Network coding for quality of service in wireless multi-hop networks

Youghourta Benfattoum

UNIVERSITE PARIS-SUD

ÉCOLE DOCTORALE D'INFORMATIQUE

Laboratoire de Recherche en Informatique

DISCIPLINE : COMPUTER SCIENCE

DOCTORAL THESIS

defended on November, 15^{th} 2012

by

Youghourta BENFATTOUM

Network Coding for Quality of Service in Wireless Multi-hop Networks

Advisor:	Khaldoun AL AGHA	Professor (Université Paris-Sud)
Co-advisor:	Steven MARTIN	Associate Professor (Université Paris-Sud)
Composition of the jury:		
Reviewers:	Jean-Marie GORCE	Professor (INSA Lyon)
	Yacine GHAMRI-DOUDANE	Associate Professor (ENSIIE Evry)
Examiners:	Nadia BOUKHATEM	Associate Professor (Telecom ParisTech)
	Daniel ETIEMBLE	Professor (Université Paris-Sud)
	Paul MUHLETALER	Research Director (INRIA Rocquencourt)

Acknowledgements

I would like to ...

Orsay, November, 15^{th} *2012* Y. B.

Abstract

In this thesis we deal with the application of Network Coding to guarantee the Quality of Service (QoS) in terms of throughput and delay for wireless multi-hop networks.

Since the medium is shared, wireless networks suffer from the negative interference impact on the bandwidth. It is thus interesting to propose a Network Coding based approach that takes into account this interference during the routing process. In this context, we first propose an algorithm minimizing the interference impact for unicast flows while respecting their required bandwidth. Then, we combine it with Network Coding and more precisely the "Alice and Bob" scheme in order to increase the number of admitted flows in the network. Next, we combine the resulting algorithm with Topology Control to still improve the interference management and optimize the network performance. Last, we show by simulation the benefit of combining the three fields: Network Coding, interference consideration and Topology Control.

We also deal with delay management for multicast flows. We use Random Network Coding and more exactly the Generation-Based Network Coding (GBNC) that combines the packets per blocks (generations). Most of the works on GBNC consider a fixed generation size. Because of the network state variations (end-to-end delay, packet loss, congestion...), the delay of decoding and recovering a block of packets can vary accordingly degrading the offered QoS. To solve this problem, we propose a network-and content-aware method that adjusts the generation size dynamically to respect a certain decoding delay. We also enhance it to overcome the issue of acknowledgement loss. We then propose to apply our approach in a Home Area Network for Live TV and video streaming. Our solution provides Quality of Service and Quality of Experience for the end user with no additional equipment.

Finally, we focus on a more theoretical work on which we present a new Butterfly-based network for multi-source multi-destination flows. Our new scheme outperforms the traditional Butterfly one. We characterize the source node buffer size using the queuing theory and show that it matches the simulation results.

Résumé

Dans cette thèse, nous nous intéressons à l'application du codage réseau pour garantir une qualité de service (QoS) en termes de débit et de délai dans les réseaux sans fil multi-sauts.

Comme le support de transmission est partagé, les réseaux sans fil souffrent de l'impact négatif des interférences sur la bande passante. Il est alors intéressant de proposer une approche basée sur le codage réseau qui prenne en compte ces interférences durant le processus de routage. Dans ce contexte, nous proposons d'abord un algorithme minimisant l'impact des interférences pour des flux unicast tout en respectant la bande passante qu'ils exigent. Puis, nous le combinons avec le codage réseau et plus précisément le schéma d'"Alice et Bob" afin d'augmenter le nombre des flux acceptés. Par la suite, nous combinons l'algorithme résultant avec un contrôle de topologie pour améliorer davantage la gestion des interférences et optimiser les performances du réseau. Enfin, nous montrons par simulation l'intérêt de combiner les trois domaines : codage réseau, gestion des interférences et contrôle de topologie.

Nous abordons également la gestion du délai pour les flux multicast. Nous utilisons le codage réseau aléatoire et plus exactement le codage réseau basé sur les générations (GBNC) qui combine les paquets par bloc. La plupart des travaux portant sur le GBNC considèrent une taille de génération fixe mais à cause des variations de l'état du réseau (délai de bout-en-bout, pertes de paquets, congestion,...) le délai de décodage et de récupération du bloc de paquets peut varier, dégradant ainsi la QoS. Pour résoudre ce problème, nous proposons une méthode qui ajuste la taille de la génération de façon dynamique pour respecter un certain délai de décodage avec prise en compte des contextes réseau et contenu. De plus, nous améliorons notre approche pour contrecarrer les pertes des acquittements. Puis, nous proposons de l'utiliser dans un réseau de domicile pour la diffusion de vidéo à la demande. Notre solution améliore la qualité de service et la qualité d'expérience pour l'utilisateur final sans équipement additionnel.

Finalement, nous nous intéressons à un sujet plus théorique dans lequel nous présentons un nouveau réseau basé sur le schéma *Butterfly* pour des flux multi-sources multi-destinations. Notre nouveau schéma est meilleur que le réseau *Butterfly* traditionnel. Nous caractérisons la taille du buffer du nœud source en utilisant la théorie des files d'attente et montrons qu'elle correspond aux résultats de simulation.

Contents

List of Figures

Part I

Introduction and background

Chapter 1

Introduction

This chapter explains the context of the present thesis. It first gives the classification of wireless networks and their applications. Next, it details the reasons why a lot of effort was done to meet the recent applications requirements and cope with the wireless medium limitations. The main contributions are briefly summarized at the end of the chapter.

1.1 Context

These last decades, wireless networks have been deployed at a remarkable rate in different contexts: office buildings, universities, hotels and so on. The proliferation of lightweight wireless equipments made these networks the first solution to provide any-where any-time Internet access for the users. One of the first applications of wireless communications was cellular networks. Eventhough the core network is endowed with an infrastructure allowing wired communication, the transmission between access points and users' devices remains wireless.

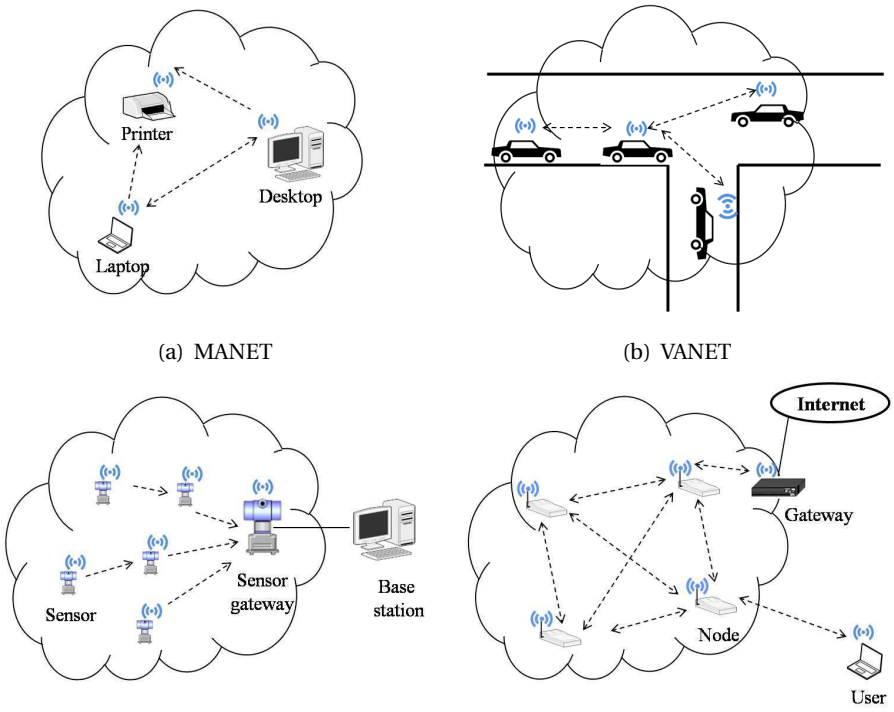
Setting an infrastructure is costly and sometimes impossible such as in a post-disaster context. Consequently, a new category of networks appeared, the wireless multi-hop networks. The nodes are self configured and build a network on their own. A source node sends information directly to the destination if it is located in its reception area, otherwise, the information is forwarded by other nodes hop-by-hop until the destination node. In general, there are several types of multi-hop wireless networks:

Mobile Ad hoc Network (MANET) configures the nodes and adapt them to their mobility in order to guarantee a continuous connection. An example of application of this kind of networks is rescue operations after a natural disaster. However, MANETs can also be used to simply connect two or more computers (see Figure 1.1(a)).

Vehicular Ad hoc NETwork (VANET) allows communication between vehicles with dynamic mobility. Its goal is to exchange information among the users and recover necessary information to manage the traffic (see Figure 1.1(b)).

Wireless Sensor Network (WSN) is used to collect measurements such as temperature and velocity. The nodes collaborate in order to deliver the information to a special node called sink or sensor gateway (see Figure 1.1(c)). This network is used in many applications such as environmental and habitat monitoring, indoor climate control, surveillance, treaty verification and intelligent alarms [8].

Wireless Mesh Network (WMN) broadens the coverage to a given area. It is mainly used when extending a wired network infrastructure is not feasible due to financial or geographical difficulties. Special wireless nodes collaborate to set a network in which any access to an external network, such as Internet, must be done by the gateway (a node that plays the role of an access point to Internet). Then, the users use the WMN such as a core network to get connected (see Figure 1.1(d)).



(a) MANET (b) VANET

(c) WSN (d) WMN

Figure 1.1: Miscellaneous types of wireless multi-hop networks.

Because of the inherent lossy wireless medium and device structure, wireless multi-hop networks are faced to different issues: low throughput, long delay (latency), weak security and limited energy power. On the other hand, the end-user applications are numerous more than ever before: Voice over IP (VoIP), video streaming, Live TV, file sharing, etc. These appli-

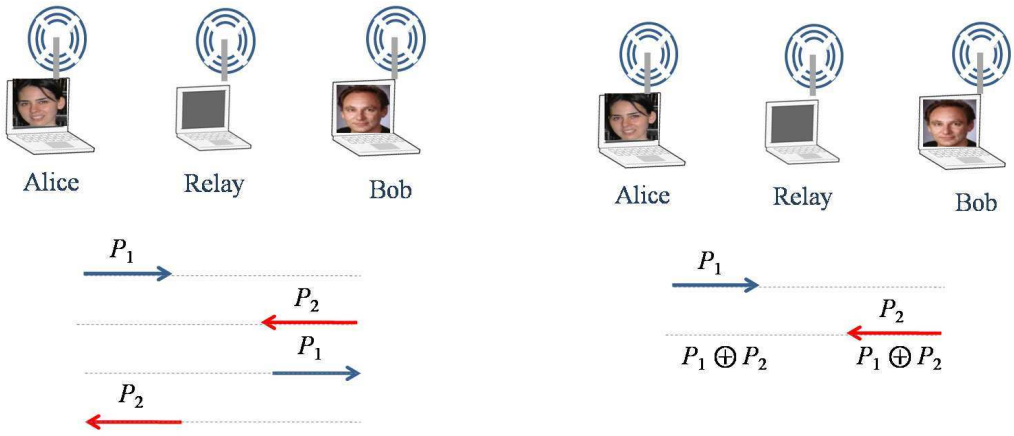
cations require high quality voice, pictures, videos and multimedia communications. This is why it is necessary to improve the features of such networks.

Providing an efficient and reliable communication while guaranteeing a Quality of Service (QoS) is challenging the wireless networks nature. QoS is the ability to provide a certain level of performance to the user's application. In networking, it consists on satisfying requirements on some aspects of a connection such as bit rate, loss and delay. Many efforts were done to improve the wireless network performance in terms of bandwidth and energy consumption, data compression and security.

The shared wireless medium reduces the quality of reception and the amount of received information. If, for instance, two nodes a and b transmit data to node c at the same time, node c can not understand the real information. This phenomenon is called interference. In this case, the bandwidth of links $a \rightarrow c$ and $b \rightarrow c$ i.e the amount of traffic that can be sent in a duration of time through those links, is used unproductively since a sequential retransmission is needed.

Some solutions try to prevent interference by (i) choosing an appropriate scheduling among the links, (ii) selecting routes that reduce the interference or (iii) adapting the signal modulation of a node and its closest nodes (neighbors) as done in the OFDMA approach.

A surprising way to avoid interference is to consider it as a good phenomenon and besides take the advantage of it in terms of bandwidth consumption. Indeed, let us take the "Alice and Bob" network depicted in Figure 1.2.



(a) without Network Coding

(b) with Network Coding

Figure 1.2: The Alice and Bob scenario.

In this scenario, there is one flow from Alice to Bob and another from Bob to Alice. When the intermediate node (relay) receives the data packet P_1 from Alice and P_2 from Bob, it simply forwards P_1 to Bob and P_2 to Alice. This scenario needs four transmissions (see Figure 1.2(a)). A new approach would allow the relay to smartly combine the packets, using an

exclusive OR (XOR) for instance, and broadcast the resulting combined packet $P_1 \oplus P_2$. Finally, Alice decodes the combined packet by XORing it with the already sent packet i.e. $P_1 \oplus (P_1 \oplus P_2)$ and thus recovers the original packet P_2. Similarly, Bob decodes $P_2 \oplus (P_1 \oplus P_2)$ and recovers P_1. Here, only three transmissions were needed (see Figure 1.2(b)). This approach that allows the intermediate nodes to mix the packets instead of simply forwarding them is called "Network Coding".

We have explained here a very simple scenario of Network Coding. A more sophisticated one is to combine many packets. For the combining operation, some coefficients are needed, they are taken from a finite set. For instance, when combining k packets $P_1, P_2, \ldots P_k$ using k coefficients $\alpha_1, \alpha_2 \cdots \alpha_k$, the combined packet is $\alpha_1.P_1 + \alpha_2.P_2 + \cdots + \alpha_k.P_k$ and its size is (see Figure 1.3):

$$traditional\_packet\_size + k \times packet\_IDentifier\_length + k \times coefficient\_size.$$

The overhead due to the slightly longer header is negligible compared to the Network Coding benefits. In [9], the overhead is estimated around 3%.

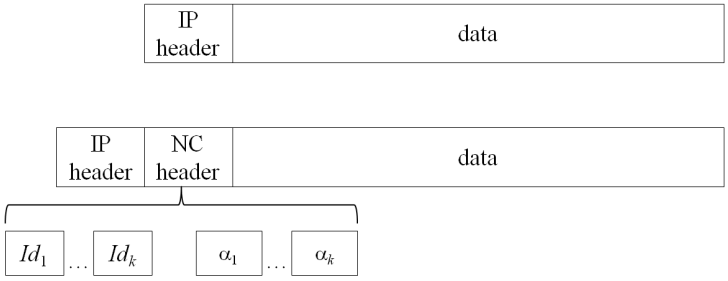


Figure 1.3: Packet structure of a native packet (top) and a coded packet (bottom).

Combining packets can be done using space-coding (coding over links) or time-coding (coding a packet with previous ones). In the wireless network depicted in Figure 1.4, the source node s wants to send two packets to the destinations node y and node z. We set the base field to $GF[2^2]$ i.e. the coefficients are chosen from the set [0, 1, 2, 3] and any operation on them is made $modulo[4]$. Node s timely combines P_1 and P_2 in order to build two combinations (or combined packets) $C_1 = P_1 + P_2$ and $C_2 = P_1 + 3P_2$ which are sent to node t and node u respectively. Node t forwards C_1 to both node x and node y. Similarly, node u forwards C_2 to both node x and node z. Node x combines C_1 and C_2 (space coding) to build one combination $C_3 = 2C_1 + C_2 = 3P_1 + 5P_2$, and using $modulo[4]$, $C_3 = 3P_1 + P_2$. This combination is then broadcast to both node y and node z. The destination node y holds two combinations C_1 and C_3. To decode the original packets, it resolves the system having two variables P_1 and P_2 and the coefficient matrix $\begin{pmatrix} 1 & 1 \\ 3 & 1 \end{pmatrix}$. A Gaussian elimination is done to recover the values of P_1 and P_2. Similarity node z uses C_2 and C_3 to recover P_1 and P_2. We notice that if Network Coding was not applied, node x would send two packets instead of one and thus would consume additional bandwidth.

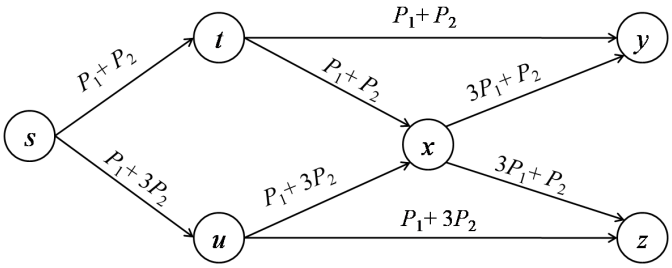


Figure 1.4: A Network Coding scenario using both time and space coding.

By introducing this new paradigm of mixing data over nodes, Network Coding widens its application areas. Indeed, it increases the throughput in wireless mesh networks; reduces energy consumption in wireless sensor networks; increases the capacity of Many-to-many broadcast networks; improves the data availability in P2P file sharing and Content Digital Networks (CDNs) and strengthens the network robustness.

1.2 Challenges and contributions

In this thesis we deal with the application of Network Coding to guarantee QoS in wireless multi-hop networks. To achieve this goal, we focus on three main points cited in the above discussion. The interference management to optimize the bandwidth consumption, the adjustment of Network Coding parameters to improve the network performance such as throughput and delay and the capacity increase using a new network scheme. More details about these three ideas are given below.

1.2.1 Bandwidth optimization

As shown in Figure 1.4, the initial goal of Network Coding was to reduce the number of transmissions and hence the bandwidth consumption in the network. However, instead of applying the coding after routing (choosing the path of) the flows, some approaches consider it during the routing phase in order to optimize the bandwidth. Other routing algorithms achieve this aim by minimizing the interference. This last has a dramatic impact on the network performance.

In a wireless network, the medium is shared by many links therefore a new flow is subject to inter-flow interference caused by the existing flows and intra-flow interference caused by the flow itself. If the interference is not considered when routing, the real bandwidth of an accepted flow can vary according to the network load. When an interference occurs, a retransmission can be necessary and in that case, the bandwidth decreases. Besides, when a flow asks for a bandwidth and we reserve the same bandwidth on each link of the path, this

is not enough, because two successive links, for instance, can not transmit at the same time and thus their bandwidth is divided by two. Finally, a flow gets a bandwidth lower than what it asked for, the QoS is then degraded.

To find the route minimizing the interference impact, a commonly used method consists in using a clique-based model. It finds exactly the best path for interference consideration and gives an upper bound of the network capacity. Due to Network Coding advantages, it is interesting to combine it with the interference consideration in the routing phase. This would take into account the interference impact and consider at the same time a coding scenario involving more than one flow such as the Alice and Bob scheme.

- We improve (Chapter 3) an existing algorithm named ROCX [10] (*Routing with Opportunistically Coded Exchanges*) by introducing the notion of interference. ROCX uses Linear Programming for routing while considering the Alice and Bob scheme of Network Coding. Said differently, it reroutes the flow in order to make them crossing as much as possible while minimizing the bandwidth consumption. As for the interference management, we enhance the clique-based model and apply it in our new algorithm, I2ILP (*Intra-flow Interference aware Linear Programming*), that uses Linear Programming to find the appropriate path from a source to a destination while minimizing the intra- and inter-flow interference impact. Using I2ILP, when a flow is admitted in the network, its offered bandwidth is kept stable and thus the QoS is guaranteed. However, the allocated path does not change, in other words, the notion of rerouting is not taken into account. We then propose a combination of ROCX and I2ILP, named IROCX (*Interference aware Routing with Opportunistically Coded Exchanges*), which introduces the I2ILP interference constraints to the ROCX linear program. Consequently, IROCX takes the advantages of Network Coding, interference consideration and rerouting.

In addition to the approaches given above, the advanced technology on hardware devices allow the wireless interfaces to control their transmission power. This new feature modifies the reception range of a node and thus the number of its neighbors. The interference impact changes accordingly. This technology referred as Topology Control, allows the nodes to change the network topology (graph) in order to minimize the interference. Combining the three fields would still increase the individual improvement of each approach taken separately.

- We combine (Chapter 4) IROCX with a Topology Control approach. This algorithm assumes that the nodes have several levels of transmission power. They are thus able to adapt the network topology in order to maximize the number of admitted flows while respecting their QoS in terms of bandwidth. Mixing the three fields (Network Coding, interference consideration and Topology Control) has led to an algorithm, named TC-IROCX (*Topology Control and Interference aware Routing with Opportunistically Coded Exchanges*), that outperforms the aforementioned algorithms.

1.2.2 Delay consideration

Most of the works on Network Coding applied it to increase the throughput. The trade-off between throughput and delay engenders an increase of throughput at the cost of delay. This is why the delay, which is a crucial performance parameter, should be well monitored when applying a Network Coding based approach.

From the above explanations, we can notice that Network Coding has various parameters: the way of coding the packets, the base field size, the number of packets to combine, etc. The goal here is to use some of them to manage the delay of receiving the packets and decoding them. Indeed, when a set of k packets (referred as generation) are combined all together, the destination in our system can not exploit them until the reception of k independent combinations. The application layer uses only native (not coded) packets and thus has to wait until the end of the decoding phase.

However, reducing the number of packets to combine decreases the benefits of Network Coding in terms of robustness and throughput. Increasing it generates a long delay for the application layer. An interesting work is to manage both throughput and delay in order to take the advantage of Network Coding while considering a QoS in terms of delay for the end user.

- We study (Chapters 5 and 6) the issue of delay for multicast traffic by proposing a way to dynamically adapt the generation size to the network state variations (end-to-en delay, congestion, packet loss). The resulting approach, named DYGES (*DYnamic GEneration Size*), keeps the advantages of using Network Coding in terms of throughput and robustness against losses while respecting a decoding-delay threshold. We then improve our approach by considering an ACK retransmission mechanism. Usually using Network Coding, each node hears and analyzes all the packets sent by its neighbors. This property is mainly used for data packets. The approach we propose, named RDYGES (*Reliable DYnamic GEneration Size*), uses this feature to analyze the ACK packets and bring more robustness to the traffic transmission.
- We propose (Chapter 7) an architecture for content delivery in a Home Area Networks (HAN). This heterogeneous network comprises an Internet box transmitting a Live TV traffic to IPTVs. Applying DYGES increases the performance of such a network and has a significant impact on the user's video quality.

1.2.3 Capacity increase

The classical example for Network Coding is the well known Butterfly network in which a multicast flow is realized with less transmissions than in a non Network Coding context. However, this scenario is used with a deterministic code assignment algorithm, which means that the

coding/decoding scheme has to be agreed upon beforehand and needs full network topology. Random Network Coding is the key evolution that impelled Network Coding to be scalable and applied in distributed systems.

Convinced that Random Network Coding is beneficial, we believe that a complete redesign of multi-source multi-destination streaming algorithm based on butterfly sub-networks can increase the network capacity and consequently improve the throughput and delay.

- We present (Chapter 7) the GBFLY (*Generalized ButterFLY*) network that is applied for multi-source multi-destination flows. It outperforms the simple Butterfly network in terms of capacity, in other works, it has higher gains in terms of number of transmissions and max-flow min-cut compared to the simple Butterfly ones. Besides, we characterize the average source buffer size using the queuing theory.

1.3 Thesis Outline

We deal in this thesis with the QoS in wireless multi-hop networks. First, we manage the bandwidth optimization using Network Coding, Interference consideration and Topology Control. Then, we tackle another QoS criterion which is the delay using a more sophisticated coding scheme. Finally, we increase the network capacity by improving the butterfly scenario. Having these goals in mind and following the challenges and contributions given in the previous section, we organize the present thesis as follows:

- **Chapter 2** introduces the background of Network Coding. It first describes its theoretical aspects and step by step brings up new notions that made Network Coding applicable to practical networks. Some examples of implementations on various kinds of networks are given to illustrate the scope of Network Coding applications.

- **Chapter 3** presents the existing algorithm ROCX and then proposes improvements by considering the interference. It explains the clique-based interference model and defines new constraints for inter- and intra-flow interference consideration for the I2ILP algorithm. To be applied to the ROCX linear program, the interference constraints are modified according to the Alice and Bob scheme. Last, it is confirmed by simulation that the resulting linear program (IROCX) respects the notion of interference while having better performance than I2ILP.

- **Chapter 4** presents some preliminaries on Topology Control and explains why its usefulness on the Alice and Bob scenario seems not obvious. The way of computing the sets of interfering links is adapted to the multilevel transmission power while the linear

program is kept as it is in IROCX. The simulation results show that the resulting algorithm, TC-IROCX, has the best flow acceptation rate due to the transmission power adjustment.

- **Chapter 5** introduces the context of multicast traffic and the Generation-Based Network Coding (GBNC). It explains that the existing GBNC approaches do not consider the impact of the network state variations on the decoding delay. Next, it proposes an algorithm, DYGES, that adjusts the generation size according to the end-to-end packet delay. The simulations show the efficiency of DYGES in terms of throughput.

- **Chapter 6** extends DYGES by considering that the ACK packets are not consistently resent by the MAC layer. It explains how the proposed algorithm, RDYGES, takes the advantage of the node's opportunistic listening mode to manage the ACK packet loss. The simulation results show that the ACK packet loss is completely hided and consequently infer that RDYGES is more reliable than DYGES.

- **Chapter 7** summarizes the context of a HAN and its applications' requirements. Next, it gives an architecture suitable to Network Coding and an application of DYGES for a Live TV scenario. It shows, by simulation, both content- and network-awareness of DYGES and emphasizes its accuracy compared to the traditional approaches. It concludes that DYGES provides Higher Definition applications to end-users with no additional equipments in the HAN.

- **Chapter 8** discusses the performance issue of multi-source multi-destination flows. It presents the GBFLY network that is based on many Butterfly sub-networks on which Random Network Coding is applied. Since it has more shared links, the new network has higher gains in terms of number of transmissions and max-flow min-cut compared to the simple Butterfly ones. The average source buffer size is characterized using the queuing theory and matches the results obtained by simulation.

- **Chapter 9** concludes the thesis and sums up the main contributions. Furthermore, it offers suggestions for future consideration.

Chapter 2

Network Coding background

This chapter seeks to synthesize the main works in the field of Network Coding. It explains in details the idea of mixing the information to increase the network performance. Then, it describes the theoretical aspects of Network Coding and step by step brings up new notions that made Network Coding applicable to practical networks. Indeed, at the beginning, the developed coding schemes were centralized, then the use of random coefficients made possible to apply Network Coding in a distributed way. Due to its evolution, it is proposed for various network topologies such as multi-hop and cellular wireless networks and integrated to other layers such as the application and transport ones.

2.1 Introduction

Traditionally in a network, the intermediate nodes involved in routing a flow from a source to a destination forwards the data packets without any processing. Network Coding allows the nodes to mix packets for improving the network performance. However, this improvement is dependent on the number of packets to combine; the way of combining them and the amount of knowledge that needs a node about the network topology and state in order to process the right coding.

Network Coding was first proposed in a theoretical context in which the coding scenario was centralized and existing flows known beforehand. Its evolution led to use random parameters and hence propose a distributed version of Network Coding applicable to practical networks.

This approach improves the network performance in terms of throughput, it achieves the maximal network capacity known as the *max-flow min-cut*. Furthermore, it brings more robustness to packet loss. If a packet is lost, the network performance is weakly affected because the information contained in that packet can be recovered from any other combined

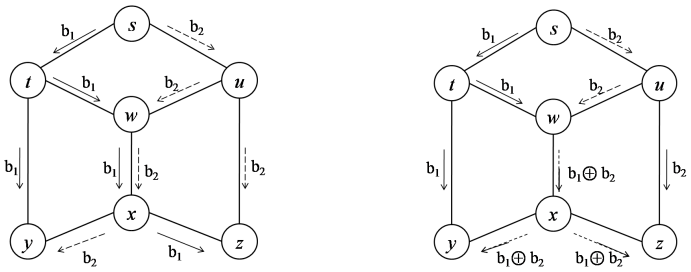
packet. In addition, Network Coding can be distributed and in that case needs only local information for the collaboration between nodes, it is thus scalable. As a coded packet is readable only if all the packets involved in the combination are known, the network security is significantly increased.

Due to all its advantages, Network Coding was proposed to be used in several network topologies (wired, cellular, *ad hoc* and so on) and for various applications (peer-to-peer, TCP, applications with QoS, etc.).

2.2 Theoretical approach

The notion of Network Coding was first introduced in the information theory by Ahlswede *et al.* in [1]. The main idea behind is that despite of considering a node as a simple relay of the information flow, Network Coding allows a node to combine more than one packet in one. The history of Network Coding is detailed in [11].

Let us explain how Network Coding works on the wired *Butterfly* network depicted in Figure 2.1:



(a) Without Network Coding (b) With Network Coding

Figure 2.1: Butterfly network with multicast from s to y and z [1]

The source node s wants to send two bits b_1 and b_2 to the destinations s and z (*multicast*). To do so, s sends b_1 to t and b_2 to u. Next, t sends b_1 to w and y . Similarly, u sends b_2 to w and z.

In a classical network (without Network Coding), assuming unit capacity links, w sends b_1 then b_2 to x (or the opposite). Finally, x sends b_1 to z and b_2 to y.

Using Network Coding, w computes a combination, for example an exclusive OR, of the two bits and sends it to x. Then, x forwards the combination to y and z. Node y (resp. z) receives b_1 and $b_1 \oplus b_2$ (resp. b_2 and $b_1 \oplus b_2$) and can consequently recover b_2 (resp. b_1) *via* the operation $b_1 \oplus (b_1 \oplus b_2)$ (resp. $b_2 \oplus (b_1 \oplus b_2)$).

We can easily deduce that the first approach needs ten transmissions while the second needs nine. The coding gain is then 10/9 for a wired context. If we consider a wireless *Butter-*

fly network, each transmission being received by all the neighbors of the transmitting node, the gain becomes 8/6. We notice that Network Coding has more impact in a wireless context.

This section explains the theoretical aspects of Network Coding and its evolution: from its analysis in terms of maximal capacity to the use of random values allowing its implementation and scalability.

2.2.1 Max-flow min-cut

The amount of information transiting in a network cannot exceed a certain value. This value is dependent of the maximal achievable network capacity. This last is defined by the *max-flow min-cut* and Network Coding allows to achieve it.

In graph theory, the *max-flow min-cut* problem consists in finding a maximum flow from a source to a destination. In networking, this is equivalent to find the paths that should take the data units sent by the source in order to maximize the flow capacity.

