Acknowledgments

To my family. Without your continuous efforts for encouraging me and shredding negative thoughts, all of this would have been much, much harder. Thank you for your presence, for the awareness that I can count on you, for walking at my side, for letting me live every day in your words and thoughts, just as much as you live in mine.

To my brother Daniele. A brother. Nothing else could encompass how much you mean to me other than this word, brother. Thank you.

To Elena, to our unforgettable moments, our shared memories, our bond that seems to never stop growing tighter and stronger, to this terrific feeling that words are not enough to describe my affection. For your support, your ability to relax or shake me when I most need it, the calm that you inject into me, as well as the pride to be all of this for you, thank you from the deep.
And thank you to Davide and Chiara, for making me feel so welcome every single time.

To all, and really all my friends, that have spiced up so many moments in my life with their invaluable presence. Feeling at home just because of your company, sharing simple as well as extraordinary things, being sad for the always too little time I spend with you, listening and be listened to, all of this has meant so much to me all the time. I owe you all.

To my "Bostonian family:" you transformed a nice experience abroad into one of the most amazing and enthralling periods of my life to date, and dug so deep into me I never stopped feeling a friend of each of you, regardless of distance.

A special and deep gratitude is for my beloved choir. It's been almost by chance that I took the lead, yet now I could hardly give it up, the main reason being this sharing, that goes well beyond a blend of voices. In particular, thanks Angelica, Leonardo and Olena, for the beat my heart loses every time I see one of our pictures together.

To all the colleagues I have shared words, feelings, thoughts, jokes, and time with, because you all contribute to make my days of hard work less exhausting and easier to face. In particular, a warm thank you to Marco, a fantastic friend, your inner strength has always been so much of an example to me.

To Prof. Michele Zorzi, my advisor. He gave me the opportunity to begin this adventure, he led me through all these years and always managed to stimulate my problem-solving skills, yet always helping when I needed it. Also, thanks to Milica Stojanovic, for the months spent with her at the Massachusetts Institute of Technology, an opportunity that revealed as an invaluable chance of professional as well as human growth.

To all of you: thank you, because, other than all of this, you have left such a deep mark in me.

Paolo

Contents

List of Figures

Abstract

Respecting a network architecture yields better guarantees of reliability, longevity, and modularity, but much better performance can be potentially achieved through wisely chosen violations to that architecture.

In a nutshell, this is the message of a recent paper (see [1] in Chapter 1) outlining pros, cons, consequences and risks of cross–layer design, a currently widely adopted paradigm for wireless networks. The increasing attention and momentum that cross–layer design has recently gained is explained by its potential advantages, namely the network performance improvements that can be achieved, especially under stringent constraints in terms of hardware and computational power. A short definition of cross–layer design identifies this technique as a means of performing information exchange among different layers in the classic ISO/OSI protocol stack model, and of harvesting the potential design opportunities and performance improvements that follow. However, by breaking the modular structure of the ISO/OSI stack, one may encounter two orders of problems: first, unwanted interactions may be introduced; second, the generality of the architecture is lost. While a careful design phase can overcome the first problem, the second one requires stronger efforts. In fact, any cross–layer design is inherently specific to the type of network and scenario it is applied to, and limits the performance improvements to that specific type. Due to this loss of generality, the same protocol hardly offers the same results as applied to different types of networks.

In this Thesis, we will show two relevant examples of successful cross–layer design applied to two very different kinds of wireless networks. The first example deals with ad hoc networks with multiple antennas and MIMO communications. Due to the specific scenario, it can be assumed that nodes have high throughput needs and can accept to, *e.g.*, spend more energy in performing the processing required by MIMO signaling in order to achieve greater communication speed. The analysis of this scenario is focused on the design of a novel PHY-aware MAC protocol for MIMO ad hoc networks and on the analysis and optimization of its performance.

A completely different point of view is required instead to handle wireless sensor networks (WSNs), the second type of wireless network considered in this Thesis. Peculiar to WSNs are the usually low communication speed, processing capabilities and energy supplies. Among others, these constraints do not allow complicated signal processing or the storage of a large amount of information. In turn this requires to limit the buffer of the nodes (the sensor hence have only a limited packet queue) and also to design protocols whose “state” can be summarized and efficiently held in the limited memory of the sensors. In the Thesis, we will provide an in-depth analysis of a geographic MAC/routing protocol for WSNs, and build upon it to yield a complete solution for channel access and packet forwarding. Part of this study is the design of an algorithm to route packets around connectivity holes, where geographic protocols alone fail.

In the appendix, the same cross–layer design concepts are applied to wireless underwater networks, a particular instance of WSNs where communications take place over long delay, low rate acoustic channels, and incur strongly frequency-dependent channel effects. All results (analysis, simulations, comparisons with other solutions) show that cross–layer design is in fact very effective, and offers valuable opportunities to leverage specific features that can lead to performance improvements in each kind of wireless network .

Sommario

Rispettare un'architettura di rete dà garanzie di affidabilità, longevità, e modularità, ma attraverso una violazione oculata di tale architettura è possibile raggiungere potenzialmente prestazioni molto più elevate.

In sintesi, questo è il messaggio di un recente articolo (vedi [1] nel Capitolo 1) che delinea pregi, difetti, conseguenze e rischi di uno dei trend recentemente più seguiti nel campo dei protocolli per reti wireless: il progetto cross–layer. L'attenzione catalizzata da questo paradigma di progetto è giustificata dalla necessità oggettiva di migliorare le prestazioni offerte dalle reti e dai protocolli esistenti, oppure di garantire performance adeguate quando i limiti dell'hardware a disposizione sono molto stringenti. Per definizione, un progetto cross–layer comporta l'interazione tra diversi layer dell'architettura ISO/OSI. Tale interazione può consistere nello scambio di informazioni o nell'interoperabilità tra layer solitamente separati, che altrimenti interagirebbero solo tramite limitate e predefinite interfacce. Gli accresciuti gradi di libertà che ne conseguono garantiscono una maggiore efficienza ai protocolli e, quindi alla rete stessa. Come detto, tuttavia, è bene tenere presenti due svantaggi quando si segue un paradigma cross–layer: l'introduzione di interazioni indesiderate e la perdita di generalità dell'architettura. Mentre un'attenta fase di progetto elimina il più delle volte il primo problema, tendenzialmente il secondo richiede più attenzione. Solitamente, un protocollo cross–layer ha prestazioni legate al tipo di rete per il quale è creato, e alle relative assunzioni di base effettuate in fase di progetto. A causa di questa perdita di generalità, è difficile che uno stesso protocollo, applicato a tipi di rete diversi, ottenga gli stessi risultati.

In questa Tesi, sono presentati due esempi di applicazione del progetto cross–layer di protocolli a due tipi di rete con necessità differenti. Il primo esempio riguarda reti ad hoc con antenne multiple e comunicazioni MIMO. Lo specifico scenario permette alcune assunzioni di massima: i nodi richiedono generalmente un throughput elevato, e sono disposti a sacrificare una maggior complessità computazionale (soprattutto nell'elaborazione dei segnali MIMO) al fine di ottenere un miglioramento delle prestazioni. L'analisi di questo scenario si concentra sul progetto di un nuovo protocollo MAC, con l'obiettivo di accrescere l'interazione tra MAC e livello fisico. Il protocollo è successivamente analizzato in profondità attraverso approcci simulativi e pseudo-analitici.

Un punto di vista diametralmente opposto è richiesto dal secondo tipo di rete considerato in questa Tesi, le reti di sensori wireless (*Wireless Sensor Networks*, WSNs). In questo caso, i terminali wireless sono molto più vincolati in termini di capacità di elaborazione, velocità di trasmissione, memoria a disposizione, risorse energetiche. Di conseguenza, i protocolli progettati devono essere estremamente efficienti e fare buon uso delle limitate risorse a disposizione. In questa Tesi, è fornita un'estesa analisi di un noto approccio geografico per l'accesso al canale e l'instradamento in reti di sensori. Tale approccio è successivamente esteso verso una soluzione completa e ottimizzata per MAC e routing. Particolare attenzione è rivolta alla soluzione del problema dell'instradamento in prossimità di interruzioni locali di connettività (*holes*), dove i protocolli geografici tendono a fallire se non adeguatamente supportati da estensioni apposite.

Nell'appendice, gli stessi concetti di progetto cross–layer sono applicati a un peculiare caso di rete di sensori, le reti di sensori sottomarini. In questo caso, i protocolli e gli schemi di comunicazione devono compensare lunghi ritardi di propagazione, basse velocità di trasmissione sul canale, ed effetti di propagazione altamente dipendenti dalla frequenza.

I risultati ottenuti (attraverso analisi, simulazioni e confronti con soluzioni esistenti) dimostrano l'efficacia del progetto cross–layer come paradigma operativo, provando come esso consenta di sfruttare le specifiche caratteristiche di ogni tipo di rete wireless al fine di migliorarne le prestazioni e l'efficienza operativa.

Chapter 1

Introduction

Contents

Recently, a trend has emerged in the field of protocol design for wireless communications. The continuous effort to improve the performance of wireless networks is pushing many researchers to adopt a new design paradigm. This new paradigm breaks the well-known architecture introduced by the OSI layered network model [2, page 20], introducing new interfaces between different layers, merging one or more of them, and controlling key parameters through the interaction of more layers than only those to which each parameter pertains.

This so-called cross–layer design is promising under many points of view. By having more network layers interact, it can grasp more degrees of freedom and make wise use of them to improve the performance of a protocol. If the correct interactions are introduced, the improvements can help achieve a greater transport capability, lower access latencies, or even higher-level objectives such as a more pleasant appearance for video streams over wireless links [3]. However, introducing new interactions requires a higher degree of awareness. In particular, one needs to explore the consequences of each specific design choice, and the effects on other interactions in the protocol stack [1].

The purpose of this introductory Chapter is to offer a broad perspective on cross–layer design, first as a paradigm and then as a means for obtaining better performance in wireless networks. In particular, Section 1.1 presents some definitions and observations on cross–layer design in general. Section 1.2 elaborates on the different types of cross–layer design and interaction. Finally, Section 1.3 summarizes the organization of this Thesis.

1.1 A Definition of Cross–Layer Design

The concept of cross–layer design is closely related to the concept of architecture. From a system-level point of view, an architecture can be seen as a decomposition of the system into multiple components. Different components collaborate to achieve a common task, and can exchange information when necessary, by obeying specific rules set up for this purpose. Such rules are described as "interfaces" between the different components.

The greatest advantage of a consistent architecture is seen both from a design and an economy of scale point of view. On one hand, the designers of a certain component can focus on the component itself. They can be sure that, so long as they conform to the previously agreed interfaces, any new version will perfectly work with the rest of the system. In turn, this enables a slower, but longer-term performance improvement. On the other hand, a good architecture means lower production costs on the long run. Kawadia and Kumar [1] observe that when a standard architecture is sufficiently widespread, the various subsystems are standardized and used across many applications, which in turn reduces the production cost and increases usage. Architecture also increases the lifetime of the system, as no change requires a complete system redesign.

In this light, it should be acknowledged that the layered architecture of networking systems is an important reason why they have spread and survived to date. Like any good architecture, the layered open system interconnection (OSI) architecture has survived many technology changes and has been a substantial building block for current global-scale networks such as the Internet. In the OSI structure [2, page 20], seven different layers are identified. The *physical* layer (PHY) provides signal-level functions and the structure needed to transmit and receive bits of information. The *data link* layer provides the abstraction of a link and functions to send "packets", *i.e.*, groups of bits over the link. In multiuser systems, most of the functions of the data link level are carried out by a sub-level, namely the Medium Access Control (MAC), that manages the use of a shared medium by different users. The rest of the data link level has logical link control functions, that are in fact simpler. These functions include the packetization of bits and error detection typically through Cyclic Redundancy Code (CRC) checks. The *network* layer manages routes, defined as successions of links forming a path from a source to a destination. The *transport* layer offers a higher-level abstraction, that of a channel. This channel can be used to transmit information between the communication parties, and can be reliable or not, depending on the specific protocol used. The difference between the upper layers, namely the *session*, *presentation* and *application* layers, is more blurred. The session-level protocols should be employed to manage sessions, where different data streams (*e.g.*, belonging to different content types in a multimedia connection) require different transport pipes to the destination. The presentation layer should provide translations of format and/or languages for the transmitted contents. Finally, the application layer makes use of the whole underlying structure. Actually, in the Transport Control Protocol / Internet Protocol (TCP/IP) stack that developed as a widespread implementation of the OSI architecture, the session, presentation and application functions are merged into the application level.

In the OSI protocol stack, only adjacent layers can communicate. In other words, interfaces exist only between subsequent layers in the stack and no messages can be directly exchanged by, *e.g.*, MAC and transport, or MAC and application. Conversely, MAC can interact with the PHY and network layer through pre-defined interfaces. The abstraction provided by the layers allows corresponding layers on different network devices to assume they can communicate with each other. For example, the network layer in a terminal can work out a path with a network layer in a router, without taking care of how the actual communication is arranged. The MAC and PHY layers will carry out that.

So, what is then cross–layer design? According to Srivastava and Motani, cross–layer design with respect to a particular architecture is "the violation of a reference layered communication architecture" [4]. Violations can be carried out by creating new interfaces, by merging layers, by jointly tuning parameters, and so forth. Each violation bears a number of advantages – and of disadvantages. While it can introduce more effective procedures, that effectively translate in better performance, a violation somehow distorts the architecture, with a detrimental impact on longevity. As explained before, a coherent architecture can boost the lifetime of a system. On the other hand, design shortcuts can lead to performance gains. When carrying out cross–layer design, this tradeoff between the long-term architectural performance gain and the short-term cross–layer gain must be taken into account.

1.2 Different Types of Cross–Layer Design

According to how the architecture is modified or violated, a number of approaches can be identified for carrying out cross–layer design. For reference, some of these are listed here, along with examples of the related design type.

A first way to break layers is to allow signaling between non-adjacent levels, whereby information and parameters are passed upward, downward or in both directions along the stack. A good example of upward information passing is provided by the Explicit Congestion Notification (ECN) extension to the TCP protocol [5]. ECN is assumed to be implemented as a standard in TCP, as it allows to mitigate some problems introduced by the wireless channel. In a nutshell, TCP works by increasing or reducing the sending rate over a connection depending on the presence of packet losses in the network. Losses are identified by "duplicate ACKs", *i.e.* acknowledgement packets that notify more than once the correct reception of a certain packet in a flow. This denotes that not all other packets after the last one received correctly have reached the destination. In a wired network, packets are most frequently dropped by routers in the presence of excess input traffic, hence duplicate ACKs are assumed to denote congestion somewhere. This is why TCP reduces the packet transmission rate. However, in wireless channels, that are much more error-prone, a lost packet does not necessarily indicate a congestion. Since TCP cannot distinguish between the two cases, it reduces the sending rate even if the channel has only momentarily entered a bad state (*e.g.*, because of fading, a peak of interference and so forth). ECN serves this purpose: when a packet is dropped because there is actually a congestion it explicitly reports this in the feedback packets, so that TCP decreases the sending rate only when necessary.

Signaling between non-adjacent layers requires new interfaces between layers, and is thus an example of cross–layer design. A slightly different yet valid example is decision-making within a certain layer based on parameters that are internal to other (even adjacent) layers. Since these parameters would not be normally known outside the layer they pertain to, making them visible is in fact a cross–layer design. The latter approach is carried out in [6]. Generally, the idea is to adapt to the channel conditions by tuning controllable transmission parameters such as the modulation and code rate.

Conversely, it should be noted that such mechanisms as the Auto Rate Fallback (ARF) are not cross–layer designs, even if they involve information flows between PHY and MAC. ARF is used in wireless networks to adapt the sending rate to the channel requirements. With ARF, the MAC layer communicates to the physical layer how to adapt its modulation and code rate to achieve a certain performance level. To do so, MAC relies on acknowledgements, which are directly visible at the MAC layer. This means that ARF is not a cross–layer design, but rather can be defined as a PHY-aware self-adaptation loop.

While passing information upward is a means of signaling the upper layers about network conditions, information passed downward allows the lower layers to understand how data should be processed as indicated by the upper layers. For example, stricter delay requirements for high-priority packets can be notified to the lower layers by the application layer, so that a certain type of priority in transmission can be applied [7]. This downward communication proves also to be very important when transmitting video over wireless links. For example, some transmission standards allow a video to be coded generating the so called *base* and *enhancement* layers. The base layer is required for the video streaming to continue, whereas the enhancement layer improves the visual perception of the video, but is not strictly required. With a downward cross–layer design, the application layers could notify which packets belong to the base layer and which belong to the enhancement layer. The MAC or PHY level can thus apply a differentiated coding to the packets, in order to give better protection to the base layer.

Information between layers can also be passed back and forth. For example, if the PHY had additional multiuser detection capabilities, PHY and MAC layers could collaborate to make collisions less likely or to exploit parallel access. Chapter 2 presents one such approach, and can thus be characterized as a back-and-forth information passing cross–layer design. Joint link scheduling and power control is another example of this design category [8]

Another way to perform cross–layer design is to create a completely new layer, encompassing the functionality of two or more layers. For example, consider Wireless Sensor

Networks (WSNs), that usually face several performance constraint and therefore must be carefully designed. Cross–layer design can achieve significant gains here. For example, protocols such as GeRaF [9] do this by jointly designing routing and MAC, effectively creating one superlayer that merges both functions. Chapter 3 develops more on this topic and presents a similar approach.

Finally, cross–layer design is also intended as finding the best choice among various combinations of techniques and protocols made available by different layers. Often, this choice must be done dynamically, with the objective to optimize a certain performance metric subject to various constraints. This approach is of particular interest in multimedia communications [10], but finds its application in other scenarios as well (*e.g.*, see [11]). It is noted in [10] that finding the optimal solution to such a cross–layer problem may prove to be difficult. First of all, deriving the relationship between parameters belonging to different layers can be very difficult. Often these dependencies involve nonlinear functions of multiple variables, and may even require approximations as close formulas are not available. Moreover, the various policies at different layers may be difficult to harmonize. Usually the application layer is more concerned with semantics and presentation, whereas the lower layers deal with issues related to transmission. Consequently it may be non-trivial to coordinate these tasks. It is also important to understand which objectives and optimizations must be carried out first, and under which practical constraints (buffer sizes, PHY configurations and so forth). Finally, attention must be paid to parameters that are independently controlled by different layers. This parallel control can be acceptable only if the control time-scales are very different, so that the slower control mechanism can be assumed to perceive an average of the value to be controlled [12].

If caution is not exercised in cross–layer design, unwanted interactions between different layers can emerge. This is the main sense of the "cautionary perspective" suggested in [1]. For example, routing protocols that create shorter paths made of longer routes usually force rate-adaptive MACs to select the lower rates available, as the chosen links can sustain no more. This may tend to reduce the maximum achieved throughput, especially with respect to a solution that chooses longer hops and a higher data rate, and perhaps transmits a lot when the channel is in a good state and slows down or stops when it is experiencing bad conditions. Similar problems due to unwanted interactions involve topology control and power control. One example of this scenario is also included in [1].

Even with all these caveats, cross–layer design offers a number of advantages and is definitely worth considering. In the following Section, some first conclusions are drawn and the objective of this Thesis is introduced.

1.3 Discussion and Organization of this Thesis

As can be seen from the previous overview, cross–layer design is gaining momentum as an emerging paradigm for performance optimization. The advantages related to cross–layer design are clear. Bridging strong boundaries between different layers yields more degrees of freedom for protocol design, and allows to pursue better optimizations. This is particularly true and needed in wireless networks, and especially at the lower network levels. Unlike in wired networks, terrestrial wireless communications rely on radio transmissions, which in turn are just spatiotemporal energy footprints. This makes the concept of "link" less meaningful, and highlights the need to take special care when keeping the physical layer features into account. Hence, cross–layer design would be a good approach even if it should boil down to, *e.g.*, physical layer awareness when designing higher-level protocols.

This Thesis presents two different cross–layer approaches, each applied to a different kind of network. In particular, we will concentrate on ad hoc networks with multiple antennas in Chapter 2 and wireless sensor networks in Chapter 3. These scenarios represent almost separate worlds.

Ad hoc networks with multiple antennas, at least at their current state of development, are designed for performance rather than for saving resources. They offer a wide range of improvements that can be exploited in the protocol design phase. These improvements include configurable directional reception and transmission (through array processing) and Multiple-Input Multiple-Output (MIMO) signal processing for spatial multiplexing and multiuser detection. Here we focus on the latter. We start from the analysis of the performance of the PHY layer in a multiuser context and derive the implication that this PHY would have on the design of higher layer protocols. We continue by designing a MAC layer that makes use of back-and-forth information exchanges with the PHY layer in order to perform multiuser detection. The whole process is driven in order to guarantee a satisfactory throughput and yet protect the wanted signals through active interference detection and cancellation.

Wireless sensor networks, instead, are tiny objects that face a lot of constraints from the point of view of PHY capabilities, processing and memory resources, and most of all available battery energy. They are often designed for long-time operations, and thus require a careful design that grasps as many performance improvements as possible. In this field, a good cross–layer design could provide the ultimate resource for increasing lifetime without sacrificing performance. In Chapter 3, we start from GeRaF, a geographic forwarding protocol for wireless sensor networks that uses a single layer to provide MAC and routing functions jointly. We will evaluate its performance and expand its cross–layer design based on the results obtained from the analysis of the protocol. This will give rise to a second protocol, called ALBA, that we evaluate and compare to GeRaF in order to prove the validity of the new approach. Another comparison is carried out against a recently proposed cross–layer protocol with similar objectives, highlighting differences between the two protocols and showing why ALBA performs better.

Furthermore, in Appendix B, we show preliminary results about physical layer-aware transmission strategies for underwater acoustic sensor networks. We analyze and compare some solutions, before discussing a broadcasting protocol based on the results of the comparison.

The main objective of this Thesis is to show how cross–layer protocol design can yield benefits in different scenarios. This is achieved through examples and research results on currently hot topics in wireless communications.

Appendix C contains a complete list of papers published or submitted during the Ph.D. program.

References

[1] V. Kawadia and P. R. Kumar, "A cautionary perspective on cross–layer design," *IEEE Commun. Mag.*, vol. 12, no. 1, pp. 3–11, Feb. 2005.

[2] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Prentice Hall, 1992.

[3] A. Ksentini, M. Naimi, and A. Gueroui, "Toward an improvement of H.264 video transmission over IEEE 802.11e through a cross–layer architecture," *IEEE Commun. Mag.*, vol. 44, no. 1, pp. 107–114, Jan. 2006.

[4] V. Srivastava and M. Motani, "Cross-layer design: a survey and the road ahead," *IEEE Commun. Mag.*, vol. 43, no. 12, pp. 112–119, Dec. 2005.

[5] S. Shakkottai, T. S. Rappaport, and P. C. Karlsson, "Cross–layer design for wireless networks," *IEEE Commun. Mag.*, vol. 41, no. 10, pp. 74–80, Oct. 2003.

[6] G. Anastasi, M. Conti, and E. Gregori, "Exploiting medium access diversity in rate adaptive wireless LANs," in *Proc. of ACM MobiCom*, Philadelphia, PA, 2004, pp. 345–359.

[7] G. Xylomenos and G. C. Polyzos, "Quality of service support over multi-service wireless internet links," *Computer Networks*, vol. 37, no. 5, pp. 601–615, 2001.

[8] T. Elbatt and A. Ephremides, "Joint scheduling and power control for wireless ad hoc networks," *IEEE Trans. Wireless Commun.*, vol. 3, no. 1, pp. 74–85, Jan. 2004.

[9] M. Zorzi and R. R. Rao, "Geographic random forwarding (GeRaF) for ad hoc and sensor networks: energy and latency performance," *IEEE Trans. Mobile Comput.*, vol. 2, no. 4, pp. 349–365, 2003.

[10] V. Srivastava and M. Motani, "Classification-based system for cross–layer optimized wireless video transmission," *IEEE Trans. Multimedia*, vol. 8, no. 5, pp. 1082–1095, Oct. 2006.

[11] S. Loretti, P. Soldati, and M. Johansson, "Cross–layer optimization of multi-hop radio networks with multi-user detectors," in *Proc. of IEEE WCNC*, New Orleans, LA, Mar. 2005, pp. 2201–2206.

[12] L. Ljung, "Analysis of recursive stochastic algorithms," *IEEE Trans. Automat. Contr.*, vol. AC–22, pp. 551–575, 1977.

Chapter 2

Cross–Layer Design in Ad Hoc Networks with Multiple Antennas

Contents

2.1 Introduction

Broadly speaking, ad hoc networks are collections of autonomous wireless terminals. The nodes that make up the network communicate over a shared channel (mainly, but not limited to, a radio one) and perform packet radio access to this channel. The main difference between ad hoc networks and common wireless networks such as WiFi, lies in the maximum flexibility paradigm that underlies ad hoc networking.

It is fairly common nowadays to use IEEE 802.11 [1] -compliant wireless networks. The widespread market of wireless devices (switches, routers, notebook cards, and so forth) as well as the integration of wireless networking chipsets on the motherboards of last-generation laptops is boosting the use of wireless networks. In a typical Small Office–Home Office (SOHO) context, several advantages can be gained from the increased flexibility and reconfigurability borne by wireless network. A typical arrangement consists of one or a few wireless access points that are connected to each other (via cables) and to the equipment that in turn connects the network to the Internet. Sometimes, these functionalities are integrated into a single device. Even though very common, this configuration is not ad hoc, in that it is infrastructured. There is something between the terminals that drives and/or manages wireless access and provides connectivity to an outside (wired) network.

Ad hoc networks, instead, are identified by the absence of infrastructure, control and maintenance devices, even of pre-specified contracts among the users and between the users and the provider. According to the commonly shared concept of ad hoc networks, the terminals should meet as needed (*i.e.,* ad hoc), set up a network, share data, and stop the connection as needed. The network should also be prepared to let new terminals in as they request to join, and to let them out when they break the connection. Furthermore, ad hoc networks are typically multihop. The limited radio coverage of a single terminal usually does not allow to reach the other end of a connection with a single hop, thus the discovery of a path to the destination, when it exists, is of primary importance.

Even if the IEEE 802.11 protocol includes a specific mode called ad hoc, it is rarely seen at work. This mode operates according to the so-called Distributed Coordination Function (DCF). In turn, DCF defines two different modes, the basic mode (with random access after carrier sensing) and the collision avoidance mode (with four-way handshaking before channel access). In the second case, the channel access scheme requires the sender to issue a first Request-To-Send (RTS) message that informs the receiver of the incoming packet and silences all nodes around the transmitter (because they could potentially interfere with the handshake). If no response to the RTS is heard (*e.g.*, because the receiver is busy), the transmitter backs off, that is, it refrains from transmitting for a certain time. Otherwise, if the recipient is free, it answers with a Clear-To-Send (CTS) packet, that clears the transmission and also silences the neighbors of the receiver to avoid interference. The data transmission follows and if the packet is correctly received, this event is confirmed through a specific acknowledgment message (ACK). This access scheme is known to help avoid the *hidden terminal* problem [2], but also to create a second problem, namely the *exposed terminal*. This means that nodes in range of the transmitter could be silenced even though they would not harm the handshake.

In fact, current ad hoc networks typically face several problems. The first and most basic impairment comes from the wireless channel. Wireless communications are subject to channel fading phenomena that may prevent a correct signal reception. For example, this might prevent the receiver from detecting the data packet, or hamper the RTS reception at a node. In this case, this node may transmit, and its packet may collide with another one, typically resulting in the loss of both.

The second problem comes from the access scheme. The RTS/CTS/DATA/ACK handshake would be safe if the transmission range were perfectly predictable. However, the aforementioned channel-related changes could impair the correct detection of part of this handshake, hence the entire communication. Furthermore, subsequent access failures increase the duration of backoffs and the delivery latency. Also, due to the so-called Auto Rate Fallback (ARF) mechanism implemented, *e.g.*, in 802.11, transmission errors cause an adaptation of the transmit rate to ensure that it can fit in the actual channel capacity. As for TCP, the bandwidth is conservatively decreased faster than it is increased. Upon subsequent errors due to fading, this may lower the channel rate even when not actually required.

The third problem is found at the network level, and is related to route discovery, maintenance and update. As detailed later, this operation is currently done through one of a few protocols that are available for ad hoc networks. However, these protocols usually represent a burden for the network. For example, reactive protocols such as Ad hoc On-demand Distance Vector (AODV) [3] and Dynamic Source Routing (DSR) [4] perform the signaling required to set up the route just as needed, and consider routes outdated after a certain time. An updated route needs to be recreated when it expires. This does not overload the network with signaling overhead, but creates excess transmission latencies, especially the first time a link is established. Conversely, proactive protocols such as Optimized Link State Routing (OLSR) [5] incur very low latencies, because they keep the state of a link/path constantly monitored through "hello" packets. This allows to synchronize routing databases, while ensuring that links are still active (for example, that the next hop in a path has not moved out of range). On the other hand, continuous control requires a certain level of signaling, that constitutes a constant overhead. Since the design of the protocols is not cross–layer, each packet sent by the routing protocol must undergo the handshake procedure as if it were a data packet, which generates further overhead. In either the reactive or the proactive case, radio propagation problems could make the routing protocol sense a link as broken when, actually, the next hop is still there. Perhaps, it is just busy in another communication, or has simply lost the packet due to channel fading. This would trigger a route rediscovery, with clear disadvantages for the network.

A final problem comes from the interactions between MAC, routing and the Transport Control Protocol (TCP), that is used by most applications to ensure reliable data delivery during a certain connection. TCP was designed to provide a reliable communication pipe. It does so mainly by means of end-to-end error control and congestion control. In a nutshell, packets are ACKed sequentially, and if some arrive out of order, the last correct packet in the flow is ACKed again. In this case, the sender receives a duplicate ACK and correspondingly reduces its sending rate. However, TCP makes a quite strong assumption on the way packets are lost in the network, namely it assumes that the reason is a congestion at some point. While in wired networks this assumption is almost always correct, in wireless networks packets are lost mainly because of transmission errors. Therefore, the congestion control mechanisms actuated by TCP are quite effective in a wired network, but can give rise to subtle issues in wireless networks. Channel errors may prevent packets from being received correctly, besides originating backoffs, and may cause multiple duplicate ACKs to reach the sender. As a reaction, TCP at the sender will interpret this as congestion and consequently reduce the packet transmission rate. Since this rate is quickly reduced but slowly increased, the sender will experience a much slower overall transmit rate. As noted in [6], this causes apparent unfairness in channel access among concurring terminals. For example, suppose that 802.11 is being used. Suppose also that a sender A requires a path of 1–2 short hops to reach its destination, and is contending for the wireless channel with another transmitter, say B. The latter, instead, needs more and longer hops. Due to the rate adaptation mechanisms at the MAC level, B is forced to a lower raw bit rate than A. Moreover, it will usually experience more errors (A's receiver is closer, thus may benefit from a sort of capture effect and receive packets correctly even in the presence of collisions). Again due to MAC effects, this turns into longer backoffs for B and hence a smaller channel share. TCP reacts to channel effects on its own, and tends to limit B's rate as well. In the end B will experience much more transmission difficulties than A. In a whole network, this definitely is a bad problem, possibly limiting the actual number of hops that it is feasible to make in an ad hoc network.

The actual impasse is that most of the applications and services made available through ad hoc networks are programmed upon TCP as of now, or will most probably be. A number of transport-level solutions have been devised in the literature to avoid the problems arising with TCP. Nevertheless, it is very unlikely that TCP will be dismissed in favor of more effective solutions for wireless networks. Given that TCP is in fact a constraint, the space left for the optimization of the protocol stack is at the MAC, possibly also at the routing level.

In particular, a cross–layer design could be of help here. TCP does not care about which routing, MAC and physical technology is being run. Therefore, any effective combination of protocols could be usefully implemented in a wireless network. The advantages coming from the interaction of one or more network layers, as outlined in the previous Chapter, could also give a significant boost to the overall performance, from raw bit rate to fairness in channel share among terminals.

This Chapter will take into account a particular case of ad hoc network, namely where each terminal is equipped with multiple antennas. As detailed in the following Section, this is a special setting. It allows a number of degrees of freedom that can be exploited to come up with an effective design. After a discussion of the work found in the literature so far, we will propose one such design that involves cross–layer interactions between the PHY and MAC layers. We will also briefly discuss the importance of extending the interaction to the routing level.

2.2 Ad Hoc Networks with Multiple Antennas

2.2.1 A brief overview of the advantages of multiple antennas

A promising direction of research to improve the throughput performance of ad hoc networks is the introduction of multiple antenna systems at the nodes. Multiple antennas potentially allow higher bit rates and enhanced communications parallelism, with clear benefits on the overall performance of the network. Smart antennas and MIMO systems have been very intensively studied in the past few years with focus on increased capacity and point-to-point link applications. This technology is now well understood and has been more recently finding its way into the networking field. On this research topic, the typical approach is to embed multiple (possibly directional) antennas in wireless nodes [7]. Multiple antennas potentially boost the system capacity and increase the degrees of freedom in the design of physical communication protocols. As each antenna brings to the node a different copy of the received signal, these copies can be processed and rearranged to yield some advantages in signal decoding, an operation known as *beamforming*. If some information is known about the "channel" between the sender and the receiver (we will go deeper on this later), the same algorithms may be applied at the transmitter and pursue further advantages. A very complete book on the topic is [8], while the reader interested in an in-depth study of directional antennas and antenna arrays is referred to [9].

When speaking of multiple antennas, one usually refers to a number of ways this technology could be fit into nodes. It is hence important to make some distinctions.

The first and simplest way to exploit multiple antennas is to perform directional communications. The easiest way to do so consists in mounting some natively directional antennas on an ad hoc terminal. Each antenna covers a limited range of transmit directions, but in turn the gain offered to the transmission is higher. This means that a greater Signal-to-Noise Ratio (SNR) is offered at the receiver for fixed distance, or that the reach of the transmission is increased for a fixed target receive SNR. However, there is one more advantage and one disadvantage with directional antennas. The advantage consists in the directional nature of the reception. Neglecting multipath phenomena for the moment, it should be noted that receptions have a limited scope, just like transmissions. This limits the effects of interference on the signal received by the antenna, because an interferer must be located in front of the antenna to disturb a reception. The disadvantage is the size of the antenna. To achieve sufficient directionality and a non-negligible transmit/receive gain, the antennas usually require form factors that do not fit the size of common portable devices. There is an interesting testbed deployment by Ramanathan *et al.* [7] that makes use of horn antennas [9]. Due to the size of the antennas, four of them are mounted on top of a vehicle, where the rest of the control and networking systems is fit into the back of the car. While this solution might be an interesting starting point for vehicular mobile ad hoc networks, it seems unlikely that it can achieve sufficient miniaturization and be used in palm desktop assistants (PDAs), laptops, cell phones, etc. In fact, other types of directional antennas exist, but those that can be integrated on a board and fit small devices are only a few. Patch antennas [9] are one such example. Usually the problem with these smaller directional antennas is the actual directivity achieved, potentially not enough to guarantee a sufficient performance gain.

Another, more complex but very promising way to exploit multiple antennas is called *beamforming*. This word encompasses a number of approaches, which have one thing in common, namely a proper weighing of the inputs (outputs) of the different antennas upon transmission (reception). Different weights allow, among other things, to create some electromagnetic radiation beams and to steer the more powerful one toward a wanted direction. Furthermore, one can also decide to place reception nulls conveniently and thus zero out interfering signals that are known to come from a certain direction. The antenna weighing technology is usually referred to as "smart antennas." Finally, one can choose to treat the multiple transmit and/or receive antennas as a Multiple-Input Multiple-Output (MIMO) system, that allows other advantages. To do either of these things the nodes must only have an array of properly spaced, typically equal antennas. The spacing is of paramount importance, because increasing the distance between different elements allows to decrease the correlation among the incoming signals, which makes beamforming algorithms perform better.

The main discriminating factor between the use of antenna beam steering and MIMO is typically the scenario. When it is easy to derive the direction of an incoming signal, beam steering is usually the best solution. Typically this requires a Line of Sight (LoS) scenario, where channel fading phenomena can be described with, *e.g.*, a Rice model [10]. In particular, the stronger the LoS component, the better the algorithms based on beam steering work. A strong multipath, in fact, tends to corrupt the information about signal directions, making it difficult to create the correct antenna radiation pattern. Nevertheless, we will detail later that most of the algorithms created for exploiting beam steering is based on the assumption that fading is negligible or absent. One more thing is also worth highlighting. When considering smart antenna techniques, it is mostly reasonable to assume that perfect Channel State Information at the Receiver (CSIR) is available. In fact, the receiver can typically apply directional scanning algorithm to sense signal peaks and thus understand the direction of the incoming signal. MUSIC and ESPRIT [8] are examples of two such algorithms. When only CSIR is available, the receiver is the only responsible of the communication performance gain, that can be achieved by increasing the gain of the antenna pattern in a certain direction or by steering pattern nulls to limit interference. Usually, the algorithm that performs beam steering works under some constraint, such as maximum wanted SNR, minimum unwanted interference, and so forth.

A deeper enhancement is reachable if Channel State Information at the Transmitter (CSIT) is also available. In the smart antenna case, this usually means that the transmitter knows the direction where to transmit to reach the receiver. This way, it can perform a proper beamforming as well and, *e.g.*, increase the power radiated toward the receiver. This can also be done in addition to other optimizations, such as steering transmit nulls toward neighbors of the transmitter, in order to reduce the interference caused on them.

This reasoning works as long as the direction of a signal can in fact be resolved with a certain accuracy. Channel variation phenomena, mainly strong multipath fading, can disrupt this condition. This tends to spoil the advantages of smart antennas. For example, a node could beamform in a certain direction without knowing that there is a stronger reflected component caused by multipath that would reach the receiver as well. Or perhaps a null steered toward a node could be ineffective in avoiding interference, perhaps because a secondary lobe of the antenna pattern causes reflected components of the signal to reach the same node.

When channel fading is a predominant effect and makes the LoS component (if any) small with respect to the overall power of the signal, it is usually better to resort to MIMO techniques. In particular, MIMO processing tends to perform better than beam steering in strong Rayleigh fading conditions, where signals propagate in a Non-Line of Sight (NLoS) scenario. In its simpler formulation, a MIMO system could be set up by a transmitter using multiple antennas to send independent symbols of a certain digital modulation. This technique is called spatial multiplexing, because multiple data flows are superimposed in the space (*i.e.*, antenna) domain. The receiver can use all of its antennas to perform diversity reception, *e.g.*, by maximal ratio combining (MRC) [10] over each independent packet. It should be noted that the receiver must have CSIR available to do this.

A more complex MIMO approach couples MRC with successive cancellation of transmitted packets during reception. Basically, the receiver estimates the channel, detects the stronger packet, and cancels it from the overall signal. If the cancellation is made correctly, the outcome is clean of the contribution of the deleted signal. This allows the detection of the second most powerful signal to be performed at a better SNR than would be without the cancellation. The process goes on like this until all packets have been detected. This technique is know as Vertical–Bell labs LAyered Space-Time (V–BLAST) and has been pioneered in [11, 12].

Another way to exploit MIMO signal processing is Space–Time Coding (STC) [13], and consists again in sending symbols through different antennas, but mixing them up in time (*i.e.*, subsequent symbol intervals), and space (*i.e.*, different antennas), according to a predefined scheme. The first and most widely known STC algorithm is the Alamouti code [14], which works on two different symbols being mixed in two different transmit antennas over two subsequent symbol intervals. More complex codes can be created where the number of antennas and symbol times involved is higher. STCs can also be layered and decoded successively using a procedure similar to V–BLAST. Actually, V–BLAST is a special case of layered STC where no space–time encoding is performed. It has also been shown [15] that there exists a tradeoff between diversity and spatial multiplexing gain in MIMO networks: in this light, we stress that V–BLAST achieves the greatest spatial multiplexing gain, whereas STCs such as [14] are optimal in a diversity sense.

As a final note, we highlight that MIMO transmissions can as well benefit from CSIT. In this case, achieving CSIT does not boil down to knowing which direction to transmit to or where to put radiation nulls. It means to estimate the channel effects on the signals that propagate from any transmit to any receive antenna, and to make this information available at the transmitter. This is actually more difficult to grasp, and usually requires a large amount of feedback from the receiver. However, it can be shown [13] that a MIMO transmission can be modeled as a number of parallel single-link transmissions. Each of these travels on a virtual subchannel that depends on the channel matrix.[1] In particular, each virtual subchannel corresponds to one of the non-zero singular values of the channel matrix. Having CSIT available allows to exploit all these channels, so that the actual channel capacity becomes the sum of the capacities of all subchannels.

Smart antenna arrays and MIMO communications are deemed to be a key technology for next-generation wireless networks. The high rates achieved through V–BLAST, in particular, are described as the first way to enable Gigabit-per-second communications in wireless networks [16]. In this light, the next Section aims at providing an overview of the current work related to the area of protocol design for ad hoc networks with multiple antennas. Various approaches will be presented, with an emphasis on the scenario of application (directional antennas, MIMO, and so on).

[1]For a definition of a MIMO channel matrix, see Section 2.3.1.

2.2.2 Related work on the use of multiple antennas in ad hoc networks

Directional communications and smart antennas

The first studies appeared on ad hoc networks involved the use of multiple directional antennas or arrays of simpler antennas used for beamforming purposes. Most of the work found in the literature focuses on the design of MAC and/or routing algorithms. These algorithms exploit either the higher directivity (thus longer reach) of the signals, or the smaller interference undergone when using directional reception.

In [17], the authors focus on the use of purely directional transmissions, designing a routing-aware MAC protocol called Multihop MAC (MMAC), that exploits the higher gains and lower overall interference achieved by directional communications for bridging longer distances. It basically exchanges control messages (RTS/CTS) over more than one hop in order to coordinate farther nodes. If an RTS is marked as "Multihop," it is propagated for multiple hops before a CTS is actually sent back. Eventually, a node understands that it can be reached directly by the RTS sender if both beamform in the direction of each other. This node answers back with a CTS specifying this fact, so that this long-range communication may take place. Both the RTS and the CTS may be sent in a directional fashion, in order to cover a longer reach while enabling greater spatial reuse.

In fact, the directivity of transmissions and receptions enables multiple communications to coexist in the same network area. This is not the case in 802.11-based ad hoc networks, where RTSs and CTSs block any communication within a certain area, to prevent interference. This potentially boosts the spatial reuse in an ad hoc network. However, directionality carries along a problem, namely increased "deafness." In fact, directional transmissions may leave nodes uncovered and thus unaware of the ongoing communication (deaf). This is not true for receivers, because a node could apply a number of predefined beamforming vectors that allow to steer the main lobe of the pattern toward different predefined directions. This makes it simpler at least to detect incoming packets, perhaps to correctly receive them, and allows to translate the solution to deafness into a signal processing problem. Quite surprisingly, this fact has not been accounted for in many papers that propose algorithms based on directional receptions. For example, a solution based on busy tones is provided in [18], which requires additional hardware.

Ramanathan *et al.* propose UDAAN in [7], a set of integrated MAC, routing, neighbor discovery and signaling protocols for ad hoc networks with directional antennas. They also built a field demonstration, using horn antennas. This is, to the best of our knowledge, the most comprehensive mobile ad hoc network testbed which has been deployed for assessing the advantages of directional antennas.

In [19], a MAC protocol is considered that has parties send and receive data directionally. Routes are created, maintained and rediscovered when needed by flooding route requests in the direction of the receiver, instead of omnidirectionally. While this protocol imposes a lighter traffic overhead, it requires topology awareness. Moreover, the model of antenna directionality is slightly oversimplified. In fact, the authors assume that N directional antennas are available, and that each one achieves a flat gain over an azimuthal extension exactly $2\pi/N$ large, which is usually not the case.

Korakis *et al.* propose Circular MAC [20], a MAC protocol with directional RTS/CTS exchange. RTSs possess a longer reach thanks to directional transmissions, but they are sent directionally using one beam at a time, so that many transmissions are needed to cover the whole horizon. This approach notifies farther nodes, thus mitigating deafness and establishing longer links, but incurs longer handshake latencies.

These papers present very interesting contributions, showing benefits of directional communications in ad hoc networks. However, very simplified propagation and antenna models are typically taken into account, *e.g.*, cone–sphere models, with the main antenna lobe having constant amplitude over a certain angular extension, with negligible or even no side lobes. This may not be always the case, especially when performing array processing, *i.e.*, when obtaining directivity through the superposition of omnidirectional signals sent from the array elements. Even if some beamforming technique is available that can steer a main beam of predefined width, and still guarantee secondary lobes to be under a given threshold [9],[2] they may still radiate a significant amount of power, potentially reducing the accuracy of simplified models. A study by Takai *et al.* [21] has also highlighted the need to account for realistic physical layer models when evaluating ad hoc networks with multiple antennas.

MIMO communications

In case MIMO communications are used, recall that all transmissions are omnidirectional, and multiple superimposed signals can be separated and detected through some specific signal processing at the receiver, such as in the V–BLAST architecture [11]. The problem of studying and optimizing MIMO links under different objectives and constraints has been widely addressed in the literature. Two complete books on the topic are, *e.g.*, [13, 22]. The specific application of MIMO systems to ad hoc networks, however, has received less attention and is only recently beginning to gain momentum.

In the existing literature, many papers take an information-theoretic point of view, and calculate throughput as the maximum mutual information between a received and a transmitted signal, proposing algorithms that, *e.g.*, pursue some throughput optimization under power constraints [23]. These works are interesting, but typically require to assume that the channel capacity is exactly reached, *e.g.*, by using Low Density Parity-Check (LDPC) or turbo channel codes in a real implementation. On the other hand, such approaches often neglect specific networking issues, for instance a particular MAC implementation.

A networking-based approach is carried out in [24] with MIMA-MAC, an access protocol tailored to ad hoc networks with up to two antennas per node. The devised MAC includes

[2]We refer here to the Dolph-Čhebyschev beamforming algorithm.

a contention and a contention–free period. In case of MIMO communications among two receivers and two different transmitters, MIMA-MAC results in each transmitter using one antenna. Each receiver instead uses two, trying to decode both transmissions. The low number of nodes considered and the limit to use at most one antenna for transmission represent significant drawbacks here.

In [25], the authors extend the busy tone approach of [2]. They suggest to use one of the sub-bands provided by a MIMO–Orthogonal Frequency Division Multiplexing (OFDM) physical layer for sending the busy tone. A maximum number of busy tones are allowed, due to limitations in the number of receive antennas. Therefore, prior to sending the busy tone, each transmitter chooses a random backoff and senses the channel. If it hears more than the preset maximum number of tones, it defers transmission, otherwise it activates its own tone and sets up the link with the receiver.

Another very interesting work on MIMO ad hoc networks is [26]. A MAC protocol is designed based on IEEE 802.11 DCF [1], and is modified for exploiting spatial diversity. RTSs and CTSs are used along with PHY preambles that allow for channel estimation. The acquired information is then used to decide the correct transmission rate for the data packet. STCs are used for achieving the full diversity order available. An analysis of the MAC impact on routing is also carried out, evaluating the relation between the delay incurred before sensing a free channel and the advancement obtained with a one-hop transmission. Directional antennas are explored as a special case of multiple antenna communications.

A different method for managing radio links with multiple antennas is given in [27]. There, a centralized controller estimates concurrent resource usage and schedules links to exploit the benefits of MIMO, such as spatial multiplexing and interference suppression, along with increased transmit rate. The final objective is a proportional fair scheduling of transmissions that accounts for bottleneck links, and is achieved by graph coloring. An online algorithm is also designed. This last contribution, although interesting, makes some very strong assumptions on the PHY layer, *e.g.*, assuming that any transmission uses the full channel capacity and that signaling at the MAC level is perfect.

A specific case with multiple transmit antennas but one receive antenna, *i.e.* Multiple-Input-Single-Output (MISO) links, is considered in [28]. In more detail, the authors focus on Virtual MISO links, whereby the antennas performing the transmissions are distributed and belong to different devices. The protocol first synchronizes a set of neighbors of the transmitter by using specific signaling messages. All synchronized nodes apply a space–time coding scheme to the transmitted symbol, by choosing a certain column of a space–time encoding matrix. Using a virtual MISO transmission allows the signal to travel longer distances, effectively shortening the path to the destination.

Similarly, the approaches in [29, 30] focus on a virtual MIMO, with multiple distributed transmit antennas and multiple antenna processing at the receiver. In either approach, the nodes organize themselves in clusters. A subset of nodes within the cluster is chosen to relay the packets to the next clusterhead, with [29] or without [30] the help of the clusterhead itself.

The use of MIMO links (and specifically STCs) is also found in [31] for addressing the problem of efficient broadcasting in ad hoc networks with multiple antennas. Such a technique, for example, could substitute the circular RTS transmissions required by CMAC [20].

2.2.3 Observations and scope of the work

When assessing the performance of an ad hoc network with multiple antennas, an accurate characterization of the underlying PHY layer is of paramount importance. As to MIMO communications, a nontrivial assessment of the receiver capabilities is needed, in particular for nonlinear multiuser detectors like BLAST [11,32]. Until now, several approximations are found in the PHY models used in the literature. On one hand, they are needed to grasp the main message provided by the various works without having to deal with too many details. On the other hand, correct evaluations can be done only with correct PHY models, which is somehow lacking in the present literature, except for a few works.

From the MAC point of view, some of the papers above rely on the exchange of signaling messages among communication parties. Unlike directional listening/transmitting, MIMO links are inherently omnidirectional and not prone to deafness in the sense used for directional transmissions. Moreover, with the use of a proper MIMO transmission scheme, significant bit rate improvements could be achieved at the price of increased complexity at the receiver side (PHY). The main effort then becomes to design a new protocol that explicitly addresses MIMO features.

In the following, we will start from an accurate PHY layer model and construct MAC and link management protocols in a completely cross–layer fashion. This choice comes from the discussion in the previous Chapter, whereby cross–layer interactions have been highlighted as a very promising way to achieve better performance. We will derive an approximate yet precise model for the behavior of the PHY layer and show that it reliably predicts network performance, which is our ultimate requirement. We will show how the protocol could be optimized from different points of view, thanks to the reduced simulation time allowed by the approximated PHY model. We will also describe how our considerations should be extended to bring routing into the cross layer design. To this end, we will present some preliminary results that show the importance of this approach.

2.3 Physical Layer Model

We proceed by providing a thorough description of the physical layer model for MIMO communications in ad hoc networks. It should be noted that, unlike in [32], the scenario here is more complex. Specifically, we suppose that nodes can transmit simultaneously to different (possibly, multiple) receivers. Additionally, each node is allowed to send independent data symbols through different antennas by means of spatial multiplexing. Hence, we need to take this into account when modeling this multiuser MIMO system.

We suppose that each node has N_A antennas. As the transmission scenario we assume a frequency-flat block fading channel. Nodes can both transmit and receive information, in a peer-to-peer fashion. For now, we also assume that all nodes are in communication range of each other, *i.e.*, we consider a completely connected network.[3] To keep the analysis simple, we assume bit-synchronous node operations, although this is not a requirement when using BLAST receivers (see [32] and references therein).

2.3.1 Node operations

Transmitting nodes

When operating as a transmitter, the generic node p splits each packet into sub-packets called *PDUs*. Each PDU is sent from one antenna, so that u_p PDUs are multiplexed using u_p antennas during transmission. Assuming that N_{Tx} nodes with indices $\{1, 2, \ldots, N_{\text{Tx}}\}$ are transmitting, the total number of simultaneously transmitted PDUs is $U = \sum_{p=1}^{N_{\text{Tx}}} u_p$. We can identify each transmit antenna with a transmit antenna index (TAI), starting with the first antenna of the first node and ending with antenna $u_{N_{\text{Tx}}}$ of node N_{Tx}.

Let us define the column vector $\boldsymbol{s}'(t) = [s'_1(t), \ldots, s'_U(t)]^T$ whose entry $s'_i(t)$ is the symbol transmitted from antenna with TAI i at time tT, where T is the symbol period and ^{T} denotes the transpose operation.

We assume that node p always transmits with a total power P_{tot} that is uniformly distributed among the u_p antennas. Hence, the power of the transmitted data signal on the antenna with TAI i belonging to node p is $\sigma^2_{s'}(i) = \mathrm{E}[|s'_i(t)|^2] = P_{\text{tot}}/u_p$ where $\mathrm{E}[\cdot]$ denotes expectation. For the sake of a simpler notation, we omit in the following the time index t in all signals.

Receiving nodes

When operating as a receiver, the generic node q uses all the N_A antennas and the column vector of the N_A received samples can be written as $\boldsymbol{r}^{(q)} = \tilde{\boldsymbol{H}}^{(q)}\boldsymbol{s}' + \boldsymbol{\nu}'^{(q)}$, where $\boldsymbol{\nu}'^{(q)}$ is the column noise vector of length N_A, and $\tilde{\boldsymbol{H}}^{(q)}$ is the $N_A \times U$ channel matrix whose entry $\tilde{\boldsymbol{H}}^{(q)}_{\ell,m}$ represents the complex baseband channel gain between the transmit antenna of TAI m and the ℓth receive antenna.

In order to limit the node complexity, we assume that each receiving node has a partial knowledge of the channels, *i.e.*, node q can only know the channel gains associated to N_d transmit antennas (called *internal antennas*, IA), whose TAIs are in the sub-set $\mathcal{N}^{(q)} = \{n_1, n_2, \ldots, n_{N_d}\}$, for which we assume perfect channel estimation. Without restriction, we assume that the TAIs of known antennas are the first N_d, *i.e.*, $\mathcal{N}^{(q)} = \{1, 2, \ldots, N_d\}$. During

[3]While this is not the most general case, it will represent a much more challenging scenario for the MAC protocol being designed. This will make results and differences more notable.

data transmission, in general, $\mathcal{N}^{(q)}$ includes the TAI of all granted transmissions intended to reach node q and the TAI of some other interfering transmissions.

We define $\boldsymbol{H}^{(q)}$ as the matrix containing the columns $\tilde{\boldsymbol{H}}^{(q)}_{\cdot,i}$, with $i \in \mathcal{N}^{(q)}$, and we define the N_d-size vector of data symbols belonging to $\mathcal{N}^{(q)}$ as $\boldsymbol{s}^{(q)} = [s'_1, s'_2, \ldots, s'_{N_d}]^T$. The variance of the entries of $\boldsymbol{s}^{(q)}$ is $\sigma^2_{s^{(q)}}(i) = \sigma^2_{s'}(i), \forall i$.

The transmitting antennas whose signals are not detected by node q are instead indicated as *external antennas* (EA) to q. According to $\mathcal{N}^{(q)}$, we assume that the TAI of EA are $N_d + 1, N_d + 2, \ldots, U$ and the corresponding channel matrix $\bar{\boldsymbol{H}}^{(q)}$ contains the columns $\tilde{\boldsymbol{H}}^{(q)}_{\cdot,i}$, with $i = N_d + 1, N_d + 2, \ldots, U$. The transmitted symbols of EA to q are denoted as $\bar{\boldsymbol{s}}^{(q)} = [s'_{N_d+1}, s'_{N_d+2}, \ldots, s'_U]^T$. With the definitions of IA and EA, we can rewrite the received signal as

$$\boldsymbol{r}^{(q)} = \boldsymbol{H}^{(q)}\boldsymbol{s}^{(q)} + \bar{\boldsymbol{H}}^{(q)}\bar{\boldsymbol{s}}^{(q)} + \boldsymbol{\nu}'^{(q)} . \tag{2.1}$$

The explicit modeling of a pure interference term is an important point in our ad hoc scenario. Indeed, terminals may be able to detect only a limited number of signals, or they may decide to neglect low-power interference and reduce processing, *e.g.*, for energy saving purposes.

In order to simplify notation we will omit in the following the index of the receive node $^{(q)}$ in all variables, since we will always refer to a single node.

2.3.2 BLAST receiver

In order to extract a sufficient statistic for detection, the receive node multiplies the vector of the received samples by a matrix matched to the channel. By defining the $N_d \times N_d$ matrix $\boldsymbol{R} = \boldsymbol{H}^\dagger\boldsymbol{H}$, where † is the Hermitian operator, the obtained vector is $\boldsymbol{z} = \boldsymbol{H}^\dagger\boldsymbol{r} = \boldsymbol{R}\boldsymbol{s} + \boldsymbol{\nu} + \boldsymbol{i}_{\mathrm{EA}}$, where $\boldsymbol{\nu} = \boldsymbol{H}^\dagger\boldsymbol{\nu}'$ and $\boldsymbol{i}_{\mathrm{EA}} = \boldsymbol{H}^\dagger\bar{\boldsymbol{H}}\bar{\boldsymbol{s}}$ accounts for the interference due to EA. We recall that the receiver is not required to know the statistics of $\boldsymbol{i}_{\mathrm{EA}}$.

The BLAST receiver performs the detection of the PDUs in stages. At each stage the PDU with the highest signal to noise plus interference ratio (SNIR) is detected, and its contribution is removed from the vector $\boldsymbol{z}$ before the next stage [11, 12]. The ordered TAI set is $\{k_1, k_2, \ldots, k_{N_d}\}$, which is a permutation of the integers $1, 2, \ldots, N_d$.

Let k_i be the TAI of the PDU detected at the ith stage and $\boldsymbol{z}(i)$ the vector obtained from $\boldsymbol{z}$ after the removal of the contributions due to PDUs with TAI in the set $\mathcal{K}(i) = \{k_1, k_2, \ldots, k_{i-1}\}$ where we set $\mathcal{K}(1) = \varnothing$ and $\boldsymbol{z}(1) = \boldsymbol{z}$. The detection of PDU k_i is performed by combining $\boldsymbol{z}(i)$ with the weighing vector $\boldsymbol{w}(i)$ to obtain the sample $\tilde{s}_{k_i} = \boldsymbol{w}(i)^T\boldsymbol{z}(i)$, which is applied to a threshold detector to provide the symbol estimate $\hat{b}_{k_i}$. The estimated symbol is multiplied by the standard deviation of the transmitted symbol to obtain $\hat{s}_{k_i} = \sigma_s(k_i)\hat{b}_{k_i}$. After detection, the contribution of PDU k_i is removed from $\boldsymbol{z}(i)$ to obtain

$$\boldsymbol{z}(i+1) = \boldsymbol{z}(i) - \boldsymbol{R}_{\cdot,k_i}\,\hat{s}_{k_i} , \quad i = 1, 2, \ldots, N_d - 1 , \tag{2.2}$$

where $\boldsymbol{R}_{\cdot,k_i}$ is the k_ith column of $\boldsymbol{R}$.

In the zero forcing (ZF) approach [11], the weighing vector aims at minimizing the interference, regardless of a possible noise enhancement. Let $\boldsymbol{R}(1) = \boldsymbol{R}$ and then compute $\boldsymbol{R}(i)$, $i = 2, 3, \ldots, N_d - 1$ by nulling the k_ith row and column of $\boldsymbol{R}(i-1)$. The weighing vector $\boldsymbol{w}(i)$ is the k_ith column of $\boldsymbol{R}^+(i)$ (the Moore–Penrose pseudoinverse of $\boldsymbol{R}(i)$ [33]), *i.e.*, its mth element is $w_m(i) = [\boldsymbol{R}^+(i)]_{k_i,m}$, $m = 1, 2, \ldots, N_d$.

Provided that the number of receive antennas is larger than the number of residual IA, $N_d - i$, and that $\boldsymbol{R}(i)$ is full rank, the given weighing vector completely cancels the interference due to PDUs $k_{i+1}, k_{i+2}, \ldots, k_{N_d}$. However, when $N_A < N_d - i$, $\boldsymbol{R}(i)$ has rank smaller than $N_d - i$, and using $\boldsymbol{R}^+(i)$ leaves some residual interference due to IA after weighing. Alternatively, a minimum mean square error (MMSE) criterion could be adopted for the choice of the weighing vector [34], which jointly considers noise and interference. Here we only consider the ZF approach, since we verified by simulation that MMSE does not bring a really significant advantage and its derivation is entirely analogous to the ZF case.

Lastly, for real constellations, a receiver with improved performance has been derived in [32]. Accordingly, in the following we shall still call the signal correlation matrix $\boldsymbol{R}$, with the implicit understanding that it represents in fact its real part, $\mathfrak{Re}[\boldsymbol{R}]$.

2.3.3 Insights on the performance of the PHY layer

It is out of the scope of this thesis to provide results on the performance of the BLAST receiver itself. However, in the following we will describe some results that help understand the impact of this PHY level on the upper layers, thus on the network performance. In Figure 2.1 we report the Bit Error Rate (BER) performance in a distributed context, where a single receiver with $N_A = 8$ antennas tries to decode U incoming PDUs of fixed length, each sent from one antenna at full power by a different user. All users are placed at the same distance from the receiver. The simulation has been conducted with 1000-bit PDUs, over many different channel realizations. As suggested by Figure 2.1, there is a fairly low probability of bit error even in the presence of substantial receiver overload. With 14 incoming PDUs, for instance, the BER falls below 10^{-5} for sufficiently high SNR. Note that a successive cancellation algorithm such as BLAST is especially suited to ad hoc networks, since a more realistic spatial distribution of transmitters typically translates into significantly different received powers, which in turn result in better performance without requiring distributed power control. Furthermore, resorting to a pseudoinverse–based decorrelating receiver translates into a soft limit on the number of incoming flows, which is not limited to N_A.

To show the reason behind the choice of real constellations and of the increased performance receiver for real constellations described in [32], Figure 2.2 depicts the BER for $U = 4, 10$ and $N_A = 6, 8$ using BPSK and QPSK. The figure contains a performance comparison of a low spectral efficiency Binary Phase Shift Keying (BPSK) modulation along with a Quaternary Phase Shift Keying (QPSK) modulation, which has a double spectral efficiency than BPSK. However, QPSK requires that the increased performance receiver in [32] is not

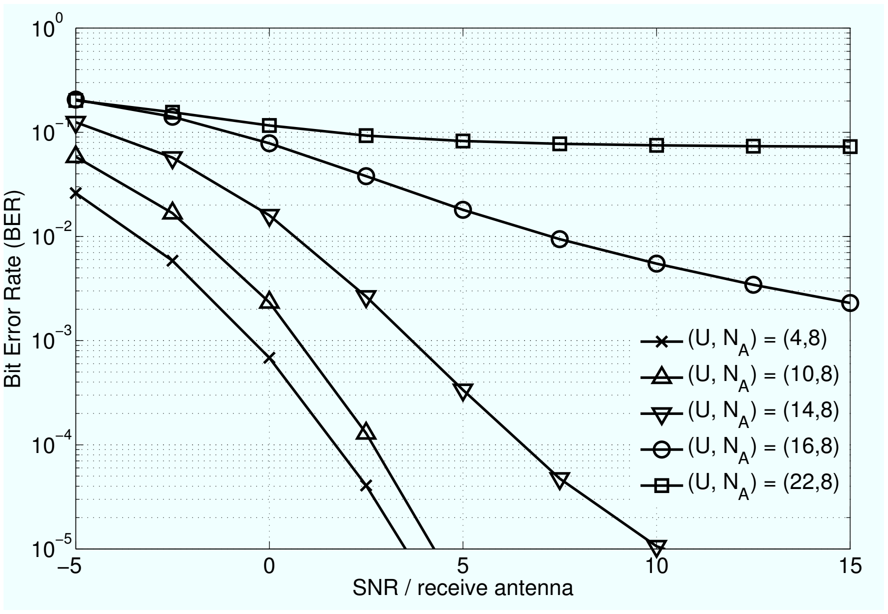


Figure 2.1. *BER performance of BLAST multiuser detection with $N_A = 8$ receive antennas for varying U. All transmit antennas are at the same distance from the receiving node.*

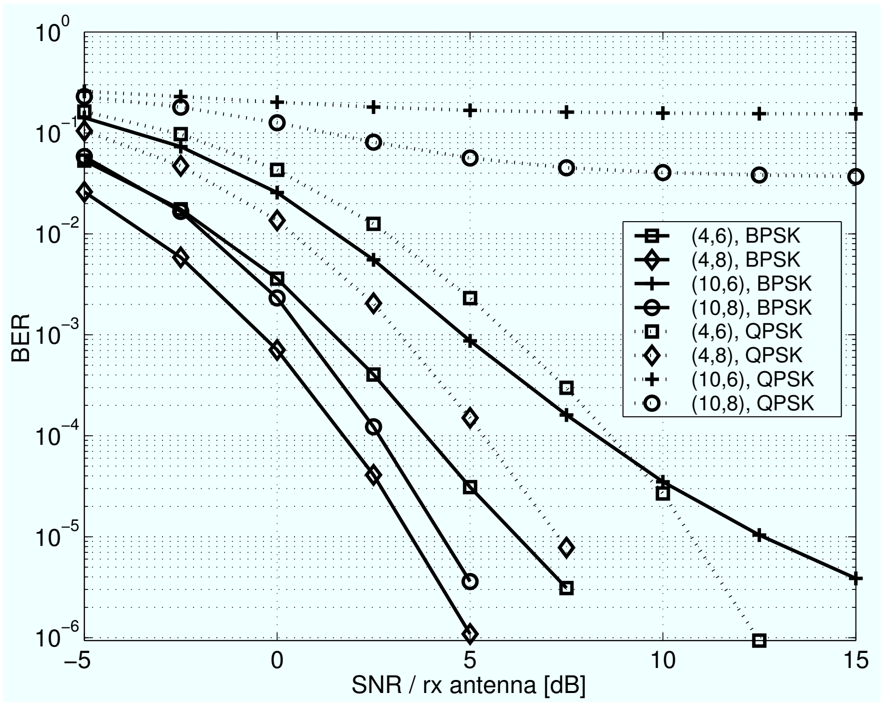


Figure 2.2. *Comparison of BERs as a function of SNR per receiver antenna for 4 different transmit/receive antenna configurations, using BPSK and QPSK modulations.*

used. Namely, one cannot take the real part of $\boldsymbol{R}$ during the decoding process, thus the overall signal processing is made in the complex domain. In turn, this means that the cross-talk between the real and imaginary parts of the constellation signals becomes a potential source

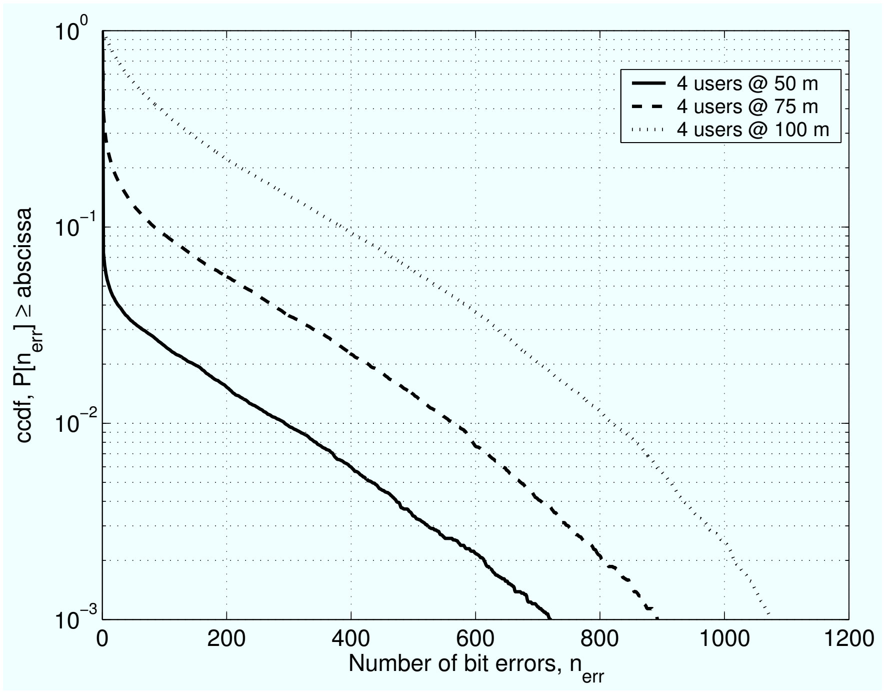


Figure 2.3. *Complementary cumulative distribution function of the number of errors, $N_A = 8$, $U = 16$, divided in 4×4 incoming PDUs, from 4 users at a distance of* $50\,\mathrm{m}$, $75\,\mathrm{m}$ *and* $100\,\mathrm{m}$ *from the receiver.*

of error. As can be inferred form Figure 2.2, BPSK decoding gives better results. This is not only a consequence of the constellation simplicity, but also of the fact that it is real and thus not affected by the aforementioned cross-talk. This explains the significant performance difference between QPSK and BPSK $(10, 6)$ and $(10, 8)$ curves in Figure 2.2. One could argue that even if it offers a somehow worse detection performance, QPSK still has double spectral efficiency with respect to BPSK, and thus may be worth considering. However, Figure 2.2 shows that BPSK's lower spectral efficiency is well compensated by the higher decoding performance. For example, in a denser multiuser context, where many PDUs may come from more than 10 different antennas, QPSK would worsen the BER performance of the system. Therefore, in the following we will always assume the use of BPSK.

Further insights come from Figure 2.3, that depicts the decoding performance in a high load scenario. It contains the complementary cumulative distribution functions (ccdf) of the number of bit errors over a packet (n_{err}), when 4 users transmit 4 *uncoded* 1000-bit PDUs each, with 4 antennas, for a total of 16 PDUs, to be decoded using 8 antennas at the receiver. It is interesting that the decoding method performs very well at limited distances (solid curve), as the probability of making any errors is very low. The motivation lies in the very nature of the channel, whose rich scattering underlying the frequency–flat fading model assumed here gives separate PDUs a different spatial signature. This allows the receiver to reconstruct the original bit sequence correctly with high probability. Furthermore, the relatively short transmit distances provide a fairly high SNR. The detection ability decays with increasing distance of the transmitters, whereby at $75\,\mathrm{m}$ it becomes more difficult to detect the various PDUs correctly. Also, this is almost impossible at $100\,\mathrm{m}$. A first conclusion we can draw from this experiment is the following. Consider a MAC protocol, whose purpose is to regulate channel access in order to get data through while serving as many users as possible. From the previous observations we see that one such protocol could afford to overload a receiver with a high number of PDUs, provided that they come from other nodes placed within a limited distance.

Another important MAC implication regards a collision avoidance (CA) scenario. We recall that signaling packets in a CA protocol (RTSs, CTSs, etc.) are approximately 5 to 10 times shorter than data packets, so that their correct reception is more likely. Other simulation results we will discuss later show that as many as 14 to 16 RTSs could be simultaneously decoded, depending on SNR conditions. What we wish to stress is that a node could receive multiple RTSs even from distant nodes, thereby conjuring up a more or less precise picture of the traffic situation in its neighborhood. This would also allow to foresee congestion, as nodes with a longer backlog would send RTSs more frequently. In such a situation, a single CTS could be employed to answer multiple RTSs, leaving to the node the freedom to decide how many transmissions it should solicit, depending on the perceived traffic conditions.

A third consequence is also of great importance for network scenarios: if some sort of synchronization in CA signaling is set up between nodes, so that all needed RTSs are first sent simultaneously, and then so are CTSs, data and ACKs, a terminal could receive multiple CTSs and decide on its own how many antennas to use for spatially multiplexed data transmission. Roughly, as more CTSs are decoded, more PDUs will be injected in that portion of the network, so that receivers will become more overloaded, and fewer PDUs should be simultaneously sent by each transmitter. This consideration paves the way for the definition of policies that control when to send a CTS in response to an RTS, and thus keep the network load under control.

The conclusions drawn so far about the dependency of the errors on distance are also supported by the results in Figure 2.4. There, the average number of bit errors over a transmission of 4 PDUs is shown as a function of distance. We suppose here that a single user is transmitting 4 1000-bit PDUs to a receiver with 8 antennas, but in the presence of EAs, that is, of transmissions whose channel cannot be tracked. These transmissions are treated as pure interference by the receiver. When there are no such EAs, no errors are registered over a large range of distances. The error tolerance is very high up to a distance depending on the number of interfering users, as each of them contributes more EAs. In particular, we suppose that each user transmits with 4 antennas, and thus contributes 4 EAs. Clearly, with 8 users the performance tends to decrease significantly with respect to the case with no interference.

