FOI-R--1059--SE
December 2003
ISSN 1650-1942
Scientific Report

Jimmi Grönkvist

Distributed STDMA Algorithms for Ad Hoc Networks

Command and Control Systems
P.O. Box 1165
SE-581 11 LINKÖPING
SWEDEN

FOI- SWEDISH DEFENCE RESEARCH AGENCY
Command and Control Systems
P.O. Box 1165
SE-581 11 LINKÖPING
SWEDEN

FOI-R--1059--SE
December 2003
ISSN 1650-1942
Scientific Report

Jimmi Grönkvist

Distributed STDMA Algorithms for Ad Hoc Networks

Issuing organization	Report number, ISRN	Report type
FOI- Swedish Defence Research Agency Command and Control Systems P.O. Box 1165 SE-581 11 LINKÖPING SWEDEN	FOI-R--1059--SE	Scientific Report
	Research area code 4. C^4ISR	
	Month year December 2003	**Project No.** E7035
	Customers code 5. Commissioned Research	
	Sub area code 41. C^4I	

Author/s	Project manager
Jimmi Grönkvist	Mattias Sköld
	Approved by Martin Rantzer
	Sponsoring Agency Swedish Armed Forces
	Scientifically and technically responsible Jan Nilsson

Report title

Distributed STDMA Algorithms for Ad Hoc Networks

Abstract

Spatial reuse TDMA (STDMA) has been proposed as an access scheme for multi-hop radio networks where real-time service guarantees are important. The idea is to achieve high capacity by letting several radio terminals use the same time slot 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. Due to the mobility of the users, this schedule must be constantly updated. To make this possible such updates must be made in parallel with only local information, i.e. distributed STDMA algorithms must be used.

In this report we investigate and list the properties a distributed STDMA algorithm must have to be efficient in a military scenario. We also study the existing STDMA algorithms and see what methods they use to fulfill these properties and show that no existing algorithm can fulfill all these properties,

The main contribution of this report is the description of the central parts of an interference-based distributed STDMA algorithm. This is the first distributed algorithm that uses a interference-based model of the network, which is important since this property can make STDMA really efficient and competitive.

The algorithm is evaluated to show that this distributed algorithm can achieve as high network capacity as a centralized scheme using the same information.

Keywords

STDMA, ad hoc networks, multi-hop networks, MAC, distributed algorithms

Further bibliographic information	**Language** English
ISSN 1650-1942	**Pages** 64 p.
	Price acc. to pricelist

Utgivare Totalförsvarets Forskningsinstitut-FOI Ledningssystem Box 1165 581 11 LINKÖPING	**Rapportnummer, ISRN** FOI-R--1059--SE	**Klassificering** Vetenskaplig Rapport
	Forskningsområde 4. Spaning och ledning	
	Månad, år December 2003	**Projektnummer** E7035
	Verksamhetsgren 5. Uppdragsfinansierad verksamhet	
	Delområde 41. Ledning med samband och telekom och IT-system	
Författare Jimmi Grönkvist	**Projektledare** Mattias Sköld	
	Godkänd av Martin Rantzer	
	Uppdragsgivare/kundbeteckning Försvarsmakten	
	Teknisk och/eller vetenskapligt ansvarig Jan Nilsson	
Rapportens titel Distribuerade STDMA-algoritmer för ad hoc-nät		

Sammanfattning

Spatiell TDMA (STDMA) är en accessmetod för radionät med flerhoppsfunktion där realtidstjänster är viktiga. Grundidén är att låta flera radioterminaler använda samma tidlucka när de är tillräckligt långt från varandra geografiskt så att interferensen mellan dem är tillräckligt låg. Sändningsrättigheterna för terminalerna beskrivs med ett schema. Mobiliteten gör att detta schema kontinuerligt måste uppdateras. För att göra detta praktiskt måste dessa uppdateringar göras parallellt i olika delar av nätet med bara lokal information, dvs distribuerade STDMA-algoritmer är nödvändiga.

I den här rapporten listas och undersöks vilka egenskaper som är viktiga för att en distribuerad STDMA-algoritm ska vara effektiv i ett militärt scenario. Vi studerar också de distribuerade algoritmer som existerar idag samt undersöker vilka metoder de använder för att uppfylla de listade egenskaperna. Med hjälp av detta kan vi ge argument varför existerande algoritmer inte är tillräckliga för att uppfylla villkoren.

Det centrala i rapporten är en beskrivning av de väsentliga delarna av en interferensbaserad distribuerad STDMA-algoritm. Detta är den första distribuerade algoritmen som är interferensbaserad, viket är viktigt eftersom denna egenskap väsentligt kan öka effektiviteten hos STDMA.

Algoritmen utvärderas och vi visar att den kan ge lika hög kapacitet som en centraliserad variant om den får tillgång till lika mycket information.

Nyckelord

STDMA, ad hoc-nät, multihopnät, MAC, distribuerade algoritmer

Övriga bibliografiska uppgifter	**Språk** Engelska
ISSN 1650-1942	**Antal sidor:** 64 s.
Distribution enligt missiv	**Pris: Enligt prislista**

Contents

Chapter 1

Introduction

1.1 Background

In future military operations, the armed forces must be able to operate against a variety of threats (from heavy armor to irregular forces), in a variety of environments, including urban, mountainous and forested terrain, and 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.

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 them for maximum efficiency. The network must also handle platforms that move at great speed while still retaining connectivity. A tactical network may be partitioned or fragmented into parts, which will require all parts to function autonomously, i.e. distributed network control. In many situations hostile jammers may be present, and the loss of any unit is possible.

Therefore, the fixed communication infrastructure cannot be relied upon, and fast self-configurable networks must be deployed quickly. A common feature of such networks is that they are not pre-planned, and area coverage is achieved by letting the radio units relay the messages, i.e. a multi-hop network. Distributed multi-hop radio networks are often referred to as ad hoc networks.

All services the network will provide must be simultaneously handled by the system, and each of them has different service demands. The upholding of such

requirements is usually denoted as Quality-of-Service (QoS) guarantees. One of the most challenging problems today in ad hoc network research is upholding QoS guarantees.

An important design issue for mobile ad hoc networks is Medium Access Control (MAC), i.e. how to avoid or resolve conflicts due to simultaneously transmitting radio units.

Most existing MAC protocols for ad hoc networks use 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) [1], 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, which can lead to the so-called hidden terminal problem, i.e. a user senses the channel is unused, but the receiver is occupied by another user beyond the user's sensing range. A way around this is to first transmit a short request-to-send (RTS) and then send the message only if a clear-to-send (CTS) is received. This is the general principle of the IEEE 802.11 standard [2], which at present is the most investigated MAC protocol. However, several successive RTSs can be lost, which makes delay guarantees difficult.

Efforts have been made to guarantee QoS in CSMA-based medium MAC, see e.g. [3], but contention-based medium access methods are inherently inappropriate for providing QoS guarantees.

One of the most important QoS parameters in many applications is delay guarantee, which is specifically sensitive to the MAC.

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 multi-hop 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 will affect only a local area.

However, to achieve both high capacity and delay guarantees one can use spatial reuse TDMA (STDMA) [4], which is an extension of TDMA where the

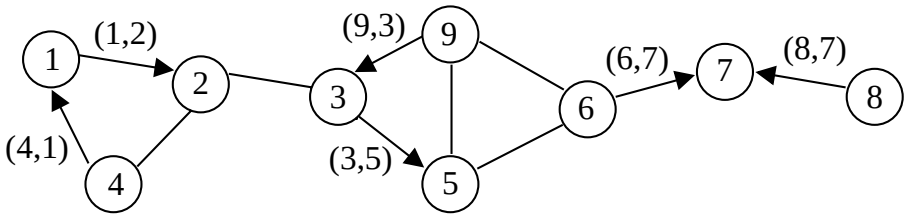

Figure 1.1: A 9-node network.

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 only minor interference is obtained.

An STDMA schedule describes the transmission rights for each time slot.

In figure 1.1 we show an example of a graph representation of a 9-node network. In this 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 $(8, 7)$ since, 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.

Unfortunately, the nodes will be moving, and nodes that can transmit simultaneously without conflict at one moment will probably not be able to do this later. Therefore, the STDMA schedule must be updated whenever something changes in the network. For example, 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 since the central node has all the information about the network. Several centralized algorithms have been proposed [5–7]

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, since 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 upon it and update their schedules without the need to collect information into a central unit.