For example, in the *Butterfly* network depicted in Figure 2.1, assuming unit capacity links and considering the source s and the sink y, s can send at most two data units to y *via* $s \rightarrow t \rightarrow y$ and $s \rightarrow u \rightarrow w \rightarrow x \rightarrow y$. In this case, the *max-flow min-cut* capacity between s and y (number of independent paths from a source to a destination in the case of unit capacity links) is equal to two. Similarly, the *max-flow min-cut* capacity between s and z, independently of the flow from s to y, is equal to two.

Let us assume that the source s wants to send two data units to both destinations y and z. The maximum achievable capacity equals the smallest capacity for each destination taken separately. Hence, this capacity is equal to two. However, without Network Coding, the unit capacity link $w \rightarrow x$ cannot deliver simultaneously the two data units of the *multicast* flow sent by nodes t and u. Therefore, the maximum capacity from s to its destinations cannot be achieved, except if the capacity of link $w \rightarrow x$ is doubled.

On the other hand, using Network Coding allows to send one combination through the unit capacity link $w \rightarrow x$. Consequently, it becomes possible to simultaneously deliver two data units to both destinations and thus achieve the maximum capacity of two.

The remainder of this section tackles the feasability study of a communication scenario using Network Coding and achieving the *max-flow min-cut* bound.

2.2.2 Admissible code

In [1], the authors define for the first time Network Coding and give a characterization of the admissible coding rate.

The capacity of the graph edges is admissible if and only if the information rate of the source is less than or equal to the *max-flow min-cut* value obtained with this capacity. On the other hand, an $\alpha-$code is represented by a sequence of K transactions that describes a scenario of communication. Each transaction transfers an information (combination of data already received) from a sender to one or many receivers. An $\alpha-$code is $\alpha-$admissible if this scenario of transactions exists.

The main contribution of [1] is to prove that the set of admissible codes is the same as the set of $\alpha-$admissible codes. In other words, if there is a network traffic with a capacity less or equal to the *max-flow min-cut*, a corresponding $\alpha-$admissible code, using notably Network Coding, exists. The demonstration is done for one source node in a cyclic and acyclic graph.

For multiple sources, if the two information sources are independent, the solution is obtained by solving each problem individually (superposition principle). However the solution is not optimal in general. We introduce in the following paragraph various coding techniques.

2.2.3 Linear Code

There are various ways to combine packets. It was proved in [12] that for *multicast* traffic, *Linear-Codes Multicast* (LCM) are sufficient to achieve the maximum capacity bounds. Using LCM makes the encoding/decoding process easier to implement in practice and faster [13].

A linear combination is a sum of packets weighted by coefficients. For example, when combining k packets $P_1, P_2, \dots P_k$ using k coefficients $\alpha_1, \alpha_2 \cdots \alpha_k$, the combined packet is given by $\sum_{i=1}^{k} \alpha_i \times P_i$. There are many ways to choose coefficients. For example, a greedy algorithm was proposed in [12] for code construction. The following paragraphs presents simpler coding techniques.

2.2.4 Algebraic resolution

The way of coding packets to achieve the maximal capacity is not unique. The authors of [2] propose to solve the network coding problem in an algebraic way. They give necessary and sufficient conditions under which a set of connections is feasible in a network. In the *multicast* case, a connection transfers data (stochastic processes) from one source node to many destination nodes. In the example of Figure 2.2, node a sends three data units to node d. The transfer matrix from the input X to the output Z is $M = A \cdot (I - F)^{-1} \cdot B^T$, where the three matrices A, F and B are defined as follow:

- A: transfer matrix from the input processes to the intermediate ones;
- F: adjacency matrix of the graph in which every vertex is a link, relevant to intermediate nodes;

- B: transfer matrix from the intermediate processes to the output ones.

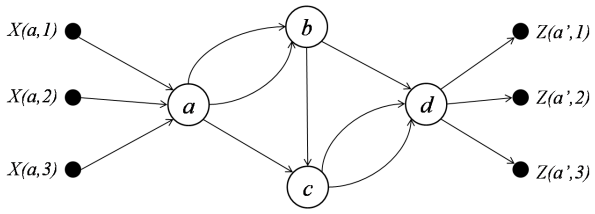


Figure 2.2: A network representing the stochastic processes [2]

The set of the output processes $Z = X \cdot M$ depends on the set of the input processes and the transfer matrix. In the case of one source and one destination, the destination can recover the original information by inverting the matrix M. The elements of the matrices are chosen through a resolution of an algebraic system that finds a variety[1] assigning values to those elements while keeping a non zero determinant.

For a network with one source and many destinations, a connection achieves its rate if and only if the *max-flow min-cut* is achieved for each connection relevant to the source and one destination. Thus, each destination has a knowledge only about its part of the matrix. The authors give a greedy algorithm to solve the system and prove its correctness.

For the case of many sources and many destinations, the system is solvable if and only if the sub-matrices in the diagonal of M are invertible and the remainder of the matrix is equal to zero. That means there is no interference between two information sent to two destinations.

As we can see it from Figure 2.1, the coding/decoding scheme has to be agreed upon beforehand and needs full network topology. This limitation makes Network Coding not straightforward to be implemented directly in real networks. Practical networks have many constraints: the packets are subject to random delays and losses, a centralized knowledge is difficult to obtain and the solutions should be simple with low complexity.

2.2.5 Random code

To cope with the aforementioned network constraints, the Linear Random Network Coding was proposed in [14]. This randomized coding allows a distributed transmission and a higher robustness against the packet loss. This approach consists on extracting the combinations coefficients *randomly* from a Galois field[2]. The coding process is characterized by the generation size and base field. The generation is the set of packets used to create the combinations.

[1] Algebraic variety: the set of solutions for a system of polynomial equations.

[2] Galois field is a finite field i.e. it contains only finitely many elements. For example in a field based on modulo 2^3, we have (6, 4, 1) + (5, 6, 3) = (3, 2, 4).

The base field is the basis of the vector space used to generate the coefficients of the combinations.

In the scenario depicted in Figure 2.3, six packets are randomly combined to generate eight coded packets of which two represent a redundancy. Two combined packets are lost, however due to redundancy, the receiver recovers the original data upon the reception of six combinations.

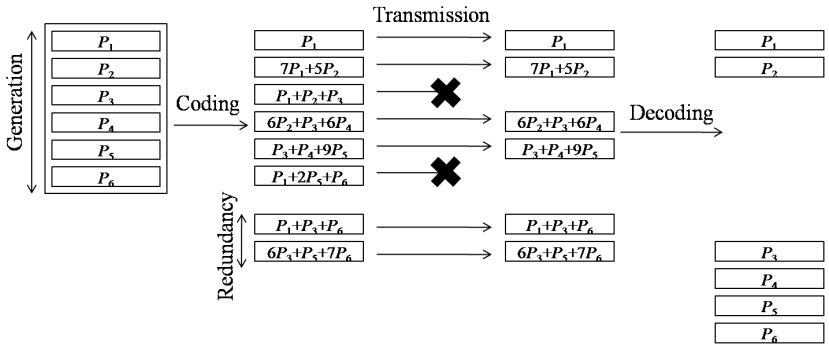


Figure 2.3: Coding/decoding phase

About the base field, in Figure 2.3, the first two packets are generated using the vectors (1,0,0,0,0,0) and (7,5,0,0,0,0) respectively. The base field should be large enough to guarantee the linear independence between the combinations.

Assuming this hypothesis, a redundancy can be added at any point of the network to cope with the packet losses. In addition, only a local knowledge is needed to determine the packets to combine. Indeed, the nodes do not need to have the knowledge of the whole transfer matrix. For the decoding process, each node receives packets which are linear combinations of source packets and it stores their coefficients into a matrix. Then, it has just to solve the algebraic system to recover the original packets.

Network Coding has many advantages such as improving the throughput and making the network more robust against losses. Therefore, it has miscellaneous applications in P2P wired, ad hoc, cellular and sensor networks [15].

In the next section, we will discuss about some implementations of Network Coding mainly in wireless networks.

2.3 Practical approach

Network Coding is used in various network topologies: multi-hop wireless networks (*mesh* and *ad hoc*), cellular networks and wired networks [15, 16]. Even if the first theoretical works on Network Coding dealt with wired networks, its impact on wireless networks is more signif-

icant. As shown in the example of *Butterfly* network depicted in Figure 2.1, Network Coding is more advantageous in a wireless context and especially multi-hop networks. On that account, works on Network Coding mainly focus on this kind of networks.

This section introduces some implementations of Network Coding at the routing layer for various topologies: probabilistic Network Coding and random Network Coding applied to multi-hop wireless networks and Network Coding applied to cellular networks. In addition, it presents Network Coding scenarios on other layers: P2P application, TCP application, QoS application.

Note that Network Coding is also applied at the physical layer, this version is called *Physical Layer Network Coding* (PLNC) [17]. It is quite different from traditional Network Coding that deals with discrete symbols (not signals). This approach is not in the scope of this thesis and will not be studied.

2.3.1 Probabilistic Network Coding

In this category of approaches, we mainly present COPE protocol and its variants [4, 18, 19] in addition to an optimal algorithm, ROCX, that uses Linear Programming and considers a particular case of COPE, the well-known Alice and Bob scenario.

COPE: It is the first architecture for Network Coding in a wireless *mesh* network [3]. It applies Network Coding for both *unicast* and *multicast* flows. It allows a node to combine (XOR) many received packets in one. However, the intermediate nodes handle with only native packets. Consequently, they have to decode all the packets they receive.

We use the word probabilistic because for coding, the node computes the probability that the packets involved in the combination will be decoded by the receiver. If it is greater than a certain threshold (0.8) the node sends a combined packet, otherwise, it sends a native packet. To compute this probability, the node needs the delivery probability of the links it will use for the transmission and should know which neighbor holds which packet. To do so, each node sends information to update the knowledge of its neighbors. COPE adds a header between the frame and the IP headers, it allows the node to:

- Give the identifiers of the packets combined in the data field;
- Send ACKs of the native packets it recovered;
- Send its reception report (the native packets contained in its packets pool).

Figure 2.4 shows that a node can have many possible combinations. Node b has to send P_1, P_2, P_3 and P_4 to nodes a, b, c and d in this order. It also knows that each of nodes a, c, d contains in its packets pool $[P_3, P_4]$, $[P_1, P_4]$ and $[P_1, P_3]$ respectively. Node b has various possibilities to combine the packets and should choose the one that maximizes the number of

packets delivered within a single transmission, here, $P_1 \oplus P_3 \oplus P_4$. Indeed, this combination makes the three nodes decoding and recovering the original packets.

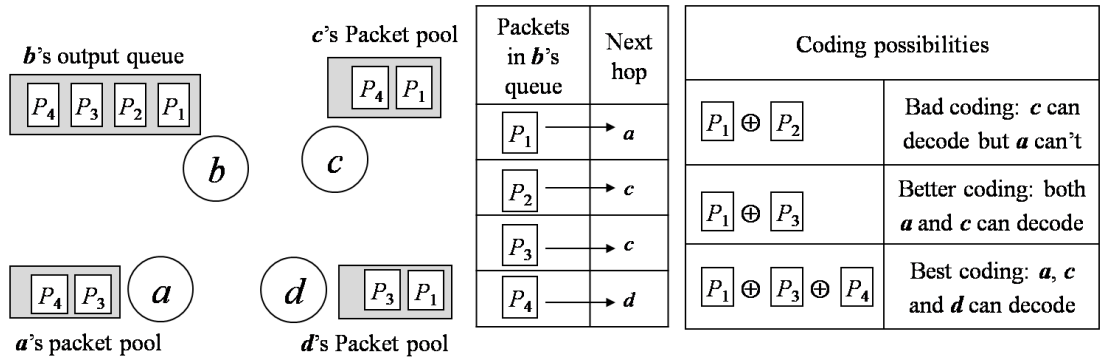


Figure 2.4: COPE coding scheme [3].

The implementation on an *ad hoc* network shows that throughput using COPE can be increased by a factor 3 (resp. 1.33) for UDP (resp. TCP) flows. COPE is sensitive to flow symmetry: the more symmetrical the flows are, in other words, the flow throughput from a source s to a destination d is close to the flow throughput from d to s, the more efficient is the coding.

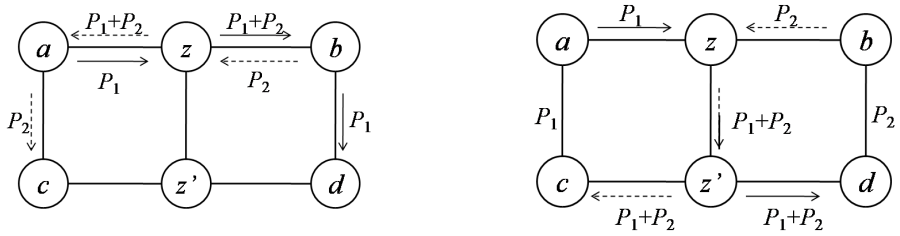
Due to its advantages, COPE is one of the most well known protocols among the Network Coding applications. Furthermore, miscellaneous works deal with COPE-like protocols proposing new variants.

BFLY: Using COPE, each intermediate node decodes the received packets before recoding and transmitting them. Hence, no combined packet is directly forwarded. Based on this remark, BFLY an improvement of COPE was proposed in [4]. The authors observed that they can increase the gain in a *Butterfly* network if they allow the forwarding of a combined packet (without decoding it).

Indeed, in the wireless network depicted in Figure 2.5, node a wants to send packet P_1 to node d, and node b wants to send P_2 to node c. We notice that COPE needs five transmissions while BFLY needs four. We recall that when node z, for instance, broadcasts the packet $P_1 \oplus P_2$ through the wireless links $z \rightarrow a$ and $z \rightarrow b$, it is considered as one transmission.

The authors propose to combine BFLY and COPE using a probabilistic model to characterize the achievable throughput. The throughput depends on the delivery probability of the links that are used. Therefore, the node evaluates whether the BFLY Network Coding worth it in terms of link delivery probability, and then it sends a combined packet, otherwise it checks with COPE and if it is not worthy, it sends a native packet.

DCAR: [18] extends the COPE scheme by considering two source nodes of which each one (i) is more than two hops far from the other source, (ii) has a destination neighbor of the



(a) COPE (b) BFLY

Figure 2.5: Coding scheme using COPE and BFLY [4]

other source and (iii) has a common node in its path to the destination with the other source flow (see Figure 2.6). During the routing phase, the source indicates the list of its neighbors, the intermediate nodes then check if they are involved in flows respecting the DCAR scheme and code the packets accordingly.

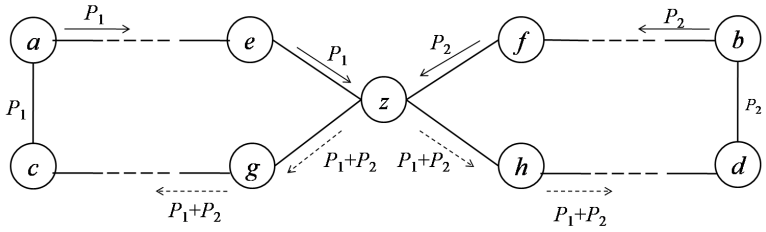


Figure 2.6: DECAR coding scheme

COPE with delay: Unlike the other COPE variants that are based on the improvement of the coding scenario, the approach presented in [19] modifies the way of selecting the packets to combine. In fact, in order to consider QoS in terms of delay, each packet has a deadline that must not be exceeded. Thus, when selecting packets to combine, the ones having a close deadline are prior to be coded.

ROCX: In the previous works, the routing decision is made independently of the coding phase. The authors of [10] enhance the Network Coding gain by taking into account the coding opportunities during the routing process. The *Routing with Opportunistically Coded eXchanges* (ROCX) algorithm is applied for mesh networks, it uses Linear Programming (LP) to introduce the constraints of coding while minimizing the total number of transmissions.

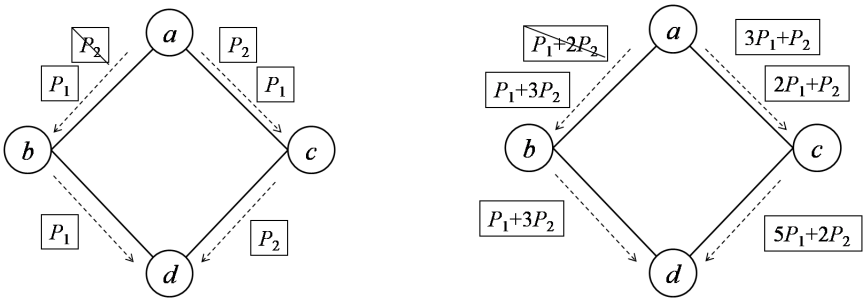
This approach was proved to be efficient especially when the paths are long but has some drawbacks. LP has a high complexity and can not be applied in networks made up of low computation devices. It is not scalable because the LP complexity grows exponentially according to the number of nodes. Moreover, each node must know which traffic is passing through which link all over the network, this condition is difficult to satisfy in practice. Another drawback is that ROCX is based only on the "Alice and Bob" scheme illustrated in Figure 1.2. Furthermore, there is no opportunistic listening in this scenario, in other words, a

node considers only the packets that are sent to it even though it can hear other packets in the network. ROCX can not exploit opportunities of another scheme like the BFLY's one [4]. It was shown in COPE that the opportunistic listening is useful to increase the throughput [3].

2.3.2 Random Network Coding

To explain this category of coding, we present MORE protocol. ExOR is a protocol that uses opportunistic routing but does not apply Network Coding, MORE improves it by introducing Network Coding.

ExOR: The traditional routing for *unicast* traffic consists on forwarding a packet through the network hop by hop from one node to one node. [20] describes *Extremely Opportunistic Routing* (ExOR) a *unicast* routing technique to increase the throughput in wireless multihop networks. It is important to recall that ExOR does not apply Network Coding but does improve the network performance. Using ExOR, the source divides data into batches (generations), then broadcasts the packets while precising a priority among the forwarders (the next-hop neighbors responsible of data transfer). If the highest priority forwarder receives a packet, it waits a duration of time according to its ranking then sends an ACK. Otherwise, a lower priority forwarder is responsible of acknowledging that packet. Consequently, a forwarder would broadcast only packets not acked by higher priority nodes. In Figure 2.7(a) for instance, node *b* is prior than node *c*.



(a) ExOR (b) MORE

Figure 2.7: A scenario of transmission using ExOR and MORE.

MORE: ExOR increases the throughput and avoids simultaneous transmissions using a local scheduling. However, it is MAC-dependent, prevents spatial reuse and is not applicable to *multicast* traffic. An other opportunistic routing named MORE was proposed in [21], it introduces Network Coding to cope with these drawbacks. When a forwarder receives a packet, it checks whether it is innovative or not. A combination is innovative if it is linearly independent of the previous received packets which amounts to saying that it holds a new information. If so, the forwarder builds a random linear combination of all the received packets that belong to the current batch and sends it. When the medium is not available, MORE exploits the time to code packets in order to send the coded packet immediately when the medium

is available. When the destination receives enough combinations to decode, it sends an ACK that triggers the transmission of the following batch and makes the intermediate nodes flushing the packets of the current batch from their memories.

One of the main differences between ExOR and MORE is that ExOR needs a scheduling to build a certain ranking among the forwarders. For instance, in Figure 2.7(a) the source has to express that the forwarder b is more prior than the forwarder c. Another difference is that the forwarding needs coordination among the nodes. c has to wait in order to deduce that b received P_1 and did not receive P_2. MORE does not need coordination, nodes b and c send respectively their combinations to d that decodes and recovers the original data.

Probabilistic *versus* random Network Coding

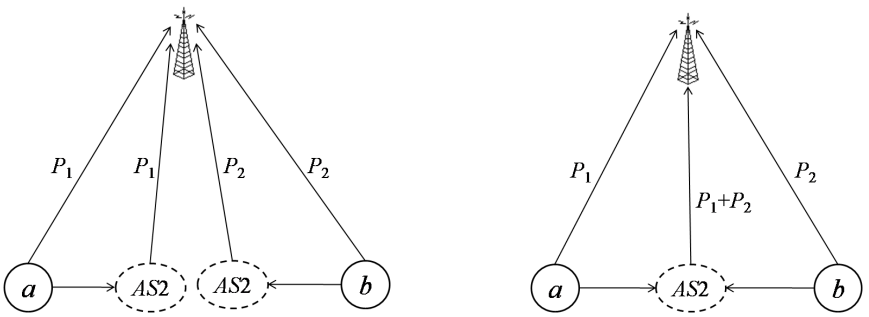
We compare here the most representative algorithms of probabilistic and random Network Coding, namely COPE and MORE. COPE is an *inter-flow* Network Coding because it combines packets from various sources. As for MORE, it is *intra-flow* since it combines only packets of the same source and more exactly relevant to one batch. MORE manages the time in a better way since it codes when the medium is unavailable. COPE requires more information control (reception report and ACK from each next-hop neighbor). The COPE's decoding process is simpler, it consists on a XOR operation, while MORE computes the inverted matrix using Gauss elimination. This concludes that both approaches have pros and cons and are hence applied in different contexts. However, it is possible to combine both inter and intra-flow coding. In [22] for instance, the authors propose the Intra-flow and Inter-flow MIXing (I^2MIX), a design which can benefit from integrating inter- and intra-flow wireless Network Coding.

2.3.3 Network Coding for cellular networks

Due to its benefits, Network Coding was proposed to be applied in cellular networks and more precisely in a *Distributed Antenna System* (DAS).

In [5], the authors prove that when using network coding, the system is more robust to transmission failure, uses less hardware resource (assisting antennas) in addition to a high spectral efficiency. Indeed, in Figure 2.8 we can see that when the system does not use Network Coding, each node needs an ASsisting antenna (AS) to replicate data. Using Network Coding reduces the number of assisting antennas and thus the number of transmissions. Consequently, there is less bandwidth expenditure.

If the nodes can cooperate with each other and there is no assisting antenna, Network Coding yields a lower system outage. We consider that an outage happens when the base station can not recover the data sent by a station. Figure 2.9 shows that for a fair cooperation



(a) Without Network Coding (b) With Network Coding

Figure 2.8: Transmission scenario in a DAS [5].

between all the users, each user, after sending its native packet, is offered three independent paths to deliver the redundant information against two without Network Coding. Without Network Coding, a, b and c send $[P_1; P_2]$, $[P_2; P_3]$ and $[P_3; P_1]$ respectively. Consequently, the packet P_1 for instance, can be transmitted through two paths. However, using Network Coding, a, b and c send for example $[P_1; P_2 \oplus P_3]$, $[P_2; P_1 \oplus 3P_3]$ and $[P_3; P_1 \oplus 2P_2]$ respectively. In this case, the packet $P1$ can be sent through three paths and then recovered. We can easily see that if three packets are lost, unlike for a traditional system, Network Coding guaranties to recover the three packets and avoid the outage whatever which packet is lost.

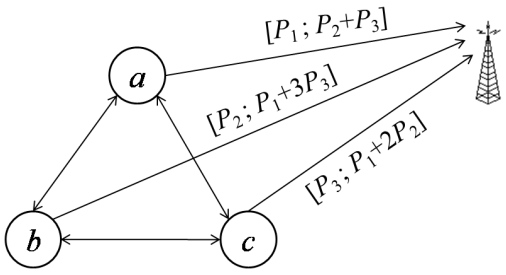


Figure 2.9: Cooperation with Network Coding [5]

2.3.4 Network Coding and TCP

Many approaches use Network Coding to improve the throughput at the routing layer. However, applying this paradigm is not limited to this layer and can be extended to other layers. For example, [6] presents *TCP-NC* an enhancement of *TCP* using Network Coding. This approach adds a sub-layer between the third and fourth layers in order to combine messages (segments). The main idea is to acknowledge the degrees of freedom (number of linearly independent combinations received by the destination) instead of acknowledging the decoded packets. Thus, the number of retransmissions decreases significantly in a network with lossy links.

A random Network Coding is chosen to generate independent combinations using a large enough Galois field size. Instead of sending an original packet, the source generates a random linear combination of the packets in its coding window and sends it. A redundancy is added to cope with the lossy links issue.

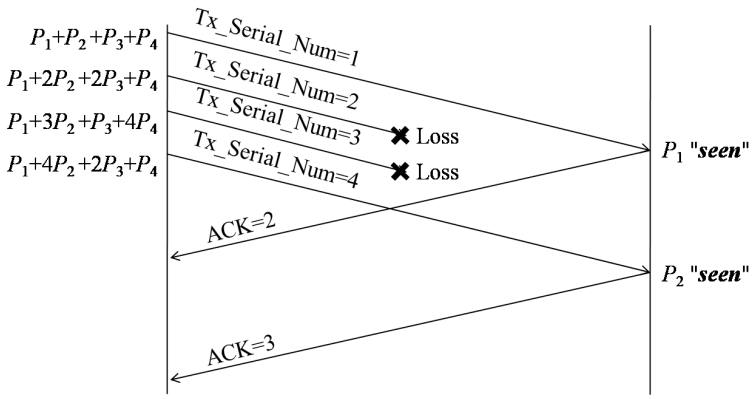


Figure 2.10: An example of coding and ACKs [6].

Upon the reception of a combination, the destination (sink) stores it in a matrix and uses the Gaussian elimination to on the one hand, find the degree to acknowledge and on the other hand, decode and recover the original packet if it is possible. Assuming independent combinations, every combination is associated to a freedom degree to acknowledge. It seems like to retrieve from a combination the smallest indexed packet that is not *seen* (acked) yet even if it is still not recovered. This technique adds redundancy because it can be impossible to recover the data before the end of the transmission of all the combined packets.

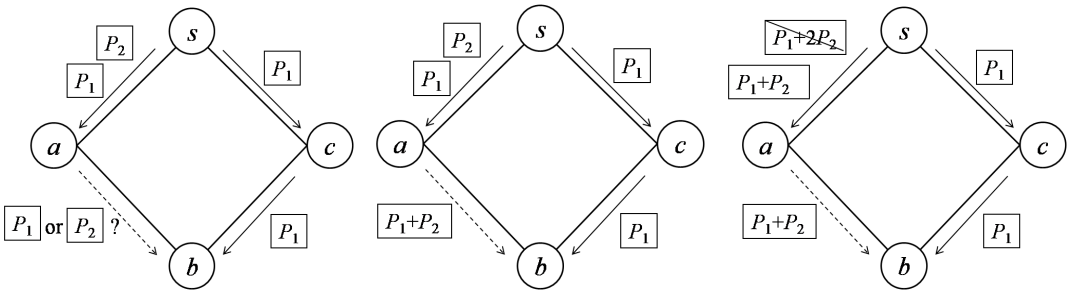
Let us explain the example depicted on Figure 2.10. The source sends four combinations, two of them are lost. When the destination receives the first packet, it considers that packet P_1 is *seen* and acknowledges it even though it is not decoded yet. Then, when it receives the second packet, since P_1 is already *seen*, P_2 is the *seen* packet and is consequently acked.

Simulations show that in a lossless context, *TCP-NC* can coexist with *TCP* without affecting fairness. With lossy links, *TCP-NC* is much more efficient than *TCP*. A variant of *TCP-NC*, namely *TCP-NCint*, allowing the intermediate nodes to re-encode data, was proved to be more efficient [6].

2.3.5 Network Coding on P2P

Network Coding is also applied at the application layer. For example, the authors of [23] describe a practical system based on Network Coding for file distribution to a large of cooperative users over a *Peer to Peer* (P2P) network. The most important advantage brought by Network Coding is that the nodes make decision for the packet propagation based only on local information. Indeed, in Figure 2.11 for instance, *a* downloads two packets P_1 and P_2

while c downloads P_1. b wants to download P_1 and P_2. Without Network Coding and without any knowledge of the transfers in the rest of the network, a can not decide the right packet(s) (P_1 or P_2) to transmit to b. Using random Network Coding, a sends a combination of P_1 and P_2 to b. On the other hand, b receives P_1 from c so it decodes and recovers P_2. These transmissions do not require the nodes to keep information about how the information is propagating in the network.



(a) Without Network Coding (b) With Network Coding at node A (c) With Network Coding at nodes A and S

Figure 2.11: A P2P network in which nodes only have local information.

Simulations show that using Network Coding improves the download rates [23]. Moreover, the system is more robust to the departure of the server and the peers containing the copies. Indeed, if they suddenly depart the system, the nodes can recover the file. For example, considering the topology of Figure 2.11, we assume that b wants to download a file from the source s. Node s uses Network Coding, it divides the file into blocks and sends combinations to the intermediate nodes a and c. Assuming the linear independence between the combinations (enough large base field), if the server departs the system before than a has the whole parts of the file, b can recover the file through the combinations received by a and c.

2.3.6 Network Coding with priority

Usually, Network Coding considers all the packets equal in priority. The authors of [24] describe a system that combines the idea of *Priority Encoding Transmission* (PET) with the Network Coding technique. For some applications like video streaming, the packets do not have the same priority because the first sequential packet has to be decoded before the following one.

PET is already used in traditional routed networks, it guarantees to recover the packets with respect of their priority [25]. The issue when using PET with Network Coding is that a contaminated packet (erroneous packet introduced by malicious users) can lead to the impossibility of correction of more prior packets that are combined with it. This problem can not be solved with the traditional codes used in networks which do not use Network Coding.

In order to be robust against losses and corruption (erroneous packets) while using Net-

work Coding, the approach described in [24] uses Gabidulin Codes [26] that are more sophisticated than Hamming codes[3]. The goal is to guarantee the recovery even when these codes are used jointly with random Network Coding. The simulations on a streaming media broadcasting scenario show that the performance are better and the delay is acceptable compared to a scenario without Network Coding.

2.4 Conclusion

In this chapter, we have introduced theoretical then practical aspects of Network Coding. Due to its several advantages, Network Coding was suggested to be applied in real networks. It was first proved that it allows to achieve the maximum network capacity. Next, various algorithms were proposed to find the most appropriate way of coding. Then, the Random Linear Network Coding allowed to implement Network Coding in a distributed way in practical networks.

Through some examples, we have explained the various implementations of Network Coding in different types of networks, namely, wireless multi-hop and cellular networks. Network Coding can be taken into account at several layers: routing, transport (TCP) and application (P2P, QoS). For each of these implementations, simulation results showed the efficiency of Network Coding.

Based on some works presented in this chapter, we propose in the remainder of this document enhancements and contributions that outperform those classical approaches. The next chapter focuses on ROCX algorithm and proposes some improvements to consider the notion of interference and hence guarantee the QoS in terms of bandwidth.

[3] A Hamming code is a linear error-correcting code. It can detect and correct bit errors based on the Hamming distance (minimum number of the bits that differs from a codeword to another).

