Network coding application for online games platformes

Marwa Dammak

THÈSE DE DOCTORAT

En co-tutelle entre
Université Paris Seine
ET Université de Sfax
École Nationale d'Ingénieurs de Sfax

En vue de l'obtention du
DOCTORAT
Dans la discipline
Génie Ingénierie des Systèmes Informatiques (EDST)
Sciences et Technologies de l'Information et de la Communication (E2MP)
par
Marwa Dammak

Network Coding Application For Online Games Platforms

Préparée à
ETIS, UMR 8051, Université Paris Seine, Université Cergy-Pontoise, ENSEA, CNRS, France
ET LETI-Ecole nationale d'ingénieurs de Sfax, Université de Sfax, Tunisie
Soutenue le 20 Novembre 2018
Devant le jury composé de :

Examinateurs	PROF.	Lamia CHAARI	ISIMS, TUNISIA
	PROF.	Nadia BOUKHATEM	Télécom ParisTech, France
Rapporteurs	PROF.	Aref MEDDEB	ENISO, Tunisie
	PROF.	Michel KIEFFER	Centrale Supélec, France
Invité	CR	Frédéric GIROIRE	CNRS, France
Directrices de thèse	PROF.	Noura SELLAMI	ENIS, Tunisie
	PROF.	Inbar FIJALKOW	ETIS-ENSEA, France
Co-encadrants	MA	Yassine BOUJELBEN	EnetCom, Tunisie
	MCF	Iryna ANDRIYANOVA	ETIS-ENSEA, France

Abstract

Massively multiplayer online games provide a large part of the global internet traffic. The traffic is typically encapsulated in TCP segments of small size information (the so called payload) resulting in a high volume of headers transmitted through the network. This implies the following: 1) the packets' size is too small for efficient routing and; 2) the bandwidth required by the server seems to be increasing. Therefore, it is necessary to find more efficient transmission and routing techniques to reduce the traffic volume and to increase the network efficiency in order to support the growing number of players. On the other hand, Quality of Experience (QoE) in the context of online games is strongly influenced by consistency. This consistency is influenced by the quality of service (QoS) offered by the network, mainly in terms of delay, jitter and order of the packets. As a result, a solution that enhances these parameters will help in satisfying more players and subsequently improving QoE.

The goal of this thesis is to propose solutions to enhance the QoE of online games by increasing the consistency of view, reducing the delay and increasing the efficiency of the network. For consistency, we propose a cyclic network topology. This ordered topology facilitates the implementation of transmission management and control procedures to impose a causal order between all players. As for the delay and the efficiency problem, some have proposed the application of Tunneling, Compression and Multiplexing (TCM) technique. However, the transmitted packets with TCM are larger than the original ones, which requires more delay to transmit them and increases the risk of congestion. We propose the use of the Network Coding technique (NC) which makes it possible to increase the bit rate of the network under certain topological and routing conditions. This technique allows intermediate nodes to encode the data they receive rather than perform a simple "store and forward" function. It can reduce the traffic load, reduce the delay

and increase the network efficiency.

In this thesis, we propose some modifications to enhance the TCM technique and evaluate its performance. Besides, we propose to add the players partition approach and change the topology from a tree to a forest. Afterwards, we investigate the cyclic topology. We design an optimized routing protocol over a cycle topology based on the network coding technique and evaluate its performance in terms of delay and order. The results show that by using NC coding, one can gain up to 14% of latency over an optimized routing protocol for the cycle topology without the use of network coding. Afterwards, we propose a practical implementation scenario of this solution over a device-to-device (D2D) infrastructure. We finally validate the theoretical limits of delay using network simulations and discuss a number of practical constraints.

Key-words: Online games, network coding, cycles, consistency, traffic load, QoE, QoS.

Résumé

Le jeu en ligne massivement multijoueurs fournit une grande partie du trafic Internet global. Le trafic est généralement composé d'une petite partie de données utiles encapsulé dans des segments TCP , entrainant un volume élevé d'entêtes transmis via le réseau. Cela implique que: 1) la taille des paquets est trop petite pour un routage efficace et; 2) la bande passante requise par le serveur augmente. Par conséquent, il est nécessaire de trouver des techniques de transmission et de routage plus efficaces afin de réduire le volume de trafic et augmenter l'efficacité du réseau permettant de prendre en charge le nombre croissant des joueurs. D'autre part, la qualité d'expérience (QoE) dans le contexte des jeux en ligne est fortement liée à la consistance. Cette consistance est influencée par la qualité de service (QoS) offerte par le réseau, principalement en termes de délai, de gigue et d'ordre de paquets. Par conséquent, une solution qui améliore ces paramètres aidera à satisfaire davantage de joueurs et à améliorer la qualité de service.

L'objectif de cette thèse est de proposer des solutions pour améliorer la qualité d'expérience des jeux en ligne en augmentant la consistance de vue, en réduisant les délais et en augmentant l'efficacité du réseau. Pour le problème de consistance, nous proposons une topologie de réseau cyclique. Cette topologie ordonnée facilite la mise en place des procédures de gestion et de contrôle de la transmission pour imposer un ordre causal entre tous les joueurs. Concernant le délai et l'efficacité du réseau, certains ont proposé l'application de la technique du tunnel, compression et multiplexage (TCM). Cependant, les paquets transmis avec TCM sont plus grands que ceux d'origine, ce qui nécessite plus de temps pour les transmettre et augmente le risque de saturation au niveau des files d'attente. Nous proposons l'utilisation de la technique de codage réseau (NC) qui permet d'augmenter le débit dans certaines conditions de topologies et de routage. Cette technique permet aux nœuds intermédiaires d'encoder les paquets qu'ils reçoivent

plutôt que d'effectuer une simple fonction de stockage et de transfert. Cela peut réduire la charge de trafic, réduire les délais et augmenter l'efficacité du réseau.

Dans cette thèse, nous proposons des modifications pour améliorer la technique TCM et nous évaluons ses performances. De plus, nous proposons d'ajouter le concept de partition des joueurs et de changer la topologie d'un arbre à une forêt. Nous étudions ensuite la topologie cyclique. Nous concevons un protocole de routage optimisé sur une topologie en cycle basée sur la technique de codage réseau. Puis, nous évaluons ses performances en termes de délai, de charge et d'ordre. Les résultats montrent que l'utilisation du codage NC permet de réduire la charge et le nombre de paquets transmis, garantir un ordre de paquet par période et de réduire le délai. En effet, on peut gagner jusqu'à 14% de latence avec notre protocole par rapport à un protocole de routage optimisé sans codage réseau. Par la suite, nous proposons un scénario de mise en pratique de cette solution sur une infrastructure Device-to-Device. Nous validons les limites théoriques du délai en utilisant des simulations réseau et nous discutons ensuite des contraintes pratiques qui s'imposent lors de l'implémentation dans un réseau réel. Finalement, nous proposons des solutions pour ces contraintes.

Mot clés: Jeux en ligne, codage réseau, cycles, consistance, charge du traffic, QoE, QoS.

الخلاصة

يوفر تطبيق الألعاب متعدد اللاعبين عبر الإنترنت الكثير من حركة الإنترنت الشاملة. تتألف الحركة عادة من جزء صغير من الحمولة المغلفة التي يتم تغليفها في أجزاء TCP مما ينتج عنه حجم كبير من الرؤوس المرسلة عبر الشبكة. هذا يعني 1) حجم الحزمة صغير جدًا للتوجيه الفعال و ؛2) زيادة النطاق الترددي المطلوب من قبل الخادم. لذلك ، من الضروري إيجاد تقنيات نقل وتوجيه أكثر كفاءة لتقليل حجم حركة المرور وزيادة كفاءة الشبكة لدعم العدد المتزايد من اللاعبين .من ناحية أخرى ، ترتبط بجودة التجربة (QoE) في سياق الألعاب عبر الإنترنت ارتباطًا وثيقًا بالاتساق. ويتأثر هذا الاتساق بجودة الخدمة (QoS) التي توفرها الشبكة، ولا سيما .من حيث التأخير والتردد وترتيب الرزم ,

الهدف من هذه الرسالة هو اقتراح الحلول لتحسين نوعية تجربة للألعاب على الإنترنت عن طريق زيادة الاتساق في وجهات النظر، وتقليل الوقت وزيادة كفاءة الشبكة. بالنسبة لمشكلة الاتساق ، نقترح طوبولوجيا شبكة دورية. هذا الهيكلية النظامية تسهل تنفيذ إجراءات إدارة الإرسال والتحكم لفرض نظام السببية بين جميع اللاعبين.فيما يتعلق بتأخير الشبكة وكفاءتها ،اقترح البعض تطبيق تقنية الأنفاق والضغط وتعدد الإرسال(TCM) ومع ذلك، فإن الحزم المرسلة تصبح أكبر من الأصلي، الأمر الذي يتطلب مزيدا من الوقت ,

لنقل .نقترح استخدام تقنية ترميز الشبكة (NC) التي تجعل من الممكن زيادة التدفق في ظل ظروف معينة من الطبولوجيا والتوجيه. تسمح هذه التقنية للعُقد المتوسطة بتشفير الحزم التي تتلقاها بدلاً من إجراء وظيفة تخزين ونقل بسيطة. هذا يمكن من زيادة كفاءة الشبكة ، والحد من التأخير. في هذه الأطروحة ، نقترح تعديلات لتحسين تقنية TCM و نقوم بتقييم أدائها. بالإضافة إلى ذلك، نقترح إضافة مفهوم تقسيم اللاعب وتغيير طوبولوجيا شجرة إلى غابة. ثم ندرس الطوبولوجيا الدورية. نقوم بتصميم بروتوكول توجيه مُحسَّن على طبولوجيا الدورة استناداً إلى تقنية تشفير الشبكة. ثم ، نقوم بتقييم أدائها من حيث التأخير والحمل والنظام. تظهر النتائج أن استخدام تشفير NC يقلل من الحمل وعدد الحزم المنقولة ، ويضمن ترتيب الحزم في كل فترة ويقلل من التأخير. في الواقع ، يمكننا الحصول على ما يصل إلى 14 ٪ من زمن الانتقال مع بروتوكولنا مقارنة ببروتوكول التوجيه الأمثل دون تشفير الشبكة. بعد ذلك ، نقترح سيناريو التنفيذ العملي لهذا الحل على بنية تحتية من جهاز إلى جهاز. نحن نتحقق من الحدود النظرية للتأخير باستخدام محاكاة الشبكة ثم نناقش المشاكل العملية المطلوبة عند التنفيذ في شبكة حقيقية. وأخيرًا ، نقترح حلولًا لهذه المشاكل.

المفاتيح: الألعاب عبر الإنترنت ،ترميز الشبكة ،الدورات، الاتساق ،تحميل الحركة, QoS ، QoE

Acknowledgements

At the end of writing my thesis dissertation, I am convinced that this thesis research could never have been achieved without the support of a large number of people.

First, I would like to thank my thesis directors Mrs Noura SELLAMI Professor at ENIS and Mrs Inbar FIJALKOW Professor of Universities at ENSEA, as well as my co-supervisors Mr Yassine BOUJELBEN Assitant Professor at Enetcom and Mrs. Iryna Andriyanova Assistant Professor at ENSEA for the trust that they have given me by agreeing to lead this work with enthusiasm and pedagogy. I wish to express my gratitude for their excellent coaching and wise advice that allowed me to carry out this thesis work.

My thanks also go to the members of the jury for accepting to honor me with their presence today.

I would also like to thank my fellow doctoral students for their support and cooperation. In addition, I would like to express my gratitude to the staff of both laboratories ETIS and LETI. I would like to thank my friends, too, for always being by my side when I needed them. Last but not the least, I would like to thank my parents, my brother and my sister for supporting me throughout writing this thesis dissertation and in every step in my life in general.

This accomplishment would not have been possible without you all. Thank you.

Marwa DAMMAK

Dedication

This thesis is dedicated to my best friends, Tesnim, Islem, Latifa, Asma, Imen and Omnia who have always been a constant source of support, help and encouragement for me, and also to my brother Mohammed, my sister Abir, my aunts and uncles; I am truly grateful for having them in my life. This work is also dedicated to my lovely parents, Zohra and Abdelwaheb, who have always loved me unconditionally and who have constantly been good examples for me.

Contents

List of Figures

List of Tables

Overview

Internet traffic is remarkably increasing, especially with the evolution of Internet services and the ease of connecting to Internet via smartphones and tablets. This traffic carries different types of flows for various applications. Some of these applications, such as video streaming and multimedia file transfer, used to be the main contributors to this traffic. Hence, the majority of works focused on those types of applications in order to study the network's performance, the quality of the received signal, the throughput, the delay, etc. However, nowadays, new applications are getting more popular and contribute a lot to the global Internet traffic. One of the growing applications is the Massively Multiplayer Online Gaming, a real-time group communication application. The growing popularity of this application results in an increasing share on the global Internet traffic. The traffic share of online games in the global Internet traffic is growing. This traffic is composed of a large number of small-sized packets which carry a little payload volume compared to the overhead added by the transport protocols. This fact decreases the efficiency of the network. As a real-time application, online gaming presents special constraints, which are basically the delay, the jitter and the throughput. This is translated into the impact of the quality of service (QoS) provided by the underlying network on the quality of experience (QoE) perceived by the users. Depending on the nature of the game, users may be more or less tolerant to some of these parameters. The QoE of online gamers is strongly influenced by the consistency of the game view which represents the synchronization and the coherence of the game's state perceived by the players. The consistency is influenced by three main parameters: transmission latency, the jitter introduced by the network and the order of packets. Thus, online gaming is characterized by the following problems: 1) a low payload with a large overhead resulting in a huge degradation of the network's efficiency and; 2) a picky end user that requires a certain level of quality of experience

in order to continue the game.

Naturally, a solution that increases the efficiency of the network by reducing the overhead and by increasing the throughput will improve the quality of service of the network and will help it to support the growing number of players. Besides, a routing scheme that can provide a compromise between these constraints and the consistency of the game by ordering the game packets will offer an acceptable QoE to online players. As for the network efficiency, the network coding technique can reduce the overhead of online games traffic and increase the throughput. This technique has shown efficiency in both wired and wireless networks, and over different network topologies such as the tree topology, the star topology or the cycle topology. Besides, it can be applied not only to the multicast traffic flow case, but to multiple unicast traffic flows as well. In some schemes, NC can also help in reducing the transmission delay. As for the consistency issue, we propose to use a cycle topology. With cycles, ordering mechanisms can be easily implemented thanks to some management keys that can be added, such as a token. The cycle topology has recently gained much interest for different applications. However, the use of the cycle topology will result in a larger delay which can be unacceptable by certain online games. In order to exploit cycles and make the cycle typology a practical alternative for online games, one should reduce the delay of transmission. In this thesis, we have design a NC-based routing protocol for online gaming applications over cycles with a careful choice of the node transmission scheduling. This protocol reduces the transmission delay over the cycle, reduces the traffic load and increases the throughput while maintaining a certain order of packets. After designing the protocol, we evaluate its performance by calculating its theoretical bounds and validating them via simulations. The proposed protocol reduces the load over the cycle and decreases the latency by up to 14% compared to a simple routing protocol with no network coding. We also propose a practical scenario implementation over a device-to-device infrastructure and discuss some constraints that can be introduced by real networks and propose solutions.

The remainder of this dissertation is organized as follows. In the first part, we introduce the characteristics and problems of online games. Then we give an overview of the related works dealing with these problems. In the second part, we propose some enhancements to an existing solution and evaluate its performance for an online game using simulations. The last part is devoted to the description of our proposed solution, an illustration of a possible implementation example, the discussion of some practical constraints

and the proposition of some solutions. Finally, we conclude with a conclusion and an overview of some perspectives and possible extensions of our work.

Part I

State of the art

Chapter 1

Massively multiplayer online games

Contents

1.1 Introduction

In this chapter, we will present the online games applications. We will introduce the game types, the architectures adopted for online games, the specificities of traffic generated by online games as well as the performance metrics usually considered for these applications.

1.2 Online games

In this section we will present the specifications of the online gaming application in terms of protocols, traffic characteristics, quality of experience and its metrics. Finally, we will point out the problems that the actual online games face.

1.2.1 Online game categories

Massively Multiplayer Online Games, known also as the MMOG, are attracting people from all over the world and with different game preferences. Nowadays, we talk about tens of millions of players per month for some online games. As a result, the Internet traffic consumption is increasing, as shown in Figure 1.1, with a compound annual growth rate (CAGR) of 62% between 2016 and 2021 according to cisco[1]. The categories of online games are: MMORPG (role play games), MMOFPS (first person shooting games), MMORTS (real time strategy games), MMOR (racing games), MMOSG (social games), MMORG (rythm games) and many other types. In the following, we will describe the three main categories of multiplayer online games which are responsible for most of the gaming traffic.

MMOFPS: MMOFPS means Massively Multiplayer Online First Person Shooting games; one of the most well-known examples of this type is Counter Strike. It is characterized by its fast pace. For MMOFPS, latency is crucial and the duration of the game session is not long.

MMORPG: MMORPG stands for Massively Multiplayer Online Role Playing games. This category includes the majority of the online games.

[1]https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html

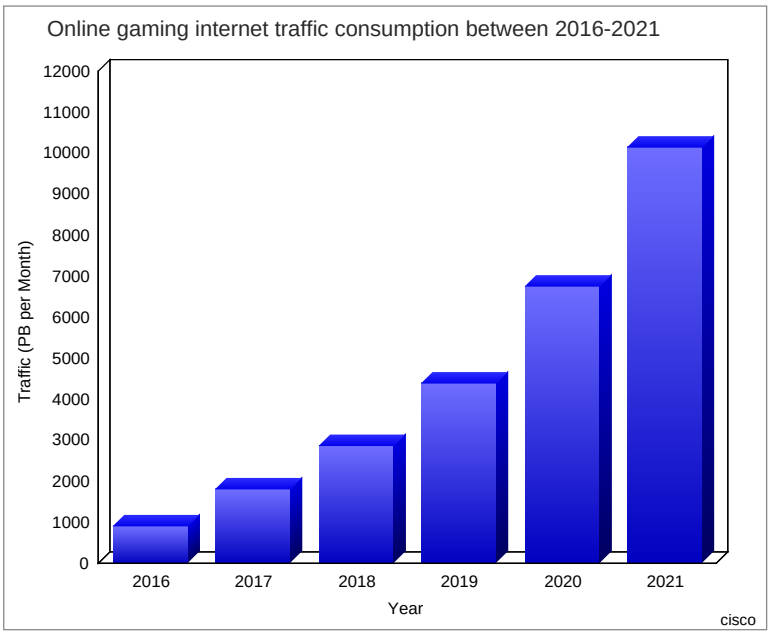


Figure 1.1: The online gaming Internet consumption according to Cisco's statistics in 2017

One of the most famous ones is the "League Of Legends". It has a remarkably growing number of active players which rose from 11.5 billions/month in 2011 to more than 100 billions/month in 2016. These adventurous games can create interactions and possibly social relationships between players [4] which make them less restrictive on delay compared to MMOFPS.

MMORTS: MMORTS refers to Massively Multiplayer Online Real Time Strategy games. This type of games is slower than the other two types, takes a longer time to be concluded and its players are more patient as they take a long time to make their moves. Among the famous examples of MMORTS games, let us cite "starcraft".

As a real-time application, each one of these types has its own constraints in terms of pace, maximum tolerable latency and bandwidth requirements. A comparison of the specificities of each of these types is illustrated in Table 1.1 [5, 6].

Game type	Game Pace	Maximum latency	Bandwidth requirements
MMOFPS	fast	150-250 ms	40 kbps
MMORPG	slow	500-1000 ms	7kbps
MMORTS	very slow	< 1.5 s	N/A

Table 1.1: Comparison between the main types of MMO games.

1.2.2 Game network architecture

When it comes to massively multiplayer online games, two possible choices of architecture are present: the Client/Server and the Peer-To-Peer (P2P) architectures. Each of them has its advantages and disadvantages. We will describe these points in more details in the following.

1.2.2.1 Client/Server

The client/server architecture is the favourite architecture and the one most commonly used by games developers [7]. Indeed, almost all the multiplayer online games are based on this model where all the players are connected to

a central server which is responsible for calculating the states' updates based on the actions received from different players. This architecture provides numerous advantages but also presents some disadvantages.

1.2.2.2 Peer-To-Peer

Peer-To-Peer is the architecture model in which all the nodes are connected to one another. This architecture is widely used for different applications such as file sharing, multimedia communications owing to the advantages that it can afford. P2P implies that the traffic circulates between these nodes (peers) in order to update their state. Each peer, in this case, is responsible for calculating his version of the game state based on the information received from the other peers.

Even though P2P architecture is not very popular when it comes to online games, some of these games are designed based on this architecture such as MiMaz [8], Empires [9], as well as one of the famous MMORTS games: the Starcraft series [10].

1.2.2.3 Discussion

The client/server architecture is adopted for the majority of the online games in order to solve the capacity problem. For instance, the MMOG server can be represented by a group of machines with dedicated responsibilities [11, 12]. The choice of the criteria used to divide the responsibilities can vary from one game to another. It can be based on functionalities such as 'players logging in, chatting and patching', or based on the players' interactions. As a result, the players can be grouped depending on their local environment on the game map or the so-called *area of interest* (AOI) [13] (This concept will be discussed in more details later in section 3.3.1). Another solution to reduce the bandwidth required for the server is the *delta encoding* method [13]. This method is based on the fact that an objects state changes just a little from one update to another. Thus, the server can calculate the difference between the last and the new update and send this difference to the clients. One of the games that uses these approaches is Quake. However, despite using these solutions, the server is always exposed to the problem of overload and computation heaviness. In this context, the concept of cloud gaming can solve the problem. Cloud gaming is a relatively new concept that exploits

the ease offered by the cloud computing platform in order to lighten the load on players' terminals. The objective of this concept is to open the doors to new potential players who are using light terminals with limited performance such as smart phones or tablets. In fact, these terminals can't provide the minimum hardware performance requirements of traditional online games. All the computational operations are performed by the cloud gaming system as shown in Figure 1.2. However, this solution has its limits and challenges. In fact, the video coding, transmission and decoding operations are costly in terms of delay [14, 15, 16, 17]. As the client/server and the P2P architectures present some advantages as well as some disadvantages, some researchers proposed new architectures called hybrid architectures, where both of these architectures are used [18, 19].

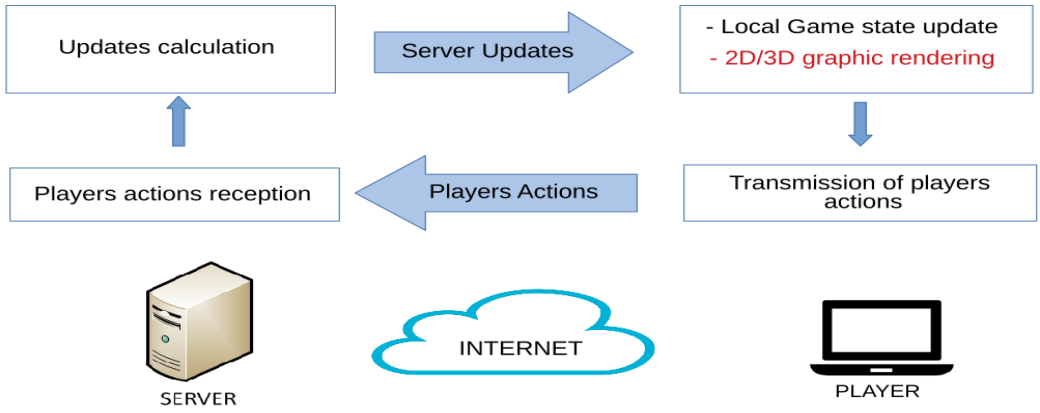
1.3 Network aspects of online games

In this section, we will focus on the traffic of online games. Let us only consider the client/server architecture as the most commonly used one for online games. A presentation of a typical online game infrastructure using the client/server architecture is illustrated in Figure 1.3.

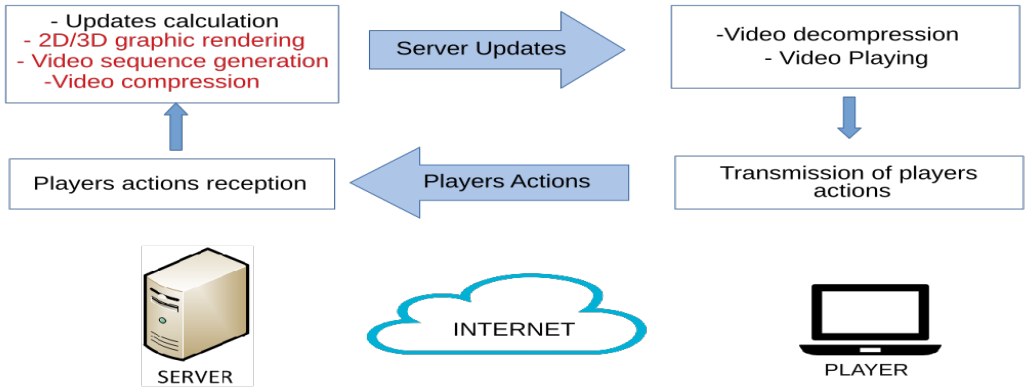
1.3.1 Traffic characteristics

Online gaming traffic consists of two different types of traffic: client-to-server and server-to-client traffic. In the following, we will discuss the difference between these two traffics and their specific features.

- Client-to-server packets
  The clients of online gaming need to establish a connection to the game session through the game server using a simple unicast transmission. Afterwards, all their traffic will be forwarded to that server. Therefore, in a game session of n players, n unicast traffic flows are sent the server: one per each player. Mainly, the traffic generated by the clients represents their actions in the game. These actions will be treated by the server in order to generate the new game state update. Then the server sends it to the clients so that they all apply the changes and synchronize their game view. Besides their actions, other packets such as TCP acknowledgments and multimedia files for audio chatting can be sent from the clients to the server.



(a) Traditional online gaming model



(b) Cloud gaming model

Figure 1.2: Comparison between traditional online gaming and cloud gaming models

- Server-to-client packets
  The server is the principal component of the conventional architecture of online gaming. It is responsible for a multitude of tasks including authentication, clients' accounts' management, handling the requests of joining or leaving and mainly managing the game state modifications and maintaining a synchronous game state for all the players. Hence,

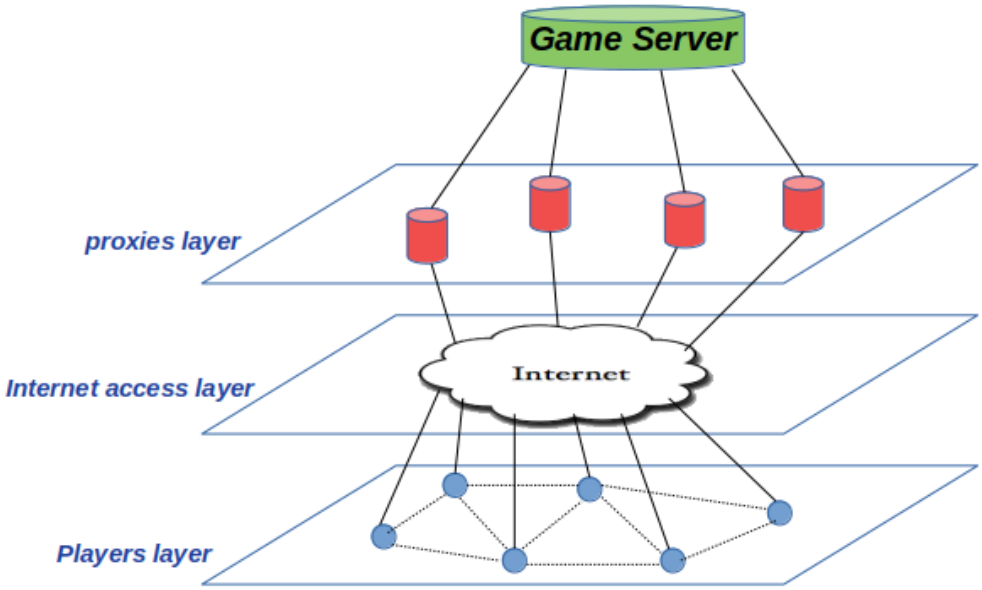


Figure 1.3: Typical online game infrastructure.

the majority of the messages generated by the server represent the game updates sent to the clients. These messages are multicasted to all the concerned players in a periodic way. Here, the multicast is not necessarily performed in the network layer. It can be performed using application layer services [20, 21].

- Packet size
  Client packets and server packets differ in terms of size. In fact, the size of the server packets is generally larger than the size of the client packets [22]. This is due to the correlation between the server's update and the number of players of a game session. While the players send only their own actions, the server should calculate the changes produced by all the players. For instance, the size of the server packets for the QuackIII game can be calculated depending on the number of players as follows [23]:

$$S_{server} = \begin{cases} 81.9 + 13 * (n-2) \text{ if } n > 2, \\ 81.9 + 13 \text{ else} \end{cases} \tag{1.1}$$

  where S_{server} is the size of the server's packet in bytes and n is the

number of players. However, compared to other Internet applications' traffic, both of clients' and server packets' sizes are considerably small. For the ShenZhou Online game, 98% of the client packets have a size smaller than 72 bytes and the average size of the server packets is 154 bytes [6]. In fact, the online gaming traffic consists generally of a huge number of small-size packets. These packets have a very particular form. The length of the payload is variable but it is usually much smaller than the length of the header. Moreover, as the most popular transport protocol for gaming is TCP, which generates a relatively large header, the overall header size at the transport layer represents up to 73% of a game packet [6]. This results in an extremely inefficient transmission protocol for online gaming and a sharp decrease in the quality of service (QoS) when the number of players increases. This degradation of the QoS of the underlying network represents the main reason for the player to decide to quit the game [24]. This leads us to the following section where we discuss the quality of experience of the online games.

