NTNU
Innovation and Creativity

Energy-Efficient Link Adaptation and Resource Allocation in Energy-Constrained Wireless Ad Hoc Networks

Even Krogsveen

Master of Science in Communication Technology

Submission date:	June 2007
Supervisor:	Geir Egil Øien, IET
Co-supervisor:	Changmian Wang, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem Description

Wireless ad hoc networks are quickly emerging as the most interesting new paradigm for wireless technology in a variety of applications, including industrial automation, sensor networks, personal area networks, military applications, home networks, and so on. This MSc thesis project will only be focusing on the communication aspects of such networks, not the type of data or the applications running on top. Since the nodes in most applications of such networks will be battery-powered, energy-efficiency is a matter of great concern.

The MSc project will commence with a literature study where the aim is to collect and summarize knowledge about the state-of-the-art in research within this field as well as to propose promising avenues for further research. In particular short-range applications, where both transmission energy and hardware energy consumption is of importance, are targeted. Both solutions for high-capacity, low-latency applications as well as low-capacity, medium-to-long latency applications are of interest. Evaluation of the channel state information needed to perform link adaptation is also an issue.

After the initial literature study, the emphasis will be on development and evaluation of novel energy-efficient link adaptation schemes for energy-constrained wireless ad hoc networks, based on knowledge gained from the literature about promising research directions. In particular, the schemes reported in [Goldsmith] and [Yang] are to be used as a background for the solution of a particular route configuration optimization problem. The project is associated with Workpackage 2 "Energy Efficient Link Adaptation" in the Nordic (NORDITE/NFR) collaborative research project CROPS: Cross-layer optimization in short-range wireless sensor networks (cf. http://www.s3.kth.se/commth/projects/CROPS/).

Responsible supervisor: Prof. Geir E. Øien, rom C349, oien@iet.ntnu.no
Co-supervisor: PhD student Changmian Wang

Assignment given: 17. January 2007
Supervisor: Geir Egil Øien, IET

Acknowledgements

This thesis is the concluding part of my 5-year Master of Science degree at the Norwegian University of Science and Technology, department of electronics and telecommunications. The given problem description is inspired by the ongoing NORDITE Cross-Layer Optimization in Short-Range Wireless Sensor Networks (CROPS) project. The thesis has also resulted in a paper, which is submitted to the the IEEE International Symposium on Wireless Communication Systems (ISWCS) '07, awaiting assessment. Co-authors are Geir Øien, Changmian Wang and Saska Lindfors.

A special thank goes to my supervisors, professor Geir Øien and PhD student Changmian Wang, for their guidance and support. I would also like to thank Dr. Anna Kim and professor Saska Lindfors for valuable input and constructive criticism.

Even Krogsveen,
Trondheim, June 2007

Abstract

Wireless ad hoc networks have a number of advantages over traditional, infrastructure-based networks. Robustness and easy deployment are two of the main advantages. However, the distributed nature of such networks raises a number of design challenges, especially when energy-efficiency and QoS requirements are to be taken into consideration. These challenges can only be met by allowing closer cooperation and mutual adaptation between the protocol layers, referred to as a *cross-layer* design paradigm.

In energy-constrained wireless ad hoc networks, each node can only transmit to a limited number of other nodes directly. Hence, in order to reach distant destinations, intermediate nodes must relay the traffic of their peer nodes, resulting in *multihop* routes. The total energy consumption associated with a end-to-end transmission over such a route can be significantly reduced if the nodes are correctly configured. A cross-layer, optimization scheme, based on adaptive modulation and power control, is proposed in this thesis. The optimization scheme assumes that an existing route has been found, and allows QoS requirements in terms of end-to-end bit error rate and delay. Both transmission and circuit energy consumption is taken into consideration. By jointly optimizing all nodes throughout the route, the total energy consumption can be reduced by more than 50%, compared to a fixed-rate system. The adaptive system also exhibits superior capabilities to meet stringent QoS requirements.

Results for both continuous and discrete rate adaptation is produced, and it is found that discrete adaptation causes only a small performance degradation, compared to the optimal, continuous case. Simulations also show that the system is vulnerable to inaccurate link state information. Finally, the effects of maximum-rate limitation and ignoring the circuit power consumption is investigated.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Wireless ad hoc networks is one of today's most interesting emerging technologies. The development of this agile network technology has opened for a vast number of new and exciting applications, in which flexibility and easy deployment and configuration are important properties. An ad hoc network consists of nodes that must communicate with each other without relying on any infrastructure or pre-defined hierarchy. This is in contrast to e.g. WLAN and cellular mobile phone systems, in which base stations coordinate the user nodes and control all traffic flows. In other words, an ad hoc network must be self-organizing and function without any dedicated controlling unit.

A self-configuring, autonomous wireless network obviously has a huge potential in a great number of situations. Data networks made up of handheld devices, 'smart' homes, communication and data systems for military field operations, industrial control systems and wireless sensor networks are just a few examples where wireless ad hoc networks can be applied. Common for all these examples is that they utilize the distributed nature and independence of fixed infrastructure inherent in wireless ad hoc networks.

A distributed system is by nature much more error-tolerant than one with some form of centralized control, since there is no single critical point of error. This property is of major importance for many military applications and sensor networks where nodes are placed in a hostile environment, where a high rate of node failure is to be expected. Many redundant nodes will increase the robustness of the system. For other applications, the reduced time and cost of system deployment when no infrastructure is needed, or where existing infrastructure cannot be used, is the most useful aspect of an ad hoc network. This is true for many commercial applications, both industrial and domestic, and also military.

In short, there is an abundance of applications that could utilize wireless ad hoc networks. But there are also a number of technological challenges associated with this kind of networks systems. The nodes are typically small, often battery-powered and can be highly mobile. Some applications have stringent QoS requirements, while others might have a required system lifetime measured in decades. The distributed nature of wireless ad hoc networks, combined with these requirements makes existing protocols and techniques, developed for fixed and wired networks, unsuitable. This calls for novel thinking that challenges the traditional design paradigms.

1.1 Related Work, Definitions and Scope

Wireless ad hoc networks are a hot topic for research, resulting in a great number of published research papers and articles. As pointed out, there are countless conceivable applications for this type of networks, all with different design requirements and limitations. This makes it impossible to define at a common framework that encompass them all [1]. However, there are some key design goals, common for all applications of wireless ad hoc networks. These will form the basis for this thesis.

It is commonly accepted that wireless ad hoc networks calls for a rethinking of the traditional, strictly layered OSI reference model for protocol design. Many of the challenges encountered in application of wireless ad hoc networks can only be solved by allowing a closer integration between the protocol layers, commonly referred to as a *cross-layer* design paradigm [2, 3, 4].

Energy-efficiency is another point emphasized in the vast majority of relevant publications, since energy reserves are usually considered to be finite. In e.g. [5, 6], it is shown that energy awareness must be incorporated in all protocol layers in order to obtain truly energy-efficient solutions, thereby motivating the cross-layer design paradigm. The limited resources of the network nodes imply that the transmission range is also limited [3, 7]. In order to transmit to remote destinations, the traffic must therefore be relayed by intermediate networks nodes to the final destination, creating a *multihop* route [1].

Another recurring topic is link adaptation, which is shown to offer tremendous performance improvements in the time-varying environment of a wireless communication system [2]. This is particularly important in resource-limited wireless ad hoc networks, and link adaptation is shown to greatly improve the performance of such networks, e.g. in [8, 9].

The distributed nature of wireless ad hoc networks makes QoS sensitive traffic extra challenging. Nevertheless, many applications generate traffic that requires some form of QoS capabilities, e.g. multimedia sensor networks [10].

This brief overview defines the area of focus for this thesis, and the design goals and mentioned will be further investigated. To summarize, energy-constrained, short-range wireless ad hoc networks with QoS sensitive traffic will be the main focus. Only the communication aspect of such systems will be reviewed. The focus will be on the middle to lower layers.

ZigBee and Bluetooth [11, 12], are two commercially available technologies that are targeted at many of the same applications as described here. Further, they share many of the same characteristics, such as emphasis on energy-efficiency, multihop routing (ZigBee) and distributed network control. However, these standards does not fully utilize the cross-layer design approach [3, 13], and will not be further investigated here.

1.2 Objective

This thesis has two main objectives:

- To give a brief introduction to the domain of wireless ad hoc networks, and to point out some of the most important related design goals and challenges, as well as proposed techniques on how these problems can best be met. Cross-layer design and its advantages will be emphasized, with focus on the middle to lower protocol layers.
- To investigate a route optimization problem, using a cross-layer approach. The problem is defined based on knowledge acquired by the initial literature study, and the proposed solution is based on relevant research papers. An in-depth presentation of the system model and the optimization scheme will be given. Then the model is implemented in Matlab, simulations are run and conclusions drawn.

1.3 Outline

The rest of the thesis is organized as follows:

Section 2 Literature survey. Further motivation for cross-layer design paradigm. Brief overview of important design goals for lower through middle protocol layers.

Section 3 Presentation of system model, the route optimization problem and the proposed solution.

Section 4 Description of simulations and numerical results.

Section 5 Result analysis and review of the model, based on the results.

Section 6 Final conclusions and propositions for further work.

Chapter 2

Background and Context

The flexibility of wireless ad hoc networks opens for a number of new, exciting applications. Since no infrastructure is needed, new applications can be quickly and easily deployed. The distributed nature also makes applications fault-tolerant, as there is no single point of which the functionality of the entire system depends. Figure 2.1 visualizes the structure of a generic ad hoc network.

There are many technological challenges associated with the design of wireless ad hoc networks, many of which remains unsolved. The field is a very hot topic for research, and there is clearly a demand from the industry for this technology. New techniques and protocols are frequently proposed. However, the distributed nature of wireless ad hoc networks always carries with it an inherent penalty in terms of degraded performance, compared to centrally controlled networks [1, 2].

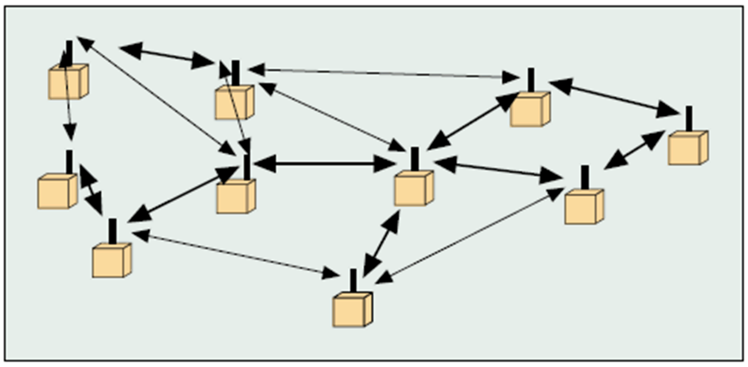


Figure 2.1: Conseptual sketch of wireless ad hoc network. ([1, Fig. 1])

The problems and limitations inherent in wireless ad hoc networks are in many applications acceptable, in light of the increased flexibility and possibilities such networks present. Due to the vast variety of applications of wireless ad hoc networks, each with different requirements, it is difficult to define a common framework for the design of such systems; especially since in order to achieve acceptable performance and energy efficiency, many system designs need to be tailored to a specific application [1, 14, 15]. This is in particular true for systems with stringent QoS requirements [3, 4, 10].

But there are a number of important design goals and key challenges that are relevant for most applications of wireless ad hoc networks, of which the cross layer design paradigm is the most important. The reminder of this section will give a brief motivation for this design paradigm, and present important related design issues for the three bottom protocol layers.

2.1 Motivation for Cross-Layer Design

Traditionally, the design of any data network or communication system have been based on the generic OSI reference model [2, 16]. E.g. the well-known 5-layer TCP/IP stack, the workhorse of the Internet, is based on the OSI model.

A fundamental design principle of this model is modularity. Each layer is designed to perform its tasks in isolation, and offer its services to the neighboring layers. The implementation and operational details, however, are hidden within each layer. Communication between the layers only takes place through well-defined interfaces, and only between neighboring layers. This approach makes it easy to modify any layer without interfering with the operation of the others. The high degree of modularity also makes understanding and designing networks easier. And finally, it allows equipment from different vendors cooperate seamlessly, as long as they obey the standards [2, 16, 17].

Whereas the principle of modularity has its obvious benefits, it also dictates some limitations. This is especially true when considering wireless networks. The OSI model was originally designed for wired networks [2, 16] which are fundamentally different from wireless networks in a number of important areas, some of which are summarized here. The principle of cross-layer design is illustrated by Figure 2.2. A cross-layer design (right) allows information flow across layers, thereby allowing joint adaptation and optimization, while in the traditional model (left), each layer it performs its operations in isolation, only capable of self-adaptation. This generic, 5-layered model is frequently used as a reference model in the literature, and will also be adopted here.

Channel Fluctuations

A wireless communication channel is in nature far more unstable than a wired one. Fading, shadowing, interference, doppler shifts and refractions

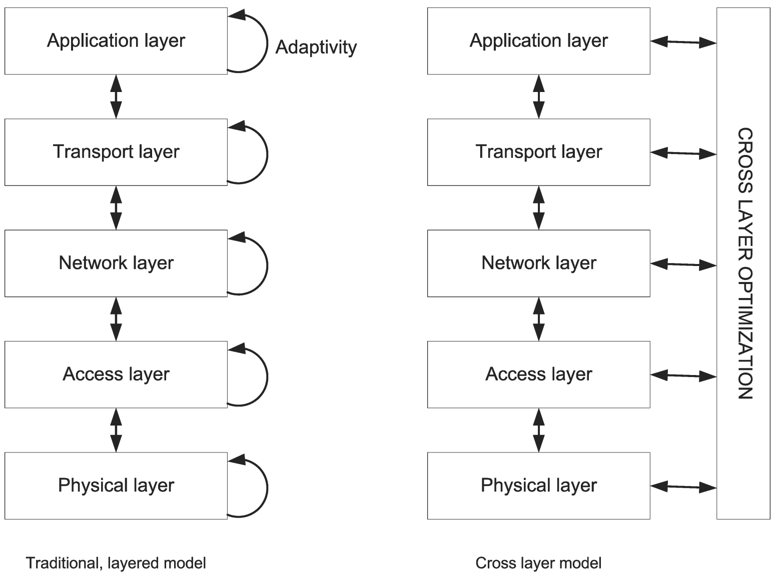


Figure 2.2: Reference model for protocol stack. Copy of [2, fig. 16.2 (modified)]

all contribute toward making the wireless channel a challenging medium for communications [1, 2, 18].

Relatively stable, time-invariant wired channels with small fluctuations in link quality and Signal to Noise Ratio (SNR) can appear to be transparent for most practical purposes by over-designing the physical protocol layer slightly, thereby compensating for the small variations that do exist [6]. That is, the capacity of the channel appears to the higher protocol layers as constant, and the details about the transmission medium can remain hidden within the physical protocol layer, as dictated by the OSI model [16].

Wireless channels, however, exhibit much larger variations, and the capacity of the channel will drop when the link quality, referred to as the link state, is low. The channel capacity can be increased by increasing the transmit power [2], but in resource-limited wireless nodes, the transmission power is usually limited. In other words, adaptation within the physical layer in order to even out these fluctuations is often not possible. The information about the channel should therefore be shared with the higher layers, so that they can adapt their operation to the time-varying communication channel.

Distributed network control combined with resource-limited nodes and time-varying conditions makes QoS sensitive traffic a particularly difficult challenge. This topic is given much attention in the literature. It is a common conclusion that allowing closer cooperation between protocol layers and mutual adaptation have the potential of providing better performance while lowering overall energy consumption, as compared to the traditional, layered approach. Link adaptation is a key tool in order to obtain these goals [7, 8, 9, 15].

Multiple Access

The design of multiple access protocols for wireless ad hoc networks presents some major challenges. Without any fixed infrastructure, it is up the nodes themselves to coordinate their transmission in an efficient manner. This is a highly complex problem, since all nodes share the same communication medium. To save energy, nodes that are not transmitting or receiving data should ideally be powered down to conserve energy [1, 5]. But many nodes must also act as routers, and forward packets from other nodes to their final destination. Also, the transmission scheduling can be adapted to the time-varying nature of the wireless channel, e.g. by transmitting only when channel conditions are good, called *opportunistic scheduling* [2]. This illustrates that optimal scheduling of transmissions is impossible without taking the operation of the other layers into consideration.

Network Topology and Routing

Routing principles for wireless ad hoc networks and their wired counterparts are fundamentally different. Wired networks will be mostly static, with infrequent topology changes and relatively stable communication links. There exists a number of routing algorithms for finding optimal paths in this type of networks, based on different routing metrics [16]. The relatively stable environment makes it possible to separate the routing operation from details in the lower layers.

In wireless ad hoc networks, on the other hand, routing protocols must be able to adapt to fluctuations in the underlying channel characteristics, as well as frequent topology changes (in mobile networks) [1]. QoS sensitive traffic complicates the problem even further. On top of this, energy awareness should be incorporated into the routing metrics, to maximize the network lifetime [15]. All this shows that routing in wireless ad hoc networks is a complicated problem that cannot be solved in isolation.

Summary

As pointed out, protocols designed for static, wired networks will not perform very well if used directly on a wireless network, especially when stringent QoS requirements must be satisfied. There are many problems in wireless ad hoc networks that cannot be addressed by any single layer. There is broad acceptance in the literature for the fact that re-thinking of the original, strictly layered protocol stack reference model is necessary when considering wireless ad hoc network systems, and that the stringent design requirements can only be met by utilizing the possibilities this model provides [3, 6, 15, 19].

2.2 Design Goals and Issues

As mentioned, the design of a wireless ad hoc network is highly dependent on the applications. E.g. a sensor network with nodes embedded in a concrete bridge to monitor the structural integrity must have a lifetime of years or decades, but the requirements on delay and throughput are minimal. A totally different scenario is the temporarily deployment of video surveillance equipment, e.g. in a urban riot area. This application will have a much shorter lifetime, maybe a few days, while the data traffic and QoS requirements will be much more demanding. The number of nodes in these two applications will also be very different. In the first scenario, some thousand nodes could be deployed, to ensure sufficient redundancy and reliability, while in the second, perhaps only 10 - 50 nodes is required.

This simple example illustrates the diversity of applications in which wireless ad hoc networks can be utilized. To narrow down this vast field, some ground assumptions must be made.

The network nodes are typically battery powered, so that the energy reserves are finite. In some applications, energy-harvesting from the environment is possible [4, 14], and batteries can be replaced for others. Nevertheless, energy should in most cases be considered as a scarce resource. For all techniques and applications discussed in this thesis, energy efficiency is emphasized as a key design goal.

Energy-limited devices are also usually peak-power limited [1]. That is, the instantaneous power that can be drained from the batteries is limited. Limited power means limited transmission range [7, 20]. Thus, many applications of wireless ad hoc networks can be classified as *short range*, e.g. Wireless Sensor Networks (WSNs) and applications for home automation [1, 21]. While this term is not well-defined, the definition from [3] will be adopted here: Short range systems is interpreted as systems with communication ranges (between the nodes) so short that the the nodes' circuit power consumption is of the same order of magnitude as power used for communication.

2.2.1 Hardware

The hardware platform is where all the higher layer protocols are implemented. Functionality defined by higher layer protocols must be supported in the hardware system design, and virtually all design choices made here will affect the other layers [2, 5, 15]. This thesis will not give an in-depth discussion about energy-efficient hardware design, but simply point out the most important aspects and desired properties of a generic low-power hardware platform.

First of all, different application have different requirements. This calls for tailored hardware design to deliver the best possible performance. At the same time, many applications of wireless ad hoc networks, such as home automation systems and WSNs, rely on cheap, mass-produced hardware to become commercially feasible. This leads to the first of many trade-offs: Tailored hardware will be more expensive, but will perform better in the targeted application(s), whereas a more generic hardware platform would be cheaper ot produce, at the cost of reduced functionality and performance.

For all applications considered here, energy-efficiency will be a primary concern. Many papers on energy-efficiency in wireless ad hoc networks address only transmission energy. But recent research has shown that for ultra-low power applications, energy consumption associated with e.g. signal processing, coding, start-up and even leakage power when the node is powered down is non-negligible and should be included in the total energy budget.

It is shown that incorporating circuit energy consumption into the optimization process of a given system, tremendous energy savings can be obtained [5, 8, 22, 23]. A desired property of a generic hardware platform is the ability to trade performance can be traded for power savings [5]. This means that a system can adapt adapt its hardware operation to meet given requirements, and not waste valuable energy by over-performing.

2.2.2 Link Adaptation

Every communication channel have a fundamental capacity limit, which will vary in time-variant channels [2, 24]. In resource-limited wireless ad hoc networks, it is a crucial design goal to adapt to these fluctuations, so that the available resources can be utilized efficiently [1].

Link adaptation can be carried out in a number of ways, and at different protocol layers, from simple power control to sophisticated cooperative diversity schemes [2]. Power control can be used for channel inversion, so that the received Signal to Noise and Interference Ratio (SNIR) always satisfies a certain threshold. The transmission rate can also be adjusted, based on the capacity of the channel, by making use of adaptive modulation or coding [2]. These techniques are frequently utilized in research papers and proposed protocols. In [5, 6, 22], it is shown that these techniques can greatly improve the system performance, while saving energy and meeting stringent QoS requirements. However, the performance of adaptive modulation, coding and power control is dependent on the accuracy of the available link state estimates. Estimating the channel in extremely fast fading environments, e.g. highly mobile ad hoc networks, is difficult, and other techniques should be considered [2].

Multiple Input Multiple Output (MIMO) techniques can be used to mitigate the effects of fading channels, thereby dramatically increasing the capacity. Equivalently, if the data rate is constant, energy can be saved [1, 2]. This is true when only considering transmission power, but the special characteristics of energy-constrained ad hoc network nodes must be taken into consideration. MIMO-techniques require sophisticated signal processing that will require extra energy [2], which for some scenarios might actually exceed the savings obtained by applying such techniques in the first place.

Another approach for producing reliable point-to-point transmission over time-varying links is to simply retransmit packets that are received in error [9]. This can be very beneficial in systems where link accurate state estimation is difficult. In fact, retransmitting over a fast-varying channel is a simple form of time-diversity, where the number of allowed retransmissions corresponds to the diversity order [25].

Retransmission schemes does not require advanced signal processing procedures. On the other hand, performing retransmissions causes increased time- and energy consumption, both of which are limited in applications considered here. Especially, in applications with a tight delay constraints, retransmissions might not be feasible.

To summarize, appropriate link adaptation is crucial for successful design of wireless ad hoc networks. But for any adaptation scheme, any associated additional energy cost (or other penalties) must be carefully weighted against the provided benefits, bearing in mind the special characteristics, requirements and limitations of energy-constrained wireless ad hoc networks. So exactly which technique is best suited will vary from application to application.

2.2.3 Resource Allocation and Multiple Access

Medium access is another of the major challenges in wireless ad hoc networks. In particular, the lack of a single control unit makes efficient resource utilization difficult, especially when combined with stringent requirements to keeping the energy consumption as low as possible.

In any wireless communication system, the bandwidth is limited, and must be shared by all users/nodes. The available bandwidth is usually divided into a finite number of channels. This is typically done by time, frequency or code (orthogonal codes) division, or a combination of these. In order to communicate, a node must somehow be granted access to a channel.

Code division can also utilize non-orthogonal codes, thereby making the systems *interference-limited*. Here, simultaneous transmissions are allowed, but will interfere with each other. The level of interference from other users decides the capacity of each channel. In contrast to orthogonal channelization schemes, there is no hard limit on the maximum number of simultaneously transmitting users [1]. Thus, nodes can in theory transmit whenever they want, without any prior negotiating for channel access.

However, there are some difficulties associated with this scheme. First of all, semi-orthogonal code multiple access schemes are vulnerable to the near-far problem [24], thus requiring precise power control. This could lead to conflicts, as power control affects multiple layers, and should be optimized accordingly. Further, code division systems requires complex transceivers and near perfect synchronization between nodes [24], thus making their usefulness in energy-constrained ad hoc networks questionable. This is confirmed by the overweight of attention given to simpler channelizations schemes in the literature.

So in general, there will be more nodes in the networks considered here than available channels. As a consequence, the nodes must share communication channels. Due to the lack of fixed infrastructure or a centralized control mechanism, it is up to the nodes themselves to synchronize their transmissions and utilize the available communication channels in the most efficient manner possible.

When a node is not communicating, it should enter a sleep state, which is the best way to save energy [21, 26]. But a sleeping node is no longer a part of the network, and cannot participate in relaying packets for other nodes, which is crucial for keeping the network connected.

While energy efficiency traditionally has not been a concern for the access protocol layer [26], it is a key design in energy-constrained ad hoc networks. In fact, proper scheduling of transmission and sleep cycles is crucial if truly energy-efficient solutions are to be obtained [1, 6, 23]. In [5], it is shown that for ultra-low power systems, even powering up and down a node carries with it a non-negligible energy penalty, that should also be taken into consideration when deciding whether or not putting a node to sleep.

Any MAC protocol must ensure a fair sharing of communication resources. Note that the term *fair* in some applications of ad hoc networks does not necessarily mean fairness with respect to nodes. Fairness can also refer to fairness to the system as a whole. Especially in WSNs with a lot of redundancy, prolonging the network lifetime might be more important than ensuring a strict per-node fairness policy [5, 21]. Other applications generate QoS sensitive traffic. Here, strict per-node fairness must be emphasized.

There are two main approaches for granting channel access: Random access and scheduling [1]. In the first scheme, nodes contend over channel access, and the 'winner' is allowed to transmit. Scheduling refers to some deterministic schedule for channel access having been found, and all involved nodes are granted access in due time. The distributed nature of ad hoc networks obviously goes well with some form of random access scheme, while scheduling has the potential of offering a more efficient utilization of available resources.