The problem of designing distributed STDMA schedules is well addressed in the literature. Several algorithms for mobile ad hoc networks have been proposed. Few of these have been implemented into functional systems, but one of these, USAP [8], is used for the generation of multi-channel STDMA schedules in the soldier phone radio [9], which is designed as an ad hoc radio for military use in mobile environments. USAP will be further described in chapter 3.

In this report we describe the properties a distributed STDMA algorithm should have in order to be efficient. We also study the existing STDMA algorithms and see what methods they use to fulfil these properties. No existing STDMA algorithm can fulfill all these properties, although the most complete USAP [10] fulfills several of them. STDMA has great potential, but so far work is still required to make it sufficiently efficient to be competitive. Therefore we focus on how to use distributed information and describe an algorithm that can efficiently handle different amounts of information.

We evaluate this algorithm and show that it can give the same capacity as a centralized scheme with the same information. Furthermore, we show how capacity decreases with information reduction.

1.2 Outline

In Chapter 2 we describe the interference-based network model we have used. This gives us information about when links can be used simultaneously without a conflict. We also describe the difference between node and link assignment. Finally, we describe the more traditional graph model since most existing work is based on this.

In Chapter 3 we give a more detailed description of which properties a distributed STDMA algorithm should have to be of use to us. Furthermore we describe the previous work in the area of distributed STDMA scheduling and how well the different algorithms fulfil the properties we require.

In Chapter 4 we describe the central parts of a distributed interference-based STDMA algorithm and in Chapter 5 we evaluate this algorithm and show that it gives as good a result as a centralized approach if they are given the same

information. We also evaluate the algorithm for different input information.

Finally, in Chapter 6 we make some concluding remarks and discuss further work.

Chapter 2

Network model

This chapter introduces the network model we use and the assumptions required. We start by describing the interference-based model of a radio network. This model gives a realistic description of when users can transmit on the channel without conflicts. Traditionally for ad hoc networks, the most frequently used model is graph-based. Such a model is much simpler to use, but its ability to describe the communication channel is not very good. This model will also be described, since it simplifies the description of already existing algorithms in Chapter 3.

2.1 Interference-Based Scheduling

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), Γ_{ij}, as

$$\Gamma_{ij} = \frac{P_i G(i,j)}{N_{\mathrm{r}}}, \tag{2.1}$$

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_{r} 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*, γ_{C}. That is, the set of

links in the network, $\mathcal{L}$, is defined:

$$\mathcal{L} = \{(i,j) : \Gamma_{ij} \geq \gamma_{\mathrm{C}}\} \ . \tag{2.2}$$

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

$$V_{\mathrm{T}}(L) = \{v_i : (i,j) \in L\} \ .$$

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

$$I_L(i,j) = \sum_{v_k \in V_{\mathrm{T}}(L) \setminus v_i} P_k G(k,j). \tag{2.3}$$

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

$$\Pi_L(i,j) = \frac{P_i G(i,j)}{(N_{\mathrm{r}} + I_L(i,j))}. \tag{2.4}$$

We assume that any two radio units can communicate a packet without error if the SIR is not less than a *reliable communication threshold*, γ_{R}.

Furthermore, 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. For simplicity we assume a fixed transmission power and omni-directional antennas. However, the use of power control and directional antennas is quite easy to include in interference-based scheduling compared with graph-based scheduling. This is important, since it has been shown that adaptive antennas can have a vast improvement on network capacity [11].

2.2 Node and Link Assignment

Traditionally, there are two different assignment methods for STDMA.

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. In a *link-assigned* schedule, the directed link is assigned a slot. A node can then 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 network throughput [12] (at least for unicast traffic).

In this report we will concentrate on link assignment, often referred to as link activation. This is mainly done because it more intuitively (and efficiently) can handle advanced nodes with abilities like power control and adaptive antennas. Furthermore, using link assignment we can extend the transmission rights (LET – Link assignment with Extended Transmission rights), which gives huge improvements [13].

In the following, we describe the criteria needed 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 link $(i, j) \in L$ iff $\{i, j\} \cap \{k, l\} \neq \emptyset$. 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.

The assumptions 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, can also be described as that a set of links L and the set of its adjacent links $\Psi(L)$ must be disjoint:

$$L \cap \Psi(L) = \emptyset. \tag{2.5}$$

The signal-to-interference criteria (2.4) gives the following condition

$$\Pi_L(i, j) \geq \gamma_{\mathrm{R}} \ \forall \ (i, j) \in L. \tag{2.6}$$

If the above two conditions, (2.5) and (2.6), hold for a set of links $L \in \mathcal{L}$, we say that the links in L can *transmit simultaneously*.

We will say that the schedule generated by a distributed STDMA algorithm is conflict-free if equations (2.5) and (2.6) are valid for all links in any time slot.

2.3 Graph-based Scheduling

The traditional approach in designing reuse schedules is to use a graph model of the network.

Given a graph, a reuse schedule can be obtained by studying the set of edges. One problem with this approach is that, depending on how the graph is chosen, it may result in schedules with serious interference in terms of SIR.

In the graph-based method, a graph representation G_γ is chosen. To represent the radio network as a directed graph, we denote by G_γ the directed graph that is obtained by defining the set of nodes $\mathcal{V}$ 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 γ.

The schedule is then designed from the graph G_γ. Interferences from other nodes are not taken into account. The traditional method for link assignment given a set of edges is to say that two links (i,j) and (k,l) can use the same time slot if and only if:

- $i, j, k,$ and l are all mutually distinct, and
- $(i,l) \notin E$ and $(k,j) \notin E$.

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. For a more precise description of this problem see [14].

Observe that the above criteria are not sufficient to guarantee that the assignment is conflict-free in terms of SIR, as defined in the previous section. The assignments that fulfil the above two criteria do not necessarily fulfil the SIR criterion given in condition (2.6). They may therefore not be able to *transmit simultaneously* according to our definitions. We illustrate this with a small example.

Example 1

In figure 2.1 we see the edges obtained for a sample network by choosing the threshold γ_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, the SIR calculated at node 5 will only be 1.6 dB. This is because the SNR between node 5 and 8 is just below what is needed for communication, and SNR between 5 and 7 is just above.

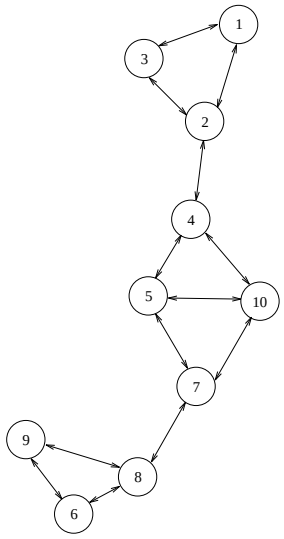

Figure 2.1: The graph G_γ obtained for $\gamma = 13$ dB of a small sample network consisting of ten nodes.

From the example we see that the graph approach if applied as above will result in serious interferences.

However, graph-based algorithms can still be useful. One method of avoiding the serious interference levels shown in the example above is to base the schedule on a graph where node pairs with SNR less than γ_{C} are also included as interference edges [15]. The edges with SNR lower than γ_{C} will not be assigned time slots, but only be used in the test criterion. By considering a graph G_γ and letting γ take a value γ_{I} smaller than γ_{C}, the set of edges will contain not only the links but also interference edges, which represents 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 γ_{I} the *interference threshold*. This is a threshold that will also be used for distributed interference-based scheduling.

This two-level graph model can be used to create interference-free schedules. However, the two-level graph-based scheduling must be done in a more careful manner than interference-based scheduling, since it has less information and thereby needs higher margins to generate conflict-free schedules [16]. Up to one third of the network capacity can be lost. This is highly undesirable, es-

pecially if the network changes slowly. In such cases it is not necessary to send information about the network often, which results in low overhead independent of the used model.

Notice that an alternative way to do graph-based scheduling is to assume that there are interference edges from all nodes at exactly two hops distance. By doing this we include most of the worst interference edges with the additional benefit that we do not have to determine exactly which interference edges there are. This is a more practical way to include interference information than the two-level graph we describe above since it might be difficult to detect all interference links (or at least detemine which one that is the sender).

A drawback with this method is that it may overestimate the interference caused by two-hop neighbors thereby give a more careful form of scheuling (assigns less time slots) than the two-level graph model (which is more careful than interference-based scheduling). We may also have interferences not considered from nodes further away. To alleviate this last problem, this method can be generalized to all nodes at k hops distance, but that of course makes the scheduling even more careful. This method has for example sometimes been used in USAP [8].

