Wireless Communication in Orienteering

Sergio Angel Marti

March 4, 2005

Abstract

Orienteering is a popular competition where the participants have to pass by a number of control points in numerical order, and in the fastest time, in order to win. In the current competitions of orienteering, it is very difficult for the public to follow the race due to the fact that these competitions are carried out in forests.

In this project, we apply ad-hoc network to this scenario so that the control points send the information of each participant to a main server, in a multihop way, in order to show the information to the public. The goals are to save power and decrease the delay.

This project studies the main MAC protocols for ad-hoc, and compares both of the main branches: contention-based and TDMA, by measuring the sleeping time and the delay in different networks, always with the orienteering scenario in mind.

Even though TDMA reaches better efficiency, because of being simpler, a final solution based on contention-based has been implemented that works efficiently with a small number of hops.

Acknowledgements

I would like to thank all the people that have made possible this project. First of all, I would like to thank my supervisor, Christian Rohner, for his endlessly help along this period. I did not know very much about ad-hoc networking when I started. Thanks to him, I have learned a lot. He always had time for me when I needed help.

I would like to thank also Adam Dunkels, the author of the contiki operating system, for answering all the questions I have had. I could not have gone so far without his support.

I would like to thank the CoRe group. I have really enjoyed working there. Thanks to all the members for making of such group a nice place to work in.

Besides, I would like to thank all the people that has been part of my social life during this period. Thanks to all my friends, for all the chats, parties and trips, that have made of my stay in Uppsala a nice period to remember.

Finally I would like to give my deepest thanks to my girlfriend Inma. Being with her, the effects of the hardest and most stressful moments of the project were reduced to the half, while the greatest were doubled.

Contents

Chapter 1

Introduction

Day by day, wireless networks go deeper into our lifes in an unstoppable way. Nowadays, it is not strange to see people sharing files, visiting websites, playing online games, with nothing more than a laptop with a wireless card. The technology has improved so much, that now it is not necessary to make a hole in a wall to connect two computers, instead of that we just have to buy a couple of wireless cards, and we will be able to communicate two computers, regardless of their location at home.

Normally, wireless networks are used to connect an existing wired network to a group of clients, offering them complete access to the internet resources. This mode of wireless network is called infraestructure mode. However, it is also possible that a group of clients connect between them directly, and form an independent network, where nodes must help each other in routing messages in a multihop way, if two nodes are too far away. This mode of wireless network is known as ad-hoc wireless network.

There are a lot of situations where ad-hoc wireless networks can be used. For instance, some scenarios could be a couple of classmates exchanging documents after class, or a group of inhabitants of a town far from a city, who do not have internet connection, and want to chat and share files. But we can go further. The flexibility of ad-hoc networks make possible their location not only in cities and towns, but also in forests, lakes, montains, even in damaged terrains by some disaster. A rescue team that needs to keep contact can be a good example.

In order to build an ad-hoc network, nodes must have some rules about how to send the data, to whom, and in which order. The set of these rules is called the MAC (Medium Access Control) protocol, and it has to be created carefully, having into account the scenario where the network will be used,

and optimized to achieve some goal. The typical goal to achieve in most networks is the efficiency, but in ad-hoc wireless networks, as they normally have limited battery, an important goal is to reduce power consumption as well. In this project, the work will be focused on the creation of an ad-hoc network with the main goal of reducing power consumption, and the increase of the efficiency as a secondary goal, and the scenario will be the sport of orienteering.

Orienteering is a popular competition not only in the Nordic countries, but also around the world. In this sport, each participant is helped by a map and a compass to run a course, passing by a number of control points marked in numerical order. Each of these controls are drawn in the map. The goal is to arrive to the finish line as fast as possible.

Because of being usually carried out in forests, the competition is hard to be followed by the public, since the participants disappear in the beginning in the forest and turn up eventually in the finish line. In order to make it more exciting, the public should be aware of the times where each participant passes by a control point.

This can be done with the implementation of an ad-hoc network, where each control point (helped by other control points) send the exact time of each participant to a main server, and this server reproduce the data in a screen, visible by all the public. This will be the main purpose of this project. For the control points, a small sensor node, with a limited battery, will be used. The goal will be to save as much battery as possible, but having into account that a long delay is not acceptable, since the public has to follow the race in real-time.

This project is structured as follows. The next section introduces and explains the typical MAC protocols. Section three talks about the specific MAC protocols for ad-hoc. Section four will compare the main protocols, with the orienteering scenario in mind. Section five will describe the final algorithm, and will show the results and some improvements. Final conclusions are summarized in the last section.

Chapter 2

MAC protocols

In any computer network, nodes cannot just send data whenever they want. If that happened, many nodes would probably send the data at the same time and a lot of collisions would take place. This would lead to a very chaotic network where messages take a long time to arrive to their recipient or even they do not arrive. To avoid this, networks usually have some rules to decide which node or nodes are allowed to send. This rules are defined by the Medium Access Control layer protocol, known most commonly as the MAC protocol.

The aim of a MAC protocol is to allocate the media between the nodes in the best way as possible, so that the main goals are achieved. Sometimes, it would be good to allocate the media between the nodes in a fair way, that means that there is no discrimination between them and there are not some nodes using the media more than others. In other situations maybe it is favourable for our goals to have some nodes using the media more than others. Besides, the MAC protocol has to avoid collisions as far as possible.

MAC protocols can be divided in two groups: Static and Dynamic. Next, the main protocols in each group are explained.

2.1 Static MAC Protocols

To solve the channel allocation problem, it is possible to assign each node a portion of the media. This portion can be equal to all the nodes, or proportional to the traffic of each node. The protocols that divide the channel between the nodes in this way are called static, since this division is mantained during all the life of the network. The main static MAC protocols are the Frecuency Division Multiple Access (FDMA), Time Division Multiple

Access (TDMA), and Code Division Multiple Access (CDMA), where the bandwith is divided based on frequency, time, or codes, respectively.

2.1.1 FDMA

When trying to allow multiple stations to access the same medium simultaneously, a first and logical approach would be to divide the channel into several subchannels, as many as the number of users in the net, so that each one has its own channel to transmit, private and completely separated from the rest.

These are the basics of the Frecuency Division Multiple Access protocol. In the FDMA, and as it's shown in the picture, the whole bandwith of the channel is divided into N portions, being N the number of stations willing to transmit. This assures a private way of sending data for each station, and needs that a station trasmit always by the same frecuency band, and receive from the whole bandwith.
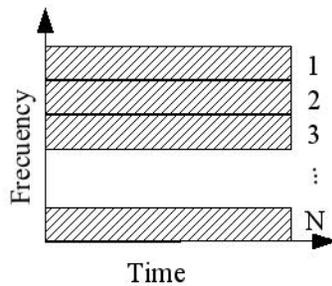
Figure 2.1: In FDMA, frecuency is divided between the N nodes

As there is no interference between users, synchronization is not needed. This makes this protocol very simple, and efficient when there is a small and constant number of users, and all of them have a similar load of traffic. However, this is not what happens in the real world. Usually the number of stations is constantly changing, and the traffic is bursty. When less than N stations are connected to the network or some of them are quiet or just sending few packets, some bandwith is lost. Besides, FDMA networks are not scalable: changes in all the stations would be required when adding a new user. That is why this protocol is not efficient in the real networks.

2.1.2 TDMA

Like the FDMA, the TDMA (Time Division Multiple Access) divides also the channel between all the users. The difference is that, while in FDMA the channel is divided by its frecuency, in TDMA it is divided by a very different way.

With TDMA the time line in the channel is divided into time slots, each slot asigned to each user. Therefore each station is able to use the whole bandwidth of the channel, but only during its corresponding time. We can have a better understanding of this protocol by looking at the picture.
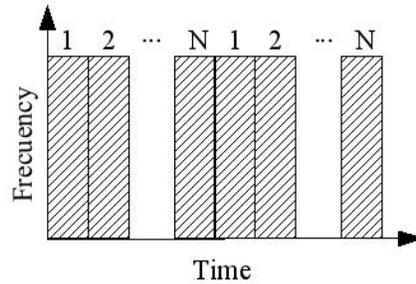

Figure 2.2: In TDMA, time is divided between the N nodes

The efficiency of this protocol is also very similar to the FDMA. It works very well when the load of traffic is held constant, and it is unefficient when for example a station is quiet some time, because its time slot is wasted. However, there is a difference that makes this protocol less simple than FDMA, that is the need of synchronization between users, to avoid them sending data in another slot. This has also an advantage, because it allows resynchronization when the number of users changes, so that there is no time slot wasted.

To improve the performance of this protocol, it is also possible to assign time slots of variable size to every station, depending on how heavy is their load of traffic.

2.1.3 CDMA

With Code Division Multiple Access (CDMA), there is a unique code for each station, called the chipping code, which is used to encode the data every time a user is going to transmit. The receiver must decode the signal to understand the data. It doesn't matter if two or more stations transmit

at the same time, the resulting signal will arrive at the receiver and, as it knows the chipping code of each station, it will decode the signal and get the original data.

As we can infer, this protocol does not have the main drawback of FDMA and TDMA, since there is no bandwidth lost when a station is quiet. The disadvantage is the long time it takes to enconde and decode the data every time a transmission takes place. Moreover, an exact synchronization between the senders is needed, so that the receiver gets the right sum of all the signals.

2.2 Dynamic MAC Protocols

The Static MAC Protocols can offer a simple and good solution when the number of nodes of the network is small and constant. However, most often networks are composed by an undefined number of nodes, with an unpredictable load of traffic. In these cases the allocation of the channel should vary. This is what the other main group of MAC Protocols do.

Dynamic MAC Protocols involves a more complex scheme where the division of the channel between the nodes does not remain the same during the whole life of the network, it adapts itself instead. Some dynamic protocols are the ALOHA and the Carrier Sense Multiple Access (CSMA).

2.2.1 ALOHA

The ALOHA protocol was developed in the 1970s at the University of Hawai, and it is one of the most popular of the multiple access protocols.

The basics of the ALOHA protocol are very simple. A station transmits whenever it has data to send. If there is a collision, it waits a random amount of time and transmits again. Usually, it is possible to know when a collision occurs by listening to the channel. In the mediums where it is not possible to listen and send at the same time, acknowledgements are used to find out if a frame was correctly received.

This is called the pure ALOHA. In this variant, if the last bit of a frame occupies the channel at the same time that the first bit of another frame, then a collision occurs. There is a variant of the ALOHA called slotted-ALOHA, where the time is divided into slots, so that the probability of having a collision is decreased.

The ALOHA protocol is simple and efficient with a small number of users, but has a poor performance when the load of traffic is very high, since more collisions take place.

2.2.2 CSMA

In the ALOHA, stations transmit when they have data to send. As they do not find out first if someone else is using the channel, collisions can often take place. Carrier Sense Multiple Access protocols (CSMA), as we can infer from its name, are based in the fact that stations sense the channel before trying to use it. The goal is to avoid most of the collisions that ALOHA systems suffer.

