HIDING SWITCHING DELAY AND IMPROVING BROADCAST EFFICIENCY IN MULTICHANNEL MULTI-INTERFACE WIRELESS AD HOC NETWORKS

BY

XAVIER RODOLFO FRANCO

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2011

Urbana, Illinois

Adviser:

Professor Nitin Vaidya

ABSTRACT

Multichannel multiradio wireless networks allow one to utilize the available spectrum more efficiently to improve network performance. When nodes operate with multiple channels, the contention among the nodes operating in a particular channel is reduced, which translates to more opportunities to transmit and consequently improved throughput.

Among the architectures proposed to implement a multichannel network, the hybrid multichannel protocol has been shown to outperform other strategies with respect to throughput. However, there are still some open problems that have not been studied in detail. One of these is how to improve the efficiency of broadcasting, since currently the hybrid multichannel protocol performs broadcasting by flooding on each channel. Another area that has not been studied enough is the benefits of hiding the switching delay.

In this thesis we first propose a smart probabilistic protocol for broadcasting in a multichannel multiradio (MC-MR) wireless network, operating with the hybrid multichannel protocol. We present simulation results that show that our protocol can significantly reduce redundant traffic while keeping the delivery ratio at useful levels. In the second part we study the benefits of hiding the switching delay by using two switchable interfaces with the constraint that only one interface can transmit at a time.

To my wife, for her dedicated support in these two years

ACKNOWLEDGMENTS

I would like to thank my adviser, Dr. Nitin Vaidya, for guiding me through the research and thesis writing process. I would especially like to thank Vijay Raman for always helping me with understanding the hybrid multichannel protocol. I would also like to thank all members of my research group, for their help and support through discussions and suggestions. Last but not least, thanks to my family for supporting me every step of this work.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Despite significant advances in physical layer technologies, today's wireless LAN throughput is considerably less than its wired counterparts. The advertised 72.2 Mbps bandwidth for 802.11 a/g is almost halved when all the overheads (MAC contention, 802.11 headers, 802.11 ACK) are accounted for. The throughput problem is further aggravated for multihop wireless networks due to interference from adjacent hops on the same path as well as from neighboring paths. Wireless technologies, such as 802.11, provide multiple non-overlapping channels, and recent advacements in hardware for wireless devices render the usage of multiple radios affordable. The ability to utilize multiple channels substantially increases the effective bandwidth available to wireless network nodes. Kyasanur [1] showed that the capacity of a multichannel wireless network with n randomly distributed nodes scales linearly with the number of channels when the ratio of the number of channels to the number of interfaces is of the order of $O(\log n)$.

One key point in the derivation of the results found by Kyasanur is that in order to utilize all the channels available, the interfaces have to be switched; however, the switching delay was assumed to be zero, which is not the case in practice. Just to be aware of the degradation that switching delay can cause in the network performance, suppose that the transmission time for each packet is T and that the switching delay is S; thus, in the worst case scenario where a channel switch is required for every packet, the capacity will be only $\frac{T}{T+S}$ of the capacity when there is no switching delay. The problem can be very serious if the packet transmission time is small and the switching delay is significant. Indeed, in practice the switching delay can be in the order of milliseconds. Kyasanur proposed to hide the switching delay by having a few interfaces in each node and schedule transmissions on the interfaces in such a way that while one interface is transmitting, another interface is switching the channel; the requirement for this solution is that the switching delay has to be less than or equal to the interval during which other interfaces are transmitting.

Current experimental architectures that use multiple channels and allow channel switching use only two interfaces per node. In this thesis we focus on MC-MR wireless networks which use the hybrid multichannel protocol. The drawback of this solution is that when a node has to forward multiple flows, it will have to invest considerable time just switching channels. In the first part of this thesis we study the effects of hiding the switching delay in an MC-MR wireless network which operates with the hybrid multichannel protocol. We evaluate the gain offered by the use of a third interface with the constraint that only one interface per node can transmit at a time. We experiment with different types of traffic, TCP and UDP, determining for each the advantage of using a third interface.

Another important question that we address in this work is how to perform *efficient* broadcast in MR-MC wireless networks. Broadcasting is a common service in multihop networks used not only for data dissemination but also for neighbor discovery and route discovery in reactive unicast protocols. The easiest and most common approach to perform broadcasting is to flood the network with the message. However, this generates excessive redundant traffic and interference in the shared medium among neighboring nodes (which is called the broadcast problem [2]). Many broadcasting protocols have been developed for wireless ad hoc networks. Some of them try to ensure 100% reliability; i.e., every node in the network is guaranteed to receive the broadcast message. For other protocols the focus is to achieve a minimum broadcast latency, and others reduce the redundant transmissions. However, most of these protocols assume a single-radio single-channel (SC-SR) model.

The use of multiple channels creates new challenges to design a broadcast protocol. In the case when a single channel is used across the network, and the nodes are equipped with omnidirectional antennas, a transmission can potentially be received by all neighboring nodes that lie within the communication range. This is called the *wireless broadcast advantage*. However, when multiple channels are used, if a node broadcasts a packet on a particular channel, the packet will be received only by those nodes that are within the communication range and are listening on that channel. Since we are considering nodes equipped with multiple radios, it is possible that a node may need to broadcast the same packet on different channels in order to reach all its neighbors. In multichannel networks, the broadcast strategy depends on the channel assignment scheme implemented in the network. There exist three different strategies to exploit the multiple channels available: *static, dynamic* and *hybrid* (see details in Section 3). We consider the hybrid channel assignment strategy, where the problem of having nodes tuned to different channels is solved by allowing some of the radios to switch across all the channels in the network. Ding et al. [3] have shown that the hybrid channel assignment can achieve higher throughput than other channel assignment strategies. In previous implementations of this protocol [1], broadcast is performed by transmitting a copy of the message on all the available channels, which implies that each node has to spend significant time switching channels. Another problem is that this approach is basically a flooding strategy on multiple channels, so the overhead introduced to the network is large and the energy consumption in the nodes is also high when compared to the single channel case. Among the techniques used for SC-SR models, the probabilistic approach has demonstrated several desirable features, such as scalability and ease of implementation. In this work we propose a smart probabilistic approach to improve the efficiency of broadcasting on MC-MR WMNs with hybrid channel assignment.

The rest of this thesis is organized as follows. In Chapter 2 we discuss previous work and present a brief summary of broadcasting techniques in SC-SR WMNs, and we review protocols for the MC-MR model. In Chapter 3 we describe the operation of the hybrid multichannel protocol and routing protocol which we are working with. In Chapter 4 we study the throughput improvement introduced by hiding the switching cost using three interfaces per node. In Chapter 5 we propose a smart probabilistic algorithm to perform broadcast in MC-MR wireless networks and present simulation results of the performance. In Chapter 6 we review the modifications required in *ns-2* to implement MC-MR wireless networks with the hybrid multichannel protocol. Finally, we conclude with Chapter 7.

CHAPTER 2

RELEVANT WORK

Several protocols have been proposed in the literature to exploit the benefits of multiple channels. In this chapter we provide a brief survey of the various work that adresses the use of multiple interfaces per node and we summarize previous work on broadcasting over MC-MR wireless networks.

2.1 Using Multiple Radios on Multichannel Ad Hoc Wireless Networks

In the capacity analysis of MC-MR networks [1], Kyasanur did not consider the impact of switching delay; however, he proposed that this delay can be hidden by using a few interfaces per node in such a way that while one interface is transmitting, another interface is switching the channel. The requirement for this solution is that the switching delay has to be less than or equal to the interval during which other interfaces are transmitting. Shen [4] perfomed experiments with three interfaces per node, setting two of them as switchable. He first found that when the two switchable interfaces are allowed to transmit simultaneously, the number of useful channels is reduced to three. Shen compared the performance of using one switchable and two switchable interfaces over a single flow while varying the number of hops and found that TCP has a significant improvement, but the improvement in UDP is less. The reason is that in TCP a node has to forward data packets and also acknowledgments, and they are likely to be transmitted on different channels; therefore, using two interfaces to transmit on different channels reduces the switching delay. In contrast, in UDP the flow is unidirectional, so if all the packets are being routed in the same chanel, the node does not incur switching delay, and it is the same to have one switchable interface or two. Robinson et al. also conducted experiments in [5] using more than two interfaces and found that even when a third interface is idle, without transmitting or receiving, the radiation leakage affects the performance of active interfaces. However, he also found that the degradation is smaller when each interface is protected with an appropriate shielding.

2.2 Broadcasting over Multichannel Multiradio Wireless Networks

Broadcasting has been broadly studied for at least 30 years. We can sort the techniques found in the literature into four main categories [6], [7]: flooding, probabilistic methods, area-based methods and neighbor-knowledge-based methods. Flooding is the easiest way to solve the problem. In this technique, upon reception of a new broadcast packet, a node simply sends it to all its neighbors. However, this technique causes serious network congestion and collisions, and it wastes nodes' resources. All the other methods try to improve the efficiency of flooding, basically applying different ideas to reduce the number of rebroadcasting nodes. In probability-based methods each node determines a forwarding probability. In its simple form each node is assigned a fixed forwarding probability, but the nodes can also adjust the probability dynamically depending on local information (such as the number of neighbors). In area-based methods, each node estimates its potential contribution to the overall broadcasting; for example, the decision could depend on the distance from the previous hop: if the receiving node is too close, then it could decide not to rebroadcast since the additional area covered will not be significant. Neighbor-knowledge-based methods are more complex; the idea basically is to select a small set of nodes to form a connected dominating set (CDS) to forward the packet. In a connected dominating set, each node is either a member of the set or it is one hop away from some member of the set. It has been shown that building a CDS is NP-hard when global network information is available, and the problem is even more chanllenging when only local information is available. Protocols based on this technique usually provide heuristics for constructing the CDS.

All the techniques described previously assume a SC-SR network. Qadir et al. [8] address the problem of minimum latency broadcasting in MC-MR multirate wireless networks. They consider a network with static channel assignment and design a set of centralized algorithms to achieve minimum latency broadcasting. The protocol basically consists of constructing a broadcast delivery tree and then scheduling transmissions in a centralized manner. The main problem with this solution is its centralized approach. Ahuja and Ramasubramanian [9] consider the problem of broadcasting in an MC-MR network employing directional antennas. They propose a set of heuristic solutions to find a broadcast tree from a given root and then assign channels in such a way that all the links in the broadcast tree can be active simultaneously without interfering with each other. Jiang and Xie [10] consider the minimum energy broadcast problem with directional multibeam antennas (MEB-MB). Their solution exploits the Broadcast Incremental Power (BIP) algorithm, which uses the wireless broadcast advantage property to choose the transmitting nodes and the power they have to use. The target is to minimize the power comsumption while covering all the nodes in the network. Li et al. [11] consider a multichannel multiradio wireless network with static channel assignment and reduce the broadcast problem to the minimal strong connected dominating set problem of the *interface-extended* graph. This is a self-pruning protocol, which collects information about network topology using hello messages. Song et al. [12] consider the problem of broadcasting jointly with channel assignment, and construct a tree using channels with the minimum possible interference. They propose a link quality metric to select the links that will be part of the tree.

All these solutions consider a multichannel network where the nodes have their radios tuned to the same channel all the time. However, to the best of our knowledge, there is no approach proposed for the case when the radios can switch across channels dynamically, specifically, when the channel assignment used is hybrid. In [13], Kyasanur et al. resolve the broadcast problem in this kind of network simply by flooding the packet on all the channels available; however, as we explained for the single channel case, this is very inefficient, and the problem is worse if multiple channels are transmitting the same packet. Our work focuses on MC-MR ad hoc wireless networks which employ the hybrid channel assignment strategy.