Fig. 2.4 also gives a second insight. During data reception, a user may afford to send uncoded PDUs at a far distance, even in strong uncanceled interference conditions. Thus, the receiver need not always estimate and cancel interfering components for all signals it hears, but can afford to leave some of them as EAs, reducing the complexity of the decoding

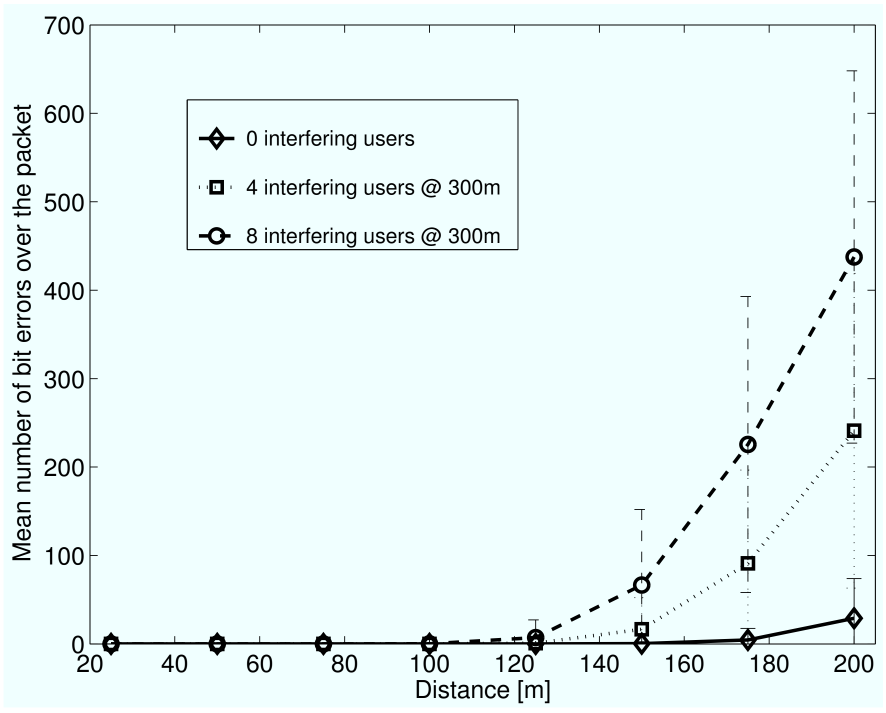


Figure 2.4. *Average and standard deviation of bit errors over a packet as a function of distance, 8 rx antennas,* 4 *incoming* 1000*-bit PDUs from one user, with* 0*,* 4 *and* 8 *interfering users.*

process accordingly. This may be of help during both signaling and data packet reception, as a receiver may not require (or not be able) to know every signaling transmission coming from too large a portion of the network, but may eventually afford to take MAC or routing decisions based on the status of only the neighboring nodes. In fact, we recall that the status of the neighbors can be known with fairly good accuracy, because short signaling packets can be demonstrated to travel far away without errors.

As a next step in drawing MAC layer implications of layered space–time multiuser detection, we inspect the probability of correct packet reception as a function of distance. Following the general rule that the number of bits per transmitting antenna is fixed, we allow some coding to be applied to the data packet, so that if N_D is the number of bits in a data packet and r the code rate, then the number of bits to transmit becomes N_D/r. In order to make a fair comparison between all evaluated configurations, we force the nodes to use more antennas to send more bits, so that the global duration of the transmission is the same in both the coded and uncoded case. As an example, a rate $2/3$ coded flow of 4000 bits becomes 6000 bit long, and is transmitted with 6 antennas to comply with the 1000-bits-per-antenna constraint. The code used is the rate $1/2$ convolutional code described in the 802.11 standard [1]. A rate $3/4$ version of the code is obtained by puncturing the coded bits according to a predefined scheme.

Figure 2.5 shows the correct packet reception probability, P_{corr}, for the case where a single user transmits a coded or uncoded flow using the appropriate number of antennas,

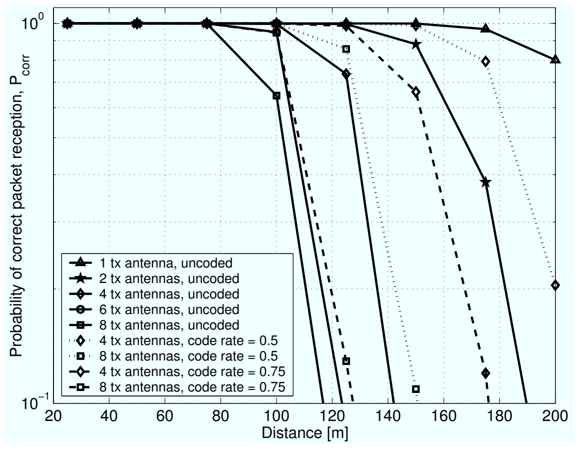


Figure 2.5. *Probability of correct packet reception, with and without coding, for different numbers of transmit antennas.* 1 *transmit node.*

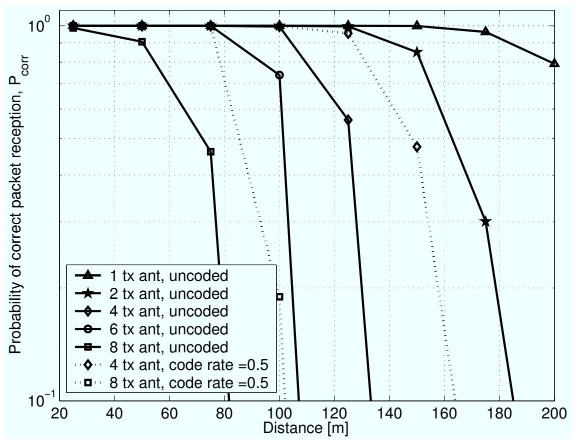


Figure 2.6. *Probability of correct packet reception, with and without coding, for different numbers of transmit antennas.* 2 *transmit nodes.*

e.g., 1000 bits with a single antenna, 2000 bits with two antennas, and so on. The receiver always uses all of its 8 antennas. An average interference level is obtained by having 4 users send data from 300 m away. As can be inferred, the distance at which an uncoded transmission becomes excessively error-prone varies as a function of the number of used antennas. The two extreme cases show a maximum reachable distance of about 150 m (when a single antenna is used) which falls to roughly 75 m when the complete set of available antennas is engaged in transmission. We only consider the values of P_{corr} ranging from 0.1 to 1, because lower values represent nothing more than a waste of resources.

Again, Figure 2.5 tells us that in low traffic conditions and at small distances, a high degree of spatial multiplexing could be used, hence reaching very high transmission bit rates. On the other hand, lower bit rates allow to reach farther destinations. It is interesting to highlight that, for the single user case, a proper encoding of the PDU could translate into a longer covered distance. For example, the curve representing a 2000 bit 1/2-coded PDU (for a total of 4000 bits) sent with 4 antennas outperforms the curve representing a 2000 bit uncoded packet sent with 2 antennas, and the same happens for the 8000 bit 3/4-coded packet sent with 8 antennas and the 6000 bit uncoded flow transmitted with 6 antennas. This means that, in low traffic conditions, coding makes it possible to reach farther distances at the price of an increased number of transmitting antennas. A MAC protocol might be designed to exploit this favorable condition and force users to change their coding and antenna configuration adaptively. This should be done according to the nodes' bit rate requirements and to the neighboring nodes' status, which could be inferred from signaling packets.

However, the case with two transmitting users leads to different observations. Figure 2.6 reports the values of P_{corr} following the same concept used in Figure 2.5, with the difference that here we consider two users transmitting simultaneously from the same distance, with the same number of antennas and the same coding configuration. The information we draw from this figure is that coding is no longer of help in beating the interference from the other data flow we have introduced, when the objective is to reach a farther distance. The system still has a very high performance even for a high number of transmitting antennas, if the distance from the receiver is kept below 75–$80\,\mathrm{m}$, but when the transmission distance increases, it is better to introduce a smaller overall interference by sending uncoded packets over fewer antennas. Also, we infer it would be preferable for a MAC protocol to fragment longer packets into smaller units, and to transmit these units sequentially using fewer antennas, so as not to increase the system load.

This last result suggests that the use of channel coding (increasing the number of antennas) is not a very good choice. The lower transmit power and the increased receiver load tend to null the advantage introduced by the coding scheme. A similar problem would be found using, *e.g.*, space–time codes. Hence, in the following design, we decided to assume that no PDU is actually coded. Our MAC protocol will focus on traffic control among neighboring nodes rather than on bit rate and coding scheme adaptation.

2.3.4 Impact of PHY on MAC

The well known collision avoidance option in the 802.11 protocol [1] prescribes the exchange of control messages (RTSs and CTSs) in order to mitigate the hidden terminal problem, thus preventing collisions that would result in loss of data and waste of resources. In a MIMO ad hoc network, however, this leads to strong inefficiencies. Specifically, the receiver structure we presented in Section 2.3.3 is able, given some channel knowledge, to separate incoming PDUs which would then not result in a collision, but could instead be detected separately. This crucial channel knowledge at the receiver is obtained through training preambles preceding packet transmission. The networking protocols may then choose how many and which channels to estimate, taking into account that the limited receiver capabilities allow locking onto at most N_S^{max} sequences simultaneously. While doing this, the protocols must be aware of the tradeoff existing between the amount of wanted data to detect and the interference protection granted to those data. In other words, trying to detect too many wanted data packets could leave limited resources for interference cancellation, leading to data loss.

In particular, designing an efficient MAC protocol requires to account for the specific scenarios of interest. In an ad hoc network, nodes transmit from different distances. The average per–PDU received power thus varies among all transmitters, which could result in improved detection performance. Moreover, ad hoc networks typically experience concurrent channel access. Collision avoidance schemes, such as 802.11, try to avoid this concurrency by blocking those nodes that receive the signaling messages. Instead of blocking, we want to encourage simultaneous transmissions and to exploit the receiver's ability to separate multiple incoming signals. To this aim, we start with an assessment of the receiver performance when receiving data PDUs and signaling packets. Even if not exhaustive, this study is indeed important for two reasons. First, it yields some insights on data transmission capability and, more specifically, on the affordable spatial multiplexing as a function of distance. Second, it allows to understand the probability of correctly receiving superimposed signaling messages. This latter parameter is quite crucial, since if this probability is sufficiently high, signaling packets can be relied on as a source of information on neighboring traffic and handshakes.

For this study, we place a node in the center of a circular area of given radius R to act as a receiver. The intended transmitter is moved from 40 to 140 m away from this receiver. The transmitter sends data in blocks of 1000 bits per antenna (*e.g.*, 2000 bits spatially multiplexed through 2 antennas, 4000 bits through 4 antennas, and so on). The maximum power is constrained, and equally divided among the used antennas, since this is the best choice in the absence of channel state information at the transmitter. Moreover, we randomly place inside the area some further (interfering) data senders, which always transmit 1000 bits of data at full power through one antenna. Those nodes falling below a threshold distance $d^* < R$ are considered trackable and their contribution can be detected and canceled, whereas those falling beyond d^* are treated as unknown interferers. The reasoning here is that a node could either have a limited knowledge of its neighborhood, or not wish to detect all incoming signals, but only those with a minimum received power, in order to guarantee detection performance. We have set R and d^* so that the probability that a signal is detected is 50%.

The results of this test are given in Figure 2.7, using 8 interferers in addition to the intended transmitter. The curves show that there exists a tight relationship between the number of used antennas (thus, bit rate) and the maximum coverage distance affordable. For example, with a 90% minimum success ratio objective, a transmitter could reach 70 m, 90 m and 110 m, using 8, 4 and 2 antennas respectively.

To encourage parallelism, RTS and CTS messages do not block transmissions in our scheme, but rather are used for traffic load estimation. Since signaling packets are shorter and transmitted with a single antenna at full power, we expect them to be detectable in large quantities without significant errors. To gather further insight on that, we have considered a similar scenario as before, with 1 to 20 nodes transmitting simultaneously 200 bits long RTSs to a receiver placed at the center of a circular area. Again, all nodes beyond d^* are considered as unknown interferers. Besides the previously explained reason, here d^* also functions as a measure of a node's knowledge of its neighboring network activity. The receiver acquires better traffic awareness if it is able to decode correctly many RTSs, which is possible only if many parallel channels can be tracked (greater N_S^{max}). We simulate such higher tracking capabilities through a greater d^*, and vary it such that the average number of interfering nodes (IN) over all nodes is 30%, 10% and, as a limit case, 2%. Fig. 2.8 summarizes signaling packet capture performance. With the same settings as in Fig. 2.7 (IN $= 30\%$), there is still a fairly high probability of detecting a good percentage of the signaling packets, translating into 13 to 15 correct detections with $N_A = 8$ antennas, even if more packets are sent. This

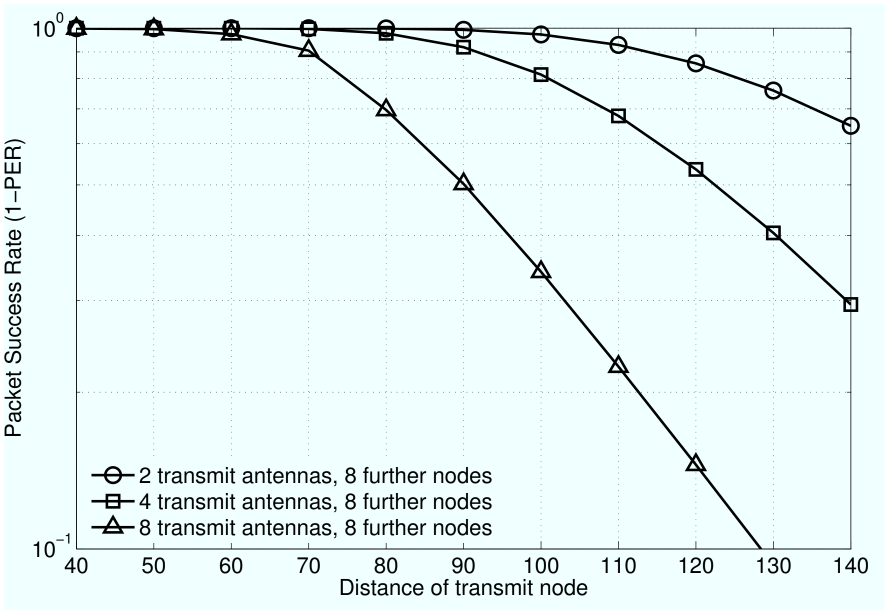


Figure 2.7. *Probability of capturing a data packet in the presence of interfering traffic, for varying number of antennas used by the wanted receiver.*

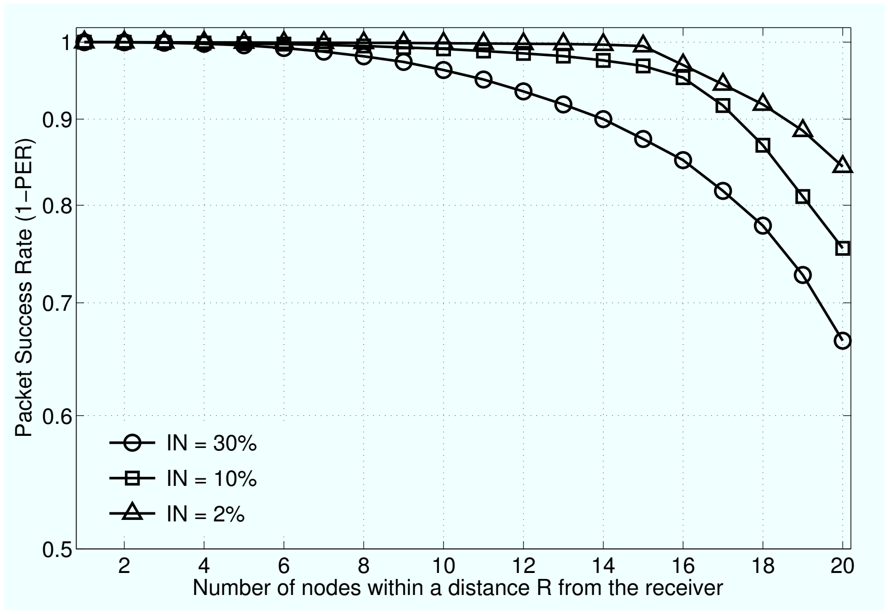


Figure 2.8. *Probability of capturing a signaling packet for varying number of interfering nodes over the total number of transmit nodes.*

value improves if the node can afford to increase its neighborhood knowledge (IN $= 10\%$, 2%). The insight gained here is that relying on the exchange of signaling packets prior to data transmission is a good choice, because it is highly likely that a substantial fraction of these packets is received correctly.

2.4 Cross–layer MAC Design

To gather most of the aforementioned advantages, we resort to a framed communication structure, with four phases. To work correctly, all nodes have to share the same frame synchronization. These phases are designed according to the standard sequence of messages in a collision avoidance mechanism, and are summarized as follows.

2.4.1 RTS phase

In this phase, all senders look into their backlog queue and, if it is not empty, they compose transmission requests and pack them into a single RTS message. Each packet in the queue is split into multiple PDUs of fixed length, such that each PDU can be transmitted through one antenna. For this reason, any request has to specify the number of PDUs to be sent simultaneously, in addition to the intended destination node. How to associate a destination node with a suitable number of transmit antennas is an *RTS policy*, and depends on the degree of spatial multiplexing sought, as well as the local traffic intensity, thus the queue level of the sender. Any RTS may contain several such requests. Moreover, an RTS is always sent with one antenna.

2.4.2 CTS phase

In this phase, all nodes that were not transmitters themselves receive multiple simultaneous RTSs, and apply the reception algorithm of Section 2.3 to separate and decode them. When responding to the correctly received RTSs, nodes have to account for the need to both receive intended traffic (thus increasing throughput) and protect it from interfering PDUs (thus improving reliability). The constraint in this tradeoff is the maximum number of trackable channels, *i.e.*, the maximum number of training sequences a node's circuitry can lock onto. We name a *CTS policy* the way the former is traded off for the latter, *e.g.*, controlling the number of allowed senders and/or the number of allowed antennas. Since this is a design decision, we defer the description of the compared CTS policies to the next Section. CTSs are also sent out using one antenna.

2.4.3 DATA phase

All transmitters receive superimposed CTSs and, after BLAST detection, they follow CTS indications and send their PDUs. Each PDU has a fixed predefined length and is transmitted through one antenna, but a node can send multiple PDUs simultaneously, possibly to different receivers.

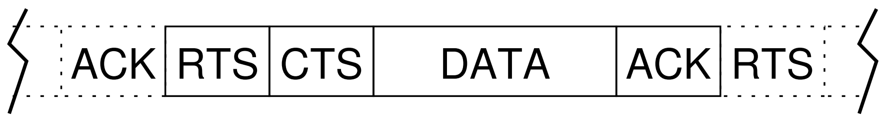


Figure 2.9. *Sequence of messages in a frame.*

2.4.4 ACK phase

After detection, all receivers evaluate which PDUs have been correctly received, compose a cumulative PDU–wise ACK, and send it back to the transmitters. After this last phase, the data handshake exchange is complete, the current frame ends and the next is started. Note that this corresponds to the implementation of a Selective Repeat Automatic Repeat reQuest (SR–ARQ) protocol, where PDUs are individually ACK'ed and, if necessary, retransmitted. The sequence of messages in a frame is depicted in Figure 2.9.

2.4.5 Other MAC details

Before going more deeply into CTS policy definition, we remark that a random backoff is needed for nodes that do not receive a CTS, as otherwise persistent attempts would lead the system into deadlock. In the forthcoming results and in the following Sections, we will make use of a standard exponential backoff. However, we differentiate between node-blocking backoff (or node–lock) and destination-wise-blocking backoff (or dest–lock). The difference is to which extent a node must refrain from sending other RTSs. With the more conservative node–lock, a node does not send any RTS for the whole duration of the backoff. Conversely, dest–lock only blocks RTSs sent to neighbors that did not reply with a CTS at a previous attempt. In either case, the duration of the transmit gap is set to a random number of frames. This amount is uniformly distributed in the interval $[1, BW(i)]$, where i tracks the current attempt, and $BW(i) = 2^{i-1}W$, with W a fixed backoff window parameter. We will elaborate more on backoff strategies later.

Before proceeding, we also highlight that it suffices that nodes be frame–synchronous, even if for simplifying the system description, we have referred to [32] in Section 2.3, where a tighter symbol–level synchronization is assumed. In fact, instead of operating on a per–symbol basis, the receiver can first detect one whole PDU and then cancel it, detect and cancel the second PDU and so forth, until the last one is detected. Frame synchronization is a much weaker assumption, and can be easily implemented with current technology.

2.4.6 RTS and CTS Policies

The last details we need to specify about our MAC protocol are RTS and CTS policies, which are especially important in this context, since efficient data exchange requires that the receivers' detection capabilities are not exceeded, and that sufficient knowledge of the neighborhood is available. Before dealing with MIMO-specific policies, we introduce here a simpler baseline protocol that we will use later for comparison. The definition of this protocol is necessary, since the approaches described in Section 2.2.2 cannot be directly compared to our solution, because of either the absence of a specific MAC scheme [23], the tailoring of MAC around some fixed PHY parameters such as the number of antennas [24], the diverse issues related to different modulation and signaling schemes [25], the attention devoted to achieving full diversity instead of full parallelism [26], or the idealized assumptions about a MIMO PHY level and MAC signaling [27].

Our baseline, instead, is meant as an example of how a layered networking solution would behave when set up on top of a spatial multiplexing-capable MIMO PHY level. Furthermore, it is directly comparable with our policies, as it takes into account the PHY used (unlike, *e.g.*, [27] that focuses on link capacity) and is sufficiently general not to depend on the number of antennas per node (unlike, *e.g.*, [24]). Our baseline works as follows. When a node has a packet to transmit, it senses the channel, gaining access if it finds it free. In order to obtain an optimistic upper bound on the performance of this protocol, we assume that relay selection is "ideal", in the sense bthat one node among the RTS senders is chosen to transmit, whereas the others back off. Also note that the random choice of a node does not yield significant drawbacks, because the nodes of the network simulated in Section 2.5 are within coverage of one another (see the same Section for details about this choice) and therefore any transmission would silence all other handshakes. When a node is granted access rights, it sends an RTS and waits for a CTS from the recipient. To be consistent with the following MIMO transmission policies, data packets are divided in PDUs, each 1000 bits long. Now, the best transmission enhancement obtainable within this protocol is to increase the raw bit rate as much as possible. To this aim, PDUs are evenly split among *all* N_A available antennas and transmitted in succession, *i.e.*, if $N_A = 8$, a block of 125 bit per PDU is sent through each antenna. Before returning to the idle state or performing another transmission attempt, the node waits for an ACK from the receiver, reporting which blocks were detected correctly. In case of errors, only the erroneous blocks are retransmitted.

This baseline is a reasonably simple protocol, yet it makes use of MIMO capabilities and maintains other features, such as carrier sense and contention-based channel access (thus, *no* interference cancellation), similarly to 802.11 DCF. Results based on this scheme will show that a straightforward porting of a layered solution on top of the more powerful MIMO PHY is a suboptimal choice.

RTS policy

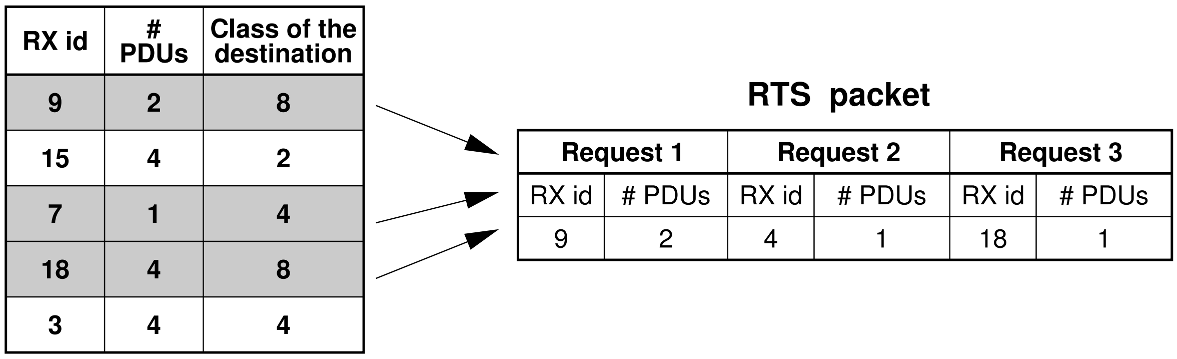
The way to correctly associate a neighbor with the maximum achievable degree of SM relies on considerations about the local traffic intensity and on the distance of the receiver. Let the set of neighbors of a given node s be denoted as $\mathcal{V} = \{v_1, v_2, \ldots\}$. Also let a_{sv_j} be the *maximum* number of antennas to be used by node s when transmitting to a set of nodes that includes v_j, $j = 1, 2, \ldots$. Since we wish to encourage spatial multiplexing, we restrict a_{sv_j}

to be either $a_1 = 2$, $a_2 = 4$, or $a_3 = 8$. Referring to Fig. 2.7, we set three threshold distances δ_1, δ_2, δ_3 corresponding to the maximum reach affordable with a_1, a_2, a_3 transmit antennas, respectively. Then, we let s set $a_{sv_j} = a_i$ if and only if $\delta_{i-1} < d(s, v_j) \leq \delta_i$, where $d(x, y)$ is the distance between node x and node y, and $\delta_0 = 0$. Note that distances are directly related to the average received SNR value, so that an objective function can be chosen for setting threshold distances based on either metric.[4]

Let us focus on node n. The algorithm begins with step $i = 1$. If the node queue is non-empty, a request is created according to the following algorithm. The node sets $k_1 = 1$ and scans the k_1th packet's destination, d_{k_1}, and the number of the packet's unsent PDUs, p_{k_1}. Then, it compares p_{k_1} with $a_{nd_{k_1}}$. If $p_{k_1} > a_{nd_{k_1}}$, it inserts in the RTS the request pair $(d_{k_1}, a_{nd_{k_1}})$ and does not consider any further request. Otherwise, the pair (d_{k_1}, p_{k_1}) is put in the RTS. Node n keeps memory of the queue indices of all packets selected for transmission, maintaining them in set $\mathcal{S}_i$, where i is the step index. It also accumulates in the variable $A(i)$ the number of antennas allocated for selected PDUs until step i. At step 1, $\mathcal{S}_1 = \{k_1\}$, and calling $M(1) = \min\{a_{nd_{k_1}}, p_{k_1}\}$, the maximum feasible number of antennas, then $A(1) = M(1)$.

If $p_{k_1} < a_{nd_{k_1}}$, d_{k_1} could potentially sustain a transmission with $a_{nd_{k_1}} - p_{k_1}$ further antennas (in the absence of interference). In this case there is still room for sending one or more further PDUs belonging to other packets. Therefore, the node proceeds to step $i = 2$ and scans its queue, until it finds a packet k_2 that matches the condition $a_{nd_{k_2}} > A(1)$. This means that the selected packet's destination is reachable with at least as many antennas as its component PDUs (which would be sent through one antenna each). The node sets $\mathcal{S}_2 = \mathcal{S}_1 \cup \{k_2\}$, calculates the maximum number of PDUs allotted to k_2 as $M(2) = \min\big\{\min\{a_{nd_{k_1}}, a_{nd_{k_2}}\} - A(1), p_{k_2}\big\}$, so as not to overcome the antenna constraints $a_{nd_{k_1}}$ and $a_{nd_{k_2}}$ and taking into account that $A(1)$ antennas have already been allotted. Then, it inserts in the RTS the pair (d_{k_2}, M), and finally updates $A(2) = A(1) + M(2)$. If there is still available space for transmission without violating antenna constraints, *i.e.*, $\min_{j\in\mathcal{S}_2}\{a_{nd_j}\} > A(2)$, then the node proceeds to step $i = 3$, looking again in its queue for a packet k_3 such that $a_{id_{k_3}} > A(2)$, and so on. In general, at step i, the node explores the queue for a packet k_i such that $a_{id_{k_i}} > A(i-1)$. Then $\mathcal{S}_i = \mathcal{S}_{i-1} \cup \{k_i\}$, $M(i) = \min\big\{\min_{j\in\mathcal{S}_i}\{a_{nd_j}\} - A(i-1), p_{k_i}\big\}$, and the pair $\big(d_{k_i}, M(i)\big)$ is put in the RTS. The algorithm then proceeds to step $i + 1$ if and only if $\min_{j\in\mathcal{S}_i}\{a_{nd_j}\} > A(i)$ and a packet such that $a_{id_{k_{i+1}}} > A(i)$ can be found in the queue. As an example, consider Figure 2.10. There, and in the following, the maximum number of antennas that can be allowed toward a certain node is shortly referred to as "class." The RTS is formed by first allotting the 2 PDUs required by node 9, whose class is 8. Node 15 cannot be accommodated, because the only 2 antenna transmissions it can sustain have been already allotted to node 9. However, node 9 can sustain 6 more antenna transmissions, which allows to accommodate 1 PDU for node 7. Notice that now node 7's class (4) represents the

[4]Later, we will use a 90% target success rate, and accordingly set $\delta_1 = 70$, $\delta_2 = 90$ and $\delta_3 = 110$ m using Fig. 2.7.



RX id	# PDUs	Class of the destination
9	2	8
15	4	2
7	1	4
18	4	8
3	4	4

RTS packet

Request 1		Request 2		Request 3	
RX id	# PDUs	RX id	# PDUs	RX id	# PDUs
9	2	4	1	18	1

Figure 2.10. *Example of application of the RTS policy. No request for nodes 15 and 3 is included in the RTS, because the maximum number of antennas allowed toward these nodes is too small. In addition, allowing a transmission to node 3 would overload the reception capability of node 4.*

most restrictive constraint, and that 3 PDUs have already been requested so far. This allows a third request to node 18 to ask for only 1 PDU, which completes the RTS construction.

CTS policies

At first, the node sorts all requests contained in every correctly decoded RTS in order of decreasing received power, and divides them in two subsets, namely $\mathcal{W}$ and $\mathcal{U}$, respectively standing for *wanted* and *unwanted*. The first set contains all requests directed to the node, the second set all other requests. Recall that if a request by node s_k implies the transmission of, say, r_k PDUs, the receiver has to account for channel estimation resources that will be needed for all PDU transmissions. Since the maximum number of simultaneous PDUs per receive antenna is limited to N_S^{max}, each time a transmission is granted the number of available tracking resources is decreased by r_k. Therefore, for each request considered, the receiver inserts in the CTS the pair $(s_k, \bar{r}_k)$, where $\bar{r}_k = \min\{r_k, N_S^{max} - \sum_{j=1}^{k-1} r_j\}$, until there are no more available tracking resources. Grants are given according to one of the following policies, with the understanding that no more than N_S^{max} PDUs can be granted.

do Not Follow Traffic (NFT) In this case, the node grants the requests in $\mathcal{W}$ until either they are all granted or all available channel estimation resources are used, and does not consider $\mathcal{U}$ at all.

Follow Traffic (FT) In FT, the node always grants the first (highest-power) request in $\mathcal{W}$ and then considers all requests in $\mathcal{W} \cup \mathcal{U}$, re-ordered by decreasing received power. At step k, if the processed request belongs to $\mathcal{U}$, no grants are given in the CTS, but the number of estimation resources available is decreased according to $\bar{r}_k$. An example is given in Figure 2.11, for 3 wanted and 3 unwanted traffic requests, each with a different number of associated PDUs. By assigning the channel estimation resources to the requesting nodes in order of decreasing SNR, the receiver can accommodate all wanted requests. Moreover, it

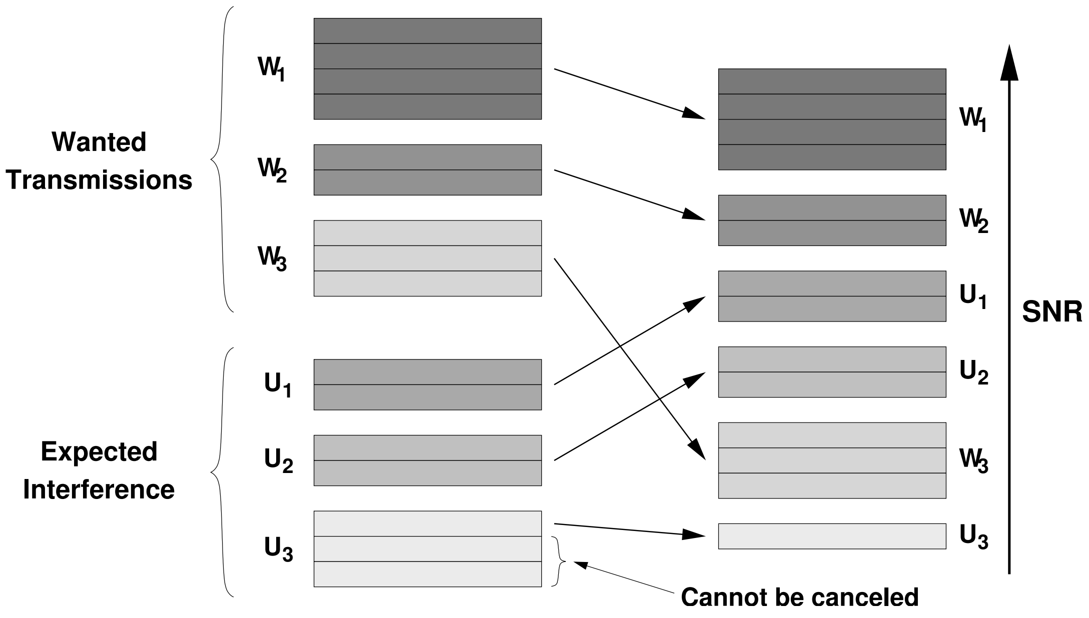


Figure 2.11. *Example of application of the FT policy. Darker grays represent higher SNRs at the receiver. Some of the unwanted PDUs by U_3 cannot be canceled due to limited channel estimation capabilities, and are left as unknown interference.*

can detect (and cancel) the interference from U_1 and U_2, whereas only one PDU from U_3 can be canceled, due to lack of further resources. This policy strives to guarantee some throughput through the allowance of one transmission in $\mathcal{W}$ but prioritizes protection from strong interference through merging $\mathcal{W}$ and $\mathcal{U}$ when choosing which channels it is more convenient to track. In order to show that both these are necessary, we also consider the two following modifications of FT.

Partially Follow Traffic (PFT) With PFT, nodes gives priority to the allowance of wanted transmissions, processing first all requests in $\mathcal{W}$. If there is any tracking resource left, it then begins to consider requests in $\mathcal{U}$ until all resources are exhausted, enabling the cancellation of some neighboring interference. This variant privileges throughput over protection against interference.

Follow Traffic Without Interference Cancellation (FT–WIC) This policy operates as FT, but does not perform cancellation of interfering requests in $\mathcal{U}$. Therefore, the only means of protecting data is refraining from transmission if too many powerful interferers are detected, without explicitly canceling any signal. This scheme is therefore expected to have poor performance and is considered here to stress the importance of interference cancellation.

Observe that all described policies are cross–layer. On the one hand, they manage network access by selecting which incoming PDUs to decode. They perform this control by accounting for the physical layer, which processes PDUs in order of average received power. On the other hand, they force the multiuser detector to decode a subset of PDUs that opti-

Description	**Value**
Number of nodes	25
Antennas per node, N_A	8
Operating band	5.8 GHz ISM
Data rate per antenna	7.5 Mbps
Digital modulation	BPSK
Type of traffic	Poisson, constant rate λ
Backoff window parameter, W	1
Maximum backoff window	32 frames
Signaling packet length	200 bits
PDU length	1000 bits
PDUs per packet	k, randomly chosen in $\{1, 2, 3, 4\}$
Queue buffer capacity	120 PDUs
Packet timeout	2500 frames
Max no. of trackable sequences, N_S^{max}	32
$\{\delta_1, \delta_2, \delta_3\}$	$\{70, 90, 110\}$

Table 2.1. *Relevant simulation parameters.*

mizes the throughput–reliability tradeoff. All these decisions are taken based on information about per–PDU powers, a parameter provided by the PHY layer which is simple (as suggested in [35]) but crucial. Note also that i) CTS policies are the only way to reduce data traffic in this kind of networks, since RTSs/CTSs are not used for channel reservation, but rather as an indication of intention/clearance to transmit, and ii) both RTS and CTS policies favor the creation of multiple point–to–point links, all potentially making use of spatial multiplexing. This is made possible by inserting multiple requests (grants) in the RTS (CTS), each composed of multiple PDUs.

Notice also that the policies proposed here are not bound to a MIMO PHY, but on the contrary, are suitable for use with any decision-feedback multiuser detection-capable PHY. In other words, our policies can operate on top of any PHY that successively detects multiple signals, and cancels their contribution from the received signal prior to the following detections. We chose V–BLAST as one such PHY, since it is a good representative and has recently received a lot of attention [16]. An accurate comparison of the CTS policies is carried out in the following Section.

Before proceeding to the description of simulation results, we remark that the cross–layer design of a network including a MIMO PHY brings many details into the picture, and forces to make choices in order to focus on the main protocol details and effects, without compromising the general understanding of the protocol behavior. Such choices have been made and tuned based on a wide range of preliminary simulations, before the results hereon could be obtained.

2.5 Simulation Results

2.5.1 Simulation Setup

In order to evaluate the MAC design introduced in the previous Section, we deploy 25 nodes with 8 antennas each in a square grid topology with 5×5 nodes and nearest neighbors $25\ \mathrm{m}$ apart. All nodes are static, and we assume that the frame synchronization assumption holds throughout the simulation.[5] Traffic is generated according to a Poisson process of rate λ packets per second per node. Each generated packet is made of k $1000\,\mathrm{bits}$-long PDUs, with k randomly chosen in the set $\{1, 2, 3, 4\}$. Unsent packets are buffered. We test this specific configuration because nodes are all within coverage range of each other—a demanding scenario in terms of interference, required resources, and efficient protocol design. Transmissions are performed in accordance to the MAC protocol described in Section 2.4 and the policies of Section 2.4.6. Finally, we will only make use of node–lock backoff policy here. All other relevant simulation parameters are given in Table 2.1. We have built a fully detailed MATLAB simulator that accurately reproduces the multiuser detection algorithm at the symbol level, on top of which we stack the framed MAC described in Section 2.4 and either the baseline or one of the MIMO-specific RTS/CTS policies.

2.5.2 Comparison among CTS Policies

In Fig. 2.12 we compare all CTS policies in terms of aggregate network throughput as a function of traffic. Throughput is measured here in $\mathrm{Mbit/s}$ in the whole network. NFT shows very poor performance for all traffic values, for two reasons: i) it allows the transmission of all requested PDUs, regardless of whether the receivers can separate them, and ii) it does not cancel any interferer. PFT performs slightly better since, while still granting every requested PDU, it incorporates a mechanism that exploits unused estimation resources for canceling the strongest interfering PDUs (recall that every policy always considers received powers in decreasing order when selecting what to grant or to cancel). This gives PFT some benefit over NFT, because it allows for the cancellation of at least the strongest interfering PDUs. Yet, PFT cannot cope with excessive traffic load. In fact, beginning from λ between 700 and 800, the amount of requested traffic gets heavier, leaving less room for cancellation of unwanted signals, and causing a throughput decrease. This is the key reason why FT performs better than PFT. It merges wanted and interfering PDUs and allots estimation resources to both, while still ensuring that at least one wanted PDU is granted. In the worst case, under exceedingly high traffic, 1 wanted PDU (the one with highest power) would be protected against the $N_S^{max} - 1$ highest-power interferers, in an attempt to let some wanted

[5]How to keep node synchronized in mobile topologies is beyond the scope of this thesis. However, if some mechanism can keep nodes synchronized up to a certain drift, this drift can be compensated for by oversampling the signals received by each antenna. Applying zero forcing as in Section 2.3 to the new correlation matrix (whose is increased according to the oversampling factor) allows to decouple imperfectly synchronized signals, solving the problem at the price of greater signal processing efforts.

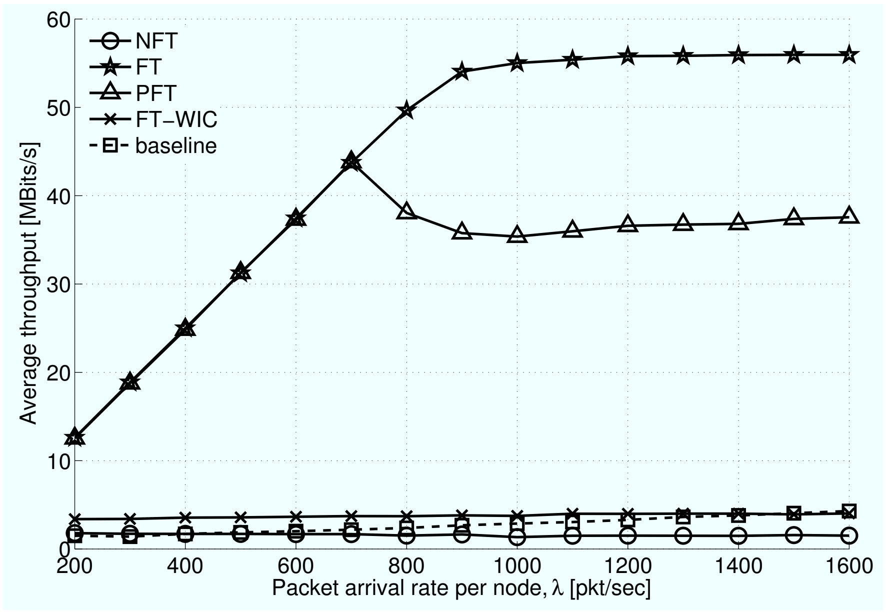


Figure 2.12. *Throughput for all CTS policies as a function of traffic.*

data get through. The net effect is to activate more frequently short-distance links that can sustain more spatial multiplexing, as will be shown in Fig. 2.14.

The importance of interference protection is well highlighted by the FT-WIC policy results. From Fig. 2.12, we recognize the same trend experienced by NFT, just shifted up to some extent. This shift is explained by the FT–like behavior, as FT-WIC limits traffic in the presence of interfering PDUs. Nonetheless, FT-WIC still exhibits poor performance, because it lacks the most important interference cancellation feature. Also note that the baseline transmission policy performs poorly as well, because it does not make full use of the MIMO capabilities, resulting in very low throughput. We highlight that both NFT and FT-WIC experience a slight linear increase at very low traffic, before reaching a saturation throughput value that remains then constant at higher traffic. The initial increase is not shown here both because it is an expected behavior, and because we wish to focus on more specific policies such as PFT and FT. Moreover, it is expected that the throughput does not fall to zero, since some of the signals are eventually transmitted to nearby nodes, where the SNR allows some non-zero probability of correct reception.

As a side remark, the simulations show that the decision feedback multiuser detector of Section 2.3, with FT, is able to support up to 12 successful PDUs per frame on average, which is larger than the maximum number of antennas per node, *i.e.*, 8, even in a fully connected network. This is a very interesting result: it substantiates the need for both a well-designed physical layer and a management protocol, and shows that the number of antennas is a soft limit in MIMO ad hoc networks, if efficient RTS/CTS policies are provided. In this case, to be "efficient" means to favor the effective rejection of multiple access interference.

Figure 2.13 shows the average ratio of successfully received to sent PDUs, and basically

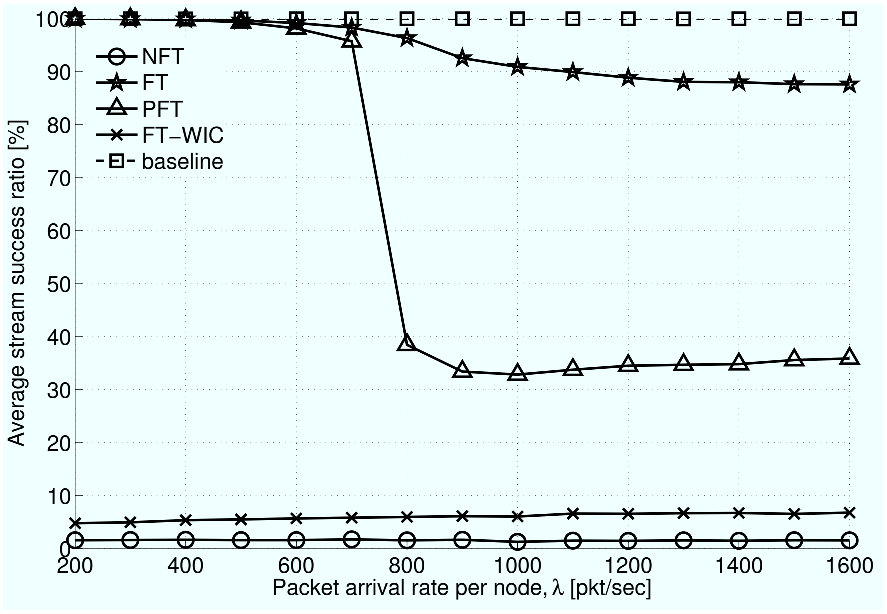


Figure 2.13. *Transmission success ratio of a PDU for all CTS policies as a function of traffic. Notice the more effective interference protection capabilities of FT, that allow a good success ratio even at high traffic.*

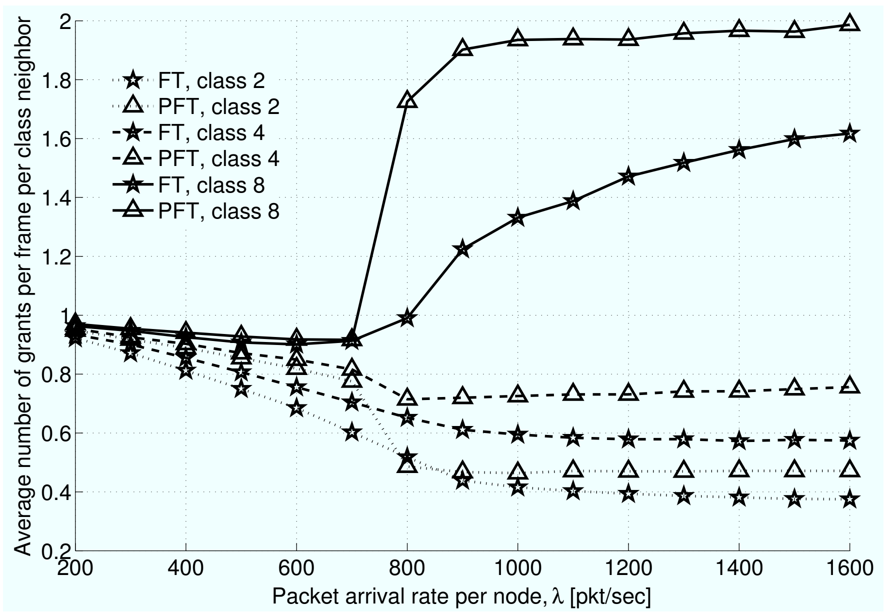


Figure 2.14. *Number of grants given to neighbors with different reception capabilities per frame as a function of traffic. Only the FT and PFT policies are displayed.*

confirms the previous statements. FT achieves the best results and still almost ensures a 90% probability of correct detection at the highest traffic. On the contrary, NFT and FT-WIC incur a very low probability of detection success, and PFT stands in between, its chances being smaller than 40% at high λ. Conversely, the success ratio of the baseline protocol is nearly 100% as expected, since very few transmissions take place due to the collision avoidance mechanism.

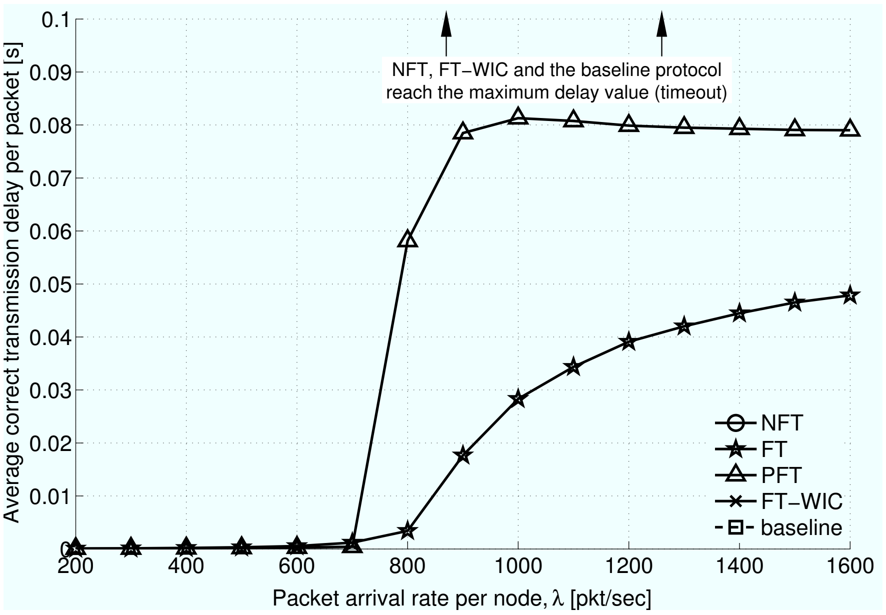


Figure 2.15. *Delay before a correct packet transmission for FT and PFT as a function of traffic. Some curves are not displayed to focus on the comparison between FT and PFT.*

To corroborate the claim that the use of FT at high traffic turns into a more likely activation of short links, we depict in Figure 2.14 the number of grants given to each neighbor depending on the maximum number of antennas allowed for use with that neighbor (shortly referred here as its "class"). We show just PFT and FT, which achieve the most significant results, since they do not reach the same degree of congestion as NFT and FT-WIC. We observe that after starting from nearly one grant per node per class, both FT and PFT incur a progressively stronger decrease in the number of transmissions allowed toward neighbors in classes 2 and 4. On the other hand, transmissions toward class 8 neighbors increase more steeply than the others decrease. This is due to FT's conservative behavior: it is highly likely that the request with stronger power comes from a close class 8 node and that other resources are dedicated to dealing with interference. The number of class 8 grants increases also for PFT, but this is only a consequence of the greater flexibility given by class 8 nodes, that can afford higher SM and thus allow the receiver to give more clearances (see the RTS policy in Section 2.4.6).

The results described before are also confirmed by Figures 2.15 and 2.16, which show the average packet delay in seconds and the average queue length, respectively. Consistently with previous results, we observe that only PFT and FT provide some limited delay, even for higher traffic values. More specifically, PFT reaches a saturation delay of approximately 375 frames. For the same traffic values, the higher throughput achieved by FT is still capable of keeping the network uncongested, explaining the smoother increase in delay. Similar considerations apply to the behavior of the queue length as a function of traffic. In this case, the lower PFT throughput does not allow sufficient packet delivery capabilities, hence the node buffers are filled at $\lambda \geq 800$. In FT, instead, a higher amount of data gets through, re-

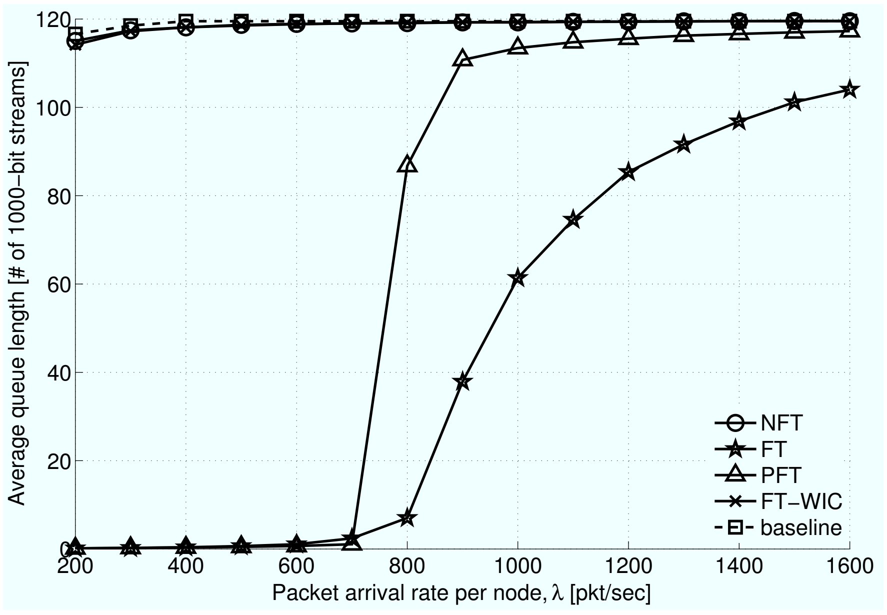


Figure 2.16. *Queue length for all CTS policies as a function of traffic*

sulting in a shorter queue length. All other policies including the baseline protocol perform much worse, as their average delay is close to the upper bound imposed by the value of the timeout.

Overall, the presented results show that the effective cross–layer design that led to FT achieves satisfactory performance, as it allows high throughput and success ratio, hence limited delay and backlog. The results also highlight the difference between FT-WIC and FT, thus the importance of interference cancellation when protecting wanted data. This is particularly important when simultaneous channel access is enforced in order to exploit spatial multiplexing-capable receivers. Finally, we remark that considering received requests in order of decreasing received power tends to favor shorter links (with greater SINR) at high traffic. As a side note, this might have an impact on routing, as under heavy traffic it may be more convenient to set up longer paths with multiple, more robust hops.

2.6 Analytical Model for the BLAST Receiver

The results commented in the previous Section have been obtained through fully-detailed simulations. This requires to reproduce PHY-level operations completely, *i.e.*, to fully implement every step of the BLAST detection, from signal ordering based on received SNR to the correlation matrix pseudo-inversion to obtain weights, to the actual successive cancellation of detected signals. In particular, the most expensive operation is the pseudo-inversion, which is however a fundamental step of the process.

Due to the complexity of the BLAST detection, obtaining statistically meaningful network-level results becomes computationally very demanding. This is especially true when combining the bit–level time scale (required to simulate MIMO receiver processing) with that on which network protocols typically operate, that is much longer. In principle, it could be possible to simulate all scenarios of interest, but this would take a very large amount of time

In order to alleviate the task of reproducing every single MIMO transmission and reception step, two approaches may be followed. The first is to use approximations of the overall "gain" (over traditional links) that could be obtained when operating MIMO links. This is a viable solution, but may lead to some unwanted oversimplification and to results that may not be completely meaningful. The importance of a good and realistic physical model when assessing ad hoc networking issues has been stressed in the recent literature (*e.g.*, see [21]). The second option is to develop approximate formulas for the whole process involving spatial superposition, interference cancellation, and any other MIMO feature that takes part in the symbol transmission and detection phases, rather than just reducing it to a simple global gain factor. Such formulas should be fast to evaluate, while being able to reproduce the real behavior of the implemented technology with sufficient precision. If both these conditions can be met, one could proceed to network evaluation by means of a pseudo-analytic approach, whereby every protocol stage is simulated, whereas the behavior of the PHY layer is reproduced through approximated formulas.

The purpose of this Section is to introduce and compare two such formulas under different points of view, namely complexity, precision, and suitability to network simulations. As to the last two points, we wish to understand where is the best tradeoff between a precise PHY model and a good network emulation. In other words, we seek an approach that yields good network-level results, although possibly looser PHY-level ones.

Before proceeding with the description of the approximations, recall the model and terminology introduced in Section 2.3. In particular, let us define again Internal Antennas (IAs) as those sending signals whose channel is estimated. These signals are detected and canceled within the BLAST process. Conversely, External Antennas (EAs) are those whose channel cannot be estimated and that are treated as pure interference. Now, the performance of a spatial multiplexing system is affected by a) imperfect IA cancellation due to detection errors, b) EA interference, c) residual interference due to the pseudo-inverse of the correlation matrix $\boldsymbol{R}(i)$, and d) noise. We model here, with two approaches, the impact of imperfect cancellation on the detection of forthcoming PDUs. In the first approach, denoted *Gaussian technique* and described in Section 2.6.1, the interference is considered as additional Gaussian noise. In the second approach of Section 2.6.2, denoted *enumeration technique*, we exhaustively enumerate all the various configurations of detection errors and compute the conditional BER for each configuration. Although the enumeration technique is more accurate than the Gaussian technique, the number of configurations to be explored increases exponentially with the number of IA, with a consequent increase in computation time. Hence, in Section 2.6.3 we derive a suboptimal technique, denoted*pruned tree technique*, where only the most relevant error configurations are explored.

In all cases, EA interference is approximated as Gaussian noise, with autocorrelation matrix

$$\boldsymbol{R}_{\mathrm{EA}} = \mathrm{E}[\boldsymbol{i}_{\mathrm{EA}} \boldsymbol{i}_{\mathrm{EA}}^H] = \boldsymbol{H}^H \bar{\boldsymbol{H}} \bar{\boldsymbol{\Sigma}} [\boldsymbol{H}^H \bar{\boldsymbol{H}}]^H \,, \tag{2.3}$$

where $\bar{\boldsymbol{\Sigma}}$ is a diagonal matrix with entries $\bar{\boldsymbol{\Sigma}}_{\ell,\ell} = \sigma_{s'}^2(N_d + \ell)$, $\ell = 1, 2, \ldots, U - N_d$. Then the power of EA interference after weighing can be written as $\sigma_{\mathrm{EA}}^2(k_i) = ||\boldsymbol{w}(i)^T \boldsymbol{R}_{\mathrm{EA}}||^2$.

2.6.1 Gaussian technique

We model the residual interference due to errors in the detection process as an *error signal* with Gaussian statistics, zero mean, and variance depending on the error rate. This approach has already been considered in literature, for some particular transmission scenarios. Using the Gaussian approximation the average BER has been obtained for Rayleigh fading channels when $N_d \leq N_A$, [36, 37]. In this subsection we provide the general expression even for the case of $N_d > N_A$ and for any channel $\boldsymbol{H}$, rather than for a specific channel statistics. This provides a tool for evaluating network performance even when active nodes have different channel statistics.

Assuming a BPSK transmission, the transmitted signal s_k takes a value $\pm\sigma_s(k)$, that depends on the number of active antennas. Then, the error signal for PDU k is

$$e_k = \hat{s}_k - s_k \in \{-2\sigma_s(k), 0, 2\sigma_s(k)\} \,. \tag{2.4}$$

When the received PDU k is affected by a BER $P_e(k)$, $e_k = 0$ with probability $1 - P_e(k)$, $e_k = -2\sigma_s(k)$ with probability $P_e(k)/2$ and $e_k = 2\sigma_s(k)$ with probability $P_e(k)/2$. Note that an erroneous cancellation actually doubles interference. Hence, the variance of the interference signal due to PDU k can be written as

$$\sigma_e^2(k) = \mathrm{E}[e_k^2] = 4\sigma_s^2(k) P_e(k) \tag{2.5}$$

and the SNIR for PDU k_i can be written as

$$\gamma(k_i) = \frac{|\boldsymbol{w}(i)^T \boldsymbol{R}_{\cdot,k_i}|^2 \sigma_s^2(k_i)}{\frac{N_0}{2}||\boldsymbol{w}(i)^T \boldsymbol{H}^H||^2 + \sigma_{\mathrm{EA}}^2(k_i) \;+ \displaystyle\sum_{k \in \mathcal{K}(i)} |\boldsymbol{w}(i)^T \boldsymbol{R}_{\cdot,k}|^2 \sigma_e^2(k) \;+ \displaystyle\sum_{k \in \mathcal{N} \setminus \mathcal{K}(i+1)} |\boldsymbol{w}(i)^T \boldsymbol{R}_{\cdot,k}|^2 \sigma_s^2(k)} \,. \tag{2.6}$$

The first term of the denominator in (2.6) is the power of noise, with power spectral density $N_0/2$ since we are considering a real constellation. The second term of the denominator accounts for the interference due to EA. The power of the residual interference due to imperfect cancellation is provided by the third term in the denominator. With the last term in the denominator we have also inserted the interference due to IA not yet canceled, since when the number of receiving antennas is less than the number of undetected IA, the vector $\boldsymbol{w}(i)$ is not able to completely remove the interference due to PDUs $k_{i+1}, k_{i+2}, \ldots, k_{N_d}$. Note that, from the definition of $\mathcal{K}(i)$, $\mathcal{N} \setminus \mathcal{K}(i+1)$ is the set of all IA not yet canceled, not including the IA k_i.

The BER for a BLAST transmission over channel $\boldsymbol{H}$ with an uncoded BPSK modulation, is given by $P_e(k_i) = Q\big(\sqrt{\gamma(k_i)}\,\big)$, where $Q(\,\cdot\,)$ is the complementary Gaussian distribution.

2.6.2 Enumeration technique

As we will show in Section 2.7.1, the Gaussian technique is not always accurate, especially with an increasing number of transmitting antennas. Hence, we propose here the *enumeration technique* which takes into account the exact interference statistics by enumerating, at stage i of BLAST, all possible error configurations of the previously detected PDUs $k_1, k_2, \dots, k_{i-1}$ and their impact on the probability of erroneous detection of PDU k_i. A similar approach has been considered in [38] for the computation of the average BER with the assumptions of two transmit antennas, high signal to noise ratio, and Rayleigh fading. Here we generalize the technique for any channel statistics and any number of antennas.

Define the ordered error vector until detection of PDU k_i as $\boldsymbol{e}^{(i)} = [e_{k_1}, e_{k_2}, \dots, e_{k_{i-1}}]^T$, where entries are provided by (2.4). The error probability on PDU k_i can be conditioned on the error configuration of previously canceled PDUs $\bar{\boldsymbol{e}}$, obtaining, for $i = 2, 3, \dots N_d$,

$$\mathrm{P}[e_{k_i} = a] = \sum_{\substack{\bar{e}_\ell \in \mathcal{V}_\ell \\ \ell=1,\dots,i-1}} \mathrm{P}[e_{k_i} = a \big| \boldsymbol{e}^{(i)} = \bar{\boldsymbol{e}}]\, \mathrm{P}[\boldsymbol{e}^{(i)} = \bar{\boldsymbol{e}}]\,, \tag{2.7}$$

where $\mathcal{V}_\ell = \{-2\sigma_s(k_\ell), 0, 2\sigma_s(k_\ell)\}$, contains the variance of $\bar{e}_\ell$, $\ell = 1, \dots, i-1$. Moreover, $a \in \{-2\sigma_s(k_i), 2\sigma_s(k_i)\}$, the summation is taken over all possible error configurations $\bar{\boldsymbol{e}}$ of the $(i-1)$ detected PDUs, and $\mathrm{P}[\cdot]$ denotes probability. For the first detected PDU, we have no previous errors, thus

$$\mathrm{P}[e_{k_1} = +2\sigma_s(k_1)] = \mathrm{P}[e_{k_1} = -2\sigma_s(k_1)] = P_e(k_1)/2\,. \tag{2.8}$$

The expression (2.7) can be computed more efficiently by considering the associated tree of error configurations. An example of error tree is shown in Figure 2.17 (where for simplicity we set $\sigma_s(i) = 1$, $i = 1, 2, \dots, N_d$). Starting from the root (level 0), the ith level of the tree corresponds to the detection of PDU k_i, and each node corresponds to a different error configuration of the previously detected symbols. In particular, for the first PDU k_1, corresponding to the first level after the root, there is only one configuration, since there are no previous detections. Three branches depart from the root, corresponding to the three possible error configurations of $\hat{s}_{k_1}$, *i.e.*, $-2\sigma_s(k_1), 0, 2\sigma_s(k_1)$. The second level corresponds to the detection of the second PDU k_2 and there are three possible error configurations of the previously detected PDU k_1.

The general level i has a total of 3^i nodes, each representing one term of the summation in (2.7). A node at level i is identified by an error configuration vector $\boldsymbol{e}^{(i)}$ and an error value e_{k_i}. For the computation of the error probability of each term of (2.7) we must consider the signal at the input of the detector, given a specific error configuration. Given an error vector $\boldsymbol{e}^{(i)}$, the signal at the detector input can be written as

$$\tilde{s}_{k_i} | \boldsymbol{e}^{(i)} = s_{k_i} + \sum_{\ell=1}^{i-1} \boldsymbol{w}(i)^T \boldsymbol{R}_{\cdot,k_\ell} \boldsymbol{e}_\ell^{(i)} + \sum_{k \in \mathcal{N} \setminus \mathcal{K}(i+1)} \boldsymbol{w}(i)^T \boldsymbol{R}_{\cdot,k} s_k + \boldsymbol{w}(i)^T \boldsymbol{\nu} + \boldsymbol{w}(i)^T \boldsymbol{i}_{\mathrm{EA}}\,. \tag{2.9}$$

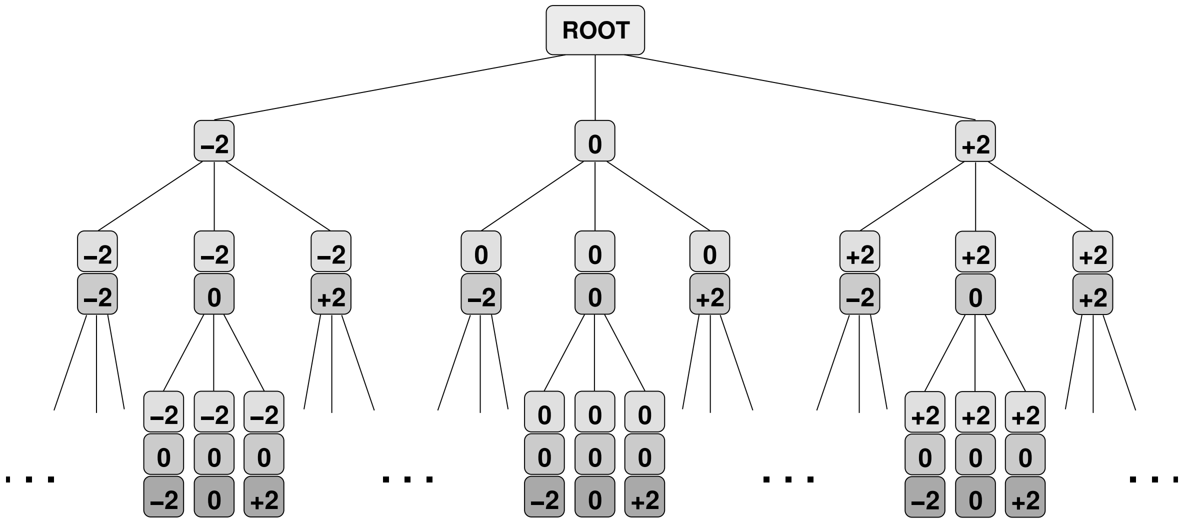


Figure 2.17. *Error configuration tree for* $\sigma_s(i)=1$, $i=1,2,\ldots,N_d$.

Let us define the two conditional SNIRs

$$\gamma_{s_{k_i}=\pm\sigma_s(k_i)|\boldsymbol{e}^{(i)}} = \frac{\left| \pm\, \boldsymbol{w}(i)^T \boldsymbol{R}_{\cdot,k_i}\sigma_s(k_i) + \sum_{\ell=1}^{i-1} \boldsymbol{w}(i)^T \boldsymbol{R}_{\cdot,k_\ell}\boldsymbol{e}_\ell^{(i)} \right|^2}{\frac{N_0}{2}||\boldsymbol{w}(i)^T\boldsymbol{H}^H||^2+\sigma^2_{\mathrm{EA}}(k_i)+\sum_{k\in\mathcal{N}\setminus\mathcal{K}(i+1)} |\boldsymbol{w}(i)^T\boldsymbol{R}_{\cdot,k}|^2\sigma_s^2(k)}\,, \tag{2.10}$$

where we observe that the interference due to detection errors produces a shift in the position of the received signal, and hence changes its power in the numerator of the SNIR.

We must now consider the following cases, according to the transmitted signal and the level of interference due to error propagation. The conditional error probability for PDU k_i, assuming equally likely data symbols, can be written as in equations (2.11)–(2.14):

$$\mathrm{P}\left[e_{k_i} = 2\sigma_s(k_i) \middle| \boldsymbol{e}^{(i)}, \boldsymbol{w}(i)^T\boldsymbol{R}_{\cdot,k_i}\sigma_s(k_i) \geq \sum_{\ell=1}^{i-1}\boldsymbol{w}(i)^T\boldsymbol{R}_{\cdot,k_\ell}\boldsymbol{e}_\ell^{(i)} \right] = Q\left(\sqrt{\gamma_{s_{k_i}=-\sigma_s(k_i)|\boldsymbol{e}^{(i)}}}\right) \tag{2.11}$$

$$\mathrm{P}\left[e_{k_i} = -2\sigma_s(k_i) \middle| \boldsymbol{e}^{(i)}, \boldsymbol{w}(i)^T\boldsymbol{R}_{\cdot,k_i}\sigma_s(k_i) \geq -\sum_{\ell=1}^{i-1}\boldsymbol{w}(i)^T\boldsymbol{R}_{\cdot,k_\ell}\boldsymbol{e}_\ell^{(i)} \right] = Q\left(\sqrt{\gamma_{s_{k_i}=\sigma_s(k_i)|\boldsymbol{e}^{(i)}}}\right) \tag{2.12}$$

$$\mathrm{P}\left[e_{k_i} = 2\sigma_s(k_i) \middle| \boldsymbol{e}^{(i)}, \boldsymbol{w}(i)^T\boldsymbol{R}_{\cdot,k_i}\sigma_s(k_i) < \sum_{\ell=1}^{i-1}\boldsymbol{w}(i)^T\boldsymbol{R}_{\cdot,k_\ell}\boldsymbol{e}_\ell^{(i)} \right] = \left[1-Q\left(\sqrt{\gamma_{s_{k_i}=-\sigma_s(k_i)|\boldsymbol{e}^{(i)}}}\right)\right] \tag{2.13}$$

$$\mathrm{P}\left[e_{k_i} = -2\sigma_s(k_i) \middle| \boldsymbol{e}^{(i)}, \boldsymbol{w}(i)^T\boldsymbol{R}_{\cdot,k_i}\sigma_s(k_i) < -\sum_{\ell=1}^{i-1}\boldsymbol{w}(i)^T\boldsymbol{R}_{\cdot,k_\ell}\boldsymbol{e}_\ell^{(i)} \right] = \left[1-Q\left(\sqrt{\gamma_{s_{k_i}=+\sigma_s(k_i)|\boldsymbol{e}^{(i)}}}\right)\right] \tag{2.14}$$

The probability of a generic node at level i characterized by the error vector $\bar{\boldsymbol{e}}^{(i)}$ and the error value e_{k_i}, can be obtained as $\mathrm{P}\left[e_{k_i} = \bar{e}_{k_i} | \boldsymbol{e}^{(i)} = \bar{\boldsymbol{e}}^{(i)}\right], i = 2, 3, \ldots, N_d$, where $\mathrm{P}[e_{k_1} = \bar{e}_{k_1}]$ is provided by (2.8). This calculation needs averaging over the transmitted symbol, since the same $\boldsymbol{e}^{(i)}$ has a different impact on the error probability, according to the signal sent. Further, averaging $\mathrm{P}\left[e_{k_i} = \bar{e}_{k_i} | \boldsymbol{e}^{(i)} = \bar{\boldsymbol{e}}^{(i)}\right]$ over all $\boldsymbol{e}^{(i)}$ yields the total probability of wrong detection for PDU k_i.

Lastly, note that we could also consider the various error configurations of EA for the SNIR evaluation. In this case, since EA are not detected, the error signal would be equal to $-\sigma_{s'}(k)$ or $\sigma_{s'}(k)$ with equal probability $1/2$, for $k = N_d + 1, N_d + 2, \ldots, U$. Even though this modeling would provide more accurate results than approximating EA as Gaussian noise, it would further increase the computational complexity of the enumeration technique.

2.6.3 Pruned tree technique

In order to reduce the complexity of the enumeration technique, we limit the exploration of the tree to just a few branches. Sub-trees departing from nodes with more than n_e errors are approximated with an upper bound on the error probability, assuming that when more than n_e errors occur in interference cancellation, the average error probability in forthcoming detections is high. Hence, all nodes of the subtrees having a sub-root with $n_e + 1$ errors are characterized by an error probability of 0.5.

For example, for $n_e = 0$ we obtain

$$\begin{aligned} \mathrm{P}[e_{k_i} \neq 0]|_{n_e=0} &\approx \mathrm{P}[e_{k_i} \neq 0 | \boldsymbol{e}^{(i)} = \mathbf{0}]\mathrm{P}[\boldsymbol{e}^{(i)} = \mathbf{0}] \\ &+ \frac{1}{2}\left[1 - \mathrm{P}[\boldsymbol{e}^{(i)} = \mathbf{0}]\right] . \end{aligned} \tag{2.15}$$

This case is similar to the upper bound derived in [39], where it was assumed that given a decision error in an earlier stage of BLAST, the probability of a subsequent decision error is 1. In that case the upper bound is as in (2.15) with 1 instead of 0.5 as weight of the summation.

