Interference-Based Scheduling in Spatial Reuse
TDMA

JIMMI GRÖNKVIST

Doctoral Thesis
Stockholm, Sweden 2005
Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av teknologidoktorsexamen fredagen den 14 oktober 2005 klockan 14.00 i sal C1, Electrum, Kungliga Tekniska Högskolan, Isafjordsgatan 22, Kista.

© Jimmi Grönkvist, september 2005

Tryck: Universitetsservice US AB
Abstract

Spatial reuse TDMA has been proposed as an access scheme for multi-hop radio networks where real-time service guarantees are important. The idea is to allow several radio terminals to use the same time slot when possible. A time slot can be shared when the radio units are geographically separated such that small interference is obtained. The transmission rights of the different users are described with a schedule.

In this thesis we will study various aspects of STDMA scheduling. A common thread in these various aspects is the use of an interference-based network model, as opposed to a traditional graph-based network model. While an interference-based network model is more complex than a graph-based model, it is also much more realistic in describing the wireless medium.

An important contribution of this thesis is a comparison of network models where we show that the limited information of a graph model leads to significant loss of throughput as compared to an interference-based model, when performing STDMA scheduling.

The first part of this thesis is a study of assignment strategies for centralized scheduling. Traditionally, transmission rights have been given to nodes or to links, i.e., transmitter/receiver pairs. We compare these two approaches and show that both have undesirable properties in certain cases. Furthermore, we propose a novel assignment strategy, achieving the advantages of both methods.

Next we investigate the effect of a limited frame length on STDMA schedules. We first show that the required frame length is larger for link assignment than for node assignment. Further, we propose a novel assignment strategy, the joint node and link assignment, that has as low frame length requirements as node assignment but with the capacity of link assignment.

In the last part of this thesis we describe a novel interference-based distributed STDMA algorithm and investigate its properties, specifically its overhead requirement. In addition we show that this algorithm can generate as good schedules as a centralized algorithm can.
Acknowledgment

I am grateful to many people for helping me carry this thesis to completion. First, I wish to thank my advisor Professor Jens Zander. His scientific guidance and support, especially in structuring my work, has been invaluable.  

Many thanks also to Anders Hansson, who has always been there to listen to all my odd ideas. Many ideas have matured during our fruitful discussions.  

I have appreciated the help of Pelle Zeijlon, both in helping me with the implementation of the distributed algorithm, but perhaps more the discussions around them. It was a great help in making the algorithm more complete.  

Thanks also to Ashay Dhamdhere whom I had the pleasure to work with during a part of this last year. Apart from that working together have been fun, our cooking and eating of Indian cuisine have also been a pleasure.  

Furthermore, I would like to thank the present and the former Head of Department, Sören Eriksson and Christian Jönsson, as well as my present and former project managers Mattias Sköld and Jan Nilsson, for giving me the opportunity to do this work.  

In addition, I am grateful to all my colleagues at the Department of Communication Systems at the Swedish Defence Research Agency (FOI) for providing a pleasant work environment. In particular, I wish to thank the network research group.  

Finally, there is Elisabeth, never forgotten.
# Contents

1 Introduction .................................................. 1
   1.1 Military communications ....................... 1
   1.2 Ad hoc Networks .................................. 3
   1.3 Medium Access Control .......................... 4
   1.4 STDMA Scheduling .................................. 5
   1.5 Research Area and Previous Work .............. 8
   1.6 Research Strategy and Contributions .......... 12
      1.6.1 Delimitations ............................... 13
   1.7 Related Work ....................................... 14
   1.8 Outline of the Thesis ............................ 15

2 Network Model .............................................. 17
   2.1 The OSI Model ..................................... 17
   2.2 Data Link layer and Physical Layer .......... 19
   2.3 Interactions with the Link Layer ............... 21
   2.4 Transport and Network Layer .................... 22
   2.5 Routing assumptions for Mobile Networks .... 25
   2.6 Traffic Estimations ............................... 25
   2.7 Performance Measures ............................ 26

3 Node and Link Assignment ................................. 29
   3.1 STDMA scheduling ................................. 29
   3.2 Link Assignment .................................... 32
   3.3 Node Assignment ..................................... 33
   3.4 Analysis ............................................. 34
      3.4.1 Maximum throughput for link assignment 35
      3.4.2 Maximum throughput in node assignment 36
Chapter 1

Introduction

1.1 Military communications

In future military operations, the armed forces must be able to operate against a variety of threats (from heavy armor to irregular forces) and in a variety of environments, including urban, mountainous and forested terrain, under jamming threats, sometimes while remaining covert. This will require a very flexible command, control and communications system that can be adapted to the prevailing situation.

The communication system will be a central part of a future network centric defence. Fast and reliable exchange of information between different parts of the military organization will be essential and international operations will be increasingly more important as will cooperation with other countries military forces.

Communications capability must be maintained over extended ranges and interoperation with strategic-, operational- and tactical-level information systems must be possible in order to support autonomous operations with highly dispersed organizational elements. The network must be scalable to permit use over small as well as large areas of operation, possibly with large number of users.

This will require a dynamic re-configurable network that can exploit all sensors and information sources available for maximum efficiency so that information can be quickly acquired and assimilated at all levels of the command’s hierarchy.

Furthermore, much of the information and its requirements must be avail-
able even at the level of the individual soldiers. The system must be able to provide its users with robust communication, situation awareness, planning, tasking and coordination, precise geolocation and navigation, and possibly other services not yet foreseen [1].

Since different parts of the network experience very different situations, these different parts must be able to autonomously adapt to the prevailing local situations and exploit these for maximum efficiency. Such situations can vary from a merely portable Headquarter LAN to the rapid changes on the battlefield itself. In all situations there may be hostile jammers present, and the loss of any unit is possible.

Even under the most severe conditions, the network must provide each soldier with the ability to transmit and receive command and control data at a minimal information rate, regardless of combat situation, position or environment. The network must also function if divided into sub-segments.

All these requirements mean that robust, secure, and efficient wireless communication networks will play a very essential part of the future military communication network.

To avoid weak points, we do not want to rely on centralized control. Furthermore, if communication links break due to mobility or hostile jamming we want the network to be able to reconfigure itself and find other routes for the traffic.

These requirements differ considerably from most civilian networks, which usually are geared to low cost with pre-installed wireless infrastructures. Civilian networks are often hierarchical in the sense that mobile units will communicate through a central, static node. The loss of this central node leads to network failure of all units in the surrounding area.

All services the military communication network will provide must be simultaneously handled by the system, each with different service demands. For example, voice transmissions put high demands on the delay. The human ear is especially sensitive to long delays, which can come from large delay variations (buffered in the end), large delay mean values or a combination of both. On the other hand, a rather high bit error rate can probably be accepted. Other information updates may have high demands on the avoidance of bit errors, while delay may be of lesser importance.

The upholding of these service requirements is usually denoted as Quality-of-Service (QoS) guarantees. QoS can be seen as a performance contract between the network and the application. In a mobile radio network no absolute guarantees can be given since there is always the possibility that the network
1.2. Ad hoc Networks

An ad hoc network consists of (mobile) radio units (nodes) that are spread out in some (possibly unknown) terrain without any form of pre-planning or fixed infra-structure. Every node is both transmitter and receiver and can also function as a relay node for nodes further away (multi-hop functionality), outside direct radio transmission range, i.e. nodes close to each other communicate directly while nodes further away use intermediate nodes as relay nodes.

Ad hoc networks can be robust and flexible since the loss of one of the relaying nodes can be handled by finding other intermediate nodes that can be relays, no node are essential for the functionality of the network. Furthermore, by relaying traffic we do not need line-of-sight communications between the communicating nodes, thereby decreasing necessary transmission power.

Ad hoc networks have been suggested for use in several situations where the wired communication infrastructure is not sufficient. Beside the obvious — military communications, it can also be used in emergency situations, e.g. after a earthquake where parts of the wired infrastructure have been destroyed or is lacking power supply. Another is to complement cellular systems in order
to extend the range of base stations. In this case we need fewer base stations resulting in a cheaper fixed infra-structure.

There has been research on ad hoc networks since the 70’s, then under the name packet radio networks, but in recent years there have been a considerable increase in interest for ad hoc networks. Despite this, there is still problems that need to be solved before they can be efficiently used in our military scenarios.

One problem is caused by the multi-hop functionality itself. The relaying of traffic can enable communication between units further away than what would otherwise be possible, but it also introduces the problem of finding the path from source to destination. This problem is generally referred to as routing and it has gotten a lot of attention from researchers. The problem is that ad hoc networks usually changes much faster than other networks and new paths must constantly be found without too large overhead. See for example [3] for an overview of different routing methods in ad hoc networks.

Another important design issue is Medium Access Control (MAC), i.e. how to avoid or resolve conflicts due to simultaneously transmitting radio units. Less research then for routing have been done here and this will be our main area.

Before we discuss MAC in more detail we can also mention some other issues of importance for the performance of ad hoc networks. One of them is the previously mentioned QoS. The rapid changes in an ad hoc network makes reliable guarantees, delays or other, difficult. It is probably necessary to solve this problem on all levels of the communication system. We can for example demand that both routing and MAC attempt to give guarantees. For routing this means finding appropriate ways (in terms of what should be guaranteed) and for MAC it may for example mean reservations or prioritizing important data.

Another issue that also is important to solve is security. Ad hoc networks are supposed to function autonomously without any centralized unit, it is therefore difficult to use methods that exists for fixed networks. However, this is also an issue we will not study further in this work.

1.3 Medium Access Control

Traditionally, MAC protocols for ad hoc networks are based on contention-based access methods, i.e. a user attempts to access the channel only when it actually has packets to send. The user has no specific reservation of a channel and only tries to contend for or reserve the channel when it has packets to transmit. This has clear advantages when the traffic is unpredictable. More
specifically, the most frequently used protocols are based on carrier sense multiple access (CSMA) [4], i.e. each user monitors the channel to see if it is used, and only if it is not will the user transmit. However, this is done in the transmitter while collisions appear in the receiver. This can lead to the so-called hidden terminal problem. A way around this is to first transmit a short request-to-send (RTS) and then only send the message if a clear-to-send (CTS) is received. This is the general principle of the IEEE 802.11 standard [5], which at present is the most investigated and used MAC protocol. However, several RTS can be lost in a row which makes delay guarantees difficult.

Efforts have been made to guarantee QoS in CSMA-based medium MAC, see e.g. [6], but contention-based medium access methods are inherently inappropriate for providing QoS guarantees.

One of the most important QoS parameters in many applications that are specifically sensitive to the MAC is the delay guarantees previously mentioned. One approach where delay bounds can be guaranteed is time division multiple access (TDMA), i.e. the time is divided into time slots and each user receives its own time slot.

Unfortunately, in sparsely connected networks this is usually inefficient. But, due to the multihop properties, the time slots can often be shared by more than one user without conflicts. This will automatically be the case with dynamic MAC protocols like CSMA, since a user’s access to the channel only will affect a local area.

However, to achieve both high capacity and delay guarantees one can use spatial reuse TDMA (STDMA) [7], which is an extension of TDMA where the capacity is increased by spatial reuse of the time slots, i.e., a time slot can be shared by radio units geographically separated so that small interference is obtained.

1.4 STDMA Scheduling

The problem is to design STDMA schedules that fulfill required properties, e.g. minimizing delay or being able to update the schedules in a distributed fashion. An STDMA schedule describes the transmission rights for each time slot.

Before we go into more detail we will show a small example of how we can go from a scenario with tanks to an STDMA schedule. In the top part of Figure 1.1 we see a group of nine tanks heading to the right.

Now, to represent this as a radio network each of these tanks will be shown
as a node in the middle picture (circles), giving us a 9-node network. Furthermore, the ability to directly communicate between two of the tanks is represented by an line. Such lines are called links, and the lack of a link represents that the two units cannot communicate directly, but must instead relay their message by intermediate nodes. Direct communication is usually possible if the units are close to each other, but objects like a hill can prevent direct communication even when units are close, see for example nodes 6 and 8 in the figure.

Furthermore, we can see that communication between nodes 1 and 9 must be relayed by nodes 2 and 3. An STDMA schedule could assign link (1, 2), (9, 3), and (6, 7) to transmit simultaneously, since they are sufficiently far from each other. Another set could be (4, 1), (3, 5) and (8, 7), but not (1, 2), (3, 5), and (8, 7) because, at least in this example, we are using omni-directional antennas,
and the transmission of node 3 would interfere with the reception of node 2.

In the bottom part of the figure we show a possible schedule where each link receives one time slot each, as can be seen, at least ten time slots will be needed since the ten directed links between the four nodes in the center cannot share time slots.

Unfortunately, the nodes will be moving, and nodes that can transmit simultaneously without conflict at one moment will probably not be able to do so later. Returning to the above example we can see that if tank 3 moves slower than the three tanks to the left, the transmission of node 1 will eventually create sufficient interference when node 3 receives on link 2 so that links 1, 2, and 3 cannot share the same time slot anymore.

Therefore, the STDMA schedule must be updated whenever something changes in the network. This can be done in a centralized manner, i.e. all information is collected into a central node, which calculates a new schedule. This schedule is then propagated throughout the network. The schedules designed this way can be very efficient because the central node has all information about the network.

However, for a fast-moving network this is usually not possible. By the time the new schedule has been propagated it is already obsolete, due to node movements. Furthermore, it is not a robust solution, as the loss of the central node can be devastating for network communications.

Another way to create STDMA schedules is to do it in a distributed manner, i.e. when something changes in the network, only the nodes in the local neighborhood of the change will act on it and update their schedules without the need to collect information into a central unit. In this thesis we will look at both methods.

In the above example, all transmission rights are assigned to the links, i.e. both transmitting and receiving nodes are determined in advance when the schedule is created. This is called link assignment or link activation. An alternative would be to assign transmission rights to the nodes instead. In this case only the node is scheduled to transmit in the time slot. Any of its neighbors, or all, can be chosen to be the receiving node. This is called node assignment or node activation.

Generally, node assignment is used for broadcast traffic, and link assignment is used for unicast traffic. However, in the multipurpose networks of the future, the network must be able to handle both these types of traffic simultaneously, which means that it would be preferable if one assignment strategy can be used for all traffic types.
1.5 Research Area and Previous Work

In this section we will go further into the problems that have been studied for STDMA and from this motivate the specific problems we have chosen to investigate in this thesis. In the next section we will describe our research problems in greater detail.

The problems surrounding the design of STDMA schedules are well addressed in the literature, but areas remain that need more work.

Networks Models

In order to determine which nodes or links that can transmit at the same time when we do the scheduling, we need a description of the network. We usually denote such a description with a network model. The network models used by STDMA algorithms have varied in complexity.

Most algorithms assume that transmission ranges are limited (usually circular), and beyond this no interference is caused. This allows the problem to be transformed into a graph-theoretical problem, i.e. the network is represented as a directed graph as we did in the previous section. In this graph, an edge between two nodes indicates that they can communicate with each other directly, and the lack of an edge indicates that they cannot affect each other even as interferences. This is commonly referred to as the Protocol Interference Model, see for example [7, 8, 9].

Using this model, scheduling can be transformed into the coloring of the nodes or links in a graph, which can be solved with the help of graph theory. This model thereby makes it simple to create STDMA algorithms. In addition, the maximum distance nodes can effect each other is two hops, which makes it easy to design distributed algorithms. However, a disadvantage is that this model does not describe the wireless medium very well as it does not take into consideration capture or that the combination of interference from several nodes can cause transmissions to fail.

A more complex network model uses a two-level graph model, see e.g. [10]. Here interference edges are added, meaning that there is not sufficient signal power to receive the packet without error, but it is strong enough to interfere with reception from other users. This model gives a better description of the network and is still useful for a mobile scenario where the schedule must be updated often.

A more realistic, although even more complex, model is the use of the
signal-to-interference ratio, *Interference-Based Scheduling*. This is also known as the Physical Interference Model. In this case, a node is assumed to be able to receive a packet without error if the received signal strength is sufficient compared with the noise and all interfering signals (from simultaneously transmitting nodes in the network). The first use of this model for STDMA scheduling that we have found is [11].

The use of the Physical Interference Model is the main focus of this thesis. One reason to this is that it gives a so much better description of the network than a graph model. We will generally use interference-based scheduling for all scheduling in this thesis.

The only exception to this is a comparison of network models. Because so many algorithms have assumed the use of the Protocol Interference Model, the comparison of the efficiency of these models is one of the research areas in this thesis. In chapter 5, we will show that graph scheduling can cause severe problems.

It should be noted, however, that there have also been other approaches to using graph-based scheduling. In [12] a truncated graph model is used that gives probabilistic guarantees for the throughput by bounding the maximum number of simultaneous transmitting units.

**Assignment Strategies and Minimizing Frame Length**

Since both broadcast traffic and unicast traffic have been considered important in ad hoc networks, most work on STDMA has generally assumed the two types of assignment strategies previously mentioned, i.e. node assignment and link assignment.

Node assignment have also been called node activation or broadcast scheduling. Examples of algorithms that generate node assignment schedules can be found in [8, 13, 14, 15, 16]. Link assignment is often referred to as link activation, and examples of such algorithms can be found in [17, 18, 19, 20].

In practice, both assignment strategies are similar so it is usually simple to design algorithms for both strategies if an algorithm is designed for one of them, which has, for instance, been done in [21, 22]. Another example is [23], in which algorithms for both link and node scheduling are described that focus on generation on short schedules. In [24] a more general description of the assignment problem is presented. The different assignment methods are seen as constraints in a unified algorithm for the assignment problem given in the paper.

The most common research problem in respect of STDMA has probably
been finding algorithms that generate schedules that give each node or link a time slot with as short a schedule as possible. For arbitrary graphs, this has been shown to be an NP-complete problem [25, 26] for both link and node assignment. To overcome this the radio networks can instead be modeled as restricted graphs, for example trees, planar graphs or close to planar graphs for which it is easier to find solutions, see [27] for an overview.

However, although node scheduling has generally been assumed to be used for multicast traffic and link scheduling has been assumed to be used for unicast traffic, this does not provide the full picture. Little work exists that studies which of these assignment strategies is preferable in different scenarios with different types of traffic. In [28], the different properties of node and link assignment have been studied in respect of schedules in which nodes or links are assigned one slot each. In this they find that node assignment is preferable for all traffic loads.

**Traffic Sensitivity**

However, giving each node or link a single time slot is not necessarily good. There is considerable variation of traffic over the different links of the network due to the relaying of traffic in multi-hop networks. An STDMA algorithm must adapt to this and give some nodes more capacity (time slots) in order to be efficient. Algorithms that do this are usually denoted as *traffic sensitive* or traffic controlled. The ability to give some nodes or links extra time slots related to the traffic loads on the links has been considered in several papers, see e.g. [17, 29].

Furthermore, studies on traffic sensitivity have also shown that it improves capacity considerably compared with giving each node or link a single time slot [11, 30].

The results in [28] are therefore somewhat limited because traffic sensitivity is less of a problem for node assignment as there is less variation of the traffic over the nodes than the links. A comparison under the assumption of traffic sensitivity will be studied in this thesis. We will show that this gives a different result, namely that link assignment can achieve higher throughput than node assignment. Furthermore, we have chosen to use traffic sensitivity for all scheduling in the thesis.

As we will see, neither node assignment or link assignment will be sufficient in all situations so an important part of the thesis will focus on the suggestion of new assignment strategies with better performance.
1.5. Research Area and Previous Work

Centralized and Distributed Algorithms

We have left one of the most important problem areas regarding STDMA until now. The issues concerns how to design algorithms that can generate schedules in a distributed manner. There are still many reasons why centralized scheduling can be useful. For example, for static networks, a schedule can be generated centrally and be distributed to the nodes in the network. However, the main reason we study centralized algorithms in this thesis is that it gives us an upper limit on performance and provides some ideas about which properties are important when we do the distributed scheduling. Many such centralized algorithms have been suggested, see for example [14, 19, 31].

However, in most scenarios where STDMA is envisioned, for example in this thesis, mobility is assumed. Being able to update the schedule in a distributed manner is often considered vital. Many such algorithms have been suggested, see for example [17, 32, 18, 15, 21, 33, 34], but they all assume a graph model to describe the network. As previously mentioned, according to such a model, nodes will only affect each other at a maximum of two hops, which makes the distribution of information much simpler than for an interference-based model that may let nodes at much further distance affect each other, depending on the terrain. Few distributed algorithms have been implemented into functional systems, but one, USAP [21], is used for to generate multi-channel STDMA schedules in the soldier phone radio [35] that is designed as an ad hoc radio for military use in mobile environments.

No fully distributed STDMA algorithm that can generate schedules based on the physical Interference Model exists, and the first step towards such a algorithms will be an important part of this thesis.

In addition we can return to the previous problem regarding traffic sensitivity, but now specifically for distributed algorithms.

For distributed scheduling, traffic control is only rarely included. Although many algorithms include the ability to assign more than one time slot to a link or node, cf. [36, 37], a more specific description of how and when some of the links receive extra time slots is usually omitted.

One exception is [38] which attempts to generate fair time slot allocations for link assignment given traffic demands on each link. Another is [39], where each link can request a bandwidth, although the actual bandwidth it receives is proportional to its request compared with the other links’ requests.

Another solution is described in [18], where virtual circuits are assigned time slots (or rather each link along the virtual circuit). This gives traffic sensi-
tivity because a link can carry more than one virtual circuit.

1.6 Research Strategy and Contributions

We will study the behavior of STDMA in two different types of mobility scenarios. First, centralized algorithms in static networks (in terms of both node mobility and traffic). We use centralized scheduling to give us an upper limit on performance, and to provide some ideas about which properties that are important when we do the scheduling.

Second, we will study interference-based distributed scheduling in mobile scenarios. We will also describe which properties a distributed STDMA algorithm should have in order to be efficient. No existing STDMA algorithm can fulfill all these properties, although USAP [36] fulfills several of them. Investigations of distributed STDMA will give us a better picture of how well STDMA performs.

The present thesis consists of the following studies:

1. We compare the two most common assignment methods used today, i.e. node and link assignment. This comparison is performed both in an analytical manner and via simulations, in order to determine when node or link assignment should be used. We will show that link assignment behaves better for high traffic loads, achieving a higher throughput for unicast traffic. However, this comes at a cost of higher delay than node assignment for low traffic loads. Our results also indicate that only the size and the connectivity of the network are necessary to determine when each of these methods should be used.