Part II

QoS with bandwidth optimization

Chapter 3

Network Coding with Interference Consideration

The previous chapter gave an overview of the works on Network Coding and showed the importance of this mechanism. In this chapter, we are concerned by a particular applicative approach of Network Coding on wireless mesh networks, namely ROCX algorithm proposed by *Ni et al.* [10]. ROCX increases the Network Coding gain by taking into account the coding opportunities during the routing process. It is based on the 'Alice and Bob' scheme and uses Linear Programming. However, it does not take into account the bandwidth limitation and the interference impact. In a wireless network, a flow is subject to inter-flow interference caused by the existing flows and intra-flow interference caused by the flow itself. For a flow requiring a given bandwidth, its QoS could not be respected if the interference impact is not taken into account beforehand. We first improve an interference model based on clique computation and propose I2ILP a routing algorithm that considers both kinds of interference. Then, we combine it with ROCX and obtain IROCX that maximizes the benefits of Network Coding while considering the interference impact. The simulation results show its effectiveness.

3.1 Introduction

A wireless multi-hop network is a decentralized network without a preexisting infrastructure. Each node participates in routing data of other nodes besides transmitting and receiving its own data. The routing process consists in determining the path along which the nodes will forward the data packets from a source to a destination node. Using Network Coding, the intermediate nodes are not limited to forward the data packets but can mix them in order to increase one of the most precious resources in wireless networks: Bandwidth. Due to its advantages in terms of bandwidth, reliability, security and energy consumption, Network Cod-

ing has many applications in P2P wired networks, ad hoc, cellular and sensor networks [15].

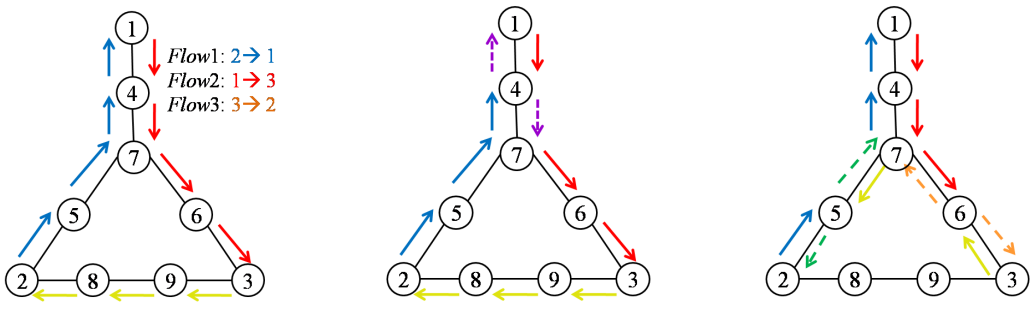
An interesting work, namely Routing with Opportunistically Coded Exchanges (ROCX), was proposed in [10]. It uses Linear Programming to find which flows can be mixed in order to maximize the network performance. Its main advantage is that the routing decisions are made with the awareness of coding. However, this work does not deal with neither the bandwidth constraints nor the interference impact.

Wireless networks suffer from the interference phenomenon. The bandwidth is reduced because two links can not transmit simultaneously if they interfere with each other. To compute the exact sets of mutually interfering links, we improve the clique based interference model. Next, we apply it in the shortest path linear program generating the Intra-flow Interference-aware Integer Linear Programming (I2ILP) algorithm.

The final algorithm we propose here, namely Interference-aware Routing with Opportunistically Coded Exchanges (IROCX), is an enhancement of ROCX by adapting the I2ILP clique constraints to the Network Coding case and integrating them to guarantee the interference awareness. A description of ROCX is given in the next section.

3.2 Network Coding with ROCX

A Network Coding issue is to define which packets have to be combined on which link in order to maximize the network performance. The theoretical works on Network Coding [12, 2] made possible its application in practical networks. Consequently, many practical approaches were proposed such as COPE [3] and BFLY [4].



(a) Shortest path (b) COPE (c) ROCX

Figure 3.1: An example with shortest path, COPE or ROCX algorithms.

Mostly, in the practical works, the routing decision is made independently of the coding phase. The authors of [10] enhance the Network Coding gain by taking into account the coding opportunities during the routing process. The proposed algorithm (ROCX) is applied for mesh networks. It uses Linear Programming (LP) to introduce the constraints of coding

while minimizing the total number of transmissions and taking into account the links delivery probabilities. It is based on the 'Alice and Bob' scheme presented in Figure 1.2 of Chapter 1. Figure 3.1 shows the efficiency of ROCX compared to COPE and traditional routing. There are three flows in the network. The dashed arrows indicate the combined packets.

Compared to traditional routing, COPE allows node 4 to combine two packets reducing the number of transmissions by 1. In this case, routing is first done by choosing the Shortest Path, then according the coding possibilities, coding is processed. As for ROCX, it reroutes the flow from 3 to 2 in order to increase the opportunity of combining the packets. Hence, nodes 4,5 and 6 combine the packets and lead to decrease the number of transmissions by 2.

For simplicity of notation, we rename Alice, Bob and the relay as nodes 2,3 and 1 respectively. Let $r_{i,j}$ be the delivery probability of link $i \rightarrow j$. Considering that node 2 (resp. node 3) wants to send one packet to node 3 (resp. node 2) through node 1 that uses Network Coding, the authors proved that the expected number of transmissions for a successful exchange with coding is (the gain with coding is the last term):

$$\frac{2}{r_{1,2}} + \frac{2}{r_{1,3}} - \frac{1}{r_{1,2} + r_{1,3} - r_{1,2} r_{1,3}} \tag{3.1}$$

Let N be the number of nodes and T the number of traffic flows (also considered as the set of flows). For a given flow t requiring an amount of traffic M_t, s_t and d_t are respectively its source and destination nodes. $r_{i,j}$ is the two-way successful delivery probability between i and j. For link $i \rightarrow j$, $x_{i,j}$ is its total amount of traffic of which $u^t_{i,j}$ is the part relevant to flow t. For the traffic on link $k \rightarrow i$ with next hop j, $w_{k,i,j}$ is its total amount of which $m^t_{k,i,j}$ is the part relevant to flow t. $c_{k,i,j}$ is the amount of traffic between k and j that can be coded at i. The linear program aims at finding $m^t_{k,i,j}$ that satisfies the following objective function :

$$\text{Minimize} \sum_{i,j=1}^{N} \frac{x_{i,j}}{r_{i,j}} - \sum_{i,j,k=1}^{N} \frac{c_{k,i,j}}{r_{i,k} + r_{i,j} - r_{i,k} r_{i,j}}$$

The constraints are given in [10] as follows:

Variables definition

$$\begin{cases} \sum_{t=1}^{T} u^t_{i,j} - x_{i,j} = 0 \\ \sum_{t=1}^{T} m^t_{i,j,k} - w_{i,j,k} = 0 \\ \sum_{i=1}^{N} u^t_{s_i,i} - M_t = 0 \\ \sum_{i=1}^{N} u^t_{i,d_i} - M_t = 0 \end{cases}$$

Flow conservation

$$\begin{cases} \sum_{k=1}^{N} m^t_{i,j,k} - u^t_{i,j} = 0, \ j \neq d_t \\ \sum_{k=1}^{N} m^t_{k,i,j} - u^t_{i,j} = 0, \ i \neq s_t \\ \sum_{i=1}^{N} u^t_{i,s_i} = 0 \\ \sum_{i=1}^{N} u^t_{d_i,i} = 0 \end{cases}$$

Coding

$$\begin{cases} c_{k,i,j} - w_{k,i,j} \leq 0 \\ c_{k,i,j} - w_{j,i,k} \leq 0 \\ c_{k,i,j} = c_{j,i,k} = 0 \end{cases}$$

Even if it considers Network Coding and rerouting, ROCX formulation does not take into account neither the notion of interference nor the bandwidth limitation of a link. Before solving this problem, we describe in the next section some preliminaries on the interference consideration and present the I2ILP algorithm.

3.3 Interference Consideration

In wireless networks, a node shares the medium with all its neighbors. Even if two paths have no node in common, interference can occur between them consequently reducing their throughput. This is known as inter-flow interference. Besides, when a link interferes with another link of the same path, the interference is said to be intra-flow. When flows have bandwidth requirements, it is necessary to take into account this interference during the routing process. Indeed, if the network depicted in Figure 3.2 were wired, it is clear that the available bandwidth for a flow from a to c would be equal to the least of the bandwidths of links $a \rightarrow b$[1] and $b \rightarrow c$. Because the network is wireless, the available bandwidth must be divided by two since both considered links cannot transmit simultaneously.

Most of the routing algorithms with QoS do not take into account the interference caused by the new flow with itself. Consequently, the new flow can be accepted while the required bandwidth is not available on the chosen path. The clique based interference model is an interesting way to find the exact sets of mutually interfering links.

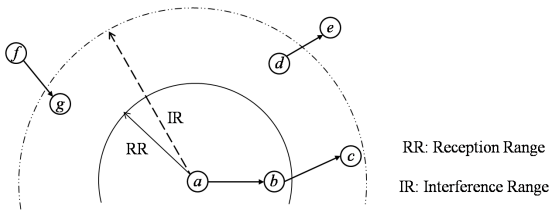


Figure 3.2: Intra-flow and inter-flow interference.

3.3.1 Interference and Widest-Path

Let us take for example Dijkstra's widest-path algorithm: since the path width is equal to the least bandwidth on its links, we can see on Figure 3.3 that the algorithm is not optimal in a wireless context. Indeed, there are two paths from a to v with the same length. The weights of the links represent their available bandwidth. The path chosen by Dijkstra's algorithm is on the right because the smallest bandwidth is equal to 90, versus 80 for the one on the left. However, there is more intra-flow interference on the chosen path. For example, when link

[1] In the remaining, both $a \rightarrow b$ and (a,b) notations represent a link from node a to node b and will be used interchangeably.

$h \rightarrow i$ is transmitting, the seven links $d \rightarrow e$, $e \rightarrow f$, $f \rightarrow g$, $g \rightarrow h$, $i \rightarrow j$, $j \rightarrow k$ and $k \rightarrow v$ should be inactive. Along the left path, when $o \rightarrow p$ is transmitting, only six links $l \rightarrow m$, $m \rightarrow n$, $n \rightarrow o$, $o \rightarrow q$, $q \rightarrow r$ and $r \rightarrow s$ should be inactive. If the bandwidth required by the flow is equal to 14, then the chosen path is not valid, because the real bandwidth cannot exceed $100/(7+1) < 14$. On the other hand, the left path satisfies the flow request since the real bandwidth is $100/(6+1) > 14$.

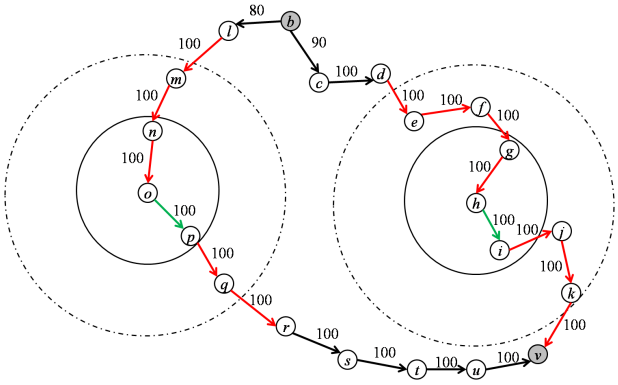


Figure 3.3: Finding a path with interference consideration

3.3.2 Mutually Interfering Links

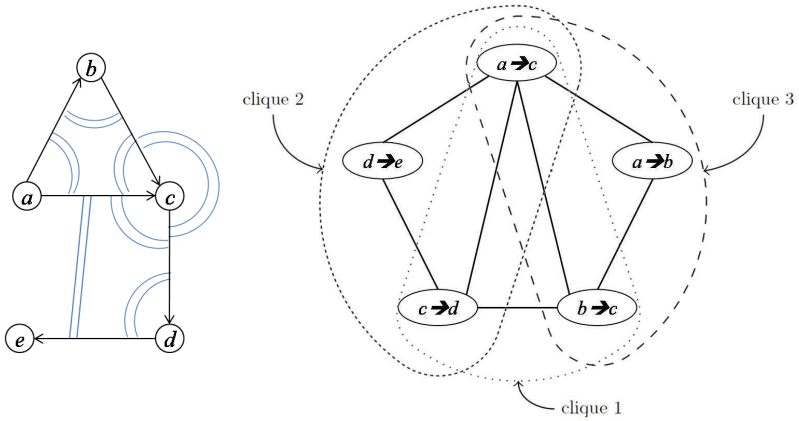
We consider a network as a graph $G(V,E)$, with V and E the sets of nodes and links respectively. The conflict graph CG is an undirected graph in which the vertices represent links of the network G and the edges interference relations between links. Figure 3.4 shows an example of five nodes in which

- link $a \rightarrow b$ interferes with links $a \rightarrow c$ and $b \rightarrow c$;
- link $b \rightarrow c$ interferes with links $a \rightarrow b$, $a \rightarrow c$ and $c \rightarrow d$;
- link $a \rightarrow c$ interferes with links $a \rightarrow b$, $b \rightarrow c$, $c \rightarrow d$ and $d \rightarrow e$;
- link $c \rightarrow d$ interferes with links $a \rightarrow c$, $b \rightarrow c$ and $d \rightarrow e$;
- link $d \rightarrow e$ interferes with links $a \rightarrow c$ and $c \rightarrow d$.

We represent a set of mutually interfering links by a maximal clique in the conflict graph i.e. a complete sub-graph of CG that is not contained in another sub-graph. In this example, there are three maximal cliques. Finding all maximal cliques[2] in a graph is an NP-complete problem and the most efficient algorithm for finding all the cliques [27] is linear according to the number of cliques.

[2]We shall use the term "clique" instead of "maximal clique" in the following, for simplicity's sake.

There are other methods to determine the sets of mutually interfering links in a network, some based on a N-hops interference model, others on cross-layer approaches. The first category gives an approximation of the cliques in the conflict graph, whereas the second uses information from the MAC layer to find interfering nodes [28, 29]. However, most of the currently available off-the-shelf devices assume the use of a TCP/IP stack and thus are not adapted to use cross-layer methods [30]. To be effective, a node applying a routing algorithm with QoS in terms of bandwidth should have a local knowledge of its links but also a global view of the sets of mutually interfering links to which its links belong.



(a) Network graph (b) Conflict graph

Figure 3.4: Violation of the optimality principle in a wireless network [7].

Besides the cliques computation, finding the optimal path from a source to a destination node is also NP-complete. Indeed, it has been shown in [7] that the principle of optimality is not respected in a wireless network. In other words, the best path from a source to a destination is not necessarily the best one from the source to an intermediate node due to interference. Let us assume that all the links of Figure 3.4 have the same capacity C. The widest path from a to e is $a \rightarrow b \rightarrow c \rightarrow d \rightarrow e$ because its available bandwidth is $C/2$ since there are at most two links in the same clique. The path $a \rightarrow c \rightarrow d \rightarrow e$ is not the widest because its three links all interfere with each other and thus its bandwidth is equal to $(C/3)$. We can see that the widest path from a to e does not contain the widest path from a to c that is $a \rightarrow c$. We thus infer that the best path towards the destination is not the succession of the best paths towards intermediate nodes. This explains the high complexity of the problem. Indeed, a comprehensive solution consists in testing all the possible paths from the source to the destination. The following section presents the existing routing algorithms.

3.3.3 Routing algorithms

In the following, we present the main routing algorithms based only on the network layer, namely Dijkstra adapted to widest path, Shortest-Widest-Path (SWP), the algorithm based

on Integer Linear Programming (ILP) and Ad hoc SWP (ASWP). Each of these algorithms searches the best path from a source s to a destination d, using different criteria:

Dijkstra [31]	The best path is the widest in terms of bandwidth.
SWP [32]	The best path is the shortest having a bandwidth greater or equal to what is required by the flow.
ILP [33]	The best path is the shortest in terms of number of hops, it is possible to add other constraints to respect the QoS requirements.
ASWP [7]	The best path is the widest in terms of bandwidth with intra-flow and inter-flow consideration during the width computation.

Mostly, the width of a path from s to v is computed as follows:

$$width_{s,v} = \min_{(i,j)\in path_{s,v}} \left(S_{i,j}\right) \ . \tag{3.2}$$

It depends on the smallest residual bandwidth on it. The residual bandwidth of a link $i \rightarrow j$ is defined as:

$$S_{i,j} = C_{i,j} - R_{i,j} \ , \tag{3.3}$$

where $C_{i,j}$ is the initial bandwidth of $i \rightarrow j$ and $R_{i,j}$ the bandwidth already reserved by the existing flows on this link. This solution does not consider the interference impact.

ASWP takes into account both kinds of interference and uses the clique-based interference model. This heuristic uses the Bellman-Ford algorithm [34, 35] combined to k-best-paths approach. It is suitable for our problem, but considers the same initial capacity for all the links and this is not realistic.

3.3.4 Enhancement of interference consideration

When introducing a flow in the network, the routing algorithm must find a path on which:

- the interference generated by the new flow on those already accepted must not degrade their guaranteed bandwidths;
- the interference of the new flow on itself must not prevent it from getting the required bandwidth.

A redefinition of the link residual bandwidth by integrating clique constraints solves the first point. Let r_q be the utilization rate of clique q. It is the sum of utilization rates of all its links

$$r_q = \sum_{(i,j)\in q} \frac{R_{i,j}}{C_{i,j}} \ , \tag{3.4}$$

The residual bandwidth of link $i \rightarrow j$ is then a function of the highest utilization rate of the cliques to which the link belongs. It is defined as follows:

$$S_{i,j} = C_{i,j} \cdot \max\left(1 - \max_{\substack{q \in Q \\ (i,j) \in q}} (r_q), 0\right) \quad . \tag{3.5}$$

The second point implies to redefine the path width by considering the intra-flow interference of the new flow. We give a formula inspired by the ASWP approach [7, 36] to compute the path width. According to its position in the network, a node can have various link capacities with its neighbors. For a variable channel capacity, we adapt the computation of the path width. Let $path_{s,v}$ be a path from s to v. Its width, noted $width_{s,v}$, is a function of residual bandwidths divided by a quantity $z_q^{s,v}$ which is the number of links of the considered path which belong to clique q.

$$width_{s,v} = \min_{q \in Q}\left(\frac{1 - r_q}{z_q^{s,v}} \cdot \min_{(i,j) \in q \cap path_{s,v}} C_{i,j}\right) \tag{3.6}$$

According to these new definitions, we have adapted the routing algorithms listed above and kept the most efficient, namely I2ILP, the enhanced version of ILP.

3.3.5 I2ILP

The shortest path problem is modeled as a linear program (ILP [33]). We add in it clique constraints based on Equations 3.5 and 3.6 to consider the interference impact (see the last equation of the I2ILP linear program presented below).

For a network $G(V, E)$ having a set of cliques Q, each time we want to introduce a flow from s to d requiring a bandwidth M, the following program is solved.

Minimize $\sum_{(i,j) \in E} l_{i,j}$

s.t.

$$\begin{cases} \sum_i l_{i,k} - \sum_j l_{k,j} = 0 \ , & \forall k \in V \setminus \{s,d\}, \quad \forall (i,k),(k,j) \in E \\ \sum_i l_{i,s} - \sum_j l_{s,j} = -1 \ , & \forall (i,s),(s,j) \in E \\ \sum_i l_{i,d} - \sum_j l_{d,j} = 1 \ , & \forall (i,d),(d,j) \in E \\ \sum_{(i,j) \in q} \frac{R_{i,j} + M \cdot l_{i,j}}{C_{i,j}} \leq 1 \ , & \forall q \in Q \\ l_{i,j} \in \{0,1\} \ , & \forall (i,j) \in E \end{cases}$$

To each unidirectional link $i \rightarrow j$ of the network topology is associated an integer variable $l_{i,j}$. When the linear program is solved, the variables $l_{i,j}$ that are equal to 1 indicate the links of the best path from s to d.

I2ILP represents an upper bound for the network capacity since it always gives a feasible path if there is one in the network. Nevertheless, it does not use Network Coding nor rerouting. Indeed, rerouting makes a better distribution of the load in the network. Moreover, as we have seen it in the example depicted in Figure 3.1, this property is crucial to exploit the coding opportunities in order to achieve an efficient coding. In the next section, we cope with this problem by combining I2ILP and ROCX.

3.4 Our solution: IROCX

In this section, we first use the clique model detailed in Section 3.3 and explain how to adapt the I2ILP clique constraints to ROCX linear program. However, for the accordance between the two formulations, instead of handling the amount of traffic (in Kb), we handle the corresponding bandwidth (in Kb/s) but keep the same notation as ROCX.

The bandwidth consumed on a link cannot exceed its initial capacity. Having this in mind, we add the following bandwidth limitation constraint:

$$\frac{x_{i,j}}{r_{i,j}} \leq C_{i,j}, \quad \forall (i,j) \in E \tag{3.7}$$

that can be written as:

$$\frac{x_{i,j}}{r_{i,j} \cdot C_{i,j}} \leq 1, \quad \forall (i,j) \in E \tag{3.8}$$

However, we do not need the bandwidth limitation constraint because it is less restrictive than the clique constraint explained below. Indeed, using the bandwidth limitation constraint allows each link to use 100% of its capacity while using the clique constraint, a link shares the bandwidth with other links belonging to the same clique and thus its exploitable bandwidth is lower. For example, if a clique contains three links and two of them use 20% of their capacities, the third link can not exceed 60% of its capacity.

To consider the interference in our formulation, let us explain the example depicted in Figure 3.5. The network comprises four nodes in which the cliques are given as follows:

- Clique q_1: $a \rightarrow b, \quad b \rightarrow a, \quad b \rightarrow c, \quad c \rightarrow b$
- Clique q_2: $b \rightarrow c, \quad c \rightarrow b, \quad c \rightarrow d, \quad d \rightarrow c$

Let us assume two flows $a \rightarrow b \rightarrow c$ and $c \rightarrow b \rightarrow a$ with requested bandwidths equal to M_1 and M_2 respectively. Clique q_1 is concerned with links $b \rightarrow c$ and $c \rightarrow b$ while clique q_2 is concerned with links $a \rightarrow b$, $c \rightarrow b$, $b \rightarrow a$ and $b \rightarrow c$.

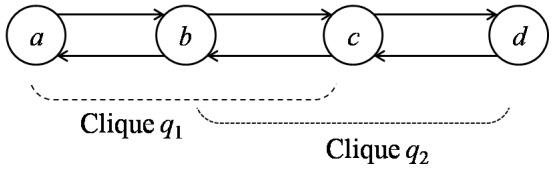


Figure 3.5: An example of network and its cliques.

Without Network Coding, each of cliques q_1 and q_2 decreases their utilization rate, after introducing the flows, by respectively:

$$\frac{M_1}{r_{b,c} \cdot C_{b,c}} + \frac{M_2}{r_{c,b} \cdot C_{c,b}} \tag{3.9}$$

and

$$\frac{M_1}{r_{a,b} \cdot C_{a,b}} + \frac{M_2}{r_{c,b} \cdot C_{c,b}} + \frac{M_2}{r_{b,a} \cdot C_{b,a}} + \frac{M_1}{r_{b,c} \cdot C_{b,c}} \ . \tag{3.10}$$

Using Network Coding, clique q_2 is not affected, while it is not the case for clique q_1 because it is concerned with both links $b \rightarrow a$ and $b \rightarrow c$ that transmit simultaneously the same (coded) information (with bandwidth M_3). These links will consume the equivalent of $M_3 = min(M_1, M_2)$[3] on one link (or twice a half) in clique q_1. Consequently, q_1 decreases its utilization rate by:

$$\begin{aligned} &\frac{M_1}{r_{a,b} \cdot C_{a,b}} + \frac{M_2}{r_{c,b} \cdot C_{c,b}} + \frac{M_2}{r_{b,a} \cdot C_{b,a}} + \frac{M_1}{r_{b,c} \cdot C_{b,c}} - \frac{M_3}{2 \cdot (r_{b,a} + r_{b,c} - r_{b,a} r_{b,c}) \cdot C_{b,a}} \\ &- \frac{M_3}{2 \cdot (r_{b,a} + r_{b,c} - r_{b,a} r_{b,c}) \cdot C_{b,c}} \end{aligned} \tag{3.11}$$

We generalize this idea in order to introduce it as the last constraint to the model. M_1 and M_2 will be represented by the variable $x_{i,j}$, which is the bandwidth of uncoded traffic passing through the links and M_3 by $c_{k,i,j}$, the bandwidth of coded traffic at node i.

We first rewrite the model as follows:

$$\text{Minimize} \sum_{(i,j) \in E} \frac{x_{i,j}}{r_{i,j}} - \sum_{\substack{(i,j) \in E \\ (k,i) \in E}} \frac{c_{k,i,j}}{r_{i,k} + r_{i,j} - r_{i,k} r_{i,j}}$$

[3] When mixing two flows, the throughput of the coded packets cannot exceed the minimum of the two flows throughputs.

$$
\begin{cases}
\sum_{t\in T} u^t_{i,j} - x_{i,j} = 0 \ , \forall (i,j) \in E \\
\sum_{t\in T} m^t_{i,j,k} - w_{i,j,k} = 0 \ , \forall (i,j),(j,k) \in E \\
\sum_{i,(s_t,i)\in E} u^t_{s_t,i} - M_t = 0 \ , \forall t \in T \\
\sum_{i,(i,d_t)\in E} u^t_{i,d_t} - M_t = 0 \ , \forall t \in T \\
\sum_{k,(j,k)\in E} m^t_{i,j,k} - u^t_{i,j} = 0 \ , \forall t \in T, \forall (i,j) \in E, j \neq d_t \\
\sum_{k,(k,i)\in E} m^t_{k,i,j} - u^t_{i,j} = 0 \ , \forall t \in T, \forall (i,j) \in E, i \neq s_t \\
\sum_{i,(i,s_t)\in E} u^t_{i,s_i} = 0 \ , \forall t \in T \\
\sum_{i,(d_t,i)\in E} u^t_{d_i,i} = 0 \ , \forall t \in T \\
c_{k,i,j} - w_{k,i,j} \leq 0 \ , \forall (k,i),(i,j) \in E \\
c_{k,i,j} - w_{j,i,k} \leq 0 \ , \forall (k,i),(i,j) \in E \\
c_{k,i,j} - c_{j,i,k} = 0 \ , \forall (k,i),(i,j) \in E
\end{cases}
$$

Finally, the clique constraint must ensure that the clique utilization is less than 1 to anticipate the interference impact and to guarantee that the flow will not suffer from a bandwidth degradation. By generalizing Equation 3.11 we obtain:

$$
\sum_{(i,j)\in q} \frac{x_{i,j}}{r_{i,j}\cdot C_{i,j}} - \sum_{\substack{(i,j),(i,k)\in q \\ k<j}} \frac{c_{k,i,j}}{2\cdot(r_{i,k}+r_{i,j}-r_{i,k}r_{i,j})\cdot C_{i,j}} + \frac{c_{k,i,j}}{2\cdot(r_{i,k}+r_{i,j}-r_{i,k}r_{i,j})\cdot C_{i,k}} \leq 1, \quad \forall q \in Q \tag{3.12}
$$

and by factorizing the two last terms, we get:

$$
\sum_{(i,j)\in q} \frac{x_{i,j}}{r_{i,j}\cdot C_{i,j}} - \sum_{\substack{(i,j),(i,k)\in q \\ k<j}} \left(\frac{c_{k,i,j}}{r_{i,k}+r_{i,j}-r_{i,k}r_{i,j}}\right)\cdot\left(\frac{1}{2C_{i,j}}+\frac{1}{2C_{i,k}}\right) \leq 1 \ , \quad \forall q \in Q \ . \tag{3.13}
$$

The condition $k < j$ is used to avoid counting the amount of the coded traffic twice ($c_{k,i,j} = c_{j,i,k}$).

This formulation allows IROCX to exploit the advantages of both ROCX and I2ILP. We sum up in Table 3.1 the routing algorithms features. I2ILP is *interference-aware*, ROCX is *coding-aware* and IROCX is both *interference*- and *coding-aware*.

Table 3.1: Comparison among I2ILP, ROCX and IROCX.

	I2ILP	ROCX	IROCX
Interference consideration	Yes	No	Yes
Network Coding	No	Yes	Yes
Rerouting	No	Yes	Yes

3.5 Performance Evaluation

We now present the numerical results for the network model proposed in the previous section. Our simulator, written in C++ under Opnet 15.0 and using the GLPK Linear Programming Solver [37], implements only the network layer.

The network consists of a connected graph of 20 nodes randomly placed on a surface of 600×600 $units^2$. The reception range radius (RR) is set to 100 $units$ and the interference range radius (IR) to 200 $units$. The initial bandwidth of a link is uniformly chosen between 500 and 1000 Kb/s and its delivery probability between 0.8 and 1. The flows are consecutively submitted to the network and live until the end of the simulation. Each of them starts from a source s to a destination d randomly chosen, and requires a QoS consisting of a bandwidth equal to 10 Kb/s. It is shown in [10] that ROCX performs better when the flows traverse more hops. Therefore, we focus our study on flows having shortest path lengths longer than 2 hops.

Figures 3.6(a), 3.6(b), 3.6(c) and 3.6(d) show the evolution of some performance criteria according to the number of submitted flows. The proportion of the violated cliques is the ratio between the number of cliques having exceeded a utilization of 100% and the total number of cliques. The plotted bandwidth gives an estimation of the real bandwidth that the network offers to a given flow, it indicates the bandwidth degradation that occurs according to the various approaches.

ROCX accepts all the flows (Figure 3.6(a)) because there is no link bandwidth limitation. However, after 15 submitted flows, some cliques are overloaded (Figure 3.6(d)) leading to the degradation of the offered bandwidth (Figure 3.6(c)). In this case, the QoS is not guaranteed. Indeed, the first flow for example, after being accepted, sees its reserved bandwidth decreasing from 10 to 9.582 after the 16^{th} flow insertion and to 4.476 after the 28^{th} flow insertion.

I2ILP has the lowest flow acceptation rate though it respects the clique constraints (Figure 3.6(d)) and thus the QoS of the flows (Figure 3.6(c)) since the real bandwidth is equal to 10 which is the requested bandwidth.

As for IROCX, it appears to be the best approach since it has a higher flow acceptation rate than I2ILP (Figure 3.6(a)), respects the clique constraints (Figure 3.6(d)) and guarantees the QoS of the accepted flows (Figure 3.6(c)). In the phase between the submission of the 3^{rd} and the 12^{th} flow, both I2ILP and IROCX accept all the flows nevertheless IROCX underloads its cliques compared to I2ILP (Figure 3.6(b)). The reason is the bandwidth gain made

by IROCX when broadcasting combined packets. This confirms one more time the benefit of Network Coding in term of bandwidth.