Whenever a station wishes to transmit, first it senses the channel, if no one is sending, then it sends the data. On the contrary if the channel is busy, then it waits until it is idle, and then it transmits again. Most of the collisions are avoided, but they can still occur, for example if two or more stations were waiting for the channel to be idle and then they transmit at the same time. If that happens, the station waits a random amount of time and starts the algorithm again.

This variant of CSMA is called 1-persistent. Persistent because when the channel is busy it keeps listening until it becomes idle. There is a variant called non-persistent, where the station waits a random amount of time whenever it finds the channel busy, instead of keep listening all the time.

This reduces the use of the channel, and more collisions are avoided, since stations don't start sending all at the same time once the channel becomes idle.

CSMA protocols reach a very high throughput and work well in LAN networks, where all the stations are able to listen to all the rest, and thus carrier sense prevents an important number of collisions. That is why they are broadly used in wired systems. For example, a variant of CSMA is used in the IEEE 802.3 standard.

Chapter 3

MAC protocols for ad-hoc

In the previous chapter, it was showned how CSMA achieves high levels of throughput in local area networks. This was because, in networks where all the stations can connect to all the rest directly, carrier sense does an important job avoiding most of the collisions, since when a station starts to send, all the rest are aware of that transmission by listening to the channel. Besides, thanks to the fact that the stations in wire networks can listen and send at the same time, a station that is sending data can listen at the same time and find out if there has been a collision, in order to stop the transmission as soon as possible.

Nevertheless, when trying to apply CSMA to ad-hoc wireless networks, due to the considerable differences between wire and wireless systems, some problems are encountered. First, in wireless, nodes do not have the ability of listen and send at the same time, as wire stations do. This leads to the problem of taking a long time to detect collisions, since nodes cannot find out if their message has collided with another one, until they have finished the transmission.

Another main feature is that in wireless, nodes are not connected to all the rest. Each node has a characteristic range instead, and only can send data to the nodes inside its range. Thus, when a node wants to send data to some other node, this data is sent along the whole range and not just to the space between both nodes[1], so all the nodes in range receive the data.

This features make CSMA a bad adviser in wireless networks. Imagine a network with three stations: A,B, and C, where B can communicate with A and C, and C can communicate with B. Let us consider that A is sending

---

[1]This would be possible with directional antennas. In this project it will be supposed that all the antennas are omnidirectional

data to B at the moment, and that C wants to comunicate with B. If using CSMA, as C is out of the range of A, it will conclude that the channel is idle and it will send the data to B, causing a collision. This is called the *hidden terminal problem.*
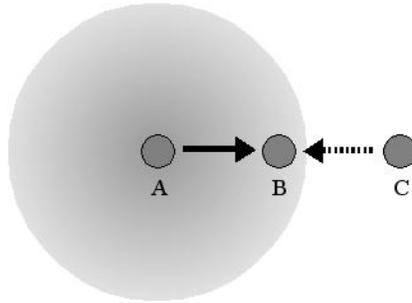
Figure 3.1: The hidden terminal problem: As C is unaware of A's transmission, decides wrongly to send to B

Let us now imagine that B is transmitting to A, and C wants to communicate with a four node D, in range of C. As C is in the range of B, it will notice that the channel is being used, so it will not send data to D, despite it could transmit without disturbing B to A transmission. This is called the *exposed terminal problem.*

Figure 3.2: The exposed terminal problem: C decides wrongly not to send to D because of B's transmission

Because of all this problems, CSMA is not optimum and other MAC protocols must be used when dealing with ad-hoc networks. There are already a lot of MAC protocols specific for ad-hoc networks, and they are

clasified in two groups: contention-based and TDMA. Some of them, the most important, will be studied in this project.

3.1 Contention-based

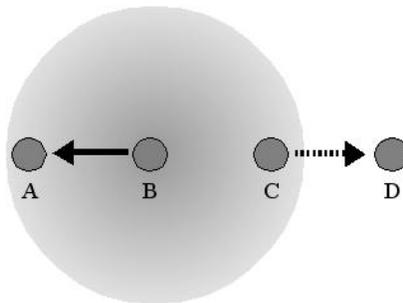Contention-based are the MAC protocols for ad-hoc where the nodes are likely to send at any moment, so they have to use some contention technique in order to decrease the number of collisions as far as possible. Next, the most important ones, MACAW [3], PAMAS [4], and S-MAC [5], will be analyzed.

3.1.1 MACAW

MACAW, one of the first protocols in offering an alternative to CSMA in wireless, uses a very different way of controlling the access to the medium. While CSMA senses the activity of the medium around the sender before sending, in MACAW nodes exchange some control packets before transmitting. The purpose of this idea is to let all the neighbours know about this transmission, specifically to let them know where is the receiver, that is, the critical node in wireless, and thus avoid a big number of collisions. MACAW is based on another protocol called MACA [2], and it is used in the standard IEEE 802.11.

Main features

The basics of MACAW is that nodes exchange some control packets before sending. These kind of control packets are called Request to Send (RTS) and Clear to Send (CTS). The first one is sent by a node before it wants to send something, and the second one is sent by another node as an answer of the RTS. As an example, imagine a network with two nodes A and B. When the node A wishes to send data to node B, first it sends the RTS packet, that is a short packet containing the length of the data to send. When the node B receives the RTS, then it answers with the CTS packet, which is also a short packet containing the length of the transmission. Once node A receives the CTS, it starts sending the data to B.

The key of this idea is how neighbours react while the transmission takes place. To avoid collisions, nodes close enough to a node that is going to receive something should remain silent during that transmission. In MACAW, any node hearing a RTS packet will remain silent time enough for the sender to receive the CTS. And any station hearing a CTS packet from a node will

defer on sending during all the transmission. For example, imagine we have also one node C, in range of A but far from B, and another node D, in range only of B, as shown in the picture. On the one hand, when C hears the RTS from A, it will understand that A intends to send something to B, and will remain silent some time enough for A to receive the CTS packet from B. On the other hand, when D hears the CTS from B, it will remain idle during all the transmission, since sending would interfere at B.
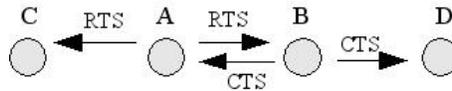

Figure 3.3: RTS and CTS exchange in MACAW

Whenever a node sends one RTS, it sets a timer. When that timer expires, it goes to the contention state, where the binary exponential backoff algorithm will be executed before retransmitting the packet. That will prevent several nodes sending at the same time after a collision.

Untill this point, these are the same basics for MACA and MACAW. Next, the improvements of MACAW will be mentioned.

Backoff

When more than one station is waiting for a transmission to end, it is likely that all of them will try to send its data at the same time after the transmission, causing a collision. The backoff algorithm is used to minimize the number of those collisions in a network. In original MACA, it works as follows: each node has a counter, called backoff value. Whenever a node wants to send, it first waits during a random amount of time slots between 0 and the number of the backoff value, and then transmits. If the transmission is succesful the backoff value is reseted to the minimum value, and if there is another collision it is increased.

This approach decreases the number of nodes sending at the same time, but it is not optimal. Let us imagine that we have 2 nodes that have an infinitive amount of packets to send to a third node. In the beginning there will be collisions, but as long as their respective backoff values increase, eventually one of them will win the medium, and will send. As a consequence the backoff value of the winner will be set to the minimum, while the value of the loser will increase still more. Once the transmission is finished they both will have still packets to send, but the winner will have more chances

to send again. This may result in an unfair distribution of the medium, since one node will send all the time and the other one will not be able to access the medium.

If we want to achieve fairness in a network, all nodes must have the same backoff counter, so that all of them have the same chancess to use the medium. MACAW deals with this problem by adding to the packet header the backoff value of the sender. When a node hears a packet then substitutes its own backoff value by the one in the header of that packet. Thus all nodes have the same value and the network is fairer. Besides, instead of reseting the value to the minimum after each succesfull transmission, which would lead to reseting the whole network and thus practically assure a collision if several nodes are waiting, MACAW decreases the value by 1 each time data is succesfully sent and multiplies it by 1.5 when there is a collision. Therefore the backoff value of the whole network adapts itself to the traffic.

Multiple Stream Model

In the above section, we achieved fairness by giving all the nodes the same probability to access the medium. However, there are some cases where there is still some unfair situation. Let's imagine this situation.
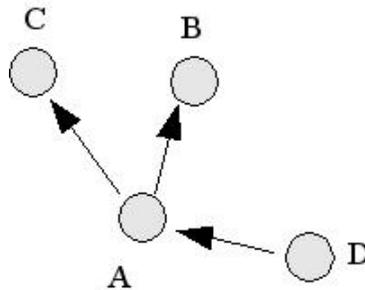
Figure 3.4: In this situation, D to A transmission will have half the bandwith of the network

In the example that the picture shows, where one node A wants to transmit some data to B and some data to C, and at the same time node D wants to send data to A, since nodes have their own backoff value, and bandwith is divided between them, transmission from D to A will have half of the bandwidth of the network, while the other half will be divided between the other 2 transmissions. It depends on the situation, but normally in this case

we would like to give each transmission the same bandwidth, treating all streams similarly instead of treating all the nodes similarly. In MACAW, each station has a different queue per transmission, and the backoff algorithm is executed for each transmission independently. Thus streams contend to the medium, and all of them have the same chances to win the medium.

Message Exchange

Usually wireless networks are very unreliable. That means that, compared to wired networks, a lot of packets are lost, and thus they have to be sent again, causing a loss of bandwidth. In original MACA, when a collision occurs, or when a packet is lost because of the noise, the problem has to be dealed at the transport layer. This produces a delay that could be reduced if the error was handled at the link-layer.

MACAW takes this into account, and turns the message scheme of MACA RTS-CTS-DATA into RTS-CTS-DATA-ACK. That means that after sending the data, the sender waits for an acknowledgement. If it does not arrive, the sender will suppose that some problem in the communication has occured and it will start again all the process by sending the RTS. If it is the acknowledgement what has been lost, then when the sender sends the RTS, the receiver will answer with the ACK, and thus the sender does not have to send all the data again.

In a network with no error rate this change would be a loss of throughput, but in unreliable networks, like wireless, the gain in throughput is very significant.

DS

In the exposed terminal scenario, that ilustrates the picture, B is sending to A, and C wants to transmit to D. In a wireless network this situation could be possible, as B and C are sending and do not interfere between them. However, in our scheme where the sender as well as the receiver have to send control packets, note that this situation will not be possible, since C could send the RTS but not receive the CTS. The way that MACAW solves this problem is making C remain silent during the whole transmission from B to A.

One trivial way of achieving this would be to make C remain quiet whenever it hears a RTS. This will prevent C from sending, but will also force C to stay idle if B sends a RTS that receives no answer. Thus C should

only defer on transmitting when it hears a RTS that leads to a success-
full transmission. To do that, MACAW adds another control packet, called
Data-Sending packet (DS). This packet is sent by the sender after receiving
the CTS and just before the DATA. And any node hearing it will be aware
of a succesfull transmission and thus will remain silent during all the time it
takes. In our example, C will hear the DS from B and will stay silent. If it
hears no DS after the RTS it will conclude that that transmission was not
succesfull and will send to D.

