UNIVERSITY OF CALIFORNIA
Los Angeles

Fast Online Gaming over Wireless Networks

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Claudio Enrico Palazzi

2007

The dissertation of Claudio Enrico Palazzi is approved.

_____

Kung Yao

_____

Richard R. Muntz

_____

Leonard Kleinrock

_____

Mario Gerla, Committee Chair

University of California, Los Angeles

2007

To My Family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This PhD thesis has been developed as part of my joint PhD program. Thanks to the Interlink project, in fact, I am a PhD student in both the Department of Computer Science, UCLA, and in the Dipartimento di Scienze dell'Informazione, Università di Bologna. As part of my education and research work, every year I have spent a period in each of these two institutions, under the supervision of Professor Mario Gerla, and Professor Marco Roccetti, respectively. My first thanks goes hence to them for the great opportunity they gave me and for their fundamental guidance in these years.

My sincere gratitude goes to all the members of my dissertation committee: Professor Leonard Kleinrock, Professor Richard R. Muntz, and Professor Kung Yao, for their time, comments, and suggestions.

I cannot forget Giovanni Pau that conceived, first, this great adventure of a joint PhD. He has been a true friend, a great co-worker, and an unlimited source of astute suggestions.

Many thanks also to Stefano Cacciaguerra, Vincent Chavoutier, Jiwei Chen, Ling-Jyh Chen, Brian Chin, Shirshanka Das, Maria Fazio, Stefano Ferretti, Rosario Firrincieli, Salvatore, Frizzoli, Rafit Izhak-Ratzin, Uichin Lee, Eugenio Magistretti, Emiliano Manca, Cesar Marcondes, Gustavo Marfia, Daniela Maniezzo, Tammara Massey, Silvia Mirri, Alok Nandan, Fabio Parmeggiani, Anders Person, Paul Ray, Matteo Roffilli, and Cesare Roseti, Prof. Medy Sanadidi, Alexandro Sentinelli, James Stepanek, Tony Sun, et. al. With them my papers have been better works, my time in the office enjoyable, and my life richer of precious friends. My thanks have to be extended also to all the people in the Network Research Lab, Radio Lab, and Studio 16 for all the interesting discussion we had.

This work is dedicated with love to my family.

VITA

1974	Born, Thalwil, Switzerland (Italian citizen)
2002	B.S. (Computer Science). Università di Bologna, Italy
2005	M.S. (Computer Science). University of California, Los Angeles
2006	Ph.D. (Computer Science). Università di Bologna, Italy
2007	Ph.D. (Computer Science). University of California, Los Angeles

PUBLICATIONS

S. Ferretti, M. Roccetti, C. E. Palazzi, "Adaptive Playout Buffering Schemes for IP Voice Communication", in Encyclopedia of Information Science and Technology, Second Edition, Idea Group Inc., 2007, will appear in 2007.

M. Roccetti, C. E. Palazzi, S. Ferretti, G. Pau, "Wireless Home Entertainment Center: Protocol Communications and Architecture", Encyclopedia of Wireless and Mobile Communications, (B. Furht Ed.), CRC Press, will appear in 2007.

G. Marfia, C. E. Palazzi, G. Pau, M. Gerla, M. Y. Sanadidi, M. Roccetti, “TCP Libra: Exploring RTT-Fairness for TCP”, IFIP/TC6-LCNS NETWORKING 2007, Atlanta, GA, USA, May 2007.

M. Roccetti, M. Gerla, C. E. Palazzi, S. Ferretti, G. Pau, “First Responders' Crystal Ball: How to Scry the Emergency from a Remote Vehicle”, 1st IEEE International Workshop on Research Challenges in Next Generation Networks for First Responders and Critical Infrastructures (NetCri 07) - 26th IEEE International Performance Computing and Communications Conference (IPCCC 2007), New Orleans, LA, USA, Apr 2007.

R. Mazzeo, C. E. Palazzi, M. Roccetti, G. Sciutto, “Computer-assisted Pigment Identification in Artworks”, IASTED European Conference on Internet and Multimedia Systems and Applications, EuroIMSA 2007, Chamonix, France, mar 2007.

S. Ferretti, M. Roccetti, C. E. Palazzi, “An Optimistic Obsolescence-Based Approach to Event Synchronization for Massive Multiplayer Online Games”, International Journal of Computers and Applications, Acta Press/IASTED, vol.29, no.1, Jan 2007, 33-43.

M. Fazio, C. E. Palazzi, S. Das, M. Gerla, “Facilitating Real-time Applications in VANETs through Fast Address Auto-configuration”, 3rd IEEE CCNC International Workshop on Networking Issues in Multimedia Entertainment (CCNC/NIME 2007), Las Vegas, NV, USA, Jan 2007.

C. E. Palazzi, S. Ferretti, M. Roccetti, G. Pau, M. Gerla, "How Do You Quickly Choreograph Inter-Vehicular Communications? A Fast Vehicle-to-Vehicle Multi-Hop Broadcast Algorithm, Explained", Proc. of IEEE International Workshop on

Networking Issues in Multimedia Entertainment (CCNC/NIME 2007), Las Vegas, NV, USA, Jan 2007.

M. Fazio, C. E. Palazzi, S. Das, M. Gerla, “Vehicular Address Configuration”, 1st IEEE Workshop on Automotive Networking and Applications (AutoNet) - GLOBECOM, San Francisco, CA, USA, Dec 2006.

C. E. Palazzi, S. Ferretti, M. Roccetti, G. Pau, M. Gerla, “What's in that Magic Box? The Home Entertainment Center's Special Protocol Potion, Revealed”, IEEE Transactions on Consumer Electronics, IEEE Consumer Electronics Society, vol. 52, no. 4, Nov 2006, 1280-1288.

C. E. Palazzi, G. Pau, M. Roccetti, M. Gerla, “Digital Entertainment Delivery in a Wireless House: Time for a MAC Tuning”, China Communications, CIC, vol. 4, no. 5, Oct 2006, 94-101.

S. Ferretti, C. E. Palazzi, M. Roccetti, G. Pau, M. Gerla, “FILA in Gameland, a Holistic Approach to a Problem of Many Dimensions”, ACM Computers in Entertainment, vol. 4, no. 4, Oct 2006.

M. Fazio, C. E. Palazzi, S. Das, M. Gerla, “Automatic IP Address Configuration in VANETs”, Third ACM International Workshop on Vehicular Ad Hoc Networks (VANET 2006), MOBICOM 2006, Marina del Rey, Los Angeles, CA, USA, Sep 2006.

C. E. Palazzi, S. Ferretti, S. Cacciaguerra, M. Roccetti, “Interactivity-Loss Avoidance in Event Delivery Synchronization for Mirrored Game Architectures”, IEEE Transactions on Multimedia, IEEE Signal Processing Society, vol. 8, no. 4, Aug 2006, 874-879.

C. E. Palazzi, G. Pau, M. Roccetti, S. Ferretti, M. Gerla, "Wireless Home Entertainment Center: Reducing Last Hop Delays for Real-time Applications", 3rd ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE 2006), Hollywood, CA, USA, ACM, Jun 2006.

C. E. Palazzi, S. Ferretti, M. Roccetti, G. Pau, M. Gerla, "Buscar el Levante por el Poniente: In Search of Fairness Through Interactivity in Massively Multiplayer Online Games", 2nd IEEE International Workshop on Networking Issues in Multimedia Entertainment (CCNC/NIME 2006), Las Vegas, NV, USA, IEEE Communications Society, Jan 2006, 1183-1187.

C. E. Palazzi, "Fast and Fair Event Delivery in Large Scale Online Games over Heterogeneous Networks (Ph.D. thesis)", UniBo Technical Report UBLCS-2006-10, Mar 2006.

S. Ferretti, C. E. Palazzi, M. Roccetti, G. Pau, M. Gerla, "FILA, a Holistic Approach to Massive Online Gaming: Algorithm Comparison and Performance Analysis", 3rd ACM Annual International Conference in Computer Game Design and Technology (GDTW2005), Liverpool, UK, Nov 2005, 68-76.
*Awarded with the **Best Full Paper Award** by the GDTW 2005 Program Committee.*

G. Marfia, C. E. Palazzi, G. Pau, M. Gerla, M. Y. Sanadidi, M. Roccetti, "TCP Libra: Exploring RTT-Fairness for TCP", UCLA CSD Technical Report #TR050037, Sep 2005.

C. E. Palazzi, G. Pau, M. Roccetti, M. Gerla, "In-Home Online Entertainment: Analyzing the Impact of the Wireless MAC-Transport Protocols Interference", IEEE

International Conference on Wireless Networks, Communications and Mobile Computing (WIRELESSCOM2005), Maui, HI, USA, Jun 2005, vol. 1, 516-521.

C. E. Palazzi, S. Ferretti, S. Cacciaguerra, M. Roccetti, “A RIO-like Technique for Interactivity Loss Avoidance in Fast-Paced Multiplayer Online Games”, ACM Computers in Entertainment, vol.3, no.2, Apr 2005.

C. E. Palazzi, C. Roseti, M. Luglio, M. Gerla, M. Y. Sanadidi, J. Stepanek, “Enhancing Transport Layer Capability in HAPS-Satellite Integrated Architecture”, Wireless Personal Communications, Springer Science+Business Media B.V. (formerly Kluwer Academic Publishers B.V.), vol. 32, no. 3-4, Feb 2005.

C. E. Palazzi, S. Ferretti, S. Cacciaguerra, M. Roccetti, “A RIO-like Technique for Interactivity Loss Avoidance in Fast-Paced Multiplayer Online Games: a Preliminary Study”, 2nd ACM Annual International Workshop in Computer Game Design and Technology (GDTW 2004), Liverpool, UK, Nov 2004.

C. E. Palazzi, “Buddy-Finder: A Proposal for a Novel Entertainment Application for GSM”, 1st IEEE International Workshop on Networking Issues in Multimedia Entertainment (NIME'04), GLOBECOM 2004, Dallas, TX, USA, Nov 2004.

C. E. Palazzi, S. Ferretti, S. Cacciaguerra, M. Roccetti, “On Maintaining Interactivity in Event Delivery Synchronization for Mirrored Game Architectures”, 1st IEEE International Workshop on Networking Issues in Multimedia Entertainment (NIME'04), GLOBECOM 2004, Dallas, TX, USA, Nov 2004.

C. E. Palazzi, “Residual Capacity Estimator for TCP on wired/wireless links”, WCC2004 Student Forum IFIP World Computer Congress 2004, Toulouse, France, Ago 2004.

C. E. Palazzi, C. Roseti, M. Luglio, M. Gerla, M. Y. Sanadidi, and J. Stepanek, “Satellite coverage in urban areas using Unmanned Airborne Vehicles (UAVs)”, IEEE Semiannual Vehicular Technology Conference, VTC2004-Spring, Milan, Italy, May 2004.

ABSTRACT OF THE DISSERTATION

Fast Online Gaming over Wireless Networks

by

Claudio Enrico Palazzi

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2007

Professor Mario Gerla, Chair

Interactive, massive online games are becoming popular and begin to receive the attention of researchers. Wireless, mobile gaming represents a particularly interesting case because of the proliferation of smart personal devices, the increasing availability of high speed wireless access points, and the emergence of vehicular networks and applications. With this vision, we propose a holistic solution that enables a top quality online gaming experience regardless whether the player is wired, wireless, even riding a vehicle.

Main contributions of this work:

i) synchronization mechanism based on an Internet server overlay that enables scalability, fairness, and interactivity;

ii) elastic (TCP) and real-time (games) coexistence in a wireless access point;

iii) a fast multi-hop broadcast scheme for efficient game event delivery to players in vehicles.

CHAPTER 1

Introduction

The current status of the Internet as a widely utilized tool and the overwhelming development of wireless access technology lead us to a future in which the synergy between wireless and the Internet will be integral part of our everyday life. Virtual libraries, remote-working, video-telephony and voice over IP, traffic control, remote-medicine, video and music on demand, on-line games, location based resource discovery, navigation support, are only a few of the innumerable services that will be ubiquitously available [5, 6, 9, 10, 12, 37, 40, 42, 45, 46].

People will be continuously connected during the whole day, regardless of their location and utilizing a plethora of traditional or new devices. Furthermore, connectivity will follow the *always best connected* paradigm so that the connection will seamlessly switch from one access technology to another guaranteeing always the best conditions for customers (i.e., cost, bandwidth, coverage, personal preferences) [13].

We are crossing a technology threshold that will revolutionize every area of our lives. It will affect all of our everyday habits and businesses in ways far more pervasive than people may imagine. Devices that we use today for a limited range of special purposes will become multiple application platforms. Even common objects such as wristwatches, cars, PDAs are evolving and their enhancement toward multipurpose tools will accelerate as we move forward.

Wristwatch capabilities will be augmented making it able to communicate, download/play music, keep personal/medical information, identify us to our car/home/devices, etc. Cars will be elevated from a simple transportation vehicle to an office on the move, as well as an information provider and entertainment center. Passengers will be allowed to access the Internet, engage in teleconferencing, play

distributed videogames, learn location based information as low traffic paths to the destination or special offers for hotel reservations [114, 115, 116], participate in ad hoc or mesh networks [92], be part of an urban grid [93], etc.

As a proof for this forthcoming scenario, hot spots are rapidly increasing in number providing people with wireless connectivity in almost all the buildings they enter (e.g. home, work, cafeteria, etc.) and all the streets they walk and drive in. A large amount of static and nomadic users relies today on a wireless access to the Internet to run their favorite applications: email, web surfing, P2P file sharing, chatting, and video/audio downloading/streaming.

Mobile and highly mobile users will soon be using both traditional and innovative applications. Novel services will be provided whose utility sensibly rises for mobile users: location-dependent information, information exchange with people around, mobile market, safe driving alert system, urban grid for traffic control, Video/Voice over IP, Text/Voice-Chat on the move, and many others that we are still not able to imagine [16, 22, 23, 47, 82, 113].

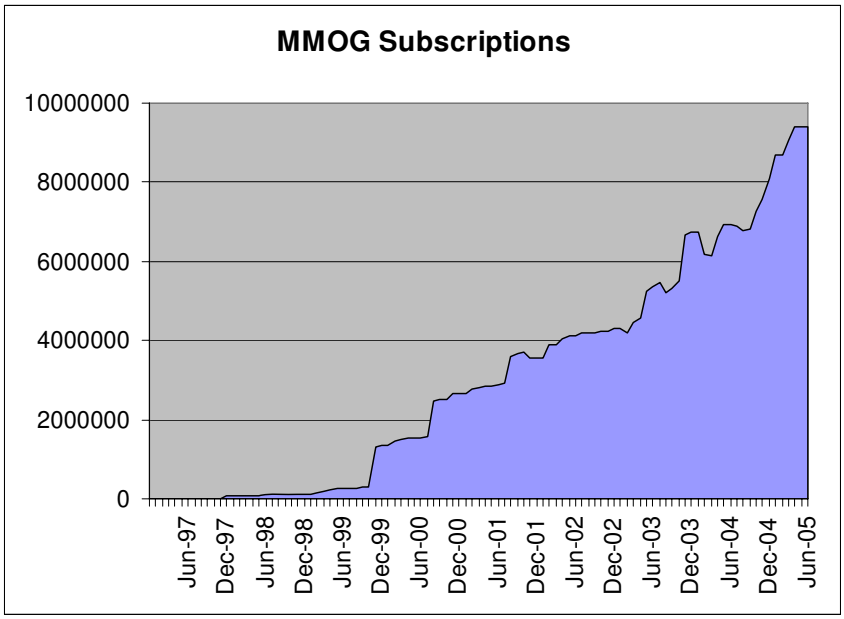


Figure 1.1: The exponential increase of MMOG subscriptions (1997-2005) [25].

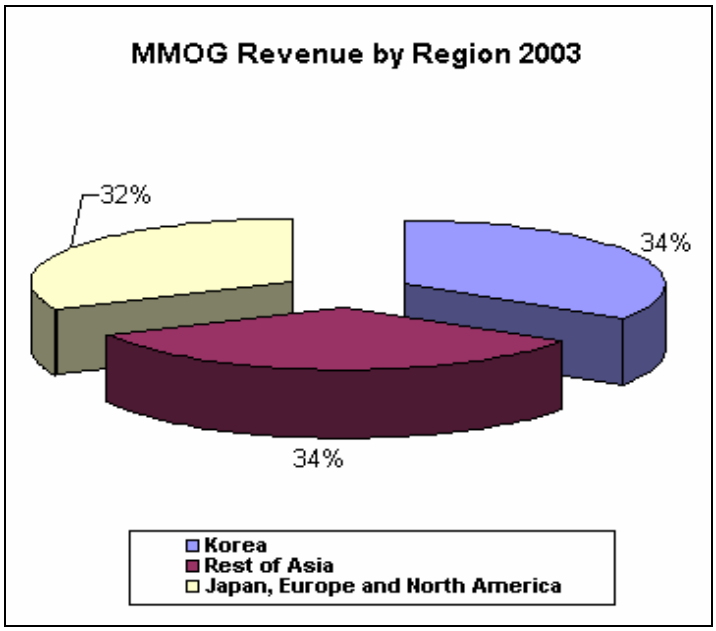


Figure 1.2: MMOG revenue by Region (2003) [25].

Houses will not be spared by this *wireless revolution*. Nowadays, home networking is still limited to few PCs and electronic equipments located in different rooms of the house; yet the vision for the near future includes many devices networked within a single household and connected to the Internet [117]. The *domotic philosophy* is steadily increasing the number of habitations which combine technology and services to improve living in the areas of safety, comfort and technical management. *Smart houses* will be endowed with devices, appliances and sensors, communicating through the means of wireless technologies and embodying active partners in managing our everyday life.

The same can be said by cars: they represent the next frontier in mobile communication for the forthcoming years. Since they are becoming more and more endowed with entertainment technology, it is not hard to foresee a future where automobiles will all be capable to connect to the Internet. However, vehicular networking needs to face several challenges before becoming a real everyday technology. The highly mobile scenario, in fact, affects the functioning of all the applications and traditional protocols, worsening the already known problems with wireless connections, and requiring the design of novel effective solutions. In this scenario several proxies along the road could be endowed with traditional or long range antennas to reach wireless cars

participating in the same virtual arena. Clients could also establish ad-hoc communication among themselves to create their own gaming network, or just to exploit other cars as bridges to the closest, even if not in range, proxy.

The trend of the market strongly suggests that entertainment applications are going to play a major role in these scenarios and, among them, video games are gaining more and more attention. In the last decade, in fact, thanks to their impressive progression in plunging players into terrifically realistic and capturing virtual worlds, videogames have expanded their market with a persistent and accelerating growth (see Fig. 1.1) [24, 25].

Furthermore, this market is still far from being mature and presents large growth margins. In fact, as can be seen in Fig. 1.2, a very large portion of MMOG revenues comes exclusively from South Korea (~50 millions of habitants). By projecting this value over the whole world and pretending that all the other developed country will eventually reach the same percentage of online players on their whole population, we can have a clue of the astonishing potentiality which hides behind this market.

Nowadays, two main reasons above the others attract an increasing number of researchers and practitioners toward the MMOG field. The first one is the explosive growth of the computer games market, with an increasing trend whose end is still not in sight [26]. The second reason is represented by the correlation between problems that emerge in developing innovative game experiences and those typical of various other conventional research fields in Computer Science and Engineering. Indeed, creating enjoyable online games requires the convergence of solutions belonging to extremely diverse technical areas. Examples are represented by networking, computer graphics, animation, music and sound, multimedia design, Artificial Intelligence (AI), human-computer interaction, software engineering, virtual environments and distributed simulation [27, 28, 29, 30, 31, 32].

Massively Multiplayer Online Games (MMOGs) are further extending the boundaries of what has been defined “the tenth art” with the possibility of contemporary engaging, in the same virtual scenario, millions of players located all over the world. Indeed, one of the main elements that determine the success of a game has always been represented by

the possibility to engage in multiplayer sessions. Humans are social beings and generally consider challenging other humans as funnier than playing alone against an artificial intelligence. Nowadays, the boom in the Internet usage has brought the logistic advantage of an always available virtual arena where millions of players can contemporary participate in electronic multi-user amusements.

Furthermore, the exploding market of connectable handheld devices, always looking for new killer applications, pushes the game industry to propose effective distributed game platforms proficient at engaging an unlimited number of contemporary users [33, 34]. This large and emerging market is driving researchers and practitioners to develop novel distributed solutions able to efficiently sustain interactive multiplayer networked game sessions over best effort networks such as the Internet [7, 35, 36, 38].

1.1 Problem Statement and Thesis Contribution

In this Thesis we focus on problems that arise in heterogeneous networks that involve also wireless users when trying to deploy a highly interactive entertainment system; in this context, MMOGs represents an emblematic and challenging example.

We define the *game state* in the system as the set of information that univocally describes the current configuration of the game. Players perform moves based on their perception of the game state. Specifically, their terminals periodically render a projection of the game state on screens. Players can hence be aware of the surrounding virtual world and influence it by generating *game events* which includes, but are not limited to, the movement of an avatar, hitting/missing a target, changing difficulty level.

In its simplest version, with no particular techniques employed, Game events are propagated from clients to some *decision point(s)* which utilize(s) them to determine the new game state that will be broadcasted. Decision point(s) could be represented by a centralized server, or a constellation of mirrored servers, or just other players, depending on the underlying architecture (see Section 2.6). Indeed, it is of particular interest to discuss potentially efficient game server architectures by analyzing network and

computational issues related to the maintenance of a consistent game state in the whole game platform.

```
typedef struct usercmd_s
{
  // Interpolation time on client
    short lerp_msec;
  // Duration in ms of command
    byte msec;
  // Command view angles.
    vec3_t viewangles;
  // intended velocities
  // Forward velocity.
    float forwardmove;
  // Sideways velocity.
    float sidemove;
  // Upward velocity.
    float upmove;
  // Attack buttons
    unsigned short buttons;
  //
  // Additional fields omitted…
  //
} usercmd_t;
```

Figure 1.3: Half-Life message format [171].