- Periodicity and inter-packet interval
  Studies on online games' traffic showed that the packets are generally sent in a periodic way either from the client or the server side [6, 7]. In fact, considering the server-to-client flow, the periodicity of the server packets is the result of its round-based behavior. The traffic is hence sent periodically to all the players with a server period that depends on the concerned game [25] [6]. Let's note this period as τ_{sc}. On the other hand, the studies on the clients' traffic also showed a periodic pattern of packets' arrival to the server. The periodicity of the clients' packets is mainly due to the automatically-generated commands [25]. Actually, the player can switch to an "auto-movement" or an "auto-attack" mode which helps the avatar to automatically move to a specific destination in the game map or automatically shoot or hit another avatar for several consecutive times with one click, for example. The periodicity of the clients' traffic could be canceled if the clients' timers are not synchronized. However, the clients' timers are synchronized (though not intentionally) because their initialization is based on the reception of the server packets, which are sent simultaneously to all clients as it has a round-based behavior as we mentioned. Thus, although the game developers didn't intend to have synchronous clients, the design choice

of the server ensures this synchronization. We will note the period of traffic from the clients to the server as τ_{cs}.

1.3.2 Quality of Experience

We will present the sensitivity of online games and their Quality of service (QoS) and quality of experience (QoE). One should note that, when the QoS focuses on the parts between the end users but not on the end users themselves, the QoE is in fact about the experience and the satisfaction of the end user. In the following, we will describe the QoE of online games, its factors and the relation between this QoE and the network QoS.

1.3.2.1 QoE parameters of online games

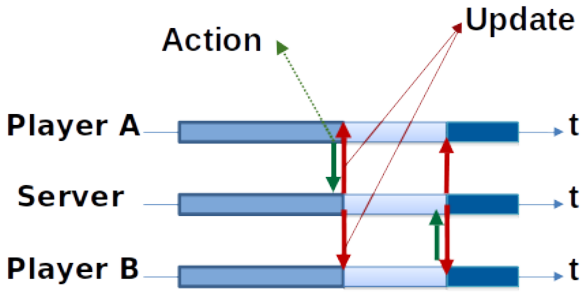
Several studies have been recently done in order to elaborate objective models of the quality of experience (QoE) of players. For instance, the authors of [26] have identified three key parameters to measure the QoE which are responsiveness, precision and fairness.

- Responsiveness: it is related to the player's perception of the overall process of the game. It is strongly connected to the time used by the system to undertake an event and to update the player's screen.
- Precision: it represents the player's perception of the results of his actions. It is related to the closeness between the player's action and the system's response.
- Fairness: refers to the degree of difference between all the game environments perceived by the players.

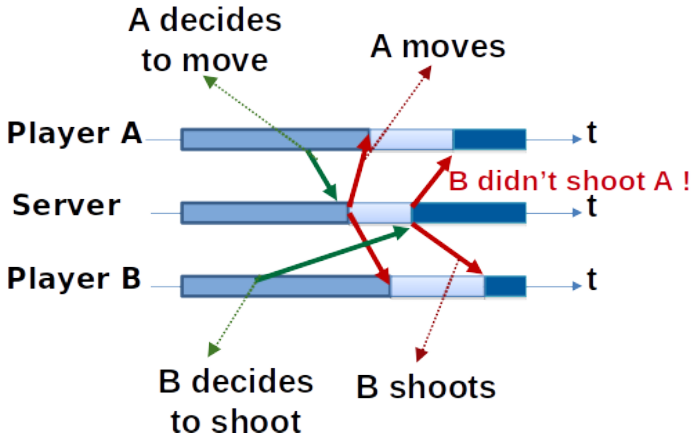
All these parameters are influenced by the consistency of the game which will be detailed further in the following.

1.3.2.2 Consistency

As we move from single-player games to multiplayer games where each interaction between the players has an influence on the game result, the consistency issue appears. Recently, consistency has also been identified as a critical parameter of the QoE [26, 27]. It reflects the coherence and synchronization of the video sequences, observed by different players. A loss of



(a) Consistency case.



(b) Inconsistency case

Figure 1.4: Example of consistent and inconsistent actions in a game.

consistency makes the game meaningless due to the loss of causality between inter-dependent events.

As an example, the upper picture in Figure 1.4 illustrates the consistent case, where each action is immediately updated by all players and all of them have the same view (game environment). The lower picture in Figure 1.4 illustrates the case where a change in the network transmission delay entails serious causality problems. In this scenario, Player B shoots at Player A at time t_1, and Player A moves at time $t_2 = t_1 + \Delta$. However, the packet from Player B had larger delay than the packet from Player A. Thus, the action of A reaches the central server faster that the action of B. As a result, even though the action of B was taken before the action of A, Player B couldn't hit Player A because, for the central server, he has already moved elsewhere.

The consistency can be decomposed into three main categories [26]:

- Causal consistency: This consistency is derived from the event-based model of online games. It assumes that, at any instant of the game, the game state should not violate the causal-effect, i.e the happened-before relationship [28], between the different players' actions.

- Time-Space consistency: This category is based on the fact that the judgments and the actions of the players on the game objects are based not only on their relative positions, but also on the duration of this situation. The time-space consistency metrics are defined in [29].

- View consistency: With a perfect network performance, all players will perceive the same game state at the same time. All actions will be performed in time and in order. However, the delay introduced by the network connection between the players and the server leads to different game views for different players. The View consistency is considered the most important category of consistency as it is directly perceived by the game's players.

If the network connecting the players to the game server was perfect, then all actions and updates would be performed instantly (real clock) and the game consistency would be guaranteed. Unfortunately, the network always presents some impairments which decrease the consistency of the game. Besides, consistency is proven to be also influenced by the packets' ordering. In fact, to ensure consistency, one needs to guarantee a causal order between client packets since player actions are dependent when the avatars are interacting (see [30] [31]).

To explain more the effect of disorder in online games, we will re-use the example illustrated in Figure 1.4. The variation of delays caused a cancellation of the causal relationship between the action of Player A and that of Player B. If some ordering mechanisms were applied, then the server would have to execute the actions in the adequate order and inconsistency would be avoided. Thus, Player B would hit Player A. If some solutions assuring reordering mechanisms are implemented in the game, the inconsistency is avoided.

1.3.2.3 QoS metrics and relation to QoE

To improve the satisfaction of online gamers, one needs to quantify their quality of experience. However, online games represent a special type of application as players are very demanding: they want the best service in terms of graphics, frames rate, speed, interactivity, sound, etc. They are in fact very difficult customers, which makes the evaluation of the QoE crucial and difficult. In fact, the satisfaction of the players can be influenced by the type of the game itself, by their experience or even the experience of the other players in the team and some possible social relationships between the players [32, 4]. Two basic techniques were proposed in order to evaluate the QoE of online games. The first one is single stimulus, where the players need to rate some specific parameters using the MOS tests, for example [29] [33]. The second is double stimuli, where the players need to compare two choices based on different parameters and decide which choice is better [34]. The results of all these models have proven that the QoE of online games and their consistency depend on the network's performance, mainly these three major QoS parameters: latency, jitter and packet loss. In fact, it was shown in [24] that the dissatisfaction of players caused by the degradation of these parameters can even lead them to quit the game.

- Latency: It is defined as the time needed to transmit the player's actions from their application layer to the server's application layer. In some works, it is defined as the round-trip time RTT but we will not use this definition. If the latency is high, it will influence the pleasure and the satisfaction of the player during the game session. Let's note the latency of a packet P_j between the player c_i and the server s as $D^j_{(c_i,s)}$ and the mean latency between the player c_i and the server s as $D_{(c_i,s)}$. This latency can differ from one player to another, which will lead, when no ordering mechanisms are present, to the inconsistency as described in 1.3.2.2.

- Jitter: According to [35], the jitter is the mean variation of the latency $D^j_{(c_i,s)}$ of two packets P_j and P_{j+1} from a source c_i to a destination s. We will note the jitter as J. It can be expressed as follows;

$$J = \mid D^{j+1}_{(c_i,s)} - D^j_{(c_i,s)} \mid \tag{1.2}$$

  It may be caused by the network's congestion or the fact that some packets from the same source do not follow the same path. Another

possible reason is sharing the Internet bandwidth with a TCP connection such as FTP. Some games consider the late packets as lost since they are only interested in the newer actions. This sometimes can be acceptable but it can cause a loss of consistency and of the logic of game if the late packets carry crucial actions.

- Packet loss: Different reasons can lead to the loss of some packets transmitted through the network. The loss rate differs depending on the network type. As a fact, the loss probability is higher for wireless network due to possible obstacles and signal degradation during the transmission. For wired networks, the loss is mainly caused by buffers as they drop the received packets when they become full. For TCP, the packet loss is not considered as a problem as the lost packets will be retransmitted. However, this retransmission will delay the well received packets until the lost ones are correctly received. When using UDP, the lost packets will not be retransmitted. Hence, some solutions are proposed such as predicting the lost action based on the behaviour of the player. Although this solution is complicated since it needs adding some new calculation and decision making procedures in the application layer, it does not necessarily ameliorate the QoE of the game since the prediction can be different from the real actions, which will lead to unsatisfied clients. The packet loss rate will be noted as $Loss$ for the rest of the report.

- Packet ordering: The definition of an ordered-packet arrival given in [36] is as follows:

  Definition 1 *Ordered arrival is a property which is found in packets that transit their path and where the packet sequence number increases with each new arrival and there are no backward steps.*

  We will propose a mathematical formulation of this definition. Let's consider a source S with k packets to send to a destination $Dest$. Let the source attribute to each packet P a sequence number N_P that represents the order of sending the packet. The index of the first packet sent is 1, the second is 2, etc. $Dest$ will receive the packets $RP_1, ..., RP_k$ where RP_j represents the j^{th} received packet. The receiver will check the sequence number of the received number to calculate the disorder.

The disorder d can be defined as follows:

$$d = \max_{i \in \{1,..,k-1\}} \mid N_{RP_i} - N_{RP_{i+1}} - 1 \mid \tag{1.3}$$

If $d = 0$, then the packets are received in order. Otherwise, the packets are received in disorder, i.e a different order than the order of transmission by the source.

These parameters are the base of every QoE prediction model for online games. However, they could be considered with different weights and importance from one model to another and from one game to another depending on the game category, the targeted users and the mechanisms that can be implemented in the game in order to mask a part of these imperfections.
In [24], the authors proposed a prediction model of players' departure based on the network conditions. They calculate the probability that the player decides to leave the game after 10 minutes of playing as follows:

$$lp = 12.5 * rtt.mean + 86.1 * rtt.sd + 1.1 * \log(c_{loss}) + 1.2 * \log(s_{loss}) \tag{1.4}$$

$$Pr[stay < 10min] = \frac{\exp(lp)}{1 + \exp(lp)} \tag{1.5}$$

where rtt.mean is the round trip average RTT or $2.D_{(c,s)}$, rtt.sd is the variation of the RTT which we defined as the jitter, J, c_{loss} is the loss rate $Loss$ of the clients' packets and s_{loss} is the loss rate of the server's packets.

1.3.3 Transport layer protocols for online games

The transport protocol used for massively multiplayers online games is either TCP [37] or UDP [38]. However, each of these transport protocols has its advantages as well as its disadvantages. The choice of the adequate protocol in this case is just a matter of prioritization that the game development company makes. In this section, we will expose the characteristics of each of these two transport protocols and, hence, their advantages and disadvantages in the context of the online games application. In other words, we will talk about the performance of these transport protocols based on the requirements of the quality of service of the online games that we discussed in 1.3.2, specifically the ordering, the delay and the packet loss.

Even though TCP does not seem the best choice for a real-time application such as the MMOG, the majority of the famous online games use TCP. In fact, it has been considered a strange choice by many researchers such as Griwodz et al. in [39] where they discussed the "fun fact" of using TCP for online games. The reason simply resides in the priorities of the game designers. As a matter of fact, both transport protocols TCP and UDP have their advantages as well as disadvantages. Hence, neither of them can be the best choice for online games applications. The use of either of them means to prioritize some features over others. In fact, there are even some online games that use both of them for different phases of the game. For instance, the famous MMOFPS game, Counter Strike, uses TCP as well as UDP. It uses TCP during the first phase, the connection phase, where the client is logging in and downloading the game maps and the environment data. Regarding this large influence that the transport protocol have on the quality of the game, many researchers focused on the study of other alternatives that can afford better performance than TCP and UDP for online games applications [40, 41, 42].

1.3.4 Open problems and motivation of our work

Online games are becoming very popular, generating an increasing amount of Internet traffic as illustrated in Figure 1.1. This traffic is mostly composed of headers followed by very small payloads [43] [44], especially under the TCP transport protocol. This results in a degradation of the efficiency of the underlying network. The second main characteristic of the traffic is its periodicity, which results in the flash crowds effect: the network receives a huge number of packets at almost the same time. The bursty behavior brings some challenges to the network's infrastructure in terms of processing, buffering, delay [44] and the requirements of the server in terms of bandwidth. As a result, the server becomes threatened by the overload problem which limits the scalability of these games. Thus, it becomes crucial to find mechanisms to decrease the network load generated by the online games and to increase the network's efficiency and the game's throughput.

Remark 1 *The traffic load is depending on the number of packets generated as well as the size of these packets. When we will deal with network coding technique, we suppose that all the packets have the same size and thus the load depends only on the number of the transmitted packets.*

On the other hand, the QoE of online games is strongly related to the consistency of the game. An inconsistent game is essentially caused by the delay, the jitter and the packet disorder introduced by the transmission network. Therefore, the network should provide some acceptable QoS performance to meet the satisfaction constraints of the increasing number of online gamers.

In this context, our objective is to design a routing protocol C that provides a certain packet order with a minimum number of transmissions and reduced bandwidth requirements for the server with the minimum latency $D_{(c,s)}$ that meets the latency tolerance limits D_{max} for online gamers.

In the next chapter, we will present the related works on the possible solutions to solve online games' main problems that we identified.

Chapter 2

Related Works

Contents

2.1 Introduction

After introducing online game applications, their specifications and their problems, we will present in this chapter the related works and the proposed solutions for these problems, mainly related to the consistency and to the traffic load. We will discuss two possible topologies for online games: the tree and the cycle. For each one, we will introduce possible solutions for both consistency and load optimization.

2.2 Network support for group communications

For group communication applications, the use of multiple separate unicast transmissions is not efficient and can't allow the scalability of the network. As a solution, one can use shared topologies for different flows. This topology can be either: 1) a structured topology such as a tree or a cycle or; 2) a peer-to-peer topology. "Online games" is one of these group communication applications. As the main architecture used for online games is a client/server one, we are more interested in the tree and the cycle topologies. In the tree topology model, presented in Figure 2.1, nodes from the players' layer will be connected with the upper layers until the server, going through intermediate nodes. The players in this case represent the leaves of the tree (see the red-colored branch in Figure 2.1) but they can also be relays (see the blue-colored branch in Figure 2.1). The intermediate nodes will aggregate the received packets and send one packet to the next node over the tree. As for the cycle topology case, the cycle will interconnect the players to the server. All the clients' and the server's packets are transmitted to their destinations through this cycle. However, in some cases, the cycle will be very large with a huge latency and will not be practical to implement. Thus, in order to limit the size and the delay of the cycle, we can focus on the players' layer and divide the cycle into multiple cycles, composed of groups of players and a gateway node, as illustrated in Figure 2.2. For now, we will call this gateway node a local server and we will explain its role with more details in the next chapters.

We suppose that the same topology will be maintained for the upstream and downstream flows. Note that the consistency and the traffic load issues are more critical in the upstream flow, from the players to the server. In the following, we will discuss these two topologies and the corresponding

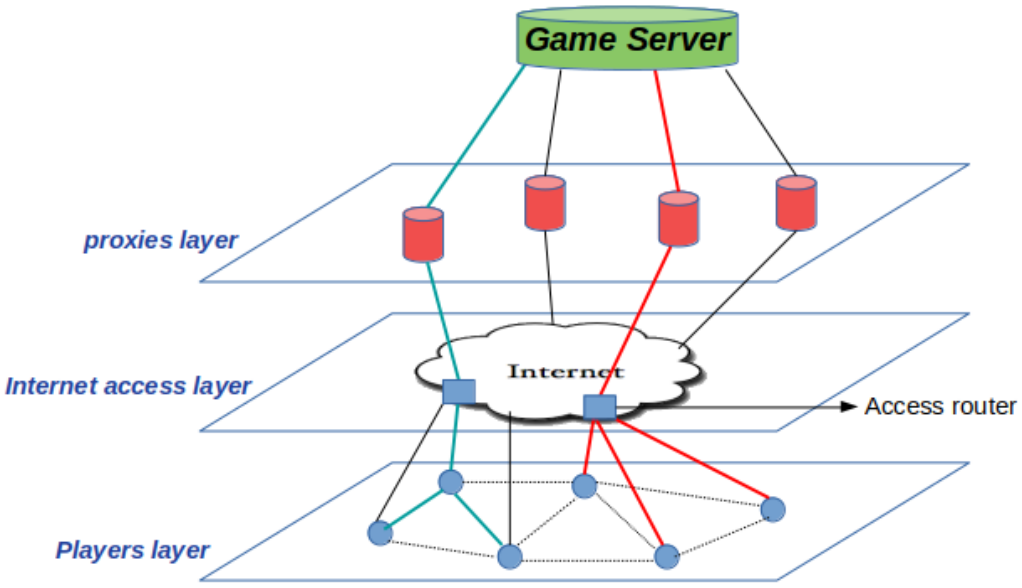


Figure 2.1: Tree topology introduced to online games infrastructure

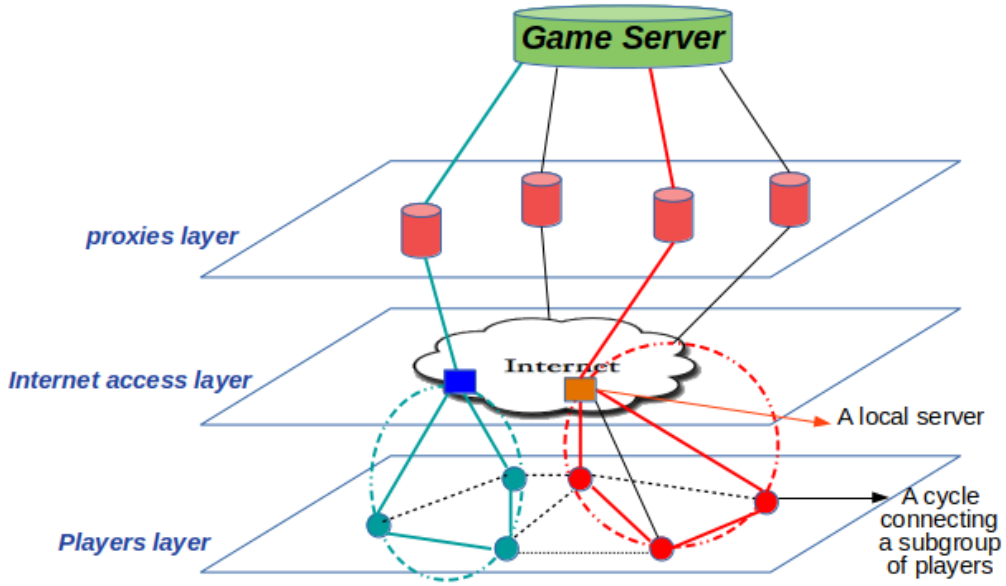


Figure 2.2: Cyclic topology introduced to online games infrastructure

solutions for consistency and traffic load.

2.3 Tree-based solutions

Tree topology is one of the conventional topologies when it comes to a group communication application. Indeed, the authors in [45] and [21] have proposed tree-based topologies for online games application. As the tree can be used for online games, we will discuss tree-based solutions for online game consistency and traffic load issues, then we will describe their effect on online video gaming.

2.3.1 Consistency

To provide consistency, some tree-based ordering protocols have been proposed in the literature such as the Tree-based Ordered Multicast (TOM) protocol in [46], a Total Ordering Multicast Protocol in [47] and a multiple-group ordering protocol in [48]. The packet ordering concept in these propositions is globally the same: All the packets need to go through a special node, generally called a sequencer, which is responsible for assigning an indicator to each packet. This indicator will help the receiver to order the received packets. The indicator can be, for example, a sequence number or a timestamp that represents the time at which the packet was sent. However, these protocols are rather complex and generate a very high signaling overhead, which will result in an additional delay. Besides, all the packets should go through the sequencer before being transmitted to their destination, which causes an additional delay and a bottleneck issue at the sequencer.

2.3.2 Load optimization

To reduce the traffic load over a tree topology, one can use several techniques such as efficient routing schemes for traffic balancing over the links [49], aggregation and header or data compression. Some of these solutions have been proposed for online games. For instance, data compression has been proposed in [50] to reduce the traffic from the server to the players. In [51], the authors proposed to aggregate the game traffic and showed that this technique can reduce the traffic load from the clients to the server and improve the game performance. A more elaborated solution, called Tunnelling,

Compressing and Multiplexing (TCM), has been proposed in [52, 53] based on a combination of header compression and multiplexing is adopted to optimize online games from the players to the server. As we believe that for online games the uplink traffic load issue is more crucial than the downlink one, we are rather interested in the TCM solution. Let us describe the TCM in more details in the following section.

2.3.2.1 Concept of Tunnelling, Compressing and Multiplexing

Packets are first received by a TCM agent, located at the access provider proxies or directly at the client end as a software. The TCM agent ensures successive header compression, multiplexing and tunnelling (see Figure 2.3) as follows:

Header compression: Once a packet is received by a TCM agent, its UDP/IP or TCP/IP header is compressed using a specified header compression protocol, IPHC or ROHCv2 for instance [53]. This compression reduces the IP header from 20 bytes to 2 bytes approximately, and the UDP/TCP header of 8-20 bytes to 2 bytes.

Multiplexing: At this step, multiple packets are merged into one packet (multiplexed). The multiplexing technique from [54, 53] is a periodic one– the TCM agent receives packets during a period T and multiplexes all of them into one packet. In the case where the agent receives only one packet, no multiplexing is performed. The multiplexing procedure is illustrated in Figure 2.4.

Tunnelling: At the last step the multiplexed packet is sent to the server using a layer-2 tunnelling protocol.

At the receiver end (main server), the TCM packets are demultiplexed and decompressed to their original form.

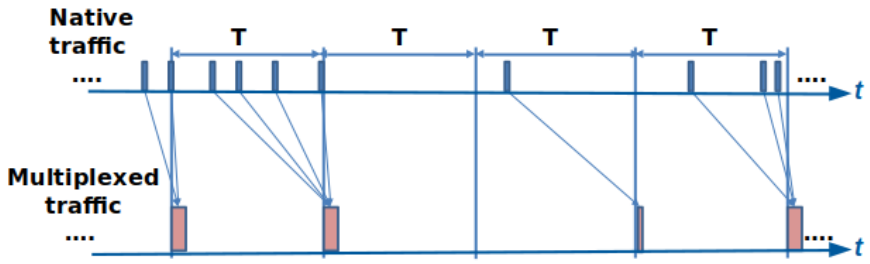


Figure 2.4: Multiplexing method.

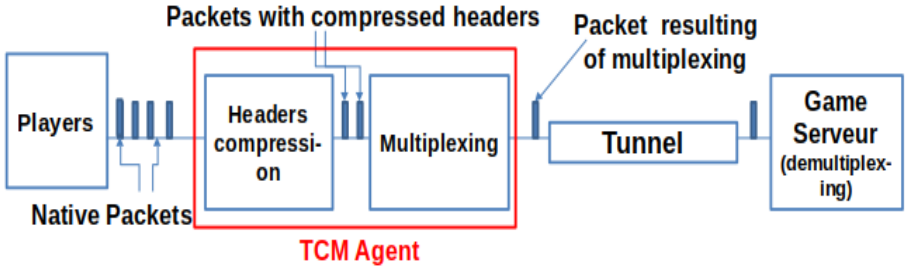


Figure 2.3: Intermediate functions within the TCM.

Figure 2.5 illustrates the protocol stack and the structure of one TCM packet. The packet is composed of the following parts:

- Common Header (CH): for the whole packet.
- PPPMux header (MH): added at the beginning of each compressed packet.
- Reduced header (RH): includes the IP/UDP compressed header of each native packet.
- Payload (P): the UDP payload of the native packets.

The size of the multiplexed packet S_{mux} is then expressed as follows when the number of the multiplexed packets k is larger than 1 [54]:

$$S_{mux} = CH + k(MH + E[RH] + E[P]) \tag{2.1}$$

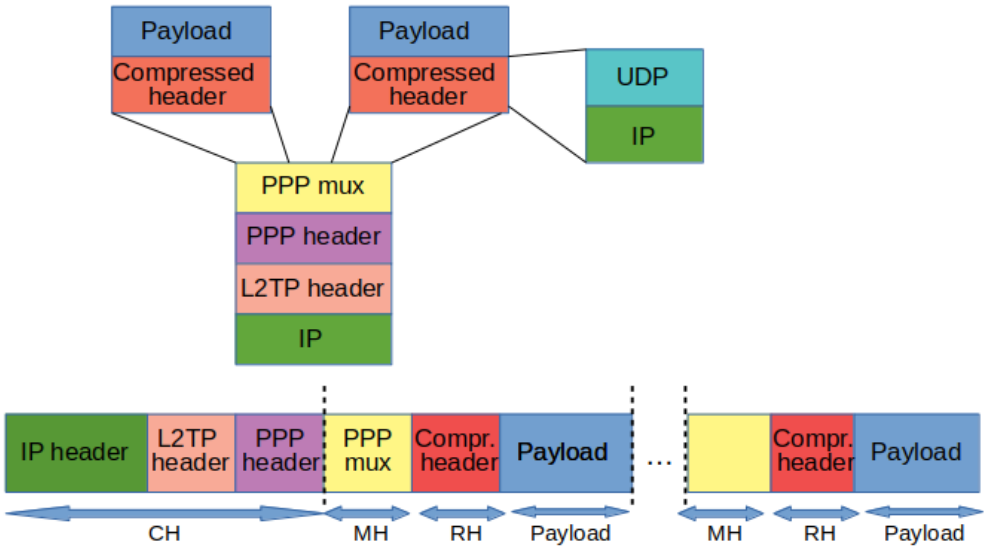


Figure 2.5: Compression method.

2.3.2.2 TCM limitations

There are three limitations in the TCM approach:

1. An additional delay will be incurred given that the packets will only be sent at the end of the multiplexing period T, independently of their arrival time. This delay is proportional to T and is equal in average to $T/2$.

2. Since this additional delay changes from one packet to another, depending on the time the initial packet was received by the TCM agent, a variable jitter delay is also added. This is illustrated in Figure 2.6. Given three initial packets 1, 2 and 3, the delay between the multiplexed packets, carrying packets 1 and 2, has increased, while the delay between the multiplexed packets, carrying packets 2 and 3, has decreased.

3. If the number of initial packets, gathered during the period T, is large, the size of the multiplexed packet will also become large, which may lead to the packet segmentation at lower network layers and, thus, to increase the delay.

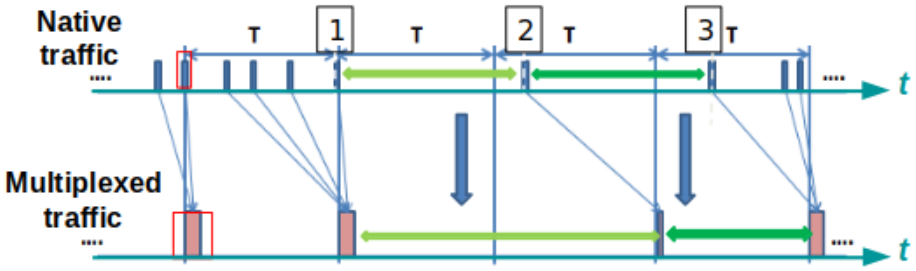


Figure 2.6: Example of a jitter caused by the TCM.

To conclude, the TCM technique can reduce the network's inefficiency by increasing the utility of the online games traffic. The gain of the TCM technique is dependent on the multiplexing period and the number of connected players. Hence, an ideal implementation should be associated with a dynamic choice of this period based on the mentioned parameters. The disadvantage of this technique is its additional delay which can damage the game quality specially for games using TCP or with very interactive activities [55]. However, it was shown that even with this delay, TCM can manage to afford an acceptable delay by the online gamers if well studied limits to the multiplexing period are established. Certainly, these limits will reduce the throughput gain, but it will guarantee an acceptable QoE [55, 56]. Another problem is the variation of the delay for the online gamers which can increase the inconsistency of the game. Finally, the packet size can be very large which may introduce segmentations through the transmission. Hence the probability of packet loss, retransmission and waiting will increase.

2.3.2.3 TCM advantages

In order to decide whether the TCM is useful in online gaming scenarios, one should evaluate the impact of the first two limitations onto the QoE of players, which will be done in the next chapter. As for the third limitation, it can be circumvented with a new version of the TCM algorithm, presented in Section 3.2. In [53] and [54], TCM has been proposed as a solution to reduce the traffic charge of online games. However, there were no simulations to confirm the benefits of TCM. The gain has been demonstrated using an emulation based on fixed values. Hence, we set up different simulations to

evaluate the effect of applying TCM. Besides, we will simulate the forest topology and evaluate its performance with and without TCM.

2.4 Cycle topology

Cycle topologies offer an appealing alternative to tree topologies for group communications because of the attractive benefits that they can provide such as ease of management, reliability and survivability. However, the use of cycles increases the transmission latency which might not be acceptable for delay-sensitive applications such as multiplayer online gaming.