2.4 Traffic in Multi-hop Networks

The relaying of traffic in multi-hop networks causes a considerable variation of the traffic load on the links. To achieve large capacities, efficient traffic-controlled schedules have to compensate for this problem.

In a traffic-controlled schedule, links or nodes can use several slots, see [17], according to the traffic load. We define h_{ij} as the number of slots allocated to link (i, j) within a frame in a schedule.

In our traffic model we assume point-to-point traffic, i.e. a packet entering the network has only one destination. However, it is easy to expand the traffic model to other forms of traffic, e.g. multicast traffic.

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 probabil-

ity 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 variations caused by the input traffic model.

Let λ 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 cannot be calculated until the traffic has been routed.

Now, let R denote the routing table where the list entry $R(v, w)$ at v, w is a path from entry node v to exit node w. Let the number of paths in R containing the *directed* link (i, j) be equal to Λ_{ij}.

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

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

The maximum traffic load giving bounded packet delay is commonly referred to as the throughput of the network. We define the throughput as the number λ^* for which the following expressions hold for all traffic loads λ

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

The maximum throughput for a link assigned schedule can be written as [13]

$$\lambda_{\mathrm{L}}^* = \min_{(i,j)} \frac{N(N-1)h_{ij}}{T_{\mathrm{L}}\Lambda_{ij}}, \tag{2.7}$$

where T_{L} is the length of the link-assigned schedule, i.e. the number of time slots in the schedule.

Chapter 3

Distributed STDMA Algorithms

In this chapter we will give a more formal definition of the properties we seek in a distributed algorithm. We will also describe the existing algorithms for distributed STDMA scheduling with their advantages and disadvantages for our purposes. Specifically, we will describe some of the newer (and most interesting) algorithms.

Most older algorithms are either node or link assigned, commonly referred to node and link activation, whereas many of the newer algorithms 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. This is one of the reasons most newer algorithms are adapted for both. As previously mentioned, in this report we will concentrate on link assignment.

In the following we list some of the desired properties of a distributed STDMA algorithm.

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. *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 [7]. 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.

6. *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. In fact, in order to use a graph-based model we need to be more "careful" in our scheduling, resulting in much lower efficiency. Furthermore, a graph-based model has more difficulty in handling properties 7 and 8.

7. *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.

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.

Most older algorithms need at least information about the number of nodes and node density. Actually, several of the papers, see e.g. [18, 19], 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 older algorithms [18, 20, 21] we end up with a schedule of fixed length (or less than this length) depending on the network density and size.

Newer algorithms use different approaches. The most interesting of these is probably what we could call the 2^n principle, used in [10, 22]. 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.

However, it is difficult to say how well this works and how efficient sched-

ules are created if we have large changes in the number of nodes and/or density, but it is a method worth investigating further.

Traffic control is only rarely included in the algorithms. Although many newer algorithms include the ability to assign more than one time slot to a link or node, see e.g. [10, 23], a more specific description of how and when some of the links receive extra time slots is usually omitted. One exception is [24], where each link can request a bandwidth, although the bandwidth it receives is proportional to its request compared with the other links' requests.

Another solution is described in [25], where *virtual circuits* are assigned time slots (or rather each link along the virtual circuit). This gives traffic sensitivity because a link can carry more than one virtual circuit.

Most distributed STDMA algorithms assume a very simple graph model, which can give poor results [16], 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 throughput compared with an interference-based model. One exception to the use of graph-based scheduling is [26], 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 SIR 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 end up with the same information as a graph model gives.

No existing algorithm so far has considered heterogeneous nodes, although some investigation on more advanced node capabilities such as adaptive antennas has been done [27].

3.1 Common Solution

Several different proposals for a distributed STDMA algorithm exists, but many of them have a similar principle. We will first describe this common algorithm,

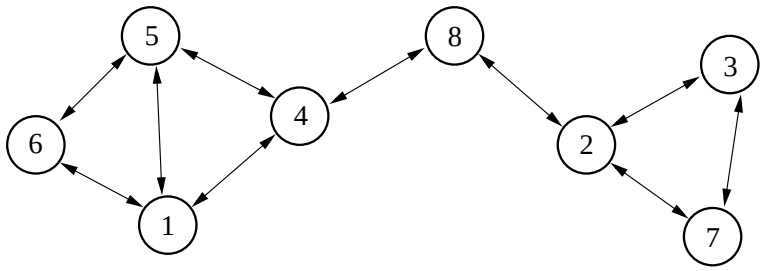

Figure 3.1: An 8-node network.

after which we will continue by describing how the different algorithms differ.

This common algorithm can be described by the following steps:

- Nodes that have entered the network exchange local information with their neighbors (depends on the algorithm but is usually a list of neighbors and of assigned slots and their priority (node identity))
- The node/link with highest priority (without an assigned slot) in its local surroundings assigns itself a time slot.
- The local schedule is then updated, and a new node/link has highest priority. This process is then continued until all nodes receive the number of time slots they are supposed to have; usually one because few of these algorithms are traffic sensitive.

Two different versions of this class of algorithms can be seen. One method is for a node to wait until all links of higher priority have received a time slot and then assign itself a time slot (usually the lowest numbered). Another alternative is to finish the scheduling for each time slot, i.e. each link waits until others of higher priority are assigned or blocked. If any of the other links are assigned, the link will be blocked, otherwise it may assign itself the time slot. In practice these two alternatives give the same result, but they are implemented in slightly different manners.

In figure 3.1 we give a small example of this. For illustrational purposes we use node assignment here. This is because we can show how the algorithm works with a smaller network and fewer time slots. For the first time slot, no

node has any time slot assigned to it. We assume that the node with the highest node ID has highest priority. When we use a graph-based model, the local neighborhood for node assignment is a two-hop radius. No node two hops away can be allowed to transmit, since the intermediate node can be the receiver. A node three hops away, or more, will not affect any of the possible receivers.

In this example, two nodes have highest priority in their local neighborhood in the first time slot. These two nodes are number 6 and 8, and they assign themselves the first time slot. In the second time slot these nodes have been removed from competition, and nodes 5 and 7 now have the highest priority. These are also assigned and consequently removed. In time slot three, we have nodes 4 and 3. Finally, the last slot is number four, in which nodes 1 and 2 receive their time slots.

One difference between algorithms of the common solution is how the link in step two is chosen, i.e. which node has the highest priority in a time slot. The usual method [18, 25, 28] is to assume that all links have some kind of pre-defined priority and the highest priority will win (as in the example). Another method to determine priority is to study how many other links a link is blocked by. The link that can be blocked by most other links will have the highest priority [22]. The effect will be that links that are difficult to schedule will be scheduled first, thereby avoiding time slots in the end of the frame with only one assigned link.

In [14] the scheduling starts with a schedule where each node has its own time slot, i.e. the usual TDMA slotting. Then the common algorithm is run in each of these time slots to increase spatial reuse. The TDMA slots can then be used to transmit the information required for the common algorithm. This solution obviously has problems with changes in the number of nodes in the network, and unlike most other algorithms it is not easy to transform into link assignment. Node ID is used as priority here (lowest ID - highest priority). This gives highly unfair schedules, since the node with the lowest node ID will be assigned to all slots in which it is not blocked by the owner of the time slot. In [20] a reservation channel is used instead for the nodes to assign themselves time slots.

The other big difference between all these algorithms is in how information is distributed in the local neighborhood. Exactly how this is done is not always clearly defined, TDMA slots in [14] are one example, another is [20], which assumes a separate reservation channel that is used to assign time slots. The

control channel is divided into a request segment and a confirmation segment, both of which are divided into N slots, one for each node. The earlier mini-slot a node has in these segments the higher priority the node has. This is mainly described for a situation where reservation is done for each time slot. However, they also describe how to make long-term assignments. This reservation method gives nodes with mini-slots early in the reservation phase a higher likelihood of assigning themselves many time slots. In order to avoid unfairness, they suggest that a round robin algorithm should be used, i.e. the priority of the nodes is cycled. This solution can also be used in [14].

In [25] some of the time slots in the frame belong to a control segment. Three consecutive time slots of these control slots belong to each node. In the first of these slots *(request time slot)* a node v_i may send a request for an assignment of a forward link (i, j) (including information on slots in which this might be possible). In the second slot *(announce time slot)* the receiver of the link, node v_j, answers with an assignment of slots which is possible to use for both nodes. In the third slot *(confirm time slot)*, node v_i transmits the assignment again (and more information). The last information is mainly to update v_is neighbors that might not have been able to hear node v_j.