In the simplest possible random access scheme, nodes transmit whenever they have data ready. This rather naive approach will obviously lead to many collisions, especially in a dense network. Carrier Sense Multiple Access (CSMA) is another approach, in which the nodes sense the channel before they transmit, and refrain from doing so if the channel is busy. But due to the hidden terminal problem [24], there will still be collisions. In the

IEEE 802.11 MAC protocol, this is avoided by using a RTS/CTS handshake procedure [27]. The overhead introduced by this scheme might make it inappropriate for use in energy-constrained ad hoc networks. Random access MAC protocols for wireless ad hoc networks are proposed in e.g. [26, 28], but none of these target low-energy systems in particular. Regardless, random access schemes have their main strengths in networks dominated by bursty traffic [1], since fewer collisions will occur and each node occupies the channel for a relatively short period of time when granted access.

The other approach is a deterministic scheduling of the order in which the nodes are granted access to the medium [1]. This scheme is typically used in systems with centralized control, such as cellular mobile phone systems like GSM [29]. Compared to random access, scheduling have the potential to make more efficient use of the transmission medium, especially in systems with many users and/or traffic dominated by long transmissions [1]. However, implementing scheduling in an ad hoc network (with distributed control) can be very difficult. The complexity of the scheduling problem grows fast with the network size [1], and requires accurate synchronization between nodes [30]. Both of which might pose serious problems in energy-constrained wireless ad hoc networks.

A number of MAC protocols have been proposed for wireless ad hoc networks, utilizing different techniques and targeting different applications. The importance of including circuit energy consumption in the sleep scheduling is highlighted in e.g. [15, 31, 32]. The importance of incorporating time-varying conditions into the protocol design is discussed in e.g. [19, 23, 33, 34]. The variety of proposed approaches confirms that design of MAC protocols is highly dependent on system characteristics, especially traffic patterns and QoS requirements. The cross-layer design allows many parameters to be taken into consideration, dramatically increasing the complexity of the problem, but also opens for high performance and resource utilization.

2.2.4 Topology & Routing

Routing in a wireless ad hoc network is the procedure of finding a path from one node to another. Usually, this will be a multihop route, involving intermediate relay nodes. A route is found by making decisions based on *routing metrics*, which typically include e.g. link capacities and buffer states. Although a simple procedure in principle, there are many difficulties introduced by the particular characteristics of energy-constrained wireless ad hoc networks. There will be frequent topology changes, e.g. due to node mobility and link fluctuations, and routes might be broken due to battery depletion. Higly mobile networks poses a particular challenge for applications with QoS requirements [35]. Finally, energy-awareness must be incorporated into the routing protocol. All must be taken into consideration, which greatly complicates the route setup and maintenance.

The stability of the network is decisive for which routing method is best suited. It is customary to divide routing schemes into 3 main categories; flooding, proactive and reactive routing [1].

In flooding, every packet is multiplied and forwarded to all reachable neighbors, making the probability for successful delivery at the final destination large. The redundancy makes this routing strategy the most robust, as well as the most resource-demanding. En energy-constrained networks, this approach should be avoided. But in highly mobile networks, or other systems where network topology changes extremely fast, other routing strategies will fail, and the brute-force approach of flooding might be the only alternative [1].

The other routing strategies are proactive and reactive routing. In proactive routing, routing tables are used when routing a packet through the network to their final destination. The routing tables are maintained by updating the routing information with regular intervals, which can be done in a centralized or a distributed fashion. In reactive routing, on the other hand, a route is only initiated and maintained when needed [16].

In centralized routing, all relevant network information is forwarded to a single location, which then calculates the routing tables. Thus, centralized routing will in general be able to find globally optimal routes [1]. But there are a number of issues with this strategy that is hard to combine with the special characteristics of wireless ad hoc networks, especially when energy-efficiency is important. First of all, there is a lot of overhead associated with the forwarding of routing information, and the network will respond slowly to topology changes. Route computation can be a complex problem that requires heavy computations, and the size of the routing problem grows fast with network size [2]. The idea of having a centralized unit is also in conflict with the definition of an ad hoc network, but could be interesting e.g. in small-scale WSNs, which typically have a centralized data fusion center or gateway [21].

Distributed proactive routing is more commonly used in wireless ad hoc networks [1]. In distributed routing, there is no central point at global routing information is gathered. Rather, the nodes only have knowledge of the routing information about their immediate neighbors, and form routing tables based on this information. Since routing information is only transmitted to each nodes immediate neighbors, there is far less overhead associated with distributed, compared to centralized routing. Further, the system will react to network topology changes much faster. On the downside, this strategy will typically give sub-optimal routes, since not all routing information is known where routing decisions are made.

The last main category, reactive routing, creates a route only when needed. The route setup can be either source- or destination initiated. That is, either the source or destination node, respectively, initiates the route setup procedure. When the transmission is over, the route is no longer maintained. Different variations of reactive routing is frequently proposed as a suitable choice for wireless ad hoc networks, e.g. [33, 36].

This approach saves the system the overhead of maintaining routes that are not used, at the cost of a initial setup delay. Also, this strategy will react faster to topology changes than proactive routing [2]. Reactive routing typically means that for each end-to-end transmission, there is a setup phase, in which the route is found (and configured) and a subsequent steady-state phase, where data is sent. This can be exploited to optimize a given route, subject to energy-efficiency and QoS requirements [9, 23]. The special class of routing referred to as *data-centric* or *attribute-based* routing [21] is also enabled by reactive routing, in this case usually destination-initiated. This class of routing is of particular interest for WSNs, and enables functions like ‘Get information for all nodes where temperature is over 100°C’.

In general, there is no clear answer to what routing strategy that is best suited for wireless ad hoc networks. As with most other design issues for this type of networks, this is highly dependent on the application characteristics and requirements. But in general, the large amount of overhead associated with centralized proactive routing, this is rarely a good choice for this kind of networks. Rather, some form of distributed proactive or reactive routing, or combinations of these, are preferred [37, 38].

Network topology and node mobility dictates many aspects of the design of a routing protocol for ad hoc networks [39]. It should be emphasized that network topology is not only decided by the physical positions of the nodes, but also by link adaptation schemes, interference from other nodes, and the transmission power of the nodes [1, 7, 19]. It is therefore desirable that also the network layer should be able to control transmission parameters, such as transmission power, data rate, etc. Clearly conflicts will arise when the same parameters are to be jointly optimized over multiple layers, thereby creating trade off-situations.

In systems where maximization of the network lifetimse is a major concern (typically WSNs), finding a single, minimum-energy route is not the whole solution. Using the same route heavily will cause the nodes along this route to deplete their batteries quickly, which might cause the network to become disconnected. Therefore, traffic should be evenly distributed throughout the network over time, in order to maximize the system lifetime. This can be done by cycling through different routes between a given source/destination node pair, although this might be sub-optimal from a local point of view [37, 40].

This all points to the fact that close integration between multiple protocol layers is required for designing routing protocols for wireless ad hoc networks. But while this interaction allows tremendous performance improvements, it also makes the protocols complicated and highly application-specific.

2.3 Summary

This brief overview illustrates that the field of wireless ad hoc networks carries with it a great number of technological challenges. Protocols that are not designed with the special characteristics of this type of networks in mind, will perform badly when faced with these challenges. This calls for the development of novel techniques and protocols, tailored to meet the stringent requirements of energy-constrained wireless ad hoc networks. Key features, such as energy-awareness and QoS support must be incorporated in all layers.

As pointed out, this is only possible by allowing protocol layers to cooperate more closely, thereby motivating the cross layer design paradigm. The cross-layer design paradigm offers potentially dramatic performance improvements, especially in resource-limited systems and in systems with stringent QoS requirements.

Despite the advantages of cross layer design, it also carries with it some problems. Taking all parameters into account can lead to prohibitily large optimization problems, especially as the network grows larger. Direct conflicts between layers might also occur. Transmission power control is a prime example of this, as it an important parameter in both the physical layer (link adaptation), the access layer (interference to other nodes) and network layer (network connectivity). This creates a trade-off situation, where the particular characteristics of the system in question decide what aspects should be emphasized over others.

Another important point is that the cross-layer design paradigm violates the modularity principle, which has proven itself as a highly efficient way of designing communication and data network systems [16, 2]. Closer integration between layers will make modifications and cooperations between different systems more difficult. However, many applications of wireless ad hoc networks will be autonomous systems, that could communicate with other systems through gateways. [41] highlights some of the dangers inherent in cross-layer design, such as over-complex systems, 'spaghetti-design' and the problems of further developing tailored systems.

The conclusion is that cross layer design is necessary in order to meet the requirements of wireless ad hoc network systems. It allows optimization of a system with respect to defined performance requirements, e.g. in terms of traffic throughput and QoS, across layers, thereby yielding tremendous energy savings. This is fundamentally different from other, typically infrastructure-based systems, where maximizing the overall system performance is the only objective, and energy-efficiency is not emphasized. However, care should be taken when applying cross-layer design, keeping in mind the advantages of the modularity principle.

Chapter 3

Problem Description

The previous section provided a general overview of the most important design goals and challenges of wireless ad hoc network. The focus will now be shifted to a specific problem of minimizing the energy consumption associated with a packet transmission over a given multihop route. The proposed system and optimization routine is based on some assumptions that makes it especially relevant for the field of multi-media WSNs [10], such as delay sensitive traffic and relatively stationary nodes.

3.1 Background

In recent research and literature, much attention has been given to the subject of energy-aware routing in wireless ad hoc networks. However, [9] proposes an interesting, closely related approach; energy-efficient optimization of existing routes. It is argued that by optimizing the transmission parameters of the nodes that make up a multihop route, the total energy consumption can be significantly reduced. At the same time, the route lifetime can be extended, thereby reducing the need for energy-demanding route setup procedures. Such a route optimization scheme could in theory be combined with any existing routing protocol [9].

In the model proposed in [9], it is assumed that a multihop route between a source and a sink node has been found by some routing protocol. The energy associated with sending a packet, using this route, is now to be minimized, while certain QoS constraints must be satisfied; the packet must be delivered with a given probability of success within a time limit. Each hop in the route is associated with an independent, Rayleigh-distributed link gain, which is assumed to be constant over the time required to transmit a packet. The system is transmitting at a fixed rate and utilizes an Automatic Repeat Request (ARQ) scheme. For each hop, the ARQ scheme automatically retransmits packets that are received in error. All nodes can adjust their transmission power from zero up to a maximum threshold.

Without any energy awareness, each node would retransmit a packet that is not successfully delivered to the next node until so happens, or a maximum threshold is reached. But if many retransmissions are necessary at one of the early nodes, there will be a large delay, and the probability for a successful delivery within the delay constraint goes to zero [9]. In this case, the packet should be dropped, since it will not be delivered to the destination in time. This best-effort does ensure the best end-to-end delivery rate, but the energy consumption can potentially be unacceptably large in energy-constrained networks.

To remedy this, the proposed scheme calculates an optimal number of allowed retransmissions for each node throughout the route, and the transmission power associated with each (re-)transmission. This ensures that packets are not retransmitted in vain. The optimal route configuration minimizes the total energy consumption, while ensuring that the packet is delivered at the sink within the given time constraint with the required probability. The optimization routine is run in the sink, based on link state information gathered from all hops throughout the route during the route setup process. The optimized transmission parameters are then propagated back to the intermediate nodes by piggybacking the information on a route reply packet. Numerical results show that this energy-greedy, truncated ARQ scheme can give energy savings up to 70%, compared to the best effort scheme, while meeting the same throughput and delay constraints [9].

The model and optimization scheme proposed in [9] are not directly targeted at ultra-low power applications. Circuit energy consumption is ignored, and the transmission power levels considered are considerably higher than what is typical for short-range, ultra-low energy applications [8, 21, 31]. Further, the model is based on a number of idealized assumptions. In the access layer queuing/buffering delay and overhead and latency associated with channel access is ignored. Also, interfering traffic and packet collisions are not discussed.

In [8], energy-optimization is also discussed, but in a different framework, and with focus on the physical layer. An adaptive modulation scheme for a point-to-point path loss Additive White Gaussian Noise (AWGN) channel is investigated. Circuit energy is included in the total energy budget. This creates a trade-off: Transmitting at a high rate requires more transmission power, but a shorter period of time is required to transmit the packet, thereby saving circuit energy, and vice versa. Also, the transmission power is limited by a peak-power constraint. It is shown that by optimizing the modulation order, up to 80% energy savings can be obtained for a point-to-point transmission, compared to the non-optimized case [8]. This work is followed up in [32] and [20], where optimal transmission scheduling and routing are discussed for some simple network examples.

Here, the work presented in [9] and [8] will be combined, and applied to a randomly generated multihop route in a network of a given size and density. That is, the nodes in a multihop route will be configured so that the total energy consumption associated with an end-to-end packet transmission is minimized, subject to QoS requirements as in [9]. Bit Error Rate (BER) will be used as the throughput quality metric, instead of Packet Error Rate (PER). The ARQ scheme and Rayleigh-fading channel model will be replaced with the AWGN channel and the physical layer models from [8]. The optimization scheme in the latter will be extended to the multihop case, so that all nodes in the route are jointly optimized. The idealized assumptions from [9], regarding competing traffic, channel access and buffer/queuing delay are adopted here.

3.2 System Model

The area in which the network is deployed is modeled as a square, two-dimensional space of a given size. In this region, a given number of nodes are distributed randomly. At some point in time, a random node, the source, generates a fixed-size packet of Q bits. This packet is sent over a multihop route, and must be delivered to its destination, the sink, within a time constraint T_D, and with a minimum end-to-end BER $P_{b,req}$.

Furthermore, assume that a route of length L hops from source to sink has been found. The physical length of each hop is denoted $d_i, i = 1, 2, \ldots, L$. Each hop is modeled by a path-loss AWGN channel, so that for a given hop length d_i, there is a one-to-one relationship with the associated channel gain, denoted g_i. An example route between two random nodes, found by a simple algorithm that minimizes the hop lengths, is shown in fig. 3.2.

The nodes are capable of adaptive modulation and demodulation, and power control. In [8], both coded and uncoded QAM and FSK is used, but only uncoded QAM is followed up in subsequent work by the authors in [20, 32]. Also here, uncoded QAM will be considered, due to the simple relationship between modulation order and transmission time of a fixed-length packet. Including coding would also complicate the circuit power consumption model further, as power consumption required for coding/decoding would have to been taken into consideration.

In the three above mentioned articles, the minimum allowed modulation order, in terms of bits per symbol, is 2 (4-QAM). This limit is based on that for a given BER, the required power per bit is the same for 4-QAM (equivalent with 4-PSK) is the same as for BPSK [18, 32]. But since 4-PSK requires shorter transmission time than BPSK, for the same number of bits, the overall energy consumption is lower for QPSK, as less circuit energy is consumed. So setting 2 bits per symbol as the lowest modulation order makes sense in terms of keeping the energy consumption to a minimum in a short-term perspective.

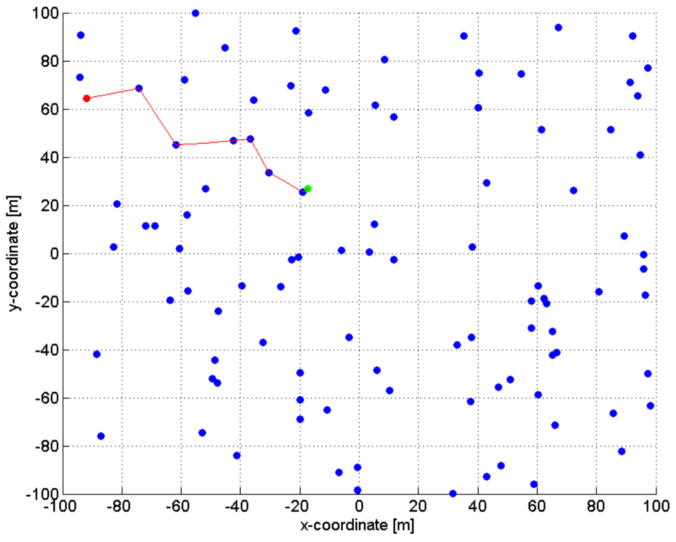


Figure 3.1: Example route of length 7 hops. 200 nodes randomly distributed in 100×100 meters area.

However, since the output power is limited in the system considered here, allowing modulation with less than 2 bits/symbol will extend the nodes' maximum transmission range for a given target BER, since more power per bit is then available. Allowing a larger transmission range means that more routes can be configured to meet the QoS requirements, or that an existing route can be maintained longer, thereby saving the system from trigging a new route setup procedure. This might be more energy-efficient in terms of maximizing a system's overall lifetime. So in the following, 1 bit per symbol will be the minimum modulation order. For $b < 2$, PSK will be used, since it share the same basic structure as MQAM, which will be used for $b \geq 2$. The same hardware model is used here, regardless of which of the two modulation techniques are used.

For PSK and QAM, there exists a simple, one-to-one relationship between the modulation order and the transmission time for a fixed-length packet. Allowing individual adaptation for each hop is therefore equivalent to dynamic time slot allocation throughout the route. The total time spend on the transmission is the sum of the time required to transmit and receive the packet L times and any queuing delay in the nodes. As in [9], propagation delay is ignored. The total energy consumption is the sum of all energies spent in all nodes along the route, both transmission and circuit energy.

The goal is now to configure all nodes throughout the given route optimally, so that the total energy consumption is minimized, while satisfying the given QoS constraints. The configuration is completely described by the vectors $\mathbf{b} = [b_1 b_2 \dots b_L]$ and $\mathbf{P}_{tx} = [P_{tx,1} P_{tx,2} \dots P_{tx,L}]$, that dictates the modulation order and transmission power, respectively, for all nodes. As in [9], it is assumed that the link state information for the entire route is known in the sink, where the optimization routine is run. The routine is run relatively infrequent (i.e. many packets are transmitted using the same configuration). If more frequent updates are required, the proposed scheme is not very well suited. Hence, energy and time required to perform the optimization is ignored for now.

The system model is implemented in Matlab and simulations are run on a number of generated routes to obtain numerical results. It turns out that for a given network size and density, the hop lengths can be well modeled with statistical tools. The remainder of Section 3.2 describes the system model more in-depth, while 3.3 explains the techniques used for optimizing the route.

3.2.1 Channel

The route consists of L hops, each of physical length $d_i, i = 1, 2, \dots, L$. The links are modeled using a path-loss AWGN channel model, which is also appropriate within one block of a slowly block-fading channel. That is, each hop is associated with a link gain $g_i, i = 1, 2, \dots, L$, through a deterministic function of the physical channel length and other parameters [8]:

$$g_i = \frac{1}{G_1 d_i^{\kappa} M_l} \tag{3.1}$$

Here, d_i is the distance of the i'th hop. G_1 is the gain factor at $d = 1$ m, κ is the path loss exponent and M_l the link margin, which are all assumed to be the same for all hops. A route is completely described by the link gain vector $\mathbf{g} = [g_1 g_2, \dots g_L]$.

When transmitting over the i'th link with transmission power $P_{tx,i}$, the received signal power $P_{rx,i}$ is found as [8, (8 - modified)]:

$$P_{rx,i} = P_{tx,i} g_i \tag{3.2}$$

and the corresponding SNIR [8, (modified)]:

$$\gamma_i = \frac{P_{tx,i} g_i}{2B\sigma^2 N_f + I} = \frac{P_{tx,i} g_i}{P_{NI}} \tag{3.3}$$

where B is the system bandwidth, σ^2 the power spectral density of the AWGN on the channel, and N_f the receiver noise figure. I is interference power from the other nodes or other sources. Note that competing traffic on the route is assumed to be eliminated by a suitable orthogonal MAC protocol, as in [8] and [9]. However, in a large-scale network, it is likely that there is simultaneous communication in a different part of the network. If the number of interference sources is large, the total interference power can be modeled as an extra term of white noise. If there is no interference, the term I can simply be set to zero. However, the term SNIR is used throughout this thesis, also if the interference power is set to zero.

If a transmission over a single hop is to satisfy a minimum BER requirement, the received SNIR must be larger than a given threshold. On an AWGN channel, the BER of MQAM modulation is bounded by [8]:

$$P_b \leq \frac{4}{b}\left(1-\frac{1}{\sqrt{2^b}}\right) Q\left(\sqrt{\frac{3}{2^b-1}\gamma}\right) \tag{3.4}$$

where $b = \log_2(M) \geq 2$ denotes bits per symbol, and the Q-function is defined as [18, (2.1-97)]:

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-t^2/2} dt, \ x \geq 0 \tag{3.5}$$

which is bounded by [18, (5.2-26)]:

$$Q(x) < e^{-x^2/2} \tag{3.6}$$

By applying this bound, (3.4) can be expressed as [8]:

$$P_b \leq \frac{4}{b}\left(1-\frac{1}{\sqrt{2^b}}\right) e^{-\frac{3\gamma}{2(2^b-1)}} \tag{3.7}$$

From (3.7), a lower limit for the received SNIR γ can be found by replacing the inequality with equality and rearranging the equation:

$$\gamma_{lim,b} = \gamma(b) \approx \frac{2}{3}\left(2^b-1\right) \ln\left(4\frac{1-1/\sqrt{2^b}}{P_b b}\right) \tag{3.8}$$

The required transmission power for transmitting over a channel with gain g_i and noise- and interference power P_{NI} using b_i bits per symbol and meeting a BER requirement P_b can now be found by combining (3.3) and (3.8):

$$P_{tx,i} = \frac{P_{NI}}{g_i}\left(\frac{2}{3}\right)\left(2^{b_i}-1\right) \ln\left(4\frac{1-1/\sqrt{2^{b_i}}}{P_b b_i}\right) \tag{3.9}$$

For $b < 2$, the bound (3.4) is no longer valid. A similar approximation for MPSK is given as [18, (5.2-61,62)]:

$$P_b \approx \frac{2}{b} Q\left(\sqrt{2\gamma}\sin\pi/2^b\right) \tag{3.10}$$

By applying the same bound for the Q-function and following the same procedure, the required SNIR to meet a target P_b for PSK can be expressed as

$$\gamma_{lim,b} \approx \ln\left(\frac{2}{P_b b}\right)\frac{1}{\sin^2 \pi/2^b}, \tag{3.11}$$

and the required transmission power P_{tx} to satisfy this SNIR limit becomes:

$$P_{tx,i} = \frac{P_{NI}}{G_i}\ln\left(\frac{2}{P_b b}\right)\frac{1}{\sin^2 \pi/2^b}. \tag{3.12}$$

In the following PSK is used for $b < 2$ and QAM for $b \geq 2$, since the SNIR requirement for PSK increases much faster with the modulation order than for QAM, as shown in Figure 3.2.

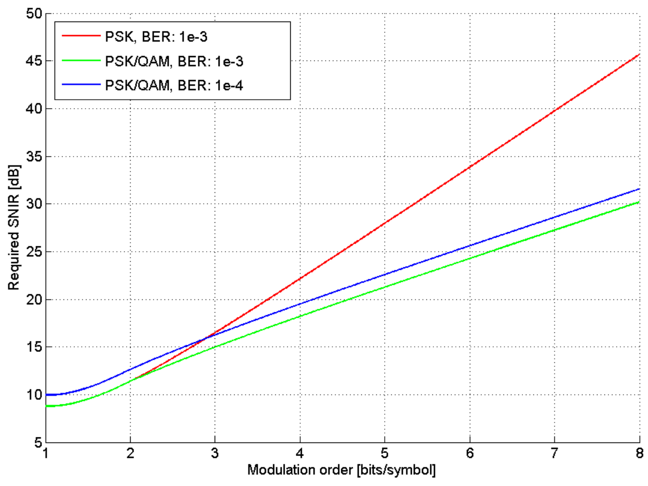


Figure 3.2: SNIR requirements for PSK and QAM over a AWGN channel for different target BERs. For the two bottom lines, denoted PSK/QAM, PSK is used for modulation order $b < 2$ and QAM for $b \geq 2$.

Combining (3.9) and (3.12) yield the expression for the required transmission power using modulation order b_i over a link with a given gain g_i:

$$P_{tx,i} = \begin{cases} \frac{P_{NI}}{g_i}\left(\frac{2}{3}\right)\left(2^{b_i}-1\right)\ln\left(4\frac{1-1/\sqrt{2^{b_i}}}{P_b b_i}\right)(1+\beta) & , b \geq 2 \\ \frac{P_{NI}}{G_i}\ln\left(\frac{2}{P_b b_i}\right)\frac{1}{\sin^2 \pi/2^{b_i}}(1+\beta) & , b < 2 \end{cases} \tag{3.13}$$

This is a key result that will later be used in the route optimization procedure. In the last term, $\beta \geq 0$ is a constant that can be adjusted as a 'safety margin'; i.e. to ensure that the received SNIR is above the minimum threshold.

3.2.2 Power Consumption

The hardware's energy consumption characteristics are highly dependent on the design choices made. Truly power-efficient systems require tailored hardware design, so that no extra energy is wasted in performing better than what is required [5, 22]. An in-depth discussion about the details of hardware design, however, is well beyond the scope of this thesis. Thus, a slightly modified version of the model used in [8] will be adopted here. The model includes only power consumption in the RF circuitry, ignoring power used on baseband signal processing, which is a reasonable assumption for systems with fairly low baseband symbol rate [22].

Further, each node is at any time in one of three different states: *Active* (transmitting or receiving), *sleeping* (not communicating) or *transient* (powering up from sleep to active). The circuit power consumption is fixed, but different for each of the three state. When an active node is transmitting, the total power consumption is found as the sum of transmission and circuit power consumption. The nodes are peak-power limited, i.e. the instantaneous power that can be drained from the batteries is limited by the constraint P_{max}.