2.4.1 Consistency

Thanks to its ordered nature and network-failure resistance, cycle topology offers more simplicity and robustness when implementing ordering protocols compared to the tree. Therefore, many researchers were interested in the cycles when elaborating their ordering protocol. The main techniques used in a cycle-based ordering protocol are: 1) a token: it is a short message conveying global status and is used for the coordination of the different sites in the group and; 2) a timestamp. In this context, token-based ordering protocols are famous and attractive considering their simplicity, flexibility and high performance. As a result, they were chosen to be deployed in many practical messaging services such as the Spreed toolkit [57], the Corosync cluster engine [58] and the Appia communication framework [59].

Among the first cycle-based ordering protocols propositions, we cite TPM [60], where a token is used to control broadcasting and message sequencing. It offers a reliable ordered multicast communication for distributed process groups in the presence of failures and network partitions. This protocol offers a limited management of group partitioning that can handle the division of the group but not the merging of these groups.

ISIS is another protocol proposed in [61]. It provides three types of delivery order: unordered messages or BCAST, causal ordered messages or CBCAST and totally ordered messages or ABCAST. Both the vector clock and the token techniques were used in this protocol: the vector clock is used to ensure the causal ordering and the token-based sequencer is used in order to provide the total order. The total order is obtained by converting a partial message order previously established. These protocols were the base for

more enhanced ordering protocols. As an example, the Totem protocol was proposed in [62] as a total ordering reliable token-based protocol that can handle more efficiently the failure cases, the partitioning and the merging of groups. This protocol offers lower computational cost and higher throughput than older propositions such as Trans and Total protocols [63]. With Totem, consistency and packets ordering is maintained despite network partitioning and remerging, or processor failure and recovery with a stable storage intact. Totem was designed for a local area network. However, the implementation of the Totem protocol presented some difficulties and challenges in maintaining the total order, especially with possible topology changes.

To solve these challenges, a more general and extended version of the Totem protocol, called The Totem multiple-ring protocol, was proposed in [64]. This protocol is built on top of the original Totem single-ring protocol where multi-cyclic topology is used to interconnect different groups. The protocol succeeded to provide a reliable, totally ordered and consistent message sequencing even in the presence of network partitioning, reemerging, processor failure and recovery. The global total order of messages across the multiple interconnected local-area networks is conserved even between the different cycles thanks to the use of a combination of sequence numbers and timestamps.

As the major disadvantage of using a cycle is the delay, other propositions focused on reducing the delay and providing fast total ordering solutions. In [65], the authors were interested in developing an efficient and fast ordering protocol. Considering the simplicity of cycles and token-based flexible semantic protocols, they chose to use a logical cycle as a topology. They then proposed the Accelerated Ring protocol as a solution to improve the performance of standard token-based protocols. It minimizes the delay by allowing the processes to pass the token even before the end of their data transmission. Results showed that by using this solution, the delay is reduced and the throughput is increased. This protocol is also efficient for partionable network models.

2.4.2 Load optimization

To reduce the load over cycle network links, one can use some optimized routing protocols. Another alternative is the network coding technique. Although this technique was initially introduced for acyclic networks, it has been shown in the literature that it can provide gains in term of load and

throughput even in cyclic topology. In the following, we will introduce the concept of network coding and its advantages. Then we will present some of its applications. Afterwards, we will describe how we can exploit this technique in a cyclic network, which we will use in our proposed solution in Chapter 4.

2.4.2.1 Introduction to network coding

Network coding (NC) has been proposed in 2000 by Ahlswede *et al.* [66] in order to achieve the maximum throughput rate in a wired network for a simple source transmission case, which cannot be achieved by the traditional receive-and-forward routing method. Afterwards, NC has become considered as an efficient solution to increase the throughput of either wired or wireless networks and solve the min-cut max-flow problem as illustrated in Definition 2 and Theorem 1 [67].

Definition 2 *A cut between a source S and receiver R is a set of graph edges whose removal disconnects S from R. A min-cut is a cut with the smallest (minimum) value. The value of the cut is the sum of the throughputs of the edges in the cut.*

Theorem 1 *Consider a graph G=(V,E) with unit capacity edges, a source vertex S, and a receiver vertex R. If the min-cut between S and R equals h, then the information can be sent from S to R at a maximum rate of h.*

The basic idea of network coding is to allow the intermediate nodes to not simply forward the messages they receive but, instead, to algebraically combine several received messages from its neighbors and possibly its own messages and transmit the resulting message to the next hop. Using network coding technique, the max-flow min-cut equality can be achieved for an acyclic multicast transmission case as implied by the following main theorem of NC [67].

Theorem 2 *Consider a directed acyclic graph $G = (V, E)$ with unit capacity edges, h unit rate sources located on the same vertex of the graph and N receivers. Assume that the value of the min-cut to each receiver is h. Then there exists a multicast transmission scheme over a large enough finite field $\mathbb{F}_q$, in which intermediate network nodes combine their incoming information symbols over $\mathbb{F}_q$, that delivers the information from the sources simultaneously to each receiver at a rate equal to h.*

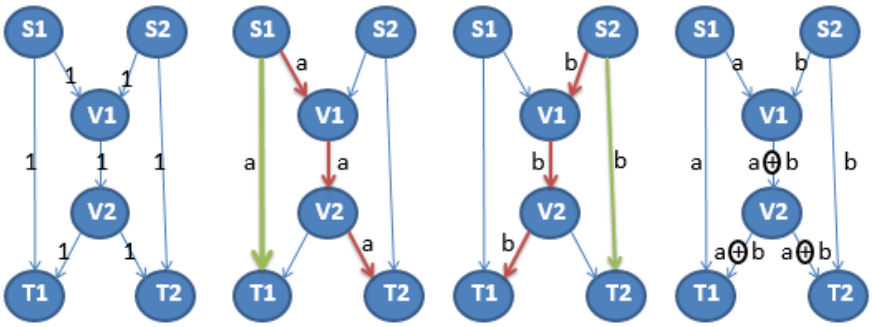


Figure 2.7: Basic network coding example : the butterfly case [1]

One of the most well-known examples that explain the advantage of the network coding technique is the butterfly example that we see in Figure 2.7 [1]. In this example, both sources S_1 and S_2 need to send their packets a and b respectively to both terminals T_1 and T_2. These two packets need to go via the intermediate nodes V_1 and V_2. As each link can only transmit one packet at a time slot, the link between V_1 and V_2 becomes the bottleneck of this network. Hence, by simple forwarding, we need 5 time units for both packets a and b to be received by both terminals. However, if we apply the network coding technique, the intermediate node V_1 can XOR both packets a and b and generate a new packet. The resulting coded packet will be forwarded through the bottleneck during one time slot and then will be transmitted to the terminals. The terminals will recover their needed packet by XORing the received coded packet with the native packets. Hence, both packets a and b have got through the link between V_1 and V_2 at the same time and the total time units needed is reduced to 3 instead of 5. In this example, we can see that the maximum throughput of the network passed from 1.5 time flows to 2 time flows thanks to the use of the network coding technique.

2.4.2.2 Network coding applications

Network coding has gained a remarkable interest and has been widely studied thanks to its advantage in terms of increasing the network throughput and minimizing the number of needed transmissions. Numerous network coding based solutions have been proposed and applied for multiple areas and applications such as point-to-point communications, content distribution and

storage applications [68], wireless mobile networks, video delivery [69], peer-to-peer networks, sensor networks [70], device-to-device communications [71] and even satellite networks [72]. Besides the throughput, network coding has achieved gains in terms of energy consumption for wireless applications where all nodes need to transmit information to all the other nodes, i.e. in a multicast scenario [70, 73]. Some specific examples showing this advantage can be found in [74] for a sensor network and in [75] for an adhoc network. Another area of application of network coding is ensuring robustness against transmission errors and failures. In this context, network-error-correcting codes such as erasure codes are largely studied [76]. Network coding has been proposed even for delay-sensitive and real-time applications such as video streaming [77, 78], and vehicular adhoc networks [79].

2.4.2.3 Linear network coding

Linear network coding (LNC) is one of the simplest and the most-used network coding types. It consists in combining the messages using some linear combinations and the resulting message will be decoded by the receiver by Gauss elimination after receiving a sufficient number of combinations. Practically, each node combines all the received packets using some specific coefficients that are assigned to each input link. These coefficients form the coding vector of a node. Let us consider a general setup with v sources $S_1, ..., S_v$. Each source S_i wants to send one symbol x_i to a set of w receivers. Let us note the set of resulting transmitted symbols as the vector $X = (x_1, x_2, ..., x_v)$. If the min-cut of the network for all the flows is equal to h, then each receiver i will receive the vector $Z_i = (z_{i,1}, z_{i,2}, ..., z_{i,h})$. Each symbol of Z_i is a linear combination of the symbols sent by the v sources. Hence, we can represent each received symbol $z_{i,j}$ as the multiplication of X by a coding vector $V_{i,j}$ as shown in Equation 2.2.

$$z_{i,j} = X \cdot V_{i,j}^T \tag{2.2}$$

Hence, the received vector Z_i can be represented as follows:

$$Z_i = X \cdot M_i \tag{2.3}$$

where M_i is the coding matrix corresponding to the receiver i and composed of the different coding matrix $V_{i,j}$. Note that if the symbols of X are over the finite field $\mathbb{F}_q$, then all the elements of the coding matrix must also be over the same field $\mathbb{F}_q$. In order to recover the original sent message, the receiver i should calculate the inverse of the coding matrix, ${M_i}^{-1}$.

The LNC method fulfills the maximum throughput and solves the min-cut problem as defined in the basic NC theorem 2. There are two well-known types of network coding: the random linear network coding (RLNC) and the XOR coding which we will describe in the following.

2.4.2.3.1 Random Linear Network Coding

A special case of linear network coding is the random linear network coding (RLNC). In this type of coding, the node combines the received packets linearly as usual but using a coding matrix that is generated randomly. Afterwards, the coded packet is sent to the next node accompanied by its coding vector. When receiving the coded packet, the sink can recover the initial packets by calculating the inverse of the coding matrix and use it to decode the received packet by using Gaussian elimination. When it comes to the wireless domain, two types of networks can be distinguished: a network with a specific infrastructure and a network with no infrastructure, also called adhoc network. RLNC's advantage is the simplicity of its concept, allowing it to be adopted to the network topology, even to adhoc networks. However, RLNC is costly in terms of bandwidth as it requires an additional overhead in order to transmit the coding vector, and in terms of calculation complexity for decoding. All these characteristics make RLNC more suitable for adhoc networks than for static networks. In [78], the authors proposed an RLNC that can provide better trade-offs between bandwidth efficiency, computing complexity and delay. On the other hand, in [80], it has been demonstrated that instantly-decodable network coding (IDNC) is more efficient in terms of decoding delay. Hence, IDNC is more suitable for centralized storage networks. In [81], the authors preferred working with IDNC when designing codes for real-time applications.

2.4.2.3.2 The XORing method

Network coding was initially proposed to solve congestion issues in wired networks [66]. However, its application in the wireless network case is being intensely investigated [82, 70, 83]. Wireless network coding (WNC) exploits the broadcasting nature of wireless channels to provide a throughput gain as can be shown in the example illustrated in Figure 2.8[2]. In this example, nodes A and B need to exchange messages via the relay node R as they are too distant to communicate directly. This so-called the Alice-and-Bob topology is a simple multihop wireless network case where NC can be applied. A sends

a message α to B while B sends a message β to A. Without the wireless network coding, four time slots are needed to ensure the communication between A and B, as shown on the left side of Figure 2.8. However, when applying the wireless network coding, only three time slots are needed, as illustrated on the right side of Figure 2.8.

In time slot $T3$, the intermediate node R encodes the received messages, α and β, by XORing them and generates a new message γ. γ is then broadcast to A and B. A decodes the received message γ by XORing it with its own message α. As a result, A obtains the needed message β. The same goes with B. It XORs γ with its message β to obtain the awaited message α. Hence, as illustrated in this example, WNC can improve the wireless network throughput by using the broadcast transmissions to decrease the number of total time slots needed to ensure the communication.

2.4.2.4 Use of NC in practice

As mentioned in previous section, two types of networks can be distinguished in the wireless field: networks with a fixed topology and adhoc networks. Based on the type of the network, two NC approaches, the deterministic and the probabilistic network coding schemes, are proposed.

2.4.2.4.1 Deterministic NC schemes

In the case of a well-known (static) network infrastructure, transmitting nodes can be selected on a permanent basis and NC can be applied by well-determined nodes using specific received symbols. These parameters are calculated in advance in an optimal way in order to guarantee the maximum possible throughput with a minimum delay. This approach is referred to as deterministic network coding. In the case of a static network, an entire knowledge of the network topology can be provided. However, this cannot be always guaranteed. Hence, some researchers designed deterministic network coding schemes for wireless networks based on local information about the neighborhood information [84, 85]. These propositions are based on a new parameter called pruning which presents the local information about neighbors. We differentiate between several types of pruning depending on the depth of knowledge: self-pruning for 1-hop information knowledge, dominant pruning for 2-hope information [84], called also a partial dominant pruning in [86, 85] and finally the total dominant pruning for 3-hop neighborhood information.

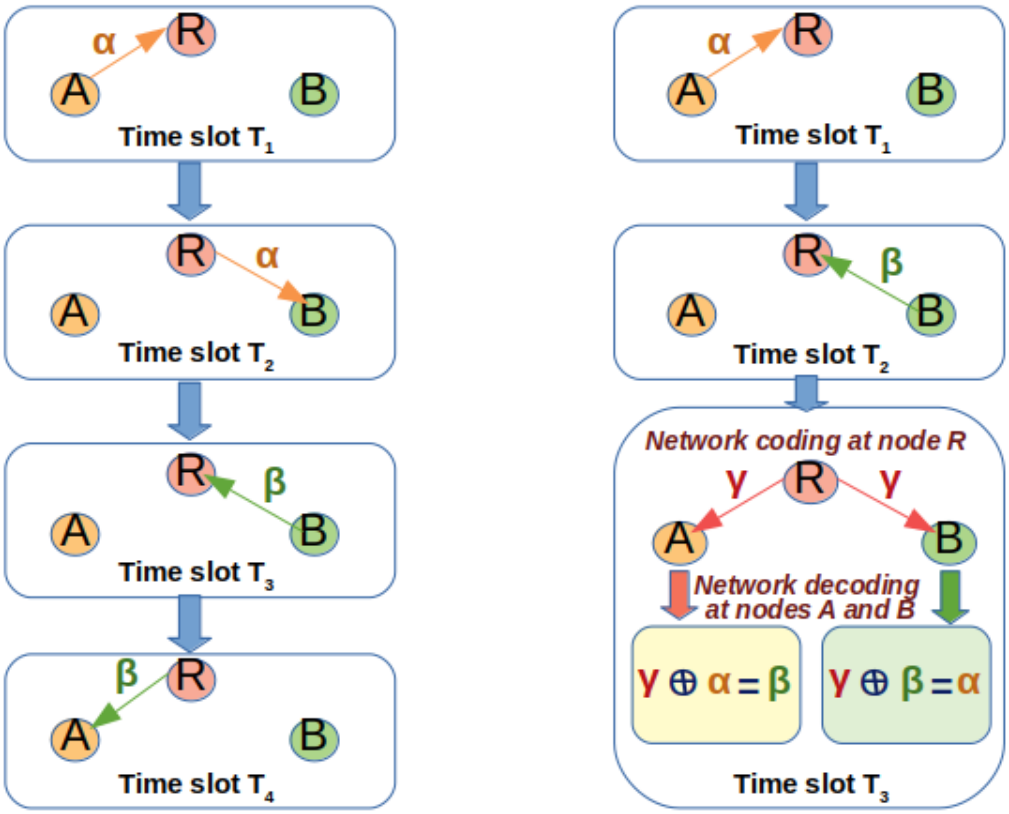


Figure 2.8: Network coding example in a wireless network [2]

2.4.2.4.2 Probabilistic NC schemes

In the absence of any guaranteed information about the topology, other works proposed to apply probabilistic approach to network coding. In this approach, as we are not sure about the topology, the nodes will code the received packets with a certain probability or coding factor P_{NC} and send the received packets without coding with the probability $1 - P_{NC}$ [87]. This factor is calculated by taking three parameters into account: the throughput gain, the delay and the probability threshold of a successful decoding operation [88]. Different algorithms have been proposed in order to efficiently calculate this factor [87, 88, 73, 89, 90, 91].

RLNC can also be used in the case of topology as it does not need a full knowledge of the topology in order to code or decode.

As we are focused on online gaming applications, there aren't many chances that the players will change their location after starting a game session as they need full concentration. Actually, players are generally located at their home or at cyber cafés. Therefore, in our proposed solution, we will consider a deterministic approach for online gaming, where the topology is a logical cycle and all the coding vectors are previously calculated. However, if the players are rather mobile and change their location frequently, then the logical cycle topology construction should be performed periodically to guarantee the connectivity between the nodes. This construction can be based on the neighboring information or pruning as explained in Section 2.4.2.4.1. When the topology is constructed, we can apply our deterministic protocol. Some metrics should be set to calculate the period of the topology construction. These metrics can include a threshold of undecodable packets,

2.4.2.5 Network coding over cycles

Network coding technique was proposed to achieve the maximum throughput for an acyclic network as established in theorem 2. When working with an acyclic network topology, an upstream-to-downstream ordering among the nodes is established [92]. This ordering enables the synchronization of nodes so that each message can be treated individually. However, if a cycle is present in the topology, this property is no longer guaranteed. Let us consider for instance the cycle example ABCD in Figure 2.9, where two sources S_1 and S_2 send their messages to sinks T_1 and T_2 respectively. In this case, an S_1 stream is transmitted from AB to CD, while an S_2 stream is transmitted from CD to AB. Hence, no upstream-to-downstream order is preserved.

One way to deal with this issue and to make sure that the acyclic network coding is still applicable in case of cycles is to actually prevent cycles by imposing some restrictions on routing. This method is not always possible. Another way to resolve the problem is to expand the network in time and to regain the acyclic graph characteristic by considering the transmitted information as a pipeline of messages [93]. For instance, in [92], the authors defended that even in case of cyclic topologies, the upstream-to-downstream ordering can be established thanks to the time dimension. In fact, the transmission medium, when observed as a space-time domain, is still acyclic. The authors of [92] also proposed a method to converse the cyclic graph into a layered acyclic one. However, this approach has many drawbacks such as encoding and decoding complexity and a too large delay [94]. Besides,

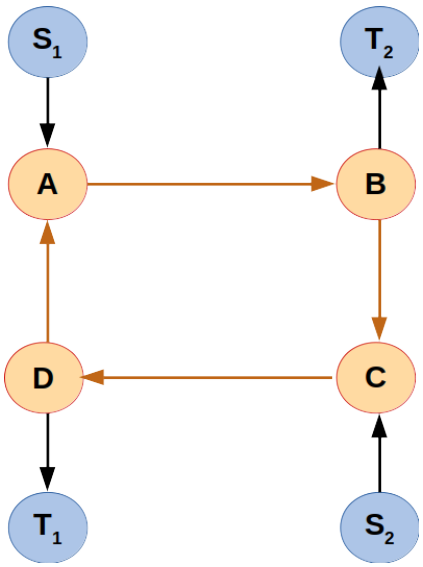


Figure 2.9: A cyclic network case in which the upstream to downstream order is not preserved [3]

in some cases, the optimal transmission rate cannot be achieved using an acyclic graph [93]. Thus, it may be more efficient to exploit the recursive nature of cycles instead of trying to avoid them. Also, the authors in [93, 3, 95] proposed to switch from a delay-free model of networks and consider delays in order to deal with cycles, which makes it natural to focus on convolutional codes. In [94], the authors proved that it is not necessary to have a delay in every edge of the cyclic graph. One just needs to have at least one edge in each cycle that has a non-zero delay which can ensure stability and causality of the information flows.

As a result, the equations proposed for linear coding over an acyclic network can be adjusted to suit this new convolutional approach, taking into account the delays introduced by the edges of the network. For instance, Equation 2.3 can now be presented as follows :

$$Z_i(D) = X(D) \cdot M_i(D) \tag{2.4}$$

where D is the delay operator variable.

Based on this approach, several propositions have been made for the construction and the design of an efficient convolutional network code for cycles like [95, 94, 96, 97]. These works, however, have focused on one source

node case and on achieving the maximum throughput more than on reducing the transmission delay over the cycle. We will use this convolutional network coding approach afterwards when we introduce the proposed routing protocol in Section 4.3.3.2.

When it comes to case of wireless networks and wireless coding based on broadcasting, other approaches for network coding over cycles have been studied focusing on the multicast scenario where each node has a message to send to all the other nodes in a time-slotted and synchronized network. The basic idea is to allow the intermediate nodes to send the combination of the received packet from their two neighbours. Each time a node receives two coded packets, it uses the already-sent ones to decode them, code the recovered packets and send them to both neighbors . Hence, the intermediate nodes alternate between coding and decoding. The operation is a simple XOR between the received packets with coefficients always equal to 1 [75, 98, 74] which reminds us of the wireless-relaying network-coding example shown in Figure 2.8. Using network coding in this cyclic multicast setup increases the throughput and helps to achieve the minimum number of transmission limits over such a network that has been proved in [99] to be equal to :

$$T_w \geq n - 2 \tag{2.5}$$

$$T_{nc} \geq (n-1)/2 \tag{2.6}$$

where n is the number of nodes of the cycle, T_w denotes the number of transmissions required for an information unit to reach all the nodes without network coding and T_{nc} denotes the number of transmissions required for an information unit to reach all the nodes with network coding. This leads us to a maximum gain in terms of transmissions number equal to 1/2.

$$\lim_{n \to +\infty} T_{nc}/T_w = \lim_{n \to +\infty} (n-2)/((n-1)/2) = 1/2 \tag{2.7}$$

Besides, it has been shown in [74] and [75] that network coding in a wireless multicast cyclic transmission case provides lower energy consumption, which is an important advantage when it comes to wireless terminals. The authors in [98] proved that their network coding solution decreases also the delays of transmission over the cycle. In the next chapter, we will introduce our solution which is inspired by these works.

2.5 Cycles versus Trees

Even though the tree topology may seem a straightforward choice for such a group communication application, it is not necessarily the best choice for every application. It offers a fast transmission and a simplicity of construction compared to the cycle-based topologies. However, when dealing with consistency, the tree-based ordering solutions are generally costly in terms of overhead and signaling. Besides, as they are based on a sequencer to which all packets need to be transmitted in a first step, an additional delay is generated in order to reach the sequencer and the sequencer will suffer from bandwidth issues. Moreover, if a problem occurs to the links between the nodes and the sequencer or if the sequencer itself breaks down, then all the ordering procedure will fail and the transmission will be interrupted. On the other hand, it was shown in previous works by Wang et al. that providing a reliable and survivable transmission is more complicated over a tree topology than over the cyclic topology, as many key management schemes may be used on the cycle [100, 101, 102]. The authors in [102] also proposed a multi-cyclic topology design to cover the case where the transmitting nodes are distant and to provide scalability to the network. Moreover, it has been stated in [103] that the cyclic topology is even better than the tree topology for real time group multicast when active nodes can change in number or in distribution as it will require signaling protocols, which is not the case with cycles. Given their numerous advantages, cycle topologies have recently gained particular interest in adhoc networks such as energy efficient routing for sensor networks [104] and reliable delivery of control packets for industrial networks [105]. Considering all these facts, it is interesting to investigate cyclic topology further as it seems a strong possible alternative to the tree topology for many applications, particularly for the multiplayer online video games. However, the presence of cycles might increase the transmission latency and, thus, worsen the QoE in terms of responsiveness. Hence, it will be interesting to provide solutions to reduce this latency in order to make the cyclic topologies more appealing and applicable for real time applications. Note that no researches have compared the tree and the cycle topology for the case of online gaming. Indeed, the cycle topology has not been yet considered for online games' application.

2.6 Conclusion

In this chapter, we discussed the conventional topologies for group communication applications: the tree and the cycle. For each topology, we presented possible solutions for online game issues, mainly traffic load and consistency. Both of these topologies are interesting given their advantages. For instance, the tree topology is better in terms of delay but this adds a large delay and complexity when it comes to consistency and ordering. As for cycle topology, it is simpler when it comes to consistency and ordering, but it introduces larger delay than the tree. Despite the numerous advantages of the cycle topology, at our knowledge, no one considered this topology for online games. One of the contributions of this thesis is the investigation of cycle topology performance for online games.

In addition to the cycle topology, we will propose an optimized routing protocol coupled with network coding. Network coding will not only reduce the latency over the cycle, but will also increase the efficiency of the network by decreasing the traffic load. Note that, for online game traffic, NC cannot be beneficial if applied to the tree topology as the min-cut in this case is equal to one. Hence, thanks to the cycles, we can investigate the impact of applying the network coding technique for online games.

Part II

Packet multiplexing for online gaming

Chapter 3

Performance impact of packet multiplexing and player partitioning on MMOG games

Contents

3.1 Introduction

In this chapter, we will focus on the study of TCM, the solution proposed to increase the traffic efficiency based on a tree topology. There are two main contributions is this chapter. First we introduce an ameliorated version of the TCM technique for online game and evaluate its performance via simulations. Second, we propose to apply the player partitioning approach coupled with the enhanced TCM. We refer to the resulting topology as a forest. This work was published in [106].

3.2 Modified TCM protocol

In Section 2.3.2, we presented the TCM technique and explained its benefits in terms of throughput gain and network efficiency. However, this gain is not always guaranteed. In fact, the best results for TCM can be obtained when the number of the multiplexed packets is large. Therefore, the solution found is to increase the multiplexing delay. This will result in:

- increasing the additional delay caused by the multiplexing period;
- increasing the jitter: the delay variation between different packets;
- a very large number of packets to be multiplexed and hence a large multiplexed packet that may have to be segmented before being sent.

The segmentation procedure will cause an additional delay in order to wait for all the segments to arrive before treating the packets and also a problem in case of packet loss: if many segments are lost, then waiting for all the successful retransmissions induces a larger delay. Hence, if one limits the number of multiplexed packets to a threshold calculated based on the packets' size and the maximum MTU to be transmitted through the network, the resulting multiplexed packet will not need any segmentation and the packets can be rapidly multiplexed if enough packets are received, even if the multiplexing period is not finished yet. The solution we propose to avoid the segmentation of multiplexed packets consists in a dynamic multiplexing period that can get two possible values: 1) a fixed value, T, which is the maximum acceptable multiplexing period calculated based on the generated additional delay and the network's performance (the transmission delay) and;

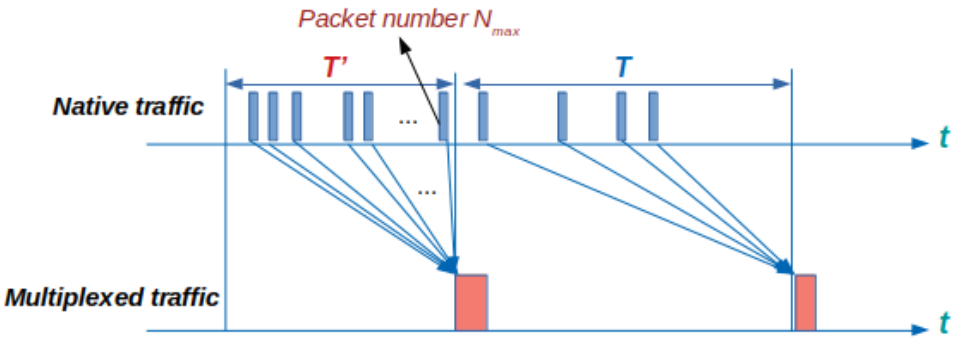


Figure 3.1: The modified TCM version.

2) a variable value, T', corresponding to the sufficient time needed to accumulate the maximum number of packets N_{max} so that S_{mux} do not exceed the network's MTU. The value of N_{max} is calculated using (2.1) as follows:

$$\begin{aligned} \mathrm{S}_{mux} &\leq MTU \\ CH + N_{max}(MH + E[RH] + E[P]) &\leq MTU \\ N_{max} &= \lfloor \frac{MTU - CH}{(MH + E[RH] + E[P])} \rfloor \end{aligned} \tag{3.1}$$

An illustration of the proposed solution is presented in Figure 3.1.

The proposed modifications are summarized in the pseudo-code of Algorithm 1. It allows multiplexing and sending packets by the TCM agent after having received N_{Max} initial packets even before waiting for the end of period T. In this way, the maximum size of the multiplexed packet is controlled by the TCM agent, and the additional delay is reduced.

3.3 Forest topology

As discussed in 1.2.2, the main architecture used for online games is the client/server one. However, it was defended in many works that hybrid architectures where the server is distributed can be more effective for online games [107]. One way to distribute the server is to partition the players as we see in the following.

