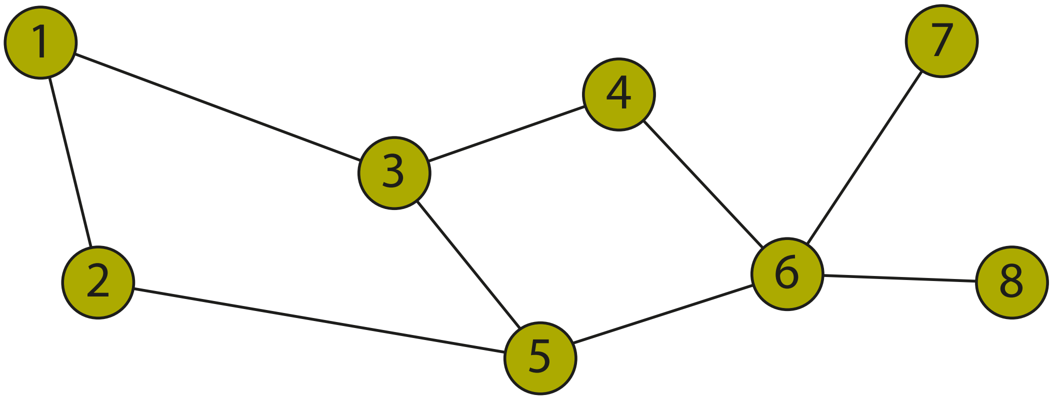
Distributed Spatial Reuse Time Division Multiple Access

Algorithm Description

ARWID KOMULAINEN, JIMMI GRÖNKVIST AND ULF STERNER



1	2	3	4	5	7	3
6	4	8	2		1	7

FOI-R--3960--SE
ISSN 1650-1942 October 2014

Arwid Komulainen, Jimmi Grönkvist and Ulf Sterner

Distributed Spatial Reuse Time Division Multiple Access

Algorithm Description

Titel	Distribuerad Spatial Reuse Time Division Multiple Access Algoritmbeskrivning
Title	Distributed Spatial Reuse Time Division Multiple Access Algorithm Description
Rapportnr / Report No.	FOI-R--3960--SE
Månad / Month	Oktober / October
Utgivningsår / Year	2014
Antal sidor / Pages	36
ISSN	1650-1942
Kund / Customer	FM
Forskningsområde	4. Informationssäkerhet och kommunikation
FoT område	Ledning och MSI
Projektnr / Project No.	E36055
Godkänd av / Approved by	Christian Jönsson
Ansvarig avdelning	Informations- och aerosystem

Abstract

In this report we give an overview of the Spatial Reuse Time Division Multiple Access (STDMA) algorithms presented in current literature. We list the requirements that should be fulfilled by STDMA algorithms in order to make them viable in military networks. Seeing as none of the algorithms currently available meet the requirements listed, we present an STDMA algorithm developed at FOI that is tailored to military ad hoc networks. The algorithm presented is a fully distributed, traffic adaptive algorithm, able to generate schedules with spatial reuse in mobile networks. This STDMA algorithm can be of great use in simulation studies and in performance evaluations and comparisons of commercially available solutions.

Keywords: ad hoc networks, distributed, STDMA, scheduling

Sammanfattning

I denna rapport ges en översikt av de schemaläggningsalgoritmer med spatiell återanvändning (STDMA-algoritmer) som finns beskrivna i litteraturen. Vi visar på vilka krav som bör ställas på STDMA-algoritmer för bruk i militära nätverk. Då inga föreslagna lösningar uppfyller de krav som ställs, beskriver vi en algoritm framtagen på FOI som är anpassad för militära ad hoc-nätverk. Algoritmen som presenteras är helt distribuerad, trafik-adaptiv och genererar scheman med spatiell återanvändning i mobila nät. Denna STDMA-algoritm är av nytta vid simuleringsstudier och prestandautvärderingar av kommersiella lösningar.

Nyckelord: ad hoc-nät, distribuerad, STDMA, schemaläggning

Contents

1 Introduction

An *ad hoc* network is a wireless network consisting of nodes that are able to communicate without the need of a fixed infrastructure, as opposed to a cellular system for instance. The nodes in an ad hoc network act both as transmitters and receivers as well as routers, relaying its neighbours packets and thereby providing multi-hop functionality. As there is no fixed infrastructure, the control of the network in terms of routing and medium access control (MAC) needs to be performed by the nodes themselves. The nodes of an ad hoc network are typically mobile making these control algorithms quite complex. The routing protocol needs to quickly adapt to varying channel conditions due to the mobility of the nodes and access to the shared channel needs to be controlled efficiently, all without too much overhead being sent. These demands make protocols for ad hoc networks difficult to design and adapting these protocols according to the type of traffic that will be sent is required in order to meet requirements for Quality of Service (QoS).

Medium access control in ad hoc networks can be performed in two different manners, contention- or schedule-based. In a contention-based scheme, nodes will try to send packets as they arrive at the node and all nodes that want to send packets will have to contend for the shared channel and use carrier sensing and collision avoidance algorithms to prevent colliding transmissions. With these types of MAC protocols it becomes hard to give guarantees for the QoS, as the delay grows quickly under heavy traffic load, due to a large number of collisions. A way of providing better QoS is to use a schedule-based MAC protocol such as *Time Division Multiple Access* (TDMA), where time is divided in slots and each node is assigned certain slots in which they are allowed to transmit, in order to avoid collisions.

In protocols based on a TDMA solution the important question is how to perform the scheduling of the time slots, i.e. assigning transmission rights to the nodes. To make best use of the resources available, the channel bandwidth, the scheduling needs to be done adaptively according to the nodes' varying traffic loads. This means that the protocol should assign more time slots to a node that has a high traffic load, in order to avoid bottlenecks in the network. To further utilize the channel as effectively as possible, nodes that are spatially separated should be able to transmit simultaneously, referred to as *Spatial Reuse* TDMA (STDMA). Lastly, in a tactical scenario a distributed scheduling algorithm is a necessity in order to avoid being vulnerable to single node failures and also to reduce scheduling delays and overhead costs.

