UNIVERSITÀ DI PADOVA  FACOLTÀ DI INGEGNERIA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

TESI DI LAUREA

NETWORK CODING AWARE QUEUE MANAGEMENT IN MULTI-RATE WIRELESS NETWORKS

RELATORE: Ch.mo Prof. Michele Zorzi

CORRELATORE: Ch.mo Prof. Srikanth V. Krishnamurthy

LAUREANDO: *Nicola De Coppi*

Padova, October 24th 2011

Contents

Abstract

This thesis presents a queue management algorithm to improve Network Coding in a Multi Rate wireless scenario. Network Coding is a technique that allows to increase the network capacity. It has been proposed as an alternative to the store and forward paradigm. Network Coding aims to reduce the number of transmissions when the network is congested. Recently some papers have presented a rate control adaptation for network coding. They show that changing the transmission rate can improve the performance of network coding.

In this thesis we focus on network coding for unicast flows, in particular we take into consideration the COPE architecture. We study what happens when two nodes transmit with different rates and we propose a Markov chain to model this scenario. Furthermore, we propose a queue management algorithm to increase the coding opportunity and the throughput of the network. In COPE, a node codes two or more packets together when packets are headed to different nexthops. When there are just packets for the same nexthop, a node loses a coding opportunity. The queue management algorithm increases the coding opportunity prioritizing the channel access of sender nodes based on the queue information and on PER of the links. We simulate this algorithm on top of COPE in a multi rate scenario with NS 2 and we show that our algorithm yields throughput gains of up to 57% compared to COPE.

Chapter 1

Introduction

This thesis has been developed during the exchange program at the University of California, Riverside. During this period abroad, I had the opportunity to work in the Networking Lab under the supervision of Prof. Srikanth V. Krishnamurthy.

Wireless networks are every year more popular and indispensable. They provide connectivity for different types of devices: from laptops to sensors, from smartphones to tablets. However the wireless channel puts fundamental limitations to the performance of devices due to errors and collisions.
Several solutions have been proposed to mitigate the problem related to wireless channels and consequently to improve the throughput: one of these is Network Coding.
The fundamental idea of Network Coding is to reduce the number of wireless transmissions coding together different packets. In contrast to the traditional *store* and *forward* paradigm, Network Coding uses a *store*, *code* and *forward* approach.
Network Coding has been applied to both multicast and unicast traffic. In this thesis we focus on unicast traffic, in particular we base our analysis and simulations on COPE [2]. This architecture for wireless mesh-networks has shown that it can improve the throughput in the presence of dense networks and bursty flows. Recently, rate adaptation algorithms have been proposed [3] and [4] on top of COPE architecture to increase further the capacity of wireless networks.
In this thesis we propose a queue management algorithm to increase the probability of coding packets in a multi-rate wireless network.

1.1 Motivation

In COPE, a node codes together just packets for different nexthops. When a node has packets with different nexthops, it computes the probability that receiver nodes can decode the coded packet. If the probability is greater that a certain threshold, it XORs native packets and it sends the coded packet on the wireless channel.
However, when a station has packets just for the same nexthop, it sends the native packet and it loses the coding opportunity. Each node keeps a virtual queue for packets with the same nexthop. Since the *Packet Delivery Ratio* (PDR) can be different for each wireless link, virtual queues can have different length. To have virtual queues unbalanced can increase the probability that one of these becomes empty and the node loses the coding opportunity.
Furthermore the physical access CSMA/CA is fair with nodes in the same *Carrier Sense* range. This solution may not be optimal in the presence of heavy traffic. Let's consider three stationary nodes Alice, Bob and Jack. Alice and Bob send packets to the relay node Jack and Jack forwards these packets to other nodes. Each node has the same probability to access the channel. Since the input traffic for node Jack is double of its output capacity, it starts accumulating packet in the queue. If we suppose that Jack uses Network Coding, it drains the packets queue faster. However, every time that Jack loses a coding opportunity, it reduces its speed draining and it accumulates packets in the queue.
Therefore, our scope is to increase the coding opportunity and we do it prioritizing the channel access based on the queue information at the node that does the coding.

1.2 Proposed solution

One solution could be to use a perfect scheduling. The node, which codes packets together, sends the information on when to transmit to other nodes. However, this node cannot know if other nodes have packets to transmit, thus using a perfect scheduling is not a good solution.
For the same reason, also delaying packets is not a good solution, indeed COPE designs the coding scheme with a principle of never delaying packets.
From these considerations, we propose to change the probability of accessing the channel tuning the *Contention Window* (CW) at the MAC layer. Based on how many packets are in the virtual queues, the relay node sends to neighbor nodes the initial value

of CW_{min} with the goal of balancing its virtual queues. Other queue management algorithms have been proposed at top of Network Coding, however no one proposes to change the probability of accessing the channel.
Furthermore, we propose the algorithm in a multi-rate wireless network, considering that each node can transmit with a different rate. We analyze when two nodes transmit with different rates and we present a Markov chain to model this behavior.
The thesis is organized in six chapters and we present topics with a top down approach. We first introduce Network Coding and previous studies on it in Chapter 2. We present basic information on IEEE 802.11 in Chapter 3 and fundamental analysis on wireless channel in Chapter 4. Our innovative contributions to Network Coding are presented in Chapters 5, 6 and 7. We first introduce the problem analysis and the model with Markov chain and the proposed algorithm in Chapter 5. This is followed by evaluation of our algorithm through simulation with NS 2 and the conclusions respectively in Chapters 6 and 7.

Chapter 2

Network Coding

Network Coding is a technique that allows reducing the number of transmissions mixing data packets at intermediate nodes.

2.1 Introduction

Network Coding is used in wireless networks where the wireless channel is shared between several stations. This allows nodes to overhear packets that are not direct to them and it creates the opportunity for the coding process.
Basically, intermediate nodes code together two or more data packets whenever there are the conditions that the receiver nodes can decode the coded packet. Reducing the number of transmissions increases the capacity of a wireless network.

2.1.1 Multicast vs Unicast

Most of the network coding papers in the literature are related to multicast traffic. Network Coding has been introduced for the first time by Ahlswelde *et al.* [5]. He showed that routers can achieve multicast capacity mixing information in different messages. This paper was followed by Li *et al.* [7] who showed that linear codes achieve the maximum capacity bound. Li proposes the *linear network coding* where the output flow of a given node is obtained as a linear combination of its input flows. However this approach needs a centralized knowledge about the network topology.
Koetter and Medard [9] propose *random network coding* where the linear coding is

substituted with a polynomial algorithm for encoding and decoding.
Chou *et al.* [8] proposes a distributed scheme for pratical network coding which does not need a centralized knowledge of the network. Furthermore, this solution is robust to random packet loss and delay.
Katti *et al.* [2] were the first to show how to apply network coding to unicast traffic. They proposed COPE as a distributed scheme where each station is in promiscuous mode. Each station stores the overheard packets for a short time T and it needs to know which packets its neighbors have to perform the coding. This information can be sent in periodic reception reports or it can be estimated from the routing protocol that uses ETX/ETT metric. Furthermore, Katti shows how network coding can be integrated in the current network stack.
In this thesis we focus on unicast traffic considering COPE's architecture.

2.1.2 Examples

We first consider three nodes Alice, Bob and Jack as in Figure 2.1. Alice wants to communicate with Bob and Bob with Alice. They are not in the transmission range and they use Jack to forward packets.

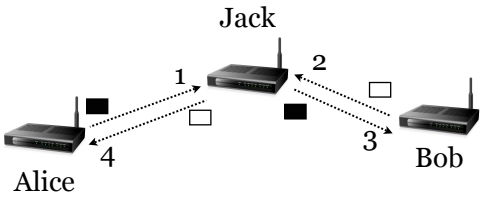


Figure 2.1: Current Approach

In current approaches, Alice first sends her packet to Jack, then Bob sends his packet to Jack. Jack forwards Alice's packet to Bob (third transmission) and Bob's packet to Alice (fourth transmission). All this process requires four transmissions.

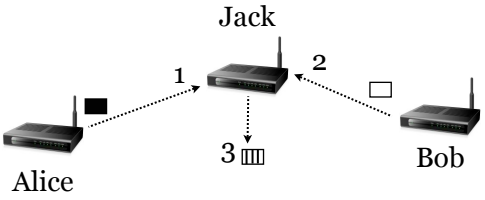


Figure 2.2: Network Coding - COPE

Using Network Coding (Figure 2.2), Jack transmits only one packet which is the bitwise sum of the packets from A and from B. Therefore the number of transmissions is reduced from four to three.

Five nodes topology

Let's consider an example that we will use again in this thesis. We consider a five nodes topology as in Figure 2.3. In this network there are five nodes Alice, Bob, Chloe,

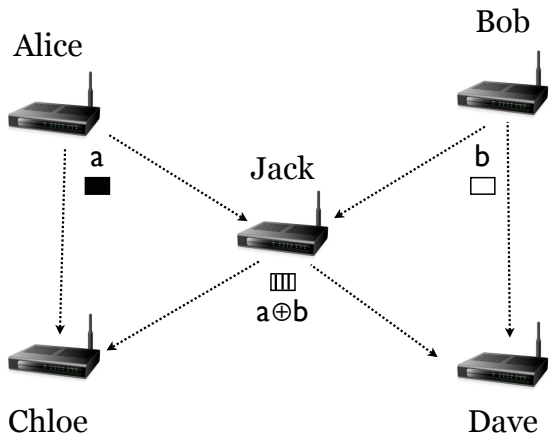


Figure 2.3: Example five nodes using network coding

Dave and Jack and there two flows from Alice to Dave and from Bob to Chloe. As in the first example, Jack is the relay node and it is responsible to forward packets. We consider that network coding is used and Jack sends the XOR packet ($\mathbf{a} \oplus \mathbf{b}$) of those from Alice ($\mathbf{a}$) and Bob ($\mathbf{b}$).
Using network coding the number of transmissions is reduced from four to three. However, in this example, to decode successfully the XOR packet, stations Chloe and Dave have to overhear correctly respectively packet $\mathbf{a}$ and $\mathbf{b}$.
Radio channel condition and the transmission rate of the stations Alice and Bob affect the overhearing property. If a receiver node cannot overhear the packet from the sender, it will not be able to decode the XOR packet from Jack and the native packet will be retransmitted.
In the next section we will see how COPE works. In the third section of this chapter, we consider a rate adaptation algorithm to improve the performance of the network, in particular we present Kim's paper [3].

2.2 XORs in the Air - COPE

Katti *et al.* [2] proposes COPE, a new forwarding architecture for wireless mesh networks. COPE inserts a new coding layer between the IP and MAC layer.
Katti shows that this architecture can improve the capacity of wireless networks considering unicast traffic, dynamic and potential bursty flows.
COPE identifies coding opportunities and it codes multiple packets together in a single packet. It works with both TCP and UDP flows.
The throughput of a wireless network with COPE can be 3-4 times greater than the same network without coding. Especially without congestion control (e.g. UDP) COPE reduces the probability that a congested router drops packets and this involves a better performance for the network.

2.2.1 Overview

COPE is based on three main techniques:

Opportunistic Listening: Each station is in promiscuous mode and it stores the overheard packets for a short time T (the default time is $T = 0.5s$). In addition, each node broadcasts *reception reports* to inform its neighbors of the packets it has stored. When a node has a packet to transmit, it includes the list of stored packets. If it has no packet to transmit, it sends the list in a special control packet.

Opportunistic Coding: *What packets should a node code together to maximize throughput?* The idea is that each node codes together as many packets as possible considering the decoding probability of the receiver nodes. When a receiver node decode the packet, it extracts its native packet. If it is not able to decode the packet, the native packet will be retransmitted.
The coding algorithm is based on the following rules:

To transmit n packets, $p_1, ..., p_n$, to n nexthops, $r_1, .., r_n$ a node can XOR n packets only if each next-hop r_i has all n-1 packets p_j for $j \neq i$.

Learning Neighbor State: A node learns what packets its neighbors have from the reception reports. However, if this information is not available, the station needs

to guess whether a neighbor has a particular packet.
The wireless routing protocol computes the delivery probability between every pair of nodes and uses it to identify good paths (e.g. ETX/ETT). Similarly COPE uses the delivery probability to know if a packet has been received correctly. When a node makes an incorrect guess, the native packet is retransmitted.

2.2.2 Coding Gain

A really important concept in network coding is the Coding Gain.
The Coding Gain is defined as:

$$G_{coding} = \frac{\#\ of\ transmissions}{\#\ of\ transmissions\ with\ COPE}. \tag{2.1}$$

The numerator counts the number of transmissions required without COPE, the denominator counts the minimum number of transmissions used by COPE to deliver the same amount of native packets.
Let's compute the Coding Gain for some wireless topologies presented in Figure 2.4. In the cross topology of Figure 2.4 (b), there are two flows of packets intersecting at

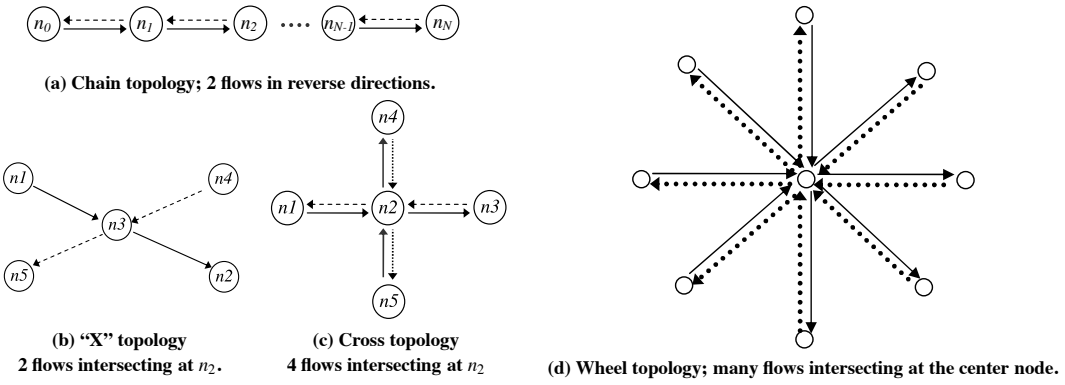


Figure 2.4: Simple topologies present in a wireless network from [2]

node $n3$. This example is equal to that presented in Figure 2.2. We have seen that using COPE the number of transmissions is reduced from 4 to 3, thus the Coding Gain is $G_{coding} = \frac{4}{3} = 1.33$.

2.2.3 Coding+MAC Gain

COPE brings improvement also to the MAC layer, indeed if a node drains the queue at MAC layer faster, it reduces the probability to drop packets.
When node $n3$ is not coding and nodes $n1$,$n4$ have continuously packets to transmit, node $n3$ becomes a bottleneck of the network. Indeed the MAC protocol tries to be fair and it divides the bandwidth between the 3 contending nodes: $n1$,$n4$ and $n3$. Node $n3$ receives an average of two packets for each packet that it transmits and it starts accumulating packets in the queue. COPE permits to drain the queue twice as fast doubling the throughput of this network. Therefore considering also at MAC layer, the Coding+MAC gain for "X" topology is equal to 2.
Let's consider Figure 2.4 (c) , where each edge node is both a sender and a receiver. There are 4 transmissions from sender nodes and these 4 packets can be coded together and transmitted with a single transmission. The total number of transmissions using COPE is 5 and the Coding Gain is $\frac{8}{5} = 1.6$. The MAC+Coding gain turns out to be 4, since node $n2$ can drain the queue four times faster.
In the Wheel topology of Figure 2.4 (c), the Coding+MAC gain increases depending

Toppology	Coding Gain	Coding+MAC Gain
Infinite Chain (a)	2	2
"X" (b)	1.33	2
Cross (c)	1.6	4
Infinite Wheel (d)	2	inf

Table 2.1: Gain for each topology presented in Figure 2.4

on the number of outer nodes. Let's assume N outer nodes, the inner node codes N packets together and it drains its queue N times faster than the same topology without COPE. The Coding Gain and Coding+MAC Gain for each topology is reported in the Table 2.1.

2.2.4 COPE's Architecture

This section presents more details about COPE and how it is implemented in wireless nodes. COPE inserts a coding layer between MAC and IP layers, see Figure 2.5. The COPE header is made by three fields: Encoded, Reports and ACKs.
The Encoded field identifies native packets that form XOR packet. When a station

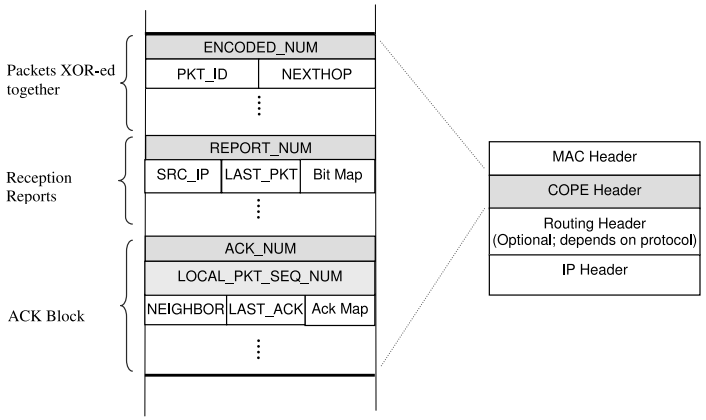


Figure 2.5: COPE header from [2]

receives a coded packet, it searches if it is one of the nexthop in the Encoded field. If so, it tries to decode the packet.
The Reports field records the reception reports. These reports are used to know which packets each node has overheard and they are needed to compute the probability of decoding.
Since each decoded packet must be ACKed at COPE layer, Cope uses cumulative ACKs to reduce the overhead in the network. It reports these on the ACKs field.
Let's take a look at COPE's main phases: coding, transmission and decoding, acknowledgment and retransmission.

Coding

COPE is implemented with the principle of never delaying packets. If in the queue there are packets with the same nexthop, nodes transmit the native packet when the wireless channel is available. This principle is based on the fact that nodes cannot know if they will receive a packet with a different nexthop, so it's better not to wait and send the native packet.
The coding phase can be listed as follows for each node:

1. dequeue the packet at the head of the OUTPUT queue
2. check if there are packets of similar size and different nexthop
3. code those packets together (XOR)
4. broadcast the coded packet

COPE never codes together packets headed to the same nexthop, otherwise the nexthop will not be able to decode them. Each node keeps a virtual queue with packets headed to the same neighbor. The VIRTUAL queue contains pointers to the packets in the OUTPUT queue. The VIRTUAL queue simplifies the searching operation. When COPE dequeues the first packet from the OUTPUT, it chooses which packets to code together from those at the head of each VIRTUAL queue.
The most important thing is to ensure that the receiver has high probability of decoding its native packet. Therefore a node estimates the probability that each of its neighbors has already heard packets that form the XOR packet. Sometimes a node can be sure that the neighbor has heard the packet, for example if the neighbor is the previous hop of the packet or when the node gets the receptions reports. If this information is not available, the node estimates the probability from those computed by the routing protocol.
In the case that a node encodes together n packets, the probability P_D for a node to decode its native packet i is equal to the probability that it has heard all the $n-1$ native packets XOR-ed with it, i.e.:

$$P_D = P_1 \times P_2 \times \cdots \times P_{n-1} \tag{2.2}$$

where P_j is the probability that the receiver node has heard packet j. COPE codes n packets together if the probability to decode them P_D is at least 0.8.

Transmission and Decoding

COPE uses a *pseudo-brodcast* technique to send packets. The MAC field of the coded packet is set to one of the receivers. In the COPE header, there is a XOR field where there is a list of all nexthops of the packet. When a node receives a packet with a MAC address different from its own, it checks if it is included in the list of the nexthops. If it is included, it proceeds decoding the packet even if the MAC address is different. Since all packets are sent using 802.11 unicast, the MAC detects collisions and it uses the backoff. Each node keeps a copy of each native packet that it has received or sent out. When it receives a XOR-ed packet, it reads the ID of each native packet used in the coding and it checks if it has all $n-1$ native packets. If it has $n-1$ native packets, it XORs them with the coded packet to obtain its native packet.