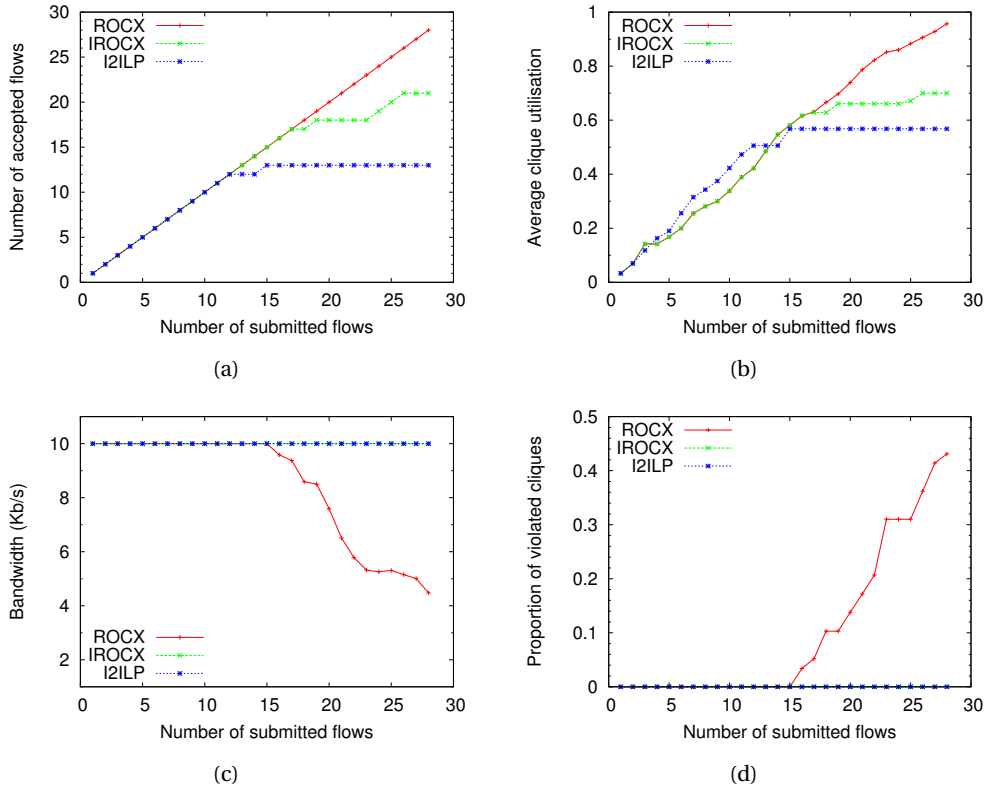


Figure 3.6: Evolution of (a) the flow acceptation, (b) average clique utilization, (c) bandwidth and (d) proportion of the violated cliques according to the number of submitted flows.

3.6 Conclusion

We discussed in this chapter the QoS in terms of bandwidth in wireless multi-hop networks. We first focused on an existing coding-aware routing algorithm (ROCX) that uses Linear Programming for maximizing the Network Coding opportunities. However, it does not take into account the notion of interference and hence can not guarantee the QoS.

To consider the interference impact, we first proposed I2ILP, an interference-aware routing algorithm that uses a clique-based model to take into account all the mutually interfering links. However, it is applied for simple routing and does not apply Network Coding.

Consequently, we proposed IROCX, an interference- and coding-aware routing algorithm. It uses Linear Programming to maximize the benefits of Network Coding while considering the interference impact. Simulation results showed that IROCX respects the QoS and improves the throughput due to the Network Coding effect.

Till now, we have considered that the nodes have only one level of transmission power and considered fixed interference and reception ranges. In the next chapter, we will modify this constraint in order to allow nodes to have multi-level transmission power. We will combine this technique, called Topology Control, with IROCX to still improve the network performance.

Chapter 4

Topology Control on IROCX

After proposing IROCX, a routing algorithm that uses Network Coding and takes into account the interference impact, we extend it in this chapter by allowing the nodes to transmit with several power levels in order to still improve the network performance. Said differently, we combine IROCX with the Topology Control approach. The resulting algorithm, TC-IROCX, takes the advantage of Network Coding, Interference consideration and Topology Control to maximize the number of accepted flows while respecting their QoS requirements.

4.1 Introduction

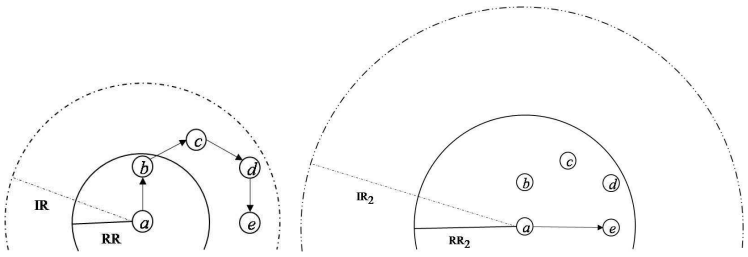
By combining packets, Network Coding reduces the number of transmissions and thus the bandwidth consumption. The interference consideration imposes to respect the QoS in terms of bandwidth for a given flow. That is why IROCX combines these two approaches. It uses Linear Programming to find which flows can be mixed to maximize the network performance. Its main advantage is that routing decisions are made with the awareness of coding and interference impact. However, the nodes are assumed to have the same fixed transmission power.

Instead of using a unique transmission power, nodes, due to Power Control, transmit with a variable-range transmission power to define a new network topology. The goal of Topology Control is to reduce the interference impact and save energy. Indeed, it is shown in [38] that a variable-range transmission power reduces the interference and bandwidth consumption. By merging three fields, we propose a routing algorithm that uses Network Coding and Power Control while considering the interference impact. More precisely, it extends IROCX by using Topology Control. In other words, it answers to the following question **"How does IROCX behaves if the nodes have several transmission power levels?"**. This approach aims at maximizing the number of accepted flows while respecting the QoS in terms of required bandwidth. The following section presents an overview of Topology Control.

4.2 Topology Control

Traditionally, Power Control is used in cellular networks to keep the reception quality independent of the distance between the wireless device and the Base Station. When applied on multi-hop networks, the topology can vary according to the chosen power level, we then use the term "Topology Control".

The main goal of Power Control in multi-hop wireless networks is to lower node energy consumption. One can say that setting the transmission power at its lowest value while keeping the network connected reduces the interference, maximizes the spatial reuse and increases the achievable throughput. However, this implication was disproved and it was shown that a variable-range transmission power increases the amount of traffic in a network [38].



(a) Unit transmission power (b) Doubled transmission power

Figure 4.1: An example of Topology Control.

Indeed, in Figure 4.1, there is a flow from a to e. Using a unit transmission power, the flow $a \rightarrow b \rightarrow c \rightarrow d \rightarrow e$ uses four transmissions (see Figure 4.1(a)). As for a doubled power level, the topology changes and a becomes a neighbor of e but its interference range, denoted IR_2 becomes wider. Then, the flow can use the path $a \rightarrow e$ (see Figure 4.1(b)). According to the traffic in the new interference area of a, the second solution could be better or worse (generates more interference) than the first one.

The transmission power has a direct impact on topology, routing, interfering links and thus energy consumption and network capacity. That is why it should be well adjusted according to the traffic status. Works on Topology Control can be organized on two categories:

The first one is based on the assumption that minimizing the power increases the capacity. Using the network graph, [39] builds a new graph that has the same nodes and reassigns links that keep the graph connected. [40] and [41] assume, in addition, a multi-level transmission power and find respectively a $2-connected$ and $k-connected$ graph. [42] extends this work using nodes equipped with directional antennas.

The second category is based on minimizing the interference since it was proved in [43] that reducing the transmission power does not imply reducing the interference. Moreover it was shown that a variable-range transmission power increases significantly the network

capacity [38]. The Low Interference Forest Establisher (LIFE [43]) approach computes a set of trees that minimizes the interference while keeping the network connected.

In this chapter, we focus on the last category, we consider that the power assignment depends obviously on the topology but also on the traffic. The next section describes in details the combination of IROCX with Topology Control.

4.3 Our solution: TC-IROCX

We consider an IROCX-like algorithm with several levels of transmission power. The Topology Control on Interference-aware Opportunistically Coded Exchanges (TC-IROCX) algorithm adapts dynamically the transmission power of the nodes according to the traffic in order to maximize the number of accepted flows.

We describe in this section the context of TC-IROCX compared to other approaches. Then we present how to compute the mutually interfering links. Finally we recall the adopted linear program.

4.3.1 Context

The scheme depicted in Figure 4.2 shows the progressive evolution of various approaches until the conception of TC-IROCX.

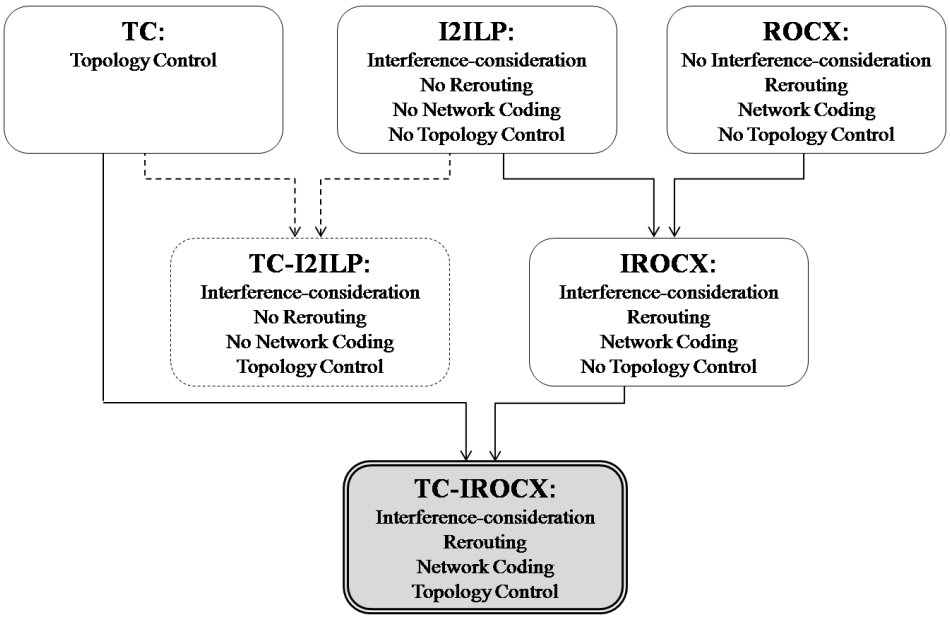


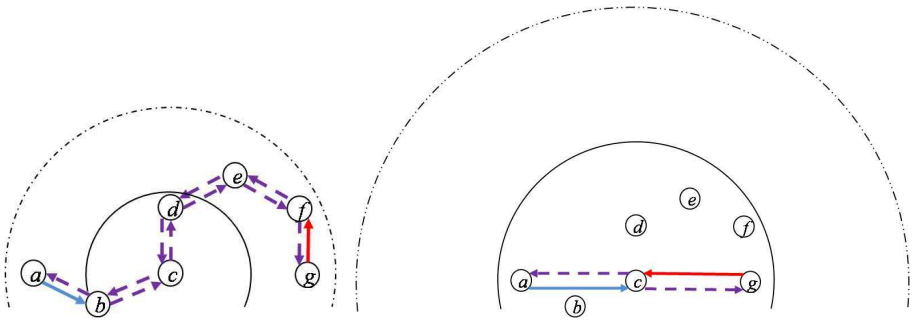
Figure 4.2: Evolution of TC-IROCX.

On one hand, I2ILP uses Linear Programming to route the flows while considering the interference impact. On the other hand, ROCX [10] finds the best path that maximizes the coding opportunities using Linear Programming. Then, IROCX combines them to use Network Coding while considering the interference impact.

TC-IROCX is the combination of IROCX with Topology Control. It is a coding-, interference- and topology control-aware algorithm. Moreover, it allows rerouting. We notice that the combination of I2ILP with Topology Control, namely TC-I2ILP, does not apply the Network Coding approach. The latter algorithm will be used in the performance evaluation for the sake of comparison.

4.3.2 Principle

The idea behind TC-IROCX is to give the nodes the opportunity of transmitting with a higher power level and thus changing the network topology. Then, some nodes become accessible using new routes. Without loss of generality, we use here two transmission powers: a unit and a doubled one.

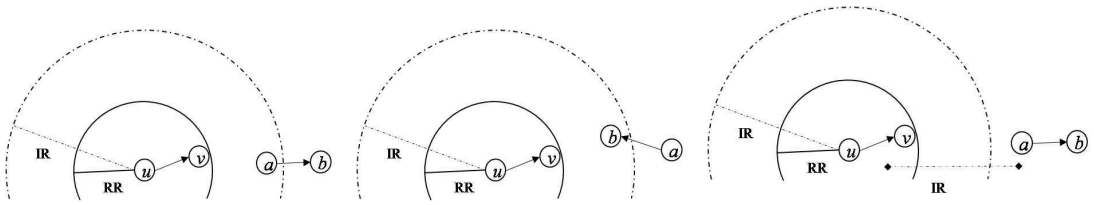


(a) Unit transmission power (b) Doubled transmission power

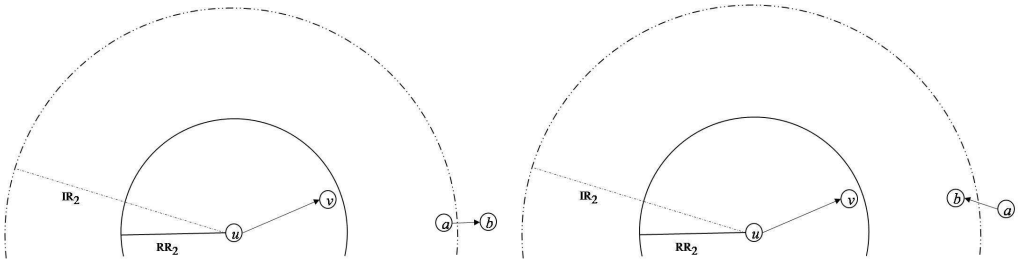
Figure 4.3: An example of Topology Control with TC-IROCX.

In Figure 4.3(a), there is one flow from a to g and another one from g to a. The nodes use Network Coding to reduce the number of transmissions. The dashed arrows indicate the combined packets. Using a unit transmission power, TC-IROCX works as IROCX. When a doubled transmission power is used, only one intermediate node is needed and the number of transmissions is significantly decreased (see Figure 4.3(b)). The counterpart is a wider interference area. That is why the efficiency of Topology Control on an Alice and Bob scheme is not obvious. According to the traffic around the considered nodes, TC-IROCX would choose the solution that maximizes the number of flows. If for instance, using a doubled transmission power prevents the admission of another flow in the network (due to the interference caused by the doubled interference zone), and a unit transmission power allows it, this last is chosen by TC-IROCX.

The doubled transmission power generates a set E_2 of new links. To determine the interference impact of the new links (referred as "long links") on the traditional links (referred as "short links") and on themselves, we will study different cases of interference inspired by the traditional cases.



(a) Case 1: u disturbs a from sending (b) Case 2: u disturbs b from receiving (c) Case 3: a disturbs v from receiving



(d) Case 4: u disturbs a from sending (e) Case 5: u disturbs b from receiving

Figure 4.4: Various cases of interference.

Let a and u be two nodes emitting to b and v respectively. For the traditional cases, nodes a and u are transmitting with a unit power i.e. $a \rightarrow b$ and $u \rightarrow v$ are short links. Here, we apply TC-IROCX for an asymmetric model[1] i.e. the links $a \rightarrow b$ and $u \rightarrow v$ are mutually interfering if $Distance(u,a) \leq IR$ (see Figure 4.4(a)), $Distance(u,b) \leq IR$ (see Figure 4.4(b)) or $Distance(a,v) \leq IR$ (see Figure 4.4(c)).

When u is emitting with a doubled transmission power and a with a unit one, the computation of the interference differs. Indeed, there is an interference between $a \rightarrow b$ and $u \rightarrow v$ if $Distance(u,a) \leq IR_2$ (see Figure 4.4(c))[2] or $Distance(u,b) \leq IR_2$ (see Figure 4.4(d)). The case $Distance(a,v) \leq IR$ is included the two previous cases.

Similarity, when $a \rightarrow b$ is a long link and $u \rightarrow v$ a short one, there is an interference if $Distance(u,a) \leq IR_2$ or $Distance(a,v) \leq IR_2$.

Finally, such as the traditional cases, when both links are long, there is an interference when $Distance(u,a) \leq IR_2$, $Distance(u,b) \leq IR_2$ or $Distance(a,v) \leq IR_2$.

[1] Using a symmetric model, the two links also interfere when the condition $Distance(b,v) \leq IR$ is satisfied. TC-IROCX can be adapted accordingly.

[2] For simplicity's sake we set IR_2 (the Interference Range generated when using a doubled transmission power) to $2 \times IR$.

Based on the new graph $G_2(V, E \cup E_2)$ and on the knowledge of the interfering links, we build a conflict graph CG_2. Then we compute the set of cliques and launch the linear program each time we want to add a flow in the network.

4.3.3 Linear Program

We keep the linear program as it is described in IROCX and adapt it to the set of long links. We use the same notations with $C_{i,j}$ the initial capacity of link (i, j), E_2 the set of new links obtained using a doubled power and Q_2 the set of the cliques generated using the graph CG_2.

The objective function finds the values of $m^t_{k,i,j}$ that:

$$\text{Minimize} \sum_{(i,j)\in E\cup E_2} \frac{x_{i,j}}{r_{i,j}} - \sum_{\substack{(i,j)\in E\cup E_2 \\ (k,i)\in E\cup E_2}} \frac{c_{k,i,j}}{r_{i,k}+r_{i,j}-r_{i,k}r_{i,j}}$$

s.t.

$$\begin{cases}
\sum_{t\in T} u^t_{i,j} - x_{i,j} = 0 \ , \forall (i,j) \in E \cup E_2 \\
\sum_{t\in T} m^t_{i,j,k} - w_{i,j,k} = 0 \ , \forall (i,j),(j,k) \in E \cup E_2 \\
\sum_{i,(s_t,i)\in E\cup E_2} u^t_{s_t,i} - M_t = 0 \ , \forall t \in T \\
\sum_{i,(i,d_t)\in E\cup E_2} u^t_{i,d_t} - M_t = 0 \ , \forall t \in T \\
\sum_{k,(j,k)\in E\cup E_2} m^t_{i,j,k} - u^t_{i,j} = 0 \ , \forall t \in T, \forall (i,j) \in E \cup E_2, j \neq d_t \\
\sum_{k,(k,i)\in E\cup E_2} m^t_{k,i,j} - u^t_{i,j} = 0 \ , \forall t \in T, \forall (i,j) \in E \cup E_2, i \neq s_t \\
\sum_{i,(i,s_t)\in E\cup E_2} u^t_{i,s_i} = 0 \ , \forall t \in T \\
\sum_{i,(d_t,i)\in E\cup E_2} u^t_{d_i,i} = 0 \ , \forall t \in T \\
c_{k,i,j} - w_{k,i,j} \leq 0 \ , \forall (k,i),(i,j) \in E \cup E_2 \\
c_{k,i,j} - w_{j,i,k} \leq 0 \ , \forall (k,i),(i,j) \in E \cup E_2 \\
c_{k,i,j} - c_{j,i,k} = 0 \ , \forall (k,i),(i,j) \in E \cup E_2 \\
\sum_{(i,j)\in q} \frac{x_{i,j}}{r_{i,j}\cdot C_{i,j}} - \sum_{\substack{(i,j),(i,k)\in q \\ k<j}} \left(\frac{c_{k,i,j}}{r_{i,k}+r_{i,j}-r_{i,k}r_{i,j}}\right)\cdot\left(\frac{1}{2C_{i,j}}+\frac{1}{2C_{i,k}}\right) \leq 1 \ , \forall q \in Q_2
\end{cases}$$

The following section shows the experimental results of TC-IROCX and compares it to other approaches.

4.4 Performance Evaluation

This section presents the numerical results for the network model relevant to TC-IROCX. The simulator, written in C++ under Opnet 15.0 and using the GLPK Linear Programming Solver [37], implements the network layer dealing with the flows and the links capacities. It was shown in the previous chapter that IROCX outperforms the existing works, we thus compare here TC-IROCX to IROCX and TC-I2ILP, the enhancement of I2ILP.

The network is a connected graph of 10 nodes randomly placed on an area of 800×800 $units$. The reception range radius (RR) and the interference range radius (IR) are respectively set to 100 and 200 $units$. When transmitting with a doubled power, RR_2 and IR_2 are set to 200 and 400 $units$ respectively. The link initial bandwidth is uniformly chosen between 500 and 1000 Kb/s and its delivery probability between 0.8 and 1. The flows are consecutively submitted to the network and have a lifetime until the end of simulation. The source and destination of a flow are randomly chosen. Each flow requires a QoS consisting of a bandwidth of 10 Kb/s. To make sens of using Network Coding, we focus our study on flows having shortest path lengths greater than 2 hops.

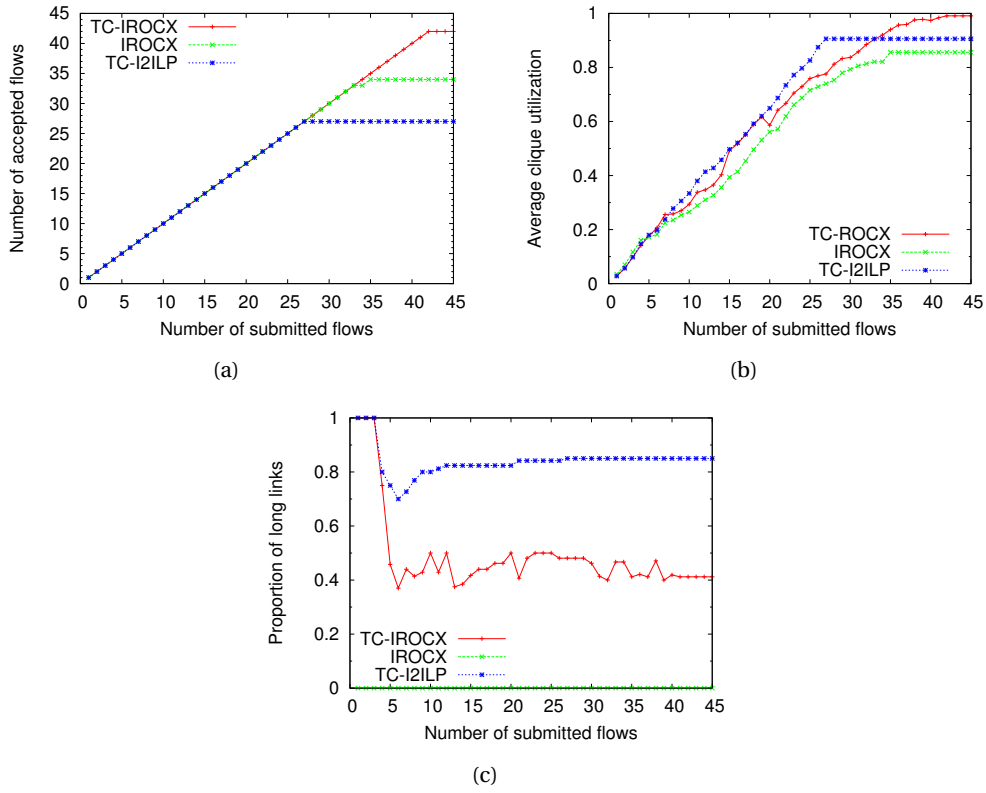


Figure 4.5: Evolution of (a) the flow acceptation, (b) average clique utilization and (c) proportion of long links according to the number of submitted flows.

Figure 4.5 plots the evolution of some performance criteria according to the number of submitted flows: We recall that the utilization rate of a link is the ratio of the bandwidth already reserved on it and its initial bandwidth. The clique utilization rate is the sum of the utilization rates of all the links belonging to that clique (refer to Chapter 3 for more details).

Figure 4.5(a) shows that TC-IROCX has the best performance in terms of admission control. TC-I2ILP differs from TC-IROCX in rerouting and using Network Coding. The low acceptation rate of TC-I2ILP proves the importance of these two mechanisms.

Figure 4.5(b) plots an average of the utilization rates of all the cliques in the network. A unit transmission power leads to lower this rate and thus is less bandwidth consuming than the doubled power. Indeed, the objective function of TC-IROCX is to minimize the overnall network load while respecting the clique constraints and not to minimize the individual clique loads. However, TC-IROCX exploits the cliques capacity until almost 100% when the network load increases.

The major result of this chapter is deduced in Figure 4.5(c) that plots the proportion of the used long links on the total links chosen by each approach. Based on minimizing the numbers of hops, TC-I2ILP mainly uses long links but has the least acceptation rate. IROCX uses only short links but does not have the best results. TC-IROCX uses both unit and doubled transmission powers and has the best acceptation rate. It modifies the transmission power level according to the network load, this shows its network-awareness. These results prove the efficiency of Topology Control on routing algorithms and confirm that choosing one level of transmission power is not the best way to improve the network performance.

4.5 Conclusion

In this chapter we have discussed the application of Network Coding in wireless mesh networks. We have proposed an enhancement of IROCX, an algorithm that considers the coding opportunities and the interference impact during the routing phase.

To still increase the network performance, we have used Topology Control. This approach aims at reducing the interference by adjusting the transmission power. The resulting algorithm, namely TC-IROCX, has the advantages of Network Coding, rerouting, interference consideration and Topology Control.

Since the nodes are allowed to transmit with a higher power level, the network topology changes. We have adapted the way of computing the mutually interfering links. Next, we have slightly modified the original linear program to meet the TC-IROCX features. Simulation results show that TC-IROCX has a better flow acceptation rate compared to the other approaches while respecting the flows' QoS. In the next chapter, we takle another QoS metric which is delay. We use the Random Network Coding scheme to maximize the coding benefits.

Part III

Delay consideration for QoS

Chapter 5

A network-aware Generation-Based Network Coding for multicast flows

We have decribed in Chapter 2 different coding scenarios. Then, we handled in the following chapters an 'Alice & Bob' based scheme and improved it to guarantee the QoS in terms of bandwidth.

We now focus on another QoS metric which is delay. For this aim, we use a more sophisticated coding scheme to maximize the benefits of Network Coding. Indeed, Generation-Based Network Coding (GBNC) deal with generations of packets and allows time and space coding. It increases the throughput and brings more robustness to the system. However, most of the works on GBNC consider a fixed generation size. A large generation size maximizes the Network Coding benefits but leads to a long delay while a small generation size reduces the delay but decreases the throughput. This chapter presents the DYnamic GEneration Size (DYGES) approach. This network-aware method adjusts the generation size according to the network variations (network size, congestion, losses) for multicast flows to keep the delay steady. Our goal is to guarantee a Quality of Service (QoS) in terms of delay. The simulation results show the accuracy of our approach.

5.1 Introduction

We address in this chapter the problem of Generation-Based Network Coding (GBNC) for multicast flows. Using this approach, the source divides the file to send into blocks called generations (also referred as batches[1]). When sending a packet, it combines only the packets belonging to the same generation using coefficients randomly and uniformly selected from a Galois Field while preventing all-zero coding coefficients. Due to the variations of the net-

[1] We refer in the remaining to batch and generation interchangeably.

work characteristics, the delay for decoding a generation can vary dramatically. Our goal is to adjust the generation size to insure a steady decoding delay.

A potential application of our work is to guarantee the QoS for video streaming and conferencing in either wireless or wired networks. In this context, the end-user application is required to display information within a certain delay. Moreover, video conferencing needs the display synchronization among the participants. An example of the targeted networks are limited size networks such as Home Area Networks (HANs).

5.2 Problem statement

In GBNC systems, the generation size is a crucial parameter, it affects both delay and throughput. A large generation size maximizes the Network Coding benefits such as increasing the throughput but leads to a long delay. As for a small generation size, it reduces the decoding delay but decreases the throughput. Section 5.4 gives more details about the relationship among the generation size, throughput and delay.

Most of the works on GBNC consider a fixed generation size. The network state is not smooth, it is subject to variations such as the number of nodes, congestions and losses.

A small generation size is likely the most suitable when the network load is high since it reduces the decoding delay. As for a large generation size, it is more appropriate for a low loaded network since it increases the throughput. However, for a given context, if the chosen generation size is not adequate, the network performance decreases in terms of delay, throughput and/or robustness. Using for instance, a large generation size when the network is high loaded increases significantly the decoding delay. Even without being critical systems, some applications need to have a reasonable bounded decoding delay to guarantee a QoS for the end user.

Having this in mind, we propose DYGES which adapts dynamically the generation size according to the network load for multicast flows. Its goal is to keep the delay steady around a given threshold while maximizing the throughput corresponding to that delay. The next section gives a state of the art related to our approach.

5.3 Related Work

The idea of GBNC was first introduced in [9]. It uses generations and random Network Coding to reduce the coding/decoding complexity. This decentralized scheme does not need to know the whole network topology. However in [9], the packets do not have the same priority and their loss is acceptable consequently there is no need for acknowledgements (ACKs).

An opportunistic coding approach inspired by COPE was proposed in [19] to minimize the delay. The packets to encode are chosen according to their delay threshold for minimizing the number of packets which miss their deadline.

A Network Coding approach with Multi-Generation Mixing for video communication is presented in [44]. It consists of combining a packet with packets of the same generation and packets of previous generations. It increases the decoding rate for networks with sparse connectivity and high loss rates but does not consider the delay issue.

[45] investigates the throughput performance when applying GBNC to scalable multicast. It shows by simulation that the throughput is significantly dependent on the choice of the coding parameters such as generation size, redundancy (or stretch) factor, field size and topology. However, the QoS in terms of delay is not managed and the generation size is fixed.

MORE [21] applies GBNC on an opportunistic routing algorithm to overcome the ExOR [20] limitations (refer to Section 2.3.1 for more details). It considers a fixed generation size and does not deal with delay constraints.

For the best of our knowledge, there is only the approach of [46] that adjusts the generation size in a Network Coding context. For a unicast flow, the destination estimates the Round Trip Time (RTT) using an empirical approach and sends it in the ACK packet. The source uses this information to compute the generation size and respect the probability that the block-decoding delay is below a given threshold. It is not straightforward to use this approach for multicast flows since the bottleneck is not relevant to one destination. Moreover, the relationship between the given threshold and the throughput is not obvious.