There are no real standards in terms of how to generate STDMA schedules and the properties of the algorithms typically depend on the scenario in which they are intended to be used. Therefore, in this paper we will start by listing the properties we see as important for military ad hoc networks. Based on the listed properties, we give an overview of the state of the art algorithms presented in literature along with comments on their applicability to a military network. Motivated by the fact that no current protocols fulfill our requirements, the rest of the report describes an algorithm developed at FOI to be used in simulations and evaluations of network solutions.

1.1 Wanted Properties in a Distributed STDMA protocol

In the following we list some of the desired properties of a distributed STDMA (DSTDMA) algorithm. The first five of these are specific for distributed scheduling. The last three properties are general for all STDMA scheduling but they are so relevant for performance that they are stated here anyway.

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 an arbitrary 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.* 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.* In particular, changes in the network density will be common in military scenarios and situations where all nodes are gathered at one place (with maximum density as a result) will occur.

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

6. *Adaptivity to traffic; the algorithm should be able to adapt to the different needs of the different links.* There is considerable variation of traffic between different nodes in the network due to the relaying of traffic in multi-hop networks. An STDMA algorithm must adapt to this in order to be efficient [1]. Traffic adaptivity is only rarely included in distributed algorithms.

7. *Using an interference-based network model.* The graph-based network model is currently the most used network model for ad hoc-networks. However, this model does not reflect reality sufficiently well in many of our scenarios and is outperformed by interference-based models, see [2].

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. Of the existing algorithms, USAP [3] is interesting for military communications but not even this algorithm has all the listed properties that are desired for reliable and efficient communication on the battlefield. See [4] for further discussions.

2 Related Work

Most of the research on STDMA scheduling is focused on achieving schedules with shortest possible frame length and the authors often call these schedules optimal. The goal of the algorithms is usually to create a schedule that makes sure that each node transmits at least once. Under most interference models and scenarios, the time complexity of finding an optimal length schedule has been proven to be NP-complete [5, 6]. Most approaches therefore consist of devising a suboptimal, heuristic algorithm that produces shortest possible schedules within a constant factor of the optimal length schedule and proving that it can perform in polynomial time.

In a more realistic scenario, however, there are more important factors affecting the performance rather than just short schedules. In a military scenario these factors can for instance be traffic adaptivity, quality of service and distribution of control information. Optimizing the schedule length also means that a variable schedule length usually is needed, something that may not be practical in realistic scenarios. Many of the works discussed in this section also rely on a central scheduler, making them unsuitable for use in military applications. Other aspects that are rarely accounted for is the effect mobility has on the capacity. In a scenario with up to hundreds of nodes moving at high speeds, links tend to be very volatile which has a significant effect on the performance of both routing and medium access algorithms.

Looking at recent works we will focus on what interference models are used, how traffic adaptivity is handled and how the amount of control traffic that is needed to perform the scheduling is handled.

In [7] a centralized scheduling algorithm with variable schedule length is presented. The presented algorithm is based on the physical interference model and schedules links in a sorted order, based on the amount of interference generated by the link to be scheduled. Traffic adaptivity is performed by assigning weights to the links depending on their traffic and then assigning as many slots as the link's weight in the scheduling process, thus requiring a varying schedule length to be used. Simulations are performed under the assumption that global knowledge being available to the central scheduler but no overhead costs are accounted for. Since the algorithm is designed with Wireless Mesh Networks in mind, no results regarding the impact of mobility are presented either.

In [8] a centralized scheme focused on providing high spatial reuse is presented. The authors present a graph-based model in combination with SINR computations to better handle the interference. The proposed scheme is verified with simulations and the metric used in evaluation is the number of successfully received packets per time slot. The work does not take into consideration how the SINR information will be distributed and what the costs of the distribution will be. The system is also assumed to consist of static nodes with long packet queues, hence paying no attention to neither traffic adaptivity nor mobility.

A further extension to the regular SINR model is explored in [9, 10], where a second SINR threshold is added and packets received with SINR between the thresholds, in this so called *Gray* or *Graded* area are, received with a error probability given as a function of the SINR. Scheduling algorithms that make use of these kinds of suboptimal allocations are presented and evaluated in terms of increased throughput. No results are presented on the amount of extra overhead generated due to increased retransmissions though, hence the actual benefits of using such a model are unclear.

The problem of scheduling multicast group communications is investigated in [11]. The scheduling is formulated as an *Integer Linear Programming* (ILP) problem and the authors present a centralized scheme based on source-specific trees. The nodes within the source-specific trees are organized in levels based on the the hop-distance from the source and the construction of conflict-free graphs and scheduling is performed on a per-level basis to increase the spatial reuse. This method is still just an extension of graph-based scheduling as no SINR calculations are made and the scheduling is very reliant on a central scheduler.

In [12] a line graph approach to the scheduling problem is presented. A weight function is used to calculate weights for pairs of links based on the interference between them. These weights are assigned to the vertices of the line graph and coloring is performed under SINR constraints. The nodes are assumed to be static during simulations, hence it is uncertain if the algorithm can be applied to a network with mobile nodes. Traffic adaptivity is briefly mentioned by using a weighting function and giving more slots to links with larger weight, but it is not examined further.

Vergados et al. [13] present a distributed TDMA scheduling approach with the aim of providing QoS on an end-to-end basis. This is achieved by performing scheduling on path-basis rather than node-basis and thereby better adapting the scheduling according to the current traffic. The algorithm is evaluated in simulations and compared to two centralized schemes, showing improvements in terms of end-to-end delay for the topologies investigated. The algorithm is however designed and tested only for static networks and lacks detail in several aspects.