Acknowledgment and Retransmission

Encoded packets are headed to multiple nexthops, but the sender get synchronous ACKs only from the MAC destination of the coded packet. How do nexthops ack the native packet if they decode the XOR packet successfully?
COPE addresses this problem with asynchronous ACKs and retransmissions. When a node sends an encoded packet, it schedules a retransmission events for each native packet that is encoded. If the native packet is not acked within T_a seconds (T_a is quite larger than a single link round trip time), the node resends the native packet. When the nexthop receives an encoded packets in promiscuous mode, if it is able to decode and to extract its native packet, it schedules an ACK event. To reduce the overhead of the network, the node sends cumulative ACKs.
Figure 2.7 reports a flow chart for the sender side and for receiver side.

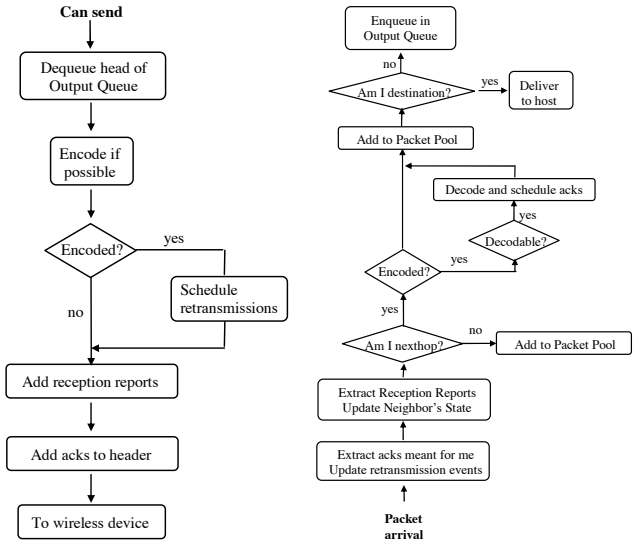


Figure 2.6: Flow chart for COPE implementation from [2]

2.2.5 Experiments and Results

Katti et al. implemented COPE in a 20-node wireless testbed. They found that:

- When the wireless medium is congested and the traffic consists of many random UDP flows, COPE can increase throughput 3-4 times. In this case, the congestion

control is not used and COPE's throughput reaches the Coding+MAC gain.

- When TCP control is used, the throughput improvement with COPE agrees with the expected coding gain unless there are many hidden terminals. In this case the number of collisions increases and TCP reduces its packet sending rate, limiting the coding opportunities.

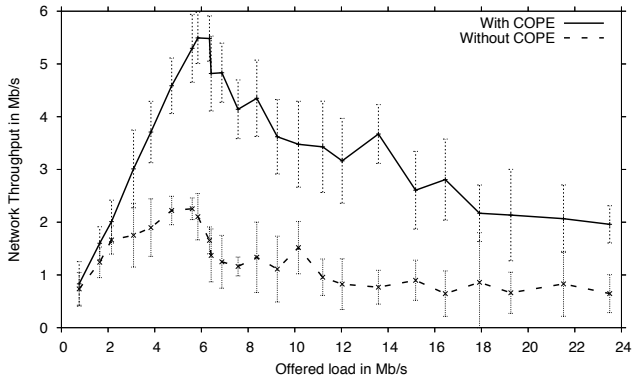


Figure 2.7: Results for UDP flow with randomly picked source-destination pairs and Poisson arrivals from [2]

2.3 Rate Control in Network Coding

COPE uses by default the lower rate available (i.e., for 802.11g it is 6Mb/s) so each node can overhear a packet in its transmission range.
Recently some papers [3] and [4] have shown that rate adaptation can improve further the capacity of wireless networks.
From one side the transmission rate should be selected to increase the probability that nodes overhear native packets and transmissions with lower rate have a higher probability to be received correctly. From the other side, decreasing too much the transmission rate reduces the throughput of the wireless network. Kim et al. [3] considers rate controls applied to network coding to optimize the throughput of the network.
In this section we present the study of Kim et al. who presents a rate control framework that works over COPE and increases the performance of COPE. This framework is based on two techniques: rate control and ACKer selection.

2.3.1 Rate Control

Many rate control algorithms have been proposed in wireless networks [10], [11] but they cannot be used directly with network coding. Indeed these algorithms maximize the throughput of the link between the sender and the receiver without considering the overhearing property.
Let's consider the "X" topology of Figure 2.8 to explain how a rate control algorithm should work.

The five nodes topology has been already presented, here we consider it with a rate

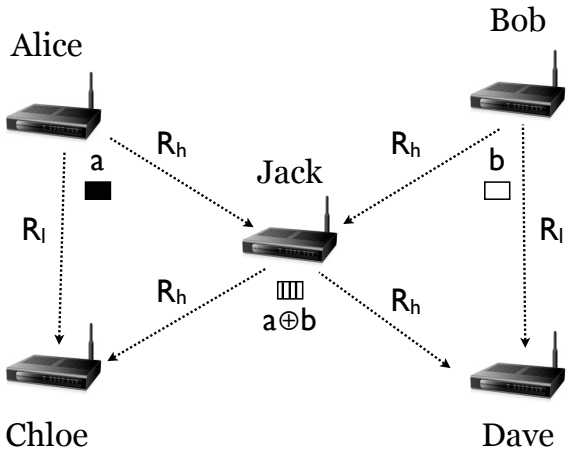


Figure 2.8: Each station uses a rate adaptation algorithm to improve network coding.

adaptation algorithm.
Alice and Bob want to transmit packets respectively to Dave and Chloe. These flows are intersecting at node Jack which makes the coding. All nodes operate in promiscuous mode and they start using the basic rate to transmit R_f. For this rate the Packet Delivery Ratio (PDR) is equal to 1 for each link. When Alice transmits packet **a** to Jack, also Dave overhears packet **a** and the same happens for Chloe with **b**. Jack applies a linear encoding function on Alice and Bob's packets ($\mathbf{a} \oplus \mathbf{b}$) and transmits this coded packet. When Chloe and Dave receive the coded packet, they decode applying the linear function to the coded and overheard packet.
If a typical rate control algorithm [10] or [11] is applied, Alice may choose a rate $R_h > R_f$ to get better performance on the link (Alice $\rightarrow$ Jack). However using the rate R_h can decrease the probability that Chloe overhears the packet and it reduces the probability of decoding successfully.

Therefore if Alice chooses to transmit at rate R_l with ($R_f < R_l < R_h$), the throughput of the link (Alice → Jack) is reduced but the throughput of the entire network may be increased since Chloe can overhear the packet and decode successfully.
So the rate control has to take care not only of the direct link (Alice → Jack) but also of indirect links (Alice → Chloe). In this case, using a lower rate brings a higher coding gain.

2.3.2 ACKer selection

We have seen that COPE uses a *pseudo-broadcast* technique to transmits XORed packets where one receiver station is chosen to be the MAC layer ACK. Hovewer both stations schedule an asynchronous ACK transmission at COPE layer for packets decoded successfully.
Kim et al. presents a trivial example to explain how to choose the ACKer selection module. He states that if the receiver nodes have different PDR, the choice of the ACKer is important.
He considers to have the same topology of Figure 2.8 and different PDR for links (Jack → Dave) and (Jack → Chloe), respectively 1 and 0.1. In this configuration, they first suppose that Chloe is the ACKer. When Chloe receives correctly the packet from Jack, also Dave has received. Therefore on average 10 transmissions are needed to deliver correctly two packets to Chloe and Dave.
However if Dave is chosen as ACKer, every time that Jack receives an ACK from Dave, it codes new packets $B_2, B_3, .., B_{10}$ with A_1 (A_i and B_i are packets for Chloe and Dave). In this case 11 packets are delivered correctly in 10 transmissions. In conclusion, choosing Dave as ACKer increases the number of packet delivery.

2.3.3 Throughput analysis

In this subsection, we report the throughput analysis presented in [3]. They consider the Figure 2.8 and the following hypothesis:

- Alice and Bob send packets **a** and **b** through the relay node Jack.
- Each rate has an associated PDR.

- A scheduling process [6] is assumed according to which packets **a** and **b** arrive to Jack before the latter transmits any of these two packets.

The probability of a successful packet delivered from Alice to Jack is equal to $P^{r_{Alice}}_{Alice,Jack} \cdot P^{r_{ACK}}_{Jack,Alice}$ where the first term is the PDR of link (Alice $\rightarrow$ Jack) at bit rate r_{Alice} and the second term is the PDR for the ACK of link (Jack $\rightarrow$ Alice).
The probability that Chloe overhears packet A is given by:

$$\begin{aligned} P^{r_{Alice}}_{(Alice,Jack),Chloe} &= \sum_{i=1}^{\infty}\{(1 - P^{r_{Alice}}_{Alice,Jack}P^{r_{ACK}}_{Jack,Alice})^{i-1} \cdot P^{r_{Alice}}_{Alice,Jack}P^{r_{ACK}}_{Jack,Alice} \\ &\quad \cdot \sum_{j=1}^{i}(1 - P^{r_{Alice}}_{Alice,Chloe})^{j-1} \cdot P^{r_{Alice}}_{Alice,Chloe}\} \end{aligned} \tag{2.3}$$

where $\sum_{j=1}^{i}(1 - P^{r_{Alice}}_{Alice,Chloe})^{j-1} \cdot P^{r_{Alice}}_{Alice,Chloe}$ is the probability that Chloe successfully overhears a packet from Alice, when the latter performs exactly i transmissions.
The total number of bits delivered to Chloe and Dave is:

$$\begin{aligned} L_t &= P^{r_{Jack}}_{(Jack,ACKer),Dave} \cdot P^{r_{Bob}}_{(Bob,Jack),Dave} \cdot L_{Alice} \\ &\quad P^{r_{Jack}}_{(Jack,ACKer),Chloe} \cdot P^{r_{Alice}}_{(Alice,Jack),Chloe} \cdot L_{Bob} \end{aligned} \tag{2.4}$$

where L_{Alice} and L_{Bob} are the length of packet that Alice and Bob send. Probabilities of links (Jack $\rightarrow$ Chloe) and (Jack $\rightarrow$ Dave) depend which node is the ACKer and it values $P^{r_x}_{(x,ACKer),y} = \sum_{i=1}^{\infty}\{(1 - P^{r_x}_{x,y}P^{r_{ACK}}_{y,x})^{i-1} \cdot P^{r_x}_{x,y}P^{r_{ACK}}_{y,x}$ if node y is the ACKer, otherwise $P^{r_x}_{(x,ACKer),y} = P^{r_x}_{(x,z),y}$ if node z is the ACKer. The expression above requires that Chloe and Dave overhear native packets.
The duration for delivering the packets from sources to the destinations is:

$$D_t = N^{r_{Alice}}_{Alice,Jack} \cdot T^{r_{Alice}}_{L_{Alice}} + N^{r_{Bob}}_{Bob,Jack} \cdot T^{r_{Bob}}_{L_{Bob}} + N^{r_{Jack}}_{Jack,ACKer} \cdot T^{r_{Jack}}_{max(L_{Alice},L_{Bob})} \tag{2.5}$$

where $N^{r_{Alice}}_{Alice,Jack} = \frac{1}{P^{r_{Alice}}_{Alice,Jack}P^{r_{ACK}}_{Jack,Alice}}$ is the number of transmissions needed for a successful one and it follows a Bernoulli process. $T^{r_{Alice}}_{L_{Alice}}$ is the transmission time of the packet with length L_{Alice} transmitted at bit rate r_{Alice}. In conclusion the expected throughput with network coding is:

$$S_{NC} = \frac{L_t}{D_t} \tag{2.6}$$

The data rates should be selected to optimize the equation (2.6). However each node should have an omniscient view of the traffic of the network but this in practice is

not possible. When Alice transmits, she cannot know if also Bob has a packet to send. If Jack has just a packet from Alice, it sends the packet when the channel is idle.
Furthermore in a more general case, with n transmitters that all use the same relay Jack, the candidate rate set is R^n. This set grows exponentially with n and finding rates that maximize the throughput becomes an NP-Hard problem[22].

2.3.4 Proposed approach

Once abandoned the idea of finding rates from the equation 2.6, Kim et al. shows that a simpler approach is possible.
First, each sender identifies all his one-hop neighbors that are also neighbors of Jack. Second, the sender chooses the rate that maximizes the expected throughput to Jack, such that the neighbors overhear the packet with a probability greater than β.
The threshold β depends on the probability of decoding of each receiver. We have seen the probability of decoding for COPE in a general case with n node (equation 2.2). Similary, when n nodes uses its own data rate, the decoding probability for the $n-th$ receiver is :

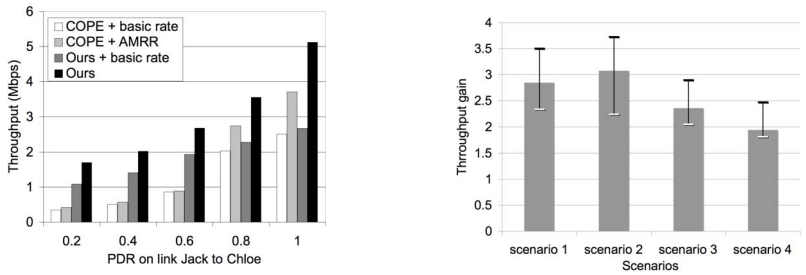
$$P_{dec(D_n)} = P^{r_{S_1}}_{(S_1,D_j),D_n} \times P^{r_{S_2}}_{(S_2,D_j),D_n} \times \cdots \times P^{r_{S_{n-1}}}_{(S_{n-1},D_j),D_n} \tag{2.7}$$

where $P^{r_{S_k}}_{(S_k,D_j),D_n}$ is the probability of overhearing the packet transmitted at rate r_{S_k} from the sender k to Jack.
In COPE the probability of decoding has to be greater than a threshold $\theta = 0.8$. At the same way, Kim et al. set $\theta = \beta^n$ and from the observation that $P_{dec(D_n)}$ is no greater than the minimum of probabilities $P^{r_{S_k}}_{(S_k,D_j),D_n}$, they require that each term be at least β.
Finally, the problem of rate control for each source (e.g. Alice) can be translate in the following optimization problem:

$$\begin{aligned} \max_{r_{Alice} \in R} \quad & \frac{L_{Alice}}{N^{r_{Alice}}_{Alice,Jack} \cdot T^{r_{Alice}}_{L_{Alice}}} \\ s.t \quad & P^{r_{Alice}}_{(Alice,Jack),Chloe} \geq \beta \\ & P^{r_{Alice}}_{(Alice,Jack),Bob} \geq \beta \end{aligned} \tag{2.8}$$

2.3.5 Experiments and Results

The framework has been tested in simulation and implemented in testbed. They have done different simulation with different topologies and with diverse traffic characteristics. They show that their framework performs better of COPE in different scenarios. In Figure 2.9 is reported some results from simulations and testbed.



(a) (b)

Figure 2.9: (a) throughput from simulation with different PDR, (b) throughput from testbed with different scenarios.[3]

In Figure 2.9(a) they performs simulation on an "X" topology 2.8 with different PDR on link (Jack $\rightarrow$ Chloe). They compare throughputs of four cases. COPE by default uses basic transmission rate (6 Mbps). They simulate also COPE with the AMRR rate adaptation algorithm [10] and their framework with and without rate adaptation. In cases with small PDR, rate unawareness can impact severely the performance.
In Figure 2.9(b) they show results from implementation on their framework on real hardware. They experiments in four different scenario: "X" topology with high quality links, cross topology with high quality links and both configurations with low quality links. Also in these cases they performs better than COPE.

Chapter 3

IEEE 802.11 Wireless Networks

Wireless communications are every year more popular and indispensable. They provide access to the Internet and Mobile Communication to different types of devices.
Over the past few years the number of smartphones and tablets has grown. These devices require more data traffic than common phones. Also the number of laptops that are connecting on Internet through wireless communications is increasing. People want to access the Internet everywhere with different devices and at higher rate. The challenge is to create communication systems that are more reliable and have higher data rates.
The wireless channel puts fundamental limitations to the performance of devices due to collisions and errors. We will give an explanation of channel errors in Chapter 4; in this chapter we focus on the collision problem and how devices access the channel using protocols, in particular the IEEE 802.11 family.
The wireless channel is shared by similar devices which are within transmission range and belong to the same network. When a device wants to transmit some information, it has to access the wireless channel. While the first device is transmitting, other stations have to wait. If two or more devices access the channel in the same time slot, they collide and have to retransmit their information.
However, different wireless devices can transmit at the same time if they are using different frequencies. For example, we can listen to the radio while making a phone call. Indeed, the radio and mobile phone use different carrier frequencies.
Similar devices use the same frequencies if they belong to the same network. For example, the Wi-Fi network in the 2.4 GHz band is divided into 13 channels spaced 5 MHz apart. If two laptops are connected to the same router, they use the same Wi-Fi

channel and they share a protocol to access the channel. A communication protocol is a set of shared rules to permit communications with the same type of devices. These rules describe all parts of the communication (signaling, message format, authentication, error detection, etc.).

3.1 Wireless Channel Access Protocols

The simplest protocol to access the channel is called ALOHA (now "Pure ALOHA") and it was presented in 1970. The idea of the protocol is simple: when a node has data to send, it sends it. If the message collides with another transmission, the node will resend after a certain amount of time depending of its back off. When the traffic load goes over a threshold, the throughput decreases exponentially.
A slightly better implementation of ALOHA is Slotted ALOHA. It introduces time slots, so a node can send only at the beginning of the time slot. This protocol reduces collisions and increases throughput with respect to Pure ALOHA.
These protocols still have a problem: they don't sense the channel before transmitting. Carrier Sense Multiple Access (CSMA) protocol introduces this feature: the idea is to "listen before talk". A node senses the channel before transmitting. If the channel is free, the node sends the packet, otherwise it reschedules the transmission by generating a random time.
CSMA is a probabilistic Media Access Control (MAC) protocol and there are different versions of it. The case that we just presented is called *nonpersistent* CSMA. There are other two versions: *1-persistent* CSMA and *p-persistent* CSMA.
In the *1-persistent* CSMA, if the channel is busy, the node sends the packet immediately after the channel becomes idle. In this case, if there is another node which is waiting to transmit, the two nodes will collide when the channel becomes idle.
The *p-persistent* CSMA doesn't have this problem. In this case, when the medium becomes idle, each station transmits with a probability p at the beginning of each slot and they don't transmit with a probability *1-p*. This process continues until the packet is sent. The *p-persistent* CSMA is used in CSMA/CA (where CA means *Collision Avoidance*) system including IEEE 802.11.

3.2 IEEE 802.11

IEEE 802.11 is a standard for Wireless Local Area Networks (WLANs). It was proposed for the first time in 1997 [17]. The standard defines two operational modes for WLANs: *infrastructure-based* and *infrastructure-less* or *ad-hoc*.
A group of corresponding stations is called a *Basic Service Set* (BSS). In the *infrastructure-based* the BSS is organized around an Access Point (AP). The AP provides to other nodes wireless access to the Internet. Members of the BSS talk to the AP only. When the cost associated to the infrastructure is high, or when it is not possible to use the infrastructure, for example after a disaster, an *ad-hoc* network can be the right solution. In this operational mode, there isn't an AP and members of the BSS talk between themselves directly. The collection of stations that are able to communicate with each other directly is usually called *Indipendent Basic Service Set* (IBSS).
The 802.11 standard provides detailed Medium Access Control (MAC) and Physical Layer (PHY) specification for WLAN.

MAC

The MAC Layer offers two different types of service: the *Distributed Coordination Function* (DCF) and the *Point Cordination Function* (PCF).
DCF is a random access scheme. It is based on the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol.
PCF is a centralized MAC protocol able to support contention-free (CF) and time bounded services. In this chapter we just consider the DCF protocol. Information about other protocols can be found in the IEEE standard [17].

PHY

The physical layer can be implemented using four different technologies: Infrared (IR), Orthogonal Frequency Division Multiplexing (OFDM), Frequency Hopping Spread Spectrum(FHSS) and Direct Sequence Spread Spectrum (DSSS).

The 802.11 Family

Today the 802.11 family consists of four different versions, denominated with letters (a, b, g and n). These versions have different modulations and data rates. 802.11a uses

the 5GHz frequency, other versions use the 2.4GHz band. 802.11n can use both 2.4 GHz and 5GHz.
802.11n, which is the most recent protocol, can reach a maximum data rate of 600 Mbit/s using Multiple Input Multiple Output (MIMO) and a larger bandwidth (40 MHz instead of 20 MHz). A brief summary of these versions is reported in Table 3.1. 802.11g operates in the same frequency band as 802.11b, and it is required to remain backwards-compatible. So 802.11g can use also the DSSS modulation to communicate with 802.11b stations.