Note that this could also be achieved by sensing the carrier before send-
ing. The purpose of MACAW by choosing the other way was to avoid carrier
sensing hardware.

RRTS

Let's imagine the next example, where C is in range of B and D and D is in
range of C but far from B. Imagine that A is sending to B and D wants to
transmit to C. D will send and RTS to C, but C will not be able to answer
with a CTS since he is defering to the transmission from B to A. If B has
a lot of data to send to A, transmission from D to C will hardly take place,
since the moment when D sends would have to coincide with the moment
when B is quiet, what is very difficult.

MACAW adds one type of control packet called RRTS (Request for
RTS), that tries to solve this problem. In some moment C will receive a
RTS from D, and will not be able to answer with a CTS because of the
other transmission. Then C will contend, and after the transmission, he will
send a RRTS to D, and then D will inmediately answer with the RTS, and
the normal communication process will take place. Node B will defer for
two time slots, and then for the whole transmission when it hears the CTS.

Figure 3.5: In this situation, D to A transmission will have half the bandwith
of the network

17

Multicast

When trying to send the same data to several receivers at the same time, the control packets exchange explained above is no longer suitable, because several CTS will probably collide. In MACAW, if one station needs to send multicast data, it sends the RTS followed by the DATA all together. The receivers will identify a multicast RTS and will not send back a CTS. Besides, other stations will defer for the length of the whole data transmission.

Conclusion

MACAW is a protocol that follows the basic scheme of MACA, and improves it by adding some features, in order to form a more complete medium access protocol.

The fact that nodes exchange control packets between them, whenever they want to transmit something, and the fact that other nodes respect that transmission, by remaining silent during the time it takes, gives this protocol a good avoidance of collisions, what as a consequence leads to a reduction of the power consumption caused by collisions and retransmissions.

However, there are some aspects of this protocol that make it unfair and less optimal to most real networks. Imagine the situation of the picture, where B is sending to A, and it has an infinite amount of data to transmit. D also wants to send data to C, but when it sends a RTS, it happens that C is unable to hear because B is sending all the time. In this situation, communication between D and C could only take place when D send the RTS just in the right moment between a complete data transmission between B and A. This is quite difficult, and here the RRTS cannot be applied, since C cannot hear the RTS. As a consequence, transmission from B to A will obtain the whole throughput of the medium for itself, while D transmission could not access the medium.

Figure 3.6: If B is always sending data it is difficult that C receives something from D

Besides, there are also some situations in wireless where several transmissions can take place at the same time withouth interference, which are not possible in MACAW. These situations are ilustrated in the next pic-

18

tures. In the first one (left), A and D are sending to B and C, respectively, and both transmissions do not interfere between them because B only hears A transmission and C only hears D transmission. In the second one (right), B is sending to A and C is sending to D, and there is also no interference, since the receivers only hear also one transmission. They are not possible in MACAW, and the reason lies in the nature of the control packets exchanged between both senders and receivers. These control packets, that gives the protocol a good avoidance of collisions, makes that both participants in a transmission have to send and receive something, and thus do not allow some transmissions to take place at the same time.

Figure 3.7: Because of the control packets, these transmissions cannot occur at the same time in MACAW

Concluding, the MACAW protocol may work with high performance in situations where traffic is not bursty, and where power is not a critical issue. However, it would not be the optimal mac protocol to choose in a wireless sensor network, for example, where battery life is a critical issue and MACAW does not offer any special technique that puts the nodes to sleep in order to save power. Besides, it would also not be a good idea to apply it in a network with a very high load of traffic, because some of the scenarios related above would may take place, and some nodes could access the medium more often than others, what will produce an unfair distribution of the medium.

3.1.2 PAMAS

PAMAS means Power Aware Multi-Access with Signalling and is a mix of the original MACA and the idea of using a separate control channel. That means that the behaviour is very similar to the protocol explained above, with the main difference that here the control packets are sent by a separate channel. Besides, PAMAS is one of the first protocols in adding power reduction support, by putting the nodes into sleep when they do not need to be awake. Next, I will explain how this protocol works and how reduces the power consumption.

How it works

PAMAS works in a similar way as MACA. RTS and CTS messages are exchanged in order to avoid collisions, but the main difference is that in PAMAS, these packets are sent by an other channel, different from where the data is sent. See below the state diagram.
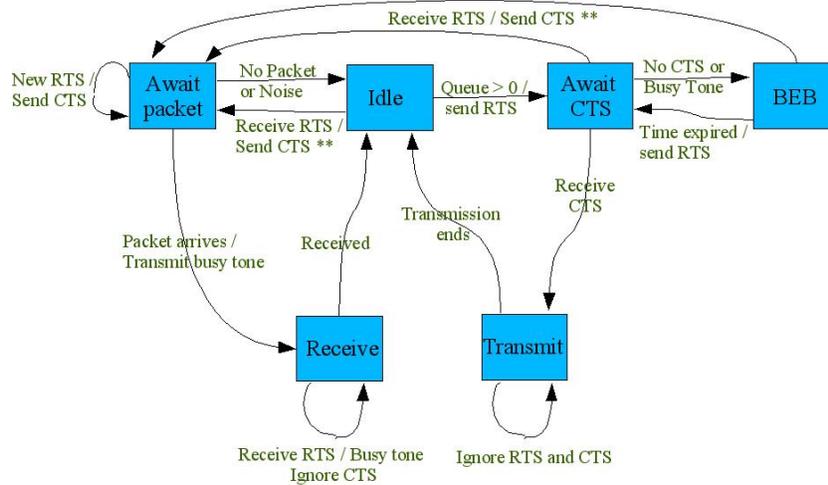

Figure 3.8: Diagram of PAMAS

Initially a node is in the idle state, that is, the state where a node is not sending or receiving. When it has data to transmit, then it sends a RTS to the receiver and goes to the Await CTS state. In this situation, the node sets a timer and waits for the CTS from the receiver. When that packet arrives, the node will turn then to the Transmit packet state, where it will send the data to the receiver. Once the data is sent, it will go again to de idle state.

On the other hand, a node that is in the idle state and receives a RTS, will send a CTS to the sender and will enter the Await packet state, where it will set a timer. It will not leave this state unless the timer expire or data start arriving. If that happens, it will enter the Receive packet state, and when the data is received, it will go again to the idle state.

As the diagram shows, a node in the receive mode transmits a busy tone through the control channel whenever it receives a RTS packet. Thanks to this feature a sender will know if the receiver is already busy with another transmission, avoiding the hidden terminal problem. This node will enter the

BEB state, where the binary exponential backoff algorithm will be executed, and when the timer expires will send again the RTS packet. These events will repeat over and over until the receiver answers with the CTS, meaning that the receiver is ready to receive data. Also, the sender will leave when it receives another RTS from another node. Thus the sender is not blocked for a long time if the receiver has a lot of data to receive.

Power issues

The waste of power is a critical issue in most ad-hoc networks. Nodes spend a lot of power when sending, listening, and even when they are idle. Ideally, nodes would just listen when they are the receivers of some packet, but actually most of them overhear information that is not for them. Because of overhearing, ad-hoc networks may feel an important waste of power.

In PAMAS, overhearing is reduced by making the nodes go to sleep when they hear a transmission nearby. Specifically, they identify two situations where a node should go to sleep. The first situation takes place when a node has nothing to transmit, and hears a transmission nearby. If it was awake with nothing to transmit, it would overhear the transmission of its neighbour, wasting power. To avoid that, it ought to sleep. The other situation takes place when a node has at least one neighbour transmitting and other one receiving. Here, this node should go to sleep even if its transmit queue is non-empty. If it sent, collisions would occur with the neighbour that is receiving.

Nodes put themselves into sleep whenever they detect one of this two situations. Obviously, each node knows whether it has an empty queue or not, and also each node knows if a neighbour is sending something by listening to the data channel. Detecting a neighbour who is listening may seem more difficult, but if we take a look to the diagram explained above we can figure it out. A node that enters the receive packet state transmits a busy tone by the control channel, and also each time it receives a RTS packet. Thus a node will know that its neighbour is receiving something by listening to the control channel.

The aim in PAMAS is to reduce power consumption withouth reducing the throughput. To achieve that goal, nodes ought to sleep just in this two situations, and as long as they hold. Thus, in PAMAS, the node that wants to sleep exchanges some special packets with its neighbours in order to know for how long it should sleep. When it wakes up and notices that there is another transmission nearby, it exchanges again this special packets and goes again to sleep. The special packet the node sends to know the duration

of the sleep is called t_probe(l), where l is the maximum packet length. The neighbours that end its transmission in the interval [l/2,l] will answer with a t_probe_response(t) packet, where t is the time where the transmission will finish. If there is no answer then the node that is about to sleep will probe another interval, on an on, until it receives a t_probe_response(t) packet. All this is a binary search made by the node who is about to sleep, to find out for how long it should go to sleep.

When a node wakes up and notices that its transmitting queue is non-empty, it should find out if the second situation hold, and if not, start transmitting. To know if a neighbour is receiving it sends a RTS packet, and will listen to a busy tone. If several busy tones collide or a busy tone collides with another control packet, the node will probe the receivers with a binary search scheme similar as above, but this time with packets called r_probe(l) and r_probe_response(t). It will probe also the senders, and when it receives the time of both sender and receiver, it will sleep during the minimum time of those two values.

With these power issues PAMAS gets an important power reduction, above all in complete networks, where all nodes are in range of all the rest and there is a lot of overhearing. The most important thing is that PAMAS takes profit of the moments where a node cannot send or receive to put it into sleep. This makes that this savings of power have no effect on throughput.

Conclusion

The main feature of PAMAS is the fact of using a separate channel for signaling. This idea makes possible some situations that in MACA or MACAW were not possible. For example let us think about the situation in the next picture, where A is sending to B and D wants to send to C. In MACAW, after D sends the RTS to C, C could not answer with a CTS, since this sending would interfere in B. However, in PAMAS, C can answer without interfering, because the control packets are sent by a different channel. We observe here that the transmission of data is not affected by the transmission of RTS or CTS packets, so this protocol offers a good avoidance of collisions. However, it is true that the fact of dividing the channel divides also the bandwith, making the bandwith dedicated to send data smaller.

The fact of putting the nodes into sleep reduces the waste of power due to overhearing, and without effect on the throughput. If we add the good avoidance of collisions, we deduce that this protocol increases the power reduction if we compare with MACAW. However, the power savings could be better, as we will see in the next protocol. Let us imagine the case of a

Figure 3.9: In PAMAS, nodes can answer to a RTS by the control channel

network where no one is sending during a long time, remaining all the nodes silent. Nodes, in PAMAS, will be awake and wasting power all the time, while they could be sleeping and saving power. We will see in next sections how to improve this situation.

There is another drawback that makes this protocol also non-fair. In the situation where a node A is sleeping because there is a transmission nearby, if some other node B wants to communicate with A, B has to wait until A wakes up. This will happen when the transmission close to A ends. However, if the sender of that transmission has an infinite queue, it will be sending all the time and A will be also sleeping, so B will not be able to communicate with A. In the previous protocol, the MACAW, this problem was solved by sharing a commun back-off value by all the nodes.