Other methods for updates also exist, but they are all based on a simple graph model. It is not always easy to upgrade these methods to more advanced network models, even to a two-level graph model.

In general, if we assume a graph-based model, the common solution requires knowledge about all two-hop neighbors and how they are scheduled.

3.2 PANAMA

The Pairwise Link Activation and Node Activation Multiple Access (PANAMA) protocol is so far the most advanced of a series of algorithms developed by L. Bao and J.J Garcia-Luna-Aceves. The most interesting aspect of these algorithms is that a node only needs information about which two-hop neighbors it has, i.e. it requires no information about the scheduling of other nodes, as algorithms of the common solution would need.

The Collision-free Topology-dependent Multiple Access (CTMA) [24] protocol was the first of these algorithms. It is node assigned scheduling with no traffic adaptivity. Time is divided into blocks, sections and parts. One specific

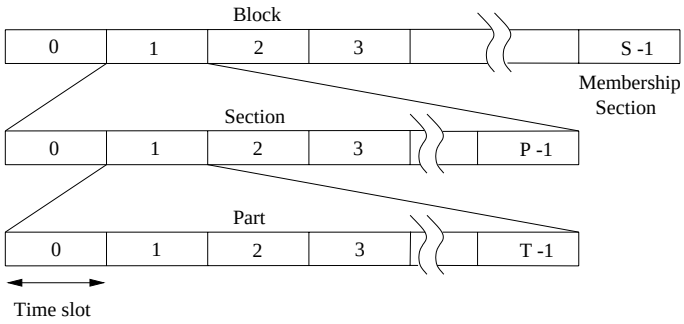

Figure 3.2: The CTMA slot schedule.

section of a block is a membership section used for updating neighbor information. This is shown in Figure 3.2.

A so-called message digest md is calculated for each node with the help of a section seed and ID number for itself and its two-hop neighbors.

- A node is only allowed to transmit in one (chosen) part of each section.
- Within this part, a node transmits in time slot $t = md \bmod t_p$, where t_p is the length of a part.
- If two or more nodes contend for a time slot (they have the same t), they concatenate their md and node ID, and the node with the highest value wins the slot.
- If no node has occupied the slot within the local area of a node, any node may compete for the slot using a similar principle as that in step three. However, a node may only compete for slots in this way in its chosen part.

As long as a node has correct information about its local neighborhood, all these calculations can be done simultaneously in each node, and exact scheduling information is therefore not necessary.

In [29], CTMA have been developed into three different algorithms: Node Activation Multiple Access (NAMA), Link Activation Multiple Access (LAMA), and Pairwise link Activation Multiple Access (PAMA). NAMA is close to CTMA,

but the frame structure have been changed; a special periodic random access slot is added after a number of regular scheduled slots. Nodes use their regular slots for neighbor information if possible and random medium access control of the special slot if this is not possible, e.g. new nodes.

LAMA and PAMA uses receiver-oriented Direct-Sequence Spread Spectrum (DSSS) in addition to time slotting, i.e. a transmitter chooses a code corresponding to the receiving node. PAMA is fully link assigned scheduling, whereas LAMA allocates groups of outgoing links, which is useful for multicast traffic. The DSSS codes (as well as scheduling time slots) are determined with a similar principle as the message digests for CTMA. There is no description of how neighbor information is updated for these protocols, but since neither of them is scheduled to reach all neighbors with one transmission, they probably require more overhead than NAMA.

These algorithms acheive traffic adaptivity by assigning multiple *pseudo identities* for each node or link.

The PANAMA protocol [30] was developed to handle unidirectional links, but also has other improvements of the previous algorithms. Instead of a message digest, a pseudo-random function giving values between 0 and 1 is used. The priority of a link (i, j) is then given as $p = \sqrt[Bw]{Rand(t + i + j)}$, where Bw is the required bandwidth of the link and t is the time slot. Traffic adaptivity can therefore be done without pseudo-identities.

Time is assumed to be divided between node activation (NAMA-UN) and link activation (PAMA-UN), where UN stands for Uni-directional Networks. Both of these use DSSS, but with transmitter-oriented codes unlike LAMA and PAMA, i.e. each transmitter is given a code of its own.

Except for the dividing in time between the assignment methods, the frame structure of PANAMA is similar to NAMA and PAMA.

The overhead of all these algorithms is minimal. However, this minimal amount of information also gives very little information for the scheduling to perform well under relatively static situations and to take advantage of more advanced nodes. For example, if a node has lower priority than another node in its local neighborhood, it will refrain from transmitting even if the other node with higher priority cannot transmit because of another node with even higher priority further away. For high mobility cases, it might be interesting, though, since overhead is minimal.

Another problem can be delay guarantees, since it is difficult to predict ex-

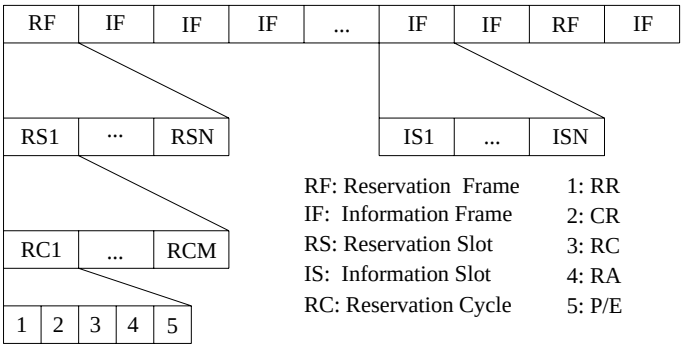

Figure 3.3: The FPRP slot schedule.

actly when a node or link will be able to transmit due to the unpredictability of a pseudo-random function.

3.3 Five-Phase-Reservation Protocol and E-TDMA

These algorithms were developed by C. Zhu and M.S. Corson [23, 31, 32]. In these algorithms, nodes may reserve slots by contention-based access. We start with a description of the Five-Phase-Reservation Protocol (FPRP) and then continue with Evolutionary TDMA (E-TDMA).

The FPRP is a contention-based protocol that uses a five-phase reservation cycle to establish TDMA slot assignment. The frame structure of FPRP starts with a Reservation Frame (RF) and continues with a sequence of Information Frames (IF). Both of these frame types consist of N slots. Each reservation slot is dedicated to the reservation of a corresponding information slot (in all following IFs until next RF). A reservation slot is composed of M Reservation cycles, which each consist of a five-phase dialog. The slot structure of FPRP is shown in figure 3.3.

These five phases are:

1. *(Reservation Request phase - RR)* If a node wants to make a reservation, it sends a message during this phase with probability p. Such a node is referred to as a Requesting Node (RN). The rest of the nodes listen to the

channel for RR packets.

2. *(Collision Report phase - CR)* A node can detect zero, one or more RR packets. If more than one is received, it is detected as a collision. Nodes that detect such collisions send CR packets.

3. *(Reservation Confirmation phase - RC)* If an RN hears no CR packets, it assumes there were no collisions and it will become a Transmission Node (TN). Such nodes send RC packets to its neighbors.

4. *(Reservation Acknowledge phase - RA)* In this phase a node acknowledges an RC it has received. If a TN is not connected to any other node, it will not receive an RA packet and the node will be aware of its isolation. The same result also happens if the node's only neighbor is another TN. It also informs nodes two hops away that the node has become TN in the slot.

5. *(Packing/Elimination phase - P/E)* Every nod that is two hops away from a TN sends a Packing packet, because such nodes cannot contend for the slot anymore and this can be used for changing the contention probability for nodes three hops away. TNs send elimination packets with a probability of 0.5 in case there is another TN adjacent to it. If a TN receives such a packet, it will receive in the slot instead.

FPRP has a significant overhead, and there is always a risk that the assignment will fail. The problem with random updates is that slots may be unused because of colliding packets.

Evolutionary TDMA (E-TDMA) is an extension of FPRP, which is more advanced [23]. The TDMA channel here is divided into two parts, interleaved in time: the control epoch and the information epoch. In the control epoch, nodes follow the control schedule, which is a node-assigned schedule in which each node is assigned one slot. This slot is used to exchange information between the nodes. In the information epoch, nodes follow the information schedule and the data traffic is transmitted. Both node and link assignment can simultaneously be used in the information epoch. The slot structure of E-TDMA is shown in figure 3.4.