Table 3.1: Parameters of IEEE 802.11 standard

802.11	a	b	g	n
Release	1999	1999	2003	2009
Freq. (GHz)	5	2.4	2.4	2.4 or 5
Band. (MHz)	20	20	20	20 or 40
Max Data Rate	54	11	54	600
MIMO	1	1	1	4
Modulation	OFDM	DSSS	DSSS, OFDM	OFDM

3.2.1 Protocol Description - DCF

The *Basic Access* mechanism for packet transmission is a two-way handshaking. When a station transmits a packet, it receives immediately an acknowledgment (ACK) from the receiver station. If the ACK is positive, the sender station considers the packet as received and moves forward transmitting other packets. Explicit transmission of an ACK is required since transmitter cannot determine if the packet is received correctly or not.

DIFS

DCF, as we have seen, uses a CSMA/CA protocol. Before transmitting a packet, the station senses the channel to determine the channel status. If the medium is sensed idle for a Distributed InterFrame Space (DIFS) time, the station transmits. Otherwise, if the medium is busy, the station waits until the channel is sensed idle for a DIFS time.

After this, a random backoff time is generated and it starts decreasing the backoff counter at the beginning of each free Slot Time. This feature of the protocol is called Collision Avoidance and is used to reduce the probability that two or more stations transmit at the same time. To avoid that a station captures the channel, after each successful transmission a new random backoff time is generated.
Figure 3.1 reports an example of Basic Access mechanism. It represents the axis time for two stations, A and B. After station B has finished to transmit a packet, it generates a random backoff time (in this case 8). Station B starts to count down its backoff, but station A, after sensing the channel idle for DIFS, transmits its packet. B freezes its counter and resumes its after the channel is sensed idle for a DIFS. Finally, when its counter reaches 0, it transmits a new packet.

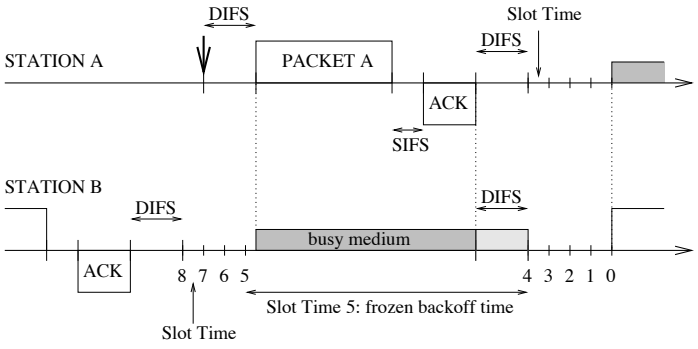


Figure 3.1: Example of Basic Access Mechanism from [1]

SIFS and EIFS

When a station receives a packet, it senses the channel idle for at least a Short Inter-Frame Space (SIFS) and sends the ACK. The SIFS is shorter than a DIFS thus the ACK can be sent immediately after receiving a packet (DIFS = SIFS + 2 x Slottime). Since SIFS plus the propagation delay is shorter than a DIFS, no other stations can sense the channel idle for a DIFS, so ACK cannot collide with other transmission. The ACK is not transmitted if the received packet is corrupted. A Cyclic Redundancy Check (CRC) algorithm is used for error detection. The probability that the CRC doesn't find the corruption is very low and it is often negligible.
When a collision or transmission error happens, the sender station doesn't access the channel for at least an Extend InterFrame Space (EIFS) interval. The relationship

between InterFrame Space is: EIFS $\geq$ DIFS $\geq$ SIFS.

Contention Window (CW)

DCF implements a discrete-time backoff scale that keeps stations syncronized. As we have seen, the backoff counter decreases at the beginning of each slot. The Time Slot length (σ) depends on the physical layer and is reported in Table 3.2.
Let's see in more detail the backoff scheme adopted by DCF. After the channel is sensed idle for a DIFS time, the backoff time is uniformly chosen in the range $(0, w-1)$ where w is the Contention Window. When a node sends a new packet, w assumes the minimum value of contention window (CW_{min}). When the transmission fails, the value of w doubles. w can reach a maximum value equal to CW_{max}. CW_{min} and CW_{max} values depend on the physical layer and are reported in Table 3.2.

PHY	FHSS	DSSS	IR	OFDM (802.11a)	OFDM (802.11g)
Slot Time(σ)	50 μs	20 μs	8 μs	9 μs	9 μs
DIFS	128 μs	50 μs	26 μs	34 μs	28 μs
SIFS	28 μs	10 μs	10 μs	16 μs	10 μs
CW_{min}	15	31	63	31	31
CW_{max}	1023	1023	1023	1023	1023

Table 3.2: Timing for the three PHY specified by the 802.11 standard. For the OFDM modulation there are two values for the SIFS. We remember that the 802.11g uses also the DSSS modulation. These values are taken from the standard [17].

RTS-CTS

In addition to the *Basic Access* scheme, a *Request-to-Send/Clear-to-Send* mechanism can be used. In the *Basic Access* mechanism, the sender stations know that a packet collides after a time that depends on the length of the packets.
The RTS/CTS mechanism aims to reduce the duration of a collision. Furthermore, this mechanism reduces the problem of hidden terminals, which occurs when two stations are in the opposite direction of a common destination station and they cannot hear each other when they transmit. The RTS/CTS mechanism is shown in Figure 3.2. When the backoff counter of a station reaches 0, a RTS is sent on the channel. This special packet

is shorter than a data packet, so it reduces the time in the case of collision. If the RTS packet is received by the destination station, this station replies with a CTS packet. When a CTS packet is received by the sender station, it proceeds sending the message. The destination and sender station wait for a SIFS before transmitting RTS/CTS.
The RTS and CTS frames contain the total duration of the transmission of the packet and the ACK. This information can be read by any listening station and can be used to set up a timer called Network Allocation Vector (NAV). The NAV information can be received also by hidden terminals and it delays its packet transmission. The RTS/CTS mechanism performs well when packets are large. It's not used when packets are under a certain size since the overhead would be greater than the effective benefit.

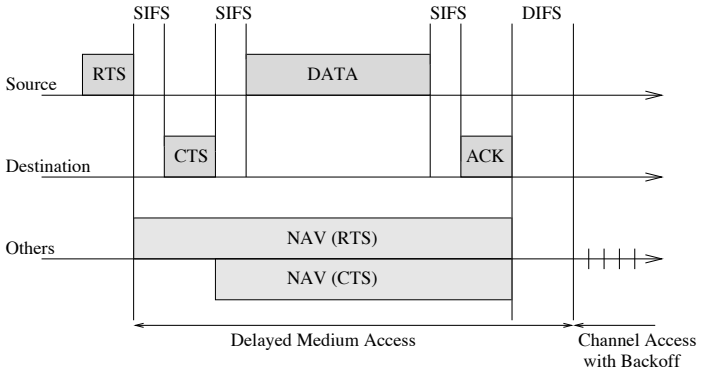


Figure 3.2: RTS/CTS Access Mechanism from [1]

3.2.2 Analytical Evaluation of the Saturation Throughput

In this section we present Bianchi's analysis of saturation throughput of 802.11 wireless network [1].
For the analysis Bianchi makes this assumptions:

- ideal channel conditions (no hidden terminals and capture)
- fixed number of stations n
- each station has a packet available for transmission (*saturation* conditions)
- p is the conditional collision probability and is assumed constant.
- discrete time that represents time slots

We want to find the probability that a station transmits in a randomly chosen slot time. This depends on the backoff model. Bianchi shows that the backoff window size can be represented with a discrete-time Markov Chain (Figure 3.3). Each state in the Markov chain counts for two stochastic processes $\{s(t), b(t)\}$.
The first stochastic process $s(t)$ represents the backoff stage. The backoff stage i is the number of retransmissions for the same packet and it is defined by $W_i = 2^i W$, where $i \in (0, m)$ and m is the maximum value that i can assume from the relation $CW_{max} = 2^m W$.
The second stochastic process $b(t)$ represents the backoff time counter for a station. The backoff time counter k can assume values in $(0, W_i - 1)$ where i was defined above. When a station wants to transmit a packet for the first time, it chooses a random backoff counter uniformly between $(0, W_0 - 1)$.
The non zero one-step transition probabilities of the Markov Chain are:

$$\begin{cases} P\{i,k|i,k+1\} = 1 & k \in (0, W_i - 2) \quad i \in (0,m) \\ P\{0,k|i,0\} = (1-p)/W_0 & k \in (0, W_0 - 1) \quad i \in (0,m) \\ P\{i,k|i-1,0\} = p/W_i & k \in (0, W_i - 1) \quad i \in (1,m) \\ P\{m,k|m,0\} = p/W_m & k \in (0, W_m - 1) \end{cases} \tag{3.1}$$

The first equation in (3.1) describes that at the beginning of each slot the backoff counter is decremented. When station wants to transmit a new packet, the backoff stage starts from 0 and it is shown in the second equation. The third equation explains when an unsuccessful transmission occurs: the backoff stage is incremented by 1 and the following state is chosen uniformly with a probability equal to p/W_i where W_i is the contention window at the i-th backoff stage. The last equation accounts for the fact that when the contention window has reached the maximum, it cannot increase and the transition probability is p/W_m.

The stationary distribution of the chain is:

$$\pi_{i,k} = \lim_{t \to \inf} P\{s(t) = i, b(t) = k\} \quad i \in (0,m), \quad k \in (0, W_i - 1). \tag{3.2}$$

It is possible to obtain a closed-form solution of the Markov chain, all passages are reported in [1], we just report the result. The probability τ that a station transmits in a randomly chosen slot is:

$$\tau = \sum_{i=0}^{m} \pi_{i,0} = \frac{\pi_{0,0}}{1-p} = \frac{2(1-2p)}{(1-2p)(W+1) + pW(1-(2p)^m)} \tag{3.3}$$

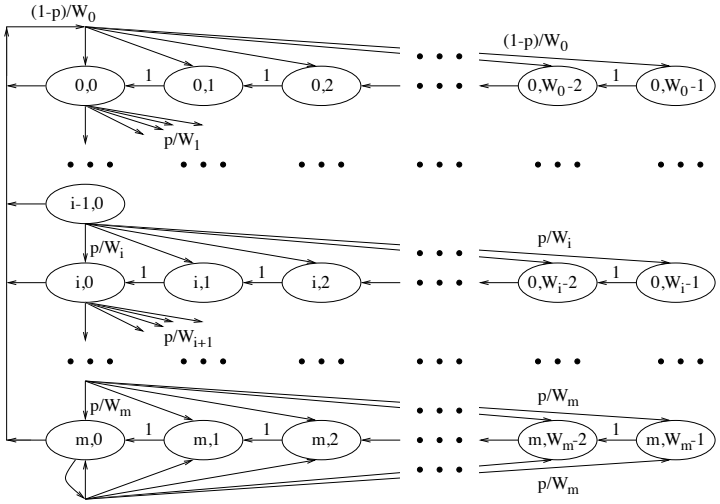


Figure 3.3: Markov Chain model for the backoff window size from [1]

The probability p that a packet collides is equal to the probability that at least one of $n-1$ stations transmit in the same slot and is:

$$p = 1 - (1-\tau)^{n-1} \tag{3.4}$$

with the assumption of independence between stations.

Throughput

The probability of a successful transmission is equal to the probability that just one station transmits:

$$P_s = \frac{P[1\ station\ TX]}{P[\geq 1\ station\ TX]} = \frac{n\tau(1-\tau)^{n-1}}{1-(1-\tau)^n}. \tag{3.5}$$

The throughput S, defined as the fraction of time the channel is used for successful transmissions, is equal to:

$$S = \frac{E[payload\ TX\ in\ a\ time\ slot]}{E[length\ of\ a\ slot\ time]} = \frac{P_s P_{tr} E[P]}{(1-P_{tr})\sigma + P_{tr}P_s T_s + P_{tr}(1-P_s)T_c} \tag{3.6}$$

where at the numerator P_s is the probability of successful transmission, P_{tr} is the probability that at least one station transmits in that slot and it is equal to $P_{tr} = 1 - (1-\tau)^n$ and $E[P]$ is the average packet payload size. The denominator of equation (3.6) is composed by three factors. The first $(1-P_{tr})\sigma$ accounts for the time wasted without any transmissions, the second $P_{tr}P_sT_s$ describes the time for a successful transmission

and the last term accounts for the time wasted by collisions.
Let's see in more details the time T_s for a successful packet transmission and the time T_c wasted for a collision presented in theFigure 3.4.

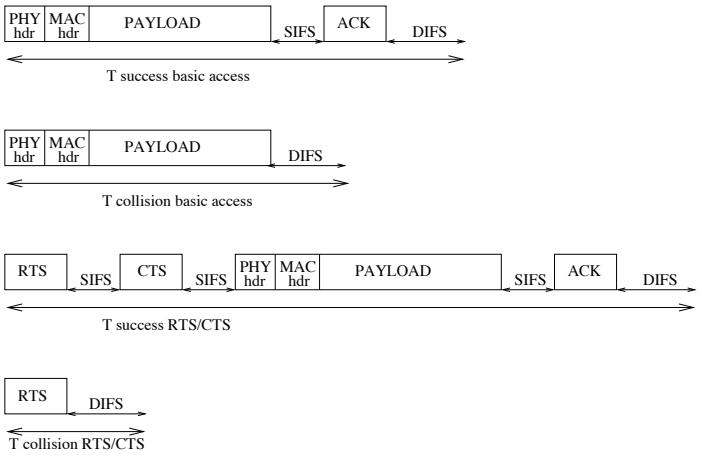


Figure 3.4: T_s adn T_c for Basic Access and RTS/CTS mechanism from [1]

For the Basic Access mechanism :

$$\begin{cases} T_s^{bas} = H + E[P] + SIFS + \delta + ACK + DIFS + \delta \\ T_c^{bas} = H + E[P^*] + DIFS + \delta & \textit{non colliding stations} \\ T_c^{bas} = H + E[P^*] + DIFS + \delta + ACK_{timeout} & \textit{colliding stations} \end{cases} \quad (3.7)$$

where δ is the propagation time and is usually assumed equal to 1 μs, $E[P]$ is the average time spent transmitting a packet and $E[P^*]$ is the average time of the longest payload involved in a collision. If it is assumed for simplicity that all packets have the same length $E[P^*] = E[P] = P$. The time of collision for colliding stations reported in equation (3.7) accounts for the fact that stations before sensing the channel again need to wait for an ACK timeout.
In the case of RTS/CTS mechanism:

$$\begin{cases} T_s^{rts} = RTS + 3SIFS + 4\delta + CTS + H + E[P] + ACK + DIFS \\ T_c^{rts} = RTS + DIFS + \delta & \textit{non colliding stations} \\ T_c^{rts} = RTS + DIFS + \delta + CTS_{timeout} & \textit{colliding stations} \end{cases} \quad (3.8)$$

Equation (3.8) shows clearly that there is more overhead to transmit with RTS/CTS instead of Basic Access. When packets are colliding, the time wasted is just the RTS time.

Colliding stations need to wait for a $CTS_{timeout}$ that it is similar to the $ACK_{timeout}$. Now we have all elements to compute the throughput in equation (3.6). Bianchi's analysis is really useful to understand the performance of wireless networks. However this analysis assumes that all stations use the same rate and the same probability accessing the channel. In the next section we will see what happens when a station uses a different Data Rate.

3.3 Multi-Rate

Multi-Rate is the ability of a station to automatically operate at several different bitrates. A station changes its rate depending on the channel quality. When the distance between two hosts increases, the channel quality decreases. Thus, using a lower rate keeps the Packet Error Rate (PER) low.
Each Data Rate uses a different Modulation (BPSK,QPSK,16-QAM,64-QAM) and a different Coding Rate ($1/2, 2/3, 3/4$). A lower rate uses a modulation (e.g BPSK) where there is a greater distance between adjacent points of the constellation and consequently the error is lower than other modulations. Table 3.3 shows which rates are used by 802.11 family.

	PHY Data Rate (Mb/s)
802.11b	1, 2, 5.5, 11
802.11g	6, 9, 12, 18, 24, 36, 48, 54
802.11a	6, 9, 12, 18, 24, 36, 48, 54

Table 3.3: PHY Data Rate for the 802.11 standard.

3.3.1 Rate Adaptation

We have said that rates change depending on the channel quality. Several algorithms have been proposed to adapt rates, here we consider: ARF, AARF and SNR-based.

ARF

Auto Rate Fall-back [13] is a simple rate adaptation algorithm. The idea is that a station increases its PHY Rate when a certain number of transmission successes occur consecutively and it switches back to a lower rate after 1 or 2 consecutive failures.

However, this algorithm suffers when the channel conditions change very quickly since ARF cannot adapt effectively and when the channel conditions do not change at all since ARF would try to increase the rate. An interesting throughput analysis of ARF in a lossy channel is presented by Yun in [14].

AARF

Adaptive Auto Rate Fall-back is the evolution of ARF. When the channel conditions do not change, ARF is not able to stabilize for a long period and trying higher rate decreases the application throughput. Indeed, AARF changes continuously the threshold to reflect the channel conditions. When a packet fails, AARF switches back to a lower rate and multiplies by two the number of consecutive successful transmissions (maximum 50) required to switch to a higher rate. AARF uses a more conservative approach with respect to ARF and has been shown to perform better than ARF in[15].

SNR-based

The SNR-based algorithm estimates the Signal to Noise Ratio of a packet for a given wireless link. Depending on the SNR value, the algorithm chooses the best rate to optimize the application throughput. For that rate the PER is low enough such that the number of retransmissions is low. In the case of fixed stations or low mobility, this algorithm is similar to AARF. The ARF and SNR-based algorithms are included in the external library *dei802.11mr* [20] of ns2. In our simulations we use a SNR-based algorithm since we consider fixed stations.

3.3.2 Performance Anomaly

We have seen some rate adaptation algorithms, each of them try to optimize the throughput of a station varying the data rate. However, what happens if in the same transmission range there is a station that uses a lower rate with respect to the others? When a station uses a lower bit rate, it captures the channel for a long time, so the performance of all the stations is considerably degraded. This problem has been analyzed by Heusse in [16] for 802.11b.
He considered N stations transmitting with different rates, $N-1$ nodes use the high transmission rate $R = 11$Mb/s and one uses a lower rate $r = 5.5, 2$ or 1 Mb/s. He analyzed the performance of this system assuming that each node has packet to send. He showed that the throughput at MAC layer of stations transmitting with a higher

rate R is the same of the station transmitting with a lower rate r.
This is a consequence of the fairness of CSMA/CA; indeed the long term channel access probability is equal for all the stations. Therefore, when the node transmitting with a lower rate, (e.g $r = 1$Mb/s) accesses the channel, it captures the channel eleven times longer than stations transmitting at 11 Mb/s.

Chapter 4

Wireless Channel Model

In this chapter we report a brief overview of wireless channel. We show some problems related to the wireless channel and we present two models to characterize the wireless channel. In the last part of the chapter we talk about the channel model used for the simulations with ns2.

4.1 Introduction

The wireless channel puts fundamental limitations to the performance of wireless communications systems. If the transmitter or the sender moves during the transmission, the channel changes fast and the performance is reduced drastically.
The first characteristic of the wireless channel is that it attenuates the transmitted signal. Consider a signal $s(t)$ of power P_t transmitted over the channel and the received signal $r(t)$ of power P_r. The radio channel attenuation is :

$$P_a = \frac{P_t}{P_r} \tag{4.1}$$

The attenuation depends on three factors: *Path Loss*, *Shadowing* and *Multipath*.

- *Path Loss* is the main cause of signal attenuation over the distance. It is caused by the dissipation of the power radiated by the transmitter. (variations over 100 meters)
- *Shadowing* is caused by obstacles which are between the transmitter and the receiver. These obstacles attenuate the signal power through absorption, reflection, scattering, and diffraction. (variations 10-100 meters)

- *Multipath* is the effect that happens when the radio signal received is composed by signals that followed different paths. This brings to constructive and destructive interference and phase shifting of the signal. (variations of signal wavelength λ)

These three components are presented in Figure 4.1.