For a general value of n_e, the error configurations with at most n_e errors before the detection of the ith PDU are in the set

$$\mathcal{E}^{(i)}(n_e) = \left\{ \boldsymbol{e} : \sum_{\ell=1}^{i-1} \frac{|e_{k_\ell}|}{2\sigma_s(k_\ell)} \leq n_e \right\} . \tag{2.16}$$

The error probability for node k_i is then approximated as

$$\begin{aligned} \mathrm{P}[e_{k_i} \neq 0]|_{n_e} &\approx \sum_{\bar{\boldsymbol{e}} \in \mathcal{E}^{(i)}(n_e)} \mathrm{P}[e_{k_i} \neq 0 | \boldsymbol{e}^{(i)} = \bar{\boldsymbol{e}}]\mathrm{P}[\boldsymbol{e}^{(i)} = \bar{\boldsymbol{e}}] \\ &+ \frac{1}{2}\left[1 - \sum_{\bar{\boldsymbol{e}} \in \mathcal{E}^{(i)}(n_e)} \mathrm{P}[\boldsymbol{e}^{(i)} = \bar{\boldsymbol{e}}]\right] . \end{aligned} \tag{2.17}$$

The last term weighed by $1/2$ accounts for the probability of configurations with more than n_e errors. We observe that the error propagation tree can then be pruned of all nodes that

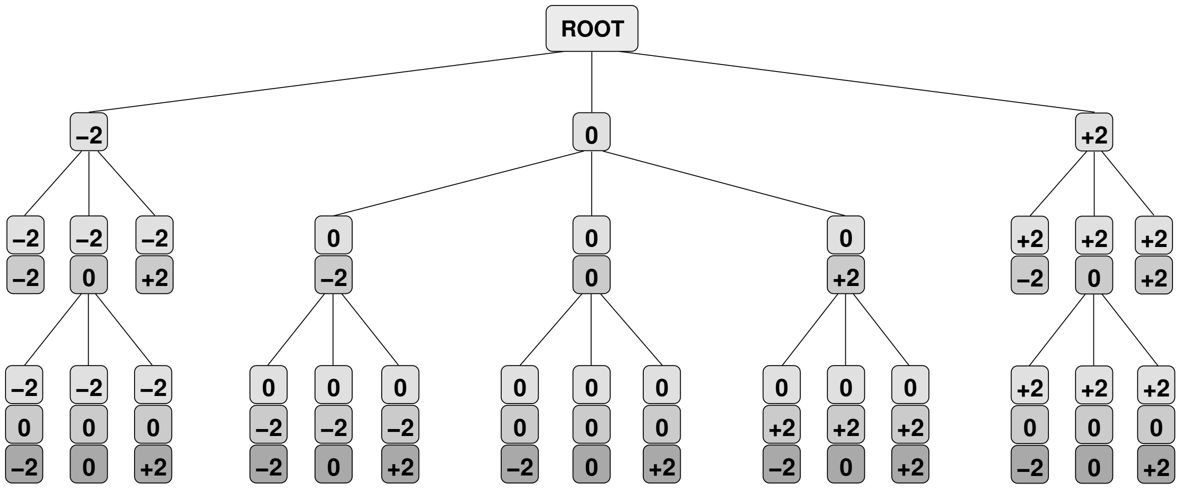


Figure 2.18. *Pruned error tree for $n_e = 2$ and $N_d = 4$, $\sigma_s(i) = 1$, $i = 1, 2, 3$.*

have more than n_e errors, since subtrees departing from these nodes are approximated with an error probability of 0.5. Figure 2.18 shows an example of a pruned tree for $n_e = 2$, and $N_d = 4$ total IA PDUs, where at most one detection error is explicitly accounted for and any configuration with more than one error is assumed to yield a correct detection probability of 0.5 in all subsequent stages (corresponding to dead leaves in the tree).

2.6.4 Computational complexity comparison

Various algorithms have been proposed for an efficient implementation of BLAST. In [40] a recursive algorithm for the matrix inversion has been proposed. In [41] a square-root based algorithm is proposed, which has been further optimized in the improved square-root (ISR) algorithm [42]. Here we consider ISR as the most efficient and reliable technique, and hence all simulations and semianalytical techniques have a common base complexity of $(2/3)N_d^3 + (7/2)N_d^2 N_A + \mathcal{O}(N_d^2 + N_d N_A)$ complex multiplications.

For the Gaussian technique, the function Q is computed N_d times, to obtain the corresponding BERs. The computation of Q can be performed by a look-up table and we bound the complexity of each function computation with the equivalent of one complex multiplication. Moreover, the SNIR must be computed N_d times, according to (2.6), which requires

$$\begin{aligned} C_\gamma &= (N_A{+}3{+}1) + [(1{+}1{+}N_A) + N_d(N_A{+}1) + 1] \\ &= N_d N_A + \mathcal{O}(N_d{+}N_A)\,. \end{aligned} \tag{2.18}$$

products and ratios. Hence, the Gaussian technique has a complexity

$$C_G = (C_\gamma{+}1)N_d = N_d^2 N_A + \mathcal{O}(N_d^2{+}N_A N_d)\,. \tag{2.19}$$

For the enumeration technique, we must compute N_d BER functions for each leaf of the error tree. The computation of the conditional SNIRs (2.10) has the same complexity as (2.6). The complete error tree has 3^{N_d-1} leaves, leading to an overall complexity

$$C_E = 2 \cdot 3^{N_d-1} C_G \,. \tag{2.20}$$

For the pruned tree technique, the number of explored leaves is $\lambda(n_e) = \sum_{k=0}^{n_e} \binom{N_d}{k} 2^k$. Hence, the complexity of the pruned error tree algorithm is

$$C_P(n_e) = 2 \cdot \lambda(n_e) C_G \,. \tag{2.21}$$

Note that in (2.20) and (2.21) the factor 2 accounts for the double computation of Q as required by (2.11)–(2.14). We observe that the complexity of the enumeration and pruned tree techniques grows exponentially with N_d, while the Gaussian technique grows as its third power. For example, with $N_d = 8$ the ratio between the enumeration and the Gaussian technique is $C_E/C_G = 39366$, while for $n_e = 1$ we have $C_P/C_G = 42$. We conclude that the enumeration technique has a significantly higher complexity than the Gaussian technique and it may be even more demanding than the bit-by-bit simulation. The pruned tree technique instead involves a limited increase of complexity with respect to the Gaussian technique.

2.7 Numerical Results using Approximations

In this Section, we carry out relevant physical level as well as network level simulations that exploit the analytical framework deployed in Section 2.6, and compare results with fully detailed bit–level simulations. A very important conclusion that can be drawn from these results is that a complete and accurate physical layer modeling is crucial both for deciding how different layers interact and for tuning relevant communication parameters, since oversimplified models may reduce the statistical meaning of simulation results. PHY results are reported in subsection 2.7.1, whereas MAC level metrics are given and commented in subsection 2.7.3.

2.7.1 Bit error rate

In this Section we compare the simulated performance of a generic node in the network with the analytical results derived in Section 2.7.1, considering channel gains having complex Gaussian statistics with zero mean and unit variance. We have compared the simulated results for different configurations of transmitted PDUs N_d and receive antennas N_A. The receiving node is assumed to detect all PDUs so that N_d is also the number of detected PDUs. We begin by examining this simpler setup as it corresponds to a special-case network with lower load, where all transmitters are placed at the same distance from the receiver, and each of them uses a single antenna at full power. Even though this setting may not be representative of a real network, it is a significant benchmark to consider before moving to more realistic scenarios.

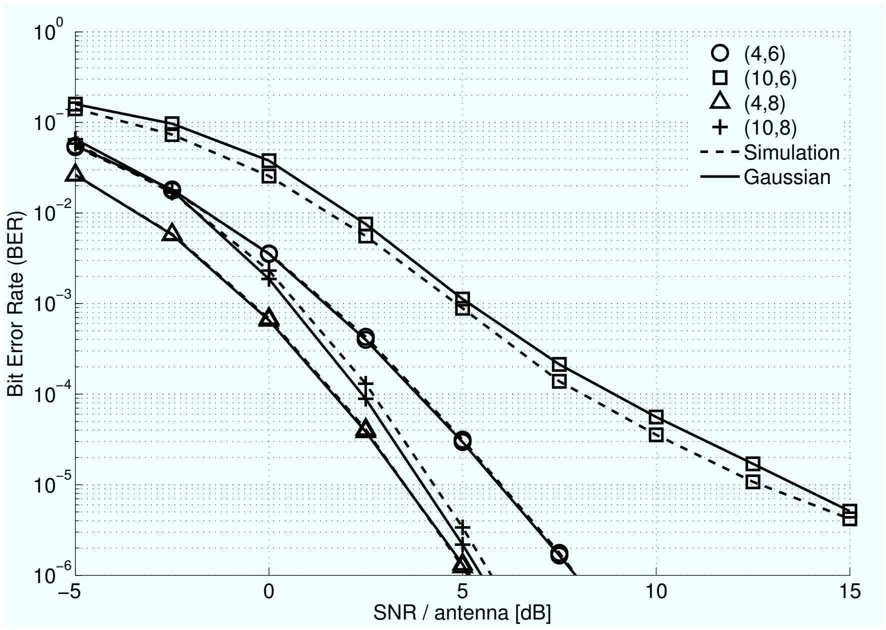


Figure 2.19. *Comparison between analytical (using the Gaussian technique) and simulated BER results for various configurations (N_d, N_A), with N_d the detected PDUs (all assumed to be IA) and N_A the receive antennas.*

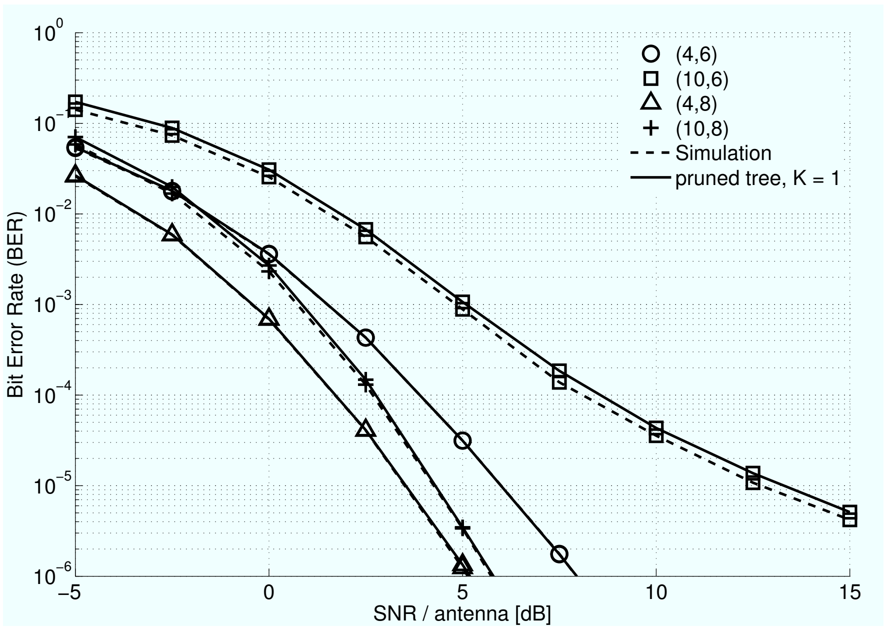


Figure 2.20. *Performance of the pruned tree technique for BLAST BER evaluation for various configurations (N_d, N_A), with N_d the detected PDUs (all assumed to be IA) and N_A the receive antennas.*

In Figure 2.19 we plot simulation results as compared to the Gaussian technique for the case of a single link. The BER is averaged over all detected PDUs and various channel realizations and is shown as a function of the average signal to noise ratio (SNR) per antenna. The average BER obtained with the pruned tree technique is shown in Figure 2.20.

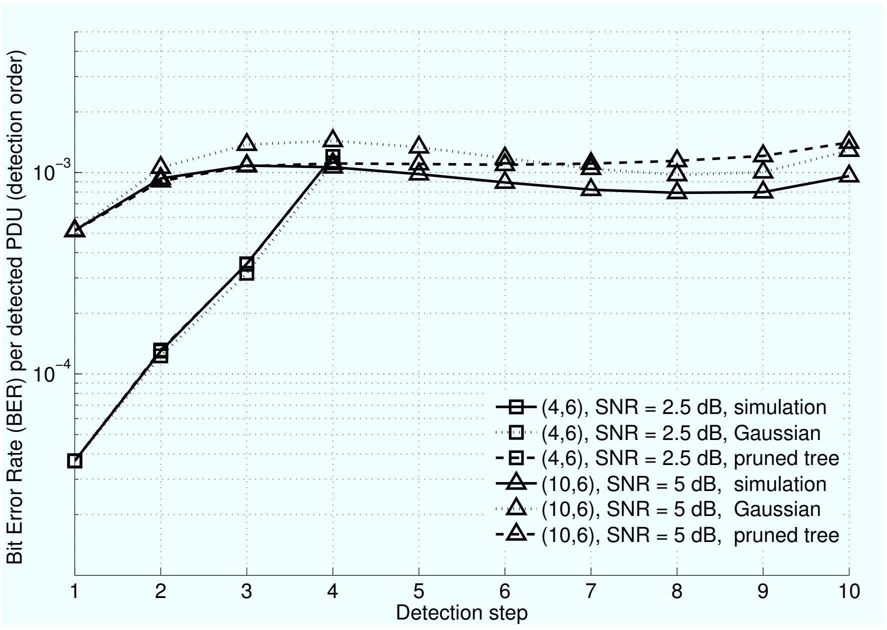


Figure 2.21. *Analytical and simulated BER* per detected PDU *for antenna configurations* $(4, 6)$ *and* $(10, 6)$ *and* $\mathrm{SNR} = 2.5$ *dB and* 5 *dB, respectively. The abscissa lists the PDU index as per the detection order.*

Results predicted by the Gaussian approximation technique differ from the simulated ones for two main reasons: i) the interference due to error propagation is not Gaussian, due to different received power levels and signal ordering, and ii) the interference due to imperfect zero forcing (ZF) is also not Gaussian. Therefore, the approximation is more accurate at higher SNR, where fewer errors occurs and the error propagation phenomenon is reduced [37]. Moreover, as long as there are fewer PDUs than receive antennas, ZF is effective in removing all interference and analysis is in accordance with simulation. When $N_d > N_A$, ZF BLAST yields additional interference that is not Gaussian, providing a further mismatch with simulated results.

The evaluation of the average BER confirms that the enumeration technique has a better match than the pruned error tree approach. For the enumeration technique, simulated and analytical BER exhibit a perfect match when $N_d \leq N_A$, while for $N_d > N_A$ a slight mismatch is present, due to the approximation of imperfect ZF as Gaussian interference.

Results reported until now are for the average BER over all detected PDUs. However, the BER of individual PDUs may differ from the average behavior, due to different level of interference and power. The per-user BER, averaged with respect to the channel statistics, is shown in Figure 2.21 for two antenna configurations. The PDU index refers to the detection order and variations in the BER among different users result from the combination of different received powers, error propagation and mutual interference. It is important to observe that the Gaussian approximation can be more accurate than the pruned error tree for the estimate of the BER of some PDUs. Note in fact that approximating the error probability as

0.5 after n_e errors flattens the analytical BER curve of the pruned tree approach.

2.7.2 Network simulation environment

The behavior of a network in a general arrangement can not be directly derived from the BER results given in the previous subsection. Moreover, network performance directly depends on MAC choices. Hence, BER results can not provide the full picture about the accuracy of the analytical methods, since relevant differences at the physical layer may be smoothed out or amplified by network behaviors. Before designing or optimizing protocols based on pseudo-analytical results, it is therefore very important to assess the validity of the approximate techniques in a more general networking scenario.

To this end, we consider again the same network setting described in Section 2.5.1. The only difference is that we will now carry out a comparison involving both dest–lock and node–lock. A key point of our analysis is to verify whether and to which extent using a given approximation method corresponds to accurate networking results, since MAC protocols are likely to smooth out the discrepancies arising from lower level models. Our purpose is to understand which simulation is more accurate in predicting network behavior. After that, we will move to a more in-depth evaluation of the same MAC protocol that will exploit the greater versatility and speed of the pseudo-analytical approach.

2.7.3 Network results

In this subsection, bit-by-bit simulations of the whole system, including complete physical and MAC layer modeling, are provided and compared with those obtained using the Gaussian and the pruned tree approximations. The complete tree exploration is not considered for network results, since it has an exceedingly high complexity. Note that the pruned tree approach provides a better match to simulated average BER than the Gaussian approximation technique, while the Gaussian approximation better follows BER behavior for individual PDUs. On the other hand, the Gaussian approximation is much less complex than the pruned tree approach.

The BER results for the single link are closely related to the PER and its complementary function, the network average success ratio ($1-\text{PER}$). In Figure 2.22 we compare the average success ratio obtained by simulation and analytical techniques, considering the average ratio between the number of correctly detected 1000-bit PDUs and the total number of transmitted PDUs per frame. We observe that the conclusions derived from the average BER results for a single link are modified by the MAC behavior and the accuracy of the analysis for each PDU becomes more relevant. In particular, the pruned tree technique, which provides a good BER match, yields PER values that differ significantly from simulation results due to the limited number of branches considered. On the contrary, the Gaussian technique performs well under both dest–lock and node–lock policies and under all considered traffic levels, since

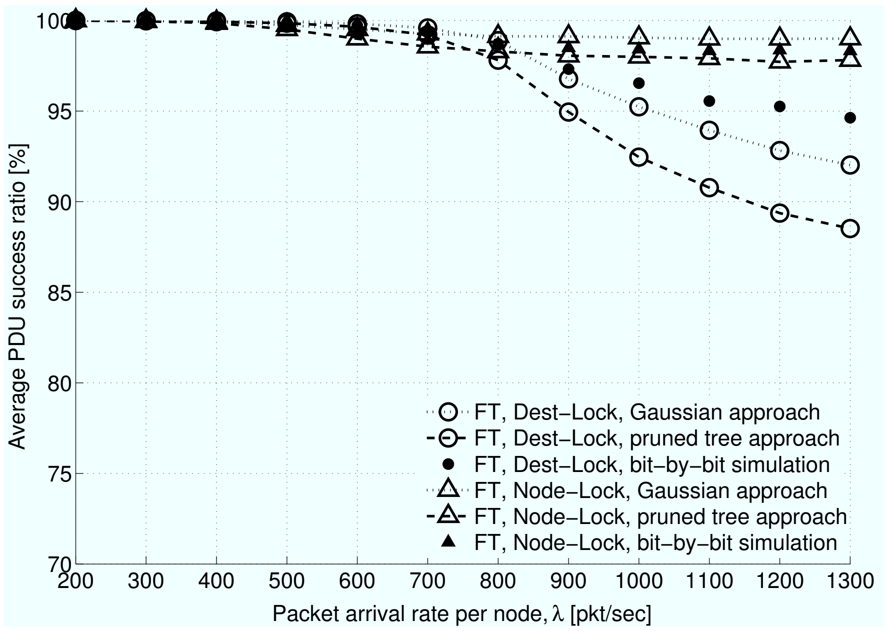


Figure 2.22. *Average 1000-bit PDU transmit success ratio as a function of λ, dest–lock and node–lock.*

as the number of PDUs increases, the distribution of interference becomes Gaussian by the central limit theorem.

In order to explore the accuracy of the analytical techniques we considered other network metrics, *i.e.*, average network throughput, average queue length, and delay.

Figure 2.23 shows the average network throughput, defined as the number of correctly detected 1000-bit PDUs per frame, as a function of the offered traffic λ. As in Figure 2.22, the analytical techniques are accurate for low traffic, and the pruned error tree approach exhibits a mismatch with respect to simulated results due to flattening of BER per user.

Figures 2.24 and 2.25 depict the average queue length and the delay, defined as the average time elapsed from the packet generation to the ACK reception following a successful packet transmission. We note that also in this case network parameters are well approximated by the MAC protocol evaluated by the Gaussian technique. About the latency, the smaller throughput predicted by the pruned tree translates into a greater delay in the dest–lock case. On the contrary, with node–lock, both approaches provide very close approximations. Figure 2.26 shows the average number of transmit/receive data links that a single node activates per frame, possibly containing more than one 1000-bit PDU transmission per link. Both the pruned tree and the Gaussian techniques are indeed very accurate in the dest–lock and the node–lock cases.

We conclude that the Gaussian approximation is well suited to predict network behavior for a wide range of traffic intensities and different MAC policies, and also has the advantage of a limited complexity when compared with the pruned error tree approach. It is then suitable to obtain fast results, for example to compare the two proposed MAC polices. The dest–lock policy favors transmissions, as it blocks communications only toward a sin-

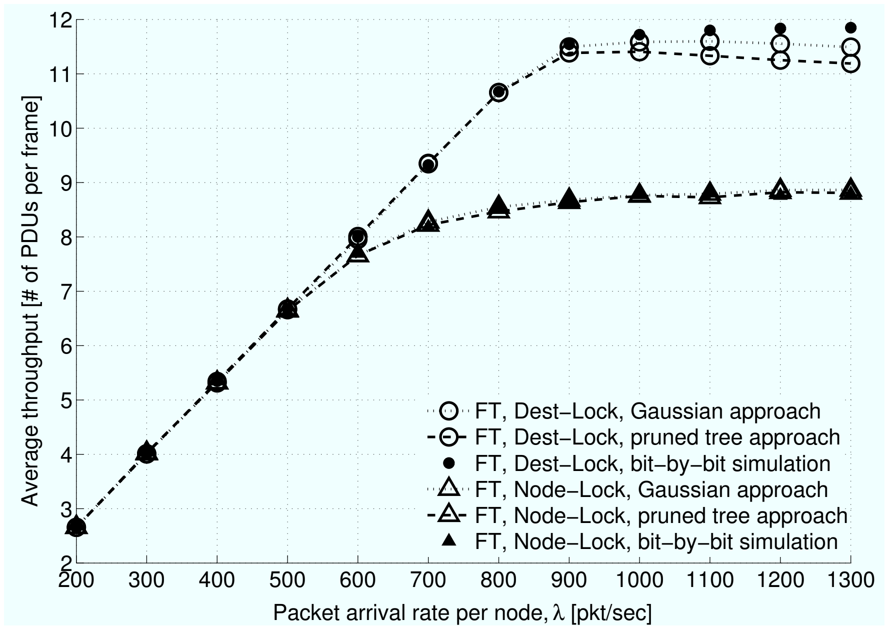


Figure 2.23. *Average network throughput as a function of λ, dest–lock and node–lock.*

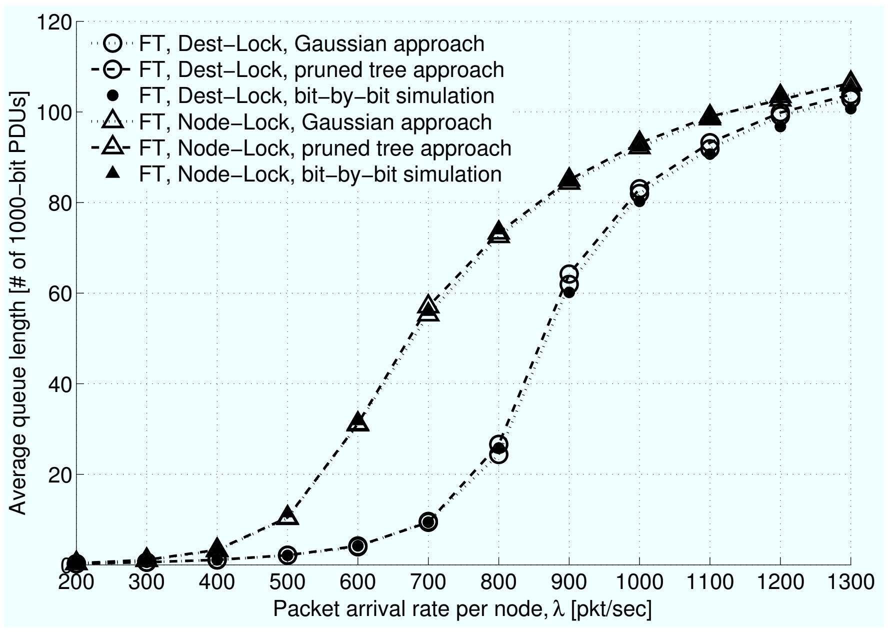


Figure 2.24. *Average queue length as a function of λ, dest–lock and node–lock.*

gle unavailable receiver each time a failure occurs. As more nodes transmit simultaneously, receivers must perform more cancellation stages, which results in a greater probability of detection errors. This explains the slight throughput decrease of the dest–lock policy. On the other hand, from Fig. 2.26, dest–lock generates more RTSs than node–lock, and for very high packet arrival rates dest–lock may cause more collisions than node–lock. In fact, while node–lock is conservative (the node refrains from all transmissions), dest–lock better exploits the available links, letting more nodes transmit, each with fewer active links.

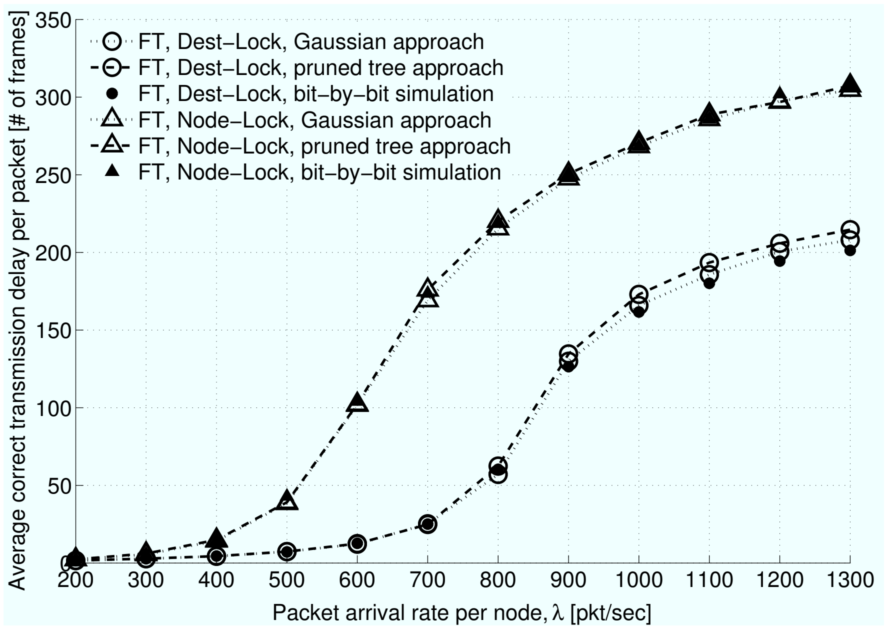


Figure 2.25. *Average correct transmission delay as a function of λ, dest–lock and node–lock.*

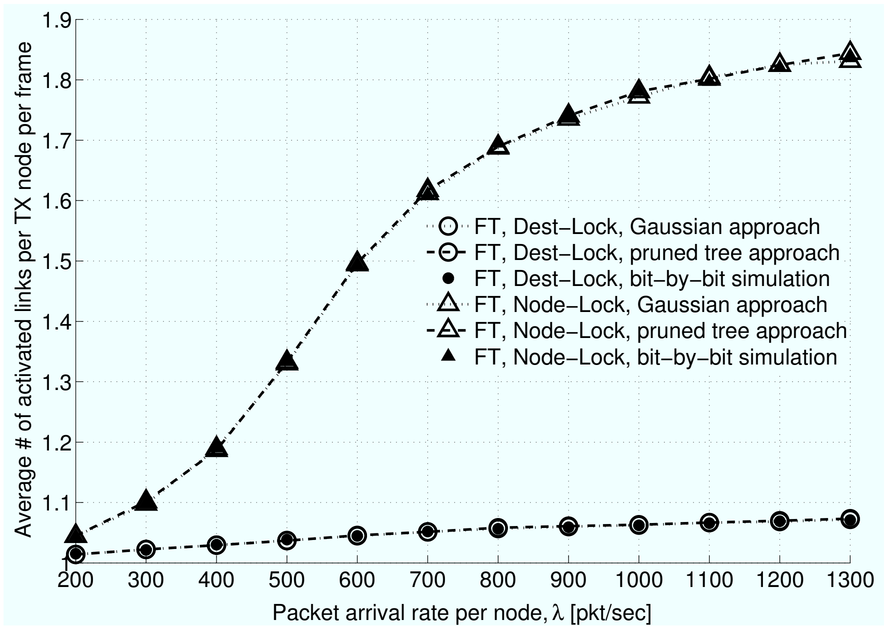


Figure 2.26. *Average number of activated links per transmitting node per frame as a function of λ, dest–lock and node–lock.*

2.7.4 Conclusions on approximation techniques

The techniques discussed so far were studied in order to achieve fast performance evaluation of ad hoc networks using BLAST. The match of the approximations with the results obtained by detailed simulations has been assessed both for a single link and for network performance. From the numerical results, and considering the dramatic decrease of complexity achieved using the approximation instead of bit-by-bit simulations, we can conclude that these approximations provide a useful tool for the design and evaluation of ad hoc networks using multiple antennas. Indeed, the Gaussian approximation of the interference shows the best match with network simulation results and is to be preferred to error enumeration techniques, even though those offer better results for a single link.

Having this important tool available, we will now use it to improve the behavior of the MAC protocol and to detail the effect of backoff on the network. The first task is dealt with in Section 2.8, and allows to show the importance of having a fast simulation technique available, when designing new protocols. The second task is carried out in Section 2.9, and will prove the usefulness of the approximation when analyzing the details of a protocol in more depth.

2.8 DSMA: An Improved Scheme for Channel Access Control

Until now, we have given only some insight on the importance of limiting a node's channel access attempts. We have also shown that severe interference (*e.g.*, uncanceled such as in the NFT CTS policy introduced in Section 2.4.6) can limit performance. Unregulated access can be a non-negligible source of such interference. The MAC described in this Chapter employs exponential backoff as the only means of limiting channel access persistence. Here we introduce a different protocol that can be integrated almost seamlessly into the MAC. This protocol is called Distributed Scheduling for MIMO Ad hoc networks (DSMA) and is basically a way to have nodes rotate between the role of sender and receiver. Among other advantages, this is expected to yield a better traffic balance and to enhance throughput, as nodes are guaranteed to a controllable extent that their intended receiver will not be engaged in another communication.

2.8.1 DSMA Channel Access Scheme

DSMA is based on the following consideration: instead of leaving nodes to access the radio medium in a completely uncoordinated manner, we provide a means for receivers to control the state of other nodes at the beginning of the next frame. This would ensure that a destination node is actually listening and ready to receive.

In more detail, let us focus on a given frame. Nodes that were receivers in this frame are supposed to send an ACK message to confirm correct data reception. In DSMA, with probability $P_{reserve}$, receivers scan their backlog queue before composing the ACK, pick the destination of one or more of the head packets (up to a maximum depth d_{max}), and embed in the ACK a *reservation* directed toward those nodes. We recall that ACKs are short signaling messages, and hence are received correctly with high probability, as shown in Section 2.3.4. Nodes receiving the ACKs and recognizing themselves as reserved destinations, refrain from transmission in the following frame.[6]

Note that with DSMA, ACKs reserve only those nodes who were transmitters or idle, because in the final part of the frame, every receiver is transmitting its own ACK and thus cannot receive other messages. Furthermore, the nodes performing a reservation in the current frame are surely receivers, and will be given a higher chance to transmit successfully in the following frame. This contributes to an overall fairer rotation of transmit and receive roles, and to a better adaptation to local traffic needs.

Idle nodes (*i.e.*, nodes that are neither reserved nor reserving) can decide to act as transmitters in the following frame with probability P_{Tx}. When choosing who to transmit to, they privilege reserved nodes and, clearly, do not send anything to reservers. Thus, they properly exploit the information contained in the previously detected ACKs, which they surely heard.

Unlike previous work in which we made use of a random access policy, we believe that a partially controlled but indeed distributed management of transmissions and receptions could provide benefits to the performance of MIMO ad hoc networks using spatial multiplexing, without introducing any new communication overhead, as reserving information are piggy–backed in control messages that would be sent anyways.

2.8.2 Results

Relevant network parameters

The simulation scenario is the same introduced for the previous experiments, (see also Section 2.5). This allows a more direct comparison. Once more, it is worth stressing that all results presented here have been obtained using the pseudo-analytical approach with the Gaussian approximation of Section 2.6.1 as a model for the physical layer. The method proved to be fast and allowed to span over a number of different configurations. Consequently, it was possible to explore the inter-dependence between DSMA's parameters and their effect on the network performance. All results presented hereon are obtained assuming the use of the FT CTS policy.

Comparison between DSMA and exponential backoff

As a first result, Figure 2.27 shows a comparison between the average throughput for the DSMA and exponential backoff strategies, for varying offered traffic. By throughput, we mean again the average number of 1000-bit PDUs per frame that actually get through. Exponential backoff is intended as dest–lock, with window parameter $W = 16$.[7] In Figure 2.27,

[6]Recall that frames are made of synchronized RTS, CTS, DATA and ACK transmissions, so that all ACKs are simultaneously received, and all nodes take on the requested receiver roles accordingly (see Sections 2.3.4 and 2.8.2).

[7]This is a good value to reach good performance when using dest–lock, see also Section 2.9.

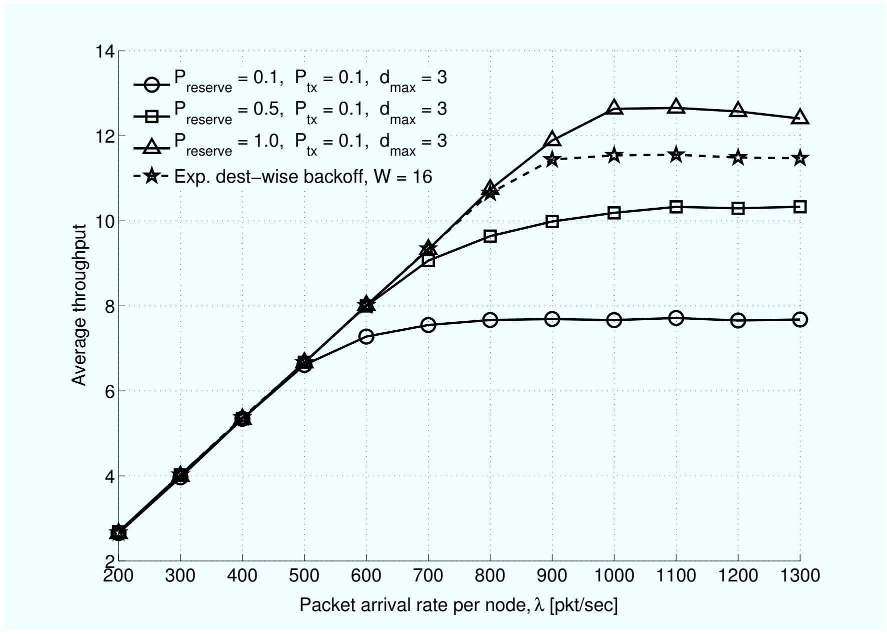


Figure 2.27. *Comparison between average throughput of DSMA and dest–lock for* $P_{Tx} = 0.1$*,* $d_{max} = 3$ *and varying* $P_{reserve}$*.*

we set DSMA parameters by considering a fixed probability of transmission by idle nodes ($P_{Tx} = 0.1$), and fixed queue search depth for reservations ($d_{max} = 3$). $P_{reserve}$ is varied to be 0, 0.5 or 1.

Figure 2.27 shows that exponential backoff is outperformed by DSMA if $P_{reserve}$ is sufficiently high. Otherwise, the maximum saturation throughput that can be reached by the DSMA algorithm is inferior, and decreases for decreasing $P_{reserve}$. A first reason is as follows. For low P_{Tx}, if $P_{reserve}$ is also low, a very conservative behavior is set up in the network, because these values limit both the number of reserving nodes and the number of idle nodes deciding to transmit. Conversely, a high $P_{reserve}$ maximizes the effects of the DSMA distributed coordination method, achieving a higher throughput than exponential backoff. Hence, the intuition that originates this contribution is confirmed. Recall also that the maximum throughput (as we defined it) reachable by 802.11 DCF in a completely connected network is 1, hence both MIMO MACs offer a considerable improvement. Further gain may be achieved by properly tuning P_{Tx} and $P_{reserve}$, as will be explained in the following Section, which discusses a high offered traffic scenario.

Besides the higher level of coordination among transmitters and receivers, another reason behind the higher throughput values reached by DSMA is visible in Figure 2.28, that shows the ratio of successfully received to sent PDU transmission requests, for both DSMA and exponential backoff. The parameters are set as in Figure 2.27. The PDU success ratio is a very important metric, because of the way network operations are administered. Recall that the FT policy relies on the knowledge of communications that will superimpose in the next data transfer phase, which is obtained directly from information contained in

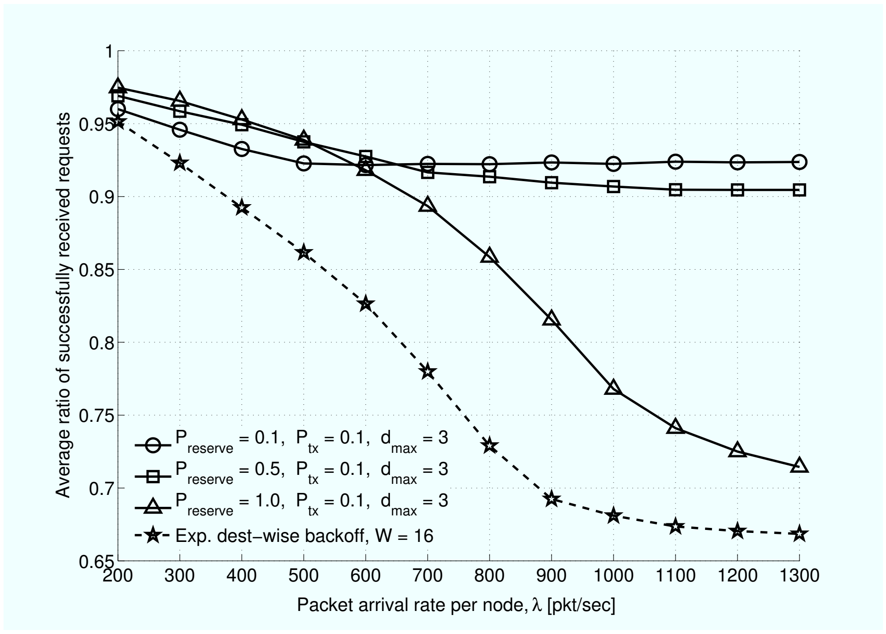


Figure 2.28. *Comparison between success ratio of DSMA and dest–lock for* $P_{Tx} = 0.1$*,* $d_{max} = 3$ *and varying* $P_{reserve}$*.*

RTS messages. If an RTS is sent, but the corresponding destination does not send a CTS, other neighboring receivers might take the request as granted anyway, and spend training sequences to estimate the corresponding channel, even if the transmission does not actually take place.[8] At high traffic loads, this may be a limiting behavior, because important degrees of freedom are wasted for potential transmitters that remain inactive. In turn, this leaves receivers less protected against other interfering transmissions. DSMA helps solve this problem. Reservations help send RTSs to destinations that will be able to grant them with higher probability. This bestows the chance that FT correctly drives the interference cancellation process, thereby increasing the global network throughput. Note that the intersection between the DSMA curves is expected because, at high loads, lower P_{Tx} and $P_{reserve}$ allow few nodes to transmit their RTS, hence improving the probability that the few ones getting through are granted.

To sum up, FT can benefit from a higher–level coordination as provided by DSMA in order to correctly exploit channel information, but DSMA always needs FT to reach a good balance between throughput and reliability (obtained through interference cancellation and load bounding).

[8]Clearly, receivers cannot cope with this problem by extrapolating information from CTSs, as they are CTS senders themselves.

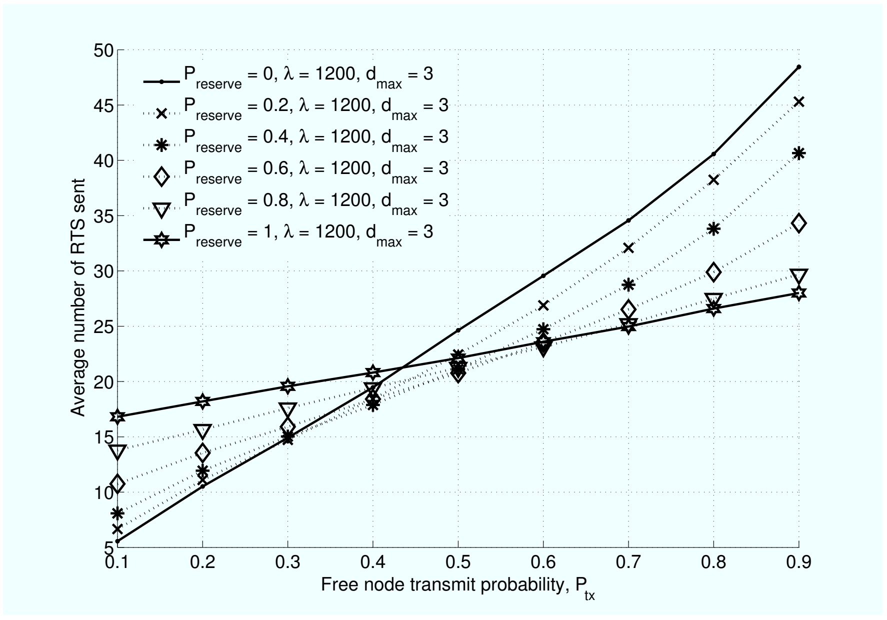


Figure 2.29. *Average number of sent RTSs with DSMA at constant high traffic ($\lambda = 1200$), for varying $P_{reserve}$ and P_{Tx}.*

Results on the Variation of $P_{reserve}$ and P_{Tx}

In this subsection, we describe in deeper detail the effects of tuning the two parameters $P_{reserve}$ and P_{Tx} that are part of the DSMA protocol. In Figures 2.29, 2.30 and 2.31, we give more insights on how the variation of $P_{reserve}$ and P_{Tx} affects the behavior of nodes and the network performance. The Figures depict the number of RTSs sent by nodes during handshakes preceding data transmissions (more RTSs may be included in a single packet to be sent in the RTS phase), the average throughput (*i.e.*, the average number of 1000-bit PDUs per frame that actually get through), and the average ratio of successful PDU transmissions to the total number of attempts to send data PDUs, respectively. Results are considered for a large offered traffic ($\lambda = 1200$), so that the system exhibits saturation conditions and the effect of the values of the parameters is better highlighted (*e.g.*, at low traffic P_{Tx} has an almost imperceptible influence).

From Figure 2.29, we first observe that if $P_{reserve} = 0$, *i.e.*, nodes are not allowed to reserve any destination, the behavior is quite predictable. In fact, the number of RTSs increases with the probability that an idle node decides to send a packet, P_{Tx}. As $P_{reserve}$ is increased, the impact of P_{Tx} is reduced, and even if the relationship between the number of RTSs and P_{Tx} is still approximately linear, the slope is smaller. This happens because an increasingly higher number of nodes are asked to act as receivers, thereby leaving a smaller set of transmitters per frame. With high $P_{reserve}$, there is in fact a high probability that transmitters in a given frame are nodes that made a reservation request in the previous frame.

Now let us concentrate on Figures 2.30 and 2.31, representing average throughput and success ratio in a high traffic scenario, respectively. In the case $P_{reserve} = 0$, at first (low

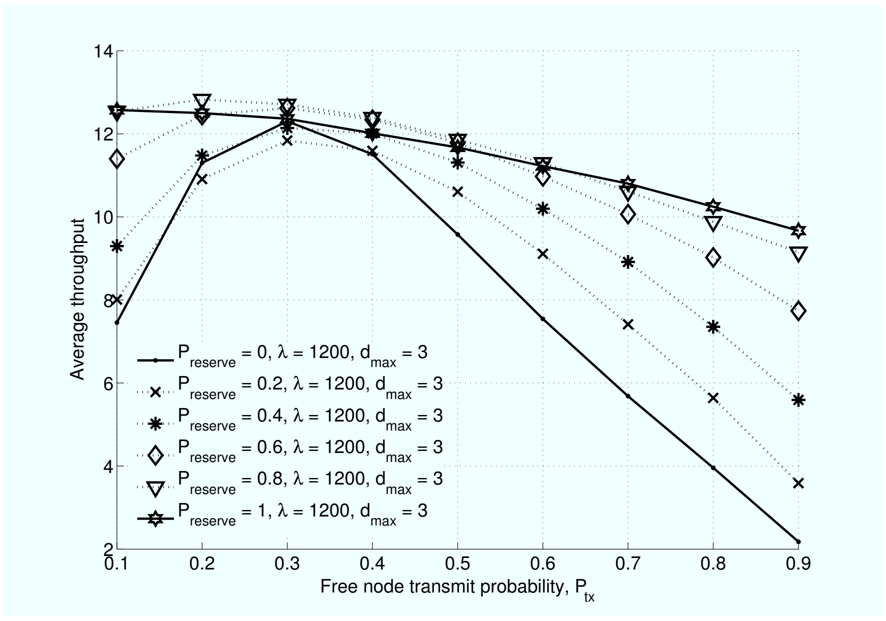


Figure 2.30. *Average throughput with DSMA at constant high traffic ($\lambda = 1200$), for varying $P_{reserve}$ and P_{Tx}.*

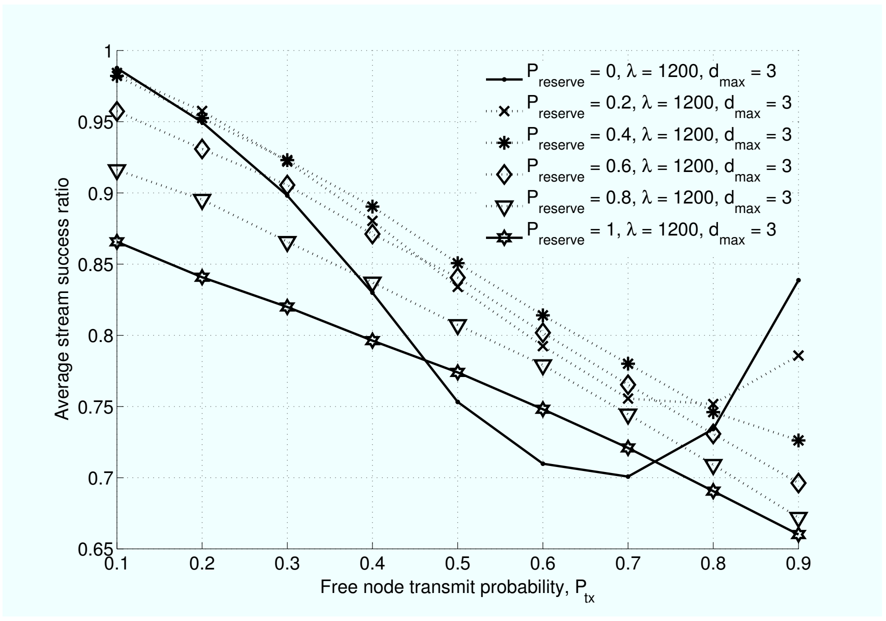


Figure 2.31. *Average ratio of successfully received to transmitted PDUs with DSMA at constant high traffic ($\lambda = 1200$), for varying $P_{reserve}$ and P_{Tx}.*

P_{Tx}) the number of transmitters per frame is low, and so is the throughput as well. As P_{Tx} is increased, the number of transmissions per frame also increases: the system is quickly loaded by senders, thereby experiencing a steep throughput increase. Anyway, nodes inject traffic in the network in a "dumb" way, as they greedily try to send with probability P_{Tx} without any coordination. Therefore, throughput first reaches a maximum, but then falls, because receivers are too overloaded to decode incoming transmissions correctly. Success ratio decreases as well, until P_{Tx} is so high that almost all nodes try to send data in each frame. In this scenario, they require most probably to transmit toward nodes that chose to be transmitters themselves, hence causing a failure that forces both to stay silent during the data phase. Therefore, the few handshakes taking place between a sender and an idle receiver succeed almost surely, increasing the success ratio for high P_{Tx} and $P_{reserve} = 0$. Note that, even with a higher success ratio, nodes rarely find a free receiver, hence the throughput continues decreasing with P_{Tx}.

If the DSMA reservation feature is activated by increasing $P_{reserve}$, we observe a progressively higher throughput and success ratio, explained by the deeper level of coordination that is distributely achieved among nodes. In particular, transmitting nodes are more frequently asked to act as receivers. Furthermore, idle nodes that choose to transmit preferably address reserved nodes. This both improves the probability that sent RTSs are granted and reduces the impact of a higher P_{Tx}, thereupon achieving a better coordination of traffic flows and a globally higher network performance. Specifically, lower $P_{reserve}$ has a behavior which is very similar to the case $P_{reserve} = 0$, but with higher success ratio, and thus higher throughput for greater P_{Tx}; on the other hand, while proceeding toward the limiting case $P_{reserve} = 1$, curves show a more linear behavior (due to the smaller influence of P_{Tx}), with higher throughput and smoothly decreasing success ratio. In these cases, the level of coordination among nodes is pushed to its maximum depth, obtaining improved performance despite the slightly lower success ratios. Such results are due to the very high chance that an RTS is directed toward a node which was asked to be a receiver, improving the likelihood that they are accepted and that more data PDUs are spatially multiplexed in the network. Of course, the resulting heavier receiver load tends to imply a lower success ratio, but the overall effect is typically a higher throughput nonetheless.

Note that we do not claim at all that increasing $P_{reserve}$ is always the best strategy for achieving a better throughput: performance depends on traffic and node density, and the best values of $P_{reserve}$ and P_{Tx} are to be tuned adaptively. For the sake of a better understanding of DSMA's behavior, we show in Figure 2.32 how the maximum allowable throughput is jointly affected by $P_{reserve}$ and P_{Tx}. A very interesting thing to note there is the *saddle point* forming at about ($P_{Tx} = 0.3$, $P_{reserve} = 0.1$), *i.e.*, the point of local maximum in P_{Tx} and of local minimum in $P_{reserve}$. It can be inferred that, at high traffic and for P_{Tx} around 0.3, reserving nodes with low probability is worse than not reserving at all, which is in turn worse than reserving with higher probability. In fact, at high traffic, keeping the reservation mechanism too limited has the only net effect of increasing the number of transmissions, whereas the most beneficial effect of DSMA, *i.e.*, the chance to send RTSs to nodes that are ready to listen is still negligible. This sheds some light on the need to set P_{Tx} and $P_{reserve}$ jointly, in order to achieve the maximum performance given the offered traffic. In the discussed scenario, with $\lambda = 1200$, this maximum point is found to be around ($P_{Tx} = 0.2$, $P_{reserve} = 0.8$). Thus, for processing the highest amount of traffic and achieving the maximum throughput,

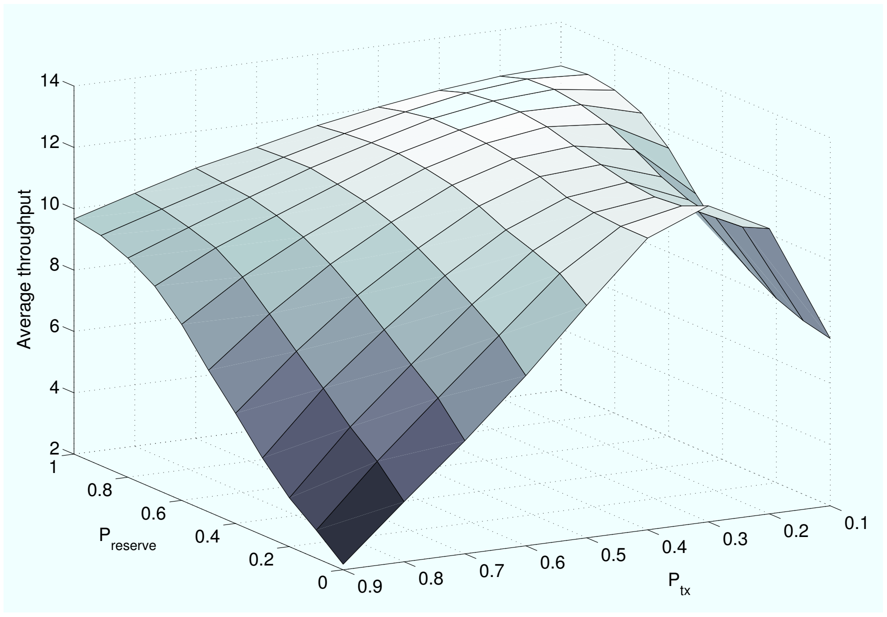


Figure 2.32. *3D view of throughput with DSMA at constant high traffic ($\lambda = 1200$), for varying $P_{reserve}$ and P_{Tx}.*

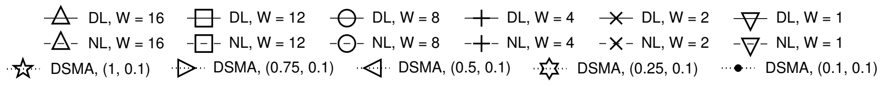
one has to rely on a high probability of reserving receivers in the ACK transmission phase and on a low probability that an idle node decides to initiate a transmission.

2.8.3 Conclusions on DSMA

The results presented in this Section have shown that the DSMA scheme can be very effective in achieving throughput and success ratio benefits in a MIMO ad hoc network. The scheme relies on a distributed scheduling and coordination of transmitter and receiver roles, which improves the chance that transmissions are directed toward nodes that are in fact available to receive and process them. Also note that DSMA integrates almost seamlessly in our cross–layer design for MIMO ad hoc networks. It uses the same messaging and framed communication structure, with the introduction of only three further parameters. The insights on the effect of tuning two of these parameters have shown that DSMA can adapt to low as well as high traffic conditions.

2.9 Performance Comparison for all Access Strategies

As a final step of our study involving the approximation of MIMO PHY performance, in this Section we provide a wider comparison among the different backoff techniques and the DSMA approach introduced before. This study aims at showing which backoff technique is more suited to MAC access and how it drives the transmissions in terms of created links, spatial multiplexing over each link, and so on. The network scenario is the same considered before.



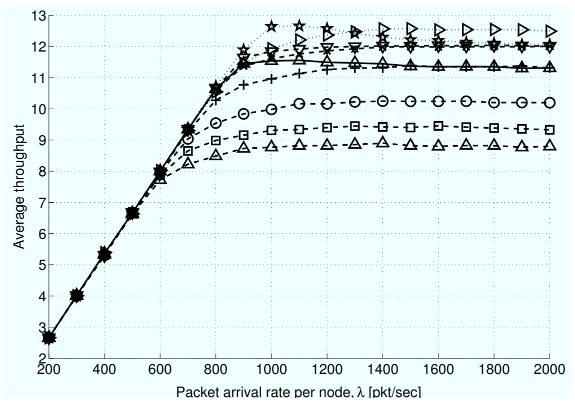


Figure 2.33. *Throughput, node–lock.*

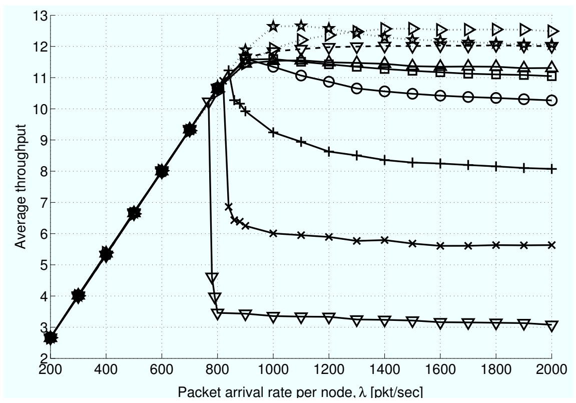


Figure 2.34. *Throughput, dest–lock.*

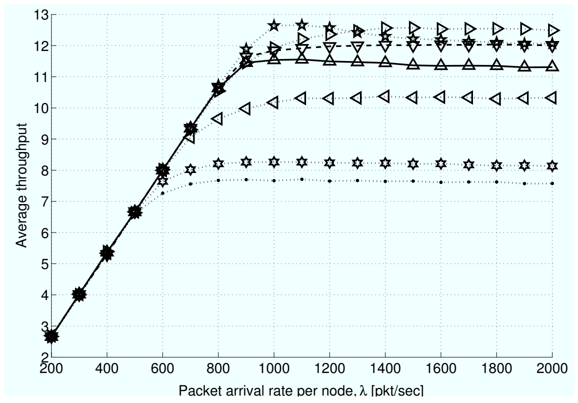


Figure 2.35. *Throughput, DSMA*

The first results we show are throughput (Figures 2.33, 2.34 and 2.35), latency (in number of frames) before a correct transmission (Figures 2.36, 2.37 and 2.38), and success ratio (Figures 2.39, 2.40 and 2.41). Each set of pictures is devoted to a different metric. Each picture contains the curves related to one of the techniques. Additionally, it depicts with the curves of the other techniques that show the best results, for an easier comparison.

When studying backoff, different curves are obtained by changing the value of the backoff window parameter W (see also Section 2.4.5). For DSMA, the pair $(P_{reserve}, P_{Tx})$ is varied. The legend is shown only once per each set of pictures, and reports the parameters associated to each curve. Note that dest–lock is depicted using solid lines, node–lock using dashed lines and DSMA using dotted lines. Furthermore, dest–lock and node–lock are abbreviated as DL and NL, respectively.

First, consider Figures 2.33 to 2.35, depicting average throughput. DL and NL have different behaviors for varying W. In particular, DL is a more aggressive policy. It allows nodes to send more requests by blocking just one unavailable destinations per failed attempt. As a

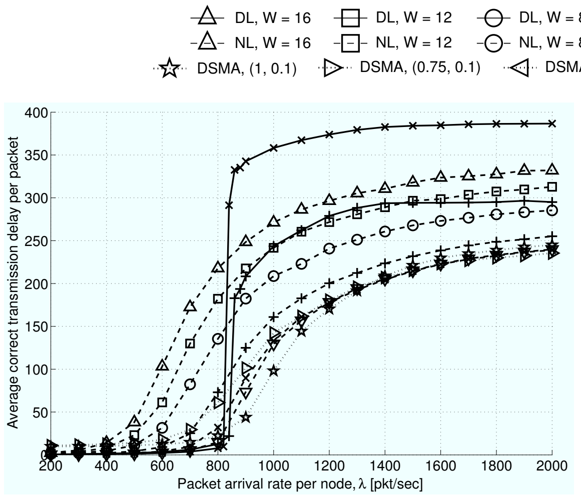


Figure 2.36. *Latency, node–lock.*

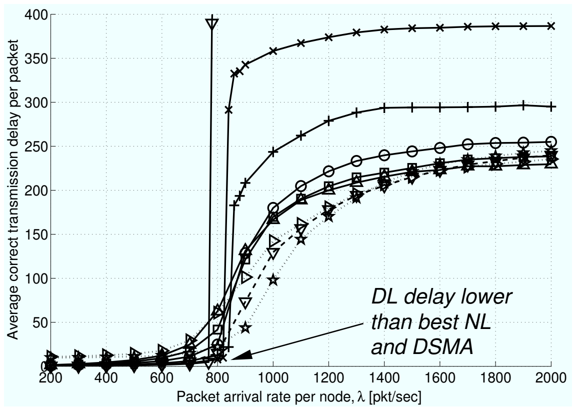


Figure 2.37. *Latency, dest–lock.*

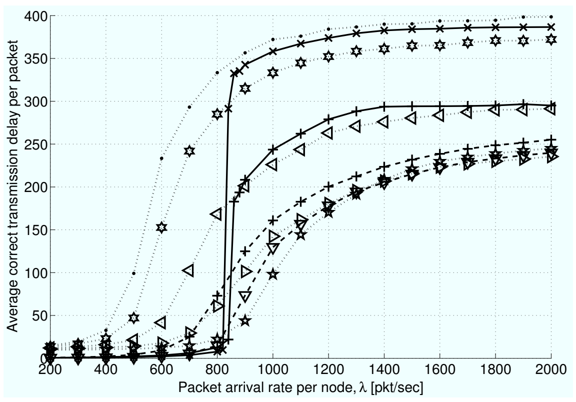
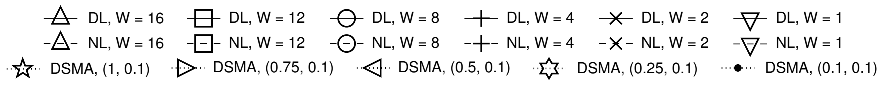


Figure 2.38. *Latency, DSMA*

consequence, DL performs better than NL only if W is sufficiently high, such that congestion does not occur. For example, for $W = 1, 2, 4, 8$, DL is subject to a decay in throughput performance which is progressively mitigated by increasing W. This decay is mainly caused by the unsustainable amount of traffic generated due to node persistence in transmission attempts. This persistence eventually overloads the receiving stage, and prevents a correct detection. Conversely, $W = 12, 16$ force longer silences on average, hence receivers are less loaded on average.

NL, on the other hand, imposes to defer any communication, having any transmitter turn into an available receiver for a given amount of time upon any failure. Anyway, if W is too large the throughput saturates to a suboptimal value. With sufficiently low W, instead, NL outperforms the best throughput reached by DL.

Even if outperformed by NL from a throughput point of view, DL is very useful for keeping transmission latency as low as possible. Figures 2.36 to 2.38 detail this fact, which is a direct consequence of DL's aggressiveness. With DL, nodes can transmit more often, so



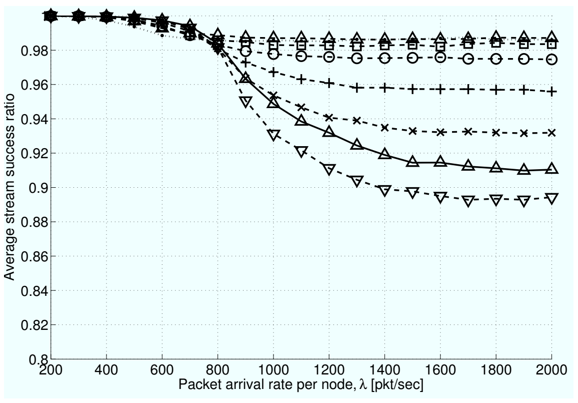


Figure 2.39. *Success ratio, node–lock.*

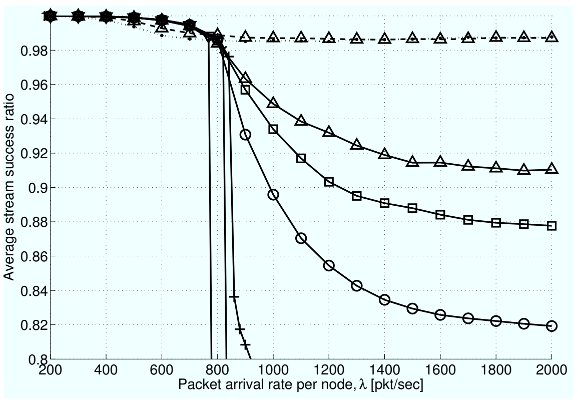


Figure 2.40. *Success ratio, dest–lock.*

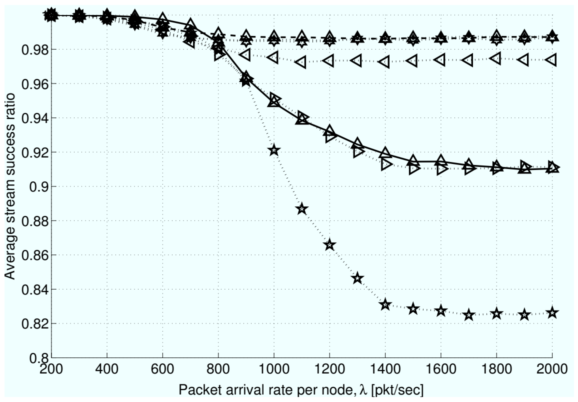


Figure 2.41. *Success ratio, DSMA*

that in low traffic scenarios they still experience a fair PDU success probability with lower delay. Figures 2.39 to 2.41, depicting the ratio of the correctly received 1000-bit PDUs to those sent, support this deduction. Such considerations suggest that DL should be used when low traffic is expected, while switching to NL at higher traffic and using, *e.g.*, the average experienced delay as a measure of local network congestion for deciding when to switch from DL to NL. Studying and designing adaptive protocols is out of the scope of this paper, and is currently being addressed.

The behavior of distributed scheduling (DSMA) is different. By enforcing some coordination among receivers and transmitters, DSMA reaches a higher throughput and outperforms both backoff schemes. The two parameters $P_{reserve}$ and P_{Tx} as defined in Section 2.8.1 are crucial in determining the protocol behavior. For this reason, they have been chosen so that an appreciable difference among curves is observed in the given results. As seen from Figures 2.33–2.35 and 2.36–2.38, DSMA outperforms both NL and DL even in strong traffic scenarios, provided that $P_{reserve}$ is sufficiently high, *i.e.*, that an adequate number of reser-

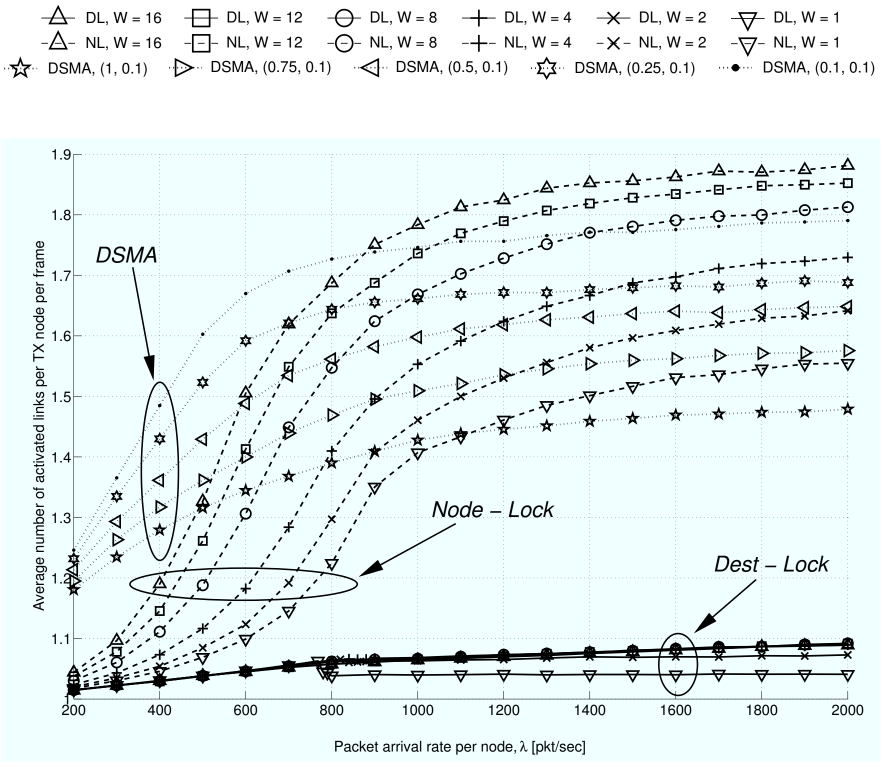


Figure 2.42. *Average number of links per TX node per frame as a function of traffic.*

vation messages are sent by receivers. In particular, for $P_{reserve} = 0.75$ and $P_{reserve} = 1$, the network is pushed to a throughput of as much as 12.8 PDUs per frame. We also observe that for $P_{reserve} = 0.75$, the throughput increases more gradually but then achieves the highest value among all strategies. This interesting result shows that relying only on distributed scheduling of transmissions and receptions is not the most advantageous choice; the best performance is instead achieved by preserving a certain degree of randomness.

To obtain further insights on the behavior and applicability of the schemes presented, we depict in Figures 2.42 and 2.43 the average number of links activated per node per frame, and the average number of transmitters per frame, respectively. As before, by "link," we mean a node–to–node connection, regardless of the amount of spatial multiplexing used. In Figure 2.43, only three curves per policy are reported. With these figures, it is possible to understand whether high throughput strategies prefer to load single connections with many PDUs, or to create multiple links each with less spatial multiplexing. In the former case, for example, the policy would prove to be more suited to delay–constrained connection–based networking, where it is important to convey high traffic on a given link (*e.g.*, as part of a longer multihop path toward a final recipient). In the latter, the policy would be more applicable to information distribution scenarios, where a single source may want to address several destinations in order to spread traffic faster.

DL tends to allow more transmitters than NL (Figure 2.43) and correspondingly more one–to–one connections (Figure 2.42), with each connection having stronger spatial mul-

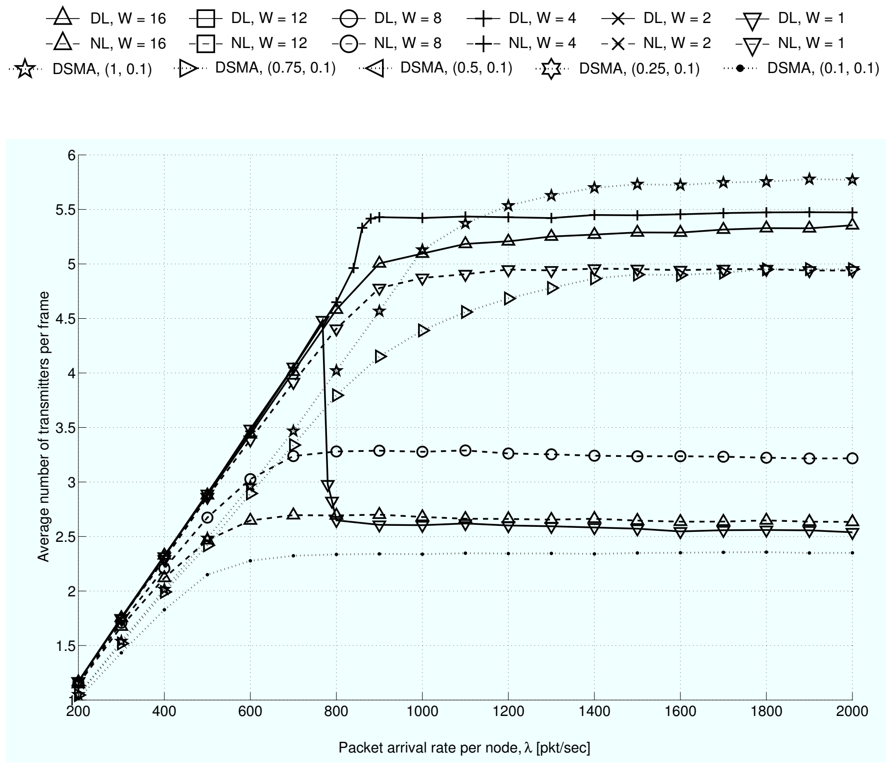


Figure 2.43. *Average number of transmitters per frame as a function of traffic.*

tiplexing, and thus higher throughput. Conversely, even the most permissive NL policy (for $W = 1$) enables a lower number of transmitters, each likely to connect to more than one receiver. NL thus achieves a lower data rate per link, but an overall better aggregate throughput.

DSMA instead tends to create multiple links originated from the same transmitter. As a result, DSMA curves in Figure 2.42 all start from greater values than both backoff techniques, and then saturate to a maximum that is mainly determined by $P_{reserve}$ (the probability that a receiver decides to make a reservation). Correspondingly, the higher the number of simultaneous links created, the lower the number of transmitters allowed, as seen from Figure 2.43. This behavior results from DSMA design, whereby reservations are used to cycle the transmit role among nodes. If a receiver wants to transmit in the following frame, it may ask for up to a certain number of listeners each time. This engages one–to–many links with high probability, and is exploitable whenever a high degree of parallelism is needed along with higher bit rates. Such a scenario is found, *e.g.*, in multihop wireless networks with many coexisting multimedia connections, where parallelism ensures a fast data spreading, and transmit role cycling helps mitigating starvation (the time before a receiving node can become a transmitter again).

Note also that the parameters $(0.75, 0.1)$ drive DSMA to nearly the same transmit behavior as NL with $W = 1$. The slight difference experienced in throughput performance (higher for DSMA) is explained by the slightly higher PDU success ratio, which in turn depends on the ability of DSMA to coordinate transmitters and receivers so that intended destinations are reached with high probability.

As a final remark, consider again Figures 2.33–2.35 and Figures 2.39–2.41. We note that DL and NL experience high throughput in correspondence of a success ratio near 99%, whereas the max throughput DSMA configuration $(0.75, 0.1)$ undergoes 90% success only. This non-trivial result is explained by the different way DSMA organizes transmission parallelism. Namely, as many transmitters exist as in the best NL policy ($W = 1$), but each has slightly more active links per frame and with stronger SM than NL; moreover, due to the reservation mechanism, intended destinations are more easily reached than in random backoff policies. This reorganizes traffic in a sufficiently distributed fashion, such that success probability is still acceptable and performance is globally better. To sum up, DSMA allows for more SM traffic and enforces parallel one–to–many communications still achieving satisfactory success rates and the lowest average delay among all strategies. These are very important features for an ad hoc network where all nodes have heavy traffic requirements.

2.9.1 Conclusions on access strategies comparison

In this Section we have aimed first at showing the importance of correct transmission management in a network that relies on multiuser detection, since interference control is a primary issue to address. Moreover, we have characterized all strategies under the point of view of throughput performance, success ratio, transmission parallelism, number of links per transmitter and delay incurred by correct transmissions. We also highlighted relevant differences and analogies in how the network is driven by each strategy, and showed how spatial multiplexing and multiuser detection are used for enhancing throughput performance and how transmitting nodes establish parallel links to different destinations. We have also studied how some relevant parameters affect the behavior of each policy, obtaining insights on how they could be tailored to match different traffic needs.

This study has been made possible by the flexibility and higher speed offered by the pseudo-analytical approach developed in Section 2.6. Without it the study would have been still possible, yet much more lengthy and difficult to pursue.