3.1.3 S-MAC

Designed for wireless sensor networks, S-MAC (sensor-MAC) is a protocol which has energy conservation and self-configuration as primary goals. Following the scheme of PAMAS, S-MAC makes the nodes sleep when there are other transmission nearby, and adds new energy features to reduce the power consumption even more. The fact of putting the nodes into sleep periodically is the most characteristic. In the next sections I will explain this features in detail.

Periodic listen and sleep

In a wireless network, it can happen that all the nodes are idle for a long time. In that situation, nodes are awake, waiting por a possible packet reception. In a network were the load of traffic is very low, the loses of power due to idle listening can be quite important. Because of that, S-MAC introduces the periodic listen and sleep technique, where all the nodes divide their idle period in two portions, one where the node is listening and the other one where the node is sleeping. Thus the energy savings increase, although the latency also increases. See the next figure.
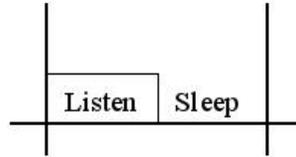
Figure 3.10: Periodic listen and sleep in S-MAC

The exact moment of time when a node goes to sleep and the moment of time when it begins listening is called schedule. In S-MAC, all the nodes try to have the same schedule, so that communication can take place properly. To achieve that, nodes must synchronize together. S-MAC divides the nodes into two kinds, synchronizers and followers. At first, a node listens for a certain amount of time, if it does not hear a schedule from another node it declares itself a synchronizer, and choses a random schedule. From now on, that node broadcasts periodically its own schedule. On the other hand, if a node hears another schedule, it adopts that schedule, waits a random amount of time and broadcasts it again. That node is a follower. The goal of this algorithm is to have a network with one synchronizer and the rest followers, then all the nodes would have the same schedule. However, it is possible that two or more nodes become synchronizers at the same time, broadcasting different schedules over the net. In S-MAC, if a node hears another schedule, after it selects one, it adopts both of them. Thus all the nodes are able to listen to its neighbours, although the nodes with more than one schedule have less time to sleep.

Once the schedule has been selected, nodes must update their schedules periodically, otherwise the clock drift would be larger and larger. To prevent that, SYNC packets are sent between nodes periodically. This king of packets indicate the relative time where the sender is going to sleep. Therefore all the neighbours can update the corresponding schedule.

In S-MAC, the listening time is divided also in two portions. The first one for listening to SYNC packets and the second one for listening to data. This scheme allows the nodes to hear both SYNC packets and data.

Collision and Overhearing avoidance

In order to avoid several nodes sending packets to another node at the same time. S-MAC, like other contention-based protocols, incorporates some techniques that decrease the number of collisions. One of them is the physical

carrier sense. Before sending either the SYNC or the data, a node sense the medium during a random amount of time slots, if it detects no activity in the medium, then sends. And on the contrary, if it detects another transmission, then it sleeps until the end of the communication.

Besides, S-MAC follows the RTS/CTS scheme each time a communication is going to take place, and all the neighbours that hear either a RTS packet or a CTS packet go to sleep until the entire transmission finishes. This scheme is controlled by the NAV variable. NAV stands for Network Allocation Vector and is a variable that indicates for how long a node should not send because of another transmission. Every time a node overhears a packet which is not for him, it updates the NAV. Therefore, if the NAV is bigger than 0 the node should sleep until it gets zero. This technique gives the node the oportunity to measure the activity of its neighbours and act in consequence.

Message Passing

A important feature of S-MAC is the message fragmentation. Due to the nature of wireless networks, sending long packets can produce low rates of efficiency, since an error in one bit invalidate the whole message. Having that into account, S-MAC sends a long message, which has only one RTS and one CTS packet, in fragments. And each time a fragment is received, the receiver sends an ACK packet. These packets attach also the duration field, so if a neighbour wakes up in the middle of a transmission, it will hear the ACKs and will know that a transmission is still taking place. Thus, there are less retransmissions of packets and the efficiency is increased.

Conclusion

S-MAC is a contention-based protocol whose primary goal is saving energy. In order to reduce collisions, nodes contend for a random amount of time before transmitting, this does not avoid all the collisions, but contribute to reduce them considerably. The technique of sleeping when another transmission is taking place, and the technique of sleeping from time to time. Reduce the energy even when the load of traffic is very low. However, as only one portion of the time is used for listening, if a message-generating event takes place during sleep time it has to wait until the listening time, producing an increase in the latency. This makes this protocol very suitable in networks where throughput is not important, but energy conservation. And makes it unsuitable in networks where throughput is the firs goal.

3.2 TDMA

Unlike contention-based, in TDMA nodes cannot send at any moment, they are assigned each one a time slot, where they are allowed to send. TDMA protocols have the advantages of avoiding collisions and control packets, but usually the disadvantage of the delay produced until the right slot arrives. One example of TDMA protocol is the DE-MAC [6].

3.2.1 DE-MAC

Based on TDMA, DE-MAC (Distributed Energy-Aware MAC protocol) is a MAC protocol for Wireless networks whose main goal is to save power while maintaining efficiency. Because of being a TDMA protocol, DE-MAC does not waste power produced by collisions or control packets. However, its main feature is that in DE-MAC nodes are not treated in the same way, as most protocols do. It balances the network by making the nodes that spend more power sleep more time. To achieve this, nodes can operate in two different phases: the normal phase, where nodes exchange data packets between them normally, and the voting phase, triggered by the critical nodes, where they decide who should sleep more than the rest.

Normal phase

Like any other TDMA protocol, in DE-MAC, time is divided into slots, and each node is assigned a number of them. During a time slot, only the node assigned to it is allowed to send, and the nodes that are connected to that node must be awake for a possible reception.
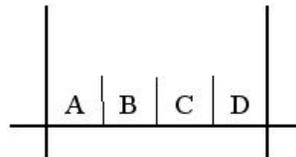

Figure 3.11: In TDMA, time is divided into slots

Thus, in the normal phase, each node will turn off its radio and go to sleep when it is in its own time slot and has nothing to send, or when it is in another time slot but none of its neighbours transmit in that one. A node has to be awake during the time slot of another node, just if there is connectivity between them.

Voting phase

In order to balance the energy consumption, nodes adjust their number of time slots (eihter one or two) when a node reaches a critical value of its battery. The way this is done is by means of a voting phase, where nodes exchange their energy levels, and compare them. The node or nodes with less energy are the winners, and the rest the losers .

The way this is done is as follows: First, when a node enters a critical situation of its battery, it starts the voting phase by sending to all its neighbours a packet with its energy value. The rest of the nodes compare that value with their own, if the value received is smaller they answer a positive value, and if the value is bigger then they answer a negative value. If the critical node has the smallest battery level then it declares itself the winner and sets for himself twice the number of time slots of the rest. Then it will sleep during more time if it has nothing to transmit and also will be awake for less time since its neighbours have less number of time slots.

After the voting phase, which is integrated in the TDMA scheme, nodes go back again to the normal phase, and the value of the previous winner is set as the next threshold that will trigger another voting phase.

Conclusion

Because of being a TDMA protocol, DE-MAC has the main advantages of these kind of protocols: absence of collisions and control packets. Besides, this protocol has into account that in wireless networks, normally not all the nodes spend the same quantity of power, so the network should adapt to balance the battery levels of all the nodes and therefore make the life of the network longer.

However, in DE-MAC nodes do not sleep too much, they are awake most of the time if they have a lot of neighbours and all of them are connected. In that case nodes will only sleep during its own time slot and just if they have nothing to transmit. Scalability is not good here, the more nodes the network has the more delay, since nodes have to wait till their own time slot to send. Besides, in DE-MAC nodes listen to each other even if they have nothing to receive, that means that if there is little traffic the power consumption increases significantly.

Chapter 4

Comparison of protocols for Orienteering

As it was pointed out before, the selection of a good MAC protocol depends strongly on the scenario it is to be applyed. In some cases, a general MAC protocol, like the ones explained in the last chapters, will suit well with no modifications. In other cases, some changes will be required to achieve the goal in an acceptable way.

In this chapter, we will start studying the characteristics of the scenario of orienteering, and after that we will analyze and compare the protocols explained in the last chapter, always with the scenario of orienteering in mind. The goal will be to conclude which components of those protocols are good for the final protocol, that will be described in the next chapter.

4.1 Features of Orienteering

Even that it is possible that the protocols of the last chapter work well in most situations. Normally, it is always possible to improve the protocol having into account factors like the number of nodes of the network, the type and size of the data is exchanged between them, the load of traffic, the special role of some nodes, the disposition of the nodes, etc. In this section, we will study the scenario of orienteering, and its main features. That will help us to know what is exactly what we want to design.
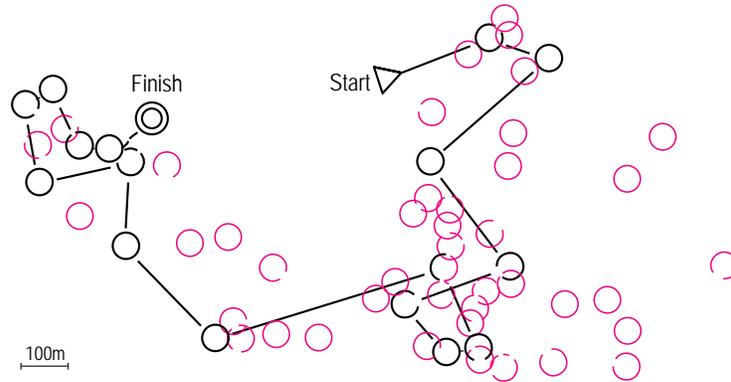
Figure 4.1: A real orienteering example

4.1.1 Goals

Because of the lack of power support in the forest, nodes have to be battery-powered in order to work. For comfort reasons, it is also not expected that the batteries are changed in a long time. Because of that, the main goal to achieve will be that the nodes save as much power as possible.

However, we cannot forget that a specific level of throughput has to be mantained. Since the information that has to arrive to the server represents the data corresponding to a race, and as we want to make a representation as real as possible of what happens, we cannot afford to have long delays. To be more concrete, we will not accept delays longer than 2 or 3 seconds.

In conclusion, the goal will be to reduce the power consumption as much as possible, keeping the enough throughput that allows the proper following of the race, so that the excitement of the race is not removed.

4.1.2 Type of data exchanged

Whenever a participant triggers a control point, the sensor node just has to send two pieces of information to the server: the identification of the participant and the current time. Because of sending such small packet, we will have to find out if it is worth to send control packets.

As it was mentioned in the last chapter, the contention-based protocols exchange two kinds of control packets before starting a transmission: an RTS and a CTS. These packets usually just attach the size of the message it is going to be send, and its goal is to let the neighbours know that a

transmission is going to take place, so that they defer on sending.

When a message is very long, having control packets is very useful. If the long message was sent alone, and there was a collision, the sender would find it out after the whole transmission, with the corresponding delay, and it would transmit it again, with the consequent risk of having more collisions. If the message is sent with control packets, the risk of collisions is reduced to a minimum sized packet.