Among the few works which presents thorough results regarding protocol overhead is [14]. The authors present a distributed, adaptive protocol, complete with extensive simulation results for the overhead needed in order to maintain the schedule. The results show that the overhead generated quickly increases as the mobility and communication distances of the nodes increase. The algorithm presented is, however, using a graph-based interference model and no discussion is made in regard to traffic adaptivity or the underlying traffic assumptions.

Luo et al. [15] present a TDMA-based protocol specifically designed for urban warfare, aimed at providing multi-hop communications with a minimal amount of overhead costs. This is achieved by using a combination of different slot types, both slots that are evenly distributed between the nodes and repeated every frame, and slots that can be either reserved or contended for depending on the traffic situation. The system is only aimed at networks consisting of up to 20 nodes and seemingly short range communications. The simulation results show no details in respect to node mobility or traffic adaptivity.

In [16] the authors approach the overhead problem in a cross-layered manner by making the scheduling aware of the OLSR routing protocol. The idea presented is to use the MPR selector set as a measure of priority. This information is then sent in unused fields in the HELLO messages used by the OLSR algorithm, thereby not generating any additional overhead when sending information about priorities. The MPR selector set, however, gives only a very limited view of the traffic distribution and hence the approach is too simple to achieve good traffic adaptivity.

A tactical MANET system is presented in [17]. The system is fully distributed with a focus on providing support for both voice and data by the use of an adaptive frame structure containing some slots dedicated to data traffic and some slots available for voice traffic. The slot allocation is based on a request and release principle that is performed at each node using the data slots that it has acquired, hence no explicit control slots are used. The system is further developed in [18] where the scheduling is improved by using a physical interference model in the channel access scheme. The scheduling is performed in two steps, first a 2-hop interference model is used then slots that seem to allow more spatial reuse are scheduled further under SINR constraints. The algorithm is evaluated for a 20 node network in simulations and measured in terms of receiver collisions and slot usage. The work presents some interesting ideas, especially the support for real-time voice, but lacks details regarding the implementation and performance, for instance how much of the traffic sent that actually is control traffic or how well the protocol scales with growing network size.

3 Central Design Concepts

In this chapter, we elaborate on some of the central concepts on which the algorithm is based. The chapter describes the SINR-based channel model that is used to generate conflict free assignment of timeslots. Following is a description of the concept of priority which is needed to determine allocation order and to provide traffic adaptivity. Last, we discuss how link-layer information is abstracted and what kind of traffic types that are regarded.

3.1 SINR-based Scheduling

The purpose of an STDMA-scheduler is to assign transmission rights to nodes in a conflict-free manner. This is achieved if the scheduled transmissions fulfill two constraints, logical constraints and SINR constraints. The logical constraints stems from the fact that nodes are assumed to be half-duplex, i.e., nodes cannot transmit and receive simultaneously or receive multiple transmissions simultaneously. For the SINR constraints we require that at each receiving node, the SINR is above the *threshold for reliable communication*, γ_r.

$$\frac{P_i G_{i,j}}{N + \sum_{\substack{k \neq i \\ k \in T}} P_k G_{k,j}} \geq \gamma_r \tag{3.1}$$

Here 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 is the noise power at the receiver, while T is the set of transmitting nodes having a link to receiving node j. Each node that receives an allocation attempt will check the SINR constraint for all nodes that are receivers in the considered timeslot, for which there exists a link. Two nodes are considered having a link if the SNR is above the *communication threshold*, γ_c

$$\frac{P_i G_{i,j}}{N} \geq \gamma_c \tag{3.2}$$

The choice of this threshold is of course dependent on several factors, such as the actual modulation method of the signal, properties of the receiver noise, data rate and required BER. It is also possible to add additional margin to the SNR thresholds in order to only use links that tolerate some interference and also to add some stability to the set of links that are used. Different margins can be used for links that are considered in the interference calculation and links that are actually used for transmission, thus accounting for interfering effects from weak links but only transmitting on better links. The choice of thresholds will, however, affect the connectivity of the network and too strict margins might cause network segmentation. The actual values of the thresholds are considered design parameters.

Since the exact channel gains, $G_{i,j}$, are not known, the SNR for links will be estimated from the control messages sent during the control slots of the schedule, see Section 4.1.

3.2 Priority-based Resource Allocation

In a distributed scheduling algorithm there is a need to determine the order in which nodes should be allowed to acquire resources. A way to determine the allocation order serves two purposes, firstly to avoid several nodes trying to obtain the same resource at the same time, which would lead to conflicts. Secondly, in order to have traffic adaptivity, nodes that have a larger traffic load should be able to acquire more resources. We have chosen to use priorities in order to determine the allocation order and a node's priority is defined as the quotient of the traffic load and the allocated resources.

All the incoming traffic and allocated resources for a node is summed up to an aggregate priority value. In the case of link slots, each link will instead have a separate priority value. The priority value for a node, i, will be calculated from the sum of all incoming traffic flows, λ_i and all resources μ_i as

$$\lambda_i = \sum_{k=1}^{K} \lambda_{i,k}\,, \quad \mu_i = \sum_{l=1}^{L} \mu_{i,l} \tag{3.3}$$

for node i with K incoming traffic flows and L allocated resources, where both traffic and resources are given as average bitrates (in bits per second). The priority value for node i will then simply be

$$\text{Prio}_i = \frac{\lambda_i}{\mu_i}. \tag{3.4}$$

The priority values calculated assume that we have some way of estimating the traffic loads. These estimates can include both information from arriving packets, queue lengths, as well as routing information, if available. How the estimate is obtained is irrelevant to the STDMA protocol though, although bad traffic estimation will have a negative effect on how well the scheduling can be performed.

The concept of priority presented here can be further generalized to include measures such as different traffic classes in order to give, for example, voice traffic precedence over email and ftp services, regardless of the traffic load for the different services.