Residential broadband connectivity is currently becoming more and more common. Moreover, MMOGs generally utilizes very small packets thus easing the bandwidth requirement for this kind of application. As a confirmation, Fig. 1.3 shows the format of packets sent from client to server in Half-Life [74, 171]. Fields represent basic information that could be contained within few tens of bytes [146].

In this context, we intend to provide cutting-edge MMOG experience both to wired and wireless users, even when traveling in cars. To this aim, we propose a scalable

system able to support fairness and interactivity even when players are connected through the wireless medium. More in detail, in order to ensure optimal performance to players we have to split the problem into three sub-parts which requires specific solutions applied by different subjects.

The first sub-part regards the communications and synchronization among game servers and represents the portion of the total connectivity that can be handled by the MMOG service provider (either directly, or through ISP-domain managers). It is up to the MMOG service provider to deploy the ultimate MMOG technology in this portion of the game platform.

The second sub-part, is concerned with the links between game servers and their engaged players, thus including also the wireless hops. In a wireless home scenario, this portion of the connectivity is out of the control of the MMOG service provider: customers decide on their own whether they are going to subscribe cable/DSL connectivity, set up a wireless last hop, and/or install a particular Media Center. Yet, it is a critical part, as it generally includes the bottleneck of the connection and may include large queuing delays.

Instead, considering an infrastructured vehicular network, the last wireless link, with the relative Access Point (AP), may be controlled by the government, or by companies in charge of highways' services, or even by some local commercial store. However, certain locations may be poorly covered by APs, thus cars may have to relay on ad-hoc connectivity as a only way to communicate. The third sub-part of our system is hence represented by the multi-hop ad-hoc networking among traveling vehicles that are engaged in the same online game. We call this group of vehicles a *gaming car platoon*.

Therefore, to obtain an optimal solution we propose to proceed through successive steps and address the three sub-problems independently one from the others. Our mechanisms complement each other since their scopes are detached (even if connected, especially the first two) and, although they generate the best performance results when combined, they produce benefits even if singularly applied.

In particular, for the first part we exploit a hybrid architecture combining both the advantages of client-server and peer-to-peer paradigms. Our solution deploys over the network a constellation of communicating replicated *Game State Servers* (GSSs), each of which locally maintains a redundant version of the game state. Each GSS manages and updates its copy of the game state as follows: i) it collects game events coming both from its engaged players and from other GSS peers; ii) it forwards to all other GSSs the events generated by its connected players; iii) it updates the game state considering the set of received game events; iv) it finally delivers the newly updated game state to its connected players.

It goes without saying that, within this scenario, an efficient event synchronization scheme among GSSs needs to be employed to guarantee a consistent and responsive evolution of the game state. Indeed, one of the key factors in determining the success of an online game is represented by the ability to rapidly deliver events among the various GSSs. While simpler turn-based games do not have to face this problem, as only one player is allowed to perform an action at any given time, the task of providing players with responsiveness and real-time interactions is probably the most stringent requirement for MMOGs (especially for fast-paced MMOGs). In fact, in case of intense traffic in the network or when excessive computational loads are slowing down some GSSs, the game delivery activity turns out to be quite complex. As a consequence, the responsiveness of the distributed game system may be jeopardized.

With this in mind, we propose a scheme that takes inspiration from Active Queue Management techniques to maintain the game event delivery delays under a human perceptivity threshold and uplift the playability degree of MMOGs. Our scheme follows a holistic approach aiming at preserving also consistency and fairness. At the basis of our scheme lies the idea of exploiting the semantics of the game and, in particular, the notion of obsolescence. Simply put, obsolescence entails that during a game session some events can lose their significance as time passes, i.e., new actions may make the previous ones irrelevant. For example, where there is a rapid succession of movements performed by a

single agent in a virtual world, the event representing the last destination supersedes the previous ones (obsolete events).

Obsolescence allows the system to drop those game events that lose their importance during the game evolution. Discarding superseded events for processing fresher ones may be of great help for delay-affected GSSs. This means that, during the game event exchange activity, while responsive GSSs may deliver all game events to provide their connected clients with a fluent game state evolution, those GSSs that are experiencing loss of interactivity may skip the execution of obsolete events in order to speed up the event processing activity, thus gaining interactivity.

Upholding interactivity may be useful also to the aim of ensuring fairness. In fact, we demonstrate how our system is able to take advantage of the reduced transmission time to magnify the efficiency of a local lag-type algorithm in ensuring fairness without compromising interactivity. This represents a very important result since it contradicts the general belief that interactivity and fairness embodies antithetic objectives.

However, even if our scheme is proficient in maintaining a high degree of responsiveness among game servers, still problems may arise at the edges of the considered topology, where users in their homes or cars may be engaged in an online gaming through an AP (see Fig. 1.4). This represents the aforementioned second part of our problem.

Concurrent traffic may generate queues that build up at the last (or first) link of the connection, thus delaying the game event delivery. This problem is worsened in case of players relying on wireless connectivity, regardless whether from their homes or while traveling in vehicles. The wireless medium, in fact, is naturally prone to be easily shared by several contemporary users who may interfere with each other. The applications run in this context may vary and we demonstrate as some of these may be particularly harmful toward real time traffic (online games but also video-streaming, video-chats, etc.). In particular, we show how the very popular TCP-based FTP application for downloading files increases queuing delays to such an extent that responsiveness may be completely jeopardized.

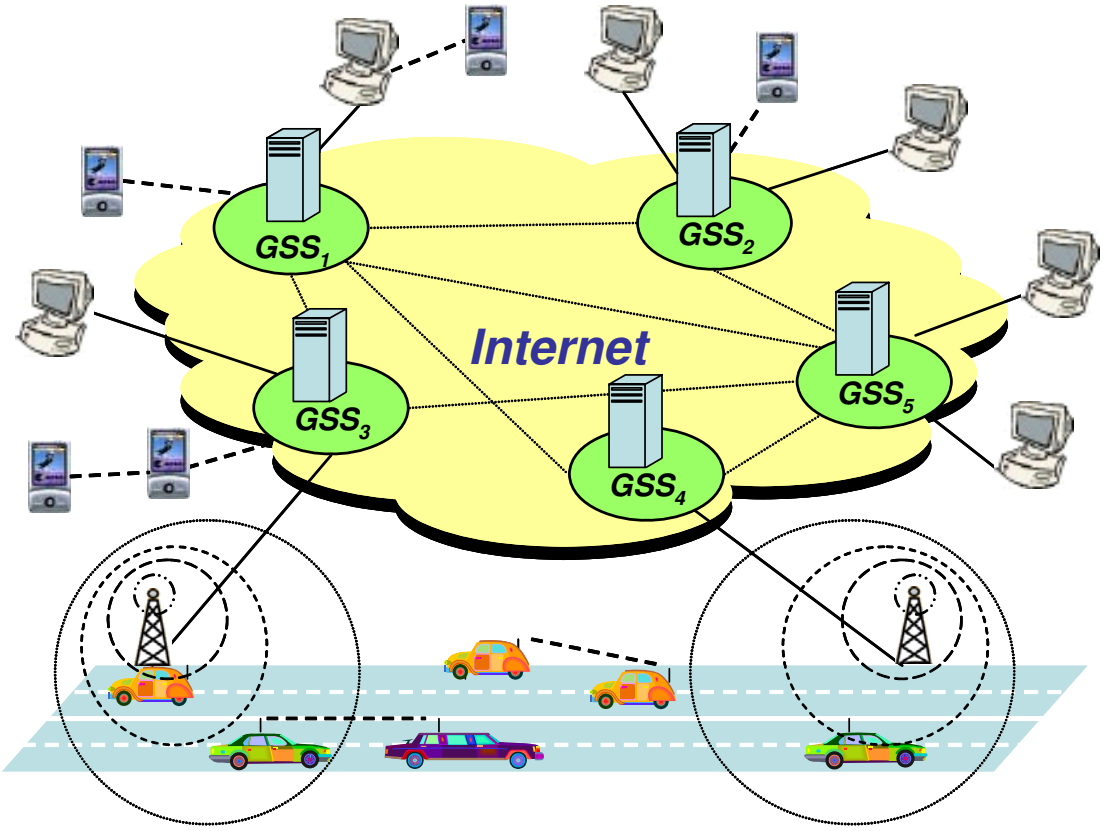


Figure 1.4: Hybrid architecture for distributed game entertainment in heterogeneous scenarios including vehicular networking.

To this aim, we propose the use of a *smart* AP that aims at achieving best performance for both elastic and real-time applications [203, 204] by appropriately limiting the advertised window for TCP flows. We evaluated the use of this solution both for a wireless home environment and for an infrastructured vehicular network.

Finally, as vehicular networks are still far from being supported by an epidemic set of APs specifically deployed to this aim, ad-hoc connectivity represents a fundamental resource to have players engaged in online gaming among themselves or to widen the coverage of sporadic APs through multi-hop wireless connections. Having players located in a group of cars moving in the same direction implies that game message exchange among vehicles in the same gaming car platoon will follow a many-to-many paradigm. In this context, the best solution to propagate game messages to all other

players is clearly that of having them sent in (multi-hop) broadcast over the vehicular network; moreover, the longer these broadcast hops, the quicker all the players will be reached by each game event.

Our proposal for this issue is that of utilizing priorities so that when a car sends a game event, the farthest car in its transmission range will be the one that will take upon itself the task of forwarding it onto the next hop. As a main contribution in this scheme, we have designed a transmission range estimator that works in a distributed way, just exploiting game message exchange, to have each vehicle aware of how far each game message will go and compute its own probability in becoming the next forwarder.

Summarizing in very few words, the contributions of this Thesis follow.

1. Synchronization mechanism for a mirrored game server architecture able to holistically support scalability, fairness, and interactivity by exploiting the semantics of the game. Furthermore, through our mechanism we contradict the general belief that interactivity and fairness are antithetic objectives. Instead, with the practical example represented by our solution we demonstrate that it is possible to improve (also) fairness by aiming at interactivity.

2. Smart AP able to make elastic (e.g., FTP/TCP) and real-time (e.g., games) flows coexist without affecting the performance of one another (i.e., throughput and delay). Indeed, it is well known in scientific literature that UDP-based real time flows can be harmful toward TCP-based elastic flows as the former may not implement any form of congestion control. However, we demonstrate here that even the latter can harm the performance of the former as TCP continuously probes the channel for more bandwidth, thus eventually generating queues (delays) on the connection. The efficacy of our scheme as been evaluated for both a wireless home and an infrastructured vehicular network.

3. Fast multi-hop broadcast scheme that exploits a novel transmission range estimator to quickly deliver game events to all players in a certain gaming car platoon. Thanks to our estimator, we are able to relax one of the most common, yet

unrealistic, assumptions made in scientific papers on this topic: “the transmission range is a constant for all vehicles known a priori”.

1.2 Thesis Outline

The remaining of this Thesis is organized as illustrated below.

In Chapter 2 we introduce the reader in the MMOG field by analyzing the most important issues in this research area. In particular, we overview key requirements and fundamental problems shared with traditional field in computer science, we summarize previous work that constitutes helpful background, we propose a model for MMOG delays, and we analyze possible system architectures.

Chapter 3 depicts the general framework that we use to guarantee fast synchronization among servers. It introduces the concepts of obsolescence and correlation and explains how to use them in combination with queuing management techniques in order to uplift the interactivity level of the system. Extensive results are provided to show the benefits attainable through our scheme.

Our proposed synchronization scheme is further refined in Chapter 4 where we show how increasing interactivity is not incompatible with aiming at fairness among players. On the contrary, the latter could be more easily achieved through the means of the former. We present experiments and outcomes that prove our assertion.

In Chapter 5 we describe the wireless home as an important scenario where MMOGs are going to play a major role. In particular, we demonstrate how MMOG performance could be affected by concurrent traffic of diverse nature. We then propose and evaluate different possible techniques, find in a *smarter* AP that we designed an efficient and effectively deployable solution (named SAP-LAW).

The infrastructured vehicular scenario is discussed in Chapter 6, where we assess our SAP-LAW solution in this context. Both the an urban and the highway contexts are evaluated to demonstrate its efficacy in provide efficient coexistence to heterogeneous flows even in a highly mobile environment.

Chapter 7 discusses the challenging scenario of online gaming over a VANET. Through an extensive set of experiments we demonstrate how our transmission range estimator factually permits to have vehicles prioritized so as to sensibly reduce the number of hops and transmissions, and hence delays, needed to propagate game events to all players in the gaming car platoon.

Finally, Chapter 8 concludes this Thesis.

CHAPTER 2

Background

We present a general introduction to the MMOG research field. In particular, we first discuss the major MMOG issues and wrong assumptions generally made by online game developers. We then explain fundamental problems that MMOGs share with other traditional field in computer science, thus elevating themselves to the rank of scientific topic. After this, we propose a model for delays in MMOGs. Finally, we analyze possible architecture for online games and survey related work in this area.

2.1 MMOG Issues

Several elements contribute to the success of a game: capturing graphics and visual effects, original plot, fair difficulty, multiplayer sessions, high interactivity level, etc. MMOG is, by definition, a class of games devised to engage a multitude of users, contemporary present in the same virtual arena even if physically located far away from each other. This scenario could be realized through the Internet; however, the best effort nature of the Net poses several challenges before being able to proficiently deploy really scalable and interactive game sessions.

In this sense, a MMOG is subject to similar major issues and fundamental principles which emerge when developing a general distributed application [75]. These are well summarized by Peter Deutsch's *"Eight Fallacies of Distributed Computing"* [130].

Essentially, developers building a general distributed application (or a MMOG) may fall in one or more of the following wrong assumptions generating a system that is intrinsically unable to guarantee high performance:

1. *"The network is reliable"*. Errors, losses, and wrong order delivery are common issues when utilizing a network to transmit messages. The larger is the scale of our MMOG, the higher is the probability that one of these events happens contradicting this assumption. Therefore, the application should be able to face these issues through mechanisms able to provide reliability without affecting the interactivity of the game system.

2. *"Latency is zero"*. Depending on the topology of the game network and on its conditions, delays may conspicuously increase. Objective delays as perceived by players are impacted both by network delays and by processing workload delays. At the same time, since MMOG requirements for interactivity and fairness, latency is the main issue to address in this kind of application.

3. *"Bandwidth is infinite"*. In a LAN scale network game this assumption may still hold since the available bandwidth is generally much larger than the required one to run a MMOG. On the other hand, when moving to wide area networks, congestion due to a very large and unpredictable traffic may generate bottlenecks that could invalidate this assumption.

4. *"The network is secure"*. Security in a large scale MMOG includes various issues that can be shared with traditional distributed systems or even be completely new. Authentication, subscription transactions, and cheating represent the most relevant among them.

5. *"Topology doesn't change"*. In a LAN, failures can suddenly lead to having isolated PCs that cannot anymore communicate among themselves nor, obviously, be engaged in a MMOG. Considering the larger scale case of the Internet, failures are addressed by the presence of redundant alternatives. Yet, the consequent topology change and congestion increase may noticeably impact on the system performance.

6. *"There is one administrator"*. MMOGs can be deployed across large geographic areas requiring a decentralized control of networking resources. Therefore, multiple domains and administrators may be involved in the resource allocation.

7. *"Transport cost is zero"*. The exchange of messages among nodes is a characteristic that belongs to the nature of a MMOG. These messages have to be forwarded through some infrastructure that needs to be provided and therefore bought/rented. Moreover, as the involved network spreads across different domains, resources among shared infrastructures have to be allocated to support the MMOG. Transport costs are hence inflated by providers' rates.

8. *"The network is homogeneous"*. The Internet is heterogeneous in nature and this characteristic is going to become even more evident in future. As new wireless and mobile technologies are coming into the picture, different network accesses, as well as different devices, are offered to players. Moreover, the core of the Internet evolves gradually and slowly since the cost and the large number of domains involved. Evolution at its edge, instead, proceeds by leaps and bounces, thus generating an interoperating mix of components that increases the heterogeneity of the infrastructure.

As a demonstration of the wide diffusion of these wrong assumptions, many games have been initially developed to be singularly played on PCs. Since more and more houses have became endowed with high speed connectivity such as cable and DSL, game vendors started to endow their products with the possibility of playing online with other users. However, this feature has been added as an extension of single use designed games, thus clearly inheriting the above wrong assumptions. Instead, MMOGs aimed at providing an amusing experience to their players have to be designed since the beginning, as intended to be played over the Internet and taking into account all the involved implications.

2.2 Fundamental Problems

Delving into the Internet, the first video game appears to be a simple *Tennis for Two* created by Higginbotham in 1958 to entertain visitors of the Brookhaven National Laboratory, a US nuclear research lab in Upton, New York. Since then, video games have evolved from a simple pastime into a real business and an important research field.

Indeed, certain games simulate, relatively cheaply and safely, situations which could be much more expensive or dangerous in the real world. Technical solutions introduced by game developers are now employed in medical surgery, military simulations, distributed simulations, virtual reality, interactive collaboration, distance learning, e-commerce and manufacturing systems. In this sense, several works have been presented that describe the convergence between game technology and non-game applications.

For example, interactive storytelling may gain benefits by employing synchronization schemes developed for online games [64, 125]. Collaborative applications require augmented reality interfaces to support group interaction [210]. Combat video games are utilized to enhance strategic, combat, and decision-making skills of military commanders [126, 129, 178]. Psychology is requesting for immersive collaborative virtual environments to investigate human behaviors and interaction [208, 209], as well as to support traumatized patients [121]. Real time video-streaming, voice over IP, and online games are all extremely delay sensitive applications that need fast delivery support [40, 99]. Finally, surgery simulations and video games share a quest for realistic object behavior, immediate response to given commands, and high-quality images [128].

These represent only a few of the innumerable intersections between video games and non-game applications. It is hence easy to see how issues emerging with online games represent fundamental problems also shared by other traditional fields in computer science, and how adapting elements of computer games may enable the creation of compelling user experiences in several domains [120].

Under a networking point of view, distributed multiplayer games are characterized by four main requirements which are intrinsically correlated and correspond to major

research challenges involved with MMOG, namely: interactivity, consistency, fairness, and scalability.

Interactivity (or ***responsiveness***) refers to the delay between the generation of a game event in a node and the time at which other nodes become aware of that event. Therefore, it includes both the network latency and the processing time. Having a high level of interactivity represents a fundamental quality for a MMOG. In order to assure an enjoyable playability to the final user, external stimuli generated by players need to be processed under a human-perceptivity threshold. This means that the time elapsed from the game event generation at a certain node and its processing time at every other node participating in the same game session must maintain a low average value. Unfortunately, not only could this be very hard to be accomplished in a best-effort network, but we will probably face also a high variance in the delivery time of game packets. Variable congestion conditions in Internet could, in fact, result in sudden slow down of the perceived game fluency on screen. Moreover, players in the same virtual arena can increase in number, even sensibly, with almost no predictability. Some game server may thus experience impulsive computational load and loose interactivity. These problems are obviously amplified when plunged into a wireless scenario.

Consistency, on the other hand, regards the contemporary uniformity of the game state view in all the nodes belonging to the system. Depending on the features of the game, consistency requirements may be absolute or partial. The easiest way to guarantee absolute consistency would be that of making the game proceed through discrete locksteps [98]. At each step, the system waits until having received all the actions generated by the final users; only at this moment a new instance of the game is produced and propagated to all the nodes. Having a single move allowed for each player and synchronizing all the agents before moving toward the next round, for sure grants absolute consistency but, on the other hand, impairs the responsiveness of the system. Obtaining both absolute consistency and high interactivity would require the employment of almost unlimited network and computation resources (very high bandwidth, very low

latencies, very high speed at server to process events). A trade-off between the two attributes needs to be found in order to develop a proficient game platform.

Fairness among users is another major issue in MMOGs. In fact, to ensure a rewarding game experience to users, every player needs to posses the same chances of winning than any other, regardless of the different network connections. Indeed, we are here interested in networking fairness for MMOGs as it represents one of the major issues that need to be addressed when developing new online games. In this context, relative delays have to be considered as important as absolute ones. Simultaneous game evolution with identical speed should be guaranteed as much as possible to all the participants [78]. To this aim, introducing appropriate artificial delays before displaying both generated and received game events may represent a feasible solution. However, similar to consistency, aiming at a full fairness may result in excessively increasing game delays, thus jeopardizing interactivity. Indeed, it is generally believed that interactivity and fairness/consistency embodies antithetic requirements [104, 105, 127].

Scalability regards the capability of the system to provide an efficient support to a large community of players. Regarding this point, it should be noticed that the interest of companies in online gaming emerges from the huge revenues that may be generated by a very elevated number of customers. Besides, humans are social beings which enjoy the presence of others in most of their amusement activities (i.e. team sports, movies in theatres) and the competition in challenging their skills against real adversaries. However, especially in the case of fast-paced MMOG, scalability is sometimes sacrificed to maintain a high degree of interactivity. In some cases, in fact, the system could deny the MMOG access to some users depending on their experienced delays [179]. These delays could have been generated by several factors such as the location of these users with respect to the server, the network conditions, and the current computational load of the system. Limiting the access to some customers obviously eases the achievement of interactivity, consistency, and fairness, but at the cost of a reduced scalability.