   This has been published in [40, 41].

2. We suggest a novel assignment strategy that achieves the advantages of both link assignment and node assignment. Our proposed strategy is based on a link schedule, but in which transmission rights are extended. This strategy is evaluated in comparison with the other two methods by using approximations and simulations.

   This has been published in [42, 43, 41].

3. We investigate the loss of efficiency (in terms of throughput) when the STDMA algorithm only has knowledge about the two-level graph model of the network compared with having full knowledge of the attenuation
between all pairs of nodes. We show that the traditional graph-based scheduling can lead to significant loss of throughput.

A somewhat different approach to this has been published in [44].

4. We investigate the effect of a limited frame length. We will show that the required frame length is larger for link assignment than for node assignment, but we also suggest a novel assignment strategy—joint node and link assignment—that has as low frame length requirements as node assignment but with the capacity of link assignment.

This is a joint study in collaboration with Ashay Dhamdhere.

A version of this is submitted to [45].

5. We describe a novel interference-based distributed STDMA algorithm that can give results as high as what a centralized algorithm can. This is done for an investigation on how to efficiently handle (use) distributed information. This is mainly published in [46] and [47].

6. We show how to reduce the overhead requirement of the above described algorithm by choosing good parameter settings. With overhead requirement we mean how much control information is required to convey the desired network information. This can be done to indicate how much network information should be transferred for different networks.

This is not yet published.

1.6.1 Delimitations

We will not study all extra features that can be added in order to further improve STDMA. Examples of such features are power control, rate control, adaptive antennas, and similar methods. We have enough parameters to handle without these, but they can further improve STDMA. One goal of this work is to simplify their inclusion later.

However, rate control will partly be included for distributed scheduling since because some form of rate control is probably necessary for interference-based scheduling in mobile networks.

Other practical issues concerning how to make an STDMA scheme work will not be studied. These include slot synchronization, exact functionality of data link layer, effects of Tx/Rx turnaround time, hostile jammers, and similar effects which do affect the performance of the network.
1.7 Related Work

There are also other areas of research for STDMA that are of interest. These will be out of scope for this thesis, but much of the research on STDMA that is being performed today is done in the following areas.

**Optimal Scheduling**

So far most STDMA algorithms (specifically if they are distributed) have been designed to give an acceptable solution, rather than a solution that handles the channel as efficiently as possible under different situations. However, in [19, 48] the optimal scheduling problem was formulated using graph-based scheduling. However, both of these assumed spread-spectrum signal modulation so only avoidance of nodes or links transmitting and receiving at the same time was necessary.

Only much more recently has more work been done on this, with most of the studies using interference-based scheduling. In [49, 50] capacity regions for ad hoc networks are studied. One method examined is a time slotted MAC system, i.e. what we call STDMA. In addition to this, several papers on optimal scheduling, with different assumptions have been published [51, 52, 53, 54, 55, 56].

**Cross-Layer Issues**

Several of the optimization papers also consider cross-layer issues as well, in respect of both higher and lower layers. Joint scheduling with power control and/or routing are some examples. As these issues are not the main focus of the present thesis we will not provide more details about how these papers differ.

Cross-layer issues have not only been studied for optimal scheduling. Power control have been studied in several cases. See [57] as one example. In [58] variable-rate is also added. Distributed power control is used in [59, 60] (the first one without scheduling—however) and with routing in [61].

Adaptive and directional antennas have been studied for centralized scheduling in some cases and have been shown to give considerable improvements [62, 63]. Some results for distributed scheduling also exist [64].

So far optimization methods are mainly useful as reference methods, because they require all information about the network and it takes a long time to calculate the schedules, especially for large networks. In addition, the best
possible schedule in terms of throughput can be very long in order to handle traffic sensitivity. Finding the best schedule with a given frame length is still a very difficult.

1.8 Outline of the Thesis

In Chapter 2 we describe the network model we have used. We also describe the layers, according to the OSI model, that are of interest to us, i.e. data link layer, network layer and transport layer. The data link layer describes the functionality of the links and when a link can be used without conflicts.

The transport layer basically gives us a model of the external traffic of the network, whereas the network layer includes the routing of this traffic. The main purpose of this is to calculate the traffic on the links and nodes in the network. We also define the evaluation parameters, which are the average end-to-end packet delay and the maximum throughput.

The rest of the thesis can be divided into two parts: centralized scheduling and distributed scheduling. In Chapters 3 to 6 we will deal with centralized scheduling for static networks. In Chapters 7 and 8 we will study distributed scheduling for mobile networks.

In Chapter 3 we define and exemplify node assignment and link assignment. We also provide approximations for the maximum throughput and the average packet delay for these assignment methods. These are then used to compare the efficiency of the algorithms. We conclude the chapter with simulations of delay and throughput to determine how well the approximations work.

In Chapter 4 we describe a novel assignment method LET. Here, too, we give an approximate formula for the delay and use this in comparison with simulations. These results are then compared to node and link assignment, showing the advantages of LET.

In Chapter 5, we make a comparison between using the traditional graph model when designing an STDMA schedule and using an interference-based model.

In Chapter 6 we study the effect of different frame lengths and introduce a new scheduling strategy, joint node and link assignment, that performs well under all frame lengths.

Chapter 7 starts the study on distributed scheduling by giving a list of properties. Furthermore, we describe the distributed algorithm that we use. In chapter 8 we continue this by studying the overhead traffic required when using this
algorithm for a mobile network.

Finally, in Chapter 9 we conclude the thesis.

In appendix A we present more information about how simulations are performed.
Chapter 2

Network Model

This chapter introduces the network model we use and the assumptions required. These can be divided into two parts: first, the assumptions on the data link layer and then the assumptions from the network and transport layers. The data link layer is described in the first section. The assumptions on the higher layers describe how the traffic is generated and routed.

Furthermore, we also describe how the performance will be evaluated.

2.1 The OSI Model

To reduce design complexity, most networks are organized as stacks of layers, each one on top of the one below it. Each layer offers a set of services to the layer on top of it, shielding the above layers from details of the implementation of the layer below. The number of layers and content may vary from network to network, but layer \( l \) on a node can be seen as it carries out a conversation with layer \( l \) on another node. The rules of this conversation are called the protocol of layer \( l \). A collection of layers and protocols is called a network architecture.

In this thesis we will use the Open System Interconnection (OSI) reference model as a basic description of the network architecture, it is not a complete architecture, however, because the protocols of each layer are not specified. More about network layering and the OSI reference model can be found in [65]. The OSI model has seven layers dealing with different issues, see Figure 2.1. The different layers in ascending order are,

- **Physical layer** - Deals with transmission of raw bits over the channel.


Data Link Layer - Creates a virtual link for reliable transmission between two nodes, by, for example, adding error correction to the raw bits. In addition, it handles MAC to avoid simultaneous transmissions on the media.

Network Layer - Controls the subnetwork, dealing in particular with routing.

Transport Layer - Creates a virtual end-to-end link

Session Layer - Creates sessions between users. This can allow users to restart after a crash.

Presentation Layer - Deals with the transmitted information, making it possible for computers with different data representations to communicate.

Application Layer - These are the user applications, such as HTTP for web browsing, SMTP for electronic mails, or FTP for file transfers, see [65] for more information.
2.2. Data Link layer and Physical Layer

As can be seen, the four highest layers deals with end-to-end communication that is only be needed in the end nodes, whereas the three lowest layers are needed in all intermediate nodes. In an ad hoc node all layers are necessary.

Since an STDMA schedule controls when a link should be used, it thereby has protocols in the network layer as routing on top of it and the specific link issues in the Data Link Layer and Physical Layer below it.

2.2 Data Link layer and Physical Layer

The radio network considered, consists of a number of radio units spread out in some terrain. If the received signal power from one radio unit is sufficient compared in relation to noise and interfering signal power, it is assumed that any two radio units can communicate, i.e., establish a link.

In this section we describe our model for the data link layer and physical layer. In essence, it is an interference-based model of the radio network, which is represented by a set of nodes \( V \) and the link gain \( G(i, j) \) between any two distinct nodes \( v_i \) and \( v_j, i \neq j \).

For the link level we will make the following assumptions:

- All antennas are isotropic.
- All nodes use equal transmission power.
- There is only one fixed required BER on the links.
- Slot synchronization is perfect.
- All packets are of equal length.
- A node cannot transmit more than one packet in a time slot and a node cannot receive and transmit simultaneously in a time slot.

The assumption on isotropic antennas and equal transmission power is mainly for simplicity, but in section 4.2 we will present a brief discussion of the consequences of directional antennas and varying transmission power specified on LET, since this assignment strategy will be affected most by these assumptions.

For any two nodes, \( v_i \) and \( v_j \), where \( v_i \) is the transmitting node and \( v_j \neq v_i \), we define the signal-to-noise ratio (SNR), \( \Gamma_{ij} \), as

\[
\Gamma_{ij} = \frac{P_i G(i, j)}{N_r},
\]

(2.1)
where \( P_i \) denotes the power of the transmitting node \( v_i \), \( G(i, j) \) is the link gain between nodes \( v_i \) and \( v_j \), and \( N_i \) is the noise power in the receiver. For convenience, we define \( \Gamma_{ii} = 0 \) corresponding to the physical situations of a node not being able to transmit to itself.

We say that a pair of nodes \( v_i \) and \( v_j \) form a link \( (i, j) \), if the signal-to-noise ratio (SNR) is not less than a communication threshold, \( \gamma_C \). That is, the set of links in the network, \( \mathcal{L} \), is defined:

\[
\mathcal{L} = \{(i, j) : \Gamma_{ij} \geq \gamma_C\}. \tag{2.2}
\]

Links are graphically depicted as in Figure 2.2.

For a set of links, \( L \subseteq \mathcal{L} \), we define the transmitting nodes:

\[
V_T(L) = \{v_i : (i, j) \in L\}. \tag{2.3}
\]

For any link, \( (i, j) \in L \), we define the interference as follows

\[
I_L(i, j) = \sum_{v_k \in V_T(L) \setminus v_i} P_k G(k, j). \tag{2.3}
\]

Furthermore, we define the signal-to-interference ratio (SIR):

\[
\Pi_L(i, j) = \frac{P_i G(i, j)}{(N_i + I_L(i, j))}. \tag{2.4}
\]

We assume that any two radio units can communicate a packet without error if the SIR is not less than a reliable communication threshold, \( \gamma_R \). A schedule \( S \) is defined as the sets \( Y_t \), for \( t = 1, 2, \ldots, T \), where \( T \) is the period of the schedule. The sets \( Y_t \) contain the nodes or links assigned time slot \( t \). A schedule is called conflict free if the SIR is not less than the threshold \( \gamma_R \) for all receiving nodes in all sets \( Y_t \).

However, due to mobility and limited information conflict-free schedules are very difficult to create and uphold. In order to make comparisons for distributed
2.3 Interactions with the Link Layer

scheduling we assume that links that have a lower SIR than $\gamma_R$ in a time slot can decrease its data rate as compared to the nominal data rate $R_N$ used by links with SIR above $\gamma_R$, i.e.

$$\frac{SIR}{\gamma_R} = \frac{\text{Used Rate}}{R_N},$$

(2.5)

The choice of these thresholds is of course dependent on several factors, such as the actual modulation method of the signal, properties of the receiver noise, data rate and required BER.

The threshold $\gamma_R$ will be determined by the factors described above. However, these factors only decide the lowest possible $\gamma_C$. That is, we can choose a higher $\gamma_C$, thereby excluding some node pairs from communicating with each other. By doing this we can create an interference margin so that all links can handle some interferences. However, this comes at the price of longer routes and the risk that the network will divide into sub-segments. For simplicity we will assume that $\gamma_R = \gamma_C$ for centralized scheduling. For distributed scheduling we may need the margin due to limited information and in these cases we will use $\gamma_C = 1.5\gamma_R$.

We have also chosen $\gamma_R$ equal to 10 in all simulations.

2.3 Interactions with the Link Layer

The network model we have described in this chapter is somewhat simplistic (although more complex than a graph model). In reality, all packets will not be perfectly received if the SIR is above $\gamma_C$, and all packets will not be lost just because SIR is below the threshold. If the sent packets had infinite size, such assumptions would be more accurate.

In addition, exact path gains will not be available and fast and slow fading will complicate the situation even further. An exact representation on the details of the link is difficult to obtain and even if we could get such information from the links in the network, an STDMA algorithm could not handle such fast changes anyway.

Instead, we have to hide the volatility of the link from the MAC protocol. The link gain given from lower layer will be an expected average. Variable data rates for the link (as seen from STDMA point of view) must be average data rates, not what actually is transmitted on the link. An adaptive radio node may actually change data rate on the link from one time slot to the next (in extreme cases we could even change it within the time slot).
The purpose of the STDMA algorithm is to create sets of simultaneously transmitting links that can be efficiently handled locally by the links (without any further interaction between them in terms of transmitted information).

Nevertheless, the actual functionality of the link will be ignored in this report and we will concentrate on the functionality of STDMA when the details of the link layer already have been hidden. This will also be the case for evaluations.

2.4 Transport and Network Layer

The traffic arriving at the network can be separated into two types. The first is unicast traffic, with a single source and destination. This type of traffic can be the carrier of many types of information, e.g. file transfer or telephone conversations. The second type of traffic is multicast or broadcast, i.e. a packet has one source but many destinations. With broadcast we mean the entire network. Broadcast traffic is very usual in military networks, e.g. group calls or situation awareness data.

Since all nodes cannot directly communicate with all other nodes in the network, due to limited transmission power, obstacles and large distances, all nodes in the network are assumed to be able to relay packets. In order to do this, each node is assumed to have a routing table with entry’s for all other nodes in the network. We will not elaborate on how this information is obtained but merely note that the routing will have an effect on the traffic in the network. Unless otherwise stated, we assume that all networks are connected, i.e. there is always a path between any pair of nodes.

In the following we first discuss the effects of this for unicast traffic and then the effects for broadcast traffic.

Unicast traffic assumes that a packet entering the network has only one destination. Packets enter the network at entry nodes according to a probability function, $p(v), v \in V$, and packets exit the network at exit nodes. When a packet enters the network, it has a destination, i.e. an exit node from the network. The destination of a packet is modeled as a conditional probability function, $q(w|v), (w, v) \in V \times V$, i.e. given that a packet has entry node $v$, the probability that the packet’s destination is $w$ is $q(w|v)$. For simplicity we will assume a uniform traffic model, i.e. $p(v) = 1/N$, and $q(w|v) = 1/(N - 1)$, where $N$ is the number of nodes, $N = |V|$. This assumption will not affect our results since we use traffic controlled schedules, thereby compensating for
2.4. Transport and Network Layer

variations caused by the input traffic model.

Let \( \lambda \) be the total traffic load of the network, i.e. the average number of packets per time slot arriving at the network as a whole. Then, \( \lambda/N(N-1) \) is the total average of traffic load entering the network in node \( v_i \) with destination node \( v_j \). As the network is not necessarily fully connected, some packets must be relayed by other nodes. In such a case, the traffic load on each link can be calculated only when the traffic has been routed.

For unicast traffic we use the shortest route counted in the number of hops, i.e. packets sent between two nodes will always use the path which requires the least number of transmissions. If several routes of the same length exist, all packets between two specific nodes will always use the same route.

The reason why this routing strategy is used, except for its simplicity, is that this minimizes the number of retransmissions needed before a packet reaches the destination. Since we for routing (scheduling is not yet done) are assuming a fixed data rate on the links, it would be difficult to take advantage of a strategy which uses a longer route, since we would have to generate schedules with a much better spatial reuse to compensate for this.

Now, let \( R_u \) denote the routing table for unicast traffic, where the list entry \( R_u(v, w) \) at \( v, w \) is a path \( p_{vw} \) from entry node \( v \) to exit node \( w \), where \( p_{vw} \) is given by the routing algorithm described above. Let the number of paths in \( R_u \) containing the directed link \( (i, j) \) be equal to \( \Lambda_{ij} \). From now on, we will call this parameter the relative traffic on link \( (i, j) \).

Comment: For non-uniform traffic the relative traffic should be weighted with the traffic load on each path rather than just using the number of paths.

Further, let \( \lambda_{ij} \) be the average traffic load on link \( (i, j) \). Then \( \lambda_{ij} \) is given by

\[
\lambda_{ij} = \frac{\lambda}{N(N-1)} \Lambda_{ij}.
\]

Moreover, we have, \( \lambda_i \) as the average traffic load on node \( v_i \). Here \( \lambda_i \) is given by:

\[
\lambda_i = \frac{\lambda}{N(N-1)} \sum_{j:(i,j) \in L} \Lambda_{ij} = \frac{\lambda}{N(N-1)} \Lambda_i,
\]

where \( \Lambda_i \) is denoted as the relative traffic of node \( v_i \).

This can be described in a similar way for broadcast traffic, which assumes that a packet entering the network has all other nodes as destination. Packets enter the network at entry nodes according to a probability function, \( p(v) \). We will also assume a uniform traffic model, i.e. \( p(v) = 1/N \).
Again, let $\lambda$ be the total traffic load of the network, i.e. the average number of packets per time slot arriving at the network as a whole. Then $\lambda/N$ is the total average of traffic load entering the network in node $v_i$ destined to all other nodes.

Radio is an inherent broadcast medium, especially when we are using omnidirectional antennas. This means that if we are sending a packet, all nodes within range can receive the packet if nothing else interferes with them. Since the interference allowed in the nodes is determined by the assignment strategy, this means that depending on what assignment strategy that is used this determines the actual traffic that must be transmitted by the nodes. For example, node assignment guarantees that all neighbors are collision-free, meaning that all neighboring nodes that should receive the packet will do so by a single transmission. Link assignment, on the other hand, is a single transmission over a link. If several neighbors should receive the packet, they have to receive a transmission each.

We will therefore define the average traffic of a node as the average number of different packets that are to be transmitted, regardless of whether they have one or several destinations. This means that the actual number of transmissions of the node will be at least this high, which is the case if node assignment is used since all packets will reach all their destinations with a single transmission each. Other transmission strategies may require a larger number of transmissions if more than one is required for a transmission, e.g. link assignment.

For broadcast traffic, a more advanced routing method must be used than for unicast traffic. As mentioned, radio is an inherent broadcast medium. Dependent on which assignment strategy that is used, we can use this to our advantage in a more or less efficient manner. One way of doing this is to minimize the number of retransmissions needed for a packet to reach all destinations, i.e. we want as many neighboring nodes as possible to be reached by each transmission. This can also be described as maximizing the number of leafs in the routing tree.

However, this is a very complex problem, so for simplicity we will use a heuristic algorithm in an attempts to achieve this.

The following creates a routing tree for each node.

Initiate by choosing the node as root. Find the node $v_i$ with the highest number of neighboring nodes that is not included in the tree. Include all these neighboring nodes and the edges from $v_i$ to these nodes. This is repeated until all nodes are included in the tree.

Now, let $R_b$ denote the routing table for broadcast traffic, where the list entry $R_b(v)$ at $v$ is a tree with entry node $v$ as root. Let the number of trees in
2.5 Routing assumptions for Mobile Networks

We assume that the routing is perfect and we ignore any necessary overhead caused by routing traffic. This also means that the routing protocol is assumed to be useful for the multicast transmissions that the STDMA algorithm needs. However, we also assume that the routing protocol reacts to link changes slower than the MAC layer is capable of handling, which means that new multicast routes is available only when the MAC protocol has figured out its new local neighborhood after a change.

This means that between link changes the STDMA algorithm may use the routing protocol for its updates regarding assignment of slots. But when a change takes place, the STDMA algorithm must first work out its new local neighborhood with the old routing information before the routing protocol adapts to the change.

The routing protocol in each node will have a better picture of how the whole network looks than the local picture from the STDMA algorithm in the node, but the routing protocol will be locally updated through the MAC layer.

2.6 Traffic Estimations

A similar issue is traffic from upper layers. For STDMA to work well, we will need to have accurate information on traffic loads in order to compensate for these different traffic loads. However, such information is not always available. Sometimes the applications may give some information, and sometimes we have to make estimations from the arriving traffic and existing queue lengths.

Such estimations may vary considerably over time, and this information may have to be hidden from the STDMA algorithm. How to estimate traffic
loads efficiently is a research area in itself and not much has been done for
ad hoc networks. But because this is outside the scope of this report, we will
assume that the actual expected traffic loads are known, as was calculated in
section 2.4.

2.7 Performance Measures

One important parameter which will affect the usefulness of a radio network is
the delay of information from source to destination. We will start by discussing
delay for unicast traffic and then discuss how this differs from broadcast traffic.

As seen from a network level, this can be translated to the network delay.
**Network delay** is the expected time, in time slots, from the arrival of a packet at
the buffer of the entry node to the arrival of the packet at the exit node, averaged
over all origin-destination pairs. This is the first parameter we will investigate.

To determine the network delay we need the following definitions and notations.

The stochastic variable **path delay** $D_{kl}^p$ for a path $p_{kl}$ is the time, in time
slots, from the arrival of a packet at the buffer of the arrival node $v_k$ to the
arrival of the packet at the destination node $v_l$. The stochastic variable **edge
delay** $D_{e_{kl}}^i(i, j)$ is the time, in time slots, from the packet arrives at the buffer of
node $v_i$ until it is received by node $v_j$, given that the packet is relayed on path
$p_{kl}$. This path is deterministically given by the routing algorithm as described
in the previous section.

The path delay is thus the sum of the edge delays of that path,

$$D_{kl}^p = \sum_{(i,j) \in p_{kl}} D_{e_{kl}}^i(i, j). \quad (2.6)$$

The average path delay is a stochastic variable defined as an average over
all origin-destination pairs:

$$\frac{1}{N(N-1)} \sum_{(k,l) \in N^2} D_{kl}^p. \quad (2.7)$$

The **network delay** $D$ is the expected value of the average path delay, i.e.

$$D = E\left[\frac{1}{N(N-1)} \sum_{(k,l) \in N^2} D_{kl}^p\right]. \quad (2.8)$$
Due to the relaying of packets the statistical properties of this variable are complicated, and an exact analytical analysis of the network delay is difficult [7]. Instead we will use computer simulations to determine this parameter, see appendix A.