3.3 Link Layer Abstraction

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

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

Instead, we have to hide the volatility of the link from the MAC protocol. The link gain given from lower layer will be an expected average. Variable data rates for the link (as seen from STDMA point of view) must be average data rates, not what actually is transmitted on the link. An adaptive radio node may actually change data rate on the link from one time slot to the next.

3.4 Traffic Types

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

Multicast traffic is assumed to be treated as broadcast by the MAC layer, i.e. multicast traffic will be broadcasted to all neighbours by the MAC layer and its up to the routing process in each of these neighbours to determine whether to further broadcast it.

4 Algorithm Description

In this chapter, we present a description of the STDMA algorithm and discuss some of its most relevant properties. The first thing to notice is that the algorithm can schedule both node and link allocations dynamically, depending on traffic type and traffic rate. For example, broadcast and multicast are naturally scheduled in node slots, but also unicast flows can be sent in node slots and several low rate unicast flows can share an allocated node slot. Most aspects of the algorithm are identical whether node or link slots are used, with a few exceptions. Therefore, in this chapter we will mainly describe the algorithm from a node-oriented perspective, with minor comments on the parts that differ between node and link slots. Traffic adaptivity is included through priorities, i.e., a node with much traffic will have high priority more often than a node with less traffic, with priority defined according to Section 3.2.

The algorithm is interference-based, meaning that potential allocations must fulfill SINR constraints at all intended receivers. SINR calculations are based on link estimates which are sent in the administrative time slots. We will discuss what information the nodes exchange and how in more detail later on.

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

- Nodes that have entered the network exchange local information with their neighbours.
- The node with highest priority in its local surroundings allocates a number of time slots. Allocation is performed by the transmitting node, both when using node and link allocations.
- The local schedule of the nodes is then updated, and the node with highest priority is determined. This process is then continued until all slots are occupied or all nodes have the time slots they require. Nodes that cannot allocate themselves any more slots will block themselves from the priority calculation or try to steal slots from nodes with lower priority.
- Nodes that have resources that are no longer needed will deallocate these resources.

4.1 Frame Structure

Figure 4.1 gives a description of the general frame structure that the algorithm uses. Time is divided into frames, which is further divided into subframes. In

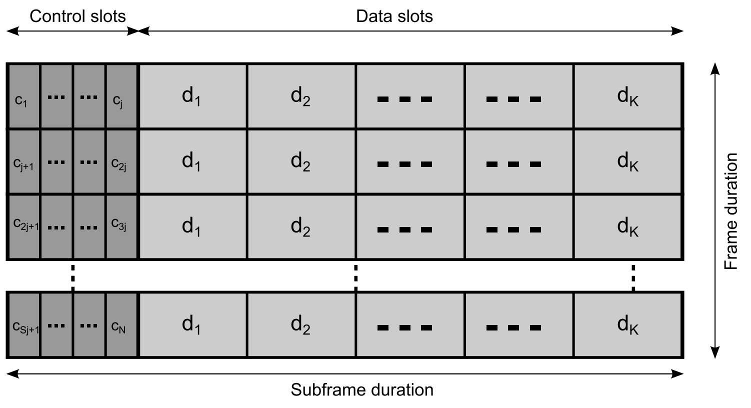


Figure 4.1: Schematic overview of the frame structure used. The frame length depends on the number of control slots per subframe, j, the number of data slots per subframe, K, and the number of nodes, N. Note that the data slots are repeated every subframe while the control slots are not. The number of subframes per frame is given by N/j as each node needs to have one control slot per frame.

each subframe there are two types of time slots. First, there are administrative control slots, these slots are divided in a round robin fashion such that the average time between a node's consecutive control slots is equal among all nodes. Different subframes typically have a different set of nodes using the control slots. The frame structure used is flexible, making it easy to adjust the amount of control traffic versus data traffic. The number of control slots and data slots are design parameters that can be varied depending on how large portion of the bandwidth is allowed to be used for overhead. However, the number of control slots per subframe, j, should be chosen as a factor of the number of nodes, to make channel access delay equal among all nodes.

The control slots are used for neighbourhood discovery, estimating SNR, and doing most of the scheduling, for further details on the control messaging see Section 5.5. The rest of the slots carry data and these slots are repeated every subframe, i.e, a node that is scheduled in a slot will be able to transmit packets every subframe.

4.2 Blocking

The concept of blocking is used in the algorithm in order to avoid certain situations causing the algorithm to get locked in a fixed state. There are three different situations that need to be handled, using three different kinds of blocks:

1. If a node tries to allocate a timeslot but fails due to the SINR being too low, the node will consider that *timeslot* blocked. This situation occurs when the allocating node and the receiving nodes have different views of the SINR for the current timeslot. This type of blocking allows the node to move on and try to allocate another timeslot, rather than trying to sync its SINR estimate with the receiving node. The reason for blocking rather than synchronizing is that SINR values tend to change rapidly, hence it is more effective to just try another timeslot.

2. When a node needs more resources but is unable to find a timeslot it can allocate, it will block *itself*. This kind of blocking is communicated to the node's neighbours. This allows neighbours of the node to ignore the blocked node when checking which node has highest priority. Without this kind of blocking, if the node with the highest priority is unable to allocate any timeslot, the algorithm would get stuck and nodes with lower priority would be stuck waiting.

3. A node which needs more resources but is not the node with highest priority in its neighbourhood will wait for the nodes with higher priority to make allocations. However, if the neighbour that has highest priority does not make an allocation in a given time, the node will block that *neighbour* from the priority list. This situation typically occurs when the node with highest priority no longer is a neighbour, due to some links being lost or a node exiting the network.

Each time a node receives updates about the network (new links, traffic loads or allocations) it checks to see whether there are any blocks which can be removed. Some special consideration is made when removing a block of type 2 (node blocking itself). When the network changes such that a node that has previously blocked itself might be able to remove the block and allocate resources, it will consider if there are other nodes in the neighbourhood who are also blocked. Only if the node in question have the highest priority, counting other blocked nodes, will it remove the block. Otherwise, the node will wait for a time, dependent on the number of nodes with higher priority, before removing its block.