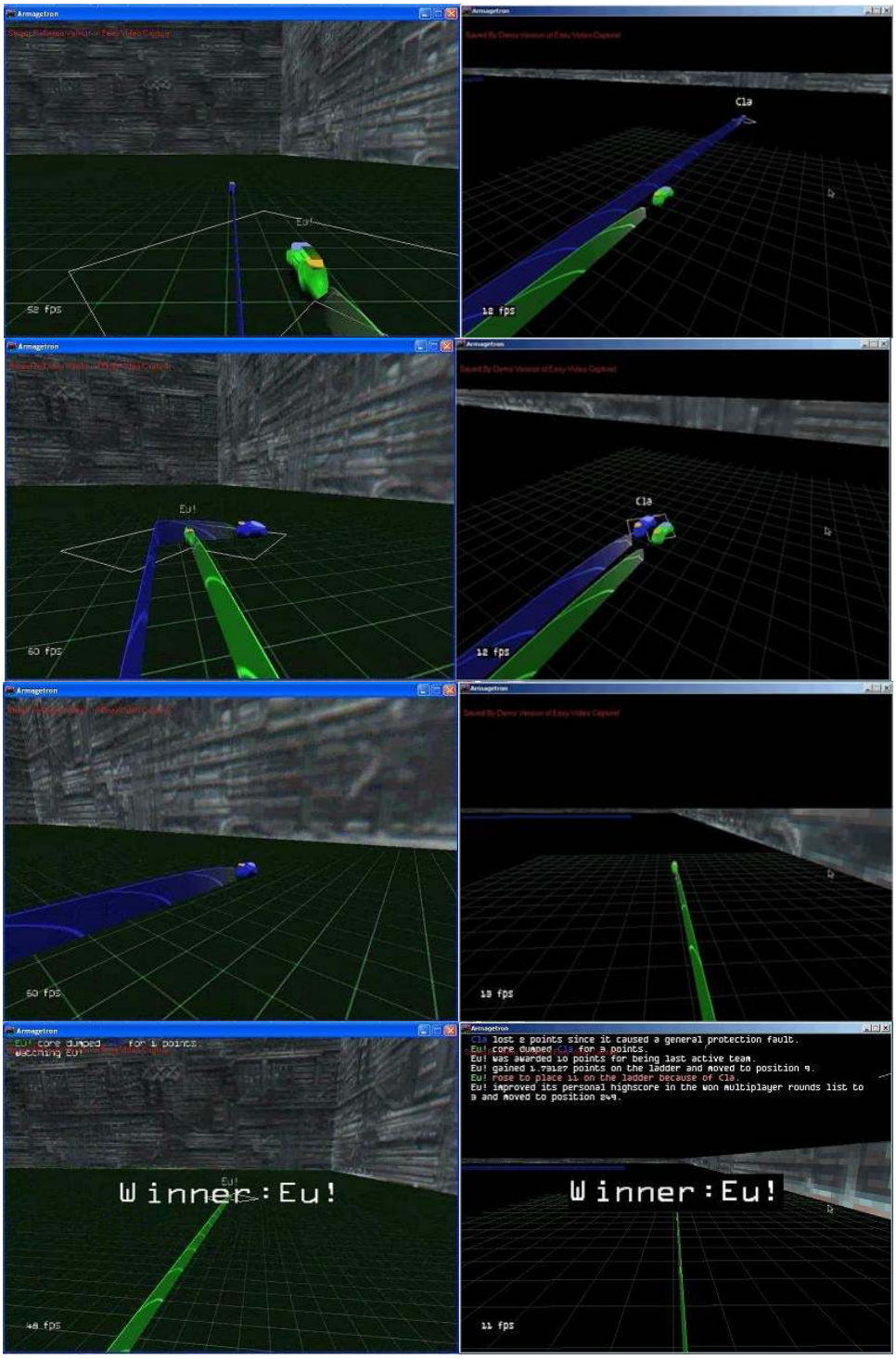


Figure 2.1: Frame sequence of an Armagetron game session: Cla's view (left column, blue player) vs Eu's view (right column, green player); evident lag differences generate inconsistencies and unfairness.

A well designed game architecture could help in devising a MMOG which possesses all the required qualities. The positioning of servers affects network latencies; thereby, they should be optimally located to efficiently serve their customers [76]. Moreover, every time a new scheme is proposed as a solution for MMOGs, all the four aforementioned key factors should be ensured and verified. Generalizing this concept, developers should follow a holistic approach when designing a new MMOG, considering the whole set of requirements and aiming at the intersection of their solutions. Addressing only one requirement, in fact, could produce the unexpected and undesired result of jeopardizing the others.

As a practical example of problems related to (different) delays in online games, we present in Fig. 2.1 the frame evolution of an Armagetron game session [211]. The game is based on the movie of the same name, released by Disney in 1982. Its rules are simple: each player controls a *light cycle* that leaves a wall behind it wherever the cycle it goes; the cycle cannot stop and can turn only at 90 degree angles. The goal of the game is to have all the other players crash into some wall while avoiding hitting others' own wall. Speed helps players in trapping other players; but the only way to speed up a light cycle is to drive very close to a wall.

In the left column of Fig. 2.1, there are frames as seen by a player, named *Cla* (blue cycle and wall), who connected to a certain game server with a 180 ms of RTT. Instead, frames on the right column correspond to the game as seen by a player, named *Eu* (green cycle and wall), who is connected to the same server with a 20 ms of RTT. Frames on the same row relates to the same instant of the game action; it is hence very easy to notice the inconsistency between the two game state views by simply comparing the position of one light cycle with respect to the other. In particular, during the second and third couple of frames in the sequence, player Cla believes he won (in frame 2 Cla sees Eu hitting his wall), whereas in the last frame he realizes that the server declared Eu as the winner.

This is a clear sign of inconsistency and unfairness. Player Cla would surely refrain from renewing his subscription to the game if he was paying money. Even if the game

was for free, player Cla would probably stop playing to avoid the frustration of sure, yet undeserved, failing.

The reason for this inconsistency and unfairness is in the different RTT experienced by the two game connections. Basically, server's view represent the real game evolution, whereas Cla and Eu visualize their own information by combining local player's movements with (differently delayed) game server's updates. As a result, Eu sees the game action to evolve in advance with respect to Cla, as the former is much closer to the game server than the latter. Furthermore, Eu's game almost identically corresponds to that of the game server, whereas Cla could see on his screen something different from the real game evolution. When considering fast paced games as in this case the consequences are very frustrating for the unlucky player.

2.3 Online Games: Related Work

Here, we present a commented summary of research contributions that have to be considered in our scenario. For each of them, we introduce the proposed technique and then highlight both the advantages and disadvantages of using it. In particular, we first evaluate existing techniques aimed at increasing the responsiveness degree of online games. We then analyze solutions proposed to ensure fairness and consistency (solutions developed to satisfy the fairness can generally be extended to satisfy consistency too, and viceversa). For the sake of completeness, we also provide a panoramic of anti-cheating mechanisms analyzing their impact on the other requirements. Finally, we are interested in efficiently extending the boundaries of MMOGs also to the wireless domain. Therefore, we discuss research about the impact of a wireless link on measured performance.

2.3.1 Responsiveness

Trying to improve the responsiveness of a distributed game architecture, two main causes for delays have to be analyzed: network latencies and computational costs. Several research works have already brought contributions to the factual developing of efficient

synchronization schemes. *Compression* and *aggregation* consider networking having a dominant position when dealing with the delays and thus with the playability of a MMOG [39].

More in detail, packet compression tries to speed up transmissions by reducing the amount of payload bytes to be transmitted. However, the provided benefits are very limited as the size of online game packets is already very small (see Fig. 1.3 and [146]). Aggregation, instead, merges packets together in the attempt of reducing the transmission overhead. Both compression and aggregation pay the achieved latency benefits with an increment in computational costs. Moreover, aggregation can generate further waste of time if a transmission is delayed while waiting for having available other events that could be aggregated.

In the attempt of reducing both the traffic load in the network and the computational cost to process each game event, *interest management* techniques rely on the *area-of-interest* concept to filter events. Players, in fact, are not impacted by game events happening far away from their current virtual position. Therefore, sending all the game events to all the players represents a waste of network and computational resources with no utility counterpart. Instead, an appropriate multicast based scheme could match every packet with the nodes that really need to receive it in order to reduce both the traffic on the channel and the processing burden at each node.

On the other hand, a tradeoff exists between the computation spared at the destination by receiving only a limited number of packets and the required one for implementing the filtering scheme at the sending GSS. Moreover, interest management techniques could further delay packet delivery when applied to games having almost all the game actions to be forwarded to the majority of the participants. Finally, area-of-interest techniques do not help in reducing latency as there is no correspondence between players' position in the virtual world and their actual location in the real one. In simple words, players positioned in the same virtual area-of-interest might be physically located very far from each other, thus not gaining any benefits in terms of propagation and queuing delay.

Slightly detaching playability from the real responsiveness of the network, *optimistic* algorithms for synchronizing game state at servers can be utilized in order to avoid delay perception at destination. In case of high delays in events forwarding between GSSs, in fact, an optimistic approach may execute game actions on clients before really knowing if ordering would require processing other on-the-way ones first. Game instances are thus processed without wasting any time in waiting for other packets that might arrive later. On the other hand, this performance gain is paid with some occurrence of temporary consistency loss. Standard *Time Warp* and *Breathing Time Warp* represent typical exemplars of this family of algorithms [52, 53, 105]. *Rollback* based techniques are exploited to reestablish the consistency of the game state but may further impact on the responsiveness of the system.

Based on this concept, a new synchronization mechanism for online games called *trailing state synchronization* was presented by *Cronin et al.* [59]. According to this approach, every GSS locally maintains a fixed number of copies of the game state, each of which is kept at a different simulation time. In essence, each copy of the game state is associated with a particular execution, and each execution is delayed for a fixed time interval. If no inconsistencies are detected, the game proceeds in rendering the copy of the game state which is more forward in time; as a result, players perceive the game evolution as being very responsive.

With this trailing state synchronization, inconsistencies are identified by comparing the leading state (that optimistically processes game events without any additional delay) with the game states of the delayed executions that reorder and then process the received game events. If an inconsistency is detected, a rollback is performed by copying the (consistent) game state from the delayed execution to the leading execution and then the rolled back game events are re-processed in the proper order. It goes without saying that a tradeoff relationship exists between the number of game state copies maintained by each GSS and the number of game events that need to be re-processed in case of a rollback. Moreover, this scheme does not avoid short-term inconsistencies: it just provides a mechanism to detect and correct them after they happened.

Optimistic algorithms can be employed also at the boundaries of the game platform. Clients could, in fact, be equipped with the intelligence required to perform predictions of other players' movements. Obviously, unless utilizing a fully distributed architecture, servers have still to validate clients' predictions. If the prediction results wrong, the server(s) will eventually correct the mistake and restore a consistent game state. However, a full Round Trip Time (RTT) may elapse between the generation of an inconsistency due to wrong prediction and its correction. If this period of time surpasses the human perceptivity threshold, players could be annoyed by it.

In general, prediction techniques rest upon the assumption that a consistent game state exists at some point. From this starting point, new actions are predicted and executed on clients' screen. For instance, 50 frames per second represent a typical refreshing rate for an online game and correspond to one new game state sent from the server to the client every 20ms. Considering a connection of 100ms of RTT, this amounts to 5 game events that could be subject to rollback if the server does not confirm the correctness of the predictions [171].

A tradeoff between the speed of executing game actions and the frequency of short-term inconsistencies is hence present in this kind of solutions. Moreover, client side prediction techniques require that clients and servers share the same predictive code; this solution could hence be not applicable on clients equipped with low computational resources. Finally, having portions of code shared between all clients and servers gives an advantage to malicious players. Cheaters, in fact, could become aware of the next expected (and predicted) actions of regular players and modify their behaviors in order to obtain the maximum benefits out of it.

Extrapolation and *Interpolation* are particular instances of the optimistic approach. With extrapolation, the next action and position of a virtual object (even an avatar) are predicted by exploiting the current available information (i.e. current position, speed, direction). Actions can thus be rendered on the screen before really receiving the related game event.

In games where physical rules of the real world are respected, the task of predicting events forward in time is made easier. Unfortunately, most of the games available in commerce present very poor similarities with real physical laws or ballistic models. Indeed, virtual object in games are usually allowed to perform sudden actions featured with huge acceleration/deceleration and instantaneous change of direction. These unrealistic movements cannot be anticipated by extrapolation and result in inconsistencies that affect the player's perception of the game evolution. Depending on the extension of the extrapolated time, these errors can be more or less evident. Therefore a tradeoff exists between the impact of these inconsistencies and the augment of responsiveness that can be simulated by this scheme.

Interpolation is another technique that tries to guess the position/action of a virtual object at a certain moment without having precise information about it. In particular, interpolation uses exact information about two valid positions/actions of an object at two different moments to determine how to display the in-between movements of that object.

When interpolation is employed the constraint for reliability in game event delivery can be relaxed. For instance, if some of the movements of an avatar, leading from an initial to a final position, are lost, the system is still able to reconstruct the whole sequence. However, some critical movements cannot be interpolated without creating inconsistencies, thus requiring reliable transmission (see the notion of correlated events in Section 3.2).

2.3.2 Consistency and Fairness

The simplest way to enforce fairness and consistency in the game is that of proceeding through discrete *locksteps* [98]. Simply stated, the game evolves by marching in step and players have to wait their turn before making any action. Every game event is thereby reliably received by all the players before any new move could be instantiated. Consistency and fairness are hence ensured by the fact that before proceeding to any new step, all the nodes in the game platform will share the same identical game state view.

Unfortunately, this scheme is affected by some important drawbacks. The most relevant one is the fact that locksteps cannot be used for interactive games since updates

are displayed on screens only after having received game events from all the players. This requires reliable transmission of game events and may sometimes require many seconds, thus making the game proceed very slowly. Moreover, if one node in the system fails, then all the others could wait indefinitely before receiving updates from that node and impeding any progress in the game evolution.

In [71, 105, 131], *Mauve et al.* presented an optimistic approach for the consistency control in networked multiplayer games. Their approach utilizes the local lag control mechanism combined with a modified Time Warp (executed only when necessary) in the attempt of solving the trade-off relationship existing between responsiveness and consistency.

In particular, *Mauve et al.* devised a synchronization approach based on the idea of intentionally decreasing the responsiveness of the application in order to eliminate short-term inconsistencies. Exploiting a local lag approach, game events are delayed for a certain amount of time before being executed by clients. This additional delay permits the reordering of the received events in order to minimize inconsistencies. However, the presence of this delay is not always sufficient to prevent short-term inconsistencies as game events might still arrive late due to jitter or packet loss in the network.

In [137], *Li et al.* presented a continuous consistency control mechanism for supporting networked multiplayer games. Similar to [105], this approach rests on the idea that game events should not be considered as discrete updates of the game state, but rather as continuous actions in the game world. In this context, the authors proposed a relaxed time-dependent consistency control scheme which gradually synchronizes the nodes in the system and ensures that the possible discrepancy among game states stored at each node never exceeds a predefined threshold.

As it is evident, this scheme does not avoid short-term inconsistency. Rather, it deals with them after a certain period of time to avoid that the divergence among the various game state views does not become excessive. However, players' gaming experience could still result affected.

Other approaches simply assume that delaying the game event processing activity for a predetermined amount of time may be sufficient to guarantee a uniform evolution of the game state at different nodes without any need to resort to rollback. Following this idea, *Diot and Gautier* depicted in [132] a synchronization mechanism which embodies an optimistic version of the well known conservative *bucket synchronization* algorithm.

Their approach assumes that there exists a processing deadline (which defines a time bucket) and that a correct evolution of the game requires that all game events are received prior to this deadline. In essence, the idea behind this scheme is that of ordering and processing game events at the end of the time bucket. If some game events are not received before the time bucket expiration, *dead reckoning* techniques are exploited to compensate event losses [134, 135]. A similar approach was presented also in [133]; in both cases, the main drawback is that dead reckoning does not ensure the full consistency of the distributed game state.

Short-term inconsistencies may still arise if the bucket size is set excessively small but, on the other hand, using large buckets may induce severe responsiveness degradation. Further, a complex problem is that of fitting the bucket size with the unstable condition of the Internet where large and variable jitter values may be experienced.

Fairness among users is a very important issue in online games. Every player has to possess the same chances of winning than any other. To this aim, it is fundamental that delays impact on all participants in equal measure. Fairness could be enforced by compensating latency differences among players with an appropriate queuing of game events at each client before presenting them onto the screen [54, 78, 94, 104, 105]. This artificial injection of an appropriate amount of delay is usually referred to as *Local Lag* technique.

In essence, this scheme normalizes the game state so as to have all the players virtually brought back at the time when a certain game event was generated. Unfortunately, pure Local Lag algorithm does not solve the tradeoff between interactivity and fairness anticipated in Section 2.2.. Lags, in fact, can be inflated only up to the delay

perceptivity threshold; otherwise all the players will be affected by excessive delays which jeopardize the interactivity degree of the whole game platform.

Finally, aiming at ensuring consistency, *Knutsson et al.* propose in [136] the use of coordinators to solve update conflicts that may occur in a peer-to-peer architecture. In particular, they split the game state management into different classes handled by different coordinators within a group of interested peers. Aiming at guaranteeing fault tolerance, they also propose the use of a primary-backup protocol to address possible fail-stop failures of coordinators. However, several issues arise in this context such as, coordinator election, fairness maintenance among different kind of nodes, authentication, and cheating avoidance from malicious intruders that become coordinators.

2.3.3 Cheating

A deep analysis on cheating and proposed solutions is out of the scope of this Thesis. Nonetheless, for the sake of completeness we provide a short panoramic of work on this topic. Moreover, research in this area generally proposes protocols which also include solutions aimed at sustaining other key requirements such as interactivity, consistency, and fairness. Conversely, none of the schemes reviewed till now proposes any cheat-proof mechanism.

A general taxonomy of cheating in online games can be found in [169]. A more detailed one, adopting a classification based on networking layers and then further categorizing the *protocol level cheats* class, is presented in [167]. In [167], the authors also present NEO, a low-latency event ordering protocol for a distributed architecture. Designed to improve the responsiveness of the game while preventing protocol level cheats, NEO takes inspiration from bucket synchronization and divides the time into equal intervals called *rounds*. Within these rounds, players send encrypted updates to each other and, in the following round, players send key to decrypt the updates sent. After this, a voting mechanism is employed to achieve *majority based* consistency and responsiveness.

Specifically, every new game state update is accepted by players only if a majority of them has received it on time. Consistency is hence maintained through a distributed

voting system which collects positive votes from players who have received the game state update in time, and negative votes from delay affected players. In order to avoid interactivity loss, NEO does not wait for receiving all the votes. The relationship between game progression and communication reliability is weakened, any non-received votes at the end of the round are considered *abstentions*, and only the receiving of a majority of votes is required to consider the election as valid.

Therefore, NEO does not guarantee continuous global consistency or interactivity. Rather, its aim is that of ensuring these properties for a majority of players, and utilizing dead reckoning techniques in order to adjust the game state view for the rest of them (a minority). Obviously, depending on the definitions of majority and minority a different tradeoff can be found for interactivity and consistency. However, the length of a round is limited by the maximum latency acceptable to still consider the system as responsive, and no effort is devoted to facilitate a more efficient solution for this tradeoff by effectively decreasing the delays caused by traffic conditions. On the contrary, a bucket synchronization mechanism which also requires multiple rounds in order to send updates, encrypt them, and voting for determining their consistency, definitely affects the responsiveness of the system, and is therefore not well suited for highly interactive games.

An extreme solution for the protocol level cheats makes use of the aforementioned concept of locksteps. In particular, *Baughman and Levine* design in [168] a protocol that forces players to reveal their moves before actually performing them and, most important, before receiving any plain-text move from any other player. Thereby, this scheme prevents anyone from performing late changes of their moves based on the knowledge about other players' actions.

Two steps characterize this protocol. First, each player has to communicate her/his next move utilizing an encrypted message; the move is hence *committed* even if not revealed ahead of time. Finally, each player sends her/his move in clear. The authors also propose the utilization of area-of-interest techniques to reduce the overhead in the system caused by the reliable transport of commitment messages. Nonetheless, any lockstep

based mechanism leaves unresolved several problems and, among the others the most critical one is the lack of responsiveness.

Another method based on locksteps is presented in [170]. In this work, *Chen and Maheswaran* developed a mechanism for P2P architecture composed by two protocols: the first one aimed at ensuring fairness among all players regardless of their latencies, and the second one able to prevent certain types of time-cheats.

The fairness ensuring protocol makes use of specific nodes called *pulser*, and of network *sensors*. The former are elected nodes which periodically broadcast game state updates (*pulses*), while the latter detect the current status of the network and adapt the pulses sending rate in order to meet the fairness requirement.

Pulsers are also used to implement encryption on game messages in order to avoid time-cheats. In particular, they forward encrypted messages at a pace which corresponds to that one achieved by the slowest player. The authors demonstrate how this is sufficient to prevent faster players from maliciously peek into the future with respect to slower players.

Unfortunately this scheme presents several lacks. First, there is no real guarantee about the fine precision of the estimated network conditions; this is particularly true in the case where nodes are dispersed and sensors are not in close proximity of each of them. Second, this scheme works only if the whole structure (i.e., sensors and pulsars) can be adequately distributed over the Internet. Indeed, providing efficient clusterization of the nodes with respect to the sensors and the pulsers remains an open problem for this scheme. Finally, the anti-cheating mechanism only addresses time-cheats, while many other possible ways to gain unfair advantages have not been treated.

2.4 Wireless Scenarios

We are witnessing a continuous proliferation of access point all around us; through cell phones, smart devices, and laptops the communication world has become nomadic and mobile, even at vehicular speeds. This popularity created an everyday life style for people

that cannot anymore renounce to their connectivity freedom and are, instead, asking for more: more mobility, more contents and services, more bandwidth, etc.

In this section, we discuss WLANs through its functionalities, scenarios, applications, protocols, and problems.

2.4.1 Infrastructure vs Ad-hoc

Wireless networks can be divided into two main classes depending on whether they use fixed infrastructures or not. Infrastructured wireless networks extends the Internet over the wireless domain. Wireless nodes connect to an AP to have access to all Internet services and each AP offers to all engaged wireless nodes the same functionalities. A mobile node could move out of the transmission range of an AP, thus losing its connectivity to the Internet. In this case, if another AP is available in the new location, the node might connect to Internet again through the new AP; however, in the mean time, the wireless node has probably lost its ongoing sessions. This problem can be solved through a smooth handoff that seamless transfers the connectivity from the old to the new AP before disconnection take place. A well known protocol to perform this task is represented by Mobile IP that transfers packets from the old AP to the new way through routing triangulation [195].

