Abstract -

Spatial reuse TDMA is a collision-free access scheme for ad hoc networks. The idea is to let spatially separated radio terminals reuse the same time slot when the resulting interferences are not too severe. In this paper we first describe the properties a distributed STDMA algorithm must have in order to be efficient. No existing algorithm fulfills all of these properties. Second we focus on how to efficiently use distributed information and describe an algorithm that can handle different amount of information.

Furthermore, we evaluate this algorithm for different information and show that it can give the same capacity as a centralized reference algorithm.

Keywords - STDMA, MAC, Ad hoc networks

I. INTRODUCTION

There are many situations where a fixed communication infra-structure cannot be relied upon for wireless communication, and where self-configurable networks must be deployed quickly, e.g., emergency relief or military networks.

Common features of these networks are that they are not pre-planned, and area coverage is achieved by letting the radio units relay the messages, i.e. multihop networks. These kind of networks are often referred to as ad hoc networks. One of the most challenging problems in ad hoc networks is to guarantee Quality of Service (QoS).

An interesting medium access control protocol that has great potential for QoS is spatial reuse TDMA (STDMA) [1], which is an extension of TDMA. In STDMA the capacity is increased by spatial reuse of the time slots. An STDMA schedule describes the transmission rights for each time slot. Different algorithms for generating STDMA schedules have been proposed, see e.g. [2], [3], [4].

However, a problem in ad hoc networks is that the nodes will be moving, and a schedule that is conflict-free at one moment will probably not be that later.

Therefore the STDMA schedule must be updated whenever something changes in the network. This can be done in a centralized manner, i.e. all information is collected into a central node which calculates a new schedule [4], [5]. Unfortunately, for fast moving or large networks this is usually not possible — by the time the new schedule has been propagated to all nodes it is already obsolete due to node movements.

Another way to create STDMA schedules is to do it in a distributed manner [2], [3], [6], i.e. when something changes in the network, only the nodes in the local neighborhood of the change act upon it and update their schedules without the need to collect information into a central unit.

In this paper we describe the properties a distributed STDMA algorithm must have in order to be efficient. No existing STDMA algorithm can fulfill all these properties, although the most complete USAP [7] fulfills several of them. STDMA has great potential but so far there is still work lacking to make it sufficiently efficient to be competitive.

Therefore in this paper we focus on how to use distributed information and describe an algorithm that can efficiently handle different amount of information.

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.

We evaluate our algorithm and show that under ideal conditions it can give as high throughput as a centralized version with all information. Furthermore, we show how the throughput decreases with information reduction.

II. NETWORK MODEL

Here we describe the interference-based model of a radio network. The network is represented by a set of nodes \( V \) and the link gain \( G(i,j) \) between any two distinct nodes \((v_i, v_j)\), \(i \neq j\).

For simplicity we assume isotropic antennas and that all nodes use equal transmission power.

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

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

where \(P\) denotes the power of the transmitting node \(v_i\), and \(N_r\) is the noise power in the receiver.

We say that the pair \((v_i, v_j)\) forms a link, \((i,j)\), if the SNR \(\Gamma_{ij}\) is not less than a communication threshold, \(\gamma_C\). That is, the set of links in the network, \(L\), is defined:

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

For a set of links, \(L \subseteq L\), we define the transmitting nodes: \(V_T(L) = \{v_i : (i,j) \in L\}\). For any link, \((i,j) \in L\),
we define the interference at node $v_j$ as follows

$$I_L(i,j) = \sum_{v_k \in V_L(i) \setminus v_i} P_k G(k,j). \quad (3)$$

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

$$\Pi_L(i,j) = \frac{P_i G(i,j)}{N_r + I_L(i,j)} \quad (4)$$

We assume that any two radio units can communicate a packet without error if the SIR is not less than a reliable communication threshold, $\gamma_R$. An STDMA schedule is called conflict free if the SIR is not less than the threshold $\gamma_R$ for all receiving nodes in all time slots.