We will now compare the above-described with broadcast traffic. When we have broadcast traffic, each arriving packet has all other nodes as destination. We can now use several different definitions of the packet delay since a packet will arrive at its destinations at different times. One example would be the maximum delay, i.e. the delay of a packet would be the delay until the arrival at the last node. However, most nodes will experience a much lower delay. So instead we will use the average delay of a packet. The total delay of a packet that originated at node $v_i$ can be written as

$$D_i = \frac{1}{N-1} \sum_{j: (i,j) \in \mathcal{N}^2} D_{ij}$$

where $D_{ij}$ can be described as the path delay between node $v_i$ and node $v_j$. This is the path in the tree with root $v_i$ given by the routing algorithm for broadcast routing described in the previous section.

The network delay, $D$, can thus be described with the following expression

$$D = \mathbb{E}\left[ \frac{1}{N} \sum_{i \in \mathcal{N}} D_i = \frac{1}{N(N-1)} \sum_{(i,j) \in \mathcal{N}^2} D_{ij} \right],$$

which is similar to the expression for unicast traffic. However, the actual traffic on the nodes and links will differ, which will result in different delays for the different traffic types.

The demand for higher data rates can often lead to highly loaded networks. One of the advantages of STDMA is that it can function well even under high traffic loads.

In particular, the largest admissible traffic load yielding a finite network delay is highly interesting. This maximum traffic load is commonly referred to as the maximum throughput of the network. We define the maximum throughput as the number $\lambda^*$ for which the following expressions hold for all traffic loads $\lambda$,

$$\begin{cases} 
\lambda < \lambda^* & \text{yields bounded } D \\
\lambda > \lambda^* & \text{yields unbounded } D
\end{cases}$$

Notice, here maximum throughput is the same as uniform capacity, i.e. the minimum bit rate at which any node can communicate with any other node in the network, because our choice of traffic model.
Maximum throughput is the second parameter we study. In mobile scenarios this parameter will be a little more complex, a problem is that the capacity of the network will vary over time and even if the expected end-to-end delay for a node pair may be infinite at some moment, due to the fact that capacity is low at that moment may not result in infinite delay since the network can change into a more beneficial situation, allowing all queued packets to reach their destination. Still, the expected maximum throughput as a function of time is a very interesting parameter to study since it gives us a good indication on whether a specific STDMA schedule is good or not. Besides, unless we can give any prediction about how units move, a good estimator will be that the network does not change.

As mentioned, a problem is that the network sometimes is not connected, node pairs in different parts of the network can then not communicate, resulting in infinite expected delay on these paths and no maximum throughput. This does neither represent the situation very well, nor can any MAC algorithm can do anything about it. In such cases we instead assume that only connected paths attempts to transmit packets when estimating throughput.

The centralized evaluation will study both network delay and maximum throughput for different assignment methods in static scenarios in order to determine which properties that are wanted on an STDMA algorithm.

Most evaluations for distributed algorithm will be comparisons to ideal (they have all information) centralized schemes, this will give estimates on how well the distributed algorithm perform compared to an upper limit on their possible performance.

For the specific assignment methods we study in this thesis, we present a good estimation of the maximum throughput given the schedule and a specific routing. Except for LET with broadcast traffic, in which case simulations will be performed, this is done in chapter 4.

We define connectivity as the fraction of nodes in the network that can be reached by a node, in one hop, on average, i.e. \( M/(N(N-1)) \), where \( M \) is the number of directed links in the network.
Chapter 3

Node and Link Assignment

This chapter describes the two most frequently used assignment methods so far, node assignment and link assignment, and describes the advantages of both of them. A preliminary comparison between these two assignment methods was done in [28] showing node assignment to be preferable. However, as mentioned in Chapter 1 they did not consider traffic sensitivity, which has a considerable impact on system performance, see for example [30]. We start the chapter by describing the centralized STDMA algorithm we use.

3.1 STDMA scheduling

In this section we describe and motivate the choice of the centralized STDMA algorithm that we use in the first part of the thesis. This algorithm is described in such a way that it can be used independently on the specific assignment strategy.

In all assignment strategies we effectively assign time slots to sets of links. In link assignment, these sets consist of single links, and node assignment can be described as all outgoing links from the assigned node. We will use the notation $x_i$ for one of these link sets and the notation $X$ for the union of all link sets. One thing to notice is that a link $(i, j)$ can belong to more than one link set, although this is not the case for node or link assignment.

A schedule $S$ is defined as the sets $Y_t$, for $t = 1, 2, \ldots, T$, where $T$ is the period of the schedule. The sets $Y_t$ contain the link sets assigned time slot $t$.

We will use the notation transmit simultaneously to denote that all possible receiving nodes of the link sets assigned the time slot have SIR above the reliable communication threshold. A schedule is called conflict-free if this is the case
The algorithm assumes full knowledge of the interference environment. It needs as an input the basic path-loss, or estimates, from all nodes to all other nodes.

This is a description of a basic algorithm.

For each new loop a time slot, numbered $t$, is created. In step two of the algorithm we first check to see if the link sets that have not yet received a time slot can be assigned to this slot, i.e. transmit simultaneously with the assigned link sets. In the third step the same check is performed on the rest of the link sets. For some $t = T \leq |V|$, all link sets will have received all their time slots and the algorithm will terminate. The output of the algorithm are the sets $Y_t$ for $t = 1, 2, \ldots T$, where $T$ is the period of the schedule and the length of the frame. When the algorithm terminates the sets $Y_t$ will contain the link sets $x_i$ that are assigned to time slot $t$.

In multi-hop networks the traffic load on the various nodes will differ considerably. This will cause "bottleneck" effects at busy nodes with long packet delays as a result. This can be compensated for by assigning the more heavily loaded link sets several time slots.

That is, the algorithm works so that each link set is guaranteed a certain number of slots in each period or frame. This number is based on the relative traffic of each link set. Furthermore, the link sets are assigned time slots according to a priority list. The priority of a link set is based both on the relative traffic load and on the number of time slots passed since the link set previously was assigned a slot. With this procedure, the slots assigned to each link set will be spread out evenly over the period, resulting in a decreased network delay.

In the following we will use the notation $\Lambda^x_i$ to denote the relative traffic of link set $x_i$. If we use link assignment, the link set is actually a link $(i, j)$ and $\Lambda^x_i = \Lambda_{ij}$. Node assignment, on the other hand, means that $x_i = \{(i, k) : (i, k) \in L\}$ and $\Lambda^x_i = \Lambda_i$.

In the algorithm, link set $x_i$ will be guaranteed $\Lambda^x_i$ number of slots per frame.

In the final schedule all link sets will have at least this many time slots, and some may have more. In general the algorithm will work for any fixed routing.

With the above procedure, link sets with a high traffic load will obtain several slots per frame. In this case the network delay will also depend on how evenly over the frame, these slots are arranged. To spread out the slots over the frame the link sets are ordered in a list of priority. The link set priority is set to $\tau_i \Lambda^x_i$, where $\tau_i$ is the number of slots that have passed since the link set was previously allocated a time slot. The link set allocation is then performed in the
order described by the priority list, highest priority first, etc.

In the following we describe the traffic sensitive algorithm obtained with the above ideas. In addition to the inputs to the basic algorithm, this algorithm needs the knowledge about the relative traffic of each link set.

**Step 1** Initialize:

1.1 Enumerate the link sets
1.2 Create a list, \( A \), containing all of the link sets and an empty list \( B \).
1.3 Set \( t \) to zero.
1.4 Calculate the number of time slots each link set is to be guaranteed and set \( h_i = \Lambda_i^x \).
1.5 Set \( \tau_i \) to zero for all link sets.

**Step 2** Repeat until list \( A \) is empty:

**Step 2.1** Set \( t \leftarrow t + 1 \) and \( Y_t \leftarrow \emptyset \).

**Step 2.2** For each link set \( x_i \) in list \( A \):

2.2.1 Set \( Y_t \leftarrow Y_t \cup x_i \).
2.2.2 If the link sets in \( Y_t \) can transmit simultaneously:
   - If \( h_i = 1 \), remove the link set from list \( A \) and add to list \( B \).
   - Set \( h_i \leftarrow h_i - 1 \), and set \( \tau_i \) to zero.
2.2.3 If the set of link sets in \( Y_t \) cannot transmit simultaneously, set \( Y_t \leftarrow Y_t \setminus x_i \).

Set \( \tau_i \leftarrow \tau_i + 1 \).

**Step 2.3** For each link set \( x_i \) in list \( B \) but not in \( Y_t \):

2.3.1 Set \( Y_t \leftarrow Y_t \cup x_i \).
2.3.2 If the link sets in \( Y_t \) can transmit simultaneously, set \( \tau_i \) to zero.
2.3.3 If the link sets in \( Y_t \) cannot transmit simultaneously, set \( Y_t \leftarrow Y_t \setminus x_i \)
   and set \( \tau_i \leftarrow \tau_i + 1 \).

**Step 2.4** Reorder lists \( A \) and \( B \) according to link set priority,
\[ \tau_i \Lambda_i^x, \]
highest priority first.
3.2 Link Assignment

In link-oriented assignment, the directed link is assigned a slot. A node can thus only use this slot for transmission to a specific neighbor. In general this knowledge can be used to achieve a higher degree of spatial reuse. The effect is higher maximum throughput.

Below, we describe the criteria for a set of links to be able to transmit simultaneously with sufficiently low interference level at the receiving nodes.

We say that a link \((k, l)\) is adjacent to any other link \((i, j)\) \(\in L\) iff \(\{i, j\} \cap \{k, l\} \neq \emptyset\) \((i, j) \neq (k, l)\). Furthermore we define \(\Psi(L)\) as the union of all adjacent links to the links in \(L\). We assume that a node cannot transmit more than one packet in a time slot and that a node cannot receive and transmit simultaneously in a time slot. Alternatively, we say that a set of links \(L\) and the set of its adjacent links \(\Psi(K)\) must be disjoint:

\[
L \cap \Psi(L) = \emptyset. \tag{3.1}
\]

We also require that the SIR value is sufficiently high for reliable communication, see section 2.2,

\[
\Pi_L(i, j) \geq \gamma_R \ \forall (i, j) \in L. \tag{3.2}
\]

If the above two conditions, (3.1) and (3.2), hold for a set of links \(L \in \mathcal{L}\), we say that the links in \(L\) can transmit simultaneously.

Figure 3.1 shows a small network. Assuming that interference between nodes without communication links is small, we can see that links 1, 2 and 3 can transmit simultaneously.

One problem with this assignment method is that it does not take advantage of the inherent broadcast properties of the radio medium. Each transmitted packet will only be received by the assigned receiver despite being sent in all directions (omnidirectional antennas). This is no problem with unicast traffic, but for broadcast traffic, where each packet should reach several destinations, it is inefficient. In these cases the packet has to be retransmitted for each of these receivers.

For example, if a broadcast packet has node \(B\) as source, it must be transmitted three times from node \(B\). Link 1, 5 and 6 must transmit the packet at different time slots. Then to reach the last destination, the packet must be transmitted on links 2 and 3. As these two links can transmit simultaneously, it is possible to use only four time slots in order to reach all destinations.
3.3 Node Assignment

In a node-assigned schedule, a node is allowed to transmit to any of its neighbors in its slot. If the schedule is to be conflict-free, this means that we have to guarantee that we will not have a conflict in any of the neighboring nodes.

The advantage is that if the packet is supposed to reach more than one of the neighbors to a node, one transmission of the packet is sufficient to reach them all. This makes node assignment very efficient for broadcast traffic.

There are two necessary conditions for the situation when all the nodes in a set $V$ are allowed to transmit packages simultaneously. Let the neighbors $\Omega(v)$ to a node $v \in V$ be the set of all nodes that have a link from $v$ to itself. The neighbors are the nodes that $v$ possibly can transmit a packet to. Similarly, let $\Omega(V)$ denote the union of all neighbors of all nodes in $V$.

The first condition is that two neighbors cannot transmit at the same time. Another way to say this is that the sets $V$ and $\Omega(V)$ must be disjoint:

$$V \cap \Omega(V) = \emptyset. \quad (3.3)$$

Let $L(V)$ be the set of all links from the nodes in $V$ to their neighbors in $\Omega(V)$. 

![Figure 3.1: A small example.](image)
Since it must be possible to use all the links in $L(V)$ for transmission simultaneously, we state the second condition:

$$\Pi_{L(V)}(i, j) \geq \gamma_R$$

for all $(i, j) \in L(V)$. (3.4)

If the above two conditions, (3.3) and (3.4), hold for a set of nodes $V \in \mathcal{V}$, we say that the set of nodes can transmit simultaneously.

If we return to the example network and compare it with link assignment, we see that the transmitting nodes to the three links that could transmit simultaneously cannot be permitted to transmit simultaneously when we use the node-assignment strategy, since both node $A$ and $C$ may choose to transmit to $B$.

In the broadcast traffic example, nodes $A$, $C$, and $E$ can be reached by one transmission of $B$, thereby decreasing the number of necessary transmissions. However, to reach the last two nodes we must transmit on $A$ and $C$, and these nodes cannot transmit simultaneously. This results in the use of three time slots in order to reach all destinations, which is one time slot lower than for link assignment.

### 3.4 Analysis

In this section we present some analytical results that are useful when evaluating the properties of node and link assignment. We first discuss maximum throughput and then continue with an approximation of the network delay at low traffic arrival rate.

In [66] the maximum throughput in a network with fixed capacities on the links and fixed routing is determined. Here, we will do the same specifically for a link-assigned schedule and a node-assigned schedule.

A link-assigned schedule contains, for each time slot in the frame, the set of links $(i, j)$ that are allowed to transmit in that time slot. A node-assigned schedule contains the set of nodes $v_i$ that are allowed to transmit in each time slot. We assume that the number of time slots in a frame, $T$. We do not consider packet failures and retransmissions.

Each link may be assigned several time slots per frame. Let $h_{ij}(S_L)$ denote the number of slots in schedule $S_L$ where link $(i, j)$ can access a time slot and similarly let $h_i(S_N)$ denote the number of slots in schedule $S_N$ where node $v_i$ can access a time slot.
3.4. Analysis

3.4.1 Maximum throughput for link assignment

The following is done for link assignment. Later we will do the same for node assignment. As was previously defined, the maximum throughput is the largest admissible traffic load yielding a finite network delay. Studying equations (2.6) and (2.8), it is easy to realize that the network delay is bounded as long as the edge delay of all links is bounded; the network delay is unbounded if the edge delay of any of the links is unbounded.

So for a good estimation of the throughput, we only need to find the smallest value of $\lambda$ so that at least one of the link queues are saturated. This will happen when the offered traffic rate on the link $\lambda_{ij}$ equals the rate at which it can transmit packets, i.e. $h_{ij}/T_L$ packets per time slot, where $T_L$ is the frame length of the link-assigned schedule. This gives us the following equation for each link in the network for unicast traffic

$$\lambda_{ij} = \frac{\lambda}{N(N-1)} \Lambda_{ij} = \frac{h_{ij}}{T_L},$$

thus resulting in the following saturation traffic arrival rate for the link

$$\lambda = \frac{h_{ij} N(N-1)}{T_L \Lambda_{ij}}.$$

The maximum throughput of the network $\lambda^*(S_L)$ can now simply be calculated as

$$\lambda^*(S_L) = \min_{(i,j) \in L^*} \frac{h_{ij}(S_L) N(N-1)}{T_L \Lambda_{ij}}. \quad (3.5)$$

Furthermore, if we use schedules that are fully compensated for traffic, i.e. $h_{ij} = \Lambda_{ij}$, the formula for link assignment can be simplified to

$$\lambda^*_L = \frac{N(N-1)}{T_L}. \quad (3.6)$$

For broadcast traffic the formula for the maximum throughput will be

$$\lambda^*_L(S_L) = \min_{(i,j) \in L^*} \frac{h_{ij}(S_L) N}{T_L \Lambda_{ij}}, \quad (3.7)$$

and for a schedule that is fully compensated for traffic

$$\lambda^*_L = \frac{N}{T_L}. \quad (3.8)$$
Figure 3.2: The figure shows the ratio between estimated values of the maximum throughput and simulated values for the maximum throughput for networks of different connectivity. The ratio is plotted for 400 networks of size 10 nodes with unicast traffic.

In Figure 3.2 we show a comparison of this estimation with simulation results. As can be seen this estimation is sufficiently good for our purposes. The largest part of the deviation between simulated and estimated results is probably due to the high delay variance of the simulation results close to the maximum throughput, see Appendix A for more information.

### 3.4.2 Maximum throughput in node assignment

The corresponding notation for node assignment is $h_i(S_N)$ as the number of time slots assigned to node $v_i$, and $T_N$ as the frame length. The maximum throughput for node assignment in the case of unicast traffic is

$$\lambda^*(S_N) = \min_{i \in V} \frac{h_i(S_N)N(N - 1)}{T_N\Lambda_i}, \quad (3.9)$$

and the corresponding result for broadcast traffic can be written as

$$\lambda^*(S_N) = \min_{i \in V} \frac{Nh_i(S_N)}{T_N\Lambda_i}. \quad (3.10)$$

For schedules that are fully compensated for traffic, i.e. $h_i = \Lambda_i$ the formula...
3.4. Analysis

for node assignment with unicast traffic can be simplified to

$$\lambda^*_N = \frac{N(N-1)}{T_N}$$

(3.11)

and, similarly, for broadcast traffic

$$\lambda^*_N = \frac{N}{T_N}$$

(3.12)

The ratio between the maximum throughput of a link-assigned schedule and the maximum throughput of a node-assigned schedule can then be written as:

$$\frac{\lambda^*_L}{\lambda^*_N} = \frac{T_L}{T_N}$$

(3.13)

Notice, this is valid for both unicast and broadcast traffic.

3.4.3 An approximative formula for the network delay

We now derive an expression for the network delay $D_L$ for a link-assigned schedule with unicast traffic,

$$D_L = \sum_{(i,j) \in L} \frac{\Lambda_{ij}}{N(N-1)} d_{ij},$$

(3.14)

where $d_{ij}$ is the average expected edge delay on link $(i, j)$.

In order to determine the network delay, we make two extra assumptions. Even if they are not fulfilled, it is still a useful approximation of the network delay.

First, the slots in the schedules assigned to a node are perfectly spread over the frame, i.e. the distance between two assigned slots is equal. The algorithm used in the simulations attempts to do this, but the algorithm is not optimal and it is not always possible to spread the slots evenly.

Second, the relay traffic can be described as a Poisson process. This is certainly not the case, since relay packets can only arrive in specific time slots. However, it is normally a good approximation. This is an attempt to use the same principle as the independence assumption [66], but for a TDMA network with fixed packet size.
With these assumptions, the average expected edge delay for a link can at low traffic arrival rate be written as:

\[ d_{ij} = \frac{T_L}{2h_{ij}}, \]

which inserted in (3.14) gives the network delay of a link-assigned schedule

\[ D_L = \sum_{(i,j) \in L} \frac{\lambda_{ij}}{N(N-1)} \frac{T_L}{2h_{ij}}. \]

Furthermore, if the schedule is fully compensated for traffic, we have

\[ D_L = \frac{MT_L}{2N(N-1)}, \]

where \( M \) is the number of links in the network.

The corresponding result for a node-assigned schedule is

\[ D_N = \sum_{i \in V} \frac{\lambda_i}{N(N-1)} \frac{T_N}{2h_i}, \]

and for a schedule fully compensated for traffic

\[ D_N = \frac{NT_N}{2N(N-1)}. \]

The ratio between the network delay of link assignment and the network delay of node assignment can then be written as:

\[ \frac{D_L}{D_N} = \frac{MT_L}{NT_N} = \frac{M\lambda_i^N}{N\lambda_L}. \]

(3.15)

Simulations in the next section will show that this is a good approximation except for low connectivity.

### 3.5 Evaluation and Results

In this section we compare the delays and maximum throughput for node assignment and link assignment. The results for network delay are generated by network simulation as described in appendix A. Furthermore, for unicast traffic we compare these simulated results with the approximative results given in section 3.4. The comparison begins with unicast and concludes with broadcast traffic.
3.5. Evaluation and Results

3.5.1 Unicast Traffic

We start the investigation by studying the networks at high traffic loads. The first parameter we study here is the ratio between maximum throughput $\lambda^*_L$ of link assignment and maximum throughput $\lambda^*_N$ of node assignment, i.e.

$$\frac{\lambda^*_L}{\lambda^*_N},$$

which we know from equation (3.13) equals $T_N/T_L$.

As can be seen in Figure 3.3, this ratio exhibits considerable variations over the networks studied. One conclusion in these simulations is that link assignment provides higher throughput. This is not so surprising since the degree of spatial reuse is higher for link assignment.

To determine how much better link assignment can be, we plot in Figure 3.4 the ratio $\lambda^*_L/\lambda^*_N$ averaged over connectivity for networks of different sizes.

As can be seen, $\lambda^*_L/\lambda^*_N$ increases with the size of the network and decreases if connectivity is increased. The estimated standard deviation of this ratio is around 0.04 when it is largest.

We continue by studying the network delay at low traffic loads. The second parameter studied is the ratio between network delay of link assignment and node assignment at low traffic loads. In Figure 3.5 this parameter

Figure 3.3: The figure shows the ratio between maximum throughput for link assignment and node assignment for networks of different connectivity. The ratio is plotted for 500 networks of size 20 nodes.
can be studied. The variance is rather low and a linear relationship between delay and connectivity can be detected.

To see how well equation (3.15) approximates the delay we plot $D_L/D_N \times \lambda_N/\lambda_L$, which should be approximately $M/N$. As can be seen in Figure 3.6, this works fairly well. $D_L/D_N$ is slightly lower than predicted independent of connectivity.

The last parameter we study for unicast traffic is the input traffic load of the network which gives equal network delay for node and link assignment. This parameter is interesting since it determines for what traffic loads link/node assignment is preferable. That is, for traffic loads higher than this parameter, link assignment is preferable, and at lower traffic loads node assignment is preferable. As can be seen in Figure 3.7 the variance over the simulated networks of this parameter is less than for the other parameters, which means that the average value is highly interesting. This average is shown in Figure 3.8. From this we can conclude that for unicast traffic the preferable assignment method can be determined with knowledge only of the connectivity of, and input traffic to, the network.
3.5. Evaluation and Results

Figure 3.5: The figure shows the ratio between delay for link assignment and delay for node assignment. The ratio is plotted for 500 networks of size 20 nodes.