However, if the message to send was as small as the RTS packet, for example, it is obvious that sending this packet too would be unefficient, since we would be increasing the traffic considerably, maintaining the same risk of collisions. Besides, it is also worthless to put the neighbours into sleep during such a small amount of time.

In the case of orienteering we want to send a packet, which is slighty bigger than the RTS and CTS packets. So if the protocol chosen is contention-based, most probably the control packets will not be needed. In that case we would need some sort of acknoledgement to guarantee the proper delivery of our messages.

4.1.3   Role of the nodes

In the designed network, all nodes must sleep from time to time, and must send out the data of the participant when it arrives. They also have to be able to redirect messages in the direction of the server. The nodes which are closer to the server will have to work more than the others, so perhaps it would be a good idea to make them sleep more, in order to make the network life longer.

Besides, it is also important to note that data in the network will flow always in the direction of the server. And, excepting for synchronization and configuration issues, it is not expected that the server will send anything to the nodes. So in general, a node will receive data from the contrary side of the server, and will send the data to the side of the server.

4.1.4   Disposition of the nodes

As the nodes are placed in a forest, and their disposition will change in every different competition, the designed protocol must be open to all possibilities, and must work for every topology. The only condition is that all the nodes must be able to communicate with the server, directly or by means of other node.

The place of the server in the network is also an important issue. It could be placed in a corner or in between the nodes. If it is in a corner, messages from the nodes of the other side will have to do more hops to get to the server. Besides, if there is only one node directly connected to the server, as it has to retransmit the messages from all the network to the server, that node will have to work much more than the rest, so it will also spend more power. If there are more nodes connected to the server, there will be more roads to arrive, so the work of the nodes will be more balanced.

The designed network must work well independently of the place where the server is, as long as it is connected to the rest of the nodes. However, it is possible that the designed network might reach better performance for a specificic place, which in that case will be mentioned in this report.

Another important aspect is the mobility. As control points are not moved during the race. We do not expect that the nodes will change and that the design protocol should be able to adapt itself. Network is static, nodes will be placed in an specific place and that place will not changed during the race.

4.2 Contention-based vs TDMA in small networks

Now that our specific scenario has been analyzed, it is time to apply the MAC protocols discussed in the last chapter and compare them. More precisely, the contention-based and TDMA schemes will be compared, and the criteria will be the power efficiency and the delay.

We will start by small simple networks of four nodes before we try to generalize for a bigger network that could be typical for orienteering. We will assume that all are orienteering networks, so one of the nodes will be the server, and the messages sent by the rest of the nodes will have to arrive to the server. The sleeping time and the delay will be discussed for each of the sides.

For the contention-based, we will apply a protocol where all the nodes can send in the same time period, and we will supposed that some contention mechanism is used to reduce the number of collisions. Also, and as we want to reduce the power consumption, the periodic listen and sleep scheme will be used. Thus, it will be very similar to the S-MAC protocol studied on the third chapter.

For the TDMA side, the nodes will not share the same time period, they will have their own time slot instead, where they will be able to send, or to sleep if they do not have anything to transmit. A node will sleep also during
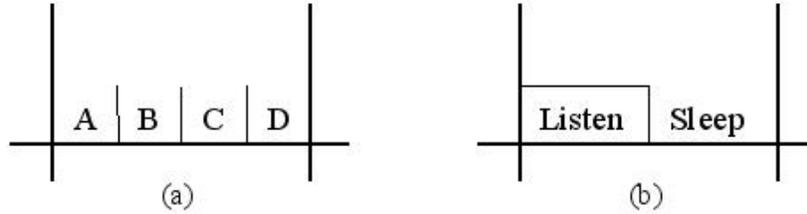
Figure 4.2: TDMA (a) and S-MAC (b) basic schemes

the time slot of another node if it does not have to receive anything from it.

4.2.1 Full network

As the picture ilustrates, the first network to analyze will be formed by four nodes, where all are connected between them, that is, full connected. The server will be node D, drawn darker. So the messages sent by the rest of the nodes must arrive to D.
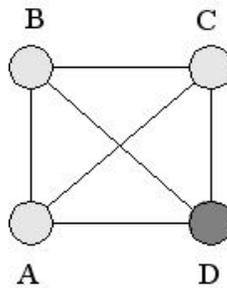

Figure 4.3: Full connected network of four nodes

Contention-based

The time period of the contention-based protocol for this network, is divided in two parts. In the first one, the four nodes listen to the medium, or send if they have to, not without contending first. And in the second part all the nodes sleep. To analyze the power consumption and the delay properly, first we need to find out which nodes are able to send data, and in which direction it will be sent, that is, the routing.

As D is the server, A, B, and C will have to send packets to D, from time to time. D, on the contrary, does not have to send packets to any node. There are also several alternatives for the data to arrive to the destination. For example, as they are in range, A could send the information directly to D. However, it could also send it to B instead, for example, and then B could retransmit it to D.

In contention-based all the nodes share the same time period to send. That means that, excepting when there are collisions, only one will win the media and send. The more times the nodes have to send, the more possibilities of having a collision. Because of this, if possible, it is better to reduce the number of hops of a message as much as possible. Thus, the best routing in our example will be A,B, and C, sending the messages directly to D.
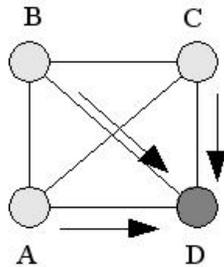

Figure 4.4: With contention-based, the less hops the better

Having chosen the best routing in this case, we can discuss now about the power consumption and the delay. The sleeping time in a contention-based protocol with periodic listen and sleep is not fixed. It could be $\frac{1}{2}T$, or $\frac{2}{3}T$, etc, being T the period time, depending on how much power is wanted to save. The advantage of this scheme is that, apart of being able to set the sleeping time at ease, all the nodes sleep the same, so the network is balanced. The main drawback relies on the fact that the whole network is idle during the sleeping time. The delay depends on the amount of data the nodes have to send. If nodes have always data to send there will be more collisions, and it will take a long time for the messages to arrive to the destination. Nevertheless, if the traffic is not bursty, the messages will reach quickly their destination. Note also that the messages that arrive to the queue during the sleeping time have to wait until the active time to be sent.

In orienteering, nodes send data just when a participant triggers the

corresponding mechanism. That, in computer terms, is not very often. If this was a small orienteering network, there would not be a lot of collisions, so the delay would be very small. The sleeping time could also be set to $\frac{2}{3}T$ for example, and the worst delay, supposing no collisions, would occur when a participant triggers the mechanism just in the beginning of the sleeping period, that would be $\frac{2}{3}T + T_c + T_s$, being T_c the contention time and T_s the time to send. This would also be acceptable.

TDMA

To apply TDMA to this network, the time period will be divided into 4 time slots. Each time slot will be assigned to a node, and only the node assigned to a time slot will be able to send data during that time. In this scheme, nodes will sleep in its own time slot if they have no data to send and also in the time slot of another node if they do not expect to receive data from it.

In contention-based, we concluded that the best routing option was that all the nodes send the data directly to D. In TDMA, however, if we applied this routing, node D (the server) would sleep just in its own time slot, since it would have to be awake in all the rest slots in order to listen to A,B and C. Thus, we would have that D just sleeps $\frac{1}{4}T$, which is quite a small amount of time compared to the levels reached in contention-based.

In order to improve the sleeping time, next we will measure it for each node and for some of the routing options. We will use λ_n as the probability of the node n to send. λ_{mvn} will be the probability that m or n sends.

Routing	A	B	C	D
	$\frac{3}{4} + \frac{1}{4}(1 - \lambda_a)$	$\frac{3}{4} + \frac{1}{4}(1 - \lambda_b)$	$\frac{3}{4} + \frac{1}{4}(1 - \lambda_c)$	$\frac{1}{4}$
	$\frac{3}{4} + \frac{1}{4}(1 - \lambda_a)$	$\frac{3}{4} + \frac{1}{4}(1 - \lambda_b)$	$\frac{2}{4} + \frac{1}{4}(1 - \lambda_{bvc})$	$\frac{2}{4}$
	$\frac{3}{4} + \frac{1}{4}(1 - \lambda_a)$	$\frac{2}{4} + \frac{1}{4}(1 - \lambda_{avb})$	$\frac{2}{4} + \frac{1}{4}(1 - \lambda_{avbvc})$	$\frac{3}{4}$

34

As we advanced above, in the first routing scheme, as all the nodes send the data to D, it sleeps very few ($\frac{1}{4}T$) and A,B, and C sleep a lot (more than $\frac{3}{4}T$). On the second routing, D sleeps $\frac{2}{4}T$, because it has to listen to A and C, and in this case C sleeps less than before, now that it has to receive data from B. The last routing option saves more power in the network. The node that sleeps less is C, with a little bit more than $\frac{2}{4}T$. If we just had into account the sleeping time we could propably choose right now the last routing scheme. However, we have to think also about the delay.

The delay in a TDMA protocol depends strongly on the order in which the time slots are disposed. If a node A has to receive data from another node B, it would be good if the time slot of A was just after the time slot of B, so that when the message arrives to A, it can be inmediately retransmitted to another node. So for our example a good allocation could be this order: *ABCD*.
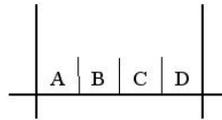
With this disposition of the time slots, the delay in the last routing scheme would not be so bad, even that messages from A have to visit B and C. As these time slots are in the order of the routing, the messages are inmediately retransmitted and the delay is improved. However, the delay in the first routing scheme is smaller. Messages arrive inmediately to the server and the longest delay occurs when the message arrives to the queue of the node just after its time slot.

It is an interesting point to think that, if D was not the server, but another node that has to send data to a node E, for example, there would not be a big difference between the delays of the first routing scheme and the third one. This is because, even that with the first routing scheme, a message from A would reach D sooner than with the third routing, D will have to wait anyway until its time slot to retransmit it to E. The difference is that with the first routing scheme, the message from A has to wait in D during B and C time slots, when with the third routing, it is continuosly being retransmitted.

If this was a small orienteering network and D was the server, as the nodes do not have to send data very often, we would probably choose the routing option that offers more sleeping time, that is, the third one. Besides, and in the same way as contention-based, here it would be possible to make

the sleeping time of the nodes even bigger by making some time slots bigger, or by creating another time slot where all the nodes sleep. This however, would increase the delay.

4.2.2 Ring

The next network to analyze will be a ring of four nodes. As the last network, D will be the server, and the rest of the nodes will have to make the information arrive to it.
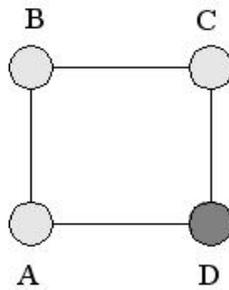

Figure 4.5: A ring with four nodes

Contention-based

As it was deduced in the full network, for the contention-based scheme, we will try to find the routing with less hops. The situation is quite similar to the full network, the only difference is that now B is not able to send the data to D directly, and it has to send it to A or C. We will pick up for example the situation where B sends the data to C.
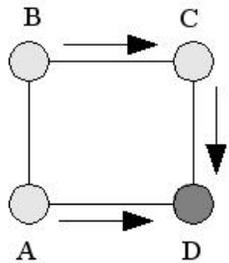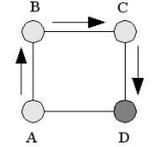
The sleeping time of the nodes will be the same for all of them, and it will be able to change it depending on our interests. However, the delay of the messages of B will be longer than the one of the messages of A and C. As A and C are connected directly to D, their messages will arrive inmediately, if there are not collisions. The messages of B will travel first to C, and then to D. So they will last one time period more, if we supposed that only one message can be sent per period, and that there are no collisions. In this case the delay for B is better in the full network.