CHAPTER 3

BACKGROUND

All the work in this thesis considers an MC-MR wireless network, which operates with the hybrid multichannel protocol. For that reason, we dedicate this chapter to an overview of the functioning of the network we are dealing with. We want to highlight that this network has been evaluated experimentally in the Net-X testbed project [1]. This chapter mainly describes the hybrid multichannel protocol operation and the routing protocol, which are important to understand the next chapters.

3.1 Hybrid Multichannel Protocol Operation

The first thing that we need to think about when dealing with multichannel wireless networks is how to ensure that nodes that are operating on different channels can communicate with each other. Different channel assignment strategies have been proposed to solve this problem. In *static channel assignment*, each node is assigned a set of channels, and its radios are always tuned to the same channels. In *dynamic channel assignment*, the radios are allowed to switch channels over time. In the *hybrid channel* approach, a subset of the radios are fixed for long intervals and the others can be switched across the available channels frequently. We will focus only on the last approach since it is used in the architecture we are considering. For simplicity in the explanation, we will consider only two radios per node. In this strategy, connectivity is ensured between nodes by the radio that can switch across channels; we call this radio the *switchable* interface and the radio that operates on a fixed channel the *fixed* interface. Roughly speaking, reception of packets is only possible on the fixed interface, but packet transmission is possible on both interfaces, fixed or switchable; the interface to use depends on the channel of the neighbor node which the node is transmitting to. If the neighbor node is operating on the same channel as the current node, then the trannsmission can be through the fixed interface; otherwise, the switchable interface has to switch to the appropriate channel to transmit. Therefore, any node can potentially transmit and receive simulatenously if the receiving channel is different from the transmitting channel. However, in order to do this, every node has to be aware of the fixed channel on which its neighbor nodes are operating; that is, two neighboring nodes cannot communicate with each other if one of them is not aware of the channel of the other. This awareness requires a neighbor discovery mechanism, which is implemented with hello messages that contain the channel information. A copy of the hello message is broadcast two hops away on all the channels available with the intention that all the neighboring nodes recieve it. Another important point to consider in the operation of this protocol is that the switching delay is non-negligible. In practical experiments, it is around 5 ms [14]. For that reason, once a node switches to a channel, it stays on that channel a predetermined time before switching to the next channel. Also, there exists a limit on the amount of time that a node is allowed to stay on a particular channel if it has packets to transmit on other channels. In order to handle the multiple channels, each interface maintains packet queues for each channel; thus, the restricition on the time that an interface is allowed to spend on a channel has the goal of providing "fair" scheduling for the queues.

Since the channel of the switchable interfaces is assigned depending on the neighbor to which the node is transmitting, we note that this protocol only requires assigning channels to the fixed interface. This is implemented with the information provided by the hello messages received from neighbors. At the beginning, all the fixed interfaces randomly choose a channel. The idea is that each node uses hello messages received to maintain a *neighbor* *table* containing the fixed channels being used by its two-hop neighbors. Hello messages are sent periodically, and the idea behind assigning the channels is that every time a node has to send a hello packet, it switches its fixed channel to the least used channel among its two-hop neighbors. However, this is done only with probability 0.5 [14], since otherwise we could easily have a situation where the fixed channels are switched constantly. Figure 3.1 shows an example of the operation of the protocol. We can see that the relay node B can receive and transmit simultaneously.

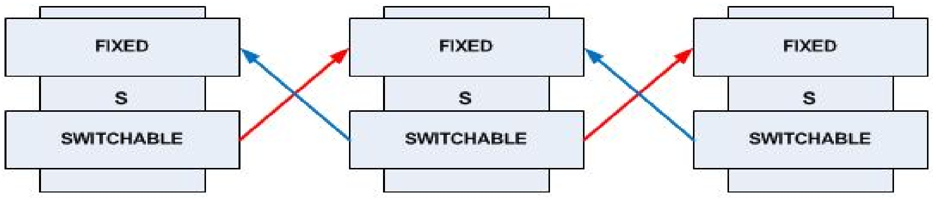

Figure 3.1: Hybrid multichannel protocol operation

3.2 Multichannel Routing Protocol

The routing mechanism we will consider is basically based on AODV routing protocol [15], modified for multichannel operation. When dealing with multiple channels, we need a routing protocol that considers new metrics to evaluate the route cost. For that reason the protocol we consider incorporates a mechanism for finding a channel diverse route, avoiding bottlenecks, and reducing the overall expected transmission time in addition to reducing the number of hops. Also this protocol tries to avoid those routes that may require frequent channel switching. In order to find a channel diverse route, it is necessary to make sure that a flow experiences lower intra-flow interference, and this is achieved by selecting routes such that adjacent hops operate on different channels as much as possible. The hybrid channel protocol requires the swicthable interface to switch channels to transmit to different neighbors, which implies a switching delay; therefore, it is preferable to avoid routing multiple flows through a single node, as this may result in frequent channel switching. All these requirements are incorporated in the form of a metric called MCR [13]. This metric uses information about channel usage among neighbors to estimate the cost of switching to a particular channel. Also, the cost of a link is estimated with the *expected transmission time* (*ETT*) [16] metric on every channel. The MCR metric basically couples the switching cost with the ETT metric and sums them up over the entire path. For instance, if ETT_i is the estimated transmission time in the i^{th} hop and $SC(c_i)$ is the channel switching cost of channel c_i used in the i^{th} hop of transmission, then the MCR metric is given by

$$MCR = (1-\beta) * \sum_{i=1}^{h} (ETT_i + SC(c_i)) + \beta * max_{1 \leq j \leq c} X_j \tag{3.1}$$

where β is a weight between 0 and 1 (in practical experiments β has been set to 0.4), h is the number of hops on the path, and c is the number of channels available. X_j is the total ETT cost on channel j and is given by

$$X_j = \sum_{\forall i : c_i = j} ETT_i \tag{3.2}$$

The ETT of a link is calculated using the expected number of retransmissions ETX on the link, and it is given by $ETT = ETX * \frac{S}{B}$, where S is the average packet size and B is the data rate of the link. ETX is estimated based on the packet loss probability of the link, which is calculated using the hello packets.

In addition to the new metric, the routing protocol also considers other modifications. Since the nodes have two interfaces and multiple channels to use, the routing entry has to indicate which interface and channel to use to forward the packet. This protocol does not include optimizations like route caching, which is available in the original AODV, because channel allocations and the corresponding costs may change frequently, and the actual change can only be estimated at the destination.

CHAPTER 4

HIDING SWITCHING DELAY WITH THREE INTERFACES

It has been shown [1] that the use of two interfaces to handle multiple channels provides significant improvement in throughput. Shen [4] performed extensive experiments using three interfaces per node on the Net-X testbed. He found that, if three interfaces are used simultaneously, the actual number of channels that can be used in the network is reduced to three; otherwise, the channel interference degrades the throughput considerably. In this study, Shen used two switchable interfaces and one fixed. He performed experiments on linear topologies with different numbers of hops. He found that, even with the reduced number of channels, using a third interface offers improvement in throughput. The improvement is significant when the traffic is TCP, since there is a need to route packets in forward and reverse directions because of the acknowledgments.

The time spent by a node switching channels can represent a significant fraction of the total operation time. For instance, consider a switching delay of 5 ms and a node forwarding packets on three different channels; in order to serve the three queues, the node will spend 15 ms switching channels. Now, if the packet transmission time is just a fraction of a milisecond, then, during the time spent on switching channels, a node could easily transmit a few tens of packets.

Kyasanur proposed the idea of hiding the switching delay by using multiple interfaces. This idea works if, for a particular interface, the time between succesive turns to transmit is longer than the switching delay; so, while one interface is switching its channel, another one could be transmitting. In this chapter we adopt this idea using three interfaces, setting one as fixed and the others as switchable. Only one switchable interface can transmit at a time. We do not allow simultaneous transmissions on the switchable interfaces of the same node. The goal is to avoid the reduction, found by Shen, in the actual number of channels that can be used when three interfaces are available in a node. We conduct several experiments on *ns-2* to study the benefits of hiding the switching delay in the hybrid multichannel protocol.

4.1 Changes Required to Use Two Switchable Interfaces

Protocols for MR-MC wireless networks with hybrid channel assignment that consider two radios per node may require some modifications to be used in scenarios where there are more than two interfaces. For instance, consider the hybrid multichannel protocol described in Chapter 3. In this protocol, when a node needs to transmit a packet on a specific channel, it uses the switchable interface if the fixed interface is not tuned to the required channel. However, if the node has more than one switchable radio, the protocol should be able to decide which switchable interface has to be used to transmit the packet. This example shows us that, when dealing with more than one switchable interface, we need to specify new rules about how to use the interfaces.

When the node has two switchable interfaces we need to define which channels are supported on each interface. We could divide the set of channels available, such that each switchable interface transmits only on a specific subset of channels. However, this can potentially affect the desired performance. It is possible that all the packets that a node has to forward use channels in the same subset. To avoid this, we have decided that any interface can use any channel available in the network. For this, we need to modify slightly the structure of the routing table. In the routing protocol that we are considering, each table entry consists of three fields: destination IP address, interface and channel. We propose that each entry consists only of the pair destination IP address and channel. The interface can be selected dynamically trying to balance the load on the switchable interfaces. We propose algorithm 1 to select the interface. Broadcast packets are treated in the same way.

Notation:

$C_{I(j)}$: Channel used by interface j.

$I(j).queueLength(c)$: Length of channel queue c on interface j.

Algorithm 1 Returns interface to transmit packet on channel c

```
1: if ($C_{I(fixed)} = c$) then
2:     Return Fixed
3: end if
4: for $j = \{Switchable_1, Switchable_2\}$ do
5:     if ($I(j).queueLength(c) > 0$) then
6:         Return j
7:     end if
8:     if ($C_{I(j)} = c$) then
9:         Return j
10:    end if
11: end for
12: Return interface with less non-empty queues
```

We describe the way we hide the switching latency with an example. Let us consider node A. Assume first that node A has to forward packets on two channels: c_1 and c_2. According to algorithm 1, node A will use one switchable interface, S_1, to serve queue c_1 and the other switchable interface, S_2, to serve queue c_2. Therefore, S_1 switches to c_1 and S_2 switches to c_2. Suppose that A selects S_1 to transmit first. While S_1 is transmitting, S_2 remains idle. In order to hide the switching latency, once S_1 completes its transmissions, S_2 inmediately starts serving pending packets on queue c_2. Next, when S_2 completes its scheduled transmissions, S_1 can transmit again on channel c_1. We notice that using the two switchable interfaces in this way, node A may transmit continuously.

Now, let us assume that node A has to forward packets on four different channels: c_1, c_2, c_3 and c_4. According to algorithm 1, node A will assign packets corresponding to two queues to S_1. The other two queues will be served by S_2. Let us assume that S_1 serves queues c_1 and c_2, and that S_2 serves queues c_3 and c_4. Suppose that S_1 will serve first c_1. Similarly, S_2 will serve first c_3. So, S_1 and S_2 will switch to c_1 and c_3, respectively. Suppose that node A selects S_1 to transmit first. At the beginning, S_1 will serve queue c_1; once S_1 completes its transmissions, S_2 will inmediately serve queue c_3. While S_2 is transmitting packets, S_1 will switch its channel to c_2, such that once S_2 is done with its transmissions, S_1 inmediately starts serving c_2. Similarly, while S_1 is transmitting, S_2 will switch its channel to c_4. Once S_1 completes its transmissions, S_2 may transmit inmediately. Notice that now node A is transmiting packets on four different channels; however, node A may transmit packets without interruptions. So, by hiding the switching latency, we mean that the node is able to transmit packets continuously, even if the packets must be transmitted on different channels. We also notice in this example that we use a round robin scheduling mechanism for the interfaces and also for the channel queues corresponding to each interface.