An information epoch has K information frames, and each consists of L information slots. The control epoch has two phases: a contention phase and an allocation phase. A contention phase is divided into N contention slots, which

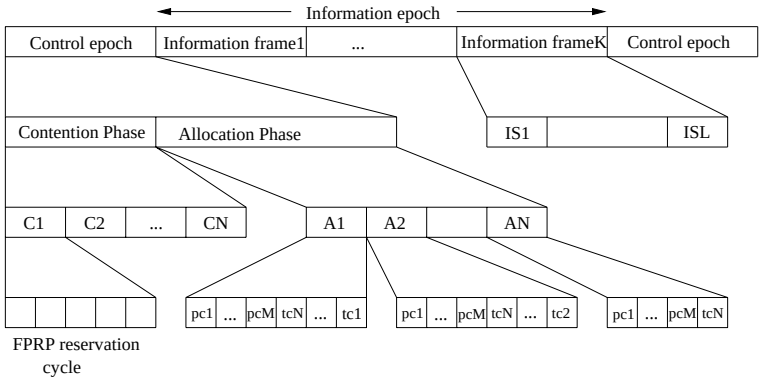

Figure 3.4: The E-TDMA slot schedule.

each consists of a number of FPRP cycles. In the ith contention slot a node can contend for the ith *temporary color*. A temporary color is a permission to reserve new information slots (in the information schedule) or one of the M *permanent colors*. A permanent color is necessary for exchanging scheduling information in the allocation phase.

An allocation phase consists of N allocation frames (A), each corresponding to a temporary color. The first A frame (A1) has $M + N$ slots, corresponding to the M permanent colors and the N temporary colors. Slots corresponding to the temporary colors are placed last in the frame in reversed order. For each consecutive A frame the number of slots decreases with one from the back, so that in the ith A frame the last slot corresponds to the ith temporary color. All of the slots prior to this one are used only to transmit information about the schedule and neighborhood. The only new assignments allowed can be done in the last slot. The node can now reserve a permanent color or information slots (or both), since it has all the information it requires to do this.

An advantage of E-TDMA over other reservation protocols [31, 33] is that it only needs to acquire the channel once, even if it needs more than one time slot. Furthermore, unlike FPRP a node or link keeps an assigned time slot and does not have to reserve the channel each time it has packets to transmit.

It is not clear how efficient these algorithms are under realistic situations. They might be especially sensitive to hostile jamming, since in many of the

phases of FPRP a node listens for collisions. For example, if a hostile node transmits noise during phase two, an RN node will sense this as a collision of CR packets, and thereby assume that its assignment failed. This is highly unwanted in a military scenario.

3.4 USAP

The Unifying Slot Assignment Protocol (USAP) was developed by D.Young [8] at Rockwell Collins to manage the TDMA slot and channel assignment in their soldier phone program, i.e. USAP can be seen as multi-channel STDMA. It handles the information needed to create the distributed database of slot allocation information. USAP to the best of our knowledge is the only distributed STDMA algorithm that has been commercially developed and implemented into a functional system.

Rockwell Collins has continued to develop USAP since the first paper was published in 1996 [8]. The most up-to-date paper is [10], which describes USAP Multiple Access (USAP-MA), which is a set of heuristics that run on top of USAP.

The description below will be based on this paper. USAP-MA can be seen as an advanced variant of the common solution. However, most of the algorithm description handles other issues than the actual assignment of time slots.

USAP-MA handles both node and link assignment in a flexible fashion so an instantaneous mix that matches the applications can be achieved. A node shares the following information with its neighbors in order to have the information required for choosing non-conflicting transmit allocations.

- Allocations where a node is transmitting.
- Allocations where a node is receiving.
- Allocations where a node's neighbors are transmitting.

When a node wants to allocate a slot, it knows which slots are available so it simply adds the information to the transmitted data. It can then either wait for acknowledgments from all its neighbors before the slot is used, or it can use it for data traffic immediately if momentary conflicts, caused by mobility or conflicting allocations, can be tolerated.

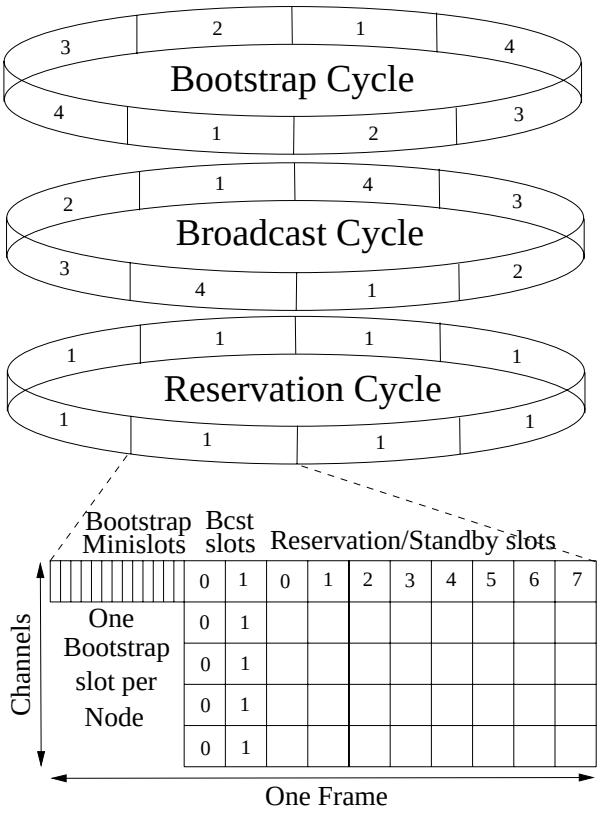

Figure 3.5: The USAP-MA TDMA slot structure.

The USAP-MA frame structure is shown in figure 3.5. Each frame is divided into three parts. First, a number of bootstrap minislots are used for exchanging the network information required for the assignment of the rest of the slots. Second, there are broadcast slots, which are node allocation slots used to support datagram services and other control traffic that nodes need to share.

Third, there are reservation slots, which are link allocation slots. A number of frames form a cycle, and all three parts can have different cycle lengths, i.e. we have a bootstrap cycle, broadcast cycle and a reservation cycle.

An example of the number of slots in each frame and frames in each type

F0												B0	B1	R0 (B0)	R1 (B1)	R2 (B2)	R3 (B3)	R4 (B4)	R5 (B5)	R6 (B6)	R7 (B7)
F1												B2	B3	R0 (B1)	R1 (B2)	R2 (B3)	R3 (B4)	R4 (B5)	R5 (B6)	R6 (B7)	R7 (B0)
F2												B4	B5	R0 (B2)	R1 (B3)	R2 (B4)	R3 (B5)	R4 (B6)	R5 (B7)	R6 (B0)	R7 (B1)
F3												B6	B7	R0 (B3)	R1 (B4)	R2 (B5)	R3 (B6)	R4 (B7)	R5 (B0)	R6 (B1)	R7 (B2)
F4												B0	B1	R0 (B4)	R1 (B5)	R2 (B6)	R3 (B7)	R4 (B0)	R5 (B1)	R6 (B2)	R7 (B3)
F5												B2	B3	R0 (B5)	R1 (B6)	R2 (B7)	R3 (B0)	R4 (B1)	R5 (B2)	R6 (B3)	R7 (B4)
F6												B4	B5	R0 (B6)	R1 (B7)	R2 (B0)	R3 (B1)	R4 (B2)	R5 (B3)	R6 (B4)	R7 (B5)
F7												B6	B7	R0 (B7)	R1 (B0)	R2 (B1)	R3 (B2)	R4 (B3)	R5 (B4)	R6 (B5)	R7 (B6)

Figure 3.6: The USAP slot schedule.

of cycle is also given in [10]. These choices are also given in figure 3.5. Each frame is 125 ms and consists of 13 bootstrap minislots, 2 broadcast slots and 8 reservation slots. The bootstrap cycle is 4 frames. If each node is given a bootstrap slot of its own, 52 users can be handled by the system. However, other solutions than giving each node its own minislot are suggested, and we will discuss them later. The broadcast cycle is 4 frames, which gives 8 nodes that can transmit on a channel in a local neighborhood. The reservation cycle repeats every frame.

If the reservation slots are not allocated in a local neighborhood, they act as standby broadcast slots. For example, as shown in figure 3.6, reservation slot zero R0 acts as broadcast slot for node zero B0 in the first frame F0, reservation slot one R1 acts as broadcast slot for node B1 and so forth. However, since reservations usually last many frames, a reservation would block a specific node if this were repeated in every frame. Instead the corresponding nodes are shifted one step for each frame, i.e. in frame two reservation slot zero R0 acts as broadcast slot for node B1 and so forth.