2.10 Summary and Conclusions

In this Chapter, we have presented a cross–layer protocol design for MIMO ad hoc networks. These are a special case of ad hoc networks where each terminal mounts multiple antennas, and signals are processed in order to transmit superimposed streams of data and yet be able to demultiplex them at the receiver. This can yield several benefits to the overall network performance, provided that the receiver is not overloaded. The purpose of our MAC protocol is exactly to keep the performance of such a network reasonably good by driving channel access, and most importantly, by exchanging information with the PHY layer to understand which transmissions should be granted or not. This decision is made based on average PHY-level metrics, which therefore impact access control. MAC then tells PHY what to decode, and what to leave as interference.

This bidirectional cross–layer interaction is very important to balance traffic needs at the MAC layer while not overloading the receiver at the physical layer. Actually, it constitutes the founding brick of our work, showing that the cross–layer design paradigm is very effective in this field. When designing protocols for higher network levels such as routing, the interaction should be expanded to routing itself, so that all chances to optimize the network can be actually exploited.

After describing our algorithm and showing the performance through several different metrics, we argued that the evaluation of a given design or the creation of new protocols is slowed down considerably by the need to simulate PHY details. Therefore, we developed two different approximations for PHY performance and showed that they provide satisfactory results when used to evaluate a whole network on the time scale of a protocol. Finally, we used these approximations to develop a new protocol and to delve into a comparison between different channel access strategies. The pseudo-analytical approach allowed to obtain a large number of result in a short time, thereby easing both protocol design and tuning.

Acknowledgment

The work presented in this Chapter has been done in collaboration with Marco Levorato, University of Padova.

References

[1] IEEE Standards Department, *ANSI / IEEE Standard 802.11*. IEEE Press, 1999.

[2] F. Tobagi and L. Kleinrock, "Packet switching in radio channels: part II–The hidden terminal problem in carrier sense multiple-access and the busy-tone solution," *IEEE Trans. Commun.*, vol. 23, no. 12, pp. 1417–1433, Dec. 1975.

[3] C. E. Perkins and E. M. Belding-Royer and S. R. Das, "Ad-hoc on-demand distance vector routing," IETF Mobile Ad-hoc Networks (MANET) Working Group RFC. [Online]. Available: http://www.ietf.org/rfc/rfc3561.txt

[4] J. Broch, D. B. Johnson, and D. A. Maltz, "The dynamic source routing protocol for mobile ad hoc networks," IETF Internet Draft, Dec. 1998. [Online]. Available: http://www.ietf.org

[5] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," IETF RFC 3626, Oct. 2003. [Online]. Available: http://www.ietf.org

[6] G. Anastasi, M. Conti, and E. Gregori, "IEEE 802.11 ad hoc networks: protocols, performance and open issues," in *Ad Hoc Networking*. New York: IEEE Press and John Wiley and Sons, Inc., 2004.

[7] R. Ramanathan, J. Redi, C. Santivanez, D. Viggins, and S. Polit, "Ad hoc networking with directional antennas: a complete system solution," *IEEE J. Select. Areas Commun.*, vol. 23, no. 3, pp. 496–506, Mar. 2005.

[8] H. L. van Trees, *Optimum Array Processing, Part IV*. New York: John Wiley and Sons, Inc., 2002.

[9] C. A. Balanis, *Antenna Theory: Analysis and Design*, 2nd ed. New York: John Wiley and Sons, Inc., 1996.

[10] J. G. Proakis, *Digital Communications*, 2nd ed. New York: McGraw-Hill, 1999.

[11] G. J. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multiple antennas," *Bell Labs Tech. J.*, vol. 1, no. 2, pp. 41–59, 1996.

[12] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V–BLAST: an architecture for realizing very high data rates over the rich–scattering wireless channel," in *Proc. of IEEE ISSSE*, Pisa, Italy, Sep. 1998, pp. 295–300.

[13] H. Jafarkhani, *Space–Time Coding: Theory and Practice*. Cambridge University Press, Sep. 2005.

[14] S. M. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE Trans. Commun.*, vol. 16, no. 8, pp. 1451–1458, Oct. 1998.

[15] L. Zheng and D. N. C. Tse, "Diversity and multiplexing: a fundamental tradeoff in multiple-antenna channels," *IEEE Trans. Inform. Theory*, vol. 49, no. 5, pp. 1073–1096, May 2003.

[16] A. J. Paulraj, D. A. Gore, R. U. Nabar, and H. Bölcskei, "An overview of MIMO communications: a key to gigabit wireless," *Proc. IEEE*, vol. 92, no. 2, pp. 198–218, Feb. 2004.

[17] R. R. Choudhury, X. Yang, R. Ramanathan, and N. H. Vaidya, "On designing MAC protocols for wireless networks using directional antennas," *IEEE Trans. Mobile Comput.*, vol. 5, no. 5, pp. 477–491, May 2006.

[18] R. R. Choudhury and N. H. Vaidya, "Deafness: a MAC problem in ad hoc networks when using directional antennas," in *Proc. of IEEE ICNP*, Oct. 2004.

[19] A. Nasipuri, S. Ye, J. You, and R. E. Hiromoto, "A MAC protocol for mobile ad hoc networks using directional antennas," in *Proc. of IEEE WCNC*, vol. 2, Chicago, IL, Sep. 2000, pp. 1214–1219.

[20] T. Korakis, G. Jakllari, and L. Tassiulas, "A MAC protocol for full exploitation of directional antennas in ad hoc wireless networks," in *Proc. of ACM MobiHoc*, Annapolis, MD, Jun. 2003.

[21] M. Takai, J. Martin, and R. Bagrodia, "Effects of wireless physical layer modeling in mobile ad hoc networks," in *Proc. of ACM MobiHoc*, Long Beach, CA, Oct. 2001, pp. 87–94.

[22] A. Paulraj, R. Nabar, and D. Gore, *Introduction to Space–Time Wireless Communications*. Cambridge, UK: Cambridge University Press, 2003.

[23] B. Chen and M. Gans, "MIMO communications in ad hoc networks," in *Proc. of IEEE VTC 2005-Spring*, Stockholm, Sweden, May 2005, pp. 2434–2438.

[24] M. Park, S.-H. Choi, and S. M. Nettles, "Cross–layer MAC design for wireless networks using MIMO," in *Proc. of IEEE GlobeCom*, vol. 2, St. Louis, MO, Nov. 2005, pp. 938–942.

[25] D. Vang and U. Tureli, "Cross–layer design for broadband ad hoc networks with MIMO–OFDM," in *Proc. of Signal Processing Advances in Wireless Communications*, Jun. 2005.

[26] M. Hu and J. Zhang, "MIMO ad hoc networks: medium access control, saturation throughput, and optimal hop distance," *Journ. of Commun. and Networks, Special Issue on Mobile Ad Hoc Networks*, pp. 317–330, Dec. 2004.

[27] K. Sundaresan, R. Sivakumar, M. Ingram, and T.-Y. Chang, "Medium access control in ad hoc networks with MIMO links: optimization considerations and algorithms," *IEEE Trans. Mobile Comput.*, vol. 3, no. 4, pp. 350–365, Oct. 2004.

[28] G. Jakllari, S. V. Krishnamurthy, M. Faloutsos, P. V. Krishnamurthy, and O. Ercetin, "A cross–layer framework for exploiting virtual MISO links in mobile ad hoc networks," *IEEE Trans. Mobile Comput.*, vol. 6, no. 6, pp. 579–594, Jun. 2007.

[29] S. Cui and A. J. Goldsmith, "Cross–layer optimization of sensor networks based on cooperative MIMO techniques with rate adaptation," in *Proc. of IEEE SPAWC*, New York City, NY, Jun. 2005, pp. 960–964.

[30] Y. Yuan, Z. He, and M. Chen, "Virtual MIMO-based cross–layer design for wireless sensor networks," *IEEE Trans. Veh. Technol.*, vol. 55, no. 3, pp. 856–864, May 2006.

[31] F. Rossetto and M. Zorzi, "On gain asymmetry and broadcast efficiency in MIMO ad hoc networks," in *Proc. of IEEE ICC*, Istanbul, Turkey, Jun. 2006.

[32] S. Sfar, R. D. Murch, and K. B. Letaief, "Layered space–time multiuser detection over wireless uplink systems," *IEEE Trans. Wireless Commun.*, vol. 2, no. 4, pp. 653–668, Jul. 2003.

[33] G. H. Golub and C. F. van Loan, *Matrix Computations*. Baltimore, MD: The Johns Hopkins Univ. Press, 1983.

[34] G. Ginis and J. M. Cioffi, "On the relation between BLAST and the GDFE," *IEEE Commun. Lett.*, vol. 5, no. 9, pp. 364–366, Sep. 2001.

[35] L. Tong, Q. Zhao, and G. Mergen, "Multipacket reception in random access wireless networks: from signal processing to optimal medium access control," *IEEE Commun. Mag.*, vol. 39, no. 11, pp. 108–112, Nov. 2001.

[36] C. Shen, Y. Zhu, S. Zhou, and J. Jiang, "On the performance of V-BLAST with zero-forcing successive interference cancellation receiver," in *Proc. of IEEE GlobeCom*, Dallas, TX, Nov. 2004, pp. 2818–2822.

[37] K. Liu and A. M. Sayeed, "An iterative extension of BLAST decoding algorithm for layered space-time signals," *IEEE Trans. Commun.*, vol. 53, no. 10, pp. 1754–1761, Oct. 2005.

[38] S. Loyka and F. Gagnon, "Performance analysis of the V-BLAST algorithm: an analytical approach," *IEEE Trans. Wireless Commun.*, vol. 3, no. 4, pp. 1326–1337, Jul. 2004.

[39] R. Narasimhan, "Error propagation analysis of V–BLAST with channel estimation errors," *IEEE Trans. Commun.*, vol. 53, no. 1, pp. 27–31, Jan. 2005.

[40] J. Benesty, Y. Huang, and J. Chen, "A fast recursive algorithm for optimum sequential signal detection in a BLAST system," *IEEE Trans. Signal Processing*, vol. 1, no. 1, pp. 1722–1730, Jul. 2003.

[41] B. Hassibi, "An efficient square-root algorithm for BLAST," in *Proc. of IEEE ICASSP*, Istanbul, Turkey, Jun. 2000, pp. 737–740.

[42] H. Zhu, Z. Lei, and F. P. S. Chin, "An improved square-root algorithm for BLAST," *IEEE Signal Processing Lett.*, vol. 11, no. 9, pp. 772–775, Sep. 2004.

Chapter 3

Cross–Layer Design in Wireless Sensor Networks

Contents

3.1 Introduction

Wireless sensor networks (WSNs) are a particular instance of ad hoc networks where wireless nodes are severely constrained in terms of critical resources such as communication rate, processing capabilities, memory and energy. The concept and motivation behind WSNs is mainly the setup of large networks of tiny wireless-enabled nodes, that perform some sort of data gathering and processing unattended or on demand, as required by a control station or by a user.

So far, large-scale electronic integration techniques have allowed the construction of small nodes that yet can perform a number of functions. They can communicate over a radio channel using digital modulation, and can potentially perform any measurement task, depending on the sensing equipment available. Furthermore, they are provided with some sort of computational capability, typically in the form of an integrated micro-controller or processor. Such a resource is shared, and must be used both to manage the low-level functions of the embedded operating system (*e.g.*, TinyOS [1]) and to run networking protocols. Secondarily, high-level tasks, involving for example data processing and formatting, compression, and so forth, also require access to this resource.

WSNs compensate for the lack of powerful resources with the number of nodes that are usually deployed in a network. If long-reach links cannot be formed due to radio power constraints usually more nodes will be available to form a path of relays, so that messages can reach the final destination in a multi-hop fashion. For this reason, similarly to ad hoc networks, medium- to large-sized WSNs are inherently multi-hop. Once deployed, neighboring nodes set up links, and multi-hop paths between communication parties are formed, usually to fit the needs of the application that will be run by the network. When the network starts its operations, in general it continues as long as possible without the need for human-assisted reconfiguration and maintenance. In some cases it may be infeasible, too risky, or even impossible to perform any maintenance (the simplest being battery replacement, or node relocation).

How to set up links, calculate paths, manage communications in an efficient manner, and improve network lifetime to reduce maintenance is currently an open problem and an important research topic [2]. As previously stated, wireless sensor nodes are constrained from many points of view, especially if compared to other wireless nodes. Such constraints include, but are not limited to

- *Limited energy reserve*: as each task performed by the node (especially receptions and transmissions) drains part of the battery energy, it is paramount to take energy issues into account when designing communication protocols.

- *Limited transmit rate and power*: as opposed to more powerful wireless nodes, sensors are usually restricted to using simple digital modulations and low transmit power. The first issue limits the transmit data rate but makes it unnecessary to implement complex synchronization and channel equalization techniques that would otherwise be needed for other kinds of digital modulations. The second issue limits the coverage area and is mainly due to the need to keep the radio equipment small, simple and low-energy.

- *Limited computational power*: to save energy, the on-board processing equipment is usually limited to simple controllers. The operating systems are also designed to be lightweight and to require minimum control overhead.

In general, all constraints are due to the effort of keeping the sensors small-sized and cheap, besides limiting the overall energy consumption of all available apparata. As a side note, wireless sensors are usually designed to be an extremely adaptable platform. They obey a general-purpose concept that allows the end user to build and run his own applications, whose objectives and operational modes may be quite different.

For instance, wireless sensors can be deployed to perform a time-limited real-time fine-grained sensing task in some sort of critical scenario. In this case, the main objective is retrieving the required data as soon as possible, with the focus on timely data delivery instead of energy-efficiency. Warfare scenarios (battlefield movement detection, mine detection, and so on), disaster-recovery and rescue missions, environmental pollution assessment after accidents (a petrol tanker shipwreck, a chemical plant explosion, ad so on), are only some examples of this kind of sensing task. WSNs are a good candidate as they would efficiently perform the required measures without risking human lives in hazardous locations. Their reconfigurability would allow to program fast data sensing and frequent reports, typically at the price of higher energy consumption. Nevertheless, note that energy is not generally an issue in this case, due to the criticality of the scenario.

As opposed to mission-critical scenarios, a number of applications can be found that require the sensors to assume a completely different behavior. Environmental monitoring over a long time period, single or multiple target tracking, ambient intelligence, structural integrity checks for bridges or buildings, intelligent farming, as well as next-generation human-machine interactions are several examples where the main focus of the network design should be posed on efficiency and long lifetime instead of fast data delivery. Protocols should be designed to enforce operational modes that save energy through efficient handshakes and periodical radio shutdown. Yet, these protocols should still have little if any impact on the capability to gather and report data to inquiring stations.

From the overview provided up to this point, it should be clear that WSNs are particular ad hoc networks, where protocol and application design faces many constraints and challenges. For this reason, the cross-layer design paradigm can be proficiently applied to WSNs. An important choice is where the interaction should be enforced. Since the PHY level in wireless sensors is typically fixed, with little programmability beyond transmit power control, it might be preferable to have higher-level functionalities interact.

In the literature, many examples can be found where the MAC and routing layers operate jointly to enable a faster online relay selection. According to the traditional layered approach, multi-hop paths should be computed using some routing algorithm and then fed to the lower layers, where a MAC protocol takes care of managing multiple access among the node and its neighbors, eventually contacting and establishing a link with the next hop as demanded by the routing protocol. As opposed to the additional overhead incurred with this approach, blending MAC and routing yields significant benefits, the most important being a straightforward automation of the relay selection phase. Suppose that the routing part of the joint layer keeps track of a certain metric. This metric measures the preference given to a specific relay. In turn, the MAC part can drive a selection mechanism among some or all neighbors, in order to extract the node with the best metric. The mechanism can involve progressive filtering of nodes offering the worst metric, as well as a contention mechanism that lets nodes exclude themselves when trying to isolate one relay out of a set of almost equivalent neighbors. On the other hand, the specific metric of choice can be adapted to protocol design objectives. In [3,4], the relay offering the maximum advancement is sought, whereas [5] suggests to employ the ratio of the advancement to the power required to reach the relay. In [6] the preferred relay offers the incremental cost that minimizes the overall cost of the path to the destination. This relay is chosen among those exhibiting equal or smaller *hop count*, *i.e.*, number of hops to reach the destination. There are also some examples available where the chosen metric includes some lower-level measures, such as the packet error rate (PER).[1] In particular, the preferred relay in [7] offers the smallest product between the

[1]Note that the PER can be a PHY or a MAC metric, depending on how one defines the role of each layer. For example, if the PHY simply detects the packets and leaves any error check to the MAC layer, the PER can be

PER and the geographic advancement. The set of metrics suggested in [8], generalizes the approach of [7] by defining a normalized advancement (NADV) as the ratio between the advancement and a link cost, and also provides some methods to estimate physical or network layer metrics (error probability, delay, or power consumption) using probe messages or perceived Signal-to-Noise Ratio (SNR) values. In [9] the used metrics include power control information, whereas [10] explores the use of location, queue length and available energy, figuring out how to weigh and account for each of them to obtain performance improvements. To sum up, the general message suggested by this set of works is that cross–layer interactions between the MAC and routing (occasionally, PHY) layers can lead to significantly improved protocol performance.

3.1.1 Focus of this work and organization of the chapter

In the following, we will undertake a cross–layer design approach applied to GeRaF, a joint MAC/routing protocol for WSNs that exploits geographic information. We will develop on the ideas that originated the first version of the protocol, in order to explain how it could be improved in terms of delivery efficiency and energy consumption. We will highlight how substantial advantages are yielded by even simple modifications to the contention algorithms used to elect the next hop. We will discuss the performance of GeRaF in networks with different density and detail the main problems that arise.

Our next step will be the definition of a new metric and a new scheme that aim at improving geographic forwarding performance in a WSN. Finally, we will show that this new scheme can be augmented with a simple algorithm that helps solving one of the major drawbacks of geographic forwarding protocols, namely the impossibility to route packets around connectivity holes using only greedy relay search.[2] A survey of other protocols and algorithms for geographic forwarding and dead-end reduction in WSNs concludes the chapter.

3.2 Geographic Forwarding in Wireless Sensor Networks

Geographic routing protocols have recently emerged as a hot research topic in multihop WSNs, and as a valid alternative to routing algorithms designed for general-purpose ad hoc networks, such as AODV [11], OLSR [12], and DSR [13]. In fact, the distributed optimization of the link costs [11] or recursive path searches [13] both generate a vast amount of signaling, that may consume a lot of energy and subtract resources to application-related traffic with unnecessary overhead. That is especially true for proactive protocols [12] that periodically check the status of network links, and thus require a continuous signaling effort.

defined as a MAC metric.

[2]In the case of a geographic routing scheme, a greedy relay search would choose only nodes that offer a positive advancement toward the destination. However, in the presence of connectivity holes, the current packet holder might be the closest to the destination among all of its neighbors. Since greedy algorithms cannot directly cope with this situation, such a case requires special attention. We will develop more on this topic later.

Geographic protocols adopt a different point of view, as they do not require the construction of any fixed route. A geographic protocol aims at advancing the packet toward the destination with each hop. While the specific rules for choosing the next hop may vary, all neighbors of a sender are generally considered more or less equivalent under a routing point of view, so long as they provide advancement. This is why most geographic routing protocols are defined *flat*, meaning that no nodes have a hierarchically higher role. Furthermore, routing does not require to establish paths proactively or to send specific inquiries before discovering a path to the destination. They allow to forward the packet in the direction of the recipient, using nodes in between as relays for the message. To accomplish this task, no routing table is required, making geographic protocols almost stateless. This is very important in WSNs, because the less information a protocol needs to maintain, the less significant is the burden on the limited sensor memory. Nevertheless, each node needs at least to be aware of its own location, which is a piece of information that is often available or can be gained at reasonable cost with specific equipment [14, 15] or using distributed techniques, possibly with some estimation error [16].

In the following, we will initially focus on a geographic routing protocol specifically designed for WSNs, namely Geographic Random Forwarding (GeRaF) [3,4], and more specifically to its one-radio version [17]. The following section provides a thorough description of this protocol.

3.3 Description of GeRaF

Geographic Random Forwarding (GeRaF) is basically a cross–layer MAC and routing protocol that seeks the maximum advancement per hop while proceeding toward the destination. The relay that best suits this requirement is searched through a greedy scan of the forwarding region, *i.e.*, the area where nodes should be located to offer a non-negative advancement.[3] In order to define the forwarding area properly, the node must then know the location of the destination. For this reason, GeRaF finds its best use in converge-casting scenarios, where all sensors report sensed data back to a single collecting node called the *sink*. Using GeRaF in scenarios where the destination can be any other node in the network would require a mechanism to retrieve its location, such as [18]. This case is conceptually less relevant than the rest of our work here, and out of the scope of the reasoning that follows.

The base version of GeRaF as introduced in [3] works as follows. Each node independently follows a sleep/awake schedule in order to save energy. The schedules are asynchronous with respect to one another, even though they have the same duty cycle d. To perform routing, GeRaF divides the positive advancement zone in N_r regions with area

[3]The forwarding region is the intersection of two circles, one centered on the current transmitter, with radius equal to the coverage radius of a node, and another one centered on the sink, with radius equal to the distance between the destination and the transmitter.

$\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_{N_r}$, each delimited by the coverage area of the transmit node and one or two circle arcs centered at the sink, as in Figure 3.1. The regions are used to discriminate between the advancement offered by different neighbors, and to help choosing the best one. GeRaF assumes that every node knows its own position and the location of the sink. Every other relevant geographic information is piggy-backed using signaling messages. For example, the relays need to know the coordinates of the sender, in order to understand if they offer any advancement and, if so, which region they belong to.

Handshakes are initiated by transmitters after a channel sensing phase that has the purpose to detect other exchanges in progress nearby. The length of the sensing interval will be discussed later. If the channel is free, the transmitter sends a Request-To-Send (`RTS`) message carrying the index of the first region, that silences all neighbors offering negative advancement and serves as a query to the awake nodes in the region closest to the sink, $\mathcal{A}_1$. A contention is then started among these nodes, whereby one of the following events may take place. If only one node is awake in the first region, it reports back with a Clear-To-Send (`CTS`) message, is chosen as a relay, and is sent the data packet. Upon correct reception, it replies back with an `ACK` and immediately senses the channel again to begin advancing the packet further. If more than one node reports back, the sender issues another `RTS` with the same region index. The relays detect this fact and interpret it correctly as a warning that a collision has taken place. Hence, they start a binary splitting process: each node chooses with probability 0.5 whether to answer with another `CTS` or not. The nodes that decided to stay silent go back to sleep if someone else answers. In the case no nodes responded, the transmitter simply triggers another toss.[4] The process eventually ends with a single node sending a `CTS` and being elected as a relay. The last case to consider is when no nodes are awake in the first area. In this case, the sender issues another `RTS` message with the index of the second region, initiating a contention among the awake nodes there. If no nodes can be found, the transmitter sends more `RTS`s to ping the other regions in order of increasing distance from the sink (thus decreasing offered progress), and if no nodes are awake, it backs off and schedules another attempt at a later time.

The original version of GeRaF [3] worked only with those neighbors that were awake at the moment the RTS was sent. However, nodes may happen to wake up during the contention process itself. The help such nodes may offer is twofold. On one hand they may achieve greater progress than the initially awake neighbors. On the other hand, they may be the only relays available, especially in particularly low density scenarios. What we consider in the following is a version of GeRaF that works with the awaking nodes as well. This version, in fact, is more general and performs better, as it can count on more degrees of freedom in the choice of the next hop. From the analysis presented later, it will also be clear that previous versions of GeRaF are special cases of this one.

[4]It should be noted that in this case the `RTS` must contain a flag that explicitly requests the nodes to toss again. Otherwise, according to the preceding rules, they would receive another `RTS` with the same region ID, assume that there has been a collision between other relays, and consequently go to sleep.

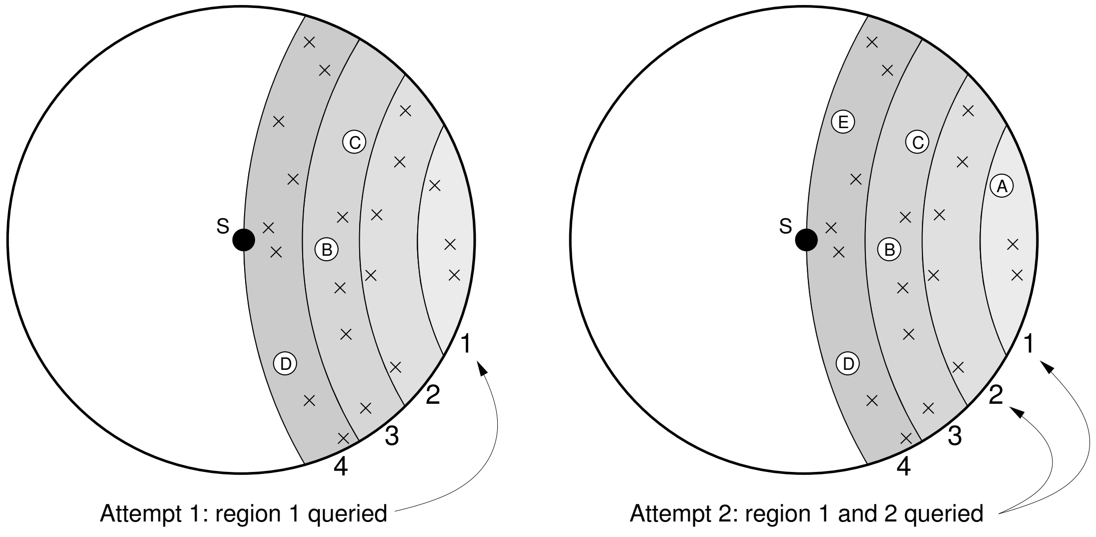

S
C
B
D
1
2
3
4
Attempt 1: region 1 queried
S
E
C
A
B
D
1
2
3
4
Attempt 2: region 1 and 2 queried

Figure 3.1. *GeRaF priority regions and an example of the wakeup process.*

So, awaking nodes can participate to the contention according to the following rule: if they belong to a region that has not yet been queried, they simply wait for their turn to come; if they belong to an already queried region (hence offering greater advancement than any other relay that might be elected in the present contention) they answer the `RTS` regardless of which region is being searched, and then handle possible contention by following the rules explained before. Depending on the specific strategy adopted for handling contentions, performance may vary (more nodes imply longer contentions on average, but also a greater probability of finding a free neighbor for relaying, which may turn out to be more convenient in sparse topologies. An example of the sequential relay search attempts and of the awakening process can be found in Figure 3.1. The different regions are highlighted using increasingly darker grays (the lightest represent the first queried region). Crosses represent sleeping nodes, whereas awake nodes are identified by a letter. At the first attempt, only region 1 is queried, and no node is found awake there. At the second attempt, both regions 1 and 2 are queried, allowing awaking nodes in both regions to participate. In this case, node A can take part in the relay selection, whereas node E cannot, since it has awakened in region 4, that is not being queried.

GeRaF has several advantages over standard approaches to routing in wireless networks. Apart from being geographic with the consequent reduction in the memory required for routing tables, GeRaF integrates MAC message exchanges and the designation of the most convenient relay (from a geographic point of view) in a single framework. This enables channel access and routing in one shot. Moreover, GeRaF is perfectly suited to be used whenever a low energy consumption is of paramount importance, so that the nodes should be kept awake only a small fraction of the time. Thanks to awake/sleep cycles and to its cross–layer design, GeRaF keeps being energy–efficient even in these difficult cases. As a last note, the algorithm itself is simple and easy to implement on real nodes, even those with a limited available memory and computational power (the most difficult arithmetical operation involved is perhaps the calculation of a distance).

3.3.1 Analysis of GeRaF

In this subsection, we detail an analytical model for GeRaF. We then present numerical and simulation results on GeRaF performance, focusing mainly on latency, throughput, and energy consumption. Such results will provide insights on the impact of the different GeRaF design choices on its performance.

3.3.2 Semi-Markov Model

In this section we describe an analytical semi-Markov model for GeRaF. The choice of the proposed model is motivated by the need to trade off accuracy and complexity. More accurate models would require to consider the current state of each network device, resulting in a non-scalable model. We will see that our model, although relatively simple, is detailed enough to keep track of non-trivial events such as the impairments due to simultaneous channel access, MAC-related effects, and their associated delays and energy consumption.

To derive the state–transition structure of the semi-Markov model, we first track the evolution of a node. A node can be in one of the following states:

1. *Sleep*: The node is asleep with no packets to transmit. The transceiver is not operational.
2. *Listen*: The node is idle with no packets to transmit.
3. *Packet Ready*: The node has a packet ready for transmission. The packet might come from the node itself, or may have been received from a neighbor. This is also the state a node goes back to at the end of a backoff.
4. *Transmit RTS*: The node initiates a handshake to select a relay. A node is in this state for the entire contention phase until either a relay is selected or none is found and the node backs off.
5. *Transmit Data*: A handshake has been successfully completed. The data packet is sent to the selected relay.
6. *Receive RTS*: This is the state a node is in while participating to a contention as a relay candidate.
7. *Receive Data*: The node has been selected as a relay. This is the state in which a node receives the data packet and sends the corresponding acknowledgment.

Let us suppose that all nodes follow an awake/sleep cycle in order to save energy, whereby the fraction of awake time over the total time in the absence of traffic (the *duty cycle*) is d. If nodes spend a total time T_{listen} on and T_{sleep} in low power, the duty cycle $d = T_{listen}/T_{duty}$, with $T_{duty} = T_{listen} + T_{sleep}$. Let us also suppose the signaling packets required by the protocol (`RTS`, `CTS`, `ACK`) are all of the same length T_{SIG}. We suppose that nodes are distributed in the network area according to a Poisson process with rate δ, such that the number of neighbors per node (those within the coverage area) is N on average. We also assume that each node has a fixed coverage radius equal to 1. Therefore, $\delta = N/\pi$. Each of the forwarding regions introduced in Section 3.3 has in general a different area, say $\mathcal{A}_j$, $j = 1, \ldots, N_r$. Define $\alpha_j = d\delta\mathcal{A}_j$ as the average number of nodes deployed in region i that are awake at any given instant. Similarly, $\sigma_j = (1-d)\,\delta\mathcal{A}_j$ is the average number of asleep nodes.

Sleeping nodes might wake up during a contention and become eligible relays. In order to model the awakening process, assume for now that a sender node has queried the forwarding region with an RTS and has found no relays. From the point of view of the transmitter, this means sampling the awake/sleep cycles of the neighbors in the forwarding area at a time instant when all of them are asleep. Recall that GeRaF works in attempts, each inquiring one more region than the preceding one. Subsequent attempts are spaced by $2T_{SIG}$, the time required to send a message and wait for a reply. Performing another attempt is equivalent to sampling the awake/asleep process again, so that there is a certain probability that some relays have woken up in the meantime. Consider one relay. Since there are no means for the transmitter to know when this relay will wake up, the wakeup epoch can be modeled as a random variable, uniformly distributed in the interval $[0, T_{sleep}]$. Therefore, the probability that a single node has awakened during a time $t \in \left[0, T_{sleep}\right]$ is t/T_{sleep}. Considering all nodes in region j and recalling that their wakeup processes are independent of each other, the average number of nodes that woke up in region j during t is $\sigma_j t/T_{sleep}$. Call β_{ij} the average number of awaking nodes available at attempt i in region j, and define the overall average number of available nodes as γ_i. For $i = 1$, only awake nodes can reply, thus $\beta_{1j} = 0$, $\forall j = 1, \ldots, N_r$, and $\gamma_1 = \alpha_1$. At the second iteration, we must also consider the nodes awaking in regions 1 and 2 in the $2T_{SIG}$ time interval between the first and second iteration. Therefore, $\beta_{2j} = 2T_{SIG}\sigma_j/T_{sleep}$, $j = 1, 2$, and $\gamma_2 = \alpha_2 + \sum_{j=1}^{2} \beta_{2j}$. At the third iteration, the awaking nodes are those that became active in a time interval of $2T_{SIG}$ in regions 1 and 2 (they were not awake during the previous iterations) and those in region 3 that exited sleep since the relay search started, *i.e.*, in a time interval of $4T_{SIG}$. Thus, $\beta_{3j} = 2T_{SIG}\sigma_j/T_{sleep}$, $j = 1, 2$, $\beta_{33} = 4T_{SIG}\sigma_3/T_{sleep}$, and $\gamma_3 = \alpha_3 + \sum_{j=1}^{3} \beta_{3j}$. In general, at iteration i, only regions up to i are considered, and we have:

$$\beta_{ij} = \frac{2T_{SIG}\sigma_j}{T_{sleep}},\ \forall j = 1, \ldots, i-1, \quad \beta_{ii} = \frac{2(i-1)T_{SIG}\sigma_i}{T_{sleep}} \quad \text{and} \quad \gamma_i = \alpha_i + \sum_{j=1}^{i} \beta_{ij}\,. \tag{3.1}$$

The version of GeRaF presented here can be described with a semi-Markov model with 7 states and 15 transitions as depicted in Figure 3.2, similar to [3]. Due to the changes in the mechanisms of the protocol (and thus in the average number of nodes involved in the transactions) most of the transition probabilities need to be derived again, even though the calculation is similar to that performed in [3]. For the sake of completeness, we will include the derivation of the transition probabilities here.

First of all, a complete description of the model includes the specification of the embed-

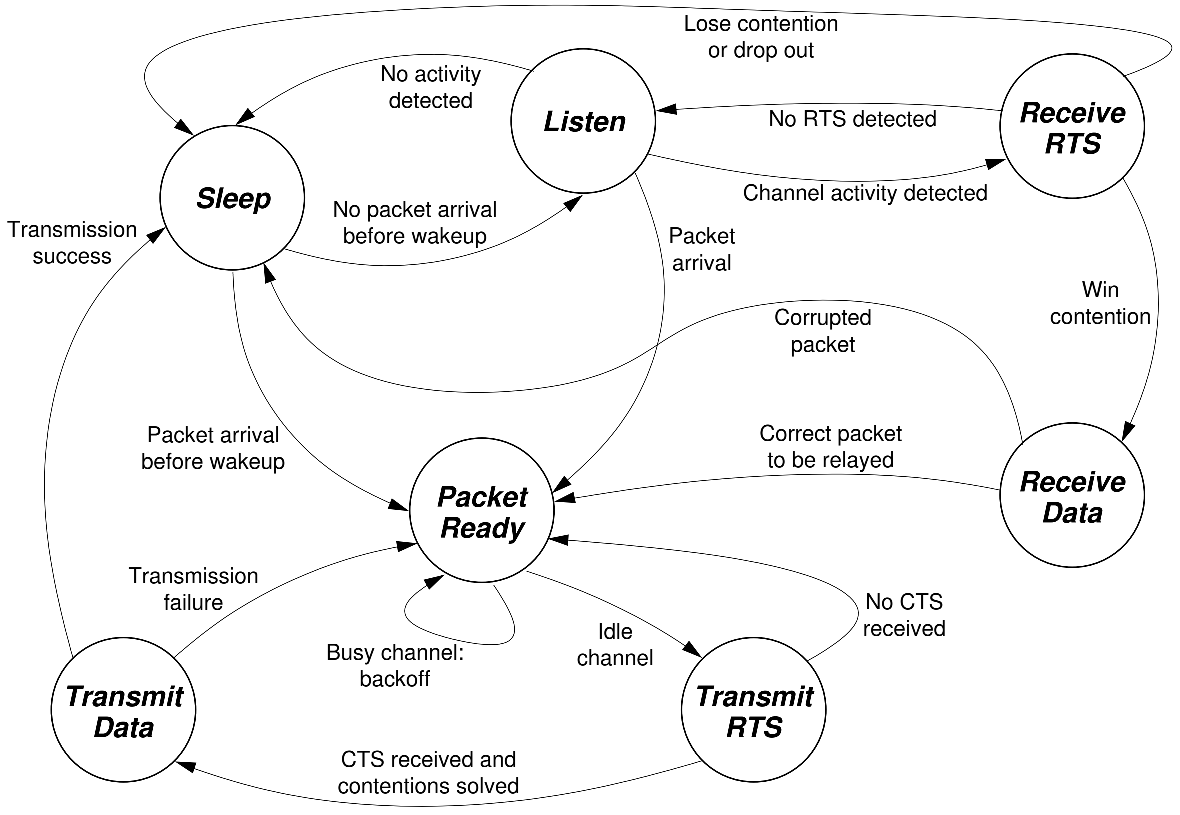


Figure 3.2. *States and transitions of the semi–Markov model for GeRaF.*

ded Markov chain and the metrics associated to each transition (in our case, the transition times suffice to calculate all metrics of interest such as delivery latency, throughput and energy consumption). The transition times are grouped in Table 3.1 on a per transition basis, including also those states whose description is skipped here. The variables used in Table 3.1 are the same introduced in the corresponding paragraph describing the state reported in the column "From." There, as well as in the following, λ indicates the average packet generation rate per node, in packets per second.

3.3.3 State *Sleep*

The node is in a low power state, with an inactive radio, yet it may generate new packets to send. This leads to state *Packet Ready*, with probability $1 - e^{-\lambda T_{sleep}}$, after an average time $t = \frac{1}{\lambda} - \frac{T_{sleep}}{e^{\lambda T_{sleep}} - 1}$ spent in sleep.

The other possible transition leads to *Listen*, and takes place if no traffic is generated while the node is asleep (with probability $e^{\lambda T_{sleep}}$). The time spent sleeping in this case is simply T_{sleep}.

3.3.4 State *Listen*

In this state, the node is actively monitoring the channel and has no queued packets to send. Idle listening may be interrupted if a packet is locally generated (by neighbors or by the node itself), and the corresponding source begins handshake operations. Since the average number of nodes within coverage is N, the arrival rate of the whole neighborhood is therefore $\lambda(N+1)$. Given that a packet is generated, it arrives on average after a time $t = \frac{1}{\lambda(N+1)} - \frac{T_{listen}}{e^{\lambda(N+1)T_{listen}}-1}$. If the packet is generated by one of the neighbors, with probability $\frac{N}{N+1}\left(1 - e^{-\lambda(N+1)T_{listen}}\right)$, an RTS transmission is detected, and the node goes to state *Receive RTS*. Otherwise, with probability $\frac{1}{N+1}\left(1 - e^{-\lambda(N+1)T_{listen}}\right)$, the node is the source and goes to *Packet Ready* for channel sense. In any other case, no signals are received, the normal duty cycle is followed, and the node returns to sleep after listening for T_{listen}.

3.3.5 State *Transmit RTS*

In this state, a node has perceived a free channel after sensing and transmits an RTS. The node could discover either some or no awake nodes. The first event has probability $P_{CTS} = 1 - \prod_{i=1}^{N_r} e^{-\gamma_i}$ and leads to state *Transmit Data*. Otherwise, state *Packet Ready* is entered with probability $1 - P_{CTS}$. In the first case, the average time to solve a contention (in number of attempts to receive a CTS, also referred to as CTS slots) is can be found by considering how many slots are left empty (due to the fact that no relay are found, and how many slots it takes to solve a contention. Let us call x_0 the average number of unsuccessful attempts. The probability that i attempts are unsuccessful is calculated as the probability that nobody is awake or has awakened up to attempt i and that somebody is found at attempt $i+1$. Hence x_0 can be found as

$$x_0 = \frac{1}{P_{CTS}} \sum_{i=0}^{N_r-1} \left(\prod_{j=1}^{i} e^{-\gamma_j} \right) \left(1 - e^{-\gamma_{i+1}}\right) i \,. \tag{3.2}$$

Let us now call x_1 the average number of CTS slots needed to isolate a single relay out of a set of responders, given that at least one relay is present. As before, we need to track which is the successful attempt. The probability that no relays are found up to the ith attempt is derived as before. Once we know that somebody is found at attempt $i+1$, the number of CTS slots required to identify a relay depends on the number of neighbors that respond. The distribution of such a number is Poisson with parameter γ_{i+1}. By averaging over the number of responders and over the attempts, x_1 is found as

$$x_1 = \frac{1}{P_{CTS}} \sum_{i=0}^{N_r-1} \left(\prod_{j=1}^{i} e^{-\gamma_j} \right) \sum_{k=1}^{\infty} \frac{e^{-\gamma_{i+1}} \gamma_{i+1}^{k}}{k!} s_k \,. \tag{3.3}$$

In other words, x_1 is the time elapsed from the first answered CTS to the time the collision is resolved. In (3.3), the second factor accounts for the probability that the first i iterations are unsuccessful, the third for the probability that $k \geq 1$ nodes contend during the $(i+1)$th iteration, and s_k is the average number of slots required to solve a contention involving k relays. It is easy to find s_k for a given k by considering a Markov chain over the states $1, 2, \ldots, k$. Let i be the number of nodes that are contending to be a relay at a given time. Each node chooses with probability 0.5 whether to transmit another CTS or to stay silent. If all nodes stay silent or if all transmit a CTS, the following contention will still involve i nodes. Otherwise, $i - j$ nodes decide to stay silent, and the following contention will involve j nodes. The transition probabilities of this Markov chain are therefore

$$p_{ii} = 2 \cdot 2^{-i}, \qquad p_{ij} = \binom{i}{j} 2^{-i}, \quad j = 1, \ldots, i-1, \quad i = 1, \ldots, k\,. \tag{3.4}$$

Hence, s_k is the average first-passage time from state k to state 1, which is computed by solving the following system of linear equations in $k-1$ variables [19]:

$$s_i = 1 + \sum_{j=2}^{i} p_{ij} s_j\,, \quad i = 2, \ldots, k\,, \tag{3.5}$$

where p_{ij} is as in (3.4). Note that (3.5) actually yields all other s_i, $i < k$ as well.

3.3.6 State *Transmit Data*

The action taken in this state is sending a data packet after a successful handshake completion. After transmission, the node waits for an ACK and then goes to the *Sleep* state. The transmission is successful only if both the data packet and the ACK are received correctly, which happens with probability P_D and P_A, respectively. In this case, with probability $P_D P_A$ the sender goes back to sleep. Otherwise, it switches to state *Packet Ready* where it will reschedule the transmission after a backoff. The related times are found in Table 3.1.

3.3.7 State *Packet Ready*

This state represents the availability of a packet for transmission. Here, the node senses the channel for a time T_{sens} and reschedules a later attempt after $T_{backoff}$ if it is found busy. The length of the sensing time must ensure a high probability that if a handshake is taking place nearby, either a transmitter- or a receiver-originated message is detected. In either case, the length of the data transmission and the message exchange during the relay search must be protected. Hence, T_{sens} must be set to

$$T_{sens} = \max\{T_D,\ T_{RTS} + (N_r - 1)(T_{CTS} + T_{RTS})\}\,, \tag{3.6}$$

where the terms T_D, T_{RTS} and T_{CTS} represent the transmission time for a packet of the respective type. The first term ensures that the data transmission is not interfered with (so that no collision on the packet or ACK should take place), and thereby protects a sender node. The second term, instead, protects an awake relay, since it represents the maximum duration of a silence before a CTS message is issued by a relay during a search. Usually, $T_{sens} = T_D$.[5]

Otherwise, it switches to *Transmit RTS* to begin a handshake. The first case takes place with probability P_{idle} and leads to *Transmit RTS*, whereas with probability $1 - P_{idle}$ there is a transition to *Packet Ready*. The related times can be found in Table 3.1.

P_{idle} is found through the following argument. Let s be the sensing node. Any neighbor of s that is involved in a communication (RTS/CTS handshake or actual data/ACK exchange) as a transmitter or as a receiver will result in the channel being sensed busy, and will cause s to back off. Let us focus on an infinitesimal area $\mathrm{d}A$ at distance r from s and calculate how this area contributes to the channel being sensed busy. On average, $(N/\pi)\mathrm{d}A$ nodes are found inside $\mathrm{d}A$. Each of these nodes can i) become a transmitter, with rate λ, or ii) become a receiver. In the latter case, in steady-state the rate at which a node is involved as a receiver is also λ. However, all data communications originated by a node which is also within coverage of s must be ignored in this case, as otherwise the same event would be counted twice. Therefore, the rate corresponding to case ii) is computed considering only the traffic originated by nodes outside s's coverage area. Since the area of the union of two unit-radius circles, one centered in s and the other centered at distance r from s, is

$$A(r) \;=\; \pi + 2\sin^{-1}\left(\frac{r}{2}\right) + r\sqrt{1-\left(\frac{r}{2}\right)^2}\,, \tag{3.7}$$

the region where originating traffic is to be considered has area $A(r) - \pi$, and hence the total rate at which a node inside $\mathrm{d}A$ is detected as active by s is $\lambda\left(1 + \frac{A(r)-\pi}{\pi}\right) = \lambda A(r)/\pi$. Now, the average time to complete a data transmission (thus the average time the channel is occupied) can be computed as $T_T = x_0 + x_1 + T_{DATA} + T_{ACK}$ (x_0 and x_1 are as for state *Transmit RTS* in Section 3.3.5). Hence, we can define a vulnerability time $\mathcal{T}(r)$ during which any handshake started in the neighborhood of s and involving a node inside $\mathrm{d}A$ forces s to back off as

$$\mathcal{T}(r) = \begin{cases} T_T + T_{sens}, & \text{with prob. } \pi/A(r) \text{ (case } i\text{) above)} \\ T_T + T_{sens} - T_{RTS}, & \text{with prob. } (A(r)-\pi)/A(r) \text{ (case } ii\text{) above)} \end{cases}, \tag{3.8}$$

because in the second case the sensed channel occupancy starts after the RTS is received. Since both the traffic activity and the node placement are Poisson processes (in time and space, respectively), the number of events that cause s to detect a busy channel is a Poisson random variable with average

$$\begin{aligned} \Lambda \;&=\; \int_{C(s)} \frac{N}{\pi}\frac{\lambda A(r)}{\pi}\mathcal{T}(r)\,\mathrm{d}A = \int_0^1 \frac{N\lambda A(r)}{\pi}\mathcal{T}(r)2r\,\mathrm{d}r && (3.9) \\ &=\; \frac{\lambda N}{4\pi}\left[(3\sqrt{3}+4\pi)(T_T+T_{sens}) - 3\sqrt{3}T_{RTS}\right], && (3.10) \end{aligned}$$

[5] If two radios were available to transmit and receive on different frequencies, a node could issue a busy tone using the second radio during the reception periods of the first radio. This way, the sensing phase can be skipped, as a busy channel is always notified by either the data or the busy tone radio.

where $\mathrm{C}(s)$ is s's coverage area. Therefore, the probability that s finds the channel idle is $P_{idle} = e^{-\Lambda}$.

As a side remark, note that a handshake initiated by a neighbor can end up in a backoff. What we considered in (3.10) is a pessimistic bound where every handshake occupies the channel for T_T on average. Removing this bound would require to keep track of the behavior of one-hop neighbors at least, thus increasing the complexity of the model. Nevertheless, we will show that our bound here is tight enough to model the situations of interest.

3.3.8 State *Receive RTS*

In this state an idle (listening) node hears a beginning transaction (an RTS) and participates in relay selection operations, if it is in the forwarding area. Otherwise it goes back to sleep. One of the following events may take place:

1. The node is not in the forwarding area;
2. The node is in the forwarding area but drops out before sending a CTS;
3. The node is in the forwarding area, participates to a contention, and either wins or loses.

The node is in the relay area if it is placed in any of the priority regions, an event with probability $\xi = \left(\sum_{k=1}^{N_r} \mathcal{A}_k\right)/\pi$. So, the node receives the RTS and goes directly to sleep with probability $1-\xi$. Otherwise, the node is in the relay area and events 2) or 3) as before may happen. For a clearer description, let

$$b_i \;=\; P\left[\text{node in region } \mathcal{A}_i \mid \text{node in fwd area}\right] = \frac{\mathcal{A}_i}{\sum_{k=1}^{N_r} \mathcal{A}_k} \tag{3.11}$$

$$c_i \;=\; P\left[\text{node is in } \textit{any} \text{ of } \mathcal{A}_j,\ j=1,\dots,i \mid \text{node in forwarding area}\right] = \sum_{j=1}^{i} b_j . \tag{3.12}$$

Let $\psi_i = 1 - c_i$ be the probability that a node cannot contend at the current attempt i because it is located in any of the regions $i+1,\dots,N_r$. Recall in fact that GeRaF is structured in subsequent attempts, each allowing nodes from one more region to respond. Case 2) takes place if the node is in region $i+1$ (and thus reachable not before attempt $i+1$) but some nodes were either awake in region $j=1,\dots,i$ or have woken up in the same regions before attempt i. In this case, the node drops out. As before, we calculate the drop out probability by considering the event that no nodes are found until attempt i, some nodes are found at attempt $i+1$, and the node cannot contend at the attempt $i+1$ (this last probability is ψ_{i+1}). This yields

$$p_d \;=\; \xi \sum_{i=0}^{N_r-1} \left(\prod_{j=1}^{i} e^{-\gamma_j}\right) \left(1 - e^{-\gamma_{i+1}}\right) \psi_{i+1}, \tag{3.13}$$

where we recall that each term γ_i jointly accounts for nodes already awake at the beginning of the handshake and for nodes that have awakened in the meanwhile, in any feasible region. The average number of involved CTS slots, conditioned on the event that the node drops out, is then

$$x_d = \frac{\xi}{p_d} \sum_{i=0}^{N_r-1} \psi_{i+1} \left(\prod_{j=1}^{i} e^{-\gamma_j} \right) \left(1 - e^{-\gamma_{i+1}}\right) (i+1). \tag{3.14}$$

Case 3) implies that the node is eligible for contending as a relay, hence it is in the forwarding area and belongs to region i while attempt m is being performed, $m \geq i$. It participates but loses if the winner is selected to be one of other, say, k nodes being available at attempt m. Due to the use of binary splitting for collision resolution, each user has the same probability of being elected as the winner. Thus, the probability that all the previous events take place and the node loses is

$$\xi \left(\prod_{j=1}^{i} e^{-\gamma_j} \right) (1 - \psi_{i+1}) \frac{e^{-\gamma_{i+1}} \gamma_{i+1}^k}{k!} \frac{k}{k+1}, \tag{3.15}$$

where we account that the node is in the relay area, that nobody responded at attempts $j = 1, \ldots, i$, that the node is allowed to participate during attempt $i+1$ and that the winner is one of k contenders other than the node. By averaging on the Poisson distribution[6] for the number of available relays, we get

$$\begin{aligned} p_\ell &= \xi \sum_{i=0}^{N_r-1} \left(\prod_{j=1}^{i} e^{-\gamma_j} \right) (1 - \psi_{i+1}) \sum_{k=0}^{\infty} \frac{e^{-\gamma_{i+1}} \gamma_{i+1}^k}{k!} \frac{k}{k+1} \\ &= \xi \sum_{i=0}^{N_r-1} \left(\prod_{j=1}^{i} e^{-\gamma_j} \right) (1 - \psi_{i+1}) \left(1 - \frac{1 - e^{-\gamma_{i+1}}}{\gamma_{i+1}} \right), \end{aligned} \tag{3.16}$$

from which the corresponding conditional average number of slots involved is $x_\ell = x_{\ell 0} + x_{\ell 1}$, where

$$x_{\ell 0} = \frac{\xi}{p_\ell} \sum_{i=0}^{N_r-1} \left(\prod_{j=1}^{i} e^{-\gamma_j} \right) (1 - \psi_{i+1}) \left(1 - \frac{1 - e^{-\gamma_{i+1}}}{\gamma_{i+1}} \right) i \tag{3.17}$$

$$x_{\ell 1} = \frac{\xi}{p_\ell} \sum_{i=0}^{N_r-1} \left(\prod_{j=1}^{i} e^{-\gamma_j} \right) (1 - \psi_{i+1}) \sum_{k=0}^{\infty} \frac{e^{-\gamma_{i+1}} \gamma_{i+1}^k}{k!} \frac{k}{k+1} s_{k+1}. \tag{3.18}$$

Here, $x_{\ell 0}$ is the number of slots in which the current attempt does not allow the node to transmit a `CTS`, whereas $x_{\ell 1}$ is the average number of `CTS` slots from when the node participates in the contention to when the contention is resolved.

[6] Notice that the Poisson distribution is calculated for k users, not $k+1$, as the it is conditioned on the presence of the node being modeled.

The final possible event is that the node wins a contention. Similarly to (3.16), the probability related to this case is calculated as

$$\begin{aligned} p_w &= \xi \sum_{i=0}^{N_r-1} \left(\prod_{j=1}^{i} e^{-\gamma_j} \right) (1-\psi_{i+1}) \sum_{k=0}^{\infty} \frac{e^{-\gamma_{i+1}} \gamma_{i+1}^k}{k!} \frac{1}{k+1} \\ &= \xi \sum_{i=0}^{N_r-1} \left(\prod_{j=1}^{i} e^{-\gamma_j} \right) (1-\psi_{i+1}) \left(\frac{1-e^{-\gamma_{i+1}}}{\gamma_{i+1}} \right) \end{aligned} \tag{3.19}$$

and the conditional average number of slots involved is $x_w = x_{w0} + x_{w1}$, where

$$x_{w0} = \frac{\xi}{p_w} \sum_{i=0}^{N_r-1} \left(\prod_{j=1}^{i} e^{-\gamma_j} \right) (1-\psi_{i+1}) \left(\frac{1-e^{-\gamma_{i+1}}}{\gamma_{i+1}} \right) i \tag{3.20}$$

is the average number of empty slots, and

$$x_{w1} = \frac{\xi}{p_w} \sum_{i=0}^{N_r-1} \left(\prod_{j=1}^{i} e^{-\gamma_j} \right) (1-\psi_{i+1}) \sum_{k=0}^{\infty} \frac{e^{-\gamma_{i+1}} \gamma_{i+1}^k}{k!} \frac{1}{k+1} s_{k+1} \tag{3.21}$$

is the average number of CTS slots up to the election of the node as the winner.

Now, if the node is not in the relay area, drops out before being engaged in a contention, or gets engaged but loses, then it turns off its radio and goes to sleep. This event has a total probability of $(1-\xi) + p_d + p_\ell$. According to the discussion at the end of Section 3.3.5, the conditional average number of slots in which the node does not send a CTS is then

$$x_0 = \frac{p_d x_d + p_\ell x_{\ell 0}}{(1-\xi) + p_d + p_\ell} \tag{3.22}$$

and the conditional average number of slots where the node may transmit a CTS is

$$x_1 = \frac{p_\ell x_{\ell 1}}{(1-\xi) + p_d + p_\ell} \tag{3.23}$$

If the node is not in the relay area, it drops out immediately, so this case gives no contribution to the above calculations. Upon winning the contention, the node goes directly to the state *Receive Data* and waits for the incoming packet. The times related to the transitions are found in Table 3.1. In the following we will also suppose that when a node is idle and detects newly incoming traffic, it always hears an RTS. This approximation is tighter at low traffic, and numerical results will indicate that this transition never plays an important role. Hence, we will always assume that the probability of detecting a packet different from an RTS is zero.

3.3.9 State *Receive Data*

Upon being elected as a winner, a node enters this state and receives the packet. If the transmission is successful (both the data and the ACK are received correctly), the node switches to *Packet Ready* in order to sense the channel and forward the data packet further. This event has probability $P_D P_A$. Otherwise, the transmission fails and the node goes back to *Sleep*.

From	To	Transmit	Receive	Listen	Sleep
Listen	*Packet Ready*	0	0	$\frac{1}{\lambda(N+1)} - \frac{T_{listen}}{e^{\lambda(N+1)T_{listen}}-1}$	0
	Receive RTS	0	0	$\frac{1}{\lambda(N+1)} - \frac{T_{listen}}{e^{\lambda(N+1)T_{listen}}-1}$	0
	Sleep	0	0	T_{listen}	0
Sleep	*Packet Ready*	0	0	0	$\frac{1}{\lambda} - \frac{T_{sleep}}{e^{\lambda T_{sleep}}-1}$
	Listen	0	0	0	$\frac{1}{\lambda} - \frac{T_{sleep}}{e^{\lambda T_{sleep}}-1}$
Transmit RTS	*Packet Ready*	$T_{RTS} + N_r T_{RTS}$	0	$N_r T_{CTS}$	$T_{backoff}$
	Transmit Data	$T_{RTS} + (x-1)T_{RTS}$	$x_1 T_{CTS}$	$x_0 T_{CTS}$	0
Transmit Data	*Sleep*	T_D	T_{ACK}	0	0
	Packet Ready	T_D	0	T_{ACK}	$T_{backoff}$
Packet Ready	*Transmit RTS*	0	0	T_{sens}	0
	Packet Ready	0	0	T_{sens}	$T_{backoff}$
Receive RTS	*Sleep*	$x_1 T_{CTS}$	$T_{RTS} + x_w T_{RTS}$	0	$x_{w0} T_{CTS}$
	Receive Data	$x_{w1} T_{CTS}$	$T_{RTS} + x_w T_{RTS}$	0	$x_{w0} T_{CTS}$
Receive Data	*Packet Ready*	T_{ACK}	T_{DATA}	0	0
	Sleep	0	T_{DATA}	0	0

Table 3.1. *Average transition times. Variables refer to the definition given in the corresponding paragraph.*

3.4 Performance Evaluation

3.4.1 Analytical framework

The main metrics we wish to evaluate in this analysis are average throughput inside coverage, energy consumption (normalized to that of an always-on radio) and channel access latency (defined as the time from when the first channel sense takes place to when the packet transmission that will end correctly starts). The theory of renewal reward processes [19] allows to derive all such metrics using the model described in Section 3.3.1. In particular, solving the semi-Markov chain yields the average fraction of time spent by a node in each of the states, in the long run. Let

$$\Pi = \left[\Pi_{\mathtt{lst}},\, \Pi_{\mathtt{slp}},\, \Pi_{\mathtt{PKTrdy}},\, \Pi_{\mathtt{txRTS}},\, \Pi_{\mathtt{txDATA}},\, \Pi_{\mathtt{rxRTS}},\, \Pi_{\mathtt{rxDATA}}\right] \tag{3.24}$$

be the vector containing such times, where the names of the states have been conveniently abbreviated. In our model, a node can generate new packets only if it is not currently handling other packets, *i.e.*, if it is in state *Listen* or *Sleep*. Therefore throughput is obtained as the packet arrival rate weighed with the probability that the node is in either of these two states. By normalizing to the duration of a data packet, T_{DATA}, the average throughput for all N nodes in coverage is thus $\lambda N T_{DATA}\left(\Pi_{\mathtt{lst}} + \Pi_{\mathtt{slp}}\right)$. The average energy consumption is obtained similarly, by weighing the time needed for each transition with the amount of power absorbed by the radio during that transition. This yields a measure of average energy expenditure in each state, which is then used as a "reward" for the process being evaluated. Average latency is obtained instead by considering the first passage times θ_{ij} between pairs of states in the chain, which are given by the solution of the following system of linear equations [19]

$$\theta_{ij} = \varphi_i + \sum_{r \neq j} P_{ir}\theta_{rj}, \tag{3.25}$$

where φ_i represents the average metrics accumulated in state i once it is entered (for latency, this means time) and is obtained by taking the weighed sum of the transition metrics from state i with the related probabilities from matrix P. In this case, the relevant path joins state *Packet Ready* to state *Sleep*, and allows to calculate the average time needed to successfully deliver a packet. Latency is derived from this first passage time by subtracting the duration of the data and ACK transmissions.

3.4.2 Simulation parameters

The analytical model has been compared with simulation results obtained with ns2. Our objective is twofold. On one hand we wish to confirm the good performance achieved by GeRaF. and use the analysis to identify optimal values for certain network parameters. On the other hand, we validate analytical results by emulating the protocol behavior in a fully detailed network, where second order effects and consequences that analysis is not able to catch may prove to have some significant impact. The network has been deployed in an area of $320\,\mathrm{m} \times 320\,\mathrm{m}$. Depending on the wanted average density, 100 or 1024 nodes are organized in a square grid within that area. The coverage radius is $40\,\mathrm{m}$. The sink is placed at the center of the network. Packets are generated according to a Poisson process with rate λ packets per second and, if possible, stored in the buffer of the sensor, which is assumed to accommodate one packet at most. The one-hop neighbors of the sink do not generate packets, which helps reducing the bottle-neck effect as data converge to the sink. It also preserves the statistical meaning of the results for high traffic, when congestion would cause the neighbors of the sink to be the only ones to forward data correctly, yielding an unrealistically small latency. Data packets are $2000\,\mathrm{bits}$ long, whereas the length of signaling packets is $200\,\mathrm{bits}$. The channel rate is $38.4\,\mathrm{kbps}$, so that $T_{SIG} = 5.21\,\mathrm{ms}$ and $T_D = 52.1\,\mathrm{ms}$. We also ideally assume $P_D = P_A = 1$ as this is in accordance with the simulation data. Other parameters have been chosen according to the following discussion.

3.4.3 Optimal protocol parameter setting

First, we aim at finding suitable values for some tunable protocol parameters and timings. Specifically, we focus on the number of relay search regions N_r, the average backoff time $T_{backoff}$, the awake time T_{listen} and the sensing time T_{sens}. First of all, let us consider N_r. This parameter is important for two reasons, namely it sets the number of relay regions and also the average duration of the interval when an asleep node could wakeup and take part in a contention. In fact, recall that it takes more time to scan the whole forwarding area if the number of regions is high, and additional nodes can wake up during this time. In turn, this improves the probability to find at least one relay. This has a twofold impact on throughput. On one hand, having more relays available yields a greater capability of handling traffic. On the other hand, a greater N_r lengthens the time required to scan the regions, thereby increasing channel access latency (albeit this might also reduce the number of nodes that

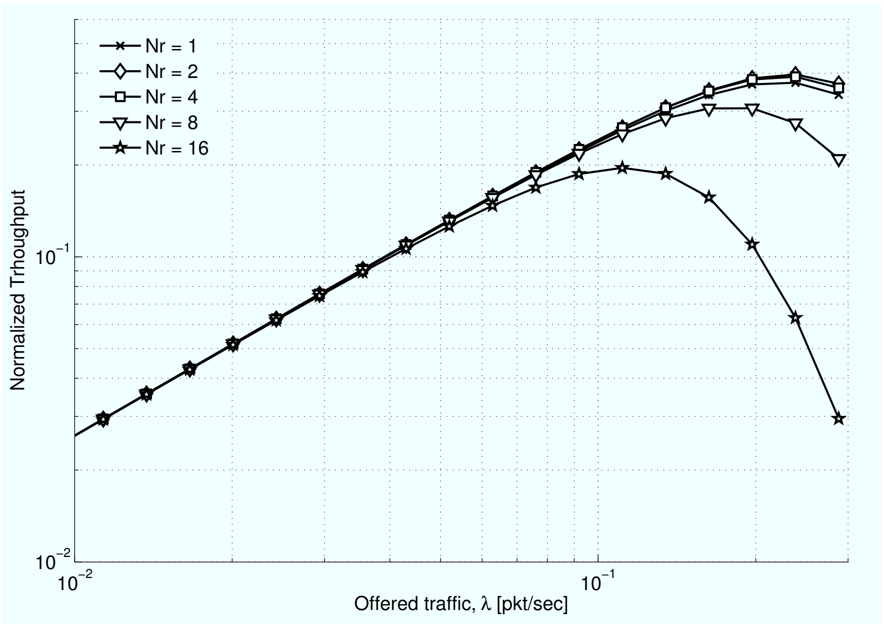


Figure 3.3. *Throughput against offered traffic for different values of N_r, $N = 50$, $d = 0.1$.*

contend, favoring short relay choices). Hence, a well-chosen N_r balances between the need for high throughput and low latency. To exemplify this choice, compare Figures 3.3 and 3.4, that show the average throughput against λ and the average latency against throughput, respectively. We consider $N = 50$ nodes within coverage on average, and a duty cycle $d = 0.1$. From Figure 3.3 we see that the optimal choice for throughput in this case is $N_r = 2$. This value gives the best latency performance as well (Figure 3.4). We stress that the best value of N_r depends on both N and d. The same analysis must hence be repeated for any considered value of N and d. For example, for $N = 50$, $d = 0.01$, we find $N_r = 6$, for $N = 5$, $d = 0.1$ we have $N_r = 4$, and $N_r = 8$ for $N = 5$, $d = 0.01$. As a side note, Figure 3.4 is plotted as a function of throughput, which is in turn a concave function of traffic. For this reason, all curves tend to fold back. This also applies to Figures 3.5 and 3.8–3.12.

Considerations similar to those presented for N_r hold for the choice of $T_{backoff}$ as well. First of all, note that this interval must be long enough for nodes to find an uncorrelated set of awake neighbors after a failed attempt. To have a practical backoff length, we impose that $T_{backoff}$ be a multiple of $T_{last} = N_r(T_{RTS} + T_{CTS}) + T_{DATA} + T_{ACK}$, *i.e.*, the duration of a handshake that finds one relay during the last region query. The analytical results suggest that, by decreasing $T_{backoff}$, both throughput and latency improve (not shown here due to lack of space). However, this also affects energy consumption. This byproduct can be seen in Figure 3.5, showing the normalized energy consumption against throughput for $N = 50$ and $d = 0.1$. At low traffic, the consumption is equal to the duty cycle of the node. When traffic increases (corresponding to approaching the maximum throughput value), two different behaviors are possible. If $T_{backoff}$ is long enough, a node stays asleep for a longer time after an unsuccessful transmission attempt. Therefore, its energy consumption tends to decrease

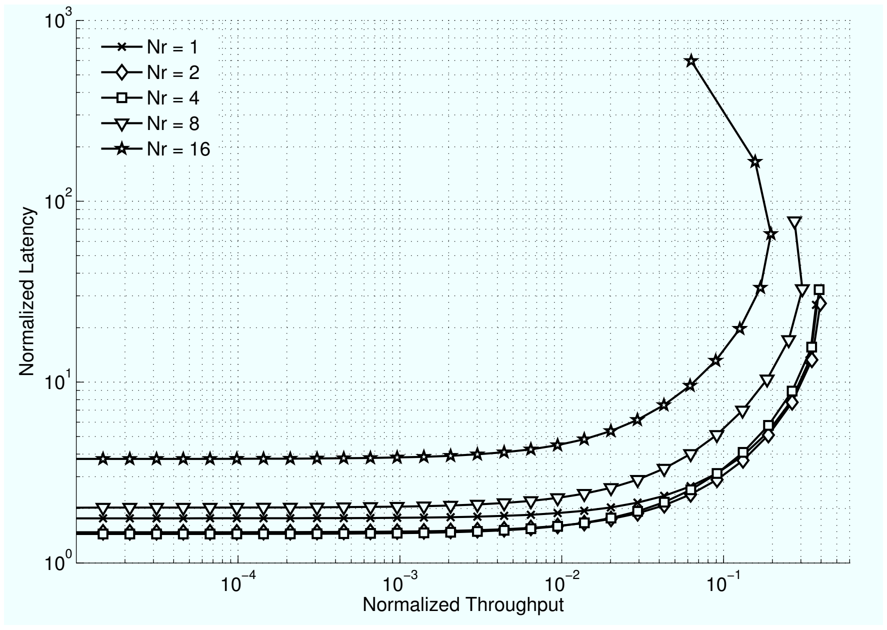


Figure 3.4. *Latency against throughput for different values of* N_r, $N = 50$, $d = 0.1$.

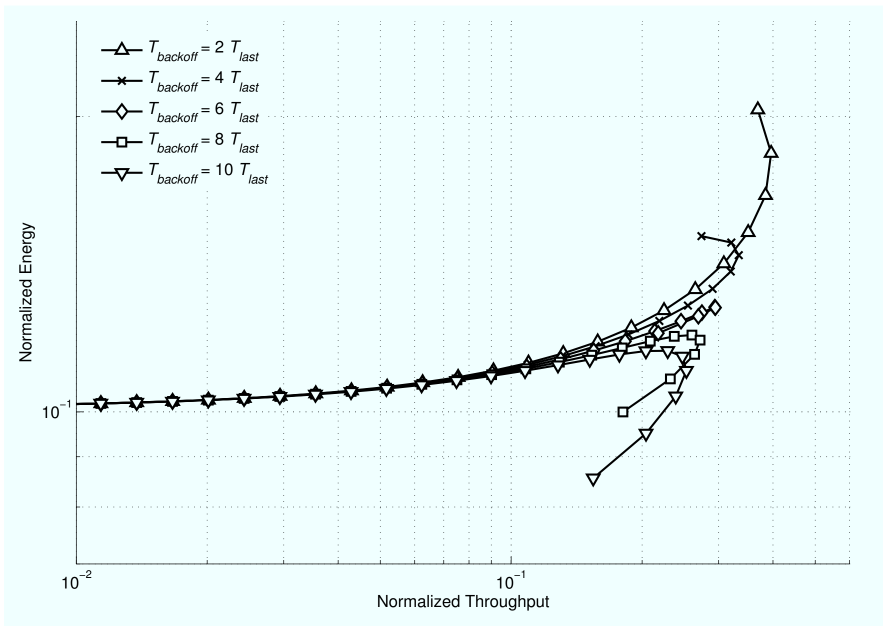


Figure 3.5. *Energy against throughput for different values of* $T_{backoff}$, $N = 50$, $d = 0.1$.

for increasing traffic. Conversely, if $T_{backoff}$ is shorter, the transmission attempts are more frequent, and the energy consumption increases. Now, one may decide that it is desirable for nodes to spend less energy when the network is congested (*i.e.*, when the throughput decreases and the curves tend to fold back), and hence a high $T_{backoff}$ is better. However, this would decrease the maximum throughput and increase latency, thereby making congestion occur at lower offered traffic. Therefore, in the forthcoming evaluation, we have set $T_{backoff} =$

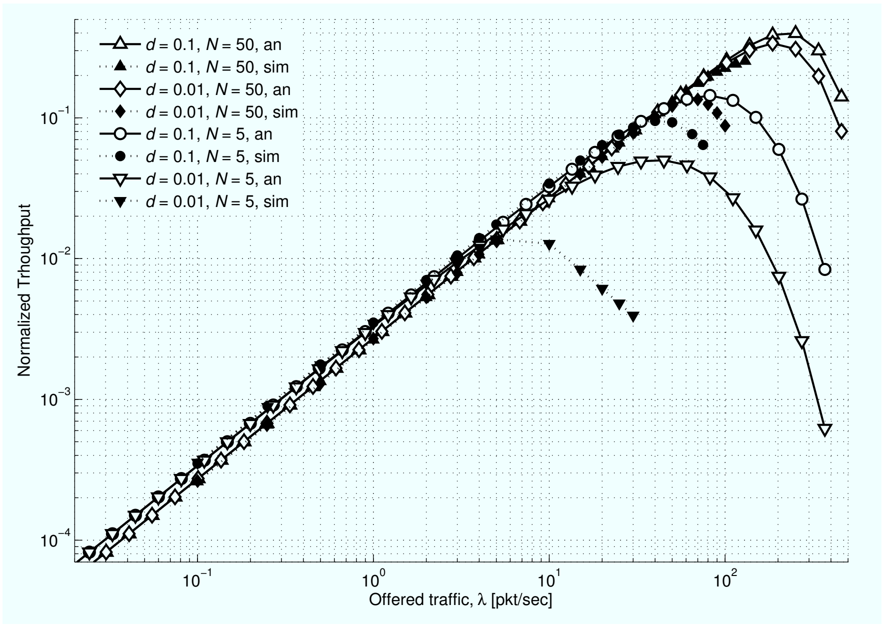


Figure 3.6. *Throughput as a function of offered traffic.*

$2T_{last}$, which allows for low latency and high throughput for any value of N and d.

All other protocol parameters have been set according to similar considerations. The analysis shows that the best values for T_{listen} and T_{sens} are in any case the lowest allowed. This is also intuitively explained, because a higher T_{sens} extends the vulnerability time of the protocol and decreases the communication efficiency. An equivalent argument holds for T_{listen}, with the additional remark that the energy consumption is not impacted significantly as the only factor that matters in this case is d, not the actual length of the listening phase.

It should be noted that, for practical reasons, the backoff interval length should be chosen randomly, so that nodes do not wake up for a new attempt at the same time. Hence, in the simulations described in the following, we choose this interval uniformly between 1 and 3 times T_{last}. This is a sufficiently long time to let the awaking node find an uncorrelated set of awake neighbors.