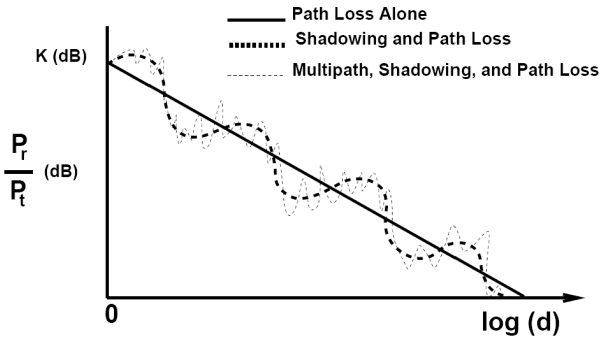


Figure 4.1: Combined effect of path loss, shadowing and multipath with increasing distance.

4.1.1 Signal Propagation

To introduce the analytical model for the wireless channel, we have first to present the analytical representation for the transmitted and received signal.
Consider a narrowband pass-band signal $s(t)$ at the carrier frequency f_c. The transmitted and received signals are real since modulators are built using oscillators that generate real sinusoids. The channel can be modeled as a linear filter with impulse response $h(t)$ (Figure 4.2).

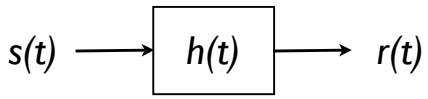


Figure 4.2: The channel can be represented as a linear filter with response $h(t)$

We can write the transmitted signal $s(t)$ as the real part of a complex signal:

$$s(t) = \Re\left\{u(t)e^{j2\pi f_c t}\right\} = \Re\{u(t)\}\cos(2\pi f_c t) - \Im\{u(t)\}\sin(2\pi f_c t) \tag{4.2}$$

where $u(t)$ is the complex envelope of $s(t)$ and f_c is the carrier frequency. $\Re\{u(t)\}$ is the in-phase component and $\Im\{u(t)\}$ is the quadrature component of the signal.

The channel changes the amplitude and the phase at each frequency of the transmitted signal and can be modeled by $h(t) = ae^{j\phi}\delta(t-\tau)$.
Therefore the received signal $r(t)$ is related to the input signal $s(t)$ and the impulse response $h(t)$ by the convolution integral:

$$r(t) = \int_{-\infty}^{\infty} h(\xi)s(t-\xi)d\xi = \int_{-\infty}^{\infty} ae^{j\phi}\delta(\xi-\tau)s(t-\xi)d\xi = ae^{j\phi}s(t-\tau). \tag{4.3}$$

In conclusion, the channel produces an attenuation, a delayed and a deterministic shift of the phase on the received signal.

4.2 Channel models

We introduce two analytical models for the wireless channel: *free-space path loss* and *two ray model*.

4.2.1 Free-Space Path Loss

Suppose a signal $s(t)$ transmitted through a free space to a receiver at distance d. We suppose that there are no obstacles between the sender and the receiver, (no *shadowing* or *multipath*). The signal that propagates from the transmitter to the receiver along a straight line, is also indicated also as *Line of Sight* (LOS) signal.
The receiver signal $r(t)$ is:

$$r(t) = \Re\left\{\frac{\sqrt{G_tG_r}e^{-j2\pi d/\lambda}}{4\pi d}u(t)e^{j2\pi f_c t}\right\} \tag{4.4}$$

where G_t and G_r are the antenna gains respectively at the transmitter and at receiver in the transmission direction. $e^{-j2\pi d/\lambda}$ is the phase shift due to the distance d which the wave travels. From equation (4.4), we can compute the ratio of the received to transmitted power:

$$\frac{P_r}{P_t} = G_tG_r\left(\frac{\lambda}{4\pi d}\right)^2 \tag{4.5}$$

Equation (4.5), also called the Friis equation, shows the relation between the path loss and the distance d. In particular, the signal power decreases proportionally to the square of the distance.
The received Power can be represented in dBm by:

$$P_r(dBm) = P_t(dBm) + 10\log_{10}(G_tG_r) + 20\log_{10}(\lambda) - 20\log_{10}(4\pi d) \tag{4.6}$$

The free-space path loss is defined as:

$$P_L(dB) = 10\log_{10}\frac{P_t}{P_r} = -10\log_{10}\frac{G_l\lambda^2}{(4\pi d)^2}. \tag{4.7}$$

4.2.2 Two Ray Path Loss

A more accurate model for the signal propagation is the two ray model. This model takes into consideration the *multipath* effect, in particular assumes that the signal at the receiver is composed by the LOS signal and a reflected component. The reflected component is the transmitted signal reflected by the ground, (Figure 4.3). The received

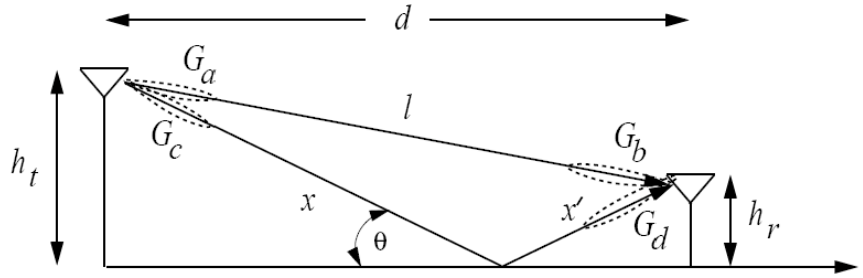


Figure 4.3: Two ray model

signal is given by the sum of the two components. Since the channel is modeled as a linear filter, we can apply superposition of the effects and find:

$$r_{2ray}(t) = \Re\left\{\frac{\lambda}{4\pi}\left[\frac{\sqrt{G_l}u(t)e^{-j2\pi l/\lambda}}{l} + \frac{R\sqrt{G_r}u(t-\tau)e^{-j2\pi(x+x')/\lambda}}{x+x'}\right]e^{j2\pi f_c t}\right\}. \tag{4.8}$$

The first component of equation (4.8) counts for the LOS signal where the wave propagates for a distance l. $\sqrt{G_l} = \sqrt{G_aG_b}$ is the product of the antenna gains in the LOS direction. The second component of equation (4.8) counts for the reflected signal. The distance in this case is equal to $x + x'$ and since $x + x' > l$, this component reaches the receiver with a time delay equal to $\tau = (x + x' - l)/c$. The $\sqrt{G_l} = \sqrt{G_cG_d}$ is the product of the antenna gains in the direction of reflected signal and R is the ground reflection coefficient.

Since the transmitted signal is narrow band relative to the delay ($\tau << B_u^{-1}$), it follows that $u(t) \approx u(t-\tau)$. From this consideration we can derive the received power:

$$P_r = P_t\left[\frac{\lambda}{4\pi}\right]^2\left|\frac{\sqrt{G_l}}{l} + \frac{R\sqrt{G_r}e^{-j\Delta\phi}}{x+x'}\right|^2. \tag{4.9}$$

where $\Delta\phi = 2\pi(x + x')$ is the phase difference between the two received signals. Furthermore, we can simplify the equation above when $d >> 0$ and assuming that, $x + x' \approx l \approx d$, $G_l \approx G_r$, $\Delta\phi \approx \frac{4\pi h_t h_r}{\lambda d}$ and $R = -1$ and we obtain:

$$P_r \approx \left[\frac{\lambda\sqrt{G_l}}{4\pi d}\right]^2 \left[\frac{4\pi h_t h_r}{\lambda d}\right]^2 P_t = \left[\frac{\sqrt{G_l} h_t h_r}{d^2}\right]^2 P_t \tag{4.10}$$

that in dBm is:

$$P_r(dBm) \approx P_t(dBm) + 10\log_{10}(\sqrt{G_l}) + 20\log_{10}(h_t h_r) - 40\log_{10}(d). \tag{4.11}$$

When the distance d between the transmitter and the receiver is large enough, from equation (4.10) we see that the received power decreases as the fourth power of the distance d.

In Figure 4.4, we plot the received power varying the distance d ($d > 10$m) and for

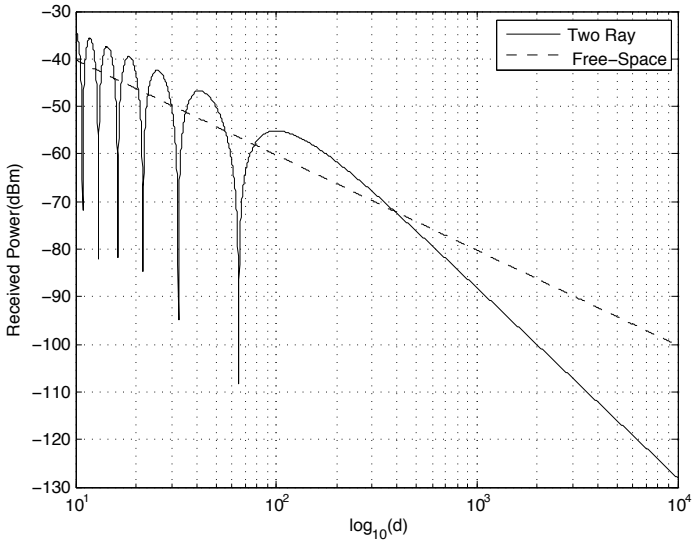


Figure 4.4: Two Ray vs Free Space

the free-space and two ray model. We use the same parameters used to simulate 802.11 stations in ns2: $f = 2437$MHz, $R = -1$, $h_r = h_t = 2$m, $G_t = Gr = 1$ and the transmit power $P_t = 0.1$W.

It is interesting to notice that with two ray model you can have a very low received power even for small distance ($10 < d < 100$) .This is caused by destructive interference of two rays.

Furthermore Figure 4.4 shows that after a distance the received power decreases faster

with two ray model than free space. This is explained by equation (4.10) in which we have seen that the power decreases with the fourth power of the distance.

4.3 Noise and Interference Contribution

The path loss is not the only source of attenuation. The received signal $r(t)$ is affected by the noise $n(t)$, so a more accurate equation is $r(t) = s(t) + n(t)$. The noise $n(t)$ is modeled as a Gaussian random process with zero mean (AWGN) and power spectral density $N_0/2$.

4.3.1 SNR and SINR

We define the Signal to Noise Ratio (SNR) as the ratio of the received power to the noise power within the bandwidth of the transmitted signal. If the bandwidth of the complex envelope $u(t)$ is denoted with B, the bandwidth of the transmitted signal $s(t)$ is 2B. From this we have:

$$SNR = \frac{P_r}{N_0 B}. \tag{4.12}$$

Until now, we had considered that there is just one sender and one receiver. When there are several transmitters, the interference of other transmissions has an important role on the quality of the received signal. We define the Signal to Interference Noise power Ratio (SINR) as the ratio of received Power to the power of the noise with the bandwidth and sum of the power of interference:

$$SINR = \frac{P_r}{N_0 B + \sum_{i \neq r} P_i}. \tag{4.13}$$

4.4 Channel model in NS 2

In this section we want just to present the channel model used for the simulations. More details about Network Simulator (NS 2) are presented in Chapter 6.
The default channel model in NS 2 is a “Black and White” model for packet reception, where two nodes are either in range (every bit is correctly received) or out of range (all bits are lost). Unfortunately this model does not allow for realistic rate selection. A typical Rate Adaptation algorithm would choose the highest possible rate in the transmission range and it would change rarely since the packet errors are just due to collisions.

For this reason, we use the external libray dei80211mr[20] which furnishes a better channel model.

4.4.1 An improved NS 2 channel model

The channel model is implemented that whenever a node receives a packet, it calculates the SINR for that packet and based on the SINR and the Transmission Data Rate it computes the Packet Error Rate (PER). The PER is calculated using pre-determined curves (PER vs SINR and packet size). Some default curves for both 802.11g and 802.11b are provided. Figure 4.5 shows the default curves for 802.11g versus experiment for 1500Byte packets for four different rates.

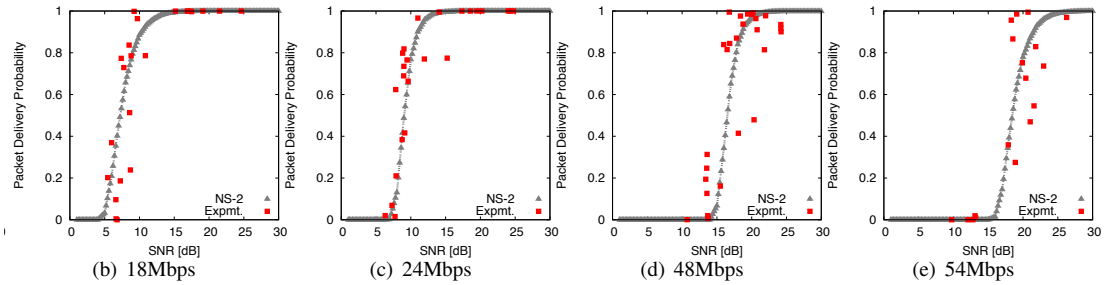


Figure 4.5: Simulations versus Experiments for 1500 Byte packets from [4]

4.4.2 Transmission Range, Interference Range and Carrier Sense Range

We have seen that the distinction between the Transmission Range, Interference Range and Carrier Sense Range is not clear in NS2 and it varies with different implementations of the channel model. In this section we want to clarify these zones.
When a node transmits a packet, receiver nodes are usually divided in three groups accordingly to the SNR received and the distance from the sender:

- *transmission range* (R): the range inside which nodes are able to receive the transmitted packet.
- *interference range* (R_i): the range inside which any new transmission may interfere with the packet reception.
- *carrier sense range* (R_s): the range inside which nodes are able to sense the signal, even though correct packet reception may not be available.

The relation between these ranges is usually $R < R_i < R_s$.
In the dei80211mr implementation R_i coincides with R_s and R varies depending on the transmission rate. In the presence of a rate adaptation algorithm, we have noticed that nodes prefer sending a packet using a low data rate to a node far away instead of sending to an intermediate node with a higher data rate. It has been shown in [31] and [32] that is better to have a sequence of two higher data rate transmissions than one with lower data rate. Moreover, in our case, we need that the transmission goes through the relay node to make the coding.
To avoid that a node chooses one single lower rate transmission instead of two higher data rate transmissions, we reintroduce the Rx Threshold in the dei80211mr implementation. This threshold permits to fix R depending on the received power (P_r). Once the transmitted power is fixed, the received power depends on the distance, so fixing the Rx Threshold is like fixing the Transmission Range.
In conclusion, we fix Rx Threshold so that:

- R is about 100 metres
- $R_i = R_s$ is 200 metres.

In [18] Deng *et al.* study the optimal tuning of the carrier sensing range in ad hoc networks and they show that tuning this parameter can improve the performance of ad hoc networks. However keeping R_s fixed and equal to R_i is reasonable and it has been adopted also in other papers like [4].

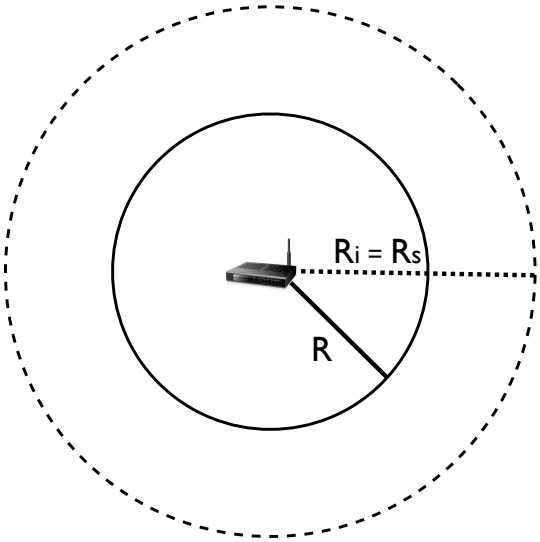


Figure 4.6: Transmission Range (R) and Interference Range $R_i = R_s$ for a wireless station.

Chapter 5

Problem Analysis and Proposed Algorithm.

We have seen in previous chapters how COPE works, the Basic Access scheme of IEEE 802.11 wireless network and wireless channel models. In this Chapter, we use these concepts for our analysis and we present our proposed queue management algorithm.
We consider the five nodes topology network presented in Figure 5.1. The five nodes are a recurrent topology in larger networks and it is often analyzed for its simplicity. We assume that there are two packet flows which intersect at the relay node. The relay node codes together packets from different flows.
First we model this scenario through a Markov chain and we compute the throughput at the relay node. This model takes into consideration nodes with different PHY data rate and shows how changing the Contention Window affects throughput.
Then, we propose a queue management algorithm to improve the throughput of the network. Other queue management has been proposed in [30].

5.1 Five nodes topology

Let's consider the five nodes topology, each node is represented by a letter in Figure 5.1. We consider two UDP flows: from A to D and from B to C. Since A can not transmit packets directly to D, it sends the packets to a relay node J and B sends packet to J, too. We suppose that nodes A and B have always packets to transmit (*saturation traffic*).
Each node can overhear packets transmitted in its transmission range and it saves over-

heard packets for a determined amount of time (usually $T = 0.5s$). Node J keeps two virtual queues with packets for C and for D. If J has packets from both virtual queues, it codes (XOR) together the native packets at the head of each queue.
However, when J has packets just for one nexthop, it transmits the native packet and loses the coding opportunity.
When a receiver node, e.g. C, receives the XORed packet, it is able to decode the packet and it extracts its native packet if it has overheard successfully the packet from A.
Since the overhearing depends on the wireless channel and it is different for each couple of nodes, node C may not overhear correctly the packet transmitted from A to J. In such a case the receiver node cannot decode the XORed packet sent by J.
We also consider that the sender nodes and the relay node use a rate adaptation algorithm to increase the capacity of the wireless network. The rate is chosen taking into account overhearing and decoding probability at the receiver nodes.

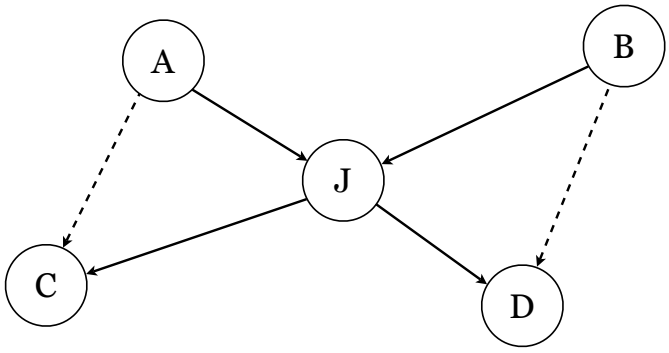


Figure 5.1: Two packet flows: A→ D, B → C. J is the relay node. C and D can overhear packets (dotted line)

Since the *Packet Error Rate* (PER) can be different for each wireless link, J can have virtual queues with different lengths. Having virtual queues unbalanced increases the probability that one of these becomes empty and J loses the coding opportunity.
Since the input traffic for node Jack is double of its output capacity, it starts accumulating packet in the queue.
For these two reasons, we studied if we can improve the capacity of this wireless network balancing the queue at the relay node J.
We aim to increase the coding opportunity and the throughout. We propose to change

the probability of accessing the channel of the sender nodes tuning the Contention Window (CW) at the MAC layer. Based on how many packets are in the virtual queues and on the PER of the links, the relay node J sends to sender nodes (A,B) the initial value of CW^i_{min} where $i \in (A, B)$ with the goal to balance its virtual queues.

5.1.1 Modeling channel access with different PHY Rates

First we consider just three nodes (A,B and J) as shown in Figure 5.2. We want to analyze channel access with different rates and we want to compute the throughput at node J.
Sender nodes (A,B) have their own Data Rate R_n which depends on the rate adaption algorithm. We assume that each station has packet to transmit (*saturation condition*).

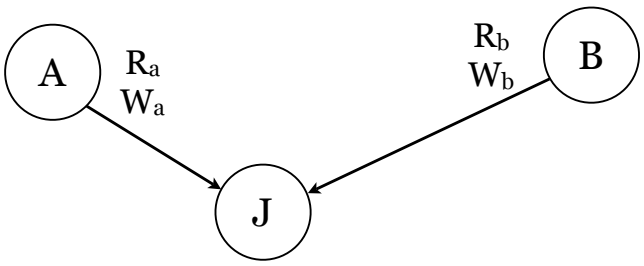


Figure 5.2: Two packet flows: A→ J, B → J. R_n is the data rate of node n, W_n is the Minimum Contention Window.

We use CSMA/CA as in IEEE 802.11 to characterize the channel access. After a node senses the channel idle for a DIFS, it chooses uniformly a backoff counter i in a Contention Window $[0, W_n - 1]$ with $n = (A, B)$.
To keep the model simple, the Contention Window is not doubled after a failed transmission but it is constant and equal to the Minimum Contention Window. We will show in the simulation that the hypothesis of fixing the Contention Window is a good approximation since there are only two nodes transmitting.