4.3 Allocation of Resources

A node having less resource than its traffic load will try to allocate more resources. In this section we will describe the flow of the allocation procedure, more precisely the allocation of a node slot. The only difference when allocat-

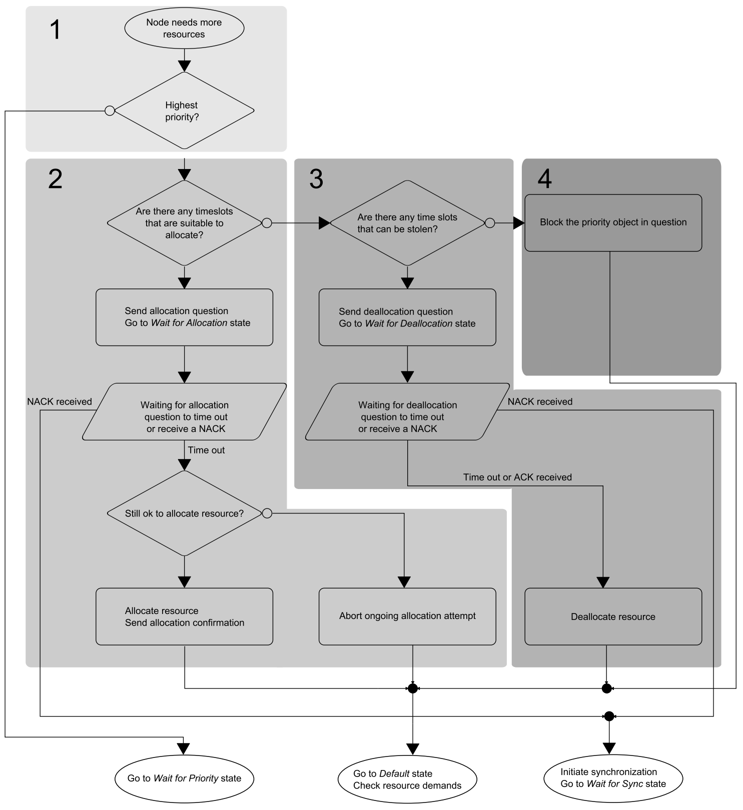

1
Node needs more resources
Highest priority?
2
Are there any timeslots that are suitable to allocate?
Send allocation question
Go to *Wait for Allocation* state
NACK received
Waiting for allocation question to time out or receive a NACK
Time out
Still ok to allocate resource?
Allocate resource
Send allocation confirmation
Abort ongoing allocation attempt
3
Are there any time slots that can be stolen?
Send deallocation question
Go to *Wait for Deallocation* state
Waiting for deallocation question to time out or receive a NACK
NACK received
Time out or ACK received
4
Block the priority object in question
Deallocate resource
Go to *Wait for Priority* state
Go to *Default* state
Check resource demands
Initiate synchronization
Go to *Wait for Sync* state

Figure 4.2: Allocation procedure. For detailed explanations of the different states see Section 5.2

ing a link slot is the priority value used, as described in Section 3.2, and the set of available time slots to allocate; the general procedure, however, is the same. Following is a description of the procedure and a flow chart can be seen in Figure 4.2. When the procedure is finished, network information and priority values are updated and the procedure is repeated as long as there are nodes that need more resources.

1. The node checks the priorities of all the nodes in the local neighbourhood to determine whether it has the highest priority in the neighbourhood. If this is not the case, the node changes state to *Wait for Priority* and awaits changes in the network that affect the priorities in the node's local neighbourhood.

2. When the node has determined that it has the highest priority in the local neighbourhood, it attempts to find a suitable time slot to allocate, broadcasts an allocation for the time slot in question and waits a given time for other nodes to reply to the allocation question. The time the node has to wait depends on the length of the schedule used and length of the time slots. This guarantees that all other nodes have time to both receive the question and respond to it before the allocation is finalized. If no NACK messages are received during the waiting time the node will assume the allocation succeeded and broadcast a message confirming the allocation, in order to give nodes that have made a temporary allocation a confirmation. If the node receives a NACK it will not send any message confirming the allocation and other nodes that had made a temporary allocation will remove it after a set time, since the lack of a confirmation message indicates that the allocation failed.

3. If there are no suitable time slots to allocate the node will check if it can steal a time slot from another node, due to having higher priority than the other node. When such a time slot is found the node sends a question to the chosen node asking it to deallocate its resource, changes its state to *Wait for Deallocation* and waits for the node in question to respond.

4. In the case that there are no resources that the node can allocate at all, the node will block itself, notify its neighbours and change its state to *Default*.

4.4 Deallocation of Resources

Allocated resources can be deallocated in two cases. The first case is that a node that realizes it has too much resources in relation to its traffic load can, contrary to [4], deallocate resources that are no longer needed, rather than just wait for another node to request a deallocation of the resource due to higher priority. The primary reason for this is to make allocation of resources faster if the network is not fully loaded.

The second case occurs when a node cannot find a suitable time slot to allocate and decides to steal a time slot from another node with lower priority, as previously mentioned in the allocation procedure, Section 4.3. When a node attempts to deallocate another node's resources, it is required that the node performing the theft has higher priority than the other node after the theft as well, to prevent oscillating behaviour incurring large amounts of overhead.

5 Algorithm Implementation

In this section, we describe how the concepts presented in Chapter 4 are implemented. The central themes discussed in this chapter are the event driven state machine and the mechanics used to spread algorithm information (such as traffic loads, allocations and link estimates) by ways of control messages and synchronization procedures.