Nevertheless, there is an advantage in the ring in front of the full network. In this network, if A and B sent data at the same time, there would be no collision, since C is not in range of A and D is not in range of B. In a same way, B and C could send data at the same time. C would not listen to the message of B, but at least D would receive C's message. So, although the delay of B is increased in this network, the probability of collisions is reduced.

Again, an orienteering network of such small number of nodes and with this disposition, could achieve our goals with the contention-based scheme.

TDMA

We will start measuring the sleeping time for every possibility of routing in this network. In this case there are only two.
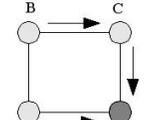
Routing	A	B	C	D
	$\frac{3}{4} + \frac{1}{4}(1 - \lambda_a)$	$\frac{3}{4} + \frac{1}{4}(1 - \lambda_b)$	$\frac{2}{4} + \frac{1}{4}(1 - \lambda_{bvc})$	$\frac{2}{4}$
	$\frac{3}{4} + \frac{1}{4}(1 - \lambda_a)$	$\frac{2}{4} + \frac{1}{4}(1 - \lambda_{avb})$	$\frac{2}{4} + \frac{1}{4}(1 - \lambda_{avbvc})$	$\frac{3}{4}$

As it is shown in the table, the results are very similar to the ones obtained in the full network. The best sleeping time is on the second routing option, where the node that sleeps less is C, with more than $\frac{2}{4}T$. The rest of the nodes sleep also during a similar amount of time, excepting A, that sleeps more (because it does not have to listen to any node).

The difference of the delay between both situations relies on the messages

37

from A. In the first routing scheme, A's messages arrive inmediately to D, while in the second routing, they have to travel through B and C.

In the same way as the full network, for orienteering we would probably choose the second routing option, since it is the one that offers more sleeping time, with acceptable levels of delay.

4.2.3   Line

For the next small network to analyze, we will use the line, that is, four nodes just connected one by one.

Figure 4.6: A line of four nodes

Contention-based

The sleeping time, as it is characteristic in the protocols with periodic listen and sleep, will be the same for all the nodes of the network, and will be set at ease.

The delay, however, will be different than the last networks. Now only one routing scheme is possible. The scheme where A send the information to B, B to C, and C to D. In this situation, only C sends directly to D. As a consequence, A and B's messages will last more time in arriving to D. For example, if we suppose that it is only possible to send one message per period, if there are no collisions and B and C are quiet, a message from A will last more than $2T$ in arriving to D, while a message from C will arrive to D inmediately.

Figure 4.7: Routing of a line of four nodes

Although the delay has increased, the probability of collisions decrease in this network. In the full connected scenario, when any two nodes sent at the same time, there was a collision. In here, C for example does not have

38

any risk of collision. It may occur that B will send at the same time as C, but C's message will arrive without collision to D, since D is not in range of B. The same happens between B and A: no matter if A is sending, B's message will arrive to C, if C is not sending.

Thus, in this network, the closer a node is to the server, the less delay and less collisions its messages will have. However, as it has to retransmit the messages from more nodes, it will spend more power.

Probably a line of just four nodes would work well for orienteering, but if the line was composed by a large amount of nodes, the delay of the further node would be too big, and that would not be an acceptable solution.

TDMA

As it has been said in the contention-based section, in the line network there is only one possible routing. As we have done for the other networks in the TDMA section, we will measure the sleeping time for each of the nodes:

Routing	A	B	C	D
A → B → C → D	$\frac{3}{4} + \frac{1}{4}(1 - \lambda_a)$	$\frac{2}{4} + \frac{1}{4}(1 - \lambda_{avb})$	$\frac{2}{4} + \frac{1}{4}(1 - \lambda_{avbvc})$	$\frac{3}{4}$

We achieve similar results to the ones obtained in the last networks. The node that sleeps less is C, with more that $\frac{2}{4}T$, and the node that sleeps more is A, with more than $\frac{3}{4}T$. All nodes sleep more than $\frac{1}{2}T$, since they just have to listen to one time slot.

To have an optimum delay, the order of the time slots must be in the direction of the routing: $ABCD$. Thus, a message from A, which is the furthest node from the server, will last less than T, if B and C are quiet. This is a very good delay having into account that the number of hops from A to the server is three.

It would have been also possible to use three time slots instead of four. That is because A and D can send at the same time without interfering in some node. Using three time slots instead of four, decreases the delay, but also decreases the sleeping time.

4.3   Contention-based vs TDMA in bigger networks

The number of control points in a orienteering race is not fixed. It can be 20, 100, or even more, depending on the race. Besides, there can be a lot of them, and only use a set of them for the race. Because of that, we must

compare also the sleeping time and the delay for the contention-based and the TDMA schemes in bigger networks. In this section, we will apply the knowledge obtained in previous sections, to do that.

Contention-based

In the same way as it occured in the small networks, applying our contention-based protocol to a network with a large number of nodes, will make that all of them sleep the same amount of time. Also, the sleeping time could be set depending on our interests. Normally it will be set as high as possible, and as long as we accomplish the wished delay.

The delay depends on the topology of the network. Let us imagine a network of 12 nodes, for example, where all of them are connected between them, and the server is one of them. If in one moment, only one of them wanted to send, and the rest of them were quiet, this message, as it is the only one, would arrive inmediately to the server. If, on the contrary, two or more nodes sent at the same time, resulting in a collision, this messages would suffer a delay, since they would have to contend, win the media, and send again in a next period. In this situation, the delay would depend on the contention mechanism, and chosing a good one would prevent most of the collisions.

In the case of a line network of 12 nodes, where the server is in a extreme. A message from the node in the other side will have to make 11 hops to arrive to the server. If we suppose that only one message can be sent by period, and we assume that no other nodes are sending, the delay of this message will be $11T$. And in the normal situation where the rest of the nodes also send data to the server, this delay will be even longer.
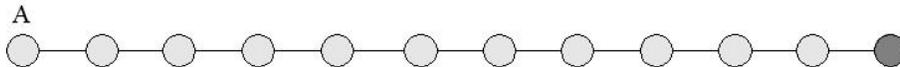

Figure 4.8: Node A has a long delay when the number of hops to arrive to the server is very high

In this case, it could be a good idea that nodes far from the server had more priority than the ones which are closer. This could be done by the nodes by putting the received messages in the first positions of their queue, so that these messages can be delivered sooner than their own messages. Thus, in the example, messages from A would not have to wait in the queues of other nodes. And the nodes close to the server, as they have a smaller

delay, can afford to give less priority to their own messages, so that far messages arrive sooner.

Therefore, if we want to apply this protocol to a big network, and we want to achieve acceptable levels of performance, we have to choose an optimum contention mechanism, to minimize the number of collisions, and reduce the number of hops as much as possible. A message from a node that has to do 20 hops to arrive to the server would probably have a too big delay for orienteering.

TDMA

Applying TDMA to a big network is not an obvious task. The number of time slots, their allocation, and the routing must be selected carefully in order to achieve a good performance. And normally a whole knowledge of the topology of the network is needed to do that.

The number of time slots, for example, must be set in a way that avoids two or more nodes sending at the same time and interfering in another node. Thus, in the full connected network of 12 nodes, there has to be at least 12 time slots, less would mean that more than one node might send ant the same time and produce a collision. In the line network of 12 nodes, however, 3 time slots would be enough. As the picture ilustrates, nodes with the same time slot number can send at the same time without interfereng in another node.
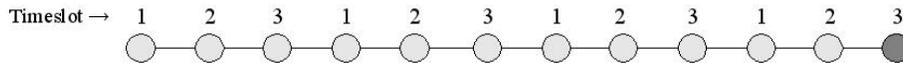

Figure 4.9: In a line network only three time slots are needed

A small amount of time slots, normally leads to a better delay. If we have a look to the next picture, we can see how a node is able to send with more frecuency if the number of time slots is smaller. If we suppose, for example, that node 1 wants to send a message to node 2, just after its time slot, in the left situation it will have to wait two time slots, while in the right example it would wait during 11 time slots. Besides, if that message was lost, because of the unreliability of wireless, in the right example it would have to wait again for another 11 time slots.

But having more time slots can also have some interesting advantages in our scenario of orienteering. Let us go back to the line network of twelve nodes again. In there, nodes in the time slot two have to be awake during
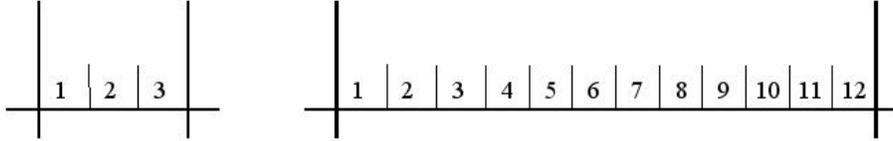
41

Figure 4.10: A three time slots scheme (left) and a twelve time slots scheme (right)

time slot one. That means that they can only sleep during time slot three and during their own time slot if they have nothing to transmit, that is a little bit more than $\frac{1}{3}T$. If we used four time slots instead of three, although the delay would be slightly increased, nodes would be able to sleep during another time slot, so their sleeping time would be bigger, more than $\frac{2}{4}T$.

Normally, in all the cases it is possible to add more time slots than needed, or making some of them bigger. However, in some situations, the number of required time slots is very big, and it is not possible to reduce it without having risks of collisions. The full network of twelve nodes is a good example of that. In that case twelve time slots are needed, and if we used less we would have to introduce some kind of mechanism that prevents collisions.

Furthermore, in the full network example, as all the nodes are connected between them, each one would have to be awake during the time slots of all its neighbours. This would be very unsuitable for our scenario, since we want to increase the sleeping time. Fortunately, we could remove most of those connections, and select a proper routing scheme, in the same way we did in the full connected network of four nodes in a previous section. That would increase the sleeping time significantly.

So important as the number of time slots, is their allocation. The order of the time slots must go in the direction of the routing, so that messages travel to their destiny as fast as possible. In the example of the line network, this allocation culminates in a very small delay. If the order of the time slots was in reverse order, we would have that messages wait a long time after each transmission, to be retransmitted again, producing a very big delay.

If all this characteristics are set carefully, TDMA can be a nearly optimum solution, where messages are always travelling and there is no risk of collisions. However, in networks where the topology is unkown, like ours, these features can be very complex to find out. Besides, in most cases it may be also possible to give more sleeping time to the nodes, by making the

time slots bigger or by adding more. However, the delay would increase.