3.4.4 One-hop GeRaF performance

We start by evaluating the network throughput performance as defined in Section 3.4.1, which is depicted in Figure 3.6 for $N = 5, 50$ and $d = 0.01, 0.1$. After a first increase proportional to λ, congestion builds up, thus throughput reaches a maximum and decays. The uncongested region is very well predicted by analysis, and it is interesting to note that i) the maximum supported traffic is greater in analysis than in simulation, and ii) the approximation is better for greater values of N and d. This result is explained by the following fact. The analysis can only follow the evolution of a single node, and must assume that all neighbors behave accordingly to the steady-state probability distribution. Jointly tracing the evolution

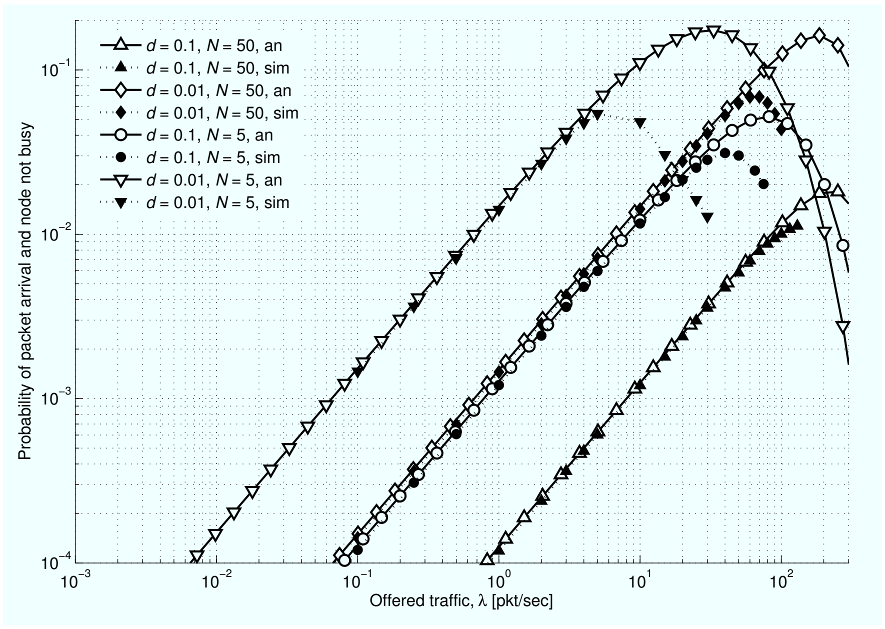


Figure 3.7. *Probability that a packet is generated and the node is free to handle it, as a function of offered traffic.*

of all nodes would result in a completely accurate Markov model. However, this model would be unmanageable due to the very large number of states and transitions involved.

Both the previous observations i) and ii) are explained in terms of the increasingly worse impairments encountered when trying to forward a packet. When a node has data to forward, it queries its neighbors to find a suitable relay. However, in simulations, in-range nodes could either have their own packets to transmit or be busy in the further advancement of previously received data. In this case, they would not respond to RTSs, so that transmitters with busy neighbors experience a lower average degree, which in turn causes more backoffs, decreasing the maximum amount of traffic that gets through. In analysis, this effect cannot be modeled, for the same reasons already described above. A similar consequence of this effect can be observed in Figure 3.7, that depicts the total probability of acceptance of a newly generated packet, *i.e.*, the probability that a node is either in state *Listen* or in state *Sleep*, thus free to handle any newly generated packet, and a data packet is in fact generated. This probability is found as $\Pi_{\texttt{lst}} \frac{1-e^{-\lambda(N+1)T_{listen}}}{N+1} + \Pi_{\texttt{slp}}\left(1 - e^{-\lambda T_{sleep}}\right)$. All curves show a behavior similar to that of the throughput, as when the nodes are either busy or in backoff the number of accepted packets begins to decrease. In simulations this tends to happen sooner.

To support the observations above, in Figure 3.8 we show the average probability of receiving a CTS in response to any RTS sent, representing the availability of at least one relay that is awake and idle. As shown in subsection 3.3.5, the analytically derived probability of such an event does not depend on the offered traffic λ. In simulation, this is true as long as traffic is sufficiently low, such that no node is held by excessive backoff events,

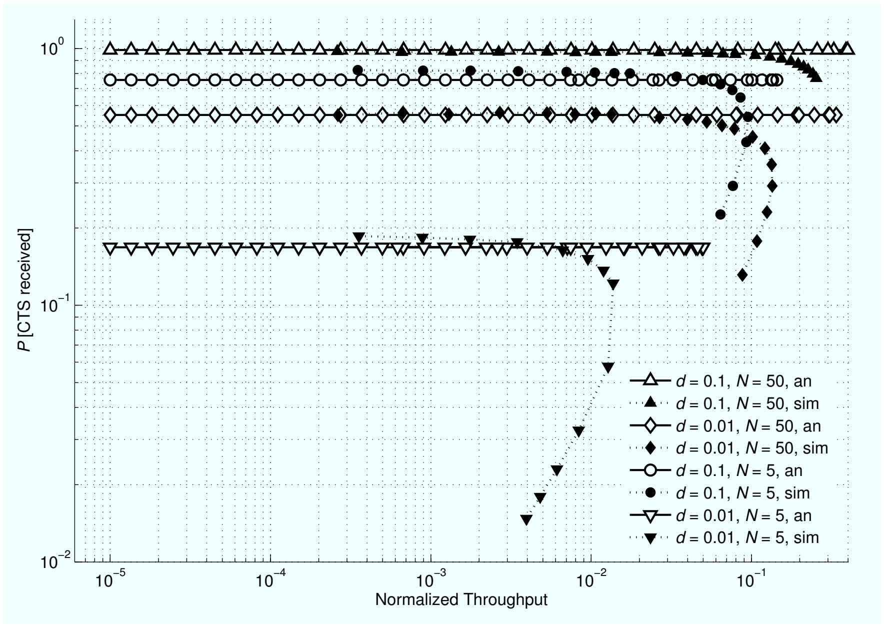


Figure 3.8. *Probability of finding at least one relay as a function of throughput.*

which prevent it from relaying other nodes' packets. As traffic increases, this impairment becomes more and more likely, so that the average number of available relays decreases and a progressively higher overall congestion level is reached. The speed at which deadlocks occur depends on the average network degree N and the duty cycle d. Notice once again that Figure 3.8 is plotted as a function of throughput, which is in turn a concave function of traffic. For this reason, all curves tend to fold back.

Further insight on these effects is given in Figures 3.9 and 3.10, depicting the average probability that a node wins and is elected as a relay given that it was contending for this role, and the probability that the radio channel is sensed idle, both as a function of throughput as before. While in analysis the first one (Figure 3.9) is only dependent on the average degree, on duty cycle (through γ_is) and on node positioning, in simulation it also shows a dependence on traffic, since probabilities first increase and then fold back. The main reason of this behavior is that, at high traffic, more nodes have their own data communications to carry on and are unable to accept relay requests. This unavailability also explains the higher overall chance that a node is elected as a forwarder, given that it was available and participated in the contention. Since nodes with data to send generate a denser signaling traffic, the probability of sensing a free channel upon one node's own transmission correspondingly folds back, as seen from Figure 3.10.

From these results, we can conclude that GeRaF's performance is strongly dependent on the number of available relays per transmission attempt. Equation (3.1) suggests that this number only depends on the product Nd, once the area of the forwarding regions, the length of the signaling packets, and the duration of the sleep interval are fixed. Therefore, Nd represents a rough figure of merit for evaluating the protocol performance. In Figure 3.6

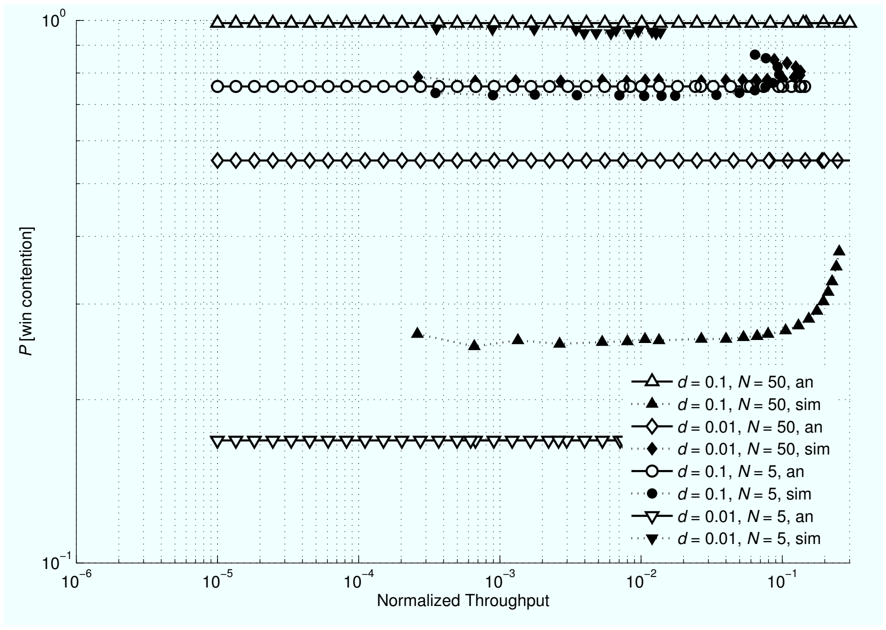


Figure 3.9. *Probability of winning a contention as a function of throughput.*

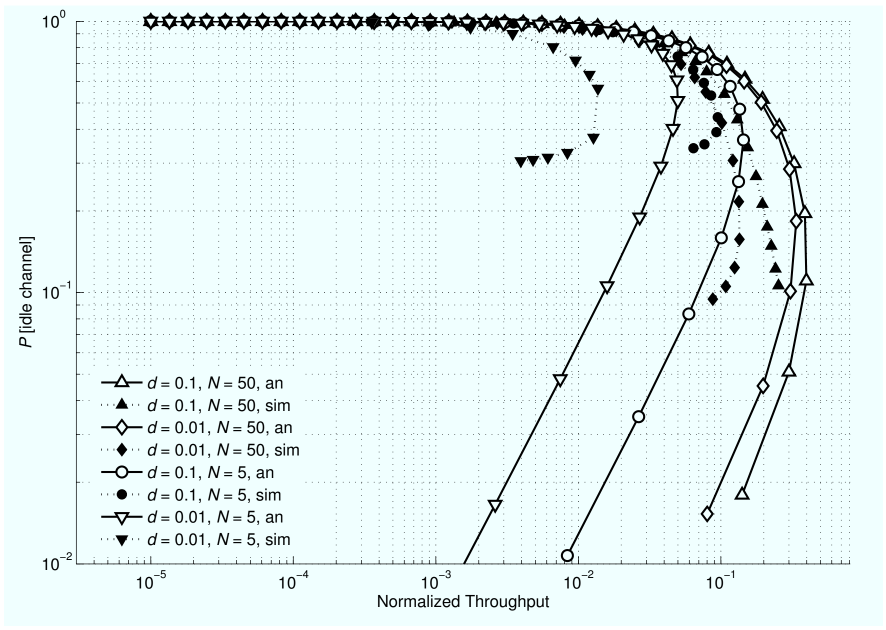


Figure 3.10. *Probability of sensing an idle channel as a function of throughput.*

such effect is clear, with the further remark that $(N, d) = (5, 0.1)$ performs slightly better than $(N, d) = (50, 0.01)$, because in the latter case nodes tend to spend more time in backoff due to a failed relay search. For example, consider a low traffic scenario. Recall that the durations of both the awake time and the relay search phase are fixed. Hence, if relays wake up more often, they are more likely to hear RTSs and contend, thereby handling more traffic and achieving a higher throughput. This is exactly what happens at higher d. Note also

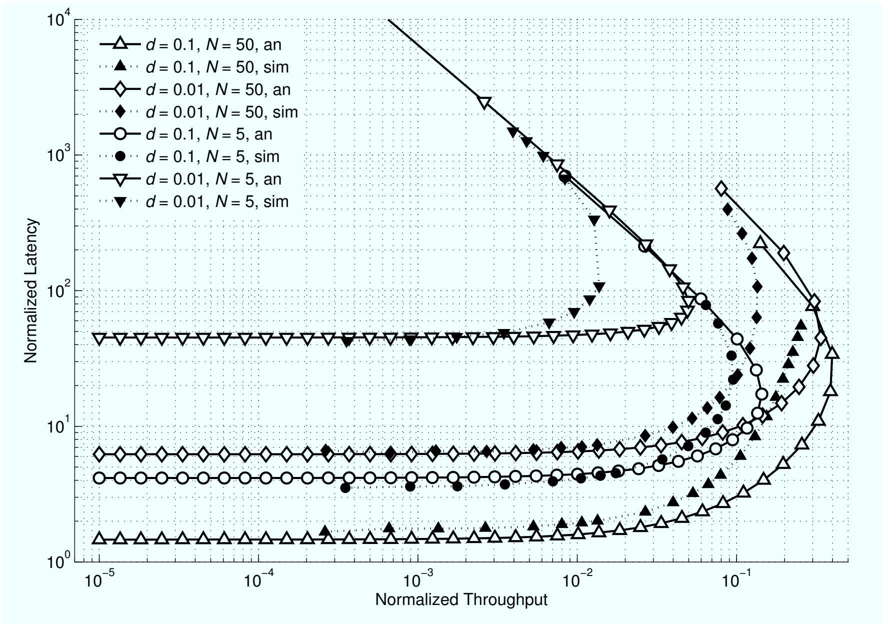


Figure 3.11. *Normalized latency as a function of throughput.*

that this effect dominates the slightly longer relay search phase used in the $(50, 0.01)$ case (as imposed by the greater N_r, see Section 3.4.3). In general, however, a greater N allows to reach a higher maximum throughput, as the bigger number of relays within coverage yields more traffic transport capability.

In the following, we focus on the evaluation of GeRaF under the point of view of energy expenditure and channel access latency, giving further insights on the difference with the version previously proposed in [3]. As mentioned before, latency is a measure of how easy it is to gain channel access, and is defined as the average time expected to elapse before successful data forwarding. Thus, it accounts for any busy channel condition, *e.g.* due to neighboring congestion, or to the absence of free relays.

In Figure 3.11, we display GeRaF's latency performance against throughput. Curves are shown for all the considered values of N and d. As expected, the analysis can predict latency quite faithfully at both low and high traffic loads. However, when congestion builds up, the analysis loses some accuracy in predicting the reasons of relay unavailability seen before. Specifically, some neighbors are attempting to forward other packets, which keeps them busy (or in backoff). This is in contrast with the assumption that N nodes are available (regardless of whether awake or asleep), and explains the lower accuracy of the analysis. At the highest traffic, when there is a very high probability of sensing a busy channel, both the analytically predicted and the simulated latencies are mostly affected by the average time spent in backoff, and thus they become aligned again. Latency performance is better, in general, for higher duty cycles and average degrees. As a secondary effect, decreasing d from 0.1 to 0.01 and increasing N from 5 to 50 slightly worsens the average latency. The reason, here, is the same that explains the lower throughput in this setting.

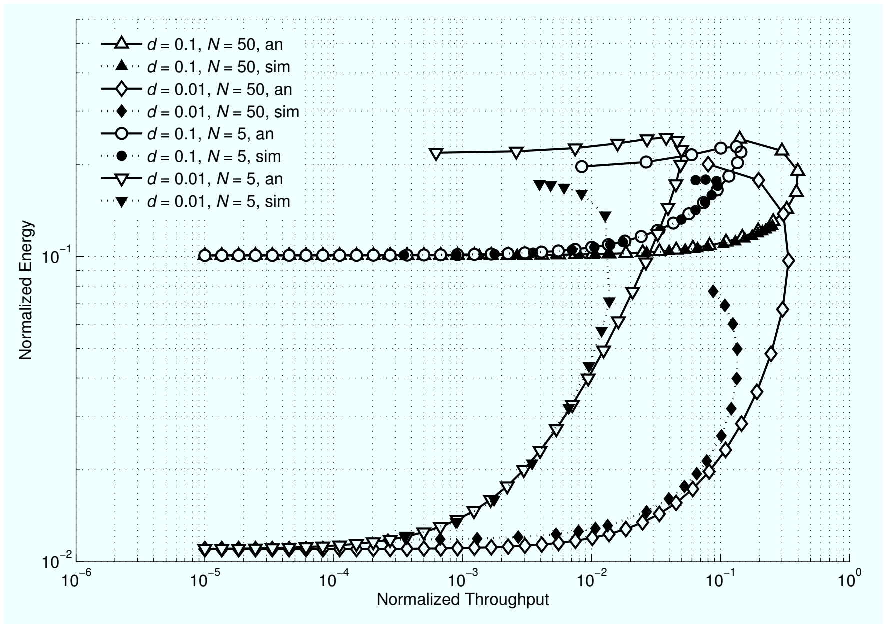


Figure 3.12. *Normalized energy as a function of throughput.*

Figure 3.12 shows the average energy consumption (the total energy consumed by a node) normalized to the energy that the node would spend if its radio were always on. In this case, at low traffic a node tightly follows the on-off schedule dictated by its duty cycle, and the analysis closely matches the simulation. As the traffic increases, and hence the throughput approaches its maximum, the energy required to sustain the network operations is progressively greater. The steeper increase is observed in the presence of more critical network conditions, *i.e.*, for lower values of N and d. The simulation results match those of the analysis in any of the considered configurations. However, for the cited reasons, the nodes back off more often in simulations, and this causes a mismatch at medium traffic values. At high traffic, instead the analysis tends to get closer to simulations again, as the energy consumption is dominated by time spent in backoff.

As a final remark, in Figure 3.13 we show the mutual relation between energy consumption and latency performance for fixed throughput (as defined in Section 3.4.4), by varying d in $[10^{-4}, 1]$. The curves are spanned from the bottom-right to the top-left corner for increasing d. We have included the version of GeRaF presented here and the initially proposed version [3], where awaking nodes cannot participate in ongoing contentions. The latter has been dubbed Noaw, for "no awaking nodes." The curves suggest that our GeRaF offers a better tradeoff between energy and latency with respect to GeRaF-Noaw, for the two values of the average network degree N we considered. This demonstrates the effectiveness and importance of including awakened nodes into contentions. The accordance between analytical and simulation results is very good at medium to high duty cycle, yielding a good match in the significant part of the energy-latency tradeoff, *i.e.*, in the Pareto-optimal region, where the curves have negative slope. There is, however, a slight discrepancy for very low

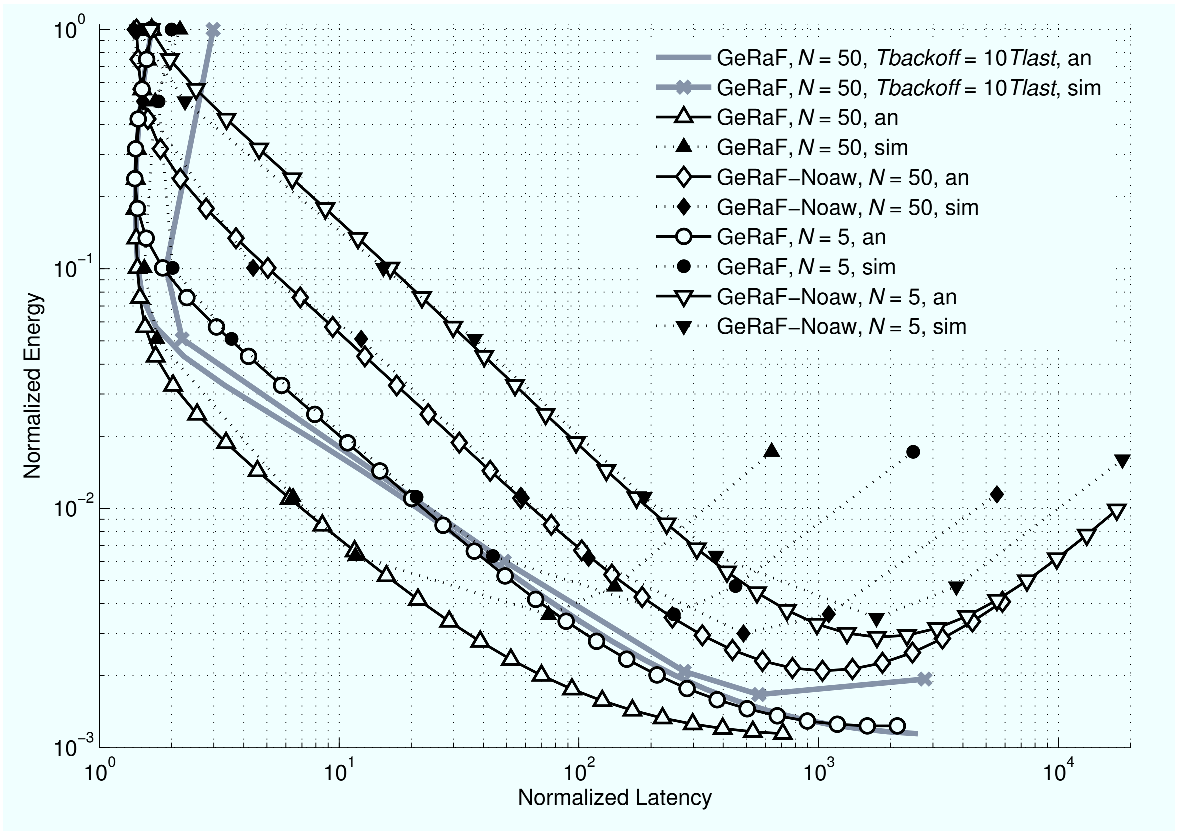


Figure 3.13. *Tradeoff between energy and latency for GeRaF and GeRaF-Noaw at* $\lambda = 10^{-5}$.

values of d. The reason is the same explained before: congestion is reproduced better in simulations than in the analysis, because the latter assumes that the neighbors of the transmitter are free to contend, once channel access is granted. On the contrary, at low duty cycle, it is very likely that nodes are busy in sending their own packets. As a further proof of this fact, we show in Figure 3.13 the performance of our version of GeRaF with $N = 50$ nodes and $d = 0.1$, where $T_{backoff} = 10T_{last}$. Longer backoffs worsen the average latency and throughput (nodes handle the same packets for a longer time and cannot accept new ones) but relieves the network by making transmission attempts less persistent. Hence, the network is less prone to congestion (curves fold later for decreasing d), and as a byproduct the match between analysis and simulation improves. Whether to prefer a lower latency or a better performance at low duty cycles is ultimately a design choice.

3.4.5 Preliminary observations

GeRaF is a lightweight protocol that bears the advantages of geographic routing, and exploits the node redundancy inherent in any sensor network in an effective manner for finding a node offering maximum advancement. From the study carried out here, we can conclude that if properly tuned with a more efficient contention scheme and correctly designed timings, it can offer very good performance in terms of latency and energy saving.

Moreover, the analytical model shows very good accordance to the simulations, and can thus be employed to determine the best parameters (such as duty cycle and, if controllable, node density) to achieve the desired performance depending on the type of traffic to support.

Since the ns2 simulator exhibits a good accordance to analysis we will use it in the following to reproduce the behavior of a network with multihop routing. This study is necessary in order to ensure that the performance evaluation carried out up to this point is still valid in a more realistic setting. In fact, a true converge-casting would require the farthest nodes to report back to the sink as well. Hence, the others sensors on the path toward the sink should help relaying the packets coming from farther areas. Ensuring that little data is lost on the path requires to implement queues on the sensors, so that they can manage the data backlog when needed. For example, they could temporarily store newly generated data while handling the transmission of the neighbors' packets.

3.5 Multihop GeRaF evaluation

The purpose of this Section is to evaluate GeRaF's multihop performance under different node density settings. We are mainly interested in the packet delivery ratio, energy consumption, number of hops and on the causes of packet losses. A joint analysis of these data allows to understand where GeRaF can be improved, and pave the way for the subsequent discussion about the ALBA protocol, that will be presented later.

The simulation scenario for this multihop evaluation is similar to the one used for single-hop performance measures. More specifically, we have considered random deployments and performed simulations according to different densities and also varied some parameters such as the density of the network and the transmit coverage range. The density has been tuned to yield a total of 5 to 50 neighbors to a node, on average, whereas the coverage range has been tuned to yield more hops when necessary. Unless otherwise specified, it has been fixed to the same values used for the one-hop evaluation, *i.e.*, $40\,\mathrm{m}$. Unlike previous results, we have fixed the duty cycle to 0.1 here. Energy consumption has been evaluated according to the first-order energy model presented in [20]. In this model, the energy consumed per transmitted bit is assumed to be constant and equal to E_{RX}, whereas the transmit energy depends on the coverage radius r according to

$$
\begin{aligned}
E_{TX}(r) &= E_{TX_e} + E_{TX_a}(r)\,, && (3.26)\\
E_{TX_a}(r) &= \varepsilon_a \cdot r^2\,. && (3.27)
\end{aligned}
$$

In (3.26), E_{TX_e} accounts for the energy drained by the transmitter electronic circuitry (and is set equal to E_{RX}), whereas $E_{TX_a}(r)$ models the energy required to cover the transmission range r. The energy cost when asleep is a very low non-zero value, equal to $1/1000$ of the receive energy. According to this model, $E_{TX_a}(r)$ and E_{TX_e} have similar values when $r = 22.5\,\mathrm{m}$, after which $E_{TX_a}(r)$ becomes the dominant factor. In the following evaluation,

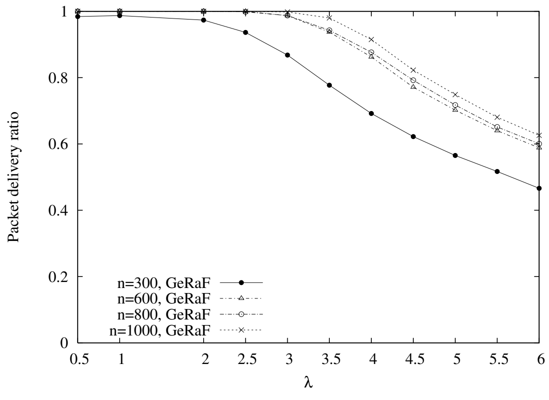


Figure 3.14. *Packet delivery ratio, GeRaF.*

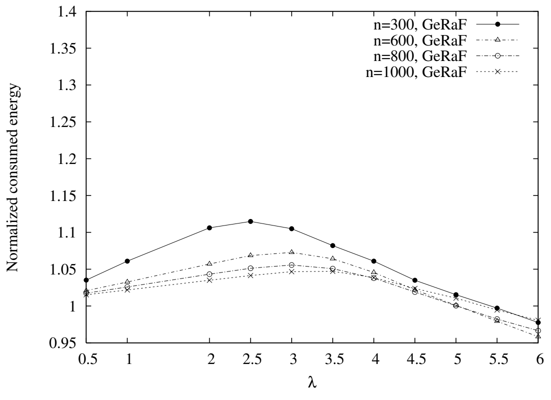


Figure 3.15. *Node energy consumption, GeRaF.*

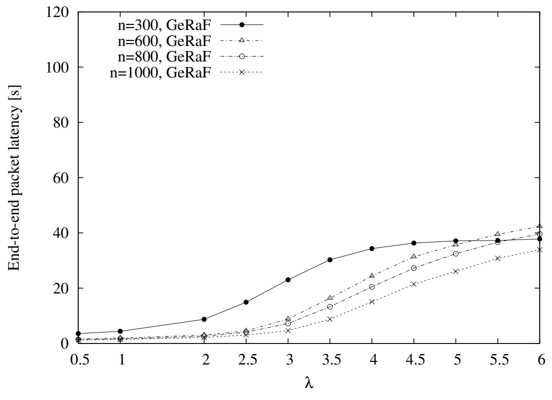


Figure 3.16. *End-to-end packet latency, GeRaF.*

the energy consumed by nodes has been normalized to the consumption incurred if the duty cycle was strictly followed, *i.e.*, in the absence of traffic.

The results presented hereon have been obtained by averaging over 100 different connected topologies per density value. All topologies have been run for 20,000 seconds.

Firstly, let us focus on Figures 3.14, 3.15 and 3.16, that depict the delivery ratio, the energy consumption and the end-to-end delivery delay. The packet delivery ratio, in particular, is defined as the ratio of the successfully delivered packet to all packets managed by the nodes (that is, not discarded due to a full queue). The main reason why a packet can be dropped is that the maximum number of attempts to forward it has been reached. In turn, this happens due to repeated backoff events caused by transmission errors, to the inability to find a relay or, in a negligibly small number of cases, to unresolvable collision events. Note that, even if a proper channel sensing time has been designed to avoid most collisions, *i*) some second-order events[7] may take place that cause collisions anyways (results indicate that this event

[7]For example, consider two different senders placed far apart, but that have relays in range of each other. If the two handshakes takes place simultaneously, there is a non-zero probability that some messages collide, perhaps corrupting the data or `ACK` message. Since there is no sensing at the receiver side, this event cannot be

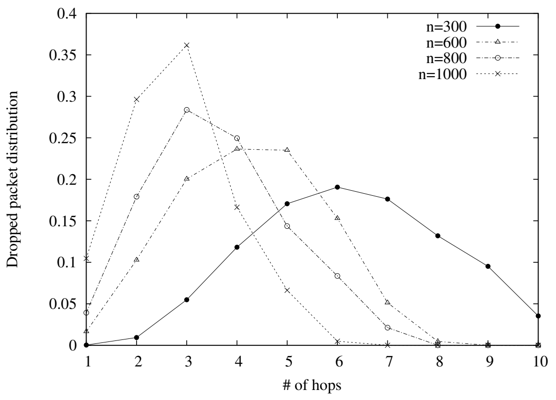


Figure 3.17. *Hop-wise probability of packet drop, GeRaF, $\lambda = 6$.*

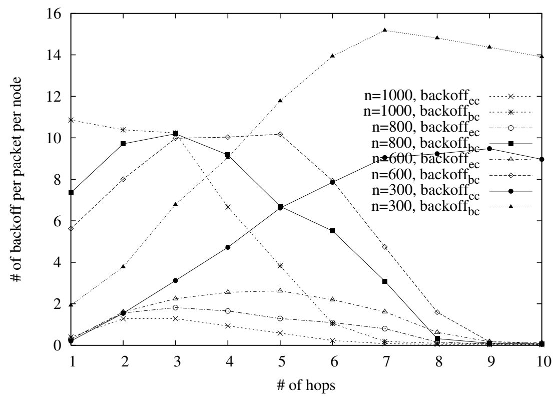


Figure 3.18. *Number of backoffs per packet per node, GeRaF, $\lambda = 6$.*

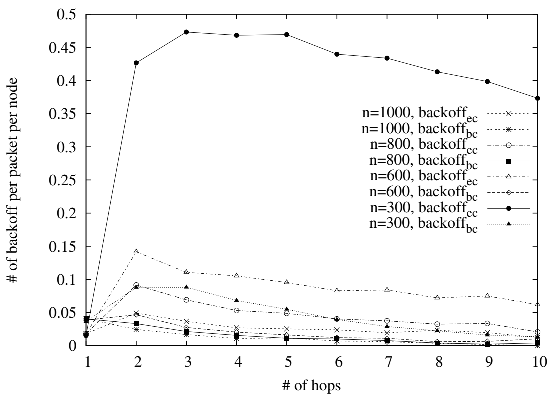


Figure 3.19. *Number of backoffs per packet per node, GeRaF, $\lambda = 0.5$.*

has a very low probability nevertheless), and *ii*) the protocol itself has a vulnerability time equal to T_{sens}, as is for all CSMA protocols. In particular, at a low density (*i.e.*, when the number of nodes is $n = 300$) the most likely cause of errors is a bottleneck effect at some point in the network. Given the network parameters, 300 nodes yield an average of 1.5 awake neighbors per attempt performed, of which less than one half offers non-negative advancement. That is a critical value because, in addition, there is no guarantee that those neighbors will be free to accept the sender's request. If the node cannot forward the packet after a (albeit high) number of attempts, it will be forced to drop the packet. Higher densities ($n = 600, 800, 1000$) suffer less from this event, as confirmed by their almost equivalent delivery ratio performance. Note also, that all curves follow the same trend. This confirms that the unavailability of relays is the most important cause of errors. In other words, even if higher densities tend to generate a little more collisions, the overall impact of this event is negligible.

completely avoided.

The normalized energy consumption (Figure 3.15) reflects these observations as well. All curves show that initially the network experiences an overall energy increase that is proportional to the offered traffic. Depending on the transport capability of the network, *i.e.*, on the density of relays, this increase is more or less apparent, being steeper for lower densities. At some point that again depends on density, the energy consumption reaches a maximum before beginning a monotonic decrease phase. As for delivery ratio, this is due to increasing backoff events caused by the impossibility to find a relay. The energy decrease shows that most of the network is actually in backoff at high traffic, with small differences depending on density.

Correspondingly, the end-to-end latency shows a sigmoid behavior, whereby it experiences a small increase when the network is still not too loaded, and grows more substantially afterwards. Perhaps counter-intuitively, at high traffic the latency settles on a more or less constant value. This behavior is actually totally reasonable, as at high traffic the network ends to shrink, and packets are generally unable to travel many hops for reaching the sink. Since the only packets that make it come in most cases from close nodes, this limits the average latency to an almost constant value.

In order to elaborate more on this analysis, consider Figures 3.17, 3.18 and 3.19, that depict the average probability of dropping a packet and the average backoff probability on a per-hop basis. These pictures can show more effectively where the forwarding problems tend to take place depending on density and traffic. Only low ($\lambda = 0.5$) and high ($\lambda = 6$) traffic are considered in this case. Figure 3.17 shows that the network gets shrank as density decreases, for high traffic. Recall that at each hop, the current sender performs a maximum of 50 forwarding attempts before dropping a packet, and that only the packets that actually enter the queue are counted for calculating the drop probability. With a lower number of nodes, the persistent forwarding requests due to fast packet generation steer the network to a congested situation. At low density, most dropping events take place far from the sink, because all traffic is converging toward a single point, and it gets increasingly more difficult for a node to let its own packets through. This is due to almost all relay having their own traffic to forward and being rarely able to volunteer as a next hop. Conversely, at higher density, the nodes tend to easily route the packets until 2 to 4 hops before reaching the sink, where most of the droppings take place. The congestion due to the convergence of traffic routes is again the reason behind losses.

In the same traffic situation, Figure 3.18 shows the distribution of the per-hop backoff probability. Backoff events are identified by means of their cause, *i.e.* a busy channel (BC) or an empty cycle (EC) during relay search. Focus for now on nodes 6 to 10 hops from the sink. At higher load the network gets congested, making it progressively harder to find relays, so that the number of EC backoffs increases from 0.4 at $\lambda = 5$ to 14–15 at $\lambda = 6$. Because of that, the traffic conveyed to the nodes closer to the sink tends to decrease. Along with the fact that the sink is always available and that 1-hop nodes do not generate traffic, this explains why performance improves hop by hop. At $n = 600, 800, 1000$, the greater average degree allows to advance packets up to 2–5 hops from the sink without significant problems (see Figure 3.17). The main reason for discarding packets is severe congestion (nodes are not able to get channel access for long times), both at high and low density. The case $n = 300$ shows a different trend. The node degree here is very low, therefore, it is less likely to find eligible relays. This turns into multiple backoffs before a successful forwarding. At higher traffic, transmissions are further impaired by the shorter time (on average) spent awake and idle by nodes, due to both duty cycles and backoff. This results in packet dropping, even close to the sources. Also, packet dropping is due both to early congestion (relay selection is performed multiple times, traffic is shared among a few nodes) and the difficulty to find an awake, available relay. Note that, in these scenarios, the nodes closer to the source benefit from the reduced network load resulting from early packet dropping (their number of backoffs per packets is lower). For comparison, Figure 3.19 presents the same backoff analysis for low traffic, where backoffs occur in general much more rarely. In this case, the only critical situation occurs at $n = 300$, where the low density is the cause of the low probability to find free relays. In fact, most backoff events in this case are due to ECs.

3.5.1 Observations

The evaluation carried out up to this point through analysis and simulation has pointed out that GeRaF behaves interestingly well in terms of latency and energy consumption, besides offering a good data transport capability. It has also shed light on its major drawback, that is, the problems related to relay election, especially at critical node density and traffic values. The choice of a relay, in particular, is critical to the network performance. If a next hop is chosen that is experiencing congestion and thus can hardly forward packets, the risk is to worsen the congestion, perhaps involving the current sender. This moves the congestion back to the nodes farther from the sink. All in all, this is the problem to solve to improve network performance. For example, augmenting GeRaF with normal duty cycling during backoffs (so that the node can receive while it refrains from transmitting) or bursty transmissions (to make more efficient use of the channel once it has been acquired) can partially relieve the problem, but does not solve it effectively. Such optimizations have the only consequence of increasing up the lowest traffic value where congestion begins to build up.

These reasons motivated us toward the design of a new contention scheme that identifies a more convenient relay from the point of view of traffic, instead of seeking only advancement. Along with some further optimization, the protocol improves substantially over GeRaF and will be the starting point of further discussions. The next Section introduces and describes this new approach.

3.6 ALBA — a New Scheme for Geographic Forwarding

ALBA stands for Adaptive Load–Balanced Algorithm, and is a cross–layer solution for converge-casting in WSNs. Like GeRaF, it integrates awake/asleep schedules, MAC and routing. In addition, it also includes network congestion control and a technique to deal with dead ends. Wake-up schedules are independent, with fixed duty cycle d.

ALBA also tries to balance network traffic among nodes *before* favoring the relays that yield a better geographic advancement. As observed before, there is little point in concentrating only on the progress toward the sink without keeping an eye open for traffic load distribution. If a node gets overwhelmed with forwarding requests, and/or is unable to quickly advance packets, it soon gets congested and cannot operate as relay for its neighbors. In turn, this decreases the number of eligible forwarders.[8] ALBA makes joint use of two ideas for alleviating in-network congestion and increasing the packet delivery ratio. The first idea relies on the observation that in typical WSNs scenarios, where nodes follow a low duty cycle, it is challenging to find eligible forwarders, making relay selection a time-consuming operation. Therefore, ALBA shares the overhead of relay selection among multiple packets through bursty transmissions. Once a relay is found, a burst of packets, whose length is tuned depending on local forwarding capabilities, are transmitted back-to-back to the relay. This is beneficial in scenarios with few eligible forwarders as well as in high traffic scenarios (where nodes tend to have longer queues). The second idea concerns the relay selection criterion. ALBA chooses a relay among the eligible forwarders offering good performance history and good progress. Similarly to GeRaF, the forwarding area is divided into N_r regions ordered according to a decreasing geographic progress criterion. Any node in region i is closer to the sink than all nodes in regions $i+1, i+2, \ldots$. Every node is characterized by a *geographic priority index* (GPI) equal to the priority region index. The information about the GPI is separately computed by each node based on information on the positions of the sender (contained in the signaling packets), the node itself, and the sink (ALBA nodes must therefore be location-aware). The second parameter is called the *queue priority index* (QPI) and is calculated as follows. Suppose that the requested number of packets to be transmitted in a burst is N_B, and that the queue occupancy of the relay is Q. Suppose also that the relay keeps a weighed average M of the number of packets it was able to transmit back-to-back without errors during the past forwarding attempts.[9] The QPI is then defined as

$$\min\left\{\lceil (Q+N_B)/M\rceil, N_q\right\}, \tag{3.28}$$

where N_q is an upper bound to the number of assignable QPIs, and is set to be equal to N_r without loss of generality. The rationale behind the QPI concept is that the sender will be

[8] As before, these are defined as the active neighbors of the sender providing a positive advancement toward the sink. Later, we will relax this concept.

[9] This mechanism is easier if the receiver separately `ACK`s every packet it receives during the transmission of a burst. Here we consider this specific implementation.

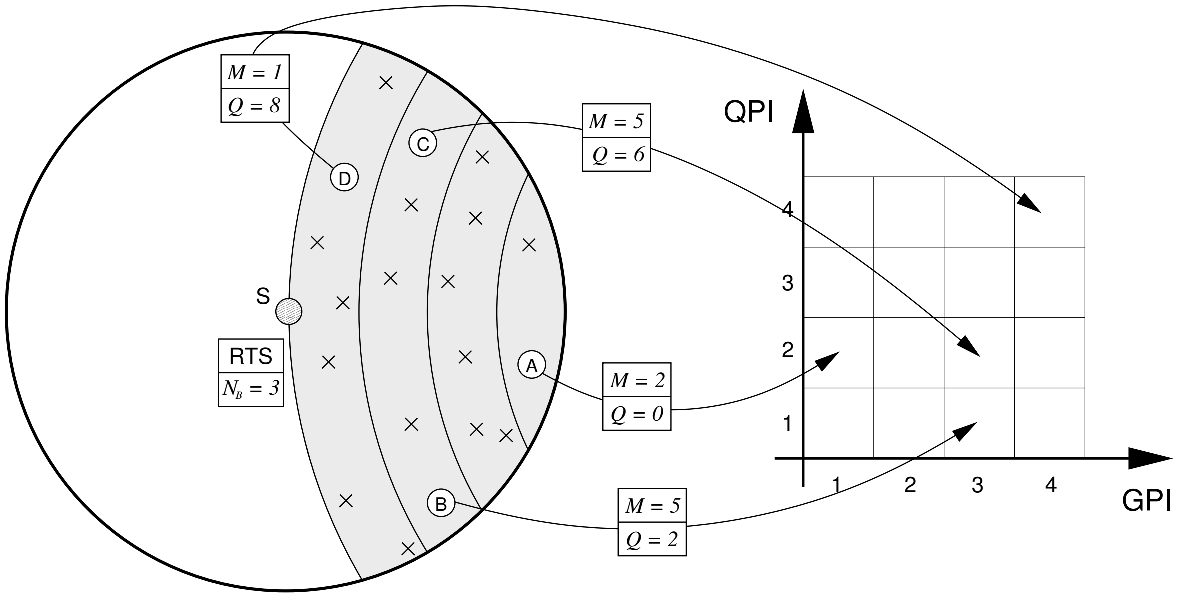


Figure 3.20. *Example of computation of QPIs and GPIs by relays upon* RTS *reception.*

able to discriminate between nodes that can easily forward the data and nodes that may be experiencing either congestion or bad channel conditions. The QPI captures the estimated number of burst transmissions needed before the new data can be relayed further. As relay selection is the most time consuming operation in contention-based multi-hop forwarding, the selection of relays with low QPI decreases end-to-end latency. If a node finds it difficult to send out packet bursts (small M), or has a long queue (large Q) then the channel is likely to be intensively used somewhere around the relay. This makes the node a bad candidate. On the contrary, if a node has a very low queue, or it is able to issue longer packet bursts (large M), then it will be chosen more frequently, because it is expected it can handle a larger amount of packets.

An example of QPI and GPI assignment is provided in Figure 3.20. The sender S is represented by a gray circle, crosses denote asleep neighbors and white circles awake ones, *i.e.*, the only nodes that are available for relaying at the time the first query is issued to the neighbors. The forwarding area is colored light gray, and the regions are delimited by arcs centered on the sink (supposed to be quite far away and not shown). S requests to send a burst of $N_B = 3$ packets. Similarly, nodes B and C both have $M = 5$, but B has a smaller queue. Therefore, B's QPI is 1, and C's is 2. Similarly to GeRaF, in ALBA the sender first performs a channel sense, that makes colliding with ongoing packet transmissions less likely. Then, it transmits an RTS asking its eligible forwarders to compute their QPI and GPI. The RTS contains the location of the sender, the location of the sink, and the requested length of the data burst to be sent, N_B. Based on the knowledge of its position and on the information included in the RTS packet, each forwarder computes its own GPI. By using the value of N_B, its queue size Q and the expected length M of a burst that it can successfully transmit, the node also calculates its QPI. In Figure 3.20, node A has an empty queue and

$M = 2$. Therefore, its QPI is 2. Node D also has a low $M = 1$ but its backlog is longer ($Q = 8$), which results in its QPI being 4 (the maximum allowed value for the QPI). Once the QPI and the GPI have been computed, nodes can respond to the RTS using the following rules.

The contention is arbitrated by first looking for the best QPI. Thus, only nodes with a QPI equal to 1 are allowed to answer the first RTS with a CTS message. As in GeRaF, three events may take place here. If only one node answers at this time, it is elected as the relay and the search phase ends. Otherwise, either no nodes, or multiple nodes answer the RTS. In the last case, a collision occurs. If no nodes respond, a second RTS is sent to query the nodes with QPI equal to 1 and 2, thereby considering a larger set of nodes that includes the neighbors addressed in the previous phase. If no node replies, the sender keeps expanding the target set for of its search. At the third attempt, the QPIs 1, 2 and 3 are looked for, and so forth, until at the N_rth attempt all QPIs are considered. If the search terminates unsuccessfully, with no relay being found, the sender backs off and reschedules a later attempt.

The case of a collision due to multiple CTS responses is handled slightly differently with respect to GeRaF. The occurrence of a collision indicates that there are nodes with the same QPI, so that the sender can choose the one that provides the best advancement. To this end, a GPI search phase is needed. A modified RTS packet is thus sent. The header of this RTS specifies that only nodes with the selected QPI and GPI equal to 1 may answer. The same answering procedure used for QPI is used here (this time based on the GPI index). If a collision occurs, then multiple nodes have the same QPI and GPI, and one is selected through a splitting tree algorithm. Other RTSs reporting the same QPI and GPI of the previous one are sent, and each one forces nodes to choose with probability 0.5 whether to stay silent or to answer with a CTS again. This process goes on until a single relay responds and is eventually chosen.

This selection process can fail in two cases, namely if no node with any QPI is found (as described above) or if the contention among nodes with the same QPI and GPI is prolonged beyond a maximum number of attempts N_{Att}. Both situations lead the sender to a backoff. Like in GeRaF, the maximum number of backoffs before a packet is dropped is fixed to, say, N_{Boff}. To exemplify the selection procedure let us go back again to Figure 3.20. The only available relay with QPI 1 after the first RTS is node B, which will send the CTS alone and hence be selected as a relay. If B were asleep, only A, C and D would be available. In this case, no node with QPI 1 exists, so that the first RTS is not answered by anyone. Both A and C will answer the second RTS, since both have QPI 2. The second phase (best GPI search) is then started, which terminates with the selection of node A, whose GPI is 1.

Once a relay is selected, the whole burst of N_B data packets is sent. The nodes that lost the contention overhear the data transmission and understand that they have not been selected as relays. Hence, they go back to sleep. All data packets are individually acknowledged. This allows to stop transmitting the burst if an ACK is missing (because of channel errors, collisions, etc.). Incorrect packets in the burst are rescheduled for a later attempt after a backoff period. During the backoff, the nodes do not go to sleep, but instead follow their normal duty cycle, and may respond to relaying requests if needed. Regardless of the final status of the transmission and the presence or absence of errors, the sender updates its expected maximum burst length M, taking into account the number of correct packets (if errors occurred), or by optimistically assuming that a burst of M_B packets was received correctly (if all sent packets were received with no errors).

Recall the observation that significant performance improvements can be obtained by allowing eligible forwarders to join a relay selection phase that has already started (see also the comparison between GeRaF and GeRaF-Noaw in Figure 3.13). In ALBA, we decided to incorporate this optimization. Therefore, upon waking up, nodes can enter the QPI search phase and can answer an `RTS` packet with a `CTS` packet, provided that their QPI index is lower than or equal to the one that is currently searched for. When a QPI region is found with one or more eligible forwarders in it, and the best GPI search starts, the set of eligible forwarders is frozen. No node that wakes up after this time can enter the contention. This choice has been made to favor a fast relay selection once a region has been found with active neighbors in it.

3.6.1 ALBA multihop performance evaluation

To discuss the effectiveness of the new relay search scheme and of the other optimizations introduced in ALBA (duty cycling during backoff and bursty packet transmissions), we carry out here the same analysis performed for GeRaF in Section 3.5. The considered scenario is the same described before, therefore we will omit its description here for brevity.

Figures 3.21 to 3.23 display the packet delivery ratio, the average normalized energy consumption and the end-to-end packet latency for ALBA. Figure 3.22 shows that duty cycling is the main source of energy consumption at low traffic (in all the curves the normalized energy consumption has values close to 1). That was true for GeRaF as well. However, in GeRaF the normalized energy first increases and then starts decreasing, whereas in ALBA it steadily increases with the traffic load. This is due to the different relay search scheme, that favors nodes with a better traffic carrying capability. Secondarily, ALBA nodes continue to follow the regular duty cycle and keep participating in contentions while waiting for the end of a backoff interval. This also improves the chance to find a suitable receiver. On the contrary, whenever a GeRaF node is handling a packet it stops volunteering as a relay for its neighbors. Even incorporating the same feature in GeRaF does not yield the same benefit (see the discussion in Section 3.5.1). The experiments show that with ALBA a node backs off on average only 2.8 (4.15) at $n = 300$ and $\lambda = 4$ $(\lambda = 6)$. The lower the number of potential relays of a node (*i.e.*, the lower the density) the more apparent this effect is. In fact, not only ALBA relies on a higher average number of potential relays, but it is also able to forward traffic more effectively at high load. In addition, the possibility to transmit bursts of packets back-to-back ensures a better use of the time and overhead needed to search for a relay. As

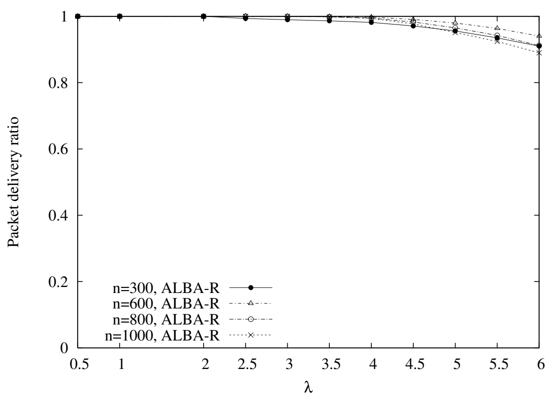


Figure 3.21. *Overall packet delivery ratio, ALBA.*

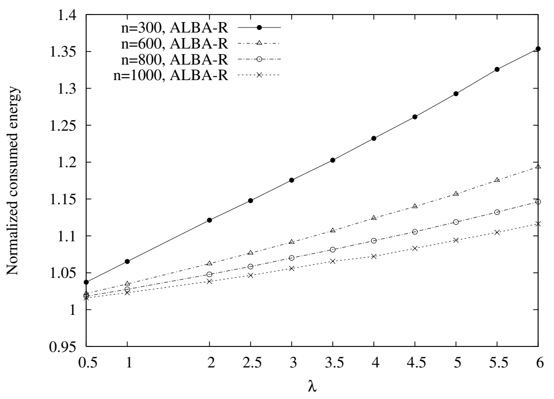


Figure 3.22. *Node energy consumption, ALBA.*

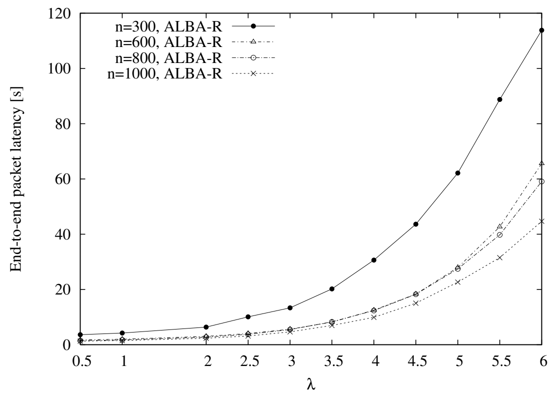


Figure 3.23. *End-to-end packet latency, ALBA.*

a consequence ALBA can successfully direct toward the sink a higher number of packets, which however imposes a proportionally higher energy consumption.

Following the normal duty cycle when in backoff, ALBA forces nodes to stay awake longer with increasing traffic in order to complete relay selections that might have started. The higher the traffic load the higher the normalized energy consumption. In GeRaF, instead, nodes tend to be in backoff (asleep) for long times at high load. Figure 3.21 shows the packet delivery ratio for ALBA and confirms the above observations, especially as compared to Figure 3.14. Even at high packet generation rates ($\lambda = 4, 6$) ALBA correctly delivers more than 90% of the packets. ALBA's QPI-based relay selection, the back-to-back packet transmission scheme, and the higher number of potential relays prove to be very effective in supporting higher network loads. At the same loads the performance of GeRaF is significantly degraded, with a delivery ratio of 70-90% when $\lambda = 4$, and 50-60% when $\lambda = 6$, depending on the node density.

The higher packet delivery ratio justifies the higher end-to-end packet latency of ALBA. As shown by Figures 3.23 and 3.16, in case of low traffic ALBA and GeRaF show comparable latencies. ALBA could potentially result into longer contentions (up to 8 rounds instead of the 4 required by GeRaF). However both contentions are very fast and ALBA typically finds a relay much earlier than in the worst case, resulting only in a couple of extra RTS–CTS exchanges with respect to GeRaF. On the other hand, ALBA incurs a significantly higher latency than GeRaF for high network traffic. The increase is particularly significant for low network densities, due to the higher traffic delivered to the sink (which results into higher queuing delays) and the QPI-based relay selection mechanisms. This, at high loads, causes a slight route length increase (10-15% with respect to GeRaF). However, this is an acceptable disadvantage, given the major improvements yielded by ALBA's features. In particular, it is worth noticing that the size of a correctly received packet burst grows, on average, from 1.1 packets ($\lambda = 2$) to 1.6 ($\lambda = 4$), up to 2.6 ($\lambda = 6$). This means that ALBA can effectively distribute traffic even at high loads, where more nodes are in backoff than at low traffic. Most of these packets reach the sink correctly.

Figures 3.24 to 3.26 show the distribution of the probability to drop a packet at certain hop, as well as the fraction of backoffs that take place in the network on a per hop basis. Providing ALBA with the load-balancing relay selection, duty cycling during backoff and bursty transmissions shows one more advantage. Only few packets are dropped, and only at very high load ($\lambda = 6$). Moreover, when $n = 600, 800, 1000$ the dropped packets are discarded mostly when they are 1 or 2 hops from the sink. There, all packets begins to converge over the same zone, and load balancing can only partially alleviate congestion. Backoff and packet dropping are mostly due to difficulties in channel access, *i.e.*, busy cycles (BCs). As density decreases, traffic flows through fewer neighbors, leading to a more severe congestion which extends also to nodes far away from the sink. When $n = 300$ the highest packet dropping is observed at nodes 4 to 5 hops away from the sink.

As a final note we compute the ratio between the average time an ALBA node uses the channel for relay selection, for transmitting data packets and receiving their ACKs, and the minimum time needed for relay selection and data transmission (*i.e.*, the time to perform carrier sense, transmit a RTS, receive the CTS, send the packet and receive its ACK). This metric, denoted as normalized overhead per packet in the following, is higher in protocols whose relay selection scheme requires heavy traffic exchange (high overhead). Figure 3.27 depicts the normalized overhead per packet when n varies from 600 to 1000 and λ varies between 0.5 and 6. We observe that the per-packet overhead required by ALBA is very limited. When $\lambda = 0.5$ it is only 20% greater than the minimum, whereas it expectedly grows for higher loads. Nonetheless, the increase is fairly smooth, thanks to the light relay selection scheme adopted (only a few packets need to be exchanged for each contention), and also to the fact that the overhead for relay selection is shared among multiple packets (those in a burst). When $\lambda = 4$ ($\lambda = 6$) the normalized overhead is only 1.21 (1.37) when $n = 1000$.

GeRaF always shows worse performance than ALBA, for the relay selection difficulties mentioned before. This results in a significant increase in the normalized overhead, which

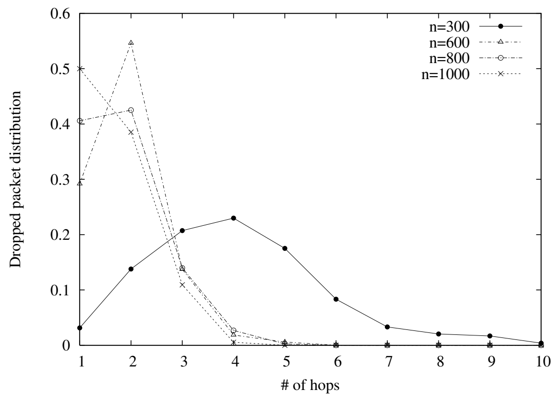


Figure 3.24. *Hop-wise probability of packet drop, ALBA,* $\lambda = 6$.

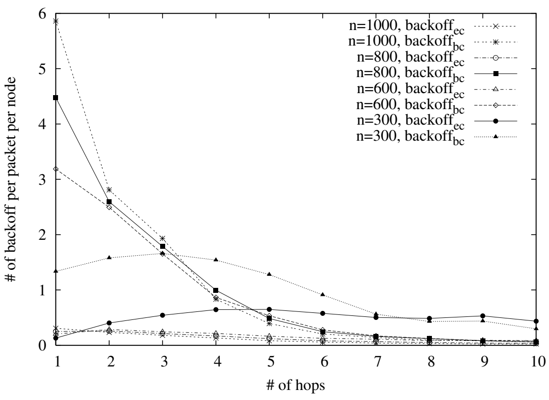


Figure 3.25. *Number of backoff per packet per node, ALBA,* $\lambda = 6$.

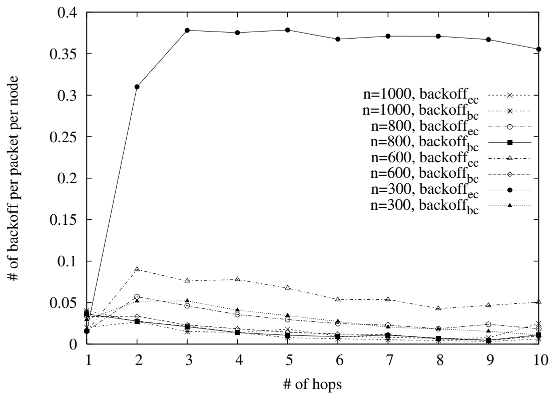


Figure 3.26. *Number of backoff per packet per node, ALBA,* $\lambda = 0.5$.

can be as high as 3.2 when $n = 600$ and $\lambda = 6$. Similar trends are observed for the normalized per-hop latency (Figure 3.28). This metric is defined as the ratio between the average time required to advance by one hop and the best case latency, *i.e.*, the time needed to perform carrier sense and send data with minimum overhead (one `RTS`, one `CTS` and one `ACK`). Queuing delay is not considered here. Both ALBA and GeRaF perform quite well considering that they have to deal with nodes following asynchronous awake-sleep schedules with a low duty cycle. When $\lambda \leq 2$, the normalized per-hop latency is very low, never exceeding 3.13. In GeRaF, as the load increases and the congestion builds up, nodes back off more often, and the greater time required to complete handshakes correctly degrades the forwarding performance. At high loads, ALBA's back-to-back transmissions improve efficiency, hence latency. Also, it is easier to find awake neighbors, since nodes in backoff follow the duty cycle. This significantly decreases ALBA's normalized per hop latency which is around one third of that experienced by GeRaF when $\lambda = 6$.

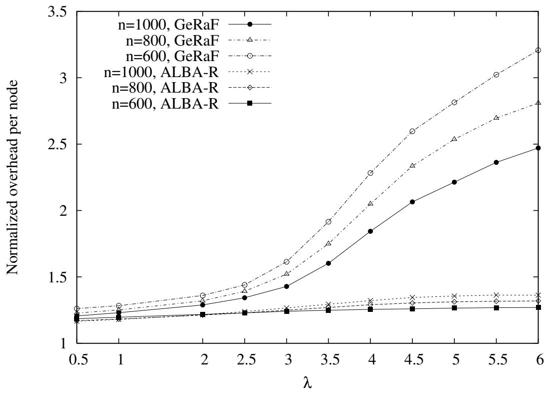


Figure 3.27. *Normalized overhead per packet per node, ALBA and GeRaF.*

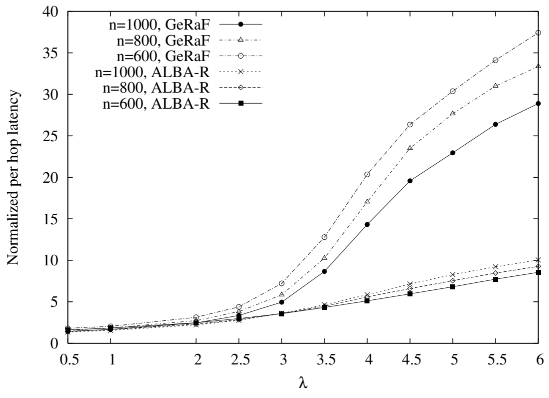


Figure 3.28. *Normalized per-hop latency, ALBA and GeRaF.*

3.6.2 Observations

While ALBA substantially outperforms GeRaF's transport capabilities, it still suffers from a problem typical of all greedy geographic protocols, namely forwarding around connectivity holes most likely fails. With a greedy (thus stateless), relay search, it is impossible to move packets out of nodes that happen to be the closest to the sink among all their neighbors. GeRaF's and ALBA's geographic search mechanisms cannot cope with this, since they do not keep track of dead ends, and always work on the set of nodes found during the relay search procedure. Stateful techniques such as those presented in [21] or [22] are examples of how to solve this problem by activating only a specific set of links or by marking nodes so that a way out of the hole is found.

In the following, we propose an algorithm that works according to this paradigm. The algorithm is designed to integrate itself into ALBA and to impose no further overhead on standard protocol operations. The following Section is devoted to the description and study of this feature in low density scenarios and in particularly difficult topologies.

3.7 The Rainbow Backtracking Algorithm and ALBA–R

Rainbow is the mechanism through which ALBA deals with dead ends. It is a blind adaptation algorithm whereby the nodes try to understand if they are dead ends by measuring how many of their forwarding attempts fail. If they think they are in fact dead ends, they stop volunteering as relays and change the way they forward packets. By doing this iteratively in a whole network, all nodes can find a path to a node from where greedy geographic forwarding can be resumed.

Differently from the many solutions proposed for dead ends (Section 3.9), Rainbow does not need to make the network graph planar. Nevertheless, if the topology is connected, Rainbow guarantees that all generated packets can be delivered to the sink (errors are still

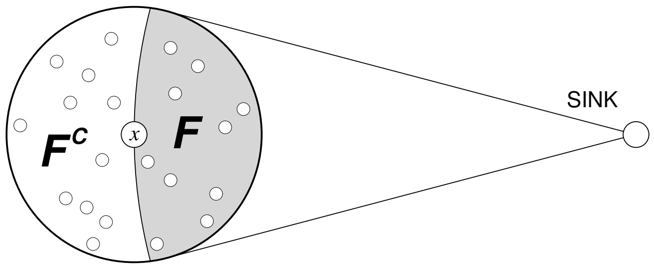


Figure 3.29. *The F and F^C regions.*

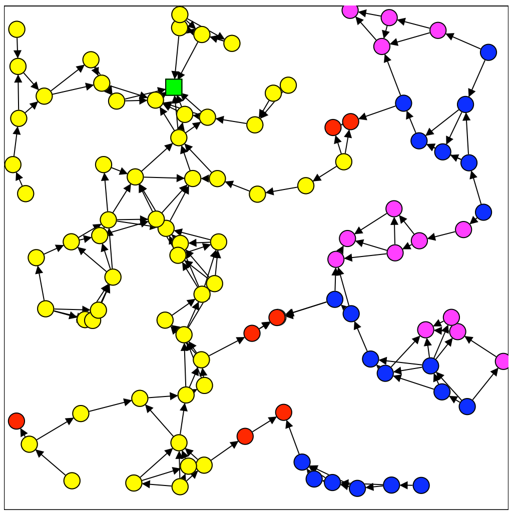


Figure 3.30. *Sample* 100 *nodes topology with nodes colored according to Rainbow.*

possible due to congestion, in the sense that packets can be dropped after too many failed attempts)

Rainbow works as follows. Consider a node engaged in packet forwarding. Let us denote with F the portion of its transmission area where relays offering positive advancement are located. Similarly, we call F^C the remaining part of the transmission area (see Figure 3.29).[10] Finally, let $C_1, \ldots, C_h$ be a set of h labels (or *colors*) that nodes assume according to their perceived ability to forward packets directly to the sink or not.

Initially, all nodes are colored C_1 (say, yellow) and function according to the standard ALBA rules (Section 3.6). If there are no connectivity holes, all nodes remain yellow. However, if a node is unable to relay a packet for more than a fixed number of attempts, it infers that it may be a dead end. The number of attempts before drawing this conclusion, N_{hsk}, must be carefully selected to avoid false positives. In the forthcoming results, we have tuned N_{hsk} considering the worst case (lowest duty cycle, and only one eligible forwarder in coverage). In particular, N_{hsk} has been set so that there is a negligible probability of finding eligible forwarders after more than N_{hsk} unsuccessful attempts in the worst case. In our simulations we never observed false positives: a perceived dead end was always a dead end.

In case the node decides it is a dead end, it stops participating in contentions initiated by other nodes. This avoids the risk that the node becomes overloaded if it cannot advance packets. When the probability of participating in contentions has decreased to zero it turns to color C_2 (say, red). These actions are important: the node must stop uselessly volunteer-

[10]The nodes located exactly at the same distance from the sink as the current sender are placed in F or F^C depending on their unique ID. If their ID is bigger than the sender's, they are placed in F. Otherwise they are located in F^C. This rule is used to prove Rainbow correctness (see Appendix A).

ing as a relay, and modify its behavior so as to route both incoming and locally originated packets around the hole ahead.

Red nodes behave differently in the relay search phase, as they try to send the packet away from the sink by searching for yellow or red relays in region F_C. When a yellow node (which has a greedy route available) is reached, regular ALBA operations are resumed. From that point on, the path going toward the sink will be made only of yellow nodes (each hop provides a positive advancement toward the sink).

Red nodes may be unable to find routes that lead to the sink through red or yellow nodes only. In this case, a red node progressively stops offering itself as a relay for red nodes and changes its color again, turning to C_3 (*e.g.*, blue). According to this new color, it resumes relay searches in F (instead of in F_C) but only looks for blue or red neighbors. Blue nodes do not volunteer to be next hops for red or yellow nodes, but can resort to them to find a route. Packets generated by a blue node will advance toward the sink through blue nodes until they reach a red node. Then they will travel away from the sink (via red nodes) until they reach a yellow node. From that point on, they will be taken care of by yellow nodes, and thus be ensured progress toward the sink.

If blue nodes cannot find blue or red eligible forwarders (blue or red neighbors in F) they switch color again, turning to C_4 (*e.g.*, violet). Similar to red nodes, violet nodes search into F_C for eligible relays. However only blue or violet nodes can answer violet nodes searching for relays. An example of this situation can be found in Figure 3.30. There, the nodes have one among the 4 colors introduced before. It should be noted that if properly walked, the network links offer paths toward the sink even from the worst positions. For instance, the packets from the violet nodes in the top right corner can get to the sink through blue, red, and finally yellow nodes. In the process, note that violet and red nodes send the packets away from the sink, as specified before.

The Rainbow process can be generalized for any number of colors, $C_1, \dots, C_h$. In particular, C_1-nodes are part of a direct route to the sink passing only through other C_1-nodes. When a node is forced to change to color C_k, it searches region F_C (if k is even) or region F (if k is odd). In both cases, nodes colored C_k only participate in the relay selection process initiated by C_k or C_{k+1}-nodes, and can forward their packets only to nodes colored C_k- or C_{k-1}. We will show that the Rainbow mechanism does not generate loops and that the nodes in paths to the sink with no less than h *alternations*[11] converge to the correct color C_h in a finite time. The formal proof of these claims is reported in the Appendix.

In the following, we will refer to ALBA-R$_h$ as ALBA incorporating the Rainbow mechanism with h colors in a single protocol. ALBA is then a special case of ALBA-R$_h$ when $h = 1$.

[11]An *alternation* is a change in the region queried for relays, corresponding to selecting relays with a different color than that of the sender. For example, if the sender looks for relays in F, the next hop will search in F_C, and vice-versa.

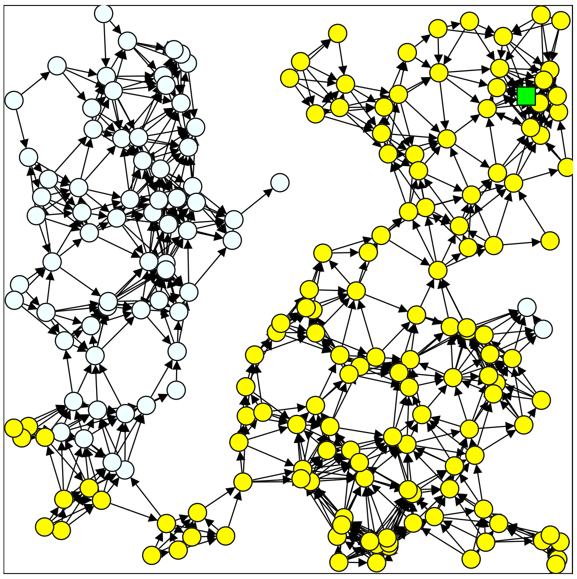


Figure 3.31. *Sample deployment with* 200 *nodes.*

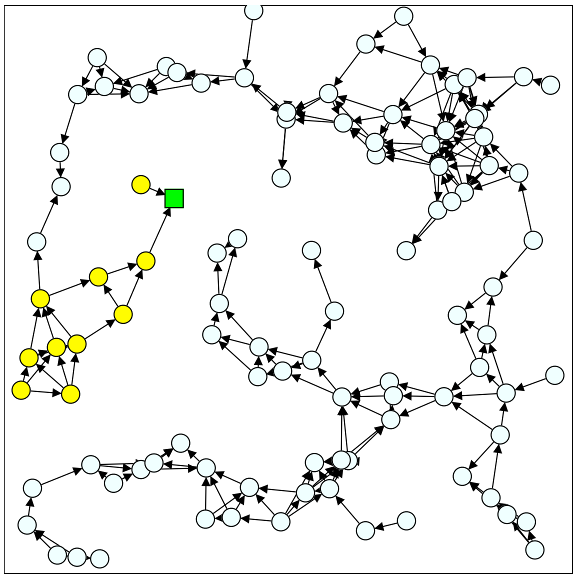


Figure 3.32. *Sample deployment with* 100 *nodes.*

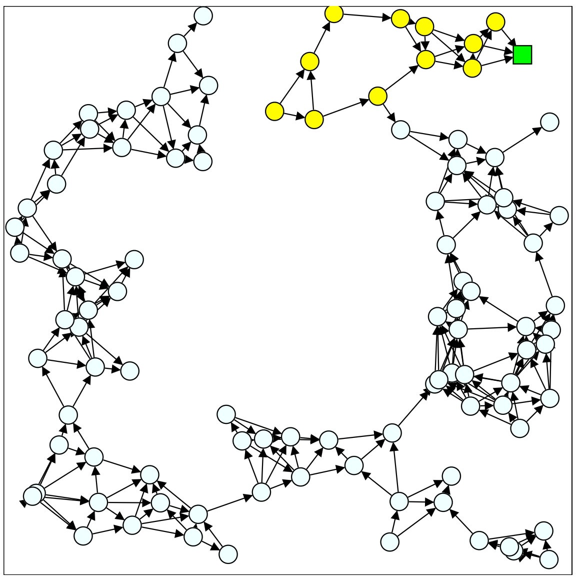


Figure 3.33. *Sample hole topology with* 100 *nodes.*

3.7.1 ALBA–R in sparse topologies

This first set of experiments specifically tests the effectiveness and capabilities of the Rainbow protocol. The first group of results refer to a scenario in which $n = 100, 200$ nodes are randomly and uniformly scattered in the deployment area. In either case, the network is very sparse, and each node has a small number of possible relays. Therefore, dead ends occur more frequently, and greedy forwarding is more likely to fail. An intuition of the impact of dead ends on protocol performance in these scenarios is given by the two sample

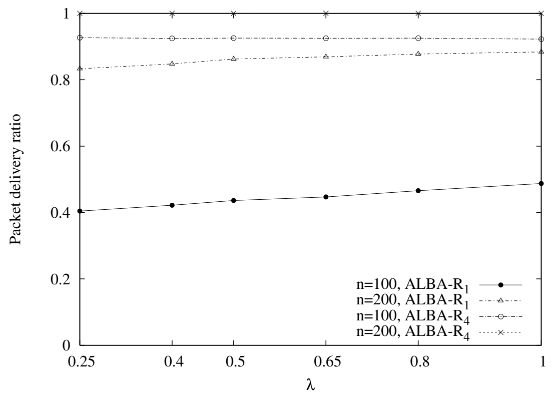


Figure 3.34. *Packet delivery ratio, ALBA–R$_1$ vs. ALBA–R$_4$, $n = 100, 200$.*

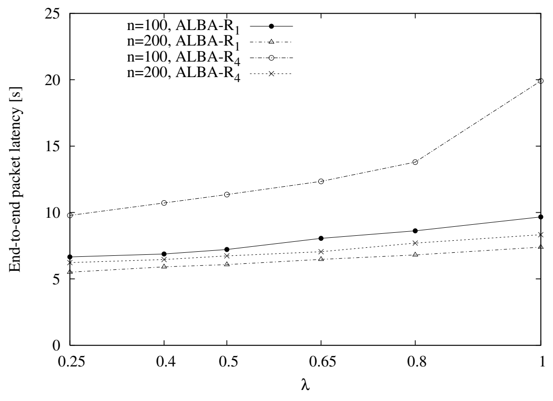


Figure 3.35. *End-to-end latency for yellow nodes, ALBA–R$_1$ vs. –R$_4$, $n = 100, 200$.*

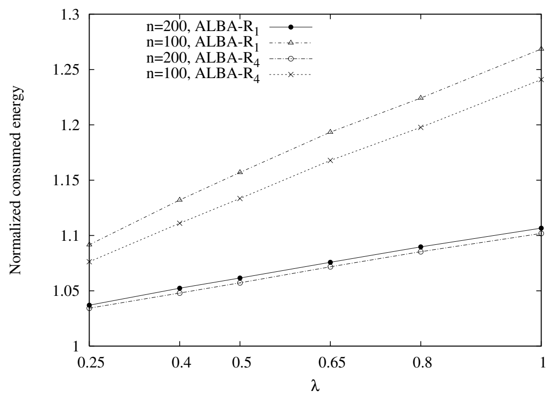


Figure 3.36. *Normalized node energy consumption, ALBA–R$_1$ vs. –R$_4$, $n = 100, 200$.*

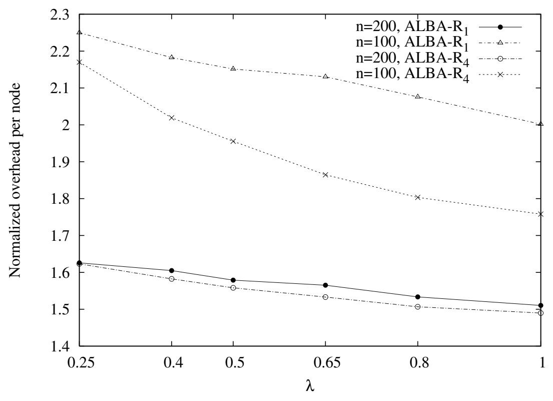


Figure 3.37. *Average normalized overhead per node, ALBA–R, $n = 100, 200$.*

"worst-case" sparse topologies depicted in Figures 3.31 and 3.32 ($n = 200, 100$). In these pictures, an edge (i, j) indicates that node j is an eligible forwarder located in the forwarding area F of node i. Nodes are colored yellow whenever they belong to a direct route to the sink. Otherwise their color is white. In both figures, some connectivity holes prevent greedy forwarding. More precisely, in the sample topology with 200 nodes nearly half of the nodes are white, *i.e.*, they are dead ends or connected only to paths that lead to dead ends (when using the greedy relay search). This percentage grows to as many as 90% white nodes in the topology with $n = 100$ nodes. When considering all the possible sparse topologies the average fraction of white nodes is a bit less than in these worst-case scenarios, though it is still significant. On average 40% (5%) of the nodes are white when $n = 100$ ($n = 200$).