4.2 Experiments with Single-Hop Flows

In this section we conduct a simple experiment with the single-hop network shown in Figure 4.1. We have a single source of traffic and multiple destinations. The destinations are listening on different fixed channels. We study the difference in the throughput using two and three radios. We vary the number of destinations, which implies that we vary the number of channel switchings that the source node has to perform. The goal of the experiment is to study how a single node is affected when we increase the number of channel switchings.

We use a CBR traffic generator at node S with UDP protocol. The packet generation rate is 2 Mbps and the packet size is 512 bytes. The channel data rate is 11 Mbps. Figure 4.2 shows the throughput while we increase the number of flows. First, we can see that when only one flow is active, the throughput is practically the same in both cases. We could expect a little improvement in this case, but it is strictly related to the way that hello packets are

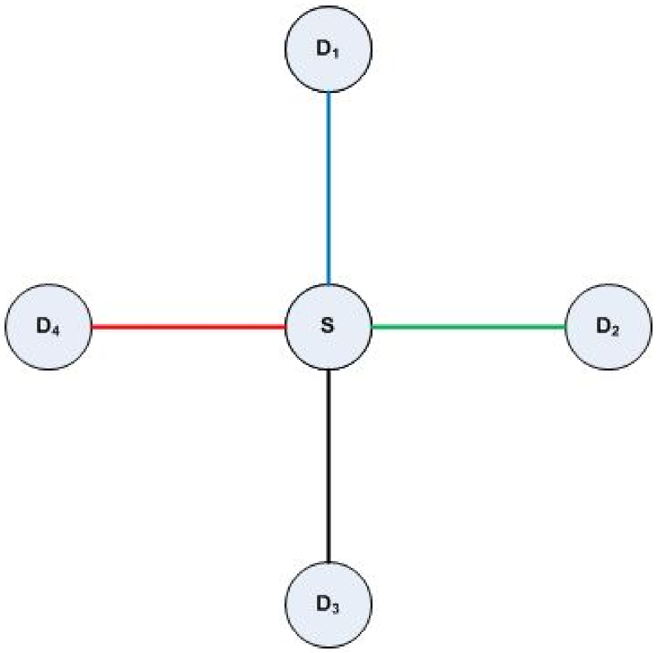

Figure 4.1: Single-hop network

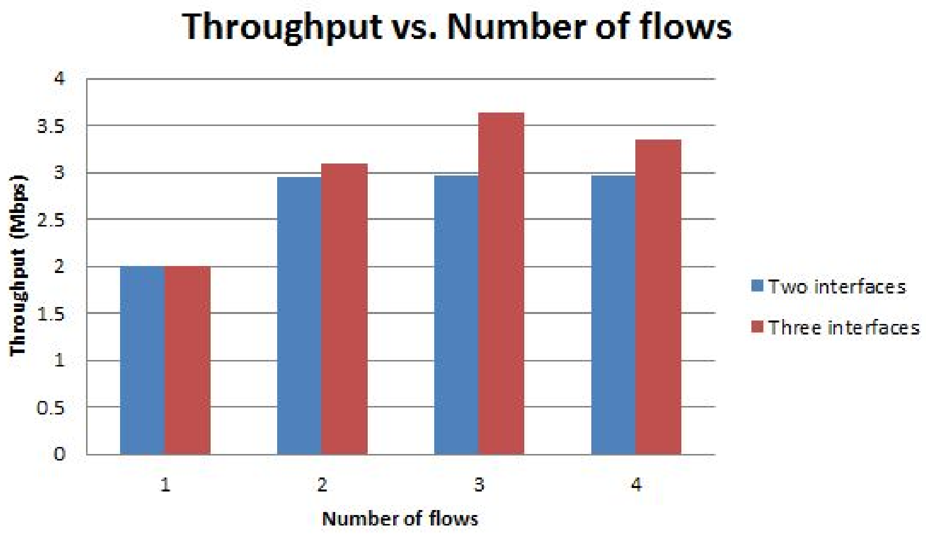

Figure 4.2: Throughput vs. number of flows

transmitted. Remember that when three interfaces are available we try to balance the load on the interfaces. Next, when we have two active flows we see an improvement of around 4%, while when the source is required to transmit on four different channels the improvement is around 13%. Of course, these results are expected since with three interfaces we are hiding the switching delay. Notice that when two interfaces are used, the maximum achievable throughput is around 3 Mbps, which is achieved with two flows. This upper bound is determined by the two factors: first, the packet generation rate, and second, the switching delay. We can see that even when we increase the number of flows, the aggregate throughput cannot be increased by more than 3 Mbps. In contrast, when we use three interfaces we can see that it is possible to achieve a throughput larger than 3 Mbps. Also, we can see that the improvement for the case when the source transmits on three channels is larger than the case when the transmissions occur on four channels. The reason for this is that, when the node has to serve three queues, two of them will be served by one interface and one by the other. Let us say that interface 2 is serving one queue and interface 3 is serving two queues; therefore, according to the round robin scheduling used by the hybrid multichannel protocol, those flows served by interface 3 will have roughly half as many opportunities to forward packets as the flows served by interface 2. This results in a high throughput for the last flow. Basically the large improvement in the aggregate throughput is due to the high throughput achieved by the flow which does not share interface. In contrast, when we have a single switchable interface, all the flows share the interface.

4.3 Experiments with Two-Hop Flows

In this section we conduct experiments with simple two-hop networks with up to five nodes. The three scenarios tested are illustrated in Figures 4.3-4.5. We use different fixed channels for all the nodes. Our goal is to evaluate the impact of a third interface in typical scenarios found in real networks. We evaluate throughput for TCP and UDP traffic. We expect to see improvement in throughput only when the nodes have to perform channel switching. Additionally, from the results found in the previous section we expect that the more channel switching required, the more the improvement.

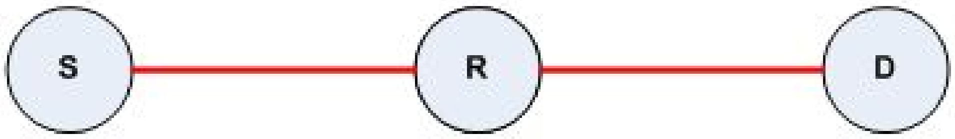

Figure 4.3: Scenario 1: two-hop network with single source and single destination

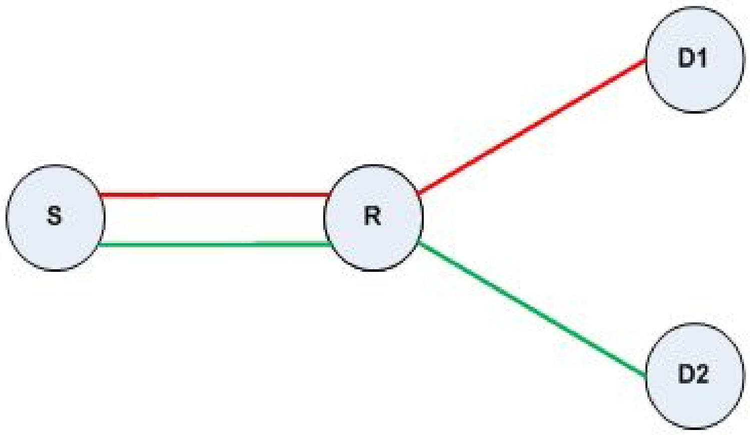

Figure 4.4: Scenario 2: two-hop network with single source and two destinations

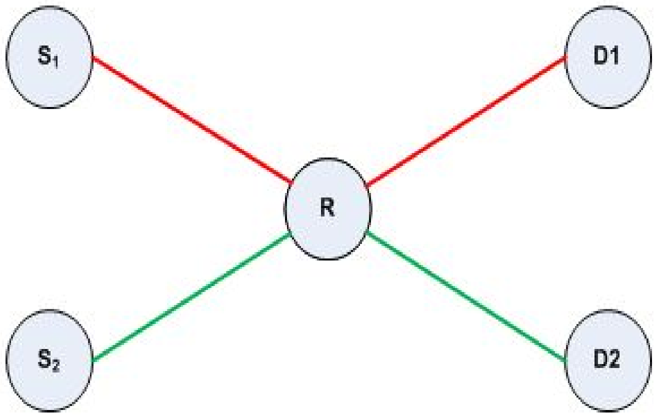

Figure 4.5: Scenario 3: two-hop network with two sources and two destinations

Scenario 1, shown in Figure 4.3, consists of a simple flow with one relay node. To evaluate performance with TCP, we set up an FTP connection between the source and the destination, and run the simulation for 60 seconds. As we can see in Figure 4.6, the use of a third interface improves the throughput by about 5%. The improvement in this case is introduced by the relay node, which has to forward data packets and acknowledgments. Given the channel assignment strategy used by the multichannel protocol, the relay node has to use different channels to transmit acknowledgments and data packets, so it has to perform channel switching whose latency is hidden when an additional switchable interface is available.

For the same network, we set up a CBR flow of 2 Mbps between source and destination using the UDP protocol. In Figure 4.7 we can see that for scenario 1 the throughput is practically the same; there is only a slight improvement. This is due to the fact that with UDP, packets are forwarded only in one direction; there are no acknowledgments, so the relay node does not need to perform channel switchings for transmit data packets. The small improvement is due to the load balancing strategy used to transmit hello messages.

Now, we add one more destination for scenario 2, as shown in Figure 4.4. We have two flows from the same source to two different destinations. When both connections are TCP, the relay node will have to transmit on three different channels: two for data packets and one for acknowledgments. In Figure 4.6 we can see that for this scenario the improvement in the throughput is greater than for scenario 1, around 13%, mainly because the saving in switching latency is larger, since now we have to perform two channel switchings instead of one. On the other hand, when the flow is UDP, we see in scenario 2 in Figure 4.7 an improvement in the throughput of around 7%, which is smaller than in the TCP case. This is expected since for UDP the relay node has to transmit only on two different channels, so just one channel switching is required, and thus the saving in latency is smaller.