5.4 Generation size, throughput and delay

We have explained that the generation size for the GBNC is a crucial parameter. In this section, we detail its impact on network performance such as delay and throughput. This parameter affects the decoding delay that is the duration between sending the first packet (or combination) of a generation at the source side and decoding the last packet of that generation at the destination side. It also affects the overhead (number of ACKs sent to acknowledge the generations). Adjusting the generation size is useful, this is proved by the relationship among throughput, delay and generation size given below.

Let us take the following example: the source node a wants to send a file of F packets to the destination node b. We consider that the data and ACK packet sizes are respectively L and L'. We assume that the propagation time is negligible and the capacity of link $a \rightarrow b$ is equal to $C(bits/s)$ and greater than or equal to the arrival rate at the source. The generation size is k. The number of generations is equal to F/k, this value indicates the number of ACKs and thus the overhead. Let D_{total} be the delay for reliably sending the whole file, which

means sending all the data packets and receiving all the ACK packets by the source. It is given by:

$$D_{total} = \frac{Amount\ of\ sent\ packets + Amount\ of\ ACKs}{Capacity} = \frac{F \times L + \frac{F}{k} \times L'}{C} \tag{5.1}$$

$$\text{then} \quad C = \frac{F \times L}{D_{total}} + \frac{F \times L'}{k \times D_{total}} = goodput + overhead \tag{5.2}$$

Assuming a fixed capacity C, we can infer from these formulas the impact of the generation size k on:

- Throughput: the higher is k (the lower is the overhead[2]), the higher is the goodput[3].
- Overall Delay: the higher is k, the lower is the overall delay.

A large generation size appears to be more appropriate to increase the network performance. However, the destination can exploit the native packets of a generation only after receiving and decoding the k coded packets sent by the source. The higher is k, the longer is the decoding delay, the more the information exploited by the destination's higher layers is lately received. That is why this delay is crucial for real time applications.

Due to the variations of the network characteristics (congestion, end-to-end packet delay, network size), a fixed generation size does not guarantee a fixed decoding delay and hence can affect the QoS in terms of delay. Our goal is to build a network-aware approach that adjusts the generation size dynamically to maximize the throughput while respecting a fixed block-decoding delay. The next section explains our approach in details.

5.5 Our solution: DYGES

Our approach performs a feedback control mechanism to adjust the next generation size of a given multicast flow based on some parameters deduced from the last sent generation. We describe below the DYGES method. We present in Figures 5.1 and 5.2 detailed diagrams of our MORE-like system for the source and the forwarders/destinations sides.

Before sending the first packet of a multicast flow, the generation size is set to its default value and a new batch is created. A *batch* or *batch matrix* is a table associated to a given multicast flow and that is used to save a copy of the current generation packets.

The source (see Figure 5.1) receives a packet from higher layers and adds it to a temporary queue. This queue is useful when the application layer has a high throughput, the

[2] Overhead: throughput of ACK packets.
[3] Goodput: throughput of data packets.

packets can be saved in it until the batch is flushed. When a new batch is created, packets of the queue are transferred to the batch. However, the number of these packets should not exceed the generation size. Moreover, each time a packet is inserted into the batch, a variable NPB (Number of Packets in the Batch) is increased.

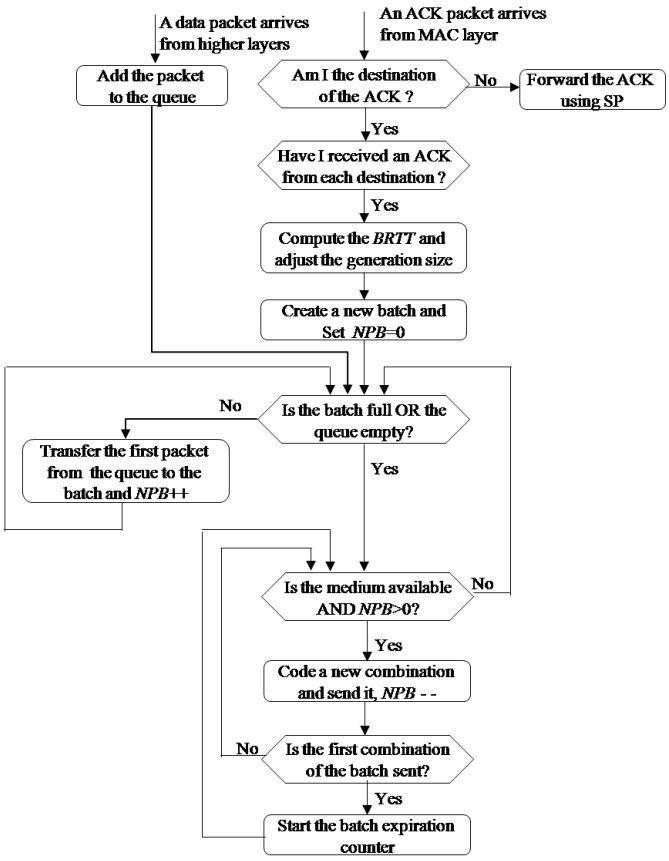


Figure 5.1: Sending a packet and receiving an ACK by the source.

If the medium is available, as long as there are combined packets to send from the current batch ($NPB > 0$), the source builds a combination using all the packets belonging to the current batch matrix and sends it. We recall that the coefficients are randomly taken from a Galois field (finite field). The value of NPB is then decreased. If the medium is not available, the batch not full[4] and another packet arrives from higher layers, this packet is inserted into the batch and NPB increased. Consequently, this packet will be taken into account to build the combinations to be sent. Note that for the first sent combination of each generation, the source sets a timer for the batch expiration. In case of loss, the timer expiration triggers the retransmission.

Upon receiving a packet P, each intermediate or destination node (see Figure 5.2) belonging to the multicast tree compares its current batch identifier to the one of P denoted

[4] Necessary condition: if the batch is full, all the arriving packets are kept in the temporary queue.

$P.batch$. If it is smaller, this means that the source has already received the ACK for the previous generation and started the new generation with packet P. In this case, the node flushes its batch and resets it with P. Otherwise, if the node has a current batch equal to the one of P it adds it to its batch verifying that the combination P is innovative i.e. linearly independent of the previous combinations it has already received and belonging to that batch. If so and the node is a forwarder (not a leaf in the tree), it is allowed to send a combination. Therefore, when the medium is available, it builds a random combination using the combinations of its batch and sends it. Moreover, if the node is a destination, it checks if it has received the whole batch and if so, sends an ACK asking for a new batch. We want to consider all the packets of a given batch for building a combination for maximizing the robustness against losses while respecting a steady decoding delay. Consequently, the decoding is launched after receiving the whole generation. Note that the ACK is routed using the Shortest Path (SP) and is prior than data packets when forwarding.

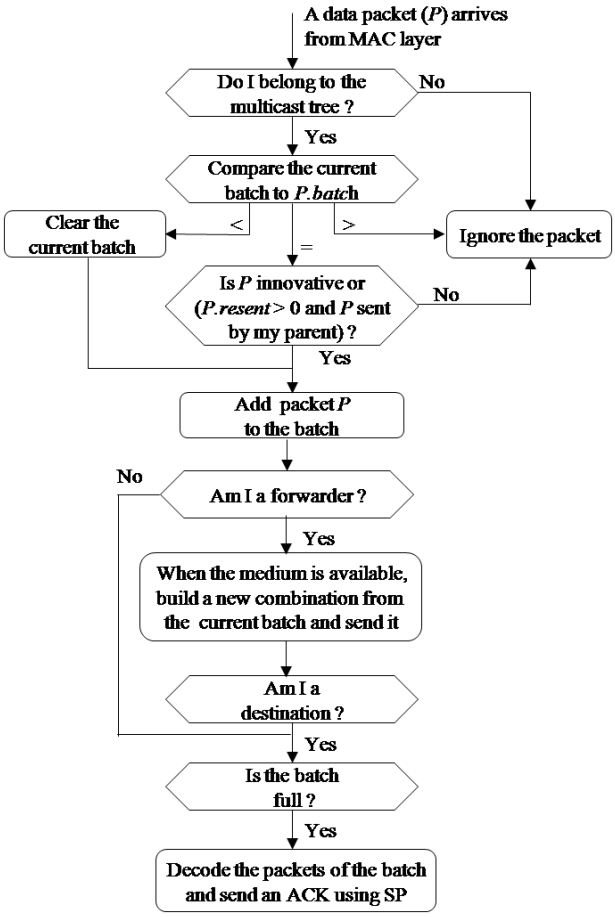


Figure 5.2: Receiving a packet by the forwarders and the destinations.

In addition to manage multicast traffic and multiple ACKs, our system differs from MORE in its post batch expiration behavior. Indeed, in the case of retransmission (indicated in the packet with $P.resent > 0$), an intermediate node is allowed to forward a non-innovative com-

bination sent (only[5]) by its parent in the tree. For the targeted applications, we consider that even if a batch expires, it is necessary to resend the batch to make the receiver decoding the generation and retrieving some packets that did not exceed their individual deadline.

When the source (see Figure 5.1) receives the ACK, it checks whether it has received one ACK from each destination, if so, it resets the batch. To build the new batch, the packets of the temporary queue, no more than the generation size, are transferred to the batch according to their arrivals. Then, the previous sending process is applied.

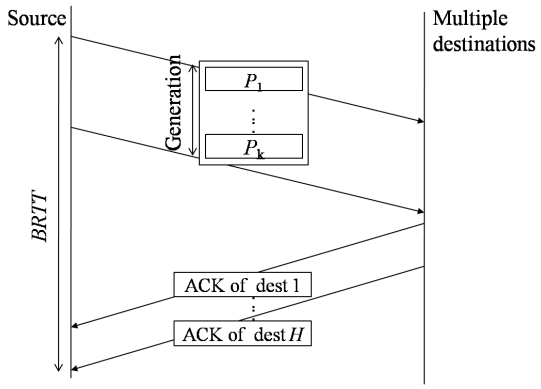


Figure 5.3: $BRTT$ computation after sending a k-packet generation to H destinations.

Our system uses the DYGES approach to estimate the next generation size. It computes the $BRTT$ (Batch Round Trip Time), as shown in Figure 5.3, it is the span between sending the first packet of a generation and receiving the last ACK (from the last destination) for this generation.

After computing $BRTT$, DYGES compares it to a threshold $thresh$: if $BRTT$ is below $thresh$, the generation size is increased, otherwise, the generation size is decreased. $thresh$ depends on the fixed maximum decoding delay required by the application. This feature brings more content awareness to our system.

$thresh$ should be computed by simulation according to each scenario, To give an example of computation, we consider that destination's application layer requires a certain rate of packet arrival. One has to set $thresh$ to a low value, then increases it gradually until the rate of late packets at the destination's application layer is not negligible (QoS not guaranteed). For our simulations, we will consider that $thresh$ is already known.

DYGES is inspired by TCP but instead of adjusting the congestion window to maximize the throughput, DYGES adapts the generation size dynamically to losses and congestion for managing the decoding delay. It is network-aware i.e. it intends to have a smooth decoding delay independently of the network variations. The next section shows the experimental results.

[5] This solution avoids the flooding phenomenon.

5.6 Simulation results

We implement in our event-based network simulator, written in C++ under Opnet 15.0, a routing layer endowed with Network Coding and an ideal MAC layer. We use the multicast tree algorithm given in [47] to find the nodes implied in the flow forwarding.

We deploy a connected network of $N = 30$ nodes on a square of 1000x1000m^2. At the beginning of the simulation, a multicast flow is generated and has a lifetime until the end of the simulation. Its packet inter-arrival follows a Poisson distribution with a parameter $\lambda 1$. This flow will be used for the study, it has 12 destinations and its tree contains 15 nodes.

Many nodes are chosen to generate unicast flows with randomly chosen destinations. They have a packet inter-arrival according to a Poisson distribution with a parameter $\lambda 2$. Their lifetime follows a Poisson distribution with a parameter $\lambda 3 = 5s^{-1}$ and the flow interarrival follows a Poisson distribution with $\lambda 4 = 20s^{-1}$. The goal of these flows is varying the network state to investigate the behavior of the studied multicast flow.

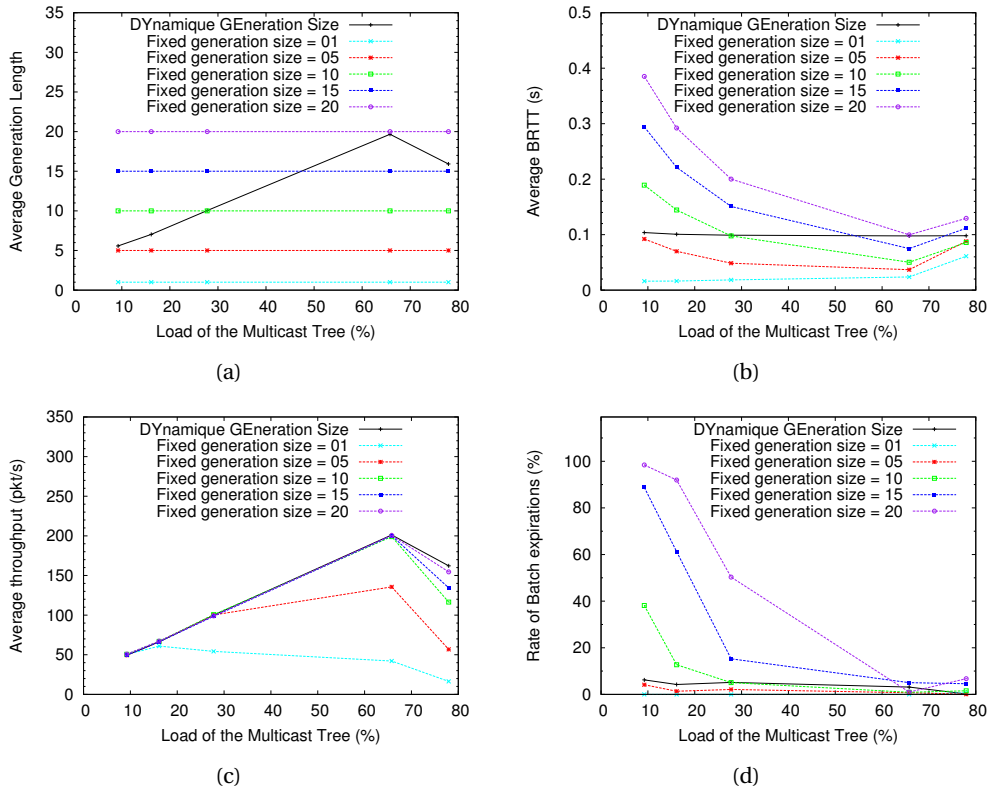


Figure 5.4: Evolution of (a) the average generation size, (b) $BRTT$, (c) throughput and (d) rate of batch expirations according to the load of the multicast tree.

Each link is lossless and transmits a packet in $1ms$. The sending buffer of an intermediate node (at the MAC layer) has an unlimited capacity. To build a combination the coeffi-

cients are chosen in a Galois Field GF[251]. The batch expiration time is set to $200ms$ and $thresh = 95ms$. Here, the increment used for adjusting the generation length is ± 1 packet.

We focus on the load of the studied multicast tree, we vary the network load by varying the packet inter-arrival parameters ($\lambda 1$ and $\lambda 2$) and the number of unicast flows. We consider the reference (a load of 100%) when all the nodes of the tree are sending packets. The simulation duration is $100s$.

Figure 5.4 shows the simulation results for the studied flow. We compare DYGES with different fixed generation size systems to investigate the impact on decoding delay, throughput and batch expiration rate.

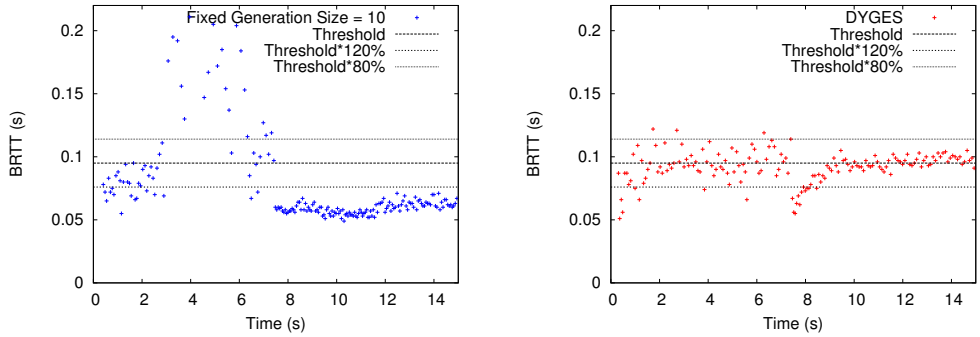
- For the low load ($\leq$ 28%), the packet inter-arrival is much lower than the network capacity and this makes the receivers waiting idly for the complete generation. This explains the high $BRTT$ obtained using a high generation size on Figure 5.4(b) and the number of their $BRTT$ expirations (see Figure 5.4(d)). The throughput of the different approaches, plotted on Figure 5.4(c) is identical except for the generation size that is equal to one, it has a very high overhead because each packet is ACKed individually. Even if it appears that the generation size of 5 has a lower decoding delay than DYGES while having the same throughput, the goal of DYGES is not to find the optimal values of delay and throughput, but to keep the decoding delay as steady as possible without dramatically losing performance in terms of throughput.

- When the network load increases ($28\% < load \leq 65\%$), the idle time decreases leading to still decrease the $BRTT$ of the fixed generations. The small generation size has a low decoding delay but at the cost of throughput because it does not maximize the benefits of Network Coding. As for DYGES, it keeps the decoding delay steady and the throughput at its maximum value. To do so, it adapts the average generation size until 19.7 (see Figure 5.4(a).).

- For the high load (> 65%), there is more congestion engendered by the other flows and thus a longer end-to-end packet delay. Consequently, the $BRTT$ delay increases and the throughput decreases. A large generation size leads to more $BRTT$ expirations (compared to the load of 65%). As for DYGES, it decreases its average generation size in order to, on the one hand, have a high throughput and on the other hand, still keep the decoding delay around the threshold.

Figure 5.5 shows a comparison between the Fixed Generation Size and DYGES approaches in terms of instantaneous $BRTT$. We see on Figure 5.5(a) that $BRTT$ is highly affected by the presence of other flows and congestion. Then, when the network state stabilizes, $BRTT$ decreases and remains around a value below $thresh$. As for DYGES, it keeps $BRTT$ around $thresh$ in any case.

If we compute the ratio of the $BRTT$ points that are out of the range $thresh \pm 20\%$ for example, we get 53.43% for a Fixed generation size and 16.56% for DYGES. We figure out DYGES brought around 70% of the exceeding points into the considered interval. Even if we can not exactly know the future end-to-end, DYGES tries to predict the network variations according to the last $BRTT$. As a consequence, DYGES decreases the high $BRTT$ points to respect the delay, and increases the low $BRTT$ points to increase the throughput. This is done instantly and shows how fast DYGES converges (see Figure 5.5).



(a) (b)

Figure 5.5: Evolution of the instantaneous $BRTT$.

We can conclude that DYGES adapts dynamically the generation size to keep the decoding delay steady and thus to guarantee the QoS in terms of delay under the different loads, this shows its network-awareness.

5.7 Conclusion

In this chapter we have studied the issue of QoS in terms of delay for GBNC systems. Most of these systems consider a fixed generation size. However, the delay can vary significantly according the network state and then degrade the application performance.

Therefore, we proposed DYGES, an approach that adjusts the generation size dynamically. We have investigated by simulation the generation size according to the network load variations. The results showed that DYGES is *content-aware* since it adapts the generation size according to $thresh$ that depends on the fixed maximum decoding delay required by the application. It is also *network-aware*, it adapts the generation size according to the congestion in order to maintain the decoding delay steady.

For our simulations, we have considered DYGES in a lossless context, we hence focus in the next chapter on data and ACK packet loss and explain how DYGES can be enhanced to cope with this issue.

Chapter 6

Reliable DYGES for wireless networks

We have proposed in the previous chapter the DYGES approach which adjusts the generation size dynamically according to network load. In this chapter, we show the packet loss impact on DYGES and present RDYGES, an enhancement of DYGES in a lossy environment. In addition to combine the packets, a redundancy is used to offset the data packets loss. As for the ACK loss, it can not be offset by combining the ACK packets, therefore, we take the advantage of the opportunistic listening feature of the nodes to retransmit the lost ACKs when it is appropriate. The simulation results show the RDYGES efficiency in a lossy network.

6.1 Introduction

Based on DYGES, we propose in this chapter an enhancement taking into account the packet loss in a lossy environment. Indeed, the version of DYGES we have considered for simulations relies on the MAC layer to retransmit the packets. Therefore, it appears to the routing layer that the network is lossless. However, this implies more sophisticated mechanisms at the MAC layer to make a reliable transmission. Since the chosen Network Coding scheme imposes the opportunistic listening mode to the nodes, it is interesting to take the benefit of it to offset the ACK loss. Our contribution is mainly based on this idea.

6.2 Related Work

When the delivery probability in the network is low, a retransmission mechanism can be necessary to guarantee the transmission reliability.

A comparison between Network Coding and other error control techniques is given in [48]. It considers a tree-based reliable multicast and analyzes the achieved gain in terms of num-

ber of transmissions. Three approaches based on Automatic Repeat reQuest (ARQ) and Forward Error Correction (FEC) were implemented:

- End-to-end ARQ: the source node retransmits the packets until their reception by all the destinations. The intermediate nodes only forward those packets.
- End-to-end FEC: this rateless coding allows each node to code blocks of native packets and transmit them. It is necessary to decode a block before re-encoding and forwarding it.
- Link-by-link ARQ: each intermediate node retransmits the packets until all its children in the tree receive it.

Network Coding can be considered as a Link-by-link FEC approach. It was proved that it can significantly reduce the number of transmissions compared to the other three schemes. Other works such as [49] and [15] prove the efficiency of Network Coding for respectively one and two hops scenarios. However, they consider reliable feedback and a fixed generation size.

In the literature, many works dealing with Network Coding such as [45] and [9] present their schemes but do not address the retransmission issue.

An efficient Network Coding based retransmission algorithm for wireless multicast is presented in [50]. Its main idea is to generate combinations from a given generation according to the previously received (native) packets and send them without waiting the reception of the whole generation. The packets to code are sorted according to their "utility" that is computed in function of their delivery/loss. Moreover, the proposed algorithm, named BENEFIT, considers that it is not necessary for a combination to be decoded immediately after its sending. Its goal is to reduce the number of retransmissions and the average individual decoding delay.

A COPE-like approach is presented in [51], it aims at minimizing the number of transmissions. The retransmissions are managed using timers and the packets to code are chosen according to several heuristics such as sorting by time (First In First Out) and sorting by utility (the number of receivers that need the packet).

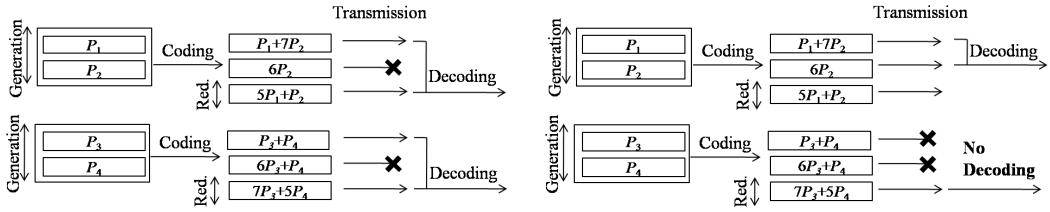
The authors of [52] propose an algorithm that schedules the transmission time slots while considering (dedicating) a part for retransmissions. Retransmission happens only when the percentage of the lost packets achieves a given threshold, this criterion minimizes the delay.

The schemes of [50], [51] and [52] are based on a one-hop transmission and apply Network Coding only for retransmissions (new packets are sent native). Moreover, the generation size is considered fixed. We present in the following section the relationship between the generation size and packet loss.

6.3 Generation size and packet loss

One of the Network Coding benefits is the robustness against losses. The robustness is quantified by the probability of decoding the generation (the probability of receiving all the combinations of a generation) and is affected by the packet loss probability. The packet loss can be caused by a congestion at the node level or the channel transmission error at the link level.

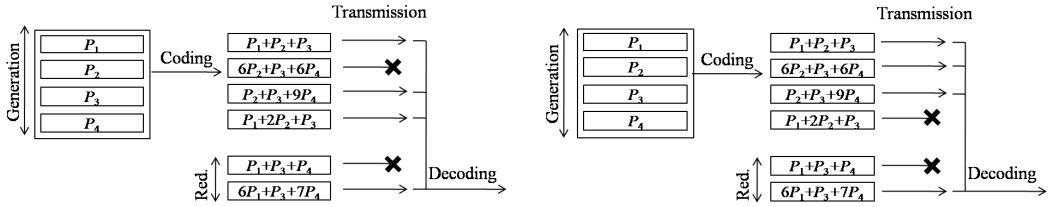
For a lossy network, the higher is the generation size the more robust is the system against losses and vice versa. Let us take the two systems depicted in Figures 6.1 and 6.2, where a source node wants to send four packets to a destination node. Since the loss probability is equal to 1/3, a redundancy (Red.) of two combinations is added to thwart the loss of two packets. For example, in the first scenario (Figures 6.1(a) and 6.2(a)), the second and fifth combinations are lost. In the second scenario (Figures 6.1(b) and 6.2(b)), the fourth and fifth combinations are lost.



(a) Loss of the 2^{nd} and 5^{th} packets

(b) Loss of the 4^{th} and 5^{th} packets

Figure 6.1: Impact of a 2-packet generation on the system robustness.



(a) Loss of the 2^{nd} and 5^{th} packets

(b) Loss of the 4^{th} and 5^{th} packets

Figure 6.2: Impact of a 4-packet generation on the system robustness.

The second system recovers the initial information in both loss scenarios (see Figures 6.2(a) and 6.2(b)). As for the first system, it does not recover the whole information in the second scenario (see Figure 6.1(b)) without transmitting more than three combinations of the second generation. Here, the first system is less robust than the second one because its generation size is smaller.

When generalizing this idea to a system generating n combined packets from a generation of k native packets, the Success Probability ($SProb_{(k,n)}$) of decoding the generation on a

path having a delivery probability q is proportional to k. It is given by:

$$SProb_{(k,n)} = \sum_{i=k}^{n} C_n^i . q^i . (1-q)^{n-i}$$

Indeed, the success probability of receiving exactly k and loosing $n-k$ coded packets follows a binomial distribution $B(k,n)$ and is equal to: $q^k.(1-q)^{n-k}$. Then, $SProb_{(k,n)}$ is the probability of receiving *at least* k packets.

After presenting the theoretical aspect of the generation size and the parameters affecting it, we explain in the following section how to apply Redundancy to cope with data loss.

6.4 Redundancy Factor for DYGES in a lossy context

As we have seen it, Network Coding allows more robustness to losses not only by combining packets but also by adding Redundancy. Therefore, each forwarder z has a Redundancy Factor Red_z given as follows:

$$Red_z = \max_{y \in Children[z]} \frac{1}{d_{z,y}} \tag{6.1}$$

where $Children[z]$ are the neighbors of z that belong to the multicast tree except the parent of z, and $d_{z,y}$ is the delivery probability of link $z \rightarrow y$.

A forwarder z has the knowledge of its Redundancy Factor and uses a variable NUM_z initially null. When receiving an innovative combination relevant to the current batch, z computes the number of packets to send as described in [6]:

$NUM_z = NUM_z + Red_z$,

Send $\lfloor NUM_z \rfloor$ random combinations of the current batch,

NUM_z = fractional part of NUM_z.

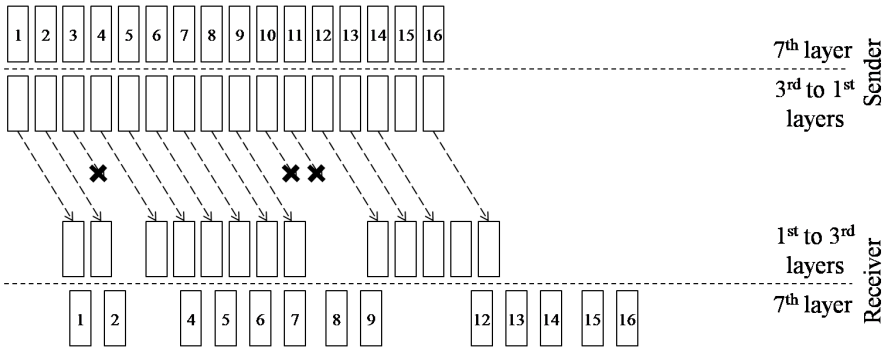
Taking into account the delivery probability of the network links increases the network-awareness of DYGES. The next section summarizes the effect of DYGES.

6.5 DYGES effect in a lossy context

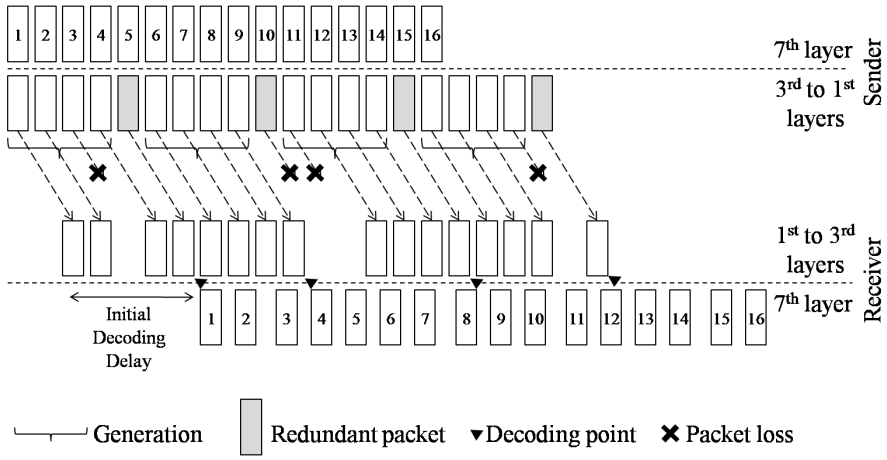
We show in this section the behavior of DYGES in a lossy network. We first focus on data packets loss only, then on both data and ACK packets loss to show the dramatic impact of ACK loss on DYGES.

6.5.1 Robustness against network variations

We first recall the impact of Network Coding by comparing it to a system with no Network Coding. Then we show the advantage of using DYGES by comparing it to a fixed generation size scenario. In the example depicted in Figures 6.3 and 6.4, the source wants to send 16 packets. Assuming a real-time application such as video streaming, the information should be displayed at the receiver side with a certain rate. This means that a packet has a given individual deadline for being received.