Let's consider the example shown in Figure 5.3: A has just finished to transmit a packet and it has another one to transmit. It chooses a random backoff time to wait to access the channel for the new transmission. When the channel is IDLE, node A

can decrease its backoff ($f_{a,i}$); when node B transmits, the channel is sensed BUSY and node A freezes its backoff counter($t_{xb,i}$). It restarts to countdown after B finishes the transmission and the channel is sensed IDLE ($f_{a,i'}$ with the backoff counter $i' < i$). When the backoff counter reaches 0, node A transmits its packet on the channel.
Figure 5.3 shows two different time axis depending on the rate of node B; when node B uses a lower rate than A ($R_b < R_a$), station A has to wait more time to transmit the new packet. The throughput of node A is reduced because the channel is occupied more time by station B.

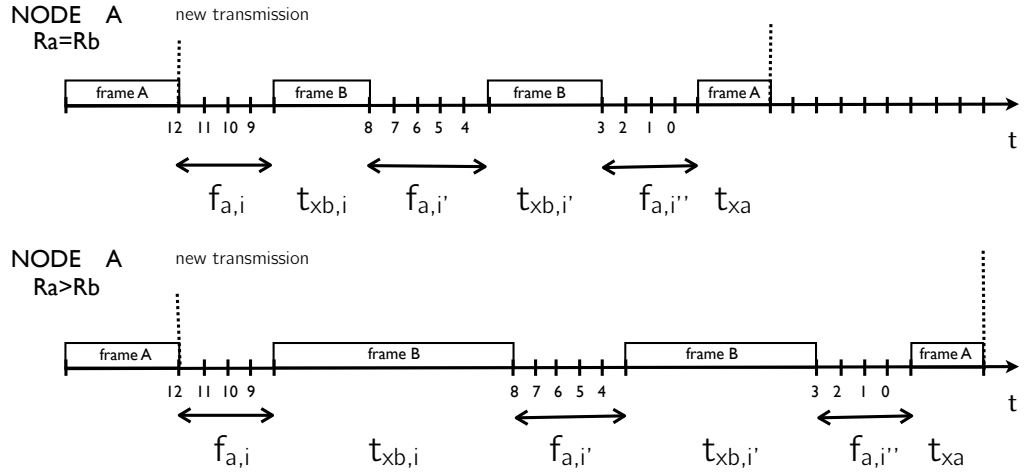


Figure 5.3: Time axis of node A. In the first case if the rate of node B is equal to the rate of node A. In the second case with different rate.

The discrete Markov Chain in Figure 5.4 wants to represent the example above. The discrete time of the chain is equal to the slot time T of the backoff counter.

Markov chain states

The state t_{xa} represents the time spent by node A transmitting a new packet and its relative ACK. The time spent in t_{xa} depends on the length of the packet and on the transmission data rate. We model this time with a geometric variable of parameter K_a (it is represented in the Markov chain with a loop).
The station chooses a new backoff counter when it has new packet. It chooses with the same probability K_a/W_a one of the state of the backoff counter.

After choosing the backoff counter, node A senses the channel at the beginning of each slot and if the channel is idle, it decrements its backoff counter. This is represented by the transition probabilities $1-p$ where p is the probability that node B transmits in a free slot.
With a probability p it goes to state $t_{xb,i}$ that represents that node B transmits while the backoff counter of A is i. Also the time spent in $t_{xb,i}$ is modeled with a geometric variable of parameter K_b.
When node B has finished to occupy the channel, station A goes back sensing the channel and counting down the backoff. When the backoff counter reaches 0, there is a successful transmission with probability $1-p$ and there is a collision with p. The packets collision is represented by state C. The time spent in this state is a geometric time with parameters K_c and it depends on the EIFS and on the transmitted packet.

Modeling time spent in states with a geometric variable

To model the time spent in states t_{xa}, $t_{xb,i}$ and C, we use geometric variables of parameter respectively K_a, K_b and K_c. These parameters depend on the length of the packet and on the transmission data rate.
In the following equations, we report the time spent respectively to transmit the payload $T^i_{payload}$, the packet T^i_{pck}, the packet and the ack T^a_{frame} and the time for a collision $T_{collision}$ considering a Basic Acces mechanism:

$$\begin{cases} T^i_{payload} = E[L_{payload}]/R_i & i = \{a,b\} \\ T^i_{pck} = L_H/R_{basic} + T^i_{payload} + SIFS + \delta & i = \{a,b\} \\ T^i_{frame} = T^i_{pck} + L_{ACK}/R_{basic} + DIFS + \delta & i = \{a,b\} \\ T_{collision} = EIFS + T_{slot} + T^a_{pck} \end{cases} \tag{5.1}$$

where L_H and L_{ACK} are respectively the length of the packet and of the ACK, R_{basic} is the lower rate that is used to transmit the header of the packet and of the ACK and δ is the propagation time.
From equations (5.1), we can derive the value of K_i with $i = \{a,b\}$:

$$\begin{cases} K_i = \frac{T_{slot}}{T^i_{frame}} & i = \{a,b\} \\ K_c = \frac{T_{slot}}{T_{collision}} \end{cases} \tag{5.2}$$

Stationary distribution of the chain

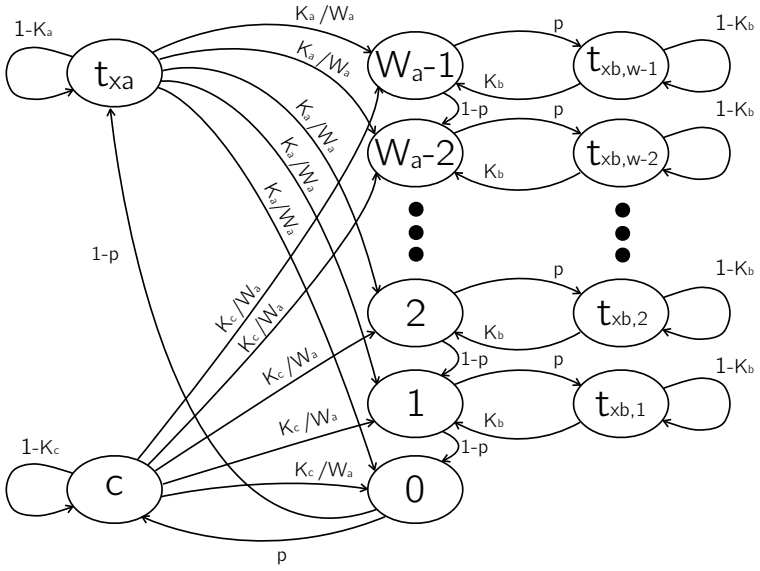


Figure 5.4: Markov chain modeling a packet transmission of node A.

Let's find the stationary distribution of the Markov chain.
First, we write all stationary probabilities as a function of π_0:

$$\pi_{t_{xa}} = \frac{1-p}{K_a}\pi_0 \tag{5.3}$$

$$\pi_c = \frac{p\pi_0}{K_c} \tag{5.4}$$

$$\pi_{W-1} = \frac{K_a}{W_a}\pi_{t_{xa}} + \frac{K_c}{W_a}\pi_c + K_b\pi_{t_{xb,W_a-1}} \tag{5.5}$$

$$\pi_{t_{xb,W_a-1}} = p\pi_{W_a-1} + (1-K_b)\pi_{t_{xb,W_a-1}} \tag{5.6}$$

Using equations (5.5) and (5.6), we derive the following equation:

$$\pi_{t_{xb,W_a-1}} = \frac{p}{K_b}\pi_{W_a-1} \tag{5.7}$$

Now we substitute equations (5.3), (5.4) and (5.7) in equation (5.5) and we find:

$$\pi_{W_a-1} = \frac{(1-p)}{W_a}\pi_0 + \frac{p}{W_a}\pi_0 + p\pi_{W_a-1} \qquad \pi_{W_a-1} = \frac{1}{1-p}\frac{1}{W_a}\pi_0 \tag{5.8}$$

and

$$\pi_{t_{xb,W_a-1}} = \frac{1}{1-p}\frac{1}{W_a}\frac{p}{K_b}\pi_0 \tag{5.9}$$

We can now consider $i \in (1, W_a - 2)$ and we have these equations:

$$\pi_{t_{xb,i}} = \frac{p}{K_b}\pi_i \tag{5.10}$$

$$\pi_i = \frac{K_a}{W_a}\pi_{t_{xa}} + \frac{K_c}{W_a}\pi_c + K_b\pi_{t_{xb,i}} + (1-p)\pi_{i+1} \tag{5.11}$$

Equation (5.11) can be simplified using (5.3),(5.4) and (5.10):

$$\pi_i = \frac{1}{(1-p)W_a}\pi_0 + \pi_{i+1} \tag{5.12}$$

Before using the normalization condition we observe that:

$$\begin{aligned}\sum_{i=1}^{W_a-2}\pi_i &= \sum_{i=1}^{W_a-2}\frac{1}{(1-p)W_a}\pi_0 + \pi_{i+1} = (W_a-2)\pi_{W_a-1} + \frac{(W_a-2)(W_a-1)}{2}\pi_{W_a-1} \\ &= \frac{(W_a-2)(W_a+1)}{2}\pi_{W_a-1} = \frac{(W_a-2)(W_a+1)}{2}\frac{1}{1-p}\frac{1}{W_a}\pi_0 \end{aligned} \tag{5.13}$$

Now we impose the normalization condition and we obtain:

$$\begin{aligned} 1 &= \pi_{t_{xa}} + \pi_c + \pi_{W_a-1} + \pi_{t_{xb,W_a-1}} + \sum_{i=1}^{W_a-2}\pi_i + \sum_{i=1}^{W_a-2}\pi_{t_{xb,i}} + \pi_0 \\ &= \pi_{t_{xa}} + \pi_c + \pi_{W_a-1} + \pi_{t_{xb,W_a-1}} + (1+\frac{p}{K_b})\sum_{i=1}^{W_a-2}\pi_i + \pi_0 \\ &= \frac{1-p}{K_a}\pi_0 + \frac{p\pi_0}{K_c} + (1+\frac{p}{K_b})\frac{1}{1-p}\frac{1}{W_a}\pi_0 + \\ &\quad +(1+\frac{p}{K_b})\frac{(W_a-2)(W_a+1)}{2}\frac{1}{1-p}\frac{1}{W_a}\pi_0 + \pi_0 \\ &= (\frac{1-p}{K_a} + \frac{p}{K_c} + 1 + (1+\frac{p}{K_b})\frac{1}{1-p}\frac{1}{W_a}\frac{(W_a-2)(W_a+1)+2}{2})\pi_0 \end{aligned} \tag{5.14}$$

finally we can write that

$$\pi_0 = \frac{1}{(\frac{1-p}{K_a} + \frac{p}{K_c} + 1 + (1+\frac{p}{K_b})\frac{1}{1-p}\frac{1}{W_a}\frac{(W_a-2)(W_a+1)+2}{2})} \tag{5.15}$$

and finally we obtain:

$$\begin{aligned}\pi_{t_{xa}} &= \frac{1-p}{K_a}\pi_0 \\ &= \frac{1-p}{K_a}\frac{1}{(\frac{1-p}{K_a} + \frac{p}{K_c} + 1 + (1+\frac{p}{K_b})\frac{1}{1-p}\frac{1}{W_a}\frac{(W_a-2)(W_a+1)+2}{2})}\end{aligned} \tag{5.16}$$

Equation (5.16) represents the percentage of time that the channel is used by station A. We can notice that this equation depends on the contention window W_a, the probability

p that the other station transmits and the time spent in transmission or collision defined by K_a, K_b, K_c.
To find a numerical solution for $\pi_{t_{xa}}$ we have to consider a symmetric Markov chain for station B. These two models are related by the probability p. For Markov chain of node A, p is the probability that node B transmits when the channel is free. The relation between these two models is given by equation (5.17):

$$p = \frac{\pi_0^B}{\sum_i \pi_i^B} \tag{5.17}$$

where π_0^B is the state representing the backoff counter reaches 0 for station B and $\sum_i \pi_i^B$ represent the time that the channel is idle.

Throughput

We can now compute the throughput at node J of the flows from A and B as follows:

$$S^i_{model} = \pi_{t_{xi}} \frac{T^i_{payload}}{T^i_{frame}} R_i \qquad i = \{a, b\} \tag{5.18}$$

$\pi_{t_{xa}}$ represents the fraction of time spent for the transmission included the ACK. We multiply this by the factor $\frac{T^i_{payload}}{T^i_{frame}}$ to find the fraction of time spent for the transmission of the packet. Finally, we multiply this by rate R_i to obtain the throughput.

5.1.2 Simulations to test the model

We consider three nodes A,B and J. We compare the throughput S^i_{model} that we obtain from equation (5.18) with the throughput $S^i_{simulation}$ of the simulations with NS 2. We use a rate adaptation algorithm SNR-based and we set to have a $PER < 0.01$. In our model we do not consider channel errors. For the model and for the simulations we use the parameters presented in Table 5.1.

Since we use the IEEE 802.11g, the PHY_{header} and the time to transmit a packet depends on the PHY rate. We use equation (5.19) to compute the time to transmit a packet. Table 5.2 shows the time spent to transmit a packet of 1500 Bytes for different PHY rates.

$$T_{pck} = OFDM_{preamble} + \lceil ((OFDM_{ServiceTail} + L_{pck})/nBits) \rceil * OFDM_{SymbDuration} \tag{5.19}$$

Name and Symbol	Value
Payload Size ($L_{payload}$)	12000bit
UDP Header (L_{UDP})	160bit
MAC Header (L_{MAC})	272bit
Ack Size (L_{ACK})	112bit
ACKtimeout	300 μs
Propagation Delay (δ)	1 μs
Time slot (T)	9 μs
DIFS	28 μs
SIFS	10 μs
CW_{max}	1023
$\text{OFDM}_{ServiceTail}$	22bit
$\text{OFDM}_{preamble}$	26 μs
$\text{OFDM}_{SymbDuration}$	4μs

Table 5.1: Parameters used for the model and for simulations in NS 2, some of them are from 802.11g standard.

PHY Rate	nBits	T_{pck}
6 Mb/s	24 bit	2102 μs
9 Mb/s	36 bit	1410 μs
12 Mb/s	48 bit	1066 μs
18 Mb/s	72 bit	718 μs
24 Mb/s	96 bit	546 μs
36 Mb/s	144 bit	374 μs
48 Mb/s	192 bit	286 μs
54 Mb/s	216 bit	258 μs

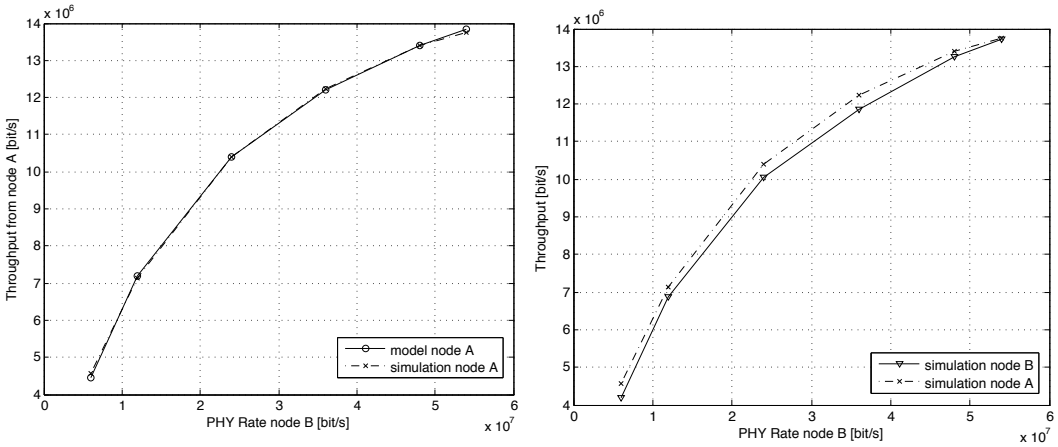
Table 5.2: For each PHY rate of 802.11g the nBits used for the relative modulation and the time T_{pck} to transmit a packet of 1554 Bytes are reported.

First scenario

We consider the PHY rate of node A fixed to R_a = 54Mb/s and PHY rate of node B varies R_b=[6,12, 24, 36, 48, 54] MB/s. Rates 9 Mb/s and 18 Mb/s have not been selected during the simulation and they have not been considered. Both nodes use a minimum contention window of $CW_{min} = 31$. Each simulation last 100 seconds and it is averaged over 5 different simulations.
We compare the NS 2 simulation with the analysis in Figure 5.5 (a). The error between the model and the simulation is at maximum 5%.
When the same value of $CW_{min} = 31$ is used for both nodes and the channel is error free, the throughput of both nodes is almost the same as you can see in Figure 5.5 (b). Even if station A uses a rate R_a=54Mb/s, its throughput is reduced to almost 4.5 Mb/s when station B uses R_b=6Mb/s. This is because station B occupies the channel most of the time. Indeed, as you can see in Figure 5.6, when station B uses R_b=6Mb/s it occupies the channel for 80% of the time.
Figure 5.6 shows the percentage of time spent in each state of the Markov chain. From this graph we can deduce that the channel is occupied most of the time by the station that uses a lower rate (in this case $\pi_{t_{xb}}$). When both nodes use a data rate $R_a = R_b = 54$Mb/s the utilization of the channel is the same.
Furthermore, we can see that the time spent for collisions π_c is around 5%. This is because both stations use $CW_{min} = 31$ which reduces the collision probability.
The time spent in one backoff stage ($\sum_{i=0}^{W-1} \pi_i$) counts more in percentage when both nodes are using a higher rate.

R_b	$S^a_{simulation}$	S^a_{model}	error^a	$S^b_{simulation}$	S^b_{model}	error^b
6 Mb/s	4.557 Mb/s	4.450 Mb/s	2.34%	4.196 Mb/s	4.415 Mb/s	5.22%
12 Mb/s	7.141 Mb/s	7.191 Mb/s	0.7 %	6.864 Mb/s	7.1342 Mb/s	3.93%
24 Mb/s	10.387 Mb/s	10.409 Mb/s	0.22 %	10.053 Mb/s	10.327 Mb/s	2.72%
36 Mb/s	12.226 Mb/s	12.218 Mb/s	0.06%	11.853 Mb/s	12.218 Mb/s	3.08%
48 Mb/s	13.406 Mb/s	13.409 Mb/s	0.03%	13.262 Mb/s	13.410 Mb/s	1.11%
54 Mb/s	13.746 Mb/s	13.839 Mb/s	0.68 %	13.729 Mb/s	13.840 Mb/s	0.8%

Table 5.3: Comparison of the Throughput from A and from B between the model and the simulation. R_a = 54Mb/s, CW^a_{min}=CW^b_{min}=31



(a) (b)

Figure 5.5: (a) Throughput node A: Analysis versus simulation node A (b) Throughput: Simulation node A versus node B. R_a = 54Mb/s and CW^a_{min}=CW^b_{min}=31

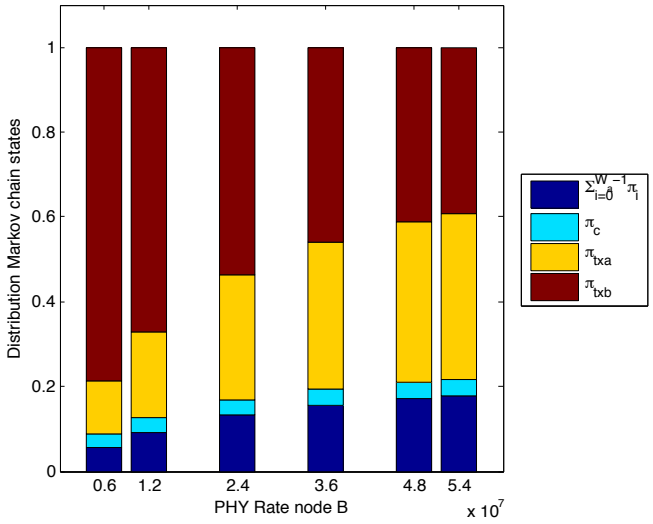


Figure 5.6: Distribution of Markov chain states. It shows the percentage of time that is used by each states ($\pi_{t_{xa}}$,$\pi_{t_{xb}}$,π_C and $\sum_{i=0}^{W-1}\pi_i$). R_a = 54Mb/s and CW^a_{min}=CW^b_{min}=31

Second scenario

What happens if the minumum contention window is different between stations A and B? We consider the case that station B uses a CW^b_{min}=63 and station A CW^a_{min}=31. We consider, as in the first scenario, the PHY rate of node A fixed to R_a = 54Mb/s and PHY rate of node B varies R_b=[6,12, 24, 36, 48, 54] MB/s.
We can see from Figure 5.7 that also in this case the model represents correctly what happens during the simulations. The error between the throughput values of the model and of the simulation is at maximum 5%.
We can see from Figure 5.7 that the throughput of node A is greater than that of node B. Increasing the contention window of node B (CW^b_{min}=63) allows node A to access more frequently the channel and this translates to a throughput gain. From the other side the throughput of node B is reduced because it has less opportunity to transmit.
Figure 5.8 shows the channel utilization in this case. We notice that node A occupies the channel for more time than in the first scenario. When R_b is 6 Mb/s, node B occupies the channel for about 65 % that is less than 80 % when the CW_{min} was the same for both nodes. When R_b is 54 Mb/s, that is the same as that of R_a, station A occupies the channel for more time because it accesses it more frequently.
Furthermore the time spent in π_c is lower than the first scenario since setting CW^b_{min}=63 reduces the collision probability.