Ad-hoc networks are composed by several nodes with wireless connectivity capabilities that connect one another to establish communications without the need of any AP. Every node in the network is able to communicate with any other node in its transmission range. Transmissions can happen both directly between two end hosts that are close to each other, or through multi-hop when a node needs to send a message to another node that is out of its range. Ad-hoc networks can be composed by static nodes but the most challenging case is clearly when nodes move: they are also known as Mobile Ad-hoc Networks (MANETs). Ad-hoc networks have received the attention of researchers and practitioners since their high deployment flexibility, low cost, and robustness, which make them perfectly suitable for a whole plethora of scenarios where the infrastructure is missing, e.g., away from towns, in areas hit by a major disaster or just not covered by APs, in military battlefields.

Currently, a hot research topic is represented by Vehicular Ad-hoc Networks (VANETs) because of the challenges involved in having communicating nodes traveling at very high speed. Indeed, traveling vehicles could easily find themselves in a situation with no AP along their road. Yet, several useful applications could be run over an ad-hoc network connectivity: safe driving, text/audio/video chats or online gaming among passengers sit in vehicles within a certain geographical area, peer-to-peer file sharing, etc. Depending on the application, two possible transmission models have to be considered: the *pull model* and the *push model*. With the former nodes (vehicles) explicitly asks to receive certain data from another node, whereas with the latter messages are proactively broadcast to every node in a certain area of interest. Both models can also involve multi-hop in their transmissions. Since the shared nature of the wireless channel, communications involving several nodes in the same VANET (e.g., online games) are clearly more efficient if performed through broadcasting (i.e., pull model) instead of sending as many messages as the number of recipients.

Focusing on the two main scenarios we are interested in, handoffs are not very frequent in a home environment, where a single AP is generally enough to provide connectivity to every room; instead, they represent a critical issue in a vehicular network as cars' high mobility make them stay in the coverage area of each AP for a very limited time (even just few seconds). Moreover, ad-hoc networking is rarely utilized in a home scenario where an AP is generally available, whereas it can often represent the only connectivity option with vehicular scenarios, especially as long as very few APs are available along roads.

2.4.2 Elastic vs Real-time Applications

Elastic applications have been the earliest in appearing on the Internet scene, they still represents the majority of Internet traffic and are naturally going to play a major role even in future vehicular networks. Prominent examples are the email (SMTP), the World Wide Web (HTTP), and file downloading (FTP). These applications do not require any specific minimal amount of bandwidth to work, yet, they try to utilize as much bandwidth as possible and the more the bandwidth available is, the better they perform.

Wireless environments generally affects elastic applications. Indeed, disconnections and error losses are very badly tolerated by these applications and can cause sever performance degradation. At the same time, delays are not of crucial importance, especially when considering the single packet delivery.

Real-time applications are becoming more and more popular among users; think for instance to Skype and World of Warcraft phenomena or, more in general, to video streaming, IP-telephony, teleconferencing, online gaming, interactive IP-radio/TV, and e-learning. The utilization of these services is growing everyday thanks to their entertaining nature.

Regardless of whether they are utilized through a traditional client-server architecture or through a P2P paradigm, the main requirement for these interactive applications is embodied by their little tolerance to delays [76]. On the other hand, sporadic losses do not affect their performances. The bandwidth required by real time applications is higher for rich-media applications such as high quality movie streaming, however, most of interactive applications, and online gaming is among these, only require a continuous availability of a certain (little) amount of bandwidth to guarantee a continuous flow of data.

2.4.3 Transport Layer Protocols

The success of the Internet is based on several factors; among these, one of the most important is the ability to provide a reliable medium for information exchange and file downloading. In particular, traffic control functionalities for the most common applications (i.e., FTP, HTTP, SMTP, Telnet) are provided by the Transmission Control Protocol (TCP). TCP was initially designed to provide an end-to-end, connection-oriented, and reliable service in the ARPANET [138], and later, in the Internet. TCP addresses two major issues: reliability and congestion control [139]. To achieve the second goal, TCP adapts the sending rate to avoid network overflow or falling into service starvation. TCP congestion control has been studied by the research community for the last 25 years, leading to several TCP variants with and without the explicit intervention of the network layer (a survey can be found in [140]).

The most popular version, TCP New Reno, implements a congestion control algorithm, known as the AIMD (Additive Increase, Multiplicative Decrease) algorithm. In this context, the *sending window* utilized by the sender represents the number of packets that the sender can send towards the destination without having yet received any acknowledgment about their delivery from the receiver. Indeed, for every packet received, the receiver sends back to the sender an acknowledgment (ack) that identifies the next packet expected and implying that all the precedent packets in the sequence where successfully delivered. Acks (and *Time Outs* [139]) are also used to determine packet losses and to communicate the *advertised window* back to the sender. The advertised window provides the sender with a limit to the maximum transmissible sending rate. The final sending window, in fact, is the minimum between the advertised window and a variable named *congestion window*.

The very basic concept of the congestion control scheme can be summarized as follows:

- The number of packets sent out without having yet received back their corresponding acks cannot be higher than the current sending window.
- For each successfully delivered packet, a new one is sent.
- When a whole congestion window of packets has been successfully delivered, the congestion window value is increased by 1.
- When a packet loss is detected the TCP sender assumes congestion on the path to the receiver and decreases its congestion window by half (or to 1 if a Time Out occurred).

This scheme has been developed following the end-to-end paradigm by which the two involved end nodes do not have any explicit information about the links connecting them. In essence, the Internet is seen as a *black box* whose contents remain unknown and all the intelligence is left at the edge. Sender and receiver are unaware of the available bandwidth on the links among them and of the possible presence of other flows along the

same path. The sender has hence to continuously probe the channel to make use of bandwidth that might be available and to back off when congestion is detected.

The User Datagram Protocol [147] offers a lighter service: it just takes messages from the application layer and sends them toward destination with no guarantee that messages are actually going to be delivered and in which order. Indeed, UDP has no retransmission, flow control, and congestion control. At the same time, its simplicity of use ensures a reduced overhead; furthermore, the fact that no connection needs to be established makes it faster than TCP. These reasons make UDP ideal for real time applications or streams that need a constant amount of minimal bandwidth at their disposal to work properly.

It is hence evident that elastic applications are well matched with TCP, whereas real time ones are usually supported by UDP. Indeed, UDP is usually employed by applications characterized by stringent real-time constraints and that can tolerate sporadic packet losses (i.e., audio/video streaming, online games). The lack of congestion control functionalities of UDP had lead the scientific community to wisely consider UDP as unfair toward TCP. Indeed, citing from [198]: *"Although commonly done today, running multimedia applications over UDP is controversial to say the least. [...] the lack of congestion control in UDP can result in high loss rates between a UDP sender and receiver, and the crowding out of TCP sessions - a potentially serious problem."*

Even if this is true when the available bandwidth is very scarce, the broadband connectivity offered today may overturn this situation [157, 166, 203, 204].

Larger and larger bandwidths are offered even to home consumers so that the traffic generated by UDP-based applications can be accommodated. Yet, a problem emerges when real-time applications (UDP-based) coexist with downloading ones (TCP-based) on a wireless channel, causing the former to experience a scattered flow progression.

Major causes for this problem can be found in the TCP's congestion control functionality. In particular, TCP continuously probes for higher transfer rates, also queuing packets on the buffer associated with the bottleneck of the connection. If one considers that the same wireless connection might be shared by several devices and

applications thus increasing the congestion level and queue lengths, it is even more evident how packets can be delayed in queue, jeopardizing requirements of real-time applications.

This negative situation is further worsened by the following three factors due to the wireless nature of the link. First, the wireless medium allows the transmission of only one packet at a time and is not full-duplex as wired links. Packets have hence to wait their turns to be transmitted. Second, as interference, errors, fading, and mobility may cause packet loss, the IEEE 802.11 MAC layer reacts through local retransmissions (4 at most, [22]) which, in turn, cause subsequent packets to wait in queue until the preceding ones or their retransmissions eventually reach the receiver. Last but not least, the back-off mechanism of the IEEE 802.11 introduces an increasing amount of time before attempting again a transmission [102].

2.4.4 Performance Evaluation over Wireless Links

In recent years, many researchers have focused their studies on the problems encountered in a wireless environment [15, 56]. We try to limit the scope of our survey to those works that are related to problems we are trying to address in this Thesis and to the solutions we propose and evaluate in Chapter 5.

Focusing on MAC layer retransmissions, TCP and UDP flows have been tested by *Nam et al* over a IEEE 802.11 wireless link when different signal levels were present. They showed that without retransmissions implemented at the link layer, loss rates become unacceptable for any application [3]. The claim that MAC layer retransmissions improve TCP performance was confirmed also by *Xylomenos* and *Polyzos*, who experimented TCP and UDP on a WLAN and analyzed their behavior with different interfaces and bidirectional TCP traffic [4].

Said that, we may ask ourselves whether the current number of MAC layer retransmissions represents the optimal choice to support both TCP-based traffic and real time applications. Indeed, a high number of repeated retransmissions could still be not enough to prevent TCP from experiencing timeouts and retransmitting the same data as the MAC layer. At the same time, MAC retransmissions can be wasteful and potentially

harmful for time-sensitive applications, such as real time video/audio or online games over UDP [166].

It should be said that a vast collection of research papers focusing on 802.11 could be found by delving into the Internet. They present analysis, problems, and solutions. Nonetheless, the vast majority of them provides results that focus on a throughput/losses point of view [57, 172, 173, 174, 176, 177], while the performance of real time applications depends on the measured per packet delay and jitter [58]. Even if some recent works present delay measurements for real time applications over IEEE 802.11, a deep analysis of this issue with respect to MMOG is still missing, as well as efficient solutions aimed at reducing queuing delay over wireless links [166, 175].

2.4.5 MAC Layer Protocols: the IEEE 802.11 Family

As the availability of digital entertainment devices increases rapidly, the need for interconnecting them is felt as ever more urgent, as well as the necessity to extend the reach of entertainment centers to the wireless domain. In the today's market, IEEE 802.11 based wireless LANs are *de facto* emerging as the candidate to lead the mobile revolution providing wireless connectivity and advanced functionalities in terms of flexibility, security and throughput to support entertainment applications ranging from networked games to in-house digital audio/video distribution and live conferencing etc. [141]. In this context, technical standards are currently being defined that address both the methods of wired/wireless interconnection and the means to guarantee a full interoperability between digital entertainment appliances. Yet not much work has been done in the direction of understanding how can the Internet native language (i.e. the TCP/IP protocol) take over this complex scenario for efficiently delivering digital contents to entertainment devices, and which is the impact of diverse MAC layer settings over the Internet native transport protocols (e.g. TCP, UDP) during the distribution of in-house entertainment.

The MAC layer is positioned at a low level in the OSI model [186]. Its classic implementation for mobile networks is represented by the family of IEEE 802.11 family

of standards, whose specifications also includes the physical layer. All of IEEE 802.11 versions can work both in infrastructure and in ad-hoc mode.

Recalling its main characteristics, the IEEE 802.11 MAC layer protocol attempts to face the packet loss problem by implementing its own retransmission scheme [142]. In particular, lost packets are retransmitted after a certain period of time without having received any corresponding ack. Successive retransmissions for the same packet are repeated up to a maximum number of time, which is by default set to 4 in the standard IEEE 802.11, or until receiving a successful ack. A backoff mechanism determines the retransmission timeouts. This scheme hides wireless error losses from the TCP's congestion control mechanism, thus avoiding deleterious multiple reductions of the data sending window. On the other hand, local retransmissions affect packet delivery delay by increasing its variability and thereby affecting time-constrained applications such as audio or video-stream.

Focusing on the various versions, the IEEE 802.11b represents the first one in the 802.11 family that had an epidemic diffusion. It implements a CSMA/CA mechanism to handle the channel contention and has a nominal bandwidth of 11 Mbps and a factual one of circa 5 Mbps. The utilized channel frequency is the 2.4 GHz, which is prone to a lot of interference from microwave ovens, cell phones, Bluetooth devices, etc.

The IEEE 802.11a and IEEE 802.11g protocols have been developed to increase the bandwidth available over wireless connections and offer a nominal speed of 54 Mbps, which corresponds to about 20 Mbps of factual bandwidth [174, 205]. The former utilizes the 5 GHz reserved frequency and has the possibility to reduce interference through the use of 12 non-overlapping channels, whereas the latter exploits the traditional 2.4 GHz frequency and is hence fully compatible with the IEEE 802.11b. Since its high speed, its ability to coexist with legacy IEEE 802.11b WLANs, and its low cost, the IEEE 802.11g is rapidly becoming very popular in homes, offices, educational institutions, and public hot spots.

Aimed at providing Quality of Service (QoS) capabilities to WLANs, the IEEE 802.11e has been proposed [102]. Its design allows do discriminate among different kind

of flows, assigning priorities through different parameter settings. In particular, flows with different priorities are enqueued into different buffers to which corresponds different contention times. As a result, packets belonging to high priority traffic will have higher chances to be transmitted first with respect to low priority ones. Yet, it is not clear how the AP will be able to classify the incoming traffic; probably, the sender will have to mark each of its packets or flows with a priority level. Unfortunately, this would imply the modification of all senders' operability or applications, thus strongly affecting the actual deployment of this protocol.

The increasing request for higher and higher bandwidth has pushed the IEEE in developing the IEEE 802.11n standard that promises a theoretical speed around 250 Mbps and a factual one around 100 Mbps. Moreover, the IEEE 802.11n will also be able to utilize MIMO (multiple-in, multiple-out) directional antennas thus spatially increasing the available bandwidth. In January 2007, the IEEE 802.11 working group has approved Draft 2.0.

The IEEE 802.11p is also known as WAVE (Wireless Ability in Vehicular Environments) and represents the standard specifically designed to provide wireless connectivity to driving vehicles. Initially designed for Intelligent Transportation System (ITS) applications, it is know object of studies for being utilized for several other purposes that includes both traditional Internet services and novel applications. Even if the IEEE 802.11p protocol is still just a draft, some of its future features are assumed to be known. In particular, it will make use of the reserved 5.9 GHz frequency to limit interferences and will support the American DSRC (Dedicated Short Range Communication) standard for interconnecting vehicles among themselves and with the infrastructure on the curb [187]. Other promised features of IEEE 802.11p are its ability to transmit even at very long distances (up to 1000 m) and very high vehicular speeds (at 200 Km/h it should transmit up to 6 Mps at 300 m), which make it very appealing for any kind of application, from those related to increase traffic safety to entertainment ones.

Many other protocols, even if less popular, than those just discussed compose the family of IEEE 802.11 standards; a comprehensive list can be found in [186]. Since in

this work we are mainly interested in realistic home and vehicular wireless scenarios, we focus our attention on IEEE 802.11g and IEEE 802.11p protocols.

2.4.6 Broadcast in a VANET

Sending online games among (many) players in a VANET amounts to propagate game events over a wireless channel, through multi-hops, where the recipients can be many even within the same transmission range. This clearly corresponds to the pull model and is hence more efficiently resolved through (multi-hop) broadcast, rather than resorting to many resource-consuming unicast transmissions. Therefore, one of the problems we address in this thesis regards fast broadcasting of a message to all cars in a given strip-shaped area-of-interest [191].

Experts report that principal reasons behind a slow broadcast delivery are due to a non-optimal number of hops experienced by a message to cover all the involved cars and, more in general, to an excessive number of vehicles that try to simultaneously forward the message [189, 190, 191, 192]. To tackle this problem a theoretically optimal broadcast algorithm has been recently proposed which propagates messages to cars making use of the notion of Minimum Connected Dominating Set [193]. This leads to great practical difficulties in the implementation of such algorithm as it would require a complete and continuously updated knowledge of the network topology. For instance, in an attempt to implement this algorithm with n cars, its authors have developed a scheme employing as many as O(n log n) control messages [194]. It goes without saying that this is not a scalable solution.

Addressing the fast-delivery broadcast problem from a more practical standpoint, various 802.11-based solutions have been proposed. For example, [191] proposes a backoff mechanism that reduces the frequency of message retransmissions when congestion is causing collisions. In [190], instead, as soon as a car receives a broadcast message from a following vehicle along a strip, it refrains from forwarding it as the reception of this message is a clear confirmation that subsequent cars have already received it. Unfortunately, both these two schemes do not consider a very important

factor in determining the final propagation delay of a message: the number of hops a broadcasted message traverses before covering its whole area-of-interest.

In [192], hops' minimization is achieved by individuating the farthest car within the source's backward transmission range, which has to forward the message. To this aim, jamming signals are emitted by each car with a duration that is directly proportional to the distance between the considered car and the message's source. The car with the longest jamming signal is clearly the farthest car from the source. Even if this guarantees a minimum number of hops to cover the whole area-of-interest, the time wasted to determine the next forwarder through jamming signals could make this scheme not suitable for a tight time delay scenario as the one we are considering.

A final scheme trying to statistically achieve a minimum number of hops when propagating a broadcasted message is discussed in [189]. In particular, different contention windows are here assigned to each car. The contention window represents the maximum number of time slots a car waits before taking upon itself the task of propagating the broadcasted message: each car randomly select a waiting time within its contention window. In [189], the authors propose that nodes set their respective contention windows with an inverse proportion of the distance from the sender. With this scheme, no control traffic is generated that causes useless overhead. Yet, it is assumed that there is a unique and constant transmission range for all cars in every moment; this is obviously not realistic in a VANET since its high and fast mobility.

Moreover, all discussed schemes only discuss the case where few messages are sporadically sent to other cars around; none of them considered the possibility to have many messages continuously generated by many cars in the same VANET.

2.5 System Model

Aimed at providing a holistic solution for MMOGs and being aware of related works, we can now analytically study the four main requirements seen in Section 2.2 and the tradeoff relationship existing among them.

Every class of game is featured by a peculiar (and fixed) *Game Interactivity Threshold* (*GIT*) that represents the maximum delay endurable before visualizing a game event on players' screens if one wishes to preserve interactivity. The typical *GIT* for fast paced games (i.e. vehicle racing, first person shooter) corresponds to 150-200 ms but this value can be increased up to seconds in case of slow paced games (i.e. strategic, role play game) [77, 78, 80, 106, 107].

If we call $t^{g(e)}$ the generation time of event *e* and $t_i^{v(e)}$ the visualization time of the same event at player *i*, then interactivity is preserved at *i* during the delivery of *e* when the following condition is satisfied:

$$t_i^{v(e)} - t^{g(e)} \leq GIT. \tag{2.1}$$

Both consistency and fairness regards having the same game state contemporary viewed in all the nodes of the system. Therefore, the same class of techniques is generally used to achieve each of them (or both). The easiest way to guarantee consistency and fairness is to make the game proceeding through discrete locksteps. Unfortunately, as discussed in Section 2.3.2, this scheme cannot be applied to interactive games.

To ensure fairness (and consistency) in continuously evolving games several studies propose schemes based on the introduction of artificial delays in order to contemporary visualize game events on all the players' screens (i.e., local lag schemes) [54, 78, 94, 104, 105].

With local lag, game advancements are delayed for a sufficient amount of time in order to guarantee that all the clients in the system process and perceive the generated game events at the same time and in the same order. Indeed, since the generation time of each event is unique and considering CC, the set of clients, we can say that we have event-related fairness [109] for event e if condition (2.2) is satisfied, simply stated, if there is a unique $t^{v(e)}$ value for all the players:

$$t_i^{v(e)} = t^{v(e)} \qquad \forall i \in CC. \tag{2.2}$$

Since a single game event experiences different overall delays (*OD*) in its paths from the source to all the diverse destinations, different amounts of artificial delay δ should be added in order to contemporary visualize the same event *e* on all the players' screens and hence to satisfy the following condition:

$$t^{g(e)} + OD_i(e) + \delta_i(e) = t^{v(e)} \quad \forall i \in CC. \tag{2.3}$$

A possible value typically chosen for the unique $t^{v(e)}$ is represented by the highest *OD* in transmitting events amongst nodes. When the highest *OD* is greater than *GIT*, however, fairness is preserved at the cost of jeopardizing interactivity for all the players. Conversely, if we use *GIT* as an upper bound to $t^{v(e)}$, then we can guarantee interactivity but not fairness.

Consequently, in order to maximize the possibility to obtain both interactivity and fairness $t^{v(e)}$ should be set as

$$t^{v(e)} = t^{g(e)} + GIT. \tag{2.4}$$

The $OD_i(e)$ experienced by an event *e* when it finally reaches client *i* is composed by several delay components, respectively: physical latency $l_i(e)$, queuing time $q_i(e)$ on nodes along the path, and processing time $p_i(e)$. Therefore, $OD_i(e)$ can be written as

$$OD_i(e) = l_i(e) + q_i(e) + p_i(e). \tag{2.5}$$

Even when the network latency would allow having values of *OD*, and hence also of $t^{v(e)}$, lower than *GIT*, a large number of players generating a huge amount of traffic may raise the value of the other two components (i.e., $q_i(e)$ and $p_i(e)$), thus leading us again to the crossroad between fairness and interactivity.

To conclude, the efficiency and applicability of popular delayed-based algorithms such as local lag strongly depend on the network conditions and on the interactivity degree required by the game. Yet, guaranteeing both interactivity and full fairness through local lag can sometimes be achieved only at the cost of limiting the scalability of

the game by bounding the number of contemporary participants and the geographical extension of the target player market.

It hence becomes evident how MMOGs require the use of architectural solutions and algorithms able to reduce the delay components in (2.5) in order to find the most efficient tradeoff among interactivity, consistency, fairness, and scalability.

2.6 Architectures

Typically, network architectures supporting MMOGs can be distinguished based on three main categories as depicted by Fig. 2.2: *centralized client-server*, *fully distributed*, and *mirrored game server*.

In the centralized client-server architecture, we have a single authoritative point which is responsible to run the main logic of the game. We report here only a few of its tasks: execute players' commands, enforce consistency, send back to the client the new game state update, etc. Clients have only to receive the new game state update, render it on the screen, and forward player's commands. The single authoritative point is usually represented by a single server; however, a cluster of computers could be utilized as well in order to increase the performance of the system [97].

The centralized client-server architecture represents the simplest solution for authentication procedures, security issues, and consistency maintenance [35, 95, 96, 97]. Moreover, assuming to have N simultaneous players, the generated messages are in the order of $O(N)$. On the other hand, the unique bottleneck limits the efficiency and scalability of this solution.