Algorithm 1 Our TCM protocol

```
1: procedure TCM ROUTING
2: Define
3:     T_TCM as the multiplexing period
4:     Timer is set to be equal to T_TCM
5:     N_Packets as the number of packets received at the TCM Agent
6:     N_Max as the max. number of packets to multiplex
7: loop (forever):
8:     start Timer
9:     while ! Timer timeout do
10:
11:        if receive packet then
12:        N_Packets = N_Packets + 1
13:        if N_Packets = N_Max then
14:        compression and multiplexing of packets
15:        send packet to server
16:        N_Packets = 0
17:        start Timer
18:
19:        wait until Timer timeout
20:
21:
22:    end while
23:    if timer timeout then
24:        if N_Packets = 1 then
25:            send original packet
26:        else
27:            compression and multiplexing of packets
28:            send packet to server
29:            N_Packets = 0
30:            start Timer
31:        end if
32:    end if
33: end of loop forever
34: end procedure
```

3.3.1 Client partitioning

In order to distribute an online game system and balance the traffic load, players should be divided into subgroups. The partitioning criteria is generally the Area Of Interest (AOI) of the players. In fact, in virtual environment systems, each participant is usually represented by an avatar. This avatar is only concerned by what is happening in his proximity, which is referred to as the Area of Interest. Hence, the actions can be sent based on this criterion resulting in a remarkable decrease in the traffic overhead. In fact, based on the defined AOI of each avatar, the decision of sending some updates to that avatar will be made. Using this concept, a virtual world can be divided into sub-worlds where every sub-world contains a set of the avatars' area of interest. The number of sub-worlds can be calculated depending on the popularity (the density of the population) of some areas and on the available resources that will be controlling these sub-worlds. An example of a distributed architecture based on the AOI concept is illustrated in Figure 3.2.

This approach can be easily applied in the case of online games. In fact, in online games, each player has a certain range of vision of the game world. This range includes a limited number of environmental objects and other avatars. As an example, Figure 3.3 represents the game view of one player of the League of Legends (LoL) game. As we can see in the figure, the player has only the vision of the nearby objects, with a limited global vision of the rest of the game map with very few details (the zone limited by a dashed red line). The vision range is the same for each player and it can be calculated based on the avatar's position on the map and the parameters of the game [13].

3.3.2 Our proposition: the forest

The partitioning procedure reduces the bandwidth required by the server and increases the game throughput and the server's interactivity. Others considered partitioning the players based on their geographical location in the real world in order to bring some services such as the audio effects of online games closer [108], which can reduce the round trip delay. When the TCM was proposed for online games, it was based on the possibility of having a number of players located in approximately the same geographical area. One possible scenario is the case of cyber coffees where a group of players are

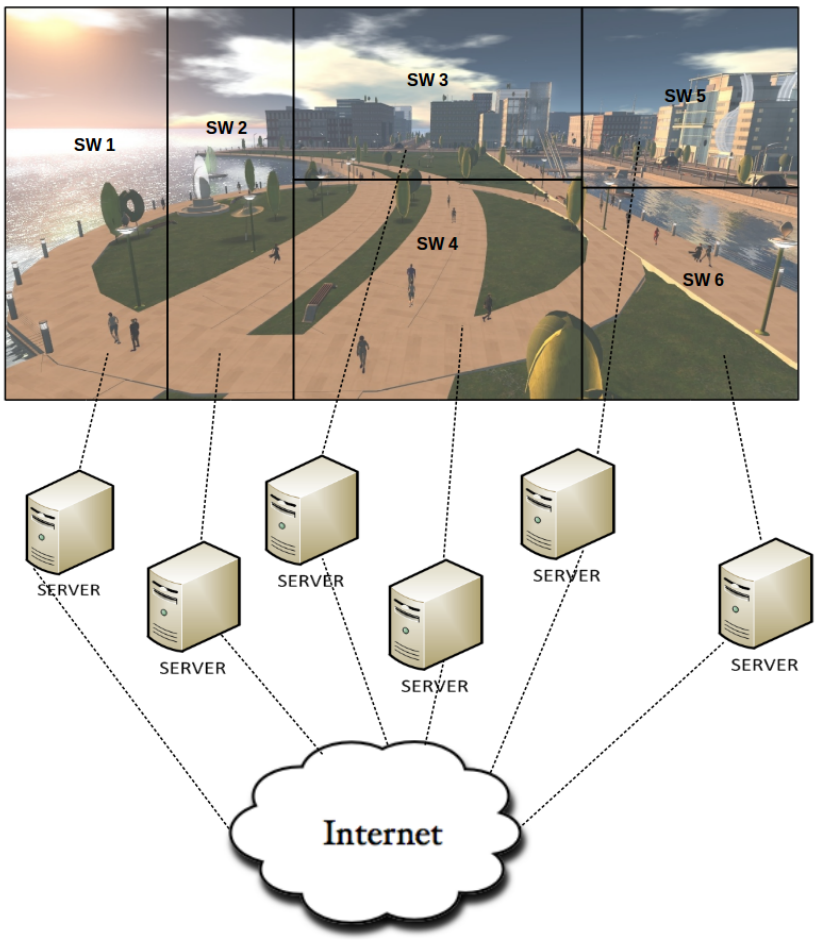


Figure 3.2: Example of distributed architecture for a virtual world based on the AOI criteria.

Figure 3.3: Illustration of the global vision and the detailed vision of League of Legends players.

playing the same game at the same time, or a game party where friends are gathered to play against each other. In this case, one can partition the players based on their location in the game world as well as their location in the real world. A subgroup of players in our model is a set of players that are located in the same geographical area and the same area in the game. Each subgroup is associated with a local server. A local server is a server that treats the messages of its subgroup and calculates local updates. Then, it sends their updates to the subgroup in order the change the game's state. These updates are calculated locally without the central server's intervention. However, a global vision of the game and a synchronous view for all the players should be provided. Therefore, the local servers are still connected to the central server. The local servers send their new states to the central server from time to time. Afterwards, the central server will calculate the global game's state's update and send it to all the local servers. Thus, all the players get the same global view of the entire game world. Note that the traffic between the servers is less frequent than the traffic between the local servers and their subgroup members. We call the resulting architecture a "forest" and we represent it in figure 3.4.

According to the graph theory, a forest is defined as follows:

Definition 3 *A forest is a collection of trees which are not necessarily in-*

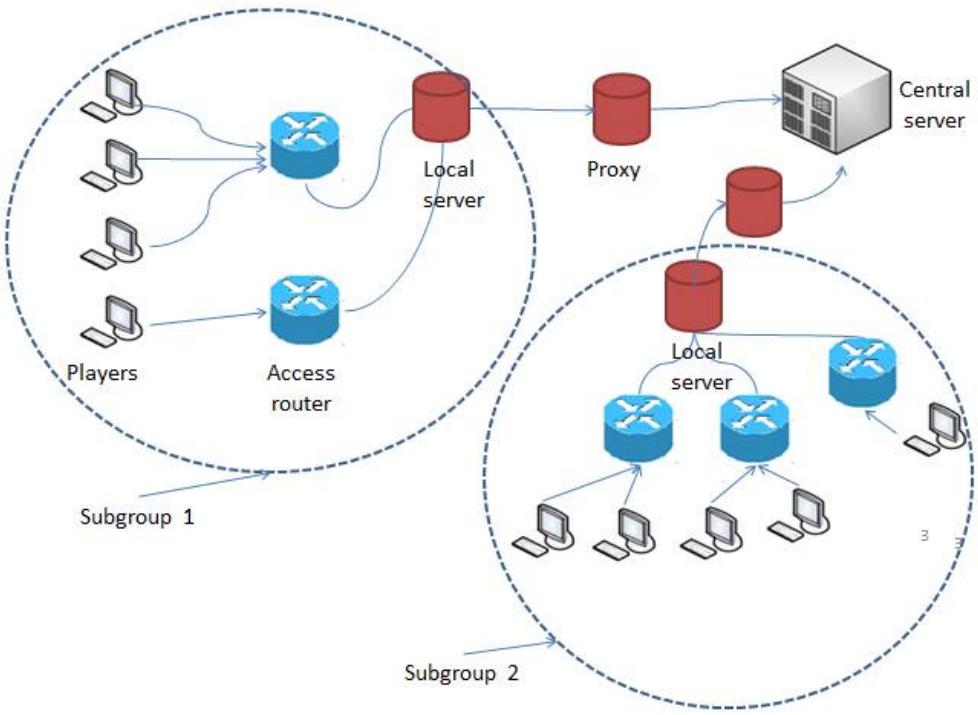


Figure 3.4: The forest topology for online games.

terconnected.

In this work, we only use interconnected subtrees which represent a tree from a topological point of view. However, each subtree connects a group of players chosen based on their location on the game's map. Since the traffic in one subtree may be different from another subtree, we will call this topology a forest with respect to the traffic independence.

The forest is then composed of multiple subtrees having the local servers as roots. These subtrees are connected to the central server which represents the root of the forest as shown in Figure 3.4.

Such a partition is expected to minimize the traffic to the central server and also to reduce the delay in local updates among the players of the same group. In the forest topology, there are four types of traffic:

1. Local unicast: from players in a group to their local server. This traffic is similar to unicast traffic for the tree.

2. Local multicast: from the local server to the players in the group. It has the characteristics of the multicast traffic in the tree topology.

3. Global unicast: from the local servers to the central server. In order to conserve the 80/20 rate relation between the local and the global traffic, we set up the local server to send an update packet about its group after having received the packets from 4 players. The size of the global unicast packet is calculated using (3.2) [23].

$$S_{server} = \begin{cases} 81.9 + 13 * (n-2) \text{ if } n > 2, \\ 81.9 + 13 \text{ else} \end{cases} \tag{3.2}$$

   where S_{server} is the size of the server packet in bytes and n is the total number of players

4. Global multicast: from the central server to all players. Once the server receives a global unicast packet, it generates the update packet whose size is given in (3.2).

3.4 Simulations and Results

In order to evaluate the impact of TCM on online games and to measure the effect of the group partition, we set up four simulation scenarios: 1) tree topology without TCM; 2) tree topology with the TCM; 3) forest topology without TCM and 4) forest topology with the TCM. For each input parameter, 10 simulations are run and the average of their results is taken.

3.4.1 Simulation setup

For simulations, we use the NS-3 simulator. The graphs are generated randomly using the GT-Itm graph generator. Then, a tree or forest topology is generated with the help of the Dijkstra algorithm, based on the shortest path calculation. Finally, the adjacency matrix is generated, to be used within NS-3. The used tree and forest topologies are given in Figure 3.5 and 3.6 respectively. Further, an FTP client is added to the NS-3 simulation. It generates a Poisson background traffic at the level of 50% of link capacities. The link capacities are set as follows. Access links are set to 1 Mbps, metro links to 5 Mbps and core links to 10 Mbps. These values have been chosen

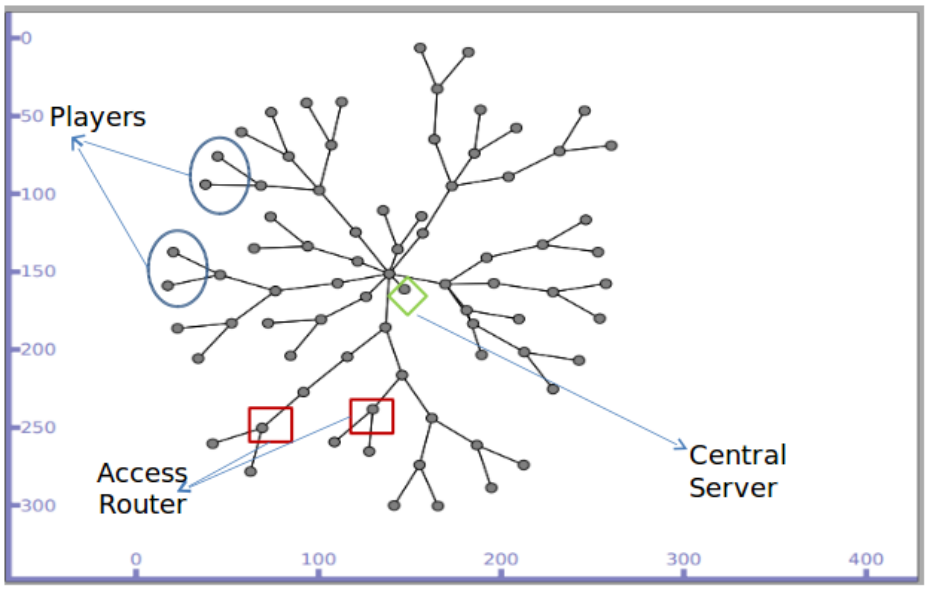


Figure 3.5: Tree topology with NS-3.

so that one could reach the network congestion when the number of players increases. The players send their packets periodically with a small random delay, representing the difference between their processors.

3.4.2 End-to-End delay

We investigate the impact of the TCM on the end-to-end delay when the number of players increases (and so does the traffic load). Fig. 3.7 summarizes our simulation results. The following comments can be made.

For tree topologies, the TCM induces a larger delay, given that there are only a few players in the game. In fact, in this case the throughput gain of the TCM is not very important. Besides, the additional delay caused by the multiplexing period T is added. When the number of players increases, the delay increases exponentially. However, it grows without the TCM faster than with the TCM. Indeed, the TCM allows to have a better performance at heavy network loads. Note that the difference in delay increases with the number of players. Also, note the big impact of the multiplexing period – the delay decreases for larger values of T. Clearly, this may not be a general trend, and one of the figures below will illustrate the delay behaviour w.r.t

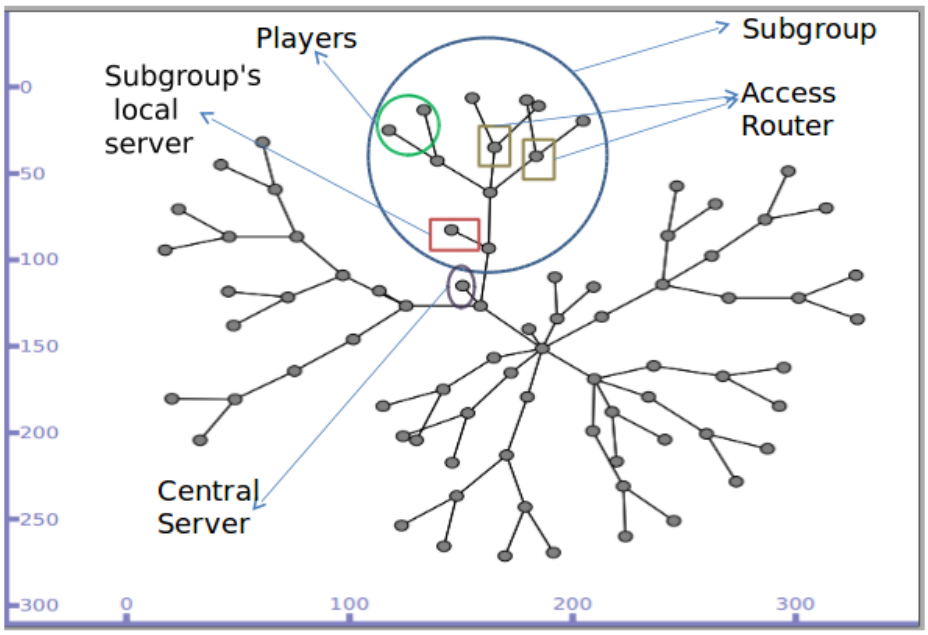


Figure 3.6: Forest topology with NS-3.

the multiplexing period.

As for forest topologies, the end-to-end delay is on average 400 ms lower than for the tree topology, even with the TCM. Moreover, we note the advantage of the partition onto the overall network load – for the same number of players, there is not much difference between the cases with and without the TCM because the traffic load has been greatly reduced, especially on the bottleneck links.

Finally, let us investigate the impact of the multiplexing period on the delay performance. Let the number of players be 500 and let us increase the value of the TCM period from 0 to 100 ms. Fig. 3.8 shows the behaviour of the delay w.r.t. the TCM period for a tree topology. It can be noted, that, when T increases, the delay decreases since the occupied bandwidth is reduced. However, T cannot grow infinitely. Indeed, at some point the delay saturates and becomes constant. This matches well with the result presented in [54]. Actually, when the period or the number of players is large, the bandwidth gain meets the following asymptotic limit:

$$BWR_a = \frac{MH + \mathbb{E}[RH] + \mathbb{E}[P]}{NH + \mathbb{E}[P]} \tag{3.3}$$

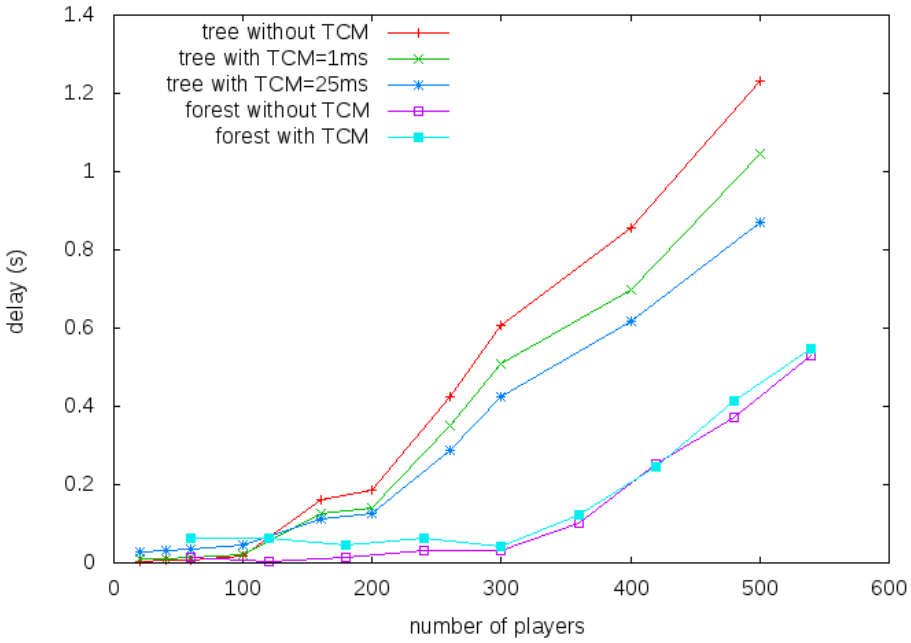


Figure 3.7: End-to-end delay vs. number of players, with/without TCM.

where BWR is the ratio of bandwidths with and without multiplexing, MH is the PPPMux header included at the beginning of each compressed packet, RH is the reduced header, NH is the UDP/IP header and P is the payload of initial (uncompressed) packets. Given all that was said above, there is no interest in multiplexing periods of more than $T = 50$ ms.

3.4.3 Jitter

Let us study the jitter – the second QoE parameter in online gaming. To do this, we evaluate the jitter variation with the number of online players, see Figure 3.9 for results. For tree topologies, the TCM traffic has a larger jitter for a small number of players. This is is due to the TCM multiplexing period as explained above. However, when the number of players is larger than 200, the jitter for the initial traffic becomes too important. Indeed, when there are many players, there are more successive packets, and the average jitter caused by the TCM decreases. Moreover, since the traffic throughput is reduced with the TCM, the jitter becomes smaller. As for the TCM, and

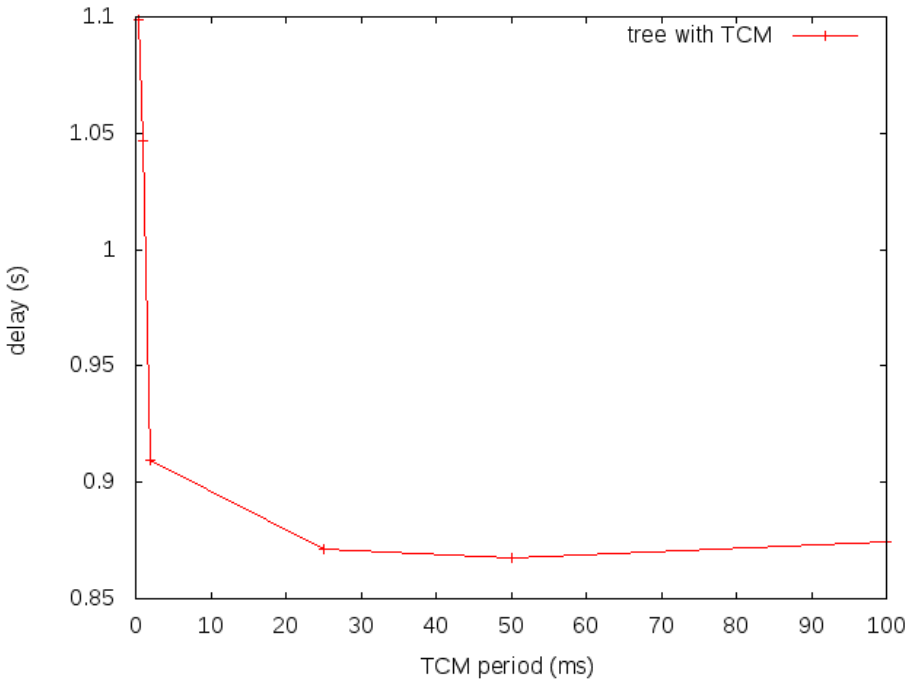


Figure 3.8: End-to-end delay vs. T.

with a small number of players, the larger T is, the larger the jitter is. The situation only changes when the number of players is more than 300, i.e., when the number of multiplexed packets becomes larger (see Figure 3.10). The jitter also experiences an asymptotic limit.

As for forest topologies, the jitter behaves even better than for a tree topology with the TCM for a large number of players. With the TCM, the jitter is very low, and thus the QoE of players is high.

3.4.4 Arrival order of packets

Since consistency is a critical requirement for online players, we are interested in evaluating the number of out-of-order packets delivered by the network. Here we assume that the inconsistency is due to the packets displayed in the wrong order. First, we evaluate the number of packets delivered out of order as the number of players increases. Fig. 3.11 shows that the number of out-of-order packets increases with the number of players for the tree

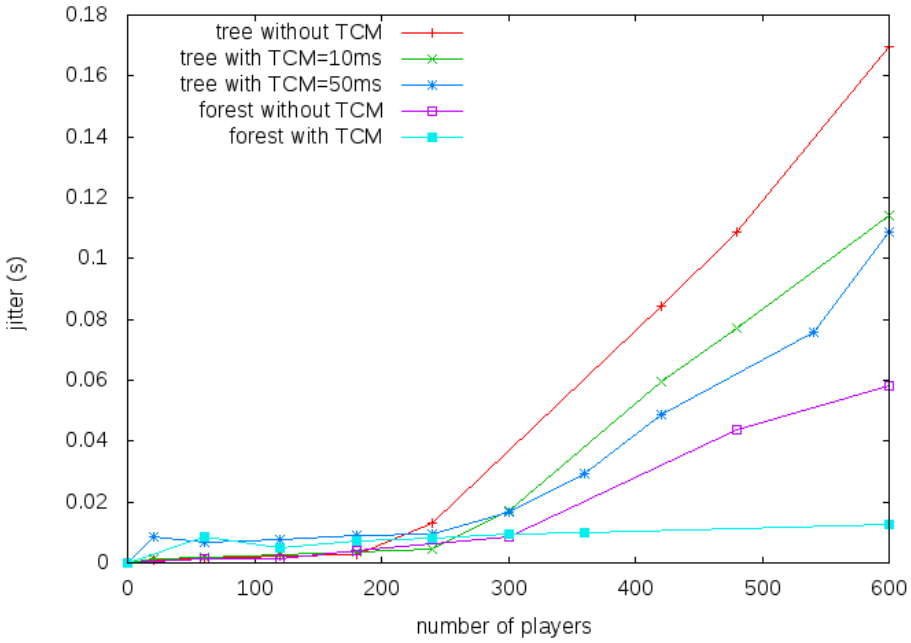


Figure 3.9: Jitter variation vs. the number of players.

topology. Even though the TCM reduces the traffic load, its number of out-of-order packets is always larger than with the initial traffic. This result can be explained by the fact that for the TCM, if a packet is not received by the server in a good order, it will be the same for all the native packets multiplexed in that packet, thus leading to a burst of out-of-order packets. For forest topologies, the group partition can considerably decrease the rate of out-of-order packets, and the situation gets even better with the TCM. This difference between the tree and the forest topology is explained by the difference in the network load.

Next, let us set up the number of players to 200 and simulate the rate of out-of-order packets for various TCM periods. Fig. 3.12 shows that the disorder increases with T. In fact, with the TCM, the disorder is important since one out-of-order packet implies the disorder of all the multiplexed packets. Moreover, for a larger T, more packets are multiplexed to one packet. Thus, the disorder increases with T and reaches the case where almost all the packets are out-of-order, requiring an important effort of reordering which induces an additive processing delay.

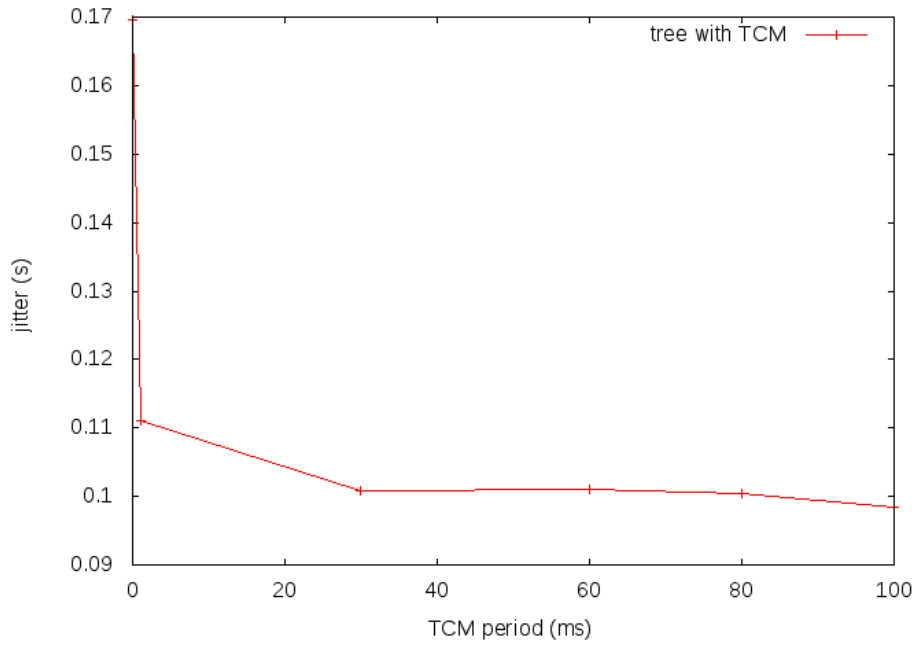


Figure 3.10: Jitter as a function of the TCM period T for 600 online players.

3.5 Conclusion

In this chapter we investigated the effect of packet multiplexing on the QoE of online multiplayer games. The three major QoE metrics have been studied: delay, jitter and packets order. Our simulations show that the modified TCM behaves better than the simple routing in terms of delay and of jitter when the number of players is large. However, the TCM behaves worse in terms of disorder. Moreover, we studied the scenario when, in addition to TCM, the online players are partitioned into groups based on their location on the game's map. This led us to introduce the so-called forest topology. The use of the forest topology decreases the delay and the disorder, and thus improves the QoE of online game for a large number of players.

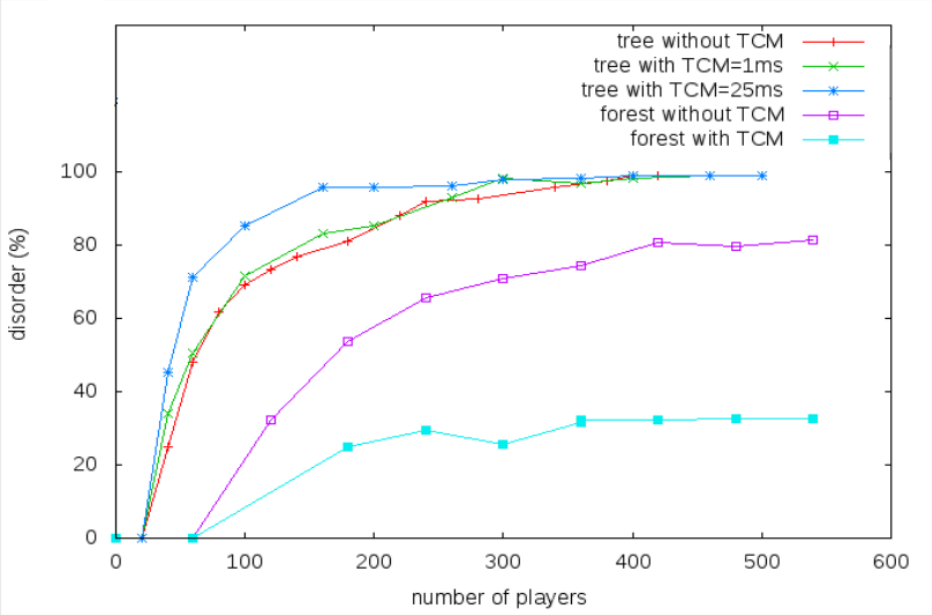


Figure 3.11: Number of out-of-order packets vs. the number of players.

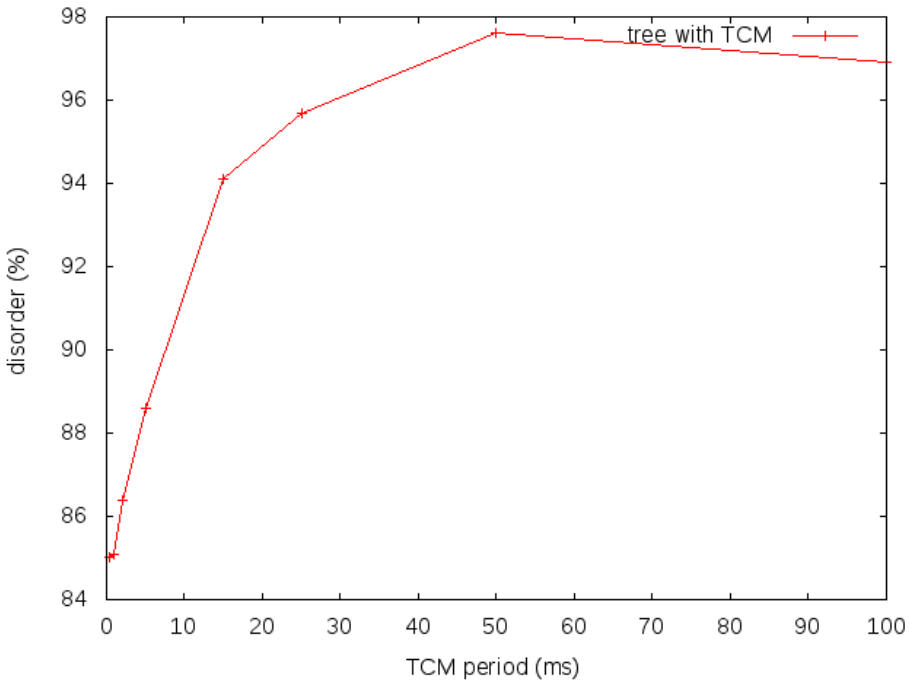


Figure 3.12: Variation of the number of out-of-order packets vs. T.