In active mode, the power consumption is denoted P_{ctx} and P_{crx}, depending on if the node is transmitting or receiving, respectively. In sleep mode, it is assumed that the power consumption can be ignored, as most of the circuitry is powered down. When powering up a node, there is a non-negligible delay. This is mainly caused by the time required for the Phase Locked Loop (PLL), which is required for coherent PSK/QAM modulation, to settle down. By carefully choosing the order in which the different elements of the circuitry are powered up, the power consumption in transient mode, denoted P_{tr} can be kept as low as possible [8, 22], but there is still a (small) net energy loss every time a node is powered up. Accumulated over time, this term can significantly degrade the overall network lifetime in ultra-low power applications, if the sleep schedule is not properly planned [22].

When calculating the total energy consumption associated with a point-to-point transmission over a given link, the power consumptions of the transmitting and receiving node are added together and denoted $P_c = P_{ctx} + P_{crx}$, which is constant (independent on modulation order). For simplicity, it is assumed that all nodes are initially in the sleep state. Hence, there will be a small energy penalty for powering up the receiving node. The total energy consumption for a packet transmission over the i'th hop can now be expressed as:

$$E_i = T_{on,i}\left(\left(1+\alpha\right)P_{tx,i} + P_c\right) + T_{tr}P_{tr} \tag{3.14}$$

where $T_{on,i}$ is the time required for the node to transmit the packet, i.e. the active period. Assuming square pulses and system bandwidth B, the time required to transmit a packet of length Q bits, using b_i bits per symbol is given by:

$$T_{on,i} = T_{on}(b_i) = \frac{Q}{Bb_i} \tag{3.15}$$

Further, $P_{tx,i}$ is the output transmission power, as referred to in (3.13). T_{tr} is the power-up transient time for the receiving node. α is a measure on the Power Amplifier (PA)'s power efficiency, modeled as a function of the signal's Peak-to-Average Ratio (PAR). The total power consumption when transmitting with output power $P_{tx,i}$ is then given as $(1+\alpha)P_{tx,i}$. The PA's efficiency is here modeled as independent of the output power, which might not be entirely accurate for short-range applications such as this [42]. However, the same basic model is used in several publications by the authors of [8] (e.g. [6, 20, 32]), and will also be adopted here. In the mentioned papers, α is modeled as:

$$\alpha(b) = \frac{\xi(b)}{\eta} - 1 \tag{3.16}$$

where η is the PA's drain efficiency and ξ is the modulation scheme's PAR, which for for this model is given as [8, (modified)]:

$$\xi(b) = \begin{cases} 3\frac{\sqrt{2^b}-1}{\sqrt{2^b}+1}, & b \geq 2 \text{ (QAM)} \\ 1 & b < 2 \text{ (PSK)} \end{cases} \tag{3.17}$$

The peak power constraint P_{max} limits the total power consumption in each node, since $(1+\alpha)P_{tx,i} + P_{ctx} \leq P_{max}$ is required. It is assumed that the peak-power constraint is always met for a receiving node, i.e. $P_{crx} < P_{max}$. For a given modulation order, the output transmission power is then limited by:

$$P_{tx,max}(b) = \frac{P_{max} - P_{ctx}}{1+\alpha} \tag{3.18}$$

The final equation (3.18) states that the available output transmission power is a function of the modulation order (the argument b is omitted on the right handside). By combining (3.1), (3.13) and (3.18), the maximum transmission range for a given modulation order and target BER can be found.

3.2.3 QoS Requirements

The QoS requirements are given per end-to-end packet transmission, in terms of maximum allowed delay T_D and BER $P_{b,req}$. The total transmission time is calculated as the sum of the transmission and transition times for all nodes. In addition, there might be a queuing delay at each node, denoted $T_{q,i}, i = 1, 2, \ldots L$. If no other traffic exists on the route, and all processing and buffering delays associated with the forwarding of the packet are ignored, $T_{q,i} = 0,\ \forall i$. This assumption is made in [9], and will be adopted here. Although this assumption is not possible to fulfill in an implementable system, it does provide an theoretical upper performance bound. For an end-to-end delay constraint T_D, a valid configuration must satisfy:

$$T_{tot} = \sum_{i=1}^{L} (T_{on,i} + T_{q,i} + T_{tr}) \leq T_D \tag{3.19}$$

where $T_{on,i}$ is defined by (3.15).

The probability for a successful end-to-end transmission over L hops can be found as the product of the probability for successful transmission for each hop, assuming bit error events are independent between the hops, i.e. a decode-and-forward scheme is used. The following product must be satisfied:

$$(1 - P_{b,tot}) = \prod_{i=1}^{L} (1 - P_{b,i}) \geq (1 - P_{b,req}) \tag{3.20}$$

For simplicity, the target BER is required to be equal across all hops. This value is readily found from (3.20) by replacing the inequality with equality and assuming that bit error events are independent from hop to hop:

$$P_{b,i} \equiv P_b = 1 - \sqrt[L]{1 - P_{b,req}} \tag{3.21}$$

The index i is omitted, and P_b is used as the minimum required BER for all hop throughout the route. The SNIR required to meet P_b can now be found from (3.8) and (3.11), and the required transmission power to meet this target for a given link gain from from (3.18).

3.2.4 Network Estimate

To this point, it has simply been assumed that a feasible path from source to sink has been found, without discussing how. In order to run simulations, a large number of routes must be generated. Implementing some energy-aware routing protocol to provide a route in an actual network is beyond the scope of this thesis. Instead, a statistical approach will be taken. In particular, the hop lengths are modeled as random variables of some distribution, which is found in a rather qualitative manner.

The area in which the nodes are deployed is modeled as a square, two-dimensional space of a certain size. A given number of nodes are distributed randomly in this area. A proper routing protocol would now have found a feasible path between a given source/sink pair. Which route is found is, of course, highly dependent on the protocol and the routing metrics used. But as shown, the nodes have limited transmission range. Moreover, they also have a finite number of reachable neighbors. Regardless of the routing protocol details, it is reasonable to assume that the next hop for any node will be one of the y closest nodes which are closer to the sink than the node itself. The exact value of y will depend on node density, network size and of course, the routing metrics. But in a dense network, it is reasonable to assume that y is larger than in a sparse. Here, the value $y = 3$ is arbitrary chosen for the particular network size and density used for the simulations.

Now, for a node n in a randomly generated network, the mean of the distances to the y closest nodes, which are closer to the sink than the node n itself, is calculated and denoted $d^y_{avg,n}$. In order to find a statistical average for this metric, a fixed sink is chosen randomly, and $d^y_{avg,n}$ is averaged over all nodes in the network. Averaging this over a large number of generated networks yields an estimate of the distribution of d^y_{avg}. A set of L random variables of this distribution is now used to model the hop lengths of a multihop route of length L.

Figure 3.3 shows the normalized histogram of $d^3_{avg,n}$ $\forall$ n for 500 network realizations of a 100×100 m network with 200 nodes. A Rayleigh distribution with parameter set equal to the peak of the histogram, which in this case is 11.5 m, is shown in the same plot. For these parameters, d^3_{avg} is well approximated by this Rayleigh distribution. I.e. the hop lengths are modeled as Rayleigh-distributed random variables.

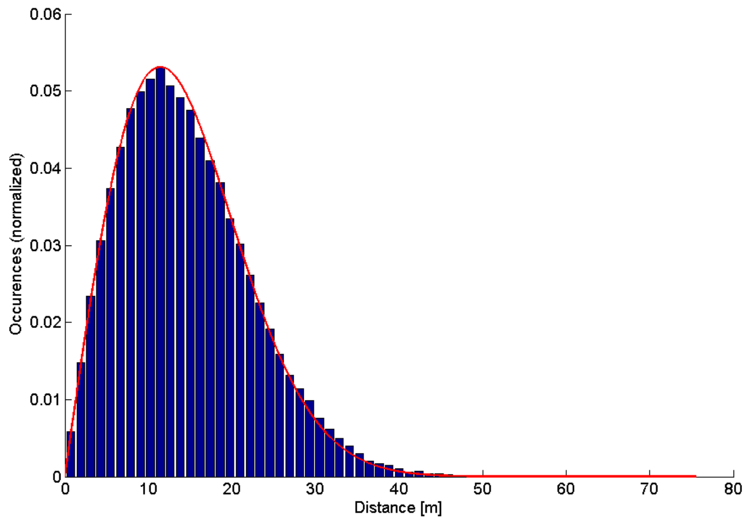


Figure 3.3: Normalized histogram of average hop lengths in randomly generated network, shown with Rayleigh PDF with parameter adjusted to peak of the histogram.

The similarity of the distribution of the average hop length and the Rayleigh distribution was found using a heuristic approach, and is valid only for the given numerical values. The general distribution of the average hop length is unknown. For other values, it seems like a χ^2-distribution fits the histogram better. The shape of the histogram depends heavily on how many of the closest nodes are used when finding the average, as well as the node density.

3.3 Route Optimization

With this framework, the goal is now to find the optimal configuration for a given route that minimizes the total energy consumption per packet transmission, while ensuring that the QoS requirements are satisfied. The total end-to-end delay must not exceed the delay constraint and the received SNIR in all hops must be above the minimum threshold for the target P_b. Finally, the total power consumption is peak power limited. This problem can be formulated as:

$$\begin{array}{ll} \text{Minimize} & E_{tot} = \sum_{i=1}^{L} \left(\frac{Q}{Bb_i} \left(\left(1+\alpha\right) P_{tx,i} + P_c \right) + T_{tr} P_{tr} \right) \\ \\ \text{Subject to} & \sum_{i=1}^{L} \left(\frac{Q}{Bb_i} + T_{q,i} + T_{tr} \right) \leq T_D \\ & \frac{P_{tx,i} g_i}{P_{NI}} \geq \gamma_{lim,b_i}, \quad \forall i \\ & P_{tx,i} \leq \frac{P_{max} - P_{ctx}}{1+\alpha}, \quad \forall i \\ & b_i \geq b_{min}, \quad \forall i \end{array} \tag{3.22}$$

The first line expresses the total energy consumption for transmitting a packet over the route $\mathbf{g}$ using the configuration described by $\mathbf{b}$ and $\mathbf{P}_{tx}$. The second line is the delay constraint, the third is the BER constraint (per hop), the fourth is the peak-power constraint, and the fifth simply states the minimum modulation order. In this problem, there are two vector variables to be optimized, $\mathbf{b}$ and $\mathbf{P}_{tx}$. But it can be simplified to a problem of one variable by observing that the required transmission power for each hop $P_{tx,i}$ can be expressed as a function of the modulation order b_i by applying (3.13). By using this and the transmission power constraint, defined by (3.18), a upper bound for $\mathbf{b}$ can be found as:

$$\mathbf{b}_{max} : b_{max,i} = \arg\max_{b} \left\{ \gamma_{max,i,b} \geq \gamma_{lim,b} \right\}, \ i = 1, 2, \ldots, L \ , \tag{3.23}$$

where $\gamma_{lim,b}$ is the minimum received SNIR to meet P_b using for modulation order b, as defined by (3.8) and (3.11). $\gamma_{max,i,b}$ is the maximum received SNIR for the i'th hop, using b bits per symbol, which is found by combining (3.3) and (3.18):

$$\gamma_{max,i,b} = \frac{P_{tx,max}\left(b\right) g_i}{P_{NI}} \tag{3.24}$$

This $\mathbf{b}_{max}$ is defines the maximum allowable b_i for each hop in the given route. It is readily shown from (3.13) and (3.16), respectively, that both P_{tx} and α are strictly increasing functions of b if all other parameters are fixed. Hence, decreasing the modulation order will only increase the maximum available transmission power. At the same time, the required SNIR threshold decreases with the modulation order, as shown in Figure 3.2. This implies that any route configuration $\mathbf{b}$, such that $b_i \leq b_{max,i}, i = 1, 2, \ldots, L$ will also meet the SNIR requirement for the given P_b, and therefore satisfies the third and fourth lines of (3.22).

Now, by expressing P_{tx} as a function of b and applying the upper bound $\mathbf{b}_{max}$ on the solution vector $\mathbf{b}$, (3.22) is simplified to:

$$\begin{array}{ll} \text{Minimize} & E_{tot} = \sum_{i=1}^{L} \left(\frac{Q}{Bb_i} \left((1+\alpha) P_{tx,i} + P_c \right) + 2T_{tr}P_{tr} \right) \\ \text{Subject to} & \sum_{i=1}^{L} \left(\frac{Q}{Bb_i} + T_{q,i} + T_{tr} \right) \leq T_D \\ & b_{min} \leq b_i \leq b_{max,i}, \quad \forall i \end{array} \tag{3.25}$$

This is an optimization problem of a single, L-dimensional vector variable, that can be solved using different techniques. Finding the optimal $\mathbf{b}$ yields the minimum possible energy consumption for the given route and input parameters. This problem can be solved for two cases; continuous and discrete rate adaptation. That is, allowing the modulation order b to take on any positive value larger than 1, or restrict it to integer values.

Continuous adaptation is expected to give the optimal solution. However, implementing modulation for a non-integer modulation order is quite complex [2], and not feasible in a practical system. Thus, both continuous and discrete rate adaptation is implemented, and the results compared. For reference, the performance of the adaptive systems will be compared to fixed-rate systems, with and without power control.

3.3.1 Continuous Case

When allowing continuous rate adaptation, the optimization problem (3.25) can be solved using methods found in the Matlab Optimization Toolbox. First, (3.23) is applied to find the maximum allowed modulation order vector $\mathbf{b}_{max}$ for all hops throughout the given route, which is described by the link gain vector $\mathbf{g}$. This configuration yields the minimum end-to-end transmission time. If it does not meet the delay constraint, no valid configuration exists for the given route.

If a valid configuration does exist, the function `fmincon` is used to find the optimal modulation order vector $\mathbf{b}_{opt}$, that lies within the feasible region, i.e. $b_{min} \leq b_i \leq b_{max,i}, \ \forall\ i$. Finally, the corresponding transmission power vector $\mathbf{P}_{tx}$ is readily found, using (3.13).

For the special case $L = 1$, the problem can be further simplified. The end-to-end delay (second line in (3.25)) can be manipulated to yield the minimum boundary for the modulation order:

$$b_{min} = \min \left\{ 1, \frac{Q}{B\left(T_D - T_{tr}\right)} \right\}, \tag{3.26}$$

where the term for queuing delay is here ignored. Now there is only one constraint left, the allowed range of modulation orders. Figure 3.4 illustrates the steps of the optimization process, and the given constraint. Except for a lower minimum allowed modulation order, this simplified problem is now very similar to what is described in [8], which can be shown to be a convex function over the feasible region for the modulation order [6]. As stated, the net energy consumption when transmitting a packet over a single hop using a modulation order $b < 2$ will be larger than when transmitting at $b = 2$, since the power requirement per bit is the same while a longer transmission time is required. Thus, this problem is also convex, and efficient methods can then be applied to find the optimal solution [8]. However, if the general L-dimensional problem can also be shown to be convex, remains an open question.

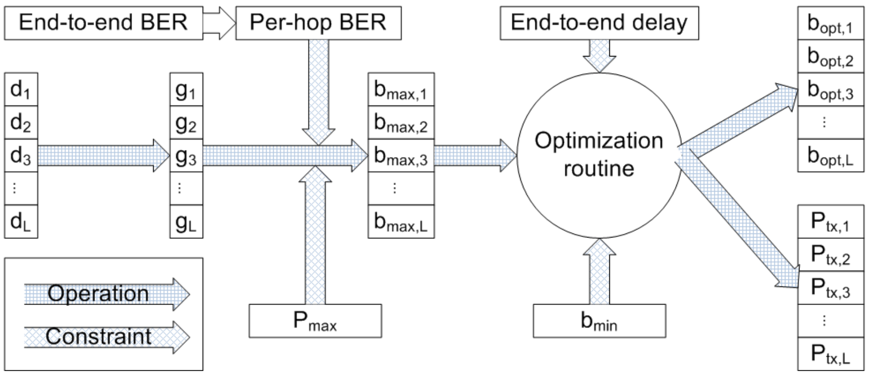

End-to-end BER
Per-hop BER
End-to-end delay
d_1 d_2 d_3 ⋮ d_L
g_1 g_2 g_3 ⋮ g_L
$b_{max,1}$ $b_{max,2}$ $b_{max,3}$ ⋮ $b_{max,L}$
Optimization routine
$b_{opt,1}$ $b_{opt,2}$ $b_{opt,3}$ ⋮ $b_{opt,L}$
$P_{tx,1}$ $P_{tx,2}$ $P_{tx,3}$ ⋮ $P_{tx,L}$
Operation
Constraint
P_{max}
b_{min}

Figure 3.4: Steps in route optimization routine

Also note that if the delay constraint is loose, i.e. $\sum_{i=1}^{L} \left(\frac{Q}{Bb_i} + T_{q,i} + T_{tr} \right) \leq T_D$ is met for $b_i = 1$, $\forall i$, the second line of (3.25) becomes redundant, as the delay constraint is satisfied by all valid configurations. Transmitting more bits per symbol at any hop will only decrease the end-to-end delay. In this case, there is no need for end-to-end optimization, and configuring each node in isolation will give the optimal solution. However, for tighter delay constraints, this will not be the case, and the nodes needs to be jointly optimized to guarantee that the global, optimal solution is found.

3.3.2 Discrete Case

Restricting the modulation order to integer values requires a different approach, since the methods found in the Matlab Optimization Toolbox are only applicable for continuous variables. An iterative energy-greedy algorithm, presented in pseudo-code in Figure 3.5, is proposed here. Simulations show that this algorithm finds a near-optimal solution, while it is far less computationally demanding than the method used in the continuous case.

The initial step is, as in the continuous case, to find the maximum modulation order for all hops throughout the route. This is done simply by following the same procedure as in the continuous case, and then round all elements down to the nearest integer value.

Assuming that the discrete $\mathbf{b}_{max}$ is a valid configuration, the corresponding energy consumption is calculated and the main part of the algorithm is run. Again, a key point is that as long as all elements of $\mathbf{b}$ is within the feasible region, i.e. $b_{min} \leq b_i \leq b_{max,i}$, the peak-power constraint is met. It is then sufficient to check if the end-to-end delay constraint is met.

In each iteration, the constellation size for each hop is decreased by one bit, relative to the best configuration from last iteration. This is repeated in a round-robin fashion for all hops. If one or more of these alternative configurations give a lower total energy consumption than the previous best configuration, the best is selected and set as the new best configuration. This procedure is run until no better configuration is found, and the optimal configuration is returned.

The function $T(\mathbf{b})$ in line 4, calculates the total time used for the transmission using the configuration $\mathbf{b}$, as in (3.19). $\mathbf{P}_{tx}(\mathbf{b})$ on line 6 calculates the power vector, i.e. the required transmission powers for all hops when transmitting at the rates defined by $\mathbf{b}$. The function implements equation (3.13). Finally, the function $E(\mathbf{b}, \mathbf{P}_{tx})$, first encountered in line 7 calculates the total energy consumption for the given configuration, as described in the first line of (3.25).

This algorithm ensures that the largest possible energy savings are obtained in each iteration. It does not guarantee that the optimal solution is found, but simulations indicate that the results are close the optimal solutions found by allowing continuous rate adaptation.

3.3.3 Fixed Rate

The modulation order can be set to a fixed value for all nodes in a given route. This can be done with or without power control, i.e. adjusting the transmission power so that the target BER is just satisfied (channel inversion). Both these cases are implemented, and serves as a reference to which the results from the adaptive schemes are compared. For a given modulation order b_{fix}, the required transmission power for each hop can be found directly from (3.13). Without power control, each node transmits at maximum power for the given modulation order, found from (3.18). Clearly, no joint optimization of the nodes is required when using fixed modulation order, since the power can just as well be adjusted locally at each node.

```
procedure optimizeRouteConfiguration($b_{min}, \mathbf{b}_{max}, T_D$)
    $\mathbf{b}_{opt} \leftarrow \mathbf{b}_{max}$    ▷ Initial best configuration
    $N \leftarrow \text{length}(\mathbf{b}_{opt})$    ▷ Number of hops in route
    $T_{tmp} \leftarrow T(\mathbf{b}_{opt})$    ▷ Calculate end-to-end delay
    if $T_{tmp} \leq T_D$ then    ▷ Check if time constraint met
        $\mathbf{P}_{tx,opt} \leftarrow \mathbf{P}(\mathbf{b}_{opt})$    ▷ Calculate transmission powers
        $E_{opt} \leftarrow E(\mathbf{b}_{opt}, \mathbf{P}_{tx,opt})$    ▷ Calculate total energy consumption
        $run \leftarrow TRUE$
    else
        $run \leftarrow FALSE$    ▷ No valid route configuration exists
        $\mathbf{b}_{opt} = \mathbf{0}$
        $\mathbf{P}_{tx,opt} = \mathbf{0}$
    end if
    while $run == TRUE$ do
        $\mathbf{b}_{prev} \leftarrow \mathbf{b}_{opt}$    ▷ Save previous best configuration
        for $n \leftarrow 1, N$ do    ▷ For all hops
            $\mathbf{b}_{tmp} \leftarrow \mathbf{b}_{prev}$    ▷ Go back to previous best configuration
            $\mathbf{b}_{tmp}(n) \leftarrow \mathbf{b}_{tmp}(n) - 1$    ▷ Decrease b with 1 for n'th hop
            if $\mathbf{b}_{tmp}(n) \geq b_{min}$ then    ▷ Check $b_{min}$ constraint
                $T_{tmp} \leftarrow T(\mathbf{b}_{tmp})$
                if $T_{tmp} \leq T_D$ then
                    $\mathbf{P}_{tx,tmp} \leftarrow \mathbf{P}(\mathbf{b}_{tmp})$
                    $E_{tmp} \leftarrow E(\mathbf{b}_{tmp}, \mathbf{P}_{tx,tmp})$
                    if $E_{tmp} < E_{opt}$ then    ▷ New configuration is better?
                        $E_{opt} \leftarrow E_{tmp}$
                        $\mathbf{b}_{opt} \leftarrow \mathbf{b}_{tmp}$    ▷ Set new configuration as best
                        $\mathbf{P}_{tx,opt} \leftarrow \mathbf{P}_{tx,tmp}$
                    end if
                end if
            end if
        end for
        if $\mathbf{b}_{prev} == \mathbf{b}_{opt}$ then
            $run \leftarrow FALSE$    ▷ No better configuration found. Terminate
        end if
    end while
    return $\mathbf{b}_{opt}, \mathbf{P}_{tx,opt}$    ▷ Return optimized configuration
end procedure
```

Figure 3.5: Energy-greedy route optimization algorithm for discrete rate adaptation.

3.3.4 Error Scenarios

Setup Error

Of course, not all routes can be configured to satisfy the given QoS requirements, i.e. end-to-end delay and minimum BER requirement. For a given end-to-end BER $P_{b,req}$ and a route length L, the per-hop BER P_b is found from (3.21). Due to the peak-power constraint, there is a maximum transmission range associated with each modulation order, for a given link gain g_i and target P_b. If any of the hop lengths in a route exceeds this maximum transmission range for the smallest allowed modulation order $b_{min} = 1$, the SNIR will fall below the minimum threshold and the BER constraint will be breached. E.g. for $P_{b,req} = 10^{-3}$ and a route of 5 hops, the per-hop BER P_b is approximately 2×10^{-4}. This gives a maximum transmission range of $d_{max} = 47$ meters.

The error probability for this type of setup error can be expressed as $P(error_{range}) = \sum_{i=1}^{L} P(d_i > d_{max}) = L \times P(d_i > d_{max})$, since the hop lengths are modeled as independent and identically distributed random variables. As the node density grows, this type of setup error rate will become arbitrary small. Note that this type of setup error is independent of the delay constraint.

The other type of setup error is caused by breaching the delay constraint. The minimum end-to-end delay for a given route is obtained when using the maximum allowed modulation order at all nodes. If this configuration does not satisfy the delay constraint, no valid route configuration exists. The probability for this can be expressed as $P(error_{delay}) = P(T_{min} > T_D)$, where T_{min} is found from (3.19) using the largest possible modulation order at all hops, which is found from (3.23): $T_{min} = \sum_{i=1}^{L} \left(\frac{Q}{Bb_{max,i}} + T_{tr} \right)$. Queuing delay is ignored. The simulations does not separate between these two types of setup errors. Whenever they occur in the simulations, they are both referred to as a *setup error*.

Transmission Error

The route configuration schemes presented above are based on link state information (except the fixed-rate, fixed-power scheme), which can easily be calculated in the link layer [2]. However the link state estimates might not be entirely accurate, due to node mobility, estimation errors and latency introduced when forwarding the link state estimates to the sink node.

If a link state of a given hop is estimated to have a smaller signal attenuation that its real value, the SNIR will fall below the minimum threshold, and the BER over this link will be larger that expected. In the opposite case, valuable energy is wasted, since the node is configured to use a higher transmission power then what is actually needed. Hence, the performance of a system that is optimized with respect to link state information will be degraded if this information is inaccurate.

In the simulations, when inaccurate link state information is used[1] in the optimization procedure, the received SNIR may fall below the required threshold during transmission at one of the hops. If this happens, the end-to-end BER will fall below its target value $P_{b,req}$. The packet is therefore dropped, and a *transmission error* is registered. To avoid this situation, the transmission power must be increased to allow for some variation in the link state estimates. That is, setting the parameter β in (3.13) to a value larger than zero, according to the uncertainty of the link state estimates. Note that this new, adjusted value of the transmission power P_{tx} is also limited by the peak-power constraint P_{max}, as defined in (3.18). Clearly, if perfect link state information is used, β can be set to zero, and there will be no transmission errors.