Finally, we experiment with scenario 3, in which two flows with different sources and destinations use the same relay node, as shown in Figure 4.5. When we set up the two flows

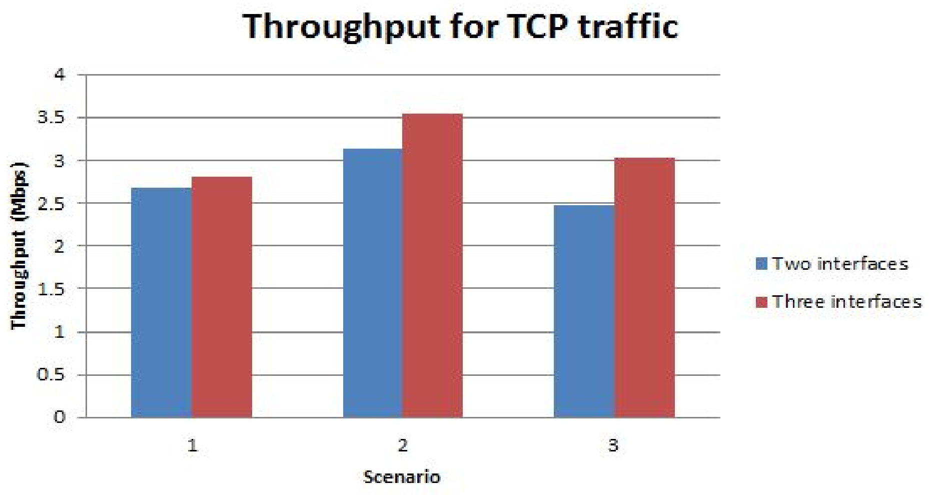

Figure 4.6: Throughput for the different scenarios, TCP traffic

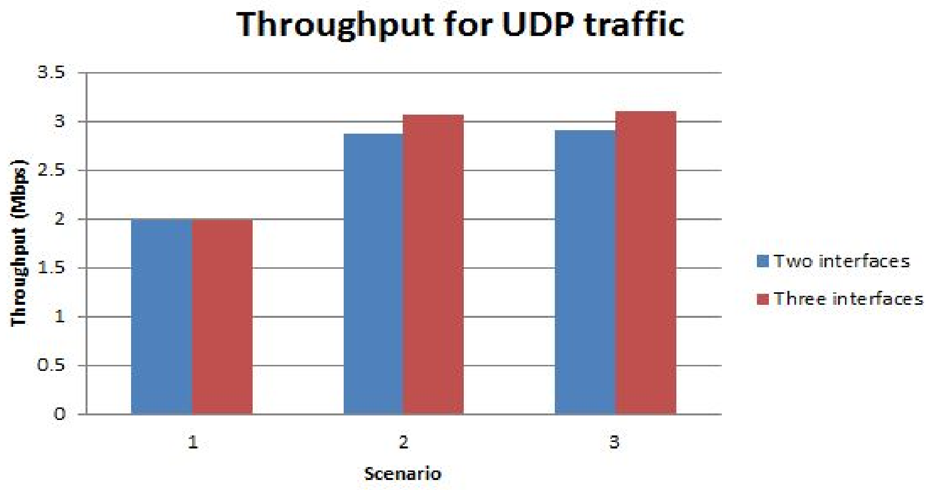

Figure 4.7: Throughput for the different scenarios, UDP traffic

as TCP, the link layer on the relay node has to serve four different channel queues, so this scenario requires more channel switching than scenario 2. We can see in Figure 4.6 that the improvement in the throughput is larger than in scenarios 1 and 2, around 20%. The reason for this is the larger saving in the switching latency by having more switching involved. On the other hand, in Figure 4.7 we can see that if both flows are UDP, the improvement in the throughput is practically the same as in scenario 2, around 7%. This is expected, since the flows' having different sources does not affect the number of channels that the relay node uses for UDP traffic.

4.4 Experiments with Multiple-Hop Flows

Experiments in previous sections have shown the benefits of hiding the channel switching latency in simple scenarios. In this section we study the impact of the third interface on a larger network, with multiple flows and different numbers of hops. The fixed channels are assigned such that for each node, all the one-hop neighbors use different channels. We set up flows with different lengths in the network shown in Figure 4.8. We have defined the following order to activate the flows: 0-4, 7-10, 12-8 and 6-11. When we say that only one flow is active in the network, that means that node 0 is sending packets to node 4. If we say that two flows are active, those flows are 0-4 and 7-10, and so on. We study the aggregate throughput in the network and the throughput of flow 0-4. Notice that the flows have been chosen in such a way that some nodes are obligated to perform channel switching. We study the throughput when the nodes operate with one and two switchable interfaces and obtain results for TCP and UDP traffic.

We will start by analyzing the results for TCP traffic. Figure 4.9 shows the throughput in Mbps as a function of the number of flows active in the network. Notice that in this case, even when there is a single active flow, the use of a third interface introduces an improvement of around 10%, which is due to the fact that all intermediate nodes forward data packets and acknowledgments, so they are obligated to perform channel switching. We can see that while we increase the number of active flows, both the aggregate throughput and the improvement from hiding the switching delay also increase. This is expected since the more flows that are active, the more channel switchings involved, which implies that a larger switching latency can be saved. We see in Figure 4.9 that when the four flows are active, the use of a third

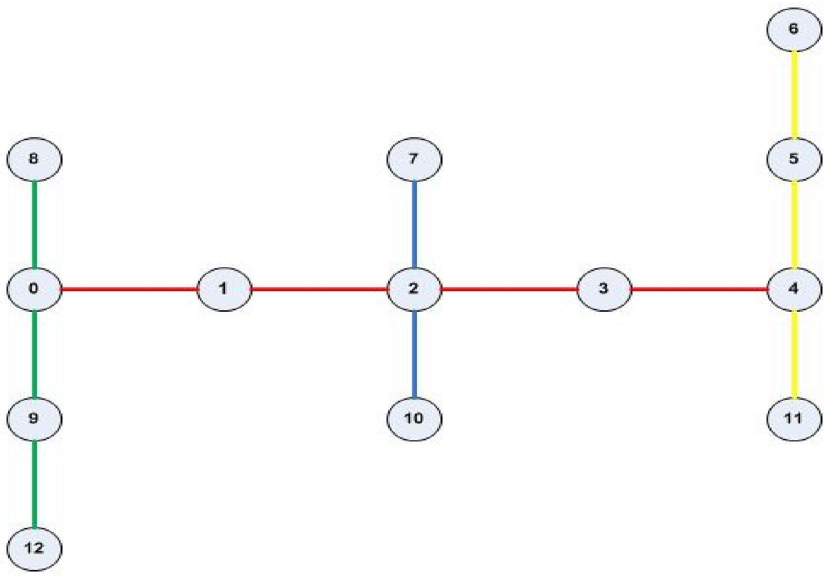

Figure 4.8: Network to study the performance of flows with multiple hops

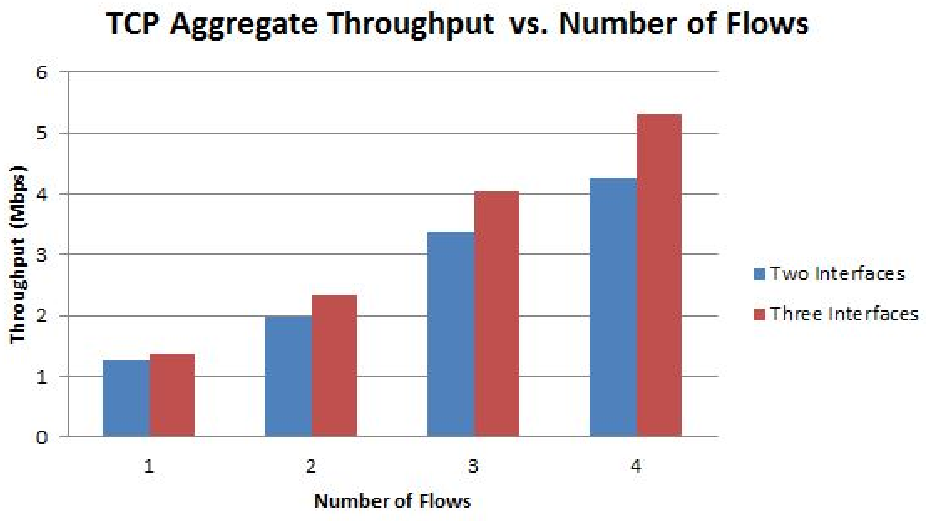

Figure 4.9: Aggregate throughput for TCP traffic vs. number of active flows

interface creates an improvement of around 25% in the aggregate throughput.

Now, we analyze the impact of hiding the switching delay on a particular flow. We plot the throughput for the flow 0-4 as we increase the number of active flows. Notice that all the other flows affect the chosen one, so by increasing the number of flows we are increasing the number of channel switchings performed on flow 0-4. Figure 4.10 shows the throughput as we increase the number of active flows. We notice that in all the cases, the throughput is larger when we hide the switching delay. In fact when all four flows are active, the improvement in the throughput of flow 0-4 is around 40%. Remember that in this case, three of the five nodes in the route 0-4 will have to forward packets on at least three different channels (considering data packets and acknowledgments); therefore, we are saving at least 10 ms each time one of those nodes serves three different channel queues. Another important fact to notice is that when we activate the flow 7-10, the throughput is almost halved; however, when we activate the other flows, the difference is not so large. This is mainly due to the fact that once flow 7-10 is activated, node 2 becomes a bottleneck; it has to forward packets on four different channels (two for data packets and two for TCP acknowledgments), increasing considerably the RTT and consequently degrading the throughput. The activation of other flows does not reduce the throughput by the same magnitude, since now the TCP source has reduced the number of packets sent to the destination.

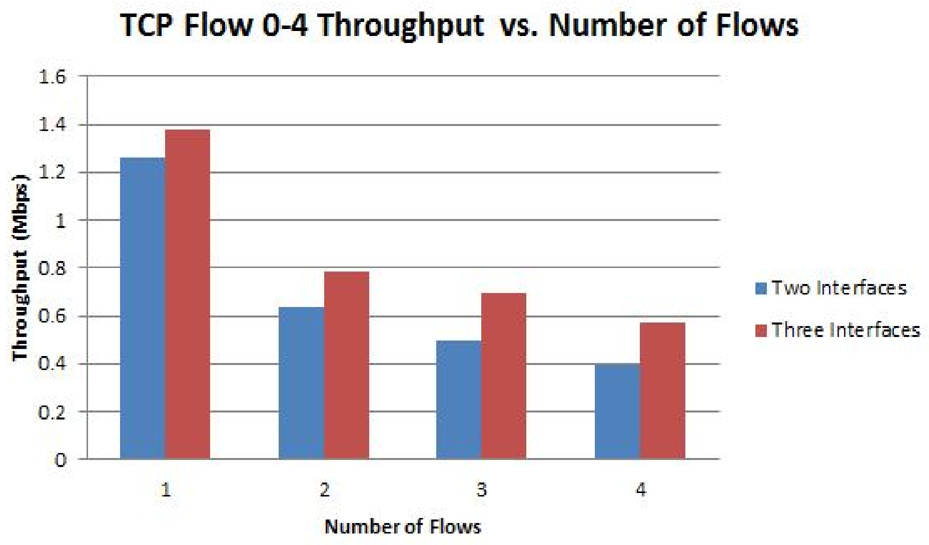

Figure 4.10: Throughput of the flow 0-4 for TCP traffic vs. number of active flows

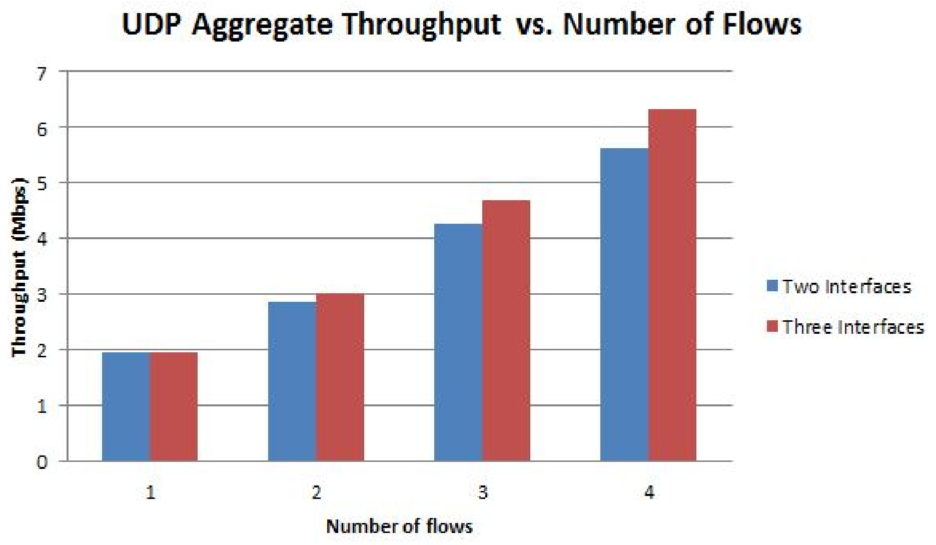

Figure 4.11: Aggregate throughput for UDP traffic vs. number of active flows

We have also simulated the same scenarios for a CBR traffic generator using the UDP protocol. In Figure 4.11 we plot the aggregate network throughput while we increase the number of active flows. One difference from the TCP case is that the improvement is smaller for UDP traffic. This is explained by the fact that UDP does not require acknowledgments, so fewer transmissions and consequently less channel switching are involved. We can see that when all four flows are active the difference is about 12%.