In conclusion we derive from the model that:

- When a station uses a lower bit rate, it captures the channel for a long time. The throughput of the other stations is considerably degraded and it is almost the same for both stations (as in performance anomaly Section 3.3.2).
- Changing the minimum contention window CW_{min} of a node gives more opportunity to transmit to the node with a lower contention window CW_{min} and consequently it can increase the throughput for that node.

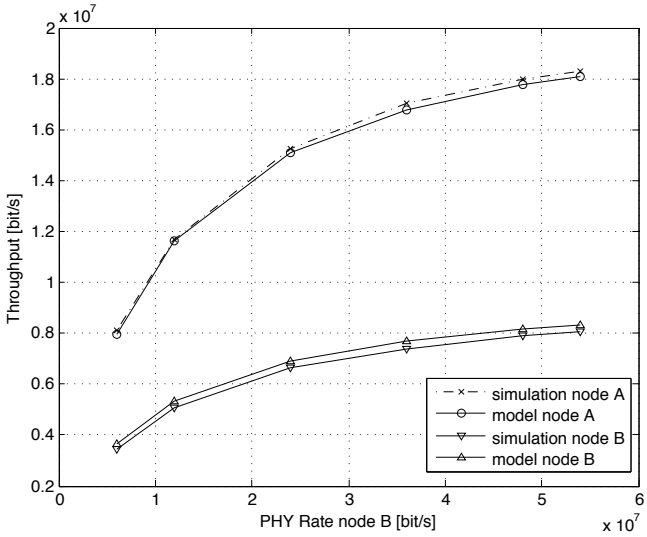


Figure 5.7: Throughput: Analysis versus simulation node A and node B. $R_a = 54$Mb/s, CW^a_{min}=31 and CW^b_{min}=63

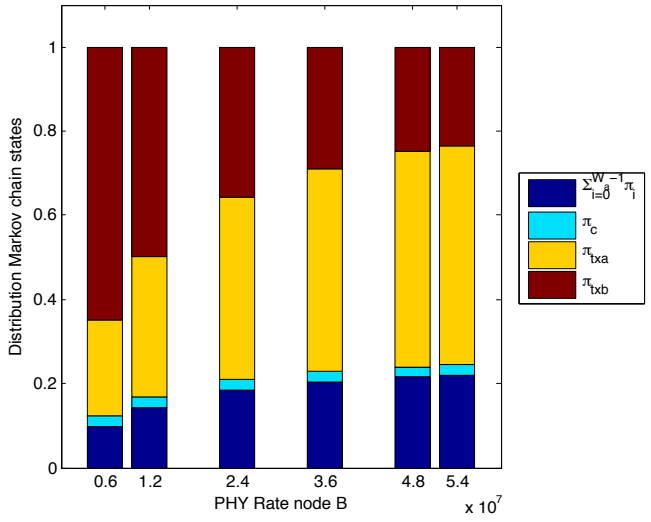


Figure 5.8: Distribution of Markov chain states. It shows the percentage of time that is used by each states ($\pi_{t_{xa}}$,$\pi_{t_{xb}}$,π_C and $\sum_{i=0}^{W-1}\pi_i$). $R_a = 54$Mb/s, CW^a_{min}=31 and CW^b_{min}=63

5.2 Rate Adaptation on top of COPE

We consider a Rate Adaptation algorithm to increase the capacity of the network as shown by papers [3] and [4]. In particular, we consider the IEEE 802.11 g which allows transmission at 6, 9, 12, 18, 24, 36, 48 and 54 Mbps.
Since we deal with static networks, stable Signal-to-Noise Ratios (SNRs) are observed at nodes. The SNR is measured from packets sent in the wireless link and it depends on the distance between two nodes and on the noise power.
For the same SNR, transmitting at a lower rate will tend to result in a lower error rate. We used a rate adaptation SNR-based. For a given SNR, it chooses the higher rate that satisfies a maximum PER ($MAXPER$).
Furthermore, nodes need to overhear packets in the channel, thus for a given node, the minimum rate is chosen to allow all as its neighbors to overhear packets with $PER \leq MAXPER$.
Let's consider node A in Figure 5.9; it chooses the rate R_a that satisfies both $PER_{aj} \leq MAXPER$ and $PER_{ac} \leq MAXPER$ since node C needs to overhear the packet from A. Also nodes B and J select the PHY rate such that for each neighbor the PER is less than $MAXPER$.

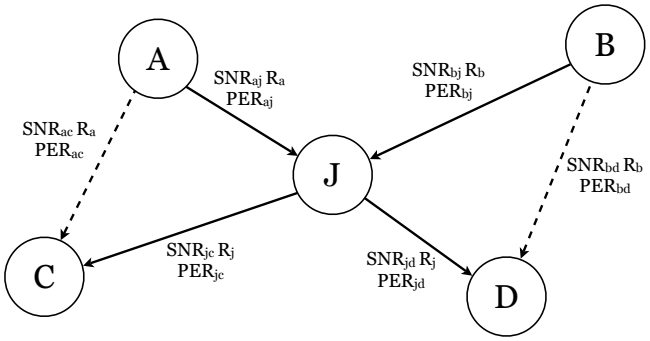


Figure 5.9: For each link the SNR, PHY Rate chosen and PER are reported. The PER is derived by the SNR and PHY Rate.

5.2.1 Tuning the MAXPER

Having a MAXPER too low reduces the system throughput, since nodes choose a low data rate that satisfies the MAXPER. Using a MAXPER too high can reduce the throughput because nodes have to transmit more times the same packet for a successful

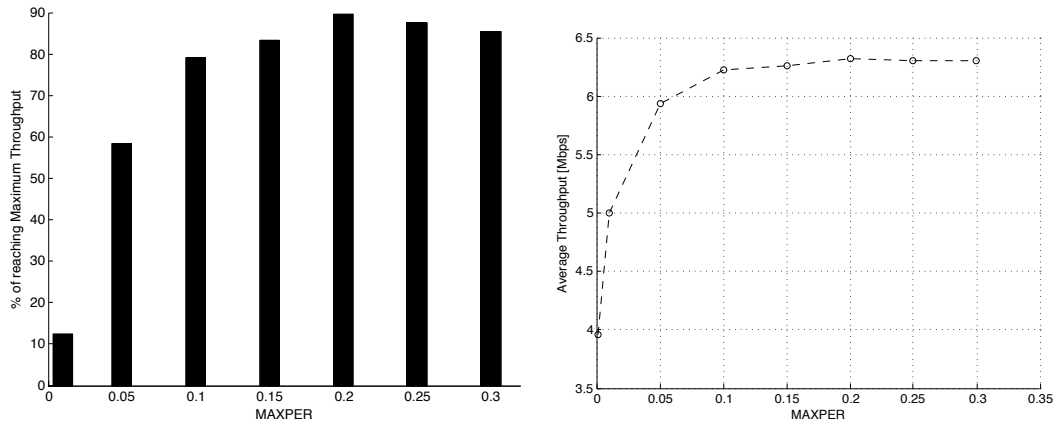
transmission. Thus it is important to find the right value for MAXPER.
We considered:

- MAXPER=[0.001 0.01 0.05 0.1 0.15 0.2 0.25 0.3]
- 48 random five nodes topologies
- two saturated UDP flows

We used the external library *dei80211mr* [20]. This library offers a better channel model since it introduces the PER for each packet.
Each topology is different from the others for the positions of nodes or for the noise power. However we selected just topologies where the relay node codes at least one packet.
Figure 5.10(a) shows for a given MAXPER the percentage of time that the system performs above 95% of the maximum throughput reached. We can see that the percentage is growing until the MAXPER reaches 0.2. When the network uses MAXPER=0.2, it can reach the maximum throughput in 90 % of the cases. For a greater value of MAXPER, the percentage decreases.
The Figure 5.10(b) shows the average throughput. We can see that after MAXPER=0.1 the average throughput is almost constant.
We fixed MAXPER=0.2 for the simulations, that is also used in COPE paper [2]. Furthermore, it leaves room for improvement with queue management algorithm since the links can have different PERs.

5.3 Queue Management

We have seen from the analysis how changing the contention window affects the throughput. However the Markov chain cannot be used in the queue management algorithm since finding the numerical solution of equation (5.16) requires too much computational time. Furthermore, the model is not complete since we haven't considered the PER that is a fundamental component in the simulation and in real world.
Therefore, we look for a queue management algorithm that can adapt fast to the virtual queues size taking into consideration the links PER.
A queue management algorithm has been proposed also by Seferoglu *et al.* in [30]. However, they considered TCP flows and made modifications to congestion control mechanism and to queue management schemes. In their solution the intermediate



(a) (b)

Figure 5.10: (a) For a given MAXPER this graph shows the percentage of time the system performs above 95% of the maximum throughput. The value 0.001 has never reached the maximum throughput. (b) shows the average throughput of the system.

nodes drops packets based on the network coding information. Our solution aims to increase the performance prioritizing the channel access based on the queue information.

5.3.1 Packet queues in COPE

Each station store a maximum of 50 packets in the Output queue. When COPE is active, each packet enqueued in the Output queue is enqueued also in the Virtual queue relative to the packet's nexthop.
When the relay node J senses the channel idle, it dequeues the first packet in Output queue and it searches at the head of the Virtual queues if there is a packet with a different nexthop to code together with the dequeued packet (see Figure 5.11). The Virtual queues are needed to speed up the search.

Since the PER can be different for each wireless link, J can have virtual queues with different lengths ($Q_c \neq Q_d$). Having unbalanced virtual queues increases the probability that one of these becomes empty and J loses the coding opportunity.
With our queue management algorithm we want to reduce the probability that one virtual queue becomes empty and we do that prioritizing the channel access.

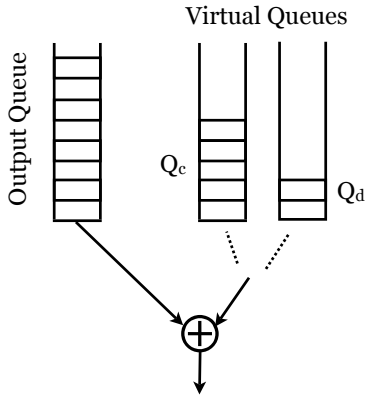


Figure 5.11: Output Queue and Virtual Queues at relay node for different nexthops. Q_c and Q_d are the sizes of the virtual queues headed respectively to C and to d.

5.3.2 Parameters considered in the Algorithm

Based on the information of virtual queues (Q_c,Q_d) and on the PERs of the links of the network, our algorithm computes the contention window to balance the virtual queues. Our algorithm has Q_c,Q_d,PER_{aj},PER_{bj},PER_{jc} and PER_{jd} as input and values of CW^a_{min}, CW^b_{min} as output (see Figure 5.12).

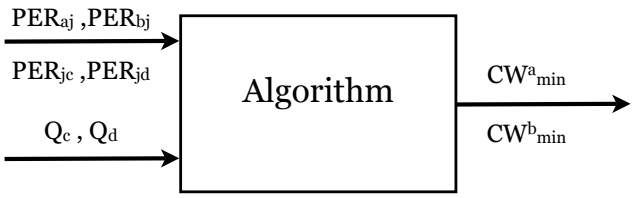


Figure 5.12: Block representation of our queue management algorithm.

How the values of CW_{min} are sent to sender nodes

We insert the value of CW_{min} in the COPE header (see Figure 2.5). We propose to add a new field for the contention window (see Figure 5.13). Thus, whenever J sends a packet to C or D, nodes A and B, which are in promiscuous mode, can overhear the packet and extract the minimum Contention Window values. The overhead for the network is almost zero since the field is about 64 bits per node and is sent in an existing packet.

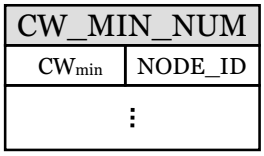


Figure 5.13: Field inserted in COPE header for sending the Contention Window values.

5.3.3 How links with different PER reflect on the virtual queues

We have seen during the simulations, and it is easy to understand, that links with different PERs play an important role on the virtual queues unbalancing. In particular, considering Figure 5.9, we have seen that:

- for sender links, when for example $PER_{aj} > PER_{bj}$, node A has to transmit more times the same packet for a successful transmission and the virtual queue with packets from A will have less packets than that from B.
- for receiver links, when for example $PER_{jc} > PER_{jd}$, node J accumulates packets headed to C due of retransmissions.

Considering the entire path of packets ($A \rightarrow D$ and $B \rightarrow C$) , we can derive which virtual queue will tend to have less packets than the other and which more.
We consider the fraction of $\frac{PER_{receiver}}{PER_{sender}}$ and we identify which sender between a and b is $send_{min}, send_{max}$:

$$send_{min} | \min(\frac{PER_{jd}}{PER_{aj}}, \frac{PER_{jc}}{PER_{bj}}) \qquad send_{max} | \max(\frac{PER_{jd}}{PER_{aj}}, \frac{PER_{jc}}{PER_{bj}}). \tag{5.20}$$

The worst case is when PER_{sender} is close to MAXPER and $PER_{receiver}$ is almost 0; this means that J receives a lot of packets from the receiver but is not able to send them all and will start accumulating them.

Which values to assign to CW^a_{min},CW^b_{min}

Once we found $send_{min}$ and $send_{max}$, depending on the output queue size at the relay node, we increase the probability of packet transmission of $send_{min}$ or we reduce the probability of transmission of $send_{max}$.
Assuming to have a link with PER_{link}, the average number of packet transmissions is:

$$\frac{1}{(1 - PER_{link})}. \tag{5.21}$$

The transmission probability in the case of a fixed contention window W is:

$$p = \frac{2}{(W+1)}. \tag{5.22}$$

We can compensate the number of transmissions required to transmit successfully a packet changing the transmission probability of nodes. The idea is that when we want to increase the transmission probability of node $send_{min}$ we set the Contention Window:

$$CW^i_{min} = K * (CW^j_{min} + 1) - 1 \tag{5.23}$$

where i is the $send_{min}$ node and j is the other node.
Thus, supposing that a is the $send_{min}$ and we want to increase the transmission probability $send_{min}$ keeping fixed $CW^b_{min} = 31$, we compute:

$$CW^a_{min} = \frac{(1 - PER_{aj})(1 - PER_{jc})}{(1 - PER_{bj})(1 - PER_{jd})}(CW^b_{min} + 1) - 1 \tag{5.24}$$

where the coefficient $K = \frac{(1-PER_{aj})(1-PER_{jc})}{(1-PER_{bj})(1-PER_{jd})} \leq 1$ because a is the $send_{min}$. This coefficient is needed to balance the delivery probability of the packets of the two paths ($A \rightarrow D$ and $B \rightarrow C$).
In case we want to reduce the probability of b, we fix $CW^a_{min} = 31$ and we compute CW^b_{min} from equation 5.24. In this case the new coefficient $K' = K^{-1} \geq 1$. Since MAXPER=0.2, the coefficients K and K' take values in the interval [0.64 1.56].

This solution is part of our queue management algorithm. However we have seen that this by itself is not sufficient. Thus, in case virtual queues are highly unbalanced, we reduce or double the Contention Window.

5.3.4 Proposed Algorithm

First we asked: *how should the output queue at the relay node be to optimize network coding?*
The Output queue should have enough packets from both paths so a coded packet can always be sent. Furthermore, the relay node should not have too much packets otherwise it drops packets.
We divide the Output queue in three segments:

- if the queue size is $(Q_c + Q_d) \leq 15$, we increase the probability of the $send_{min}$ node following equation (5.24). If the virtual queues become highly unbalanced we divide by 2 the CW_{min} for the node with less packets.

- if the queue size is $15 < (Q_c + Q_d) \leq 30$, we think that this is the right size for the Output queue, and we balance the virtual queues following the (5.24).
- if the queue size is $(Q_c + Q_d) > 30$, there is a risk that the queue becomes full and starts to drop packets. In this case, we reduce the transmission probability of the node with more packets: the higher the difference $|Q_c - Q_d|$, the higher the CW_{min} of the node with more packets.

In the third case, we prefer to reduce the probability of the sender node with more packets in the queue, thus also J has more probability to access the channel and it is able to drain its queue faster.

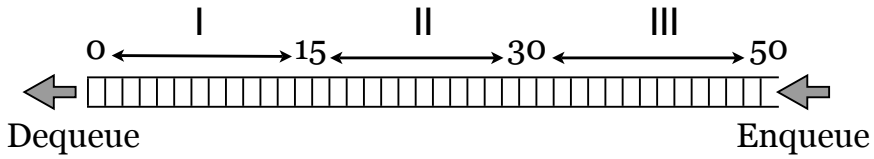


Figure 5.14: FIFO Output queue divided in three segments.

The full algorithm is reported in Algorithm 1. Simulations to prove the efficacy of this algorithm and to extensively evaluate its performance are reported in the next Chapter.

input : Q_c,Q_d, PER_{aj},PER_{bj},PER_{jc},PER_{jd}
output: CW^a_{min},CW^b_{min}

At relay node J; $CW^{default}_{min} = 31$;
find $send_{min}$, $send_{max}$, K and K';
if $(Q_c + Q_d) \leq 15$ **then**
  if $|Q_c - Q_d| < 2$ **then**
    if a==$send_{min}$ **then**
      $CW^a_{min} = K * (CW^{default}_{min} + 1) - 1$; $CW^b_{min} = CW^{default}_{min}$;
    else
      $CW^a_{min} = CW^{default}_{min}$; $CW^b_{min} = K' * (CW^{default}_{min} + 1) - 1$;
    end
  else
    if $Q_d > Q_c$ **then**
      CW^a_{min}=$CW^{default}_{min}$; CW^b_{min}=$\frac{1}{2} * CW^{default}_{min}$;
    else
      CW^a_{min}=$\frac{1}{2} * CW^{default}_{min}$; CW^b_{min}=$CW^{default}_{min}$;
    end
  end
end
if $15 < (Q_c + Q_d) \leq 30$ **then**
  if $|Q_c - Q_d| < 4$ **then**
    if a==$send_{min}$ **then**
      $CW^a_{min} = K * (CW^{default}_{min} + 1) - 1$; $CW^b_{min} = CW^{default}_{min}$;
    else
      $CW^a_{min} = CW^{default}_{min}$; $CW^b_{min} = K' * (CW^{default}_{min} + 1) - 1$;
    end
  else
    if a==$send_{max}$ **then**
      $CW^a_{min} = K * (CW^{default}_{min} + 1) - 1$; $CW^b_{min} = CW^{default}_{min}$;
    else
      $CW^a_{min} = CW^{default}_{min}$; $CW^b_{min} = K' * (CW^{default}_{min} + 1) - 1$;
    end
  end
end
It continues in the next page

if $(Q_c + Q_d) > 30$ **then**
  if $|Q_c - Q_d| < 2$ **then**
    if $a{=}{=}send_{max}$ **then**
      $CW^a_{min} = K * (CW^{default}_{min} + 1) - 1; \quad CW^b_{min} = CW^{default}_{min};$
    else
      $CW^a_{min} = CW^{default}_{min}; \quad CW^b_{min} = K' * (CW^{default}_{min} + 1) - 1;$
    end
  end
  if $2 \leq |Q_c - Q_d| < 10$ **then**
    if $Q_d > Q_c$ **then**
      $CW^a_{min}{=}2 * CW^{default}_{min}; \; CW^b_{min}{=}CW^{default}_{min};$
    else
      $CW^a_{min}{=}CW^{default}_{min}; \; CW^b_{min}{=}2 * CW^{default}_{min};$
    end
  end
  if $10 \leq |Q_c - Q_d| < 15$ **then**
    if $Q_d > Q_c$ **then**
      $CW^a_{min}{=}4 * CW^{default}_{min}; \; CW^b_{min}{=}CW^{default}_{min};$
    else
      $CW^a_{min}{=}CW^{default}_{min}; \; CW^b_{min}{=}4 * CW^{default}_{min};$
    end
  end
  if $|Q_c - Q_d| \geq 15$ **then**
    if $Q_d > Q_c$ **then**
      $CW^a_{min}{=}8 * CW^{default}_{min}; \; CW^b_{min}{=}CW^{default}_{min};$
    else
      $CW^a_{min}{=}CW^{default}_{min}; \; CW^b_{min}{=}8 * CW^{default}_{min};$
    end
  end
end

Algorithm 1: Pseudo Code of the Queue Management Algorithm

Chapter 6

Simulations with NS 2

In this chapter we present the simulations with Network Simulator 2 (NS 2). We compare the performance of the system in three cases: without COPE, with COPE and COPE with the queue management algorithm. To compare the performance of these systems we use as metrics throughput. Furthermore, we are interested in the percentage of coded packets at the relay node and its virtual queues.
The chapter is structured as follows. First we give a brief introduction of NS 2 and we present COPE's implementation on NS 2. We conclude the chapter with simulation results.
Since multi-rate is not supported by NS 2, we used the external library (dei80211mr) developed by the University of Padua [20]. This library has also an improved channel model. For the implementation of COPE we started from the google project [21], keeping the basic structure of the project. We completely changed the coding and decoding process and we fixed some bugs in the acknowledgments.