Figure 3.6: The figure shows the ratio between delay for link assignment and delay for node assignment multiplied by the ratio between the maximum throughput for link assignment and maximum throughput for node assignment. This should approximately be equal to the connectivity of the network multiplied by $N - 1$, which is the line plotted in the figure. The ratio is plotted for 500 networks of size 20 nodes.
Figure 3.7: The figure shows the input traffic level giving equal network delay for different network connectivity. This is plotted for 500 networks of size 20 nodes.

Figure 3.8: The figure shows the traffic load which gives equal network delay for 500 networks of sizes 10, 20, and 40 nodes.
3.5. Evaluation and Results

3.5.2 Broadcast Traffic

We start the investigation on broadcast traffic by returning to the case of networks at high traffic loads. The parameter studied is the ratio between maximum throughput $\lambda^*_L$ of link assignment and maximum throughput $\lambda^*_N$ of node assignment, i.e.

$$\frac{\lambda^*_L}{\lambda^*_N},$$

which equals $T_N/T_L$ for broadcast traffic, too.

This ratio can be studied for networks of size 20 nodes in Figure 3.9. Similar to unicast traffic, this parameter exhibits considerable variations over the networks studied.

However, unlike for unicast, the decrease in variation with connectivity is significant. For high connectivity there is very little variation left, less than 0.01 and less for medium and high connectivity.

Not very surprisingly, node assignment behaves better than link assignment for nearly all networks, especially for high connectivity, although there are networks where link assignment achieves the higher throughput.

To make a more complete comparison, we plot, in Figure 3.10, the ratio $\lambda^*_L/\lambda^*_N$ averaged over connectivity for networks of different sizes. As can be seen, $\lambda^*_L/\lambda^*_N$ decreases with both network size and connectivity. This is because

![Figure 3.9: The figure shows the ratio between maximum throughput for link assignment and node assignment for broadcast traffic. The ratio is plotted for 500 networks of different connectivity of size 20 nodes.](image)
Chapter 3. Node and Link Assignment

Figure 3.10: The figure shows the average ratio between maximum throughput for link assignment and node assignment for broadcast traffic. The ratio between maximum throughput for networks of different size 10, 20, and 40 nodes.

the number of neighbors a node has increases when any of these parameters are increased, which node assignment can take better advantage of. The estimated standard deviation of the average ratio is here as much as 0.07 at very low values, however, it quickly decreases to 0.01 and less for medium and high connectivity giving us a better estimate for these connectivities.

We continue by studying the network delay at low traffic loads. In Figure 3.11 we see that node assignment always performs better that link assignment, especially for networks of high connectivity. In Figure 3.12 this is averaged over the networks. We see that this effect increases with network size and connectivity in the same way as for throughput. The estimated standard deviation of the average ratio is here less than 0.5 for all values.

From this we can conclude that link assignment can never compete with node assignment for pure broadcast traffic.

3.6 Conclusions

In this chapter we have studied the two most common assignment strategies used for STDMA. We conclude that, in the unicast case, link assignment behaves better for high traffic loads, achieving a higher throughput. However, this comes
at a cost of higher delay than node assignment for low traffic loads.

For broadcast traffic, node assignment is always preferable. Basically, for heavily loaded networks with mainly unicast traffic, link assignment is to be preferred. If we have a considerable part which is broadcast traffic, we should use node assignment.

Both methods have drawbacks, however, so in the next chapter we will study a novel strategy which attempt to combine the advantages of these methods.
Figure 3.12: The figure shows the average ratio between delay for link assignment and delay for node assignment for networks of different connectivity with broadcast traffic. The ratio between delay for networks of different size 10, 20, and 40 nodes.
Chapter 4

Extended Transmission Rights

In this chapter we present a novel assignment strategy which is based on link assignment.

4.1 LET principle

Notice that the interference term in (2.3) depends only on which nodes that are transmitting and not on the nodes that are receiving packets. Assume that a node is assigned as a transmitter in a slot, i.e. an outgoing link of the node is assigned the slot. If this node redirects the transmission to a node, other than the assigned receiving node, the inequality in (3.2) still holds for all links originally assigned to the slot. This means that the interference level of the other simultaneously receiving nodes will not change. (Recall the assumption of omnidirectional antennas.) The redirected transmission in itself cannot always be guaranteed to be conflict-free.

Based on these observations, we suggest the following scheme for extending transmission rights for any given link-assigned schedule. When a link is assigned a time slot, the node first checks whether there is a packet to transmit on that link. If there is no such packet, any other link with the same transmitting node might be used if the node has a packet to transmit. Preferably links that are conflict-free should have priority, so as to avoid unnecessary packet loss.

We call this strategy Link assignment with Extended Transmission rights (LET).

To illustrate how it works we present a simple example. Assume links 1, 2, and 3 in Figure 4.1 have be scheduled to transmit in the same slot. Let us
study node B in more detail. If node B does not have any packets to transmit to E, it is permitted to transmit on either of the links 5 or 6 in the slot assigned to link 1. Now, if both links 2 and 3 are used (or if these nodes also use the LET property), neither transmission on 5 or 6 will be successful. However, for low traffic the probability that this would happen is small. If none of the other two nodes use their slot, the redirected transmission will be successful; and if only one of them transmits, we still have 50 percent probability of a success. This is because node B cannot know which one (if any) of the others will transmit.

We now continue by proving that by redirecting the transmissions the nodes in the network will not cause any conflict at any node which has not redirected their transmission. Assume that \( L \) is a set of links such that they can transmit simultaneously according to equations (3.1) and (3.2), i.e.

\[
\Pi_L(i, j) \geq \gamma_R \quad \forall \ (i, j) \in L.
\]

Furthermore, assume that the transmitting nodes of \( L_R \subseteq L \) redirect their transmissions to other receiving nodes than scheduled in the initial link schedule and that \( L_{NR} \) is the rest of the links, i.e.

\[
L_{NR} = L \setminus L_R.
\]
Let \( L_U \) be the set of links used by the redirecting nodes, therefore,

\[
V_T(L_U) = V_T(L_R).
\]

If \( L_{NR} \) is to be conflict-free, the following inequality must be valid

\[
\Pi_{L_{NR} \cup L_U}(i, j) \geq \gamma_1 \quad \forall \ (i, j) \in L_{NR}.
\]

For any \((i, j) \in L_{NR}\), we can write

\[
\Pi_{L_{NR} \cup L_U}(i, j) = \frac{P_i G(i, j)}{(N_t + I_{L_{NR} \cup L_U}(i, j))}.
\]

and

\[
I_{L_{NR} \cup L_U}(i, j) = \sum_{v_k \in V_T(L_{NR} \cup L_U) \setminus v_i} \frac{P_k}{L_b(k, j)}.
\]

However,

\[
V_T(L_{NR} \cup L_U) = V_T(L_{NR}) \cup V_T(L_U) = V_T(L_{NR}) \cup V_T(L_R) = V_T(L),
\]

resulting in, \( I_{L_{NR} \cup L_U}(i, j) = I_L(i, j) \) and \( \Pi_{L_{NR} \cup L_U}(i, j) = \Pi_L(i, j) \), which of course fulfills (3.2).

For broadcast traffic, the situation will be a little more complicated. Each packet transmitted may have more than one receiver. This means that some, but not all, packets may be received correctly. A worst case scenario would be that only transmissions over scheduled links would be received, and all of the rest of the links must be retransmitted. LET can then behave exactly as link assignment for high traffic loads. On the other hand, for low traffic loads, the behavior will be that of node assignment, due to the low interference in the network. (The interference may be higher for broadcast traffic than for unicast traffic since a packet will be split and then might be relayed simultaneously by two nodes.)

However, there is also a possibility that more links than scheduled happen to be collision-free. This can easily be the case if two nodes lie close each other. Any transmission with sufficiently high SIR to one of them will also reach the other with sufficiently high SIR.

The variable distance between the nodes in this type of network would be another reason for an improvement compared to link assignment. If the distance between transmitter and receiver is small enough, the signal strength will be sufficient to handle very high interference, which can often be the case even if the link is not scheduled to the time slot.
We conclude this section by giving a more formal description of the LET algorithm that we use in our simulations. This is done in each node:

**Step 1** If there is a packet in the scheduled link’s queue:

1.1 Transmit this packet on all links that it is supposed to be transmitted on.

1.2 Remove the packet from all link queues where this transmission was successful.

**Step 2** Otherwise if there is no such packet. Choose one of the node’s other outgoing links that can be exchanged conflict-free in the schedule that has a packet in queue:

2.1 Transmit this packet on all links that it is supposed to be transmitted on.

2.2 Remove the packet from all link queues where this transmission was successful.

**Step 3** Otherwise if there is no such packet. Choose one of the rest of the node’s outgoing links that has a packet in queue:

3.1 Transmit this packet on all links that it is supposed to be transmitted on.

3.2 Remove the packet from all link queues where this transmission was successful.

**Step 4** Otherwise, do nothing.

### 4.2 Basic Properties

In Figure 4.2, network delay for different $\lambda$ is shown schedules using for three different assignment strategies in a network of 30 nodes with unicast traffic. The assignment strategies are node assignment, link assignment and LET.

We see from this figure that in this network, link assignment is preferable to node assignment for high traffic loads. For low traffic loads, node assignment achieves a smaller delay. The LET method combines the advantages of the two methods and in this case achieves a smaller delay for all traffic loads.
4.2. Basic Properties

Figure 4.2: Network delay in a 30 node network.

In this figure some areas of interest can be seen; at low traffic loads, the ratio between delay of node assignment and delay of LET. Moreover, the ratio between delay of link assignment and delay of LET is interesting.

At high traffic loads, the ratio between maximum throughput of node and link assignment is interesting. The ratio between the maximum throughput of LET and link assignment is also interesting, but at very high traffic levels most of the links in the network will have packets in queue. For this case, LET will appear mainly as link assignment. LET will achieve the same maximum throughput as the link-assigned schedule it is based on if the link-assigned schedule is fully traffic compensated.

This is because at very high traffic loads, the probability that a link will have a packet to transmit in its time slot will be close to one. The LET property will not be used, and the network will appear exactly as a link-assigned schedule. No conflicts will appear, resulting in the same maximum throughput as the link-assigned schedule (for unicast traffic).

If the link-assigned schedule is not fully compensated for varying network traffic, LET will give at least as high maximum throughput as the link-assigned schedule. This is because no packet transmitted on a link assigned in the time slot is lost in LET, and a highly loaded link can use one of the lower loaded outgoing links from that node.

Although any link-assigned schedule may be used to extend transmission rights, some link schedules may give LET more or less desirable properties.
Here we discuss what effect the link schedule has on delay.

At low traffic load, the nodes will normally only have at most one packet in queue at a time. In this case, LET behaves as node assignment, with the node-assigned schedule as the transmitting nodes in the link-assigned schedule. A problem when generating STDMA schedules is how to determine which slots to give a node or link that is going to receive more than one slot, since delay through the node depends on which specific time slots the node is given.

For example, assume that a node has received two time slots and these are spread in such a way that the distance between them is approximately half the frame length. Then for low traffic loads, the delay will be at most half the frame length. If the node is given two consecutive time slots, the maximum delay might be the entire frame length. That is, it is usually efficient to spread a node’s time slots evenly over the frame. This problem gets worse if nodes receive many time slots, especially since a large part of the traffic usually flows through these nodes.

In some algorithms, see e.g. [30], an effort is made to spread the time slots a node or link is given equally over the frame. However, even if the link schedule has perfectly spread time slots, this may not be the case for the transmitting nodes. Therefore, LET might give considerably higher delay than a node-assigned schedule if the link schedule tends to give the transmitting nodes consecutive time slots.

Any assignment algorithm, especially if it is of a greedy type, has a set of rules to determine which link to assign to a time slot. One method used is node ID, i.e. the lower the ID number, the higher the priority. To assign links, the pair of node IDs of the transmitting and receiving nodes can be used. A sorted list according to link priority would then give the outgoing links from a node consecutive places in the list. Even if another system for link priority is used, node ID is probably eventually used if priority is equal for several links.

Now, assume we have a fully connected network, i.e. all nodes can communicate with all other nodes without relaying. Furthermore, in this case we can assume that there is no spatial reuse, although in a link schedule using an interference-based model, this might be possible due to capture. In this case the assigned schedule would be the sorted list described above, which gives high network delay.

One way to avoid this problem is to give the links a link ID which is random, although different, for each link.

The assumptions used in this paper is the use of omnidirectional antennas and equal transmission power of all nodes in the network. If this is not the case,
a node cannot redirect its transmission to any other of its neighbors since this might require an increase of signal power or redirection of the antennas. Any change of the outgoing power strength and direction can ruin the conflict-free properties of the other receiving nodes.

However, some of the nodes may still be reached without such a change, and the LET properties can still be used with these nodes, although this is less efficient than if all nodes could be reached.

An alternative way of designing a LET schedule would be to base it on a node-assigned schedule instead and replace each node in the schedule with one of its outgoing links. Using LET on this schedule would result in a schedule that would behave as a node-assigned schedule for all traffic loads and traffic types, thereby achieving the advantages of node assignment for broadcast traffic. The actual order in which packets are transmitted can be different, though.

Unfortunately, this schedule would not have the positive properties of link assignment for unicast traffic.

**4.3 Analysis**

The same assumptions as we made for node and link assignment in previous chapter can be made for LET as well. However, since LET behaves like a node-assigned schedule at low traffic loads, the link-assigned schedule would have to try to spread the time slots for the transmitting nodes of the links evenly over the frame instead of the link time slots. However, the algorithm used in the simulations does not attempt to do this. This results in a larger discrepancy than for node assignment or link assignment when compared with simulations.

The network delay for unicast traffic using LET for low traffic loads under these assumptions can be written as:

$$D_{LET} = \sum_{i \in V} \frac{\Lambda_i}{N(N-1)} \frac{T_L}{2h_i}. \quad (4.1)$$

For a schedule fully compensated for unicast traffic $h_i = \Lambda_i$, this results in,

$$D_{LET} = \frac{NT_L}{2N(N-1)}. \quad (4.2)$$

The ratio between network delay of link assignment and network delay of LET for unicast traffic can be written as

$$\frac{D_L}{D_{LET}} = \frac{M}{N}. \quad (4.2)$$
and the ratio between network delay of node assignment and network delay of LET can be written as

$$\frac{D_N}{D_{LET}} = \frac{T_N}{T_L} = \frac{\lambda_L^*}{\lambda_N^*}. \quad (4.3)$$

### 4.4 Evaluation and Results

In this section we compare the delays and maximum throughput for LET with node assignment and link assignment. The results for network delay are generated by network simulation. No comparison for maximum throughput in the case of unicast traffic is included since LET achieves the same maximum throughput as link assignment. For broadcast traffic such a comparison is included, using simulations to determine the maximum throughput for LET.

Furthermore, we compare these simulated results with the approximative results for unicast traffic given in section 4.3.

In the simulations of LET we have the extra simulation assumption that the link ID is random, as described in section 4.2.

#### 4.4.1 Unicast Traffic

The first comparison is node assignment and LET. Therefore, the parameter studied is the ratio between network delay of node assignment and network delay of LET at low traffic loads, i.e.

$$D_N / D_{LET}.$$ 

If this parameter is greater than one, LET is always preferable. If it is less than one, node assignment is preferable for low traffic loads.

In Figure 4.3, this parameter can be studied for networks of size 20 nodes. As can be seen there are some variations over the different networks. In Figure 4.4 we plot the ratio $D_{LET} / D_N$ averaged over connectivity for networks of different sizes. As can be seen, $D_{LET} / D_N$ decreases with the connectivity. This is because the gain in spatial reuse of link assignment compared with node assignment decreases with connectivity. Its increase with network size is consistent with the approximation in equation (4.3), since it should be close to $\lambda_L / \lambda_N$. But it should also be noted that the estimated standard deviation of the average ratio is on about the same order as the difference between the different network sizes. It is here on the order of 0.1 for the low values and less than 0.03 above a connectivity of 0.4.
4.4. Evaluation and Results

Figure 4.3: The figure shows the ratio between the delay of node assignment and the delay of LET. The ratio is plotted for 500 networks of size 20 nodes.

Figure 4.4: The figure shows the average ratio between delay for node assignment and delay for LET for networks of different connectivity. The ratio between delay for networks of different size: 10, 20, and 40 nodes.

It can be concluded that for the chosen assignment algorithms, except for high connectivity, a link-assigned schedule with extended transmission rights gives lower delay than a node-assigned schedule. For very high connectivity,
LET can give a higher delay than node assignment because the link-assigned schedule our method is based on does not attempt to spread the time slots a node is assigned evenly over the frame.

We conclude the study of network delay for unicast traffic by examining the ratio between network delay of link assignment and network delay of LET at low traffic loads $D_{LET}/D_L$. In Figure 4.5, this ratio is averaged for networks of different sizes. It can be concluded that LET decreases the network delay considerably compared with link assignment. This effect increases with network size and connectivity. This is not a surprising result, since increasing network size or connectivity increases the number of outgoing links of a network node, thereby giving LET more opportunities. The estimated standard deviation of the average ratio is here less than 0.5 for all values.

From these simulations and the knowledge that LET always achieves at least as high maximum throughput as link assignment, we can see that for unicast traffic LET is preferable to both link and node assignment except for networks of very high connectivity and low traffic.

Figure 4.5: The figure shows the average ratio between delay for link assignment and delay for LET for networks of different connectivity. The ratio between delay for networks of different size; 10, 20, and 40 nodes.
4.4. Evaluation and Results

4.4.2 Broadcast Traffic

We will now study LET in the case of broadcast traffic. In figure 4.6 we plot the ratio between delay for LET and the delay for node assignment. As can be seen there is a considerable variation over the different networks. However, for all except the highest connectivities, LET achieves the lower delay.

In Figure 4.7, this ratio is averaged for networks of different sizes. It can be concluded that for high connectivity, LET achieves approximately the same result as node assignment with small dependence on network size. However, for low connectivity we see a considerable improvement when using LET for increasing network size. Only the results for the low connectivity have a reasonably low estimated standard deviation of the average ratio though. For connectivity values above 0.5 it is around 0.1, giving a rather uncertain result. For low connectivity it is around 0.05.

We conclude the investigation of broadcast traffic by returning to the case of networks at high traffic loads. The parameter studied then is the ratio between maximum throughput $\lambda_{\text{LET}}^*$ of LET and maximum throughput $\lambda_N^*$ of node assignment, i.e.

$$\frac{\lambda_{\text{LET}}^*}{\lambda_N^*}.$$ 

This ratio can be studied for networks of size 20 nodes in Figure 4.8. Ex-
except for very low connectivity, in most networks node assignment gives considerably much higher maximum throughput than LET; a factor two or more. It is interesting to see that for very low connectivity we have networks where LET outperforms node assignment.

We can study this parameter averaged over size in Figure 4.9 and see that this effect increases with the network size.

### 4.5 Concluding remarks

In this chapter we have presented a novel assignment strategy (LET), which is based on link assignment. We have shown that for unicast traffic LET is preferable to both link and node assignment except for networks of very high connectivity and low traffic.

For broadcast traffic we see that LET outperforms node assignment for low traffic arrival rates. However, this comes at a considerable cost in terms of maximum throughput. One exception is low connectivity, where LET can give the same maximum throughput as node assignment.

We can therefore conclude that LET is very useful for low connectivity networks independent of the traffic type. For a higher network connectivity we
4.5. Concluding remarks

Figure 4.8: The figure shows the ratio between maximum throughput for LET and maximum throughput for node assignment for networks of different connectivity. This ratio is plotted for 500 networks of size 20 nodes with broadcast traffic.

have to consider what kind of traffic the network is designed to handle when choosing the assignment strategy.

We will not look further into LET in this thesis, but as seen in this chapter it can be used for improving all link schedules including those generated by the distributed algorithms discussed in the later chapters.
Figure 4.9: The figure shows the ratio between maximum throughput for LET and maximum throughput for node assignment for networks of different connectivity. This is plotted for networks of different size 10, 20, and 40 nodes with broadcast traffic.
Chapter 5

Model Comparison

In this chapter we investigate the loss of capacity (in terms of throughput) when the STDMA algorithm has limited information about the network. We will still use a centralized algorithm to generate the schedules but limit the network information when we decide which links can transmit simultaneously. We will start by describing the traditional graph model and continue with more advanced models, with more information.

A somewhat different approach to solving this problem have been taken in [12]. In which they use a truncated graph model to give probabilistic guarantees for the maximum throughput by bounding the maximum number of simultaneous transmitting units.

5.1 Graph-based Network Model

The traditional approach of designing reuse schedules is to use a graph model of the network. A graph consists of a set of nodes and a set of edges connecting pairs of nodes. In a graph representation of a radio network, each radio terminal is represented by a node, and the set of edges represents the pair of nodes that can communicate directly, i.e. radio links. In the simplest form of scheduling it is assumed that the lack of an edge means that two nodes cannot affect each other even as interferences. In such scheduling, a node can be assigned to send in a time slot if it has no edge to a node assigned to receive, and a node can be assigned to receive if no node at one-hop distance is assigned to send.

This method can easily be expanded to distances at more that one hop. For example, one way to do this is to assume that all nodes at exactly two hops...
distance are also creating too much interference for simultaneous transmission. This means that a node cannot be assigned to receive on a link in a time slot if any node at distance one or two is scheduled to send (and the opposite).

This can be generalized to all nodes at $k$ hops distance, and we will denote the method as $k$-hop graph scheduling. This method has sometimes been used in USAP [21]. The traditional way of scheduling will be the 1-hop graph scheduling.

By using a $k$ larger than 1, we include most of the nodes that cause the most interference with the additional benefit that we do not have to determine exactly which nodes they are. A drawback of this method is that it may overestimate the interference caused by neighbors further away, thereby causing fewer assigned time slots than would otherwise be possible. We may also have interferences not considered from nodes further away than $k$ hops.

In the graph-based method, a graph representation $G_{\gamma}$ is chosen. To represent the radio network as a directed graph, we denote by $G_{\gamma}$ the directed graph that is obtained by defining the set of nodes $V$ as vertices and the set of edges $E$ as follows

$$ (i, j) \in E \text{ if and only if } \Gamma_{ij} \geq \gamma, $$

i.e. the set of edges is the set of node pairs with SNR not smaller than $\gamma$.