Figure 4.12 shows the throughput for the flow 0-4. We can see that, contrary to TCP when the flow 0-4 is the only active flow in the network, there is no improvement by using the third interface since there is no channel switching required. The difference increases while we activate more flows. When the four flows are active the difference is about 25%, which is smaller than the 40% obtained in the TCP case.

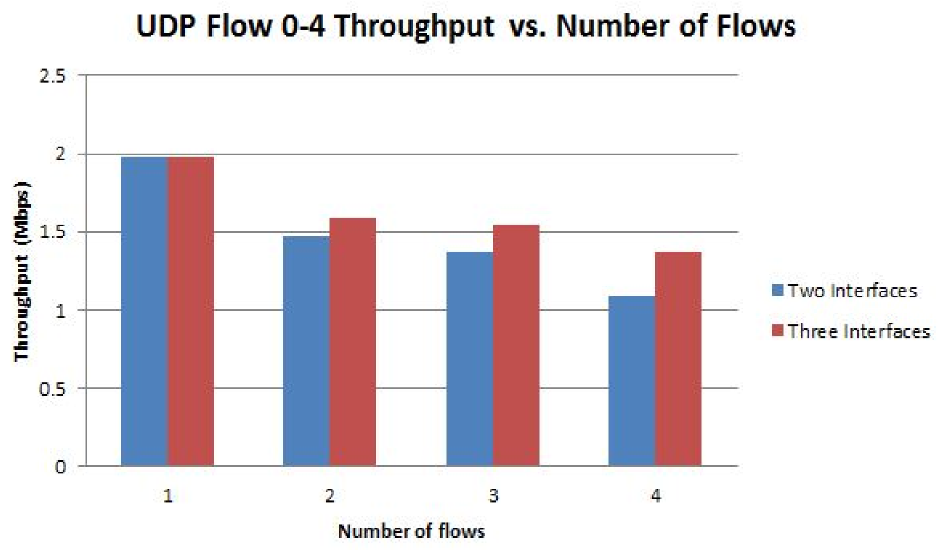

Figure 4.12: Throughput of the flow 0-4 for UDP traffic vs. number of active flows

4.5 Summary

We have performed a series of experiments to show the benefits of hiding the switching latency. Considering a latency of 5 ms, we have obtained different levels of improvement for TCP and UDP traffic. For instance, for the network shown in Figure 4.8, the improvement in aggregate throughput was about 25% for TCP traffic, when four flows are active in the network; however, for UDP traffic, the improvement is only 12%. We found that the improvement is larger if more channel switching is involved. A TCP connection requires that nodes forward packets in both directions (data packets and acknowledgments). So, given the mechanism used by the hybrid multichannel protocol to select fixed channels, it is likely that each relay node has to transmit data packets and acknowledgments on different channels. Therefore, TCP requires more channel switching. Consequently, the benefit of hiding the switching latency is larger for TCP than UDP.

CHAPTER 5

IMPROVING BROADCAST EFFICIENCY IN MC-MR WIRELESS NETWORKS

In this chapter we will describe the algorithm proposed to perform broadcasting over MC-MR ad hoc wireless networks. First, we will describe the network model, then we formulate the problem and present the proposed solution and performance results.

5.1 Network Model

We consider an MC-MR multihop wireless network. All nodes communicate with one another using the IEEE 802.11 MAC protocol. It is assumed that there is a set C of $|C|$ non-overlapping orthogonal channels in the system and each node v is equipped with two interfaces. A hybrid channel assignment scheme is implemented in the network.

We use an undirected graph $G = (V, E)$ to model the MC-MR network topology. V is the set of vertices and E is the set of edges. A vertex in V corresponds to a wireless node in the network. An edge $e = (u, v, c)$ corresponding to a communication link between nodes u and v on channel c is in the set E if $d(u, v) \leq R$, where $d(u, v)$ is the Euclidean distance between u and v, and R is the transmission range. We assume that all the nodes in the network have equal transmission range and equal carrier sensing range. Remember that given the hybrid channel assignment assumed, a node v knows all the fixed channels of all nodes w such that $d(v, w) \leq R$; therefore, it is not necessary that nodes v and u use the same fixed channel to communicate.

5.2 Problem Statement

In its simplest form the broadcast problem consists of a source node sending a message to all the nodes in the network. Usually the main goal is to achieve perfect reachability, i.e., all the nodes in the network receive the same copy of the message. This can be easily achieved by flooding the network with the message, but as we already described, flooding is inefficient. In this work, our problem consists in performing broadcasting in the network model described above and reducing redundant transmissions while keeping high levels of reachability. We assume that there are no adversaries in the network.

Given an MC-MR wireless network with the model assumed, we define $FC(v)$ as the forwarding channels that node v uses to relay broadcast packets, $FC(v) \subseteq C$. We also define $FN = \{v \mid v \in V, FC(v) \neq \emptyset\}$ to denote the set of forwarding nodes. Note that the problem of achieving efficient broadcasting (with respect to redundant transmissions) can be stated as a minimization problem with the objective function f:

$$f = \sum_{v \in FN} |FC(v)| \tag{5.1}$$

The constraint is that all $v \in V$ have to receive a copy of the broadcast message. Previous studies have shown that it is NP-hard to find a deterministic solution for this problem [11]. The optimal solution would be to construct a minimal connected dominating set (MCDS), but this requires global network information. The goal of broadcasting schemes is to reduce overhead in the network, so we cannot accept the luxury of the overhead introduced by learning global network information. Of course, the problem is more challenging in the absence of global network information. Heuristic methods are normally used to balance the cost of collecting network information and the size of the forwarding node set. However, most of the protocols based on these methods present scalability problems, since usually they build either an MCDS or a broadcast tree for each broadcast source. In this work our goal is not to find the global minimum of the function in (5.1), but rather to design a scalable strategy able to reduce the value of f when compared to flooding while achieving an acceptable delivery ratio (i.e., above 0.9). Implicitly, the objective function that we are considering affects the brodacast latency. Reducing the number of retransmitting nodes and forwarding channels, we also reduce redundant transmissions in the network. This reduces the contention and collisions per channel, and consequently the latency will be smaller.

5.3 Proposed Scheme

In the previous section we saw that finding a perfect broadcast strategy (perfect reachability, minimum latency and minimum number of retransmissions) is NP-hard. Approximation algorithms have been proposed, but many of them are not suitable for real networks because of scalability issues, overhead introduced to the network or impractical assumptions. We are looking for a broadcast protocol with the following characteristics:

- Small overhead introduced to the network
- Low computational complexity when compared to protocols that construct trees or CDS
- Scalability with network size
- Acceptable delivery ratio

Some services that rely on broadcasting, such as route discovery, do not need a perfect delivery ratio. This give us some advantages because we can use a scheme that does not necessarily achieve perfect reachability, but decreases the number of retransmissions. In this work we have chosen the probabilistic approach. This technique has been widely studied for the case when a single channel is available in the network [7]. However, to the best of our knowledge, there is not a probabilistic approach to reduce the impact of the broadcast storm problem in MC-MR networks with hybrid channel assignment. Our goal is to reduce the number of forwarding channels for each node. Of course, with a probabilistic approach we will not have a deterministic method to minimize this, but we will see, using simulations, that a probabilistic strategy can reduce significantly the overhead of flooding, while providing enough reachability for basic services like route discovery.

5.3.1 Experimenting with the Rebroadcasting Probability

In this section we experiment with a simple probabilistic broadcast protocol. This protocol uses information stored in the *neighbor table*; it reduces the set of channels to those used by one-hop neighbors. It takes as an input a fixed broadcast probability, then each node, upon receiving a packet for the first time, rebroadcasts the packet with this probability on each channel. If $C_{N(v)}$ represents the set of channels used by one-hop neighbors of node v, then v will rebroadcast the packet on channel $c \in C_{N(v)}$ with a fixed probability. In the context of the problem statement, this solution basically uses a fixed probability to add channels to the set $FC(v)$.

We first analyze the performance of broadcasting data packets across the network. We will assume that all the nodes have up-to-date neighbor information (later we will propose a simple mechanism to reduce the overhead of the neighbor discovery process). We consider a random network of 30 nodes in a 1000 m x 1000 m area. There are five channels in the network, with a data rate of 11 Mbps. The simulations consist of performing a route discovery for each connection; then we plot two metrics, delivery ratio and number of retransmissions. Each data point is the average of 50 simulation runs. For clarity of the plots, we show curves only for a rebroadcast probability of 0.2, 0.4, 0.6 and 0.8. The metrics considered are defined by:

Delivery ratio: Ratio of nodes that receive the broadcast message to the total number of nodes that are reachable, directly or indirectly, from the source host.

Number of Retransmissions: $\sum_{v \in FN} |FC(v)|$, where $FC(v)$ is the set of forwarding channels of node v and FN is the set of forwarding nodes.

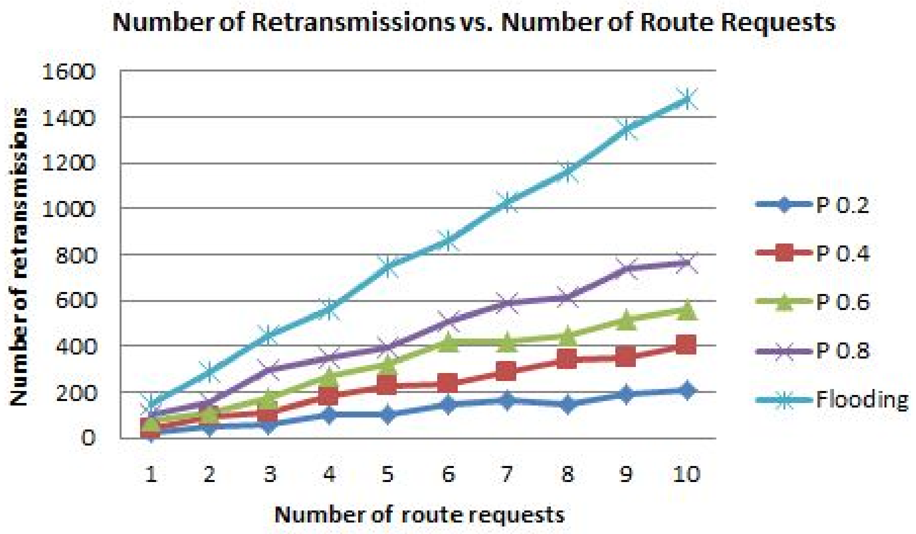

Figure 5.1: Number of retransmissions vs. number of connections

In Figure 5.1 we plot the number of retransmissions as a function of the number of packets being broadcast. We can see that, even when we use a high rebroadcast probability, the saving in rebroadcast transmissions is considerable, around 40% when the rebroadcast probability is 0.8. The improvement comes from the fact that we first reduce the set of tentative forwarding channels to those used by one-hop neighbors and then select the channels with the probability specified. Also, in Figure 5.2 we see that the delivery ratio is not affected too much when the probability is 0.8; we can see in the plot that it is around 94%. Notice that even for flooding, the delivery ratio is not always perfect since the overhead introduced is so high and collisions prevent the broadcast message from being received by all the nodes.