6.1 Introduction to NS 2

Network Simulator (NS) is an open-source discrete event simulator. NS provides support for different types of simulations in the networking field. Simulation of wired as well as wireless network functions and protocols (TCP, UDP) can be done using NS.
It has been started to develop by 1989. The project has received contributions from several Universities, in particular from University of California and Cornell University. Since 1995 the Defense Advanced Research Projects Agency (DARPA) supported the development of NS, as did later also the National Science Foundation (NSF).

NS 2 uses two key languages: C++ and Object-oriented Tool Command Language (OTcl). NS 2 modules are written in C++ and they are linked to OTcl objects using TclCL (Figure 6.1).
Why using two languages? C++ code is fast to run but it is slow to change since it has to be compiled. Instead OTcl is slow to run but fast to change. Tcl scripts are used for simulations and to interface with C++ code.
After the installation of NS 2, an executable file *ns* is created in the home directory. The simulation file written in Tcl can be invoked from the shell environment:

```
$ns example.tcl
```

This invocation creates a trace file that is used to calculate some metrics like throughput and delay.

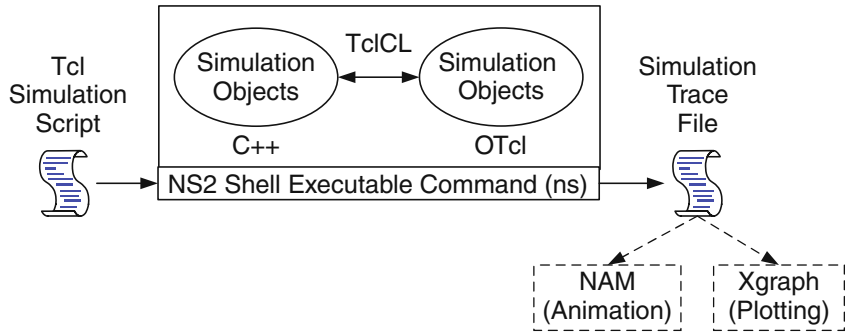


Figure 6.1: Basic Architecture of NS

6.1.1 Implementation of IEEE MAC 802.11 on NS 2

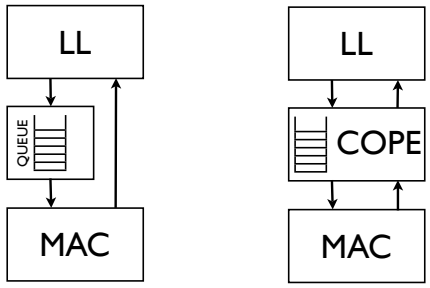
NS 2 follows the OSI model for the layering. Each layer is represented by one or more C++ files in NS 2.
The function *recv()* sends packets through different layers. Each packet contain a field *direction* that is used to send packets UP or DOWN the stack. The connections between different layers are controlled by TCL files.
Since COPE has been implemented between MAC and LL, we are interested in the Data Link Layer (DLL). Figure 6.2 (a) reports the default implementation of Data Link Layer in NS 2. It is composed by:

- *Link Layer* maps the protocol address (IP) to the Hardware address (MAC).
- *Output Queue* has only the down-target, in other words, it is used just when a node transmits a packet and not when it receives a packet. There are two main implementations Droptail and PriQueue.
- *Media Access Control (MAC)* keeps communication between 802.11 stations by coordinating access to a shared radio channel.

A complete overview of MAC layer of NS 2 with functions of each layer is reported by Liu in [23] and by Robinson in [24].

6.2 Implementation of COPE on NS 2

COPE inserts a new coding layer between the LL and MAC layers as shown in Figure 6.2 (b). This layer contains the output queue since the coding is done during the dequeuing of packets. Furthermore, other queues are presented in the COPE layer to speed up the search of packets to code together (**VirQueue**) and to manage the acknowledgment and retransmissions (**NonAckQueue** and **PendAckQueue**).
To perform the *Opportunistic Listening*, each node can overhear packets through the function *tap()* present at the MAC layer. This function allows the upper layers to make a copy of all packets overheard. Each native packet overheard is saved in **PacketPool**. When the MAC layer receives a packet, it sends it up to the COPE layer. Thus, if the packet is coded, COPE layer can decode it and it schedules an ACK packet.



(a) (b)

Figure 6.2: (a) default DLL implementation of NS 2 (b) modified implementation to insert COPE layer

6.2.1 Architecture

COPE needs several accessories functions for the coding and decoding process, the acknowledgment and the retransmission. These functions have the correspondent C++ files in the NS 2 implementation and they are presented in Figure 6.3. For each block the principal functions are reported. The file *cope.cc* contains the function *recv()* to communicate with LL and MAC layer.

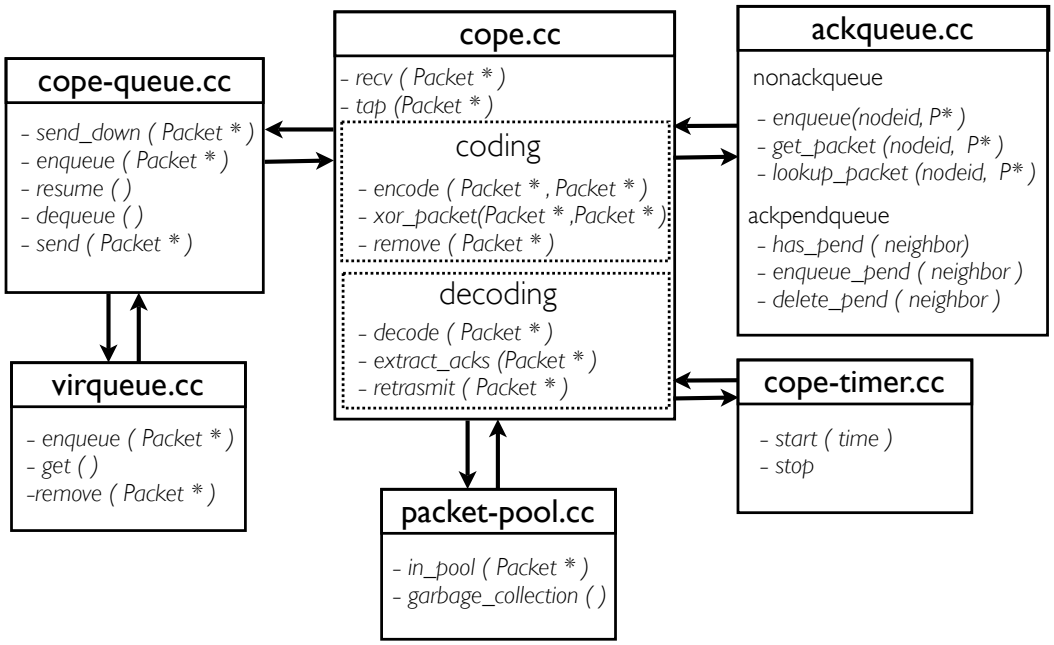


Figure 6.3: Files present in COPE implementation.

CopeQueue

The CopeQueue is the Output queue used by COPE. It inherits the PriQueue class of NS 2. Its function to enqueue and dequeue the packets of the station. As PriQueue, the CopeQueue assigns each packet to a priority queue. The Output queue can store most 50 packets.

VirQueue

Whenever a packet is enqueued in the Output queue, a copy of it is inserted in the Virtual queue. This is needed to speed up the search of a candidate packet to code.

NonAckQueue

NonAckQueue is used to store all native packets that are in an encoded packet and that have not been acknowledged. Packets, which are sent as native, are not put into NonAckQueue.
Whenever a coded packet is sent out, each native packet is enqueued in NonAckQueue and COPE starts a timer for the native packet. The timeout is a little longer of the Round Trip Time (RTT). If the Acknowledgment is not received before the timer expires, the native packet is inserted at the head of the Output queue. On the contrary, if the Acknowledgment is received, the timer is stopped and the packet is deleted. The retransmission threshold is set to 2, after which the packet is discarded.

AckPendQueue

Each node keeps a local sequence number for each packet sent to the same neighbor. This is needed for the acknowledgment. When a coded packet is received, if it is decodable, the node inserts the previous hop's MAC and the local sequence number of the native packet in the AckPendQueue. After either 8 pending ACKs or a duplicate pending ACK, a cumulative ACK is sent out.
The cumulative ACK can be inserted in an already existing packet or a new Control packet is created. The cumulative ACK is composed by three fields: Neighbor, LastAck and AckMap. LastAck is the local sequence number of last ACK. When the Neighbor receives the cumulative ACK, it knows which packets has been received and which packets to retransmit. The AckMap is made of 8 bit and each bit represents previous packets. For example, if the LastAck is 52 and the Ack Map is 1010111, the Neighbor retransmits packets with local sequence number 46 and 48.

PacketPool

The PacketPool is needed to save the overheard native packets. These packets are needed in the decoding process and they are saved in a hash map to speed up the search during the decoding. The PacketPool is garbage collected every few seconds.

6.2.2 Packet processing

In this section we show what are the most significant steps that COPE follows when sending (DOWN) or receiving (UP) a packet.

DOWN

The station wants to send a packet.

Enqueue

1. The file *cope.cc* receives the packet from the LL and calls the function *send_down()* of *cope-queue.cc*
2. The packet is enqueued in the output queue and in the virtual queue of its next-hop.

Dequeue

1. If the channel is idle or when *resume()* is called the MAC, a packet is dequeued from the Output Queue and from the virtual queue.
2. If there is a packet with a different next-hop in the virtual queues, it is encoded together with the dequeued packet. Both packets XORed are inserted in the NonAckQueue and a timeout is started.
3. The acks and reception reports are added to the packet.

UP

The station receives a packet.

1. The file *cope.cc* receives a packet from the MAC and processes it.
2. Acks and reception reports are extracted and acked packets are removed from NonAckQueue.
3. If the packet is a control packet, it is discarded. It has been used just to send acks and reception reports.
4. If the packet is Native, a copy is inserted in the packet pool. If the current node is the destination, the packet is sent up to the LL, otherwise it is discarded.

5. If the packet is Coded and at least one of the XORed packets is for the node, the station tries to decode. If station has the packet that composes the coded packet in PacketPool, it can decode the packet and extract its native packet. After that, it sends the packet to LL and it inserts a copy in PendAckQueue.

6.3 Simulation Results

We used NS 2.34 [19], the external library dei80211mr [20] and the COPE module.
The packet size is set to 1500 Bytes and we consider the five nodes topology. The transmission range is set to 100 m and the interference range and the carrier sense range are set to 200m. Each node has a transmission power of 20dBm and it uses the 2.4Ghz frequency. For the channel model we use the Free-Space Path Loss.
We consider the static scenario in Figure 5.9 and two saturated UDP traffic flows from node A to node D and from node B to C.
The positions of the nodes are generated randomly and the noise power is chosen in the set N=$\{10e^{-11}, 20e^{-11}, 30e^{-11}, 40e^{-11}, 50e^{-11}\}$. We consider about 70 combinations of topology and noise power. The simulations last for 100 seconds and it is shown the mean result over three simulations.
We compare the throughput of the system obtained without COPE, with COPE and with COPE + queue management.
Figure 6.4 shows the throughput improvement of the system in the 73 topologies with COPE and with COPE +QUEUE MGMT with respect to the throughput without COPE. We can notice that our algorithm increases the performance of COPE when COPE's gain is low. On average our algorithm increases COPE's performance by 7.5%.
Figure 6.5 shows the throughput improvement of the system with COPE + QUEUE MGMT with respect to the system with COPE. We can see that our algorithm is always increasing the performance of COPE. The average improvement is 7.5% and the maximum improvement is 57%.

Figures 6.6 (a) (b) show the fraction of coded packets at the relay node with COPE and with COPE+QUEUE MGMT. COPE codes about 70% of the packets sent out, our algorithm brings the fraction of coded packets to 85%.
Figures 6.7 (a) (b) show the Output queue at the relay node. The size of the queue is measured every time the node has sent a packet and is averaged over the total number of transmissions. The average difference between the virtual queues is 10 packets with

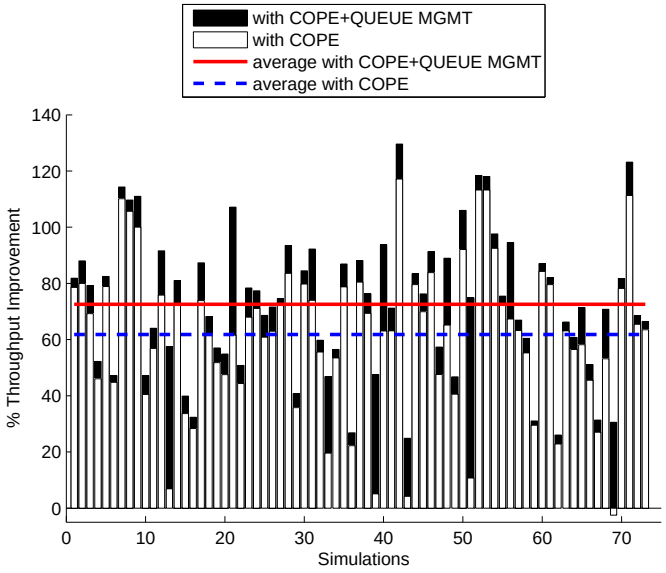


Figure 6.4: Throughput gain over a typical 802.11 system in two cases: with COPE and with COPE+QUEUE MGMT

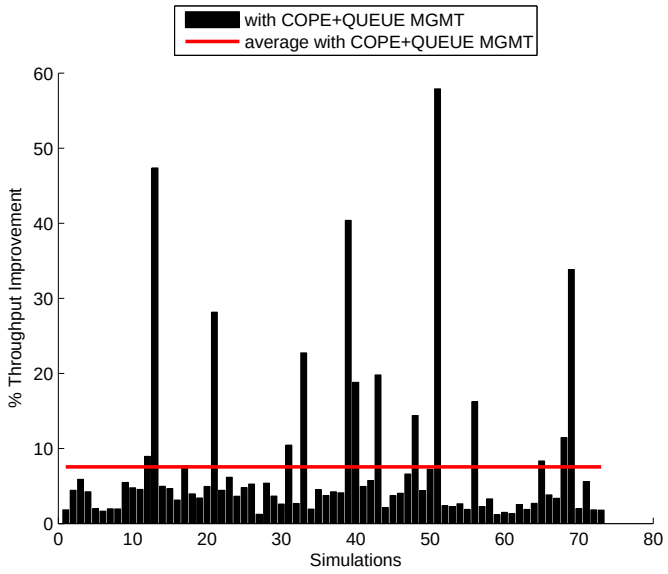
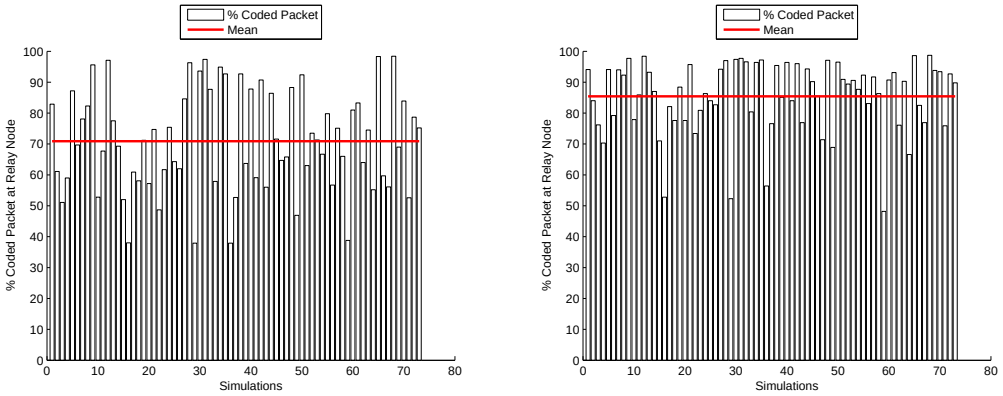


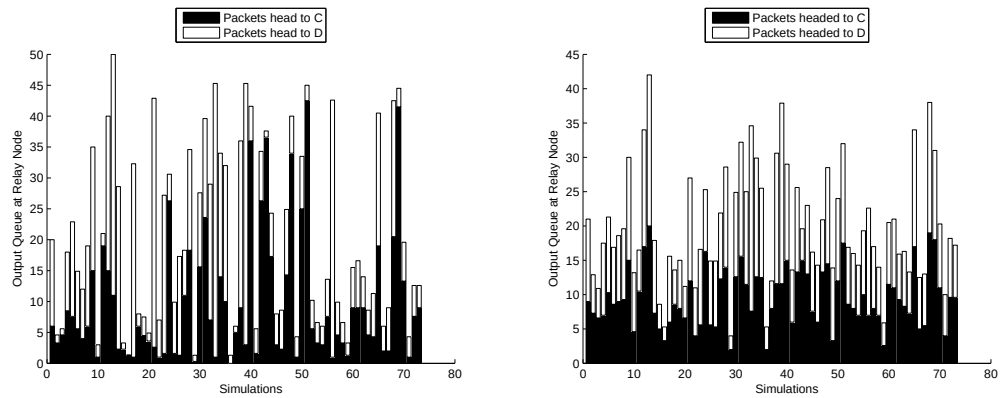
Figure 6.5: COPE+QUEUE MGMT improves gain by up to 57 % over the system with just COPE. The average improvement is 7.5%.

COPE and 2.7 packets with our system. This shows that our algorithm is able to balance the virtual queues at the relay node.



(a) (b)

Figure 6.6: Fraction of coded packets over total number of sent packets at the relay node with COPE (a) and COPE+QUEUE MGMT (b). The average goes from 70% to 85%.



(a) (b)

Figure 6.7: Size of the output queue with packets headed to C and D with COPE (a) and COPE+QUEUE MGMT (b). In the second case the virtual queues are more balanced.

Best Improvement

Let's consider the case where our algorithm brings the system gain to 57%. This case is shown in Figure 6.8. The noise power is fixed to N=$40e^{-11}$.
We can notice that the receiver links have different values of PERs ($PER_{jc} = 0.18$ and $PER_{jd} = 0.009$). Packets headed to C will tend to accumulate at the relay node queue (as shown in Figure 6.10 (a)).