Fully distributed architectures are well represented by the peer-to-peer paradigm. In this case, all the involved nodes share the same intelligence and are responsible for running the whole logic of the system. In this case, in fact, each client has to autonomously update the game state view based on its player's commands and on game actions received from other players. This obviously requires terminals endowed with higher computational capabilities.

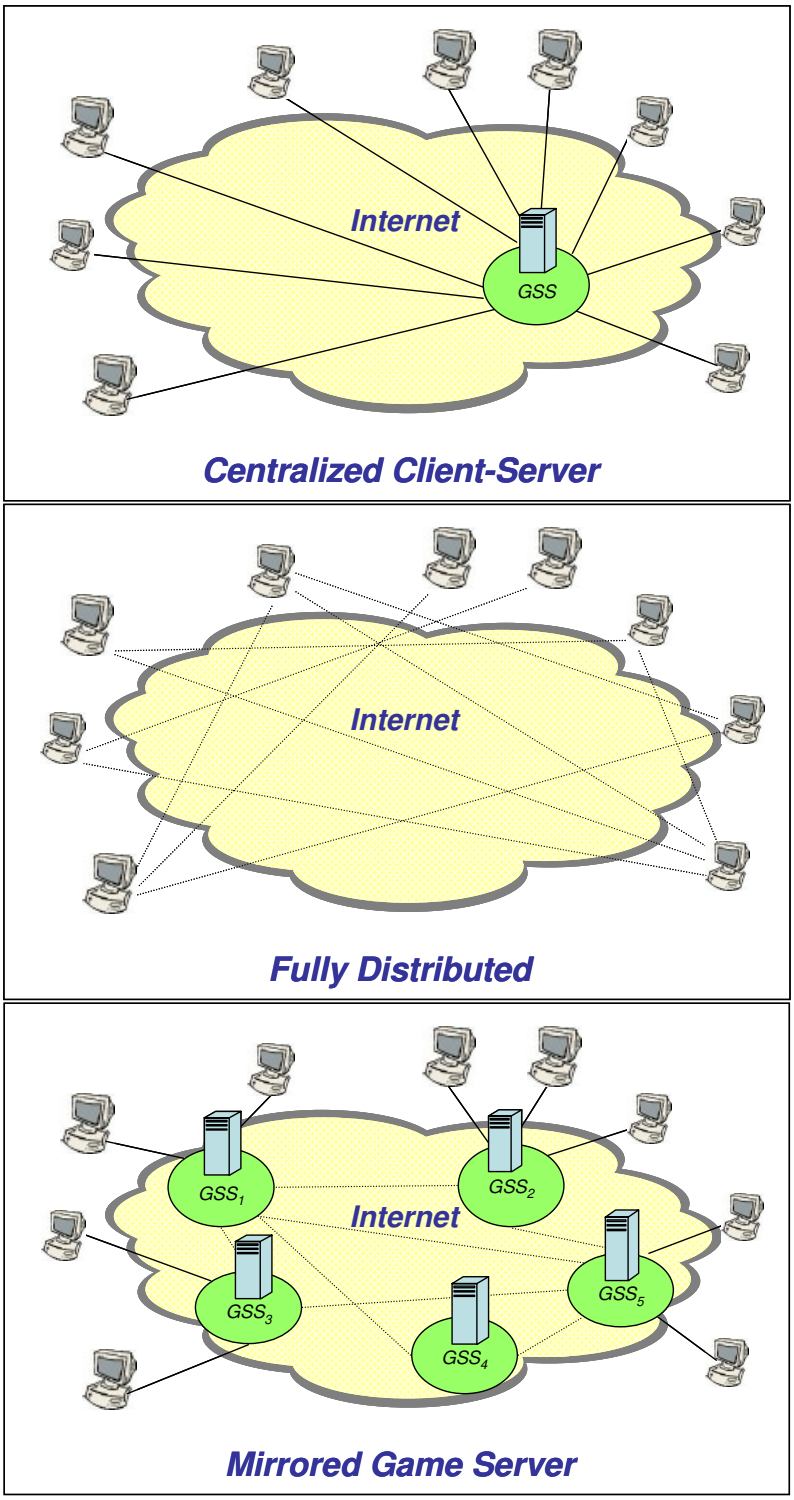


Figure 2.2: Online game architectures.

The main advantage in employing a fully distributed architecture is that of spreading the traffic load among many nodes thus generating a more scalable and failure resilient system [94, 112]. However, identical copies of the current game state need to be stored at each node. This requires some complex coordination scheme among peers; in fact, this scheme has to be distributed over the set of involved nodes and has to be able to guarantee the coherence of all game state views. Moreover, with fully distributed architecture, multicast should be employed to reduce the bandwidth requirements, but multicast technology is neither generally available nor mature enough for the specific application we are considering here. The exchanged messages could hence raise to the order of $O(N^2)$. Finally, authentication, cheating, and general consensus among all the peers are harder to be addressed than when a centralized architecture is employed.

Mirrored game server architectures represent a hybrid solution which efficiently embraces all the positive aspects of both centralized client-server and fully distributed architectures [59]. Based on this approach, GSSs are interconnected in a peer-to-peer fashion over the Internet and contain replicas of the same game state view. Players communicate with their closest GSS through the client-server paradigm. Each GSS gathers all the game events of its engaged players, updates the game state and regularly forwards it to all its players and GSS peers.

The presence of multiple high performance GSSs helps in distributing the traffic over the system and reduces the processing burden at each node [112]. Moreover, having each player connected to a close GSS reduces the impact of the player-dependent access technology (e.g., dial-up, cable, DSL) on the total delay experienced [110]. In this case, in fact, the communication among players results mainly deployed over links physically connecting GSSs, which can exploit the fastest available technology (e.g., optical fibers) to reduce latency. As a result, this architecture helps one in finding better solutions for the tradeoff among interactivity, consistency, fairness and scalability.

Other advantages in employing mirrored game server architecture are the absence of a single point of failure, the networking complexity maintained at server side, and the possibility to easily implement authentication procedures. Even if synchronization is still

required to ensure the global consistency of the game state held by the various servers, this requirement is made easier with respect to fully distributed architectures thanks to the lower number of involved nodes. Assuming to have N players and M GSSs, for example, the generated game messages amount to $O(N+M)$, which is again $O(N)$ unless considering the unlikely case of having more servers than players.

CHAPTER 3

Fast Synchronization Framework

3.1 Proposed Architecture

Since the analysis of advantages and disadvantages about the various possibilities illustrated in Section 2.6, mirrored game server emerges as the most appropriate architecture in order to efficiently manage large-scale distributed games. Indeed, this architecture embodies the advantages of both client-server and fully distributed paradigms and, in particular, it preserves the two most important features required by a MMOG architecture: scalability and controllability. The former is required to allow a multitude of players to engage in the same virtual arena. The latter regards the possibility to control the access to the game, avoid cheating, and have a centralized core where new techniques could be easily deployed to improve performance. Indeed, based on this architecture, we have devised an efficient synchronization scheme among GSSs able to enforce a high interactivity degree while guaranteeing a uniform view of the current game state.

Moreover, in Chapter 4 we will demonstrate how to exploit this architecture and our synchronization scheme to achieve also fairness, while in Chapter 5 and Chapter 6 we will focus on preserving interactivity and fairness also in the last hop of the connection. We will show how to enhance the mirrored game server architecture to make it able to sustain MMOG applications regardless of the connectivity type exploited on the client-server link.

3.2 Obsolescence and Correlation: Maintaining Responsiveness and Consistency

Absolute Consistency can be attained through the employment of a totally ordered event delivery scheme [60, 61]. On the other hand, this would imply an increment of the complexity and of the total delay experienced by the system. Waiting for the next in order action to be processed, while having other events ready in queue, may sensibly slow down the evolution of the game thus jeopardizing responsiveness.

Exploiting the semantics of the game can be put to good use to relax the total order delivery requirement and augment responsiveness [63, 64]. Some events, in fact, can lose their significance as time passes: new actions could make irrelevant previous ones. For example, player's movements are generally represented by final absolute position and, in case of rapid succession of movements of a single agent, the event representing its last destination makes obsolete the older ones.

Obsolescence can thus be defined as the relation between two received events e_1 and e_2, generated at different times $t(e_1) < t(e_2)$, by which the existence of event e_2 diminishes the importance of processing also event e_1, without affecting consistency (see Fig. 3.1-a). Dropping obsolete events before processing them clearly reduces computation at GSSs and speeds up the execution of fresher events.

To define as obsolete a game event, we have to be sure that consistency would not be weakened. To this aim, we have also to take into account the notion of ***correlation***. Two events, say e_1 and e_c, are correlated if the final game state depends on their execution order. Correlation has to be taken into account to determine the obsolescence of an event. As depicted in Fig. 3.1-b, it might be the case when e_2 would make obsolete a previous event e_1 but a further event e_c (correlated to e_1), temporary interleaved between e_1 and e_2, rescinds this relationship of obsolescence. However, they are the only events that really need to be reliably delivered to all destined GSSs and in the same order as they were generated.

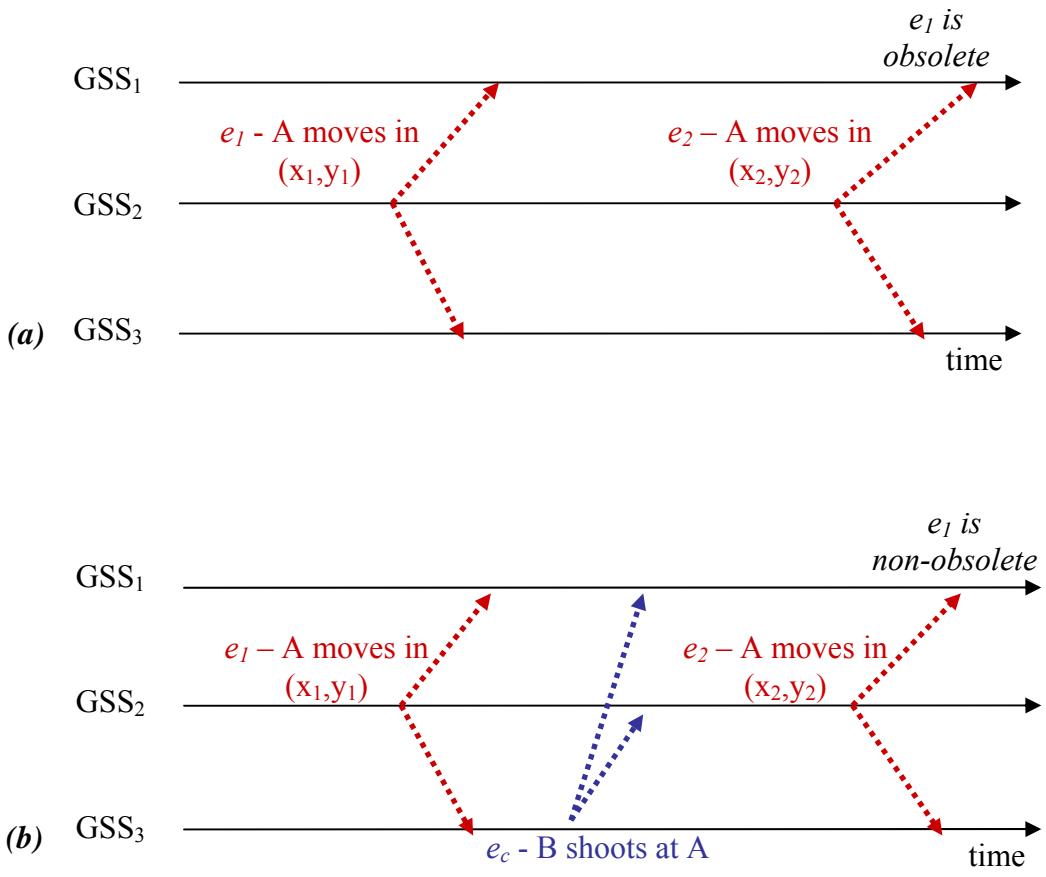


Figure 3.1: Examples of obsolescence and correlation

(a) Event e_2 makes obsolete e_1; (b) Event e_c is correlated to e_1 and rescinds the obsolescence of e_1.

Total order delivery requirement can thus be relaxed in case of non-correlated game events. Their semantic independence, in fact, allows different GSSs to process them in diverse orders without affecting consistency. This means that non-correlated game events can be processed as soon as they are received without wasting any time in waiting preceding ones, thus augmenting responsiveness; missing game events can be interpolated and rendered on players' screens. Interested readers may find a deeper analysis on obsolescence and correlation in [63, 64].

On the other hand, our experiences with online games over a best effort network lead us to claim that there exist cases where even dropping all the obsolete events in a game is

not enough to ensure an adequate interactivity degree. This is particularly true for a class of games that requires frenetic, and often redundant, players' actions. This class of games is widely recognized in the gaming community as fast-paced (or fast and furious) games: shoot/beat'em up games represent typical exemplars. For this class of games, even discarding some non-obsolete events may be preferred to compromise responsiveness and consequently users' perceived playability. Discarding some non obsolete event may generate some sporadic inconsistencies in the game state; nonetheless, we deem that partial consistency for a small amount of time becomes acceptable for the specific class of fast and furious online games. In this scenario, in fact, the necessity of a very high interactivity degree emerges as overwhelming even on the full-consistency requirement.

3.3 Interactivity Restoring Mechanism

Exploiting the notions of obsolescence and correlation, Ferretti and Roccetti propose a mechanism to restore interactivity in mirrored game server architecture [63]. The player's actions are collected by the closer GSS, transformed into events and finally forwarded to other GSSs in order to maintain a global identical view of the game state. Events are marked at their creation with a generation timestamp and then sent to destination: they are hence orderable. Obviously, a global concept of time has to be maintained by all the GSSs. Different solutions proposed in literature can synchronize the clocks at each GSS [65, 66, 67]. Alternatively, this can also be obtained by exploiting some new technological synchronization device such as GPS.

Since UDP is used as the transmission protocol, missing packets are handled at the application level. Game state updates transmitted by GSSs, in fact, are identified by sequence number. By the means of NACK (Negative ACKnowledgment) packets it is possible to determine which ones are missing and decide whether to retransmit them or not.

Each receiving GSS considers the arrival time of the event and measures the difference elapsed since its generation; the resulting value is named *Game Time Difference* (*GTD*). The *GTD* of the event is then compared with the predefined constant

GIT and normal delivery operations are performed until the former value is lower than the latter. When the *GTD* value exceeds the *GIT*, the GSS turns on a stabilization mechanism which exploits the obsolescence notion to drop useless events so as to bring the *GTD* back within the *GIT*. Moreover, messages are sent to the connected GSSs in order to make them aware of the witnessed lack of responsiveness. These GSSs can thus avoid to forward obsolete and missing events to the non interactive GSS thus helping in restoring faster

3.4 RED/RIO Techniques

Random Early Detection (RED) algorithm is an active congestion avoidance mechanism enforced at routers [68]. Traditional queue management employs simple "tail drop" schemes that drop packets only when the queue overflows. Conversely, RED algorithm randomly performs early packet discards to notify sources about the incipient congestion. In this way, a single loss experienced by a sender smoothly decreases the entire congestion level of the network and keeps the average queue size at a low level. The rationale lies in the gained capability of better accommodating occasional bursts of packets and avoiding situations in which several connections simultaneously decrease their sending rate. Summarizing, RED avoids severe congestion and maintains a stable traffic level in place of dealing with congestion after already occurred.

Every time the router receives a packet, the RED algorithm calculates the new average queue size and the probability to discard the packet. The computing method utilizes a uniform random variable, which has been proven to be more adequate to this aim than a geometric random variable. In fact, a uniformly distributed discarding function avoids global synchronization thus attaining an unwavering course of transmissions. The dropping probability is bounded by two thresholds of the queue size: min and max. Within this interval, the probability to drop a packet increases from 0 to a maximum discarding probability (Pmax). Under min, no packet is dropped and beyond max all packets are discarded.

RIO (RED with In and Out) scheme is an enhanced version of RED mechanism able to discriminate between two different classes of traffic, non-prioritized (Out) and prioritized (In), and calculates two distinct dropping probabilities [70]. The two dropping functions work independently and are featured with specific boundaries and slopes. They hence permit to discard packets utilizing different probabilities and congestion levels depending on the packets' traffic class.

3.5 Enhancing Interactivity with RED/RIO Techniques

Taking inspiration from the RED approach in case of incipient congestion in best effort networks, we have recently enhanced the aforementioned Interactivity Restoring mechanism with the Interactivity-Loss Avoidance (ILA) approach [62]. The main innovation is the capability to preempt responsiveness disruption instead of restoring it after having already lost playability. To this aim, the system discards some packets when the responsiveness among GSSs descends significantly. In practice, ILA replaces the basic binary dropping mechanism for obsolete events (OFF when interactivity is present and ON when interactivity is lost) with a continuously-working proactive mechanism that drops obsolete events with a probability that depends on the level of responsiveness of the system.

Even if, similarly to RED, ILA utilizes a uniformly distributed dropping function, the parameter taken under control is the average *GTD* instead of the average queue size. Upon each packet arrival, in fact, each GSS determines the *GTD* of the arrived event, namely *sample_GTD*, and feeds a low pass filter to compute the updated average *GTD*, namely *avg_GTD*. When *avg_GTD* exceeds a certain threshold, the GSS drops obsolete events with a certain probability *p*, without processing them. If *avg_GTD* exceeds a subsequent limit, *p* is set equal to 1, and all obsolete events waiting for being processed are discarded.

Obviously, if the low interactivity degree perceived by some GSSs is not affecting other GSSs, the latter can process and forward to their clients even those packets discarded by the formers. Indeed, since obsolete events are actions that can be considered

non-critical for the game evolution, there is no unfairness generated by processing them only in some GSSs. On the other hand, not processing those events in highly interactive GSSs would represent an unjustified limitation, even if light, to the number of events visualized at some player's side. For instance, an intermediate movement of a game character, when the final position is further ahead and no shooting nor other hazards could harm it, could receive different treatments: it can be dropped by some GSSs to preserve interactivity, or can as well be processed by other GSSs for the sake of a complete and fluent game visualization on the screens of their clients without relying on imprecise and computationally expensive interpolations.

To ensure an adequate playability degree even to the class of fast and furious games we have then further enhanced our ILA scheme. For this class of games, in fact, the core attractiveness for players emerges from a feverish, sometimes even chaotic, action sequence of user's actions. We have hence enhanced our ILA scheme with features derived from the integration of a RIO-like algorithm in place of the RED-like one [69]. The additional dropping probability provides the possibility to discard even non-obsolete game events when dropping all the obsolete ones is not yet sufficient to maintain an adequate level of responsiveness. The two discarding functions are featured with specific parameters; they work independently one from the other and take action in sequence with the increasing of the game event *GTDs* at the GSSs.

Dropping non-obsolete events can be done without consequences only for a category of games where little inconsistencies are not highly deleterious for the aim of the game and for player's fun (e.g., fast-paced games). Even in this case, if the number of dropped non-obsolete events becomes significant, a consistency restoring mechanism may be required to re-establish a coherent game state view among all GSSs [71]. For the sake of clarity, from here on we are going to call ILA-RED the ILA version that discards only obsolete events, while ILA-RIO represents the version with two discarding functions. If referring to both the algorithm with no distinction we just use the word ILA.

In Fig. 3.2 we depict the two discarding functions of ILA-RIO. Three parameters (and three phases) characterize each of the twin algorithms: min_o, max_o and $Pmax_o$, for

obsolete events, and min_v, max_v and $Pmax_v$ for valid (i.e., non obsolete) ones. In the graph, the y-axis represents the dropping probability corresponding to the *avg_GTD* indicated by the x-axis. Focusing on obsolete events, for values of *avg_GTD* in [0, min_o) the mechanism performs normal operations, with no packet drops, while in [min_o, max_o) obsolete packets are discarded with a computed probability, and finally in [max_o, ∞) all obsolete packets are thrown away. The intervals [0, min_v), [min_v, max_v), and [max_v, ∞) define the corresponding phases for valid events. The dropping probabilities are computed as a function of *avg_GTD* and of $Pmax_o$ or $Pmax_v$, respectively. Persistent situations of low interactivity degree result in high *avg_GTD* and hence in high discarding probabilities. High dropping probability values (for $Pmax_o$ or $Pmax_v$) will make the GSS discarding events without processing nor forwarding them, thus helping in restoring an adequate level of time interaction between servers.

Since valid events are strictly linked to consistency, the possibility to discard them should be taken into account only as last resort, in case of heavy disruption of responsiveness. For this reason, ILA-RIO starts dropping obsolete packets much earlier than valid ones. In addition, the algorithm throws away all the obsolete packets before considering any dropping probability on valid events; simply stated: max_o is smaller than min_v. Finally, diverse aggressiveness in dropping packets, depending on their class, can be decided by adjusting the values of $Pmax_o$ and $Pmax_v$.