Part III

Network coding for online gaming

Chapter 4

Network coding for online video gaming over a cyclic network topology

Contents

4.1 Introduction

In the previous chapter, we studied TCM, a tree-based solution to enhance the online games' throughput and reduce their latency, then we studied TCM coupled with the forest topology after dividing the players. In this chapter, we will also divide the players into subgroups but we will connect the local servers to their subgroups using a cycle topology. Besides, we will apply the network coding technique over the cycle in order to reduce the transmission delay and increase the network efficiency and the traffic throughput.

The topology interconnecting the local server with the central server is not the focus of our work. We are interested in the design and the performance evaluation of an adequate routing protocol for online games over a cycle.

As we will only treat the traffic of one subgroup and for the sake of simplicity, we will refer to the local server by "the server" for the rest of the chapter.

This work in an object of one journal article [109].

4.2 System model

For the sake of simplicity, the network with a single cycle corresponding to one proximity-defined logical subnetwork within a game will be considered. The designed protocol will focus only on the routing of the data packets. The construction and the control of the cycle topology are outside the scope of this work. In [110], a sequential algorithm for constructing multi-cycles for massive group communications under delay constraint is proposed. This algorithm can serve as an inspiration for our case.

Let the logical network of an online game contain a single cycle representing one subgroup of the players, connecting the local game server S and n game players located in the same proximity on the game map. It is represented by a cycle $G = (V, E)$ as illustrated in Figure 4.1 with the set of nodes $\mathcal{V} = \{V_0 = S, V_1, \ldots, V_n\}$, and the set of links E. For simplicity, let the network be homogenous so that each link $e \in E$ may transmit one packet per time unit (t.u.). Online game traffic is periodic with *a communication period* T [t.u.] and contains $n + 1$ flows as follows:

- a broadcast flow from V_0 to all $V_i \in \mathcal{V} \backslash V_0$ to communicate the current game instance;

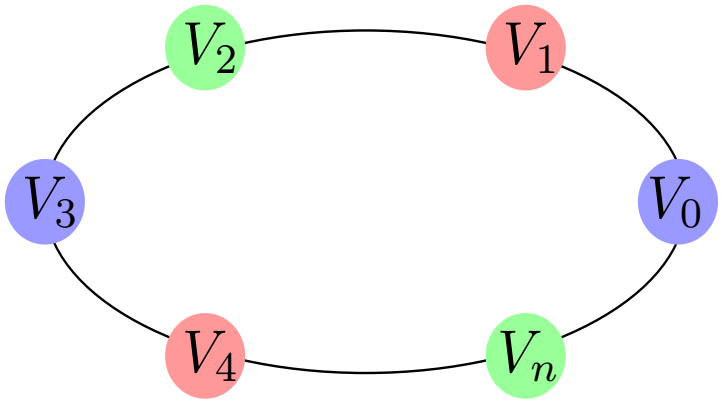


Figure 4.1: A cycle topology with $n = 5$ and $V_0 = S$. Node subsets $\mathcal{V}_1$, $\mathcal{V}_2$ and $\mathcal{V}_3$ are given by blue, red and green.

- n unicast flows from any $V_i \in \mathcal{V} \backslash V_0$ to V_0 carrying the players' actions.

Also, let L denote the total number of packets transmitted through the network during one *communication period* T.

The directions of the $n + 1$ flows described above imply two source-destination sets:

- $\mathcal{S}_1 = \{V_0\}$ and $\mathcal{D}_1 = \{V_1, \ldots, V_n\}$: the node V_0 broadcasts a common message M_0 to the set $\mathcal{D}_1$;
- $\mathcal{S}_2 = \{V_1, \ldots, V_n\}$ and $\mathcal{D}_2 = \{V_0\}$: each node V_i in S_2 has a private message M_i to send to V_0.

Given n, $\mathcal{S}_1$, $\mathcal{D}_1$, $\mathcal{S}_2$ and $\mathcal{D}_2$ defined above, we distinguish the *routing problem* which aims to find a routing $\mathcal{R}$ minimizing the couple (T, L), and the *network coding (NC) problem* which consists in finding a network code $\mathcal{C}$ and a routing protocol $\mathcal{R}_\mathcal{C}$ minimizing (T, L). Finally, for later use, let $D \triangleq \lceil n/2 \rceil$ and $d \triangleq \lfloor D/2 \rfloor = \lfloor \frac{n+1}{4} \rfloor$.

Given this system model, we need an efficient routing protocol that minimizes T and the number of packets sent L.

We propose a transmission protocol inspired by the wireless network case. We consider a general transmission case where packets are eligible to collisions. Hence, when a node V_i sends a packet, the neighbor nodes V_{i-1} and V_{i+1} (the nodes located in V_i's coverage area as shown in Figure 4.1) must neither receive nor send a packet in order to avoid collisions.

This scheduling is organized by the server by initializing the transmission itself with a special message M_{00} carrying all the information needed for the nodes to establish their transmission pattern. Afterwards, every node will follow a synchronized pattern and alternate between three states: sending, receiving from the right and receiving from the left.

The transmission protocol rules are summarized as follows:

- when a node V_i sends a packet, it broadcasts it to its two closest neighbors $V_{(i-1)}$ and $V_{(i+1)}$;
- the system is half duplex, so each node is either in the receiving or in the sending state during one time unit;
- a node cannot receive two simultaneous messages from its neighbors in order to avoid collisions.
- we can find intermediate nodes which are not game players but are needed to relay between the cycle nodes.

Considering the transmission rules defined above, the best transmission schedule with the highest collision avoidance is determined as follows [98]. If $n+1$ is divisible by 3 ($(n+1)|3$), then the set of nodes V is divided into 3 disjoint subsets $\mathcal{V}_1$, $\mathcal{V}_2$ and $\mathcal{V}_3$ with the condition that a node from a subset $\mathcal{V}_i$ is at least 3 hops away from any other node from the same subset. Figure 4.1 illustrates this condition where each color represents nodes that can be member of the same subset. Afterwards, the transmission is organized in *rounds*, each lasting 3 t.u. where every $\mathcal{V}_i$ is allowed to broadcast during one t.u. in order to avoid collisions.

Remark 2 *If $(n+1) \not| 3$, then 4 disjoint groups $\mathcal{V}_1, \ldots, \mathcal{V}_4$ are formed as follows.*

Let $r = (n+1)$ mod 3 and let $\mathcal{V}_4 = \emptyset$ if $r = 0$, $\mathcal{V}_4 = \{V_{\lceil n/2 \rceil}\}$ if $r = 1$ and $\mathcal{V}_4 = \{V_{\lfloor n/2 \rfloor}, V_{\lceil n/2 \rceil}\}$ if $r = 2$. Then

$$\begin{aligned}
\mathcal{V}_1 = &(\{V_i : i \bmod 3 = 0, i \leq \lfloor n/2 \rfloor\} \cup \\
&\{V_i : i \bmod 3 = r, i > \lfloor n/2 \rfloor\}) \setminus \mathcal{V}_4 \\
\mathcal{V}_2 = &(\{V_i : i \bmod 3 = 1, i \leq \lfloor n/2 \rfloor\} \cup \\
&\{V_i : i \bmod 3 = (r+1) \bmod 3, i > \lfloor n/2 \rfloor\}) \setminus \mathcal{V}_4 \\
\mathcal{V}_3 = &(\{V_i : i \bmod 3 = 2, i \leq \lfloor n/2 \rfloor\} \cup \\
&\{V_i : i \bmod 3 = (r+2) \bmod 3, i > \lfloor n/2 \rfloor\}) \setminus \mathcal{V}_4
\end{aligned}$$

Also, Algorithm 3 is modified accordingly: the round $t = 0$ lasts 4 t.u. (each $\mathcal{V}_i$ broadcasts during 1 t.u. and stays silent during 3 t.u.). Thanks to the careful choice of $\mathcal{V}_4$, the rounds for $t > 0$ stay unchanged. Note that, if $|\mathcal{V}_4| = 2$, it seems there will be be a collision in transmitting $M_{\lfloor n/2 \rfloor}$ and $M_{\lceil n/2 \rceil}$ at $t = 0$. However there is no loss for $M_{\lfloor n/2 \rfloor}$ and $M_{\lceil n/2 \rceil}$, as they are successfully received by $V_{\lfloor n/2 \rfloor - 1}$ and $V_{\lceil n/2 \rceil + 1}$ respectively.

4.3 Routing protocols

In this section, we present three different routing protocols over cycle. The first one is proposed in the literature and is designed based on NC for a multicast traffic case. The second and the third protocols are our propositions. As for the second, it is a shortest-path-based routing protocol for online games over a cycle. Finally , the third protocol is an enhanced version of the second that we proposed in which the network coding technique is applied.

4.3.1 NC-based multicast routing protocol

4.3.1.1 Description

The state-of-the-art of network-coded communication protocols over cyclic topologies is mostly based on the multicast scenario, where each node V_i has a message M_i to send to all other nodes in the network, for all possible values of i. In this case the use of NC is beneficial. Let us illustrate it on Examples 1 and 2 below, one with and one without NC.

Example 1 *Consider the circular routing for multicast over a single-cycle network with $n + 1$ nodes $V_0, \ldots, V_n$. Here the messages are forwarded over the cycle in one direction (clockwise or counter-clockwise), following the 3-phase or 4-phase transmission schedule described above. The transmission continues until any message M_i reaches all the nodes in $V \backslash V_i$.*

In this scenario, a routing protocol based on Network Coding was proposed and a gain of nearly 50% of the communication period was proven [98]. This leads us to the second example.

Example 2 *Consider the multicast problem over the single-cycle network, where the nodes are allowed to perform NC operations. Let Algorithm 2 below be used. Note that the algorithm was first described in [98] and here we give it when* $(n+1)|3$ *only, for the sake of simplicity.*

Algorithm 2 Multicast with NC [98] when $(n+1)|3$

Initialisation: V_i is allocated to the subset $\mathcal{V}_j$, $j = (i \mod 3)+1$, $0 \leq i \leq n$. Each V_i has a message M_i to multicast.

For $0 \leq t \leq \lceil n/2 \rceil$**, perform the 3-phase transmission:**

During 3 t.u., a node V_i $(0 \leq i \leq n)$ does the following (the order of operations depends on its subset index j):

1) Reception of a message from the right $M^{\rightarrow}_{i+1}(t)$;
2) Reception of a message from the left $M^{\leftarrow}_{i-1}(t)$;
3) Broadcast of the current message $M_i(t)$, where

$$M_i(t) = \begin{cases} M_i, & t = 0; \\ M^{\leftarrow}_{i-1}(t-1) \oplus M^{\rightarrow}_{i+1}(t-1), & t > 0. \end{cases}$$

4.3.1.2 Performance regarding delay and transmitted packets

The following Lemma1 and Lemma2 summarize the performance of the multicast algorithm with and without NC respectively in terms of communication period T size and the number of transmitted packets L.

Lemma 1 *The minimum communication period* T *for Example 1 is bounded as* $3n \leq T \leq 4n$*, while the total number* L *of messages to transmit over the network is* $L \leq \lfloor (n+1)/3 \rfloor T$*.*

Proof. If $(n+1)|3$, then exactly n rounds of the 3-phase transmission are needed so that M_i reaches all its destinations. Otherwise, n rounds of the 4-phase transmission will be used. As for the result on L, note that at most $\lfloor (n+1)/3 \rfloor$ messages are sent at each t.u.

Lemma 2 *Given the NC-based multicast protocol from Example 2, the minimum communication period* T_{NC} *is bounded as* $3\lceil \frac{n}{2} \rceil \leq T_{NC} \leq 4\lceil \frac{n}{2} \rceil$*, and the number of messages* $L_{NC} \leq \lfloor (n+1)/3 \rfloor T_{NC}$*.*

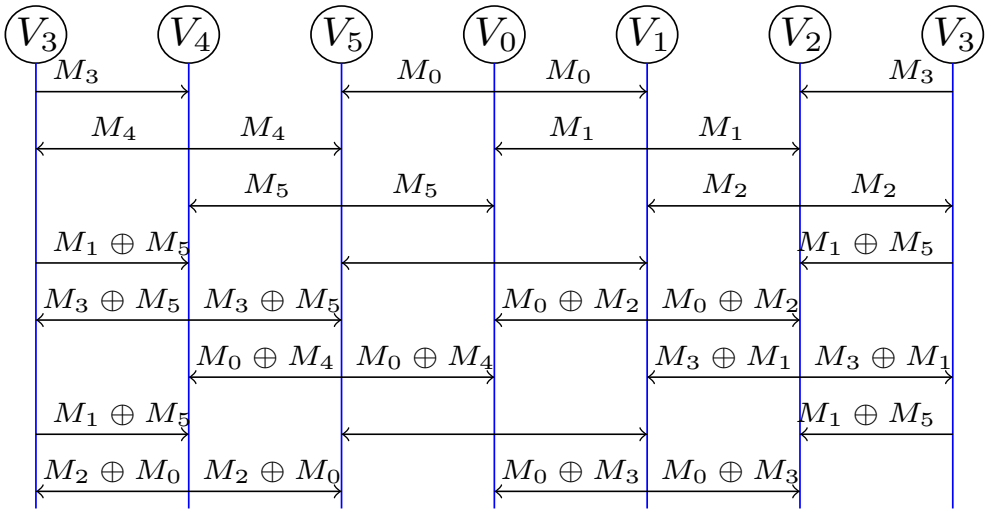


Figure 4.2: A Communication Period with the Multicast-NC protocol for $n = 5$.

Proof. At round $t = 0$, V_i receives M_{i-1} from the left and M_{i+1} from the right[1]. At round $t > 0$, V_i possesses already the messages M_{i-t} and M_{i+t}. V_i receives $M_{i-1}(t) = M_{i-t-1} \oplus M_{i-t+1}$ and $M_{i+1}(t) = M_{i+t-1} \oplus M_{i+t+1}$. So it decodes M_{i-t-1} and M_{i+t+1} by XOR-ing: $M_{i\pm t\pm 1} = M_{i\pm 1}(t) \oplus M_{i\pm t}$. At round $t = \lceil n/2 \rceil$, V_i receives the last missing message from its farthest node(s). The calculation of lower and upper bounds on T_{NC}, as well as the upper bound on L_{NC} follow directly from this procedure.

Note that the NC-based multicast routing protocol can be applied to online games and it reduces the communication period as explained in Lemma2. However, as this traffic is not fully multicast and contains also unicasts, it is possible to find a more efficient routing protocol for online games that takes into account these specificities. Therefore, we propose first an optimized routing protocol based on the shortest-path routing approach and second an enhanced version of this protocol by applying the Network Coding when possible. This protocol also minimizes the number of transmitted packets, which reduces the load of the links. In the following, we will give a description of these two protocols.

[1]here and below the index addition/subtraction is performed modulo $(n + 1)$

Algorithm 3 Routing protocol when $(n+1)|3$

Initialisation: V_i is allocated to the subset $\mathcal{V}_j$, $j = i \mod 3$, $0 \leq i \leq n$. Each V_i has a message M_i to transmit.

3-phase round for $t = 0$: A node V_i $(0 \leq i \leq n)$ performs the operations as in Algorithm 2 for $t = 0$.

4-phase round for $1 \leq t \leq d$:

During first 3 t.u., at each time unit ℓ $(\ell = 1, 2, 3)$ the set $\mathcal{V}_i$ broadcasts while the other sets are silent. Moreover,

- if $V_i \in S_{\rightarrow} = \{V_0, \ldots, V_{D-t}\} \backslash V_0$, it broadcasts $M_i(t) = M_{i+1}^{\rightarrow}(t-1)$;
- if $V_i \in S_{\leftarrow} = \{V_{n-D+t+1}, \ldots, V_{n+1} = V_0\} \backslash V_0$, it broadcasts $M_i(t) = M_{i-1}^{\leftarrow}(t-1)$.

During the last t.u., nodes V_t and V_{n+1-t} broadcast M_0.

3-phase round for $d+1 \leq t < D$:

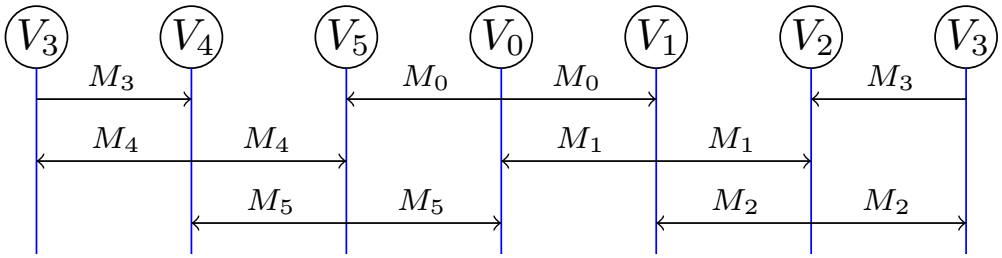
During first 3 t.u., at each time unit ℓ $(\ell = 1, 2, 3)$ the set $\mathcal{V}_i$ broadcasts while the other sets are silent. Moreover,

- if $V_i \in S_{\rightarrow} = \{V_0, \ldots, V_{D-t}\} \backslash V_0$, it broadcasts $M_i(t) = M_{i+1}^{\rightarrow}(t-1)$;
- if $V_i \in S_{\leftarrow} = \{V_{n-D+t+1}, \ldots, V_{n+1} = V_0\} \backslash V_0$, it broadcasts $M_i(t) = M_{i-1}^{\leftarrow}(t-1)$;
- V_t and V_{n+1-t} broadcast M_0.

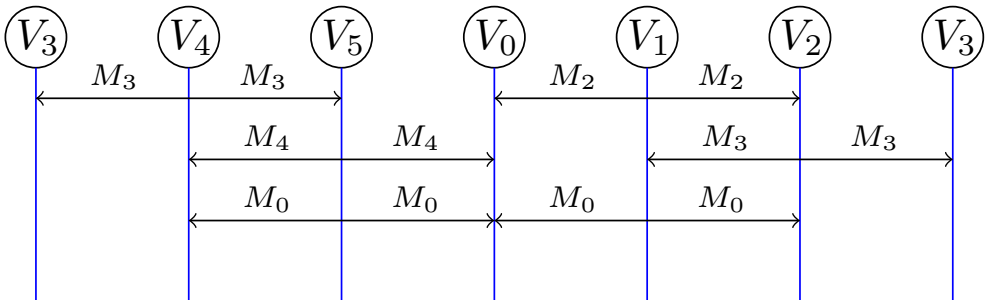
4.3.2 Shortest-path routing protocol

4.3.2.1 Description

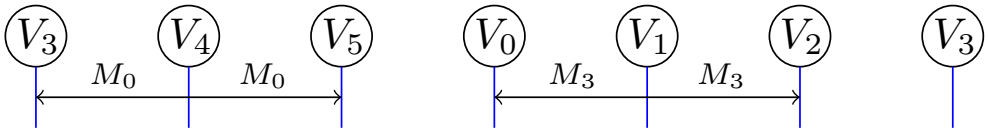
We propose Algorithm 3 as a routing algorithm without NC for online gaming. It is in fact an optimised version of the *shortest-path routing* algorithm. For simplicity, the algorithm is described when $(n+1)|3$. For an illustration, Fig.4.3 presents a time diagram for $n = 5$ ($D = 3$ and $d = 1$). At $t = 0$, the nodes send their own messages in 3 t.u. At $t = 1$, V_0 and V_3 remain silent, V_1 and V_5 send two messages each, and V_2 and V_4 send messages of their neigbours. During $t = 2$, the nodes $V_2, \ldots, V_5$ should remain silent, except



(a) Round $t = 0$. V_0 receives M_1 and M_5. M_0 is received by V_1 and V_5. At this instance all nodes possess their own messages.



(b) Round $t = 1 = d$. V_0 and V_3 do not transmit. V_0 receives M_2 and M_4. M_0 is received by V_2 and V_4.



(c) Round $t = 2$ ($t > d$). V_0, V_2, V_3 and V_5 do not transmit. V_0 receives M_3. M_0 is received by V_3. End of the communication period T.

Figure 4.3: Example for $n = 5$. Similar to Fig.4.1, $\mathcal{V}_1 = \{V_0, V_3\}$, $\mathcal{V}_2 = \{V_1, V_4\}$, $\mathcal{V}_3 = \{V_2, V_5\}$.

V_4 which communicates M_0 to V_3. V_1 forwards M_3 to V_0.

4.3.2.2 Performance

Theorem 3 *The period T of Algorithm 3 is bounded as*
$3\lceil n/2 \rceil + \lfloor \frac{n+1}{4} \rfloor - 2 \leq T \leq 3\lceil n/2 \rceil + \lfloor \frac{n+1}{4} \rfloor + 1$,
and $L = \lceil n/2 \rceil (\lfloor n/2 \rfloor + 3) - 1$.

Proof. Let $(n+1)|3$. By shortest-path routing over the cycle with $n+1$ nodes, a message M_i will be received by a destination in at most D hops. The hops are part of the rounds of Algorithm 3, thus, for $t = 0$, 3 t.u. will be used

for one hop. The total number of messages sent at $t = 0$ is $n + 1$. Moreover, for $1 \leq t \leq d$, the nodes V_i with $1 \leq i \leq d$ and with $n+1-d \leq i \leq n$ have two messages to forward: a message M_j, $j \in \{i+1, \ldots, i+d\} \cap \{i-d, \ldots, i-1\}$, and M_0. These nodes will use one additional t.u. so all the rounds will last 4 t.u. Note that the rest of nodes will be silent as they have no new messages to send. As for the rounds with $d+1 \leq t < D$, M_0 is now to be transmitted by nodes with indices in $\{d+1, \ldots, \lceil n/2 \rceil - 1\} \cap \{\lfloor n/2 \rfloor + 2, \ldots, n-d\}$. These nodes have no other messages to forward, thus they send M_0 within 3 t.u. during which the nodes V_i with $1 \leq i \leq d$ and with $n+1-d \leq i \leq n$ forward messages to V_0. Also, it can be shown that at any round $1 \leq t < D$, $n + 2 - 2t$ messages will be sent in total. Finally, if $(n+1) \not| 3$, one extra t.u. will be used at $t = 0$, and the rest of the protocol will be unchanged. This gives us the upper bound on T, $T \leq 4d + 3(D - d - 1) + 4$, as well as $L = n + 1 + \sum_{t=1}^{D-1}(n + 2 - 2t)$.
Also, if for $t = D - 2$ and $t = D - 1$ there are many silent nodes in the network, and the simultaneous transmission by nodes from different subsets does not create collisions, one can save up to 3 t.u. by a careful transmission scheduling.

4.3.3 NC-based routing protocol

4.3.3.1 Description

Let the nodes perform NC operations. Then the routing protocol above can be modified for rounds $1 \leq t \leq d$, as it is stated in Algorithm 4. To illustrate the new protocol, let us again consider the example in Fig. 4.3. The new protocol only modifies the transmission at $t = 1$ (see Fig.4.4), and allows to save 1 t.u. due to NC operations. More generally:

Theorem 4 *For Algorithm 4, one has* $3\lceil n/2 \rceil - 2 \leq T_{NC} \leq 3\lceil n/2 \rceil + 1$, *and* $L_{NC} = \lceil n/2 \rceil (\lfloor n/2 \rfloor + 3) - 2\lfloor \frac{n+1}{4} \rfloor - 1$.

Proof. Owing to NC operations, the nodes, having two messages to forward, send their XORs. Note that the nodes receiving XORs are always able to decode new messages. Thus the rounds with $1 \leq t \leq d$ last 3 t.u. instead of 4, and the number of transmitted messages is decreased by 2 in each round. By counting, as in the proof of Theorem 3, one obtains results on T_{NC} and L_{NC}.

Algorithm 4 NC-based protocol when $(n+1)|3$

Initialisation: V_i is allocated to the subset $\mathcal{V}_j$, $j = i \mod 3$, $0 \leq i \leq n$. Each V_i has a message M_i to transmit.

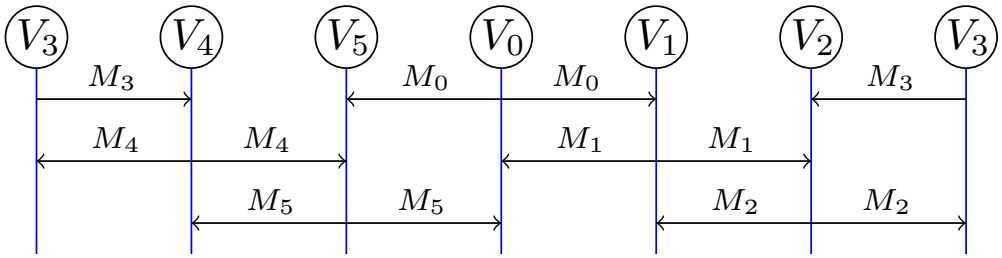
3-phase round for $t = 0$: A node V_i $(0 \leq i \leq n)$ performs the operations as in Algorithm 2 for $t = 0$.

3-phase round for $1 \leq t \leq d$: At each time unit ℓ $(\ell = 1, 2, 3)$ the set $\mathcal{V}_i$ broadcasts while the other sets are silent.

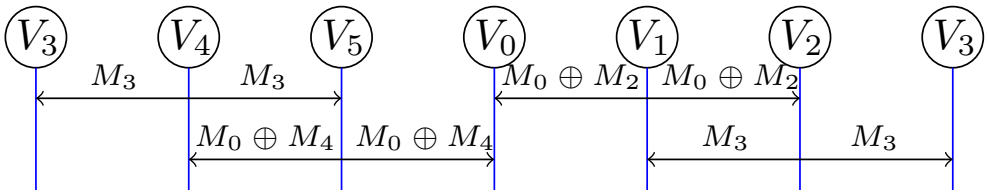
- if $V_i \in S_{\oplus} = \{V_t, V_{n+1-t}\}$, it broadcasts $M_i(t) = M_{i-1}^{\leftarrow}(t-1) \oplus M_{i+1}^{\rightarrow}(t-1)$;
- if $V_i \in S_{\rightarrow} = \{V_0, \ldots, V_{D-t}\} \backslash V_0$, it broadcasts $M_i(t) = M_{i+1}^{\rightarrow}(t-1)$;
- if $V_i \in S_{\leftarrow} = \{V_{n-D+t+1}, \ldots, V_{n+1} = V_0\} \backslash V_0$, it broadcasts $M_i(t) = M_{i-1}^{\leftarrow}(t-1)$.

3-phase round for $d+1 \leq t < D$:
During first 3 t.u., at each time unit ℓ $(\ell = 1, 2, 3)$ the set $\mathcal{V}_i$ broadcasts while the other sets are silent. Moreover,

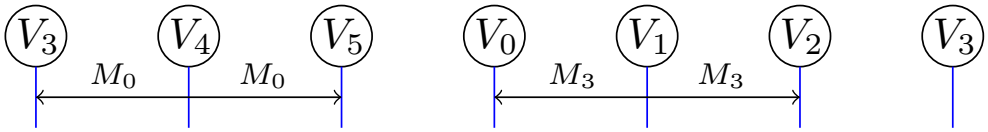
- if $V_i \in S_{\rightarrow} = \{V_0, \ldots, V_{D-t}\} \backslash V_0$, it broadcasts $M_i(t) = M_{i+1}^{\rightarrow}(t-1)$;
- if $V_i \in S_{\leftarrow} = \{V_{n-D+t+1}, \ldots, V_{n+1} = V_0\} \backslash V_0$, it broadcasts $M_i(t) = M_{i-1}^{\leftarrow}(t-1)$;
- V_t and V_{n+1-t} broadcast M_0.



(a) Round $t = 0$. V_0 receives M_1 and M_5. M_0 is received by V_1 and V_5. At this instance all nodes possess their own messages.



(b) Round $t = 1 = d$. V_0 and V_3 do not transmit. V_0 receives M_2 and M_4 (encoded). M_0 is received by V_2 and V_4.



(c) Round $t = 2$ ($t > d$). V_0, V_2, V_3 and V_5 do not transmit. V_0 receives M_3. M_0 is received by V_3. End of the communication period T.

Figure 4.4: Example of using the NC protocol for $n = 5$ with $\mathcal{V}_1 = \{V_0, V_3\}$, $\mathcal{V}_2 = \{V_1, V_4\}$, $\mathcal{V}_3 = \{V_2, V_5\}$.

Remark 3 *Both Algorithms 3 and 4 can be easily adapted to imperfect transmission conditions, namely to erasures of messages during the transmission.*

Note that, if a receiving node treats an erased message as a trivial all-0's packet, the sending node auto-detects that the message was lost and can retransmit it at the next round. This is true for all receiving nodes except V_0, $V_{\lfloor n/2 \rfloor}$ and $V_{\lceil n/2 \rceil}$; these exceptions are to be treated separately. Thus a message erasure would create an extra delay of one round but would not corrupt the functioning of the whole transmission scheme.