[1]Described in Section 4.1

Chapter 4

Simulations and Results

This section briefly describes how the system model in the previous section is implemented in Matlab, and how the simulations are executed. The source code is found in appendix B. Further, key results from the simulations are presented. The results are commented here, while a more thorough discussion and a review of the system model, based on the simulation results, are found in Section 5.

4.1 Simulations

The system is implemented using Matlab. Only standard built-in functions are used, except for the `fmincon` function from the Optimization Toolbox, which is applied when solving the route optimization problem (3.25) in the continuous case. Figure 4.1 illustrates the simulation process.

All simulations are run on a network made up of 200 nodes, distributed randomly in a 100×100 m area. As explained in Section 3.2.4, based on some simple ground assumptions, the hop lengths are approximately Rayleigh-distributed. A route of length L is therefore approximated by simply generating L independent and identically Rayleigh-distributed random variables, describing the hop lengths. The parameter of this distribution is set to 11.5 m. Link gains are then readily calculated by applying (3.1).

To simulate inaccurate link state estimates, the desired amount of uncertainty can also be added to the channel gains of the generated route, according to the following model:

$$\hat{g}_i = g_i \left(1 + R\frac{z}{100}\right) \tag{4.1}$$

Here, $g_i \in \mathbf{g}$ (The vector containing the link gains of all hops in the route), R is a zero-mean, unit variance Gaussian-distributed random variable and z a measurement of the severity of the estimation error, given in %. Assuming perfect link state information is equivalent to setting $z = 0$.

This procedure yields a new link gain vector $\hat{\mathbf{g}} = [\hat{g}_1\hat{g}_2, \ldots \hat{g}_L]$, which is used in the optimization procedure. Transmissions are then simulated, simply by applying (3.3) to the optimized transmission parameters for each hop and checking the resulting SNIR against the threshold found from by (3.8) and (3.11).

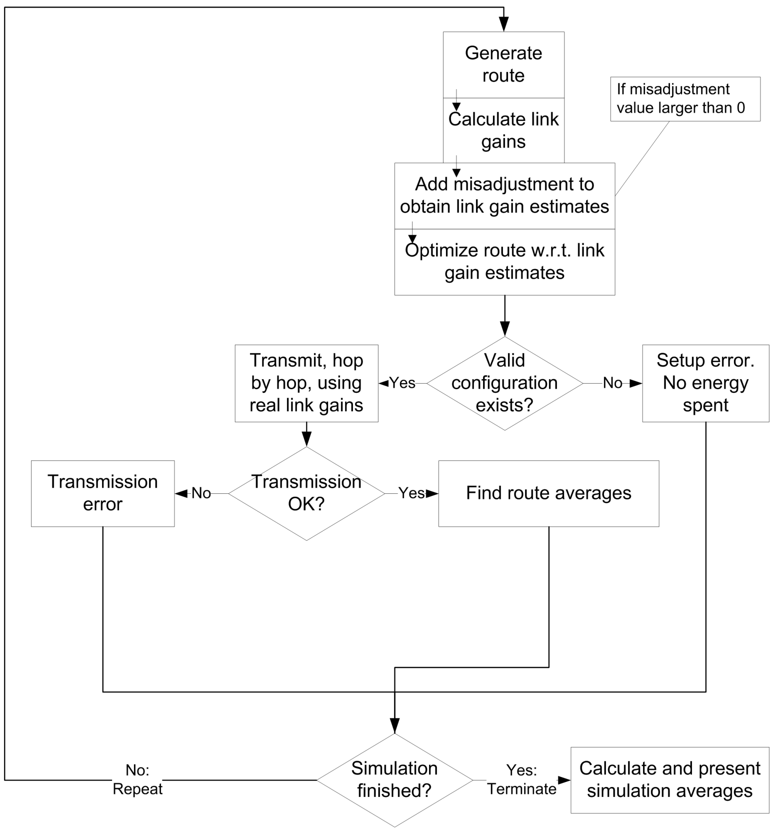

Generate route
Calculate link gains
If misadjustment value larger than 0
Add misadjustment to obtain link gain estimates
Optimize route w.r.t. link gain estimates
Transmit, hop by hop, using real link gains
Yes
Valid configuration exists?
No
Setup error. No energy spent
Transmission error
No
Transmission OK?
Yes
Find route averages
No: Repeat
Simulation finished?
Yes: Terminate
Calculate and present simulation averages

Figure 4.1: Execution of simulations

The optimization routines are run for all four cases: Fixed rate with and without power control, and continuous and discrete rate adaptation. In total, this yields four sets of four route configurations. For all successful configurations (i.e. where setup errors does not occur), the transmission routine is run. If a valid configuration cannot be found, a setup error is registered, and no transmission takes place.

The transmission procedure is done on a per-hop basis. That is, if inaccurate link state estimates are used for the optimization routine and the SNIR falls below the required threshold for one of the hops, a transmission error is registered, and the transmission is terminated for that particular route.

The generation of one route, followed by the four optimization procedures and transmission simulations, form one iteration. After each iteration, the total time and energy consumption and the average modulation order and transmission power is calculated. In routes where a transmission error occurs, the energy spent on up to, and including, the failing hop is added to the total energy consumption. The rest of the transmission is then terminated, as the QoS requirements will not be met. A number of iterations are run, and the final results found as an average over all iterations.

The implemented system is to a large extent adopted from [8]. Also the numerical values of most system parameters are the same as in the original model. Table 4.1 contains parameters used in the simulations. Unless stated otherwise, these parameters are used in all simulations. Details and definitions of the parameters are found in Section 3.

Hardware parameters	*Channel parameters*	*Other parameters*
$P_{ctx} = 98.20$ mW	$G_1 = 30$ dB	$N_f = 10$ dB
$P_{crx} = 112.50$ mW	$\kappa = 3.5$	$B = 10$ kHz
$P_{max} = 250$ mW	$M_l = 40$ dB	$Q = 2$ kB (pkt length)
$P_{tr} = 50$ mW	$\sigma^2 = -174$ dBm/Hz	$d_{avg} = 11.5$ m
$T_{tr} = 5$ μs	$I = 0$ (no interference)	$P_{b,req} = 10^{-3}$
$\eta = 0.35$		

Table 4.1: System parameters.

4.2 Single Hop Results

In order to understand the system model, it is useful to examine how it behaves in the single-hop case. In particular, the trade-off between circuit and transmission energy can easily be illustrated by this simple case. For the path loss model used here, the received power decreases exponentially with the physical channel distance, as defined in (3.1) and (3.2). The available output transmission power also decreases as the modulation order increases, as defined in (3.18), due to the PA characteristics. When combined, a maximum range for each modulation order can be found, for a given target BER. Figure 4.2 shows the required transmission power[1] as a function of modulation order, for hop distances 5, 10, 20, and 40 m. The transmission power is limited upwards by the peak-power constraint. Observe that the maximum allowed modulation order decreases fast as the hop length increases. The maximum range for the lowest modulation order $b = 1$ is approximately 50 meters for a target BER 10^{-3} (for that single hop).

[1]Note that *transmission power* here refers to actual *output* transmission power, i.e. the P_{tx} found in e.g. (3.13) and (3.18). The total power consumption when transmitting at this power is higher, and is dependent on the PA's efficiency, as defined in (3.14) and (3.16). Throughout this section, *transmission power* will refer to the *output* transmission power.

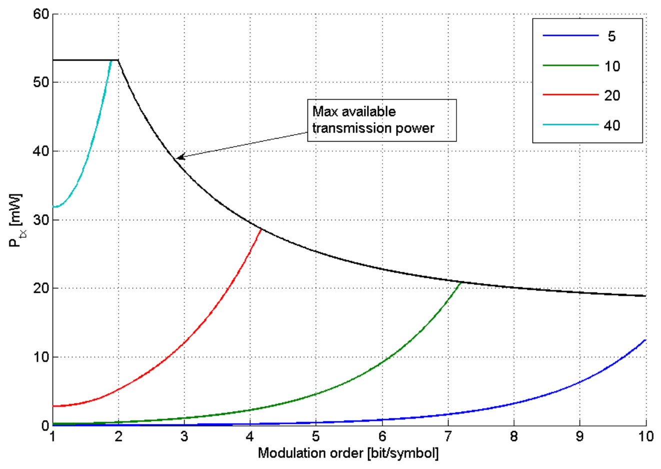


Figure 4.2: Required transmission power P_{tx} as function of modulation order b for discrete values of hop length, limited by maximum available transmission power.

The optimal configuration for each hop is found as the best trade-off between saving circuit energy at the cost of transmission energy, and vice versa. This is illustrated in figure 4.3, where the total energy consumption per bit and required transmission power are plotted against modulation order. A single packet transmission over a hop of length 11.5 m and a target BER of 10^{-3} is considered. The total energy includes energy consumption in both the transmitting and the receiving node. The abrupt fall in the graph indicates that the peak power constraint is breached, i.e. the maximum allowed modulation order for this link is approximately $b_{max} = 6.9$ bits/symbol. The corresponding, discrete value is found by rounding the optimal value down to the nearest integer, $b_{max}^{disc} = \lfloor b_{max} \rfloor = 6$ bits/symbol. The configuration that yields the lowest total energy consumption is approximately $b_{opt} = 5.8$ bits/symbol, or $b_{opt}^{disc} = b_{max}^{disc} = 6$ bits/symbol for the discrete case. If a delay constraint had been given, there would also have been a lower bound on the modulation order, as defined in (3.26).

Figure 4.4 shows the energy consumption for a single packet transmission, in terms of energy per bit, over a single hop as function of modulation order and hop length, for hop lengths 1 - 50 m, modulation order 1 - 10 bits/symbol and target BER 10^{-3}. The dark blue area below the sharp threshold marks the region for which no valid configuration exists, due to the peak-power constraint. As previously stated, the minimum allowed modulation order is 1 bit/symbol. No upper limit on the modulation order is defined, it is only limited by the peak power constraint in transmitting node for the given channel. As seen from the graph, a modulation order of more than 10 bits/symbol is theoretically possible for hop lengths smaller

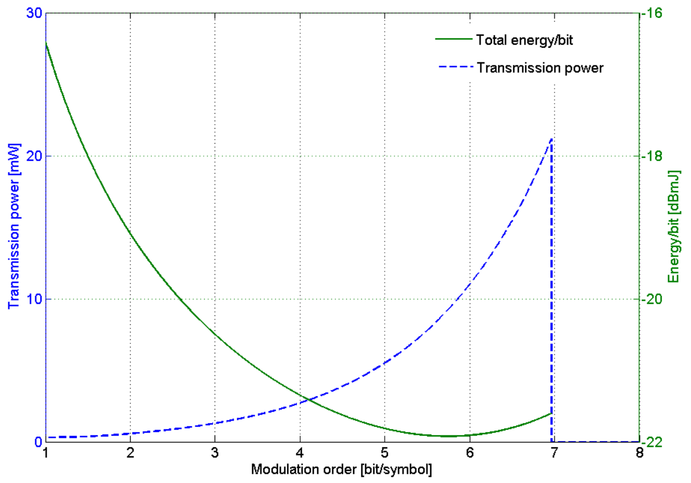


Figure 4.3: Total energy consumption and required transmission power as function of modulation order for a single hop of length 11.5 meters.

than 5 meters. Of course, such large constellation sizes are not feasible in a real wireless system. Nevertheless, the assumption that the constellation size can be arbitrary large is used in all simulations throughout this chapter, unless stated otherwise. This way, the theoretically best performance bound is found. The effect of limiting the maximum modulation order will also be investigated.

In a non-adaptive system, a fixed modulation order must be set, thereby greatly affecting the flexibility and energy efficiency of the system. That is, a low modulation order would be necessary to allow a reasonably long transmission range. But as seen from figure 4.4, this would waste valuable energy for transmission over shorter distances, where a higher modulation order is far more energy-efficient. This is an important point that will be further illustrated by simulations of transmission over multihop routes.

4.3 Multihop Results

Now, the simulation results of end-to-end packet transmissions over a multihop route are presented. The four different optimization schemes are denoted

- FRFP: Fixed rate, fixed power
- FRVP: Fixed rate, variable power (Channel inversion)
- CR: Continuous adaptive rate, variable power
- DR: Discrete adaptive rate, variable power

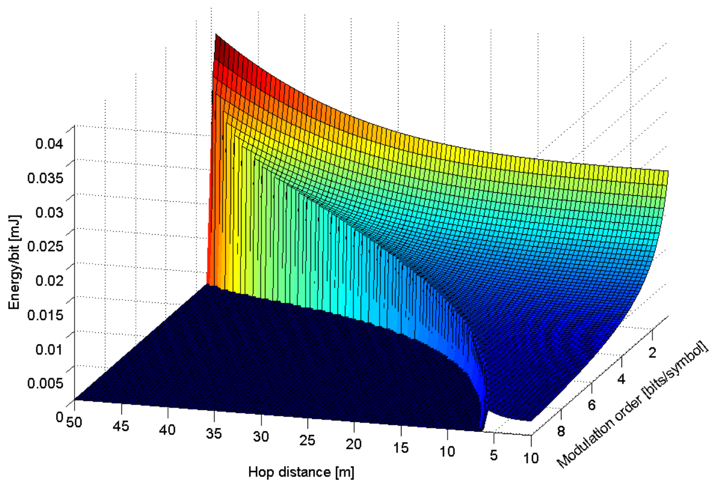


Figure 4.4: Energy consumption per bit for single hop transmission as function of modulation order and hop length.

throughout this section. By using the built-in timer function in Matlab, it is found that the discrete rate optimization routine runs much faster than the continuous version, while the fixed-rate, channel inversion scheme requires negligible time. On average, the discrete optimization routine runs 85-90% faster than its continuous counterpart.

4.3.1 Detailed Route Configuration

Table 4.2 shows the detailed results for transmitting a packet over a 3 hop route, with hop lengths 5, 12, and 35 m. The modulation order, output transmission power and time, energy per bit, and received SNIR are shown for each hop, for all four route configuration schemes. The fixed modulation order is set to 2 bits/symbol, the end-to-end delay constraint is 350 ms, and the end-to-end BER requirement 10^{-3}. Perfect link state information is assumed. With the fixed rate set fairly low, the largest energy savings can be obtained on the short distances, where the adaptive schemes can transmit at a much higher rate. Note that the penalty, in terms of time and energy, when going from continuous to discrete rate adaptation is quite small.

4.3.2 Basic Comparison

Averaging the results over a large number of iterations gives a better picture of the system's performance. The script is run 5000 times, with the route length is set to 5 hops. Fixed rate transmission over a 5-hop route yields a total delay of 512.03 ms, including transition time for each node. The end-to-end delay constraint it therefore set to 515 ms. The end-to-end BER

Hop #	*Hop length [m]*	*Hop gain [dB]*	*Trans. scheme*	*Modulation order*	*Trans. power [mW]*	*Trans. time [ms]*	*Energy/bit [μJ]*	*Rec. SNIR [dB]*
1	5.00	-94.46	FRFP	2.00	53.13	102.41	18.13	43.78
			FRVP	2.00	0.04	102.41	10.54	12.04
			CR	8.90	5.06	23.02	2.81	33.57
			DR	9.00	5.42	22.77	2.81	33.87
2	12.00	-107.77	FRFP	2.00	53.13	102.41	18.13	30.47
			FRVP	2.00	0.76	102.41	10.64	12.04
			CR	5.42	10.02	37.79	5.05	23.23
			DR	5.00	7.47	40.97	5.11	21.95
3	35.00	-124.04	FRFP	2.00	53.13	102.41	18.13	14.20
			FRVP	2.00	32.34	102.41	15.15	12.04
			CR	2.18	37.78	94.16	15.05	12.72
			DR	2.00	32.34	102.41	15.15	12.04

Table 4.2: Simulation results, single run. Example route configuration. Route length 3 hops, end-to-end target BER 10^{-3}, delay constraint 350 ms.

requirement is kept the same and perfect link state information is still assumed. The results presented in Table 4.3 show average energy consumption per bit, the nodes' average transmission power and modulation, average and maximum end-to-end transmission time and the setup error rate.

Trans. scheme	*Avg. energy/bit [dBmJ]*	*Avg. trans. power [mW]*	*Avg. mod. order*	*Trans. time [ms]*		*Setup error rate [%]*
				avg	max	
FRFP	-10.43	53.13	2	512.03	512.03	1.24
FRVP	-12.58	3.60	2	512.03	512.03	1.24
CR	-15.08	13.27	5.45	221.04	385.17	0.12
DR	-15.02	13.07	5.43	226.54	457.41	0.12
CR	-14.82	11.43	4.35	247.50	410.14	0.13
DR	-14.77	11.23	4.34	252.40	494.96	0.13

Table 4.3: Simulation results, averaged over 5000 runs. 5 hops, end-to-end BER 10^{-3}, delay constraint 515 ms. b_{max} limited to 5 bits/symbol in bottom two lines.

First of all, the simulation results show that the adaptive schemes can give huge energy savings. As one might expect, transmitting at fixed rate with maximum power yields the largest energy consumption. By allowing power control (channel inversion), 39% energy savings per bit can be obtained. The continuous rate adaptation scheme yields 44% and 66% energy savings over fixed rate with and without power control, respectively. Restricting the modulation order to integer values causes the energy consumption to rise with 1.4%. The setup error rate for the adaptive schemes is also smaller, since they allow a longer maximum hop length without falling below the SNIR threshold.

The bottom two lines of Table 4.3 show the results when the maximum modulation order b_{max} is restricted to 5 bits/symbol. As seen from figure 4.4, hops shorter than approximately 13 meters allows modulation orders larger than this limit. Hence, for all routes containing hops longer than 13 meters, there is potentially a performance degradation. Here, the energy consumption increases with 6%, compared to the unconstrained case, for both continuous and discrete adaptation. For all other simulations run here, there is no upper limit on the constellation size, other than what is imposed by the peak-power constraint.

4.3.3 Tight Delay Constraint

From Table 4.3, note that the actual end-to-end delay when using adaptive modulation is on average much smaller than the delay constraint, while less energy is consumed. The increased flexibility of the adaptive schemes can be further illustrated by running simulations with tighter delay constraints and observe the resulting setup error rates.

Table 4.4 shows the results of a number of simulations, where the delay constraint is gradually decreased, while all other parameters are kept the same. The fixed-rate modulation order is increased in integer steps when required to satisfy the delay constraints. The setup error rate and transmission time will be the same for fixed rate transmission with or without channel inversion, so FR in the table is valid for both FRFP and FRVP.

Delay constraint [ms]	Trans. scheme	Avg. mod. order	Trans. time [ms]		Setup error rate [%]
			avg	max	
400	FR	3	341.36	341.36	23.08
	CR	5.45	221.57	392.41	0.14
	DR	5.42	226.68	399.38	0.38
350	FR	3	341.36	341.36	22.72
	CR	5.47	220.54	350	0.12
	DR	5.46	223.96	349.89	1.38
300	FR	4	256.03	256.03	62.00
	CR	5.46	219.95	300.00	0.50
	DR	5.50	220.37	299.26	4.72
250	FR	5	204.83	204.83	87.80
	CR	5.59	213.21	250.00	5.34
	DR	5.75	208.12	249.84	19.78

Table 4.4: Simulation results, averaged over 5000 runs. 5 hops, end-to-end BER 10^{-3}, various delay constraints.

The setup error rate increases dramatically for the fixed rate schemes as the delay constraint is tightened. Using continuous rate adaptation, the setup error rate remains reasonably low much longer. As seen from Tables 4.3 and 4.4, the setup error rate for discrete rate adaptation at a delay constraint of 350 ms is roughly the same as for the fixed rate/channel inversion scheme at 515 ms. The energy consumption for the two cases are 31.1 μJ and 55.2 μJ, respectively. Thus, discrete rate adaptation allows an almost 40% tigher delay constraint than the fixed rate/channel inversion scheme, while consuming approximately 40% less energy, on average.

Energy consumption is not included in the results in Table 4.4. The setup error rates are very high for some of the cases with a tight delay constraint. As illustrated in figure 4.1, no energy is spent if a route cannot be successfully configured. Clearly, the failing routes contains more long hops than routes that can be configured, and will therefore be more energy-demanding. Thus, when the setup error rates are high, only the less energy-demanding routes are actually used, leading to statistically incorrect results. In fact, the average energy consumption is lower for some of the cases with tight delay constraints and a large setup error rate than those with a looser delay constraint and a smaller setup error rate. Only when the setup error rates are roughly the same does it make any sense to compare the average energy consumption.

4.3.4 Inaccurate Link State Information

All results so far are based on the assumption that perfect link state information is available in the sink. When there is uncertainty in the link state estimation, the received SNIR may fall below the threshold that is required in order to satisfy the QoS requirements. To counteract the errors caused by the uncertainty in the link state information, the transmission power must be increased accordingly.

Table 4.5 shows the transmission error rates for all four transmission schemes for various degrees of estimation error severity and extra transmission power. All other parameters are the same as for the initial simulation run (Section 4.3.2). As expected, the system is vulnerable to link state estimation errors. Due to the tight adaptation, the transmission error rates becomes very large, even for small estimation errors. The transmission power must be increased with a factor roughly three times larger than the link state estimation error in order to keep the setup error rate below 1%. Obviously, when transmitting at maximum power, the transmission error rate is much lower. In fact, no transmission errors occurred at the fixed rate/maximum transmission power scheme during these simulations.

In Table 4.6, the energy consumption per bit is given for the fixed rate/channel inversion and the two adaptive rate schemes, for the levels of misadjustment and extra transmission power that yields a transmission error rate that is roughly equal to 1 % (from Tables 4.3 and 4.5. Of course, increasing the transmission power will also increase the total energy consumption. But the results show that the relative increase in energy consumption is larger for the adaptive schemes than for the fixed rate scheme.

Misadjustment	Extra power	FRFP	FRVP	CR	DR
1	2	0	10.84	11.00	11.00
	3	0	0.38	0.44	0.44
3	5	0	21.14	21.30	21.30
	7	0	4.78	4.88	4.88
	10	0	0.14	0.14	0.14
5	10	0	10.90	11.06	11.06
	15	0	0.80	0.80	0.80

Table 4.5: Simulation results, averaged over 5000 runs, 5 hops, end-to-end BER 10^{-3}, delay constraint 515 ms. Transmission error rates, given in %, for transmission with various degrees of errors in link state information.

Using 15% extra transmission power causes the energy conumption per bit to rise with approximately 10% for both the adaptive schemes, but only with 1% for the fixed rate/channel scheme. This difference can be explained by observing (from Table 4.3) that the average transmission power is much larger for the adaptive schemes than for the fixed rate scheme, due to the (on average) higher transmission rate.

Misadjustment	Extra power	FRVP	CR	DR
[%]		[dBmJ]		
0	0	-12.58	-15.09	-15.03
1	3	-12.57	-15.04	-14.97
3	10	-12.56	-14.97	-14.91
5	15	-12.55	-14.67	-14.62

Table 4.6: Simulation results, averaged over 5000 runs, 5 hops, end-to-end BER 10^{-3}, delay constraint 515 ms. Energy consumption per bit when increasing transmission power to mitigate link state information errors.

4.3.5 Ignoring Circuit Energy

Now, the effect of including the circuitry's contribution to the overall energy consumption, and how this affects the route configuration, is investigated. The circuit power consumption is reduced by 50% and then ignored alltogether, compared to the initial values given in Table 4.1. The peak-power constraint P_{max} is reduced accordingly. That is, $P_{max}^{new} = P_{max} - aP_{ctx}$, where $a = 1$ when circuit power is ignored alltogether, and $a = 0.5$ when it is reduced by 50%. Hence, the available transmission power remains the same as before.

Table 4.7 shows the average modulation order b_{avg} and the average end-to-end delay for both cases. The delay constraints that give a reasonably low setup error rate from Table 4.4 are also used here, to obtain statistically valid results. Perfect link state information is used. It is readily seen that as the transmission power becomes more dominant, lower modulation orders are selected, and that the end-to-end delay approaches the delay constraint. In other words, when only the transmission power is considered, the optimal configuration is to use the lowest modulation order possible. Minimizing the nodes' active time, where circuit energy is consumed, becomes increasingly important as the circuit power consumption grows larger and thus, higher modulation orders are used. This behavior is intuitively satisfying.

Delay	CR				DR			
Constraint	1/2		0		1/2		0	
[ms]	b_{avg}	T_{avg}	b_{avg}	T_{avg}	b_{avg}	T_{avg}	b_{avg}	T_{avg}
515	4.76	259.11	2.23	514.75	4.75	265.02	2.01	512.03
400	4.83	256.17	3.03	399.93	4.84	258.68	3.06	396.70
350	4.80	255.96	3.49	349.99	4.84	255.28	3.61	348.3

Table 4.7: Simulation results, 5000 runs, 5 hops, end-to-end BER 10^{-3}, delay constraint 515 ms. Average modulation order and end-to-end delay when circuit power is reduced 50% and completely ignored.

4.3.6 Single- vs. Multi-hop Comparison

Routing protocol issues are not addressed directly in this thesis. The goal is merely to optimize a given route. However, it is highly relevant to examine how the total energy consumption is affected when resorting to many short hops, compared to fewer, but longer, hops. This analysis is important when designing energy-aware routing protocols, to find what is the desirable hop length, from an energy-saving perspective.

Table 4.8 shows the maximum allowed modulation order, the corresponding required transmission power and time and the energy consumption per bit, for various hop lengths. The circuit power consumption is now set to the initial values, as given in Table 4.1, and the required per-hop BER $P_b = 10^{-3}$. No delay constraint is taken into consideration.

Obviously, the maximum modulation order decreases and the transmission time and energy consumption per bit increases as the hop length grows longer. What is interesting is to observe when multiple, short hops becomes more energy-efficient than a single, longer hop.