In essence, the algorithm repeats a block of operations (listed in Fig. 3.3) each time a new event arrives at the considered GSS. In particular, the *GTD* of the game event is calculated (*sample_GTD*, line 1) as the difference between the generation time at the sending GSS and its delivery time to the considered receiving GSS. The scheme feeds a low pass filter with the just calculated *sample_GTD* in order to update the average of the *GTDs* (*avg_GTD*, line 2):

$$avg\_GTD = avg\_GTD + w*(sample\_GTD - avg\_GTD). \qquad (3.1)$$

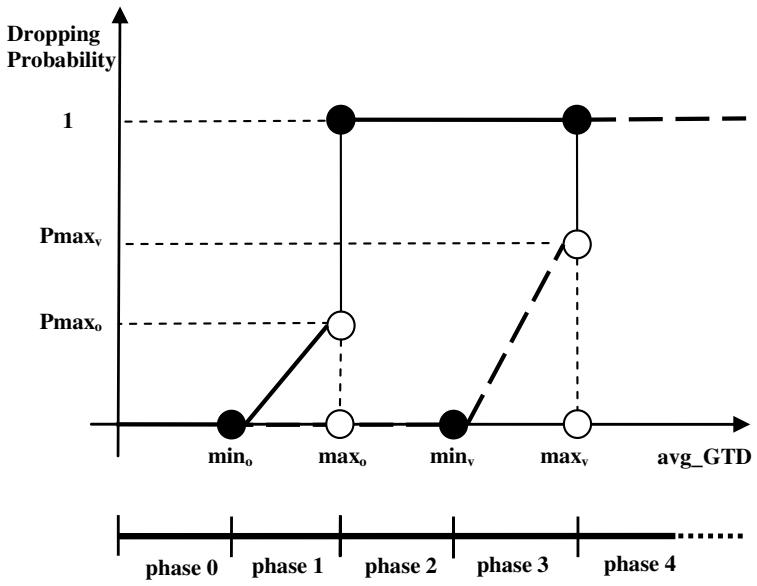


Figure 3.2:Discarding probability functions for ILA-RIO.

```
0]  for each event_packet arrival {
1]    determine the sample_GTD
2]    calculate the new average delay avg_GTD
3]    if (min_o ≤ avg_GTD < max_o) then
4]      calculate the probability P_o of dropping an obsolete event
5]      determine if ONE obsolete event has to be discarded
6]    else if (max_o ≤ avg_GTD) then
7]      drop ALL obsolete events
8]      if (min_v ≤ avg_GTD < max_v) then
9]        calculate the probability P_v of dropping a valid event
10]       determine if ONE valid event has to be discarded
11]     else if (max_v ≤ avg_GTD) then
12]       drop ALL valid events
13]     endif
14]   endif
15] endfor
```

Figure 3.3: ILA-RIO algorithm.

In (3.1), *w* is a sensitivity coefficient, with values comprised in (0, 1], that determines how closely the trajectory of the average follows the movements of the samples. Higher values of *w* correspond to heavier relative weights of the last sample in the current average.

While *avg_GTD* lies below min_o, the process stays in phase 0 and no particular operation is performed. Conversely, when *avg_GTD* is comprised between min_o and max_o, then the scheme is in phase 1 and lines 4-5 are executed. Basically, a dropping probability P_o is computed as a fraction of $Pmax_o$ in order to establish whether an obsolete event must be discarded; this fraction linearly corresponds to the position of *avg_GTD* in the interval $[min_o, max_o)$.

$$P_o = \frac{Pmax_o \times (avg\_GTD - min_o)}{(max_o - min_o)}. \tag{3.2}$$

Analogously, when *avg_GTD* is comprised between min_v and max_v, then the scheme lies in phase 3 and lines 9-10 are executed. The probability P_v to discard a valid event is then calculated as a fraction of $Pmax_v$ depending on the position of *avg_GTD* in the interval $[min_v, max_v)$.

$$P_v = \frac{Pmax_v \times (avg\_GTD - min_v)}{(max_v - min_v)}. \tag{3.3}$$

If the algorithm stays in phase 1 or 3 the corresponding dropping probability increases until an event is discarded. In particular, two couple of constants, L1 and L2 for P_o, H1 and H2 for P_v, are defined as shown in (3.4), (3.5), (3.6), and (3.7), respectively.

$$L1 = \frac{Pmax_o}{(max_o - min_o)}, \tag{3.4}$$

$$L2 = \frac{Pmax_o \times min_o}{(max_o - min_o)}, \tag{3.5}$$

$$H1 = \frac{Pmax_v}{(max_v - min_v)}, \tag{3.6}$$

$$H2 = \frac{Pmax_v \times min_v}{(max_v - min_v)}. \tag{3.7}$$

Utilizing these constants to rewrite (3.2) and (3.3), we obtain (3.8) and (3.9) which are employed respectively in line (4) and (9) of the ILA-RIO algorithm.

$$P_o = L1 \times avg\_GTD - L2 \tag{3.8}$$

$$P_v = H1 \times avg\_GTD - H2 \tag{3.9}$$

So far, the two probability functions result in geometric random variable distributions of the drops, while it would be desirable to discard events at fairly regular intervals. This can be obtained by applying an incrementing counter to augment, at each iteration, the weight of the considered discarding probability; as result, we generate a uniform distribution of the dropping dispersion, which makes the ILA scheme more resilient to temporary bursty periods [68]. To this aim, when the algorithm is in phase 1 or 3, P_o or P_v is compared with a number which is randomly generated within the interval [0, 1). If we call R_o and R_v, the two random numbers, we can see from (3.10) and (3.11) how they are used to take into account the number of iterations elapsed since the last drop to increment the discarding probability. Basically, in phase 1 an obsolete event is dropped (line 5) if

$$counter_o \geq (R_o / P_o). \tag{3.10}$$

Analogously, in phase 3 a valid event is dropped (line 10) if

$$counter_v \geq (R_v / P_v) . \quad (3.11)$$

The variables $counter_o$ and R_o are reinitialized to respectively 0 and a new random value each time the algorithm enters phase 1, or some obsolete packet is discarded. Analogously, variables $counter_v$, and R_v are reinitialized each time the algorithm drops a valid packet. Conversely, if the scheme is respectively in phase 1 or 3, the two dropping probabilities, P_o and P_v, are recomputed every new event arrival. Moreover, $counter_o$ and $counter_v$ are incremented by 1 every time ILA-RIO stays in phase 1 or phase 3, respectively, without dropping any game event.

Considering the remaining of the algorithm, if *avg_GTD* grows beyond max_o, the scheme enters in phase 2 and all obsolete packets have to be discarded in the attempt of re-establishing interactivity (line 7). Moreover if the value of *avg_GTD* surpasses even max_v, all the events, with no distinction between obsolete or valid, are dropped (line 12). At this point, in fact, the extremely jeopardized responsiveness conditions suggest resetting and reinitializing the game session.

Finally, it is easy to notice that ILA-RED can be considered a particular instance of ILA-RIO. In fact, if we set to 0 the probability of dropping a non obsolete event, or to ∞ the *GTD* value at which it takes action, then the two schemes behave identically. Simply stated, a chart representing ILA-RED behavior, equivalent to that one presented in Fig. 3.2, would contain just phase 0, phase 1 and phase 2, respectively contained in the intervals $[0, min_o)$, $[min_o, max_o)$ and $[max_o, \infty)$. Analogously, the algorithm for ILA-RED can be obtained by simply eliminating lines 8-13 from Fig. 3.3.

3.6 Optimistic Obsolescence-based Synchronization Scheme

Whereas ILA schemes are based on a conservative approach, the *Optimistic Obsolescence-based Synchronization* (OOS) scheme differs from the previous ones as it follows the optimistic paradigm [181]. Still, OOS processes events based on a correlation

order thus guaranteeing the final consistency of the game state evolution at all GSSs [184]. However, this consistency is not continuously preserved; rather, it is restored through employing the well known Time Warp algorithm [52, 53].

Simply stated, as soon as an event e_i, coming from a GSS p, is received at any GSS q, a check is executed to detect whether e_i is obsolete. In the positive case, it is simply discarded. Otherwise, another check is executed to verify whether there exist other already processed events e_j (correlated to e_i) with a generation time larger than $T_g^p(e_i)$. For each event e_j meeting this condition, the system invokes a rollback procedure which is based on the standard incremental state saving technique [181]. It may be the case that some of these events e_j may become obsolete due to the reprocessing activity triggered by the rollback procedure. Needless to say, these events are dropped as soon as they are recognized as obsolete.

This scheme guarantees that game state consistency is preserved.

```
0 procedure OOS-receive-event-procedure() {
1    e_i := received event;
2    if (e_i is obsolete)
3       drop(e_i);
4    else {
5       Event_List := {e_j | e_j already processed ∧
                              e_j correlated to e_i ∧ T_g(e_j) > T_g^p(e_i)};
6       if (Event_List ≠ NULL) {
7          Rollback(Event_List);
8          Process(e_i);
9          for (each e_j ∈ Event_List)
10            if (e_j is obsolete)
11               drop(e_j);
12            else Process(e_j);
13       }
14       else Process(e_i);
15    }
16 }
```

Figure 3.4: OOS algorithm.

The OOS algorithm is reported in Fig. 3.4, which lists all the actions accomplished by a given GSS when a new game event is received. First, the GSS verifies if e_i may be already identified as obsolete (line 2). In this case, e_i is dropped (line 3). Otherwise, a check is carried out to control whether any game events e_j, correlated to e_i and generated after e_i, have already been processed (lines 5-6). If this check succeeds, then a rollback procedure is performed where all these events e_j are rolled back (line 7). At this point, e_i is processed (line 8), followed by the execution of all those rolled back events which are not obsolete (lines 9-12). Obsolete events are discarded during the rollback (lines 10-11). If the check fails, thus implying that no rollback procedure is needed, then e_i is directly processed (line 14).

It is worth mentioning that our OOS scheme is devised to reduce the amount of rollbacks executed during the game processing activity with respect to. Time Warp. Indeed, a rollback is performed only when receiving a late event that cannot be considered as obsolete. Instead, if obsolete, the event is dropped without invoking the rollback procedure. Moreover, a rollback is needed only if correlated events have been processed out of order and, even in this case, only non-obsolete events are reprocessed. In essence, our scheme has the positive effect of reducing the computational burden typically required to maintain the game state consistency.

It is also worth noticing that when an optimistic scheme is employed to synchronize GSSs, processed game events can be transmitted to players only when those events are no longer subject to possible rollback. In other words, the rollback strategy should not affect the game evolution as seen by players; otherwise, users could have the undesired perception of characters that, for example, jump from incorrect positions to correct ones, or come back to life when they have already been shown as killed.

In point of this fact, other proposals that refer to optimistic synchronization schemes for networked multiplayer games [59, 182] do not seem to take into consideration this problem. In essence, events are forwarded to players without any control on the stability of the rendered game state. Therefore, when a rollback occurs, the corresponding reprocessing is perceived by connected players as well.

We have a different position here. We claim that stalling renderings at player's side should be avoided even when an OOS strategy is employed. This may be achieved by notifying players only with a stable game state. While this approach slightly increases the amount of time spent by game events at the server side, the final degree of responsiveness can still be improved thanks to an obsolescence-based discarding mechanism as in our case. This claim is confirmed by results provided.

We conclude this section by mentioning that our OOS strategy has been implemented by exploiting a receiver-initiated communication protocol that utilizes NACKs (Negative ACKnowledgments) to provide the delivery guarantee only for non-obsolete events [180]. It is well known that TCP-based synchronization protocols have to be avoided at all costs while developing distributed games [183]. The motivation is that much of TCP's behavior, such as congestion control and retransmissions, is detrimental to meeting the MMOGs real-time constraint. Our approach, instead, exploits UDP to transmit events among GSSs. Moreover, since we are concerned with the reliable delivery of non-obsolete events only, as soon as a receiving GSS detects that a non-obsolete event e_i is missing (through sequence number), a NACK is sent back to the sending GSS. Upon receiving a NACK, the sending GSS either retransmits e_i or transmits the most recently generated event e_j that has made e_i obsolete.

3.7 Simulation Results

We present here the most relevant simulative results among those we obtained.

In particular we compare the different schemes by showing the outcomes of simulations having the AIDT parameter set to 30ms, 45ms, and 60ms, respectively. Having a low AIDT corresponds to a higher frequency in game event transmissions at each server and therefore in a higher congestion and computational workload.

3.7.1 Simulation Assessment

To evaluate our event processing strategy, we have created a mathematical model simulating a general Mirrored Game Server architecture comprising various GSSs

connected via diverse links over the Internet. As said, we assume that the events generated in the system can be totally ordered by exploiting a global notion of time.

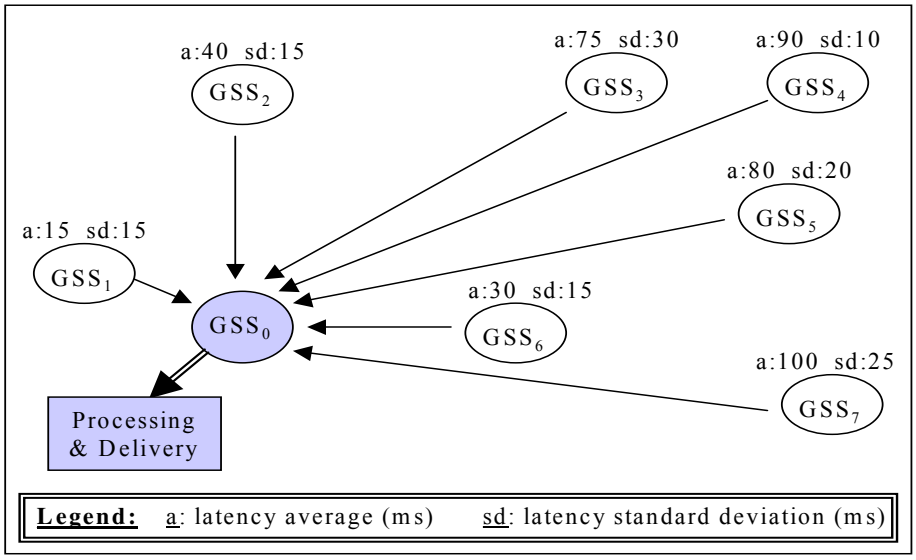


Figure 3.5: The adopted configuration.

Table 3.1: Sending GSSs involved in the simulations

Number of Sending GSSs	**Corresponding GSSs employed**
4	GSS_1, GSS_2, GSS_3, GSS_4
5	GSS_1, GSS_2, GSS_3, GSS_4, GSS_5
6	GSS_1, GSS_2, GSS_3 GSS_4, GSS_5, GSS_6
7	GSS_1, GSS_2, GSS_3 GSS_4, GSS_5, GSS_6, GSS_7

Different numbers of GSSs have been simulated in our scenario. In order to permit a deeper analysis of the dynamics related to game event traffic and processing, simulations have focused on the packet receiving aspect of a single GSS, while the other GSSs are involved as game event transmitters. Fig. 3.5 depicts the adopted configuration of the network and shows the values assigned to the simulation parameters. GSS_0 is the

receiving GSS and the others are the sending GSSs. We carried out several simulation experiments (30) with a number of sending servers varying in the range from four to seven. The involved GSSs, for each different configuration, are listed in Table 3.1.

We have been inspired by the scientific literature on MMOGs to set the values of the network latencies among the GSSs [72]. Specifically, GTDs follow a lognormal distribution having the average and standard deviation values as shown in Fig. 3.5. If each client connects to the "nearest" GSS to bound the impact of the client-server latency on the total delay experienced by the game events, we can assume to have the client-server latency portion already comprised in the network latency values mentioned above. Finally, the event generation rate at each GSS and the average event size (200 Bytes) have been chosen as inspired by the games literature and varies from a normal traffic situation to an intensely loaded one [73].

We define as Average Inter-Departure Time (AIDT) the time that elapses, in average, between two subsequent game packet departures at a single server. The AIDT values have been used to generate the lognormal distribution of the event departures from each involved server. Various simulations have been run having the AIDT parameter set to 30ms, 45ms or 60ms. The standard deviation was always equal to 10ms. The chosen AIDT values produce an amount of events as those typically generated by from 5 to tens of players for each given GSS. Based on these values, we have generated three diverse trace files containing 1000 events for each GSS. Each trace file also included the information needed to identify (correlated and) obsolete events. Where not differently stated, we have set to 90% the probability that an event makes obsolete preceding ones. This represents a realistic scenario for a vast plethora of possible games where critical (correlated) game events that cannot become obsolete have to be considered only sporadically, such as during collisions or shots, and may represent even less than the 10% of the whole set of game events.

As a confirmation of this claim, an extensive study of players' behavior on Quake 3 is presented in [76]. In that paper, a measure of the average number of kill actions per minute as a function of the median ping time between client and server is reported. Using

those measures we can provide a numerical explanation that demonstrates how 10% of correlated events and 90% of obsolescence probability may represent a realistic scenario for interactive MMOGs [81].

Specifically, in [76] it is shown that when the 80% of players is located within 180ms of range from the server, the average of kill actions per minute varies in the interval from 1.60 to 3.25. Considering the median of this interval (2.425) we obtain 0.04 kill events per second. With an AIDT of 60ms at client side [73, 77], 16.67 game events are generated every second. Therefore, the resulting percentage of kill events over the whole set of game actions amounts to just 0.24%. Pessimistically assuming that a kill event can be issued only after an average number of 40 correlated actions (e.g., various shots, movements of the character into the location where it will be shot or out of the position where it would have been shot) we get 9.6% of critical events. Therefore, 10% of correlated events and 90% of obsolescence probability represent a realistic configuration for online games simulations.

Since real commercial games employ UDP as the transport protocol, we have adopted it too for our simulations [75]. Moreover, to circumvent the problems deriving from UDP's unreliability, we have implemented an application level retransmission scheme based on NACKs (Negative ACKnowledgments).

Our set of simulations can be subdivided into three main parts. First, we consider schemes that continuously fully preserves the game's consistency. In this context, we demonstrate the efficacy of obsolescence-based schemes by comparing the outcomes of our proposed ILA-RED scheme with the ON-OFF mechanism (Interactivity Restoring as reviewed in Section 3.3) and the traditional OFF approach (having no discrimination of obsolete packets and no event discarding nor other algorithms to restore interactivity). Then, assuming a scenario involving a frenetically paced game where we can relax the consistency requirement, we evaluated ILA-RIO. Finally, aimed at assessing the potentiality of optimistic schemes coupled with obsolescence-based discarding mechanism, we compare OOS with OFF, ON-OFF, ILA-RED, and the traditional Time Warp (TW) scheme.

As to the algorithm, we have chosen to set w=1/8 in (3.1) in the attempt to make the algorithm able to filter out sporadic high *GTDs*, while being able to promptly react to a persistent decline of responsiveness. In the ILA-RED case, we have set parameters as follows: min_o = 50ms, max_o = 150ms (equivalent to the *GIT* for the ON-OFF scheme) and $Pmax_o$ = 0.2. Parameters in ILA-RIO were chosen as min_o = 50ms, max_o = 100ms, $Pmax_o$ = 0.2, min_v = 150 ms (equivalent to the *GIT* for the ON-OFF scheme), max_v = 225 ms and $Pmax_v$ = 0.3.

Choosing appropriate values for the above cited parameters is an important part of implementing ILA algorithm. The various phases, in fact, should take place in precise situations experienced by the game platform. In particular, phase 1 have to be activated when the delay between the generation of a player's action and its execution on the screens provides the first perceivable symptoms of responsiveness degradation. Since phase 2 corresponds to dropping all the obsolete events, it should take action when the lag becomes annoying and low-performance determining for players. In case of fast paced games, the parameter of these two phases should be chosen in a further conservative way, in order to anticipate the triggering times for the discarding functions. Phase 3 represents the last resort to restore responsiveness in the system, whilst still having latencies that can be temporary tolerated by players. Finally, since entering in phase 4 amounts to reinitialize the game session, its entering threshold have to be chosen as the higher latency value at which users still can compensate the poor playing conditions by anticipating their moves.

The various thresholds should be hence set differently depending on the played game or at list on the belonging game class (e.g. adventures, shoot/beat'em up, car racing, etc.). However, as rationale for our chosen values, scientific literature declares that a delay of 50 ms is not perceived at all by players while at 150 ms (i.e., our *GIT* parameters) player's performance results disturbed by the lag. Finally, 225 ms of delay could represent an upper bound for playable interaction [76, 77, 78, 79]. These limits hold for games like vehicle racing, first person shooters and fast shoot/beat'em-up, but can be augmented in case of strategic games (e.g. Starcraft, Age of Empire, etc.) [80].

The interactivity benefits attained by obsolete event discarding schemes can be analyzed in terms of delays experienced by game events before being processed. Therefore, we have compared the responsiveness of considered schemes, exploiting the following metrics: i) the number of events having a *GTD* larger than the *GIT*; ii) the cumulative function of the *GTD* values; iii) the average of the *GTD* values, their standard deviation, the minimum and maximum values.

The fluency of the game evolution on players' screens passes through having just a very limited number of discarded game events. We have hence compared the obsolete event discarding schemes by measuring: i) the number of obsolete events dropped; ii) the number of times all the pending obsolete events have to be discarded; iii) the average number of subsequent game events with *GTD* over *GIT*; iv) the total number of bursts of game events with *GTD* over *GIT*.

Finally, some results about ILA-RIO and OOS have reported by comparing them against ON-OFF and OFF mechanisms, and against regular Time Warp, respectively.

3.7.2 Obsolescence-Based Scheme vs Traditional One

As predicted, the two obsolescence-based discarding mechanisms outperform the traditional one, especially with high traffic load at the servers. In fact, Fig. 3.6 compares, for ILA-RED, ON-OFF and OFF schemes, the percentage of events arrived at GSS_0 with a *GTD* value larger than the *GIT*. This also represents the percentage of events that cannot be considered interactive.

The cumulative function of the *GTDs* (Fig. 3.7) represents another tool proficient in evaluating the efficacy of ILA-RED and ON-OFF schemes. Indeed, the more the line is concentrated in the left side of the chart, the higher is the percentage of events having a *GTD* lower than a certain threshold. In particular, Fig. 3.7 depicts the cumulative function of the *GTDs* in a scenario considering seven sending GSSs, each one sending events to the receiving GSS_0 with an AIDT of 30 ms. In this configuration, ILA-RED has 93.86% of events with a *GTD* less or equal than the *GIT* of 150 ms, ON-OFF hits the 89.40%, and OFF reaches only the 49.94%.

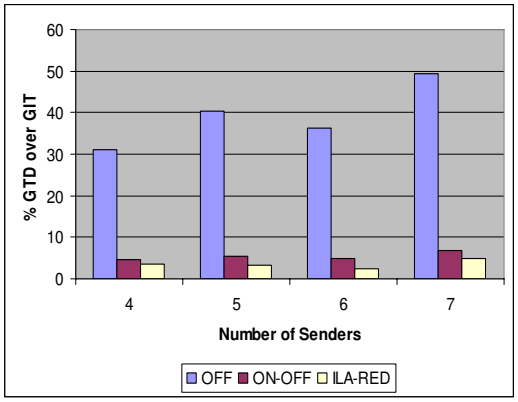


Figure 3.6: Percentage of events with GTD over GIT; AIDT = 30ms.