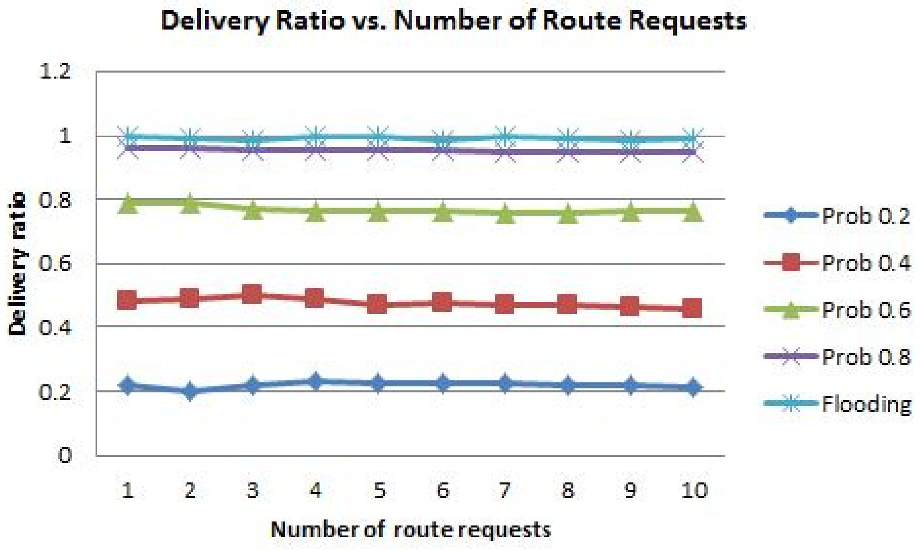

Figure 5.2: Delivery ratio vs. number of connections

5.3.2 Reducing Overhead in the Neighbor Discovery Mechanism

Now we show how to improve the process of neighbor discovery by using a probabilistic approach. First, we notice that we cannot use directly the previous idea, since we cannot rely on the *neighbor table*. It is the task of the neighbor discovery mechanism to build the table. We propose algorithm 2 to reduce the overhead introduced by hello messages:

Algorithm 2 Broadcast Hello Messages: $\forall v \in V$

```
1: $FC(v) = \emptyset$
2: repeat
3:     Choose randomly channel $c \in C \mid c \notin FC(v)$
4:     $FC(v) = FC(v) \cup c$
5: until ((Uniform(0,1) $> P$) or ($FC(v) = C$))
6: for all channel $c \in FC(v)$ do
7:     Transmit Hello Message on channel $c$
8: end for
```

Notice that in this case the strategy is different. Here we do not add a channel c to the forward channel set $FC(v)$ with a fixed probability. In this algorithm, we add a randomly chosen channel to $FC(v)$, then with probability p we add another channel and with probability $1-p$ we stop adding channels to $FC(v)$. Of course we cannot add more channels than the available channels in the network. All the channels have the same probability of being chosen. During each round of hello messages a node can transmit on channels where there are neighbors listening but also on other channels. The idea is to advertise the presence of the nodes not only to known neighbors, but also to previously unnotified nodes.

We analyze the performance of this algorithm with different probabilites P. We evaluate the number of rebroadcast transmissions and the number of neighbors discovered after each round of hello messages is complete. We say that the k^{th} round of hello messages has been completed, if all the nodes in the network have transmitted k hello messages. Since we are using 802.11 as MAC protocol, the nodes are not synchronized; however, in order to simulate the way hello messages are transmitted, at the beginning of the simulation, we allow each node to choose randomly, from the interval $[0, 1]$ seconds, the time to transmit a hello message. Once a node has chosen the time for the first transmission, it transmits periodically every four seconds. Therefore, at the end of the interval $[0, 1]$ all the nodes have transmitted one hello message, at the end of interval $[4, 5]$ all the nodes have transmitted 2 hello messages and so on. We have used a random network of 15 nodes in an area of 600 m x 600 m. There are five channels available in the network. For clarity of the plots, we show curves only for a rebroadcast probability of 0.2, 0.4, 0.6, and 0.8. Each data point is the average of 50 simulations.

We can see in Figure 5.3, that with a rebroadcasting probability of 0.8 after ten rounds of hello messages, the saving in rebroadcast transmissions is about 35%. Also with the same probability, we can see in Figure 5.4 that after three rounds of hello packets, nodes have the same knowledge of the network as in the case when we use flooding.

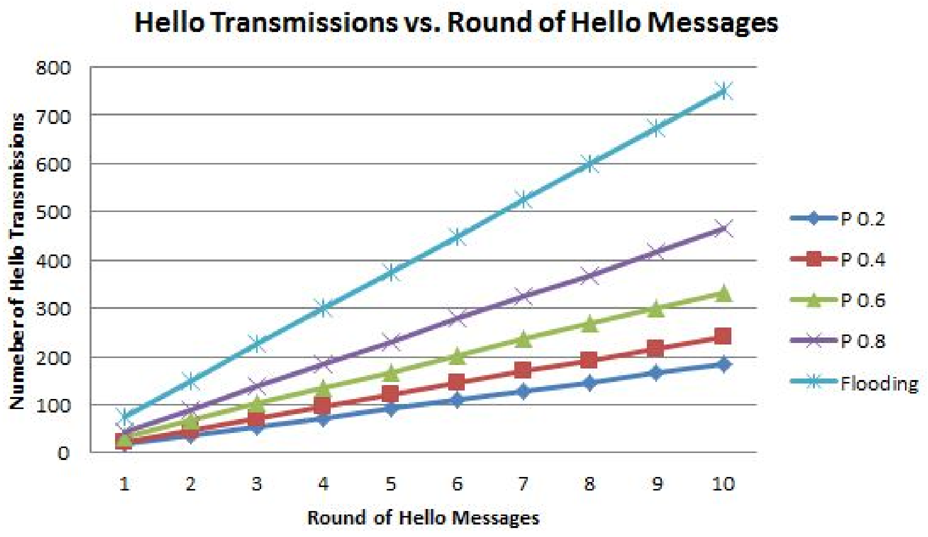

Figure 5.3: Average hello messages transmissions vs. round of hello messages

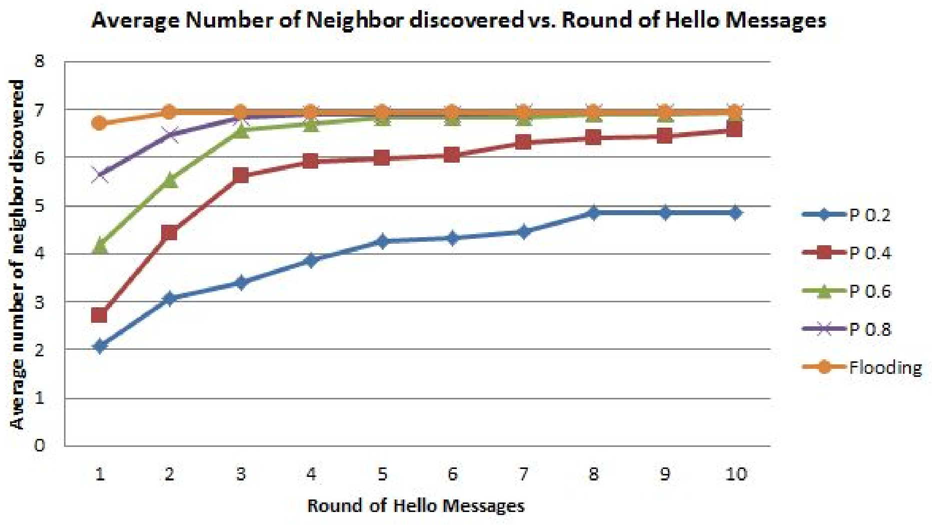

Figure 5.4: Number of neighbors discovered vs. round of hello messages

5.3.3 Multichannel Smart Broadcast Protocol

From the previous experiments we notice that a fixed probability of 0.8 provides good results for both metrics. However, we can do something else to improve the efficiency. Since the hybrid multichannel protocol requires fixed channel information to be propagated two hops away, we can adjust the probability per channel exploiting this information and the special properties of the channel assignment scheme. In this section we present a broadcast protocol that assigns different rebroadcast probability for each channel. We call it Multichannel Smart Broadcast Protocol (MSBP).

Notations used in Algorthim 3:

$N(v)$: Set of one-hop neighbors of node v

$N_c(v)$: Set of one-hop neighbors of node v with fixed channel c

$C_{N(v)}$: Set of fixed channels used by nodes in $N(v)$

$FC(v)$: Set of forwarding channels of node v

$P_c(v)$: Probability that node v rebroadcasts on channel c

$P_{MAX} = 0.8$

$P_{MED} = 0.6$

$P_{MIN} = 0.4$

Algorithm 3 MSBP: For each v receiving message from u

```
1: $FC(v) = \emptyset$
2: for all channel $c \in C$ do
3:     if $c \notin C_{N(v)}$ then
4:         $P_c(v) = 0$
5:     else
6:         if $c \in FC(u)$ then
7:             $P_c(v) = P_{MED}$
8:         else
9:             $P_c(v) = P_{MAX}$
10:        end if
11:    end if
12: end for
13: if $v.fixedChannel \in C_{N(v)}$ then
14:     $P_{v.fixedChannel}(v) = P_{MIN}$
15: end if
16: if $u.fixedChannel \in C_{N(v)\setminus u}$ then
17:     $P_{u.fixedChannel}(v) = P_{MIN}$
18: end if
19: for all channel $c \in C$ do
20:     With probability $P_c(v)$
21:     $FC(v) = FC(v) \cup c$
22: end for
23: Copy $FC(v)$ to the packet
24: for all channel $c \in FC(v)$ do
25:     Retransmit packet on channel $c$
26: end for
```

Basically this algorithm uses information about neighbors to assign the rebroadcast probabilities. A small rebroadcast probability is assigned to those channels which are expected to have scarce reuse in the neighborhood. In the hybrid channel assignment, each fixed interface tries to choose the least used channel among the two-hop neighbors. For that reason a node v assigns the smallest rebroadcast probability to its fixed channel and to the previous hop's fixed channel. Additionally, each node includes in the packet its forwarding channel set. This information is used by the receiving nodes to identify those channels where the transmitter of the previous hop transmmitted. Those channels are assigned a rebroadcasting probability of $P_{MED} = 0.6$. This part is inspired by the study performed by Ni et al. [2], which shows that after a previous broadcast, a new rebroadcast on average covers only 41% additional area. In the same paper it is shown that a probability of 0.6 offers a good tradeoff of retransmissions for delivery ratio. For all the other channels $c \in C_{N(v)}$, we use a rebroadcast probability of 0.8, which provided the best tradeoff of retransmissions for delivery ratio in the previous experiments. It is important to notice that the overhead introduced by the protocol is very small. The protocol requires that each node add to the broadcast packet only the forward channel list.

We compare the performance of this protocol with flooding and the fixed probability broadcast scheme with probability parameter 0.8. We evaluate the three protocols using five channels in a random network topology of 30 nodes located in an area of 1000 m x 1000 m. While we increase the number of route requests being broadcast (all the route requests are generated by different sources), we plot the delivery ratio, number of retransmissions and the route discovery latency. Each data point is the average of 50 simulation runs. The first two metrics are defined in the same way as in the previous section and the last metric, *route request latency*, is defined as the time elapsed since a source node initiates a route discovery and receives the corresponding route reply.