It should be noted that the algorithm can be configured to use either explicit acknowledgement through ACK-messages or implicit acknowledgement by NACK-messages. The choice of acknowledgement method will affect the amount of overhead transmitted, where a NACK-based solution should slightly reduce the overhead sent. During the rest of the chapter, it is assumed that the algorithm is used with NACK messages.

5.1 State Machine with Memory

In order for the algorithm to function there needs to be way for nodes to acquire, update and synchronize information about links, traffic and allocations in its local neighbourhood. This is solved with the help of a state machine. A true state machine is memoryless, though to simplify implementation we have used "a state machine with memory". The memory is in the form of a database that contains:

- Link information - path gain for different node pairs (does not have to be a link that can be used for communication i.e might be an interference link)
- Traffic information - information regarding the traffic load at different nodes or links
- Allocations - which links/nodes use a certain time slot
- Sequence numbers

To keep track of which information is the newest, all information is tagged with a sequence number. A higher sequence number means a newer piece of information. If a node receives updated information, i.e. information with a higher sequence number, it replaces the information in its database with the new data. If a node receives data with the same sequence number as in its database, it does nothing. However, if a node receives old data, it will transmit its newer version of the information to that sender. The usage of sequence numbers will ensure that old information is not propagated through the network from a node with outdated information.

5.2 States

The state a node is in depends on the amount of traffic and resources it currently has and the algorithm messages it receives from other nodes in its local neighbourhood. Following is a brief description of the different states and how the node acts depending on its state. The state transitions possible and the events causing them are visualized in Figure 5.1.

1. *Default:* A node is in the default state when it does not have any needs for more resources. This is also the state a node is in when it first enters a network.

2. *Wait for priority:* When a node is in the Wait for Priority state it does not have enough resources to meet its traffic demands, however, there are other nodes in the neighbourhood which have higher priority. In this state the node waits for changes in the network that gives it the highest priority in its local neighbourhood. If enough time passes without the node getting highest priority it will temporarily block itself.

3. *Wait for allocation:* A node that is in this state has sent an allocation question for a resource and is awaiting response from its neighbours. Since a NACK-based solution is used for allocations, the node will proceed with the allocation once enough time has passed since the question was sent and no NACK message has been received. In case of a NACK message, a synchronization procedure will begin as the erroneous allocation attempt was due to an outdated view of the protocol.

4. *Wait for deallocation:* A node will be in this state when it has a need for more resources but is unable to find free time slots to allocate. The node will then check if its priority is higher than some already allocated priority object of some other node. The node will then try to steal that time slot from a node with a lower prioritized object by asking it to deallocate its resource. If the other node acknowledges the deallocation or if enough time passes without an answer the node will proceed with deallocating the time slot. If on the other hand a NACK is received the node needs to sync its traffic estimations and allocations as there is a discrepancy between the nodes' view of the current protocol.

5. *Wait for synchronization:* A node will be in this state when it has been notified that it needs to sync its traffic estimations and/or allocations. When the node is in the Wait for Sync state it will not attempt make any new allocations until the sync process is considered to be finished, in order to prevent the node from making multiple failed allocation attempts.

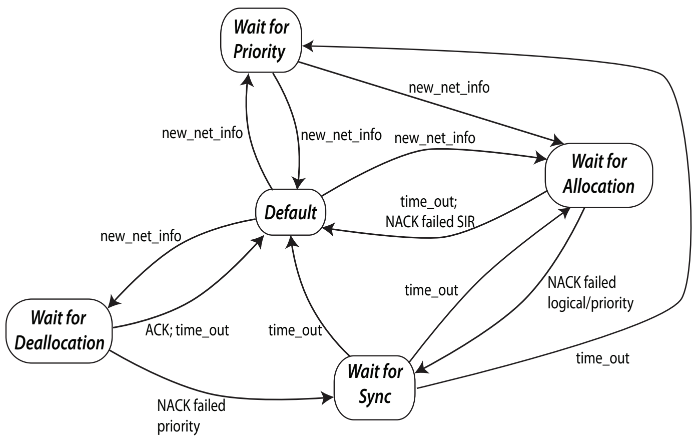


Figure 5.1: Overview of the state transitions possible and the events causing the transitions.

5.3 Events

The algorithm is driven forward by events that are triggered by receiving certain algorithm messages or as a result of a timer running out. Depending on the current state the node is in, different measures are taken. Following is description of the events that can occur and the handling of the events.

New Net Info Each time a node receives updates about the network in terms of allocations, traffic loads or links this event is triggered. The node checks if there are any resources that should be deallocated due to the SIR being too low or decreased traffic. For such resources, in which the node is either the transmitter or receiver, the node will deallocate the resource. The node will also consider if the changes in the network causes an increased need for resources and might in that case attempt to start an allocation procedure or go the *Wait for Priority* state.

- If the node is in any of the states *Default*, *Wait for Priority* or *Wait for Sync* and there are other nodes' resources that need to be deallocated, the node sends a deallocation question to the affected node and changes state to *Wait for Deallocation.* When determining which resources should be deallocated, priority is taken into consideration to keep the deallocation procedure fair among the nodes.
- If the node is in *Wait for Allocation* or *Wait for Deallocation* it will not try to deallocate other nodes resources and just remain in its' state.

Allocation Question A message is received with a question for an allocation.

Firstly the node will check to see that the question was sent from a node to which there actually exists a link, otherwise it is disregarded.

- The default behavior is then to proceed with checking whether it is OK to perform the allocation. If the allocation is considered OK, the node will make a temporary allocation of the resource and wait for the asking node to confirm the allocation. If there is a logical error, i.e. the node is either a transmitter or receiver in the concerned time slot and the asking node is a one hop neighbor, the asking node is sent a NACK message containing the affected resource and then the node proceeds to sync its' allocations with the asking node. If the allocation fails due to the SIR being too low a NACK message is sent to the asking node notifying it of the SIR error.
- When the node is in the *Wait for allocation* state it will firstly check if the allocation question concerns the same resource that node is currently asking for. If that is the case, a priority check is performed and if the node is still the highest priority a NACK message is sent to the asking node after which the node syncs its' traffic estimations and allocations to the asking node. If instead the other asking node had highest priority, the node will cancel its' ongoing allocation question and change state to *Default* and perform the default allocation question procedure.