To guarantee that our protocol remains functional even in case of packet loss and transmission delay variation, each state will take exactly one t.u. (a timeout). If during this time no packet is received, then the packet is considered lost and is replaced by an all-zero packet. The recovery of lost packets is assumed to be handled by the reliable transport layer protocols.

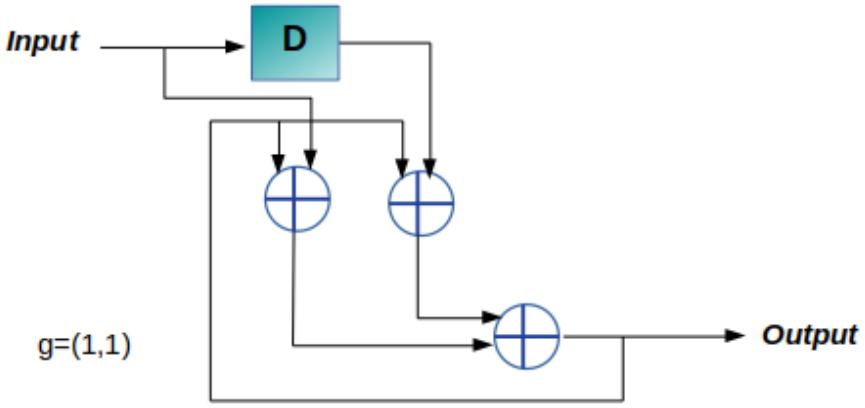


Figure 4.5: A convolutional code representation of NC-based multicast protocol.

Hence, in case of a lossy network, the end of the *communication period* can be detected by a second factor. As the nodes may not receive all the packets they are supposed to receive due to packet loss, one should add a second possible condition to stop the current communication period and start a new one. This condition is reaching a maximum communication period size that corresponds to the maximum size of T in the case of a lossless network (i.e the maximum number of $t.u.$ during one T corresponding to n client nodes).

Remark 4 *Note that the NC-based multicast algorithm can be considered as a non-systematic recursive convolutional code of rate 1 (m/n where m is the number of the old inputs used and n is the number of the output symbols) and with constraint length k=2 as shown in Figure 4.5.*

If we note the input sequence as u and the output sequence as v then the generator sequence corresponding to this code is given as: $g = (1, 1)$ verifying $v = u.g$ and the generator matrix G verifying the equation $v = u.G$ is as follows:

$$G= \begin{pmatrix} 0 & 0 & 0 & ..... & & & & \\ 0 & 1 & 1 & 0 & ..... & & & \\ 0 & 0 & 1 & 1 & 0 & ..... & & \\ ... & ... & 0 & 1 & 1 & 0 & ..... & \\ ... & ... & ... & 0 & 1 & 1 & 0 & ..... \\ ... & ... & ... & ... & ... & ... & ... & ..... \end{pmatrix} \tag{4.1}$$

We notice that the first row is an all-zero row which corresponds to the initialization period where the nodes send their own packets and not the received ones. As for the optimized routing protocol, it has no coding operations. It uses the memorized information but sends them in their original form. However, we can represent it as a convolutional code of rate 1 and constraint length $k = 2$. The corresponding generator sequence in this case is either $g = (0, 1)$ or $g = (1, 0)$ depending on the group to which the concerned node belongs.

The NC-based routing protocol that we finally developed is a combination of both of these codes with three possible cases: coding with $g_a = (1, 1)$, sending without coding with $g_b = (0, 1)$ or $g_b = (1, 0)$ or not sending at all or sending its own packet independently of the input packets (during the first period $t = 0$), corresponding to a $g_c = (0, 0)$. Each case can occur for a certain number of time slots. The resulting code is then a rate 1 non-systematic, non recursive time-variant convolutional code with a constraint length k=2. Let's note the set of periods t where the generating vectors g_a, g_b and g_c is used by A, B and C respectively. As a result, $A = T_a$, $B = \{\{1, ..., T_b\} \cup \{T'_b\}\}\backslash\{T_a\}$ and $C = \{\{T_b + 1, ..., D - 1\} \cup \{0\}\}\backslash\{T_b\}$ where the values T_a, T_b, T'_b and T_c represent the limits of time slots using each of the vectors g_a, g_b and g_c respectively. The corresponding generating matrix has the following general

subset	T_a	T_b	T_b'	T_c
P_{1-1}	i	$\lfloor n/2 \rfloor - i + 1$	/	$D - T_b$
P_{1-2}	$(n+1) - i$	$i - \lceil n/2 \rceil$	/	$D - T_b$
P_{2-1}	/	$\lfloor n/2 \rfloor - i + 1$	i	$D - T_b$
P_{2-2}	/	$i - \lceil n/2 \rceil$	$(n+1) - i$	$D - T_b$

Table 4.1: T_a, T_b, T_b' and T_c values depending of the node index i and its subset

form:

$$
\mathrm{G} = \begin{array}{c} \\ 0 \\ 1 \\ . \\ . \\ T_a \\ . \\ . \\ T_b \\ . \\ . \\ . \\ . \\ T_c \end{array}
\begin{pmatrix}
g_c^{(1)} & g_c^{(2)} & 0 & .... & & & & & & & & & \\
0 & g_b^{(1)} & g_b^{(2)} & 0 & ... & & & & & & & & \\
0 & 0 & g_b^{(1)} & g_b^{(2)} & 0 & .. & & & & & & & \\
0 & 0 & 0 & g_b^{(1)} & g_b^{(2)} & 0 & ... & & & & & & \\
0 & 0 & 0 & 0 & g_a^{(1)} & g_a^{(2)} & 0 & ... & & & & & \\
... & ... & .. & ... & ... & ... & ... & ..... & & & & & \\
... & ... & .. & ... & ... & ... & ... & ..... & & & & & \\
0 & 0 & . & . & 0 & g_b^{(1)} & g_b^{(2)} & 0 & ..... & & & & \\
0 & 0 & . & . & . & 0 & g_c^{(1)} & g_c^{(2)} & 0 & ..... & & & \\
0 & 0 & . & . & . & . & 0 & g_c^{(1)} & g_c^{(2)} & 0 & ..... & & \\
... & ... & .. & ... & ... & ... & ... & ..... & & & & & \\
... & ... & .. & ... & ... & ... & ... & ..... & & & & & \\
0 & 0 & . & . & . & . & . & . & 0 & g_c^{(1)} & g_c^{(2)} & 0 & ..
\end{pmatrix}
$$

The values T_a, T_b, $T - b'$ and T_c depend of the node's index i. We note that the generating matrix of the server, as well as the nodes $V_{\lfloor n/2 \rfloor}$ and $V_{\lceil n/2 \rceil}$ have a special form. As they do not resend the received packets, they only use the g_c generating vector resulting in an all-zero-matrix. To illustrate the difference between the different nodes, we can divide the cycle into 4 subsets of nodes $P_{1-1} = \{V_1, ..., V_{\lfloor n/4 \rfloor}\}$, $P_{1-2} = \{V_{\lceil n/4 \rceil}, ..., V_{\lfloor n/2 \rfloor - 1}\}$, $P_{2-1} = \{V_{\lceil n/2 \rceil} + 1\}, ..., V_{\lfloor 3n/4 \rfloor}$ and $P_{2-2} = \{V_{\lceil 3n/4 \rceil}, ..., V_n\}$ as illustrated in Figure 4.6. The occurrence index of each subset is summarized in Table 4.1.

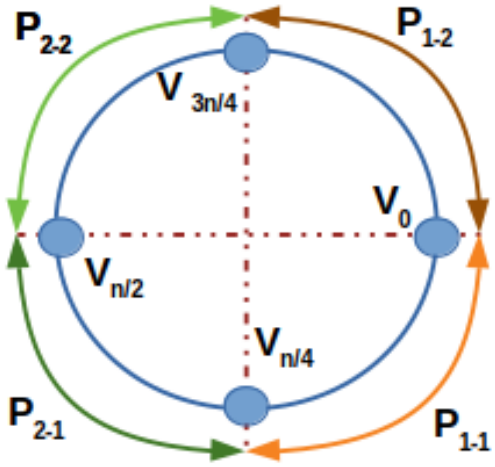


Figure 4.6: The division of the cycle nodes into 4 subsets based on their location.

4.3.3.2 Comparison of three protocols

Table 4.2 gives the values of T and L for the four protocols described previously (multicast circular routing, NC-based multicast from Algorithm 2, optimised protocols without and with NC from Algorithms 3 and 4).

Note that Algorithm 3 has a better performance in terms of T and L, compared to the NC-based multicast protocols. Moreover, Algorithm 4 allows us to obtain even larger gains. In particular, the gain in terms of T is up to 20% compared to Algorithm 2 and 14% compared to Algorithm 3 when n is sufficiently large. As for the number of transmitted packets L, it is reduced by up to 34% compared to Algorithm 2 and 12% considering Algorithm 3. The NC gain of Algorithm 4 is due to the possibility to broadcast messages to close neighbours (transmission rule 1). This condition is easy to satisfy in some kinds of networks, i.e., in wireless mesh networks [98]. In wireline networks, broadcast may be implemented by means of the IP-multicast [111]. But, if broadcast is not an option and one sends messages to the neighbours sequentially (i.e., classical routing in wireline networks), Algorithm 4 behaves

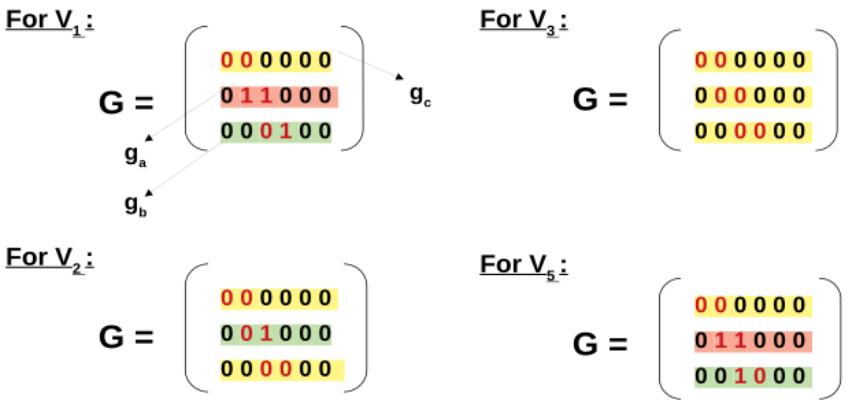


Figure 4.7: Generating matrix for nodes V_1, V_2, V_3 and V_5 for a cycle with n=5

like Algorithm 3. On the other hand, the half-duplex constraint (transmission rule 2) does not limit the usefulness of NC, Algorithms 3 and 4 can be adapted for full-duplex.

4.4 Conclusion

In this chapter, we proposed an optimized routing protocol for online games over a cycle topology, where the NC technique is applied. The proposed protocol reduces the number of transmitted packets over the cycle as well as the end-to-end delay. Indeed, with NC one can gain up to 14% of the communication period size an 12% of the number of transmitted packets considering a shortest-path routing protocol, adopted for online games, without NC. The transmission model we proposed for the cycle is based on three main characteristics: broadcast, half-duplex and collision. However, the protocol can be adopted for other transmission cases as shown in Section 4.3.3.2. The network here is chosen to be deterministic and the coding vectors are predetermined, as we assumed that the players are fix and the logical cycle topology is constructed and conserved during the game session. If the play-

n	T (LB/UB), L			
	Circular routing	Algorithm 2	Algorithm 3	Algorithm 4
7	21/28, 42	12/16, 24	12/15, 23	10/13, 19
8	24/32, 72	12/16, 36	12/15, 27	10/13, 23
9	27/36, 81	15/20, 45	15/18, 34	13/16, 30

Table 4.2: Comparison of T (lower bound and upper bound) and L for the three routing protocols.

ers are mobile, then the topology construction algorithm must be performed many times during the game session. The frequency of the topology construction depends on the mobility of the players. Each time, we calculate the adequate cycle members. After the construction, the different sets of nodes are recalculated and the routing protocol is applied normally.

Chapter 5

Network coding implementation: Cycle-based routing protocol for online games over a D2D infrastructure

Contents

5.1 Introduction

We chose to study the deployment and the performance of our protocol over a device-to-device (D2D) communication infrastructure. D2D is an emerging technology that is more and more appreciated. It will be largely exploited by the 5G technology due to its numerous advantages. Hence, it is interesting to chose this infrastructure for a possible future implementation of the proposed NC-based routing protocol. Part of this work is an object of one conference paper [112].

5.2 Device-to-Device network

We consider the case of online games where the servers are generally located far away from the players (example in the cloud network in the case of cloud gaming). The players need to communicate with this server through an Internet connection provided by the gateway node (for example a base station). Therefore, we will consider the network composed by the players and the gateway connecting them to the Internet (i.e to the distant server). This connection can be established via either a wired or wireless network. In this chapter we are more interested in a wireless network, specifically a Device-to-Device (D2D), where the players need to communicate with the gateway using consecutive wireless transmissions.

The D2D network is composed of interconnected user terminals. These devices can communicate with each other for different possible reasons. As an example, the D2D devices can help the relaying of distant devices to the base station with the minimum energy consumption and the maximum throughput [113]. Hence the online player devices can be interconnected with each other and with the base station connecting them to the game server. In this case, the cycle can be composed of the player devices, possible relay devices and the base station. The relay nodes can be chosen using the social-position relationship as proposed in [114] in order to maximize the cooperation probability. An example of a cycle over a D2D infrastructure is represented in Figure 5.1 where the dashed black lines represent the possible connections between devices and the dashed red lines represent the cycle linking the players to the base station. As for the technologies used for the communication between D2D devices, there are two approaches. We will detail them in the following.

5.2.1 Inband D2D

In this type of device-to-device infrastructure, the communication between the devices or user equipments UEs exploits the cellular spectrum and the radio interface such as LTE, LTE-A, WiMax and 5G [113]. In this type, we differentiate between the underlay and overlay D2D model.

- Underlay model: both the cellular and the D2D traffics use the cellular spectrum at the same time. In this model, interference is a crucial issue. Thus, solutions must be applied to avoid it. Interference management algorithms have been widely studied for this purpose. [115, 116].

- Overlay model: In this model, the cellular spectrum is divided between the D2D and the cellular communications. Each communication has its dedicated resources. Hence, the interference between these two communications is avoided. In this case, the partition of the spectrum should be well studied and optimized in order to preserve the resources. Besides, with the base station's assistance, the scheduling and power control can be controlled, which reduces the interference between the D2D flows [117].

5.2.2 OutBand D2D

OutBand D2D, also referred to as D2D-u, is the model where D2D communication is transmitted over an unlicensed spectrum using other wireless technologies such as WiFi Direct, ZigBee, bluetooth and LTE-u [113, 118]. In this case, the cellular and the D2D traffics are separated and interference problems between these two traffics do not exist anymore. However, using this spectrum, the devices should compete to access the medium and avoid collisions. As the devices are connected to both spectrums, the coordination between the radio interfaces is important and different solutions can be used. In this context, we differentiate between :

- The controlled D2D model: where the coordination between the radio interfaces of the user's equipment is controlled by the base station.

- The autonomous D2D model: in this model, the coordination between the radio interfaces of the user's device is performed without the intervention of the base station. Instead, it is controlled by the users themselves.

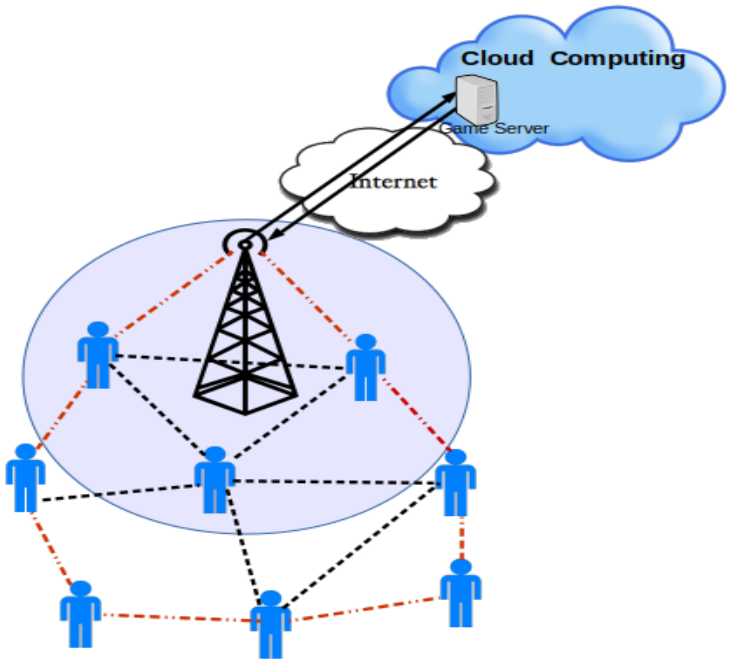


Figure 5.1: Single-cycle topology using D2D technology

We are interested in the OutBand case and specifically the WiFi Direct technology [119].

Since the studies which focus on proposing efficient and flexible medium access schemes that avoid D2D traffics collision are still in progress, we have chosen the safest solution by applying our proposed scheduling scheme, exploiting the possible synchronization of the devices using, for example, the highway addressable remote transducer (HART) protocol [120] or the adaptive distributed network synchronization (ARES) algorithm [121]. The system's model's description will be illustrated in the next section.

5.3 System model

Let the logical network of an online game be represented by a single cycle as illustrated in Figure 4.1 in Section 4.2, connecting the game server $S{=}V_0$ and n game players $\{V_1, \ldots, V_n\}$. As the server is generally located far away from the players, the node S in our model represents the gateway node between the players and the server. Hence, the server packets will reach the players

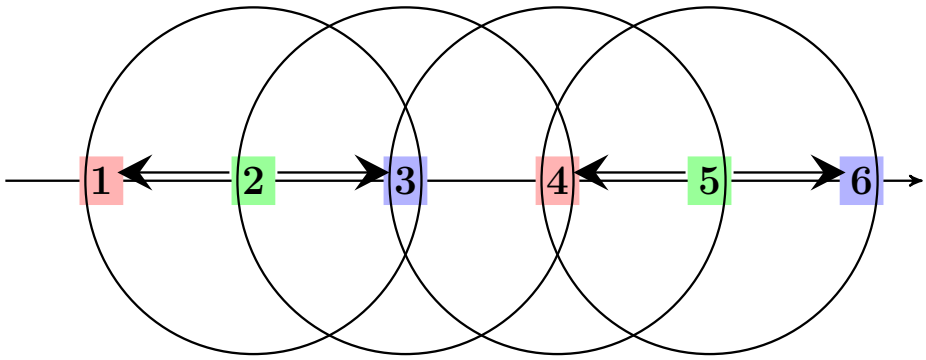


Figure 5.2: The coverage area of each node on the cycle.

in the cycle through that gateway node S. If we consider the partition of players described in Section 3.3, then S is the local server of the subgroup of players sharing the same area on the game map. Our cycle is composed of $n+1$ nodes where the coverage area of each node is limited to its two neighbors. The neighbors of a node are its closest two nodes in the cycle as shown in Figure 5.2.

As we always consider the case where collisions are possible, the system model will be based on the scheduling presented in Section 4.2. We call to mind here that the main transmission rules are as follows:

- when a node V_i sends a packet, it broadcasts it to its two closest neighbors $V_{(i-1)}$ and $V_{(i+1)}$;
- the system is half duplex, so each node is either in the receiving or in the sending state during one time unit;
- a node cannot receive two simultaneous messages from its neighbors in order to avoid collisions.
- we can find intermediate nodes which are not game players but are needed to relay between the cycle nodes.

As a result, we apply the node partition into 3 or 4 subsets depending on the number of nodes $n+1$ as shown in Section 4.2.

5.4 Communication period size

In this section, we present the set up as well as the results of the simulations we run in order to measure the communication period's size.

5.4.1 Simulation setup

To simulate the proposed routing protocol over a cycle topology, we used the network simulator $NS3$. In this chapter, we are interested in the WiFi Direct wireless technology. However, as we are not interested yet in the discovery of devices and for the sake of simplicity, we ran our simulations over a WiFi Adhoc topology which can afford the point to point communications between the nodes and provide the needed results such as the size of the communication period and the impact of the time unit size on the packet loss rate. The device discovery as well as the topology construction will be addressed in future works. The simulation parameters are set as follows:

- The wireless network is set to be lossless, with WiFi nodes and link throughput of 100 kb/s;
- The transmission delay from one node to its neighbors is of the order of 2ms, which has been fixed as the $t.u$;
- The number of players varies from $n = 5$ to $n = 50$;
- The transmission/reception power and the gains are set so that the coverage range of each node is nearly 250m;
- The nodes have been distributed on the graph so that the coverage of each node reaches only its two neighbors;
- The client packet's size is set to 70 bytes while the server packet's size is set to 110 bytes.

5.4.2 Simulation results

As we mentioned earlier, the NC multicast protocol can minimize the communication period for online games' traffic over a cycle topology. However, we argued that a more adopted routing protocol for online games' specifications, as ours, will provide a better performance. In order to validate this

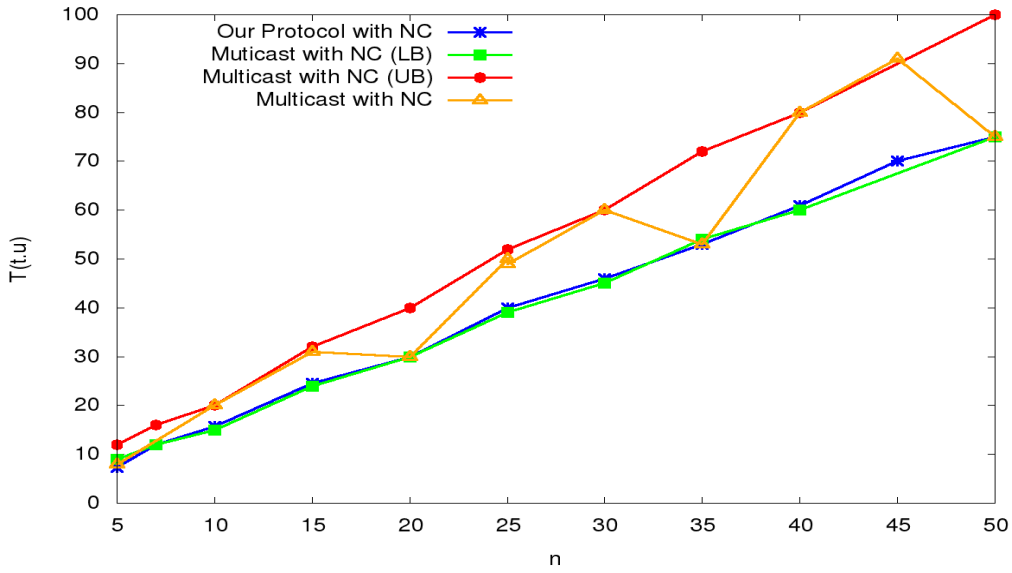


Figure 5.3: The mean communication period T

assumption, we ran multiple simulations of our protocol using the simulator $NS3$ for different values of n and compared the results found to those of the NC multicast protocol. The results are presented in Figure 5.3. The figure shows that the mean communication period of our protocol is almost equal to the lower bound of the communication period of the NC multicast protocol. Note that the lower bound of the NC multicast protocol is reachable only if $(n+1)|3$. Otherwise, for the rest of the values of n, the communication period will reach the upper bound. However, with the NC-based routing protocol, the communication period remains near the lower bound for any value of n. As a result, we are not interested in some values of n which can provide better latency than other possible values. For instance, we are not obliged to fix a cycle size that is divisible by 3 to guarantee lower latency as the case of the NC-based Multicast protocol. Hence our protocol is more flexible in terms of cycle size.

5.5 Constraints and solutions

In this section, we present and discuss some possible solutions for different constraints that are evoked when implementing the NC-based routing proto-

col over the cycle.

5.5.1 Packet size

The client packets and server packets have different sizes. In fact, The server packets' size depends of the number of players in the game. However, to perform network coding, the packets' size should be the same. This problem of sizes can be solved by applying fragmentation to the server's packets to meet the size of those of the clients. If we suppose that the server packet's size is approximately $k * S$ where S is the size of one client packet, then the server packet will be divided into k sub-packets and each sub-packet will be sent during a communication period. Hence the time unit $t.u$ will depend only on the size of the clients' packets S and the communication period will be shorter.

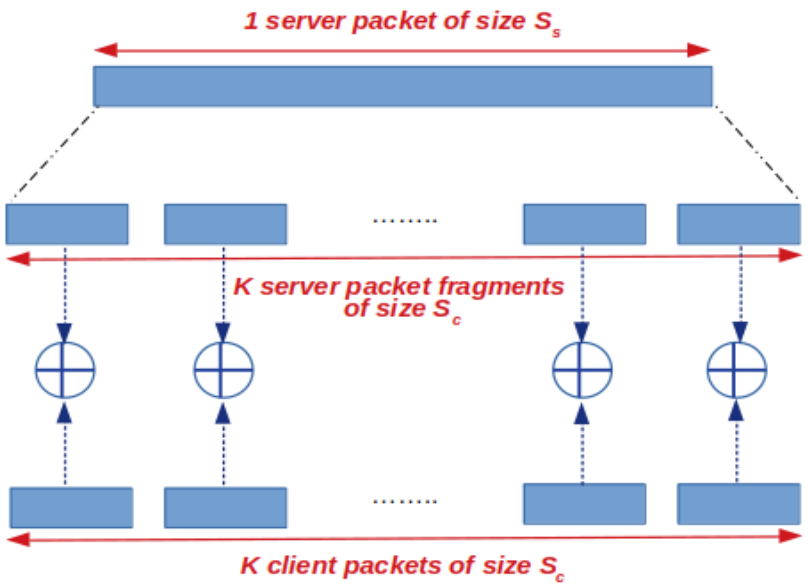


Figure 5.4: Fragmentation of the server packet to encode with the client packets

5.5.2 Delay variation

Each node in the cycle has three possible states. It is either waiting for a packet from the right, waiting for a packet from the left or sending. Each state will last for one $t.u$. When the $t.u$ allowed to each state expires, the node will change the state and will ignore any actions related to the previous state. In other words, if a packet is received from the right after the expiration of the $t.u$ allocated to the waiting from the right, then it will not be taken into account; it will be rejected and considered lost. If we are to apply our routing protocol within a perfect network, we can use the transmission delay of one packet from one node of the cycle to its neighbor as our $t.u$. Let's note D_M as being the maximum transmission delay between two successive nodes in the cycle so that:

$$D_M = \max_{i,j}(D_{(i,j)}) \tag{5.1}$$

where i and j are the indexes of the cycle's nodes. Before starting the first communication period, the server will initialize the transmission in order to synchronize the nodes. Hence, it is possible to determine an approximation of the maximum transmission delay D_M between two successive nodes in the cycle and fix $t.u$ such that $t.u$=D_M. However, the transmission delay over the cycle's links may change depending on some network specifications such as transmission window modification in case of TCP traffic, additional charge from other applications to be transmitted by the cycle's nodes, etc. As a result, if the transmission delay of a node's packet over a link increases significantly, the neighbor nodes will consider the packet as lost and the packet loss rate will increase accordingly. In order to minimize the number of lost packets, we should add an additional delay or a jitter delay D_J to D_M when calculating the time unit of the cycle :

$$t.u = D_M + D_J \tag{5.2}$$

Once the $t.u$'s value is fixed, each node will wait to receive packets during one $t.u$ (timeout) from each side. Hence, even if the packet is received before the end of the timeout, the node cannot change the state. It will wait until the expiration of the timeout. As a result, while the values of transmission delay between two successive nodes on the cycle D and the jitter delay added by the node can vary from one node to another and from one transmission to another, the time unit's size $t.u$ is always the same, as shown in Figure 5.5.

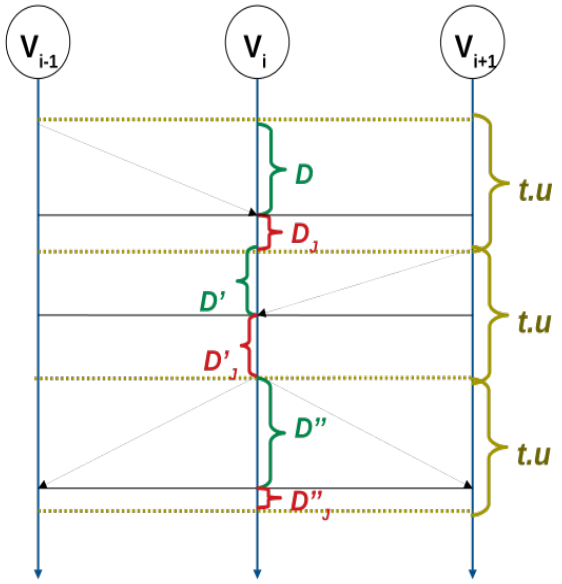


Figure 5.5: $t.u$ calculation taking into account the jitter delay D_j

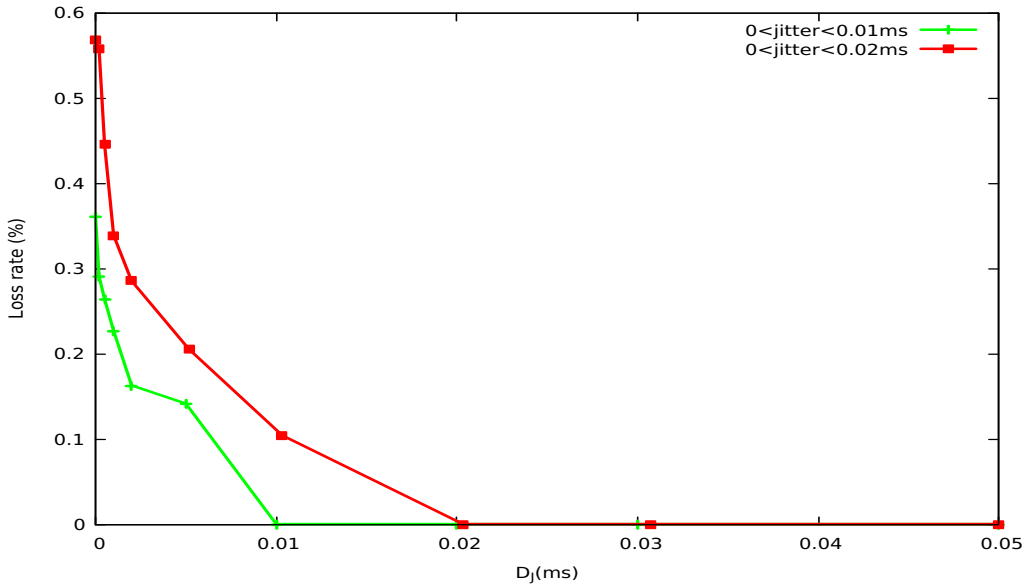


Figure 5.6: Loss rate depending of the D_J used for two variation of the jitter : up to 0.01ms and up to 0.02ms.