(a) Without Network Coding



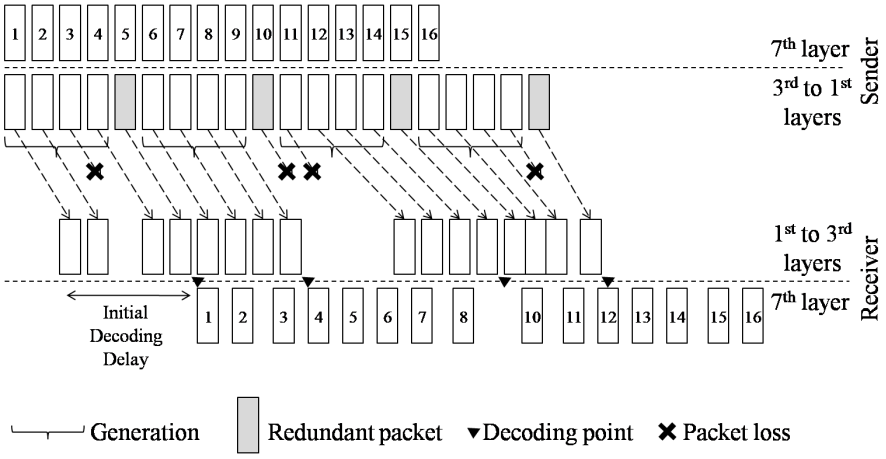
(b) With Network Coding

Figure 6.3: The effect of Network Coding in a lossy context.

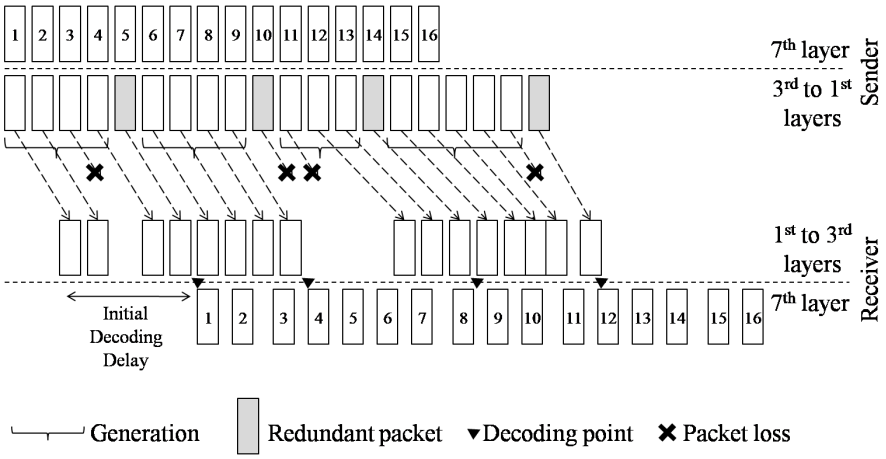
Figures 6.3 shows the effect of Network Coding in a lossy network. The system depicted in Figure 6.3(a) does not use Network Coding and thus the packet loss generates gaps at the receiver side reducing consequently the QoS of the displayed stream. As for the system depicted in Figure 6.3(b), thanks to GBNC, the redundancy offsets the lost packets. The packets are received in time and thus a higher definition video can be guaranteed. We tolerate an initial decoding delay, it can be adjusted via the first generations size.

A fixed generation size could not manage the variation of the end-to-end delay caused by either the congestion or the loss rate variations. Indeed, in Figure 6.4(a), when the end-to-end

delay increases after sending the 12^{th} combination, the current generation is lately decoded and the 9^{th} packet is not recovered in time. Even if the real-time targeted applications are not critical, we should adapt the generation size to the network context.



(a) Fixed Generation Size



(b) DYGES

Figure 6.4: The effect of DYGES in a lossy context with congestion.

Figure 6.4(b) shows the effect of having a dynamic generation size. When the end-to-end delay is high, DYGES decreases and adapts the generation size to have a smooth decoding delay. Indeed, the 9^{th} packet is decoded just in time. Then DYGES increases the generation size due to the end-to-end delay decrease.

Using a threshold to bound the block-decoding delay, compels the source node to adjust its data rate according to the network load for delivering the packets in time to the destination's application layer.

6.5.2 Loss of data packets

We use the simulator and the benchmark parameters defined in Section 5.6 of the previous chapter. We consider for the following simulations that the loss can affect *"only data packets"*. Each link has a delivery probability uniformly chosen in $[g,1]$ where g is the minimal delivery probability. Since the congestion phenomenon was already studied in the previous chapter, we focus here on packet loss and thus generate only the multicast traffic. The batch expiration time is set to $110ms$ and $thresh = 95ms$.

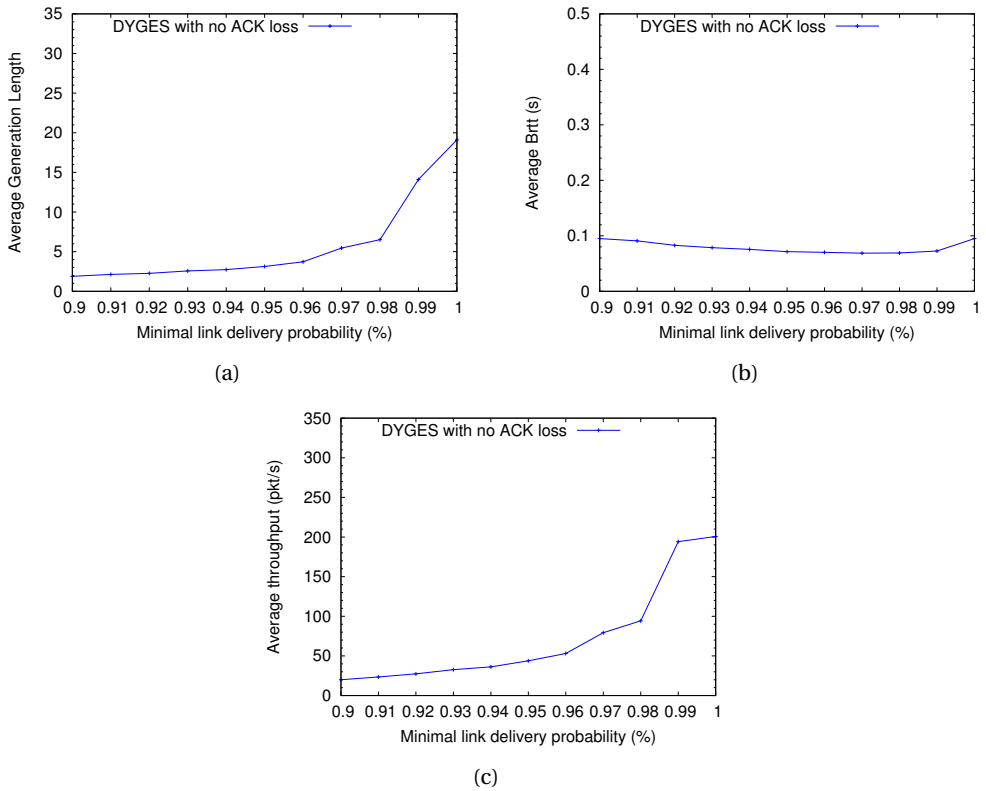


Figure 6.5: Evolution of (a) the average generation size, (b) $BRTT$ and (c) throughput when using DYGES according to the delivery probability.

Figure 6.5 shows the simulation results for the studied flow when varying the minimal delivery probability from 90% to 100%.

- Figure 6.5(a) shows the network-awareness of DYGES, it increases the generation size as the delivery probability increases.
- Figure 6.5(b) shows that even if the delivery probability changes in the network, the block-decoding delay ($BRTT$) remains quite steady.
- Figure 6.5(c) shows that DYGES is sensitive to losses. Since it is based on $BRTT$, it can not make the difference between the congestion and packet loss and hence reacts

similarly. The more the delivery probability increases the more DYGES increases the generation size and thus the throughput while keeping a smooth $BRTT$.

We can conclude that DYGES adapts dynamically the generation size to keep the decoding delay steady and guarantee the QoS in terms of delay under the different scenarios of loss. This shows another aspect of its network-awareness.

6.5.3 Loss of data and ACK packets

We consider here that the loss can affect both *"data and ACK packets"*. Figure 6.6 shows the evolution of the throughput when varying the minimal delivery probability from 90% to 100%. We can see that when there is ACK packet loss the throughput becomes close to zero. Indeed, there are many ACK packets delivered through many links (the longer is the path, the higher is the end-to-end loss probability) and the source can not start sending a new generation until the reception of one ACK from each destination.

We conclude that DYGES, as we have built it, is very sensitive to ACK losses. The data packet loss is thwarted by using combinations and adding redundancy. The ACK packets are too small to be combined and the notion of redundancy factor for ACKs is meaningless. That is why we propose in the following section a reliable mechanism to thwart the ACK loss.

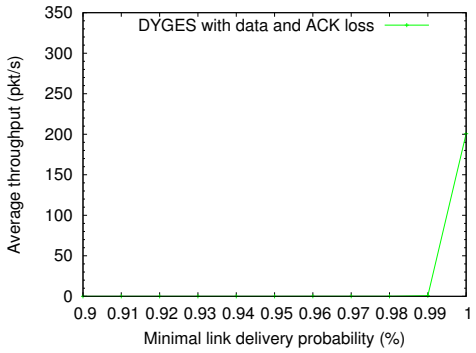


Figure 6.6: Evolution of the average throughput according to the delivery probability.

6.6 ACK recovery for a Reliable DYGES scheme

6.6.1 RDYGES principle

We present in this section an enhancement of DYGES, the Reliable DYnamic GEneration Size (RDYGES), that overcomes the ACK loss problem. Many approaches presented in the literature rely on MAC-Layer retransmissions to recover the lost ACK packets, they use of instance

the $CSMA/CA$ protocol [53]. As for RDYGES, it takes the advantage of using opportunistic listening [3] to get an implicit acknowledgment of the ACK reception. Using RDYGES, each MAC-sender sending an ACK waits until its MAC-receiver forwards it. If the ACK is not forwarded then the MAC-sender retransmits it.

Using the opportunistic listening in the context of RDYGES means each node receives and analyzes all the data and ACK packets sent by its neighbors. When the ACK is received by its final destination that is also the data flow source, it is forwarded by this destination to itself to inform the neighbor that the ACK is not lost, we call it "self ACK". When sending an ACK packet, the MAC-sender, say node a, saves it in a table $Table\_ack$ that records all the sent ACK packets. We represent the structure of a frame as $(S, D, ID, TYPE, IPH)$ where

S is the sender's MAC address,

D the receiver's MAC address,

ID the packet identifier,

$TYPE$ indicates if it is a DATA or an ACK packet and

IPH is the IP header.

We associate to each ACK packet saved in $Table\_ack$ a positive number called NCR (Number of Cycles[1] to wait before the Retransmission), it is initially set to $Initial\_value$. This value is discussed later in this chapter.

Each time a frame $(S_{rec}, D_{rec}, ID_{rec}, TYPE_{rec})$ is heard (or received), node a having a non-empty $Table\_ack$ proceeds as follows (see Figure 6.7):

For each element $(S_{sent}, D_{sent}, ID_{sent}, TYPE_{sent})$ of $Table\_ack$,

1) If $TYPE_{rec} = DATA$ and $D_{sent} = S_{rec}$, the neighbor is emitting data because it has no ACK to forward, we recall that the ACKs are prior than data packets during the sending phase. The ACK sent by node a to its neighbor might be lost. The NCR must be decremented to accelerate the retransmission of the said ACK.

2) If $TYPE_{rec} = ACK, D_{sent} = S_{rec}$ and $IPH_{sent} = IPH_{rec}$, the ACK was well forwarded. The ACK retransmission is then canceled and the corresponding element removed from $Table\_ack$.

3) If $TYPE_{rec} = ACK, D_{sent} = S_{rec}$ and $IPH_{sent} \neq IPH_{rec}$, the receiver is busy in sending other ACKs. The retransmission is delayed by incrementing NCR.

[1] A cycle corresponds to the span between two events of sending or receiving packet.

4) If $TYPE_{rec} = ACK$ and node a is the final destination of the ACK, therefore it should forward a "self ACK" to inform its neighbors it has well received it. To do so, it adds the corresponding element to $Table\_ack$ and sets its NCR to 0 in order to send it as soon as possible.

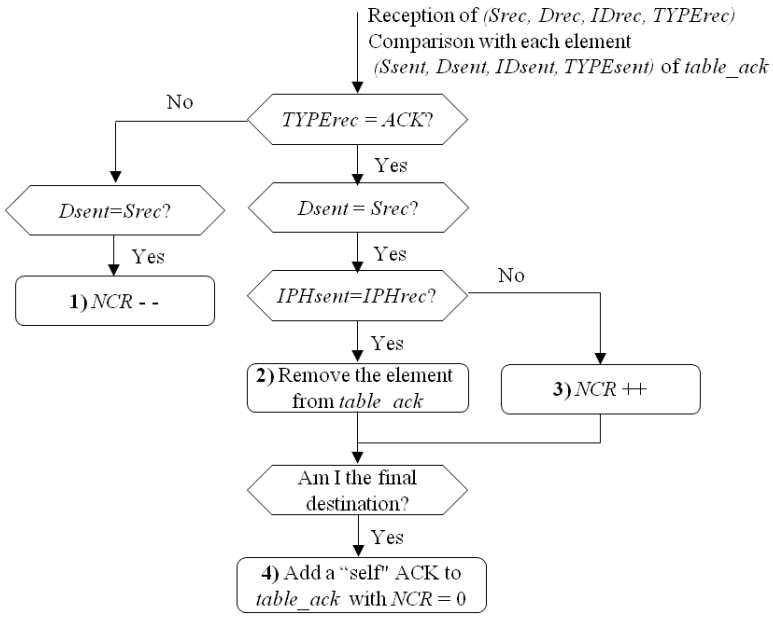


Figure 6.7: Managing the ACK retransmission during the sending phase.

For packet emission, we set the priority of retransmitting a sent ACK higher than transmitting an ACK for the first time, transmitting data packets being the least prior. Each time node a has an opportunity of transmission (the medium is free), it manages the ACK retransmission as follows (see Figure 6.8):

1) If there is an element $(S_{tosend}, D_{tosend}, ID_{tosend}, TYPE_{tosend})$ of $Table\_ack$ with a null NCR, it is ***retransmitted*** and NCR is reinitialized to $Initial\_value + 1$ (will be decremented by 1 at the end of this procedure).

2) Moreover, if $S_{tosend} = MAC\_ID$ of node a, this means that the ACK to be sent is a "self ACK". There is no deed to save it, the element is consequently removed from $Table\_ack$.

3) If there is no ACK to retransmit but there is an ACK (sent by the routing layer) to send, it should be ***sent*** and the corresponding element added to $Table\_ack$ while setting NCR to $Initial\_value + 1$.

4) If there is no ACK to send or resend and there is a data packet to send, it is sent.

5) For each element of $Table\_ack$, NCR is decremented with one cycle.

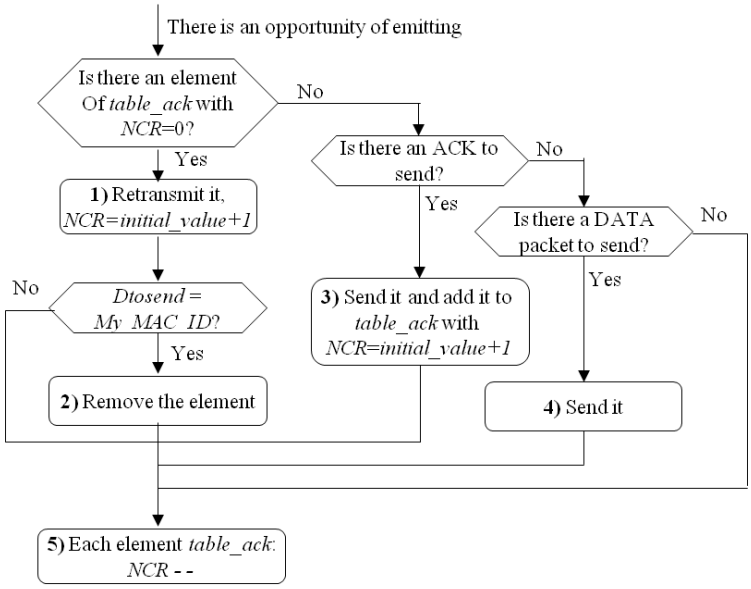


Figure 6.8: Managing the ACK retransmission during the reception phase.

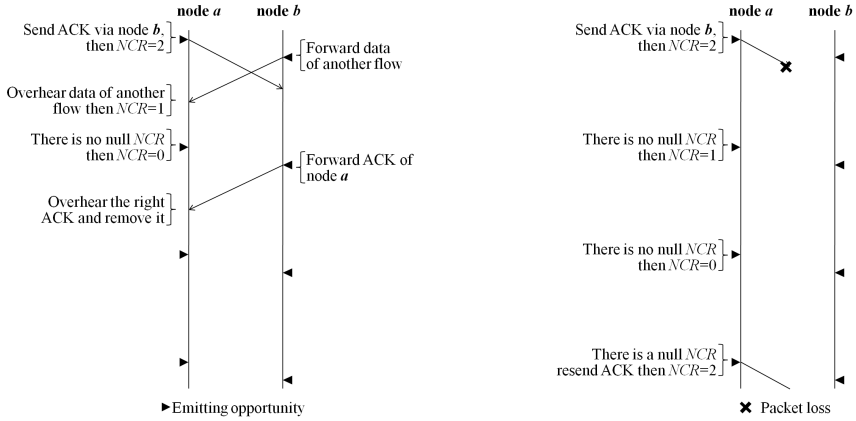
Note: This approach is applicable on ACK packets because they are much shorter and less numerous than data packets. Indeed, the later would consume much memory. The only extra cost of RDYGES is the "self ACKs" sent by the source of the multicast flow, however, it is negligible.

6.6.2 Initial value

If the initial value of NCR is too small, the retransmission would be sent early and might be useless (bandwidth and energy consumption). On the contrary if it is too large, the sender would wait idly before retransmitting and hence the throughput might decrease. We intuitively set $initial\_value$ to 2 and show on two examples that greater and lower values are not appropriate.

In Figure 6.9(a), node a sends an ACK via its neighbor node b. If $initial\_value$ would be equal to 1, node a would consider that the ACK was lost because node b transmitted a data packet and would consequently retransmit the ACK in vain. We can easily see that the reason why node b started transmitting the data packet is that it did not receive yet the ACK from node a.

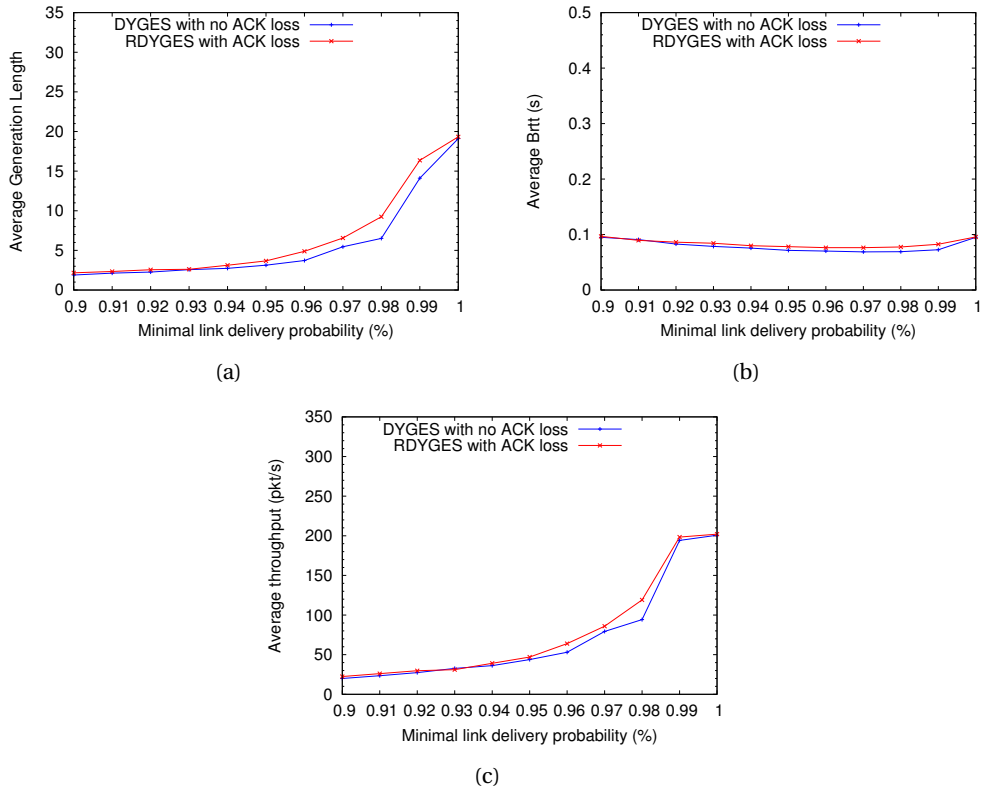
In the case that the ACK is lost and node b is idle such as in the example of Figure 6.9(b), choosing a high $initial\_value$ would uselessly increase the waiting time for retransmission. That is way we set $initial\_value$ to 2 for our simulations.



(a) (b)

Figure 6.9: An example showing the impact of setting $initial\_value = 2$.

6.6.3 Simulation results



(a) (b)

(c)

Figure 6.10: Evolution of (a) the average generation size, (b) $BRTT$ and (c) throughput when using RDYGES according to the delivery probability.

The simulation results plotted in Figure 6.10 show that RDYGES is really efficient since the obtained results are very close to the ones obtained with no ACK loss (see Figure 6.5).

We have finally overcome the issue of ACK loss while keeping the feature of smooth block-decoding delay and QoS guarantee.

6.7 Conclusion

We have proposed DYGES, a network-aware approach that dynamically adjusts the generation size according to network variations. We have showed by simulation it efficiency in a lossless context.

In this chapter, we have studied the issue of the DYGES's network-awareness in a lossy context. Since the data packet loss is thwarted using Network Coding and Redundancy, we mainly focused on the ACK loss and proposed an efficient algorithm named RDYGES to cope with this issue. Indeed, if DYGES could not relay on the MAC layer for a reliable ACK transmission, the system breaks down as it is very sensitive to ACK loss. Since the nodes use the opportunistic listening, RDYGES use this feature to analyze the ACK packets and retransmit them if needed.

It is shown by simulation that our approach keeps the block-decoding delay steady even when the link delivery probability varies and thus brings more robustness against losses.

Since The ACK loss is totally thwarted using RDYGES, we will use consider in the remaining of this thesis DYGES in a network with data loss only. We apply it, in the next chapter, for Live TV and video streaming in a particular heterogeneous network, namely, a Home Area Network.

Chapter 7

Application of DYGES on HANs

The work described in this chapter has been done in collaboration with Orange Labs, the R&D department of France Telecom.

This chapter addresses the problem of context awareness in Home Area Networks (HANs) for audio-visual services. The network is made up of devices having various transmission technologies, therefore the links have different properties leading to a multi-path scheme. Moreover, each application has its own requirements in terms of bandwidth, delay, jitter, etc.

In addition to the Network Coding benefits in terms of throughput, we use DYGES to tackle the delay issue of applying Network Coding in such an environment. We propose an architecture based on an extended Home Network to verify the effectiveness of DYGES. Through its network and content awareness, DYGES appears to be a good solution to provide Higher Definition applications to end-users with no additional resource or equipment in the HAN.

7.1 Introduction

During the last decades, many multimedia and audio-visual applications appeared such as transmission of image files, audio and video streams. Each application has its requirements in terms of bandwidth, delay, jitter, etc. Improving bandwidth and delay is a challenge for increasing the applications performance.

For the core network, replication-based networks such as Content Delivery Networks (CDN [54, 55, 56, 57]) and Peer-to-peer systems (P2P [58, 59, 60]) are very promising since they movie the data closer to the client and increase its availability. They also take into account the context (network, content, user, device) in order to improve the network performance and satisfy the application requirements and hence the QoS. However, these systems can

not be extended to the user HAN, they have limitations such as consuming more resources and requiring much management. Having less constraints and less resource consumption, Network Coding is a distributed approach that can be applied for both core network and HANs.

Random Network Coding needs only a local knowledge of the network topology to randomly mix packets in order to improve the throughput and robustness. We propose to apply DYGES in a HAN to ensure network-aware content delivery while considering the content and device contexts during the content delivery.

7.2 Problem statement

The context of our work is a Home Area Network (HAN) in which there are various equipments such as laptops, SD TV, HD TV, PDAs, personal devices, with different transmission technologies such as Power Line Communication (PLC), Ethernet, WiFi, WiMax, infrared, Bluetooth, etc. [61]. Therefore, the content can be disseminated through various paths.

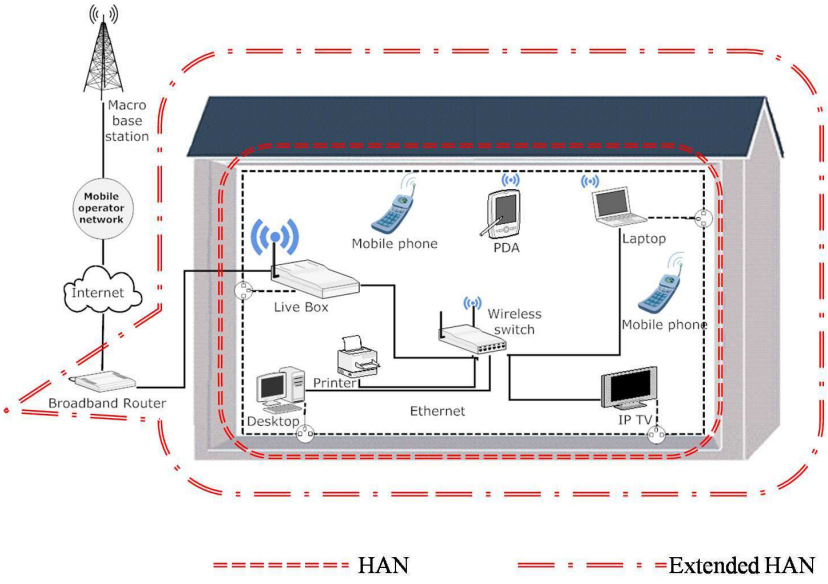


Figure 7.1: A HAN with multi-path transmissions and its Extended HAN.

Some categories of traffic such as video streaming need many resources like bandwidth. Therefore, it is difficult to guarantee a High Definition (HD) Live Video for all the consumers in their Home Networks. Our goal is to use the Network Coding collaborative nodes that increases the network performance while using the pre-existing devices. By a collaborative node, we mean a node that relays a flow even if it is not concerned by it. We also focus on an Extended HAN that contains the border router[1] and the HANs linked to it.

[1] In the remaining of this report, the term router designates the broadband router of the building.

For instance, in the extended Home Network depicted in Figure 7.1, the LiveBox could send data to more than one device via PLC, WiFi and Ethernet while using Network Coding. Each path has its own properties (Bandwidth, delay...) and each application has its requirements. The network topology depends on the mobility of the network devices. Moving a laptop, for instance, will change the preferred path and the interference impact. The goal here is to match between the network properties and the application requirements during the coding process. Improving some parameters such as the throughput and delay allows to guarantee the network Quality of Service (QoS) the Quality of Experience (QoE).

7.3 Context awareness

Network Coding is inherently network-aware since it considers the network topology for data mixing. By varying some parameters, a Network Coding based approach such as DYGES can consider the content and the device contexts. These considerations are explained as follow:

- **Network-context**
  Network Coding is built to take into account the topology changes during the routing (dissemination) process. Using redundancy, it makes a lossy network more robust against packet loss. DYGES adapts the generation size according to the network load, packet loss and congestion.

- **Service-context**
  DYGES varies the generation size to keep the block-decoding delay around a threshold based on the application requirements. For example, considering an application with a short delay constraint, the generation size can be adjusted to reduce the decoding delay.

  Applications do not have necessarily the same priority, for instance, a real time application has higher priority than file transfer. In this case, DYGES can prioritize the coding of the former application using small generations and schedule a span for coding the latter application's packets using larger generations.

  These priorities could be also set by the network operator function of the favorite traffic, for instance, if the network operator is relaying traffic for other actors (as OTT 'Over the Top' Players) weak priority could be given for this traffic.

- **Device-context**
  Considering that the network contains devices with various properties (computation and storage), DYGES can be adapted to the difference among the devices. For example, the generation size is decreased for devices with limited capacities to reduce the processing time of the Gaussian elimination.

In addition to this context awareness, we recall that DYGES improves the network reliability while increasing the throughput and respecting the QoS in terms of delay. Having all these advantages in mind, we propose in the next section a global architecture for content dissemination in a HAN.

7.4 Architecture

Our architecture has to increase the network performance by considering the existing coding opportunities with no additional equipment. An extended Home Network containing several HANs with the broadband router linked to them, is an interesting framework to verify the effectiveness of DYGES.

Our scheme uses only the devices already involved in the communication process. Besides, we assume the probability that customers watch the same TV program is not negligible. Hence, we try to take the benefits of this opportunity.

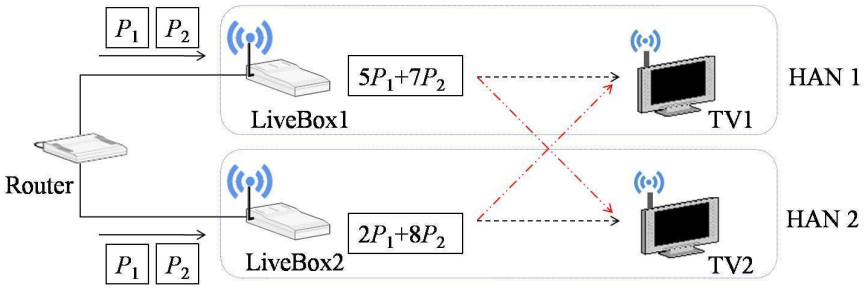


Figure 7.2: A multicast flow in an Extended HAN.

To explain our architecture design, we consider in Figure 7.2 two TVs belonging to two neighbor HANs and asking for the same packets more precisely there is a multicast flow, such as a Live TV traffic, from the router to TVs. Each Livebox must deliver the packets to its TVs. However using Network Coding, each Livebox generates a random combination and sends it to its TV through the wireless medium. We assume TV1 (resp. TV2) is in the reception range of Livebox2 (resp. Livebox1).