Allocation ACK This message will only be treated if the node is in the *Wait for Allocation* state. The node will check to see that the acknowledgement is for the resource it was asking for and if so it will perform the allocation and notify other nodes about the allocation. Thereafter the node will change to the *Default* state and perform the *New Traffic Info* procedure.

Allocation NACK, Failed Priority This message will only be treated if the node is in the *Wait for Allocation* state. If the NACK is for a resource the node was asking for, traffic estimations and allocations will be synced and the node will change state to Wait for Sync.

Allocation NACK, Failed Logical Same as above.

Allocation NACK, Failed SIR This message will only be treated if the node is in the *Wait for Allocation* state. If the NACK is for a resource the node was asking for, the node will block the time slot and then change state to *Default* and perform the New net info procedure.

Deallocation ACK This message will only be treated if the node is in the *Wait for Deallocation state*. If the ACK is for an object that is in the list that the node keeps of resources it wishes to deallocate, the node proceeds to deallocate the object and then calls the *New Net Info* procedure if the list of wanted deallocations is empty. Otherwise the node remains in the *Wait for Deallocation* state.

Deallocation NACK, Failed Priority This message will only be treated if the node is in the *Wait for Deallocation* state. If the NACK is for an object that is in the list of wanted deallocations, the node blocks the time slot and proceeds to sync traffic estimations and allocations and change state to *Wait for Sync*.

Time Out Time out events are triggered when a timer ends. Timers and time out events are used in parts of the algorithm where a node needs to wait for other nodes' responses before completing an action or switching state. For instance, if a node has sent an allocation question for a resource, the node needs to wait a given time in order for other nodes to acknowledge the allocation, before using it to send data. Another instance where a node times out is when it is notified that there seems to be errors in the node's view of the protocol and then is inactive for a given time while the synchronization is performed to avoid that multiple erroneous allocation attempts are performed.

- In the *Default* state a time out will never be performed.
- If the node is in the *Wait or Priority* state and times out it means that the traffic and resource situation has not changed. The node then temporarily blocks the priority object and changes state to *Default* and calls the *New traffic info* procedure.
- In the *Wait for allocation* state a check is made to see whether the allocation that it asked for still is OK. If so, the allocation is finalized and the node will announce the allocation to its neighbours. Then the node changes to the *Default* state and performs the *New net info* procedure.
- In the *Wait for Deallocation* state the deallocations that the node has not received NACK for are deallocated and then the *New net info* procedure is called.
- In the *Wait for sync* state the node will consider the synchronization done and proceed to check whether to perform any unblocking or allocation.

5.4 Synchronization

If an allocation or deallocation fails it is usually because the nodes involved have disjoint views of the network, in terms of traffic, allocations or link estimates. The STDMA protocol must thus be able to synchronize node databases. The synchronization errors taken into consideration so far are those detected in the receiving node (Rx) when it gets a query regarding allocation or deallocation. However, the error itself is not necessarily in Rx, it can just as well be in Tx (i.e. the node making the query). When allocating, there are a number of errors that might occur:

- Wrong priority - occurs if the receiver believes that it has higher priority than the transmitter. The nodes solve this by performing a double handshake of the concerned time slots.

- Logical error - disagreement regarding if a link or node is allocated or not. Information concerning the allocation in question is transferred to the querying node. The information is used to let the querying node determine if it has missed an allocation or if the other (receiving) node has missed a deallocation. The error is then corrected, either by adding the allocation at the querying/transmitting node or by resending the deallocation message to the other node. Logical errors typically occur when a link is broken causing several nodes to make new allocations

- SINR error - The nodes disagree on the link quality. Since the quality frequently changes, we choose to sit tight and just block the time slot at the transmitting node until the next SINR update arrives.

During the comparison of the involved objects the sequence numbers of the objects will be used by the nodes to determine which information is most up to date in order to resolve the conflict.

5.5 Control Traffic

The control messages used by the algorithm can contain two types of information; algorithm messages (allocation questions, ACK/NACK and allocation confirmations) and sync information (traffic estimates, allocations and link estimates). When a control packet is packed, a packing order is used in order to prioritize the type of control information that is most crucial for the algorithm to work. The node will start to pack as much as of the highest prioritized information it can fit in the control packet. If there is room left in the packet it continues this process with information of decreasing priority until the control packet is full.

The order is determined in part by a prioritization among different kinds of messages, depending on how essential they are for the algorithm to work, and in part by whether information is new or not. As an example, new allocations and traffic updates are prioritized as well as ACK/NACK messages, while old allocations traffic updates are only transmitted if there is room, to ensure that nodes that missed the initial updates may receive it at a later time. Link updates are the lowest prioritized information and is only sent if there is room left in the message after all other messages have been packed.

There is a drawback of handling the control traffic the way mentioned here. In order for the algorithm to work properly, both algorithm messages as well as allocations, traffic updates and link updates need to be distributed. However, if there are a lot of allocations being made, there might never be enough room to send link updates (the lowest prioritized kind of information) and the algorithm will not work properly. The most obvious solution is to increase the amount of control slots used in the frame structure, see section 4.1. Another possible solution to this problem is to allow control messages, such as link updates, to be sent in the data slots that a node has acquired. These control messages could then be assigned a higher priority, as detailed in Section 3.2.

6 Conclusions

Designing communications solutions for military networks is a challenging problem, in many aspects. Mobility and the nature of the applications puts tough demands on possible solutions in terms of robustness, QoS demands and decentralized solutions.