Figure 5.5 shows that MSBP involves 20% fewer retransmissions than fixed probability

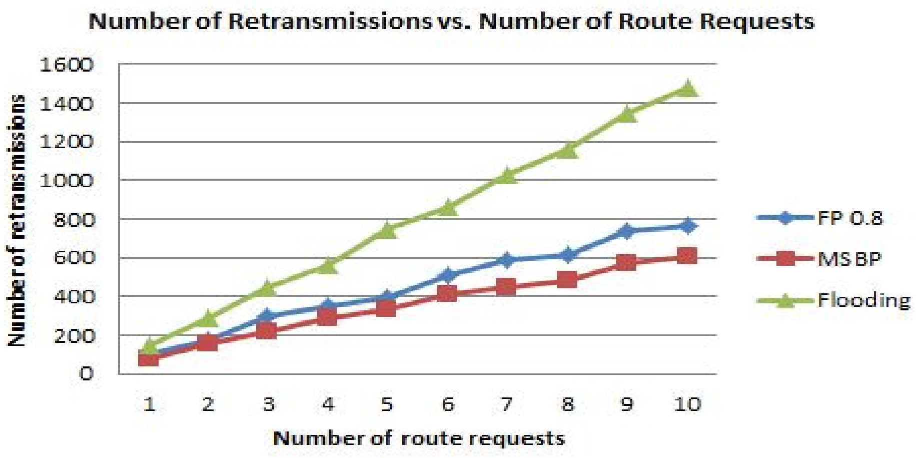

Figure 5.5: Number of retransmissions vs. number of route requests

broadcast. The difference is mainly due to the fact that MSBP assigns smaller rebroadcast probabilities to those channels where the expected additional coverage is low. In contrast, fixed probability broadcast sets the rebroadcast probability to 0.8 for all the channels where neighbors are listening, regardless of neighborhood information.

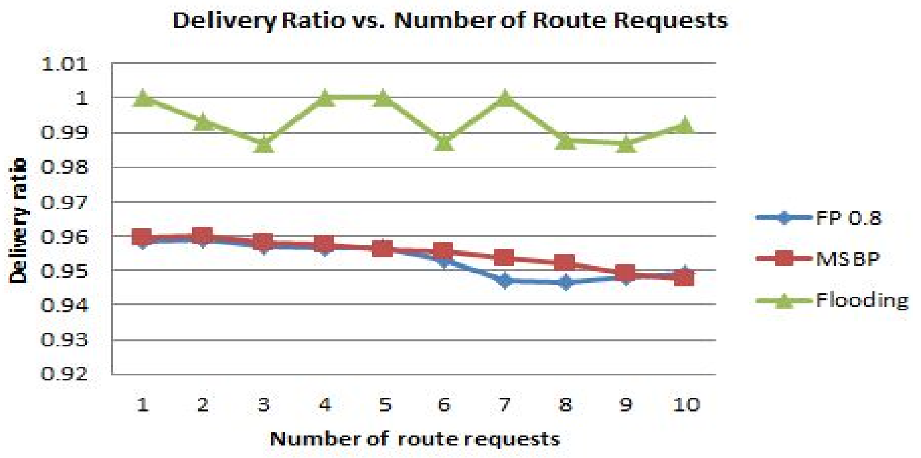

Figure 5.6: Delivery ratio vs. number of route requests

One may expect that assigning smaller rebroadcast probabilities to some channels should hurt reachability; however, as we can see in Figure 5.6, if the assigned probabilities reflect somehow what is happening in the neighborhood of the nodes, then the delivery ratio is not worsened. In Figure 5.6 we can see that the delivery ratio for MSBP is practically the same as in the fixed probability case (about 0.95).

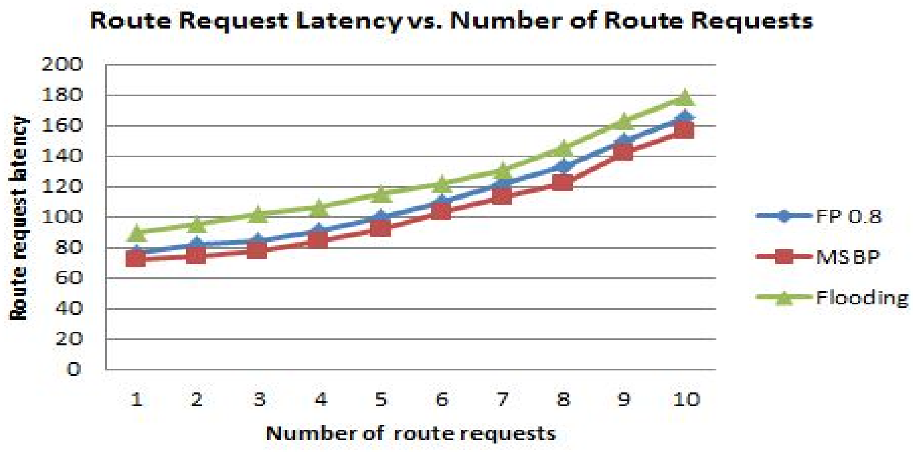

Figure 5.7: Route request latency vs. number of route requests

Figure 5.7 shows the route request latency. The routing protocol considered tries to find the best route according to the metric MCR [13]; however, due to collisions, even for flooding it is not always possible to find the best route. With a probabilistic approach, it is likely that the best route found differs from that found by flooding. In these results we have considered the time elapsed from initiation of the route request until reception of the route reply for the best possible route. In Figure 5.7 we can see that both probabilistic approaches offer improvement in the route request latency; however, MSBP offers the smallest latency, about 12% smaller than that of flooding.

5.4 Summary

In this chapter we have evaluated a probabilistic approach to improve the efficiency of broadcasting on MC-MR networks. We limit the channels that a node may use to broadcast to those used by one-hop neighbors. We find that using a fixed rebroadcast probability per channel of 0.8, the saving in the number of retransmissions is about 40%, while the delivery ratio is about 0.95. Using the properties of the hybrid channel assignment mechanism, we propose MSBP. This protocol assigns different rebroadcast probabilities to different channels. When compared to a fixed probabilty of 0.8, this protocol offers an improvement of about 20% in the number of retransmissions, while keeping the same delivery ratio.

CHAPTER 6

IMPLEMENTING THE HYBRID MULTICHANNEL PROTOCOL IN NS-2

In this chapter we describe the changes required to implement the hybrid multichannel protocol in the simulator *ns-2*. Here we detail how to extend the node model to support multiple interfaces and how to implement the different services involved in the hybrid multichannel protocol.

6.1 Multiple Interface Model

In order to enable *ns-2* to support multiple channels and multiple interfaces, non-trivial modifications are required. The first thing that we need to define is how to extend the mobile node architecture to support multiple interfaces. We have chosen the model proposed by Aguero and Perez [17], seen in Figure 6.1. In this model the idea is to use the same modules as the single interface case from the address multiplexer to the top; however, from the link layer to the bottom, we need to create as many copies of the original chain of objects as the number of interfaces the node has. For outgoing packets, the routing agent has the responsibility of choosing the interface where each packet has to be transmitted; that is, the routing agent has to pass the packet to a specific link layer module. Once the packet arrives at the link layer, the process is the same as when the node has only one interface; so the packet travels to lower layers until being transmitted to the channel.

On the other hand, for incoming packets there are few differences from the original operation of the simulator. Incoming packets arrive through the corresponding channel and travel through the different entities in ascending order. All the link layer modules have a common connection to the adress multiplexer, and from this point to above, all the packets are treated as in the original model.

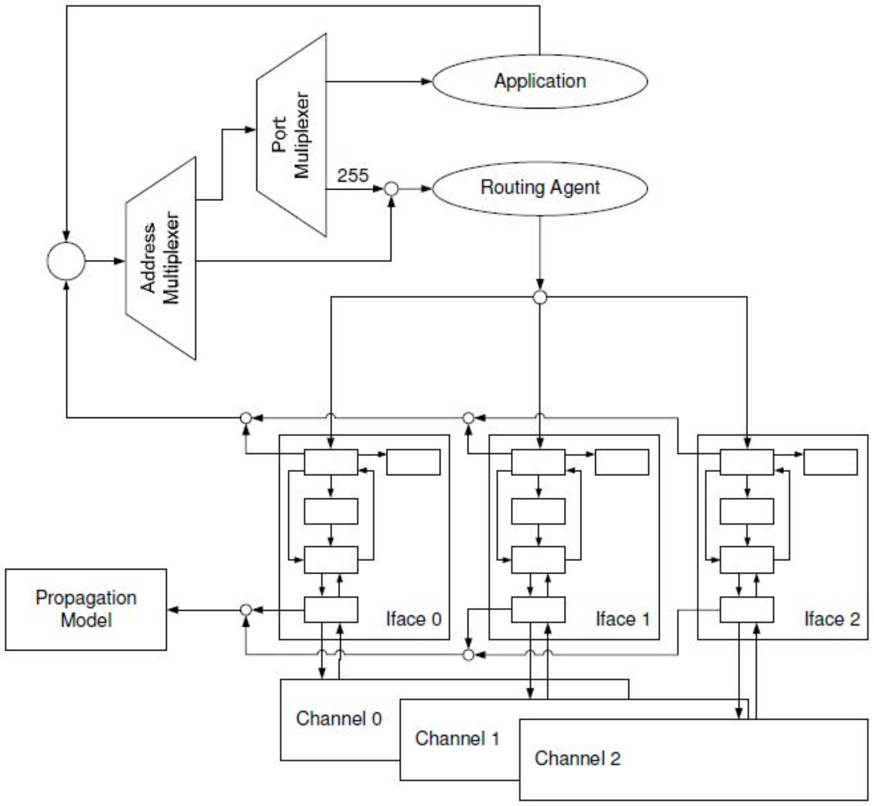

Figure 6.1: Mobile node architecture with multiple interface support

The mobile node architecture is independent of the number of channels available in the network. We can implement nodes with fewer interfaces than channels, which offers great flexibility to implement static and dynamic multichannel protocols. Each interface is tuned to a single channel, so each branch of the architecture is able to transmit and receive packets on a single channel at time. This implies that the model uses different instances of the wireless channel class. As we will see in the next section, this allows us to implement channel switching, as required by the hybrid multichannel protocol.

The mobile node model that we use is basically the same as that described in [17], but we change the way ARP tables are stored. In the model described in [17], there is an ARP table for each branch of the node, i.e., each link layer maintains its own table. However, we found this implementation inefficient since each link layer module will need to learn the MAC address of the next hop even when another link layer module in the same node already knows it. We propose to use only one ARP table common to all the link layer modules, so if one interface already performed an ARP request for a node X, the other link layer modules in the same node will not need to do it again.

6.2 Channel Switching

The class Wireless Channel maintains two lists: one for the interfaces that are tuned to the channel and another for the respective nodes to which interfaces belong. In the Tcl script we need to define the channel for each interface for all nodes. The process to tune an interface to a specific channel is divided into two steps: first, the channel adds the node to its *NodeList* and then it adds the respective interface to its *InterfaceList*. Only after these two steps are completed can the node transmit and receive packets through this interface.

Switching the channel involves many considerations. These include not only the switching channel mechanism, but also certain issues that can arise if appropriate rules are not correctly defined for the process. We start by describing the process of channel switching that we have implemented in the model described in Section 6.1. When an interface transmits a packet on a particular channel, only those interfaces included in the *InterfaceList* of that channel may receive the packet. A channel object may add or delete elements from these lists. We will call the channel where the interface is currently tuned *the current channel*, and the channel where the interface wants to switch to *the new channel*. When an interface needs to switch the channel, the process is divided into the following four steps:

1. The current channel deletes from its *InterfaceList* the interface object that invoked the

switching.