The results show that for short distances, a single, long hop is more energy-efficient than multiple short hops. E.g. will 5 hops of 4 meters each in total consume $5 \times 2.56 = 12.8$ mJ, as compared to 8.52 mJ for a single hop of 20 meters. That is, resorting to 5 hops consumes approximately 50% more energy per bit, and the total transmission time is also higher. At approximately 15 m, the situation changes; two hops of 15 m requires slightly less energy than a single hop of 30 m. The time consumption is also roughly equal, ignoring any intermediate queuing and processing delay. There is not much difference between the continuous and the discrete schemes is these results.

The modulation order is unconstrained for these results, and is very high for the shortest distances (16 bits/symbol for 1 m). This is, of course, not feasible to implement in a real system, where there will be a upper limit for the modulation order. For these short distances, the circuit power dominates over the transmission power. Thus, using a lower modulation order than the theoretical, optimal value will cause the energy consumption per bit to increase for the short hops, making a single hop more feasible. The severity of this problem decreases as the hop length increases. It should also be noted that in order to maintain the same end-to-end BER, resorting to more hops requires a lower per-hop BER, compared to if fewer hops are used, as defined in (3.21).

Hop length [m]	*Trans. scheme*	*Max. mod. order*	*Trans. power [mW]*	*Trans. time [ms]*	*Energy/bit [mJ]*
1	CR	16.45	3.70	12.45	1.47
	DR	16.00	2.72	12.80	1.46
2	CR	12.91	3.70	15.87	1.88
	DR	13.00	3.95	15.75	1.88
4	CR	9.60	4.37	21.33	2.56
	DR	10.00	5.74	20.48	2.57
8	CR	6.75	7.02	30.36	3.86
	DR	7.00	8.35	29.26	3.87
10	CR	5.83	8.16	35.12	4.53
	DR	6.00	9.17	34.13	4.53
15	CR	4.50	13.33	45.47	6.33
	DR	4.00	9.29	51.20	6.46
20	CR	3.59	18.89	56.97	8.52
	DR	4.00	25.42	51.20	8.54
30	CR	2.44	31.86	83.81	13.09
	DR	2.00	21.69	102.40	13.63
40	CR	1.88	52.62	108.94	19.20
	DR	1.00	31.92	204.80	30.20

Table 4.8: Simulation results, single run, per-hop BER 10^{-3}. Minimum-energy configuration for various hop lengths. All results given per hop.

Chapter 5

Discussion

The simulation results show that joint end-to-end route optimization across the protocol layers can potentially offer tremendous energy savings, and that the increased flexibility of the adaptive schemes enables the system to satisfy stringent QoS requirements. However, it should be stressed that the proposed model and simulations are based on a number of idealized assumptions; mainly the absence of competing traffic, perfect timing of the wake-up/sleep cycle of the nodes, and zero queuing/buffer delay. The effect of inaccurate link state information is only briefly discussed. Thus, the simulation results reflect the theoretically best performance results, and should be treated accordingly. It is difficult to directly compare the performance of the scheme proposed here with the one proposed in [9], since they are based on different ground assumptions regarding the channel models.

5.1 Model Review

The single-hop results show that for a given hop length, there exists a optimal modulation order that minimized the total energy associated with a packet transmission over this link, for a given target BER. The physical layer model and system parameters are adopted from [8], and the single-hop results comply well with the results in the article. This adaptive modulation and power control scheme belongs in the physical layer, which has no knowledge of end-to-end QoS requirements.

By jointly optimizing all nodes in a multihop route, across the physical, access and network layers, subject to end-to-end QoS constraints, the optimal configuration for all nodes can be found. While this configuration might not allow all nodes to transmit in the most energy-efficient manner possible from a local perspective, the total energy consumption for all nodes throughout the route is minimized, while the QoS constraints are satisfied. Further, the number of hops in an arbitrary route will vary, but the same end-to-end QoS requirements will still be valid. Thus, it makes sense to jointly optimize all nodes throughout the route. If each node is optimized in isolation, without knowledge of the route as a whole, it would be very difficult to satisfy such end-to-end requirements.

The allocation of modulation orders to the nodes is equivalent to optimal, dynamic time slot allocation. The modulation order/rate is increased over short hops/high quality links, where the energy cost per bit is small, resulting in a short time slot. Long hops/low quality links are allocated lower transmission rates/longer time slots, since the extra cost of increasing the rate is then much higher than for the shorter hops. This 'water-filling' is a well-known principle for communication over time-varying wireless channels. A fixed-rate system cannot adapt to the different hop lengths in other ways than channel inversion, which is a sub-optimal solution, compared to the water-filling principle [2].

As the simulation results show, the increased flexibility of the adaptive system allows it to meet far tighter delay constraints than the fixed-rate system, while consuming less energy. Continuous modulation orders yields, as expected, the optimal solution, since more accurate adaptation is then possible. However, the penalty associated when restricting the modulation order to integer values is quite small; the energy consumption per bit increases with 1-2%, on average. This is in line with other results [2, 43].

In order to obtain the optimal route configuration, accurate link state information must be available in the sink. Using inaccurate link state information in the simulations causes performance degradation, in terms of increased energy consumption and/or more frequent transmission errors. That is, in order to keep the transmission error rate reasonably low, the transmission power must be increased. The required extra power increases fast when the uncertainty associated with the link state estimates is larger than a few percent. Of course, there will always be some delay associated with the propagation of the link state information to the sink, running the optimization routine and back-propagate the route configuration, thereby causing some inaccuracy in the link state estimates. But in a relatively stable network, the estimates will still be good enough. On the other hand, a high-mobility network, where the hop lengths/link states change more frequently than what the system can keep up with, will cause the proposed scheme to fail. In such cases, some best-effort scheme, like transmitting at a fixed, high transmission power or using ARQ, would be a better solution. In fact, in the simulations, no transmission errors occurred when always using maximum available transmission power.

With the set of system parameters used for the simulations, the circuit power consumption is quite high, compared to the available transmission power, while the transition time is so short that it has negligible impact on the result. In fact, the total circuit power of the receiving and transmitting nodes added together is larger than the available transmission power. As confirmed by the simulation results, this favors short transmission times over keeping the transmission power at a low level, which is in accordance with other, related publications [6]. Reducing the circuit energy, causes the adaptive system to select lower modulation orders. When the circuit power is completely ignored, the optimal configuration yields an end-to-end delay that is as close to the delay constraint as possible.

For the given system parameters, it is shown that fewer, but longer hops are more energy-efficient than many short hops. This is valid up to approximately 30 m, where two 15 m hops requires slightly less energy than a single hop of 30 m. This indicates that the proposed ground assumption for the routing strategy, where one of the 3 closest nodes were always selected for the next hop, is probably not optimal from an energy-efficiency point of view. Rather, more distant nodes should also be allowed for the next hop, resulting in a higher average hop length. More hops also means more queuing delay (not investigated here), and more complicated multiple access problems. On the other hand, higher transmission power causes more interference to other nodes in the network. And, of course, for a given node in a random ad hoc network, it might not be possible to find a next hop that is of (near) optimal length, especially in sparse networks. To summarize, there are many things that needs to be taken into consideration when defining what is an 'optimal' route between two given nodes in an energy-constrained wireless ad hoc network. Thus, the main strength of the proposed scheme is the ability to adapt to (almost) any given route, making the best out of the situation at hand, which is shown to provide significant energy savings under the given assumptions.

5.2 Optimization Routine Scheduling

The optimization routine presented here is done for a single packet. The energy cost of doing this optimization for every transmitted packet in a practical system, however, would obviously swamp the energy savings obtained by optimizing the route in the first place. So several packets have to be transmitted on the same route, using the same configuration, for this scheme to be feasible. This raises the problem of how often a route should be re-configured (if necessary), after the initial optimization, which again depends on the propagation environment, node mobility and the traffic pattern.

Routes could be maintained in a proactive manner, with periodic re-configurations. Running frequent updates cause much overhead, which will increase the total energy consumption. On the other hand, too infrequent updates could lead to sub-optimal performance and increased transmission error rates. There is no clear answer to what is the optimal schedule, since it is highly dependent on the network mobility.

In a reactive approach, the optimization routine can be trigged by a packet loss. This reduces the overhead, but there will be some time before the route is re-configured and ready to be used again. Once a route has been broken, it might also be impossible to successfully re-configure it, in which case a new route must be found. Especially in delay-sensitive applications this will be a problem.

5.3 Distributed Power Control

In a practical system, the modulation order will be restricted to integer values, while the transmission power can be adapted more accurately. It is therefore plausible to assume that the transmission powers should be more frequently updated than the modulation orders in order to maintain the optimal route configuration, regardless of if proactive or reactive re-configuration scheduling is used. This can be exploited by making the optimization scheme partially distributed.

Link states are measured locally at each node, and must be relayed to the sink in order to run the optimization process. Relaying this information for every packet transmission is also obviously not feasible. Also, there is always some delay introduced when relaying packets to the sink, making the estimates slightly out of date. But since link states can be measured locally every time a transmission takes place, all nodes have access to recent link state information about their own in- and outgoing links. Therefore, it makes sense to adjust the transmission power locally, at each individual node and with respect to its allocated modulation order, once the initial route configuration is relayed back to the nodes. This saves the overhead of relaying the link state information to the sink, and allows more accurate power adaptation. The modulation orders, however, will remain the same until a new end-to-end optimization routine is run.

This approach is justified by the fact that the target BER for each hop is known at the involved nodes. Further, the required transmission power for each individual node is expressed as a deterministic function of the corresponding modulation order. Thus, local transmission power adjustments can be done locally, without any performance degradation. The modification makes the route configuration scheme partly distributed, and effectively divides it into a setup phase and a maintenance phase. This is similar to what is proposed in [23].

This modified version allows the transmission power to be adjusted more frequently than in the original version, without any additional overhead. A complete end-to-end route optimization operation that also updates the modulation orders (i.e. a reconfiguration) should now be run whenever the energy gain by doing so is larger than the associated overhead cost (proactive), or when the required transmission power can no longer be delivered at any of the intermediate nodes (reactive).

5.4 Circuit Power Model Re-Visited

As mentioned in Section 3.2.2, the hardware circuit power consumption model used might not be entirely correct. First of all, the assumption that the circuit power consumption is independent on modulation order should be used with care, especially in systems where a wide range of modulation orders are available. Ideally, the hardware should also adapt its operation (and power consumption) to what is required in order to satisfy a given configuration [5]. This would ensure optimal utilization of available resources, but might be difficult to implement in a practical system.

Further, the PA's efficiency is taken to be independent on the output power. This is a rather coarse approximation, as the efficiency of most PAs used for this kind of application is also dependent on the amplifier's output power [42]. Also, the type of PA used, thereby also the power consumption model, is dependent on the modulation technique used. Here, QAM/PSK is used due to the simple relationship between the constellation size and symbol transmission time. Constant-envelope modulation techniques, such as FSK, have less stringent requirements to the PA's linearity. This allows other types of PAs, with higher drain efficiency [6], to be used. For short-range applications, where the RF circuit design has a large influence on the overall energy consumption, it is vital to choose the right modulation technique for a given application in order to obtain truly energy-efficient solutions. In the implemented simulation scripts, it is straightforward to adjust all hardware parameters so that other hardware models and modulation techniques can be tested in the same framework.

5.5 Self-Congestion

Even in the absence of competing traffic, which is assumed for this model, self-congestion might pose a problem for adaptive rate multihop schemes, where the rates can be different across the hops. The weakest link will have the lowest throughput rate, thereby making it the bottleneck in the route. If packets are generated at the source faster than they can be transmitted by this bottleneck link, the buffer of the node transmitting over this link gradually be filled. This is referred to as *self-congestion* [44]. If too much traffic is sent over this route, the throughput delay will gradually increase and packets will eventually be dropped, due to buffer overflow.

This problem can be addressed at several protocol layers. In the network layer, a new route setup procedure can be initiated before the buffer capacity of the bottleneck node is breached. This requires some knowledge of the transmission duration if the network layer. Addressing the problem at the transport layer offers another possibility: The transmission rate of the source can be adjusted so that it does not exceed the capacity of the bottleneck. Yet another approach is to integrate the traffic characteristics into the route setup procedure. In this case, multiple routes between a source/sink pair can be found, if needed, thereby increasing the net capacity between the two nodes.

5.6 Computations: Resources and Energy

In this model, the time and energy required when performing the optimization routine itself have not been addressed. In resource-limited systems, this must of course be taken into consideration. Some applications, such as WSNs, will typically have a fixed data fusion center or gateway, which is the destination for many incoming connections. These nodes, which usually will have far greater computational capabilities and energy reserves, are well suited for taking the burden of optimizing its incoming connections. If no central node exists, and the traffic is evenly distributed over all peer nodes, they will all take their fair share of the burden. But if a single node is the destination for many connections, it might experience a quick depletion of the batteries, leaving it useless.

Regardless, the complexity of the optimization routine should be minimized. The energy-greedy optimization algorithm used for discrete rate adaptation is best described as a truncated brute-force search over the possible configurations. If the number of hops is reasonably low, and the maximum modulation order is limited, the optimal solution will be found after relatively few iterations. However, the complexity increases with the number of hops and range of allowed modulation orders. Other, more efficient methods to solve this problem might exist.

As previously stated, finding the optimal modulation order in the single hop case for the model used here can be shown to be a convex problem [6]. This is utilized in [45], which, for a given set of input parameters, aims to arrive at a simple, closed-form expression for finding the optimal modulation order for QAM transmission over a single hop, by means of curve-fitting. The optimal modulation order can than be found directly, without running any optimization algorithms. This work is based on the same underlying hardware and channel models as adopted here. If a similar approach can be used on the L-dimensional optimization problem of a multihop route, the energy required for running the optimization routine could probably be significantly reduced.

Chapter 6

Conclusions

Wireless ad hoc networks have the potential of becoming a dominant network technology in the near future. One of the main advantages of such networks is that they can function without any fixed infrastructure or base station, making deployment both easy and cost-effective. Further, the distributed nature makes these networks fault-tolerant, as there is no single critical point of error. However, distributed network control will always lead to some performance degradation.

Many applications will rely on small, battery-powered nodes, with limited resources and energy-reserves, which makes energy-efficiency a paramount design goal. This, combined with the other special characteristics of wireless ad hoc networks, raises many technological challenges, requiring novel techniques and aproaches. Some of these, which are believed to be among the most important, have been summarized in this thesis. Most notably, the cross-layer design paradigm is of major importance, as it has the potential of providing tremendous performance improvement over the traditional, strictly layered OSI reference model, especially in systems with limited energy and/or QoS requirements. Despite the advantages of the cross-layer design, it does come with a cost. Closer integration between layer implies that different layers of the protocol stack must be jointly designed, thereby complicating the design and making reuse and further development of existing protocols and systems more difficult.

Based on the special characteristics of energy-constrained wireless ad hoc networks, and novel techniques proposed in related literature, an energy-efficient, cross-layer route optimization scheme has been proposed. It was shown how a given multihop route consisting of AWGN links can be energy-optimized, subject to QoS requirements, in terms of end-to-end throughput and delay, by utilizing adaptive modulation and demodulation and power control in the intermediate nodes. The route configuration is based on link state information from the given route, and all nodes throughout the route are jointly optimized, across the physical, access and network layers. Random routes, on which the proposed technique was applied, were generated using statistical tools, based on an initial, qualitative analysis of the characteristics of energy-aware routing protocols.

The hardware model used in the system is targeted at short-range, low-power devices, so that the circuit power consumption is of the same order of magnitude as the transmission power. It is shown that when taking the circuit power consumption into consideration in a low-power adaptive rate system, the traditional belief that maximizing the transmission time is the most energy-efficient strategy no longer holds. Rather, the optimal modulation order for a given point-to-point transmission is found as the best trade-off between saving circuit energy at the cost of transmission energy, and vice versa. The simulation results depend heavily on the underlying hardware and channel parameters, both of which can easily be adjusting in the developed simulation script.

Simulations run on multihop routes, showed that the proposed route optimization scheme offers significant energy savings, while improving the system's capability to meet stringent QoS requirements. Compared to a fixed-rate system, the proposed optimization scheme can on average reduce the total energy consumption per transmitted bit by more that 50%, while satisfying stricter delay constraints, when allowing continuous rate adaptation. Resticting the modulation order to discrete, integer values causes 1-2% higher energy consumption, compared to the optimal, continuous case. Further simulations showed that inaccurate link state information can result in severe performance degredation, making the scheme best suited for relatively stable and immoblie networks.

The proposed cross-layer route optimization scheme allows energy-efficient link adaptation when transmitting over variable-quality links, while the end-to-end optimization ensures that the QoS requirements are satisfied and that a global, optimal solution is found. Without this end-to-end optimization, each node would have been configured in isolation from the others, potentially wasting valuable energy and definitely making it very difficult to satisfy end-to-end QoS requirents. The proposed scheme can in theory be combined with any existing routing protocol, by optimizing both new and existing routes. Re-configuring existing routes is beneficial in order to prolong their lifetime, thereby reducing the need for running energy-demanding route setup procedures.

The proposed scheme is based on a number of idealized assumptions; mainly no contending traffic on the route, zero buffer/queuing delay, no overhead associated with channel access and perfect sleep-scheduling of the nodes. These assumptions are obviously not valid in real systems. However, the proposed model can serve as a reference for benchmark testing of parts of real systems, since the idealized conditions yields the theoretical upper performance bound for the given situation. The main problems with the proposed model is the need for accurate link state information, and challenges associated with sleep scheduling and channel access of intermediate nodes. Further, when link rates throughout a route are different, low-capacity links will become bottlenecks, which might cause self-congestion.

6.1 Contributions

- Utilizing adaptive modulation link adaptation techniques in an end-to-end route optimization scheme.
- Using statistical approximations of energy-aware routing in a random wireless ad hoc network, and use this to easily generate route-approximations in a network of a given size and density.
- Developing the energy-greedy route optimization algorithm.
- Developing a framework for testing various route optimization schemes. Flexibility and easy configuration has been emphasized.

6.2 Future Work

In this thesis, a basic system model that is well suited for demonstrating the principles for the proposed route optimization scheme, has been used. The model is based on a number of idealized assumptions, resulting in the best achievable results. Only some qualitative analyses on how a real-life situation would affect the performance of the system are made. Extending the model so that contending traffic, interference from other nodes and channel access issues are included in the problem and simulations represents the next natural step. The hardware model should also be reviewed. Further, the complexity of the optimization algorithms should be scrutinized and, if possible, improved. If the optimization can be proven to be convex, this could be utilized to make more efficient algorithms. Finally, a proper routing protocol could be implemented. It would also be interesting to investigate if the proposed scheme could be combined with existing protocols and then be implemented in more advanced network simulators, such as `ns2`, in order to run more representable simulations.

Bibliography

[1] A. Goldsmith and S.B. Wicker. Design Challenges for Energy-Constrained Ad Hoc Networks. *IEEE Wireless Communications.* Vol. 9, no. 4, pages 8-27, 2002.

[2] A. Goldsmith. *Wireless Communications.* Cambridge University Press, 2005.

[3] G. Karlsson, S. Lindfors, M. Skoglund, and G. Øien. Cross-Layer Optimization in Short-Range Wireless Sensor Networks. CROPS project description and slides, 2005. Available at: http://www.ee.kth.se/commth/projects/CROPS/.

[4] R. Ramanathan, J. Redi, and BBN Technologies. A Brief Overview of Ad Hoc Networks: Challenges and Directions. *IEEE Communications Magazine.* Vol. 40, no. 5, pages 20-22, 2002.

[5] R. Min, M. Bhardwaj, S. Cho, N. Ickes, E. Shih, A. Sinha, A. Wang, and A. Chandrakasan. Energy-Centric Enabling Technologies for Wireless Sensor Networks. *IEEE Wireless Communications.* Vol. 9, no. 4, pages 28-39, 2002.

[6] S.R. Cui. *Cross-Layer Optimization in Energy Constrained Networks.* PhD thesis, Stanford University, 2005.

[7] V. Kawadia and P.R. Kumar. Principles and Protocols for Power Control in Wireless Ad Hoc Networks. *IEEE Journal on Selected Areas in Communications.* Vol. 23, no. 1, pages 76-88, 2005.

[8] S. Cui, AJ Goldsmith, and A. Bahai. Energy-Constrained Modulation Optimization. *IEEE Transactions on Wireless Communications.* Vol. 4, no. 5, pages 2349-2360, 2005.

[9] H.C. Yang, L. Yang, and K. Wu. Minimum-Energy Route Configuration for Wireless Ad Hoc Networks. *25th IEEE International Performance,*

Computing, and Communications Conference (IPCCC). Apr. 2006, Phoenix, USA.

[10] I.F. Akyildiz, T. Melodia, and K.R. Chowdhury. A survey on wireless multimedia sensor networks. *Computer networks.* Issue 51, pages 921-960, 2006.

[11] Official website for ZigBee Alliance: http://www.zigbee.org/.

[12] Official website for Bluetooth: http://bluetooth.com/.

[13] B. Zhen, J. Park, and Y.; Kim. Scatternet Formation of Bluetooth Ad Hoc Networks. *Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS).* Jan. 2003, Island of Hawaii, USA.

[14] K. Römer and F. Mattern. The Design Space Of Wireless Sensor Networks. *IEEE Wireless Communications.* Vol. 11, no. 6, pages 54-61, 2004.

[15] E. Shih, S.H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan. Physical Layer Driven Protocol and Algorithm Design for Energy-Efficient Wireless Sensor Networks. *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking.* Jul. 2001, Rome, Italy, Pages 272-287.

[16] A.S. Tannenbaum. *Computer Networks, 4th ed.* Prentice Hall, 2003.

[17] Y. Zhang and L. Cheng. Cross-layer Optimization for Sensor Networks. *New York Metro Area Networking Workshop (NYMAN).* Sep. 2003, New York, USA, Pages 247-252.

[18] J.G. Proakis. *Digital Communications, 4th ed.* McGraw-Hill Higher Education, 2001.

[19] T. ElBatt and A. Ephremides. Joint Scheduling and Power Control for Wireless Ad Hoc Networks. *IEEE Transactions on Wireless Communications.* Vol. 3, no. 1, pages 74-85, 2004.

[20] S. Cui, R. Madan, A. Goldsmith, and S. Lall. Joint Routing, MAC and Link Layer Optimization in Sensor Networks with Energy Constraints. *IEEE International Conference on Communications (ICC).* May 2005, Seoul, South-Korea, Vol. 2.

[21] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless Sensor Networks: a Survey. *Computer Networks.* Vol. 38, no. 4, pages 393-422, 2002.

[22] A.Y. Wang, S.H. Cho, C.G. Sodini, and A.P. Chandrakasan. Energy Efficient Modulation and MAC for Asymmetric RF Microsensorsystems. *International Symposium on Low Power Electronics and Design (ISLPED).* Aug. 2001, Huntington Beach, USA, Pages 106-111.

[23] M.L. Sichitiu. Cross-Layer Scheduling for Power Efficiency in Wireless Sensor Networks. *23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM).* Mar. 2004, Hong Kong, China, Vol. 3, Pages 1740-1750.

[24] R.L. Peterson, R.E. Ziemer, and D.E. Borth. *Introduction to Spread-Spectrum Communications.* Prentice Hall Englewood Cliffs, 1995.

[25] Q. Lie, S. Zhou, and G.B. Giannakis. Cross-Layer Combining of Adaptive Modulation and Coding With Truncated ARQ over Wireless Network Links. *IEEE Transactions on Wireless Communications.* Vol. 3, no. 5, pages 1746-2755, 2004.

[26] S. Toumpis and A. Goldsmith. New media access protocols for wireless ad hoc networks based on cross-layer principles. *IEEE Transactions on Wireless Communications.* Vol. 5, no. 8, pages , 2006.

[27] S. Xu and T. Saadawi. Does the IEEE 802.11 MAC Protocol Work Well in Multihop Wireless Ad Hoc Networks? *IEEE Communications Magazine.* Vol. 39, no. 6, pages 130-137, 2001.

[28] N. Bisnik and A. Abouzeid. Queuing Delay and Achievable Throughput in Random Access Wireless Ad Hoc Networks. 2006. Available at: citeseer.ist.psu.edu/bisnik06queuing.html.

[29] M. Rahnema. Overview of the GSM system and protocol architecture. *IEEE Communications Magazine.* Vol. 31, no. 4, pages 92-100, 1993.

[30] J. Elson and D. Estrin. Time Synchronization for Wireless Sensor Networks. *Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS).* Apr. 2001, San Francisco, USA, Pages 1965-1970.

[31] C.C. Enz, A. El-Hoiydi, J.D. Decotignie, and V. Peiris. WiseNET: An Ultralow-Power Wireless Sensor Network Solution. *Computer.* Vol. 37, no. 8, pages 62-70, 2004.

[32] S. Cui, A. Goldsmith, and A. Bahai. Joint Modulation and Multiple Access Optimization under Energy Constraints. *IEEE Global Telecommunications Conference (GLOBECOM).* Nov. 2004, Dallas, USA, Vol. 1.

[33] K. Sohrabi, J. Gao, V. Ailawadhi, and G.J. Pottie. Protocols for Self-Organization of a Wireless Sensor Network. *IEEE Personal Communications.* Vol. 7, no. 5, pages 16-27, 2000.

[34] T. Holliday, A. Goldsmith, P. Glynn, and N. Bambos. Distributed power and admission control for time varying wireless networks. *IEEE Global Telecommunications Conference (GLOBECOM).* Nov. 2004, Dallas, Texas, Vol. 2.

[35] S. Chen and K. Nahrstedt. Distributed Quality-of-Service Routing in Ad Hoc Networks. *IEEE Journal on Selected Areas in Communications.* Vol. 17, no. 8, pages 1488-1505, 1999.