Figures 3.34 to 3.37 depict the average packet delivery ratio, the end-to-end packet latency, the normalized energy consumption per node, and the normalized overhead experienced by ALBA–R$_1$ (ALBA without Rainbow) and ALBA–R$_4$, respectively. The Rainbow algorithm greatly improves the packet delivery ratio, even with a few colors. 100% (93%) of the generated traffic is correctly delivered when $n = 200$ ($n = 100$) and ALBA–R$_4$ is adopted.

	$\lambda = 0.25$	$\lambda = 0.5$	$\lambda = 1.0$
$n = 100$	13.38	13.42	13.43
$n = 200$	12.47	12.72	12.93

Table 3.2. *Average route length for re-routed packets in number of hops.*

	$\lambda = 0.25$	$\lambda = 0.5$	$\lambda = 1.0$
$n = 100$	24.12	26.46	42.55
$n = 200$	10.96	12.1	14.47

Table 3.3. *Average latency for re-routed packets (s).*

In ALBA–R$_1$ this percentage reduces to 83-85% (41-48%). It may seem counter-intuitive that the percentage of packets discarded by ALBA–R$_1$ is higher than the average percentage of non-yellow nodes. However, recall that ALBA–R$_1$ rules do not guarantee that packets generated by yellow nodes are successfully delivered. With ALBA's QPI-based relay selection, it is rare to select nodes on the path to dead ends. In fact, those nodes would exhibit a bad relaying history, and thus a high QPI. However, ALBA–R$_1$ cannot avoid that packets generated by a yellow node are directed toward a white node on the way to the sink. Those packets will also be lost.

Figure 3.35 reports the end-to-end latency of packets generated by yellow nodes (in ALBA–R$_1$ packets generated by other nodes are lost). As expected, ALBA–R$_4$'s higher packet delivery ratio over ALBA–R$_1$ translates into higher end-to-end packet latency. ALBA–R$_4$ also drives packets on longer routes when they are routed through yellow nodes. The reason is the QPI-based selection, that privileges good forwarding capability in spite of geographic advancement. For networks with 200 nodes, ALBA–R$_4$ has yellow routes up to 5% longer than in ALBA–R$_1$. When $n = 100$ the increase is higher (up to 20%). ALBA–R$_4$'s greater end-to-end latency for packets generated by yellow nodes reflects the increase in the length of the very yellow part of a generic route. Furthermore, the greater latency is partly due to the higher queuing delay associated to the higher load, as well as the fact that relay selection becomes more time-consuming (nodes back off more often since they cannot acquire the channel).

Tables 3.2 and 3.3 list the number of hops in source-to-sink paths and the end-to-end latency of packets which are re-routed, respectively. Packets which are generated by non yellow nodes traverse on average around 13.4 hops before they are delivered to the sink ($n = 100$). This corresponds to up to an increase of 83.3% with respect to the average length of the yellow routes, imposing a significant increase in the end-to-end latency.

Despite the increased traffic, the amount of energy consumption and overhead is smaller in ALBA–R$_4$ than in ALBA–R$_1$ (Figures 3.36 and 3.37). This non-trivial result is due to two key features of the Rainbow algorithm. Little resources are wasted in ALBA–R$_4$, because dead end nodes are no longer considered eligible forwarders. In turn, this results in faster and lighter handshakes. It is interesting to note that the overhead in sparse networks ($n = 100, 200$) decreases with increasing traffic both for ALBA–R$_1$ and ALBA–R$_4$ (Figure 3.37). The reason is that, as the traffic grows and node queues build up, ALBA makes a better use of bursty transmissions. The toll to pay for relay selection is thus shared by many packets, resulting in lower overhead per packet.

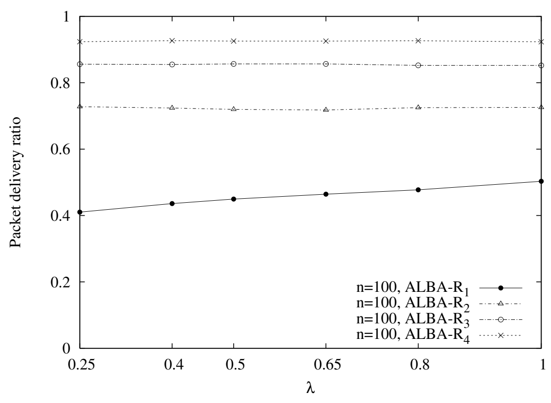


Figure 3.38. *Packet delivery ratio for $h = 1, \dots, 4$, hole topology.*

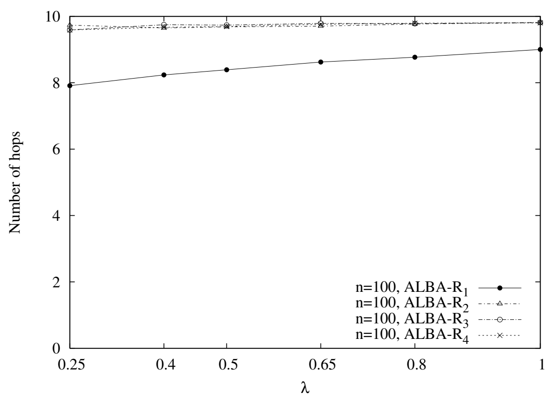


Figure 3.39. *Length of routes through yellow nodes, $h = 1, \dots, 4$, hole topology.*

We also consider different node deployments. In the following we call "hole topology" a scenario where no node is located inside a circle with diameter $160\,\mathrm{m}$, at the center of the deployment area. Nodes are randomly and uniformly distributed over the remainder of the area. An example of a hole topology is depicted in Figure 3.33. This kind of deployment is meant to represent a worst-case scenario for ALBA–R, since it requires nodes to learn routes that bypass the central hole. Let us now concentrate on hole topologies with $n = 100$ nodes (a very challenging situation). In this scenario, we studied how varying the number of colors used by Rainbow, h, impacts on protocol performance. We have set h to $1, 2, 3, 4$. As a general rule, we expect that increasing h increases the percentage of nodes that can successfully transmit their packets to the sink. Apart from rare, very congested situations, a packet is successfully delivered to the sink whenever it can be channeled on a route which changes direction (toward/away from the sink) at most h times. Therefore, we expect that the higher h, the better the delivery ratio. This intuition is confirmed by Figure 3.38 depicting the packet delivery ratio in the hole topology. When $\lambda = 0.5$, ALBA–R$_4$'s packet delivery ratio is 93%. The fraction of packets relayed by ALBA–R$_3$ is 86%. This fraction decreases to 73% with ALBA–R$_2$, and to 50% when the Rainbow mechanism is switched off.

Results obtained for the length of yellow routes (Figure 3.39) and for normalized consumed energy (Figure 3.40) are as expected. As h increases, the network traffic grows (since fewer packets are discarded). This means that nodes close to the sink (or in general on a yellow route) will be more loaded and will select relays mostly based on the QPI rather than the GPI. This results into a limited increase in the length of yellow routes. ALBA–R$_1$ is the most energy-consuming.

While using Rainbow allows to save resources that would be wasted in forwarding packets toward dead ends. However, a higher h allows the connection of more nodes to the sink, thereby increasing the delivered traffic. Ultimately, this slightly increases the consumed energy.

Let us now consider end-to-end latency in deeper detail. Packets generated by yellow

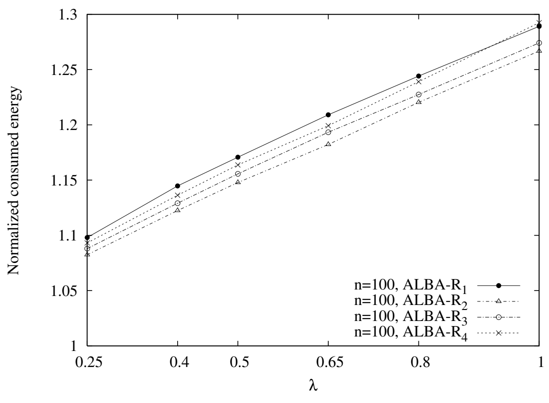


Figure 3.40. *Normalized consumed energy for varying h, hole topology.*

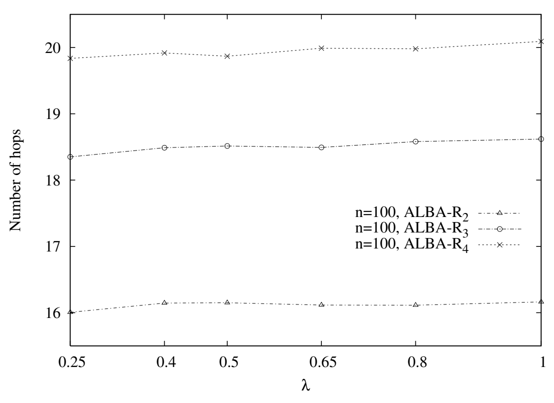


Figure 3.41. *Route length of re-routed packets for varying h, hole topology.*

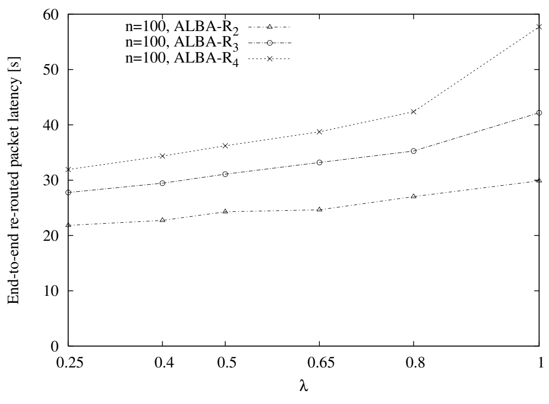


Figure 3.42. *End-to-end delay incurred by re-routed packets for varying h, hole topology.*

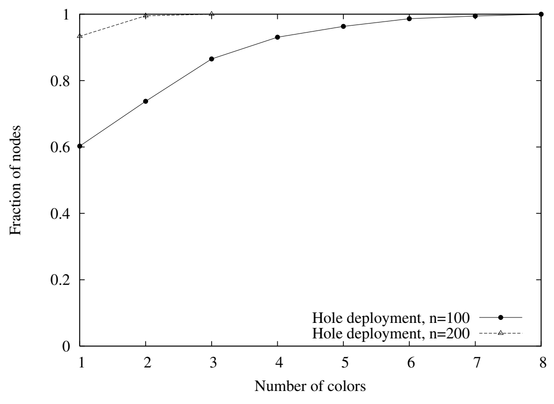


Figure 3.43. *Fraction of nodes connected to the sink for varying h, hole topology.*

nodes suffer an end-to-end latency (not shown here) that increases only slightly with h. This accounts for the larger amount of traffic traversing yellow nodes. When $\lambda = 1.0$, the end-to-end latency increases from 12.95 s to nearly 26.0 s when switching from ALBA–R$_1$ to ALBA–R$_4$. Let us now look at the end-to-end latency experienced by packets generated by non-yellow nodes Figure 3.42). In general, higher values of h induce longer routes and allow farther sources to reach the sink. However, this happens thanks to multiple alternations (*i.e.*, changes in the forwarding direction due to changes in the relay color on the way to the sink). Therefore longer routes are required on average to connect these farther sources. Correspondingly, the average latency increases with h. Figure 3.41 reports the route length for the re-routed packets (those gone through multiple colors). As more nodes find routes to the sink, the average route length increases. We observed that when varying h between 2 and 4, the number of hops traversed by re-routed packets increases from 16.2 up to 20.1. Similarly, the latency increases from 29.9 s up to 57.7 s.

Figure 3.43 shows the percentage of nodes able to deliver all packets to the sink as a function of h when $n = 100, 200$ nodes are scattered in the area according to the hole de-

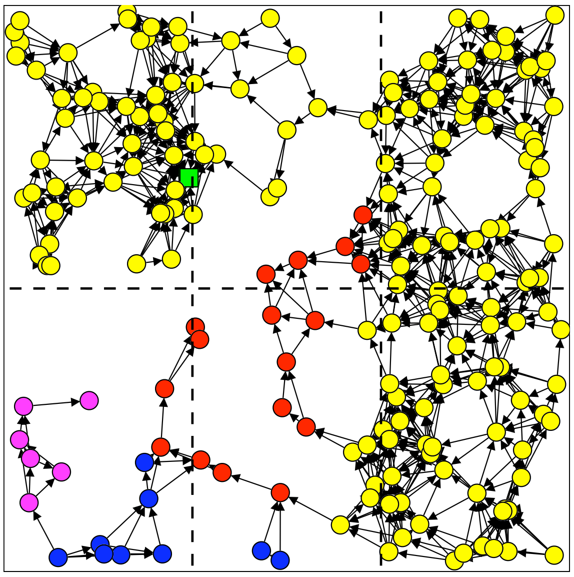


Figure 3.44. *Sample non-uniform deployment with* 200 *nodes.*

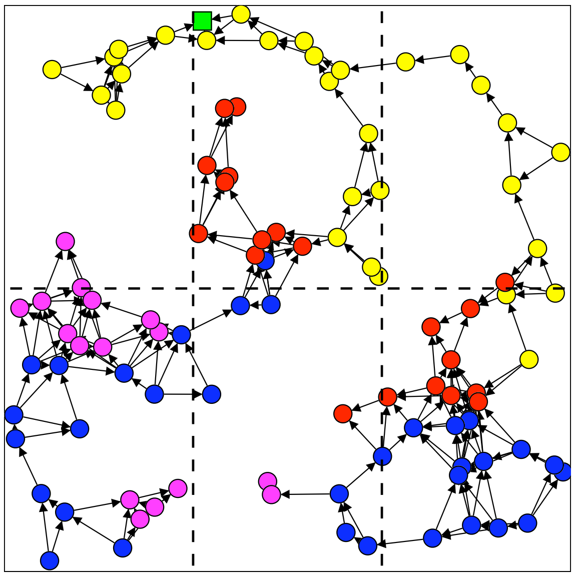


Figure 3.45. *Sample non-uniform deployment with* 100 *nodes.*

ployment. Two colors are enough to achieve 100% packet delivery ratio when $n = 200$. In the more challenging scenario with only 100 nodes, as many as 8 colors may be required for a 100% packet delivery. However, more than 90% of the nodes can successfully send packets to the sink in the same scenario when $h = 4$. Clearly, using as many colors as needed allows all nodes to be connected to the yellow nodes. It is still possible that excess backoffs lead to packet dropping in some circumstances, but still this is the only source of delivery errors.

We have also tested our protocols with different node deployments. In particular, we scattered the nodes according to a non-uniform deployment. The following Section presents this set of results.

3.7.2 ALBA–R performance in non-uniform topologies

In this subsection, we consider a different type of deployment. We wish to simulate a more realistic node scattering scenario, where rough or hilly terrain prevents a uniform node placement. To this end, we divide the same square area with side 320 m long considered for the previous results in six sub-zones. Three of them are high-density zones, whereas the other three are low-density. To create a challenging node distribution, 25% of the nodes are placed in low-density zones. This tends to limit the number of different paths found in low-density zones, and proves more difficult for ALBA–R to handle. The remaining 75% of the nodes are placed in high density zones. Two examples of a sort of worst-case node distribution with 200 and 100 nodes are shown in Figures 3.44 and 3.45, respectively.

As was the case in Section 3.7.1, there is a significant percentage of nodes (25% on average when $n = 100$, and an average of 6% when $n = 200$) which are unable to find a route to the sink ad thus can be connected only by the use of Rainbow. A first set of results compares

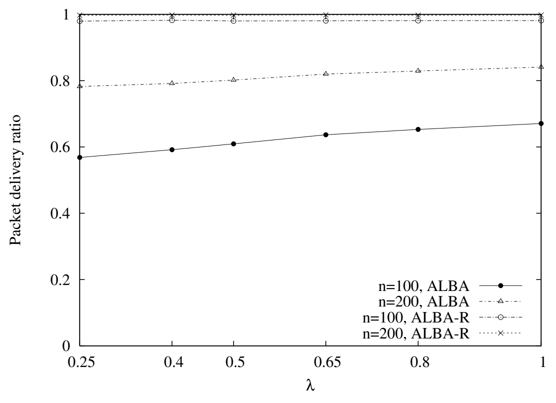


Figure 3.46. *Packet delivery ratio, ALBA–R vs. ALBA, $n = 100, 200$.*

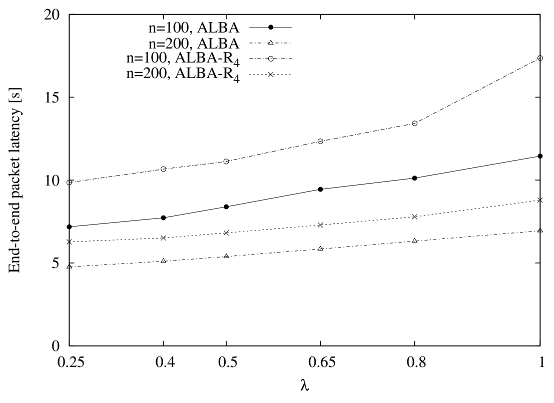


Figure 3.47. *End-to-end delivery delay, ALBA–R vs. ALBA, $n = 100, 200$.*

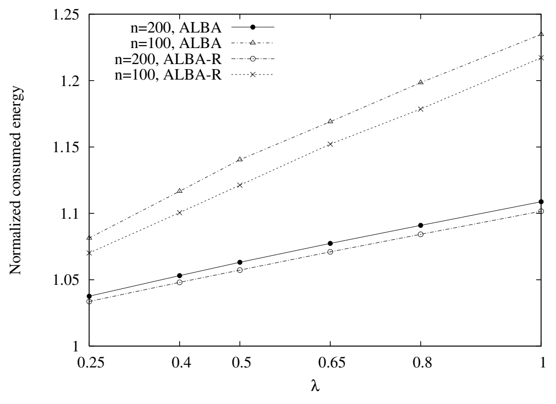


Figure 3.48. *Avg. norm. energy consumption per node, ALBA–R vs. ALBA, $n = 100, 200$.*

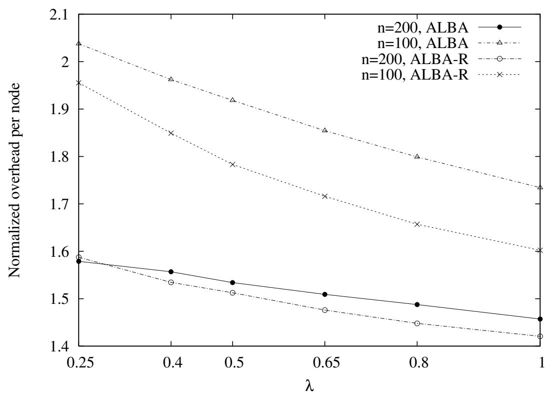


Figure 3.49. *Avg. norm. overhead per node, ALBA–R, $n = 100, 200$.*

ALBA (*i.e.*, ALBA–R_1) and ALBA–R_4. The average delivery ratio, the end-to-end latency and the energy consumption are depicted in Figures 3.46, 3.47 and 3.48, respectively. The number of colors for ALBA–R has been set to 4. At $n = 200$ The re-routing mechanism proves to be definitely necessary, as ALBA–R_4 delivers all the generated packets while ALBA only 79%–83%. This improvement is even more apparent in the sparser case ($n = 100$), where ALBA–R_4 delivers 98% of the packets while ALBA limits this percentage to 56.8–67%. Again, part of the reason why the delivery ratio of ALBA is low is that there is no guarantee that packets generated by yellow nodes are in fact delivered. Without Rainbow, it may happen that a dead end node is chosen as a relay during the QPI search phase. That would leave the packet stuck at the node, where it will be dropped. This also explains why the percentage of the packets discarded by ALBA is higher than the percentage of non-yellow nodes.

ALBA–R_4 generated more intense signaling and data traffic, which translates into higher packet latencies and energy consumption, as seen in Figure 3.47 and 3.48. Only the delay experienced by packets generated by yellow nodes is reported in Figure 3.47, since packets generated by other nodes are lost in ALBA. As traffic grows, ALBA selects more frequently

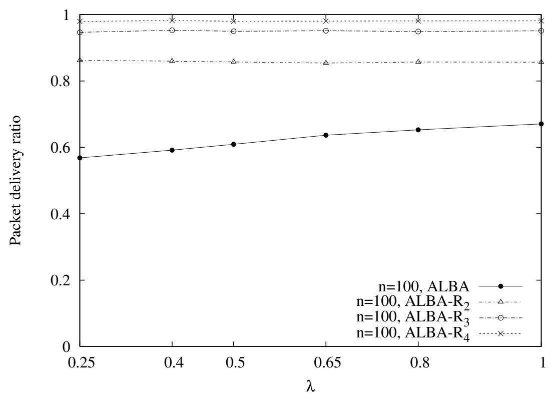


Figure 3.50. *Packet delivery ratio for varying h.*

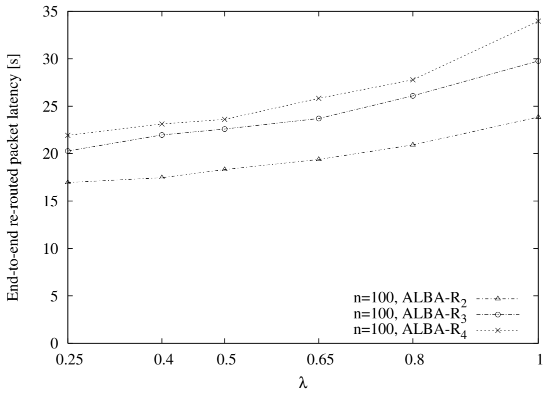


Figure 3.51. *End-to-end delay incurred by backtracked packets for varying h.*

relays offering a low QPI, but not necessarily a high GPI. This results in longer routes (up to 6% when $n = 200$ and up to 11% when $n = 100$) and consequently higher latencies. Higher network loads also make the relay selection procedure more time-consuming, as nodes are often forced to back off, mainly after having sensed a busy channel. Both facts have a detrimental effect on latency.

Figure 3.49 compares the overhead generated by ALBA and ALBA–R$_4$, and shows that ALBA–R$_4$ experiences a lower overhead. The reason is twofold. On one hand, ALBA–R$_4$ does not pay the toll of the many retransmission for the packets that have reached nodes on dead ends. On the other hand, it makes good use of the transmissions of packets in bursts.

As before, it is interesting to note what happens when the number of colors h in ALBA–R is varied. Increasing h is expected to improves the re-routing capability, and to increase the probability that a packet reaches the sink. Apart from rare, very congested situations, a packet is successfully delivered whenever it has a path to the sink which changes direction (toward/away from the sink) at most h times, *i.e.* that has at most h alternations. Therefore, the higher h, the better the delivery ratio (Figure 3.50).

Figure 3.51 shows the average end-to-end latency incurred by the packets generated by non-yellow nodes. As a rule of thumb, higher values of h induce longer routes and allow farther sources to reach the sink, although paying the price of multiple changes of direction (alternations). Correspondingly, the latency increases.

To complete ALBA–R's performance evaluation, it is interesting to comment on its behavior in large topologies, when it is very unlikely that Rainbow is resorted to. The aim is to show that Rainbow does not cause any particular performance decrease. We carry out this performance evaluation in the non-uniform deployment scenario for two reasons, because it is more general, and because the low density zones are likely to require only a localized (if any) application of Rainbow.

Figures 3.52, 3.53 and 3.54 depict the average energy consumption per node, the end-to-end packet latency, and the protocol overhead, respectively. The packet delivery ratio is not

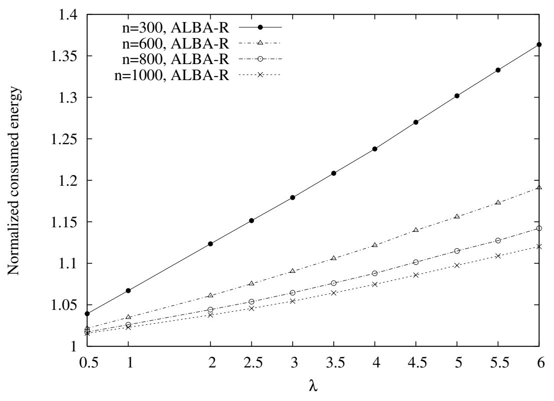


Figure 3.52. *Average normalized energy consumption per node.*

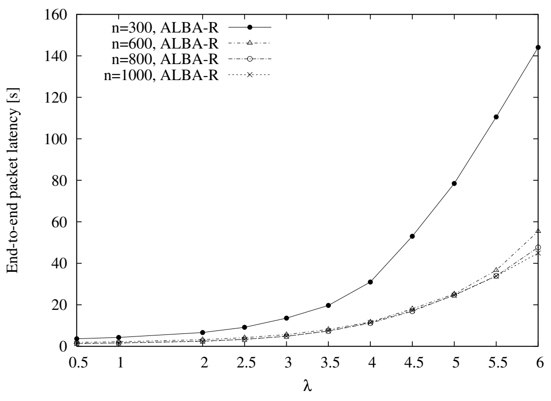


Figure 3.53. *Average end-to-end packet latency [s].*

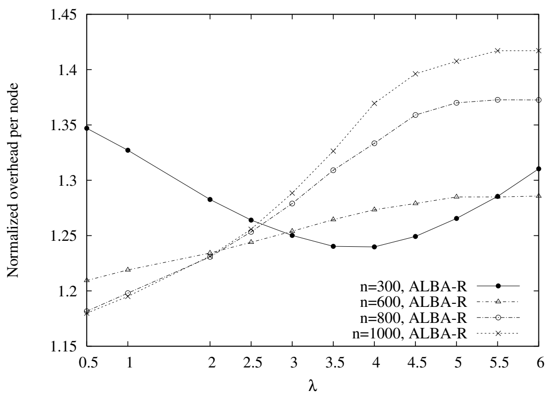


Figure 3.54. *Average normalized overhead per node.*

explicitly shown here, since it is basically 100% for $\lambda \leq 4$ and decreases only after that. When $\lambda = 6$ it can drop to as low as 85% ($n = 1000$). The reason is that at high loads the congestion around the sink is an obstacle to packet delivery. At these high loads the probability to sense the channel busy in the area close to the sink is very high. These values prove that ALBA–R_4 performs well till the network saturates. The energy consumption (Figure 3.52) and the end-to-end packet latency (Figure 3.53) show an inverse trend with respect to the packet delivery ratio. A more congested network means higher time and energy to find relays, and therefore higher end-to-end latencies. A similar trend is shown by the overhead (Figure 3.54). We observe that ALBA–R_4 has a very limited overhead per packet. When $\lambda = 0.5$ it is only 18–21% higher than the minimum when $n = 600, 800, 1000$. As the traffic load increases, ALBA maintains good performance in terms of overhead, mainly due to its light relay selection scheme (only a few packets need to be exchanged for each contention) and to the fact that the overhead for relay selection is shared among a burst of packets (since transmissions take place in bursts). When $\lambda = 4$ ($\lambda = 6$) the normalized overhead per node is only 1.37 (1.41) in large networks ($n = 1000$). The different behavior observed in networks with $n = 300$ depends on the combination of two effects. On one hand, the topology is more constrained (*i.e.*, traffic has to go though fewer nodes), whose queues build up at medium load. This allowing ALBA–R$_4$ to exploit bursty packet transmission. This explains the initial decrease in the overhead. On the other hand, with fewer eligible relay nodes the network soon becomes congested. When the relays' queues are full, such nodes refrain from offering themselves as relays, requiring senders to handle more contentions and handshakes, before a packet can actually be forwarded to the next hop.

3.7.3 Observations on ALBA and ALBA–R

Overall, the results presented up to this point suggest that Rainbow can substantially boost the performance of ALBA in topologically difficult node deployments, *i.e.*, where greedy geographic routing is not possible. Moreover, ALBA is not negatively affected by Rainbow when the density is such that packet do not need to be re-routed. However, Rainbow has one main problem, namely its transient phase when nodes are learning how to go around dead ends. During this phase, many packets may be lost, especially if the number of colors is high. If a topology requires 10 to 20 colors to be traveled correctly, the learning phase can be very long.

A solution to this problem can be to transform blind learning into training. We could set up the network so that the nodes have a very high duty cycle, or are always on. In this phase, they keep sending dummy packets, with the purpose to have node converge fast to their color. With a high duty cycle, it is very likely that if there are forwarders providing geographic advancement, then they are in fact found. In turn, this speeds up dead end detection and keeps the probability of false positives negligible.

3.8 Comparison between ALBA and MACRO

This Section is devoted to a performance comparison between two different MAC/routing protocols for wireless sensor networks. As both protocols are cross–layer designed, this is also a chance to highlight the potential problems that may arise when using this design paradigm.

The protocol we compare to is MACRO [5], an acronym for MAC/ROuting. Basically, this protocol splits the set of neighbors that are eligible to forward the packet into regions, and tries to find the best node in this set according to a certain online relay search procedure. Both the criterion and the procedure are detailed hereon. The choice of MACRO as a term of comparison is motivated by two main reasons. Firstly, the protocol has been recently proposed as a competitor for GeRaF from the point of view of energy consumption. Thus, it is interesting to understand if ALBA can do better, since its design is derived from GeRaF.

Name	*Value*	*Description*
N_r	4	Number of relay regions
T_s	$0.0521\,\mathrm{s}$	Carrier sense length
T_{bo}	$1.095\,\mathrm{s}$	Backoff interval length
N_{MaxAtt}	50	Maximum number of attempts during relay search

Table 3.4. *Common parameters.*

Secondly, MACRO pursues energy optimization objectives in an online fashion. In order to do this, it requires some design choices that may prove to seriously limit the overall effectiveness. Nevertheless, MACRO has one drawback, namely it seeks advancement in a somehow greedy way, but does not include any means of routing around dead ends. Hence, we will limit our evaluation to sufficiently dense uniform deployments, and specifically to topologies where it is not necessary to re-route a packet.

Our simulation scenario consists of a network with 600 uniformly deployed nodes whose transmission range is varied to be $30\,\mathrm{m}$, $60\,\mathrm{m}$ and $100\,\mathrm{m}$. This allows to study the relationship between the higher neighbor density yielded by a larger coverage area and the higher power needed to increase the coverage area. This is an interesting evaluation in light of the power optimization sought by MACRO. The energy consumption model is as specified in (3.26) and (3.27) [20]. In all experiments, the duty cycle of the nodes has been set to 0.1. For fairness of comparison, the parameters that the two protocols have in common have been set to the same values, as shown in Table 3.4. More specifically, the two protocols have the same maximum backoff interval T_{bo} (lasting $1.095\,\mathrm{s}$) and the same channel sense duration T_s, set to $0.0521\,\mathrm{s}$. The specific parameters of ALBA have been set as follows: the maximum number of QPIs is 4, and the maximum length of a data burst has been tuned and set to 5 packets (Table 3.5). As for MACRO, the remaining relevant parameters are the threshold probability p_{th}, the cycle duration T_c, and the wait time WT. Their values (tuned through simulations) can be found in Table 3.6.

3.8.1 Description of MACRO

MACRO is a protocol for data dissemination in wireless ad hoc and sensor networks. The work in [5] evaluated its performance in a converge-casting scenario. Similarly to GeRaF, it makes use of duty cycling in order to reduce the energy expenditure. However, unlike GeRaF, the relay search is aimed at optimizing a specific metric involving power consumption and advancement. Specifically, each relay is assigned a *weighed progress*[12] defined as the ratio between the advancement offered and the power consumed to reach the relay.

[12]In the following, we will use the terms *weighed progress* and *gain* interchangeably.

Name	*Value*	*Description*
M_B	5	Max burst length
N_q	4	Number of queue size regions

Table 3.5. *ALBA parameters.*

Name	*Value*	*Description*
T_c	0.16 s	Duration of the wakeup phase
WT	0.3 s	Maximum Wait Time for `CONTROL_ACKs`
p_{th}	0.3	Threshold prob. for achieving a better gain

Table 3.6. *MACRO parameters.*

This metric favors the selection of nodes that go farther with comparable power, or that offer similar advancement but require less power to be reached.

For relay selection MACRO uses the following protocol. First, the power for reaching the neighbors is divided into N_r^M levels. According to the unit disk graph model, each level allows to reach all relays inside a corresponding circular region, whose radius depends on the node power. Let R be the maximum transmission range: The ith region comprises all neighbors whose distance from the source is smaller than or equal to iR/N_r^M. Let also P_i denote the minimum value of the power level which enables a node to reach all the nodes in zone i ($P_i < P_{i+1}$ for all $i < N_r^M$). Note that MACRO regions are circles centered on the transmitter. As such, they are different from GeRaF's regions, that are obtained instead by intersecting circular rings centered on the sink with the forwarding area of the sender.

MACRO requires that all nodes offering positive advancement toward the sink be active before beginning a contention. This enables the selection of the best relay among all neighbors rather than among only the currently active ones (as in GeRaF). To this purpose, the sender starts transmitting short `WAKE_UP` packets over a whole wake-up cycle time T_c (the total duration of an awake/asleep cycle for any node). `WAKE_UP` packets are spaced so that at least one will fit in each neighbor's awake time. In other words, the interval between two subsequent `WAKE_UP` packets is shorter than the awake time of a node. After transmitting `WAKE_UPs` for T_c seconds (waking up all its neighbors), the transmitter sends a `GO` packet, which has the double purpose of asking for relay candidates and of soliciting the calculation of the weighed progress. Based on its own gain, each neighbor computes a random access time (the higher the gain, the lower the time) after which it answers the `GO` with a `CONTROL_ACK` packet, offering to serve as a relay. Random access times are bounded by a value WT which is a tunable protocol parameter. The `CONTROL_ACK` contains the relay identity and the associated gain. When the sender receives a `CONTROL_ACK`, it checks whether the

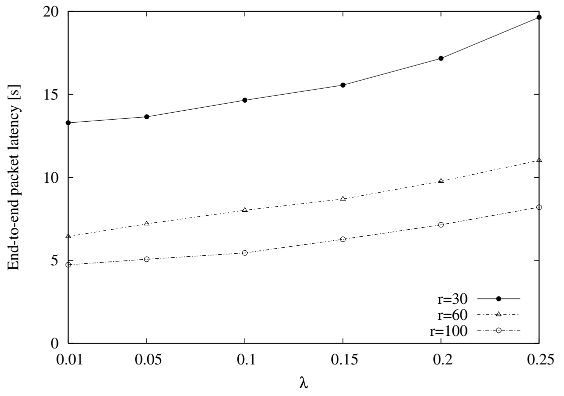


Figure 3.55. *End-to-end latency, MACRO.*

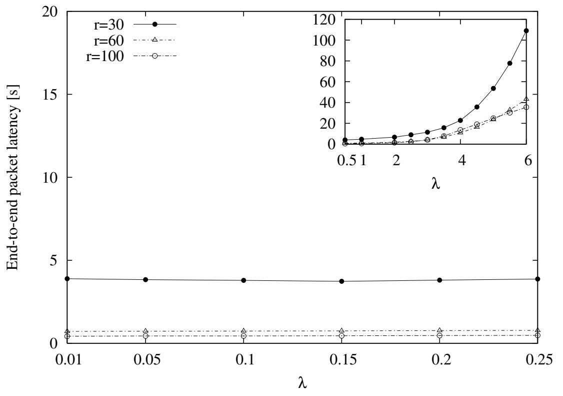


Figure 3.56. *End-to-end latency, ALBA.*

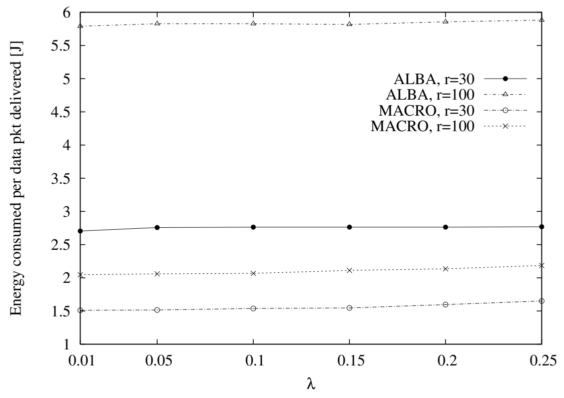


Figure 3.57. *Average energy consumed to transmit a data packet to the sink: ALBA and MACRO, $r = 60\,\mathrm{m}, 100\,\mathrm{m}$.*

gain declared inside the packet is better than the gain of the best candidate relay identified so far. If this is the case, it updates the identity of the best candidate relay and its associated gain. The sender also computes the probability that a better weighed progress can be achieved via a node in the same relay region which has chosen a longer wait time. If this probability is greater than a threshold p_{th}, it keeps waiting for additional CONTROL_ACKs. In any case, it does not wait longer than WT.

When the scan of a relay region is finished, the sender computes the probability to obtain a better weighed progress by querying the nodes in the next relay region. The interrogation starts only if this probability is greater than a threshold. At the end of this procedure the transmitter has chosen as relay the node offering the best weighed progress (among those that answered) and forwards the data packet to it. Similarly to GeRaF, MACRO is a cross–layer solution and is completely distributed. Its main merit is in the definition of the weighed progress, aiming at jointly optimizing the energy consumption needed for one hop forwarding and geographic advancement. On the other hand, MACRO is more complex

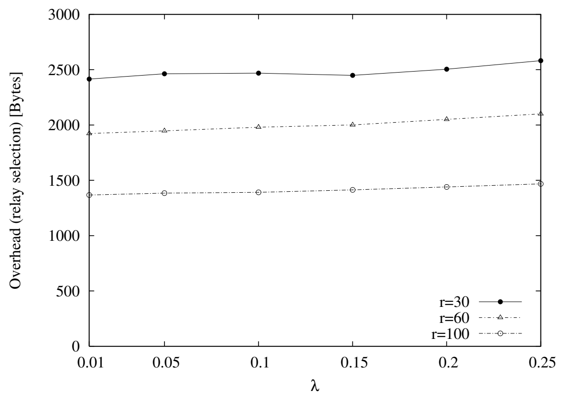


Figure 3.58. *MACRO relay selection overhead in* Bytes.

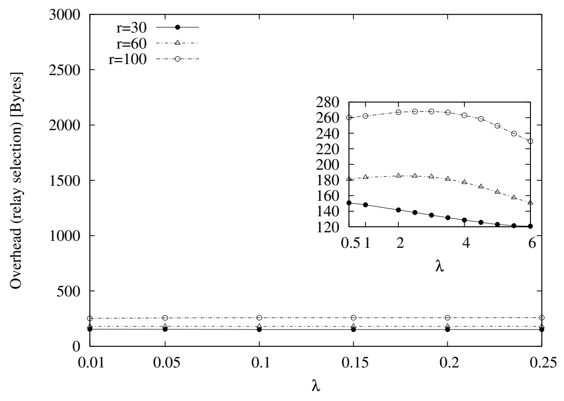


Figure 3.59. *ALBA relay selection overhead in* Bytes.

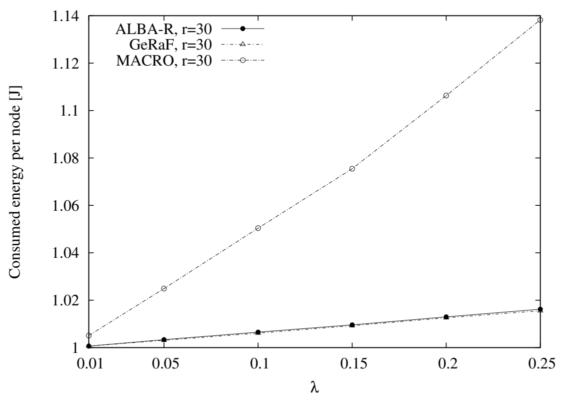


Figure 3.60. *Normalized energy consumption.*

than GeRaF, as it requires the implementation of power control into the radio hardware. MACRO also imposes higher overhead (and hence higher energy consumption) for completing the relay selection process, since it needs to wake up all the neighbors of a node before the relay selection process can be initiated. In the performance evaluation section we will show that these limits overcome the potential advantages associated to the gain-based relay selection criterion.

3.8.2 Performance comparison

Figures 3.55 to 3.60 depict the results of our comparative performance evaluation, when varying the transmission range r and the traffic load λ. The first thing to note is that ALBA scales much better with increasing traffic. MACRO shows difficulties in supporting traffic if $\lambda > 0.25$ while ALBA performs very well with 16 times as much traffic. A comparison between the two protocols is thus possible only when the traffic is low ($\lambda \leq 0.25$). Figures 3.55 to 3.59 depict the performance of the two protocols at low load only. ALBA's performance at medium-high traffic is displayed in a smaller subgraph within the main figure.

Figures 3.55 and 3.56 show the end-to-end average latency experienced in MACRO and in ALBA, respectively. In general, we observe that latency increases with traffic load and decreases with the transmission range r, since routes are formed by longer (hence, fewer) hops for larger r. MACRO performs poorly in terms of end-to-end latency. As seen from Figures 3.55 and 3.56, latency degrades quite fast for MACRO, whereas ALBA can support a much higher traffic before the end-to-end delay begins to increase noticeably. At $\lambda = 0.25$ MACRO packets experience an average end-to-end latency equal to 20s when $r = 30$m. In the same scenario ALBA reaches a latency of $20\,\mathrm{s}$ only around $\lambda = 4$ (*i.e.*, with 16 times as much traffic).

The energy consumption behavior deserves a more detailed study. Figure 3.57 shows the average energy consumed per packet, focusing on the energy needed to transmit the data packet only (*i.e.*, disregarding the energy consumption associated to control packets exchanged for relay selection). Figure 3.57 confirms that a relay selection based on MACRO's weighed progress can lead to advantages in terms of energy needed to transmit one data packet. The higher the transmission range, the more neighbors are available, the better the advantages obtained via power control and the weighed progress metric. For example, with $r = 100$ the energy consumed to transmit a data packet using ALBA can be almost three times higher than in MACRO. That is because ALBA does not exploit power control. Moreover, like GeRaF's, ALBA's regions potentially include nodes placed at the boundaries of the coverage area (see also Figure 3.20). These advantages are less apparent when considering shorter transmission ranges. In this case, the energy drained by the electronic circuitry may be a non-negligible, or even dominant, component of power consumption. Therefore, power control is of little use.

Despite the potential energy saving when performing packet forwarding, Figure 3.60 counterintuitively shows that the overall energy consumed per node is greater in MACRO than in ALBA in all the considered scenarios. When $\lambda = 0.25$ MACRO has a 12% higher normalized energy consumption than ALBA. The reason is that the advantage of MACRO is in data forwarding. Figure 3.57 neglects the energy spent for relay selection, and thus it allows us to focus on this advantage. Specifically, MACRO requires a sender to wake up all of its neighbors before relay selection begins. Nodes are awakened using beacon transmissions, that consume both time and energy. This is confirmed by Figures 3.58 and 3.59, that compare the overhead (in Bytes) required to select a relay. MACRO's overhead is shown to be one order of magnitude higher than ALBA's. Let us consider the case $r = 100\,\mathrm{m}$. In this case, the number of potential relays is very high (around 90). Hence it turns out that the selected relay can be found in the first or second relay region scanned. The overhead due to `WAKE_UP` messages sums up to between 750 and $1500\,\mathrm{Bytes}$. Thus, `WAKE_UP` messages are the major contribution to the relay selection overhead, whereas much less is due to `GO` and `CONTROL_ACK` messages.

All other control messages are quite short and give little contribution to the overhead. Namely, only one `GO` message per scanned relay region and one `CONTROL_ACK` for each potential relay are transmitted. The overhead per relay selection is hence around 1400 bytes at $r = 100$m. As r decreases, relays are selected after having scanned a higher number of regions, resulting into a higher number of WAKE_UP messages, that soon become the dominant factor in the total overhead. When nodes have fewer potential relays, fewer CONTROL_ACKs are transmitted. However, in this case the increased number of searched relay regions (and hence the increase in the number of transmitted WAKE_UP messages) imposes higher overhead. When $r = 30\,\mathrm{m}$ around $2500\,\mathrm{Bytes}$ are transmitted on average to select the next hop.

ALBA's relay selection procedure is much lighter. Relays are selected only among awake neighbors through a fast two-step QPI/GPI-based search procedure, that identifies the best forwarding condition and then the best advancement. This proves to be faster and more lightweight in all tested ranges and traffic scenarios, resulting also in lower energy consumption. As a rough estimate, ALBA's overhead is around 10 times smaller than MACRO's. Similarly, the time needed to complete a contention is 2 to 5 times shorter, depending on λ and r. Overall, ALBA shows superior performance with respect to all the relevant metrics we considered. MACRO's advantages are limited to the very data transmission phase, and are often outweighed by the quite heavy overhead needed for any other protocol operation.

3.9 Related Work

The following survey is meant to highlight and describe the first geographic approaches seen in the literature as well as the most recent advances in geographic routing algorithms ad hoc and sensor networks.

3.9.1 Traditional Approaches

Geographic routing as a paradigm for packet forwarding in multihop wireless networks was born in the 80's, with some first attempt to assess the performance of simple algorithms based on geographic information. The first to introduce the concept of "progress" (or "advancement") were Tobagi and Kleinrock in [23]. In their paper, progress is defined as the length of the projection of the segment joining the source and a neighbor to the line from the source to the destination. Their protocol, Most Forward within Radius (MFR), chooses the neighbor offering the maximum progress. Nelson and Kleinrock [24] extend the approach by choosing a neighbor that offers a random node that offers progress and arguing the existence of a tradeoff between the distance of the neighbor and the probability that the packet is not lost in a dead end (where the current holder is the closest to the destination among its own neighbors). Hou and Li [25], in particular, chose the neighbor that offered the minimum advancement in their Nearest Forward Progress (NFP). A slightly different scheme was proposed by Finn [26]. In this case, the chosen neighbor is the closest to the destination among those in coverage. A variant of this method, GEDIR [27], the message is dropped if the best choice for the current sender is to return the message to the node where it came from. This allows the detection of a dead forwarding branch. Similar to NFP, Nearest Closer (NC) [28] selects the nearest neighbor closer to the destination than the current node. Similar in concept is also Compass Routing [29], where the packet is forwarded to the node that is closest to the sender-recipient direction.

Distance Routing Effect Algorithm for Mobility (DREAM) [30] uses location information for a different purpose. DREAM is a location-based routing where the broadcast of location updates is performed according to the mobile node speed and distance from destination. Each node is assumed to own a measure of its position, and a location table, where the position of any other node in the network is stored. Routing is performed by sending the packets to all one-hop neighbors in the direction of the receiver as available in the location table, hence routing performance directly depends on how efficiently the entries in the table are updated. In [30] it is observed that for equal velocity, a mobile is seen to be moving more slowly by more distant nodes. Therefore, location updates need to travel longer distances less frequently than shorter ones. Location updates are therefore assigned a life time in [30]: shorter life times for frequent updates and smaller distances, longer life times for less frequent updates but longer distances. A recovery procedure (not specified in the paper) should be used when location information is unavailable or out-of-date. The protocol is compared to DSR, over which it is shown to exhibit smaller delivery delays.

Location-Aided Routing [31] (LAR) does not consider periodic distance-based location updates as in [30], and uses location information to perform route discovery instead of routing. Basically, LAR defines an expected zone (where a node is supposed to be located based on its las known position and movement speed) and a request zone (where a route discovery is performed on geographically local basis). With LAR scheme 1, the request zone is, *e.g.*, a rectangle, going from the query initiator (located at one vertex) to the whole expected zone (located at the vertex opposite the initiator). Every intermediate node that receives the query and is inside the request zone forwards it further, and discards it otherwise. With LAR scheme 2, instead, any intermediate query forwarder which is inside the request zone propagates the packet only of its location provides some advancement or little regression. Other formulations of the request zone are also devised, and an adaptive approach that changes the shape/orientation of the zone at each forwarding step is provided. The algorithm is compared (version 1 and 2) to simple flooding, and shows apparent advantages.

The concept of progressive refinement of the knowledge about the destination location is known since [32], where some circular zones are defined around the destination, each being smaller than the previous one and contained inside it. Inside each zone, the packet is forwarded in a greedy manner toward the center of the next circular zone, and thus undergoes a series of course corrections eventually leading to a delivery to the final destination. Simulations show that the protocol is also bandwidth-efficient.

Geographic Energy-Aware Routing (GEAR) [33] calculates the cost of links and paths and routes the packets toward the region of interest before performing a localized forwarding. When in the presence of a hole, it makes use of a learned energy metric to forward the packet to a neighbor that is convenient under an energy point of view.

A slightly different approach is presented in [34]. The authors propose Geographic Adaptive Fidelity (GAF), that is basically a geographic overlay routing protocol, that establishes a virtual grid over the network area. The grid element size depends on the coverage range, and are such that all nodes inside an element can communicate with all nodes in the adjacent elements. Routing is performed by advancing toward the destination through a different grid element at each hop. However, the use of the overlay reduces the maximum advancement by a factor of $\sqrt{5}$.

In [35], Camp *et al.* carry out an extensive comparison of LAR, DREAM [30] and DSR [13]. The authors also draw some conclusions about the usefulness of the restricted location updates in DREAM, as well as the impact of location accuracy on DSR.

Since its publication, LAR has drawn a whole set of contributions, more or less aimed at improving some aspects, introducing some new features, or assessing its performance with different technologies (such as directional antennas). Among these, we highlight PMLAR and LAKER.

Predictive Mobility and Location-Aware Routing (PMLAR) [36], by Lu and Feng, tries to improve over LAR by predicting the extension of a node's movements using a Gauss-Markov mobility model [37]. The whole protocol is very similar to LAR, except for the route maintenance procedure, where the sender is supplied with some information from the destination regarding its predicted movement zone. This information is used to initiate a new route discovery including the predicted zone. PMLAR is shown to perform better than LAR and DSR using both the Gauss-Markov and the random waypoint mobility models.

In [38], Li and Mohapatra propose to merge some features of DSR and LAR into Location Aided Knowledge Extraction Routing (LAKER). In more detail, route caching is combined with restricted geographic flooding to exploit different densities in the following way. A forwarding route is set up along with a *guiding route*, *i.e.*, a path formed of a number of waypoints, corresponding to locations where many nodes are clustered. The rationale behind this is to exploit the slowly varying structure of the group (with respect to individual nodes which may possibly appear and disappear quickly. Such clusters are cached (*e.g.*, storing a measure of the local population density) to guide the following route discovery processes. A byproduct of this approach is that such waypoints are likely to yield routes which circumvent possible obstacles in the territory.

The concept of dynamic forwarding scope is also considered in Angle Routing, by Banka and Xue in [39] with the aim of reducing overhead. In order to find a route to a destination, each sender sets an exploration scope (defined by an angle centered at the sender), traces the bisector of the angle, and follows a minimum distance to destination to select one node per section. Both nodes are required to further forward the packet, in order to provide some path redundancy. If no node is found inside the initially chosen scope, the corresponding angle is increased by 5 degrees at each failure, until a maximum extension is reached which turns into packet dropping. The protocol is shown to gain an edge over LAR and pure flooding.

Location Area-Based Ad hoc Routing (LABAR), by Záruba *et al.* [40] merges zone routing and virtual clustering to yield a more efficient protocol in terms of route length as compared to the shortest path. G-nodes (*i.e.*, those having GPS equipment for location estimation) form zones by broadcasting beacons to nearby neighbors (not necessarily one-hop only). The backbone formation is initiated by a root node and proceeds by including zones as the message is propagated among G-nodes. Upon transmission, the sender inquiries its G-node, which calculates the direction toward the G-node nearer to the destination. Routing is performed through intermediate nodes and zones, passing through G-nodes, where calculation of the direction toward the destination is repeated. If a hole should be encountered, the virtual backbone is used for reaching the correct zone.

3.9.2 Extensions of traditional approaches

Energy-Efficient Geographic Routing [41] (EEGR) is partially derived from GAF and GEAR. It inherits the virtual grid structure and the switch among active, sleeping and discovery states from GAF. Grid routing is used to forward queries from some originator to the location of interest, forwarding the packet through the active nodes path, and choosing the next grid element based on the position of the destination. When forwarding packet back to query issuers, the next hop in a multihop route is chosen according to a hybrid metric that accounts for the distance of a neighbor to the centroid of the destination region and its available energy. The results show some gain over GEAR and substantial gain over flooding and GPSR.

A square grid overlay is also considered by Liu and Yen in designing their Load Balanced Location Based Routing (LB^2R) protocol [42]. LB^2R uses grid cells whose side length is such that 9 of them (the central one and the first tier of 8 cells) are completely contained inside the coverage range of any node placed at the center of the 9 cells. Each grid elects a Grid Header (GH) that is in charge of answering to routing requests and forwarding packets. GHs are rotated whenever the node currently in charge leaves the cell.

Square grids are base to the improved reliability sought by Hornsberger and Shoja's Geographic Grid Routing (GGR) [43]. The protocol divides the network areas in a cell grid centered at the sink. The nodes located nearest to crossing points in the grid are designed to be dissemination nodes (DN) and dynamically replaced if their position varies due to mobility. The sink disseminates queries to the nearest DNs (note that the grid is designed so that DNs are outside the sink's range), through greedy forwarding to intermediate nodes. When the DN adjacent to the cells of interest are reached, they flood the message to the target cells. Inbound nodes are allowed to re-transmit the message only once. Similarly, data are routed back to the sink.

In [44], Taha and Liu apply link reliability measures to traditional greedy relay selection metrics. Their aim is to obtain a globally reliable multi-hop path using local geography-based metrics. The newly proposed hybrid metrics combine MFR, NFP, the distance to destination, and the smallest angle to projection with link reliability. Link reliability is defined here as the probability of correct reception along that link. Moreover, the authors show a convenient way of measuring link reliability which involves the use of beacons. In this approach, each node periodically sends beacons to its one-hop neighbors to disseminate information on its existence and position. The neighbors estimate link reliability by taking the ratio of the numbers of beacons successfully received to the number of beacons sent in a given time interval.

An extension to the Ad hoc On-demand Multipath Distance Vector (AOMDV) approach [45] is obtained by adding some geographic features, such as in Geography-based Ad hoc On-demand Disjoint Multipath (GAODM) [46]. GAODM distinguishes between node-disjoint and edge-disjoint paths, works using route request (RREQ) and route reply (RREP) messages just as AOMDV, with two differences. First, nodes keep a candidate list that is updated by removing the neighbors an RREQ is received from (node-disjoint version). In the edge-disjoint version, the neighbor of the destination forward the RREQ packet only once, and discards any other it receives. Secondly, the choice of the neighbor the RREQ is forwarded to is made according to its distance from the destination (selecting nearer ones). The advantage is a greater number of paths found and lower overhead than AOMDV.

Similarly, [47] explores an extension of AODV that exploits geographic locations of fixed nodes having stable and accurate position information. Such nodes are called anchors. Each anchor broadcasts messages to solicit the association of nodes to the anchor. If a node receives multiple solicitations from different anchors it is called a bridge. Each anchor and all associated nodes and bridges form a cell. When a packet need to be routed inside the same cell or among different cells, proper AODV-like route request and reply messages are broadcast. In particular, when going outside a cell the cell nearest to the destination cell is preferred for routing. A location service is required to know at least the cell that the destination node is part of. The improvements over AODV are greater for larger networks.

Multicast extensions that exploit energy and mobility information is proposed by Liang and Ren in [48]. Their protocol is dubbed Energy-Aware Geographic Multicast Routing (EM-GMR). Based on this consideration, the remaining battery capacity, the mobility, and the distance to the destination node of eligible relays within coverage range are taken into account for the selection of the next hop. A fuzzy-logic system is applied to routing decision-making. The results indicate that the proposed protocol achieves up to 40% more network lifetime compared to geographic solutions that account for the distance to the destination only.

3.9.3 Other Approaches

Park and Shin design and evaluate a holistic routing approach in [49]. Their protocol is motivated by the need to bring together location-update schemes with mobility considerations and energy awareness, in order to devise the optimal tradeoffs for distance- or time-based updates, that minimizes the routing overhead. The approach used is a well-known differentiation between proactive updates within a given local region and reactive queries to distributed location services. To this end, route search, discovery and maintenance primitives are provided. A general random mobility model is taken into account and studied, seeking the relevant metrics for the assessment of routing overhead. Simulated results for such overhead are in very good match with analytical predictions. The optimal location update policy devised in the paper is shown to offer a comparable, almost equal route discovery success ratio, with respect to overly reactive and proactive protocols, but with a lower, slowly increasing overhead.

Subramanian and Shakkottai proposed in [50], some solution to enforce geographic routing where nodes have either imprecise GPS systems, or when only imprecise information on the position of the recipient is available, or even when a fraction of the nodes have routing information, and need to resort to random forwarding to one of their neighbors. GPS impreciseness is modeled through an angular location offset, whereas by imprecise (or coarse) destination information, the authors mean that only quadrant or half-plane position knowledge is available. In any of the three cases, it is proved that the maximum delay to the destination is kept within a constant factor with respect to straight line greedy routing. The analysis is asymptotic in the number of nodes and is made using a continuum model, so that simulation results are supplied for discrete networks.

3.9.4 Contention-based Geographic Forwarding

Part of the literature on geographic forwarding focuses on the choice of a relay that best suites a given metric, among a set of neighbors that are almost equivalent from the point of view of routing. Contention-Based Forwarding (CBF), for example, makes use of timers to select the next hop. Three strategies are considered. With the first strategy, the node whose timer expires first forwards the packet. With the second strategy, the nodes allowed into the contention are only those located in a restricted area. The Releaux triangle is chosen as such area, because of its property that all nodes inside the triangle can hear each other. The third strategy issues control messages before data transmissions in order to identify all available relays and choose the most convenient. Similar to CBF, Implicit Geographic Forwarding (IGF) [51] has the relays set their response timer depending on the provided advancement and on the residual energy at the nodes.

Chen *et al.* pursued a statistical comparison of the selected forwarding area to use in a contention-based geographic forwarding [52]. The study assumes a connected network without voids, and shows when and why using the maximum forwarding area (as in GeRaF

[4]) is better or worse than resorting to the maximum communication area (where all nodes can hear each other), such as used in CBF [53] or a 60-degree sector as in IGF [51]. All three areas are compared also in a scenario with voids, using passive exploration (a node realizing to be a dead end disconnects itself from the network) as well as active exploration (gradual increase in the transmission range until a suitable relay is found).[13]

Blind Geographic Routing (BGR) [54] introduces a framework that encompasses the operation of BLR, CBF and IGF, using different timer specifications. With BGR, the forwarder initiates the process by blindly broadcasting the packet and setting a timeout timer. Upon transmission, a certain area is specified where forwarders are supposed to be located. Any relay that receives the packet waits a random backoff time before propagating it further. If a relay overhears a further packet forwarding by one of its neighbors, it restrains from further transmissions and drops the packet. If the timeout interval should expire, the chosen forwarding area is assumed to be empty, and a retransmission procedure is initiated with a different forwarding area (rotated of 60° with respect to that used for the failed attempt). The number of hops walked by the packet are included in the header, so that in case of almost simultaneous forwarding by two or more different neighboring relays, the overhearing of one another's transmission by each relay is not taken as a further forwarding hop and does not cause the packet to be dropped prematurely.

In [7], Seada *et al.* evaluate the geographic routing approach by bringing lossy radio links into the picture. First, it is argued that neither pure advancement nor pure link reliability are the best metrics to perform routing under imperfect channel conditions, even in a simple chain topology. This simple kind of topology is then analyzed, highlighting that the product of the link reliability (in terms of probability of success over that link) times the achieved advancement is the best choice if ARQ is used. After this analysis, some strategies to choose the best relay among the neighbors are quantitatively evaluated using simulations. Specifically, besides the advancement-times-reliability metric, maximum advancement, maximum reliability, maximum advancement with minimum prescribed reliability (and vice-versa) are used as decision policies. The devised metric, anyway, provides the best performance, both under a delivery delay and energy efficiency point of view.

Lee *et al.* also consider efficient geographic routing using diverse link status measures and routing metrics [8]. They account for the existence of grey propagation zones, where packet delivery is guaranteed only with a certain probability, and propose the selection of the forwarder providing the greatest normalized advancement (NADV), defined by the ratio between the advancement and the link cost. They also provide some methods to estimate link costs (error probability, delay, or power consumption) using probe messages or SNR

[13]It should be noted that this paper contains a comment on the absence of stateless schemes for routing around holes when using geographic protocols. ALBA–R is indeed such a scheme. The fact that it has been proposed so close to this paper adds to the novelty and importance of this topic. PAGER, described later, is also a stateless scheme for routing around holes.

values. These schemes are then extensively studied through simulations, showing the ability of NADV to approach ideal relay selection.

Along the same line, Li *et al.* bring power control into the picture [9], and show that i) the power control that achieves a certain constant SNR is the best choice, ii) that this value of the SNR is inside the so-called transitional region, i.e., a region in the SNR-probability of error space where the probability of error is not close to either 1 or 0, and iii) that routing based on a completely localized metric gives good results in distributed networks. More specifically, this metric is designed to be the ratio of the packet reception rate times the advancement (as in [7]), and leads to the Joint Distributed Routing and Power Control (JDRPC) algorithm. One of the assumptions, anyway, is that the network is sufficiently dense, so that no geographic routing failure occurs. An analytical framework based on a linear network is deployed, to obtain the value of the best hop distance to travel, given that power control is used. Simulation results compare JDPRC with the Dijkstra algorithm (with routes changed dynamically with time-varying link conditions), and hop-by-hop static routing (where the focus is instead on the long-term behavior of wireless links). JDPRC shows satisfactory performance in both the case of a 1-D and a 2-D network.

Savidge *et al.* recently proposed and evaluated differently weighed cost metrics in the context of image sensor networks [10]. The assumption here is that besides a typical floor of periodically reported sensings, some event triggers the generation of high-rate data (e.g., images of some environment), which has to be delivered to a sink with priority. In general, they assume a weighed cost metric accounting for geographic position (through distance, or through the angle from the destination's direction), queue level and energy spares. All metrics are accounted for in such a way that wanted values (small angles, low queue levels and high energy) lead to a significantly smaller cost that non-desirable ones. The evaluation carried out considers different combination of the weights assigned to these three parameters, and show that combined metrics accounting for all three components perform better, and offer even some robustness in case some node is unlocalized.

SPEED, by He *et al.* [55], is a routing protocol which partially inspired the work in [10]. SPEED is aimed at delay minimization rather than QoS support through one-hop metrics. It defines two subsets of nodes among those offering positive advancement (which is the only geographic component, indeed): those exhibiting a sufficiently high movement speed and those who cannot meet this constraint. The relay selection is made among the nodes in the first set. The optimization sought involves delay reduction for packets traveling along the network (to and from a base station) as compared to locally transmitted packets. This way, no support is offered for differentiated QoS. Moreover, delay measurements are made based on the previous delay history, therefore focusing on past rather than on more recent metrics such as queue length [10].

As a side topic, Shah *et al.* compare geographic routing to opportunistic routing in [56], which is basically, choosing the relay that offers the best SNR (or some combined metric) among all available forwarders. The comparison is carried out under different conditions, *i.e.*, node densities, traffic generation rates, channel qualities. With respect to geographic routing with duty-cycled nodes, opportunistic routing offers some advantages, but the greater number of hops to be traversed on average limits this performance gain. On the other hand, choosing the best relay increases the probability of a correct reception, implying smaller end-to-end latencies. The advantages of geographic algorithms (less hops) and opportunistic ones (greater link qualities) find a balance in terms of consumed energy, where geographic routing performs at least as well as opportunistic protocols.

3.9.5 Routing over Planar Graphs

Since it first appearance in [29], face routing has received a lot of attention, because it is a natural companion to greedy geographic algorithms. Face routing allows to forward around a hole, whenever one should be present, provided that an underlying structure (a planar graph) is available. A network connectivity graph is defined as planar if it contains no cross–links: as such, it is formed of multiple polygonal faces that share one or more links. Before the introduction of planar routing, the resolution of dead ends was made through breadth- or depth-first algorithms, *e.g.*, polling all neighbors and requesting them to poll their own neighbors, until a node was found that offered some positive advancement. At that point, a message was sent back to the source, carrying the whole route information needed to escape the hole. For example, this approach is considered in [57] and is shown to lead to excessive signaling steps (and therefore overhead) before a route is actually found.

The extraction of a planar graph avoids such expensive route searches. Equivalent to walking one's way out of a maze by keeping the right hand on the wall, one can route over planar graphs using a simple right-hand rule, *i.e.*, always selecting the first link placed counterclockwise with respect to the one the packet came from. This way, it is guaranteed that a packet will cross the hole along its border and eventually reach some node that may continue greedy forwarding.[14]

A known problem with face routing is hop stretch. When applying any planarizer to the network graph, the resulting subgraph is generally composed of shorter links, resulting in an increased average number of hops to reach the destination. The average increase is called stretch factor. This is a known problem that is currently under study, as new techniques are sought to distributedly planarize a network without significant stretch. Some of the works reported hereon treat planarization from this point of view. As three planarizers in particular are currently used in almost every work, or base to a number of extensions, we cite them briefly here. The Gabriel Graph (GG) was the first to be proposed for such a routing [60] (recall that [29] assumed the existence of a planar graph without devising a means to compute it). It works as follows. Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be the connectivity graph, with

[14]In general, letting n be the number of nodes and $M(n)$ the maximum transmission range, if $M(n) = K\sqrt{(\log n)/n}$, straight-line greedy routing is asymptotically almost always possible. This means that for K and n large enough, it is not necessary to resort to any technique for traveling around connectivity holes with high probability [58,59].

$\mathcal{N}$ the set of nodes (or vertices) and $\mathcal{E}$ the set of links among nodes. Assume that the Unit Disk Graph (UDG) hypothesis holds, i.e., a link (i, j) belongs to $\mathcal{E}$ if and only if i and j are within coverage range of each other. Assume also that links are undirected for now. With GG, any node x checks its neighbor list and polls each neighbor for a *witness check*. Let $\mathcal{C}_o^D$ be the circle centered at o with diameter D. Let also y be the neighbor performing the check along with x. The aim is that x and y detect if any common neighbor lies in within $\mathcal{C}_m^{d_{xy}}$, where m is the medial point of segment $\overline{xy}$ and d_{xy} is the Euclidean distance among the two points. If any such neighbor (a witness) is detected, the link (x, y) is removed from $\mathcal{E}$. The Random Neighboring Graph (RNG) works similarly, but looks for witnesses in the intersection of $\mathcal{C}_x^{2d_{xy}}$ and $\mathcal{C}_y^{2d_{xy}}$. By considering a larger area where witnesses can be found, the RNG returns in general a sparser graph than GG, resulting in a smaller hop stretch. In particular, Bose *et al.* showed in [61] that the stretch factor at constant density of nodes is $\Theta(n)$ for RNG and $\Theta(\sqrt{n})$ for GG. A third algorithm, the Restricted Delaunay Graph (RDG), based on Delaunay triangulation, also finds some use. The resulting graph has the property that for all pairs of nodes x and y and for any witness w, the circumcircle of the triangle uvw does not contain any other node in $\mathcal{N}$. Moreover, RDGs have the desirable property that their stretch factor is bounded. A planarization algorithm exhibiting such a property is called a "spanner."

If a planar graph is available, the main problem is finding an efficient technique to traverse it. Efficient means that the traversal should not get stuck at dead ends, yet it should not provide an excess route stretch, *i.e.*, an increase in the number of hops to reach the destination. Nevertheless, if a planar graph is not available one has to resort to more cumbersome ways of traversing it, such as detailed in [57].

Among the first attempts to route on a planar graph, the aforementioned paper by Kranakis *et al.* [29] describes Compass Routing II, that proposes to walk a planar graph by traversing the boundary of all faces that are intersected by the source-destination segment. When walking the face, a rule is needed to decide where to continue the exploration. In particular, the right-hand rule is suggested. It consists on taking the face boundary to the immediate left with respect to the direction walked when entering the face. Bose *et al.* propose two similar algorithms, Face and Face-II in [60]. All approaches seen so far rely only on planar graph traversal unlike Karp and Kung's Greedy Perimeter Stateless Routing (GPSR) [21]. Their protocol performs potentially better than all face-routing-only algorithms, because it contains a greedy component. In more detail, GPSR forwards the packet to the neighbor closer to the destination at each step until either the destination is reached or the current holder is the closest node among its neighbors. In the second case, a dead end is reached and face routing is resorted to. Using the right hand rule, the packet is walked out of the dead end until greedy routing can be resumed. This is done whenever a node is reached that is closer to the destination than the first node using face forwarding.

Other algorithms to perform worst-case optimal face routing were proposed by Kuhn *et al.* [62]. There, they discuss where is the best point to change face during face routing and show that one of their algorithms, Other Adaptive Face Routing (OAFR) bears the worst-case-optimal routing efficiency in face routing. They also augmented OAFR with a Greedy component to yield GOAFR and, later, GOAFR^{+}, that uses an early fallback technique similar to GPRS to convert back to greed routing during face exploration.

Other algorithms have followed that try to optimize the way a planar graph is derived, make this process simpler or find more effective ways to traverse a planar graph while keeping the convenience of greedy routing when possible.

In [63], Gao *et al.* present a spanner that makes use of Delaunay triangulation. Such method is long known to produce good planar graphs, but little is known about the property of RDGs. In a nutshell, the authors devise a clustering algorithm with clusterheads, cluster members and gateways, where the Delaunay graph is computed only withing coverage of a clusterhead. The planar structure involves link from nodes to clusterheads through gateways, and is proved to have constant stretch factor (*i.e.*, the devised algorithm is a spanner). The link with clustering is very tight, because RDG needs constant density graphs to work correctly. The algorithm presented is proved to have a smaller stretch factor than other approaches currently in use. Moreover it allows for a smaller overhead devoted to the planar graph maintenance, by forming less links than, *e.g.*, GG, and by ensuring that network changes notifications (*e.g.*, edge insertions) do not propagate farther than 2 hops away.

Geographic routing is also of interest in sensing-covered networks, where each point in a geographic area must be within sensing range of at least one sensors. In such networks, it is typically assumed that the communication range exceeds the sensing range. In [64], Xing *et al.* show that their proposed protocol, Bounded Voronoi Greedy Forwarding (BVGF) has some desirable properties, in that it is a spanner and provides a constant maximum stretch factor equaling 4.62 as long as the communication range is at least twice the sensing range.

In [65] it is argued that the real coverage of wireless nodes is not a perfect circle. Under the assumption that the maximum coverage range is some value R, and that a minimum range r exists within which any communication is correctly received, the authors deploy a three-phase routing protocol that tolerates a variation of up to $R/r = \sqrt{2}$. The routing protocol is based on a modification of face routing approaches and works as follows. Initially (first phase), every node u creates its own adjacency list, *i.e.*, the list of the nodes it can directly communicate with. In this framework, this means that both their coverage ranges in each other's direction are sufficiently large. Neighbor lists are exchanged, and each neighbor v is then informed of any other neighbor w of u that is farther than r from v (link processing). Any link set up this way is called *virtual*. Note that this phase may require a so-called virtual routing of signaling messages among nodes, which in turn may imply using more than one actual link per message. The second phase consists in the extraction of the Gabriel Graph from the network graph that includes virtual links. Finally (third phase), greedy routing is performed using GEDIR, and switching to perimeter routing on the Gabriel graph upon failures of the greedy approach, using virtual routing to cross virtual links if necessary.