2. The current channel deletes from its *NodeList* the respective node.

3. The new channel adds the node to *NodeList*.

4. The new channel adds the interface that invoked the switching to *InterfaceList*.

In order to simulate the switching delay we implement a timer between the second and third steps. We can see that the process itself is not difficult; however, a channel switching cannot be executed arbitrarily. During the implementation we found a series of problems that may arise when an interface invokes a channel switching. The following list summarizes them and describes the solution that we used to circumvent them:

- A channel switching is invoked while there are still packets pending to transmit. As we will see in the next section, a channel switching is invoked only from the link layer, and the link layer is not aware whether all the packets it passed to the interface queue have actually been transmitted; therefore, the next time the link layer requires a channel switching, it is possible that not all the scheduled packets have been transmitted. In this scenario we defer the channel switching until the last packet has been transmitted succesfully.

- A channel switching is invoked while the interface is receiving a packet. Every time an interface is scheduled to receive a packet, we block the interface such that it cannot switch the channel. We specify a deferment time, equal to the transmission time of the incoming packet plus a small guard time; therefore, only after the packet has been completely received can the switching be performed.

- One problem, which is strictly related to the node architecture used in *ns-2*, concerns ARP requests. Suppose that node A broadcasts an ARP request for node B's physical address, and node B's fixed channel is *c*. Suppose that the ARP request was scheduled as the last packet when channel *c* queue is served. The problem is that since ARP requests are broadcast packets, the MAC layer does not expect an acknowledgment for it, so the interface may potentially be changed before the ARP reply is received. During the simulations we found that due to this problem it is possible that the ARP reply is never received, which causes all the packets of the corresponding flow to be dropped. To solve this problem we treat ARP in a special way. When an ARP request is transmitted we block any channel switching until the respective ARP reply is recieved.

6.3 Hybrid Multichannel Protocol

The hybrid multichannel protocol specifies that each interface has to maintain a packet queue for each channel. In the implementation of Net-X, the proposed *channel abstraction layer* has the responsibility of selecting the interface that will be used to transmit each packet, and it also enqueues and dequeues packets from the corresponding channel queues. This layer is included between the IP stack and the interface device driver.

The node architecture in *ns-2* differs from that of a real computer. We have implemented the hybrid multichannel protocol in a different way. Given the node model we are working on, we cannot implement a channel abstraction layer. Each interface is simulated with a unique link layer, interface queue and MAC layer. We have divided the responsibilities of the channel abstraction layer in two layers: in the routing protocol, we extend the routing tables to include the interface and channel that have to be used to reach a particular destination. We implement the channel queues in the link layer as illustrated in Figure 6.2. Every time the link layer receives a packet from the routing agent, it assigns the packet to the channel specified by the routing agent. Notice that in this case the link layer does not need to specify the radio to use since each link layer has access to a single radio. It is the responsibility of the routing agent to send down the packet to the correct interface; therefore, the routing agent is kept aware of which channels are supported by each interface.

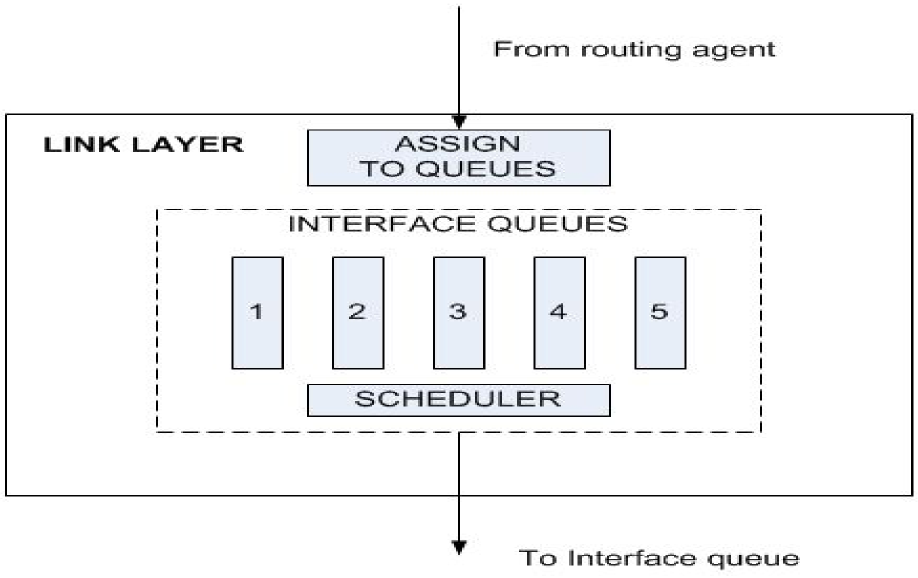

Figure 6.2: Hybrid multichannel protocol in *ns-2*

CHAPTER 7

CONCLUSIONS

This thesis studies questions related to multichannel multiradio (MC-MR) wireless networks. In the first part, we evaluate the impact of hiding the switching latency by using an additional switchable interface. In the second part, we approach the problem of reducing the cost of broadcasting in MC-MR wireless networks.

The main drawback of the hybrid multichannel protocol is the non-negligible latency incurred on channel switching. This latency increases queueing time of the packets. The throughput in a TCP connection depends directly on the round trip time, so when multiple channel switchings are required in the route, the performance of TCP degrades considerably. In Chapter 4, we have studied with several experiments the impact of using two switchable interfaces, with the constraint that only one of them can transmit at a time. That is, we study the impact of hiding the switching latency. From the results, we can conclude that the improvement is around 5% when a single switching is required in the route from source to destination. The more switching required, the more saving on switching delays and consequently the more improvement in throughput. We obtained improvement of around 25% in some cases. We notice that the use of a third interface has a greater impact in TCP than UDP, since TCP involves forwarding in both directions; therefore, channel switching is more frequent in TCP than in UDP.

From the experiments performed, we notice that the use of a third interface is only beneficial when nodes have to switch channels. In fact, there are situations where it would be better to use two fixed interfaces instead of two switchable. Consider for example the scenario shown in Figure 4.5. However, let us suppose that nodes D_1 and D_2 operate on the same fixed channel, so the relay node uses the same channel to transmit to both destinations. In this scenario one transmitting interface is enough. However, if the flows are received on different channels, the contention is reduced and the aggregate throughput would be larger than the case when both flows are received on the same channel. Therefore, even in the many situations in which two switchable interfaces offer more improvement than two fixed, the optimal is to have an algorithm that dynamically changes the radio configuration in the node. Also, if the traffic pattern is well known it is possible to have some nodes with two interfaces and others with three.

Broadcasting over MR-MC ad hoc networks has not been studied previously in the context of hybrid channel assignment. In this work we evaluated a probabilistic approach and presented MSBP, a smart probabilistic broadcast protocol for MC-MR ad hoc networks that operate with hybrid channel assignment. This protocol exploits neighbors' information and the properties of the hybrid multichannel protocol to assign different rebroadcast probabilities for each channel. Simulation results have shown that MSBP is able to introduce significant improvement compared to flooding while keeping the delivery ratio high. Most of the existing broadcast protocols try to achieve perfect reachability; however, for services like route discovery, it is usually not strictly required. We have evaluated the performance of our protocol when multiple route requests are initiated and it turns out that in all the simulations a route reply was always received.

Since our algorithm is based on a probabilistic approach, it does not fit every scenario. There are scenarios where there is a small chance that the route request cannot reach the destination. It is necessary to regenerate the route request if the previous one failed to reach the destination. For this reason, our protocol performs better in dense networks. It remains for future work to consider the case of sparse neighborhoods when assigning the rebroadcast probabilities. Additionally, all the experiments performed in this work consider static networks. It will be necessary to adapt the protocol when there is mobility in the network. When nodes are mobile there is a chance that the *neighbor table* does not contain all the actual neighbors at a given time instant. Given that our protocol uses *neighbor table* information, the probability values could reflect a scenario different from the one that is actually happening in the neighborhood of a node.

REFERENCES

[1] P. Kyasanur, "Multichannel wireless networks: Capacity and protocols," Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana, IL, May 2006. [Online]. Available: http://www.crhc.illinois.edu/wireless/papers/kyasanur2006Thesis.pdf

[2] S. Ni, S. Tseng, Y. Chen, and Y. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, Aug. 1999, pp. 151–162.

[3] Y. Ding, K. Pongaliur, and L. Xiao, "Hybrid multi-channel multi-radio wireless mesh networks," in *Proceedings of the IEEE International Workshop on Quality of Service 2009*, July 2009, pp. 1–5.

[4] T. Shen, "Experiments on a multichannel multi-interface wireless mesh network," M.S. thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 2008.

[5] J. Robinson, K. Papagiannaki, C. Diot, X. Guo, and L. Krishnamurthy, "Broadcasting protocols for multi-radio multi-channel and multi-rate mesh networks," in *WiNMee Workshop*, Apr. 2005, pp. 1–6.

[6] B. Williams and T. Camp, "Comparison of broadcasting techniques for mobile ad hoc networks," in *Proceedings of MOBIHOC'02*, June 2002, pp. 194–205.

[7] A. Hanashi, A. Siddique, I. Awan, and M. Woodward, "Performance evaluation with dynamic probabilistic broadcasting for flooding in mobile ad hoc networks," *Simulation Modelling Practice and Theory*, vol. 17, no. 2, pp. 264–375, 2009.

[8] J. Qadir, A. Mistra, and C. Chou, "Minimum latency broadcasting in multi-radio multi-channel multi-rate wireless meshes," in *3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks, 2006 (SECON '06)*, vol. 1, Sep. 2006, pp. 80–89.

[9] S. Ahuja and S. Ramasubramanian, "Broadcasting in wireless infrastructure networks employing directional antenna," in *Proceedings of the IEEE International Conference on Communications (ICC)*, Cape Town, South Africa, May 2010, pp. 1–5.

[10] A. Jiang and K. Xie, "Minimum energy broadcast in multi-channel wireless sensor network with directional antennas," in *Proceedings of the IEEE International Conference on Electronics and Information Engineering*, Kyoto, Japan, Aug. 2010, pp. V2 564–567.

[11] L. Li, B. Qin, C. Zhang, and H. Li, "Efficient broadcasting in multi-radio multi-channel and multi-hop wireless networks based on self-pruning," in *Proceedings of the IEEE High Performance Computation Conference 2007*, Houston, TX, Sep. 2007, pp. 484–495.

[12] M. Song, J. Wang, and Q. Hao, "Broadcasting protocols for multi-radio multi-channel and multi-rate mesh networks," in *Proceedings of the IEEE International Conference on Communications (ICC)*, Glasgow, Scotland, June 2007, pp. 3604–3609.

[13] P. Kyasanur, P. Chereddi, and N. Vaidya, "Net-x: System extensions for supporting multiple channels, multiple interfaces, and other interface capabilities," University of Illinois at Urbana-Champaign, Wireless Networking Group, Urbana, IL, Tech. Rep., Aug. 2006.

[14] C. Chereddi, "System architecture for multichannel multi-interface wireless networks," M.S. thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 2006.

[15] N. Vaidya, *Wireless Networks.* Class notes for ECE439, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Jan. 2010.

[16] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, June 2004, pp. 114–128.

[17] R. Aguero and J. Perez, "Adding multiple interface support to ns-2," Universidad de Cantabria, Spain, Tech. Rep., Jan. 2007.