The result of this is that when there are no active reservations, all slots are node assigned, and as reservations increase, the part of the channel that is node assigned gradually decreases.

A permanent assignment of an own bootstrap slot to each node is the most efficient solution for small networks. However, for large networks it is necessary

to dynamically assign these slots. In a dynamic solution a new node that wants to enter the network starts by listening for the bootstrap information of the other nodes. It then chooses a slot of its own. If no other node has this slot in the local neighborhood, the node will keep it. If conflicts occur, the node will learn this from the other node's USAP information and choose a new slot.

The dynamic assignment of bootstrap slots is based on the 2^n principle previously mentioned in this chapter. The bootstrap cycle length will be determined depending on the density of the local neighborhood. If the nodes try to assign themselves the lowest possible slot index, then the size of the bootstrap cycle can be chosen as the lowest power of two larger than the highest assigned slot.

Some of the more advanced features of USAP-MA deal with the problem of handling changes in network density. This is a very important problem that few algorithms handle well. USAP has two methods for handling such changes: *Adaptive Broadcast Cycles* and *Channelized Neighborhoods*.

Adaptive Broadcast Cycles (ABC) is a similar to the adaptive bootstrap cycles described above. ABC employs nested subdivisions of the fixed length cycle for the nodes to reuse unused slots.

For example, a 4-node neighborhood works as if the broadcast cycle were 2 frames, while a 16-node neighborhood would work as if it were 8 frames. However, the cycle length of the shift of acting broadcast slots can have a different value, see [10] for examples. In the 4-node neighborhood this requires 4 frames. The number of frames in the broadcast cycle will increase with powers of two. As previously mentioned, this also results in an overlap of half of the slots if the broadcast cycle is changed one step up or down. A node continuously looks for broadcast slots with lower index numbers in order to pack all nodes' broadcast slots in a local area into the first half of the schedule, thereby allowing the broadcast cycle to decrease one step.

USAP also uses the method of Channelized Neighborhoods (CN) to handle too large densities for the maximum cycle length, i.e. instead of further dividing the broadcast cycle in time we separate them in channels instead. When a node is not transmitting, it rotates its receiver on the different channels in order to hear from each transmitter. When it has rotated over the different channels, it has heard all other nodes within the neighborhood except those that transmit in the same slots as it does. The number of nodes that does this is the same as the number of channels that is used. Each transmitting node knows which of the others are listening to its transmission so it can choose packets with the correct

receiver. However, traffic that should reach all other nodes must be retransmitted as many times as the number of channels used, unless we can listen to and receive from all channels simultaneously, something that is not assumed in [10]. However, it is easy to conclude that CN would be more efficient if nodes could receive on more than one channel simultaneously, especially for broadcast and multicast traffic.

Nodes that transmit in the same time slot will not be able to communicate directly; instead they must relay their traffic through another node.

USAP is by far the most complete algorithm for creating distributed STDMA schedules. It is even implemented into a functional system. It has several positive properties, e.g. its ability to handle changes in node density. It has been developed under a rather long time for the purpose of implementing it into a product. This means that most problems with algorithms of this type probably have been encountered and dealt with. It is difficult, however, to determine how efficient it is. The conveying of information in the bootstrap slots is quite independent of the situation in the network. As much information (and the same information) will be transmitted in a static network as in a highly mobile network.

Also, USAP in its present form is graph-based scheduling (although a variant that considers interference as described in Section 2.3). It might be possible to change this to interference-based scheduling without too drastic changes, but exactly how to do this is not clear. One possibility is therefore that USAP could be used as a base for more advanced STDMA algorithms, since it has several properties that are highly interesting. It does not fulfill all our required properties in its present form, though.

3.5 Conclusions on Existing Solutions

A lot of work has been done on distributed STDMA, but no existing algorithm fulfills all our required properties. Of the algorithms we have described USAP is the most interesting 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 the algorithms described 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 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 capacity 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 next chapter we will describe an interference-based STDMA algorithm that creates an efficient schedule with a given amount of information about the network. This 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.

To do such an investigation we compare the efficiency of schedules created with different amount of information, ranging from complete information to very little information (Chapter 5). We will also discuss what information it needs and the consequences of limited information. The algorithm must be so general that it can handle all the properties we have described in this chapter.

In future work, we will develop methods to convey this information and investigate how much it will cost in overhead. Finally, we will develop a complete algorithm that includes the control information and gives as efficient schedules as possible in every situation.

Chapter 4

Interference-based Scheduling

This chapter describes the first steps toward an algorithm that fulfills the properties we listed in the previous chapter. Most specifically we will concentrate on its interference-based property.

We will start by describing how the nodes generate the schedule when they have sufficient information about their local neighborhood, and from this which information they require to do so.

4.1 How to create a schedule

We will base our distributed STDMA algorithm on the common method described in the previous chapter. This class of algorithms is most easily updated to handle an interference-based network model. In short, it 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.
- 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 will be made interference-based by using interference information, i.e. we transmitt and use interference information when we decide whether links can transmit simultaneosly. 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.

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 modes: 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. It can then 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 its new priority will be transmitted to its local neighborhood. After this, the link process can stay in this mode or change into one of the others.

- *Waiting*: In this mode, a link wants §to assign itself a time slot, but another link has higher 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.

4.1.1 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, Λ_{ij}. Since both these values are changing, the link priority is constantly changing.

The priority value of a link (i, j) will be $\frac{h_{ij}}{\Lambda_{ij}}$, where the lowest value has the highest priority. The motivation for this comes from the maximum throughput formula (2.7). 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 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.

4.1.2 Theft of Timeslots

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 similiar 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 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.

4.1.3 When does a link change mode?

We have so far described the different modes of a link process and how the priority is calculated, but we have not been very detailed about the circumstances under which a link process changes mode. 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 Λ_{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 Λ_{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.
- *A node disappears from the network:* This can be seen as the removal of several links at once.
- *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.

4.1.4 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 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.

For simplicity, and the fact that we only evaluate throughput in this report, we have chosen the first available time slot when doing the assignment in our simulations.

4.2 What information is required?

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

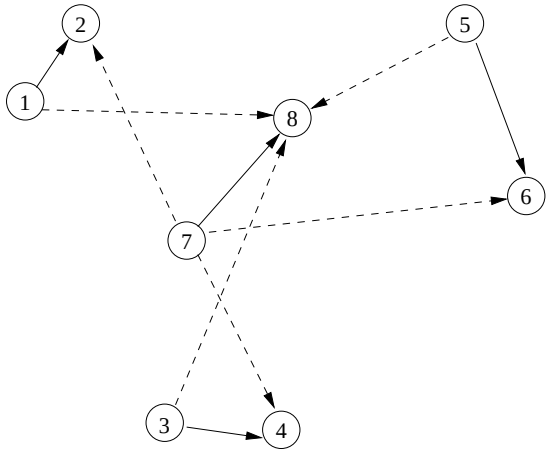

Figure 4.1: A small part of a network.

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.

Second, the interference from the transmitter is not allowed to cause interference problems in any of the other receivers already assigned.

This can be illustrated by the example shown in Figure 4.1. 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) we need the following:

- *Interference - Received Power*

  We need an estimate of the received power from each of the other trans-

mitters, i.e. $P_k G(k, j)$. This can be estimated by measuring the channel. We also assume that the channel is equal in both directions, that is

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

This assumption can then be used when a node determines whether its transmission will cause problems for somebody else.

If the received power level from a transmitter is below a value δI, it is assumed to be zero by the algorithm, i.e. we assume that such nodes do not affect one another. We will later discuss what effect they actually have, but this is a necessary assumption if we want the local neighborhood to be smaller than the entire network, because if δI is set to zero, we have all the information about the network. We will use the interference threshold, defined in Chapter 2, when we give the size of this parameter, i.e. $\gamma_I = \delta I / N_r$.

- *Local Schedule*

  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 τ, i.e. the largest value of I_{max} such that the following inequality is still valid

$$\frac{P_k G(k, l)}{(N_{\text{r}} + I_L(k, l) + I_{\text{max}}(l, \tau))} \geq \gamma_{\text{R}}.$$

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

$$P_i G(i, l) < I_{\text{max}}(l, \tau)$$

  for all assigned receivers v_l in time slot τ.

  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 the we have extra interference from outside the local neighborhood and the interference measured should be used while this lasts.

- *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.

With this information we have sufficient information for both sender and receiver to determine when to be active and which time slots can be assigned.