In order to evaluate the effect of the D_j choice on the packet loss rate, we implemented the simulations in $NS3$ considering different distributions of the delay variations. The network used is a lossless network so that a packet loss can only occur in case the packet is discarded when received after the timeout expiration. Figure 5.6 sums up the simulation results. We can see that the packet loss rate decreases when D_J increases, but at some point, it becomes meaningless to increase D_J as we will have 0% packet loss rate. Let's note this maximum delay jitter by $D_{J_{max}}$. It may seem intuitive to choose $D_{J_{max}}$ and prevent any loss. However, we are dealing with a real-time application where delay is a crucial factor. For online games, a maximum tolerable delay has been estimated for different types of games [5]. Let's note this maximum as D_{max}. Besides, we showed in Section 5.4.2 that the delay depends also on the size of the cycle. Hence, the choice of D_J should be carefully studied and calculated depending on the network's quality, the network's size or the number of clients n and the maximum delay D_{max}. The maximum value of delay jitter no longer corresponds to the 0% packet loss, but rather to all the mentioned factors and can be calculated using Equation (5.4). Figures 5.7 and 5.8 illustrate the calculation of $D_{J_{max}}$ for a

fixed n and a variable n respectively. In Figure 5.7, $D_{J_{max}}$ corresponds to the intersection of the graphs. In Figure 5.8, the red line represents the possible couple $(D_{J_{max}}, n)$ representing the $D_{J_{max}}$ value for each number of players n. The first transmission period will take $D_J = D_{J_{max}}$ at the beginning. Afterwards, the rate of packet loss is calculated, the D_J value will be reduced from one transmission period to another until reaching the threshold of packet loss rate $Loss_{max}$. This value can be decided depending on the type of game and the tolerance of the players [24]. Using $Loss_{max}$, we can define another limit of the D_J value, noted as $D_{J_{min}}$. Hence, D_J is chosen so that $D_{J_{min}} \leq D_J \leq D_{J_{max}}$ using Algorithm 5, where K denotes the pace of variation of D_J. Let's fix K as follows:

$$K = (D_{J_{max}} - D_{J_{min}})/5 \tag{5.3}$$

$$D_{NC} \leq D_{max} \tag{5.4}$$
$$T_{NC} * t.u \leq D_{max} \tag{5.5}$$
$$T_{NC} * (D_M + D_{J_{max}}) = D_{max} \tag{5.6}$$
$$(3\lceil n/2 \rceil + 1) * (D_M + D_{J_{max}}) = D_{max} \tag{5.7}$$
$$D_{J_{max}} = \frac{(D_{max} - (3\lceil n/2 \rceil + 1) * D_M)}{(3\lceil n/2 \rceil + 1)} \tag{5.8}$$

Algorithm 5 D_J calculation

Initialization: $D_J = D_{J_{max}}$
Dynamic calculation:
While $(D_J \leq D_{J_{max}})$
If $(Loss \leq Loss_{max})$
$D_J = D_J - K$
Else If $(D_J + K \leq D_{J_{max}})$
$D_J = D_J + K$
EndIf
EndIf
EndWhile

Using the example of Figure 5.8, we can determine also the maximum number of clients or players n_{max} for each value of D_J. With no delay jitter, $D_J = 0$, the value of n_{max} corresponds to approximately 68 players. If we add more players, we will exceed the maximum tolerable delay by the game players.

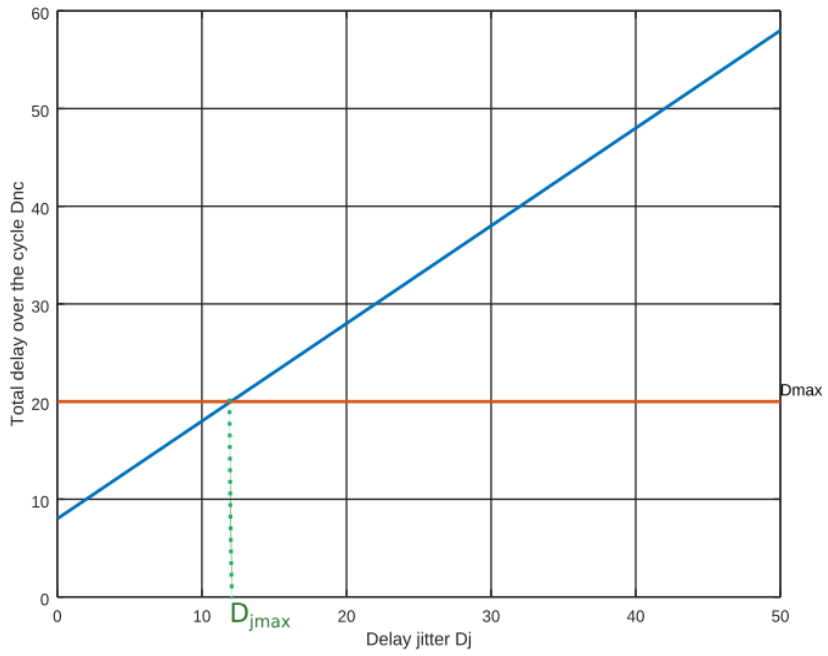


Figure 5.7: $D_{j_{max}}$ calculation for $n = 25$

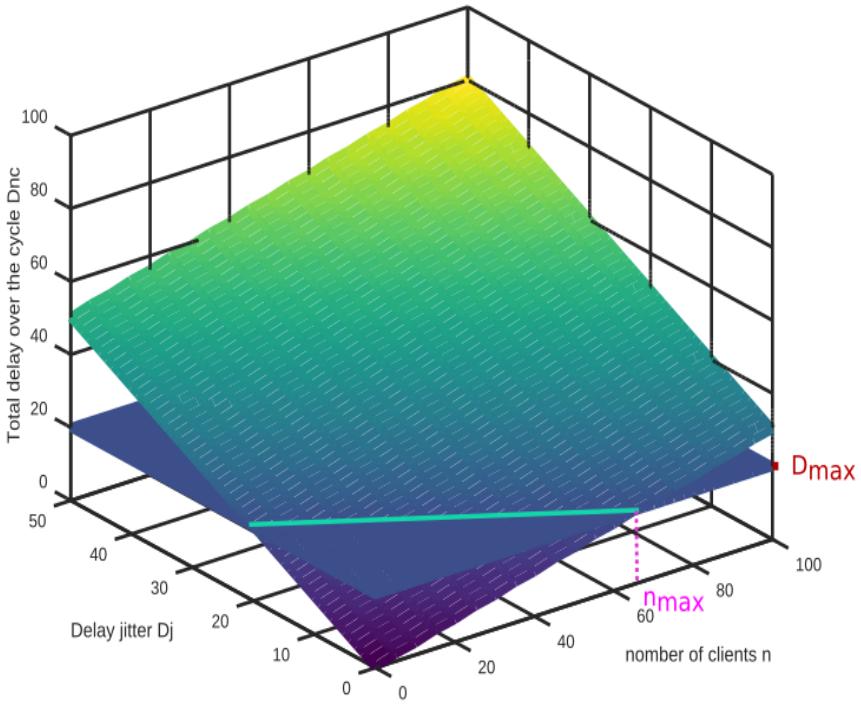


Figure 5.8: $D_{J_{max}}$ calculation depending on n

5.5.3 Packet loss

When dealing with lossy networks, feedback-based transmission protocols are the intuitive solution to recover the packet losses. In this context, TCP is widely used when a reliable transmission is needed, including online gaming applications. On the other hand, NC has been proven to be efficient even in the case of lossy networks. In fact, some researchers considered network coding schemes in order to resolve the loss issue [122, 123]. Based on these results, NC-based protocols have been proposed for various applications [123, 124], even real-time ones such as video streaming [125, 126] when the transmission channel is lossy. These propositions included wired networks [127, 126] as well as wireless networks [128, 124, 129], for unicast [128, 130], multicast [124, 131] and broadcast [132] transmissions. The approaches proposed are mostly based on either random network coding[122, 133, 123] or redundancy [128, 126, 134]. Besides, the classification of the packets into generations or batches has gained interest in this context [122, 133, 135, 136]. Some of these works were also based on the feedback information [128, 133, 129, 135] while others made it possible to deal with the packet losses without any feedback [122, 132, 136]. As for our case, we believe that the proposed NC-based routing algorithm can be naturally adopted to deal with packet losses. For this purpose, we propose two possible modifications for two different approaches:

1. Solution 1: TCP retransmission
   As the reliability of packet transmission is important for some of the game's traffic [137], many game developers choose to use the TCP protocol to guarantee the reception of the packets by their destination through successive retransmissions. Thus, our first solution is based on the belief that the lost packets will be detected and retransmitted using the transport-layer reliable transmission. In this case, we should only guarantee that the other packets (not lost packets) will be transmitted in time. To do so, the algorithm can be adopted by specifying the following:

   - For each node V_i, if the node is supposed to send $M^{\leftarrow}_{i-1}(t-1)$ or $M^{\rightarrow}_{i+1}(t-1)$ but that packet was not received (a lost packet) then no packet is sent.
   - For each node V_i, if the node is supposed to send $M^{\leftarrow}_{i-1}(t-1) \oplus M^{\rightarrow}_{i+1}(t-1)$ and one of these packets was not received (resp both

of them), then it sends only the received one (resp no packet is sent).

As we already estimated the communication period's size using our protocol for a lossless network case, we can fix T to match the upper bound of T_{NC}. Then, after T time units, the current communication period will stop and a new communication period with new native packets will start. Hence the delay will always be under control.

2. Solution 2 : Loss detection and retransmission
The retransmission procedure with TCP costs much time as it is only detected by the receiver. Hence, a solution to treat the packet loss while transmitting the traffic will reduce the additional delay caused by the packets' retransmission when using TCP and will guarantee a reliable transmission with the minimum cost if UDP is used. For this purpose, the second solution is proposed. It consists in detecting the loss of a packet and retransmitting it within the same communication period. In fact, the loss detection is quite natural with our protocol as the packets are implicitly acknowledged by all the nodes except the server and the extreme nodes ($V_{\lceil n/2 \rceil}$ and $V_{\lfloor n/2 \rfloor}$). In the following, we summarize this solution.

5.5.3.1 Protocol modifications

Let us incorporate a packet loss event into our system model. As explained before, the nodes of the cycle have a reception timeout parameter equal to $1t.u$. Therefore, each node listens to its neighbor's transmission during $1t.u.$. If the reception of a message M_i was expected but no packet has been correctly received during this time interval, then M_i is considered as lost. The modifications to add in order to recover a packet loss are summarized as follows:

- If a node has one packet to send, then no coding
- If a node has no packet to code, then an all-zero packet P_{00} is sent.
- If a node does not receive back the packet it has sent during the previous round from its neighbours, then the loss is detected.

- If a node detects a packet loss then it would retransmit the same packet (if coded then resend the coded packet, if not resend the native packet).
- If a packet is sent as a retransmission after a loss, then we should distinguish this packet (we add an indicator to the header) so that other nodes do not suppose that their own sent packets, which are supposed to be received back at that period, are lost.

As a result, the NC-based routing protocol , Algorithm 4, should be modified in order to adopt it to packet loss by incorporating the two following points:

1. **Generation of messages $M_i(t)$ to be sent by a node V_i at round t**

   - If $V_i \in S_{\oplus}$ and V_i was expecting to receive $M_{i-1}^{\leftarrow}(t-1)$ and $M_{i+1}^{\rightarrow}(t-1)$ at previous ruond $t-1$, let it broadcast

$$M_i(t) = \begin{cases} \blacktriangleright\ M_{i-1}^{\leftarrow}(t-1) \oplus M_{i+1}^{\rightarrow}(t-1), \\ \text{if no losses} \\ \blacktriangleright\ M_{i-1}^{\leftarrow}(t-1), \\ \text{if } M_{i+1}^{\rightarrow}(t-1) \text{ is lost} \\ \blacktriangleright\ M_{i-1}^{\rightarrow}(t-1), \\ \text{if } M_{i+1}^{\leftarrow}(t-1) \text{ is lost} \\ \blacktriangleright\ M_i\text{(t-1) if } M_i\text{(t-1) is lost} \\ \blacktriangleright\ P_{00} \text{ otherwise} \end{cases} \tag{5.9}$$

   - If $V_i \in S_{\rightarrow}$ (or $V_i \in S_{\rightarrow}$) and V_i was expecting to receive $M_{i+1}^{\rightarrow}(t-1)$ (or $M_{i-1}^{\leftarrow}(t-1)$) at previous round $t-1$, let it send the message $M_i(t) = M_{i+1}^{\rightarrow}(t-1)$ (or $M_i(t) = M_{i-1}^{\leftarrow}(t-1)$) if there has been no loss; let it send P_{00} otherwise.

   When using (1.1), one needs to let the receiving node know which of these possible transmission options has been chosen (coded packet, uncoded packet, retransmitted packet). This can be done by adding several extra bits to the message header of $M_i(t)$. Also note that if $M_i(t)$ is lost, its source node V_i can auto-detect the loss at the next round $t+1$ in the case when it overhears the transmissions of its neighbours (see Figure 5.9 for illustration). Thus the neighbours' broadcasts

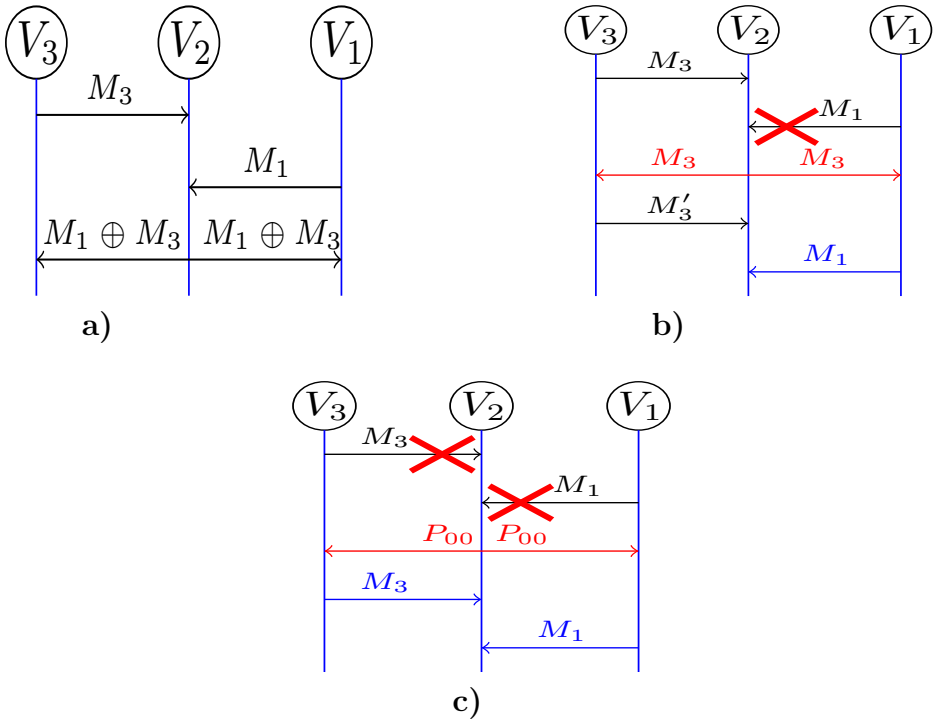


Figure 5.9: Illustration of possible loss scenarios: a) transmission protocol without losses; b) a loss of message M_1, detected by V_1 after the reception of M_3 (in red) and followed by retransmission of M_1 (blue); c) a loss of messages M_1 and M_3, detected by V_1 and V_3 after the reception of P_{00} from V_2 and followed by retransmissions of M_1 and M_3 (blue).

implicitly serve as a natural ACK feedback and can be used to correct message losses by retransmissions (see Point 3 below). Note that the only packets that are not protected by the implicit ACK are the ones which have as destination the server node V_0 or the end nodes $V_{\lfloor n/2 \rfloor}$ and $V_{\lceil n/2 \rceil}$. These three cases are treated apart (see Point 2 below).

2. **Transmission of ACK messages by nodes V_0, $V_{\lfloor n/2 \rfloor}$ and $V_{\lceil n/2 \rceil}$:** Let, at round $t-1$ a node V_i with $i \in \{0, \lfloor n/2 \rfloor, \lceil n/2 \rceil\}$ receives $M_{i-1}^{\leftarrow}(t-$

1) and/or $M_{i+1}^{\rightarrow}(t-1)$. Then, at round t, it broadcasts

$$M_i(t) = \begin{cases} \blacktriangleright\ M_{i-1}^{\leftarrow}(t-1) \oplus M_{i+1}^{\rightarrow}(t-1), \\ \text{if both messages received} \\ \blacktriangleright\ M_{i-1}^{\leftarrow}(t-1), \\ \text{if } M_{i-1}^{\leftarrow}(t-1) \text{ received} \\ \blacktriangleright\ M_{i+1}^{\rightarrow}(t-1), \\ \text{if } M_{i+1}^{\rightarrow}(t-1) \text{ received} \end{cases} \tag{5.10}$$

The messages $M_i(t)$ serve to acknowledge the reception and the neighbour nodes become able to detect their packets' loss.

3. **Retransmission of lost messages**:
When, at some round t, a node V_i detects that its message $M_i(t-1)$ has been lost, it will retransmit it at the next transmission, i.e., $M_i(t) = M_i(t-1)$. The messages received by V_i at round $t-1$, that were supposed to be transmitted forward at round t, will be stored in the memory and transmitted at the next round $t+1$.

5.5.3.2 Performance impact

Let us discuss the behavior of the modified algorithm. It can be seen from Figure 5.9 that one message loss delays the whole transmission by at most one round (i.e., 3 or 4 t.u. depending on the current protocol and topology state). In fact, thanks to the cycle topology and the broadcast transmission, the packets are sent to their destinations in two directions. Hence, even if a packet is lost, resulting in a transmission delay of other packets, it is possible to receive a copy of those delayed packets from the second direction without waiting for the delayed ones. As a result, the loss may result in a delay that varies between 1 and 3 (or 4) time units. Therefore, if some number k of message losses happens during one communication period T, the communication period T increases by at most k rounds. The upper bound is only attained in case the message losses occurred during a different period t each, and the lost packets are not yet received from the other direction of the ring. Also note that the number of transmitted messages L is also increased. In the presence of k losses, L is increased by $2 + \lceil n/2 \rceil + k$ messages, where

the term $2+\lceil n/2 \rceil$ comes from the implementation of ACK messages at Point 2 above, and k comes from the Point 3.

Taking into account the impact of the packet losses' recovery on the communication period size and the latency, the calculation of the maximum delay jitter value should be revised. If we note one packet loss cost in terms of latency as D_{Loss} then we can calculate the maximum total number of retransmissions allowed $R_{T_{max}}$ as follows:

$$R_{T_{max}} = \frac{D_{max} - D_{NC}}{D_{Loss}} \tag{5.11}$$

where

$$D_{NC} = T_{NC} * (D_M + D_J) \tag{5.12}$$
$$D_{NC} = (3\lceil n/2 \rceil + 1) * (D_M + D_J) \tag{5.13}$$

Hence, $R_{T_{max}} = f(n, D_M, D_J)$ and the choice of $R_{T_{max}}$ and the corresponding $D_{J_{max}}$ value is to be decided based on a compromise depending on the quality of the network and the tolerance to the packet loss.

The value of $R_{T_{max}}$ should be equally divided between the players. Hence, each node will have a maximum number of retransmissions $R_{max} = \frac{R_{T_{max}}}{n+1}$ to recover from packet losses. As not all the packets of online games require reliability, we can associate with each packet a priority indicator to tell the cycle nodes if they need to try to retransmit this packet in case of loss or not. In this way, we give more chance to the most important packets.

5.5.4 Packet ordering

As explained in the first chapter, consistency is a major factor influencing the QoE of the online games' players [26]. To ensure this consistency, we need to guarantee a causal order between the clients' packets since the actions of the players are interdependent when the avatars are interacting.

Definition 4 *A traffic is causally ordered if the packets sent by a client at time t are received by the server before a packet sent by any client at time $t' > t$.*

Implementing some ordering mechanisms in the end users' application layer may help to treat the packets in the right order. However, this solution

is complex and induces large delays. If the routing protocol can help in ordering the packets, then a simpler and faster reordering can be performed.

As we already stated, the client traffic is periodic. Let's note its period as P. Using this characteristic, one can consider a new definition of "order".

Definition 5 *The traffic is considered causally ordered if the packets sent at a period P are received by the server before the packets sent at period $P+1$ and after the packets sent at period $P-1$.*

In other words, packet M_i^k must be received by the server before M_j^k for each communication period $k > 0$ and each couple of nodes (V_i, V_j) where $(i, j) \in (1, ..., n+1) \times (1, ..., n+1)$. Otherwise, the sequence is said to be disordered. Note that we assume that packets sent within the same period will be ordered by the server. Hence, our objective now is to find *a well-ordering* network code $\mathcal{C}$ that ensures this order.

Proposition 1 *For a code $\mathcal{C}$, if a node sends more than one packet during a communication period (i.e. a new packet is sent before all packets from client nodes generated at period P are received by the server), then the code in not well-ordered and the causal order is not preserved.*

Proof. Let's consider our proposed code. Assume that a client is allowed to send a new packet each time it is in the transmitting state (without waiting for the end of the communication period). Hence, after the initialization step, each node will send two types of packets: a coded packet and its own new packet (see Figure 5.10). In this case, the new packets generated by the closest nodes to the server will be received by the server V_0 before the old packets generated by the furthest nodes to the server. In Figure 5.10, we can see that the second packet from nodes V_1 and V_n (surrounded by blue circles) is received before the first packet sent by other nodes such as nodes V_3 and V_{n-2} (surrounded by red dashed circles)).

5.6 Conclusion

In this chapter, we proposed a practical implementation scenario for the proposed NC-based routing protocol and discussed some of the constraints imposed by real network conditions. We present a solution for packet size difference between the server and the client packets. Besides, we discussed how the network jitter can influence the size of the time unit of the protocol. Moreover, we proposed a solution to detect packet loss and retransmit the lost

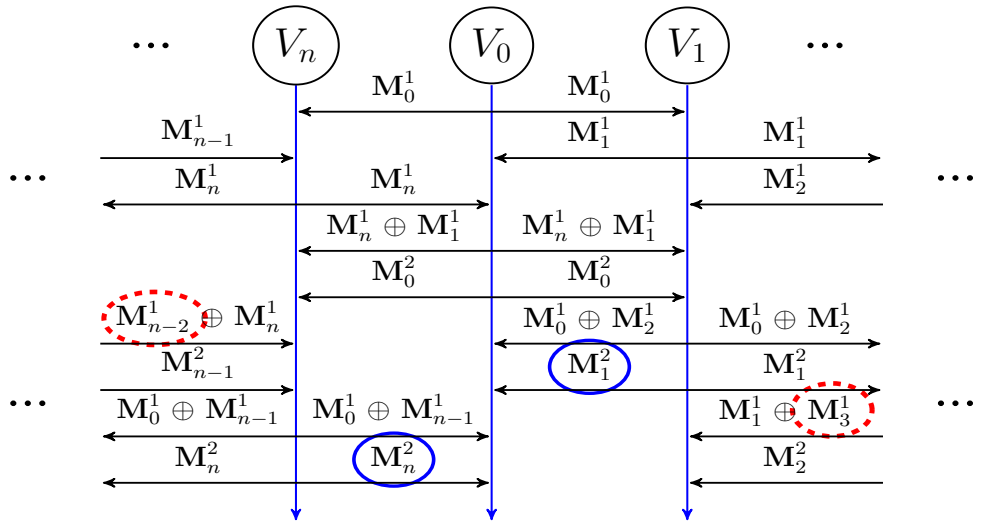


Figure 5.10: Simulation results for the modified code.

packets in a reasonable rate in order to guarantee a certain level of reliability exploiting the cycle topology. Moreover, using this solution, the lost packets will be detected and retransmitted as soon as the loss occurs and not at the receiver's side, which decreases the delay caused by packets' retransmission. Note that, using this solution, the game developers can avoid TCP disadvantages and use UDP over the cycle, which is faster and lighter in terms of traffic load due to the larger headers, the acknowledgments' transmission and the transmission window variation.

In this scenario, we conserved our proposed transmission model that allows collision avoidance. This model consists of a fixed transmission scheduling, based on the position of the node over the cycle. It uses an alternation between the three different states: sending, receiving from the right and receiving from the left. This alternation is important for our routing protocol so that each node has the packets it needs in each step. To avoid collision, we divided the nodes into subgroups and adopted a slotted synchronous transmission. If collision can be avoided using some other available transmission and resource-allocation schemes, then an asynchronous transmission can be adopted, but using the same alternation pattern between the states. In this case, the timeout concept will also be used: the nodes will wait for one time unit for the packets to be received. However, when sending, if the node has

already received the needed information and has a packet to send, it can send this packet as soon as the transmission control protocol allows it, without waiting for the expiration of the timeout. For instance, the proposed protocol in [138] for an inband d2d infrastructure, can be adopted by imposing the order of transmissions of the neighbor nodes when selecting the transmitter by the receivers. This asynchronous approach can provide gains in terms of delay in some cases.

Chapter 6

Conclusion and prospective work

In this thesis, we were focused on enhancing the quality of experience of massively multiplayer online games. The traffic of these games is composed of bursts of small sized packets with a reduced amount of data. This characteristic degrades the network efficiency and creates an overload problem at the server side, especially with the growing number of players. Besides, the online gamers are hard to satisfy and may quit the game if the Quality of Experience is not acceptable. The QoE is influenced by the consistency of the game which can degrade due to some network imperfections like high latency, excessive jitter and packets disorder.

As a solution for the consistency issue, we proposed to use a cycle topology where packet's order can be easily maintained using some adequate transmission management procedures. Then, in order to reduce the cycle size, we propose to partition the players into subgroups connected to local servers. Each local server is connected to the central server.

Our first accomplishment in this thesis is the investigation of how a cycle topology can deal with online game traffic while conserving an acceptable end to end delay for the gamers. First, we designed an optimized routing protocol over cyclic topologies, while considering the specific nature of the online games traffic. Second, in order to reduce the traffic load and the latency over the cycle, we added network coding operations to the routing protocol.

Our second accomplishment is the evaluation of the impact of NC on online game applications, which has not been investigated yet. The NC gain

was evaluated in terms of traffic load and end to end delay, using theoretical calculation as well as simulations.Concerning the delay, the achieved gain of NC, when n is sufficiently large, amounts to 20% compared to the NC-based multicast routing protocol over cycle and to 14% compared to the optimized shortest path protocol. As for the number of transmitted packets, it decreased by 34% compared to the NC-based multicast protocol and by 12% compared to the optimized shortest path protocol.

We also described a possible implementation scenario of the proposed protocol in a D2D infrastructure. Finally, to adopt this protocol to realistic network conditions, we discussed solutions and enhancement possibilities to solve the problem of packet size difference, network jitter and packet's loss.

As a result of the player partition procedure, the game delay calculation will be divided into two categories. For instance, let's call them local delay and global delay. The local delay corresponds to the delay over the subgroup: between players and the local server. It concerns the perception of the game state in the local area of the player. As for the global delay, it represents the delay between the players and the central server. It concerns the global game state perception. To each of these delays, one can associate an adequate maximum value acceptable by the players, since the global and the local view are different in terms of importance for the players. For the local delay, the bounds can be the same as the conventional maximum delays for online games. As for the global delay, it should be investigated and estimated. But generally it is less strict than the first type.

We note that, due to time constraints, we were not able to compare the performance of the TCM-based solution proposed in the literature, and the NC-based routing solution that we proposed in this thesis. In fact we simulated the TCM solution and the initial version of the NC-based routing over cycles, without the proposed modifications in section 5.5. In the future, we will continue the implementation of the final version of the protocol, evaluate its performance, propose enhancement in case of problems and compare it to TCM.

As for prospective work, it can cover the following points:

- *Global infrastructure:* In this thesis, we were interested in connecting the players of the subgroups to their local server and proposed the NC-based routing protocol over a cycle topology. As for the connection between the local servers and the central server, it was out of our scope. There are three possible ways to deal with this part of the infras-

tructure: a star topology where a point to point connection between each local server to the central server is established, a tree topology with TCM or a cycle topology with the proposed routing protocol. To decide between these solutions, one should evaluate their performance considering the specificity of the traffic between the local servers and the central server, in terms of packet size and frequency as well as the delay constraints.

- *Packets order:* The proposed protocol provides a periodic ordering for packets as we suppose that the server will order the packets of the same period. Further work can be done to extend this solution to a fully ordered one by adding some of the ordering keys usually used over cycles such as a token or a timestamp.