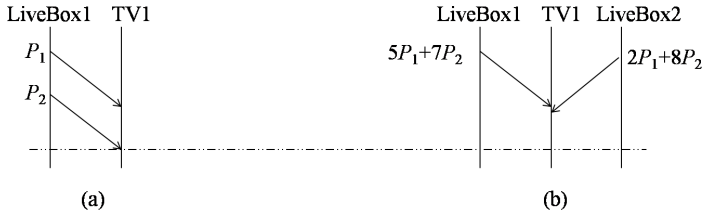


Figure 7.3: Using (a) a unique source without Network Coding versus (b) multiple sources with Network Coding.

Our main idea is to use the additional combinations heard by TVs to reduce the delay and increase the throughput. In the ideal case, each Livebox sends one combination and each TV receives two and thus doubles its throughput (considering a negligible overhead). Indeed, Figure 7.3(b) shows a shorter delay for receiving the two combinations by TV1.

Moreover, this scheme is scalable due to Random Network Coding. Indeed, in Figure 7.4, we involve another HAN without changing the coding mechanism. TV2 in this case multiplies its throughput by three in the ideal case. Thanks to multiple reception, Network Coding allows to improve the network performance while introducing no extra cost.

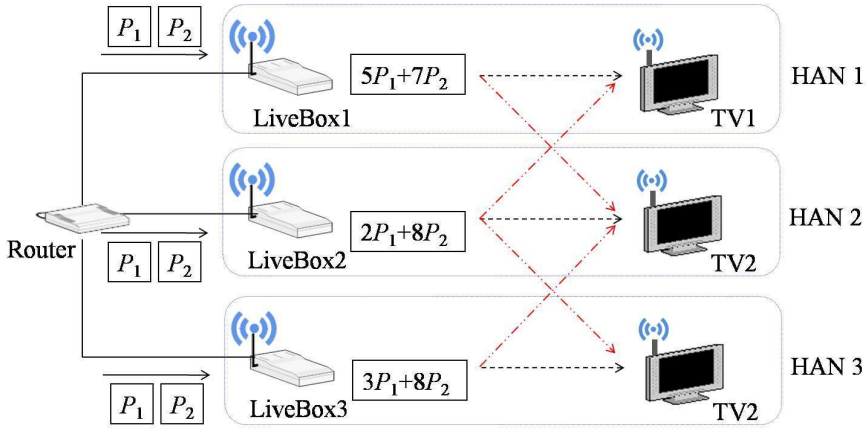


Figure 7.4: Scalability of Network Coding over HANs.

This scheme is appropriate to guarantee a higher definition for LiveTV streams using the already existing equipments[2], the various communication technologies and an open live box approach such as a femtocell [62, 63].

We describe in the next section a method for building the multicast tree that designates the different nodes involved in the communication scenario.

7.5 Communication scenario

We focus in this section on the communication scenario, in other words, how the devices communicate to set up the desired scheme. Based on our architecture, we propose two approaches according to the way the Liveboxes send their packets.

[2] In the remainder of the report, we will use the word TV to designate any device asking for a Video flow (Laptop, Desktop, Phone...).

7.5.1 Multicast

We consider here that when a Livebox sends the same packets to many nodes, it does it using a multicast frame. Consequently, we should choose which node hears to which Livebox. To do so, we need to gather the information about each HAN topology. The router is the best candidate in addition to the fact that it is the entrance of our architecture. Moreover, it is the only device allowed to have information about the traffic and topology of all the HANs it covers. We propose the following procedure:

1. The router has the knowledge of all TVs and Liveboxes concerned by a given flow (Live TV program).
2. When TV1 asks for this flow, it sends a request to the router via its Livebox1.
3. The router finds all the Liveboxes that have TV1 in their reception range and are already involved in the transmission of the given flow. It asks them to add TV1 in their destinations list.
4. TV1 hears all the packets that are destined to it.
5. If there is no TVs in HAN2, for instance, asking for the given flow, the router stops sending packets to Livebox2 (responsible of HAN2).

7.5.2 Pseudo-broadcast: Opportunistic listening

Inspired by [3], we assume that the nodes use opportunistic listening. The livebox sets only one TV address in the MAC address field. If another TV, TV2 for instance, hears that packet, it records it even if it is not destined to it. If the label (TV program identifier) does not fit with its requirements, TV2 removes it.

This approach is based on the fact that we can assign a label to each TV program and add this label in the data frame. The number of channels can be significant, therefore we focus only on the channels asked by the customers at the same time in the same building. The number of these channels is small thus we can use temporary labels: the broadband router assigns to each asked channel a label. When a channel is not asked in the building, the label is reassigned to another channel (recycling approach).

Note1: When a TV, TV1 for instance, receives a generation, it sends an ACK to the router through the path (unicast flow) TV1 → Livebox1 → router.

Note2: In our case, the Network Coding algorithms are implemented in the routing layer while inhibating the transport layer expiration timer. Since we manage the retransmission at the routing layer, we have to disable the TCP retransmission mechanism.

After choosing the appropriate architecture and communication scenario for content dissemination, we study in the next section the system behavior through simulations.

7.6 Impact of DYGES on an Extended HAN

In this section, we apply our approach on an Extended HAN scenario according to the chosen architecture.

Testbed: A pictorial representation of the extended HAN is shown in Figure 7.5. A multicast flow of a Live TV traffic is generated from the broadband router to both TVs using the Liveboxes as intermediate nodes. Other flows can be generated by the Laptops changing the network state (congestion and end-to-end delay).

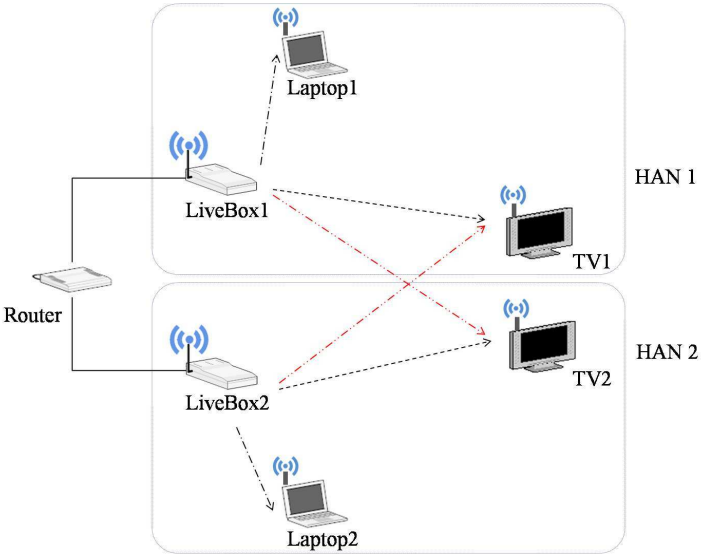


Figure 7.5: A communication scenario in a HAN.

The "No NC" approach: We compare the network performance when using DYGES and without using Network Coding, referred as "No NC". This last does not use opportunistic listening and can not take the advantage of having additional information through the links $1 \rightarrow 4$ and $2 \rightarrow 3$, referred as "cross links". "No NC" is an RTP-like (Real-time Transport Protocol) approach: the packets are endowed with time-stamps but there is no packet acknowledgement.

In addition, we manage the packet redundancy as for DYGES but by sending native packets instead of combined packets. We adapt the redundancy mechanism given for DYGES to the "No NC" approach. Each forwarder z has the knowledge of its Redundancy Factor Red_z

and uses a variable NUM_z initially set to 0. When receiving a packet, forwarder z computes the number of packets to send as follows:

$NUM_z = NUM_z + Red_z$,

Send $\lfloor NUM_z \rfloor$ times packet P,

NUM_z = fractional part of NUM_z.

Simulation parameters: The link delivery probabilities are given in the network graph shown in Figure 7.6. Various unicast flows $5 \rightarrow 1$ and $6 \rightarrow 2$ are generated sporadically. Their lifetime follows a Poisson distribution with a parameter $\lambda = 5s^{-1}$ and the flow inter-arrival follows a Poisson distribution with $\lambda = 50s^{-1}$. The batch expiration time is set to $110ms$ and the threshold ($thresh$) to $95ms$.

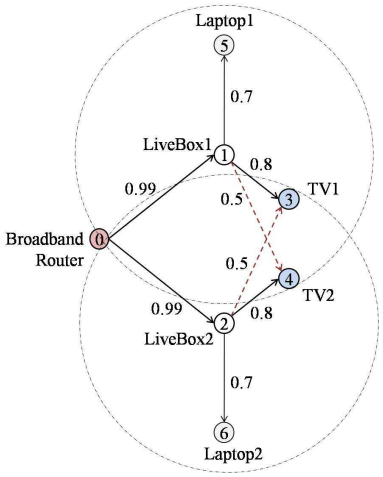


Figure 7.6: The testbed network graph.

7.6.1 Throughput analysis

We plot in Figure 7.7 the evolution, according to time, of the average throughput of the received packets (goodput) at the destination's application layer. The redundant packets are not considered since they are deleted at the destination.

At the beginning of simulation the throughput is higher for the "No NC" approach because the routing layer sends the received packets immediately to the application layer. As for DYGES, the packets have to be decoded first and are thus delayed.

Then, the unicast flows disturb the multicast flow decreasing the performance for "No NC", while DYGES seems to be more robust because it uses the additional cross links. Since

DYGES applies the redundancy mechanism more efficiently using combined packets, the resulting throughput finally stabilizes at a higher value, a gain of 18.73% compared to “No NC”.

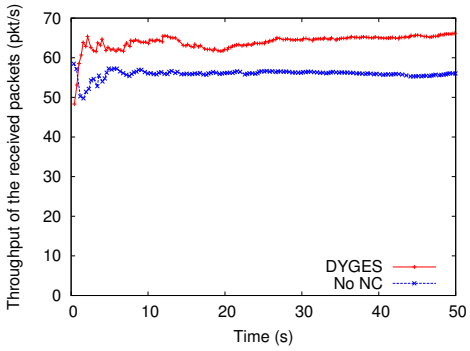


Figure 7.7: Evolution of the average throughput according to time

7.6.2 Retransmission effect

Even if a batch timer expires before decoding it, it is necessary to resend the batch in order to make the receiver decoding the generation and retrieving some packets that did not exceed their individual deadline.

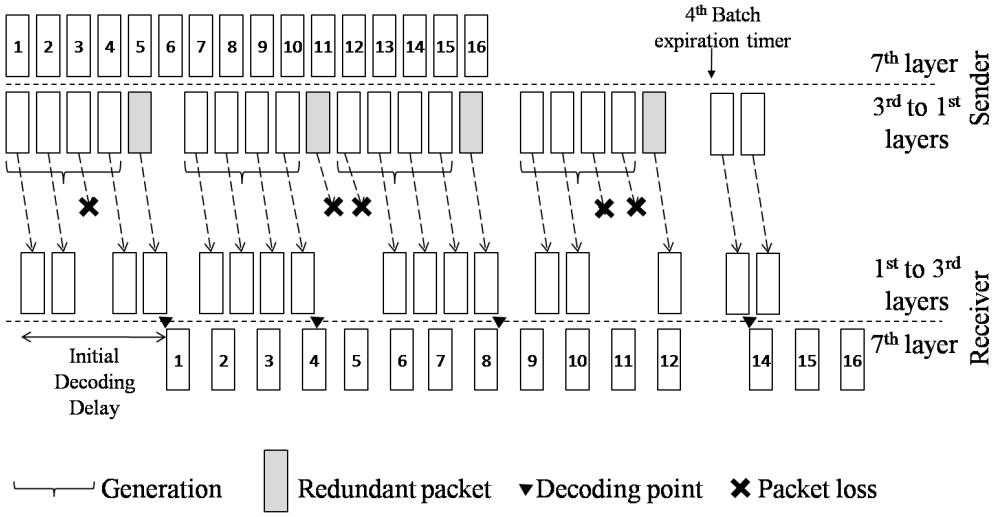


Figure 7.8: The effect of DYGES on individual packet delay.

In Figure 7.8, the source wants to send 16 packets to the destination. For simplicity's sake, we consider a generation size of four packets. Using four native packets, the source builds five combined packets, one of which is due to redundancy. As shown in Figure 7.8, the destination's application layer needs packet arrivals with a given rate, if a packet is received before a given time, it is saved in a temporary buffer, but if it is lately received, it is considered as lost. We recall that this constraint is necessary since we deal with Live TV applications.

Due to redundancy that hides the loss of one combined packet, the first generation is received and decoded in time, thus the application layer can receive the four first packets according to the expected rate. Consequently, packets P_1 to P_{12} are correctly received by the application layer assuming an initial decoding delay (sometimes referred as initial buffering delay) tolerance. The 2^{nd} and 3^{rd} generations are received similarly but the 4^{th} generation reception was delayed. Indeed, the redundancy computation is based on the average delivery probability, and in this case, unpredictably more than one loss occurred in the same generation. The source did not receive an ACK and the 4^{th} generation timer expired triggering the batch retransmission. After receiving one additional combined packet, the destination could decode the packets and send an ACK that stops the batch retransmission phase.

Here, even if the 13^{th} packet was late and considered lost by the application layer, the retransmission was useful to deliver the 14, 15 and 16^{th} packets in time. In this sense, the limitation of a decoding-delay threshold ($thresh$) is necessary for the targeted application. However, we recall that DYGES needs a tolerance of an initial decoding delay.

7.6.3 Packet loss and delay analysis

For our simulation, let's add the following constraint: the destination's application layer needs packets at a rate of $50pps$ (packets per second) i.e. a packet inter-arrival of $0.02s$ and all the late packets are considered lost. We assume a strict system that does not tolerate an initial decoding delay. We plot in Figure 7.9 the proportion of lost packets at the application layer.

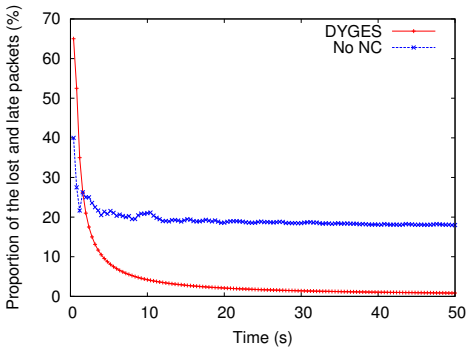


Figure 7.9: Evolution of the loss rate according to time

We recall that the loss at the destination's application layer is due to the lost packets for "No NC" and lately received packets for DYGES. Since the received packets for "No NC" are immediately sent to the application layer, there is no late packets. The DYGES's retransmission mechanism make the delivery reliable and thus no packet is lost. "No NC" can not efficiently cope with the packet loss. The redundancy mechanism reduces the packet loss rate from 20% (link delivery probability) to 17.11%, this leads to a degraded application quality (reception of 82.89% of the streaming information).

As for DYGES, there are many late packets at the beginning of simulation because, as shown in Figure 7.8, an initial decoding-delay is needed. After the first generations, there are enough packets in the buffer of the application layer to hide the generation based-arrivals and thus keep a steady throughput. Using a random Network Coding, an opportunistic listening approach and an efficient redundancy mechanism increases the overall throughput. Using a decoding-delay threshold leads to respect the individual packet delay (QoS). Consequently the QoE is increased (reception of 99.84% of the streaming information), and thus a Higher Definition application is provided for the end-user.

7.7 Conclusion

In this chapter, we have dealt with the issue of context awareness in heterogeneous networks and more particularly HANs where the links can have different properties and the applications various requirements.

In order to increase the QoS and QoE for video streaming and Live TV applications, we have chosen a TV and Video delivery architecture based on Network Coding. DYGES appears to be a good solution to increase the throughput and manage the delay constraint.

We have proposed an architecture which takes the opportunity of having many transmissions of the same information to guarantee a better performance with no additional device.

We have then applied DYGES in an example of a HAN scenario and compared it to an RTP-like approach that does not use Network Coding. The results show that the throughput obtained with DYGES is higher while respecting the individual packet delay. Consequently, for initial decoding-delay tolerant applications, DYGES provides Higher Definition to end-users with no additional equipment.

Part IV

Capacity increase in Butterfly based networks

Chapter 8

Generalized Butterfly Network

The work described in this chapter has been done in collaboration with the Department of Systems Science, Kyoto University, Japan.

As have seen it in the previous chapters, Network Coding brings many advantages such as reducing the bandwidth consumption and increasing the throughput. Its evolution started with a study of the well-known Butterfly scenario, then the theoretical works proved its efficiency and the random Network Coding made possible to apply it in a distributed way.

In this chapter, we deal with the theoretical analysis of a system using both butterfly sub-networks and Random Network Coding. Indeed in the literature, the butterfly scenario is mostly used with a deterministic code assignment algorithm. We propose a Generalized Butterfly (GBFLY) Network that applies random Network Coding for a multi-source multicast flow. Its gain outperforms the simple butterfly scheme due to shared links. We study the network performance in terms of source node buffer size using both queuing theory and simulations.

8.1 Introduction

The classical example for Network Coding is the well known Butterfly network in which a multicast flow is realized with less transmissions than in a non Network Coding context. However, this scenario is used with a deterministic code assignment algorithm, which means that the coding/decoding scheme has to be agreed upon beforehand and needs full network topology.

Using linear random Network Coding, Generation-Based Network Coding (GBNC) guarantees robust distributed transmissions. We recall that it gathers the packets into generations and combines the packets belonging to the same generation (refer to Sections 2.2.5 and 6.1).

Convinced that linear random Network Coding is beneficial, we believe that a complete

redesign of multi-source multi-destination streaming algorithm based on butterfly sub-networks is necessary to take full advantage of Network Coding. Therefore, we propose a Generalized ButterFLY (GBFLY) Network that deals with the multi-source multicast issue.

8.2 Related work

As we have shown in Section 2.2, using the Butterfly, the number of transmissions is reduced form 10 to 9. By reducing the number of transmissions, Network Coding minimizes the bandwidth consumption, increases the throughput and reduces the delay.

The contribution of [1] and [2] deals with both single-source and multi-source problems but does not investigate the buffering aspect of the studied schemes.

The authors of [64] propose R^2 an algorithm that uses random Network Coding in the push design for the P2P streaming case. However, they consider segment priority (urgency of delivery).

The butterfly network is not only used in the theoretical works, BFLY [4], for instance, enhanced the first Network Coding based protocol COPE [3] by looking for butterfly-shaped sub-networks comprising two flows.

There are various works in the literature that deal with the queuing analysis and decoding delay for butterfly and other network topologies.

TCPNC was proposed in [6], it combines the TCP segments using Network Coding and thus outperforms the traditional TCP approach. The Queuing analysis shows that in the steady state the queues size of the involved nodes is stable.

In [65], the authors study the COPE's scheme from the queuing theory point of view. They assumed a limited buffer capacity and a random number of packets to be coded. Using a Markov chain model, they propose a computation of the steady state probabilities. When studying the departure process, they show that it was not a Poisson process even if the arrivals and service durations were Poisson.

A performance evaluation of random linear Network Coding is proposed in [66]. The simulation results confirm the impact of number of packets used in a combination on increasing the coding opportunities and thus the coding gain. Moreover, they show that a large or infinite buffer size is useless.

In [67] and [68], the authors investigate the buffer size and delay of a simple Butterfly wireless network. They study the achievable rate region when using noisy and non-symmetric links and show that in some cases classical routing outperforms Network Coding.

We aim at proposing and analyzing a more general Butterfly (GBFLY) network that com-

prises various Butterfly sub-networks deployed in such a way they satisfy a multi-source multi-destination flow.

8.3 Context of study and model

We consider multiple sources of the same streaming traffic to multiple destinations. Network Coding is used by all the nodes involved in that traffic. The intermediate nodes receive the packets, combine and forward them with no need to decode them.

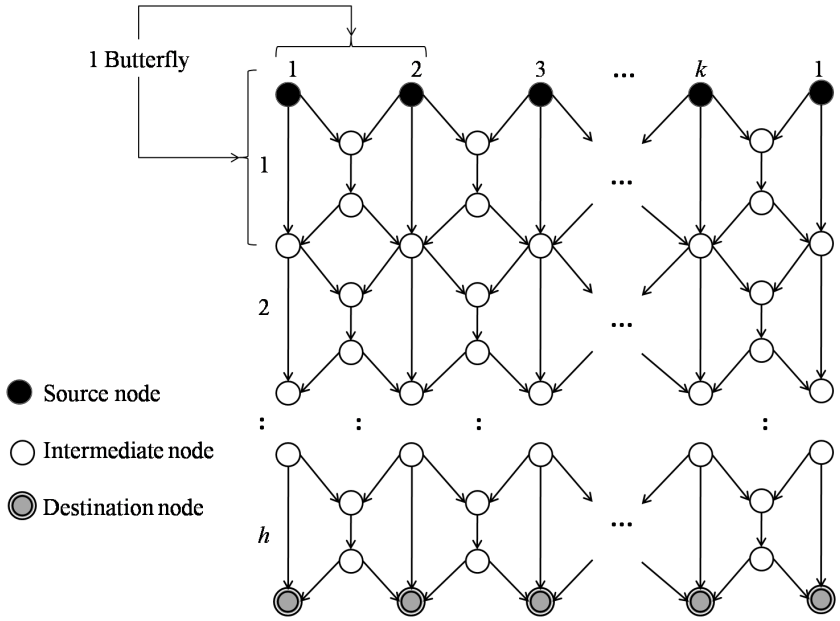


Figure 8.1: GBFLY Network with a multi-source multicast flow.

We consider k sources and k destinations such that $k > 2$. As shown in Figure 8.1, the GBFLY network is made with $k * h$ butterfly sub-networks such that $h > k$. For simplicity's sake and to avoid the boundary effect, we have considered that the 1^{st} and $k+1^{th}$ sources are the same and similarly the 1^{st} and $k+1^{th}$ destinations (*torus form*).

The packet generation size is set to $m*k$ where $m \in \mathbb{N}^*$. Every source node uses the $m*k$ native packets to generate and deliver m encoded packets. The intermediate nodes encode the received packets that belong to the same generation. Upon receiving $m*k$ independent combinations, each destination node applies a Gaussian elimination to decode and recover the original packets.

The basic case of our scheme corresponds to $k = 3$ and is depicted in Figure 8.2. Assuming a generation of three packets P_1, P_2 and P_3, each source and intermediate node (also referred as forwarder) builds a random combination $[\alpha.P_1 + \beta.P_2 + \gamma.P_3]$. Each destination receives the three independent combinations and thus recovers the three original packets.

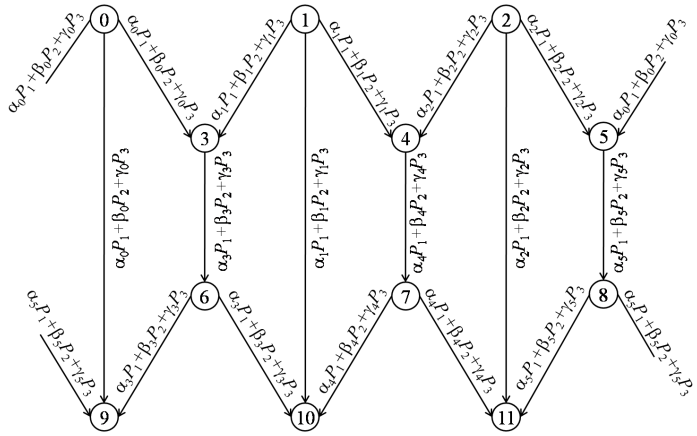


Figure 8.2: GBFLY scenario with $k = 3$.

Each destination has a buffer (See Figure 8.3) to save packets. Packets remain in the buffer until a correct decoding i.e. when the number of innovative packets in the buffer is equal to the generation size. A part of the Network Coding gain comes from the shared links where a transmission could provide additional information and hence avoid transmitting more packets. Typically, on the butterfly network, the shared link is the one in the middle of the network.

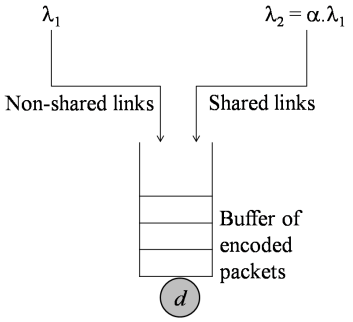


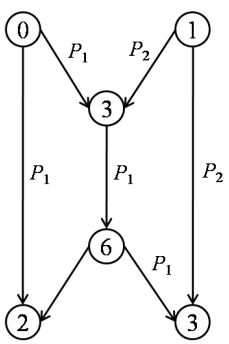
Figure 8.3: Destination node queue.

Considering this observation, the buffer is receiving the packets according to two different arrival processes:

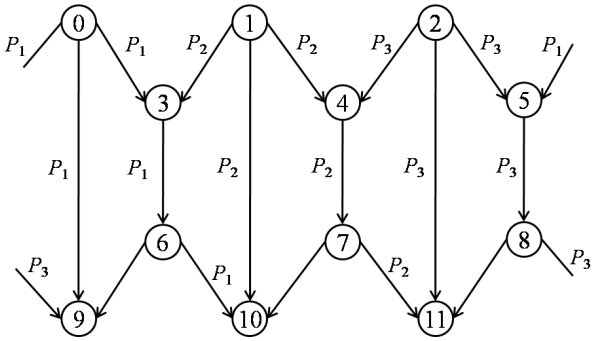
- Packets arriving from non-shared links where the arrival process should have the same arrival parameter than the closest source of the destination. Congestion is not considered in our model.
- Packets arriving from shared links where packets could accelerate the decoding process. During the same unit of time, this arrival process provides more packets than when using only non-shared links. The parameter of the arrival process here is smaller of a factor of α.

8.4 GBFLY 's Gains

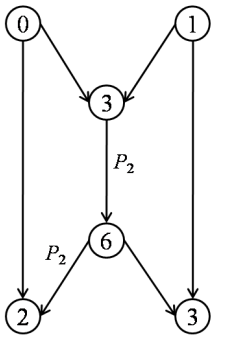
Compared to the simple Butterfly network, the design of GBFLY increases the number of shared-links by destination. This leads to increase the Network Coding gains.



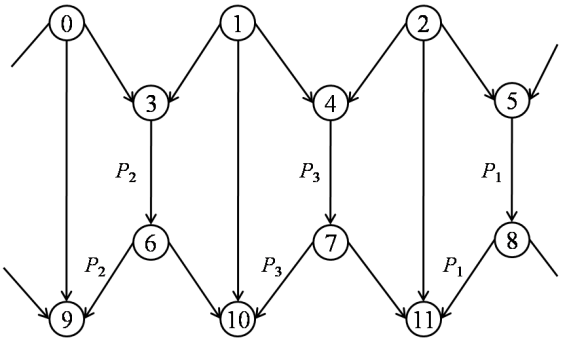
(a) Butterfly network without Network Coding



(b) GBFLY network ($k = 3$) without Network Coding



(c) Butterfly network with Network Coding



(d) GBFLY network ($k = 3$) with Network Coding

Figure 8.4: Simple-Butterfly and GBFLY networks.

The simple Butterfly wired network is known to have a gain[1] of (see Figures 8.4(a) and 8.4(c)):

8/7 in terms of number of transmissions.

2/1.5 in terms of "max-flow".

Here, we study the gains of a GBFLY network having three sources ($k = 3$) and using a generation of three packets P_1, P_2 and P_3.

Using the traditional approach, the shared links $3 \rightarrow 6$, $4 \rightarrow 7$ and $5 \rightarrow 8$ have to successively transmit the packets one by one. Figure 8.4(b) shows that 21 transmissions are needed to make the three destinations 9, 10 and 11 receive the three packets. In terms of "max-flow", the three sources receive at most $2 * 3$ flows at the same time.

Using Network Coding, the shared links are allowed to mix the packets. Figure 8.4(d) shows that only 15 transmissions are needed to make the three destinations receive the three packets. In terms of "max-flow", the three sources receive at most $3 * 3$ flows at the same time.

Consequently, the GBFLY gains show that our design takes more advantages of Network Coding. Indeed, they are given by:

21/15 in terms of number of transmissions.

3/2 in terms of max-flow.

The next section show the theoretical results using queuing theory.

8.5 Theoretical Results

In this section, we present the queuing analysis of GBFLY from the queuing theory point of view. We characterize the average source node buffer size.

We use the $M/G/1$ model[2] of the *Embedded Markov chain theory*. Let λ be the arrival rate following a Poisson distribution and B the service time, here, the transmission time (per packet) distribution with an average b. The average source node buffer size is then given by:

$$\frac{(m^2+m)\lambda b - m^2\lambda^2 b^2}{2(1-m\lambda b)^2} \times (1-\lambda b) \tag{8.1}$$

[1] A gain is the ratio of the Network Coding on a classical approach.

[2] $M/G/1$ models a single server, where arrivals are determined by a Poisson process and job service times can have an arbitrary distribution.

Proof. We first consider a model with a single packet arrival and no Network Coding.

Let π_i be the probability of having i packets in the buffer.

$$\pi_j = \sum_{i=1}^{j+1} \pi_i a_{j-i+1} + \pi_0 a_j \tag{8.2}$$

with a_i the probability of having i packets arrivals in one transmission time.