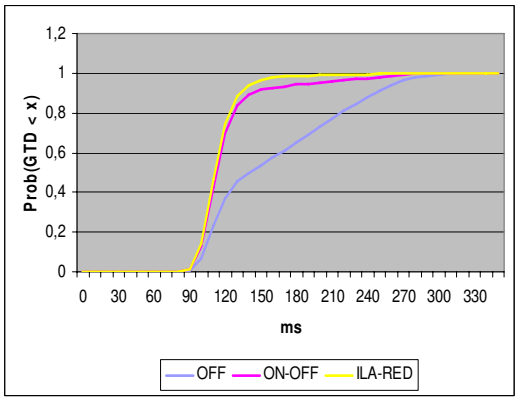


Figure 3.7: Cumulative function of the GTDs in a scenario with 7 GSSs; AIDT = 30ms.

Table 3.2: Maxi, min, average and standard deviation of the GTDs (ms); AIDT = 30ms.

	4 GSSs			5 GSSs			6 GSSs			7 GSSs		
	OFF	**ON-OFF**	**ILA-RED**	**OFF**	**ON-OFF**	**ILA-RED**	**OFF**	**ON-OFF**	**ILA-RED**	**OFF**	**ON-OFF**	**ILA-RED**
MAX	324	324	325	325	324	277	318	319	278	345	345	300
MIN	88	88	86	88	88	88	87	88	88	93	93	93
AVG	142	116	111	153	120	115	148	119	114	170	130	124
ST.DEV	52	30	20	53	32	19	50	28	18	56	32	19

These results are coherent with the values of the average and the standard deviation of the *GTDs* considering all the events transmitted. Table 3.2 shows sensible reductions in the values of both these metrics when ILA-RED or ON-OFF are implemented. Moreover, the two obsolescence-based discarding schemes result more resilient to an increased event generation activity within our mirrored game server architecture. This is evident if the case of seven sending GSSs is compared with the one employing only four sending GSSs. In this case, the average of the *GTDs* decreases from 19.72% (OFF) to 12.07% (ON-OFF) and 11.71% (ILA-RED).

3.7.3 ILA-RED vs ON-OFF: a Comparative Evaluation

The proactive mechanism implemented by ILA-RED does not just slightly improve the already good performance achieved by the ON-OFF scheme. Indeed, Table 3.2 also shows that the standard deviation of the *GTDs* attained employing ILA-RED is always sensibly smaller than the one obtained utilizing ON-OFF. The game actions flow more homogeneously on the screen thus providing a more pleasant game experience for the user.

Not only obtains ILA-RED a slightly better interaction level with respect to ON-OFF, but the total number of discarded events to attain this positive result is definitively lower. Again, with the AIDT parameter equal to 30ms at each GSS, we can analyze the behavior of the compared synchronization schemes in the challenging situation of a stressed game platform. To this aim, Fig. 3.8 shows that the results in Fig 3.6 and Table 3.2 are obtained by ILA-RED at the cost of circa only 40% of the obsolete events dropped by ON-OFF. In essence, by anticipating some event drops, ILA-RED preempts the loss of an acceptable interactivity degree and smoothes the experienced *GTDs*.

Discarding too many obsolete events can result in sudden jumps and temporary interruptions of the images/video flow on the player's screen. Even interpolation experiences an increasing number of errors when gaps are too wide and the consequent jerky rendering could be very annoying for customers. Drops should hence be avoided whenever possible. Since ILA-RED needs less pervasive interventions to be effective, we can say that even if both schemes ensure responsiveness and consistency, ILA-RED outperforms ON-OFF and founds an efficient tradeoff between the percentage of obsolete events to be discarded and a fluent visual progression of the game.

The positive effects gained by the probabilistic preemptive discarding mechanism are further highlighted by Fig. 3.9. The columns in the picture correspond to the number of times the employed scheme resorts to dropping obsolete events to restore the disrupted responsiveness. As it is evident from the great difference between the columns, the preventive probabilistic drop of some game events in phase 1 strongly reduces the number of times ILA-RED needs to discard all the obsolete packets.

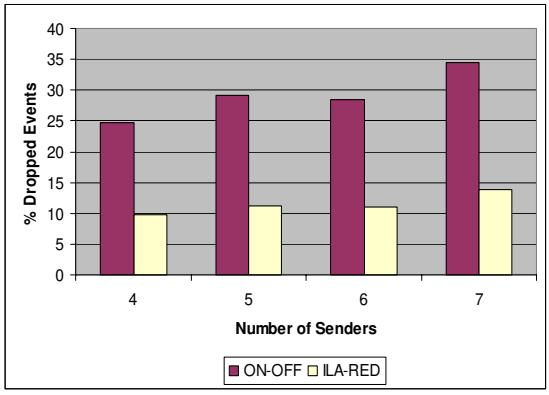


Figure 3.8: Percentage of discarded events; AIDT = 30ms.

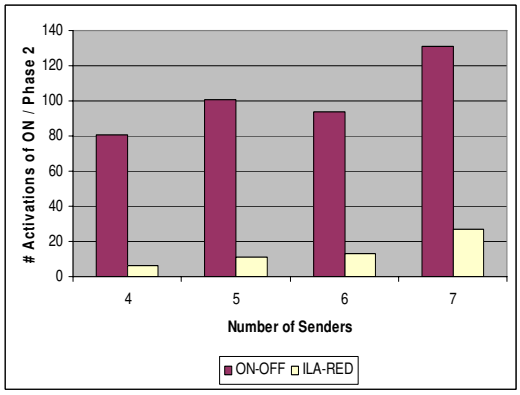


Figure 3.9: # of activations of phase ON and phase 2 for ON-OFF and ILA respectively; AIDT = 30ms.

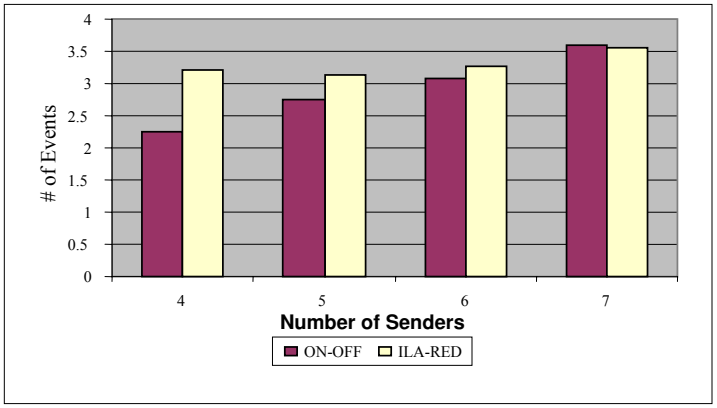


Figure 3.10: Average size of non-interactive bursts; AIDT = 30ms.

Another important metric to better evaluate synchronization schemes is concerned with the distribution of the event drops throughout the game. Fig. 3.10 and Fig. 3.11 provide insight along this direction. In particular, Fig 3.10 shows the average number of consecutive game events with an instantaneous *GTD* value larger than *GIT*. Fig. 3.11, instead, shows the total number of non-interactive bursts obtained during our simulations. As it is evident from these two charts, while ILA-RED and ON-OFF schemes have an

almost equal average size of bursts of non-interactive events, the former allows only a reduced number of these bursts, thus confirming the benefits announced by Fig 3.8.

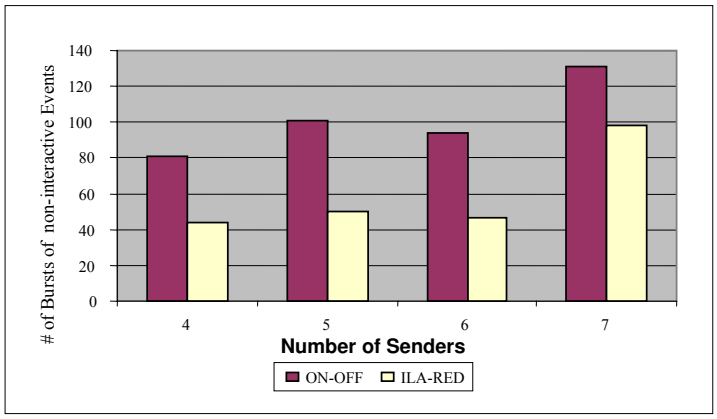


Figure 3.11: Total Number of bursts of non-interactive events; AIDT = 30ms.

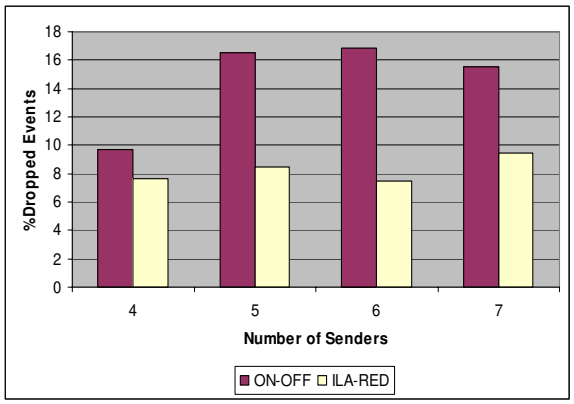


Figure 3.12: Percentage of discarded events; AIDT = 45ms.

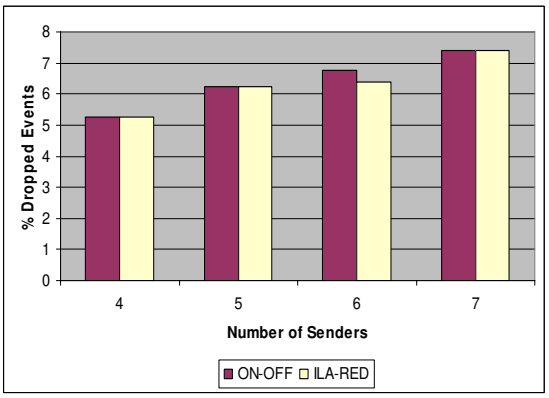


Figure 3.13: Percentage of discarded events; AIDT = 60ms.

3.7.4 ILA-RED: Sensitivity Analysis

It is evident from the previous sections that ILA-RED results more efficient than ON-OFF scheme in ensuring responsiveness whilst minimizing the number of drops. We deem that this advantage becomes more evident when the game event traffic is more

intense, while the two synchronization mechanisms behave the same when plunged into a low stress environment. Simply stated, our hypothesis is that the larger the AIDT value, the more ILA-RED and ON-OFF become equivalent in terms of the number of dropped events required to maintain a proper interactivity degree. We intend to find the breakeven point in the performance curves between ILA-RED and ON-OFF. To this aim, we have run additional simulations with the lognormal distribution of the departing game packets from each server generated with the AIDT value equal to: respectively, 45ms and 60ms.

Table 3.3: Max, min, average and standard deviation of the GTDs (ms); AIDT = 45ms.

	4 GSSs			5 GSSs			6 GSSs			7 GSSs		
	OFF	**ON-OFF**	**ILA-RED**	**OFF**	**ON-OFF**	**ILA-RED**	**OFF**	**ON-OFF**	**ILA-RED**	**OFF**	**ON-OFF**	**ILA-RED**
MAX	331	331	331	331	332	331	315	315	281	338	331	332
MIN	87	87	87	87	87	87	89	89	89	91	91	91
AVG	139	120	120	148	128	125	145	128	125	164	139	135
ST.DEV	45	24	22	46	27	22	43	27	20	49	27	21

Table 3.4: Max, min, average and standard deviation of the GTDs (ms); AIDT = 60ms.

	4 GSSs			5 GSSs			6 GSSs			7 GSSs		
	OFF	**ON-OFF**	**ILA-RED**	**OFF**	**ON-OFF**	**ILA-RED**	**OFF**	**ON-OFF**	**ILA-RED**	**OFF**	**ON-OFF**	**ILA-RED**
MAX	328	326	326	328	325	325	319	286	286	340	328	325
MIN	88	88	88	88	88	88	87	87	86	92	91	90
AVG	144	133	133	154	142	142	153	142	142	169	155	154
ST.DEV	42	29	29	41	29	28	40	29	29	42	27	27

With this greater AIDT value both the links and the GSSs result less congested. As a result, the difference in the number of discarded game events between ILA-RED and ON-OFF progressively diminishes (see Fig. 3.12 and Fig. 3.13). With AIDT = 60ms, the number of discarded events is almost the same for the two synchronization schemes. At this point, however, also the large advantage against the OFF scheme has been sensibly

reduced (see Fig. 3.14 and Fig. 3.15); despite of this, better average and standard deviation of the *GTDs* still persist (see Table 3.3 and Table 3.4).

This phenomenon has two clear explanations. First, having lower game event generation rates decreases both the latencies in the network and the queuing time at the receiving GSS, thus naturally improving the interactivity degree yet without any external intervention. Second, we have to remind that both ILA-RED and ON-OFF schemes rely on discarding obsolete events when the interactivity level decreases, without wasting time in processing them. Consequently, to be effective, these schemes require to have droppable obsolete events queued at the receiving GSS while the server is impacted by processing delays; this engulfment of events waiting for being processed is more likely to happen with lower AIDT values.

3.7.5 ILA-RIO Evaluation

We present some preliminary results related to the use of the ILA-RIO scheme to ensure high responsiveness even with fast paced games. In particular, Fig. 3.16 and Fig. 3.17 refer to specific event trace configurations, respectively utilizing different probabilities of obsolescence among events.

As observable (Fig. 3.16-a, Fig. 3.17-a), in both configurations the obsolescence-based schemes outperform again the traditional OFF method in terms of *GTDs*. Moreover, ILA-RIO further diminishes the number of events that are featured with a *GTD* larger than the predetermined *GIT* w.r.t. ON-OFF. The game evolution fluency results improved as well since ILA-RIO greatly reduces the amount of dropped events required (Fig. 3.17-b). This difference disappears in Fig. 3.16-b; having a smaller probability of obsolescence (50%), in fact, reduces the number of available obsolescent packet that could, in case, be discarded. Thereby, both for ILA-RIO and for ON-OFF, even discarding all the obsolete queued events could still not be sufficient to promptly restore the required high interactivity degree with the considered intense traffic load. Indeed, this is exactly the case when dropping even valid events may become acceptable for fast paced games.

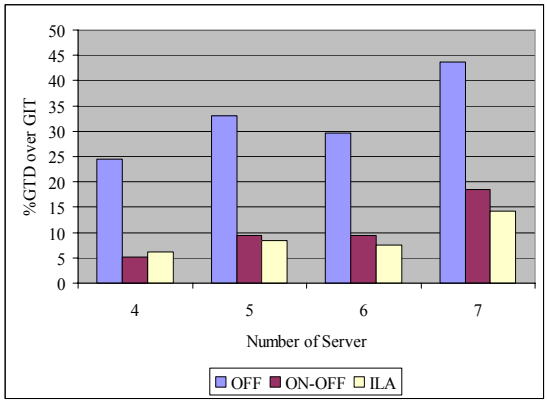


Figure 3.14: Percentage of events with GTD over GIT; AIDT = 45ms

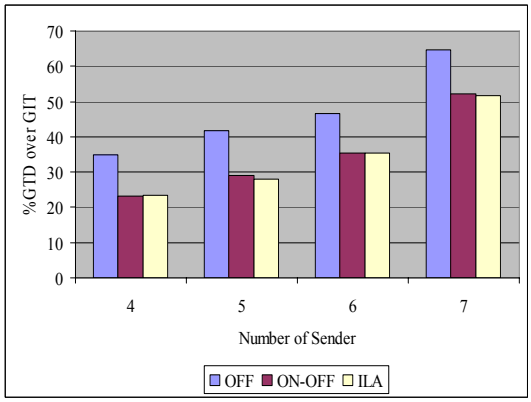
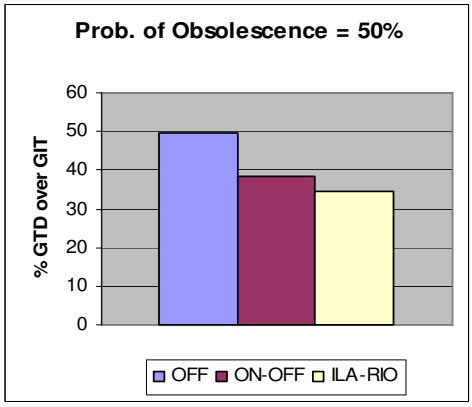
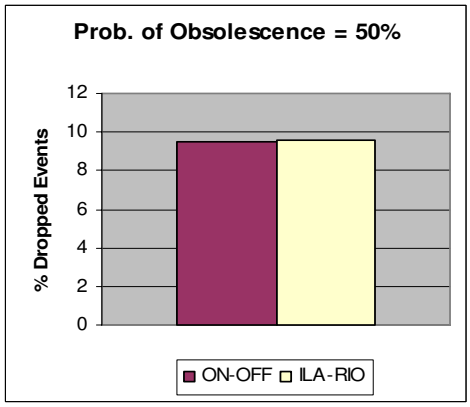


Figure 3.15: Percentage of events with GTD over GIT; AIDT = 60ms.

Finally, we evaluated the amount of valid game events dropped by our ILA-RIO approach. Table 3.5 reports the percentage of obsolete and valid events that are discarded, depending on the event trace. As expected, the amount of dropped valid game events diminishes as the percentage of obsolete ones becomes greater. This tendency is due to the fact that if an adequate number of obsolete events are available during the events exchange activity, then our scheme can exploit the drop of all these (obsolete) events to restore responsiveness when entering phase 2, with no need of resorting to valid ones.

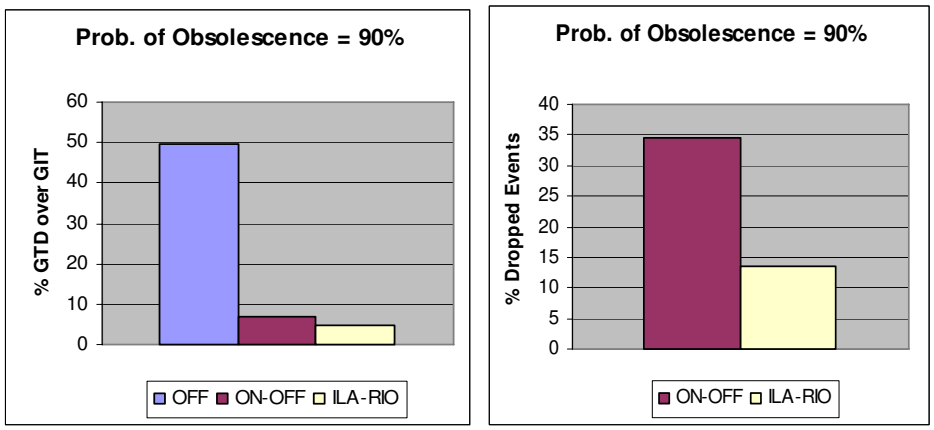
(a)





(b)

Figure 3.16: Probability of obsolescence = 50%;
(a) Event percentage having GTD > GIT; (b) Percentage of discarded events.



(a) *(b)*

Figure 3.17: Probability of obsolescence = 90%;
(a) Event percentage having GTD > GIT; (b) Percentage of discarded events.

Table 3.5: Percentage of obsolete and valid discarded events in ILA-RIO.

Obsolescence Prob.	50%	90%
Obsolete	9,46%	13,64%
Valid	0,16%	0%

3.7.6 Optimistic Obsolescence-based Synchronization: Simulation

To assess benefits on the provided interactivity degree attainable through the OOS approach, we compare it with OFF, ON-OFF, ILA-RED, and TW. In particular, we measure the percentage of game events arrived at GSS_0 with a GTD value larger than GIT. To fully appreciate the results, we have to keep in mind that our experimental analysis also includes the computational cost paid for checking obsolescence and correlation, and to perform rollbacks when necessary. In particular, Fig. 3.18 – 3.21 report the average percentage of GDTs above GIT, depending on the number of involved GSSs, as a function of the non-correlation probability. According to all experimental configurations, OOS outperforms the other schemes regardless of the number of sending

GSSs. It is also worth noticing that OFF and TW curves lie horizontal, as they do not implement any mechanism related to obsolescence and correlation.

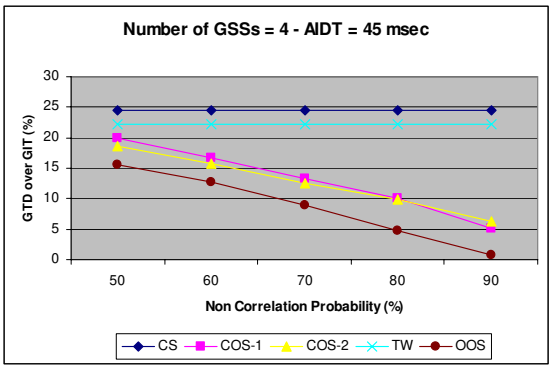


Figure 3.18: Percentage of events with GTD over GIT; 4 GSSs

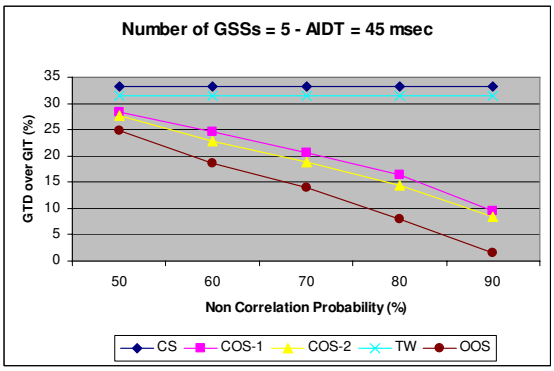


Figure 3.19: Percentage of events with GTD over GIT; 5 GSSs

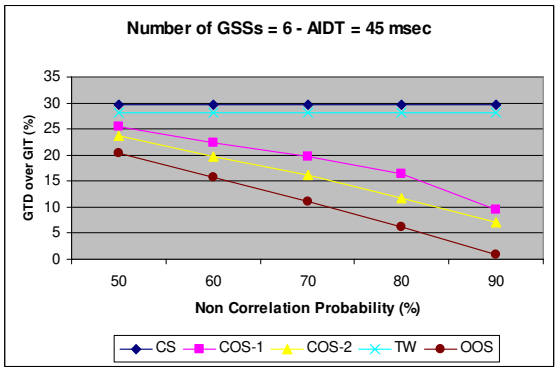


Figure 3.20: Percentage of events with GTD over GIT; 6 GSSs

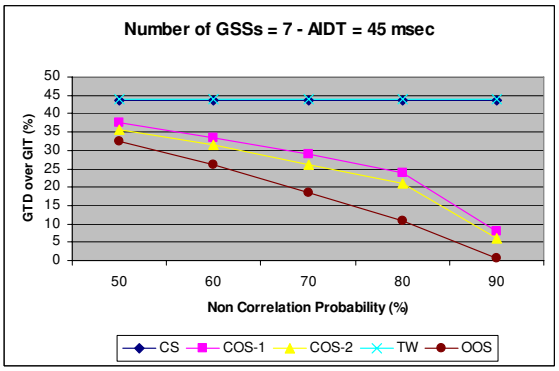


Figure 3.21: Percentage of events with GTD over GIT; 7 GSSs

Focusing on the amount of obsolete events that are dropped to maintain the interactivity degree within an acceptable value Fig. 3.22 – 3.25 shows how OOS reduces the number of discarded events in all the simulated configurations with respect to all the other schemes. This effect is derived from the fact that OOS's strategy accelerates fresher event processing, thus diminishing the possibility for an event to become obsolete.