- *Transmission model:* A synchronous transmission has been adopted in this work in order to guarantee a safe access to the channel and avoid any collision. However, if the technology used presents collision avoidance solutions and the involved terminals in the cycle support these solutions, then an asynchronous transmission can be considered and the protocol can be adopted.

- *Data compression:* Beside the headers compression proposed in the TCM technique, some works considered data compression for the server packets [50]. They were interested in the server traffic as it represents the flow with the largest packets size. Simulations proved that, applying data compression for these packets can help reducing the online game traffic. In fact, they found that using some specific coding schemes, they can achieve interesting results. For online games, the upstream traffic load poses more problems than downstream traffic. However, compressing the data of the client traffic was not considered because of the small size of the packets. We were interested in investigating the data compression for client packets. Our first tests over real traces of an online game proved that a certain level of correlation between these packets exists and can be exploited using some joint-source compression schemes. This compression will reduce the traffic charge and increase the network efficiency. Thus, we plan to continue studying this approach and evaluate its influence in terms of the QoE of online games. If the benefits of the data compression for client packets is confirmed, it can be added to the proposed solutions.

- *Construction cost:* In this work, we did not focus on the construction of the logical cycle topology. However, this construction has its own cost in terms of delay depending on the computation complexity. For instance, the computation cost of the Multiring Construction Algorithm (MCA) proposed in [110] was estimated to be equal to several minutes for a network composed of 400 nodes. This cost should be considered while choosing the adequate construction algorithm for online games application.

- *Security:* With wireless networks, providing security becomes crucial. For online games, one should protect some information, such as the players account identifiers and their chatting texts. As many online games are paid, hackers are interested in stealing players accounts and play for free. Besides, online games players can chat with each other during a game session. Their discussion can include some personal details. In this thesis we didn't treat security issues for online games. For now, we suggest to apply one of the well known encryption protocols for wireless networks, such as WEP, WPA and WPA2 [139]. Further investigation and work can determine if there are more efficient security solutions for online games or not.

Bibliography

[1] Dah Ming Chiu, R. W. Yeung, Jiaqing Huang, and Bin Fan. Can network coding help in p2p networks? In *2006 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, pages 1–5, February 2006.

[2] Nobuaki Otsukiâ and Yusuke Asai and Takatoshi Sugiyama. Research and development of wireless network coding to achieve high-efficiency multihop wireless systems, March 2010.

[3] C. Fragouli and E. Soljanin. A connection between network coding and convolutional codes. In *2004 IEEE International Conference on Communications (IEEE Cat. No.04CH37577)*, volume 2, pages 661–666 Vol.2, June 2004.

[4] Ducheneaut Nicolas and J. Moore Robert. The social side of gaming: A study of interaction patterns in a massively multiplayer online game. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work, CSCW*, pages 360–369, January 2004.

[5] Chris Gauthier Dickey, Daniel Zappala, and Virginia Lo. A distributed architecture for massively multiplayer online games. *University of Oregon*, August 2004.

[6] Kuan-Ta Chen, Polly Huang, and Chin-Laung Lei. Game traffic analysis: An mmorpg perspective. *Computer Networks*, 50(16):3002 – 3023, 2006.

[7] S. Ratti, B. Hariri, and S. Shirmohammadi. A survey of first-person shooter gaming traffic on the internet. *IEEE Internet Computing*, 14(5):60–69, September 2010.

[8] C. Diot and L. Gautier. A distributed architecture for multiplayer interactive applications on the internet. *IEEE Network*, 13(4):6–15, July 1999.

[9] Mark Terrano and Paul Bettner. 1500 archers on a 28.8: Network programming in age of empires and beyond. GDC 2001, March 2001.

[10] Choong-Soo Lee. The revolution of starcraft network traffic. In *2012 11th Annual Workshop on Network and Systems Support for Games (NetGames)*, pages 1–2, November 2012.

[11] Smed Jouni, Kaukoranta Timo, and Hakonen Harri. Aspects of networking in multiplayer computer games. In *ADCOG*, pages 74–81, November 2001.

[12] M. Merabti and A. El Rhalibi. Peer-to-peer architecture and protocol for a massively multiplayer online game. In *IEEE Global Telecommunications Conference Workshops, 2004. GlobeCom Workshops 2004.*, pages 519–528, November 2004.

[13] Bharambe Ashwin, Pang Jeffrey, and Seshan Srinivasan. Colyseus: A distributed architecture for online multiplayer games. In *Proceedings of the 3rd Conference on Networked Systems Design & Implementation-Volume 3*, NSDI'06, pages 155–168, Berkeley, CA, USA, 2006.

[14] R. Shea, J. Liu, E. C. H. Ngai, and Y. Cui. Cloud gaming: architecture and performance. *IEEE Network*, 27(4):16–21, July 2013.

[15] M. Jarschel, D. Schlosser, S. Scheuring, and T. HoÃfeld. An evaluation of qoe in cloud gaming based on subjective tests. In *2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 330–335, June 2011.

[16] R. Suselbeck, G. Schiele, and C. Becker. Peer-to-peer support for low-latency massively multiplayer online games in the cloud. In *2009 8th Annual Workshop on Network and Systems Support for Games (NetGames)*, pages 1–2, November 2009.

[17] Michael Jarschel, Daniel Schlosser, Sven Scheuring, and Tobias Hoßfeld. Gaming in the clouds: Qoe and the usersâ perspective. *Mathematical and Computer Modelling*, 57(11):2883–2894, 2013.

[18] Jardine Jared and Zappala Daniel. A hybrid architecture for massively multiplayer online games. In *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, NetGames '08, pages 60–65, New York, NY, USA, 2008.

[19] Luther Chan, James Yong, Jiaqiang Bai, Ben Leong, and Raymond Tan. Hydra: A massively-multiplayer peer-to-peer architecture for the game developer. In *Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support for Games*, NetGames '07, pages 37–42, New York, NY, USA, 2007.

[20] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, and Antony Rowstron. Scribe : A large-scale and decentralized application-level multicast infrastructure. *IEEE journal on selected areas in communications*, 20, October 2002.

[21] Knut-Helge Vik, PÃl Halvorsen, and Carsten Griwodz. Evaluating steiner-tree heuristics and diameter variations for application layer multicast. *Computer Networks*, 52(15):2872–2893, 2008.

[22] Jaecheol Kim, Jaeyoung Choi, Dukhyun Chang, Taekyoung Kwon, Yanghee Choi, and Eungsu Yuk. Traffic characteristics of a massively multi-player online role playing game. In *Proceedings of 4th ACM SIGCOMM Workshop on Network and System Support for Games*, NetGames '05, pages 1–8, New York, NY, USA, 2005.

[23] Tanja Lang, Philip Branch, and Grenville Armitag. A Synthetic Traffic Model for Quake3. *ACEâ04, Singapore*, June 2004.

[24] Kuan-Ta Chen, Polly Huang, and Chin-Laung Lei. Effect of network quality on player departure behavior in online games. *Parallel and Distributed Systems,IEEE Transactions*, pages 593–606, May 2009.

[25] Wu chang Feng, F. Chang, Wu chi Feng, and J. Walpole. A traffic characterization of popular on-line games. *IEEE/ACM Transactions on Networking*, 13(3):488–500, June 2005.

[26] Peng Chen and Magda El Zarki. Perceptual view inconsistency: An objective evaluation framework for online game quality of experience (qoe). *Network and Systems Support for Games (NetGames)*, Octobre 2011.

[27] Wladimir Palant, Carsten Griwodz, and Pal Halvorsen. Consistency requirements in multiplayer online games. *NetGames '06 Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games, USA*, pages 1–6, October 2006.

[28] Lamport Leslie. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, July 1978.

[29] Beigbeder Tom, Coughlan Rory, Lusher Corey, Plunkett John, Agu Emmanuel, and Claypool Mark. The effects of loss and latency on user performance in unreal tournament 2003. In *Proceedings of 3rd ACM SIGCOMM Workshop on Network and System Support for Games*, NetGames '04, pages 144–151, New York, NY, USA, 2004.

[30] Li Frederick W.B., Li Lewis W.F., and Lau Rynson W.H. Supporting continuous consistency in multiplayer online games. In *Proceedings of the 12th Annual ACM International Conference on Multimedia*, MULTIMEDIA '04, pages 388–391, New York, NY, USA, 2004.

[31] Sun Chengzheng, Jia Xiaohua, Zhang Yanchun, Yang Yun, and Chen David. Achieving convergence, causality preservation, and intention preservation in real-time cooperative editing systems. *ACM Trans. Comput.-Hum. Interact.*, 5(1):63–108, March 1998.

[32] Saldana Jose and Suznjevic Mirko. Qoe and latency issues in networked games. In *Handbook of Digital Games and Entertainment Technologies*, pages 1–36, August 2016.

[33] Quax Peter, Monsieurs Patrick, Lamotte Wim, De Vleeschauwer Danny, and Degrande Natalie. Objective and subjective evaluation of the influence of small amounts of delay and jitter on a recent first person shooter game. In *Proceedings of 3rd ACM SIGCOMM Workshop on Network and System Support for Games*, NetGames '04, pages 152–156, New York, NY, USA, 2004.

[34] Y. C. Chang, K. T. Chen, C. C. Wu, C. J. Ho, and C. L. Lei. Online game qoe evaluation using paired comparisons. In *2010 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR 2010)*, pages 1–6, June 2010.

[35] Henning Schulzrinne, Stephen L. Casner, Ron Frederick, and Van Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550, July 2003.

[36] Al Morton, Gomathi Ramachandran, Stanislav Shalunov, Len Ciavattone, and Jerry Perser. Packet Reordering Metrics. RFC 4737, November 2006.

[37] Transmission Control Protocol. RFC 793, September 1981.

[38] User Datagram Protocol. RFC 768, August 1980.

[39] Griwodz Carsten and Halvorsen Pål. The fun of using tcp for an mmorpg. In *Proceedings of the 2006 International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '06, pages 1:1–1:7, New York, NY, USA, 2006.

[40] Wu Chen-Chi, Chen Kuan-Ta, Chen Chih-Ming, Huang Polly, and Lei Chin-Laung. On the challenge and design of transport protocols for mmorpgs. *Multimedia Tools Appl.*, 45(1-3):7–32, October 2009.

[41] D. Burgos-Amador, J. Martinez-Cruz, and S. Recio-Perez. Reducing the impact of massive multiplayer online games on the internet using sctp. In *2013 10th International Conference on Information Technology: New Generations*, pages 8–13, April 2013.

[42] Andreas Petlund, Paul Beskow, Jon Pedersen, Espen Søgård Paaby, Carsten Griwodz, and Pål Halvorsen. Improving sctp retransmission delays for time-dependent thin streams. *Multimedia Tools and Applications*, 45(1):33–60, October 2009.

[43] Kuan-Ta Chen, Polly Huang, Chun-Ying Huang, and Chin-Laung Lei. Game Traffic Analysis: An MMORPG Perspective. *NOSSDAVâ05, Stevenson, Washington, USA*, June 2005.

[44] W.C.Feng, W.C.Feng F.Chang, and J.Walpole. A traffic characterization of popular on-line games. *IEEE/ACM Transactions on Networking,*, June 2005.

[45] V. Ramakrishna, Max Robinson, Kevin Eustice, and Peter Reiher. An active self-optimizing multiplayer gaming architecture,â autonomic

computing workshop. In *proceedings of the Fifth Annual International Workshop on Active Middleware Services (AMS)*, pages 32–41, 2003.

[46] H. . Dommel and J. J. Garcia-Luna-Aceves. Ordered end-to-end multicast for distributed multimedia systems. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, January 2000.

[47] Xiaohua Jia. A total ordering multicast protocol using propagation trees. *IEEE Trans. Parallel Distrib. Syst.*, 6(6):617–627, June 1995.

[48] H. Garcia-Molina and A. Spauster. Message ordering in a multicast environment. In *[1989] Proceedings. The 9th International Conference on Distributed Computing Systems*, pages 354–361, June 1989.

[49] Bartosz Bogdanski. Optimized routing for fat-tree topologies, 2014.

[50] Mikael Hirki. Applying compression to a game's network protocol. *CoRR*, abs/1206.2362, 2012.

[51] Wang Jinzhong. On packet aggregation mechanism for improving massive multiplayer online game performance in p2p network. In *2011 3rd International Conference on Computer Research and Development*, volume 2, pages 337–339, March 2011.

[52] J. Saldana, D. Wing, J. Fernandez-Navajas, J. Ruiz-Mas, M. A. M. Perumal, and G. Camarillo. Widening the scope of a standard: Real time flows tunneling, compressing and multiplexing. *IEEE International Conference on Communications (ICC)*, pages 6906 – 6910, June 2012.

[53] Jose Saldana, Jenifer Murillo, Julian Fernandez-Navajas, Jose Ruiz-Mas, Jose I. Aznar, and Eduardo Viruete Navarro. Bandwidth Efficiency Improvement of Online Games by the use of Tunneling, Compressing and Multiplexing Techniques. *Proc. International Symposium on Performance Evaluation of Computer and Telecommunication Systems SPECTS*, pages 227–234, 2011.

[54] Jose Saldana, Julian Fernandez-Navajas, Jose Ruiz-Mas, and Luis Casadesus. Online games traffic multiplexing: Analysis and effect in

access networks. *KSII Transactions on Internet and Information Systems*, pages 2920–2939, November 2012.

[55] J. Saldana. The effect of multiplexing delay on mmorpg tcp traffic flows. In *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*, pages 245–250, January 2014.

[56] J. Saldana, J. Fernandez-Navajas, J. Ruiz-Mas, J. I. Aznar, E. Viruete, and L. Casadesus. First person shooters: can a smarter network save bandwidth without annoying the players? *IEEE Communications Magazine*, 49(11):190–198, November 2011.

[57] Jonathan R. Stanton. A users guide to spread version 0.11. `http://www.spread.org`, October 2002.

[58] The Corosync cluster engine. `http://corosync.github.io/corosync`, March 2015.

[59] Appia communication framework. `http://appia.di.fc.ul.pt`, December 2015.

[60] B. Rajagopalan and P. K. McKinley. A token-based protocol for reliable, ordered multicast communication. In *Proceedings of the Eighth Symposium on Reliable Distributed Systems*, pages 84–93, October 1989.

[61] Kenneth P. Birman and Thomas A. Joseph. Reliable communication in the presence of failures. *ACM Trans. Comput. Syst.*, May(1):47–76, January 1987.

[62] D. A. Agarwal, L. E. Moser, P. M. Melliar-Smith, and R. K. Budhia. A reliable ordered delivery protocol for interconnected local area networks. In *Proceedings of International Conference on Network Protocols*, pages 365–374, November 1995.

[63] P. M. Melliar-Smith, L. E. Moser, and V. Agrawala. Broadcast protocols for distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, 1(1):17–25, January 1990.

[64] D. A. Agarwal, L. E. Moser, P. M. Melliar-Smith, and R. K. Budhia. The totem multiple-ring ordering and topology maintenance protocol. *ACM Trans. Comput. Syst.*, 16(2):93–132, May 1998.

[65] A. Babay and Y. Amir. Fast total ordering for modern data centers. In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pages 669–679, June 2016.

[66] R. Ahlswede, Ning Cai, S. . R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, July 2000.

[67] C. Fragouli and E. Soljanin. *Network Coding Fundamentals*. Foundations and trends in networking. Now Publishers, 2007.

[68] C. Fiandrino, D. Kliazovich, P. Bouvry, and A. Zomaya. Network coding-based content distribution in cellular access networks. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6, May 2016.

[69] C. Zhan and K. Gao. Video delivery in heterogeneous wireless networks with network coding. *IEEE Wireless Communications Letters*, 5(5):472–475, October 2016.

[70] Zunnun Narmawala and Sanjay Srivastava. Survey of applications of network coding in wired and wireless networks. In *National Conference on Communications*, pages 153–157, February 2008.

[71] J. Huang, H. Gharavi, H. Yan, and C. Xing. Network coding in relay-based device-to-device communications. *IEEE Network*, 31(4):102–107, July 2017.

[72] Fausto Vieira and Daniel E. Lucani. Chapter 9 - network coding and its applications to satellite systems. In Symeon Chatzinotas, Björn Ottersten, and Riccardo De Gaudenzi, editors, *Cooperative and Cognitive Satellite Systems*, pages 275–302. 2015.

[73] C. Fragouli, J. Widmer, and J. . Le Boudec. A network coding approach to energy efficient broadcasting: From theory to practice. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, pages 1–11, April 2006.

[74] S. Kokalj-Filipovic, P. Spasojevic, and E. Soljanin. Doped fountain coding for minimum delay data collection in circular networks. *IEEE*

Journal on Selected Areas in Communications, 27(5):673–684, June 2009.

[75] M. Ahn and Y. Kim. Network coding-based multicast scheduling for throughput enhancement in wireless ad hoc network. In *The International Conference on Information Networking 2011 (ICOIN2011)*, pages 188–193, January 2011.

[76] Takahiro Matsuda, Taku Noguchi, and Tetsuya Takine. Survey of network coding and its applications. *IEICE Transactions on Communications*, E94.B(3):698–717, 2011.

[77] H. Seferoglu and A. Markopoulou. Delay-optimized network coding for video streaming over wireless networks. In *2010 IEEE International Conference on Communications*, pages 1–5, May 2010.

[78] M. Esmaeilzadeh, P. Sadeghi, and N. Aboutorab. Random linear network coding for wireless layered video broadcast: General design methods for adaptive feedback-free transmission. *IEEE Transactions on Communications*, 65(2):790–805, February 2017.

[79] Farhan Jamil, Anam Javaid, Tariq Umer, and Mubashir Husain Rehmani. A comprehensive survey of network coding in vehicular adhoc networks. May 2016.

[80] Maxwell W. Anderson. Design and simulation of efficient network coding for minimizing the decoding delay in cellular networks. In *School of Engineering and Information Technology, UNSW Canberra at ADFA*, 2018.

[81] A. Le, A. S. Tehrani, A. G. Dimakis, and A. Markopoulou. Instantly decodable network codes for real-time applications. In *2013 International Symposium on Network Coding (NetCod)*, pages 1–6, June 2013.

[82] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. Xors in the air: Practical wireless network coding. *IEEE/ACM Transactions on Networking*, 16(3):497–510, June 2008.

[83] P. A. Chou and Y. Wu. Network coding for the internet and wireless networks. *IEEE Signal Processing Magazine*, 24(5):77–85, September 2007.

[84] H. Lim and C. Kim. Flooding in wireless ad hoc networks. *Computer Communications*, 24(3):353–363, February 2001.

[85] Wei Lou and Jie Wu. On reducing broadcast redundancy in ad hoc wireless networks. *IEEE Transactions on Mobile Computing*, 99(2):111–122, April 2002.

[86] L. Li, R. Ramjee, M. Buddhikot, and S. Miller. Network coding-based broadcast in mobile ad-hoc networks. In *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pages 1739–1747, May 2007.

[87] Y. Li, Y. Wu, B. Cao, L. Qiao, W. Zhang, and W. Tang. Retransmission mechanism with probabilistic network coding in wireless networks. In *2014 IEEE Global Communications Conference*, pages 4502–4507, December 2014.

[88] B. Cao, L. Qiao, and Y. Li. Stackelberg game theoretic approach for probabilistic network coding in retransmission mechanism. In *2014 IEEE 8th International Symposium on Embedded Multicore/Manycore SoCs*, pages 15–20, September 2014.

[89] Y. Sasson, D. Cavin, and A. Schiper. Probabilistic broadcast for flooding in wireless mobile ad hoc networks. In *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.*, volume 2, pages 1124–1130 vol.2, March 2003.

[90] Z. J. Haas, J. Y. Halpern, and Li Li. Gossip-based ad hoc routing. *IEEE/ACM Transactions on Networking*, 14(3):479–491, June 2006.

[91] Jörg Widmer, Christina Fragouli, and Jean-Yves Le Boudec. Low-complexity energy-efficient broadcasting in wireless ad-hoc networks using network coding. *First Workshop on Network Coding, Theory, and Applications (NetCod 2005), Riva del Garda, Italy*, 2005.

[92] S. R. Li, Q. T. Sun, and Z. Shao. Linear network coding: Theory and algorithms. *Proceedings of the IEEE*, 99(3):372–387, March 2011.

[93] T. Ho and D. Lun. *Network Coding: An Introduction*. Cambridge University Press, 2008.

[94] E. Erez and M. Feder. Convolutional network codes for cyclic networks. In *NetCod 2005*, April 2005.

[95] S. R. Li and R. W. Yeung. On convolutional network coding. In *2006 IEEE International Symposium on Information Theory*, pages 1743–1747, July 2006.

[96] Jiaqing Huang, Liang Wang, Tiyuan Zhang, and Hui Li. Constructing network coding in cyclic networks with the best linear independence. *2009 Fourth International Conference on Computer Sciences and Convergence Information Technology*, pages 818–823, 2009.

[97] E. Erez and M. Feder. Efficient network code design for cyclic networks. *IEEE Transactions on Information Theory*, 56(8):3862–3878, August 2010.

[98] Amr Alasaad, Hasen Nicanfar, Sathish Gopalakrishnan, and Victor C. M. Leung. A ring-based multicast routing topology with qos support in wireless mesh networks. *Wireless Networks (10220038)*, 19(7):1627–1651, 2013.

[99] Christina Fragouli and Emina Soljanin. Network coding applications. *Foundations and Trends in Networking*, 2(2):135–269, 2008.

[100] J. Xu and B.W. Wah. Consistent Synchronization of Action Order with Least Noticeable Delays in Fast-Paced Multiplayer Online Games. *ACM Trans. on Multimedia Computing, Communications, and Applications*, 13(1):1–44, December 2016.

[101] J. Wang and W. Yurcik. *A Multi-Ring Framework for Survivable Group Communications*. Defense Technical Information Center, 2004.

[102] J. Wang, W. Yurcik, Y. Yang, and J. Hester. Multiring techniques for scalable battlespace group communications. *IEEE Communications Magazine*, 43(11), November 2005.

[103] M. Baldi and Y. Ofek. A comparison of ring and tree embedding for real-time group multicast. *IEEE/ACM Transactions on Networking*, 11(3):451–464, June 2003.

[104] W. Zhang, L. Li, G. Han, and L. Zhang. E2hrc: An energy-efficient heterogeneous ring clustering routing protocol for wireless sensor networks. *IEEE Access*, 5, 2017.

[105] T. Karimireddy and S. Zhang. Guaranteed timely delivery of control packets for reliable industrial wireless networks in industry 4.0 era. In *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, July 2017.

[106] M. Dammak, Y. Boujelben, N. Sellami, and I. Andriyanova. Performance impact of packet multiplexing on massive multiplayer online games. In *Proceedings of the SCSC*, 2016.

[107] Amir Yahyavi and Bettina Kemme. Peer-to-peer architectures for massively multiplayer online games: A survey. *ACM Comput. Surv.*, 46(1):9:1–9:51, July 2013.

[108] Cong Duc Nguyen, Farzad Safaei, and Paul Boustead. Optimal assignment of distributed servers to virtual partitionsfor the provision of immersive voice communicationin massively multiplayer games. *Comput. Commun.*, 29(9):1260–1270, May 2006.

[109] M. Dammak, I. Andriyanova, Y. Boujelben, and N. Sellami. Routing and network coding over a cyclic network for online video gaming. *IEEE Communications Letters*, 22(6):1188–1191, June 2018.

[110] Yassine Boujelben, Andre Girard, and Jean-Charles Gregoire. A sequential algorithm for constructing delay-constrained multirings for multipoint-to-multipoint communications. *Telecommunication Systems*, pages 43–59, 2006.

[111] H. Gossain, C. D. M. Cordeiro, and D. P. Agrawal. Multicast: wired to wireless. *IEEE Communications Magazine*, 40(6):116–123, June 2002.

[112] M. Dammak, Y. Boujelben, and N. Sellami. Cycle-based routing protocol for device-to-device group communications. In *2018 14th International Wireless Communications Mobile Computing Conference (IWCMC)*, pages 1053–1058, June 2018.

[113] A. Asadi, Q. Wang, and V. Mancuso. A survey on device-to-device communication in cellular networks. *IEEE Communications Surveys Tutorials*, 16(4), 2014.

[114] H. Mao, W. Feng, and N. Ge. Performance of social-position relationships based cooperation among mobile terminals. *IEEE Trans. Vehicular Technology*, 65(5), 2016.

[115] David B. Johnson and David Maltz. Dynamic source routing in ad hoc wireless networks. 353, May 1999.

[116] B. Kaufman and B. Aazhang. Cellular networks with an overlaid device to device network. In *2008 42nd Asilomar Conference on Signals, Systems and Computers*, pages 1537–1541, October 2008.

[117] G. Fodor, E. Dahlman, G. Mildh, S. Parkvall, N. Reider, G. MiklÃ$^3s, and Z. TurÃ!‘nyi. Design aspects of network assisted device−to−device communications. IEEE Communications Magazine, 50(3) : 170 − −177, March 2012.

[118] R. Liu, G. Yu, F. Qu, and Z. Zhang. Device-to-device communications in unlicensed spectrum: Mode selection and resource allocation. *IEEE Access*, 4, 2016.

[119] W. Alliance. Wi-fi peer-to-peer (p2p) specification. *Wi-Fi Alliance Specification*, 1(1), January 2010.

[120] Jin Zhenghua, Wang Hong, and Yang Zhijia. Low Complexity Synchronization Algorithms for HART C8PSK. In Liangzhong Jiang, editor, *Proceedings of the 2011 International Conference on Informatics, Cybernetics, and Computer Engineering (ICCE2011) November 19-20, 2011, Melbourne, Australia*, pages 661–669, Berlin, Heidelberg, 2012.

[121] W. Sun, F. Brännström, and E. G. Ström. Network synchronization for mobile device-to-device systems. *IEEE Transactions on Communications*, 65(3):1193–1206, March 2017.

[122] S. Yang and R. W. Yeung. Coding for a network coded fountain. In *2011 IEEE International Symposium on Information Theory Proceedings*, pages 2647–2651, July 2011.

[123] Y. Keshtkarjahromi, H. Seferoglu, R. Ansari, and A. Khokhar. Device-to-device networking meets cellular via network coding. *IEEE/ACM Transactions on Networking*, 26(1):370–383, February 2018.

[124] N. Voskoboynik, H. H. Permuter, and A. Cohen. Network coding schemes for data exchange networks with arbitrary transmission delays. *IEEE/ACM Transactions on Networking*, 25(3):1293–1309, June 2017.

[125] Olfa Ben Rhaiem and Lamia Chaari. Information transmission based on network coding over wireless networks: A survey. *Telecommun. Syst.*, 65(4):551–565, August 2017.

[126] M. Halloush and H. Radha. Network coding with multi-generation mixing: Analysis and applications for video communication. In *2008 IEEE International Conference on Communications*, pages 198–202, May 2008.

[127] Desmond S. Lun, Muriel Medard, Ralf Koetter, and Michelle Effros. On coding for reliable communication over packet networks. *Physical Communication*, 1(1):3–20, 2008.

[128] Shravan Rayanchu, Sayandeep Sen, Jianming Wu, Suman Banerjee, and Sudipta Sengupta. Loss-aware network coding for unicast wireless sessions: Design, implementation, and performance evaluation. In *Proceedings of the 2008 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '08, pages 85–96, New York, NY, USA, 2008.

[129] Samih Abdul-Nabi, Ayman Khalil, Philippe Mary, and Jean-François Hélard. Efficient network coding solutions for limiting the effect of packet loss. *EURASIP Journal on Wireless Communications and Networking*, 2017(1):35, February 2017.

[130] X. Zhang and B. Li. Optimized multipath network coding in lossy wireless networks. *IEEE Journal on Selected Areas in Communications*, 27(5):622–634, June 2009.

[131] Mandar Chitre and Wee-Seng Soh. Network coding to combat packet loss in underwater networks. In *WUWNet*, 2010.

[132] R. Prior and A. Rodrigues. Systematic network coding for packet loss concealment in broadcast distribution. In *The International Conference on Information Networking 2011 (ICOIN2011)*, pages 245–250, January 2011.

[133] Szymon Chachulski, Michael Jennings, Sachin Katti, and Dina Katabi. Trading structure for randomness in wireless opportunistic routing. *SIGCOMM Comput. Commun. Rev.*, 37(4):169–180, August 2007.

[134] L. Lima, S. Gheorghiu, J. Barros, M. Medard, and A. L. Toledo. Secure network coding for multi-resolution wireless video streaming. *IEEE Journal on Selected Areas in Communications*, 28(3):377–388, April 2010.

[135] 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. *Comput. Netw.*, 57(6):1488–1502, April 2013.

[136] Hui Wang, Song Xiao, and C.-C. Jay Kuo. Random linear network coding with ladder-shaped global coding matrix for robust video transmission. *J. Vis. Comun. Image Represent.*, 22(3):203–212, April 2011.

[137] Chih-Ming Chen, Te-Yuan Huang, Kuan-Ta Chen, and Polly Huang. Quantifying the effect of content-based transport strategies for online role playing games. In *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, NetGames '08, pages 98–99, New York, NY, USA, 2008.

[138] E. Zihan, K. W. Choi, and D. I. Kim. Distributed random access scheme for collision avoidance in cellular device-to-device communication. *IEEE Transactions on Wireless Communications*, 14(7):3571–3585, July 2015.

[139] Arash Habibi Lashkari, Mir Danesh, and Behrang Samadi. A survey on wireless security protocols (wep, wpa and wpa2/802.11i). In *IEEE International Conference on Computer Science and Information Technology*, August 2009.