[36] S. Doshi, S. Bhandare, and T.X. Brown. An On-demand Minimum Energy Routing Protocol for a Wireless Ad Hoc Network. *ACM SIGMOBILE Mobile Computing and Communications Review.* Vol. 6, no. 3, pages 50-66, 2002.

[37] R.C. Shah and J.M. Rabaey. Energy Aware Routing for Low Energy Ad Hoc Sensor Networks. *IEEE Wireless Communications and Networking Conference (WCNC).* Mar. 2002, Orlando, USA, Vol. 1.

[38] Q. Li, J. Aslam, and D. Rus. Online Power-Aware Routing in Wireless Ad-Hoc Networks. *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking.* Jul. 2001, Rome, Italy, Pages 97-107.

[39] T. Camp, J. Boleng, and V. Davies. A Survey of Mobility Models for Ad Hoc Network Research. *Wireless Communications and Mobile Computing.* Vol. 2, no. 5, pages 483-501, 2002.

[40] J.H. Chang and L. Tassiulas. Energy Conserving Routing in Wireless Ad Hoc Networks. *19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM).* Mar. 2000, Tel Aviv, Israel, Vol. 1.

[41] V. Kawadia and PR Kumar. A Cautionary Perspective on Cross-Layer Design. *IEEE Wireless Communications.* Vol. 12, no. 1, pages 3-11, 2005.

[42] S. Lindfors. Power consumption of a class-b power amplifier. Draft, 2007, Helsinki University of Technology.

[43] A. Gjendemsjø, G.E. Øien, and P. Orten. Optimal Discrete-Level Power Control for Adaptive Coded Modulation Schemes with Capacity-Approaching Component Codes. *IEEE International Conference on Communications (ICC).* Jun. 2006, Istanbul, Turkey, Pages 3498-3502.

[44] B. Girod, E. Setton, and X. Zhu. Congestion-Optimized Routing and Scheduling of Video Over Wireless Ad Hoc Networks. *International Symposium on Circuits and Systems (ISCAS)*, May 2005, Kobe, Japan.

[45] C. Wang and G. Øien. Optimum Modulation Order Adaptations for MQAM in Wireless Sensor Networks. Draft, 2007, Norwegian University of Science and Technology.

Appendix A

List of Abbreviations

AWGN Additive White Gaussian Noise

CSMA Carrier Sense Multiple Access

MIMO Multiple Input Multiple Output

SNIR Signal to Noise and Interference Ratio

ARQ Automatic Repeat Request

BER Bit Error Rate

CTS Clear To Send

FSK Frequency Shift Keying

GSM Global System for Mobile Telecommunication

MAC Multiple Access

OSI Open Systems Interconnection

PAR Peak-to-Average Ratio

PER Packet Error Rate

PLL Phase Locked Loop

PSK Phase Shift Keying

QAM Quadrature Amplitude Modulation

QoS Quality of Service

RTS Ready To Send

SNR Signal to Noise Ratio

TCP Transfer Control Protocol

WSN Wireless Sensor Network

IP Internet Protocol

PA Power Amplifier

RF Radio Frequency

Appendix B

Matlab Scripts

B.1 Main Script

scr_sys.m

```
%% INITIALIZATION
clear all,close all,clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% System parameters (Adjustable)

        % Route length
         L_route = 5;
        %  %Time constraint, end-to-end [ms]
         T = 515;
        % Fixed-rate bit/symbol
         b_fix = 2;
        % Mean hop length [m]
         ch_m = 11.5;
        % Channel gain misadjustment
         g_misad = 1;
        % SNR power margin
         pwr_m = 3;
        % Target BER, from table 1 in (4)
         P_b_end_to_end = 1e-3;
         P_b = 1 - (1 - P_b_end_to_end)^(1/L_route);
%          P_b = 1e-4;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        c_sim = struct('L_route', L_route, 'T', T, 'b_fix',
            b_fix, 'ch_m', ch_m, 'P_b_EE', P_b_end_to_end, 'P_b
            ', P_b, 'g_misad', g_misad, 'pwr_m', pwr_m);
        save param_sim c_sim
        scr_constants;
        clear all

        load param_sim
        load param_sys

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   No. simulation runs

     N_run = 5000;

%     Other variables (temp stuff)
    N_bad_routes = 0;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    Some variables
    acc_res_fr = zeros(N_run, 5);
    acc_res_fr_fp = zeros(N_run, 5);
    acc_res_cr = zeros(N_run, 5);
    acc_res_dr = zeros(N_run, 5);
%     acc_res_cr_tsa = zeros(N_run, 5);
%     acc_res_dr_tsa = zeros(N_run, 5);

    route_cnf_fr = zeros(N_run, 1);
    route_cnf_cr = zeros(N_run, 1);
    route_cnf_dr = zeros(N_run, 1);
%     route_cnf_cr_tsa = zeros(N_run, 1);
%     route_cnf_dr_tsa = zeros(N_run, 1);

    T_cr = 0;
    T_dr = 0;
    T_fr = 0;
%     T_dr_tsa = 0;
%     T_cr_tsa = 0;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% START GIANT COUNTER

for bigCounter = 1:N_run

% %% VARIABLE ROUTE LENGTH
%          % Route length
%           L_route = ceil(rand*10);
%          %  %Time constraint, end-to-end [ms]
%           P_b_end_to_end = 1e-3;
%           P_b = 1 - (1 - P_b_end_to_end)^(1/L_route);
% %           Overwrite
%           c_sim.L_route = L_route;
%           c_sim.P_b = P_b;

%% ROUTE SETUP

disp([num2str(bigCounter) '/' num2str(N_run)])
% disp([num2str(bigCounter) '/' num2str(N_run) 'Route Length: '
     num2str(L_route)])

% Generate hop lengths
 ch_dist = fnc_ch_create(c_sim.ch_m,c_sim.L_route);
%     ch_dist = [5 12 35];
% ch_dist = [1 2 4 8 10 15 16 20 30 40];
% Find link gains
ch_gains = fnc_ch_calc_gain(c_sys.G_1, c_sys.k, c_sys.M_l,
    ch_dist);
% Add CSI misadjustment, 1: Gaussian dist, 0: uniform dist
ch_gains_misad = fnc_ch_gain_misad(ch_gains, c_sim.g_misad, 1);
% Check if there exists link that cannot be used
N_bad_routes = N_bad_routes + fnc_route_usable(c_sys, c_sim,
    ch_gains_misad);

%% OPTIMIZATION

% ### Find the optimal transmission scheme for route ###

% Fixed rate, adaptive power
tic
route_cnf_fr = fnc_nonad_opt(c_sys, c_sim, ch_gains_misad);
T_fr = T_fr + toc;

% Continous rate, dynamic time slot allocation
tic
route_cnf_cr = fnc_linkad_opt_c(c_sys, c_sim, ch_gains_misad);
T_cr = T_cr + toc;

% Discrete rate, dynamic time slot allocation
tic
route_cnf_dr = fnc_linkad_opt_d(c_sys, c_sim, ch_gains_misad);
T_dr = T_dr + toc;


% % % Continous rate, fixed time slot allocation
% tic
% route_cnf_cr_tsa = fnc_linkad_opt(c_sys, c_sim,
    ch_gains_misad, 0);
% T_cr_tsa = T_cr_tsa + toc;

% % % Discrete rate, fixed time slot allocation
% tic
% route_cnf_dr_tsa = fnc_linkad_opt(c_sys, c_sim,
    ch_gains_misad, 1);
% T_dr_tsa = T_dr_tsa + toc;


% ### Check if successful transmission scheme found ###

% Fixed rate
if (length(find(route_cnf_fr(:,1) > 0)) == c_sim.L_route)
    route_cnf_fr_ok = 1;
else
```

```
    route_cnf_fr_ok = 0;
end

% Continous rate, dynamic time slot allocation
if (length(find(route_cnf_cr(:,1) > 0)) == c_sim.L_route)
    route_cnf_cr_ok = 1;
else
    route_cnf_cr_ok = 0;
end

% Discrete rate, dynamic time slot allocation
if (length(find(route_cnf_dr(:,1) > 0)) == c_sim.L_route)
    route_cnf_dr_ok = 1;
else
    route_cnf_dr_ok = 0;
end

% % Continous rate, fixed time slot allocation
% if (length(find(route_cnf_cr_tsa(:,1) > 0)) == c_sim.L_route)
%     route_cnf_cr_tsa_ok = 1;
% else
%     route_cnf_cr_tsa_ok = 0;
% end

% % Discrete rate, fixed time slot allocation
% if (length(find(route_cnf_dr_tsa(:,1) > 0)) == c_sim.L_route)
%     route_cnf_dr_tsa_ok = 1;
% else
%     route_cnf_dr_tsa_ok = 0;
% end

%% TRANSMISSION

% ### Transmit only if successful configuration found ###

% Fixed rate
if (route_cnf_fr_ok)
    route_res_fr = fnc_route_tx(c_sys, c_sim, ch_gains,
        route_cnf_fr, 0);
    route_res_fr_fp = fnc_route_tx(c_sys, c_sim, ch_gains,
        route_cnf_fr, 1);
else
    route_res_fr = zeros(1,6);
    route_res_fr_fp = zeros(1,6);
end

% Continous rate
if (route_cnf_cr_ok)
    route_res_cr = fnc_route_tx(c_sys, c_sim, ch_gains,
        route_cnf_cr, 0);
else
    route_res_cr = zeros(1,6);
end

```

```
% Discrete rate
if (route_cnf_dr_ok)
    route_res_dr = fnc_route_tx(c_sys, c_sim, ch_gains,
        route_cnf_dr, 0);
else
    route_res_dr = zeros(1,6);
end

% % ### Fixed time slot allocation ###
% % Discrete rate
% if (route_cnf_cr_tsa_ok)
%     route_res_cr_tsa = fnc_route_tx(c_sys, c_sim, ch_gains,
    route_cnf_cr_tsa, 0);
% else
%     route_res_cr_tsa = zeros(1,6);
% end
%
% if (route_cnf_dr_tsa_ok)
%     route_res_dr_tsa = fnc_route_tx(c_sys, c_sim, ch_gains,
    route_cnf_dr_tsa, 0);
% else
%     route_res_dr_tsa = zeros(1,6);
% end

%% EVALUATION

% Evaluate results
acc_res_fr_fp(bigCounter,:)  = fnc_route_calc_res(c_sim,
    route_res_fr_fp, route_cnf_fr_ok);
acc_res_fr(bigCounter,:) = fnc_route_calc_res(c_sim,
    route_res_fr, route_cnf_fr_ok);
acc_res_cr(bigCounter,:) = fnc_route_calc_res(c_sim,
    route_res_cr, route_cnf_cr_ok);
acc_res_dr(bigCounter,:) = fnc_route_calc_res(c_sim,
    route_res_dr, route_cnf_dr_ok);
% acc_res_dr_tsa(bigCounter,:) = fnc_route_calc_res(c_sim,
    route_res_dr_tsa, route_cnf_dr_tsa_ok);
% acc_res_cr_tsa(bigCounter,:) = fnc_route_calc_res(c_sim,
    route_res_cr_tsa, route_cnf_cr_tsa_ok);

%% END GIANT LOOP
end

%% PRESENTATION

% Simulation parameters
disp(' ')
disp(['# Runs________________________: ' num2str(N_run)])
disp(['# Hops in Route_______________: ' num2str(c_sim.L_route)
    ])
disp(['End-to-end delay constraint___: ' num2str(c_sim.T) ' ms'
    ])
disp(['End-to-end target BER_________: ' num2str(c_sim.P_b_EE)
    ])
```

```
disp(['Link gain misadjustment_______: ' num2str(c_sim.g_misad) ' %'])
disp(['Required SNR margin___________: ' num2str(c_sim.pwr_m) ' %'])
disp(['Fixed-rate bit/symbol_________: ' num2str(c_sim.b_fix) ])

% If single run, show details
if (N_run == 1)

    % Show SNR limits
    fnc_show_snr_limits(c_sys, c_sim);

    %   Present Route information
        disp(' ')
        disp('*** Route Information ***')
    print_route_info(ch_dist, ch_gains, ch_gains_misad);
    disp(' ')
    disp('##########################################################')
    disp('Results for transmission over a single route')

%     Transmission results
    if(route_cnf_fr_ok)
        disp(' ')
        disp('*** Fixed Rate, Fixed Power ***')
        print_results(c_sys, c_sim, route_res_fr_fp);
        disp(' ')
        disp('*** Fixed Rate, Variable Power ***')
        print_results(c_sys, c_sim, route_res_fr);
    else
        disp(' ')
        disp('*** Setup Error for Fixed Rate ****')
        print_setup(c_sys, c_sim, route_cnf_fr);
    end

    if(route_cnf_cr_ok)
        disp(' ')
        disp('*** Continous Rate, Dynamic time slot allocation ***')
        print_results(c_sys, c_sim, route_res_cr);
    else
        disp(' ')
        disp('*** Setup Error for Continous Rate, Dynamic time slot allocation ****')
        print_setup(c_sys, c_sim, route_cnf_cr);
    end

    if(route_cnf_dr_ok)
        disp(' ')
        disp('*** Discrete Rate, Dynamic time slot allocation ***')
        print_results(c_sys, c_sim, route_res_dr);
    else
        disp(' ')
        disp('*** Setup Error for Discrete Rate, Dynamic time slot allocation ****')
        print_setup(c_sys, c_sim, route_cnf_dr);
    end

% % %     Fixed time slot allcoation
%     if(route_cnf_cr_tsa_ok)
%         disp(' ')
%         disp('*** Continous Rate, Fixed time slot allocation ***')
%         print_results(c_sys, c_sim, route_res_cr_tsa);
%     else
%         disp(' ')
%         disp('*** Setup Error for Continous Rate, Fixed time slot allocation ****')
%         print_setup(c_sys, c_sim, route_cnf_cr_tsa);
%     end
%
%     if(route_cnf_dr_tsa_ok)
%         disp(' ')
%         disp('*** Discrete Rate, Fixed time slot allocation ***')
%         print_results(c_sys, c_sim, route_res_dr_tsa);
%     else
%         disp(' ')
%         disp('*** Setup Error for Discrete Rate, Fixed time slot allocation ****')
%         print_setup(c_sys, c_sim, route_cnf_dr_tsa);
%     end

end
% Present averaged results
disp(' ')
disp('###########################################################')
disp('Averages')
disp(' ')
disp('# Fixed Rate, Fixed power:')
[E_fr_fp, P_fr_fp, ER_fr_fp] = print_accum_res(c_sys, c_sim, acc_res_fr_fp, N_run);
disp(' ')
disp('# Fixed Rate, Variable power:')
[E_fr, P_fr, ER_fr] = print_accum_res(c_sys, c_sim, acc_res_fr, N_run);
disp(' ')
disp('# Continous Rate, dynamic time slot allocation:');
[E_cr, P_cr, ER_cr] = print_accum_res(c_sys, c_sim, acc_res_cr, N_run);
disp(' ')
disp('# Discrete Rate, dynamic time slot allocation:');
[E_dr, P_dr, ER_dr] = print_accum_res(c_sys, c_sim, acc_res_dr,
        N_run);
% disp(' ')
% disp('# Continous Rate, fixed time slot allocation:');
% [E_cr_tsa, P_cr_tsa, ER_cr_tsa] = print_accum_res(c_sys,
    c_sim, acc_res_cr_tsa, N_run);
% disp(' ')
% disp('# Discrete Rate, fixed time slot allocation:');
% [E_dr_tsa, P_dr_tsa, ER_dr_tsa] = print_accum_res(c_sys,
    c_sim, acc_res_dr_tsa, N_run);

% Compare key results
disp(' ')
disp('
    ##########################################################'
    )
disp('Comparison')

disp(' ')
disp('# Fixed rate, fixed power (FR_FP): ')
disp(['  Energy: ' num2str(10*log10(E_fr_fp)) ' dBm, error rate
    : ' num2str(ER_fr_fp) ' %'])

disp(' ')
disp('# Fixed rate, variable power (FR_VP): ')
disp(['  Energy: ' num2str(10*log10(E_fr)) ' dBm, error rate: '
     num2str(ER_fr) ' %'])
imp = 100*(E_fr_fp - E_fr)/E_fr_fp;
disp(['    Improvement over FR_FP: ' num2str(imp) ' %'])

disp(' ')
disp('# Continous rate, dynamic time (CR): ')
disp(['  Energy: ' num2str(10*log10(E_cr)) ' dBm, error rate: '
     num2str(ER_cr) ' %'])
imp = 100*(E_fr_fp - E_cr)/E_fr_fp;
imp2 = 100*(E_fr - E_cr)/E_fr;
disp(['    Improvent over FR_FP: ' num2str(imp) ' %'])
disp(['    Improvent over FR_VP: ' num2str(imp2) ' %'])

disp(' ')
disp('# Discrete rate, dynamic time (DR): ')
disp(['  Energy: ' num2str(10*log10(E_dr)) ' dBm, error rate: '
     num2str(ER_dr) ' %'])
imp = 100*(E_dr - E_cr)/E_cr;
disp(['    Degredation from CR: ' num2str(imp) ' %'])

% disp(' ')
% disp('# Continous rate, Fixed time (CR_TSA): ')
% disp(['  Energy: ' num2str(10*log10(E_cr_tsa)) ' dBm, error
    rate: ' num2str(ER_cr_tsa) ' %'])
% imp = 100*(E_cr_tsa - E_cr)/E_cr;
% disp(['    Degredation from CR: ' num2str(imp) ' %'])

% disp(' ')
% disp('# Discrete rate, fixed time (DR_TSA): ')
% disp(['  Energy: ' num2str(10*log10(E_dr_tsa)) ' dBm, error
    rate: ' num2str(ER_dr_tsa) ' %'])
% imp = 100*(E_dr_tsa - E_cr_tsa)/E_cr_tsa;
% imp2 = 100*(E_dr_tsa - E_dr)/E_dr;
% disp(['    Degredation from CR_TSA: ' num2str(imp) ' %'])
% disp(['    Degredation from DR: ' num2str(imp2) ' %'])
%
% disp(' ')
% disp(['Error Rate (should be): ' num2str(100*(N_bad_routes/
    N_run)) '%'])

% Optimization time
disp(' ')
disp('
    ###########################################################'
    )
disp('Average Time Consumption per iteration')
disp([' #FR_VP: ' num2str(T_fr/N_run)])
disp([' #CR: ' num2str(T_cr/N_run)])
disp([' #DR: ' num2str(T_dr/N_run)])
% disp([' #CR_TSA: ' num2str(T_cr_tsa/N_run)])
% disp([' #DR_TSA: ' num2str(T_dr_tsa/N_run)])


%%
```

B.2 System Constants

scr_constants.m (user-defined system parameters)

```
clc, clear all, close all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% System parameters (fixed)

    % Reference gain factor 30dB
     G_1 = 10^3;
    % path-loss-exponent
     k = 3.5;
    % Link margin 40dB
     M_l = 10^4;
    % Bandwith [Hz]
     B = 10000;
    % Additive noise power, -174dBm
     sigma_2 = 10^-17.4;
    % Noise figure
     N_f = 10;
    % Interference power
     I = 0;
    % Packet length [bits]
     L_pkt = 2048;
    % Max Tx power [mW]
```

```
     P_max = 250;
    % Transision time [ms]
     T_tr = 5e-3;
    % PA drain efficency parameter
     eta = 0.35;
    % Calculate total noise and interference power [mW]
     N_tot = 2*B*sigma_2*N_f + I;

     %%% Circuit power

     P_mix = 30.3;
     P_syn = 50;
     P_fil_tx = 2.5;
     P_fil_rx = P_fil_tx;
     P_adc = 6.7;
     P_dac = 15.4;
     P_lna = 20;
     P_ifa = 3;

%     Total transmitter circuit power consumption (Constant)
     P_ctx = P_mix + P_syn + P_fil_tx + P_dac;
     P_crx = P_mix + P_syn + P_fil_rx + P_adc + P_lna + P_ifa;

%      P_max = P_max - 0.5*P_ctx;
%      P_ctx = 0.5*P_ctx;
%      P_crx = 0.5*P_crx;

%      Transision circuit power consumption
     P_tr = P_syn;
%      P_tr = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The bit step size

    % b: bit per symbol

    % Min b
     b_min = 1;
    % Max b
     b_max = 20;
    % b step size (continous mode)
     b_stp = 0.01;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Struct that hold system parameters, known to nodes
c_sys = struct('B', B, 'L_pkt', L_pkt, 'P_max', P_max, 'P_ctx',
     P_ctx, 'P_crx', P_crx, 'P_tr', P_tr, 'T_tr', T_tr, 'N_tot'
    , N_tot, 'G_1', G_1, 'k', k, 'M_l', M_l, 'b_min', b_min, '
    b_max', b_max, 'b_stp', b_stp, 'eta', eta);

% Save wanted constants
save param_sys c_sys

% Clear workspace
```

```
clear all
```

B.3 Route Optimization

Optimization routines: Continuous, Discrete and Channel Inversion.

B.3.1 Continuous

fnc_linkad_opt_c.m.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     function:   fnc_linkad_opt_c
%
%     Continuous modulation order route optimization. Uses
    fmincon
%
%     Input:
%     c_sys: system parameters
%     c_sim: simulation parameters
%     ch_gains: Link gain vector
%
%     Output:
%     res: result vector of dimension Lx2, consists of
%         b: Modulation order, -1 if no valid config exists
%         P_tx: transmission power
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function res = fnc_linkad_opt_c(c_sys, c_sim, ch_gains)
% Return transmission configuration: [b P_tx] (Both column
    vectors)
% b = -1 for unsuccessful configuration

%   b_ll and b_ul are vector of length L_route
%   lower and upper limit for b vector
    b_ll = c_sys.b_min*ones(1,c_sim.L_route);
    b_ul = fnc_linkad_find_b_max(c_sys, c_sim, ch_gains);

%     There is a link in the route too weak to be used.
%     Return -1 for this and largest possible b for the others
    br_ind = find(b_ul < b_ll);
    if (length(br_ind) > 0)
%         Debugging
%         disp('b_ul < b_ll in fnc_linkad_opt_c(...)')
%         b_ll
%         b_ul
        b_ul(br_ind) = -1;
        p_tx_vec = zeros(1,c_sim.L_route);
        res = [b_ul' p_tx_vec'];
        return
```

```matlab
    end

%     Check if largest b meet time requirement
    T_min = sum(fnc_calc_T(c_sys, c_sim, b_ul));
    if(T_min > c_sim.T)

%         Debugging
%         disp('T_min > T_lim in fnc_linkad_opt_c(...)')
%         T_min
%         c_sim.T

        res = [-1*ones(c_sim.L_route,1) zeros(c_sim.L_route,1)];
        return

    end

%     Optimization routine
    options_tmp = optimset('fmincon');
%     options = optimset(options_tmp,'LargeScale','off', 'TolCon', 0, 'TolFun', 1e-6, 'TolX', 0.0001, 'Diagnostics', 'off', 'Display', 'off', 'FunValCheck', 'on');
    options = optimset(options_tmp,'LargeScale','off','TolCon', 1e-5, 'TolFun', 1e-3, 'TolX', 1e-3, 'Diagnostics', 'off', 'Display', 'off', 'FunValCheck', 'on');
    [b_opt,exitflag] = fmincon(@E_tot, b_ul, [], [], [], [], b_ll, b_ul, @T_tot, options);
    % Extra output values, for debugging:
    % [b,fval,exitflag,output,lambda,grad,hessian]
    % exitflag


    if (exitflag < 0)% No solution found, exitflag -1
        exitflag
        output
        lambda
%         pause
        res = [-1*ones(c_sim.L_route,1) zeros(c_sim.L_route,1)];
    else % Solution found
        p_tx_vec = fnc_calc_P_tx(c_sys, c_sim, b_opt, ch_gains);
        res = [b_opt' p_tx_vec'];
    end

 %%%%%%%% Nested functions

%     Energy (Objective function)
    function E = E_tot(x)
%             Find transmission power associated with rate vector
            P_tx = fnc_calc_P_tx(c_sys, c_sim, x, ch_gains);
%             Find total energy consumption
            E = sum(fnc_calc_E(c_sys, c_sim, x, P_tx));

    end

%     Time (Nonlinear constraint)
    function [c, ceq] = T_tot(y)