The schedule is then designed from the graph $G_{\gamma}$. Interferences from other nodes are not taken into account. The traditional method for node assignment given a set of edges is to say that two nodes $v_i$ and $v_j$ can use the same time slot if and only if:

- edge $(i, j) \notin E$ and edge $(j, i) \notin E$, and
- there is no $v_k$ such that $(j, k) \in E$ and $(i, k) \in E$

The first criterion is based on a node not being able to receive and transmit simultaneously in the same slot. The second criterion is that a node cannot receive a packet from more than one node in the same slot. These criteria translate to the condition that nodes must be at least two hops away from each other in order to be scheduled to the same slot. For a more precise description of this problem see [8].

Observe that the above criteria are not sufficient to guarantee that the assignment is conflict-free in terms of SIR, as defined in chapter 2. The assignments that fulfill the above two criteria do not necessarily fulfill the SIR criterion given in condition (3.2). They may therefore not be able to transmit simultaneously...
5.1. Graph-based Network Model

Figure 5.1: The graph $G_\gamma$ obtained for $\gamma = 13$ dB of a small sample network consisting of ten nodes.

with the full data rate according to our definitions. We illustrate this with a small example.

In Figure 5.1 we see the edges obtained for a sample network by choosing the threshold $\gamma_C$ to be 13 dB.

Now, assume that links (2,4), (7,5) and (8,9) have been assigned the same time slot. This is possible according to the graph model of the network. If all of these nodes transmit at the same time, then the SIR calculated at node 5 will be only 1.6 dB. This is because the SNR between nodes 5 and 8 is just below what is needed for communication and the SNR between 5 and 7 is just above.

From the example we see that the graph approach if applied as above can result in serious interferences and thereby needs for a considerable decrease in data rate.

### 5.1.1 Two-level-graph-based scheduling

A more complex network model uses a two-level graph model, see e.g. [10]. Here interference edges are added, meaning that there is not sufficient signal power to receive the packet without error, but the signal power is strong enough
to interfere with reception from other users. This model gives a better description of the network, but it can be difficult to determine which interference edges exist.

The two-level graph model handles the interference more explicitly by basing the schedule on a graph in which node pairs with SNR less than $\gamma_C$ are included as interference edges [67]. The edges with SNR lower than $\gamma_C$ will not be assigned time slots, but only be used in the test criterion. By considering a graph $G_{\gamma}$ and letting $\gamma$ take a value $\gamma_I$ smaller than $\gamma_C$, the set of edges will contain not only the links but also interference edges, which represent the case when the signal from one user is too weak to be used for communication but is still strong enough to interfere. We will call $\gamma_I$ the *interference threshold*.

However, all transmissions in the time slot will add to the interference. The choice of $\gamma_I$ thereby determines the remaining interference. By choosing the threshold for a communication link, $\gamma_C$, slightly greater than what is needed for reliable communication, $\gamma_R$, we assure that the communicating link can handle these remaining interferences.

This two-level graph model can be used to create interference-free schedules. However, the two-level graph-based scheduling must still be done in a more careful manner than interference-based scheduling, since it has less information and thereby still needs margins to generate conflict-free schedules. The graphs $G_{\gamma_C}$ and $G_{\gamma_I}$ with a properly chosen $\gamma_C$ and $\gamma_I$ can now be used to generate a reuse schedule with any assignment algorithm taking a graph as a network model. In [44] we investigated this with $\gamma_I$ set to 1 thereby including interference from a node if it causes more interference than the noise in the receiver.

Here we will study how good schedules we can get with this method if we can choose the optimal value of $\gamma_I$.

### 5.2 Results

In figures 5.2 - 5.5, we plot the ratio between maximum throughput for the graph-based methods and maximum throughput for interference-based scheduling. This ratio is averaged over 1000 networks of different connectivity of size 10, 20, 40 and 60 nodes, respectively.

We define connectivity as the fraction of nodes in the network that can be reached by a node, in one hop, on average, i.e. $M/(N(N-1))$, where $M$ is the number of directed links in the network.
5.2. Results

Figure 5.2: The figure shows the ratio between the maximum throughput for SIR-based schedules and the maximum throughput for the different graph-based schedules. The ratio is plotted for 1000 networks of size 10 nodes for different connectivity.

Figure 5.3: The figure shows the ratio between the maximum throughput for SIR-based schedules and the maximum throughput for the different graph-based schedules. The ratio is plotted for 1000 networks of size 20 nodes for different connectivity.
Figure 5.4: The figure shows the ratio between the maximum throughput for SIR-based schedules and the maximum throughput for the different graph-based schedules. The ratio is plotted for 1000 networks of size 40 nodes for different connectivity.

Figure 5.5: The figure shows the ratio between the maximum throughput for SIR-based schedules and the maximum throughput for the different graph-based schedules. The ratio is plotted for 1000 networks of size 60 nodes for different connectivity.
5.3. Concluding remarks

It can be seen that the difference between all graph-based scheduling methods and interference-based scheduling increases with networks size. Two-level graph based scheduling with an optimal choice of $\gamma_I$ performs best of the graph-based methods especially for low connectivities and high traffic loads.

However, how to choose the optimal value is not so simple. The figures also show that the choice from [44], i.e. $\gamma_I = 1$ does not perform especially well for large networks, and a lower value is preferable. K-hop graph scheduling is a simpler method.

We can see that the usual 1-hop graph scheduling performs very badly compared to interference-based scheduling. The ignoring of interference levels close to being useful for communication really causes significant drops of capacity. Using higher values of $k$, e.g. 2 and 3 hops performs better, but we see that $k = \infty$, i.e. traffic sensitive TDMA performs best of the k-hop graph scheduling methods for all network sizes. This might seem a little strange, but the cause of this is the variable data rate combined with traffic sensitivity. The problem is that both 2-hop and 3-hop scheduling is not sufficient to avoid a SIR lower than $\gamma_R$, thus resulting in a lowered data rate. In worst case this afflicts all time slots that are given to a link with considerable reduction of maximum throughput as a result. This reduction cannot be estimated until the schedule is used so the algorithm cannot take this into consideration.

Without significant changes to the algorithm and increase what information that needs to be transferred this is difficult to do something about. However, this might actually be one problem that is easier to solve for a distributed algorithm. The estimated standard deviation of the average ratio is here around 0.02 for the k-hop graph scheduling except for the lowest values of connectivity where it is slightly higher. For the two-level models it is less, around 0.01 for all values.

5.3 Concluding remarks

In this chapter we have shown that the choice of underlying network model can have significant effect on network throughput. For highly connected networks the choice of graph model is not so important with the exception of the traditional method ($k = 1$) which gives very poor results unless the network is fully connected. For low connectivity the difference between the methods are significant.

The k-hop graph model can be rather useful for $k = 2$ or 3 but there are traffic sensitivity problems resulting in that traffic adaptive TDMA is better.
(It should be noted that traffic sensitivity is not simple to add which means that such a TDMA schedule is not necessarily easy to create). Results could possibly be approved with better traffic sensitivity for $k = 2$ and $3$ especially for large networks.

The two-level model usually gives good result if appropriate thresholds can be found, this however is not always that easy. The simple choice from [44] do not perform well for large networks. It would be interesting to see whether knowledge about network size would be sufficient to determine a good threshold.

However, the full interference model is always best especially for large network size and low connectivity when the gain is highest.

Notice, that for a low loaded network, the data rate may not always need to be decreased at which case the simpler methods may still behave as they should. But we are concentrating on high traffic loads here.
Chapter 6

Effect of Frame Length– Joint Node and Link Assignment

In the previous chapters we have used a variable frame length, chosen with sufficient length to obtain full traffic adaptivity. However, for large networks this is seldom possible, the overhead cost for the handling of the time slots will simply be too high. Furthermore, a variable frame length will create problems for a distributed algorithm since the frame length may change for each update, which in turn will force a global update.

In this chapter, we study what effect different frame length have on the result, i.e. the interplay between traffic sensitivity and frame length. In addition we will present an additional assignment strategy–Joint Node and Link Assignment.

However, we will only study this for unicast traffic.

6.1 A centralized algorithm

In order to create a schedule with a fixed frame length we will here describe a centralized algorithm that can do that from a given length $T$. This algorithm is very similar to the distributed algorithm that will be presented later.

Central also to this algorithm is the concept of priority values and priority. The priority value of a link set $x_i$ is defined as the number of time-slots allocated to the link set $h^x_i$, divided by the traffic load on the link set $\Lambda^x_i$. These priority values are closely connected to the maximum throughput of the network, see equation 3.5. This means that the link set that limits the maximum throughput
will be the ones that are tested first in each time slot.

This choice of priority value is similar to the saturation value used in [38], which uses an average between $\frac{h_i + 1}{\Lambda_i}$ and the value if the link receives another time slot when deciding which link to allocate next. This measure tries to maximize fairness instead of maximizing the maximum throughput. If a link with low traffic load limit the maximum throughput it may not receive an extra time slot since this would give it too much capacity as seen from a fairness measure.

The following algorithm generates a schedule with length $T$:

**Step 1** Initialize:

1.1 Enumerate the link sets
1.2 Create a list, $A$, containing all of the link sets.
1.3 Set $t$ to zero.

**Step 2** Repeat until $t = T$:

Step 2.1 Set $t ← t + 1$ and $Y_t ← \emptyset$.

Step 2.2 For each link set $x_i$ in list $A$:

2.2.1 Set $Y_t ← Y_t \cup x_i$.
2.2.2 If the link sets in $Y_t$ can transmit simultaneously:
    • Set $h_i ← h_i + 1$.
2.2.3 If the set of link sets in $Y_t$ cannot transmit simultaneously, set
    $$Y_t ← Y_t \setminus x_i.$$  

Step 2.3 Reorder list $A$ according to link set priority,

$$\theta_i = \frac{h_i}{\Lambda_i^x},$$  

lowest value first.

### 6.2 Results for Node And Link Assignment

In Figure 6.1 we plot the maximum throughput for schedules generated with different frame lengths for a 30 node network with 144 links as a function of the frame length generated with the above algorithm. This is done for both node assignment and link assignment. As can be seen the choice of frame length
6.2. Results for Node And Link Assignment

have a large impact on the performance in this case. If the frame length is not sufficient to give all nodes/links a time slot we will have zero maximum throughput, due to the definition of uniform capacity in Chapter 2.

Once all nodes/links have a time slot the maximum throughput increases to $1/\Lambda_{\text{max}} \times N(N - 1)/T$, and from there it increases irregularly depending on whether or not the nodes/links not assigned in the last added slot now have lower maximum throughput than the previous minimum value due to the increase in frame length.

That is, only links/nodes assigned in the last slot will increase their maximum throughput all the rest will decrease since the frame length increases. This may in some cases decrease the throughput. This is especially noticeable in a fully connected network, Figure 6.2, where all nodes have the same traffic load. Maximum throughput will decrease until all links/nodes have yet another slot. In this case we also see that node assignment reaches a reasonable maximum throughput with much shorter frame length than link assignment, the very large number of links in this network requires very many time slots.

In Figure 6.3 we plot the average maximum throughput for schedules with different frame lengths divided with the highest value of maximum throughput (for link assignment) averaged over 250 networks of different sizes. The left one is for low connectivity and the right one is for medium connectivity. This is done for both node and link assignment.
As for the example networks, we see that node assignment gets close to its highest value of maximum throughput much earlier than link assignment. This is because there are much fewer nodes than links and traffic adaptivity is thereby simpler. A consequence of this is that unless we have a sufficient frame length available, link assignment does not even have higher maximum throughput than node assignment for unicast traffic.

Similar to the results in chapter 3 we can also see that the difference between node and link assignment increases with network size. For a 10-node network, the frame length required to get the same maximum throughput with both methods is close to one and a half times the number of links for both low and medium connectivity. For 60 nodes, just above one half is sufficient. This means that the required frame length does not increase as fast as the number of links when the network increases in size. It may, however, do so when it only increases in connectivity.

One reason why we study variable frame length here is that we want to choose a proper frame length for distributed STDMA. As we can see, there is a knee in the maximum throughput curve, below this, there is a considerable loss in maximum throughput due to not having sufficient traffic sensitivity, above, there is very little positive effect of choosing a longer frame length. When STDMA schedules are created in a distributed manner, the overhead traffic increases with the frame length. This occurs because the same set of scheduling
6.3 Joint Node-Link Assignment

As can be seen so far node assignment performs well for short frame lengths and link assignment performs well for long frame lengths. It would however be desirable to use an assignment strategy that performs well at all frame lengths, behaving like node assignment for small frame lengths and like link assignment for large frame lengths. We propose such a strategy in this section.

The central idea behind joint node-link assignment is to assign one time-slot to each node, as a node-assigned slot. The maximum throughput of this node-assigned slot is shared among all the outgoing links of the node. Each node gets exactly one node-assigned slot, while the remaining time-slots are assigned to

operations are performed in each time-slot in the frame. While the \textit{precise} rate of increase would vary with the scheduling algorithm employed, it is clear that the overhead would increase with frame length.

In addition, using a distributed algorithm, we could have different frame lengths in different parts of the network, depending on how the networks looks locally around a node. Information on how maximum throughput depends on frame length could be useful in order to determine what a local frame length needs to be (and when it should change).

6.3 Joint Node-Link Assignment

As can be seen so far node assignment performs well for short frame lengths and link assignment performs well for long frame lengths. It would however be desirable to use an assignment strategy that performs well at all frame lengths, behaving like node assignment for small frame lengths and like link assignment for large frame lengths. We propose such a strategy in this section.

The central idea behind joint node-link assignment is to assign one time-slot to each node, as a node-assigned slot. The maximum throughput of this node-assigned slot is shared among all the outgoing links of the node. Each node gets exactly one node-assigned slot, while the remaining time-slots are assigned to
6.3.1 Sharing the Capacity of the Node-Assigned Slot

Once a node has been assigned in node-assigned mode, its capacity can be assumed to be shared among all outgoing links of that node. We illustrate this concept in Figure 6.4. Node $A$ has outgoing links $L_i$ with traffic loads $\Lambda_i$ and allocated time-slots $h_i, i = 1 \ldots 4$.

Then the priority values without considering the node-assigned slot are $\theta_1 = \frac{1}{7}, \theta_2 = \frac{1}{3}, \theta_3 = \frac{1}{4}$ and $\theta_4 = \frac{1}{5}$. If a node-assigned slot has been allocated to $A$, its capacity is shared among the outgoing links $L_i$ emanating from $A$. The capacity sharing takes the form of waterfilling as illustrated in Figure 6.4.

The waterfilling is performed in two dimensions, first across all the flows.
6.3. Joint Node-Link Assignment

of a link, and then across links. Note that the procedure leads to modified priority values for some of the links. We denote the priority of a link \((i, j)\) after waterfilling by \(\theta_w^{ij}\).

6.3.2 Handling Links with Equal Priority values

Consider a node \(v_i\) with a set of outgoing links \(L_O\). Let us assume that \(v_i\) has been allocated a node-assigned slot. Further, the set of links \(L^*_O \subseteq L_O\) have the same priority. In this case, we must decide the order in which the links should be tested. This situation occurs when a node-assigned slot has been allocated to \(v_i\), and the waterfilling operation results in equal priority values for the links in \(L^*_O\). Note that links in \(L^*_O\) may have different traffic loads. This situation is depicted in Figure 6.5 for the particular case of \(\theta_i = 0 \forall i \in L^*_O\).

We illustrate the drawback associated with an incorrect testing order with the above example. Node A has three outgoing links. In Figure 6.6(a), (1) is tested first, followed by (2) and (3). Assuming that (1) can be successfully assigned in this time-slot, the slots per flow achieved by each link are shown. On the other hand, in Figure 6.6(b), (3) is tested first, followed by (1) and (2). Once again, the slots per flow achieved by each link are as shown.

We note that the testing order corresponding to Figure 6.6(b) leads to better results at the end of the current time-slot. Intuitively, we would like to give precedence to the link which maximizes the minimum slots per flow among links.
in $L_O^*$. Thus we would like our testing order to reflect the above intuition. An exhaustive method of ordering the links in $L_O^*$ is to assume that the slot has been allocated to each element of $L_O^*$ in turn, waterfill with the capacity of the node assigned slot, and compute the minimum slots per flow. The links in $L_O^*$ should then be tested in the decreasing order of this metric.

However, it can be easily be realized that the above is equivalent to giving precedence to that link which has the minimum slots per flow, assuming it has been allocated the current slot, since this link would otherwise be the one using up most of the waterfilling node slot. This leads to a much simpler implementation wherein we compute modified $\theta$ values for links in $L_O^*$, denoted by $\theta^o$ to reflect this optimization, as $\theta^o_{ij} = \frac{h_{ij} + 1}{\Lambda_{ij}}$.

Thus the priority list is first sorted on the basis of the original priority values $\theta_{ij}$. Next, the links with equal priority values having the same transmitting node are re-sorted in increasing order of $\theta^o_{ij}$.

### 6.4 Some results on this

In Figure 6.7 we now study our joint node-link assignment strategy for low connectivity network. On the left is 250 20-node networks and on the right are 250 60-node networks. We note that the joint node-link assignment algorithm
6.4. Some results on this

matches the performance of node assignment at small frame lengths, and that of link assignment at large frame lengths. For intermediate frame lengths, it performs better than both node and link assignment. For low connectivity networks, this assignment strategy performs well at all frame lengths. The new strategy even performs better than link assignment for long frame lengths, only when the frame length is very very long will link assignment catch up. The waterfilling with only one node slot gives here a considerably improvement regarding traffic sensitivity.

Notice that this means that the frame length required to reach a certain maximum throughput can be considerably decreased as compared with link assignment, a reduction in half is common.

For medium connectivity plotted in figures 6.8 the result is not that good anymore. We can see that we get maximum throughput above zero as quickly as for node assignment but after this there is a region where node assignment is better until the higher spatial reuse of link assignment gives the joint node-link algorithm the advantage. Our new strategy is still considerably much better than link assignment though. The problem for medium connectivity, that will become even worse for high connectivity, is that one node slot simply is not enough to do an efficient water filling. A possible expansion would be to allow more than one node assigned slot to each node, this however will we leave to
Figure 6.8: Average maximum throughput compared to the highest maximum throughput for different frame lengths for networks of medium connectivity, using all three assignment strategies. 20-node networks to the left 60-node network to the right.

future work.

### 6.5 Some conclusions

In this chapter we have briefly studied the effect of limiting the frame length. As was seen, with a limited frame length some of the results from the previous chapters are changed. Node assignment can give good results with a shorter frame length than link assignment. Only for longer frame lengths do we see the results from Chapter 3.

We have also devised a new strategy, joint node-link assignment, by giving each node a single node-assigned time slot. For low connectivity, this strategy outperforms node and link assignment for all frame lengths. For medium connectivity and higher, it cannot fully compete with node assignment for shorter frame lengths but it still considerably outperforms link assignment.

This new strategy will allow us to use shorter frame lengths, which will be an advantage in reducing unnecessary overhead for distributed algorithms that need to negotiate for each time slot.
Chapter 7

Distributed Information – How to schedule Efficient?

All work so far have assumed a centralized scheduler, in practice this is unwanted, by the time new information have been collected into a central unit it may be obsolete, especially for a large network.

In addition, a centralized solution is not especially robust against node failures. If the central unit is lost we may end up in serious problem, even if we have a backup node that can take over the scheduling. Therefore, we will now study distributed scheduling, and see whether results from centralized scheduling can be used also when we design distributed algorithms.

We will start this chapter by giving a list of the properties we seek in a distributed algorithm. We will also discuss whether existing algorithms for distributed STDMA scheduling can handle these properties.

7.1 Wanted Properties

Although, many distributed STDMA algorithms are either node or link assigned, commonly referred to node and link activation, many algorithm descriptions handle both of them. There is really little difference when we design a distributed algorithm — if we can design a distributed algorithm for one of the methods, it is not so much work to do the same for the other.

For distributed scheduling we will only concentrate on link assignment.

In the following we list some of the desired properties of a distributed STDMA algorithm. The first five of these are specific for distributed schedul-
The last three properties are general for all STDMA scheduling but they are so relevant for performance so we repeat them here.

1. *No central control; the algorithm is run in parallel in every node in the network.* This is necessary if we want a robust system that can handle the loss of any node and is the basic meaning of the term “distributed”.

2. *Only local information is exchanged and needed.* As the other cornerstone of the term “distributed”, the information propagation must be limited. However, we do not make any specific definition of the term “local”, except that global information about the network is not needed.

3. *Local adaptation to topological and traffic changes must be possible.* (Ripples are permitted if the probability of updates decreases with distance from the change.) This is an addition to the previous two assumptions that prevent “unstable” algorithms.

4. *The algorithm should be able to efficiently handle large changes in the number of nodes and density of the network.* (By “efficiently” we mean that it should not just be able to create a valid schedule but also perform close to the results of a centralized algorithm in a number of very different scenarios). In particular, changes in the network density will be usual in military scenarios and situations where all nodes are gathered at one place (with maximum density as a result) will occur.

5. *The algorithm should adapt to the level of mobility.* In relatively static networks, we can get a very good picture of the situation, e.g. precise path losses and power levels which could be used to make a more efficient schedule. In a high mobility network the only information that may be possible to transmit might be the existence of neighbors. The algorithm should perform well under these circumstances too. The radio channel is the scarce resource in our network and should be used as efficiently as possible.

6. *Adaptivity to traffic; the algorithm should be able to adapt to the different needs of the different links.* There is considerable variation of traffic over the different links of the network due to the relaying of traffic in multi-hop networks. An STDMA algorithm must adapt to this in order to be efficient [30]. Furthermore, one of the main advantages of STDMA is its ability
to provide QoS. In order to go beyond *one slot per frame* guarantees (or even reach this for a single flow), traffic adaptivity is a must.

**7. Using an interference-based network model.** The graph-based network model is currently the most used network model for ad hoc networks. However, this model does not reflect reality sufficiently well in many of our scenarios. As was shown in chapter 5, graph-based scheduling do not perform very well compared to interference-based scheduling.

**8. The algorithm should handle (exploit) heterogeneous nodes in the network.** Some nodes may have more advanced abilities than others, e.g. adaptive antennas, able to use variable data rate, lower noise levels and similar abilities. Moreover, a node might have the same radio as the rest of the nodes but have other properties, e.g. in terms of mobility. A helicopter is one example of this kind of node, another is a node that is known to be immobile.