$\Pi(z)$ is the Probability Generating Function (PGF) of π_i:

$$\Pi(z) = \frac{A(z)(z-1)}{z - A(z)}(1-\lambda b) \tag{8.3}$$

with $A(z)$ the PGF of a_i, it is given by:

$$A(z) = B^*(\lambda - \lambda z) \tag{8.4}$$

where $B^*(s)$ is the Laplace-Stieltjes Transform (LST) of $B(x)$:

$$B^*(s) = \int_0^\infty e^{-sx} dB(x) \tag{8.5}$$

By considering B, the transmission time, constant,

$$B^*(s) = e^{-sb} \tag{8.6}$$

For the single arrival process, the average buffer size corresponds to the first moment of $\Pi(z)$:

$$E[N] = \Pi'(1) = \frac{2A'(1).(1-A'(1)) + A''(1)}{2(1-A'(1))^2}(1-\lambda b) \tag{8.7}$$

For the GBFLY case, during each arrival, a source node receives a generation of $m * k$ and immediately generates m combinations and save them into its sending buffer. Therefore for the batch arrival, we have:

$$A_{batch}(z) = B^*(\lambda - \lambda z^m) = e^{-(\lambda - \lambda z^m)b} = A(z^m) \tag{8.8}$$

then,

$$\Pi_{batch}(z) = \frac{A(z^m)(z-1)}{z - A(z^m)}(1-\lambda b) \tag{8.9}$$

$$\begin{aligned}\Pi'_{batch}(z) &= [(A(z^m)(z-1))'.(z - A(z^m)) - (z - A(z^m))'.(A(z^m)(z-1))] \times \frac{(1-\lambda b)}{(z-A(z^m))^2} \\ &= [(mz^{m-1}A'(z^m)(z-1) + A(z^m)).(z - A(z^m)) + (mz^{m-1}A'(z^m) - 1).A(z^m)(z-1)] \\ &\quad \times \frac{(1-\lambda b)}{(z-A(z^m))^2}\end{aligned} \tag{8.10}$$

When $z=1,\ A(z^m)=1$ then $\Pi'_{batch}(1)=\frac{0}{0} \implies$ Apply l'Hopital rule [3]

$$\begin{aligned}
&\Pi'_{batch}(z) = \\
&= [((m(m-1)z^{m-2}A'(z^m)+m^2z^{2m-2}A''(z^m))(z-1)+2mz^{m-1}A'(z^m)).(z-A(z^m)) \\
&+(mz^{m-1}A'(z^m)(z-1)+A(z^m)).(1-mz^{m-1}A'(z^m)) \\
&+(m(m-1)z^{m-2}A'(z^m)+m^2z^{2m-2}A''(z^m)).A(z^m)(z-1) \\
&+(mz^{m-1}A'(z^m)-1).(mz^{m-1}A'(z^m)(z-1)+A(z^m))] \\
&\times \frac{(1-\lambda b)}{2(1-mz^{m-1}A'(z^m)).(z-A(z^m))}
\end{aligned} \tag{8.11}$$

When $z=1$ then $\Pi'_{batch}(1)=\frac{0}{0} \implies$ Apply again l'Hopital rule.

$$\begin{aligned}
&\Pi'_{batch}(z) = \\
&[m(m-1)(m-2)z^{m-3}A'(z^m)(z-A(z^m)(z-1))+3m^2(m-1)z^{2m-3}A''(z^m)(z-A(z^m))(z-1) \\
&+m^3z^{3m-3}A'''(z^m)(z-A(z^m))(z-1)+3m(m-1)z^{m-2}A'(z^m)(z-A(z^m)) \\
&+3m^2z^{2m-2}A''(z^m)(z-A(z^m))+m(m-1)z^{m-2}A'(z^m)(1-mz^{m-1}A'(z^m))(z-1) \\
&+m^2z^{2m-2}A''(z^m)(1-mz^{m-1}A'(z^m))(z-1)+m(m-1)z^{m-2}A'(z^m)mz^{m-1}A'(z^m)(z-1) \\
&+m^2z^{2m-2}A''(z^m)mz^{m-1}A'(z^m)(z-1)+m(m-1)(m-2)z^{m-3}A'(z^m)A(z^m)(z-1) \\
&+3m^2(m-1)z^{2m-3}A''(z^m)A(z^m)(z-1)+m^3z^{3m-3}A'''(z^m)A(z^m)(z-1) \\
&+2mz^{m-1}A'(z^m)(1-mz^{m-1}A'(z^m))+m(m-1)z^{m-2}A'(z^m)A(z^m)+m^2z^{2m-2}A''(z^m)A(z^m)] \\
&\times \frac{(1-\lambda b)}{2(-m(m-1)z^{m-2}A'(z^m))(z-A(z^m))-2(m^2z^{2m-2}A''(z^m))(z-A(z^m))+2(1-mz^{m-1}A'(z^m))^2}
\end{aligned} \tag{8.12}$$

When $z=1$

$$\Pi'_{batch}(1)=[(m^2+m)A'(1)-2m^2(A'(1))^2+m^2A''(1)]\times\frac{(1-\lambda b)}{2(1-mA'(1))^2} \tag{8.13}$$

We recall that $A(z)=B(\lambda-\lambda z)$, then

$$A'(z)=-\lambda.B'(\lambda-\lambda z) \text{ then } A'(1)=\lambda.E[B]=\lambda b \tag{8.14}$$

$$A''(z)=(-\lambda)^2.B''(\lambda-\lambda z) \text{ then } A''(1)=\lambda^2.E[B^2]=\lambda^2b^2 \tag{8.15}$$

Finally, the average buffer size is:

$$\Pi'_{batch}(1)=\frac{(m^2+m)\lambda b-m^2\lambda^2b^2}{2(1-m\lambda b)^2}\times(1-\lambda b) \tag{8.16}$$

[3] L'Hopital rule:

If $\lim_{x\to c}f(x)=\lim_{x\to c}g(x)=0$, $\lim_{x\to c}\frac{f'(x)}{g'(x)}$ exists and $g'(x)\neq 0$ for $x\neq c$ then $\lim_{x\to c}\frac{f(x)}{g(x)}=\lim_{x\to c}\frac{f'(x)}{g'(x)}$.

□

Having the average buffer size theoretical computation, we compare it in the next section to the one computed by simulation within a GBFLY context.

8.6 Performance Evaluation

This section presents the numerical results for the network model relevant to our approach. We implement in our time-based network simulator, written in C++ under Opnet 15.0, a routing layer endowed with Network Coding and an ideal MAC layer.

We deploy wired GBFLY network ($k = 3$) depicted in Figure 8.2 At the beginning of the simulation, a multicast flow is created and lives until the end of simulation. It is a multi-source multi-destination flow from nodes 0, 1, 2 to 9, 10, 11 using intermediate nodes 3, 4, 5, 6, 7 and 8.

The flow has a batch inter-arrival according to a Poisson distribution with a parameter λ. For every batch arrival, each of the three source nodes receives a set of $m * k$ packets from the application layer. The batch size $m * k$ is set to $2 * 3$. To build a combination, the coefficients are chosen in a Galois Field modulo[251]. The sending buffer of an intermediate node has an unlimited capacity. The network is lossless and each link transmits a packet in $b = 10ms$.

A scenario of communication would be as follows: Upon the reception of a generation (three packets), each of the three source nodes encodes those packets to generate one encoded packet and transmits it to the three neighboring nodes. The three source nodes start encoding and transmit encoded packets at the same time. Then, the intermediate nodes re-encode the packets and forward them to the destination nodes. These last wait until the reception of $(m * k)$ packets to decode the batch and recover the original packets.

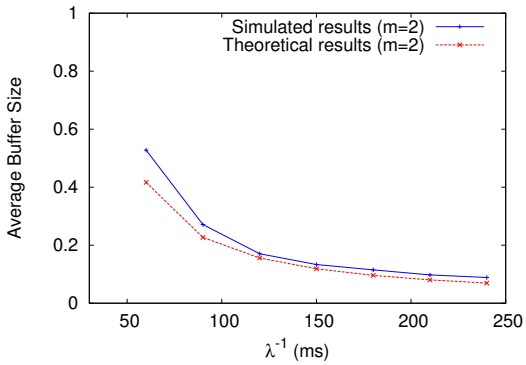


Figure 8.5: Evolution of the average source node buffer size according to the batch arrival λ for both theoretical and simulation-based approaches.

For the buffering model, we use the following *flushing policy*:

- Each source or forwarder node renews the batch if it has already sent $flush\_thresh$ packets in that batch. This value is set to 1 for our simulations.
- The destination node renews the batch after decoding all the packets belonging to the batch.

The simulation results depicted in Figure 8.5 show the evolution of the average buffer size of a source node when varying λ from 30^{-1} to $240ms^{-1}$. We notice that the higher is the batch inter-arrivals duration, the lower is the buffer size. Moreover, the difference between the average buffer size obtained using the theoretical approach and the one obtained by simulation is very small and shows the accuracy of the buffer size theoretical calculation.

8.7 Conclusion

In this chapter we have discussed the multi-source multicast performance issue. We have first proposed a new network scheme, GBFLY, for Network Coding. We have shown that, based on many butterfly sub-networks, the shared links are increased and thus the gain in terms of number of transmissions and max-flow min-cut outperforms the simple butterfly one.

A first step was to characterize the average source buffer size using the queuing theory. We have then shown that the theoretical results meet those obtained by simulation.

The next chapter presents the conclusions emerged from the analysis of the various topics of this thesis and proposes some directions for further research.

Part V

Conclusion

Chapter 9

Conclusion and future work

During the last decades, wireless networks were widely deployed due to the wireless devices evolution and proliferation. Beside cellular networks, wireless multi-hop networks represent a good solution to provide anywhere anytime connectivity. Indeed, they do not need any infrastructure and the nodes are self configuring. Furthermore, they are used to extend the wired networks to overcome geographical or cost issues.

Many applications such as video streaming and Voice over IP require a certain QoS but do not fit to wireless networks, they consider them as wired. Due to the shared nature of the medium, wireless networks suffer from interference and fading phenomena leading to increase the delay and reduce the reliability and throughput.

In this context, Network Coding appears as a very promising approach. It takes the advantage of the medium shared nature to improve the network performance. It allows the intermediate nodes to mix the information instead of simply forwarding it and consequently reduces the number of transmissions. That is why it has received much attention in the literature and was proposed to be applied in many types of networks.

Our purpose in this thesis was to use Network Coding as a tool and extend the work done to provide QoS in wireless multi-hop networks. To achieve this goal, we studied the different Network Coding schemes and chose the most appropriate ones according to the targeted applications (unicast, multicast, multi-source multi-sink, delay sensitive, bandwidth sensitive).

9.1 Summary of contributions

The first part of this thesis focused on routing unicast flows having a strict bandwidth requirement. The goal was to maximize the number of accepted flows while guaranteeing their QoS.

We improved an existing Network Coding based algorithm (ROCX) by considering the

interference impact. The idea behind ROCX is to look for Alice & Bob schemes in the network and mix the packets at their level leading to reduce the bandwidth consumption. However, ROCX does not take into account the interference phenomenon. Indeed in a wireless network, any routing algorithm should consider the interference impact during the routing process otherwise the required bandwidth is not satisfied. That is why we proposed an improvement of the clique-based interference model and applied it in the I2ILP algorithm. This last uses linear programming and achieves the upper bound for network capacity when rerouting is not considered. Next, we combined ROCX and IILP. The resulting routing algorithm, IROCX, maximizes the benefits of Network Coding (reducing the number of transmissions) while considering the interference impact. Being both interference- and coding-aware, the algorithm we proposed guarantees the QoS and improves the overall throughput.

To still increase the network performance, we used another technique that is Topology Control. It allows nodes to transmit using a multilevel power to control the interference effect. The impact of Topology Control on the Alice & Bob scheme was not obvious because several interference areas are involved. To verify the positive impact of Topology Control even on Network Coding based algorithm, we adapted the way of computing the mutually interfering links and proposed the TC-IROCX algorithm. Network Coding reduces the number of transmissions, the clique-based model guarantees the QoS, Topology Control adjusts the interference area, rerouting balances the load over the cliques. Thanks to these features, almost 100% of the network resources were used to maximize the flow acceptation rate while respecting the QoS.

The second part of this thesis dealt with delay constraints for multicast flows. We considered non critical real-time applications over a collaborative network.

Unlike the first part, we used Random Network Coding and more precisely GBNC. Using a fixed generation size, the decoding delay varies according to the network variations (network size, congestion, loss), hence, some packets can be lately decoded and the QoS degraded. That is why we proposed the DYGES approach that adjusts the generation size dynamically. It is both content- and network-aware since it adapts the generation size according to an application-dependent parameter and the network load. Aiming at maximizing the throughput while respecting a certain decoding delay, we showed in a lossless context that it maintains indeed the decoding delay steady.

Then, we studied the issue of the DYGES's network-awareness in a lossy context. Unlike the ACK loss, data packet loss is thwarted using Network Coding and Redundancy. We showed that the ACK loss has a dramatic impact on DYGES performance. To overcome this problem, we took the advantage of the opportunistic listening mode which is already used for data packets to propose the RDYGES algorithm . This last makes the ACK transmission reliable and keeps the decoding delay steady even when the link delivery probability varies and thus brings more robustness against losses.

Due to its benefits, DYGES appeared as an interesting solution to increase the QoS and

QoE for video streaming and Live TV applications in Home Area Networks. To do so, we proposed an architecture of an extended HAN that takes the opportunity of having many transmissions of the same information. Using this testbed, DYGES thwarts the loss and significantly increases the throughput. We inferred that, for initial decoding-delay tolerant applications, our solution provides Higher Definition to end-users with no additional equipment.

The third part of this thesis discussed the multi-source multicast performance issue. We proposed a torus scheme based on many Butterfly sub-networks on which we applied Random Network Coding . In the new network, GBFLY, the number of shared links is increased and thus the gains in terms of number of transmissions and max-flow min-cut outperform the simple Butterfly ones. We characterized the average source buffer size using the queuing theory and showed that the theoretical results meet those obtained by simulation.

9.2 Issues for further work

Increasing the throughput and reducing the delay using Network Coding have been the main focus of many works in the literature. Hoping that the contributions cited above close few issues but open new perspectives, we point out some thoughts for further considerations.

The approach presented in the first part is centralized and needs a global knowledge. As a future work, it would be interesting to develop a distributed algorithm that discusses the way the nodes compute the cliques locally and apply a distributed rerouting. Another illustration of a possible extension to this work is to introduce new schemes, like the Butterfly one, in the model of TC-IROCX to still improve the network performance. Furthermore, TC-IROCX does not deal directly with nodes energy constraints, introducing them will have a direct impact on the network lifetime. Besides, we considered for our simulations a unit and a doubled transmission power, it is useful to investigate the TC-IROCX behavior when there are many power levels, even smaller than the unit one. In that case, the network connectivity issue has to be carefully addressed.

In the second part, we managed the generation size but considered a fixed generation size increment. It would be useful to analyze the DYGES behavior when using a dynamic increment. However, this increment should be carefully computed to avoid frequent $BRTT$ oscillations that may increase the batch expirations and thus decrease the network performance. As a possible further improvement, this work can be extended by investigating more parameters such as the field size, the redundancy factor, the threshold and the default generation size.

DYGES is designed for limited size networks, we would like to scale it to arbitrarily large numbers of users. To do so, we plan to reduce the ACK explosion using a NACK (Negative ACK) based approach such as NORM [69, 70]. Since DYGES is intra-flow which means only packets belonging to the same generation of one flow are mixed together, it could be interest-

ing to study the behavior of an inter-flow approach mixing packets of different flows.

Finally, the third part represents a first step to characterize the average buffer size of the source node for the GBFLY scheme. A logical sequence is to characterize by queuing theory the average buffer size for both intermediate and destination nodes and compare them to a simple Butterfly network. Another point is to design an efficient flushing policy for lossy networks. It should not be complex, otherwise it would be hard to model it using the queuing theory.

In addition to the research directions cited above, there are common perspectives to the topics studied in this thesis. For example, addressing the impact of mobility would allow to investigate in a more general way the behavior of our solutions. Such as the main works on networking, we used simulations to evaluate the effectiveness of our approaches, it would be useful to implement them and propose the corresponding protocols to measure their performance in real networks. This will very likely raise some new challenges.

In a more general way, Network Coding opens the way to further research. For example, it would be interesting to investigate in deep the synergy between Network Coding applied, on the one hand, at the routing layer and on the other hand, at the physical layer (PLNC [17]). Instead of reducing the number of transmissions in the Alice & Bob scheme from four to three, the application of Network Coding at the physical layer reduces it to two. The major focus should be dedicated to make each layer's behavior aware of the other one.

Another application of Network coding could be to use it in the core network for load balancing. A Multi-Layer Switch (MLS) is an equipment that distributes the workload across multiple servers to prevent a server overload and yet it selects for each request a server that sends native packets. Integrating the Network Coding paradigm for the selection of many servers in which each one provides a part of the information seems to be a good research direction.

Appendix A

List of publications

The main results and findings of this thesis have been published prior to the defence in a number of international conferences/journals/books, as follows:

Conferences

1. Youghourta Benfattoum, Steven Martin, Ignacy Gawedzki and Khaldoun Al Agha, "I2A-SWP: Manet routing considering intra-flow interference", *Asia-Pacific International Symposium on Advanced Reliability and Maintenance Modeling (APARM'10)*, Wellington, New Zealand, pages 65-72, 2–4 December 2010.

2. Youghourta Benfattoum, Steven Martin and Khaldoun Al Agha, "IROCX: Interference-aware Routing with Opportunistically Coded Exchanges in Wireless Mesh Networks", *IEEE Wireless Communications and Networking Conference (WCNC'11)*, Cancun, Mexico, pages 1113-1118, 28-31 March 2011.

3. Youghourta Benfattoum, Steven Martin and Khaldoun Al Agha, "TC-IROCX: Network Coding with Topology Control and Interference awareness", *IEEE Wireless Communications and Networking Conference (WCNC'12)*, Paris, France, pages 1997-2012, 1-4 April 2012.

4. Youghourta Benfattoum, Steven Martin and Khaldoun Al Agha, "DYGES: A network-aware Generation-Based Network Coding for multicast flows", *IEEE Vehicular Technology Conference (VTC2012-Fall)*, Quebec, Canada, 3-6 September 2012.

Submissions

5. Youghourta Benfattoum, Steven Martin and Khaldoun Al Agha, "QoS for Real-time Reliable Multicasting in Wireless Multi-hop Networks using a Generation-Based Network Coding", *Submitted to Computer Networks Journal, in revision process*, 2012.

6. Khaldoun Al Agha, Youghourta Benfattoum, Steven Martin and Yutaka Takahashi, [1] "GBFLY: Performance Evaluation of a Generalized Butterfly Network", *Under submission*, 2012.

Book chapters

7. Youghourta Benfattoum, Steven Martin and Khaldoun Al Agha, "Network Coding: from theory to practice", Book chapter, *Network Coding*, John Wiley & Sons, Inc., pages 1-24, 2012.

8. Youghourta Benfattoum, Steven Martin and Khaldoun Al Agha, "Network Coding: de la théorie à la pratique", Book chapter, *Codage en réseau*, Hermes, 2012.

Technical reports

9. Youghourta Benfattoum, Steven Martin, Ignacy Gawedzki and Khaldoun Al Agha, "I2-SWP: Routing algorithm with intra-flow interference consideration in ad hoc network", *Technical Report LRI-1539*, 2010.
http://www.lri.fr/~bibli/Rapports-internes/2010/RR1539.pdf.

[1] The authors order is alphabetical.

Bibliography

[1] Rudolf Ahlswede, Ning Cai, Shuo-Yen Robert Li, and Raymond W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.

[2] Ralf Koetter and Muriel Médard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, 11(5):782–795, 2003.

[3] Sachin Katti, Hariharan Rahul, Wenjun Hu, Dina Katabi, Muriel Médard, and Jon Crowcroft. Xors in the air: practical wireless network coding. In *ACM Conference of the Special Interest Group on Data Communication (SIGCOMM'06)*, pages 243–254, New York, NY, USA, 2006.

[4] Soji Omiwade, Rong Zheng, and Cunqing Hua. Practical localized network coding in wireless mesh networks. In *IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'08)*, pages 332–340, 2008.

[5] Yingda Chen, Shalinee Kishore, and Jing Tiffany Li. Wireless diversity through network coding. In *IEEE Wireless Communications and Networking Conference (WCNC'06))*, pages 1681–1686, 2006.

[6] Jay K. Sundararajan, Devavrat Shah, Muriel Médard, Michael Mitzenmacher, and Joao Barros. Network coding meets TCP. In *IEEE International Conference on Computer Communications (INFOCOM'09)*, pages 280–288, 2009.

[7] Z. Jia, R. Gupta, J. Walrand, and P. Varaiya. Bandwidth guaranteed routing for ad hoc networks with interference consideration. In *IEEE Symposium on Computers and Communications (ISCC'05)*, pages 3–9, 2005.

[8] David E. Culler, Deborah Estrin, and Mani B. Srivastava. Guest editors' introduction: Overview of sensor networks. *IEEE Computer*, 37(8):41–49, 2004.

[9] Philip A. Chou, Yunnan Wu, and Kamal Jain. Practical network coding. In *Allerton Conference Communication, Control, and Computing*, 2003.

[10] Bin Ni, Naveen Santhapuri, Zifei Zhong, and Srihari Nelakuditi. Routing with opportunistically coded exchanges in wireless mesh networks. In *IEEE Workshop on Wireless Mesh Networks (WiMesh)*, pages 157–159, 2006.

[11] Network Coding Bibliography: http://www.ifp.illinois.edu/~koetter/NWC/index.html.

[12] Shuo-Yen Robert Li, Raymond W. Yeung, and Ning Cai. Linear network coding. *IEEE Transactions on Information Theory*, 49(2):371–381, 2003.

[13] Raymond W. Yeung, Shuo Yen Robert Li, Ning Cai, and Zhen Zhang. *Network Coding Theory. Part I: Single Source*, volume 2. Now Publishers, 2006.

[14] Tracey Ho, Ralf Koetter, Muriel Médard, David R. Karger, and Michelle Effros. The benefits of coding over routing in a randomized setting. In *IEEE International Symposium on Information Theory (ISIT'03)*, page 442, 2003.

[15] Christina Fragouli, Jean Yves Le Boudec, and Jorg Widmer. Network coding: an instant primer. *ACM SIGCOMM Computer Communication Review*, 36(1):63–68, 2006.

[16] Zunnun Narmawala and Sanjay Srivastava. Survey of applications of network coding in wired and wireless networks. In *National Conference on Communications (NCC'08)*, pages 153–157, 2008.

[17] Shengli Zhang, Soung Chang Liew, and Patrick P. Lam. Physical-layer network coding. In *ACM international conference on Mobile Computing and networking (MobiCom'06)*, pages 358–365, 2006.

[18] Jilin Le, John C. S. Lui, and Dah-Ming Chiu. Dcar: Distributed coding-aware routing in wireless networks. *IEEE Transactions on Mobile Computing*, 9(4):596–608, 2010.

[19] Zheng Dong, Cheng Zhan, and Yinlong Xu. Delay aware broadcast scheduling in wireless networks using network coding. In *International Conference on Networks Security, Wireless Communications and Trusted Computing (NSWCTC'10)*, pages 214–217, 2010.

[20] Sanjit Biswas and Robert Morris. Exor: Opportunistic multi-hop routing for wireless networks. In *ACM Conference of the Special Interest Group on Data Communication (SIGCOMM'05)*, pages 69–74, 2005.

[21] Szymon Chachulski, Michael Jennings, Sachin Katti, and Dina Katabi. Trading structure for randomness in wireless opportunistic routing. In *ACM Conference of the Special Interest Group on Data Communication (SIGCOMM'07)*, pages 169–180, 2007.

[22] Chuan Qin, Yi Xian, Chase Gray, Naveen Santhapuri, and Srihari Nelakuditi. I2mix: Integration of intra-flow and inter-flow wireless network coding. In *IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'08)*, pages 1–6, 2008.

[23] Christos Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *IEEE International Conference on Computer Communications (INFOCOM'05)*, volume 4, pages 2235–2245, 2005.

[24] Danilo Silva and Frank R. Kschischang. Rank-metric codes for priority encoding transmission with network coding. In *Canadian Workshop On Information Theory (CWIT'07)*, pages 81–84, 2007.

[25] Jay Kumar Sundararajan, Muriel Médard Devavrat Shah, and Parastoo Sadeghi. Feedback-based online network coding. In *Patent Publication Number: US 2010/0046371 A1*, 2009.

[26] Ernest Mukhamedovich Gabidulin. Theory of codes with maximum rank distance. *Problems of Information Transmission*, 21:1–12, 1985.

[27] Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16:575–577, 1973.

[28] Yaling Yang and R. Kravets. Contention-aware admission control for ad hoc networks. *IEEE Transactions on Mobile Computing*, 4(4):363–377, 2005.

[29] Kimaya Sanzgiri, Ian D. Chakeres, and Elizabeth M. Belding-Royer. Determining intra-flow contention along multihop paths in wireless networks. In *IEEE International Conference on Broadband Networks (BROADNETS'04)*, pages 611–620, Washington, DC, USA, 2004.

[30] Simon Odou, Steven Martin, and Khaldoun Al Agha. Idle channel time estimation in multi-hop wireless networks. *IEEE International Conference on Communications (ICC'09)*, pages 1–5, 2009.

[31] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[32] Zheng Wang and Jon Crowcroft. Quality of service routing for supporting multimedia applications. *IEEE Journal on Selected Areas in Communications*, 14:1228–1234, 1996.

[33] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986.

[34] Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

[35] L. R. Ford, Jr. and D. R. Fulkerson. Flows in networks. *Princeton University Press, Princeton, NJ*, 1962.

[36] Rajarshi Gupta, John Musacchio, and Jean Walrand. Sufficient rate constraints for QoS flows in ad-hoc networks. *Ad Hoc Networks*, 5:429–443, 2007.

[37] GNU Linear Programming Kit: http://www.gnu.org/software/glpk/.

[38] Javier Gomez and Andrew T. Campbell. Variable-range transmission power control in wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 6(1):87–99, 2007.

[39] Piyush Gupta and P. R. Kumar. Critical power for asymptotic connectivity. In *IEEE Conference on Decision and Control*, volume 1, pages 1106–1110, 1998.

[40] Ram Ramanathan Regina and Regina Rosales-hain. Topology control of multihop wireless networks using transmit power adjustment. In *IEEE International Conference on Computer Communications (INFOCOM'00)*, volume 2, pages 404–413, 2000.

[41] Ning Li and Jennifer C. Hou. Localized fault-tolerant topology control in wireless ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, 17(4):307–320, 2006.

[42] Qunfeng Dong and Yigal Bejerano. Building robust nomadic wireless mesh networks using directional antennas. In *IEEE International Conference on Computer Communications (INFOCOM'08)*, pages 1624–1632, 2008.

[43] Martin Burkhart, Pascal von Rickenbach, Rogert Wattenhofer, and Aaron Zollinger. Does topology control reduce interference? In *ACM Mobile ad hoc networking and computing (MobiHoc'04)*, pages 9–19, 2004.

[44] Mohammed D. Halloush and Hayder Radha. Network coding with multi-generation mixing: Analysis and applications for video communication. In *Conference on Information Sciences and Systems (CISS'08)*, pages 515–520, 2008.

[45] Jean-Pierre Thibault, Shahram Yousefi, and Wai-Yip Chan. Throughput performance of generation-based network coding. In *10th Canadian Workshop on Information Theory (CWIT'07)*, pages 89–92, 2007.

[46] Kui Xu, Fan Zhang, Chunming Rong, Bin Dai, and Benxiong Huang. Block size estimation for time-sensitive applications under wireless network coding. In *International Conference on Future Generation Communication and Networking (FGCN'08)*, pages 321–324, 2008.

[47] Aleksi Penttinen. Efficient multicast tree algorithm for ad hoc networks. In *IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'04)*, pages 519–521, 2004.

[48] Majid Ghaderi, Don Towsley, and Jim Kurose. Network coding performance for reliable multicast. In *IEEE Military Communications Conference (MILCOM'07)*, pages 1–7, Orlando, FL, USA, 2007.

[49] Majid Ghaderi, Don Towsley, and Jim Kurose. Reliability gain of network coding in lossy wireless networks. In *IEEE International Conference on Computer Communications (INFOCOM'08)*, pages 2171–2179, Phoenix, AZ, 2008.

[50] Jalaluddin Qureshi, Chuan Heng Foh, and Jianfei Cai. An efficient network coding based retransmission algorithm for wireless multicasts. *ACM Computing Research Repository (CoRR)*, abs/1008.1842, 2010.

[51] Eric Rozner, Anand Padmanabha Iyer, Yogita Mehta, Lili Qiu, and Mansoor Jafry. Er: efficient retransmission scheme for wireless lans. In *Conference on emerging Networking EXperiments and Technologies (CoNEXT'07)*, page 8, 2007.

[52] Pedro R. S. Lopes, Yuri C. B. Silva, and Francisco Rodrigo P. Cavalcanti. Efficient wireless multicast retransmission techniques based on multiple coded packets. In *IEEE Vehicular Technology Conference (VTC'10-Fall)*, pages 1–5, 2010.

[53] Pablo Brenner. A Technical Tutorial on the IEEE 802.11 Protocol. *BreezeCOM*, 1997.

[54] Cisco Routing Proximity:
http://data.proidea.org.pl/plnog/2edycja/materialy/prezentacje/sprevidi.pdf.

[55] Cisco IOS Optimized Edge Routing Overview:
http://www.cisco.com/en/US/docs/ios/12_4t/oer/configuration/guide/h_oerovr.html.

[56] Jeffrey Erman, Alexandre Gerber, Mohammad T. Hajiaghayi, Dan Pei, and Oliver Spatscheck. Network-aware forward caching. In *ACM International conference on World wide web (WWW'09)*, pages 291–300, 2009.

[57] Wenlan Feng, Liang Zhang, Beihong Jin, and Zhihua Fan. Context-aware caching for wireless internet applications. In *IEEE International Conference on e-Business Engineering (ICEBE 2006)*, pages 450–455, 2006.

[58] Rudiger Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *IEEE Conference on Peer-to-Peer Computing*, 2002.

[59] A P2P file-sharing system: http://www.ics.uci.edu/~bhore/papers/Hollyshare.pdf.

[60] Alex Bordetsky, Susan G. Hutchins, William G. Kemple, and Eugene Bourakov. Network aware tactical collaborative environments. In *International Command and Control Research and Technology Symposium (ICCRTS'04)*, 2004.

[61] Sue Plumley. *Home Networking Bible*. Wiley Publishing, Inc., 2004.

[62] Vikram Chandrasekhar, Jeffrey G. Andrews, and Alan Gatherer. Femtocell networks: A survey. *IEEE Communication Magazine*, Vol. 46, Issue 9, 2008.

[63] http://www.femtoforum.org/femto/.

[64] Mea Wang and Baochun Li. R2: Random push with random network coding in live peer-to-peer streaming. *IEEE Journal on Selected Areas in Communications*, 25(9):1655–1666, 2007.

[65] Shahriar E.Tajbakhsh, Mahsa Orang, Maryam H. Sohi, and Ali Movaghar. A queuing model of opportunistic network coding in wireless medium. In *Conference on Signals, Systems and Computers*, pages 1431–1435, 2008.

[66] Borislava Gajic, Janne Riihijarvi, and Petri Mahonen. Performance evaluation of network coding in middle-sized networks. In *Wireless Communications and Networking Conference (WCNC'10)*, pages 1–5, 2010.

[67] Parimal Parag and Jean-Francois Chamberland. Queueing analysis of a butterfly network. In *IEEE International Symposium on Information Theory (ISIT'08)*, pages 672–676, 2008.

[68] P. Parag and J.F. Chamberland. Queueing analysis of a butterfly network for comparing network coding to classical routing. *IEEE Transactions on Information Theory*, 56(4):1890–1908, 2010.

[69] B. Adamson, C. Bormann, M. Handley, and J. Macker. Negative-acknowledgment (NACK)-Oriented Reliable Multicast (NORM) Protocol. *RFC 3940, 2004*, 2004.

[70] B. Adamson, C. Bormann, M. Handley, and J. Macker. NACK-Oriented Reliable Multicast (NORM) Transport Protocol. *RFC 5740, 2009*, 2009.