The authors in [66] slightly extend the approach in [65] to improve routing performance.

More specifically, they mean to reduce the number of virtual links set up by the routing protocol in [65]. To this aim, they grow the number of phases to five. The first phase is link collection and implies building and exchanging lists of neighbors as in the aforementioned protocol, with the difference that upon being notified of a new node, the recipient issues a Confirm message, so that the notifier can check whether the link is bidirectional (both neighbors hear each other). The second phase is Gabriel graph construction over the only actual links. Edges are not really deleted but just marked as such. The third phase consists in adding virtual links whose length is larger than r, whereby the graph is enriched with more links between virtual neighbor. The fourth phase basically removes useless and crossing links from this graph. The fifth phase, routing, is performed using some previously devised approach. This variation ensures the same routing performance of [65] with less virtual links.

A complete protocol to address location-based ad hoc routing, namely Terminode, is presented in [67]. Terminode is in fact a more complex suite of solutions that provide support for location management, neighbor discovery, routing around holes and mobility compensation. Terminode works by combining greedy routing, face routing and restricted flooding. Whenever possible, a direct path is used for forwarding a packet. If necessary, the construction and maintenance of a planar graph allows to route around holes. In any case, when the packet arrives up to two hops away from destination, localized flooding allows to propagate the message in a larger area, thereby compensating for possible position changes in mobile scenarios. Terminode also integrates solutions to search for anchors in the network, *i.e.*, nodes that can serve as intermediate waypoints in the forwarding process. Instead of resorting to face routing directly, anchor routing allows for lower stretch factors and an overall better route found. Anchors are discovered using Friend Assisted Path Discovery (FAPD), which is based on long distance "friendship" set and maintained among nodes through proper messaging, or Geographic Maps-based Path Discovery (GMPD), which is more complex as it assumes a-priori knowledge of the node density distribution in the network area, but provides better results. The simulations show that Terminode outperforms LAR and AODV in the considered tests, involving static connected networks as well as high mobility scenarios.

Path Vector Face Routing, PVFR [68] is a face routing approach that gathers and exploits face topology knowledge. Basically, a Path Vector Exchange Protocol (PVEX) lets nodes mutually exchange face information and update which is successor and which predecessor on each face in a planar graph. If complete face information is assumed, an oblivious (memoryless) approach is deployed that, on a given face, selects the edge (and the node on that edge) nearest to the destination and either directly forwards the packet to this nearest node, or traverses the face until this node is reached. In the more realistic case that complete face information is unavailable, routing proceeds first through a greedy approach, which may eventually fail. If the node where the packet is stuck knows that on its faces lies another node nearer to the destination than itself, it forwards the packet to that node, possibly setting up virtual edges to close incomplete faces. Whenever this second routing mode fails, the forwarders resort to simple face traversal through right-hand rule to reach virtually connected nodes. The results show that satisfactory routing performance is achieved with limited face information, and suggest that even if GOAFR+ [62] is asymptotically optimal and bounds worst-case performance with expanding ellipse search, PVFR generally achieves the best average case stretch.

Face-Aware Routing (FAR) [69] is designed to provide mobicast delivery using a just-in-time approach. The idea at the base is simple: when a packet is forwarded that has some timing requirement, adjusting the forwarding speed so that delivery happens neither late nor in excessive advance provides some advantages, mainly in terms of memory occupancy. This concept is applied to mobicast scenarios, where a moving object has to be provided data, and the delivery zone for data moves and changes in time, so that older data may become out,dated if transmitted too late. A moving inquirer thus describes an interest zone. Then, FAR basically works as follows. It distinguishes among greedy and timed forwarding, the first being used for delivery to spatial neighbors, the second used by intermediate nodes whose neighbors will become part of the delivery zone in the future. Spatial neighbors are defined as those belonging to the faces the forwarder is part of. Timed forwarding uses a metric involving the expected transmission latency and the expected time for delivery zone to reach the neighbors of the relay, in order to decide whether to transmit a packet at once or to wait before sending. This implements a just-in-time policy. It is proved that FAR guarantees delivery if the delivery zone perpendicular span (the extension in the direction perpendicular to the velocity vector) is greater than the nodes communication range. The main protocol overhead is represented by neighbor discovery costs and by the messages required to maintain the planar structure (including the detection of spatial neighbors).

In [22] Fang, Gao and Guibas attack the problem of detecting connectivity holes in geographic routing through a theory involving Delaunay triangulation. With some simple conditions on the position of a node p with respect to any pair of neighbors u and v, it is possible to understand whether a packet routed to p will remain stuck there, depending on the direction the packet has to travel toward. In more detail, drawing the perpendicular bisector to the segments joining p to u and v, the center of the circumference passing through the locations of the three nodes is determined. If this point lies outside the coverage range of node p, there exists a zone where any node is unreachable from p using greedy forwarding. Running this test locally allows any node to understand if (and toward which locations) he may be a dead routing end. This algorithm is called the Tent rule. The Boundhole algorithm is used to recover from a stuck node, and works by first identifying the border nodes of a hole as follows. Starting from the stuck node (which self-identifies as such using Tent), each node locally scans the hole ahead counterclockwise, until a neighbor is located, and then forwards a hole-detection packet to it. The hole scan begins from a direction that does not include a forbidden area, defined so that the packet is not sent to previous neighboring forwarders. An efficient implementation of the algorithm is possible. Since hole-detection packets may be forwarded by any node, a suppressed start feature can be enabled, that lets nodes on the boundary of the hole discard any detection packet received by predecessors on the boundary. Moreover, nodes can periodically broadcast “heartbeats” that allow to preemptively cope with node failures. Hole detection is than usable for routing (*e.g.*, selecting the shorter path to progress around the hole), identifying regions of interest (*e.g.*, where nodes have been burnt by spreading fire), or supporting path migration, *i.e.*, dynamic routing path changes in application that track the movements of a drifting object.

Using the Tent and Boundhole algorithms [22], Fayed and Mouftah measured the size and incidence of routing holes [70], observing that doubling the number of nodes in a networks reduces the occurrence of routing holes per node by one order of magnitude. Furthermore, they observe that the majority of routing holes in all networks could be mapped by Boundhole in 10 hops or less, and circumvented in 4 hops or less.

The LDT as a spanner is also considered by He *et al.* in [71].Their protocol, Greedy and Local Neighbor Face Routing (GLNFR), lets node collect information about all nodes belonging to their adjacent faces and share this information with neighbors. The protocol has three operating modes, namely pure greedy forwarding, local face information mode and perimeter mode. The first one is the default mode. If greedy forwarding is impossible, the node spans its local face knowledge to look for a node offering advancement toward the destination. If this also fails, the packet is routed along faces using a right-hand rule.

The names “geogram” and “geocircuits” are introduced by Fotopoulou-Prigipa and MacDonald in [72]. The two terms address geographic forwarding of single packets, independently of one another, as opposed to routing using previously established paths. In [73], the same authors apply this concept to a larger extent and use it for efficient recovery from already resolved routing holes. Their protocol, namely Geographic virtual Circuit Routing Protocol (GCRP), first obtains the location of the destination, then takes forwarding decisions based on the minimization of the Euclidean distance of the relays from the destination. Dead ends are first resolved using a depth-first neighbor search, with limited depth and with the additional clause that neighbors are visited no more than once, in order of ascending distance from the destination. This method is aimed at reducing the otherwise lengthy depth-first search process. Resolved path to detour from dead ends are then cached at stuck nodes, and periodically updated to account for network dynamics.

Trajectory-based forwarding [74] is an approach mainly viable for dense networks, where it is possible to describe a certain trajectory along which the packet should be forwarded and, at each step, select the node along this trajectory that best fits a given metric (least distance from the trajectory, most forward within radius, or even most battery power left). In mobility scenarios, one could also choose nodes that promise to forward along the trajectory. Trajectory encoding is an issue. If straight lines are simple to parameterize, more complex curves would require more coefficients to be encoded. Furthermore, if multicast trees are set up which do not show any symmetry property, it could become necessary to encode the whole tree to instruct the forwarders, a weighty task. The approach is interesting nonetheless, because results show that it is robust to location errors, and makes it easy to encode multiple or braided paths in the same packet. On the other hand, trajectory routing in sparse networks would require the senders to have at least a rough estimate of the size of obstacles or voids. The protocol switches to face routing whenever greedy forwarding is not possible.

A Geographic Power Efficient Routing (GPER) protocol [75] is proposed by Wu and Candan to exploit node density whenever possible to reduce energy consumption. Within radio range, a node tries to reach the destination using intermediate neighbors offering better conditions, in terms of energy required for reliable communication. Each intermediate node is allowed to change the route again using other nodes in between. An energy model with a constant energy due to processing is accounted for, so that the energy required to ensure a target coverage range becomes significant only after a certain distance threshold. This is accounted for in the routing process. A forced routing primitive that inhibits intermediate node forwarding by nearby relays is set whenever a loop is detected. The whole network graph is planarized to yield robustness against voids if necessary. Routing over a grid overlay is also proposed to concentrate routes in the zones with a larger number of sensors, in order to yield greater energy savings.

3.9.6 Observations on Planar Spanners

Extracting and maintaining a planar graph is perhaps one the most demanding operations required by face routing. The first to propose to route over planar graph links to recover from dead ends, Kranakis *et al.* [29], assumed such a graph existed and did not propose an extraction method. Within the Face framework, Bose *et al.* proposed GG as a planar spanner [60], whereas Karp and Kung suggested the use of RNG for GPSR [21]. There exist long-known algorithms that allow to distributedly compute both GG [76] and RNG [77]. Nonetheless, such algorithms require heavy message exchanges among nodes, and the efficient extraction of planar graphs is still an open issue, as highlighted in [78]. Moreover, the approaches in [76] and [77] assume that the connectivity model complies with the Unit Disk Graph assumption, which is never the case in a real network. From this point of view, the works by Barrière *et al.* [65] (and following extensions) is a significant step. The introduction of Quasi-UDGs and the concept of virtual links to augment the graph as needed provides a means for coping with irregular reception patterns, typical of radio communications.

The first planarization experiment known to work on real networks has been recently performed by Kim *et al.* [79]. They propose the Cross–Link Detection Protocol (CLDP), a simple planarization algorithm that counterclockwise polls each available link (using a right-hand rule) and tries to detect whether some links cross each other. A distributed mechanism is devised to remove crossing links without damaging global connectivity. CLDP works on real networks, but requires a lot of overhead to maintain a planar graph. For such reasons, part of the research efforts on geographic routing are devoted to finding adequate alternatives to planarization, as reported in the following section.

A further evaluation of planarizing algorithms under more realistic propagation conditions is provided by Kim *et al.* in [80]. They show that several conditions (obstacles, collinear links, etc.) may impair planar graph formation, and possibly lead to the creation of unidirectional links. They also show that the mutual witness extensions and the CLDP [79] cannot perform correctly in every environment. They suggest some improvements to face exploration algorithms that could help in dealing with realistic environments, and simulate some face routing algorithm (best intersection, first intersection, OFR) as well as greedy routing and face routing combined (GPSR, GOFR+).

Frey and Stojmenovic [81] argue on the hypotheses behind the claim that face and combined greedy-face routing cannot guarantee delivery on arbitrary graphs. They give a survey and classification of currently available face and greedy-face routing approaches and are the first to provide some formal proofs about the correctness of such algorithms over different kinds of network graphs (GG, RNG, LDT and arbitrary).

3.9.7 Studies on geographic routing in the presence of location errors

Few studies are available that try to address at least to some extent the effect of location error on one of the instances of geographic routing. Among these, the following ones provide interesting results and are worth mentioning in light of the previous discussion.

Seada *et al.* are the first to provide a systematic study of the effects of localization errors on geographic routing for WSNs [82]. They focus on dead end recovery through planar graph traversal, and show that location errors could impair planar spanners (such as RNG and GG). Such failures are represented by loops created in the planar structure, or not removed cross-links, which in turn would lead to non-planar graphs. They also show the effects of realistic radio coverage, having non-circular shapes. Typically, this means that the network graph may no longer be undirected, or in other words, some nodes perceive some neighbors which cannot receive anything from them. The consequences of such a case are also face routing failures. The fix proposed for planar spanners is *mutual witnessing*. In a nutshell, all algorithms that extract planar connectivity graphs are based on having pairs of nodes check if there is any third node, called *witness*, inside a given portion of the coverage area. If so, the link among the inquiring pair is removed, favoring a two-hop link through the witness. Mutual witnessing consists in making sure that any witness is a neighbor of both inquiring nodes. This is shown to yield more robustness against non-ideal coverage and localization errors, but would require additional signaling.

Similar to the concept of mutual witnessing is the concept of two/hop knowledge proposed by Shah *et al.* in [83]. In this paper, they provide an analytical performance evaluation of the impact of location errors. As a baseline, they start from the protocol in [84], which performs greedy routing with an adjoint flooding phase when packets get stuck at dead ends.

They define a basic behavior (the protocol in [84] without flooding), a flooding protocol (the full approach in [84]) and the so-called second-order routing, which exploits the knowledge of second order neighbors, *i.e.*, the neighbors of the neighbors, and their respective locations. In more detail, this third version still looks for neighbors offering maximum advancement as next hops, but constrained to having a second-order neighbor which is still closer to destination (to reduce the chance a packet gets stuck somewhere). Without second order knowledge, greedy routing is shown to offer only degraded performance (depending on nodal density) for as much as a 20% positioning error with respect to the radio range. With the proposed enhancement, as much as a 40% error can be tolerated. The second-order routing is also shown to perform well in presence of physical obstacles.

3.9.8 Routing over Virtual Coordinates: an Alternative to Planarization

The use of virtual coordinates was initially conceived to provide all nodes with some sort of position information, when none or only some of them have location estimates available. This idea was first proposed by Rao *et al.* in [85]. More recently, the use of virtual coordinates has been devised as a means of enforcing greedy routing. The idea is simple: since geographic algorithms work best when greedy forwarding lasts as long as possible [64], virtual coordinates could be used not to mimic geographic ones, but to provide a warped network topology, where switching to costly dead-end recovery is less frequent. Another argument supporting virtual coordinates is that it is not required to update them upon changes in the real locations, if the connectivity remains the same, therefore enabling greater robustness to mobility.

In [85], basically, nodes are split into perimeter and internal nodes. Perimeter nodes are assumed to know their location. If this is not the case, some internal nodes are randomly elected as temporary landmarks and start broadcasting beacons, so that the nodes on the nodes may estimate to be on the perimeter through detecting maximum distance from the landmarks. After this step, perimeter nodes estimate their own position using an error-minimizing algorithm. Once perimeter nodes are provided a form of location, they flood the network with their coordinates. In subsequent steps, all internal nodes compute their location as the average of their neighbors' locations, and rebroadcast this new estimate. After a number of steps, the coordinates converge to values that allow for a forwarding success ratio of nearly 100%. This approach is also shown to work well in more difficult topologies including holes and obstacles. Nonetheless, the model accounted for in the evaluation is very simple, and does not consider, e.g., the problems incurred during flooding procedures, such as packet collisions and other sources of packet loss.

Following the interest for routing without location information, Jadbabaie [86] demonstrates some mathematical and geometrical property of the algorithm in [85]. Assuming again that the nodes on the perimeter of the network know their own location, it finds a closed form for the coordinates of all other nodes, which depends on the adjacency[15] and valence[16] matrices of a node, and on the number of peripheral nodes connected to internal nodes. In more detail, each coordinate pair computed at an internal node is proved to be a convex combination of the coordinates of perimeter nodes. Furthermore, through a mechanical systems interpretation of the results, it is claimed that the notion of "distance" in the virtual coordinates domain is different than the common concept of Euclidean distance, since virtual coordinates embed some information about obstacles or connectivity holes in the network. Therefore, geographic routing using virtual coordinates may prove to be in fact easier and less prone to disconnection than routing under the usual Euclidean distance definition.

In [87] Cao and Abdelzaher present a protocol that performs distance-based routing over logical coordinates. Some landmark nodes are supposed to exist somewhere in the network and propagate hop count information. The coordinate vector of a node is defined as the ordered tuple containing hop count distance from all landmarks. This approach allows to encode some amount of topology and connectivity information directly into logical coordinates, because the number of hops to a given landmark accounts for the need to step around holes. The basic routing protocol lets nodes forward the packet to the neighbor offering the minimum logical Euclidean distance from the destination. To avoid possible looping among neighbors (due to the coarseness of logical coordinates), a tabulist is implemented that records the last forwarding steps. Voids are countered by sending the packet back (in the logical sense) up to a certain number of steps, until a node offering progress is encountered. The paper contains also clues on the kind of distance definition and landmark number (up to 4–6 for better performance) and position (as sparse as possible).

Gradient landmark-based distributed routing (GLIDER), by Fang *et al.*, is also based on landmarks for routing [88]. The aim is to build a virtual topology where coordinates are not centered on landmarks, enabling gradient descent algorithms to work well as greedy approaches. GLIDER selects a set of nodes to operate as landmarks and has other nodes distributedly compute the so-called landmark Voronoi complex and the combinatorial Delaunay triangulation over the set of landmarks. This allows to partition the nodes in sets, or tiles, which may also be partially overlapping. Routing is performed through inter- and intra-tiles algorithms, after some algorithm is applied (e.g., a local spanning tree) that allows to understand the sequence of tiles to traverse. Inter-tile routing (from a tile to the subsequent one) is performed by selecting the neighbors that provide advancement toward the landmark of the following tile, whereas intra-tile routing is performed through minimizing Euclidean distance in the virtual coordinates. Local tile flooding is used in case a local minima is reached.

[15]The adjacency matrix of a graph $\mathcal{G}$ is defined as the matrix whose element in position (i, j) is 1 if the vertices i and j in $\mathcal{G}$ are connected, 0 otherwise.

[16]The valence matrix is a diagonal matrix where the element (i, i) is the valence of the i-th node in $\mathcal{G}$, i.e., the number of nodes adjacent to i.

Vivaldi, by Dabek *et al.*, is another decentralized approach based on virtual coordinates [89], mainly thought to operate for Internet round-trip time estimation. It uses a spring relaxation algorithm that lets nodes update their coordinates based on a calculation of a sort of spring forces acting toward equilibrium, and embeds an adaptive damping factor that helps the algorithm converge more rapidly. To resolve particularly complex topologies, the algorithm embeds a height coordinate that is used to model the time it takes to a packet to travel from a node to the core network. The algorithm is shown to converge well even in case of network topology. The drawback is that, even in the presence of the dampening factor, the algorithm requires heavy message exchange before virtual coordinates converge to the wanted values. This could be feasible in a wired network, but impractical in a wireless one.

Beacon Vector Routing (BVR), was presented in [90] by Fonseca *et al.* to be a scalable protocol for wireless sensor networks based on landmark routing. It defines a rule that lets nodes automatically elect themselves as landmarks and propagate their beacon. As in [87], all other nodes use the hop vector as a position measure and greedily route based on Euclidean distances in the hop count space. Local dead ends are resolved through restricted flooding. The main drawback of this protocol consists in the fact that many landmarks may be present, requiring large beacons to be propagated for routing, with increased overhead.

The Virtual Coordinate assignment protocol (VCap) by Caruso *et al.* [91] is affected by the opposite problem. It uses only three landmarks that are determined in such a way that their physical location are well apart. The associated overheads are very small, but on the other hand, there are set of nodes named *zones*, that share the same coordinates, requiring that routing based on hop ID is performed inside each zone. Moreover, due to the very small number of landmarks, the protocol performs extremely poorly on sparse networks.

Recently, Zhao *et al.* proposed Hop ID-based routing (HIR) [92]. Similar to previous algorithms, HIR performs a random landmark selection initiated by a coordinator node C. Each node hashes some unique characteristic such as its IP address into a landmark index and self-elect itself a landmark if its ID is sufficiently close to this index. Each landmark starts broadcasting its own beacons, so that all other nodes can measure their hop count with respect to all landmarks. The update of hop counts for mobile nodes is performed using HELLO messages to avoid flooding beacons constantly. As a baseline, HIR routes data using a greedy approach on the Euclidean distance among hop count vectors, and uses landmarks as safety destination when a dead end needs be resolved. The most noticeable drawbacks of HIR are found in the number of landmarks needed (and the consequent size of HELLO messages) and in the network density (which must be sufficiently high as in all virtual coordinates approaches).

Another algorithm to distributedly compute virtual coordinates, Greedy Embedding Spring Coordinates (GSpring), has been recently proposed in [93]. The base idea is that for greedy routing to work, the *region of ownership* (the geometrical region containing all points closer to that node than to any other) of a node s should not contain any other neighbor t. If this should happen, the best choice would be to have s be repelled by t until t gets out of the region of ownership. For this purpose,the coordinates are updated using spring relaxation as in [89]. If nodes are found inside the region of ownership, the rule changes slightly to enforce repulsion from these nodes. The initial coordinates that have to be broadcast for GSpring to unfold the network topology are either the true geographical coordinates of a subset of nodes (if available) or other values guessed through a hop count algorithm such as in [85].

3.9.9 Other Alternatives to Planar Graph Traversal

In [94], Yu Xu and Li Jun propose a somehow different technique for routing in mobile networks. They introduce the concept of virtual destination, as a means of recovering from connectivity holes. The proposed protocol can operate according to two modes: i) Greedy forwarding to real destination (GFR) lets nodes make forwarding decision based on the location of the final destination using a greedy scheme; ii) Greedy forwarding to virtual destination (GFV): in this mode, the forwarding decisions are made according to the same greedy scheme, but based on the location of some computed virtual destination, which is not a real node in the network. When greedy forwarding to the real destination fails, the packet is supposed to have reached a void. From that point on, GFV is used to recovery from the void. The approach is different from face routing, in that it does not require to maintain a planar network graph. The virtual destination is chosen so that it has an Euclidean distance of $2R$ from the current forwarder (where R is transmission range of the node), and as close as possible to the destination direction. The results also show that the proposed protocol overcomes GPSR [21] in terms of average number of hops in connected ad hoc networks.

PAGER [95] by Zou *et al.* is a protocol meant to reduce the strong overhead required for maintaining planar graphs updated. PAGER divides the network graph into two subgraphs, namely made of bright and shadow nodes. Bright nodes have a direct (greedy) path toward the destination, whereas dark nodes do not, and have to route through other dark neighbors until a bright node is found. The identification of shadow nodes is performed by having each node compute its Euclidean distance to the destination and then run an algorithm aimed at avoiding being surrounded by nodes with larger distances only. To this aim, a node increases its own distance by a predefined factor Δ and re-broadcasts it, until all shadow nodes find a way through the shadow zone. PAGER is shown (by simulation) to gain over GPSR and AODV in terms of packet delivery ratio, end-to-end delay and consumed energy, while maintaining routes whose length is close to optimum, *i.e.*, it does not suffer from excess route stretch.

Geographic routing in outdoor downtown city scenarios poses additional difficulties, mainly due to buildings causing connectivity interruptions. In [96], Lochert *et al.* show that classic routing approaches may suffer from such a configuration, for example when a packet is forwarded beyond a street junction, no other chance may be available to route around a building, and the packet may be forwarded to dead ends where greedy approaches fail and require some sort of recovery step. The proposed solution involves the use of coordinator nodes, which are elected among those placed at street junctions. To detect coordinators, any node sends beacons containing a list of neighbors, is deemed a coordinator if some nodes on the list do not see each other as a neighbor. Otherwise, a node may calculate the statistical correlation coefficient with respect to the position of all neighbors: nodes located on a street can expect to have a strongly linear relationship among coordinates, thus a coefficient close to 1. This approach is shown to offer a significant increase in the packet delivery ratio with respect to standard GPSR [21].

Greedy Distributed Spanning Tree Routing (GDSTR) is a greedy geographic algorithm that resorts to spanning tree routing upon encountering voids [93]. The idea of using distributed spanning trees is not new, *e.g.*, [97] suggested the use of distributedly computed spanning trees in ad hoc networking. There, distributed trees served to highly dynamic networks, where connectivity conditions could rapidly change. Routing over such trees is performed either through hybrid flooding (achieved through flooding the message to all leaves) or tree shuttling (*i.e.*, sending the packet up the tree up to a certain level and then letting it descend to the correct leaf. In GDSTR, spanning trees are instead a sort of super-structure helping routing. Instead of using trees for mere next-hop decisions, GDSTR lets nodes store distributed hulls, *i.e.*, convex sets containing all leaf nodes (or hulls) which descend from the node. A tradeoff is present between the accurate representation of hulls and the state space memory requested at each node, in that relaxing the hull representation may cause the hull to include additional nodes that are reachable instead through other tree branches. If a node cannot be reached through a given tree branch, then a recovery is simply administered by routing through the other branch. The overhead inherent with this approach lies in the need for building a spanning tree (minimal-depth search is suggested to have the most compact and non-intersecting hulls). Moreover, it is assumed that the destination location is known, *e.g.*, through a location service, and that nodes agree on which are their neighbors. This could potentially require more signaling among nodes [82].

3.9.10 Geocasting

Geocasting is a communication paradigm whereby all nodes in an area of interest have to be reached by a certain communication, *e.g.*, by a set of packets generated somewhere else. The term was first introduced in [98, 99] in the context of Internet routing, and has since expanded to wireless networks. Geocasting using packet radio communications poses a series of problems and advantages. On one hand it restricts the area where a message has to be propagated, thereby reducing message overhead (*e.g.*, if compared to pure flooding); on the other hand it requires recovery mechanisms to cope with possible disconnections inside the target region.

Following the lesson learned with LAR [31], Ko and Vaidya propose two versions of a region-based geocasting protocol [100]. Their work is based on the definition of a (rectangular) geocast zone contained within a rectangular forwarding zone.This last region defines where the packet is allowed to be propagated. The forwarding region may be either fixed or adaptive, *i.e.*, its dimension may be reduced by forwarders, in order to decrease the message flooding overhead. A second version, namely Adaptive Distance, requires each intermediate node to check whether they are nearer to the geocast zone than the packet sender before forwarding the packet again.

A further attempt in overhead reduction is aimed at with GeoTORA [101]. This 1protocol is based on an anycast extension of the unicast routing protocol TORA [102]. TORA is a so-called link reversal algorithms, that maintains a destination-oriented acyclic graph at each node. Whenever a link is broken, *e.g.*, by a moving node, the directed graph is updated by reversing links, so that unreachable destinations become reachable again through different routes. GeoTORA defines regions (possibly made of multiple nodes) instead of destinations. Directed graphs are maintained so that any node inside the region may is reached through at least one link. When a geocast packet enters the destination region, flooding is used to reach other nodes inside the region.

In [103] Seada and Helmy propose Geographic-Forwarding-Geocast (GFG), a protocol with close-to-minimum overhead but no guaranteed delivery, and a second protocol with Perimeter routing extensions (GFPG), which provides guaranteed delivery at the price of a much larger overhead. GFG uses a greedy geographic protocol (perhaps with face routing for recovery from holes, such as GPSR) to forward toward the center of a prescribed geocast region, until a node inside the region is reached. From there, the whole region is flooded with the packet. In dense networks with no holes or obstacles, GFG suffices to reach any node inside the region. In case of connectivity gaps inside the region, GFG fails to geocast the packet. GFPG is introduced for this reason. It resorts to perimeter routing around the region borders whenever direct flooding fails. More specifically, border nodes (nodes inside the region having neighbors outside), if unable to flood the packet, send it outside the region to provide further advancement. The packet is then routed along the region borders, until a node is eventually found inside. From then, flooding can begin again. Since GFPG exhibits a very large overhead especially in dense networks, a practical version, GFPG*, lets all nodes inside the geocast region divide their coverage range in four areas (e.g., spanning 90° each). If no region is empty, the packet is broadcast. Otherwise, a perimeter-mode packet is sent to the neighbors in the first region counterclockwise from the empty portion. Even in sparse networks, GFPG* guarantees the delivery of almost all packets, but with as much as three times the overhead required by GFG, which in turns delivers a smaller 80% of the packets only.

As far as broadcasting to the whole network is concerned, Chen and Welch proposed a location-based scheme in [104]. The whole scheme is beaconless, and based on the retransmission of messages along a series of location-based paths, *i.e.*, a sequence of locations determined over the network area. A square grid overlay perfectly fitting the area is considered, and a virtual graph is built that connects all the centers of neighboring non-empty grid cells. Cells are covered by broadcast in a depth-first order. A scheduling approach is also considered that ensures only one node per cell to be forwarding at any time.

3.10 Summary and Conclusions

In this Chapter, we have shown and developed an example of cross–layer design for wireless sensor networks. We started from GeRaF, an example of forwarding protocol where routing and MAC operations are joined together to help in the choice of the relay. Such a choice is made quicker and more effective by the cross–layer design. We optimized the relay search rules in GeRaF and the criteria for participating to contentions and showed through analysis and simulation that this new version performs better than the originally proposed one. We have argued that the multihop performance of GeRaF might be limited if the choice of the next hop is based only on advancement, and introduced a new two-fold search, where the forwarding capabilities of the relay are given greater importance. This gave rise to the Adaptive Load-Balanced Algorithm, ALBA. We have integrated ALBA with Rainbow, a mechanism to backtrack packets out of dead ends, that constitute a significant problem when using geographic forwarding. The resulting protocol, ALBA–R, proved to improve the packet delivery ratio significantly, up to 100% when no discarding is performed due to failed forwarding attempts. Finally, we compared ALBA to MACRO, a cross–layer forwarding scheme for WSNs designed with the further objective of minimizing energy consumption. Using a more realistic energy consumption model, we have shown that the accurate design that led to ALBA leads to the consumption of even less energy, overall. This was possible especially thanks to the lighter relay search phase.

The main conclusion that can be drawn from this study is that cross–layer design of protocols for wireless networks can prove to be very effective for increasing performance. Wireless networks, especially WSNs, bear much greater limitations with respect to wired networks. When limited energy consumption and increased lifetime must be pursued, the old ISO/OSI paradigm that wanted different network functions separated needs to be revisited. That paradigm was introduced for the sake of higher efficiency, easier plug-in of new functions and faster upgrades when a new technology was made available for a certain layer. In wireless networks, the stringent constraints make joint protocol design a much more efficient paradigm. Cross–layer design may speed up operations and consume less energy by blending two or more network functions into one layer. In tiny devices such as sensors, this boosts the lifetime of the network, decreases delivery latencies, and improves the number of successful transmissions.

Acknowledgment

The simulation results presented in this Chapter have been obtained in collaboration with Michele Nati and Chiara Petrioli, University of Rome "Sapienza," Italy.

References

[1] "TinyOS Community Network." [Online]. Available: http://www.tinyos.net/

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, Mar. 2002.

[3] M. Zorzi and R. R. Rao, "Geographic random forwarding (GeRaF) for ad hoc and sensor networks: energy and latency performance," *IEEE Trans. Mobile Comput.*, vol. 2, no. 4, pp. 349–365, 2003.

[4] ——, "Geographic random forwarding (GeRaF) for ad hoc and sensor networks: multihop performance," *IEEE Trans. Mobile Comput.*, vol. 2, no. 4, pp. 337–348, 2003.

[5] D. Ferrara, L. Galluccio, A. Leonardi, G. Morabito, and S. Palazzo, "MACRO: an integrated MAC/routing protocol for geographic forwarding in wireless sensor networks," in *Proc. of IEEE InfoCom*, Miami, FL, Mar. 2005, pp. 1770–1781.

[6] M. Rossi, R. R. Rao, and M. Zorzi, "Statistically assisted routing algorithms (SARA) for hop count based forwarding in wireless sensor networks," *ACM/Springer's WINET*, 2006.

[7] K. Seada, M. Zuniga, A. Helmy, and B. Krishnamachari, "Energy-efficient forwarding strategies for geographic routing in lossy wireless networks," in *Proc. of ACM SenSys*, Baltimore, MD, Nov. 2004, pp. 108–121.

[8] S. Lee, B. Bhattacharjee, and S. Banerjee, "Efficient geographic routing in multihop wireless networks," in *Proc. of ACM MobiHoc*, Urbana-Champaign, IL, May 2005, pp. 230–241.

[9] C. Li, W. Hsu, B. Krishnamachari, and A. Helmy, "A local metric for geographic routing with power control in wireless networks," in *Proc. of IEEE SECON*, Santa Clara, CA, Sep. 2005, pp. 229–239.

[10] L. Savidge, H. Lee, H. Agajan, and A. Goldsmith, "Event-driven geographic routing for wireless image sensor networks," in *Proc. of COGnitive systems and Interactive Sensors (COGIS) Symposium*, Paris, France, Mar. 2006.

[11] C. E. Perkins and E. M. Royer, "Ad hoc on-demand distance vector routing," in *Proc. of IEEE WMCSA*, Feb. 1999.

[12] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," IETF RFC 3626, Oct. 2003. [Online]. Available: http://www.ietf.org

[13] J. Broch, D. B. Johnson, and D. A. Maltz, "The dynamic source routing protocol for mobile ad hoc networks," IETF Internet Draft, Dec. 1998. [Online]. Available: http://www.ietf.org

[14] "Global Positioning System." [Online]. Available: http://www.gps.gov/

[15] "The European Union Galileo Project." [Online]. Available: http://ec.europa.eu/dgs/energy_transport/galileo/

[16] A. Savvides and M. B. Srivastava, *Mobile Ad Hoc Networking*. IEEE Press and Wiley, Inc., 2004, ch. Location Discovery, S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, *eds*.

[17] M. Zorzi, "A new contention–based MAC protocol for geographic forwarding in ad hoc and sensor networks," *Proc. of IEEE ICC*, vol. 6, pp. 3481–3485, Jun. 2004.

[18] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: a geographic hast table for data-centric storage," in *Proc. of ACM WSNA*, San Diego, CA, 2003, pp. 78–87.

[19] R. A. Howard, *Dynamic Probabilistic Systems*. John Wiley & Sons, 1971.

[20] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. of the Hawaii International Conference on System Sciences*, Maui, Hawaii, Jan. 2000.

[21] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless sensor networks," *Proc. of ACM MobiCom*, pp. 243–254, Aug. 2000.

[22] Q. Fang and J. Gao and L. J. Guibas, "Locating and bypassing holes in sensor networks," *ACM Mobile Networks and Applications*, vol. 11, no. 2, pp. 187–200, Apr. 2006.

[23] H. Takagi and L. Kleinrock, "Optimal transmission ranges for randomly distributed packet radio terminals," *IEEE Trans. Commun.*, vol. 32, no. 3, pp. 246–257, Mar. 1984.

[24] R. Nelson and L. Kleinrock, "The spatial capacity of a slotted ALOHA multihop packet radio network with capture," *IEEE Trans. Commun.*, vol. 32, no. 6, pp. 684–694, Jun. 1984.

[25] T. Hou and V. Li, "Performance analysis of multihop routing strategies in multihop wireless networks," in *Proc. of IEEE GlobeCom*, Atlanta, GA, 1984, pp. 487–492.

[26] G. G. Finn, "Routing and addressing problems in large metropolitan-scale networks," ISI, Tech. Rep. ISU/RR-87-180, Mar. 1987.

[27] I. Stojmenovic and X. Lin, "GEDIR: loop–free location based routing in wireless networks," *Proc. of the International Conference on Parallel and Distributed Computers and Networks*, Nov. 1999.

[28] ——, "Power-aware localized routing in wireless networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 11, pp. 1122–1133, Nov. 2001.

[29] E. Kranakis, H. Singh, and J. Urrutia, "Compass routing on geometric networks," in *Proc. of the 11th Canadian Conference on Computational Geometry*, Vancouver, Canada, Aug. 1999, pp. 51–54.

[30] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward, "A distance routing effect algorithm for mobility (DREAM)," in *Proc. of ACM MobiCom*, Dallas, TX, Sep. 1998, pp. 76–84.

[31] Y.-B. Ko and N. H. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks," *ACM Wireless Networks*, vol. 6, no. 4, pp. 307–321, Jul. 2000.

[32] K. N. Amouris, S. Papavassiliou, and M. Li, "A position-based multi-zone routing protocol for wide area mobile ad hoc networks," in *Proc. of IEEE VTC*, Amsterdam, The Netherlands, Sep. 1999, pp. 1365–1369.

[33] Y. Yu, D. Estrin, and R. Govindan, "Geographical and energy–aware routing: a recursive data dissemination protocol for wireless sensor networks," UCLA Comp. Sci. Dept., Tech. Rep. 010023, May 2001.

[34] Y. Xu, J. Heidemann, and D. Estrin, "Geography–informed energy conservation for ad hoc routing," *Proc. of ACM SIGMobile*, pp. 70–84, Sep. 2001.

[35] T. Camp, J. Boleng, B. Williams, L. Wilcox, and W. Navidi, "Performance comparison of two location based routing protocols for ad hoc networks," in *Proc. of IEEE InfoCom*, New York City, NY, Jun. 2002, pp. 1678–1687.

[36] T. E. Lu and K. T. Feng, "Predictive mobility and location-aware routing protocol in mobile ad hoc networks," in *Proc. of IEEE GlobeCom*, St. Louis, MO, Nov. 2005, pp. 899–903.

[37] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wiley Wireless Communications and Mobile Computing, Special Issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2, pp. 483–502, Apr. 2002.

[38] J. Li and P. Mohapatra, "LAKER: location aided knowledge extraction routing for mobile ad hoc networks," in *Proc. of IEEE WCNC*, New Orleans, LA, Mar. 2003, pp. 1180–1184.

[39] R. K. Banka and G. Xue, "Angle routing protocol: location aided routing for mobile ad hoc networks using dynamic angle selection," in *Proc. of IEEE MilCom*, Anaheim, CA, Oct. 2002, pp. 501–506.

[40] G. V. Záruba, V. K. Chaluvadi, and A. M. Suleman, "LABAR: location area based ad hoc routing for GPS-scarce wide area ad hoc networks," in *Proc. of IEEE PerCom*, Dallas–Fort Worth, TX, Mar. 2003, pp. 509–513.

[41] X. Yang, R. Li, and Y. Liu, "An energy-efficient geographic routing algorithm for wireless sensor networks," in *Proc. of IEEE International Symposium on Communications Information Technologies*, Beijing, China, May 2005, pp. 671–676.

[42] H.-I. Liu and P.-C. Yen, "LB^{2}R: a load balanced & location based routing protocol for ad hoc networks," in *Proc. of IEEE VTC*, Los Angeles, CA, Sep. 2004, pp. 3970–3974.

[43] J. Hornsberger and G. C. Shoja, “Geographic grid routing: designing for reliability in wireless sensor networks,” in *Proc. of ACM IWCMC*, Vancouver, Canada, Jul. 2006, pp. 281–286.

[44] Z. Q. Taha and X. Liu, “On the reliability–aware geographic routing,” in *Proc. of the Wireless Telecommunications Symposium*, Pomona, CA, 2005, pp. 74–78.

[45] M. K. Marina and S. R. Das, “On-demand multipath distance vector routing,” in *Proc. of IEEE ICNP*, Riverside, CA, Nov. 2001, pp. 14–23.

[46] K. Zeng, K. Ren, and W. Lou, “Geographic on-demand disjoint multipath routing in wireless ad hoc networks,” in *Proc. of IEEE MilCom*, Atlantic City, NJ, Oct. 2005, pp. 1–7.

[47] H. Li and M. Singhal, “Energy-efficient forwarding strategies for geographic routing in lossy wireless networks,” in *Proc. of ACM SenSys*, Baltimore, MD, Nov. 2004, pp. 108–121.

[48] Q. Liang and Q. Ren, “Energy and mobility aware geographical multipath routing for wireless sensor networks,” in *Proc. of IEEE WCNC*, New Orleans, LA, Mar. 2005, pp. 1867–1871.

[49] T. Park and K. G. Shin, “Optimal tradeoffs for location-based routing in large-scale networks,” *IEEE/ACM Trans. Networking*, vol. 13, no. 2, pp. 398–410, Apr. 2005.

[50] S. Subramanian and S. Shakkottai, “Geographic routing with limited information in sensor networks,” in *Proc. of ACM IPSN*, Los Angeles, CA, Apr. 2005, pp. 269–276.

[51] B. M. Blum, T. He, S. Son, and J. A. Stankovic, “IGF: a robust state-free communication protocol for sensor networks,” University of Virginia, Tech. Rep. CS-2003-11, 2003.

[52] D. Chen, J. Deng, and P. K. Varshney, “On the forwarding area of contention-based geographic forwarding for ad hoc and sensor networks,” in *Proc. of IEEE SECON*, Santa Clara, CA, Sep. 2005, pp. 130–141.

[53] H. Füßler, J. Widmer, M. Käsemann, M. Mauve, and H. Hartenstein, “Contention-based forwarding for mobile ad hoc networks,” *Elsevier's Ad Hoc Networks*, vol. 1, no. 4, pp. 351–369, Nov. 2003.

[54] M. Witt and V. Turau, “BGR: blind geographic routing for sensor networks,” in *Proc. of 3^{rd} International Workshop on Intelligent Solutions in Embedded Systems*, Hamburg, Germany, May 2005, pp. 51–61.

[55] T. He, J. A. Stankovic, C. Lu, and T. Abdelzaher, “SPEED: a stateless protocol for real–time communication in sensor networks,” in *Proc. of IEEE ICDCS*, Providence, RI, May 2003, pp. 46–55.

[56] R. C. Shah and S. Wiethölter and A. Wolisz and J. M. Rabaey, “When does opportunistic routing make sense?” in *Proc. of IEEE PerCom*, Kauai, Hawaii, Mar. 2005, pp. 350–356.

[57] I. Stojmenovic, M. Russell, and B. Vukojevic, “Depth first search, location based localized routing and QoS routing in wireless networks,” in *Proc. of IEEE ICPP*, Toronto, Canada, Aug. 2000, pp. 173–180.

[58] P. Gupta and P. R. Kumar, “The capacity of wireless networks,” *IEEE Trans. Inform. Theory*, vol. 46, no. 2, pp. 547–566, Mar. 2000.

[59] R. S. S. Shakkottai and N. B. Shroff, “Unreliable sensor grids: coverage, connectivity and diameter,” in *Proc. of IEEE InfoCom*, San Francisco, CA, Mar. 2003, pp. 1073–1083.

[60] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, “Routing with guaranteed delivery in ad hoc wireless networks,” *Proc. of ACM DIAL-M*, pp. 48–55, Aug. 1999.

[61] P. Bose, L. Devroye, W. Evans, and D. Kirkpatrick, “On the spanning ratio of Gabriel graphs and β-skeletons,” in *Proc. of the 5^{th} Latin American Symposium on Theoretical Informatics*, Cancún, Mexico, Apr. 2002, pp. 479–493.

[62] F. Kuhn, R. Wattenhofer, and A. Zollinger, “Worst-case optimal and average-case efficient geometric ad-hoc routing,” in *Proc. of ACM MobiHoc*, Annapolis, MD, 2003, pp. 267–278.

[63] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu, “Geometric spanners for mobile networks,” *IEEE J. Select. Areas Commun.*, vol. 23, no. 1, pp. 174–185, Jan. 2005.

[64] G. Xing, C. Lu, R. Pless, and Q. Huang, “Impact of sensing coverage on greedy geographic routing algorithms,” *IEEE Trans. Mobile Comput.*, vol. 45, pp. 348–360, Apr. 2006.

[65] L. Barrière, P. Fraigniaud, L. Narayanan, and J. Opatrny, “Robust position-based routing in wireless ad hoc networks with unstable transmission ranges,” *Journal of Wireless Communications and Mobile Computing (WCMC)*, vol. 2, no. 3, pp. 141–153, 2001.

[66] K. Moaveninejad, W. Song, and X. Li, "Robust position-based routing for wireless ad hoc networks," *Elsevier Journal of Ad Hoc Networks*, vol. 3, no. 5, pp. 546–559, Sep. 2005.

[67] L. Blazevic, J.-Y. Le Boudec, and S. Giordano, "A location-based routing method for mobile ad hoc networks," *IEEE Trans. Mobile Comput.*, vol. 4, no. 2, pp. 97–110, Mar. 2005.

[68] B. Leong, S. Mitra, and B. Liskov, "Path vector face routing: geographic routing with local face information," in *Proc. of ICNP*, Boston, MA, Nov. 2005, pp. 147–158.

[69] Q. Huang, S. Bhattacharya, C. Lu, and G.-C. Roman, "FAR: face-aware routing for mobicast in large-scale networks," *ACM Trans. on Sensor Networks*, vol. 1, no. 2, pp. 240–271, Nov. 2005.

[70] M. Fayed and H. T. Mouftah, "Characterizing the Impact of Routing Holes on Geographic Routing," in *Proc. IEEE Systems Communications*, Montreal, Canada, Aug. 2005, pp. 401–406.

[71] P. He, J. Li, and L. Zhou, "A novel geographic routing algorithm for ad hoc networks based on localized delaunay triangulation," in *Proc. of the International Conference on Advanced Information Networking and Applications*, Vienna, Austria, Apr. 2006, pp. 49–54.

[72] S. Fotopoulou-Prigipa and A. B.McDonald, "A novel paradigm for geographic routing in ad hoc networks: comparison of geograms and geocircuits," in *Proc. of Communication Networks and Distributed Systems Modeling and Simulation Conference*, San Diego, CA, Jan. 2004.

[73] ——, "GCRP: geographic virtual circuit routing protocol for ad hoc networks," in *Proc. of IEEE MASS*, Fort Lauderdale, FL, Oct. 2004, pp. 416–425.

[74] D. Niculescu and B. Nath, "Trajectory-based forwarding and its applications," in *Proc. of ACM MobiCom*, San Diego, CA, Sep. 2003, pp. 260–272.

[75] S. Wu and K. S. Candan, "GPER: geographic power efficient routing in sensor networks," in *Proc. of IEEE ICNP*, Berlin, Germany, Oct. 2004, pp. 161–172.

[76] K. R. Gabriel and R. R. Sokal, "A new statistical approach to geographic variation analysis," *Systematic Zoology*, vol. 18, pp. 259–278, 1969.

[77] G. T. Toussaint, "The relative neighbourhood graph of a fintie planar set," *Pattern Recognition*, vol. 12, pp. 261–268, 1980.

[78] H. Frey, "Scalable geographic routing algorithms for wireless ad hoc networks," *IEEE Network*, vol. 18, no. 4, pp. 18–22, Jul. 2004.

[79] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker, "Geographic routing made practical," in *Proc. of NSDI*, Boston, MA, May 2005.

[80] ——, "On the pitfalls of geographic routing," in *Proc. of ACM DIAL-M*, Cologne, Germany, Sep. 2005, pp. 34–43.

[81] H. Frey and I. Stojmenovic, "On delivery guarantees of face and combined greedy-face routing in ad hoc and sensor networks," in *Proc. of ACM MobiCom*, Los Angeles, CA, Sep. 2006, pp. 390–401.

[82] K. Seada, A. Helmy, and R. Govindan, "On the effect of localization errors on geographic face routing in sensor networks," in *Proc. of IEEE/ACM IPSN*, Berkeley, CA, Apr. 2004, pp. 71–80.

[83] R. C. Shah, A. Wolisz, and J. M. Rabaey, "On the performance of geographical routing in the presence of localization errors," in *Proc. of IEEE ICC*, Seoul, Korea, May 2005, pp. 2979–2985.

[84] R. Jain, A. Puri, and R. Sengupta, "Geographical routing using partial information for wireless ad hoc networks," *IEEE Personal Commun. Mag.*, vol. 8, pp. 48–57, Feb. 2001.

[85] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic routing without location information," in *Proc. of ACM MobiCom*, San Diego, CA, Sep. 2003, pp. 96–108.

[86] A. Jadbabaie, "On geographic routing without location information," in *Proc. of IEEE Conference on Decision and Control*, Atlantis, Paradise Island, Dec. 2004, pp. 4764–4769.

[87] Q. Cao and T. Abdelzaher, "A scalable logical coordinates framework for routing in wireless sensor networks," in *Proc. of the IEEE RTSS*, Lisbon, Portugal, Dec. 2004, pp. 349–358.

[88] Q. Fang, J. Gao, L. J. Guibas, V. de Silva, and L. Zhang, "GLIDER: gradient landmark-based distributed routing for sensor networks," in *Proc. of IEEE InfoCom*, Miami, FL, Mar. 2005, pp. 339–350.

[89] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, “Vivaldi: a decentralized network coordinate system,” *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 15–26, Oct. 2004.

[90] R. Fonseca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, S. Shenker, and I. Stoica, “Beacon Vector Routing: scalable point-to-point routing in wireless sensornets,” in *Proc. of NSDI*, Boston, MA, May 2004.

[91] A. Caruso, S. Chessa, S. De, and A. Urpi, “GPS free coordinate assignment and routing in wireless sensor networks,” in *Proc. of IEEE InfoCom*, Miami, FL, Mar. 2005, pp. 150–160.

[92] Y. Zhao, Q. Zhang, Y. Chen, and W. Zhu, “Hop ID based routing in mobile ad hoc networks,” in *Proc. of IEEE ICNP*, Boston, MA, Nov. 2005, pp. 179–190.

[93] Ben Wing Lup Leong, “New techniques for geographic routing,” Ph.D. dissertation, Massachusetts Institute of Technology (MIT), Cambridge, MA, May 2006.

[94] P. Y. Xu and Z. L. Jun, “Virtual destination-based geographic routing in ad hoc mobile networks,” in *Proc. of IEEE WCNM*, Wuhan, China, Sep. 2005, pp. 686–689.

[95] L. Zou, M. Lu, and Z. Xiong, “PAGER: a distributed algorithm for the dead-end problem of location-based routing in sensor networks,” *IEEE Trans. Veh. Technol.*, vol. 55, pp. 1509–1522, Jul. 2005.

[96] C. Lochert, M. Mauve, H. Füßler, and H. Hartenstein, “Geographic routing in city scenarios,” *ACM Mobile Computing and Communications Review*, vol. 9, pp. 69–72, Jan. 2005.

[97] S. Radhakrishnan, G. Racherla, and C. N. Sekharan, “DST–A routing protocol for ad hoc networks using distributed spanning trees,” in *Proc. of IEEE WCNC*, New Orleans, LA, Sep. 1999, pp. 1543–1547.

[98] J. C. Navas and T. Imielinski, “Geocast: geographic addressing and routing,” in *Proc. of ACM MobiCom*, Budapest,Hungary, Sep. 1997, pp. 66–76.

[99] T. Imielinski and J. C. Navas, “GPS-based geographic addressing, routing and resource discovery,” *Communications of the ACM*, vol. 42, no. 4, pp. 86–92, 1999.

[100] Y.-B. Ko and N. H. Vaidya, “Flooding-based geocasting protocols for mobile ad hoc networks,” *ACM/Baltzer Mobile Networks and Applications*, vol. 7, no. 6, pp. 471–480, Dec. 2002.

[101] ——, “Anycasting-based protocol for geocast service in mobile ad hoc networks,” *Elsevier Computer Networks*, vol. 41, no. 6, pp. 743–760, Apr. 2003.

[102] V. Park and M. Corson, “A highly adaptive distributed routing algorithm for mobile wireless networks,” in *Proc. of IEEE InfoCom*, vol. 3, Anchorage, AK, Apr. 1997, pp. 1405–1413.

[103] K. Seada and A. Helmy, “Efficient geocasting with perfect delivery in wireless networks,” in *Proc. of IEEE WCNC*, vol. 4, Atlanta, GA, Mar. 2004, pp. 2551–2556.

[104] Y. Chen and J. L. Welch, “Location-based broadcasting for dense mobile ad hoc networks,” in *Proc. of ACM MSWiM*, Montreal, Canada, Oct. 2005, pp. 63–70.

Chapter 4

Conclusions

This thesis has been devoted to showing the benefits of cross–layer protocol design in different kinds of networks. Cross–layer design is motivated by its greater efficiency, in that it can achieve better performance thanks to an increased number of degree of freedom. By cross–layer design, we intend the way a protocol is supposed to use information coming from different layers of the ISO/OSI stack. This information is processed and evaluated, perhaps through interactions and iterations between different layers. We have mainly dealt with two different kinds of network. In MIMO ad hoc networks (Chapter 2), our cross–layer design resulted in a back-and-forth interaction between the PHY and MAC layers, whereas in Wireless Sensor Networks (Chapter 3) a protocol was created that encompassed MAC and routing functionalities.

In Appendix B, we have also discussed a PHY-aware design and comparison of communication schemes in Underwater Acoustic Sensor Networks. These are very different both in requirements and in available resources. On one hand, WSNs are usually designed to pursue long lifetime and low energy consumption. However, certain applications may also require that data be forwarded quickly, hence posing a further constraint on delivery delay. On the other hand, MIMO ad hoc networks are expected to have more resources available (in terms of energy and computation capabilities), therefore the main focus becomes throughput, delay and success ratio performance.

In either scenario, cross–layer design has been shown to boost performance considerably, if carried out correctly. We have shown that, in MIMO networks, a higher throughput and a generally better performance can derive from jointly considering the power of foreseen transmissions and the load on the spatial-demultiplexing receiver. In WSNs, instead, we sped up the relay choice and thus relieved some traffic on the network and decreased the energy consumption through a cross–layer mechanism for routing and relay election.

The results prove the validity of the cross–layer approach, in that it accomplishes the performance enhancement objectives without noticeable problems coming from breaking the structure of the OSI stack.

Appendix A

Proof of the correctness of Rainbow

Let us define a network topology graph $G = (N, E)$ as the set N of the network nodes and the set E of links between nodes that can communicate with each other. Note that we do not make any assumption that the graph has to be a UDG. Two nodes are neighbors if they can hear each other′s transmissions. Each node x has an associated value $d(x)$, which indicates its distance from the sink (*e.g.*, $d(x)$ can be the Euclidean distance from x to the sink). Based on their weights $d(y)$ x′s neighbors y are placed either in F or in F_C. Node x has outgoing edges toward its neighbors in F and incoming edges from neighbors in F_C.

We now formally prove the claims stated in Section 3.7, regarding the properties of the Rainbow algorithm.

Theorem 1 (Rainbow is loop-free)**.** *The Rainbow extension to ALBA always finds loop-free routes to the sink.*

Proof. We start by showing that there are no cycles (loops) in routes only made up of nodes of the same color (say, all C_1 nodes). Let x_0 be a C_1 node and $x_1, x_2, \ldots, x_k$ a route through k C_1 nodes. If a k-cycle exists (*i.e.*, $x_1 = x_k$), then $d(x_1) = d(x_k)$, where $d(x)$ is the distance from x to the sink. However, C_1-nodes forward packets only if $d(x_0) > d(x_1) > \cdots > d(x_k)$, hence a contradiction.

Now let us proceed by induction on h, the number of colors. Suppose that routes are loop-free for all nodes with colors $C_1, \ldots, C_h$. We have to prove that the theorem holds true for $h+1$ colors. We consider the two cases corresponding to $h+1$ being either odd or even. If $h+1$ is odd, C_{h+1}-nodes search for relays in region F. Let us consider a route from a C_{h+1}-node x_0 to the sink S, $x_0, \ldots, x_k, x_{k+1}, \ldots, S$, passing through C_{h+1}-nodes, then C_h-nodes, and so on. Let x_k be the first C_h-node. The route $x_k, \ldots, S$ is loop-free by the inductive hypothesis. The route from x_0 to x_{k-1} is clearly loop free, since all the nodes have the same color. Finally, it cannot occur that $x_i = x_j$ for nodes $i < k$ and $j \geq k$, since this would imply that node i has changed its color from C_{h+1} to C_h, which is against Rainbow rules.

The case when $h+1$ is even is similar to the previous one, considering that the region where relays are searched for is F^C instead of F and that the direction of the inequalities becomes "$\leq$." □

The following theorem shows that if nodes are able to reliably decide whether they have neighbors in F or F_C, then they will eventually assume the proper color. In the proof by *alternation* we indicate a change in the region, whether F or F^C, queried for relays (Section 3.7)).[1]

Theorem 2. *All the nodes that have routes to the sink with h alternations (and no less than h), and only those nodes, assume the color C_h in finite time.*

Proof. Let us denote with $F(x)$ and $F^C(x)$ the regions F and F^C as seen from node x. Let also $N_F(x)$ and $N_{F^C}(x)$ be the sets of neighbors of x in the corresponding regions. We assume that if at least a relay exists in $F(x)$ and $F^C(x)$, x can eventually find it.

In the proof, we use the function $\Xi(x)$ that lists the network nodes in order of increasing distance from the sink, *i.e.*, for the node x closest to the sink $\Xi(x) = 1$, for the node x which is the second closest to the sink $\Xi(x) = 2$, etc. If two nodes y and z are at the same distance from the sink, and $y < z$, we stipulate that $\Xi(y) < \Xi(z)$. We also use the function $\Xi'(x)$ that lists the network nodes in order of increasing number of alternations required to reach the sink. If two nodes show the same number of alternations, then the closer to the sink the lower its Ξ'. Ties are broken as for Ξ.

The theorem is proven by a doubly inductive argument. First we proceed by induction on the number h of alternations required to get packets from their source to the sink. Then we perform another induction on $\Xi(x)$ and $\Xi'(x)$.

First, we consider the case $h = 1$. We start by proving that a node x gets colored C_1 if there exists a route from x to the sink where each relay is chosen in the F region of its predecessor in the route (no alternation case). We proceed by induction on $\Xi(x)$. If $\Xi(x) = 1$, either the sink is in $N_F(x)$ or not. In the first case, x is a C_1-node because it finds the sink in F. If the sink is not in $N_F(x)$ then the network is disconnected, since the sink is out of the transmission range of its closest node. Since we assumed the network connected, this case never occurs. Let us now assume that for each y such that $\Xi(y) \leq k$, $k > 1$, the claim holds true. Consider the node x such that $\Xi(x) = k + 1$. If the route $x = x_0, x_1, \ldots, S$ from x to the sink S is formed of relays chosen in each node's F region, then $d(x_i) \leq \cdots \leq d(x_0)$, where $x_i \in N_F(x_{i-1})$. If $d(x_{i+1}) = d(x_i)$ then $x_{i+1} < x_i$. Therefore, $\Xi(x_i) < \Xi(x_0)$. To each of the x_i the inductive hypothesis applies, which means that every node x_i stays colored C_1 (each node belonging to the route is a C_1-node). Now, $x = x_0$ has at least one relay in $F(x_0)$ (the node x_1), and since x_1 is a C_1-node, hence the node with $\Xi(x) = k + 1$, *i.e.*, x itself, remains colored C_1.

Let us now consider the case where x such as $\Xi(x) = k + 1$ does not have routes to the sink without alternations (the "only" part of the "all and only" claim). Since $d(y) < d(x)$ or $d(y) = d(x)$ and $y < x$, for each $y \in N_F(x)$, then $\Xi(y) < \Xi(x)$. As a result, in force of the inductive hypothesis on $\Xi(x)$, all nodes $y \in N_F(x)$ will change their color until $N_F(x)$ has no more C_1-nodes. At this time, x will no longer be able to forward in F, and will change its color as well. To sum up, for $h = 1$, the color of a node is C_1 if and only if the node exhibits a route toward the sink whose relays are always found in the region F of its predecessor node.

By using the same argument, we now prove the following claim (case $h > 1$).

[1]This awareness is achievable, *e.g.*, with a fine tuning of the number of attempts required before considering F or F^C empty (see also section 3.7).

Claim 1. *Assume that all and only those nodes that have routes to the sink with $j \leq h$ alternations, $j > 1$, take color C_j in a finite time. Then all and only those nodes showing routes with $h + 1$ alternations take color C_{h+1} in a finite time.*

Let us assume at first that $h+1$ is odd. We proceed by induction on $\Xi'(x)$. *Base case.* Let x be the node closer to the sink and with routes to the sink with $h+1$ alternations. Let also $m = \Xi'(x)$. By induction, Claim 1 is true for all nodes y having $\Xi'(y) < m$, since these nodes have routes with at most h alternations. Now, in any route with $h + 1$ alternations $x = x_0, x_1, \ldots, S$ from x to the sink S, $x_1 \in F(x_0)$, and moreover, $\Xi'(x_1) < \Xi'(x_0)$. Therefore, x_1 is a C_h-node. Node x cannot be colored with any of the $C_h, \ldots, C_1$ colors because there is no route from x_0 to the sink with at most h alternations. However, it can stay colored C_{h+1} since it has a neighbors in its F zone which is a C_h-node.

Inductive step. Assuming that Claim 1 is true for any x such that $\Xi'(x) \leq k$, we prove that it is also true for $\Xi'(x) = k + 1$.

Let x_0 be a node such that $\Xi'(x_0) = k + 1$ and let y_0 be the node such that $\Xi'(y_0) = k$. Either both x_0 and y_0 require h+1 alternations, or x_0 requires more than $h+1$ alternations. In the first case, a route $x_0, x_1, \ldots, S$ with $h + 1$ alternations leads to the sink, with $x_1 \in F(x_0)$, requiring h or $h + 1$ alternations. Since $\Xi'(x_1) < \Xi'(x_0)$, node x_1 has color C_h or C_{h+1} by induction. Node x_0 cannot be colored with any of the $C_h, \ldots, C_1$ colors because there is no route from x_0 to the sink with at most h alternations. Therefore, x_0 will assume color C_{h+1}, due to the presence of x_1 in $F(x_0)$.

In the second case, x_0 cannot assume any of the colors $C_{h+1}, C_h, \ldots, C_1$. In fact, by induction on h it cannot be colored with $C_h, \ldots, C_1$. To be colored with C_{h+1}, C_h-nodes or C_{h+1}-nodes should be in $F(x_0)$. Such nodes would precede x_0 in the sorting given by Ξ'. Therefore, they would have routes to the sink with at most $h + 1$ alternations by induction on Ξ'. But then x_0 would exhibit at least one route with $h+1$ alternations, which is a contradiction.

The case with $h+1$ even is similar to when it is odd. This time a different ordering function needs to be used, say $\Xi''(x)$, which lists nodes in order of increasing number of alternations required to reach the sink. Unlike Ξ', the node farther from the sink precedes the closer in case they exhibit an equal number of alternations. $\square$

Theorem 1 and Theorem 2 hold for any topology graph G and for any distance $d(x)$ which induces an ordering on nodes. Therefore, ALBA works also in those more realistic settings that cannot be accurately modeled by UDG graphs. More of that, ALBA is quite resilient to localization errors. Whenever a localization protocol is executed each node x estimates its position which is most of the times affected by a non-negligible error. Based on its estimated coordinates, node x computes its (estimated) distance $d'(x)$ from the sink. We observe that two nodes are neighbors independently of their distance, *i.e.*, of their own estimated coordinates. That is because the network topology graph as defined by ALBA is based only on nodes being able to communicate. Therefore, the localization errors only affect the orientation of the links between the nodes. Each node has the same set of neighbors it would have if the nodes would be able to communicate their own exact coordinates. However, the estimated distance from the sink d' may generate different sets F and F_C.

Since Theorems 1 and 2 hold also for the estimated distance d' (i.e., independently of the orientation of the links), the following corollary holds.

Corollary 1. *ALBA-R finds loop-free routes from any node x to the sink even when x's coordinates are affected by a possibly high localization error.*

Appendix B

Cross–Layer Design in Underwater Acoustic Networks

Contents

The following material is devoted to the study of PHY-aware transmission strategies and protocol design for UnderWater Acoustic Networks (UWANs) and more specifically UnderWater Acoustic Sensor Networks (UWASNs). This scenario represents another kind of wireless network where communications usually take place by means of acoustic (instead of electromagnetic) waves. This is due to a number of reasons that will be made clearer later. UWASNs are currently a very hot research topic, and represent one more case where a cross–layer design could definitely outperform a layered solution. The constraints posed by the physical layer must be properly taken into account when designing an underwater system. Such constraints can be quite limiting, especially in terms of bandwidth available on a certain link.

In the following, we start by introducing the model for the physical layer and highlighting the main implications it has on the design of an access scheme. Then, we proceed to describe our transmission technique and to carry out some comparison by means of analysis. Finally, we introduce a broadcasting protocol whose creation is based on the considerations derived from the analysis. We conclude presenting our ideas for future work on this topic.

B.1 Analysis of Different Transmission Policies for Underwater Networks

B.1.1 Introduction and Channel Model

Distributed underwater sensor networks are an emerging research area, that currently stimulates an increasing number of research contributions. The interest around this subject is well justified by the additional challenges posed by wireless underwater networking with respect to terrestrial radio communications. Radio waves scatter rapidly underwater, allowing reasonable communication performance only over very short distances. As an alternative to radio waves, optical technologies also enable underwater communications, but are feasible only within a limited reach, besides requiring further efforts to keep the transmitter and receiver aligned. However, recent developments in acoustic modem technology are paving the way to the deployment of the first underwater sensor networks, opening new perspectives to the way monitoring tasks can be carried out. Underwater networks would operate unattended according to the *ad hoc* paradigm well known in radio networks, but those networks call for the design of communication schemes and networking protocols that meet the challenges of the underwater channel and application performance requirements. As for terrestrial ad hoc networks, such needs may vary depending on the operations attended, from real-time data delivery by detection and tracking systems, to energy-efficient data harvesting as required by long term environmental sampling operations. UWASNs may also be augmented with autonomous underwater vehicles (AUVs); for example, this unmanned machinery could query the sensors on-demand, or cooperate to haul data to control stations.

In order to design effective and energy-efficient underwater communication networks, the physics of acoustic propagation must be taken into account. Acoustic waves bring a quite different propagation behavior into the picture [1]. First of all, they propagate at a slow speed $c \approx 1500\,\mathrm{m/s}$, which is five orders of magnitude smaller than for radio propagation in the air. The propagation speed actually changes with the depth, temperature, and salinity of the water. Urick has shown that this dependency can be accurately modeled as follows:

$$\begin{aligned} c(t,S,z) &= 1449.05 + 45.7t - 5.21t^2 + 0.23t^3 \\ &+ (1.333 - 0.126t + 0.009t^2)(S-35) \\ &+ 16.3z + 0.18z^2\,, \end{aligned} \tag{B.1}$$

where t is one tenth of the temperature of the water in degrees Celsius, z is the depth in meters, and S is the salinity of the water. The most important factor in (B.1) is the temperature of the water. For oceans, the temperature typically ranges between $2\,^{\circ}\mathrm{C}$ and $22\,^{\circ}\mathrm{C}$ [2]. The salinity, instead, is in the interval $[32, 37]$ parts per thousand (ppt) with an average of $35\,\mathrm{ppt}$ [3]. We stress again that this speed is very small, and is to be taken into account during the protocol design phase. For example, it should be avoided that a MAC protocol requires long handshakes, as these would consume a lot of time.

However, the most unusual feature of the underwater acoustic channel is the *dependence between the bandwidth and the transmission distance*. More precisely, the available bandwidth tends to shrink for increasing distance due to a superposition of frequency-dependent effects related to attenuation and noise.

To explain this fact, let us start with the model for attenuation. Urick models the attenuation incurred by a tone as a function of distance and frequency with the following formula [1]:

$$A(\ell, f) = d^k a(f)^d\,, \tag{B.2}$$

where d is the distance between the transmitter and the receiver, and f is the frequency of the tone. Moreover k is called the spreading coefficient, and models the geometry of the propagation. If the propagation is perfectly spherical (such as in deep water where a wave finds no boundaries until after several kilometers), $k = 2$. Conversely, if the propagation is perfectly cylindrical (such as in very shallow water), $k = 1$. Typically, a "practical" propagation is modeled by posing $k = 1.5$, to represent a mixed propagation scenario. The factor d^k is called the spreading loss. It should be noted that k is the counterpart of the attenuation exponent in the radio path loss model. Finally, the factor $a(f)$ in the above formula is called the absorption loss, and models the conversion of acoustic pressure into heat due to the resonance with certain ions present in water. This factor can be conveniently approximated by Thorp's formula [4] as follows:

$$10\log_{10} a(f) = 0.11\frac{f^2}{1+f^2} + 44\frac{f^2}{4100+f^2} + 2.75\cdot 10^{-4} f^2 + 0.003\,. \tag{B.3}$$

The formula gives an expression of $a(f)$ in $\mathrm{dB/km}$ for f in kHz. Considering the various factors, it can be seen that the attenuation increases with frequency, and that the dependence on distance is much deeper than in radio, due to the factor $a(f)^d$.

The noise power spectral density (psd) is dependent on frequency as well. In particular, four main sources of noise are identified: turbulence, shipping and other human activities, wind and waves, and thermal noise in the receiver circuitry. All of these contribute to part of the noise spectrum and can be modeled according to the following formulas:

$$\begin{aligned}
10\log_{10} N_t(f) &= 17 - 30\log(f) && \text{(B.4)}\\
10\log_{10} N_s(f) &= 40 + 20(s-0.5) + 26\log(f) - 60\log(f+0.03) && \text{(B.5)}\\
10\log_{10} N_w(f) &= 50 + 7.5\sqrt{w} + 20\log(f) - 40\log(f+0.4) && \text{(B.6)}\\
10\log_{10} N_{th}(f) &= -15 + 20\log(f)\,, && \text{(B.7)}
\end{aligned}$$

where s is the *shipping factor*, representing the intensity of shipping activities on the surface of the water and has values ranging between 0 and 1. The factor w is the *wind speed* in $\mathrm{m/s}$. The total noise psd is then given by

$$N(f) = N_t(f) + N_s(f) + N_w(f) + N_{th}(f)\,. \tag{B.8}$$

The different components impact the noise power spectral density at different frequencies. For example, in the frequency ranges encountered for distances over tens of meters, the turbulence and shipping components have very little effect, whereas the other two can become dominant. It could also be seen that noise has a "V–shaped" psd [5], that tends to decrease until roughly $40\,\mathrm{kHz}$, before starting to increase again.

We are now ready to define the average SNR of a tone transmitted at a frequency f and traveling a distance d as

$$SNR(d, f) = \frac{P_T/A(d, f)}{N(f)\Delta f}\,, \tag{B.9}$$

where P_T is the transmit power and $N(f)$ is the noise power spectral density (assumed constant in a narrow band Δf around f). In (B.9), the factor $1/A(d, f)N(f)$, or "AN factor" is the frequency-dependent component. Since $A(d, f)$ increases with frequency while $N(f)$ decreases at least to a certain point, the product between the two has a maximum in correspondence of some frequency f_0. This maximum represents minimal combined attenuation and noise effects, and the place where it is more convenient to place the tone to transmit. Around f_0, one can also define a bandwidth, *e.g.*, according to the empirical $-3\,\mathrm{dB}$ definition, and hence find two more frequencies that represent the limits of this bandwidth.

Figure B.1 shows a number of concave dotted lines that represent the AN factor for varying distance between the transmitter and receiver, from $10\,\mathrm{m}$ to $100\,\mathrm{km}$. Two bold lines show the limits of the bandwidth $B(d)$ as derived according to the $-3\,\mathrm{dB}$ definition, *i.e.*, $B(d) = \{f : SNR(d, f) > SNR(d, f_0)/2\}$. The very interesting outcome here is that *the available bandwidth shrinks for increasing distance*. This is different from what happens in radio, where increasing the distance reduces receive power and perhaps forces to decrease the transmit bit rate, in order to meet a certain SNR requirement. In underwater, that is also true, but in addition the frequency limits of the bandwidth get closer and tend to shift to the lower frequencies of the acoustic spectrum. Moreover, the optimal transmission frequency for a tone shifts toward the lower frequencies. This must be carefully taken into account when designing any protocol for underwater acoustic networks. For example, performing a few long-range hops in a multihop path, here, would require to transmit at a very high power (because of the high attenuation) and at a low bit rate (because of the small bandwidth), thus increasing the energy consumption considerably.