The first three of these properties are handled by all algorithms that claim to be distributed, although point three is difficult to assess without actual simulations. For the rest of the properties there are considerable variations.

Many algorithms need at least information about the number of nodes and node density. Actually, several papers, see e.g. [17, 32], prove that their algorithm can create schedules with length dependent on the maximum density of the network. However, unless methods not mentioned in the papers are added, this gives a fixed frame length throughout the network. This means that a change in the number of nodes or an increase in network density might force a global update of the network. This will probably not happen as often as normal updates, but it is still highly undesired. To avoid such global updates these parameters must be chosen to be worst case, which usually results in unnecessary complexity and inefficiency. Therefore, fixing the length of the schedule is a simple solution, but it does not handle the situation of varying the number of nodes in the network very well. If the number of nodes increases and the network density gets high (many neighboring nodes), the fixed frame length may not be sufficient. If the number of nodes decreases, this may result in an unnecessary long schedule with its added load of complexity. In many algorithms, see for example [15, 17, 32], we end up with a schedule of fixed length (or less than this length) depending on the network density and size.

However, there are a better approach, that we could call the $2^n$ principle, used in [36, 20, 16]. The frame length is varied in different parts of the network.
(and also over time). The frame length as seen by a node or a link is of length $2^n$. If the density increases in a local area, $n$ can be increased by one in this area. This does not cause any changes outside this local area. If all nodes in the area can receive their slots in the first half of the frame, the schedule can be decreased.

It is difficult to say how well this works and how efficient schedules are created if we have large changes in the number of nodes and/or density, but it is a method worth investigating further.

As mentioned in chapter 1, traffic control is only rarely included in the distributed algorithms. Although some algorithms include the ability to assign more than one time slot to a link or node, see e.g. [36, 37], a more specific description of how and when some of the links receive extra time slots is usually omitted.

Most distributed STDMA algorithms assume a very simple graph model, which can give poor results, see chapter 5, and usually they are not easily expanded to an interference-based model. Many can be expanded to a two-level graph model, but this gives a considerable decrease in maximum throughput compared with an interference-based model. One exception to the use of graph-based scheduling is [68], but this paper investigates extremely large systems and does not use time slotted systems.

The existing distributed algorithms are generally designed for the high mobility case, and the graph model does not convey sufficient network information for efficient scheduling in a more stable scenario. For high mobility, though, it is probably not possible to convey more than this information, which suggests that the accuracy of network information should decrease with increasing network mobility. How much information should be conveyed to the local neighborhood for a specific mobility rate (as compared to the link data rate) is still an open issue. For very high mobility we might end up with the same information as a graph model gives.

A lot of work has been done on distributed STDMA, but no existing algorithm fulfills all our required properties. Of the existing algorithms, USAP is probably the most interesting for military communications but not even this algorithm has all the listed properties that are desired for reliable and efficient communication on the battlefield.

However, although none of these algorithms can fulfill our properties they have functions that will be useful when designing such an algorithm.

All distributed algorithms have been designed with the purpose of giving an acceptable solution, rather than a solution that handles the channel as efficiently
as possible under different situations. A more systematic approach to the design of distributed STDMA algorithms is lacking. What is an efficient schedule in a specific scenario? How is such a schedule created? Exactly which information is needed and how much in each case?

We know that the more information about the network we have the better schedules we can create, thereby increasing the total maximum throughput of the network. However, increasing information also increases the overhead. This means that the amount of information the algorithm has about the network should vary depending on the situation.

The use of interference-based scheduling can give us the means to vary the amount of information. So far no distributed STDMA algorithms have tried to use this.

In the rest of the chapter we will describe our proposal to an interference-based STDMA algorithm that creates an efficient schedule with a given amount of information about the network.

In this chapter our algorithm does not care how it receives the information, but simply acts on the information it has received. The purpose of this is to investigate how efficiently we can use distributed information. In the next chapter we will study how to convey the required information.

### 7.2 An Interference-Based Distributed Algorithm

In short, the our distributed STDMA algorithm can be described by the following steps:

- Nodes that have entered the network exchange local information with their neighbors.

- The link with highest priority in its local surroundings assigns itself a time slot (this is done in the receiver).

- The local schedule is then updated, and a new link has highest priority. This process is then continued until all slots are occupied.

We will include traffic sensitivity through the link priorities, i.e. a link that needs many time slots will have high priority more often than a link with low priority.

The algorithm is interference-based since it uses interference information, i.e. interference information is transmitted and used when the algorithm decide
when links can transmit simultaneously. We will use the term *local neighborhood* of a link \((i, j)\) to mean those links that will be taken into consideration when a link determines whether it can transmit simultaneously with all other assigned links. Links outside the local neighborhood will not be considered and therefore no information about these links is assumed. Exactly how large this neighborhood is will be discussed later.

In the following we will assume that each link has a given schedule length \(T\). This length is not necessarily the same length in all parts of the network and may change over time. But this will not change the basic scheduling process.

### 7.2.1 Link States

The STDMA algorithm is run in parallel for each link, i.e. each link can be seen as a separate process which will be run at the receiving node of the link, which means that each node will run a process per incoming link. These processes can be in three states: active, waiting, or asleep.

- **Active:** In this mode, the link has the highest priority in its local neighborhood and will subsequently assign itself a time slot. Which time slot is chosen if more than one is available will be discussed later. A link process can be in this mode because of the existence of unused slots or because the link’s share of the time slots in its local neighborhood is too low. In the latter case, it can steal time slots from another link. We will describe later under which situations this may be permitted. Information about which time slot is chosen and the link’s new priority will be transmitted to all nodes in the local neighborhood of the link. After this, the link process can stay in this mode or change into one of the others.

- **Waiting:** In this mode the links wants to assign itself a time slot, but another link has highest priority. The link will wait its turn. However, since time slots are taken by active users, the link may change into asleep mode instead if all time slots are taken and the link does not have the right to steal slots.

- **Asleep:** In this mode, there are no available slots for the link and it simply waits for a change of the network, either in topology or in traffic levels.

The purpose of the sleeping phase is that to let also nodes that are not highest prioritized allocate time slot. This will allow us to use the channel more efficiently. In the next chapter we look into this in more detail.
7.2. An Interference-Based Distributed Algorithm

7.2.2 Link Priority

Link priority decides in which order the links may attempt to assign themselves a time slot. The link priority can depend on many things, but the most important will be the number of time slots the link is assigned, $h_{ij}$, and the traffic of the link, $\Lambda_{ij}$. Since both these values are changing, the link priority is constantly changing.

The priority value of a link $(i, j)$ will be $h_{ij} \Lambda_{ij}$, where the lowest value has the highest priority. The motivation for this comes from the maximum throughput formula (3.5). The links with the highest priority (lowest value) will be the links which limit the maximum throughput of the network. The network throughput will not rise above zero until all links with traffic levels above zero have received at least one time slot. An obvious consequence of this is that all links with traffic in a local neighborhood will receive at least one time slot before any of the links receive more than one slot. This will be the case, for example, when a new schedule is initiated.

7.2.3 Theft of Time slots

Sometimes, the relative traffic levels will change in a local area (or other changes may take place). This will result in a situation where a link has a smaller proportion of the time slots than its priority value merits. If there are free slots, the link may assign itself slots until it is on a similar priority level as its surrounding links. However, if no time slots are free, the link can sometimes steal time slots from other nodes.

The policy for assigning time slots in the case of free time slots is always that the link which limits the maximum throughput will be the one that receives an extra time slot. This is also the case when a link is permitted to steal a time slot. When stealing a time slot, the local network throughput must increase.

This means that a link $(i, j)$ is only permitted to steal a time slot from another link $(k, l)$ if the priority value of the stealing link is lower than the other link’s priority value after the loss of a time slot, i.e.

$$\frac{h_{ij}}{\Lambda_{ij}} < \frac{h_{kl} - 1}{\Lambda_{kl}}.$$ 

The theft of time slots means that it is the link with too few time slots that reacts to unfairness situations. This is a better solution than if links that have too many time slots would give them up since such a link can never know if
the time slot can be used by the link with too few time slots. The time slot can always be blocked by another node further away.

In this form, the algorithm permits the theft of a time slot if there is any gain at all independent on how small. Since this always will result in additional overhead traffic we may want to limit the cases where theft is allowed to only cases where the maximum-throughput gain is worth the overhead cost.

One way of doing this is to include a Theft Threshold (TT) such that theft is allowed iff:

$$\frac{h_{ij}}{\Lambda_{ij}} \times TT < \frac{h_{kl} - 1}{\Lambda_{kl}},$$

where TT is greater than or equal to one. As we will see in the next chapter, the setting of this parameter can affect the maximum throughput and overhead. To start with we will use TT equal to one.

### 7.2.4 Choice of time slots

Sometimes when a node attempts to assign itself a time slot, it will have more than one to choose from. This will especially be the case when the schedule is first initiated. The main reason why the choice of time slot is important is the packet delay of the network. For example, if a link has received two time slots and these are spread in such a way that the distance between them is approximately half the frame length, then for low traffic loads the delay will be at most half the frame length. However, if the node is given two consecutive time slots, the maximum delay might be the entire frame length. That is, it is usually efficient to spread a node’s time slots evenly over the frame. This problem gets worse if nodes receive many time slots, especially since a large part of the traffic usually flows through these nodes. From this small example we can conclude that the choice of time slots can be important.

Note that this can also affect the links that will receive only a single time slot. The reason for this is that all links will receive one slot before any of them receive their second. If we use up all slots in the first half of the schedule, for instance, we will not have an efficient spreading of time slots. This can be alleviated by choosing a time slot at random when we choose the first time slot for a link. When the link already has time slots assigned to it, we choose a slot that is maximally spread from the others.

Another reason for this consideration can be the existence of weak links, i.e. links that can only handle very small amounts of interference. These links are difficult to assign to the same slot as any other link unless they are very
7.2. An Interference-Based Distributed Algorithm

far apart. Another property of these links is that they are often long, i.e. they carry traffic a long physical distance in one hop. The links will often be used by routing algorithms that do not take into consideration the capacity of the link, a property that describes virtually all existing ad hoc routing algorithms, thus resulting in heavy traffic loads. Due to the high traffic loads these links usually receive several time slots. Now, if all the stronger links have received a time slot each, unused by any other node, it might be difficult to give the weak links their extra slots. Therefore, it can be a good idea to try to assign strong links, those that can handle larger interference levels, to slots where the interference levels already are rather high (not too high of course) and leave the unused (low interference level) slots to weak links. However, this might be a complex procedure, so we will leave such an investigation for the future.

In addition to this we have the same problem when when a link wants to steal a time slot, if there are more than one available it can be an advantage to steal the time slot, where the resulting maximum throughput is highest. That is, we may only steal time slots if there is a gain in maximum throughput by doing it, but how large that gain is can be dependent on which time slot that is stolen. The links that lose a time slot may now be the limiting links instead of the stealing link for example. The choice of time slot can thereby be of importance.

For simplicity, and the fact that we only evaluate maximum throughput in this report, we have chosen the first available time slot when doing normal assignment in our simulations. For theft we have chosen the time slot whose assigned links (only those that we actually steal) have the highest priority value after they lose a time slot.

7.2.5 When do we have a re-scheduling?

We have so far described the different states of a link process and how the priority is calculated, but we have not been very detailed about the circumstances under when a new scheduling event take place. In the following, we give events that can cause changes in an ad hoc network and which consequences they have.

- *Increased interference level on a link due to mobility.* The link cannot use the slot and deallocates it. This has two consequences. First, $h_{ij}$ decreases for the link which decreases the priority value for the link. The link may now have a value so low that it is permitted to steal a time slot from another node. Second, since the link deallocates the time slot, another link may now have a free slot, which it may allocate itself.
• **Decreased interference level on a link due to mobility.** A link may have a free time slot that was previously blocked, which it will allocate to itself. This increases $h_{ij}$, which might result in another link stealing a time slot from the link.

• **A link breaks:** This stops the link process, and all assigned time slots will be deallocated. This will have similar consequences as when a link gives up a time slot due to interference, although this can happen to many time slots simultaneously, thus affecting more links. However, a link break can also have consequences for the routing, which affects $\Lambda_{ij}$. The priority values in the neighborhood, or even further away, may change. This may lead to significant schedule changes.

• **A link is created:** This creates a new link process (or restarts an earlier process). We make the assumption that $\Lambda_{ij}$ for the new link is set to a value greater than zero, which results in a priority value of zero. This means that the link will assign itself a time slot if one is available or otherwise steal one from a link with more than one time slot. A new link can also lead to a rerouting of the traffic, which may have consequences further away.

• **A new node is added to the network:** This can be seen as several links that are added simultaneously. It can also change traffic in the network.

• **A node disappears from the network:** This can be seen as the removal of several links at once. It can also change traffic in the network.

• **Rerouting or other traffic changes:** This can result from link failures, links created or changes in the input traffic to the network. This changes the priority of the links, which can cause time slots to be stolen.

Some other things can also be mentioned. First, with more complex links we may have a variable link data rate. In such a case $h_{ij}$ may not necessarily be an integer. Second, if a link has one of its time slots stolen, it sometimes has the opportunity to steal a time slot from a link further away from the link that initiated the theft. Third, it is also possible that the local frame length is not sufficiently long for as much traffic adaptivity as we would like, or even long enough to give all links a single time slot. In this case, the local frame length must be increased. However, we will leave how this is done and how we determine whether it is necessary for future work.
7.3 What information does a node need?

As previously mentioned, the interference-based model is included by which network information is transmitted and how this information is used for determining when links can transmit simultaneously. The local neighborhood of a link \((i, j)\) is those links that will be taken into consideration when the link determines whether it can transmit simultaneously with all other assigned links. Links outside the local neighborhood will not be considered and therefore no information about these links is assumed.

The remaining issue is exactly what information the algorithm needs in order to do the scheduling.

Two things must be fulfilled if a link can be allowed to transmit in a time slot.

First, the receiver must have sufficiently low level of interference from the assigned transmitters. This is calculated in the receiver.

Second, the interference from the transmitter is not allowed to cause interference problems in any of the other receivers already assigned time slots. This is calculated in the sender and then the information is sent to the receiver (since it does the actual scheduling) as a variable called \(AVTS\) (available time slots to send).

The problem can be illustrated by the example shown in Figure 7.1. In
this case links (1, 2), (3, 4), and (5, 6) are assigned to a particular time slot. Now, link (7, 8) wants to be assigned as well. This means that the received power in node $v_8$, must be sufficiently large compared to noise and the combined interference from nodes $v_1$, $v_3$, and $v_5$. Furthermore, the interference caused by node $v_7$ in the receiving nodes $v_2$, $v_4$, and $v_6$ must not be so large that any of these fall below the SIR threshold.

In order to achieve conflict-free scheduling on link $(i, j)$ the receiver $v_j$ need the following information (which also can be seen in picture 7.2):

**Interference - Received Power**

We need an estimate of the received power from each of the other transmitters, i.e. $P_k G(k, j)$. We assume that the channel is equal in both directions, that is

$$G(k, j) = G(j, k).$$

This assumption can then be used by the transmitter to determine whether its transmission will cause problems for somebody else.

If the received power level from a transmitter is below a value $\delta I$, it is assumed to be zero by the algorithm, i.e. the algorithm assume that such nodes do not affect one another. The size of this threshold is given by the interference threshold, i.e. $\gamma_I = \delta I / N_r$.

We assume that this information can be obtained by measurements on the channel and no specific information needs to be transmitted to handle this.
7.3. What information does a node need?

To simplify notation, we say that a pair of nodes \( v_i \) and \( v_j \) form a *interference link* if the signal-to-noise ratio (SNR) is less than \( \gamma_C \) but higher than \( \gamma_I \) in a similar way as in chapter 5.

**Local Schedule**

A node needs to know the local schedule and how much more interference can be handled by the assigned receivers \( I_{max}(l, \tau) \) in each time slot \( \tau \), i.e. the largest value of \( I_{max} \) such that the following inequality is still valid

\[
P_{k}G(k, l) \geq \gamma_R \cdot \left( N_{i} + I_{L}(k, l) + I_{max}(l, \tau) \right)
\]

This information \( (I_{max}(l, \tau)) \) is required for node \( v_i \) to be able to determine whether its transmission can be handled by the other, already assigned receivers. A link can be assigned the time slot if:

\[
P_{i}G(i, l) < I_{max}(l, \tau)
\]

for all assigned receivers \( v_j \) in time slot \( \tau \).

The local schedule is also used to determine whether the receiver \( v_j \) can handle all existing interference. Measurements of the channel in the specific time slot can be of help when doing this, but is not sufficient since node \( v_j \) cannot know whether all assigned transmitters \( v_k \) actually are using their time slot. Instead the actual SIR is calculated using the local schedule and received power levels. If the interference levels that are measured are higher than this estimate, we conclude that we have extra interference from outside the local neighborhood and the interference measured should be used while this lasts.

The local schedule \( S \) and all values of \( I_{max}(l, \tau) \) needs to be sent over the channel. However, only information about nodes that \( v_i \) can cause interference to or \( v_j \) can be interfered by is needed.

**Priorities**

A node needs to know when it should be active. It also needs to know if a node in the neighborhood is asleep, since such nodes are not considered in terms of priority. The exception to this is the case with theft of time slots, in which case sleeping nodes are also considered. The node needs such information from the entire local neighborhood since we cannot allow two nodes to be active simultaneously in the local neighborhood.
**Available Time Slots to Send (AVTS)**

The receiver also needs information from the transmitter about which time slots it can assign. This can be represented by a vector of the same length as the frame length or specific information about changes of a single time slot. Only information from the transmitter is needed by the receiver.

With this information the receiver has sufficient information to determine when link \((i, j)\) should be active and which time slots it can assign.

**Local neighborhood**

Due to mobility the local neighborhood will change constantly and we also need to have accurate information about what the local neighborhood is at the present moment.

### 7.3.1 Continues Variables

A problem with interference-based scheduling that does not exist for graph-based scheduling is that some of the parameters used are not discrete variables, an example of such a parameter is path-loss. Such variables may take any value and can change continuously. Path-loss need not to be sent but \(I_{\text{max}}\) is dependent on path-loss values and need to be sent.

In practice, also path-loss values and measured values of interference will be sampled, by lower layers, but this may be of too high rate change each such value to be sent to the local neighborhood.

To avoid this we can introduce another threshold \(I_{\text{max Threshold}} (IMT)\), that will be the minimum change of \(I_{\text{max}}\) that will cause an update.

### 7.3.2 Consequences of limited information

A problem with limiting the input information on the network, i.e. using the interference threshold, is that units from far away are not considered. Although these nodes create small interference in a node, they might still cause a problem since the nodes will try to schedule as many links as possible in each time slot. This means that the scheduled SIR will be very close to \(\gamma_R\) in many receivers. The additional interference from outside the local neighborhood can then be sufficient to lower the actual SIR below \(\gamma_R\). One way to avoid this is to use an interference margin \(I_{\text{margin}}\), i.e. we assume that there will be external
interference and therefore avoid scheduling the SIR as low as $\gamma_R$. The link is then assumed to be able to use the time slot if:

$$\frac{P_i G(i, j)}{(N_i + I_L(i, j) + \text{margin})} \geq \gamma_R.$$

However, this solution may not completely solve the problem. One reason is efficiency—in order to completely avoid the problem we have to choose a rather large margin, which means that the channel will be used poorly. A second reason is network connectivity: a high interference margin will prevent the use of weak links, which might cause the network to partition. Therefore, alternative solutions may be required.

The method we will use in this thesis is to decrease the data rate for those links that have received time slots with too low SIR. To do this we measure (or estimate) the actual SIR when the schedule is used and lower the data rate until reliable communication is possible.

This is doable to a certain limit. If the external interference is very high, outside our control, we might finally give up the time slot entirely, set the time slot as unavailable, and attempt to assign a new slot if this is allowed, and hope to achieve a better SIR.

Another problem with lowering the data rate was shown in chapter 2, that is traffic sensitivity. Remember that the priority of a link is based on how loaded the link is. This is dependent on how many time slots the link is assigned. If these slots do have a lower data rate than assumed when the time slot was assigned, it means that the link in practice has lower maximum throughput than it should. This effect will be seen in the simulations. However, this can be easily solved by assuming that $h_{ij}$ is not an integer and that it decreases when the data rate decreases. For example, if a link has three time slots allocated, two at full rate and one at half rate, $h_{ij}$ will be two and a half for this link.

In the following we assume that this is done. In the next chapter we will study the consequences of these methods.

### 7.4 Instantaneous Maximum Throughput

We will now study some cases where we run the algorithm. In Figure 7.3 we show the instantaneous maximum throughput for a mobile network for both the distributed algorithm as well as the centralized algorithm presented in chapter 6. The network is a 32-node network where all nodes moves randomly with a
speed of 20 m/s during 100 seconds. This is a very mobile network, but it can be seen as a worst case scenario for a mechanized battalion (not in size though).

The parameter setting for the distributed algorithm is here chosen to default setting with $\gamma_1 = 0$ and $\gamma_1 = 1$. As can be seen the maximum throughput is even slightly higher than what can be achieved with a centralized algorithm. This is true even when we have limited the information flow. The reason to this is connected to the theft of time slots. The centralized algorithm is of the greedy type, not optimal, once a link is scheduled to a time slot it will keep it (during scheduling) even if it becomes obvious that it is a poor decision later on. The distributed algorithm can fix problems later by using theft. An example when this happens is before any of the links have any time slots, priority is then equal for all links. Both algorithms may choose to fill a specific time slot with links that are easy to allocate, but with low traffic loads. Such links could probably be assigned anywhere later which means that the time slot is being wasted. In such a case the distributed algorithm can improve the situation by reassigning the time slot by using thefts.

Something interesting to notice is the event taking place about 18s in the simulation. At this moment we suddenly lose almost a third of the maximum throughput, later at 42 seconds the maximum throughput is returned. The reason to this is that the network is circle-shaped before 18 seconds and then the link that holds the circle together breaks and we get a U-shaped network, see Figure

Figure 7.3: The figure shows the instantaneous maximum throughput for a 32 node network during 100 seconds.
7.5. Conclusions

Figure 7.4: The figure shows the network at time instance 17 seconds (left) and 18 seconds (right).