%         Time
        T_tot = sum(fnc_calc_T(c_sys, c_sim, y));
        c = T_tot - c_sim.T;
        ceq = [];

    end

end
```

B.3.2 Discrete

fnc_linkad_opt_d.m.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     function:   fnc_linkad_opt_d
%
%     Discrete modulation order route optimization. Implements
%     energy-greedy optimization algorithm.
%
%     Input:
%     c_sys: system parameters
%     c_sim: simulation parameters
%     ch_gains: Link gain vector
%
%     Output:
%     res: result vector of dimension Lx2, consists of
%         b: Modulation order, -1 if no valid config exists
%         P_tx: transmission power
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function res = fnc_linkad_opt_d(c_sys, c_sim, ch_gains)

res = zeros(c_sim.L_route,2);
E_chk = zeros(1,c_sim.L_route);
T_chk = zeros(1,c_sim.L_route);

% b_ul and b_ll are vectors of length L_route
b_ul = floor(fnc_linkad_find_b_max(c_sys, c_sim, ch_gains));
% b_ll = c_sys.b_min*ones(1,c_sim.L_route);

% Check if any links in the route are too weak
br_ind = find(b_ul < c_sys.b_min);
if (length(br_ind) > 0)

%     Debugging:
%     disp('b_ul < b_ll in fnc_linkad_opt_d(...)')
%     b_ll
%     b_ul

    b_ul(br_ind) = -1;
    p_tx_vec = zeros(1,c_sim.L_route);

    res = [b_ul' p_tx_vec'];
    return
end

% Check if the largest possible b vector meets the time
    requirement
T_min = sum(fnc_calc_T(c_sys, c_sim, b_ul));
if (T_min > c_sim.T)

%     Debugging:
%     disp('T_min > T_lim in fnc_linkad_opt_d(...)')
%     T_min
%     c_sim.T

    res = [-1*ones(c_sim.L_route,1) zeros(c_sim.L_route,1)];
    return

end

% Initial check passed, running through algorithm
b_opt = b_ul;
p_tx_opt = fnc_calc_P_tx(c_sys, c_sim, b_opt, ch_gains);
E_opt = sum(fnc_calc_E(c_sys, c_sim, b_opt, p_tx_opt));

go = 1;
while (go)

    b_prev_opt = b_opt;

    for n = 1:c_sim.L_route
%         Decrease b with 1 for each hop
        b_tmp = b_prev_opt;
        b_tmp(n) = b_tmp(n) - 1;
        %Check if valid constellation size
        %Set invalid b to large value
        if (b_tmp(n) ≥ c_sys.b_min)
            T_tmp = sum(fnc_calc_T(c_sys, c_sim, b_tmp));
            if (T_tmp ≤ c_sim.T)
                p_tx_tmp = fnc_calc_P_tx(c_sys, c_sim, b_tmp,
                    ch_gains);
                E_tmp = sum(fnc_calc_E(c_sys, c_sim, b_tmp,
                    p_tx_tmp));
                if (E_tmp < E_opt)
                    E_opt = E_tmp;
                    b_opt = b_tmp;
                    p_tx_opt = p_tx_tmp;
                end
            end
        end
    end %end FOR

    if (b_prev_opt == b_opt)
        go = 0;
    end


end%End WHILE

    res = [b_opt' p_tx_opt'];

return
```

B.3.3 Channel Inversion

fnc_nonad_opt.m.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%      function:    fnc_nonad_opt
%
%      Channel inversion, fixed rate route configuration.
%
%      Input:
%      c_sys: system parameters
%      c_sim: simulation parameters
%      ch_gains: Link gain vector
%
%      Output:
%      res: result vector of dimension Lx2, consists of
%          b: Modulation order, -1 if no valid config exists
%          P_tx: transmission power
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function res = fnc_nonad_opt(c_sys, c_sim, ch_gains)

res = zeros(c_sim.L_route, 2);

% Time Slots
% T_hop = (c_sim.T - c_sys.T_tr*c_sim.L_route)/c_sim.L_route;

% Transmission power
b_vec = c_sim.b_fix*ones(1,c_sim.L_route);
p_tx_vec = fnc_calc_P_tx(c_sys, c_sim, b_vec, ch_gains);

b_ul = fnc_linkad_find_b_max(c_sys, c_sim, ch_gains);
T_used = sum(fnc_calc_T(c_sys, c_sim, b_vec));

br_ind = find(b_vec > b_ul);

if (length(br_ind) > 0)
```

```

%     disp('b_ul < b_ll in fnc_nonad_opt(...)')
%     b_vec
%     b_ul

    b_vec(br_ind) = -1;
    p_tx_vec = fnc_calc_P_tx(c_sys, c_sim, b_ul, ch_gains);

    res = [b_vec' p_tx_vec'];
    return

elseif (T_used > c_sim.T)

%     disp('T_min > T_lim in fnc_nonad_opt_d(...)')
%     T_used
%     c_sim.T

    res = [-1*ones(c_sim.L_route,1) zeros(c_sim.L_route,1)];
    return

end

res = [b_vec' p_tx_vec'];

return
```

B.4 Transmission

fnc_route_tx.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     function:   fnc_route_tx
%
%     Packet transmission over given route, using given route
%     configuration. Hop-by-hop transmission, terminates if
    error in
%     previous hop.
%
%     Input:
%     c_sys: system parameters
%     c_sim: simulation parameters
%     g_vec: Link gain vector
%     r_cnf: Route configuration (mod.order and tr.power)
%     fix_p: Boolen value. Transmit at max power if 1, use
    given congif if
%     0
%
%     Output:
%     res_matr: Result matrix, dimestion Lx6, each row consists
     of
```

```
%                per-hop results:
%          flag: status flag. 1: OK, 0, err in previous hop, -1:
     tr. error
%          b: modulation order
%          P: transmission power
%          T: transmission time
%          E: energy consumption
%          SNR: received SNR
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function res_matr = fnc_route_tx(c_sys, c_sim, g_vec, r_cnf,
    fix_p)
% res_matr: [flag b P T E SNR]
% r_cnf: [b P]: -1 no valid b, -2 could not meet snr.

% TODO: make retransmission-routine!
% flag suggestion:
% -1: snr error
%  0: error in prev. hop
%  1: ok

% Initiate variables
res_matr = zeros(c_sim.L_route,6);
err_ind = find(r_cnf(:,1) < 1);

% Check if fixed power is to be used
if (fix_p)
   P_tx_max = fnc_calc_P_tx_max(c_sys, c_sim.b_fix);
   r_cnf(:,2) = ones(c_sim.L_route,1)*P_tx_max;
end

% Check if configuration is valid
if (length(err_ind) > 0)
    disp('Invalid configuration accepted in route_tx')
    r_cnf
    pause
end

% Transmit hop-by-hop. Cancel if a transmission error occurs
res_matr(1,:) = fnc_hop_tx(c_sys, c_sim, g_vec(1), r_cnf(1,1),
    r_cnf(1,2));
for(n = 2:c_sim.L_route)
    if(res_matr(n-1,1) == 1)
        res_matr(n,:) = fnc_hop_tx(c_sys, c_sim, g_vec(n),
            r_cnf(n,1), r_cnf(n,2));
    else
        res_matr(n,:) = [0 0 0 0 0 0];
    end
end

return
```

fnc_hop_tx.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     function:   fnc_hop_tx
%
%     Transmission over a single hop. Compares calculated,
    received
%     SNIR with required threshold for given modulation order
%
%     Input:
%     c_sys: system parameters
%     c_sis: simulation parameters
%     g: Link gain
%     b: modulation order
%     P_t: Transmission power
%
%     Output:
%     res: Result vector, consisting of:
%         flag: 1 if success, −1 if fail
%         b: modulation order
%         P_t: Transmission power
%         T: Transmission time
%         E: Used energy
%         snr_rec: Received SNIR
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function res = fnc_hop_tx(c_sys, c_sim, g, b, P_t);
% [flag b P T E snr_rec]
%

% % Debugging
% if(P_t == 1)
%     disp('*** Minimum Tx power used! ***')
% elseif (P_t < 1)
%     disp('*** Even smaller TX power used ***')
% end

% if (b > c_sys.b_max)
%     disp('b exeeds b_max in fnc_hop_tx(...)')
%     g
%     b
%     P_t
%     pause
% end

flag = 1;
% Find minimum SNR threshold
snr_min = fnc_calc_snr_min(c_sim.P_b, b, 0); % No extra margin
    here
snr_rec = P_t*g/(c_sys.N_tot);
T = fnc_calc_T(c_sys, c_sim, b);
E = fnc_calc_E(c_sys, c_sim, b, P_t);
```

```

if (snr_rec < snr_min)
    if (abs(snr_rec−snr_min) > 0.0001)
        flag = −1;
% %           Debugging
%           disp(['SNR ERROR IN TRANSMISSION!'])
%           disp(['Received SNR: ' num2str(10*log10(snr_rec)) '
    dBm'])
%           disp(['Required SNR: ' num2str(10*log10(snr_min)) '
    dBm'])
    end
end

% Return hop results
res = [flag b P_t T E snr_rec];
return
```

B.5 Result Presentation

fnc_route_calc_res.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     function:   fnc_route_calc_res
%
%     Calculates the averages of a route transmission
%
%     Input:
%     c_sim: system parameters
%     hop_res: Matrix of hop results. L rows
%     conf_succ: Boolean value if the configuration was
    successful.
%
%     Output:
%     res: Route averages, consisting of:
%         flag: status flag. 1: OK,  −1: setup err, −2: Tx err
%         E: Total energy consumption
%         T: Total end−to−end delay
%         b_avg: average modulation order
%         P_avg: average transmission power
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function res = fnc_route_calc_res(c_sim, hop_res, conf_succ)
% route_res: [flag b P T E SNR]
% acc_res: [flag E T b_avg P_avg] 1: OK,  −1: setup err, −2: Tx
     err

res = zeros(1,5);

if (conf_succ)
    %     Energy consumed also if failed transmission
    res(2) = sum(hop_res(:,5)); % acc energy
    if (length(find(hop_res(:,1) > 0)) == c_sim.L_route)
        res(1) = 1; %OK
        res(3) = sum(hop_res(:,4));  % end-to-end time, if succ
        res(4) = mean(hop_res(:,2)); % avg b, if succ
        res(5) = mean(hop_res(:,3)); % avg P, if succ
    else
        res(1) = -2;   %TX error, energy is still used
    end
else
    res(1) = -1; %Setup error
end
```

print_results.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      function:   print_results
%
%      Presents the detailed route results in table form
%
%      Input:
%      c_sys: system parameters
%      c_sim: simulation parameters
%      res: Result matrix, number of rows equals numbers of hops in route
%
%      Output:
%      none
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function print_results(c_sys, c_sim, res)
% res: [flag b P T E SNR]

disp(['|--------|---------|--------|-----------------|----------|------------|'])
disp(['|  Hop  |  flag  |   b   |  Ptx  |  T_on  |E/bit[uJ]| SNR[dB]  |'])
disp(['|--------|---------|--------|-----------------|----------|------------|'])
for m=1:c_sim.L_route
    if (res(m,6) > 0)
        snr_db = 10*log10(res(m,6));
    else
        snr_db = 0;
    end
    [s_res, e_res] = sprintf('|%-7.0f|%-8.0f|%-7.2f|%-7.2f|%-8.2f|%-9.2f|%-11.2f|', m,res(m,1:4),1000*res(m,5)/c_sys.L_pkt, snr_db);
    disp(s_res)
end
disp(['|--------|---------|--------|-----------------|----------|------------|'])

```

```
if (length(find(res(:,1) == 1)) == c_sim.L_route)
    disp('***** Transmission OK *****')
    E_total      = sum(res(:,5));
    E_per_bit    = E_total/c_sys.L_pkt;
    E_per_bit_dBm = 10*log10(E_per_bit);
    T_total      = sum(res(:,4));
    T_mean       = mean(res(:,4));
    E_per_bit_per_hop = mean(res(:,5))/c_sys.L_pkt;
    E_per_bit_per_hop_dB = 10*log10(E_per_bit_per_hop);

    disp([' Total energy consumtion: ' num2str(E_total) ' mJ'])
    disp([' Total end-to-end time  : ' num2str(T_total) ' ms'])
    disp([' Average T_on per hop: ' num2str(T_mean) ' ms'])
else
    disp('***** Transmission ERROR *****')
end

return
```

print_accum_res.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     function:   print_accum_res
%
%     Calculates simulation averages and presents results.
%
%     Input:
%     c_sys: system parameters
%     c_sim: simulation parameters
%     res: Result matrix, number of rows equals numbers of
    simulation runs
%     N_tot: Number of runs (for result validation)
%
%     Output:
%     res: result vector of dimension Lx2, consists of
%         E_b_avg: Average energy consumption per bit
%         P_tx_avg: Average transmission power
%         err_rate: Total error rate
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [E_b_avg, P_tx_avg, err_rate] = print_accum_res(c_sys,
     c_sim, res, N_tot)
% [flag E T b_avg P_avg] 1: OK, -1: setup, err, -2: Tx err

N_ok = length(find(res(:,1) == 1));
N_ok_re = length(find(res(:,1) == 2));
N_err_setup = length(find(res(:,1) == -1));
N_err_tx = length(find(res(:,1) == -2));

ind_ok = find(res(:,1) > 0);
res_ok = res(ind_ok,:);

if (N_ok + N_ok_re + N_err_setup + N_err_tx ≠ N_tot)
```

```
  disp('Err/Corr count fault!')
  return
end

N_ok_tot = N_ok + N_ok_re;
N_err_tot = N_err_setup + N_err_tx;
err_rate = 100*(N_err_tot)/N_tot;

if (N_err_tot > 0)
    err_rate_su = 100*(N_err_setup/N_err_tot);
    err_rate_tx = 100*(N_err_tx/N_err_tot);
else
    err_rate_su = 0;
    err_rate_tx = 0;
end

% % % % Not implemented
% if (N_ok_re > 0)
%     retrans_rate = 100*(N_ok_re/N_ok_tot);
% else
%     retrans_rate = 0;
% end

if (N_ok_tot > 0)

    E_p_avg = mean(res_ok(:,2));
    E_p_max = max(res_ok(:,2));
    E_p_min = min(res_ok(:,2));

    E_b_avg = E_p_avg/c_sys.L_pkt;
    E_b_avg_dbm = 10*log10(E_b_avg);

    T_avg = mean(res_ok(:,3));
    T_max = max(res_ok(:,3));
    T_min = min(res_ok(:,3));

    b_avg = mean(res_ok(:,4));
    P_tx_avg = mean(res_ok(:,5));

    disp(['  *Energy per packet [mJ]:'])
    disp(['       - Mean: ' num2str(E_p_avg) ', Max: ' num2str(
        E_p_max) ', Min: ' num2str(E_p_min) ])
    disp(['  *Energy per bit [mJ]:'])
    disp(['       - Mean: ' num2str(E_b_avg) ' (' num2str(
        E_b_avg_dbm) ' dBm)'])
    disp(['  *Time per packet[ms]:'])
    disp(['       - Mean: ' num2str(T_avg) ', Max: ' num2str(
        T_max) ', Min: ' num2str(T_min) ])
    disp(['  *Bit per symbol (mean): ' num2str(b_avg) ])
    disp(['  *Tx power(mean) [mW]: ' num2str(P_tx_avg) ])
    disp(['  *Error rate: ' num2str(err_rate) ' % (Setup: '
        num2str(err_rate_su) ', %, Tx: ' num2str(err_rate_tx) '
         %)' ])
%     disp(['  *Retransmissions: ' num2str(retrans_rate) ' %'])
```

```

    if(T_max > c_sim.T)
        if (abs(T_max - c_sim.T) > 0.0001)
            disp('Time Capacity Breached!!!!')
            disp(['Time [s]: ' num2str( T_max/1000)])
            disp(['Limit [s]: ' num2str( c_sim.T/1000)])
            pause
        end
    end

else
    disp(['  All transmissions failed!'])
    disp(['  *Error rate: ' num2str(err_rate) ' % (Setup: '
        num2str(err_rate_su) ', %, Tx: ' num2str(err_rate_tx) '
         %)' ])
%     disp(['  *Retransmissions: ' num2str(retrans_rate) ' %'])
    E_b_avg = 1;
    P_tx_avg = 0;
end

return
```

print_setup.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     function:   print_setup
%
%     Presents the route configuration details.
%
%     Input:
%     d: Hop lengths
%     g: Channel gains
%     g_misad: Channel gains with added misadjustment
%
%     Output:
%     none
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function print_setup(c_sys, c_sim, setup)

disp(['|-------|--------|-------|'])
disp(['|  Hop  |  flag  |  Ptx  | '])
disp(['|-------|--------|-------|'])
for m=1:c_sim.L_route
    [s_res, e_res] = sprintf('|%-7.0f|%-8.0f|%-7.2f|', m,setup(
        m,:));
    disp(s_res)
end
disp(['|-------|--------|-------|'])

return
```

print_route_info.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      function:    print_route_info
%
%      Presents route information.
%
%      Input:
%      d: Hop lengths
%      g: Channel gains
%      g_misad: Channel gains with added misadjustment
%
%      Output:
%      none
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function print_route_info(d, g , g_misad)

    disp(['|---------|---------|----------|---------|-----------|'])
    disp(['|   Hop   |    d    |   g_re    |  g_est  |  est_err  |'])
    disp(['|---------|---------|----------|---------|-----------|'])

for (m=1:length(d))
    g_dev = (100*(g(m)-g_misad(m))/g(m));
    [s_ch, e_ch] = sprintf('|%-9.0f|%-9.2f|%-10.2f|%-9.2f|%-11.1f|',m,d(m),10*log10(g(m)),10*log10(g_misad(m)),g_dev);
    disp(s_ch);
end
    disp(['|---------|---------|----------|---------|-----------|'])


return
```

B.6 Misc. Functions

fnc_ch_create.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     function:   fnc_ch_create
%
%     Creates a route of the desired length, each hop modeled
    as
%     a Rayleigh-distributed random variable
%
%     Input:
%     m: Distribution parameter
%     L: Route length
%
%     Output:
%     r: Route hop lengths, 1xL vector
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function r = fnc_ch_create(m,L)

    prm = ones(1,L)*m;
    r = raylrnd(prm);

return
```

fnc_ch_calc_gain.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     function:   fnc_ch_calc_gain
%
%     Calculates the channel gain vector
%
%     Input:
%     G_1: Gain factor
%     k: Path loss exponent
%     M_l: Link margin
%     d: Hop length vector
%
%     Output:
%     g: Link gains per hop
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function g = fnc_ch_calc_gain(G_1,k,M_l,d)

    g = 1./(G_1*d.^k*M_l);

return
```

fnc_ch_gain_misad.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%    function:   fnc_ch_gain_misad
%
%    Add link state information estimation error of desired
    severity. If
%    desired, uniform distribution of the error can be
    selecteder
%
%    Input:
%    g: Link gain vector
%    a: Misadjustment severity, in \%
%    gauss: Boolean value. Gaussion–distributed error if 1,
    uniform if 0
%
%    Output:
%    g_ad: New, misadjusted link gain vector.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function g_ad = fnc_ch_gain_misad(g,a,gauss)

if (gauss) % Gaussian dist of error
    v_ad = randn(size(g));
else % Uniform dist of error
    v_ad = rand(size(g)) − 0.5;
end

g_ad = g.*(1 + v_ad*a/100);

return
```

fnc_linkad_find_b_max.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%    function:   fnc_linkad_find_b_max
%
%    Calculates the maximum modulation order per hop for a
    given route
%
%    Input:
%    c_sys: system parameters
%    c_sim: simulation parameters
%    g_vec: Link gain vector
%
%    Output:
%    b_max: Maximum modulation order vector
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function b_max = fnc_linkad_find_b_max(c_sys, c_sim, g_vec)

b_max = zeros(1,length(g_vec));

% modulation order vector
b = c_sys.b_min:c_sys.b_stp:c_sys.b_max;

% SNR limits
snr_lim_vec = fnc_calc_snr_min(c_sim.P_b, b, c_sim.pwr_m);
% Find required transmission power
P_tx_max_vec = fnc_calc_P_tx_max(c_sys, b);

for n = 1:length(g_vec)

%     Largest received SNR for given hop
    snr_r_max = (P_tx_max_vec*g_vec(n))/c_sys.N_tot;
%     Find valid
    inx = max(find(snr_r_max > snr_lim_vec));

%     Select largest, if exists
    if (length(inx) > 0)
        b_max(n) = c_sys.b_min + (inx-1)*c_sys.b_stp;
    else
        b_max(n) = -1;
    end

end

return
```

fnc_calc_P_tx.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     function:   fnc_calc_P_tx
%
%     Calculate the power vector, given vectors g and b
%     g and b must be of the same lengts (scalar or vector), or
    one can
%     be a scalar
%
%     Input:
%     c_sys: system parameters
%     c_sim: simulation parameters
%     b: modulation order vectors
%     g: Link gain vector
%
%     Output:
%     p: power vector
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function p = fnc_calc_P_tx(c_sys, c_sim, b, g)

```

```
    if ((length(b) > 1) && (length(g) > 1))
       if (length(b) ≠ length(g))
          disp('ERROR IN fnc_calc_P_tx. If b and p vector, they
               must be of same length')
          b_length = length(b)
          p_length = length(g)
       end
    end

    snr_min = fnc_calc_snr_min(c_sim.P_b, b, c_sim.pwr_m);
    p = (c_sys.N_tot./g).*snr_min;

return
```

fnc_calc_P_tx_max.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     function:   fnc_calc_P_tx_max
%
%     Calculates the maximum available transmission power for a
%     given modulation order
%
%     Input:
%     c_sys: system parameters
%     b: modulation order vector (or scalar)
%
%     Output:
%     P_tx_max: Max power vector. Same dimension as b
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function P_tx_max = fnc_calc_P_tx_max(c_sys, b);

    alph_vec = fnc_calc_PA_eff_fac(c_sys.eta, b);
    P_tx_max = (c_sys.P_max - c_sys.P_ctx)./(1 + alph_vec);

return
```

fnc_calc_PA_eff_fac.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     function:   fnc_calc_PA_eff_fac
%
%     Calculates the PA effeiency factor, alpha, as function of
%     modulation order
%
%     Input:
%     eta: PA drain efficiency
%     b: modulation order vector (or scalar)
%
%     Output:
```

```
%     alpha: Power efficiency vector. Same dimension as b
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function alpha = fnc_calc_PA_eff_fac(eta, b)

    indx = find(b ≤ 2);
    b(indx) = 2;
    ksi = 3*((sqrt(2.^b)−1)./(sqrt(2.^b)+1));
    alpha = ksi/eta − 1;

return
```

fnc_calc_snr_min.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     function:    fnc_calc_snr_min
%
%     Calculates the required received SNIR threshold, given a
    target
%     BER and a modulation order
%
%     Input:
%     Pb: Target BER (scalar)
%     b: modulation order vector (or scalar)
%     pwr_m: Extra safety margin
%
%     Output:
%     gma_min: SNIR theshold vector. Same dimension as b
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function gma_min = fnc_calc_snr_min(Pb, b, pwr_m)

%     Debugging
    if (find(b ≤ 0))
        disp('b is 0!!!')
    end

    gma_min = zeros(1,length(b));

    ind_psk = find(b < 2);
    ind_qam = find(b ≥ 2);

    b_psk = b(ind_psk);
    b_qam = b(ind_qam);

    gma_psk = log(2./(Pb*b_psk)).*(1./(sin(pi./(2.^b_psk)).^2))
        *(1+pwr_m/100);
    gma_qam = log((4*(1−(1./sqrt(2.^b_qam))))./(Pb*b_qam))
        .*(((2.^b_qam)−1)*(2/3))*(1+pwr_m/100);

    gma_min(ind_psk) = gma_psk;
    gma_min(ind_qam) = gma_qam;
```

```

return
```

fnc_calc_T.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     function:   fnc_calc_T
%
%     Calcualates the transmission time for each
%     hop, including the transient time for the receiving node
%
%     Input:
%     c_sys: system parameters
%     c_sim: simulation parameters
%     b: modulation order vector (or scalar)
%
%     Output:
%     T_vec: Vector of transmission times. Same dimension as b
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function T_vec = fnc_calc_T(c_sys, c_sim, b)
    T_vec = zeros(1,length(b));

    K = (c_sys.L_pkt/c_sys.B)*1000;
    T_vec = K./b + c_sys.T_tr;

return
```

fnc_calc_E.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     function:   fnc_calc_E
%
%     Calculates the energy consumption per hop, based on route
%     configuration vector and channel gain vector
%
%     Input:
%     c_sys: system parameters
%     c_sim: simulation parameters
%     b: modulation order vector
%     P_tx: power vector
%
%     Output:
%     E_vec: Energy vector
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function E_vec = fnc_calc_E(c_sys, c_sim, b, P_tx)

%     Check if b and P_tx vecors of same dimension
    if ((length(b) > 1) && (length(P_tx) > 1))
```

```
        if (length(b) ≠ length(P_tx))
            disp('ERROR IN calc_E. If b and P_tx vectors, they
                must be of same length')
            b
            P_tx
            size(b)
            size(P_tx)
            pause
        end
    end

    E_vec = zeros(1,length(b));

%     The power effiency for each hop
    alph_vec = fnc_calc_PA_eff_fac(c_sys.eta, b);
%     Transmission time
    K = (c_sys.L_pkt/c_sys.B)*1000;
    T_on = K./b;
%     Energy penalty for transient time
    E_tr = c_sys.P_tr*c_sys.T_tr*length(b);
%     Energy vector

%     With power consumption:
    E_vec = ((1+alph_vec).*P_tx.*T_on + (c_sys.P_ctx+c_sys.
        P_crx).*T_on + E_tr)/1000;

return
```

fnc_show_snr_limits.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     function:   fnc_show_snr_limits
%
%     Shows the SNR limits for some modulation orders
%
%     Input:
%     c_sys: system parameters
%     c_sim: simulation parameters
%
%     Output:
%     none
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function fnc_show_snr_limits(c_sys, c_sim)


    b_cont = c_sys.b_min:c_sys.b_stp:c_sys.b_max;
    b_disc = 1:1:c_sys.b_max;

    gma_min_d = fnc_calc_snr_min(c_sim.P_b, b_disc, 0);
    gma_min_c = fnc_calc_snr_min(c_sim.P_b, b_cont,0);


    disp(' ')
    disp(['SNR-limits for MQAM, Target BER = ' num2str(c_sim.
        P_b)])
    disp(['|   b    |    SNR     |    dB    |'])
    disp(['|--------|------------|----------|'])
    for m=1:c_sys.b_max
        [s1, e1] = sprintf('%-7.0f', m);
        [s2, e2] = sprintf('%-9.1f', (gma_min_d(m)));
        [s3, e3] = sprintf('%-8.3f', (10*log10(gma_min_d(m))));
        disp(['|' s1 '|' s2 '|' s3 '|' ])
    end
    disp(['|--------|------------|----------|'])