The only further information required is that the sender informs the receiver which slots that are available for the transmitter to use without causing too much interference. The receiver can then assign the time slot. Information about this is then propagated to the local neighborhood.

4.2.1 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 little 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 γ_R in many receivers. The additional interference from outside the local neighborhood can then be sufficient to lower the actual SIR below γ_R. One way to avoid this is to use an interference margin I_{margin}, i.e. we assume that there will be external interference and therefore avoid scheduling the SIR as low as γ_R. The link is then assumed to be able to use the time slot if:

$$\frac{P_i G(i,j)}{(N_{\text{r}} + I_L(i,j) + I_{\text{margin}})} \geq \gamma_{\text{R}}.$$

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, additional solutions may be required.

Another alternative (or complement) 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 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 capacity 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. Although, this would give a better form of traffic adaptivity than what is described above, we will leave the implementation of this for future work.

In the next chapter we will study the consequences of the two first methods.

4.3 Concluding Remarks

In this chapter we have described an interference-based algorithm that creates efficient schedules with given information about the network. This can be seen as the first step in the development of an algorithm that attempts to meet the requirements we stated in the previous chapter. Adding interference-based scheduling to STDMA algorithms will be an important part of making STDMA efficient and competitive.

In the next chapter we will investigate how efficient schedules can be created with different information about the network.

The part that is missing, if we want this algorithm to be practical, is a more exact description on how and when information is spread to the local neighborhood. Another is how the local frame length is chosen and updated. However, these are problems other distributed algorithms have solutions for, so we should

be able to add such properties later.

Chapter 5

Evaluation

In this chapter we evaluate the algorithm given. We will study the maximum throughput of the schedules created by our algorithm and compare this to the results of a centralized approach. The maximum throughput will be evaluated for simulated networks of different connectivity and size.

5.1 Simulation setup

In the comparisons, 25 networks of three different connectivities (low, medium, and high) have been generated for networks of size 10 and 20 nodes. We have also generated 25 networks of size 40 nodes with low connectivity. The connectivity is varied by changing the transmission power, P, for a network. All networks are connected, i.e. there is always a multi-hop path between any pair of nodes.

To generate realistic networks, a terrain-data-based wave propagation model, Vogler's five knife-edge model, has been used to calculate the basic transmission path loss, see [34] for more details.

The centralized algorithm we compare with is a centralized version of our distributed algorithm described in the previous chapter. This means that the schedule is generated by one specific node in the network that contains all knowledge about the network, i.e. no local neighborhoods. This algorithm is obviously also of the common type with the same link priorities as the distributed method. The theft of time slots is not included, though.

In short the centralized algorithm can be described with the following steps: For each time slot:

- Sort all links (i, j) in a list A according to priority h_{ij}/Λ_{ij}.
- Choose the node/link with highest priority which has not yet been checked in the time slot. Assign it to the time slot if possible. Repeat until all links have been tested in the time slot.

Some additional assumptions are also made:

The shortest route, 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.

For simplicity we assume that the available data rate is linearly dependent on the SIR close to the γ_R threshold. Once the proper data rate is chosen, we assume that all packets are perfectly received, and no retransmissions are considered.

Simulations show that the factor I_{margin} can have a considerable impact for some networks but very little effect for others. Since we decrease the data rate for interfered nodes, the throughput can be limited by a single link. If a high value of I_{margin} removes some of the interference, this link may have higher throughput, which results in higher network throughput. On the other hand, if the limiting link is not interfered with by external interference, increasing I_{margin} may have negative impact. It is probably best to choose a different I_{margin} for each node in the network, but how this should be chosen is left for future work.

In our simulations we have chosen γ_C equal to 15 dB and γ_R equal to 10 dB. Furthermore, we vary I_{margin}/N_r from zero to two and for each network and choice of γ_I choose the value that gives the highest throughput. For small networks the value of I_{margin}/N_r will most often be zero.

5.2 Results

In Figures 5.1, 5.2, and 5.3 we plot the ratio between throughput of our distributed algorithm and the centralized variant for different choices of γ_I. Figure

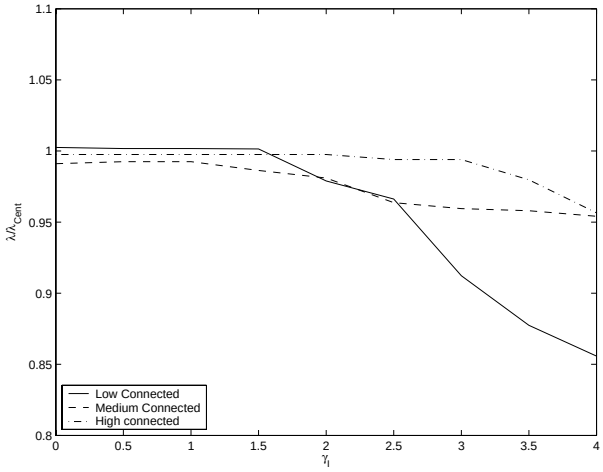

Figure 5.1: The ratio between throughput of schedules from the distributed algorithm and the centralized. The ratio is plotted for 25 networks of size 10 nodes for different values of γ_I. This is done for low, medium, and high connectivity.

5.1 is for 10-node networks, Figure 5.2 is for 20-node networks, and Figure 5.3 is for 40-node networks.

Two things can be noted in these plots. First, if we use γ_I equal to zero, i.e. the algorithm has all information about the network (the same as the centralized), the distributed algorithm achieves the same throughput as the centralized. Although the same information exists in the network in this case there will be no single node that will have all information. This result is important, because it means that the distribution of information will not have negative consequences for the capacity of the network.

Second, we see a decrease in the capacity of the network when γ_I is increased. This means that the algorithm improves its performance with more information about the network. This is an important property if we want to fulfill property 7, i.e. the ability to adapt to mobility.

We also note that the loss of capacity increases with the size of the network. This is probably because there are more nodes that can cause interference that the algorithm does not consider. A better form of traffic adaptivity will probably

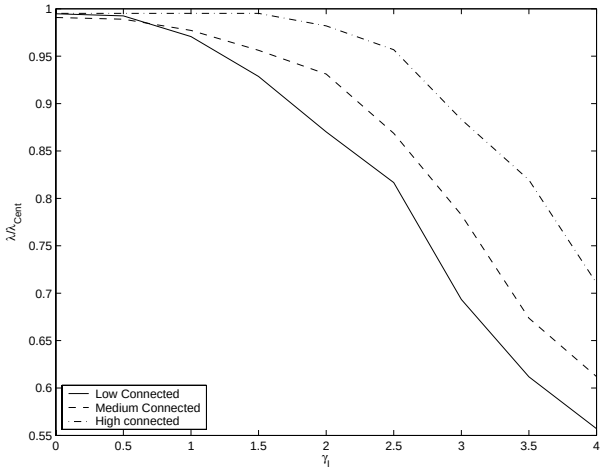

Figure 5.2: The ratio between throughput of schedules from the distributed algorithm and the centralized. The ratio is plotted for 25 networks of size 20 nodes for different values of γ_I. This is done for low, medium, and high connectivity.

improve the situation considerably.

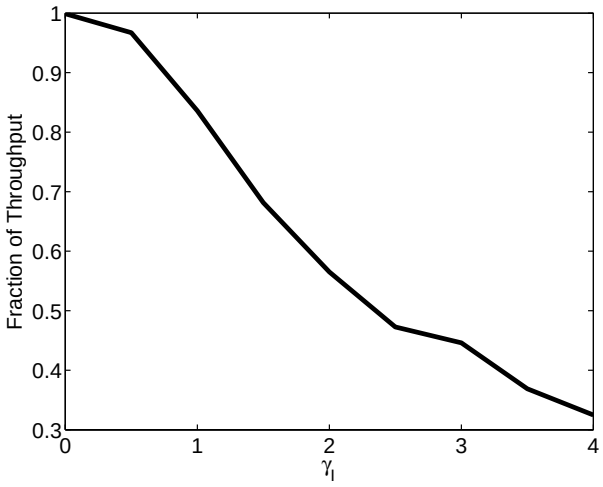

Figure 5.3: The ratio between throughput of schedules from the distributed algorithm and the centralized. The ratio is plotted for 25 networks of size 40 nodes for different values of γ_I. This is done only for low connectivity.