7.4. Much of the traffic now needs to be routed a far longer distance while interference levels almost as high as before. It should be noted of course that this problem exist also for the centralized algorithm. Added frame length for better traffic sensitivity would not really help either. At 42 seconds a new link is created returns the network to a similar form as before.

The distributed algorithm can achieve very high maximum throughput with this parameter setting, however, it is inappropriate as we will see in the next chapter due to its very high overhead. Other parameter settings will function better. This is true even when we limit the information spreading through the network by setting $\gamma_I$ to 1.

7.5 Conclusions

In this chapter we have described an interference-based distributed STDMA algorithm that we in the next chapter will use to study how much overhead information it generates. As already have been seen it can give results as high as what a centralized algorithm can. By adjusting the different thresholds we can also change how much network information that is sent.
Chapter 8

Distributed Information—At what cost?

In this chapter we will study what information the nodes need to convey to the other nodes in their local neighborhood, how often such messages need to be sent, and, finally, determine the overhead for some different networks.

8.1 What Information Must Each Node Send and to Which Receivers?

At the end of the previous chapter we showed what information a node needs to make a correct decision about whether it can assign a time slot or not. In this section we will look at the problem from the other side and study what information a node needs to send to its local neighborhood so that the rest of the nodes can make correct decisions. Returning to the previous points, we can divide them into three regions depending on how far away the information is needed.

Available time slots to send (AVTS):

The sender $v_i$ to a link $(i, j)$ must send this information to the receiver $v_j$. No one else needs the information. This can be seen as Region 1 in Figure 8.1. This information is sent every time the availability changes for any time slot due to events in the network. However, with omni-directional antennas, whether or not a node can send is independent of which link it attempts to use. This means that
Figure 8.1: The different distances information need to be transferred. Solid lines are normal communication links and dashed lines are interference links.

\( AVTS \) should be equal for all outgoing links, and it will normally be sent as a broadcast message to all neighbors. An exception is the concept of theft, in which case \( AVTS \) can differ for the different links. The message will then be sent to a subset of the neighbors. The distance to send such a message can be seen as Region 1 in Figure 8.1, or a union of such regions when it is combined for all neighbors. It is still only one transmission, though.

We call this a TimeSlotSendMessage.

**Local Schedule and \( I_{\text{max}} \)**

A node needs to know the local schedule and how much more interference can be handled by the assigned receivers \( I_{\text{max}}(l, \tau) \) in each time slot.

This information is usually updated after an assignment or deassignment of a time slot, although not necessarily by the actual link. We can also have updates due to changes on path gain caused by mobility. This information should be sent to nodes whose transmissions can interfere with the receiver of the link and nodes whose reception may be affected by the sender.

The local schedule will be sent to the Region 2 in Figure 8.1. This region consists of all nodes with a link or interference link to the receiver or transmitter. The \( I_{\text{max}} \) value can be sent to a smaller region than this as only nodes with links or interference links to the receiver need this information. However, for now we ignore this since we assume that they are sent in the same message when an assignment has taken place. Changes of \( I_{\text{max}} \) can be sent to the smaller region, however.

We call this a TimeSlotChangeMessage.
8.1. What Information Must Each Node Send and to Which Receivers?

Priority and state

Knowledge about the priority and state of links is needed even further away. The reason for this is that the receiver is in control of the link, which also gives it control of when the sender to the link transmits. This means that the node also needs information about links that the transmitter can create interference for in addition to knowledge about receivers of links whose transmissions can cause problems for the reception of the link. The sender will only test already assigned slots. It cannot prevent its neighbors from attempting to do assignments. Only if the priority and state of such links are known by the receiver can this be prevented. The range of such updates is shown as Region 3 in Figure 8.1.

These are StateChangeMessages and TrafficRateMessages.

Because these messages need to be transferred very far in the network, it would be an advantage if we could limit the number of them somewhat. StateChangeMessages will be sent every time a link wakes up or goes to sleep. If the link has a high priority value, this will happen many times without it having a chance to allocate something. Links with lower priority value will allocate the available capacity first. One way of improving the situation is to introduce a State Threshold ($ST$) which controls whether links should be awake or not.

A link that can assign slots normally (or by theft) will remain asleep if its priority value is sufficiently high compared with the rest of the nodes in the local neighborhood. However, we use a weighted average in accordance with the traffic load each link carries. We thereby reduce the effect of a few links with very light traffic load. A link will thereby remain asleep if:

$$\frac{h_{ij}}{\Lambda_{ij}} > \frac{\sum_{kl \in \text{Neighborhood}} h_{kl}}{\sum_{kl \in \text{Neighborhood}} \Lambda_{kl}} \cdot ST$$

When we change the data rate, $h_{ij}$ may also change continuous. Therefore we also introduce an $h$ Threshold ($hT$), similar to the IMT described in the previous chapter.

Link updates

Every time there is a link change, i.e. a link is created or destroyed, or if the power received from another node passes the interference threshold, it creates a change in the local neighborhood of not only the transmitter and receiver of that link but also of nodes further away. If a link has changed to a lower state, e.g. a link changes into an interference link, this handling is rather simple as it
means that the local neighborhood decreases for the nodes, and they only need information about the changing link itself. The new local neighborhood can be decided by all nodes on their own with nothing more than information about the changing link.

Sending messages after this may be a little bit more complex because the routing will change, but we assume that this problem belongs to the routing layer, which we have assumed can handle this efficiently as long as the MAC layer can keep track of its local neighborhoods (see assumptions in chapter 2).

The second case occurs if a link has changed to a higher state, e.g. an interference link into a link. This will result in a situation in which nodes on both sides of the link may need to increase their local neighborhood with new links and even new nodes. In such a case we assume that the sender and receiver of the changing link is responsible for updating their local neighborhood.

Such an update will take place in a number of steps:

1. The sender and receiver exchange their local neighborhood (or an appropriate subset, such as links that could be of interest).

2. Both these nodes receive this information and retransmit new information in their local neighborhood. Information is sent only if it is needed, i.e. information about a link is sent to those nodes that should have the link in their Region 3 neighborhood. This means that nodes at two hops distance do not need information about anything except perhaps the new link, see Region 3 as an example.

   We may also need to send information about links that is already known. A new link may make a node, previously at two-hop distance, become a direct neighbor, which should be known by other nodes in our neighborhood.

3. Nodes receiving this information upgrade their local neighborhood accordingly.

Notice that the only nodes actively involved, i.e. sending messages, in this exchange are the sender and receiver to the changing link. (This is not entirely true, however. In some cases if multiple events are taking place at more or less the same time, it might be necessary for a node to retransmit information about new links even if a node is not a transmitter or receiver. Such an occurrence can be noticed if the node studies the destination set of the message and compares it with the destination set it would send this message to. If they differ, an extra
8.1. What Information Must Each Node Send and to Which Receivers?

message may be necessary. This will only happen if the node has had link changes at about the same time as the change that the message describes.)

To clarify the problem with link updates we will give an example. In Figure 8.2 we see a 7-node network. At the start of the example we will assume that there is no link between node \(v_3\) and \(v_6\). Then an interference link is created between them, and, finally, this interference link is upgraded to a full link.

At the start of the example, node \(v_3\) has knowledge about nodes \(v_1, v_2,\) and \(v_4\) and all links between these nodes. The only knowledge about node \(v_4\) it has is the link \((2, 4)\). Node \(v_6\) has knowledge about the nodes \(v_4, v_7,\) and \(v_5\) and all links between these nodes. It also has knowledge about link \((2, 4)\), but node \(v_2\) is not included in the local neighborhood.

At some moment, nodes \(v_3\) and \(v_6\) detect that they are sufficiently close so that they can no longer ignore each other. We will discuss how this is done later in this chapter when we describe how the information is sent on the channel.

Both of these nodes now send an appropriate part of their local neighborhood to each other. How this is done is left for the routing layer, but normally we do not need six hops to find the other side as in this example. In more practical implementations one could probably set a maximum number of permitted
hops, and let the algorithm handle interferences as unidentified in such cases. But here we will assume that information is always sent if possible.

Node $v_3$ sends information about itself, $v_3$, $v_1$ and links $(1, 3)$ and $(3, 1)$ as more is not needed on the other side of an interference link (see Figure 8.1).

Node $v_6$ sends information about itself, $v_6$, $v_7$ and links $(7, 6)$ and $(6, 7)$.

This information is then to be spread in the “old” local neighborhoods of $v_3$ and $v_6$. Once again, though, we limit the necessary information. Node $v_6$ only needs to inform node $v_7$ about node $v_3$, its incoming link $(1, 3)$ and the existence of the new interference link. Node $v_3$ informs node $v_1$ about node $v_6$, its incoming link $(7, 6)$ and the new interference link.

Once these updates are done, the new local neighborhoods are updated.

Now, let us assume that nodes $v_3$ and $v_6$ move closer, and the interference link will eventually be useful as a communication link. The reaction of $v_3$ and $v_6$ are similar to the previous reaction, but the appropriate information is different than the previous update. Node $v_3$ sends information about $v_3$, $v_1$ and $v_2$, and links $(1, 3)$, $(3, 1)$ and $(4, 2)$. In addition to this, information about the existence of the interference link between $v_3$ and $v_2$ is also given.

Similarly, node $v_6$ sends information about $v_6$, $v_7$, $v_5$, and links $(7, 6)$, $(6, 7)$, $(4, 5)$, and information about the interference link between $v_6$ and $v_5$.

Some of this information could, of course, be omitted, because many of these links are already known. But to be safe, information about this can be transmitted anyway, as a precaution if the link gain changes so fast that the interference links do not have time to be generated. This could happen if we have obstacles between the nodes. This information does not need to be relayed any more since a direct link is available, however, care should be taken since the routing algorithm possibly does not know anything of this link yet.

Nodes $v_3$ and $v_6$ should now inform their local neighborhoods about the change.

In addition to this we have also added sequence numbers to these messages, because simultaneous events may cause old information to arrive later than the new information if several links are changing at the same time. When nodes in such cases send their local neighborhoods, there is no guarantee that no old information is received after the new. By adding a sequence number that only the owner (receiver) of a link may change, we make certain that old information is not used. For now, these are only necessary for links updates as these messages are sent by nodes other than the owner of the link.

In addition to this information, the local schedule must also be updated. However, this information is only needed in Region 2, which means that the
sender and receiver only needs to send the time slots in which they have outgoing or incoming links assigned. In addition, this information does not need to be further conveyed.

These are LinkChangeMessage Type 1 and 2.

**Overview of the Message Types**

We conclude this section by giving an overview of the existing messages shown in Table 8.1.

<table>
<thead>
<tr>
<th>Message</th>
<th>Purpose</th>
<th>Destinations</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVTS</td>
<td>When the sender can transmit</td>
<td>Only the receiver</td>
<td>1</td>
</tr>
<tr>
<td>Schedule</td>
<td>Assigned time slots and their $I_{max}$</td>
<td>Those nodes that can be affected by the assignment, i.e. at least a direct interference link</td>
<td>2</td>
</tr>
<tr>
<td>Priority</td>
<td>Controls access in the local neighborhood</td>
<td>Sufficient so that two simultaneous assignments do not cause problems</td>
<td>3</td>
</tr>
<tr>
<td>Link Changes</td>
<td>Updates the local neighborhood</td>
<td>Same as above</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 8.1: An overview of the different types of messages used in the scheduling.

**8.2 Message Size**

To determine the overhead traffic we must also know how large each message is. We will study this problem in this section using the same assumptions as in the rest of the report, i.e. there will be no lost messages. Therefore, each message can be very optimized.

An additional assumption is that the exact number of nodes in the network is known as this can simplify the messages.

In the following Table, 8.2, we give the required message size for a 32-node network with a frame length of 200 time slots.

For each message we use a long and short length. The long message often contains a full 32-bit multicast address, whereas the short uses the fact that
many messages are transmitted to the same multicast group many times and that nodes can save such information. This is of course, overoptimistic for a single message. However, we disregard source coding over many messages, which could decrease message length a lot as different messages often contain similar information, and several messages are often generated at the same time.

<table>
<thead>
<tr>
<th>Message Type</th>
<th>Long Size</th>
<th>Short Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>LinkChangeMessage Type 1</td>
<td>55b</td>
<td>29b</td>
</tr>
<tr>
<td>LinkChangeMessage Type 2</td>
<td>82b</td>
<td>56b</td>
</tr>
<tr>
<td>StateChangeMessage</td>
<td>48b</td>
<td>16b</td>
</tr>
<tr>
<td>TimeSlotChangeMessage - ALLOC</td>
<td>72b</td>
<td>40b</td>
</tr>
<tr>
<td>TimeSlotChangeMessage - DEALLOC</td>
<td>58b</td>
<td>26b</td>
</tr>
<tr>
<td>TimeSlotChangeMessage - Theft</td>
<td>35b</td>
<td>-</td>
</tr>
<tr>
<td>TimeSlotChangeMessage - I : max</td>
<td>72b</td>
<td>40b</td>
</tr>
<tr>
<td>TimeSlotChangeMessage - Info</td>
<td>47b</td>
<td>-</td>
</tr>
<tr>
<td>TimeSlotSendMessage - Normal</td>
<td>20b</td>
<td>-</td>
</tr>
<tr>
<td>TimeSlotSendMessage - Steal</td>
<td>21 - 52b</td>
<td>-</td>
</tr>
<tr>
<td>TimeSlotSendMessage - All</td>
<td>21 - 52b</td>
<td>-</td>
</tr>
<tr>
<td>TimeSlotSendMessage - Info</td>
<td>20b</td>
<td>-</td>
</tr>
<tr>
<td>TimeSlotSendMessage - All timeslots</td>
<td>215b</td>
<td>-</td>
</tr>
<tr>
<td>TrafficRateMessage $\lambda_{ij}$ and $h_{ij}$</td>
<td>72b</td>
<td>40b</td>
</tr>
</tbody>
</table>

Table 8.2: The size of different types of messages used by the algorithm for a 32-node network. (“Info” is a specific message type transferred at the same time as LinkChangeMessages, updating Regions 1 and 2.)

8.3 How Often Are These Messages Sent?

We also need to know how often each type of message is going to be sent.

There are two different causes of virtually all changes in an ad hoc network: link gain changes and traffic changes.

The first of these is usually dependent on mobility, but nodes that leave or enter the network can be seen as more abrupt changes in link gain. In our traffic model, traffic changes will mainly depend on rerouting due to changes in path gain but in general will also depend on the input traffic to the network.

For small changes in link gain, $I_{max}$, may change value and it is possible that time slots can now be assigned or must be deassigned because of the SIR
8.4 Exactly How Are Each of These Messages Sent?

The basic idea is to divide time into two parts, as shown in Figure 8.3, as USAP [21] uses its bootstrap slots.

- One TDMA frame with mini-slots used for administrative information (STDMA overhead) where one is given to each node.
- One STDMA frame which is used for user traffic.

In practice the different frames do not need to be divided as the figure shows, and for large networks a pure TDMA frame is probably not necessary, but we want to use two advantages of TDMA here.

First, TDMA is independent of mobility. As long as we have no new nodes, the schedule does not need to be updated. This means not just that such updates can be ignored, but also that the slots which each node uses can be known in the entire network.

Second, because only one node is sending in these slots and the rest of the nodes know which one this is, it can be used for determining both link gain and when interference links exist.
In a large network, a pure TDMA frame is not very efficient. We may also have problems with traffic adaptivity if the administrative traffic is different on the different nodes.

In such cases we may need to change the concept somewhat, because a pure TDMA frame is inefficient. Each node should still have one mini-slot each, but we might need more slots to handle the traffic variations. Which node that should send in a slot could then be decided based on priority, message history, or reservations made in the pure TDMA frame. We will not go into detail about this but leave an exact implementation for future work. As long as the simplified overhead is a small part of the total available capacity, the real overhead should not be that much larger.

As previously stated we will assume that we have no delay for these TDMA messages and that two independent events cannot take place simultaneously. (They may if they take place at the exact same moment in time.)

The overhead cost will now be the amount of the time that needs to be mini-slots. We also assume that this part can be varied over time, because the amount of overhead traffic generated that will be generated is dependent on the size of the network change that occurs. Therefore we will study the average overhead as a measurement.

8.5 What Affects the Overhead?

Before we show simulation results, it might be useful to consider which network properties that will affect the overhead:

- **Network size**

  The network size has a direct effect on the TDMA frame; a doubling of the network size doubles the required TDMA frame. It may also increase the overhead per node if the local neighborhoods increase in size. The network size is not a parameter we can control, but it will affect the performance of the algorithm. From Chapter 6 we also know that the required frame length also increases with network size. In addition to this, the required address space will be larger for a large network.

  We will mainly study a 32-node network.

- **Network Connectivity**
Network connectivity is a measure of how large a part of the network that a node can reach with one hop. Connectivity will have a direct effect on the local neighborhoods, thereby increasing the number of nodes information needs to be spread to. As above, network connectivity is difficult to control, but because it is possible to do this to some extent by power control, we will mainly study this for low connectivity.

- **Choice of $\gamma$**
  Connectivity is a measure on normal links. The interference threshold will affect the local neighborhood through interference links. The smaller this parameter, the larger the part of the network included in the local neighborhood. This is one parameter that is under our control (at least for higher values; small interference values can be troublesome to measure).

- **Mobility rate**
  Mobility rate is the rate of changes in the network. A faster network will require updates more frequent. However, as long as the STDMA algorithm manages to keep the schedule updated, the mobility rate should have an approximately linear effect as all updates only react to changes.

- **Frame length**
  The more time slots we have to assign and negotiate, the more overhead it requires. But a longer frame length also gives higher maximum throughput, as seen in Chapter 6.

### 8.6 Feedback and Stability–Discussion

For $\gamma$ equal to zero we will never have more than one link active at the same time. This means that normally when a link is assigned a time slot, this will not cause a problem for anybody else. Only mobility (and theft) can force a link to give up a time slot. For time-slot efficiency, we want to choose a high threshold when we give up a time slot so that few links use decreased data rates.

For $\gamma$ greater than zero, the situation is not quite so convenient any more. Information is now limited and an assignment of a time slot that seems to be valid in a node’s local neighborhood may create problems for the nodes outside this. A problem that appeared when we tested the algorithm the first number of times was that it did not stop for $\gamma$ greater than zero. This was because thresholds chosen for giving up a time slot were so high that external interference
from nodes outside the local neighborhood more or less always forced some links to give up their time slots. As a consequence, they tried to steal new time slots (which did not work any better) one after the other. This problem can be alleviated, as seen in the next section, by changing some of the thresholds and not allowing a link to reassign itself a time slot it already has lost for a period of time. However, the feed-back loop from external (outside local neighborhood) information is still there.

It is easy to assign links on the edge of the network and short links compared with those in the center, especially long (thereby lower SNR) links. Assigning such links will cause problems for the links outside their local neighborhood, forcing decreased data rates, which results in increased priority and possibly thefts. Links that lose time slots may then steal from links further away, possibly the link that started this. Such loops lead to poor convergence of the algorithm for $\gamma_I$ greater than zero and thereby unnecessary high overhead traffic. On the other hand, maximum throughput will be high since the algorithm will continue until it finds a good schedule.

It is interesting to notice that relatively good schedules are found very quickly after a change takes place, see Figure 8.4 for an example. But for limited information, the algorithm continues long after this, optimizing (with limited success). It stops at a value it could have found in a quarter of the time. If we could stop the scheduling this early, we could get a much lower overhead cost than

Figure 8.4: The figure shows how instantaneous maximum throughput is dependent on time during an update.
what we presently have. For the full information case this is less of a problem. The optimization phase is still here, but here it improves the schedule and is much shorter.

In the next section we will study better choices of parameter settings than the default values. Unfortunately, the optimization phase is not easy to turn off without negative consequences.

### 8.7 Parameter Settings

The messages described above will be used when the algorithm is run. However, the number of messages that will be sent is highly dependent on the exact parameter setting chosen. In this section we will try to find a better choice than the default values, i.e. when the different thresholds are not used. We will not try to optimize these parameters, because this would be very complicated, and the exact result is probably very dependent on the specific network and mobility pattern.

Furthermore, because our simulated values of the overhead will only be approximative, the value of an exact optimization even for a specific network is not very high.

We will start by studying how much overhead traffic each message type costs with the default setting. In Figure 8.3 we plot this for the different message types with $\gamma_I = 0$ and $\gamma_I = 1$. As can be seen, it is most expensive to transmit $I_{\text{max}}$, followed by State information, Traffic and Priority.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 LC1</td>
<td></td>
</tr>
<tr>
<td>2 LC2</td>
<td></td>
</tr>
<tr>
<td>3 State</td>
<td></td>
</tr>
<tr>
<td>4 ALLOC and DEALLOC</td>
<td></td>
</tr>
<tr>
<td>5 $I_{\text{max}}$</td>
<td></td>
</tr>
<tr>
<td>6 AVTS</td>
<td></td>
</tr>
<tr>
<td>7 Traffic and Priority</td>
<td></td>
</tr>
</tbody>
</table>

Table 8.3: OH traffic for different types of messages.
Figure 8.5: Maximum throughput (left) and Overhead (right) for different IM.

\(I_{\text{max}}\) is a continuous variable (almost at least, as the simulator will give time samples) and changes very rapidly. Many such messages will be sent with only very minor changes. As we will see, the Imax Threshold will decrease this considerably with only a minor decrease in maximum throughput. Priority updates have partially the same problem when the data rate is decreased.

State change messages are also very expensive, due to the distance they need to travel. Another problem with these messages is that a local change that free a time slot will awake a lot of links, but usually only one or two will receive the time slot. The rest will go to sleep again with the result that two StateChangeMessages will be sent over a large part of the network without any gain for each link in the neighborhood. The State Threshold will alleviate this problem somewhat but cannot eliminate it. In addition the AVTS messages have a similar problem, although it is local.

From this, we see that good parameter setting choices are necessary. If set to default, the overhead traffic will be significant. In this example, we have more than 2 Mbits/s just for overhead.

### 8.7.1 Imax Threshold

To avoid the problem around continuous variables, we introduced the Imax Threshold (IMT).