4.4 Conclusion

Now that we have studied the effects of contention-based and TDMA in several kind of networks, it is time to gather all the information and summarize the most important things, in order to obtain the final conclusions. Next, a table with the main advantages and disadvantages of contention-based and TDMA is shown.

	Contention-based	TDMA
delay	χ long when the number of hops is big χ collisions increase the delay χ contention also increases the delay χ synchronization makes it bigger	$\sqrt{}$ small if the routing is appropiate χ synchronization makes it bigger χ nodes have to wait for the right time slot
power saving	$\sqrt{}$ all the nodes sleep the same χ collisions increase the power consumption	$\sqrt{}$ good if the routing is appropiate χ bad in nodes that have to listen to a lot of neighbours

In delay issues, we have seen along this chapter that contention-based only gets acceptable results when the number of nodes a message has to go through is small. With contention-base, during a portion of each period the whole network is sleeping, so the messages that need several hops, will need several periods as well, and because of that they will suffer an important delay. Moreover, messages will suffer delays when they contend, and also each time a collision takes place, since the node has to realize first that there has been a collision and retransmit again.

This also makes nodes spend some power that should be unnecessary. Each time a collision occurs, the message has to be sent again, so the power a transmission takes is spent twice, for both the sender and the receiver. Although it is not possible to make the collisions disappear completely, it is possible to select a good contention mechanism that minimizes them, so that the power consumption improves, as well as the delay. Besides, the periodic listen and sleep scheme, contributes also positively to the power savings.

Contrary to contention-based, TDMA can reach good levels of delay even when the number of hops is very high. This is because in TDMA the network is not paralysed during several portions of time. Nodes work while others sleep, and vice versa. In fact, contention-based could be thought as a TDMA scheme with two time slots, where all the nodes send and receive

in the first time slot, and all of them sleep in the second slot. TDMA thus adds more time slots and makes that at each time slot there is always a node working, so the delay improves. There is an aspect, however, that affects negatively to the delay. That is the synchronization.

In TDMA, as well as in contention-based, some synchronization is needed, in order to assure the perfect working of the network. As this is carried out by sending time packets between nodes, usually it is not very precise, and leads to small drifts between the clocks of the nodes. Because of this, time slots should be a little bit bigger, to support possible drifts in the sending of these messages. In contention-based this would be also necessary, but in TDMA, the delay of this preventive time will be bigger, since there are more time slots than in contention-based.

We have said that TDMA reaches good levels of delay even in big networks. However, and as we have also seen in previous sections, this is usually bound to a good selection on the number of time slots and their allocation. A good selection will improve the delay, and also the power consumption. Let us not forget that some routing options make some nodes sleep more than others, which is not very good for our purposes. To be more concrete, the power consumption will be worse in the nodes that have to listen to a lot of neighbours. This is because these nodes will have to be awake during a bigger number of time slots and therefore will be able to sleep less. Here, the routing algorithm can help to minimize the number of connections a node receives, and thus the number of time slots it has to be awake.

Another important aspect that we should have into account is the complexity in the implementation of the algorithm. Programming TDMA is not an easy task. We need to find out how many time slots we should use, their allocation, and the order, in a proper way that makes the routing to the server faster. As in our scenario the topology of the network will change on each competition, some algorithm that obtains the topology of the whole network and decides about the time slots should be implemented. In contention-based, it could be also helpful to obtain the topology of the network, and make the routings in a way that the number of hops are minimum, but it is not as necessary as in TDMA. Nodes can direct all their messages to the node that first send a message to them.

In conclusion, TDMA has offered good levels of delay and power consumption, even in big networks. Because of this, it seems to be a better solution for orienteering than contention-based, where there are collisions and the network is inactive during certain periods of time. However, TDMA is more complex to implement, while contention-based is simpler.

Chapter 5

Implementation

As we pointed out in the beginning of this thesis, the main goal of this project is to find a solution for the problem of orienteering that saves power and decreases the delay. During the thesis, we have seen several options to achieve these goals. In this chapter, we will implement a final solution, based on the conlusions of previous chapters, and it will be programmed in small sensor devices.

Usually, this means a small device with sensing capabilities (vibration, sound, etc.), limited batteries, and capable of communicating via wireless with other sensors. The networks that have these devices as nodes are commonly called Wireless Sensor Networks.

In this project, we will implement our solution in an specific small device called Embedded Sensor Board (ESB) [7], not because of their sensor capabilities, but because of being built in a power efficient way. That is why they will fit very well in our goals.

Figure 5.1: An ESB (*http://www.scatterweb.net*)

In order to program them, we will use Contiki [9], a small operating system for tiny devices. This will give us the interface we need to control the radio, and all the other components of the sensor boards.

Next, we will see the main characteristics of the program implemented, and after that, in other section, the results and possible improvements.

5.1   My protocol

To start implementation, a first task has to be deciding between the two main ideas we have discussed along this project: contention-based and TDMA. In the last chapter, we deduced that TDMA could reach better levels of delay than in contention-based. However, the fact that it is complex to find the proper routing and time slot allocation makes this scheme difficult to implement. On the other hand, contention-based is a simpler protocol with not so good delay but with acceptable power consumption.

Because of this, and as it is our first implementation of a solution for the problem of orienteering, it would be preferable to build a simple protocol that achieves some of our goals in an acceptable way, instead of building a complex solution that achieves our goals in an efficient way. Therefore we will be able to obtain results very quickly, and that will be very useful for doing improvements and decide whether or not contention-based can definitely be used in orienteering.

So now that we have decided one of the main branches (contention-based), it is time to adapt the main scheme so that we achieve our goals in the most efficient possible way.

5.1.1   Discovery algorithm

As the topology of the network will change in different competitions, we have to implement an algorithm, so that when we turn the nodes on, they can find by themselves the direction of the server, independently of their position in the network. Thus the orienteering messages will arrive to the server properly.

As the diagram ilustrates, initially a node is in the "undiscovered" state. There it will wait until it receives a *Discovery message* (D message). Once that happens, it will contend and send another D message, so that other nodes can be discovered. It will store also the info associated to its father (the sender of the D message). Its father will be now the node to whom it will transmit all its messages. Besides, it will send a *Report message* (R message), this is just a message, intended to arrive to the server to let it

know that one node has been discovered. This message will be sent to its father, that will retransmit it in the direction of the server. After that, the node that sent the R message will set a timer and change to the "Wait for ack" state.
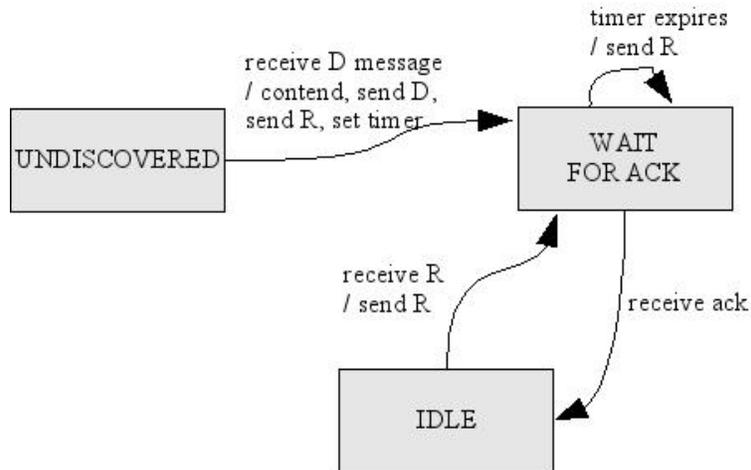
Figure 5.2: Diagram of states of the discovery algorithm

As the wireless media is very unreliable, the node will make sure that its message has arrived properly by waiting for an acknowledgment from its father. Then, it will go to the "idle" state. There it will retransmit all the R messages that receives from its possible children to its father, and thus, to the direction of the server. As before, it will make sure that each retransmitted message arrives to its father.

Eventually, the server will receive the R messages from all the nodes of the network (we suppose that the number of nodes spread is well known). If some node node has not been discovered, because has not heared the proper D message, the server will be able to send another D message, and the rest of the nodes will retransmit it.

Sometimes, it is possible that a message too far away from the server, for example so far that only receives the 20% of the messages, receives the D message properly. This would mean that this node would select the server as a father, even though most of the packets it send would not arrive. To solve this problem, all the nodes will send the D messages with less power than the R messages. Thus, we will make sure that only the closest ones will receive it.

5.1.2   Main features

As we can see, the discovery algorithm is quite simple. Here nodes do not look for the optimum routing option. The only information a node has about the rest of the network is its father, to whom it will send all the messages.

Since the orienteering message to send will be very small (just a few bytes), we will not use control packets. Messages will be sent directly. Nodes will send messages in two occasions: when a participant triggers the associated control point, and when they receive an orienteering message from another node directed to it. In both cases, the message will be sent to the father.

There will be a queue where the orienteering messages will be stored, in case the sensor is busy when a participant comes. For simplicity, the queue will be built as an static stack. This is not the best solution, because if two participants trigger the sensor in a small interval of time, the last orienteering message will have more priority. However, the sensor will give more priority to the messages received from other nodes than its own orienteering messages. As these messages come from a deeper level of the tree, and they have to make a longer trip to the server, they will be put in the first position. Thus the average delay of the messages will be reduced.

5.1.3   Acknoledgements

In order to know if messages reach destination succesfully, nodes will wait for an acknowledgement from the father each time they send a message. If the acknowledgement is not received in a random amount of time, between 1 and 5 periods, the message will be sent again. This process will be repeated until the acknolegment is received.

However, as the father has to send the message again, it is not necessary that it sends also an acknoledgment message to its son. Here, we can benefit from the fact that nodes have to retransmit the information they receive, to save messages, and thus to save power.

Whenever a node sends a message to its father, since it has also to retransmit it, the node will listen to that message as an acknolegdement. In case this message is not heared by the node, but the father has succesfully received it and retransmitted it, when the node sends again the message, the father will detect that it has already been transmitted, and will send the message again, but this time not to its father, to the son instead. Therefore messages will be sent to the server only once.

5.1.4 Periodic listen and sleep

In the implementation, we will adopt the periodic listen and sleep scheme of S-MAC. Since at many moments, the nodes in the orienteering network will be idle (because the participants are most of the time running, looking for the control points), it is worth to have a mechanism that saves power from time to time, even though some delay is lost.

Thus, the main operation scheme of the sensor boards will consist of two parts, one where the sensors are listening, or sending if they have something to send, and another part where the radio is turned off so they neither can send nor receive. To start with, we will use the same amount of sleeping time that S-MAC says, that is, half of the time. The time period will be one second, and the synchronization process is explained on section 5.1.6.
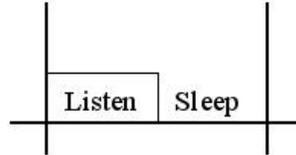

Figure 5.3: The periodic listen and sleep scheme

5.1.5 Contention

As all the nodes can send at the same time, some contention mechanism has to be introduced in the program to minimize the possibilities of having collisions. The typical way to do this is to make the nodes wait a random amount of time before sending. This time has to be enough to avoid most of the collisions but not too long, otherwise the delay would be unacceptable.