%     figure,plot(b_cont,10*log10(gma_min_c)), grid on
%     hold on
%     stem(1:b_max,10*log10(gma_min_d))

return
```

B.7 Network Analysis

scr_nw.m

```
clear all
close all
clc

x_size  = 200; %Area size
N_nd = 100; % # Nodes
N_cl = 3; % Find averages of N_cl closest for each

N_runs = 100;

res = zeros(N_nd*N_runs,1);
res_route = zeros(N_runs,6);

% big counter
for bc=1:N_runs

    clear nodes
    clear dst

    disp([ num2str(bc) '/' num2str(N_runs) ])

    % Create network
    nw = nt_fnc_create_nw(x_size,N_nd);

%     Find random sink
    sink = ceil(rand*N_nd);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Find average of N closest node, which closer to sink than
        itself, for
    % all nodes

    % Find min max and avg of N_cl nodes, in direction of
       center

    for n=1:N_nd
        if (n ≠ sink)
            res((bc−1)*N_nd+n) = nt_fnc_find_M_closest(nw,n,
                sink,N_cl);
        end
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Find a route from a random node to the sink, using
       shortest−hop
    % algorithm

    % Choose a random node
    nodes(1) = ceil(rand*N_nd);

    n = 1;
    go = 1;

    % While sink not reached
    while (go)

%         Find next node towards sink
        [hop,node_next] = nt_fnc_next_node(nw,nodes(n),sink);
        % Vector of distances
        dst(n) = hop;
        % Vector of node indices
        nodes(n+1)= node_next;
        if (node_next == sink)
            % Sink found
            go = 0;
        end
        n = n + 1;
    end

    if (bc == N_runs)
        fig_route = 1;
    else
        fig_route = 0;
    end

    %Calc route stats
    res_route(bc,:) = nt_fnc_show_route(nw, nodes, dst, x_size,
        fig_route);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Extract stats

    % Remove the nodes closest to the sink (returned negative
        values)
    indxs = find(res > 0);
    res_mod = res(indxs);

    % Extract the mean and variance of the results
    res_avg = mean(res_mod);
    res_var = var(res_mod);
    res_min = min(res_mod);
    res_max = max(res_mod);

    n_bins = 50;
    [res_bin, res_ind] = hist(res_mod,n_bins);
%     res_nrm = sum(res_bin);
    [res_bin_max, res_bin_max_ind] = max(res_bin);
    res_bin_norm = res_bin./max(res_bin);
    res_peak = res_ind(res_bin_max_ind);

    % Extract stats from routes
    % [N_hop, h_min, h_max, h_avg, d_shortest, d_route_tot]
    avg_N_hop = mean(res_route(:,1));
    d_hop_min = min(res_route(:,2));
    d_hop_max = max(res_route(:,3));
    d_hop_avg = mean(res_route(:,4));
    d_straight_avg = mean(res_route(:,5));
    d_route_avg = mean(res_route(:,6));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Show results

fig_hist    = 2;
fig_pdf_rl  = 3;
fig_cdf_rl  = 4;

% Some key statistics
disp('Network statistics:')
disp(['   Size of area___________________: ' num2str(x_size) '
    x ' num2str(x_size)])
disp(['   Number of nodes________________: ' num2str(N_nd)])
disp(['   Nodes per square meter_________: ' num2str(N_nd/
    x_size^2) ])

disp(' ')
disp(['For every node, found the mean of distance to ' num2str(
    N_cl)]);
disp(['closest nodes that are closer to sink than node itself:
    '])
disp(['   Stats from this:'])
disp(['      Mean (of means)_____________: ' num2str(res_avg)
    ])
disp(['      Variance (of means)_________: ' num2str(res_var)
    ])
```

```
disp(['      Smallest ___________________: ' num2str(res_min) ])
disp(['      Largest ____________________: ' num2str(res_max) ])
disp(['      Peak of histogram __________: ' num2str(res_peak) ])

disp(' ')
disp('Route statistics:')
disp(['   # hops (using shortest hop)____: ' num2str(avg_N_hop) ])
disp(['   Shortest hop___________________: ' num2str(d_hop_min) ])
disp(['   Longest hop____________________: ' num2str(d_hop_max) ])
disp(['   Average hop____________________: ' num2str(d_hop_avg) ])
disp(['   Dst from source to sink_______ : ' num2str(d_straight_avg) ])
disp(['   Total route length_____________: ' num2str(d_route_avg) ])


% Present plots

% Set limits for PDF/CDF x-axis
l_max = res_avg + res_var;
l_stp = l_max/n_bins;
lims = 0:1/100:l_max;
% Rayleigh PDF and CDF

pdf_rayl_pk = raylpdf(lims,res_peak);
pdf_chi2_pk = chi2pdf(lims,res_peak);
pdf_rayl_mn = raylpdf(lims,res_avg);
pdf_chi2_mn = chi2pdf(lims,res_avg);

cdf_rayl_pk = raylcdf(lims,res_peak);
cdf_rayl_mn = raylcdf(lims,res_avg);
cdf_chi2_pk = chi2cdf(lims,res_peak);
cdf_chi2_mn = chi2cdf(lims,res_avg);




fig = figure(fig_hist);
set(fig,'Name','Statistics of distance to M closest nodes, towards sink');
hold on
norm_fac = max(pdf_chi2_mn);
bar(res_ind, res_bin_norm*norm_fac)
% plot(lims,pdf_rayl_pk,'r');
% plot(lims,pdf_chi2_pk,'g');
% plot(lims,pdf_rayl_mn/sum(pdf_rayl_mn),'g');
% plot(lims,pdf_chi2_mn/sum(pdf_chi2_mn),'r');
% plot(lims,pdf_rayl_mn,'g');
plot(lims,pdf_chi2_mn,'r');
title('Chi−square, mean')

figure
hold on
norm_fac = max(pdf_rayl_mn);
bar(res_ind, res_bin_norm*norm_fac)
plot(lims,pdf_rayl_mn,'g');
title('rayleigh, mean')

figure
hold on
norm_fac = max(pdf_chi2_pk);
bar(res_ind, res_bin_norm*norm_fac)
plot(lims,pdf_chi2_pk,'g');
title('Chi−square, peak')

figure
hold on
norm_fac = max(pdf_rayl_pk);
bar(res_ind, res_bin_norm*norm_fac)
plot(lims,pdf_rayl_pk,'g');
title('rayleigh, peak')

% xlabel('Hop length [m]')
% ylabel('Relative Occurences/Probability')
% legend('Histogram','Rayleigh','Chi−squared');

% fig = figure(fig_cdf_rl);
% set(fig,'Name','CDF');
% hold on
% % plot(lims,cdf_rayl_pk,'r');
% % plot(lims,cdf_chi2_pk,'g');
% plot(lims,cdf_rayl_mn,'g');
% plot(lims,cdf_chi2_mn,'r');
% xlabel('Hop length [m]')
% ylabel('Probability')
% legend('Rayleigh','Chi−square');
```

nt_fnc_create_nw.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     function:   nt_fnc_create_nw
%
%     Creates a random, 2−dimensional network of desired size
    and density.
%
%     Input:
%     x: Network size, as side in square
%     N: Number of nodes

```

```
%
%     Output:
%     coord: x and y coordinates of N nodes, dimension 2xL
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function coord = nt_fnc_create_nw(x,N)

coord = rand(2,N)*x - x/2;

% With sink node in (0,0) and numbered

% coord = zeros(2,N);
% coord(1,:) = 1:N;
% coord(2:3,:) = rand(2,N)*x - x/2;

return
```

nt_fnc_find_M_closest.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     function:   nt_fnc_find_M_closest
%
%     Finds the average of the M closest nodes that are closer
    to another,
%     given node s than node n itself
%
%     Input:
%     nw: Coordingates of all nodes in network
%     n: Index of node in question
%     s: Index of sink node
%     M: number of nodes to be found
%
%     Output:
%     M_mean: The mean of the distances from n to the M closes
    nodes, in
%     the direction of s.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function M_mean = nt_fnc_find_M_closest(nw, n, s, M)

N = length(nw);
succ = 0;

% Large initial value, for comparison
% Store the M closest nodes, in direction of center
init_max = 10*max(nw(1,:));
dst_M_closest = 1000;

% The node in question, origin
node = nw(:,n);
% Center node, sink
sink = nw(:,s);
```

```

% Distance from origin to sink
d_o_s = nt_fnc_node_dist(node,sink);

for m=1:N
%     Distance from origin to node m
    d_o_i = nt_fnc_node_dist(node,nw(:,m));
%     Distance from other node to sink
    d_i_s = nt_fnc_node_dist(nw(:,m),sink);

%     Find the M closest nodes
%     check that considered node is not sink, or closer
%     to the sink than the other
    if ((d_i_s < d_o_s) && (m ≠ s))

%         Check if node m is among the M closest nodes by by
    comparing
%         with the one furthest away (dst_shortest is sorted)
    and that node
%         is closer to considered than to sink
        if((d_o_i < dst_M_closest(end)) && (d_o_s > d_o_i))
%             If so, add to list of shortest
            dst_M_closest(end) = d_o_i;
%             Sort the list
            dst_M_closest = sort(dst_M_closest);
            succ = succ + 1;
        end
    end

end

% If no closer nodes found, the node is one if the M+1 nodes
    closest
% to the sink. Disregard this result and return negative values
    . Otherwise,
% return the min, max and avg of the list
if (succ < M)
    M_mean = −1;
else
    M_mean = mean(dst_M_closest);
end

return
```

nt_fnc_next_node.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     function:   nt_fnc_next_node
%
%     Finds the next, closest node in direction of given sink.
    Part of
%     shortest−hop routing algorithm.
```

```
%
%     Input:
%     nw: Coordingates of all nodes in network
%     node_ind: Index of node in question
%     sink_ind: Index of sink node
%
%     Output:
%     dst: Distance to the next node
%     ind: Index of next node
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [dst, ind] = nt_fnc_next_node(nw,node_ind, sink_ind)

    N = length(nw);
    dst = 1000;
    ind = 0;
    node = nw(:,node_ind);
    sink = nw(:,sink_ind);

    %find dst to sink
    d_o_s = nt_fnc_node_dist(node,sink);
    for m=1:N %run through all other nodes

        %find dst between other node and sink
        d_i_s = nt_fnc_node_dist(nw(:,m),sink);
        %if node is closer to sink
        if (d_i_s < d_o_s)
            %find dist between node and other node
            d_o_i = nt_fnc_node_dist(node,nw(:,m));
            %check if closer then best, and not self
            if (d_o_i < dst  && d_o_i > 0)
                %Distance between node and next node
                dst = d_o_i;
                ind = m;
            end
        end
    end

%if no closer node found, return distance to origin, and 0 as
    next
if (ind == 0)
   dst = d_o_s;
   ind = sink_ind;
end

return
```

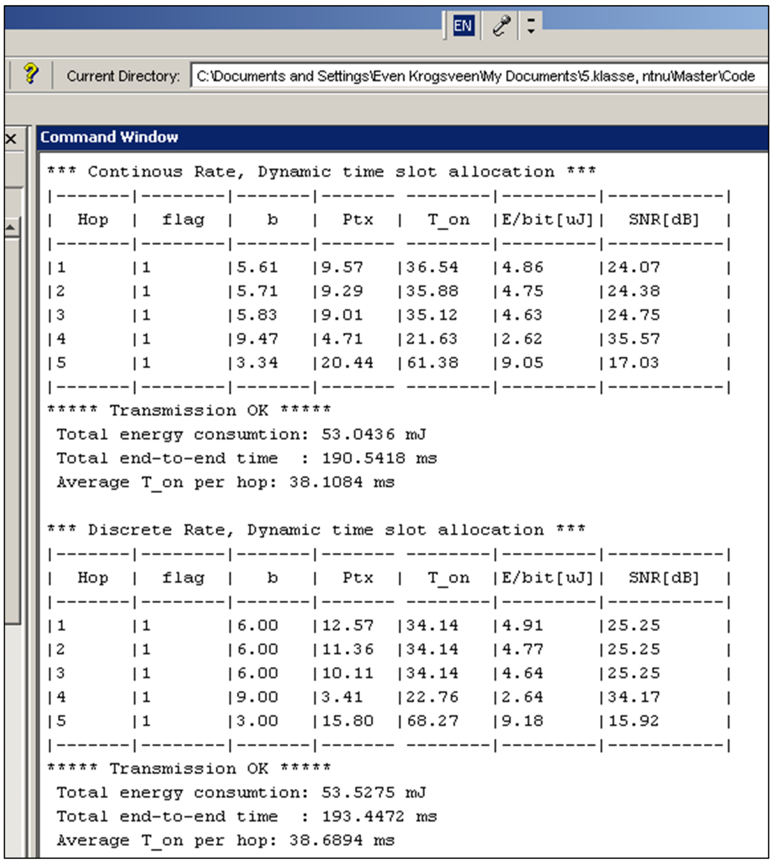
nt_fnc_show_route.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     function:   nt_fnc_show_route
%
```

```
%     Calculates Route statistics. Visualizes route if desired.
%
%     Input:
%     nw: Coordingates of all nodes in network
%     nodes: Index of nodes in route
%     x: Side of square network area
%     fig_n: Boolean value. Show route in figure if 1.
%
%     Output:
%     res: Route stats, consists of
%         N_hop: Number of hops from source to sink
%         h_min: Shortest hop
%         h_max: Longest hop
%         h_avg: Average hop
%         d_shortest: Euxledian distance between sink and
    source
%         d_route_tot: Total route length
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function res = nt_fnc_show_route(nw, nodes, dst, x, fig_n)
% [N_hop, h_min, h_max, h_avg, d_shortest, d_route_tot]

N_hop = length(nodes)−1;

if (fig_n ≠ 0)

    fig_1 = figure(fig_n);
    set(fig_1,'Name','Network sketch');
    scatter(nw(1,:)',nw(2,:)',40,'filled');
    axis([−x/2 x/2 −x/2 x/2])
    grid on
    xlabel('x−coordinate [m]')
    ylabel('y−coordinate [m]')
    hold on
    scatter(nw(1,nodes(1)),nw(2,nodes(1)),40,'g','filled');
    scatter(nw(1,nodes(end)),nw(2,nodes(end)),40,'r','filled');

end

for (m=1:N_hop)

    n1 = nw(:,nodes(m));
    n2 = nw(:,nodes(m+1));

    if (n1(1) > n2(1))
        x_neg = 1;
    end

    if (n1(2) > n2(2))
        y_neg = 1;
    end

    x_diff = n2(1)−n1(1);
    y_diff = n2(2)−n1(2);

    x_step = x_diff/2;
    y_step = y_diff/2;

    if (fig_n ≠ 0)
        plot(n1(1):x_step:n2(1),n1(2):y_step:n2(2),'r')
    end

end

d_min = min(dst);
d_max   = max(dst);
d_avg   = mean(dst);

% Shortest discance between source and sink
dst_shortest  = nt_fnc_node_dist(nw(:,nodes(1)),nw(:,nodes(end)
    ));
% Total route length
dst_route = sum(dst);

% [N_hop, h_min, h_max, h_avg, d_shortest, d_route_tot]
res = [N_hop d_min d_max d_avg dst_shortest dst_route];

return
```

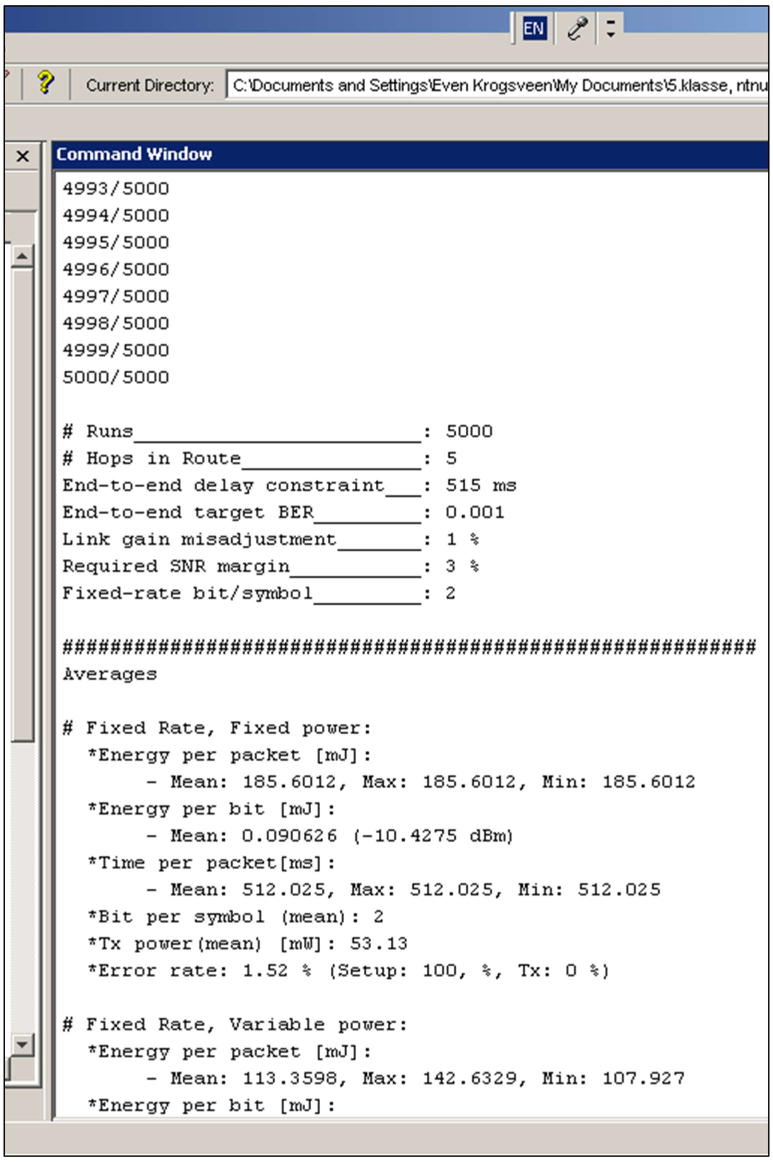
Appendix C

Screenshots


EN
Current Directory: C:\Documents and Settings\Even Krogsveen\My Documents\5.klasse, ntnu\Master\Code
Command Window

```
*** Continous Rate, Dynamic time slot allocation ***
|-------|--------|-------|-------  --------|---------|-----------|
|  Hop  |  flag  |   b   |  Ptx   |  T_on  |E/bit[uJ]|  SNR[dB]  |
|-------|--------|-------|-------  --------|---------|-----------|
|1      |1       |5.61   |9.57    |36.54   |4.86     |24.07      |
|2      |1       |5.71   |9.29    |35.88   |4.75     |24.38      |
|3      |1       |5.83   |9.01    |35.12   |4.63     |24.75      |
|4      |1       |9.47   |4.71    |21.63   |2.62     |35.57      |
|5      |1       |3.34   |20.44   |61.38   |9.05     |17.03      |
|-------|--------|-------|-------  --------|---------|-----------|
***** Transmission OK *****
 Total energy consumtion: 53.0436 mJ
 Total end-to-end time  : 190.5418 ms
 Average T_on per hop: 38.1084 ms

*** Discrete Rate, Dynamic time slot allocation ***
|-------|--------|-------|-------  --------|---------|-----------|
|  Hop  |  flag  |   b   |  Ptx   |  T_on  |E/bit[uJ]|  SNR[dB]  |
|-------|--------|-------|-------  --------|---------|-----------|
|1      |1       |6.00   |12.57   |34.14   |4.91     |25.25      |
|2      |1       |6.00   |11.36   |34.14   |4.77     |25.25      |
|3      |1       |6.00   |10.11   |34.14   |4.64     |25.25      |
|4      |1       |9.00   |3.41    |22.76   |2.64     |34.17      |
|5      |1       |3.00   |15.80   |68.27   |9.18     |15.92      |
|-------|--------|-------|-------  --------|---------|-----------|
***** Transmission OK *****
 Total energy consumtion: 53.5275 mJ
 Total end-to-end time  : 193.4472 ms
 Average T_on per hop: 38.6894 ms
```

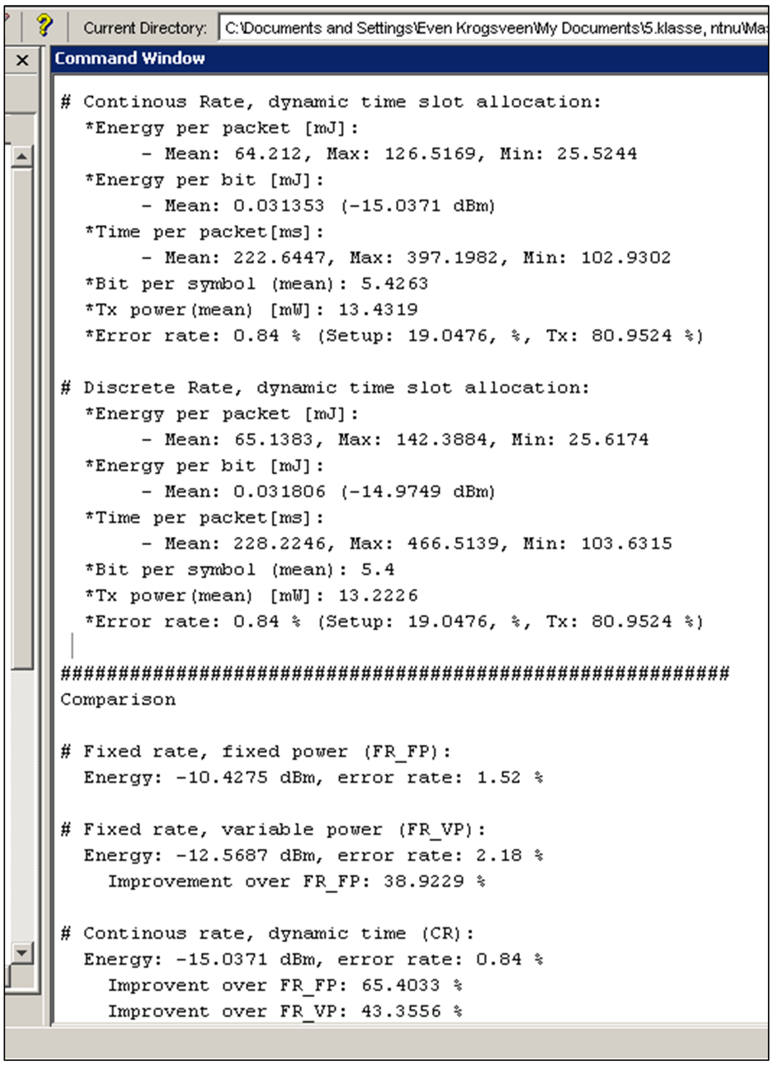
Figure C.1: Route configuration details for a single route realization.



EN
Current Directory: C:\Documents and Settings\Even Krogsveen\My Documents\5.klasse, ntnu
Command Window

```
4993/5000
4994/5000
4995/5000
4996/5000
4997/5000
4998/5000
4999/5000
5000/5000

# Runs________________________: 5000
# Hops in Route_______________: 5
End-to-end delay constraint___: 515 ms
End-to-end target BER_________: 0.001
Link gain misadjustment_______: 1 %
Required SNR margin___________: 3 %
Fixed-rate bit/symbol_________: 2

##########################################################
Averages

# Fixed Rate, Fixed power:
  *Energy per packet [mJ]:
       - Mean: 185.6012, Max: 185.6012, Min: 185.6012
  *Energy per bit [mJ]:
       - Mean: 0.090626 (-10.4275 dBm)
  *Time per packet[ms]:
       - Mean: 512.025, Max: 512.025, Min: 512.025
  *Bit per symbol (mean): 2
  *Tx power(mean) [mW]: 53.13
  *Error rate: 1.52 % (Setup: 100, %, Tx: 0 %)

# Fixed Rate, Variable power:
  *Energy per packet [mJ]:
       - Mean: 113.3598, Max: 142.6329, Min: 107.927
  *Energy per bit [mJ]:
```

Figure C.2: Simulation averages over multiple route realizations.


Current Directory:
C:\Documents and Settings\Even Krogsveen\My Documents\5.klasse, ntnu\Ma
Command Window

```
# Continous Rate, dynamic time slot allocation:
  *Energy per packet [mJ]:
       - Mean: 64.212, Max: 126.5169, Min: 25.5244
  *Energy per bit [mJ]:
       - Mean: 0.031353 (-15.0371 dBm)
  *Time per packet[ms]:
       - Mean: 222.6447, Max: 397.1982, Min: 102.9302
  *Bit per symbol (mean): 5.4263
  *Tx power(mean) [mW]: 13.4319
  *Error rate: 0.84 % (Setup: 19.0476, %, Tx: 80.9524 %)

# Discrete Rate, dynamic time slot allocation:
  *Energy per packet [mJ]:
       - Mean: 65.1383, Max: 142.3884, Min: 25.6174
  *Energy per bit [mJ]:
       - Mean: 0.031806 (-14.9749 dBm)
  *Time per packet[ms]:
       - Mean: 228.2246, Max: 466.5139, Min: 103.6315
  *Bit per symbol (mean): 5.4
  *Tx power(mean) [mW]: 13.2226
  *Error rate: 0.84 % (Setup: 19.0476, %, Tx: 80.9524 %)

###########################################################
Comparison

# Fixed rate, fixed power (FR_FP):
  Energy: -10.4275 dBm, error rate: 1.52 %

# Fixed rate, variable power (FR_VP):
  Energy: -12.5687 dBm, error rate: 2.18 %
    Improvement over FR_FP: 38.9229 %

# Continous rate, dynamic time (CR):
  Energy: -15.0371 dBm, error rate: 0.84 %
    Improvent over FR_FP: 65.4033 %
    Improvent over FR_VP: 43.3556 %
```

Figure C.3: Comparison of key results for the different optimization schemes.