Chapter 6

Concluding remarks and further work

In this report, we have listed the properties a distributed STDMA algorithm should have to be efficient in a military scenario. We have also studied the existing STDMA algorithms and seen what methods they use to fulfill these properties. Although much work has been done in this area, none of the existing algorithms can easily be used to create a sufficiently efficient distributed STDMA algorithms. Of the algorithms we have described, USAP is the most interesting, but not even this algorithm has all the listed properties that are desired for reliable and efficient communications on the battlefield.

However, although none of these algorithms can meet all our required properties, they have functions that are helpful when we design such an algorithm.

The main contribution of this report has been the description of the central parts of an interference-based distributed STDMA algorithm. This algorithm is the first distributed algorithm that uses an interference-based model of the network, which is important since this is a property that can make STDMA really efficient and competitive.

Evaluations of our algorithm show that it can achieve as high network capacity as a centralized algorithm can do with the same information. Therefore, the distribution of information will not have negative consequences for the capacity of the network.

We have also shown that our algorithm can handle different amounts of

information about the network, and that it decreases its performance with information reduction. These are important properties if we want to be able to adapt to mobility.

6.1 Further Work

Some development remains to be done before our algorithm can be used in a real scenario. One part required is a more exact description of how and when information is spread to the local neighborhood. Another is how the local frame length is chosen and updated.

Methods used by existing algorithms, e.g. USAP, can be used to spread information to the local neighborhood. An important step will also be to investigate how much it will cost in overhead for different sizes of local neighborhood. With such information we can determine what size of local neighborhood is most efficient for STDMA scheduling. The best size of local neighborhood will probably be dependent on both network size, connectivity, mobility, and link data rates.

Bibliography

[1] L. Kleinrock and F. Tobagi, "Packet switching in radio channels part I – carrier sense multiple-access modes and their throughput-delay characteristics," *IEEE Trans. Commun.*, vol. 23, pp. 1400–1416, Dec. 1975.

[2] IEEE Computer Society LAN MAN Standards Committee, *IEEE standard for wireless LAN medium access control (MAC) and physical layer (PHY) specifications*, IEEE Std 802.11-1997, The Institute of Electrical and Electronics Engineers, New York, 1997.

[3] J. Sobrinho and A. Krishnakumar, "Quality-of-service in ad hoc carrier sense multiple access wireless networks," *IEEE Journal on Selected Areas of Communications*, vol. 17, no. 8, pp. 1353–1368, Aug. 1999.

[4] R. Nelson and L. Kleinrock, "Spatial-TDMA: A collision-free multihop channel access protocol," *IEEE Trans. Commun.*, vol. 33, no. 9, pp. 934–944, Sept. 1985.

[5] N. Funabiki and Y. Takefuji, "A parallel algorithm for broadcast scheduling problems in packet radio networks," *IEEE Trans. Commun.*, vol. 41, no. 6, pp. 828–831, June 1993.

[6] B. Hajek and G. Sasaki, "Link scheduling in polynomial time," *IEEE Trans. Inform. Theory*, vol. 34, no. 5, pp. 910–917, Sept. 1988.

[7] J. Grönkvist, "Traffic controlled spatial reuse TDMA for multihop radio networks," in *Proc. of IEEE PIMRC'98*, 1998, vol. 3, pp. 1203–1207.

[8] D. Young, "USAP: A unifying dynamic multichannel TDMA slot assignment protocol," in *IEEE MILCOM*, 1996.

[9] R. Bittel et al., "Soldier phone: An innovative approach to wireless multimedia communications," in *IEEE MILCOM*, 1998.

[10] D. Young, "USAP multiple access: Dynamic resource allocation for mobile multichannel wireless networking," in *IEEE MILCOM*, 1999.

[11] F. Eklöf et al, "On the performance of antenna arrays in spatial reuse TDMA ad hoc networks," in *Proc. of MILCOM'2002*, 2002.

[12] J. Grönkvist, "Assignment methods for spatial reuse TDMA," in *Proc. of MobiHOC 2000*, 2000.

[13] J. Grönkvist, *Assignment Strategies for Spatial Reuse TDMA*, Tech. lic. thesis, Radio Communication Systems, Department of Signals, Sensors and Systems, Royal Institute of Technology, SE-100 44 Stockholm, Sweden, Mar. 2002.

[14] A. Ephremides and T. Truong, "Scheduling broadcasts in multihop radio networks," *IEEE Trans. Commun.*, vol. 38, no. 4, pp. 456–460, Apr. 1990.

[15] C. Prohazka, "Decoupling link scheduling constraints in multihop packet radio networks," *IEEE Trans. Comput.*, vol. 38, no. 3, pp. 455–458, March 1989.

[16] J. Grönkvist and A. Hansson, "Comparison between graph-based and interference-based STDMA scheduling," in *Proc. MobiHOC*, 2001.

[17] J. Shor and T. Robertazzi, "Traffic sensitive algorithms and performance measures for the generation of self-organizing radio network schedules," *IEEE Trans. Commun.*, vol. 41, no. 1, pp. 16–21, Jan. 1993.

[18] I. Chlamtac and A. Lerner, "A link allocation protocol for mobile multihop radio networks," in *GLOBECOM '85, IEEE Global Telecommunications Conference, Conference Record*, 1985, vol. 1, pp. 238–242.

[19] I. Chlamtac and S. Pinter, "Distributed nodes organization algorithm for channel access in a multihop dynamic radio network," *IEEE Trans. Comput.*, vol. 36, no. 6, June 1987.

[20] I. Cidon and M. Sidi, "Distributed assignment algorithms for multihop packet radio network," *IEEE Trans. Comput.*, vol. 10, no. 38, pp. 1353–1361, Oct. 1989.

[21] I. Chlamtac and S. Pinter, "Distributed nodes organization algorithm for channel access in multihop dynamic radio network," *IEEE Trans. Comput.*, , no. 6, pp. 728–737, June 1987.

[22] R. Liu and E. Lloyd, "A distributed protocol for adaptive link scheduling in ad-hoc networks," in *Proc. of IASTED international Conference on Wireless and Optical Communications (WOC'2001)*, Banff, Canada, June 2001, pp. 43–48.

[23] C. Zhu and M.S. Corson, "A new protocol for scheduling TDMA transmissions in mobile ad hoc networks," Tech. Rep. CSHCN TR 2001-19, Center for Satellite and Hybrid Networks, University of Maryland, 2001, http://www.isr.umd.edu/CSHCN.

[24] L. Bao and J.J. Garcia-Luna-Aceves, "Collision-free topology-dependent channel access scheduling," in *Proc. of IEEE MILCOM*, 2000, vol. 1, pp. 507–511.

[25] L. Pond and V. Li, "A distributed time-slot assignment protocol for multihop broadcast packet radio networks," in *MILCOM*, 1989.

[26] T. Shepard, "A channel access scheme for large dense packet radio networks," in *ACM SIGCOMM*, 1996.

[27] L. Bao and J.J. Garcia-Luna-Aceves, "Transmission scheduling in ad hoc networks with directional antennas," in *Proc. Mobicom*, 2002.

[28] I. Chlamtac and S. Kutten, "A spatial reuse TDMA/FDMA for mobile multi-hop radio networks," in *Proceedings of IEEE INFOCOM '85*, 1985, vol. 1, pp. 389–394.

[29] L. Bao and J.J. Garcia-Luna-Aceves, "Distributed dynamic channel access scheduling for ad hoc networks," *Journal of Parallel and Distributed Computing, Special Issue on Wireless and Mobile Ad Hoc Networking*, 2002.

[30] L. Bao and J.J. Garcia-Luna-Aceves, "Channel access scheduling in ad hoc networks with unidirektional links," in *Proc of DIALM*, 2001, pp. 9–18.

[31] C. Zhu and M.S. Corson, "A five-phase reservation protocol (FPRP) for mobile ad hoc networks," *Wireless Networks*, , no. 7, pp. 371–384, 2001.

[32] C. Zhu and M.S. Corson, "An evolutionary-TDMA scheduling protocol (E-TDMA) for mobile ad hoc networks," Tech. Rep. CSHCN TR 98-14, Center for Satellite and Hybrid Networks, University of Maryland, 1998, http://www.isr.umd.edu/CSHCN.

[33] Z. Tang and J.J. Garcia-Luna-Aceves, "A protocol for topology-dependent transmission scheduling in wireless networks," in *WCNC*, 1999.

[34] B. Asp, G. Eriksson, and P. Holm, "Detvag-90® — Final Report," Scientific Report FOA-R–97-00566-504–SE, Defence Research Est., Div. of Command and Control Warfare Tech. Linköping, Sweden, Sept. 1997.