It is very difficult to guess correctly at first the right contention mechanism. We will start by making the nodes wait a random of periods between zero to two before sending, then we will see how many collisions there are, and we will act in consequence.

5.1.6 Synchronization

In S-MAC, the way of synchronizing the network was that some nodes, the first ones in being turned on, were the synchronizers, and they sent synch packets from time to time. The nodes that turned on later, heared these packets, and adopted the same schedule. They were called followers. This

49

strategy had the risk of having several synchronizers, so some nodes could follow several schedules and sleep less than others.

In our case, to avoid that, we can make the server the only one synchronizer. It can send synch packets from time to time, and in the same way the network was discovered, these packets can be retransmitted between the nodes, so that the synch information arrives to all the network. Thus, in our network the synchronizer of each node will be its father.

Synchronizing is one the most important tasks, since it its necessary that all the nodes have the same schedule in order for the protocol to work efficiently. Besides, synchronizing is one of the most difficult tasks to do, because normally, to do this, the synchronizer sends the relative time where the period ends and, by the time this message arrives to the follower, the clock of the synchronizer has already advanced a little bit, so the position of both clocks are not exactly the same.
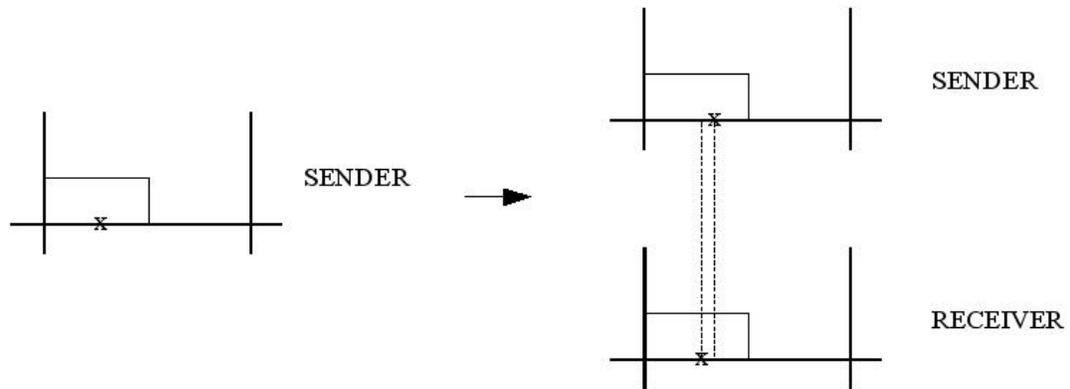

Figure 5.4: On the left, the time position where the sender of the synch is. On the right, the positions of sender and receiver when the synch packet has arrived to the receiver. The time position of the sender has advanced while the transmission, so the update of the clock is not exact

To deal with this problem, usually nodes have to send, taking into account that there is an interval of time, where it is dangerous to send, since the receiver may be still sleeping. Therefore, the nodes should have to wait a small amount of time, if they are sending in the beginning of the active time, to make sure that the receiver will be awake. This also increases a little bit the delay. In our case, however, we can benefit from the fact that data flows always in the same direction to avoid waiting during this interval.

As in our scenario, synchronization is done from the server to the rest of the nodes, all the fathers will have their clocks a little bit advanced with regard to their sons. However, as the transmissions of the orienteering messages go from the sons to the fathers, the sons can send their messages in the beginning of their period, and their fathers will most probably be awake, since they are always advanced a little bit in the time line. Because of this, it is not necessary for us to wait during a safety time if we send in the beginning of the period.

On the contrary of the orienteering messages, synch packets, as they go in the direction father-son, cannot be sent in the beginning of the active time, since most probably the sons will be still sleeping. They should be sent in the end of the active time instead, where there are more chances to find the sons awake.

Therefore, in our implementation, the server will send a synch packet each five periods. Nodes will retransmit the synch packet when they receive it. The orienteering messages will attach also the clock information, in order to avoid sending too much packets. If one node is not synchronized (by synch packets or orienteering packets) after twenty periods, it will remain awake until it receives one.

5.2   Results

Now that we have seen how the program works, in this section, the results of the first tests will be shown.

As we advanced in previous chapters, and as we expected, the messages that have to do several hops to arrive to the server, have more delay than the messages from nodes closer to the server. Sometimes they can last several seconds in arriving to the server, because some packet has been lost, or because of a collision. However, if there is not other data flowing in the network, and the packets arrive properly, the delay of these messages can be very small (less than three seconds).

Besides, if several nodes in range send at the same time the delay of their messages may be longer, since they have to contend and send again. Sometimes they can send at the same time several times, until the contention mechanism chooses the right distribution of the media. However, this is normally solved by the contention mechanism quite fast.

In conclusion, the messages arrive very fast to the server, when there are no more than three hops.

5.3 Improvements and future work

Now that the implementation is finished, it is time to write down some ideas, that have not been implemented, but that could be good in order to improve the current solution.

5.3.1 Allow new nodes to join the network by just listening to the syncks

Instead of having the D messages to discover new nodes, it could be also possible to let undiscovered nodes hear to the synch packets, and select a node as a father if it receives a concrete number of packets properly. This could make the joining process of a node easier and also could help avoiding the problem of adopting nodes with a high error rate. However, it would make the joining process longer and more complex.

5.3.2 Dedicate the sleeping time for background jobs

During the sleeping time, the sensor is doing nothing. Excepting when a participant comes, that the message is stored in the queue even if the node is sleeping, all the work is done in the active time. Even though it cannot send or receive messages during that period of time, its processor is still working. It could be a good idea to dedicate that portion of time to do other work not relationed to sending or receiving messages, like managing the received messages, preparing the next message to send, etc. In conclusion, constructing the program in a way that all the possible work is done in the sleeping time, so that the active time can be reserved just for sending and/or listening. This would increase the number of packets that the sensor is able to send or receive in the active time, and thus would increase the efficiency.

5.3.3 A dynamic contention mechanism

In the proposed solution, nodes always wait during the same random amount of periods whenever a colision takes place. It could be good if this number of periods was smaller or bigger, depending on with how many nodes in the neighbourhood it is possible to have collisions. This would avoid more collisions.

5.3.4 Improve the selection of the father

In our implementation, nodes select the first node in sending the D message as a father. In most cases, this can be acceptable, but sometimes, it may happen that a node receives a D message from a node that it is not the closest to the server, or maybe we can have the situation where too many nodes select the same father, while other nodes with the same distance to the server are not selected by anyone. The discovery algorithm could be more complex, to make the nodes select the best father, or even select several of them and use one or another depending on the situation. The best father would be a node close to the server, and with the less sons as possible, so that it can pay more attention to them. Having the optimum father selection for each topology would increase the efficiency and avoid bottle necks in some situations.

5.3.5 Send several messages at the same time

The more messages sent, the more probabilities of having collisions, and thus the delay and the power consumption increase. It would be better, if nodes could group all the messages that have to send, in only one. Therefore, if for example one node received a message from other node, and in a small amount of time a participant triggered the control point, it could send both messages together, and this multiple message would travel in the same way till the server (or bigger, if other nodes have added more messages to it). This would reduce the traffic considerably, and also the delay, since new messages would not have to wait in the queues of the nodes, they would be able to join the train to the server inmediately. To do this, control packets would be needed.

Chapter 6

Conclusion

In this project we face the challenge of making an ad-hoc network that can be used in the sport of orienteering. The goal is to make the partial times of the participants arrive to a main server, whenever they pass by a control point, so that the information can be shown in a screen to allow the public follow the race. The power consumption and the delay must be reduced as far as possible.

In order to get enough theoretical knowledge, in the first chapters we have start studying the existing MAC protocols, first the classics FDMA, TDMA, CSMA, Aloha and CSMA, and then the ones created for wireless. In these ones we have explored and analyze some of the protocols of both sides, contention-based and TDMA.

After this, in chapter three, we have started the process of thinking about the best solution for orienteering. To do that, first we have described in detail the main features of the scenario of orienteering, and then, with that scenario in mind, we have compared the main wireless MAC protocols: contention-based and TDMA, by applying each one to some specific networks. For our purpose, we have used three small networks of four nodes with different disposition of them. Then we have done the same for bigger networks. We have concluded that TDMA can reach better levels of delay and power consumption than contention-based, even though it is a more complex solution.

Because of this, in chapter four we have start implementing a solution based on contention-based, because it was simpler to implement and it could reach acceptable levels of efficiency and could allow us to obtain quick and interesting results. During all the chapter the characteristics of the algorithm have been shown, and also the results and some improvements have

been commented.

Even though a solution based on TDMA would probably reach better levels of efficiency, we have implemented a simple solution that works in an acceptable way, respecting to power consumption and delay. However, there are some aspects that can be improved, that would increase the efficiency of our goals even more.

Bibliography

[1] A. S. Tanenbaum, *Computer Networks.* Prentice Hall. March 17, 2003

[2] P. Karn, *MACA - A New Channel Access Method for Packet Radio*

[3] V. Bharghavan, *MACAW. A Media Access Protocol for Wireless LAN's*

[4] C.S. Raghavendra and S.Singh, *PAMAS - Power Aware Multi-Access protocol with Signalling for Ad Hoc Networks*

[5] W. Ye, J. Heidemann and D. Estrin, *An Energy-Efficient MAC Protocol for Wireless Sensor Networks*

[6] R. Kalidindi, L. Ray, R. Kannan and S. Iyengar, *Distributed Energy Aware MAC Layer Protocol For Wireless Sensor Networks*

[7] The ScatterWeb project, *www.scatterweb.net*, 02-2005

[8] J. Schiller, A. Liers, H.Ritter, R. Winter and T. Voigt, *ScatterWeb - Low Power Sensor Nodes and Energy Aware Routing*

[9] The Contiki Operating System, *http://www.sics.se/ adam/contiki/*, 02-2005

[10] J. Deng and Z. J. Haas *A New Medium Access Control for Packet Radio Networks*

[11] J. M. Kahn, R. H. Katz and K. S. J. Pister *Emerging Challenges: Mobile Networking for "Smart Dust"*

[12] R. C. Shah and J. M. Rabaey *Energy Aware Routing for Low Energy Ad Hoc Sensor Networks*

[13] A. Bhatnagar and J. Teifel *An Energy-Performance Metric for Mobile Ad Hoc Networks*

[14] H. Lundgren *Implementation and Real-world Evalutation of Routing Protocols for Wireless Ad hoc Networks*

[15] A. Dunkels, L. M. Feeney and B. Grnvall *An integrated approach to developing sensor network*

[16] A. Dunkels, B. Grnvall and T. Voigt *Contiki- a Lightweight and Flexible Operating System for Tiny Networked Sensors*

[17] M. Ringwald and K. Rmer *BitMAC: A Deterministic, Collision-Free, and Robust MAC Protocol for Sensor Networks*

[18] A. Chandra V. Gummalla and J. O. Limb *Wireless Medium Access Control Protocols*

[19] Wikipedia, the free encyclopedia *http://en.wikipedia.org/wiki/Main_Page*