However, due to mobility and limited information conflict-free schedules are very difficult to create and uphold. In order to make comparisons for distributed scheduling we assume that links that have a lower SIR than $\gamma_R$ in a time slot can decrease its data rate as compared to the nominal data rate $R_N$ used by links with SIR above $\gamma_R$, i.e.

$$\text{SIR} = \frac{\text{Used Rate}}{R_N}, \quad (5)$$

Furthermore, we assume that a node cannot transmit more than one packet at a time and that a node cannot receive and transmit simultaneously.

STDMA algorithms generally are of two types; transmission rights have been scheduled either to nodes or links. In the first case, only the sender is determined in advance, in the second, both sender and receiver.

In this paper we concentrate on link assignment, often referred to as link activation. This is mainly done because link activation can handle advanced nodes with abilities like power control and adaptive antennas more intuitively (and efficiently). 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.

Traffic in multihop networks

In our traffic model we assume point to point traffic, i.e., a packet entering the network has only one destination. For simplicity we will assume a uniform traffic model, i.e. all nodes are equal likely as source and destination.

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

This relaying of traffic causes a considerable variation in the traffic of the links. This will cause “bottle neck” effects at busy nodes with long packet delays as a result. To achieve optimum throughput, we have to use an traffic-adaptive schedule to compensate for this problem, i.e. some links or nodes can be assigned more than one time slot.

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

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

III. Desired properties

In the following we list some of the desired properties of a distributed 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 it is the basic meaning of the term distributed.

2) Only local information is exchanged and needed, i.e. the information propagation must be limited. The other corner stone of the term distributed. We do not make any specific definition on 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 decrease with distance from the change.) This addition to the previous two requirements prevents "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).

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 [5]. In a traffic-adaptive schedule, links or nodes with heavy traffic can use several slots according to the traffic load.

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 [8]. 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 network 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 all information that may be possible to transmit can be the existence of neighbors, and the algorithm should perform well under these circumstances as well.

None of the existing STDMA algorithms fulfill all these desired properties. Although several distributed STDMA algorithms exists, most of these have been designed with the purpose to give an acceptable solution, rather than a solution that utilizes the channel as efficiently as possible under different situations. A 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 the schedules we can create; thereby increasing the total capacity of the network. But increasing information also increases the overhead, however. Therefore, this means that the amount of information the algorithm passes between the nodes should vary depending on the situation. This is difficult to do if we use graph-based scheduling.

The use of interference-based scheduling can give us the means to vary the amount of information. This paper describes the first distributed STDMA algorithms that have been developed.

In the next section, we will describe a basic interference-based algorithm that creates an efficient schedule with given local information about the network. The algorithm does not care how it receives the information, it only acts upon the information it has received.

This basic algorithm is then used to investigate the efficiency of schedules created with different amount of information, ranging from complete information to very little information.

In further work, we will study the cost in overhead to convey the appropriate network information. This can be used to develop a complete algorithm that includes the control information and which gives as efficient schedules as possible in every situation.

IV. AN INTERFERENCE-BASED ALGORITHM

This section gives a description of the algorithm. In short it can be described with the following steps.

- Nodes that have entered the network exchange local information with its neighbors.
- The node/link with highest priority 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 slots are occupied.

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

In the following we 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 functionality of the scheduling process.

The STDMA algorithm is run in parallel for each link, i.e. each link can be considered a separate process which is run at the receiving node of the link, i.e. 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 it will subsequently assign itself a time slot. For simplicity we assume a random choice if more than one time slot is possible. A link process is in this mode when there exists unused slots or when the link’s share of the time slots in its local neighborhood is too low. In the latter case, it can steal time slots from other links. We later describe under which conditions this may be permitted. Information about which time slot that is chosen and the new priority of the link will be transmitted to its local neighborhood. After this the link process can stay in active 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 on 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 the node simply waits for a change of the network, either in topology or in traffic levels.

In this paper we assume that the receiver does the assignment. However, the sender could do this instead with minimum changes.