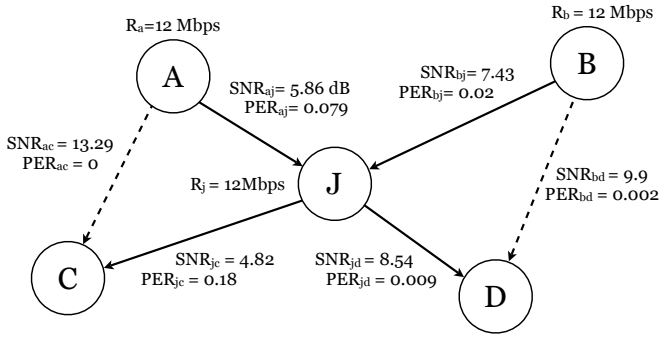
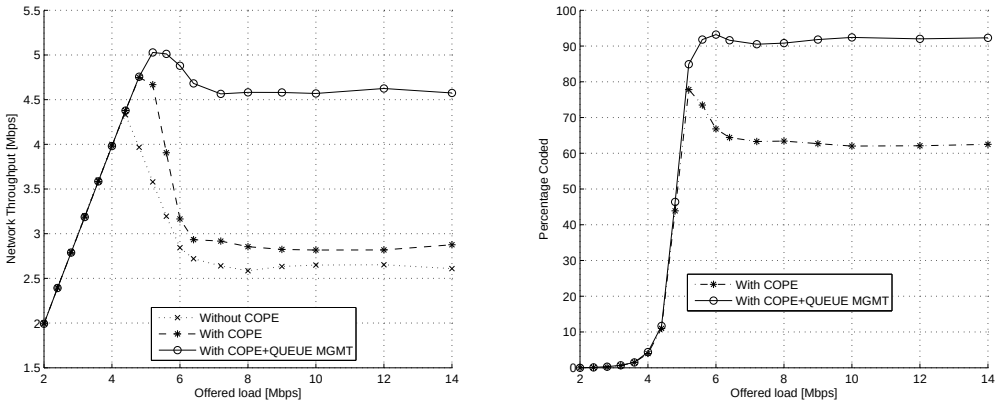


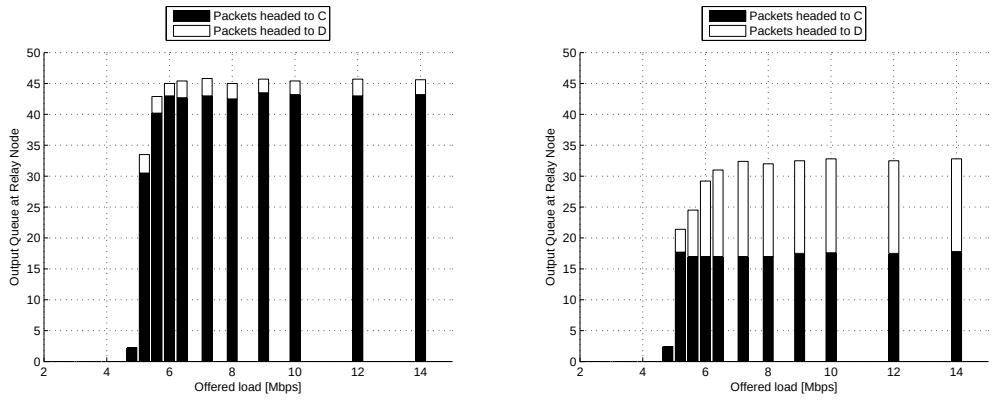
Figure 6.8: SNR, PHY rate and PER for each link in one of the cases where we obtain a great throughput improvement with COPE+QUEUE.

For this topology we consider different offered load. We can see in Figure 6.9(a) that the systems considered have the same throughput as long as the traffic offered is less than 4 Mbps. After they reach the maximum of the network throughput, the performance degrades. COPE with our algorithm is able to compensate the performance degradation optimizing the transmission probabilities.
Furthermore, the fraction of coded packets is higher with our algorithm (see Figure 6.9 (b)).
Let's consider the output queue at the relay node (Figure 6.10(a)). We can see that the relay node doesn't accumulate packets in its output queue as long as the offered load is less than 5 Mbps. After that, the average size becomes quickly 45 packets. This means that packets have a high probability to be dropped because the Output queue reaches the maximum size of 50 packets. Instead, our algorithm (Figure 6.10(b)) reduces the size of the output queue to 30 packets and balances the number of packets headed to different nexthops increasing the coding opportunity.



(a) (b)

Figure 6.9: (a) COPE+QUEUE MGMT provides 57% increase in UDP troughput in presence of saturated traffic. (b) Percentage of packets Coded at relay node J.



(a) (b)

Figure 6.10: Size of the output queue with packets headed to C and D with COPE (a) and COPE+QUEUE MGMT (b). In the second case the output queue is reduced and the virtual queues are more balanced

Average Improvement

In this case COPE+QUEUE MGMT improves gain by about 7% over COPE. The noise power is set to N=$10e^{-11}$. We can notice that in this case the different PERs are at sender links ($PER_{bj} = 0.144$ and $PER_{aj} = 0.012$) and the receiver links have a low values of PERs.

As we can see from Figure 6.13(a), the relay node accumulates packets headed to D. This is explained with the fact that the relay node receives more packets from A, since $PER_{aj} < PER_{bj}$ and it codes just 60% of packets, losing coding opportunities.

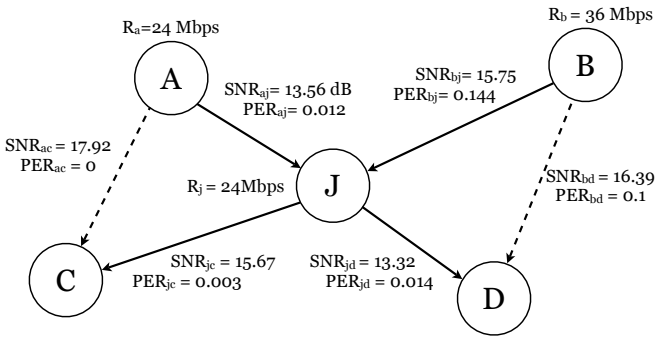
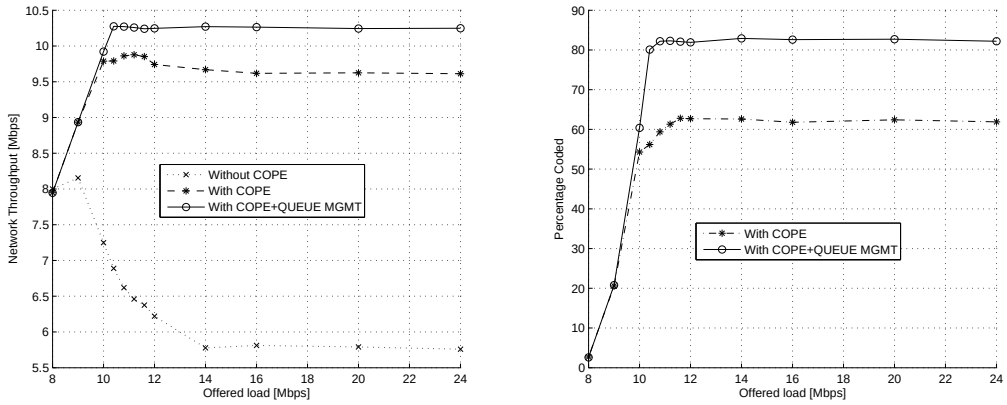
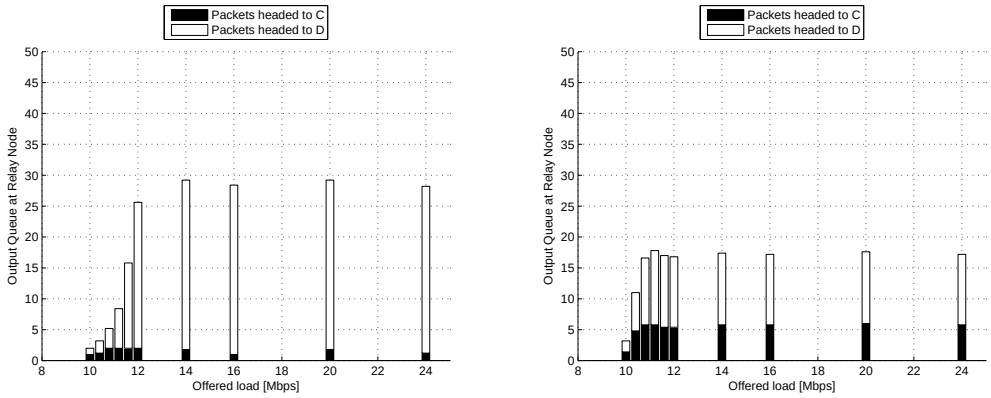


Figure 6.11: SNR, PHY rate and PER for each link in one of the case where we obtain at maximum 7% throughput improvement with COPE+QUEUE.



(a) (b)

Figure 6.12: (a) COPE+QUEUE MGMT provides 7% increase in UDP troughput in presence of saturated traffic. (b) Percentage of packets Coded at relay node J.



(a) (b)

Figure 6.13: Size of the output queue with packets headed to C and D with COPE (a) and COPE+QUEUE MGMT (b). In the second case the output queue is reduced and the virtual queues are more balanced

Little Improvement

In this case, our algorithm improves over COPE by only 1%. The noise power is set to N=$40e^{-11}$. We can notice that in this case the sender links have similar PERs and they are greater than the receiver PERs (about 0.1 greater). In this case, the queues at the relay node are often empty and the percentage of coded packets is around 50% (see Figure 6.15(b)). Even if the relay node is coding just 50% of packets, it is not receiving many packets from the senders so it doesn't accumulate packets in its output queue.

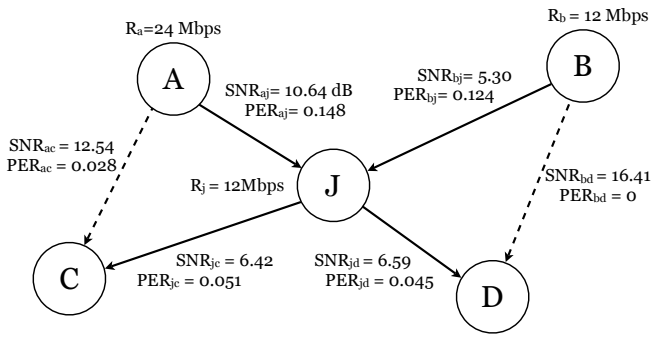
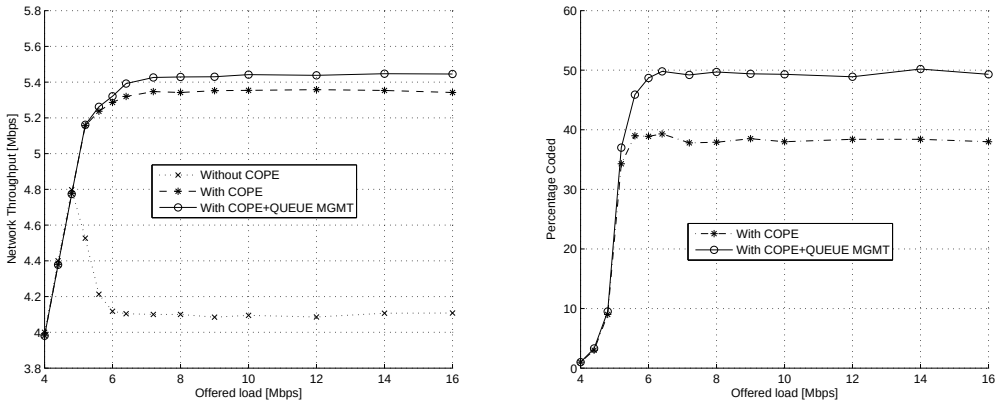
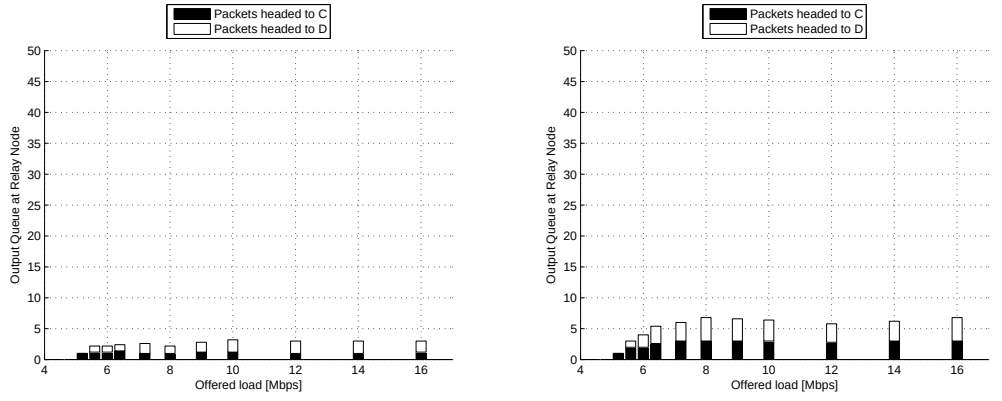


Figure 6.14: SNR, PHY rate and PER for each link in one of the case where COPE+QUEUE improves gain of 1%.



(a) (b)

Figure 6.15: (a) COPE+QUEUE MGMT provides 1% increase in UDP troughput in presence of saturated traffic. (b) Percentage of packets Coded at relay node J.



(a) (b)

Figure 6.16: Size of the output queue with packets headed to C and D with COPE (a) and COPE+QUEUE MGMT (b).

Chapter 7

Conclusion

In this thesis we studied and analyzed COPE architecture in multi a rate wireless scenario. We furnish a Markov chain to model when two nodes share the channel with different PHY rates. Therefore, we propose a queue management algorithm on top of COPE architecture to increase the coding opportunity and network throughput.
We simulate with NS 2 considering a five nodes topology and the 802.11g standard. We compare the system without COPE, with COPE and with COPE + QUEUE MANAGEMENT. We show that our algorithm allows to increase the throughput gain when COPE's gain is low.
When the channel quality between the relay node and one receiver is bad, the relay node accumulates packets in its queue and it can drop packets. Our algorithm reduces the size of the output queue at the relay node changing the contention window of the sender nodes. Furthermore, the algorithm balances the virtual queues headed to different nexthops and increase the coding opportunity.
When the PER is low and quite the same for the receivers links, our algorithm does't improve the performance of the system, since the relay node is not accumulating packets in its output queue.
Our algorithm achieves a throughput gain of up to 57% over COPE, with an average gain of 7.5%.

Bibliography

[1] G. Bianchi. *Performance analysis of the IEEE 802.11 distributed coordination function.* IEEE Journal on Selected Areas in Communications, 18(3):535-547, 2000.

[2] S. Katti, H. Rahul, W. Hu, D. Katabi , M. Médard and J. Crowcroft *"XORs in the air: practical wireless network coding"*, In ACM SIGCOMM, 2006.

[3] Kim T.S., Vural S., Broustis I., Syrivelis D., Krishnamurthy S.V., and La Porta T., *"A Framework for Joint Network Coding and Transmission Rate Control in Wireless Networks"* IEEE INFOCOM 2010, San Diego

[4] Kumar, R., Tati.S., De Mello, F., Krishnamurthy S.V. and La Porta, T. *"Network Coding Aware Rate Selection in Multi-Rate IEEE 802.11 "* IEEE ICNP 2010, Kyoto, Japan .

[5] R.Ahlswede, N.Cai, S.-Y.R.Li, and R.W.Yeung *"Network information flow"* IEEE Transactions on Information Theory, vol. IT-46, no.4,2000.

[6] H. Seferoglu, A. Markopoulou, U. C. Kozat, *"Network Coding-Aware Rate Control and Scheduling in Wireless Networks"* in Proc. of ICME09, New York, June 2009.

[7] S. R. Li, R. W. Yeung, and N. Cai. *Linear Network Coding.* In IEEE Transactions on Information Theory, 2003

[8] P. A. Chou, T. Wu and K. Jain, *Practical Network Coding*, in 51st Allerton Conf.Communication, Control and Computing, Oct. 2003.

[9] R. Koetter and M. Medard *An algebrical approach to network coding.* IEEE/ACM Transaction on Networking, 2003li

[10] S. Pal, S. R. Kundu, K. Basu, and S. K. Das. *" IEEE 802.11 Rate Control Algorithms: Experimentation and Performance Evaluation in Infrastructure Mode"*. In PAM, 2006

[11] Onoe Rate Control. http://madwifi.org/browser/trunk/ath_rate/onoe.

[12] E. N. Gilbert *"Capacity of a burst-noise channel"* Bell System Technical Journal, Vol. 39 (September 1960), pp. 1253-1265.

[13] A. Kamermen and L. Monteban. *WaveLan-II: A High-performance wireless LAN for the unlicensed band.* Bell Lab Technical Journal, pages 118-133, Summer 1997.

[14] Ji-Hoon Yun, *Throughput Analysis of IEEE 802.11 WLANs with Automatic Rate Fallback in a Lossy Channel*, IEEE Transactions on Wireless Communications (ISSN: 1536-1276, IF'08: 2.181), Volume 8, No. 2, Feb. 2009, Pages 689-693 [Link].

[15] M. Lacage, M. H. Manshaei, and T. Turletti. *IEEE 802.11 Rate Adaptation: A Practical Approach.* In ACM MSWiM, 2004

[16] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. *Performance anomaly of 802.11b.* In Proc. of IEEE INFOCOM03, April 2003.

[17] IEEE standard 802.11, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, August 1999

[18] J. Deng, B. Liang, P. K. Varshney*"Tuning the carrier sensing range of IEEE 802.11 MAC"* IEEE GLOBECOM '04., Vol. 5 (2004), pp. 2987-2991.

[19] NS 2 http://www.isi.edu/nsnam/ns/

[20] An improved 802.11 implementation for ns2 with enhanced interference model. http://www.dei.unipd.it/wdyn/?IDsezione=5090

[21] COPE on NS 2, google project by Uppsala University, http://code.google.com/p/uu-cope/

[22] C. Papadimtrou and K. Striglitz. Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, Englewood Cliffs, NJ, 198.

[23] K. Liu *Understanding the implementation of IEEE MAC 802.11 standard in NS-2.*

[24] J. Robinson *http://www.joshuarobinson.net/docs/802_ 11.html.*

[25] S. E. Tajbakhsh, M. Orang, M. H. Sohi, A. Movaghar, *A Queuing Model of Opportunistic Network Coding in Wireless Medium*, International Conference on the Latest Advances in Networks (ICLAN'2008), 2008.

[26] M. B. Iraji, M. H. Amerimehr, and F. Ashtiani, *A Queueing Model for Wireless Tandem Network Coding* in Proceedings of the 2010 IEEE Wireless Communication and Network Confer- ence (WCNC 09), 2009.

[27] S. Floyd and V. Jacobson, *Random Early Detection Gateways for Conges- tion Avoidance*, IEEE/ACM Trans. Net., vol. 1, no. 4, Aug. 1993, pp. 397413.

[28] S. Athuraliya, S. Low, V. Li, and Q. Yin. *REM Active Queue Management.* IEEE Network Magazine, 15(3), May 2001.

[29] Feng W, Kandlur D, Saha D and Shin KG *The BLUE Active Queue Management Algorithms*, IEEE/ACM. Transactions on Networking 10(4), 513528.

[30] H. Seferoglu and A. Markopoulou, *Network coding-aware queue management for unicast ows over coded wireless networks*, in Proc. of NetCod, Toronto, Canada, June 201

[31] Zhu, H., Cao, G.: rDCF: A relay-enabled medium access control protocol for wireless ad hoc networks. In: Proceedings of IEEE INFOCOM. (2005)

[32] L. M. Feeney, B. Cetin, D. Hollos, M. Kubisch, S. Mengesha, and H. Karl, *Multirate relaying for performance improvement in IEEE 802.11 WLANs,* in Proc. International Conference on Wired/Wireless Internet Communication (WWIC), Coimbra, Portugal, May 2007

Ringraziamenti

Questa volta è stata davvero dura...

Grazie a Francesca per essermi stati vicina nei momenti più difficili.

Grazie al Prof. Zorzi per i suoi preziosi consigli per la stesura della tesi e per le correzioni.

Grazie ai miei genitori e ai miei fratelli per avermi supportato.

Thanks to Prof. Srikanth for the opportunity to work in the networking lab and thanks to my lab mates for the great time we spent together.

Un ricordo particolare va a mia nonna Anna che mi è sempre stata vicina.