In Figure 8.5 we plot the maximum throughput and overhead traffic for different values of the Imax Threshold. As can be seen the maximum throughput does not depend very much on this threshold, but for values above 0.15 we start
8.7. Parameter Settings

Theft Threshold

The network is constantly changing, some links that do not need capacity at one time may need capacity later. We therefore allow the links to steal from each other if this results in an increase in the local maximum throughput. In the default settings, this is allowed if maximum throughput is higher after the theft independent of how small the gain is. The result of this is that we may have many such thefts at the end of each scheduling phase that optimizes the schedule. For a static network this may be an advantage, but for a mobile network the cost of these thefts may be higher than the actual gain.

In Figure 8.6 we plot maximum throughput and overhead traffic as a function of the Theft Threshold. As can be seen the maximum throughput decreases with a higher value of the Theft Threshold. From this we can conclude that even very small gains by theft improve maximum throughput. However these small
Chapter 8. Distributed Information—At what cost?

gains have a larger impact on overhead traffic. For now, we will choose to use a value of 1.05.

8.7.3 State Threshold

A problem for small changes in network topology is that only a few links usually benefit from the changes, but many links will attempt to do so. For example, if a link deallocates a time slot, it may suddenly seem to be available for many links in the local neighborhood, but usually only one of them will be able to assign itself the time slot. However, all of them may wake up and send a StateChangeMessage, which all must be sent to the whole local neighborhood at a large cost. Once a link has assigned itself the time slot, they will need to send a new StateChangeMessage saying that they are now asleep again.

This is a costly process. By adding a State Threshold as described in 8.1, we can decrease the number of such messages.

In Figure 8.7 we plot maximum throughput and overhead traffic as a function of the State Threshold. For $\gamma_I = 0$ the maximum throughput seems to be rather independent of this threshold, whereas for $\gamma_I = 1$ it actually increases when the State Threshold uses a low value. One reason for this may be that edge nodes will be less aggressive now, not trying to assign themselves time slots that they do not need as often. This leads to fewer cases of decreased data.
8.7. Parameter Settings

It is interesting to notice, however, that the limited information now gives higher maximum throughput than full information. Yet overhead is higher, so nothing is for free.

From these Figures it seems that the smaller values of State Threshold we choose, the better result we get. However, a choice of threshold at 1.0 or below will be difficult since that would put all links to sleep if the links have the same priority value (this happens, for example, at initialization). For now we will choose a value of 1.02.

8.7.4 h Threshold

When we change the data rate, \( h_{ij} \) may also change continuously. We have therefore introduced an \( h \) Threshold (\( hT \)).

In Figure 8.8 we plot maximum throughput and overhead traffic as a function of the \( hT \).

As can be seen, overhead traffic decreases with increasing value, whereas maximum throughput remains more or less constant. The improvement is also larger for higher values of \( \gamma_I \). A value of 0.05 seems to be a good choice.
8.8 An Alternative Possibility–Removal of States

The main problem with everything done so far is that the general overhead cost is simply too high. It might even require less overhead to do this for a centralized scheme in which all information is collected into a central node, which distributes a schedule to the rest of the network. The main reason for this is that a centralized algorithm does not need to negotiate for each time slot; only path loss values need to be transmitted. A centralized solution is less robust, and errors could cause large problems. But that a centralized scheme could do this more cheaply is still not a desirable trait.

The distributed algorithm still has some benefits. First, most links will not experience any problems when we have a change and can simply continue using the same schedule as before. Second, even if the distributed algorithm would not have sufficient assigned capacity for finishing the all the updates we will still always have a schedule, and updates will start with the most problematic links and improve things quickly for these links.

There is an alternative way of doing the scheduling that is more optimized to uniform capacity. We can remove the link state asleep. This means that links never go to sleep, with the result that no nodes will ever assign any time slots to a link if that link does not have the highest priority. In addition this means that there will be no optimization phase. All thefts will be the result of mobility.

Figure 8.9: The figure shows the instantaneous maximum throughput for a 32 node network during 100 seconds (left) and average overhead during the same time.
If we now run the algorithm with this new assumption, we can see in Figure 8.9 the instantaneous maximum throughput for the same mobile network as before. The parameter setting for the distributed algorithm is set to default here for $\gamma_I = 1$ and $\gamma_I = 0$. As can be seen, we have lower maximum throughput than we had when state was used. This is due to decreased (or removed) optimization phase. But, as can be seen, the overhead has also decreased compared to the case using nominal values of the original algorithm, see 7.3. As before we can choose values of Imax Threshold and Theft Threshold to decrease the overhead. For simplicity we choose the same values, even if additional improvement can be found by optimizing them directly for the non-states version as well.

An additional property here is that $\gamma_I = 1$ gives higher maximum throughput than $\gamma_I = 0$ even for the nominal parameter case. This is probably because a certain optimization phase still exists for $\gamma_I = 1$, which gives a slight increase both in maximum throughput and overhead.

In the following we will study algorithms that use states and those that do not.

### 8.9 Frame Length

In Figure 8.10 we plot maximum throughput and overhead as a function of the frame length. As can be seen the maximum throughput behaves similar to the static networks investigated in Chapter 6.
We can see that the overhead cost seems to increase linearly with the frame length, and that the choice of a frame length of 200 times slots appears reasonable for this network.

This also means that the joint node and link assignment scheme from Chapter 6 could be very efficient here since we would probably need less than half the frame length to reach the same maximum throughput, thereby reducing overhead by about the same. (Almost, at least, as nodes handling would need to be added.)

8.10 Results for Different $\gamma_I$

In Figure 8.11 we now plot maximum throughput and overhead as a function of $\gamma_I$. This parameter controls the size of the local neighborhood, and it is important to choose a good value for this in order to achieve good performance.

Using states is a very aggressive form of scheduling that works well if we have good information about the network. For this network it achieves the highest maximum throughput for values of $\gamma_I$ around one or less. For values around four or higher the algorithm experiences difficulties. This is because in this simulation we have chosen to let go of a time slot if the data rate has been lowered to below 30% of the nominal rate. When interference can be three or four times stronger than the noise levels, weak links will experience difficulties, because
the aggressiveness of the algorithm will try to fill the schedule as much as possible. Weak links will simply not find any time slots to assign, because they will all be blocked by external interference. All time slots will be tested without success.

In the no-states version, this is much less of a problem. The less aggressive assignments will leave many more time slots available for the weak links. It is still not very efficient to use very high values of $\gamma_I$. Eventually the maximum throughput of this version will drop as well. A considerable increase of overhead will be seen here as well, but first at much higher values ($\gamma_I \geq 7$).

### 8.11 The Effect of Mobility–Slow Networks

The network we are testing is a very mobile network, where each node moves around at 20 m/s independent of the others. This is a very difficult scenario, and it is not unexpected that the overhead cost will be very high. However, due to one of the basic properties of the algorithm that updates only take place when something happens, overhead should decrease for a less mobile network. In addition, as we have assumed that the algorithm has time to finish all updates between events (one event usually consists of many sub-events, though, for example, all path loss updates take place at the same time) mobility usually only scales the time between the events. If we also consider our traffic model, it means that traffic changes on the links only take place due to mobility. The overhead should therefore behave roughly linearly with mobility.

We have studied a slow network where nodes move at 2 m/s, but still use the same movement pattern as the usual 32-node network. It will therefore take 1000 seconds to move the same distance. We will not plot any results, but the simulation results look similar to the normal networks, but with overhead around 22 kbits/s when we are not using states, and 41 kbits/s if states are used.

### 8.12 The Effect of Connectivity–Sparse Network

STDMA is most useful in sparse networks, however, it cannot always be guaranteed that the network is connected, i.e. that there always exists a route from source to destination at the nominal data rate. The STDMA algorithm can handle such a situation, although there might be a substantial amount of the overhead traffic that will be sent over the interference links (with decreased data rate) for lack of faster links.
Chapter 8. Distributed Information–At what cost?

We will therefore study a partly unconnected 32-node network, with an average of 95 percent possible paths. One of the main benefits of the sparse network is the fewer number of links than the previously tested network. This will allow us to use a shorter frame length, 150 time slots.

In Figure 8.12 we plot the maximum throughput for the sparse network as a function of $\gamma_I$.

The curves look similar to those for the normal network. However, there is one significant difference: The cost of full information is now much more expensive. It is much more costly to send information to all nodes in a non-connected network because this requires decreased rates and the use of interference links. This suggests that full information may also be very expensive if the network size increases and the number of hops in the network increases. A value of $\gamma_I$ equal to one or higher works well for the version with states, even higher for no-states.

In addition, the frame length is reduced by a quarter, as is the overhead traffic. The requirement for a short frame length is obvious.

8.13 Comparison to Centralized Scheduling

As a final comparison we once again make a comparison to what maximum throughput that could be achievable with a centralized algorithm. In Figure 8.13 (left) we show the instantaneous maximum throughput for a mobile network for
8.13. Comparison to Centralized Scheduling

Figure 8.13: The figure shows the instantaneous maximum throughput during 100 seconds for the 32-node network.

<table>
<thead>
<tr>
<th></th>
<th>Centralized</th>
<th>Using States</th>
<th>No States</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantaneous</td>
<td>0.6034</td>
<td>0.6092</td>
<td>0.5659</td>
</tr>
<tr>
<td>Throughput</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8.4: The average instantaneous capacity during 100 seconds.

both the distributed algorithm with the chosen parameter setting as well as the centralized algorithm presented in chapter 6. This is plotted for $\gamma_I = 1$ when we are using states and for $\gamma_I = 2$ when we are not using states, because these are the best choices for this specific network scenario. In addition, in 8.13 (right) we shows the instantaneous maximum throughput from 22 seconds to 38 seconds into the simulations.

As can be seen, instantaneous maximum throughput when we are using states is on average as high as the maximum throughput for centralized scheduling, see also table 8.4, despite the fact that information is limited. When we are not using states, there is a 6% loss in maximum throughput but at half the overhead cost.

There is a considerable variation of the instantaneous maximum throughput during the 100 seconds though, which is unwanted. However, this is the case for the centralized algorithm as well and is a more fundamental network property so it is not a specific weakness in our distributed algorithm. The very quick change taking place at 18 seconds can create a problem when trying to guarantee quality of service.
These results do not consider the cost of overhead, however, but in order to do so we would need to consider data rates on the channel as well. If we have a 1 Mbit/s instantaneous link data rate, the cost of 200-400 kbit/s would be considerable and much of the capacity would be needed just to handle overhead traffic. On the other hand, for 10 Mbit/s or more these costs would be a small proportion of the total available capacity.

### 8.14 Future Work and Improvements

A number of improvements could be added to the algorithm to make it more efficient. One improvement is to allow nodes to assign more than one time slot at a time. This could be allowed if many time slots are available and the node can determine that it will receive more than one time slot for its links. This is the case for long frame lengths and could be one way to decrease the number of messages sent.

Another improvement is for the nodes to keep track of the local maximum throughput and how this changes over time. A node could use this to determine whether the network is in the first improving phase or the optimization phase in the end. Some minor tests on this have been performed that appears to give good results, but as this further increases the complexity of the algorithm simpler solutions would be preferable.

An additional way of decreasing the overhead is to attack the data rate feedback loop by ignoring the decreased data rate from the link layer as long as the data rate is not so low that the link should be deallocated from the time slot or that the algorithm calculates with modified data rates, i.e. the data rate used in calculations will be somewhere between real data rate and nominal.

In addition, we need to provide an efficient solution for how to transmit the overhead messages on the channel. A pure TDMA channel is not sufficient. We also need a more dynamic part that can handle the varying administrative traffic loads.

Node assignment could be implemented by turning control over to the transmitter instead of the receiver. A node could be assigned the time slot if all its outgoing links could be assigned in a time slot. This would also allow us to use joint node and link assignment, thereby reducing the need for long frame lengths.
8.15 Conclusions

In this chapter we have studied how much overhead the distributed algorithm generated. We have also studied how different parameter settings should be chosen. Good settings are necessary to reduce overhead. But such a setting works fairly well for the different networks we have studied.

The overhead traffic requirements can be significant, especially if we need a large frame length and the network is very mobile. For slower networks the amount of generated overhead is easily handleable. For larger, highly mobile networks (which also requires larger frame lengths) high link data rates will be needed.

We have described two different approaches: using states and not using states.

Using states gives a much more aggressive allocations, with schedules with maximum throughput much closer to that of a centralized algorithm but at a cost of high overhead. This overhead is currently probably much higher than what it could be, but we have not found a simple way to reduce the overhead. With more work, the overhead can probably be considerably decreased.

If we remove states, we get an algorithm that is much more careful in assigning time slots, resulting in lower throughput. But overhead requirement is so much lower that the algorithm can be possibly be used for the high mobility case for link data with reasonable data rates.

In addition, the required overhead can probably be reduced in the future by adding joint node and link assignment as this reduces the frame length requirement.

All problems are definitely not solved so far but this is a good start.
Chapter 8. Distributed Information–At what cost?
Chapter 9

Conclusions and Discussion

Adaptations on the link layer is common today. Interference-based scheduling is a tool that can bring this kind of adaptation also to higher layers in the communication stack. STDMA has the potential to exploit the radio channel to a much better degree than most currently used MAC protocols.

We started by looking at the two most common assignment strategies used for STDMA. For unicast traffic, link assignment behaves in the ideal case better for high traffic loads, achieving a higher throughput. However, this comes at a cost of higher delay than node assignment for low traffic loads.

To remove the inefficiencies we developed LET, which has the advantages of both methods. LET can be added to all link schedules, thereby considerably decreasing delay. However, as was shown these results are only valid if we have sufficient frame length, which is not certain in all cases, especially as overhead cost is proportionate to the frame length.

To really exploit the radio channel, the separate methods are not sufficient, only a combination of them can really give us the capacity we need with a sufficiently short frame length.

We have also looked at the underlying network models and seen what effect such a choice can have on network throughput. The traditionally used graph model can give very poor results in many cases. The full interference model is always best especially for large network size and low connectivity when the gain is highest. This shows that interference-based scheduling is very interesting if made to work.

However, the scenarios we have envisioned STDMA to work for are generally mobile, even highly mobile, and even if centralized algorithms can be a
help in finding and investigating properties of STDMA, they are not considered practically useful for us. Making STDMA distributed was one of the original goals, specifically if we could base it on the interference-based model. This has been one of the most difficult parts of the work.

Although not complete, the algorithm developed can give results close to what a centralized algorithm can, although the overhead traffic requirements can be significant, especially if we need a large frame length and the network is very mobile. For slower networks the amount of generated overhead can be easily handleable.

For larger highly mobile networks, high link data rates will be needed, but with a very good channel utilization as a result.
Appendix A

Simulation Model

Due to the relaying of packets the statistical properties of the behavior of the network is complicated, and an exact analytical analysis of the expected delay is difficult [7]. Therefore computer simulations is used in order to determine the network delay. Here we give a more detailed description of the simulation setup used.

We will first describe how the networks are generated and how the basic path loss between all nodes are calculated. Then we will describe how simulations in order to estimate network delay and maximum throughput have been done.

A.1 Generation of networks

All static networks used in the comparison are generated in the following way; $N$ nodes are spread out uniformly in a sample terrain consisting of mixed meadows and forest. The size of the area is $15km \times 25km$. The center frequency is chosen as 300 MHz, and antenna heights 3 m. Both communication threshold and reliable communication threshold are set at 10dB.

In the comparisons 500 to 1000 networks of size 10, 20, 40 and 60 nodes have been generated with different connectivity. The connectivity is varied by changing the transmission power for a network. The transmission power is always set sufficiently high so that it gives a connected network.
Mobile Networks

The mobile network is generated by spreading 32 nodes uniformly in the sample terrain, then each of them move randomly with a speed of 20 m/s in this terrain during 100 seconds. This is sampled ten times a second, i.e. a path loss matrix is generated each tenth of a second.

A.2 Link Gain

An essential part of modeling an on-ground or near-ground radio network is to model the electromagnetic propagation characteristics due to the terrain variation. A common approach is to use the basic path-loss $L_b$ between two nodes. The most simple assumption concerning the wave propagation is that no obstacles appear between the transmitting and receiving antennas, and no reflection or diffraction exist in the neighborhood of the path between the receiving and transmitting antennas. This model is the so-called free-space assumption.

Refined assumptions of wave propagation conditions near ground include terrain-height information and terrain-type information to estimate the basic path-loss. See Parsons’ [69, Sec. 2.3] for an introductory description of the problem. In the digital terrain database we have used, the height is represented as terrain-height samples at equidistant square lattice intersection points, and the terrain type is one of maximum 256 terrain types such as fresh water, salt water, forest, wet ground, etc. We assign electromagnetic ground constants such as relative dielectric constant $\varepsilon_r$ and conductivity $\sigma$ for each terrain type, and we also assign surface roughness for terrain types. Our database has a 50-meter grid for the terrain-height samples and a 25 meter by 25 meter terrain-type area.

All our calculations are carried out using the wave propagation computations library DetVag-90®, [70]. Here, we use the multiple knife-edge model of Vogler [71] with five knife edges. Currently, DetVag-90® works with narrow-band signals, i.e., the bandwidth of the transmitted signal should be of percentage order of the carrier frequency.

With the assumption of omnidirectional antennas, the link gain $G$ will be equal to $1/L_b$. 
A.3 Simulations in order to Estimate the Delay and Maximum Throughput

An essential part of the results in the thesis has been generated with the help of simulations of the network delay and maximum throughput. In this section we will discuss how this has been done and certain issues regarding the errors from such simulations. With a few exceptions, delay estimates have been made for traffic intensities far from the maximum throughput. This has the advantage of fast convergence toward a small estimation error because the delays of single messages will essentially be independent due to little or no interdependence between messages, (there is often only one message at a time in the network at low traffic arrival rates). There are some exceptions, however. We will start with a discussion on how network delay is obtained and continue with the maximum throughput.

A.3.1 Estimation of Network Delay

In the simulator, for each node, a pseudo random number generator generates a exponentially distributed number with average value \( N/\lambda \) as the time between generated messages in that node. For each time slot, this value is decreased by one. Once the value is zero (or less), a new message is generated in the node. The destination for this message is also generated randomly with equal probability for the other nodes in the network. Both creation time and destination are stamped on the message. Then the message is placed in the correct outgoing queue according to the routing matrix. In each node on the path the destination is first checked, and then the node continues relaying the message in the correct time slot according to the schedule \( S \). Once the destination is reached, the delay of this message is added to the delay of the other messages that have arrived at their destination. In addition, the number of arrived messages is recorded.

The chosen simulation time has varied somewhat depending on the situation. The delay on each path is different, which makes estimates of required simulation length somewhat more complicated. For low traffic rates, there is usually only one message at a time so the delays of different messages can be considered independent. We thereby get fast convergence toward a good estimate of the network delay. For higher traffic rates, delays for messages will no longer be independent so it will be more complicated to determine the estimation errors.

In addition we also have the time until the network reaches steady state.
Figure A.1: The figure plots the ratio between the standard deviation of the network delay and network delay versus for different traffic arrival rates.

When the simulations start, the network queues will be empty. For low rates, an empty network will be very common and the time to steady state will not affect the simulation results very much. For high rates, this can have more effect because an empty network will be extremely uncommon. To avoid using messages with too low delay (because they have been sent through a network with less traffic in queue than is likely) we remove the first part of the simulation, i.e. messages arriving within a certain percent of the total time are not considered at all.

The chosen simulation time has been 2000 times the frame length for low traffic rates. This will give approximately $2000 \times 0.01 \times 2 = 40$ messages on each path, which should be sufficient to give an acceptable estimate of the network delay. We also remove the first 10% of the simulated messages in this case. For medium rates and high rates, the value is set to 5000 times the frame length to get a better result.

It is also important to notice that the accuracy of most simulation results is not very important. Our main focus is more on the shape of the curves than on the exact values of network delay for each network. We also simulate network delay for a large number of networks all the time, which gives us more redundancy.
A.3. Simulations in order to Estimate the Delay and Maximum Throughput

An example of the standard deviation of the estimated network delay

In figures A.1 we now plot an estimate of the ratio between the standard deviation of the network delay and network delay for different traffic arrival rates. This is done for a single 20 node network with the above parameter settings by doing 200 separate simulations for each traffic arrival rate and from this estimating the standard deviation of the given results.

In the left figure this is done with low traffic arrival rates with a simulation length of 2000 times the frame length. As can be seen, the largest deviation in this region is in the relative order of a single percent. The error is largest at low arrival rates, since there are much fewer messages sent through the network at this rate.

In the right figure this is done for high traffic arrival rates with a simulation length of 5000 times the frame length. This time the errors are larger, especially close to 0.75 which is very close to the estimated maximum throughput of this network (0.757). Nevertheless, the errors are sufficiently small so that the simulation results can be useful. In addition, even if errors are on the order of tens of percent close to the maximum throughput limit this will not result in so large errors in estimations of maximum throughput since the network delay curve is very steep close to this value.

A.3.2 Estimation of Maximum Throughput

Maximum throughput is a much more difficult parameter to simulate because we want to find the largest value of the input traffic load so that we get bounded delay. The problem with this is that to detect that delay is unbounded would take infinite time to simulate, which is not possible. Instead we use the same method as when we estimate delay, but find which traffic load for which the network delay reaches a very high value. This will not give us the real value of the maximum throughput, but because the curve is very steep close to the asymptote, the error will be small if we choose a sufficiently large value of the network delay, see for example Figure 4.2. Exactly how much we underestimate the maximum throughput by is more difficult to say since we only know the approximate shape of the curve.

We also have the same problem as before with reaching steady state. The closer to the asymptote we try to simulate network delay, the longer it takes to reach an estimate that does not rise with the simulation length. In addition, due to the high variance and interdependence between the curves, it is difficult to
ascertain that steady state has been reached.

To handle this we will make partial estimates of the network delay based on the last 1000 frame lengths and compare these with the total estimate of the network delay. As long as the partial value is higher than the total value we have probably not reached steady state and will continue until we find a sample that is lower or until a maximum simulation time is reached. That maximum is now set at 30000 frame lengths, which gives approximately $30000 \times 2 \times 0.5 = 30000$ messages sent per path.

The maximum throughput will be found by increasing the input traffic load $\lambda$ until the delay passes the value $200 \times D_0$, where $D_0$ is the network delay at $\lambda = 0.01$. The step size is different for different simulation. Usually it is 0.01, but in figure 3.2 the choice is 0.001.
Bibliography