Eventually, parts of the schedule will not be valid due to mobility. A receiver on a link which detects conflicts in a time slot will deassign the time slot. This might then result in thefts of another slot (or even the same).

**Link Priority**

Link priority decides in which order the links may attempt to assign themselves a time slot. This figure depends on the number of time slots the link is assigned, $h_{ij}$, and the traffic of the link, $\Lambda_{ij}$. Since both these values are changing, the link priority is constantly changing.

The priority value of a link $(i,j)$ will be $h_{ij} \Lambda_{ij}$, when both $h_{ij}$ and $\Lambda_{ij}$ are not zero. However, the lowest value has the highest priority. The links with the highest priority (lowest value) will then be the ones which limits the maximum throughput in their local neighborhood. 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 some of the links receive more than one slot. This will be the case for example when a new schedule is initiated.

Theft of time slots

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

The policy for 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 total network throughput must increase.

This means that a link only is permitted to steal a link from another if the priority value of the stealing link is lower than the other links priority value after the loss of a time slot, i.e.

\[ \frac{h_{ij}}{A_{ij}} < \frac{h_{kl} - 1}{A_{kl}}. \]

Limited information

The algorithm will be made interference-based by using interference information, i.e. we transmit and use interference information when we decide whether links can transmit simultaneously. We use the term local neighborhood of a link \((i,j)\) to mean 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. A remaining issue is then exactly what information the algorithm needs in order to do the scheduling. A node needs the following information about the other nodes and links in its local neighborhood:

- **Received Power** We have an estimate of the received power from all other transmitters \(P_{G(i,j)}\), this is also used to determine how much interference they cause, see equation 3. However, if the received power from a transmitter is below a value \(\delta I\) it is set to zero, i.e. the algorithm ignores how such nodes affect each other. If \(\delta I\) is set to zero the local neighborhood is the entire network. We define the Interference threshold \(\gamma_I\) to be \(\delta I/N_r\). The interference threshold thereby controls the size of the local neighborhood.

- **Local Schedule** We also need information of how much more interference can be handled by the assigned receivers.

- **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.

With this information we have sufficient information for a transmitter of a link to determine whether its transmission will cause conflicts for the other assigned links and a receiver of a link will be able to decide whether the interference from other links is small enough so that it can receive without conflicts.

The only further information required for a receiver to determine if it can assign a time slot is information from the transmitter about which time slots that are available for the transmitter. The receiver can then assign the time slot. Information about this is then propagated to the local neighborhood.

V. Evaluation

In this section we compare the described algorithm with an centralized version of the algorithm. We evaluate our algorithm by studying the average maximum throughput of our algorithm compared with the throughput of the centralized algorithm.

In the comparisons, 25 networks of three different connectivities (low, medium, and high) have been generated for networks of size 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.

The centralized algorithm we compare with is a centralized version of our distributed algorithm. 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. 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, i.e. lowest value of \(h_{ij}/A_{ij}\) first.
- Choose the 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.

In Figures 1 and 2 we plot the ratio between throughput of our distributed algorithm and the centralized variant for different choices of \(\gamma_I\). Figure 1 is for 20-node networks, and Figure 2 is for 40-node networks. Two things can be noted in these plots. First, if we use \(\gamma_I\) 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 fact that the information is distributed and no node have all information will not have negative consequences for the capacity of the network.

If we limit the information for both algorithms we might have some loss for the distributed, though, since an centralized algorithm can also use knowledge about the size of
the network and can thereby at least estimate how much additional interference that will affect a node from outside its local neighborhood. But this requires an addition to the present algorithm.

Second, we see a decrease in the capacity of the network when $\gamma_I$ 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 and decreases with increasing connectivity. In both cases this is probably because there are more nodes outside the local neighborhood that can cause interference that the algorithm does not consider.

VI. CONCLUSIONS

In this paper, we have listed the properties a distributed STDMA algorithm should have to be efficient. 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 algorithm. Of the existing algorithms USAP is the most interesting, but not even this algorithm has all the listed properties.

The main contribution of this paper 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.

REFERENCES