In this paper, we have given an overview of the STDMA scheduling algorithms currently available in literature. Since ad hoc networks are used in quite varying scenarios, these algorithms can differ a lot depending on the requirements given by the scenario in which they are intended to be used. In military networks, an important factor is having decentralized solutions in order to avoid having single node failures affect the general performance of the network. A lot of the proposed solutions in literature, however, tend to be centralized solutions.

Another important aspect to consider when designing STDMA scheduling algorithms, is the network models used, more specifically the models that govern when two nodes can transmit simultaneously. Much of the work presented uses a simple graph model, in which interference is barely accounted for. Using a simple graph model yields optimistic schedules that will, however, be very susceptible to interference, especially in mobile networks. Scheduling algorithms based on such channel models tend to generate schedules with a high degree of spatial reuse in theory, however, in practice the performance will be a lot lower due to low robustness.

Given the reasoning above, we have not found any STDMA scheduling protocol, proposed in literature, that fulfill the requirements stated in Section 1.1. Hence, we have in this report presented the implementation of a distributed STDMA scheduling algorithm, based on a realistic channel model that effectively meets the requirements needed. The algorithm presented is fully decentralized, allows for spatial reuse and effectively adapts to traffic variations. The control traffic prioritization allows for the algorithm to adapt the level of overhead according to the mobility of the network, only sending the most important control messages in high mobility.

This algorithm is fully implemented in our in-house simulator and is of great use in network simulations and evaluations, as a tool against which to compare commercially available solutions and future solutions proposed in literature.

References

[1] J. Gronkvist. Traffic controlled spatial reuse TDMA in multi-hop radio networks. In *Personal, Indoor and Mobile Radio Communications, 1998. The Ninth IEEE International Symposium on*, pages 1203 –1207 vol.3, sep 1998.

[2] Jimmi Grönkvist and Anders Hansson. Comparison between graph-based and interference-based STDMA scheduling. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, MobiHoc '01, pages 255–258, New York, NY, USA, 2001. ACM.

[3] C.D. Young. USAP: a unifying dynamic distributed multichannel TDMA slot assignment protocol. 1, oct 1996.

[4] Jimmi Grönkvist. *Interference-based scheduling in spatial reuse TDMA*. PhD thesis, KTH, Communication Systems, CoS, 2005. QC 20101015.

[5] Olga Goussevskaia, Yvonne Anne Oswald, and Roger Wattenhofer. Complexity in geometric SINR. In *In MobiHoc*, 2007.

[6] Gaurav Sharma, Ravi R. Mazumdar, and Ness B. Shroff. On the complexity of scheduling in wireless networks. In *Proceedings of the 12th annual international conference on Mobile computing and networking*, MobiCom '06, pages 227–238, New York, NY, USA, 2006. ACM.

[7] Gurashish Brar. Computationally efficient scheduling with the physical interference model for throughput improvement in wireless mesh networks. In *Wireless Mesh Networks, in Proc. ACM MobiCom*, pages 2–13. ACM Press, 2006.

[8] A.D. Gore, A. Karandikar, and S. Jagabathula. On high spatial reuse link scheduling in STDMA wireless ad hoc networks. In *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, pages 736 –741, nov. 2007.

[9] Paolo Santi, Ritesh Maheshwari, Giovanni Resta, Samir Das, and Douglas M. Blough. Wireless link scheduling under a graded SINR interference model. In *Proceedings of the 2nd ACM international workshop on Foundations of wireless ad hoc and sensor networking and computing*, FOWANC '09, pages 3–12, New York, NY, USA, 2009. ACM.

[10] JinZhao Su, Jing Cao, and Wei Wu. Gray physical interference model based link scheduling algorithms. *SCIENCE CHINA Information Sciences*, 55:1337–1350, 2012. 10.1007/s11432-012-4589-4.

[11] Jung-Shian Li, Kun-Hsuan Liu, and Chien-Hung Wu. Efficient group multicast node scheduling schemes in multi-hop wireless networks. *Computer Communications*, 35(10):1247 – 1258, 2012.

[12] N. Praneeth Kumar, Ashutosh Deepak Gore, and Abhay Karandikar. Link scheduling in STDMA wireless networks: A line graph approach. *CoRR*, abs/0712.1662, 2007.

[13] D.D. Vergados, D.J. Vergados, C. Douligeris, and S.L. Tombros. QoS-aware TDMA for end-to-end traffic scheduling in ad hoc networks. *Wireless Communications, IEEE*, 13(5):68 –74, october 2006.

[14] Praveen K. Appani, Joseph L. Hammond, Daniel L. Noneaker, and Harlan B. Russell. An adaptive transmission-scheduling protocol for mobile ad hoc networks. *Ad Hoc Networks*, 5(2):254 – 271, 2007.

[15] Ming Luo, J. Hsu, Shaomin Mo, and R. Ghanadan. Distributed medium access control for multiple hop tactical networks. In *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pages 1 –7, nov. 2008.

[16] Miray Kas, Ibrahim Korpeoglu, and Ezhan Karasan. OLSR-aware channel access scheduling in wireless mesh networks. *Journal of Parallel and Distributed Computing*, 71(9):1225 – 1235, 2011. Special Issue on Advancement of Research in Wireless Access and Mobile Systems.

[17] I. Labbe, B. Gagnon, and J.-F. Roy. An adaptive VHF/UHF system for the next generation tactical MANETs. In *Military Communications Conference, 2009. MILCOM 2009. IEEE*, pages 1 –7, oct. 2009.

[18] Isabelle Labbé, Jean-François Roy, Francis St-Onge, and Benoit Gagnon. Spatial reuse and interference-aware slot scheduling in a distributed TDMA MANET system. In *Proceedings of ICWMC 2012, The Eighth International Conference on Wireless and Mobile Communications*, pages 294–303. ThinkMind, 2012.

FOI
Swedish Defence Research Agency
SE-164 90 Stockholm

Phone: +46 8 555 030 00
Fax: +46 8 555 031 00

www.foi.se