Let us proceed by deriving the channel effects over a signal with a certain spectrum $S(f)$. Assume for simplicity that this spectrum is flat, *i.e.*, $S(f) = P_T/B(d)$. The SNR in this case

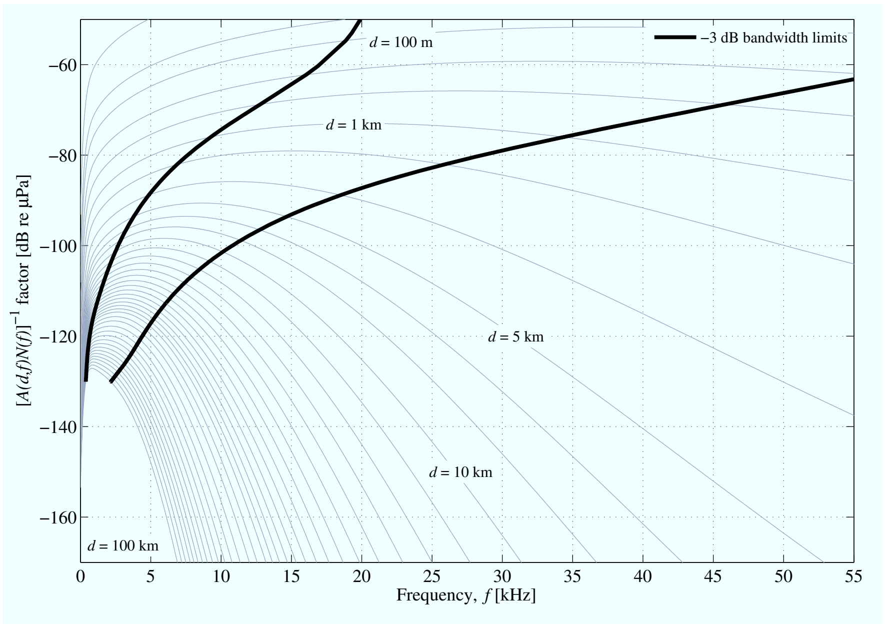


Figure B.1. *Frequency-dependent part of the SNR for an acoustic tone transmitted underwater. Bold lines represent the lower and upper limit of the available transmit bandwidth. Different grey lines represent the channel response for different distances.*

can be calculated as

$$SNR\,(d, B(d)) = \frac{\dfrac{P_T}{B(d)} \displaystyle\int_{B(d)} A^{-1}(d,f)\,\mathrm{d}f}{\displaystyle\int_{B(d)} N(f)\,\mathrm{d}f}\,. \tag{B.10}$$

Due to the one-to-one relationship between d and $B(d)$, all integrals in (B.10) are determined when d is fixed. In the following, we will use the shorthand notation

$$A_d^{-1} = \int_{B(d)} A^{-1}(d,f)\,\mathrm{d}f\,, \qquad N_d = \int_{B(d)} N(f)\,\mathrm{d}f\,. \tag{B.11}$$

B.1.2 Description of the Considered Underwater Scenario

In this study, we wish to investigate the impact of the bandwidth–distance relationship on the design of a relay acoustic link from a broad point of view. The analysis carried out in the following is aimed at highlighting relevant tradeoffs emerging from the use of different data forwarding policies.

To this aim, we focus on a linear sensor network topology, whereby r nodes are distributed evenly over a distance d_ℓ. The information collected by the sensors has to be sent to an end node which acts as a common sink. In this scenario, we analytically evaluate and compare a number of forwarding techniques that make different use of the bandwidth available at different distances, with the aim to find how delivery delay and energy consumption per packet vary with each policy, and how the parameters of each policy affect its performance. Assume that each node generates N packets that must be transmitted to the sink (*i.e.*, the last node in the line). The nodes are endowed with the chance to choose how to forward these packets. In fact, the acoustic channel allows to bridge very long distances, if the proper frequency and bandwidth are used (see the previous Section). Hence each node could choose to forward all N packets directly to the sink, or to employ some sort of relaying policy. Since direct transmissions would require a lot of energy, the nodes are allowed to co-operate, in order to improve the overall system efficiency. They do so by relaying the information, so that a data packet from a node far away can travel to the sink over multiple hops, thus consuming less power, and having a greater bandwidth available. However, relaying can be employed instead of, or in addition to, direct transmission. Specifically, we study the following transmission policies:

1. **Relaying only**: In this case, each node relays the packets from all the nodes upstream from it, in addition to transmitting its own N packets. All nodes transmit at the same bit rate. An Automatic Repeat reQuest (ARQ) procedure is used to implement reliable transmission. Two forms of ARQ are considered: a simple Stop-and-Wait (S&W), and a group S&W, in which a selective acknowledgment (ACK) is sent for a group of M packets [6]. The latter policy is hereon referred to as M–S&W.

2. **Parallel transmission**: In this case, each relay node has an option to forward some of its packets directly to the sink, while the rest are being sent to the next relay. Transmission over the two links is accomplished in two disjoint bands. The effect expected in this case is a reduction of the overall delay at the price of an increase in the energy consumption, as higher power is required to bridge longer links.

In the parallel transmission policy, the same bit rate is used on both links. However, the longer propagation delay on long links calls for more efficient error control strategies than S&W ARQ. ARQ is thus confined to short links where the delay can be tolerated. Instead, higher transmission power is used on long links. This power is set so that the probability that all packets are correctly received is $1 - p_t$, for a target probability p_t.[1] On short links, the transmission power is set so as to achieve a target value SNR_0 for the Signal-to-Noise ratio, in order to ensure a sufficiently low probability of packet error. The policies are analyzed in the following Section.

B.1.3 Transmission Policy Analysis

Relaying with S&W ARQ

This simple policy is meant to serve as a baseline for reference and comparison. Let p be the probability of success for a packet containing L bits. According to (B.10), and assuming the use of a BPSK modulation with independent channel errors on the L transmitted

[1]Note that in a more general setting that takes channel fading into account, sending all packets at higher power may not be the most efficient strategy. Some form of packet coding (*e.g.*, Reed-Solomon [7]) would help instead. The evaluation of this scenario is out of the scope of the present discussion and is left for future study.

symbols, we have

$$p = \left(1 - \frac{1}{2}\mathrm{erfc}\sqrt{SNR_0}\right)^L . \tag{B.12}$$

Note that (B.12) is the formula for the BPSK bit error rate assuming additive white Gaussian noise. Although the noise is non-white [5], when the signal bandwidth is sufficiently small, one can assume that the power spectral density of the noise is almost constant. In these conditions, expression (B.12) holds. The SNR is defined as that of an *equivalent* AWGN channel as in (B.10) [5].

Since the total distance is equally split between $h = r + 1$ hops, where r is the number of relays, each hop covers a distance of $d_r = d_\ell / h$. Hence, the transmit power is found as

$$P_T(d_r) = \frac{SNR_0 B(d_r) N_{d_r}}{A_{d_r}^{-1}} . \tag{B.13}$$

The average time required to transmit a packet between two successive nodes is $(T_d + 2\tau_{d_r})/p + T_a$, where τ_x denotes the propagation time of an acoustic wave traveling a distance x, whereas T_d and T_a are the transmission time of a data and an ACK packet, respectively. Note that in the previous formula we assumed that no ACK is transmitted if a packet is not received correctly. In this case, the transmitter sends the packet again after a time-out interval, that is long enough to accommodate the propagation delay between the sender and the relay downstream. Now, each node in the line must forward its own packets, and all those received from the previous nodes as well. Say that each node generates N packets. Hence the number of packets to send is N for the initial sender, $2N$ for the first relay, $3N$ for the second relay, and so forth. Assuming that channel access is ideal, whereby each relay transmits its own packets rightly after the previous one has completed its transmission, the total delay $D^{(R)}$ is equal to the time required to transmit $N \cdot h(h+1)/2$ packets:

$$D^{(R)} = N \cdot \frac{h(h+1)}{2} \left(\frac{T_d + 2\tau_{d_r}}{p} + T_a \right) . \tag{B.14}$$

The average energy consumed to have one packet correctly reach the following relay is found to be

$$E^{(R)} = \frac{T_d P_T(d_r) + (T_d + 2\tau_{d_r}) P_R}{p} + (P_T(d_r) + P_R) T_a , \tag{B.15}$$

where the first term represents the average energy spent until the data packet correctly reaches the receiver, and the second one is the energy required to transmit the ACK packet. P_R denotes the receive power. In the evaluation of both the total delay and the energy per packet, the return channel is assumed to be error-free, so that ACKs always reach their recipient correctly. Since ACKs are short and transmitted at the same power used for data, this assumption holds with high probability.

Relaying with M–S&W

In this case, the ARQ policy is slightly different. The sender transmits a burst of M packets back-to-back and waits for a reception confirmation. The recipient always replies with a selective ACK, to report which packets were correctly received and which were not. As opposed to plain S&W ARQ, the reply packet is needed even if no data packet is correctly received; otherwise the senders would not know which transmissions to repeat. In the following, we will call the sequential transmission of the data packet train and of the reply packet a "round."

To capture the average total delay and the average energy consumption per packet, we use the following semi–Markov model. Let p_{ij} be the probability that, after a certain transmission round, j packets are left to be transmitted, given that i packets were sent during that round, with $i, j = 0, 1, \ldots, M$. For now, focus on a group of exactly M packets, and organize the probabilities in a $(M+1) \times (M+1)$ matrix $\boldsymbol{P}_M = (p_{ij})$ as

$$\boldsymbol{P}_M = \begin{bmatrix} 1 & 0 & \cdots & \\ p & 1-p & 0 & \cdots \\ \vdots & \ddots & & \vdots \\ \binom{M-1}{M-1} p^{M-1}(1-p)^0 & \cdots & \binom{M-1}{0} p^0 (1-p)^{M-1} & 0 \\ \binom{M}{M} p^M (1-p)^0 & & \cdots & \binom{M}{0} p^0 (1-p)^M \end{bmatrix} \tag{B.16}$$

Note that $p_{ij} = \binom{i}{i-j} p^{(i-j)} (1-p)^j$ for $j \leq i$, otherwise $p_{ij} = 0$.

Define now t_i as the time required to complete a round if i packets are left to be transmitted. Define also $T(m) = mT_d + 2\tau_{d_r} + T_a$ as the time needed to perform a round when the burst to send contains m packets. We can organize the t_is, $i = 1, \ldots, M$ in a vector as

$$\boldsymbol{T}_M = \begin{bmatrix} 0 & T(1) & \cdots & T(M-1) & T(M) \end{bmatrix} . \tag{B.17}$$

Note that we do not need a matrix here, as the duration of the round depends only on the number of packets left to transmit, i. A similar argument holds for the vector $\boldsymbol{E}_M$, where each entry e_i represents the energy expenditure in a round where i packets are left to be transmitted. By defining $E(m) = (mT_d + T_a)P_T(d_r) + (mT_d + T_a + 2\tau)P_R$, we have

$$\boldsymbol{E}_M = \begin{bmatrix} 0 & E(1) & \cdots & E(M-1) & E(M) \end{bmatrix} . \tag{B.18}$$

In general, the number of packets to be sent by a certain relay x (say, N_x) may be greater or less than M, depending on the position that x holds in the line. If $N_x > M$, the matrix $\boldsymbol{P}$ describing the transmission process can be derived by replicating $N - M$ times the first row of $\boldsymbol{P}_M$ as follows:

$$\boldsymbol{P} = \left[\begin{array}{cccccc} \multicolumn{3}{c}{\multirow{3}{*}{\boxed{\boldsymbol{P}_M}}} & 0 & \cdots & 0 \\ & & & \vdots & & \vdots \\ & & & 0 & \cdots & 0 \\ 0 & \multicolumn{2}{c}{\boxed{\boldsymbol{P}_M^{(M)}}} & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & & \\ 0 & \cdots & 0 & \multicolumn{3}{c}{\boxed{\boldsymbol{P}_M^{(M)}}} \end{array} \right] . \tag{B.19}$$

It can be noticed that the elements of $\boldsymbol{P}_M^{(M)}$ "shift" to the right by one entry per line down, because no more than M transmission successes are allowed with a window of length M, even if more than M packets are left to be transmitted. In turn, this causes some states to be unreachable in only one transition. $\boldsymbol{T}$ and $\boldsymbol{E}$ are similarly constructed as follows:

$$\boldsymbol{T} = \begin{bmatrix} 0 & T(1) & \cdots & T(M-1) & T(M) \cdots T(M) \end{bmatrix}, \tag{B.20}$$

$$\boldsymbol{E} = \begin{bmatrix} 0 & E(1) & \cdots & E(M-1) & E(M) \cdots E(M) \end{bmatrix}. \tag{B.21}$$

In (B.19), the superscript $^{(i)}$ denotes the ith row of the corresponding matrix. If $N_x \le M$, $\boldsymbol{P}$ is formed by extracting the submatrix containing the first N_x rows and columns of $\boldsymbol{P}_M$, whereas $\boldsymbol{T}$ and $\boldsymbol{E}$ contain the first N_x elements of $\boldsymbol{T}_M$ and $\boldsymbol{E}_M$, respectively. The overall delay and energy consumption can then be calculated using the theory of renewal reward processes, by solving a linear system of equations of the form

$$\xi_{i0} = \chi_i + \sum_{k \neq 0} p_{ik} \xi_{k0}, \quad i = 1, \ldots, N_x, \tag{B.22}$$

where χ_i is the average metric (here, delay or energy) accumulated in state i before the transition. The desired variable is $\xi_{N_x 0}$, that represents the average metric accumulated when reaching state 0 from state N_x. To calculate the average delay $D^{(MR)}$, χ_i is set equal to $\boldsymbol{T}(i)$, *i.e.*, the ith element of $\boldsymbol{T}$. Accordingly, to calculate the average energy consumption $E^{(MR)}$, $\chi_i = \boldsymbol{E}(i)$. Note that, in this last case, the results must be divided by the total number of packets to be sent. It is worth highlighting that S&W reduces to a special case of M–S&W with $M = 1$ if a feedback message is always sent after any packet reception, either correct or erroneous.

Parallel transmission

This policy is meant to exploit the larger bandwidth available on shorter links. More specifically, it works like the relaying policy, except for the fact that instead of sending all packets to the subsequent node, each relay sends in parallel a fixed number of packets k directly to the destination node. We suppose that both the center frequencies and bit rates are properly chosen so that the two signals can be transmitted in non-overlapping bands. This policy will yield higher energy consumption, due to the longer transmission range, but also shorter delays, since each relay has to transmit only $N - k$ packets to the next neighbor, resulting in $(N - k)h(h + 1)/2$ total packet transmissions on the short links.

In order not to use S&W ARQ, which is inefficient if propagation delays are long, we choose to increase the SNR to a value that ensures a very low probability p_t that even a single packet is lost. Denoting by $\varepsilon = 1 - p_t$ the probability that all packets sent on the long link are received correctly, the required SNR SNR_0' must satisfy

$$\varepsilon = \left(1 - \frac{1}{2}\mathrm{erfc}\sqrt{SNR_0'}\right)^{kL}, \tag{B.23}$$

which yields

$$SNR'_0 = \left[\mathrm{erfc}^{-1}\left(2(1-\varepsilon^{1/kL})\right)\right]^2 . \tag{B.24}$$

Then the transmission power can be calculated as

$$P_T(d_i) = \frac{SNR'_0 B(d_i) N_{d_i}}{A_{d_i}^{-1}} , \tag{B.25}$$

where d_i is the distance separating the ith node from the destination, $i = 1, \ldots, h$. Note that as such, $P_T(d_i)$ will decrease with decreasing distance. All remaining $N - k$ packets are relayed through the nodes downstream from the source, and follow the same rules as for ordinary relaying.

The total delay for this policy depends on the number of relays between the first transmitter and the destination, as well as on the number of packets sent over the short links. Specifically, the delay is the maximum between

$$D_a^{(PT)} = (N-k)\frac{h(h+1)}{2}\left(\frac{T_d + 2\tau_{d_r}}{p} + T_a\right) \tag{B.26}$$

and

$$D_b^{(PT)} = hkT_d + \tau_{d_\ell} . \tag{B.27}$$

Note that $D_b^{(PT)}$ corresponds to the time needed by each node to transmit k packets plus one propagation delay (no feedback is received for packets sent directly to the sink). We assume that each node begins its transmission exactly after overhearing the kth packet from the node upstream, so that the packets sent on the long links arrive back-to-back to the sink. The energy consumption for this policy can be calculated as

$$E^{(PT)} = (N-k)\frac{h(h+1)}{2}\left(\frac{T_d P_T(d_r) + (T_d + 2\tau_{d_r})P_R}{p} + (P_T(d_r) + P_R)T_a\right) \tag{B.28}$$

$$+ \sum_{k=1}^{h} P_T(d_k) k T_d + \tau_{d_\ell} . \tag{B.29}$$

Also note that the last two policies listed so far, namely relaying with M–S&W and parallel transmission, can be blended into a third policy that makes use of both M–S&W ARQ (on short links) and long-range transmissions, in order to achieve a further delay improvement.

B.1.4 Results

Setting and parameters

In the scenario considered for this analysis, the distance between the first node and the sink is $50\,\mathrm{km}$. Each node generates $N = 10$ packets. Each data packet is $L = 2000\,\mathrm{bits}$ long, whereas ACKs are $200\,\mathrm{bits}$ long. The transmit power P_T is calculated so as to guarantee a target SNR of $8\,\mathrm{dB}$ at the next relay. For long-range transmissions, the target probability of error has been set to $p_t = 10^{-3}$ (the transmission power is set to guarantee a suitable SNR to

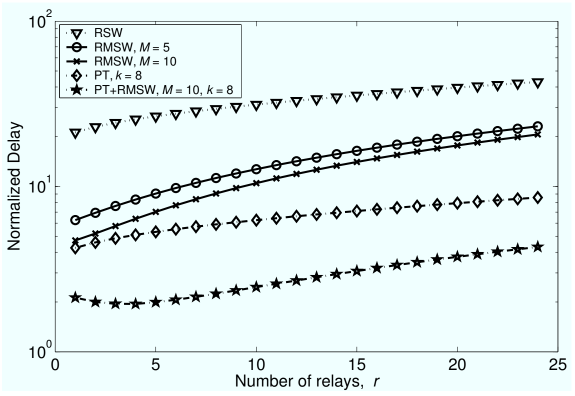


Figure B.2. *Normalized transmission delay vs. number of relays.*

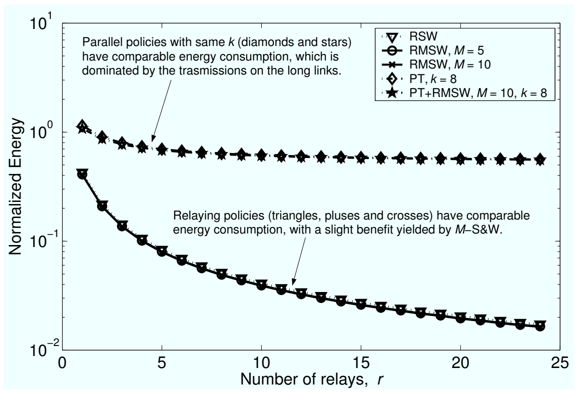


Figure B.3. *Normalized energy per packet vs. number of relays.*

enforce p_t). The channel model as outlined in Section B.1.1 is fully taken into account. The sound propagation speed in water has been fixed to $1.5\,\mathrm{km/s}$. For both long- and short-range transmissions, the data rate is fixed to $1\,\mathrm{kbps}$. The receive power is assumed to be negligibly small with respect to the transmit power. The performance of the various techniques has been evaluated analytically in terms of the total delay and total power consumption. Both metrics have been normalized for easier comparison, the delay to the time it takes to transmit all the hN packets to the sink, the energy to the amount consumed for transmitting one packet from the farthest node to the sink, assuming no errors in both cases.

Performance Analysis

In all pictures presented hereon, we will use the following abbreviations: RSW (Relaying with S&W), RMSW (Relaying with M–S&W), PT (Parallel Transmission).

Figures B.2 and B.3 illustrate the results. The first and most important observation is that parallel transmissions offer a considerably shorter average data delivery time with respect to hop-by-hop relaying, and more so for greater values of k. The corresponding price is the increase in the average per-packet energy consumption, driven by the need to ensure a higher SNR on longer links. Noticeably, an appropriate selection of k is necessary to make the energy increase worthy by improving the delay performance. From Figure B.3, it is clear that the relaying with S&W or M–S&W yields almost the same energy consumption, and that this amount is much smaller than with any other configuration of the parameters of the parallel transmission policy. Additionally, Figure B.2 shows that a more effective ARQ policy such as M–S&W can outperform parallel transmissions from the point of view of delay for some values of the number of relay nodes, if k is not properly chosen. This fact motivates us to blend parallel transmissions and M–S&W, so as to achieve the delay reduction advantages of both techniques and compensate more efficiently for the energy consumption increase.

Figures B.4 and B.5 add more to this argument, by showing the variation of delay and

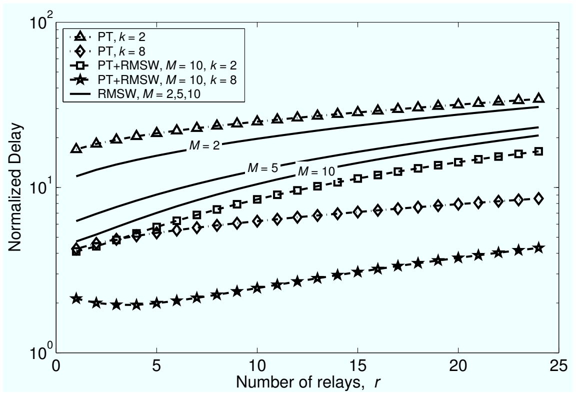


Figure B.4. *Normalized delay for varying M in RMSW vs. number of relays, compared to all policies.*

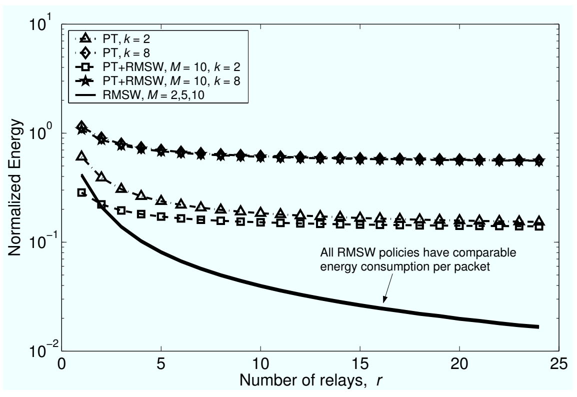


Figure B.5. *Normalized energy per packet for varying M in RMSW vs. number of relays, compared to all policies.*

energy consumption for relaying with M–S&W, for different values of M, the length of the burst of packets after which a cumulative ACK is sent. Figure B.5 suggests that relaying with M–S&W yields basically the same energy consumption for any value of M, and that joining parallel transmissions and relaying with M–S&W helps keep the expenditure lower even for the more energy-demanding parallel transmission policies. Delay is impacted more significantly by the choice of parameters. For example, the parallel transmission policy with $k = 2$ is outperformed by all relaying with M–S&W policies, for any chosen value of M.[2]

If $k = 8$, parallel transmissions yield better performance than relaying with M–S&W but at the price of a 5- to 7-fold increase in energy. On the contrary, using an effective relaying policy with M–S&W (*e.g.*, with $M = 10$) along with parallel transmissions with a low $k = 2$, improves considerably the parallel transmission delay performance, and may become even better than relaying with M–S&W from the point of view of energy consumption, if a low number of relays is present. A different design strategy with $k = 8$ and $M = 10$, instead, would give even further delay advantages, at the price of the same energy increase.

To provide more insight into the effects of k on the performance of parallel transmissions with and without M–S&W, Figures B.6 and B.7 show the delay and energy metrics for varying k in the parallel transmission policy. It is clear from Figure B.6 that a suboptimal choice of k does not decrease the average delay sufficiently to compensate for the energy increase. For example, while $k = 2$ is outperformed by any other relaying with M–S&W policy, for $k = 6$ the performance improvement begins to become noticeable, whereas $k = 8$ gives better performance than all relaying with M–S&W policies investigated. Again, parallel transmissions with M–S&W ARQ on short links yields the best delay results if k is sufficiently high, though being more energy-demanding.

Figure B.8 finally summarizes the observations reported so far, by comparing energy against delay for a number of different policies. Each curve is spanned counterclockwise

[2]Note that increasing M yields progressively smaller incremental advantages, which is in agreement with the results in [6].

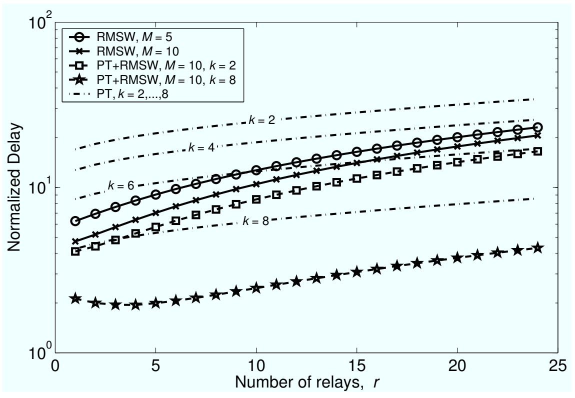


Figure B.6. *Normalized overall delay for varying k in PT vs. number of relays, compared to all policies.*

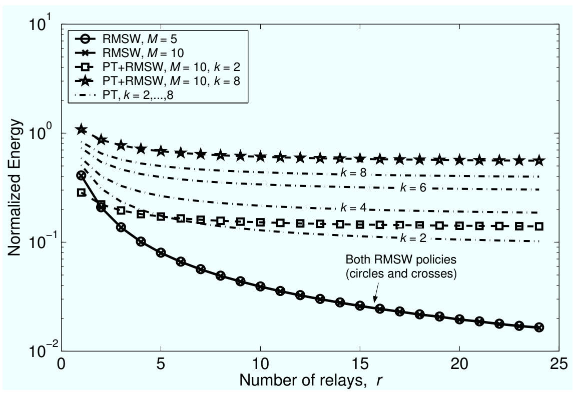


Figure B.7. *Normalized energy per packet for varying k in PT vs. number of relays, compared to all policies.*

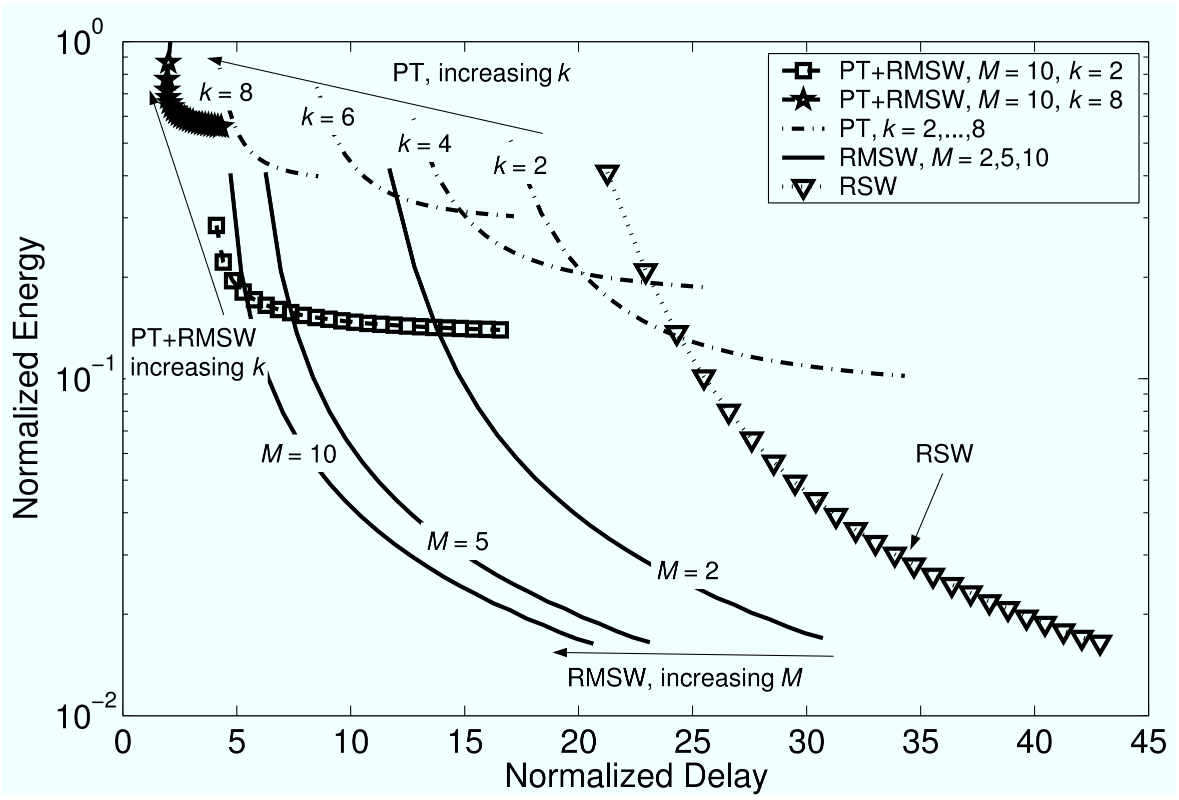


Figure B.8. *Energy per packet vs. delay comparison for all policies. Curves are spanned in a counter-clockwise sense by increasing the number of relays.*

from its top-left point by increasing the number of relays. The most "favorable corner" in this picture is the bottom-left one, where both energy and delay are low. We notice that any parallel transmissions policy immediately increases the energy consumption and yields significant delay improvements only for high k, motivating the integration of parallel transmissions and relaying with M–S&W. Moreover, increasing k limits the variation of the curves with respect to the energy axis, as long-range transmissions become the most important contributors to the energy consumption. In summary, relaying with S&W or M–S&W yields the most favorable energy performance, whereas using parallel transmissions (with or without relaying using M–S&W on the short links) can provide shorter delay at the price of a possibly substantial energy increase.

B.1.5 First Conclusions and Future Directions

Even though these results are preliminary, they allow to draw some first conclusions on the effectiveness of the analyzed transmission strategies. In particular, the use of parallel transmissions over both a short-range, high-frequency link and a long-range, low-frequency one is a feasible solution. Nevertheless, the policy parameters should be carefully set, especially the number of packets to send over longer links. This choice affects energy consumption and must be made carefully. We also pointed out that the different policies offer a way to trade off delay for energy consumption in underwater networks.

Future work on this topic will include modeling the channel fading and the effect of packet coding techniques on delay and energy performance. For now, we present in the next Section one last piece of work that stems from the analysis carried out here.

B.2 Toward an Efficient Broadcasting Protocol

Since parallel transmissions have proven to offer an advantage in the time required to complete data forwarding, we wish to exploit this to build an efficient broadcasting protocol. Broadcasting is a fundamental primitive whose application to underwater networks has received little attention to date. Reliable broadcast is required by many different applications, with the general objective of spreading a certain piece of information, or for more specialized tasks, such as in-network node reprogramming.

Broadcasting yields a different perspective into the previous scenario. While before each node had its own data to send, here the nodes cooperate in the spreading of a common message, and can thus boost even more their level of cooperation. In particular, as certain parts of a message can be overheard by certain nodes, these nodes could request the retransmission of only part of the broadcast when needed. This makes long-range transmissions more meaningful, as even if they consume more energy, they allow the broadcast to be spread more quickly.

In the following, three reliable broadcasting protocols (SBRB, FSBRB, and DBRB) are presented. These protocols are designed to address the specific challenges of the underwater channel. To confirm the validity of the approach, these protocols are compared to two standard reliable broadcast protocols through extensive simulation. In this context, we show that the designed protocols provide significant gains in terms of both energy consumption and time to complete the broadcast. Moreover, our results demonstrate the importance of addressing the peculiar relationship between bandwidth and distance exhibited by an underwater acoustic channel. In particular, we devise a way to leverage the bandwidth–distance relationship in order to reduce the number of transmissions required to complete the broadcast, with the further goal of minimizing both the overall energy consumed and the total time it takes to complete the broadcast. It is important to account for both of these metrics in the underwater environment, due to the already high energy costs of a communication and to the extremely long propagation delays.

B.2.1 Reliable Broadcast Protocols

In this section, we begin by presenting two base-line protocols, called SRB and FSRB, and our three broadcast protocols, called SBRB, FSBRB, and DBRB. Reliable broadcast protocols have been studied in detail in both wired [8] and radio-based [9,10] network environments. The main problem for reliable broadcast protocols is to efficiently correct errors affecting different parts of the message at different nodes, while avoiding retransmission storms. One way to solve this is to use forward error correction (FEC) to encode the block of packets. Then, the FEC block itself can be transmitted either proactively, or reactively, in the event of a loss. Packet FEC block codes are characterized by the number of segments of an encoded block of data that are required to successfully decode the entire message. For example, consider a Reed-Solomon code [11]. If k segments of data are encoded into a block of n packets, then a node can correct up to $\frac{n-k}{2}$ errors or $n-k$ erasures (*i.e.*, errors known to have taken place). Therefore if, *e.g.*, CRC codes are used to check the correctness of the received packets, a node can reconstruct the whole message from any k out of the n segments.

However, FEC cannot guarantee reliability. If the error rate of the channel increases beyond the corrective ability of the code, retransmission must be resorted to. Hybrid FEC/ARQ schemes for multicast and broadcast have been used in both wired and wireless environments [7] to help reduce retransmissions. In fact, in a single stream of packets, different nodes may lose different packets: in that case, FEC reduces the implosion of retransmission requests, and ARQ handles the losses FEC was not able to compensate for.

The following subsections detail three protocols we evaluated for reliable broadcast in underwater environments. The first and second protocols have two versions each, one without FEC and one using hybrid FEC/ARQ. The third protocol always makes use of FEC. For each protocol, we refer to the entire content of the broadcast as the *broadcast message*. Each broadcast message is divided into a number of packets, depending on the minimum transmission unit of the acoustic modem and the size of the message. Each packet contains a header with unique packet numbers and the total number of packets making up the broadcast message. We also consider some restrictions on what it means to reliably broadcast a message to all nodes in a network. First, we assume that no partitions exist in the network. Second, we do not consider node failures. Node failures essentially have two effects on reliable broadcast performance: the failed node will not receive the message, and a network partition could result. Since neither condition can be solved via a broadcast protocol, we believe these assumptions are reasonable.

Simple Reliable Broadcast (SRB)

The first protocol, Simple Reliable Broadcast (SRB), is not specifically suited to the underwater environment and is used as our base-line for experiments. With SRB, every node, upon receiving the broadcast message, re-broadcasts it to all its neighbors. In the event that one or more packets in a message are not received by a node, the node waits until no broadcast packets are overheard for a pre-defined time interval, and broadcasts a retransmission request to its neighbors. Upon receipt of this request, the neighbors contend for the channel by randomly choosing a backoff time in the interval. The node whose timer expires first retransmits the packets. The delay in requesting retransmissions allows time for the normal rebroadcasts from neighboring nodes to correct the transmission errors at the cost of some delay, that is indeed kept local, and does not significantly impact the dissemination delay. Channel contention at the MAC layer for SRB and all other protocols presented hereon use a carrier-sense collision avoidance (CSMA) MAC layer protocol similar to the one proposed in [12]. This protocol was developed to minimize collisions in the underwater environment.

If the error rate of the channel (either due to noise or collisions) is sufficiently high, such that retransmissions are consistently required, FEC is a good solution to achieve higher reliability without increasing the overall traffic. In the FEC version of SRB (FSRB) each message is encoded before packet transmission begins. We assume that the same encoding mechanism is used by each node; this minimizes the computational cost of message forwarding, as a node that receives all of the encoded packets does not need to decode and then re-encode the message before repeating the broadcast. Hence, SRB ensures reliability only by retransmission requests, whereas FSRB employs FEC to correct errors and resorts to retransmissions only if FEC fails.

It should be noted that with SRB and FSRB, every node repeats the broadcast. This potentially adds a number of transmissions that are not necessary in reasonably dense networks (*i.e.*, where nodes have multiple neighbors). Indeed, it has been shown that, for current radio devices, the most energy-efficient broadcast strategy is to use the maximum transmit power to reach the greatest number of neighbors with each transmission [13]. In underwater acoustic networks, however, transmitting to the longest possible distance at the highest power is not the most energy-efficient solution [14]. The protocol described in the next Section addresses this problem by leveraging the bandwidth–distance relationship exhibited by the acoustic channel.

Single-Band Reliable Broadcast (SBRB)

Essentially, Single-Band Reliable Broadcast expands SRB by employing long range communication to notify all neighbors that a broadcast has started, and then using shorter-range transmissions to send the messages to neighboring nodes. If any node does not receive the whole broadcast after a time interval, it asks its neighbors for retransmissions as specified hereon.

When a node wants to originate a broadcast, it sends a long-range signal in the appropriate frequency range at a high power level, notifying the greatest possible number of nodes of the forthcoming transmission. It should be noted that all communications other than the broadcast initiation signal take place in a high-frequency, low-power band, and thus do not collide with these long range notification signals. Along with the high power used to notify the new broadcast, this significantly reduces the probability that any notification signal will be lost.

After advertising the beginning of the broadcast, the source uses the larger bandwidth and lower power enabled by short-range communications to send the broadcast to its nearest neighbors. Once a node successfully receives the entire message, it contends for the channel to begin its own transmission using its short range bandwidth and power. If the node loses the contention and receives the broadcast from one of its neighbors, it does not attempt to forward the message any more. If the node wins the contention, it sends the first packet of the message to notify the nearest neighbors, and then switches to the long-range bandwidth and sends a broadcast notification to communicate to all other nodes that the broadcast is making progress. Once this message is sent, the node completes the broadcast transmission on the short-range band. Finally, if the node loses the contention, but does not hear the broadcast message being forwarded by any neighbor, it re-contends for the channel to transmit the broadcast.

If any segments of the broadcast should be lost, the node waits for the channel to be idle for a certain time interval and then sends a retransmission request to its neighbors, as in SRB. In case a node has received the long range broadcast notification, but no broadcast packets, it waits a longer time interval, then requests the broadcast from its neighbors and resets the timer. If a second long timeout occurs, it expands its transmission radius and requests the broadcast message again. This process continues until some node having the broadcast message receives the request and initiates a retransmission.

Note that, similar to SRB, SBRB achieves reliability through retransmission requests, whose impact is reduced through longer timeouts and slowly expanding request reaches. For this reason, SBRB is more likely to be affected by losses (fewer redundant transmissions of the broadcast message are sent). Hence, FSBRB, the FEC-enhanced version of SBRB, should exhibit much better performance. Similar to FSRB, FSBRB encodes the message prior to initiating the broadcast.

Dual–Band Reliable Broadcast (DBRB)

The final protocol, Dual–Band Reliable Broadcast (DBRB), operates similarly to SBRB, except that instead of sending short broadcast message notifications on the long distance, high power band, it uses this band to send some FEC data that can be used by nodes to correct errors. Every node that repeats the broadcast sends some redundant data using the long distance band. As a consequence, after a small number of sensors have retransmitted the broadcast, many nodes throughout the network are likely to possess sufficient redundancy to reconstruct the message completely. This allows the amount of redundancy sent on the long distance links to be tuned, with the aim to spend less energy and yet ensure a reasonable error-correction ability. However, before reaching optimal performance, this protocol experiences a transient phase that continues until sufficient redundancy is available to correct errors on the first forwarders. In fact, before redundancy spreads, these forwarders tend to require retransmissions more frequently.

B.2.2 Simulation Results

To test the performance of the five protocols described above, we implemented them in a simulator we designed to test underwater networks. Our simulator fully accounts for the model of the underwater channel as described in Section B.1.1.

A carrier-sense collision avoidance (CSMA) MAC layer protocol similar to the one proposed in [12] was used for channel access. It provides mechanisms to effectively operate in the presence of the long delays found in the underwater environment. This protocol maintains a low number of collisions per packet, except in very high traffic situations. Hence, most of the packet error events in the network come from collisions, with only a small portion being accounted for by ambient noise.

In addition to modeling the error due to noise and attenuation, as well as MAC layer collisions, we use an additional parameter in the MAC layer that allows us to increase or decrease the packet loss probability, in order to test the protocols over a wide range of error rates.

We used a topology generation algorithm that randomly places nodes in a $5\,\mathrm{km} \times 5\,\mathrm{km} \times 5\,\mathrm{km}$ network, and then adjusts node placement to make a fully connected network given the transmission range. We generated many scenarios by varying this range between $100\,\mathrm{m}$ and $2\,\mathrm{km}$. Additionally, we varied the number of nodes placed in the network between 40 and 700.

We model the acoustic modem after the WHOI micromodem specifications [15], with transmit powers between $10\,\mathrm{W}$ and $50\,\mathrm{W}$. The short-range frequency band used is $22\,\mathrm{KHz}$–$55\,\mathrm{KHz}$ whereas the long-range frequency band spans from $4\,\mathrm{KHz}$ to $13\,\mathrm{KHz}$. We used $96\,\mathrm{byte}$ packets and varied the number of packets per broadcast message from 1 to 20. Additionally, for protocols using FEC, we fixed the error correcting ability to a single packet of the broadcast message. Further optimizations for the broadcast protocols include adding adaptive FEC to tune the correction capabilities as the error rate on the channel changes.

All experiments represent the averages over 40 runs with different random topologies

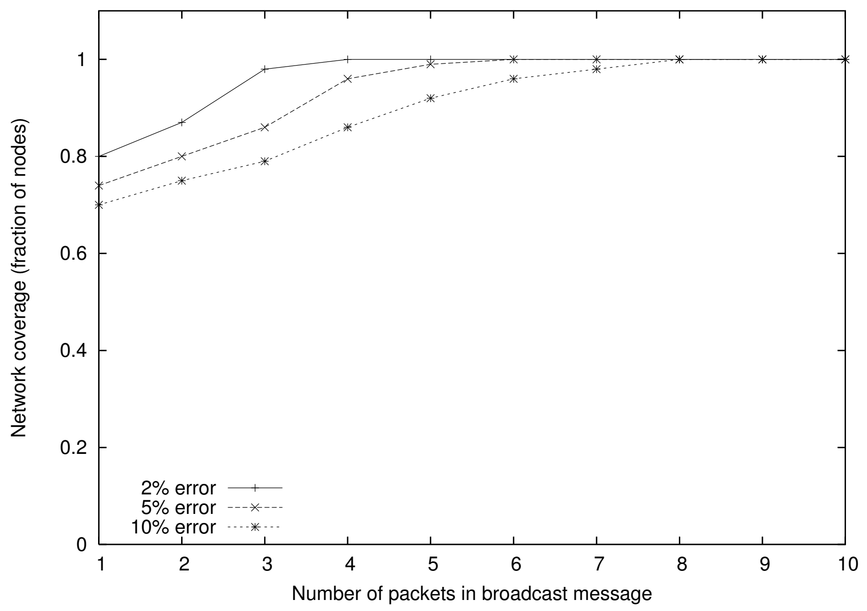


Figure B.9. *Coverage,* 512 *nodes.*

generated for each run. For all experiments, we maintained 95% confidence intervals between 1% and 2.5% of the averages.

B.2.3 Coverage

There are two components to a fully reliable broadcast protocol. The first concerns whether a node, once it receives one packet of the broadcast, receives the entire broadcast message successfully. The second concerns whether all nodes in the network receive the broadcast message. This latter concern we call *coverage*. In any wireless network, without pre-knowledge of all of the nodes in the network, it is impossible to guarantee full coverage. Reliable broadcast protocols in this environment can only promise statistical coverage. The protocols in this work deal with this situation. The network coverage of a reliable broadcast protocol is complete only if all nodes in the network are reached. As a sample situation, consider a node that is placed on the edge of the network and has only one neighbor. If that node fails to receive a broadcast packet from its only neighbor, there is no way for it to know that a broadcast is being propagated, and thus it will never request a retransmission. Additionally, there is no completely sure way for the sender to know if a node in the network has not received its broadcast. For our protocols, once a receiving node obtains part of a broadcast, it will receive the entire broadcast, through the use of either FEC or repeated retransmissions. Having multiple packets make up a single broadcast message increases the chances that the entire network will be covered, by increasing the probability that each node will receive at least one packet of the message.

Figure B.9 depicts the fraction of nodes that successfully receive the broadcast message using SBRB as a function of the number of packets making up the broadcast message for

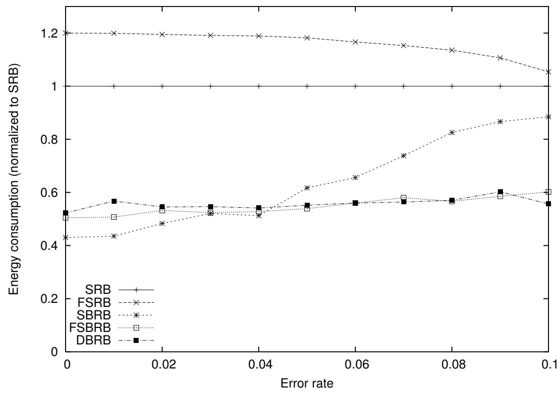


Figure B.10. *Energy consumption,* 100 m *min transmission range.*

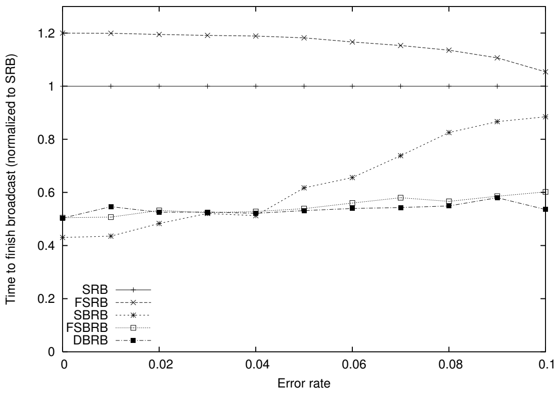


Figure B.11. *Completion time,* 100 m *min transmission range.*

different error probabilities. For all error rates tested, the fraction of nodes receiving the broadcast message converged to one after the number of packets was increased to 9. For the remainder of experiments, the broadcast message size was held constant at 10 packets (the FEC encoded messages were 12 packets long).

B.2.4 Varying the Error Rate

In the next set of experiments, we varied the error rate of the channel to see its effects on the energy consumption needed to complete the broadcast and the time from the initiation of the broadcast to when the last packet is received. In this section, we present results from 4 different network densities. Each of these network densities requires a different minimum transmit power that can be used and still have a connected network. In turn, this power translates to a minimum distance that must be covered with each transmission for keeping the network connected. This parameter is important, since it affects the energy cost of each transmission in terms of energy, as both the transmit power and the transmission time are increased at longer distances.

Figures B.10 and B.11 depict the energy consumption and time to completion, respectively, as the error rate on the channel is increased. For both graphs the y-axis is normalized to the performance of SRB. The key observation is that for error rates lower than about 3%, SBRB outperforms FSBRB and DBRB by 8% to 5%. At higher error rates, the savings from using FEC become critical and FSBRB and DBRB begin to outperform SBRB. In all cases, the gains achieved by leveraging the bandwidth–distance relationship for SBRB, FSBRB, and DBRB produce considerable savings, consuming around 40% to 50% of the energy required by SRB and FSRB. As one would expect, the time to completion follows roughly the same curve, since energy increases in these cases are directly related to the need for more transmissions. Additionally, the delay increase incurred in SBRB for using the long-range band for notification is completely dominated by the total number of transmissions needed to complete the broadcast. This is not the case in networks tested with 5 to 20 nodes, where the

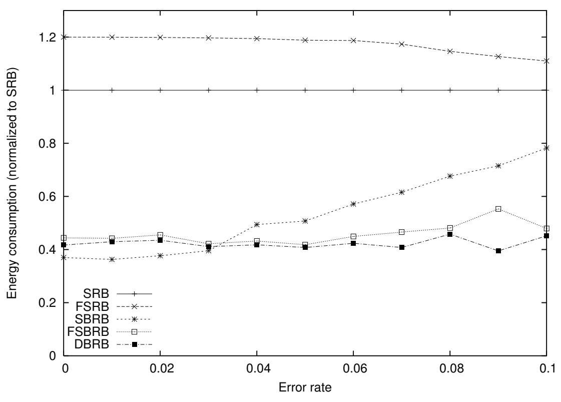


Figure B.12. *Energy consumption,* 500 m *min transmission range.*

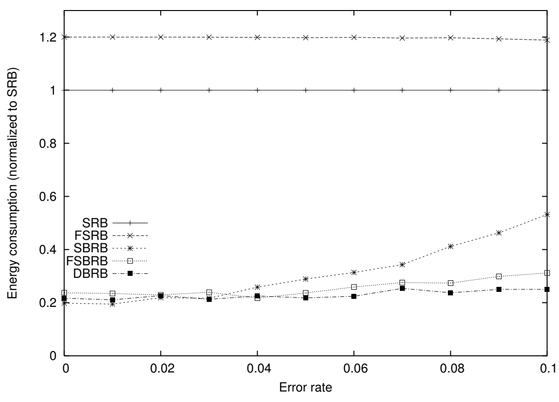


Figure B.13. *Energy consumption,* 1.2 km *min transmission range.*

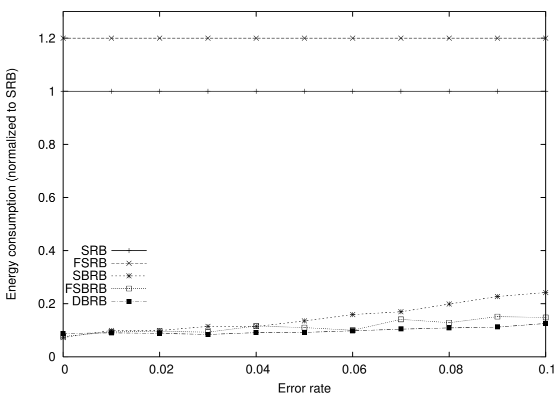


Figure B.14. *Energy consumption,* 2 km *min transmission range.*

added times for the longer range transmissions could be seen in the graphs; however, SBRB and DBRB still outperformed SRB and FSRB by no less than 16%. Since the time results are along the same lines as the energy results for all experiments in this section, we omit them due to space considerations.

Figures B.12 to B.14 depict the energy consumption normalized to SRB for minimum transmission ranges of 500 m, 1.2 km, and 2 km, respectively. As the transmission range required to avoid network partitions increases, SRB performs progressively worse. This is directly due to the large number of transmissions still needed and the increase in the cost of transmissions. Furthermore, at higher error rates and longer distances, DBRB begins to outperform FSBRB by about 3%. This suggests that, at longer minimum distances, if an adaptive FEC were used by DBRB, significant improvements over FSBRB could be realized.

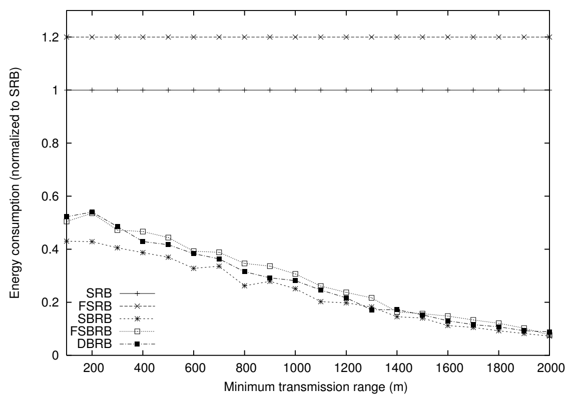


Figure B.15. *Energy consumption,* 0.00 *error parameter.*

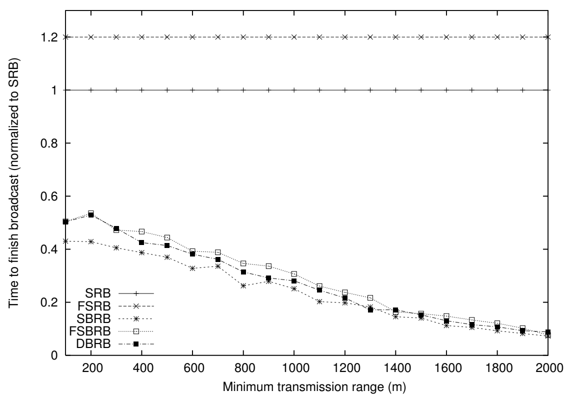


Figure B.16. *Completion time,* 0.00 *error parameter.*

B.2.5 Varying the Minimum Transmission Range

To more accurately view the effects of minimum transmission ranges on the performance of the various protocols, in this section, we present results that vary the transmission range for different average error rates.

Figures B.15 and B.16 depict the energy consumption and time to completion respectively, as the minimum transmission range to avoid network partitions is increased. For both graphs the y-axis is normalized to the performance of SRB. The first thing to notice is that minimum transmission range has no effect on FSRB, because every node repeats the message. As the transmission range increases, the total cost for sending a message increases; therefore, SBRB, FSBRB, and DBRB all save increasingly more energy compared to SRB as the minimum transmission range increases, due to their ability to minimize the number of transmissions required to complete the broadcast. As with the results in the previous subsection, the time results follow the energy trends as expected, and are thus omitted for the rest of the experiments in this Section.

Figures B.17 and B.18 depict the energy consumption as a function of the minimum transmission distance for error parameters of 5% and 10% respectively. It is worth noting that the increasing separation between SBRB, FSBRB, and DBRB as error rate increases is not affected by changes in minimum transmission distance. In other words, no matter what minimum transmission distance, if the error rate is high, DBRB outperforms the other protocols by an average of 3%.

B.3 Related Work

The use of acoustics for underwater communication has received increased interest in recent years. While the main use of acoustic waves is still sonar detection and ranging, as well as telemetry [16], relatively recent efforts have proven that reliable links can be set up

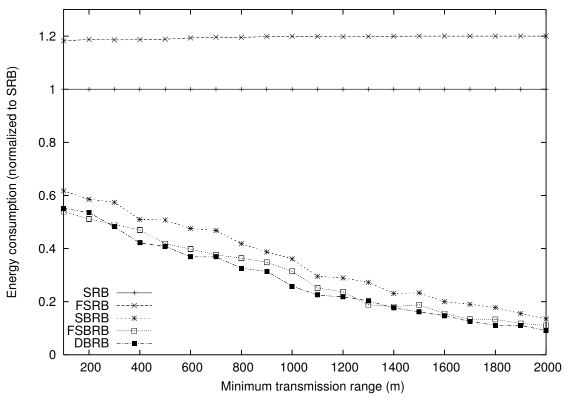


Figure B.17. *Energy consumption,* 0.05 *error parameter.*

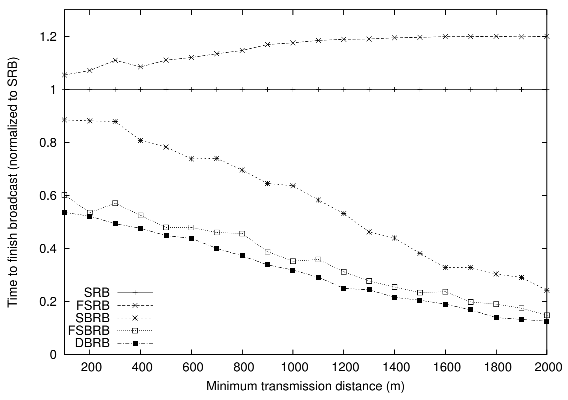


Figure B.18. *Energy consumption,* 0.10 *error parameter.*

in water, using signal processing techniques that provide good communication efficiency or speed [17–20].

While there are still many open issues in building underwater acoustic networks [21], some papers have contributed to the design of MAC protocols. A discussion of deterministic multiple access schemes for underwater networks was presented in [22]. A more comprehensive comparison of such schemes in clustered environments has been more recently carried out in [23]. Other protocols have been more specifically tailored to the underwater acoustic channel features. For example, Slotted FAMA [24] focuses on collision avoidance. It sets up shared synchronization among the sensors, whereby the time is divided into slots sufficiently long to accommodate for the maximum round-trip time in the network. Transmissions are preceded by an RTS/CTS handshake, and may take place only at the beginning of a slot, ensuring that the channel is sensed busy when another transmission is going on. PCAP [25] also pursues collision avoidance. It makes the duration of a handshake predictable, by inserting proper waiting intervals before the transmission of the CTS. This delays the setup of the link enough to "simulate" the maximum propagation delay between the two nodes. In turn, this allows the transmitter to carry out other tasks while waiting for the receiver to reply. The approach in [26] is quite different, as collision control is sought instead of avoidance. The nodes perform an RTS/CTS exchange and wait before transmitting data. During this wait time, the recipient may hear another RTS meant for another node and could be able to warn the transmitter in time to avoid the collision. Also, if the transmitter hears another RTS during the waiting time, it delays its transmission for the same reason. The protocol in [26] cannot avoid collisions completely. However, the reduced length of the waiting times ensures a globally greater throughput, outperforming Slotted FAMA. Moreover, there is no need to maintain node synchronization. Another protocol specifically tailored to underwater acoustic networks is UWAN–MAC [12]. It is designed to save energy through very low duty cycles, and focuses on collision avoidance through a sort of adaptive TDMA. Each node has an awake/sleep schedule and transmits when awake. Upon synchronizing with their neighbors through special packets, the nodes know when to wake up to hear nearby transmissions. All data packets carry schedule information so as to reduce the total overhead. HELLO packets are used to recover from erroneous synchronizations, such as waking up and hearing no transmission by the intended sender. The authors in [27] argue that the difference between transmit and receive power can be exploited in underwater networks and discuss how to manage idle time in that light. The conclusion is that near-optimal energy performance can be reached if ultra-low power transducer wakeup modes could be implemented. On a similar line of thought, Tone-Lohi [28] tries to avoid collisions by sending very short busy tones, that could be heard by other nodes during idle channel monitoring.

From the point of view of routing, most of the literature is focused on the adaptation of terrestrial radio protocols to the underwater environment. For example, Vector-Based Forwarding [29] lets nodes compute the angle of arrival of an overheard acoustic signal to understand their position with respect to a cylindrical area connecting the source to the destination. If a node is inside such an area, it is allowed to forward the packet. To achieve the maximum advancement, the eligible forwarders delay their operations proportionally to their distance from the sender, so that nearby nodes refrain for a longer time and can overhear the packets forwarded by farther neighbors. Segmented Data Reliable Transport (SDRT) [30] employs FEC to guarantee error protection. Each node encodes and forwards data continuously using a simplified version of Tornado codes, until some positive feedback is received. To avoid wasting too much energy, packet transmissions are "windowed:" the packets inside the window are transmitted at full rate, whereas a lower rate is used for those outside the window. Each receiver must decode the whole block of data before transmitting again. In [31], the authors deploy a framework for addressing delay-sensitive and -insensitive applications, involving Reed-Solomon packet coding and scheduling of packets according to their delay requirements. The focus of the investigation is on the impact of the long delays and stronger attenuation of the acoustic channel on packet routing. The variation of the available bandwidth with distance is taken into account in [14]. The conclusion is that there exists an optimal hop distance from an energy consumption point of view. Moreover, the authors infer that routing protocols should be designed to match such a distance, or if possible, to approach it from below (choosing closer neighbors), considering linear as well as 3-dimensional topologies.

Notice that even though [30, 31] rely on packet FEC as we do, they only transmit over a single channel that is suited to all nodes in the network. Unlike [30, 31], the approach we have taken here is different. We have devised different techniques to exploit the change in the available bandwidth with varying distance to convey FEC toward all nodes, with the aim to perform reliable broadcasting in an underwater sensor network.

B.4 Conclusions and Future Directions

Underwater acoustic networks have characteristics that are very different from their terrestrial, radio-based counterparts. One of the most important differences is the relationship between the available bandwidth and the transmission distance. In this Appendix, we have described preliminary results obtained on this topic. First, we have presented the analysis of a number of transmission policies that exploit the features peculiar to the underwater channel. Secondly, we have presented the design of three reliable broadcast protocols that leverage this relationship to outperform standard, radio-based broadcast approaches in terms of energy consumption and time to complete the broadcast. Our three protocols, Single-Band Reliable Broadcast, FEC Single-Band Reliable Broadcast, and Dual-Band Reliable Broadcast each build on the ideas derived from the analysis and leverage the ability to use small bands to transmit long distances to alert nodes that broadcasts are to be expected. Then, by reducing the transmission range and selecting only certain nodes from each neighborhood to repeat the broadcast, the protocols dramatically reduce the total number of transmissions required.

Once again, the results suggest that a PHY-aware protocol design outperforms an approach that is oblivious to the physical layer.

Future work includes investigating the use of adaptive FEC in FSBRB and DBRB. Our results suggest this optimization should produce even larger savings, though it is unclear whether DBRB or FSBRB would perform best. Additionally, FEC coding schemes that are not block codes may allow significant advantages for both protocols.

Acknowledgment

The analysis of transmission strategies to exploit the underwater channel presented in this Appendix has received great benefit from the insightful discussions with Milica Stojanovic, Massachusetts Institute of Technology, Cambridge, MA, USA.

The design of the broadcasting protocols for underwater networks has been done in collaboration with Albert F. Harris III, University of Illinois at Urbana–Champaign, IL, USA.

References

[1] R. Urick, *Principles of Underwater Sound*. McGraw-Hill, 1983.

[2] The National Center for Atmospheric Research, “Temperature of Ocean Water,” http://www.windows.ucar.edu/tour/link=/earth/Water/temp.html&edu=high.

[3] Office of Naval Research, “Ocean Water: Salinity,” http://www.onr.navy.mil/Focus/ocean/water/salinity1.htm.

[4] L. Berkhovskikh and Y. Lysanov, *Fundamentals of Ocean Acoustics*. Springer, 1982.

[5] M. Stojanovic, “On the relationship between capacity and distance in an underwater acoustic communication channel,” in *Proc. of ACM WUWNet*, Los Angeles, CA, Sep. 2006, pp. 41–47.

[6] M. Stojanovic, “Optimization of a data link protocol for an underwater acoustic channel,” in *Proc. of IEEE Oceans*, Brest, France, Jun. 2005, pp. 68–73.

[7] L. Rizzo and L. Vicisano, "RMDP: An FEC-based Reliable Multicast Protocol for Wireless Environments," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 2, no. 2, pp. 23–31, Apr. 1998.

[8] J.-M. Chang and N. Maxemchuk, "Reliable broadcast protocols," *ACM Transactions on Computer Systems*, vol. 2, no. 3, pp. 251–273, Aug. 1984.

[9] E. Pagani and G. P. Rossi, "Reliable Broadcast in Mobile Multihop Packet Networks," in *Proc. of ACM/IEEE (MobiCom)*, 1997.

[10] S.-T. Sheu, Y. Tsai, and J. Chen, "A Highly Reliable Broadcast Scheme for IEEE 802.11 Multi-hop Ad Hoc Networks," in *Proc. of IEEE ICC*, 2002.

[11] B. Sklar, *Digital Communications: Fundamentals and Applications*, 2nd ed. Prentice Hall PTR, 2001.

[12] M. K. Park and V. Rodoplu, "UWAN-MAC: an energy-efficient MAC protocol for underwater acoustic wireless networks," *IEEE J. Oceanic Eng.*, 2007, to appear. [Online]. Available: http://www.ece.ucsb.edu/rodoplu/Pubs/ParkRodoplu_UWANMAC.pdf

[13] C. Sengul and R. Kravets, "Heuristic approaches to energy-efficient network design problem," in *Proc. of IEEE ICDCS*, June 2007.

[14] A. F. Harris III and M. Zorzi, "Energy–efficient routing protocol design considerations for underwater networks," in *Proc. of IEEE SECON*, Jun. 2007.

[15] L. Freitag, M. Grund, S. Singh, J. Partan, P. Koski, and K. Ball, "The whoi micro-modem: An acoustic communications and navigation system for multiple platforms," http://www.whoi.edu, 2005.

[16] D. B. Kilfoyle and A. B. Baggeroer, "The state of the art in underwater acoustic telemetry," *IEEE J. Oceanic Eng.*, vol. 25, no. 1, pp. 4–27, Jan. 2000.

[17] M. Feder and J. A. Catipovic, "Algorithms for joint channel estimation and data recovery-application to equalization in underwater communications," *IEEE J. Oceanic Eng.*, vol. 16, no. 1, pp. 42–55, Jan. 1991.

[18] W. Qian and J. A. Ritcey, "Spatial diversity equalization applied to underwater communications," *IEEE J. Oceanic Eng.*, vol. 19, no. 2, pp. 227–241, Apr. 1994.

[19] M. Stojanovic, "Recent advances in high-speed underwater acoustic communications," *IEEE J. Oceanic Eng.*, vol. 21, no. 2, pp. 125–136, Apr. 1996.

[20] ——, "Retrofocusing techniques for high rate acoustic communications," *Journal of the Acoustical Society of America*, vol. 117, no. 3, pp. 1173–1185, Mar. 2005.

[21] I. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: research challenges," *Elsevier's Ad Hoc Networks*, vol. 3, no. 3, 2005.

[22] E. M. Sozer, M. Stojanovic, and J. G. Proakis, "Underwater acoustic networks," *IEEE J. Oceanic Eng.*, vol. 25, no. 1, pp. 72–83, Jan. 2000.

[23] P. Casari, S. Marella, and M. Zorzi, "A comparison of multiple access techniques in clustered underwater acoustic networks," in *Proc. of IEEE/OES OCEANS*, Aberdeen, Scotland, Jun. 2007.

[24] M. Molins and M. Stojanovic, "Slotted FAMA: a MAC Protocol for underwater acoustic networks," in *Proc. of IEEE Oceans*, Singapore, Sep. 2006.

[25] X. Guo, M. Frater, and M. Ryan, "A propagation-delay-tolerant collision avoidance protocol for underwater acoustic sensor networks," in *Proc. of IEEE Oceans*, Singapore, Sep. 2006.

[26] B. Peleato and M. Stojanovic, "A MAC protocol for ad hoc underwater acoustic sensor networks," in *Proc. of ACM WUWNet*, Los Angeles, CA, Sep. 2006, pp. 113–115.

[27] A. F. Harris III, M. Stojanovic and M. Zorzi, "When underwater acoustic nodes should sleep with one eye open: idle–time power management in underwater sensor networks," in *Proc. of ACM WUWNet*, Los Angeles, CA, Sep. 2006, pp. 105–108.

[28] A. Syed, W. Ye, and J. Heidemann, "Medium Access Control for Underwater Acoustic Networks," in *Proc. of ACM WUWNet*, Los Angeles, CA, Sep. 2006, work-in-progress paper.

[29] P. Xie, J.-H. Cui, and L. Lao, "VBF: vector-based forwarding protocol for underwater sensor networks," UCONN CSE, Tech. Rep. UbiNet-TR05-03 (BECAT/CSE-TR-05-6), Feb. 2005.

[30] P. Xie and J.-H. Cui, “SDRT: a reliable data transport protocol for underwater sensor networks,” UCONN CSE, Tech. Rep. UbiNet-TR06-03, Feb. 2006.

[31] D. Pompili, T. Melodia, and I. F. Akyildiz, “Routing algorithms for delay-insensitive and delay-sensitive applications in underwater sensor networks,” in *Proc. of ACM Mobicom*, Los Angeles, CA, Sep. 2006, pp. 298–309.

Appendix C

Complete List of Papers

This Appendix presents a complete list of papers published, accepted or submitted during the Ph.D. program. For convenience, the papers are grouped according to their main topic.

C.1 Papers on MIMO Ad Hoc Networks

The work on MIMO ad hoc networks in Chapter 2 has been presented in [1–9]. In particular, [1,2] describe preliminary studies on the performance of the MIMO multiuser detection algorithm, applied to ad hoc networks. The approximations to the detection algorithm and their effect on the simulation of ad hoc networks are described in [3, 4]. The work in [5] presents an in-depth description of the DSMA algorithm introduced in Section 2.8. DSMA is compared to backoff techniques in a comprehensive study presented in [6]. Two journals, extensively detailing the work on ad hoc networks, have been submitted for publication on the IEEE Transactions on Wireless Communications. In particular, [7] (appeared on December 2007) focuses on the study of approximations to the MIMO PHY performance, whereas [8] focuses on the cross–layer design of the network. Finally, one paper describing the implications of this cross–layer design on routing [9] is currently submitted to IEEE IWCMC 2008.

C.2 Papers on Wireless Sensor Networks

The work on wireless sensor networks in Chapter 3 has been presented in [10–16]. In particular, [10] presents an extensive simulation study of Geographic Random Forwarding (GeRaF) and the intuition behind the benefits to be obtained by handling awaking nodes in a contention. The first papers appeared about ALBA are [11, 12]. In particular, [11] deals with the protocol definition and the comparison with other similar approaches, while [12] introduces the Rainbow algorithm to face the dead end problem. A poster describing ALBA and Rainbow has also been presented, and its abstract has appeared in the ACM Mobile

Computing and Communications Review [15]. Two journal papers are currently submitted. The work on ALBA has been submitted to the IEEE Transactions on Mobile Computing [13], whereas a detailed GeRaF analysis has been submitted to the IEEE Transactions on Wireless Communications [14].

A parallel work on relieving the congestion at the sink in converge–casting scenarios (not presented in this thesis) has also been published in the proceedings of the IEEE PIMRC 2007 [16].

C.3 Papers on Underwater Acoustic Sensor Networks

The work on underwater acoustic sensor networks in Appendix B has been presented in [17–21]. In particular, [17] details the analysis of transmission strategies for underwater acoustic networks, and [18] presents a broadcasting protocol designed with the features of the underwater channel in mind. Both these works are presented in Appendix B as well.

Thanks to the collaboration with two thesis students, two more studies have been carried out. A comparative analysis of different multiple access techniques in clustered topologies is presented in [19], while an in-depth analysis of the UWAN-MAC protocol for underwater networks is detailed in [20]. Finally, a paper on optimal broadcasting policies for underwater networks using hybrid ARQ and fountain codes [21] has been accepted as an invited paper to WONS 2008.

References

[1] P. Casari, M. Levorato, and M. Zorzi, "Some issues concerning MAC design in ad hoc networks with MIMO communications," in *Proc. of WPMC*, Aalborg, Denmark, Sep. 2005.

[2] ——, "On the implications of layered space–time multiuser detection on the design of MAC protocols for ad hoc networks," in *Proc. of IEEE PIMRC*, Berlin, Germany, Sep. 11–14, 2005.

[3] M. Levorato, S. Tomasin, P. Casari, and M. Zorzi, "An approximate approach for layered space–time multiuser detection performance and its application to MIMO ad hoc networks," in *Proc. of IEEE ICC*, Istanbul, Turkey, Jun. 2006.

[4] ——, "Analysis of spatial multiplexing for cross–layer design of MIMO ad hoc networks," in *Proc. of IEEE VTC-Spring*, Melbourne, Australia, May 2006.

[5] P. Casari, M. Levorato, and M. Zorzi, "DSMA: an access method for MIMO ad hoc networks based on distributed scheduling," in *Proc. of ACM IWCMC*, Vancouver, Canada, Jul. 2006.

[6] M. Levorato, P. Casari, and M. Zorzi, "On the performance of access strategies for MIMO ad hoc networks," in *Proc. of IEEE GlobeCom*, Nov. 2006, pp. 1–5.

[7] M. Levorato, S. Tomasin, P. Casari, and M. Zorzi, "Physical layer approximations for cross–layer performance analysis in MIMO–BLAST ad hoc networks," *IEEE Trans. Wireless Commun.*, vol. 6, no. 11, pp. 4390–4400, Dec. 2007.

[8] P. Casari, M. Levorato, and M. Zorzi, "Cross–layer design of MIMO ad hoc networks with layered multiuser detection," *IEEE Trans. Wireless Commun.*, submitted.

[9] ——, "On the design of routing protocols in MIMO ad hoc networks under uniform and correlated traffic," in *Proc. of IEEE IWCMC*, Chania, Crete Island, Jun. 2008, submitted.

[10] P. Casari, A. Marcucci, M. Nati, C. Petrioli, and M. Zorzi, “A detailed simulation study of geographic random forwarding (GeRaF) in wireless sensor networks,” in *Proc. of IEEE MilCom*, Atlantic City, NJ, Oct. 2005.

[11] P. Casari, M. Nati, C. Petrioli, and M. Zorzi, “ALBA: an adaptive load–balanced algorithm for geographic forwarding in wireless sensor networks,” in *Proc. of IEEE MilCom*, Washington, DC, Oct. 2006.

[12] ——, “Efficient non–planar routing around dead ends in sparse topologies using random forwarding,” in *Proc. of IEEE ICC*, Glasgow, Scotland, Jun. 2007.

[13] ——, “ALBA: an integrated cross–layer solution for load–balancing geographic routing in wireless sensor networks,” *IEEE Trans. Mobile Comput.*, submitted.

[14] ——, “A detailed analytical and simulation study of geographic random forwarding (GeRaF),” *IEEE Trans. Wireless Commun.*, submitted.

[15] ——, “Poster abstract: Geographic forwarding and adaptive load balancing in wireless sensor networks,” *ACM Mobile Comp. and Commun. Review*, vol. 11, pp. 53–54.

[16] P. Casari, F. Zorzi, and M. Zorzi, “Efficient packet converge–casting: relieving the sink congestion in wireless sensor networks,” in *Proc. of IEEE PIMRC*, Athens, Greece, Sep. 2008.

[17] P. Casari, M. Stojanovic, and M. Zorzi, “Exploiting the bandwidth–distance relationship in underwater acoustic networks,” in *Proc. of MTS/IEEE Oceans*, Vancouver, Canada, Sep. 2007.

[18] P. Casari and A. F. Harris III, “Energy–efficient reliable broadcast in underwater acoustic networks,” in *Proc. of ACM WUWNet*, Montreal, Canada, Sep. 2007.

[19] P. Casari, S. Marella, and M. Zorzi, “A comparison of multiple access techniques in clustered underwater acoustic networks,” in *Proc. of IEEE/OES OCEANS*, Aberdeen, Scotland, Jun. 2007.

[20] P. Casari, F. E. Lapiccirella, and M. Zorzi, “A detailed simulation study of the UWAN-MAC protocol for underwater acoustic networks,” in *Proc. of MTS/IEEE Oceans*, Vancouver, Canada, Sep. 2007.

[21] P. Casari, M. Rossi, and M. Zorzi, “Towards optimal broadcasting policies for HARQ based on fountain codes in underwater networks,” in *Proc. of IEEE/IFIP WONS*, Garmisch-Partenkirchen, Germany, Jan. 2008, invited paper.