Needless to say, only schemes that are able to drop obsolete events are considered here in these figures.

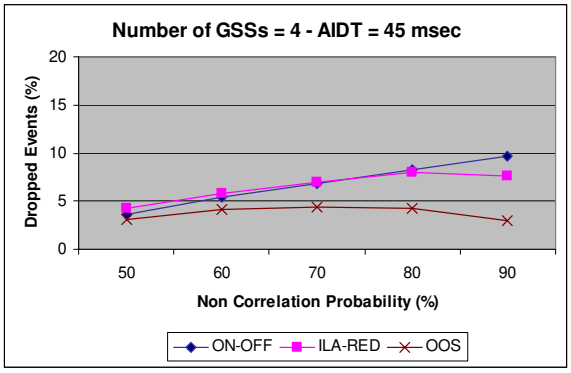


Figure 3.22: Percentage of dropped events; 4 GSSs

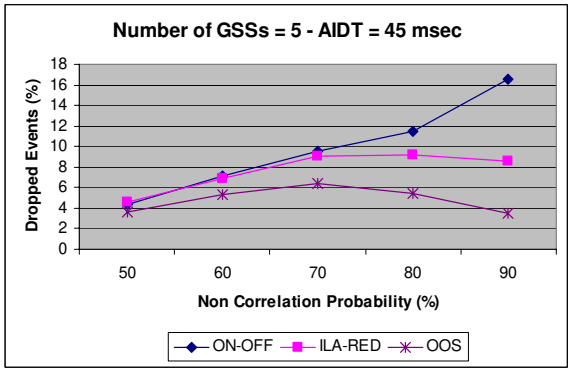


Figure 3.23: Percentage of dropped events; 5 GSSs

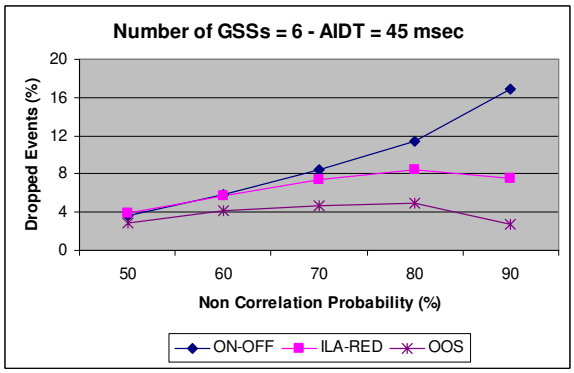


Figure 3.24: Percentage of dropped events; 6 GSSs

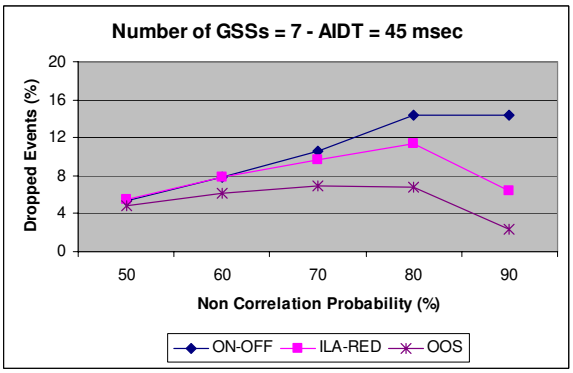


Figure 3.25: Percentage of dropped events; 7 GSSs

We measured the number of rollbacks needed to maintain the consistency of the game state by comparing the two considered optimistic synchronization algorithms: TW and OOS. Since a reduced number of rollbacks implies a higher interactivity degree, this represents a fundamental metric. Needless to say, OFF, ON-OFF, and ILA-RED never performs (nor need to) rollbacks as they are conservative schemes.

In Fig. 3.26 – 3.29 we show the rollback ratio for TW and OOS. In simpler words, we measure the total number of rollbacks in the system over the total number of generated

events for various non-correlation probabilities. Each chart refers to a different scenario with a different number of sending GSSs. It is worth noticing that the TW curves lie horizontally, as standard TW does not gain benefits from obsolescence and correlation. Moreover, in all the considered configurations, OOS is able to outperform TW since it avoids to trigger the rollback procedure for obsolete events.

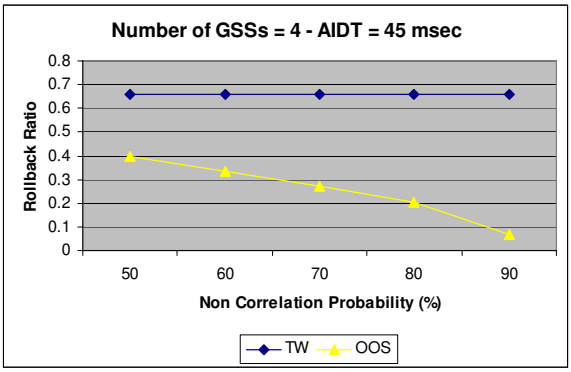


Figure 3.26: Rollback ratio; 4 GSSs

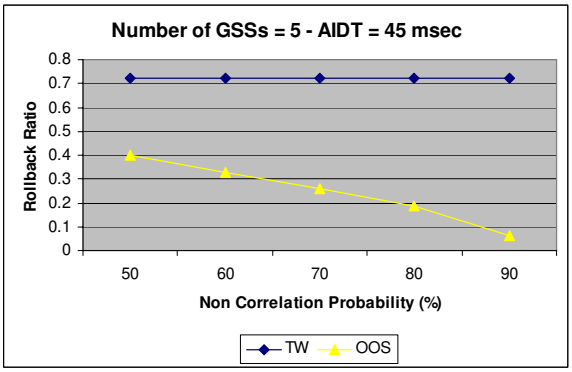


Figure 3.27: Rollback ratio; 5 GSSs

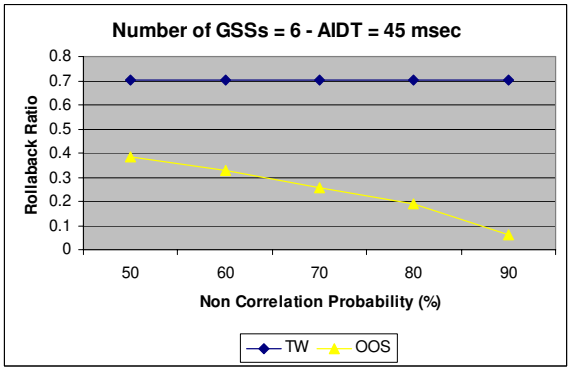


Figure 3.28: Rollback ratio; 6 GSSs

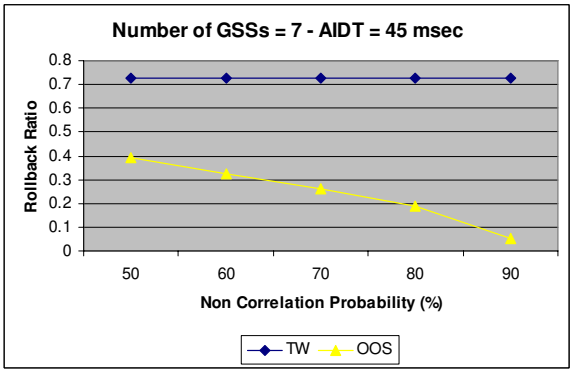


Figure 3.29: Rollback ratio; 7 GSSs

Finally, Fig. 3.30 – 3.33 report the average number of re-processed events within a single rollback. In general, OOS reduces this value with respect to TW since it avoids the re-execution of those events that become obsolete during the evolution of the game (and during the rollback).

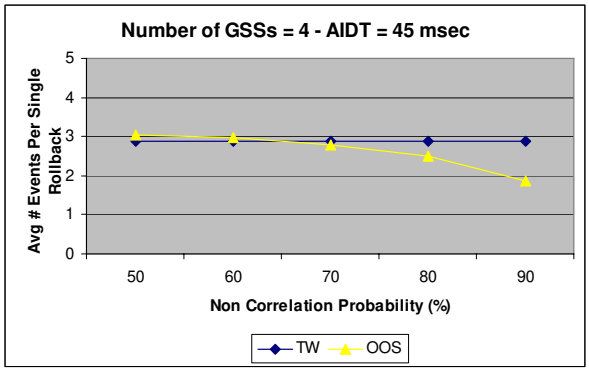


Figure 3.30: Number of events re-processed per rollback; 4 GSSs

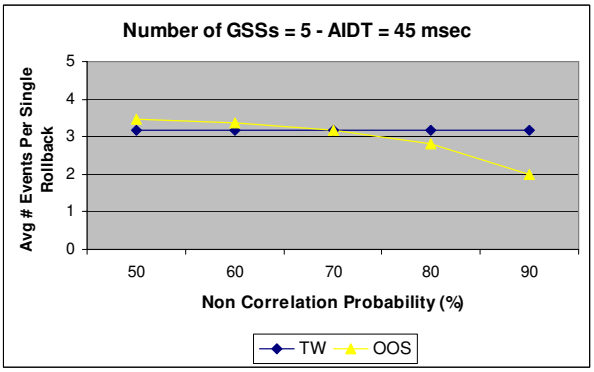


Figure 3.31: Number of events re-processed per rollback; 5 GSSs

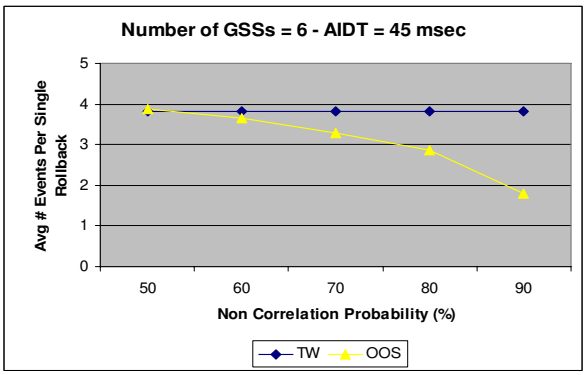


Figure 3.32: Number of events re-processed per rollback; 6 GSSs

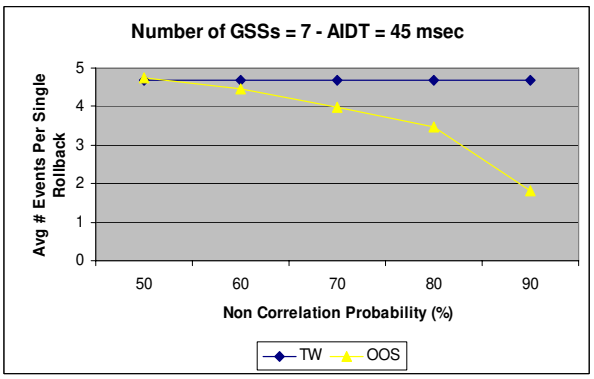


Figure 3.33: Number of events re-processed per rollback; 7 GSSs

3.8 Real Experiment Results

In previous sections, we presented results from an extensive set of simulations that has been run to demonstrate the efficacy of discarding obsolescence game events to preserve interactivity. Instead, in this section, we report on real testbed experiments that have been run by Ferretti *et al.* to compare one of the schemes utilizing the obsolescence notion against a traditional one, in a real context [185].

3.8.1 Experiment Assessment

The testbed was composed by three GSSs embodying a mirrored game architecture. One of them was placed in Bologna, Italy, another one in Cesena, Italy, and the third one was in Los Angeles. Average latencies for the two GSSs in Italy were in the order of 4 ms, whereas latencies from each of the GSSs in Italy and the one in California were in the order of 95 ms. Each server was made able to utilize the OOS scheme or the traditional TW. A periodic physical clock synchronization was taking care of maintaining the three servers on the same time evolution and minimize clocks' drift.

Clients connected to GSSs were emulated with an event generation rate following a lognormal distribution, as inspired by literature on games, and varying from a normal traffic situation to an intensively loaded one [73, 77]. Specifically, the AIDT was varying from 30 ms (intense game traffic) to 45 ms (moderate game traffic), with 10 ms of standard deviation. Each GSSs had 10 emulated clients connected which, following the approximation suggested in [146], generates an average game event size equal to 200 Bytes.

The probability that a given event is non-correlated to other events was varying from 50% to 90% [81]. A higher non-correlation probability entails a higher probability that a new event makes obsolete previous ones generated by the same players.

3.8.2 Optimistic Obsolescence-based Synchronization: Testbed

As for the simulative evaluations, even in this case, interesting metrics to evaluate the efficacy of compared schemes in supporting online gaming are: i) the amount of game events with a GTD higher than the GIT; ii) the number of time the rollback procedure has being activated; and iii) the amount of obsolete events which are dropped by the OOS scheme.

Indeed, Fig. 3.34 and Fig. 3.35 report the average amount of events that experienced a GTD value higher than the GIT, for 30 ms and 45 ms of AIDT, respectively. It is evident that the OOS scheme performs better than Time Warp, ensuring a higher interactivity degree in all the considered settings.

This happens thanks to OOS' discarding mechanism. Indeed, Fig. 3.36 and Fig. 3.37 show the average percentage of obsolete events dropped at each GSS, with AIDT = 30 ms and AIDT = 45 ms, respectively. As expected, the GSS located in Los Angeles drops a higher amount of game events, as it is affected by higher network latencies, being very far from both the other two GSSs, whereas GSSs in Italy are, at least, very close one another. It is worth noticing that in both charts, the number of dropped events initially grows with the percentage of non-correlation. This happens because the more the number of available events that become obsolete, the more each GSS is able to drop obsolete events to augment interactivity. Surprisingly, once surpassed a given percentage of non-correlation probability, the number of dropped events decreases considerably. This is probably due to the fact that, with a high non-correlation (i.e., higher obsolescence) probability, OOS is able to anticipate the loss of interactivity; thus, game events are typically processed before they could become obsolete. Obviously, performances are better with a higher AIDT, since a slower pace of game event generation results in less transmissions and processing, and hence less causes for interactivity loss.

A comparison among the average rollback ratios achieved by the two compared schemes is reported in Fig. 3.38 and Fig. 3.39. Specifically, values are computed dividing the total number of rollbacks over the total number of generated events. Since obsolete events are dropped during the event notification activity, the number rollbacks required by OOS is inferior to that of Time Warp. Yet, both optimistic schemes have to frequently resort to rollbacks even at high non-correlation probability.

Finally, these results confirm those obtained thought the use of simulations, thus confirming the efficacy of resorting to obsolescence-based discarding mechanism to improve interactive gaming experiences.

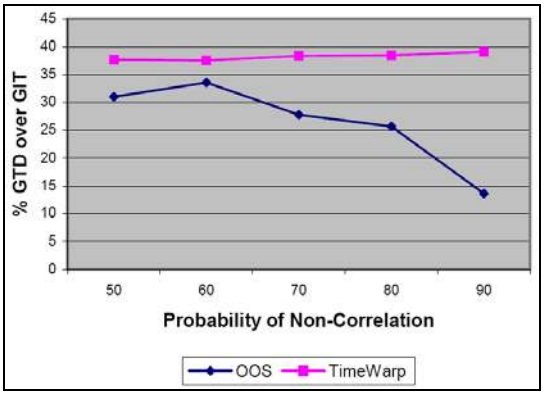


Figure 3.34: Percentage of events with GTD over GIT; AIDT = 30 ms

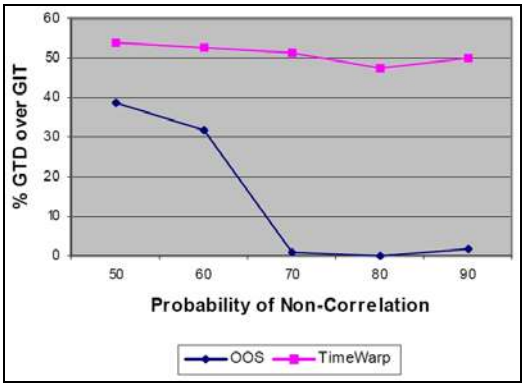


Figure 3.35: Percentage of events with GTD over GIT; AIDT = 45 ms

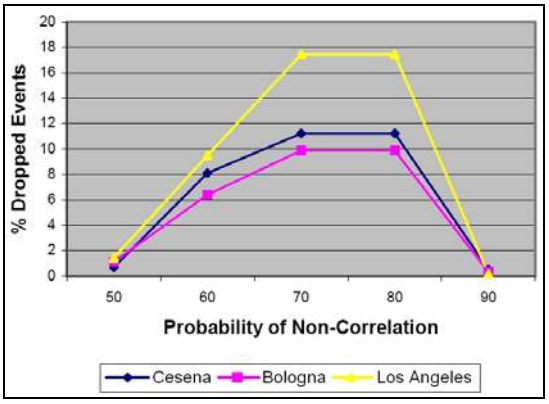


Figure 3.36: Percentage of dropped events; AIDT = 30 ms

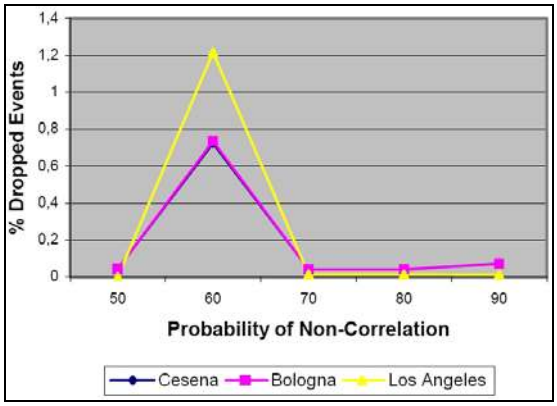


Figure 3.37: Percentage of dropped events; AIDT = 45 ms

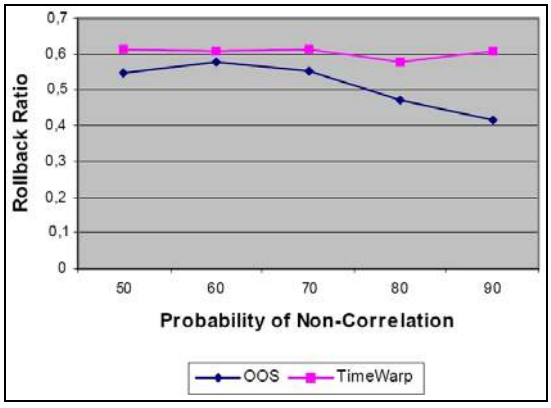


Figure 3.38: Rollback ratio; AIDT = 30 ms

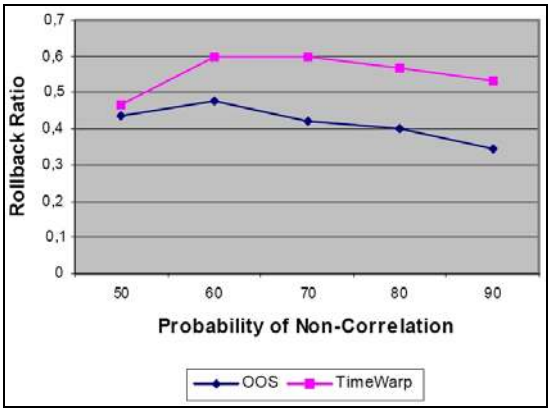


Figure 3.39: Rollback ratio; AIDT = 45 ms

CHAPTER 4

To Seek the Fairness by Way of Interactivity

Christopher Columbus's aim, when he sailed for finding Catai (ancient China) lands across the Atlantic Ocean, is perfectly described by the famous claim credited to him: *"Buscar el Levante por el Poniente",* to seek the East by way of the West. We humbly take inspiration from his genius to synthesize our work in the title of this Chapter. The analogy is represented by the fact that the scheme we propose facilitates fairness by aiming at increasing the interactivity degree in MMOGs.

We demonstrate how it is possible to take advantage of a reduced transmission time to magnify the efficiency of a local lag-type algorithm in ensuring fairness. This represents a very important result since it contradicts the general belief that either interactivity or fairness has to be sacrificed to achieve the other.

4.1 Exploiting Local Lag Techniques

As shortly discussed in Section 2.3.2, Local Lag and other similar algorithms have been proposed to ensure fairness (and consistency) among players in MMOGs [54, 78, 94, 104, 105]. The idea behind this kind of approach amounts to introducing artificial delays in the display of both generated and received game events. These delays are appropriately chosen for each client and depend on their subjective client-server latencies. The aim is that of having each game event simultaneously displayed, after a total amount of time since its creation, on all the players' screens.

Not only does this help in maintaining fairness and consistency, it could also augment the playability of the game. Up to a certain level of injected delay, in fact, players could be more comfortable with a higher but fixed delay than with smaller but variable ones

[78]. In the former case, in fact, a player can adapt her/his actions and reaction times to consider also this fixed delay and achieve higher performance than in the latter situation. For instance, car race players will learn how to constantly anticipate the steering when in proximity of a curve; first person shooter players will learn how to aim a little bit ahead of their adversary's position with respect to their moving direction; etc.

Lag compensation can thus be proficiently employed to ameliorate the negative effects of latency on MMOGs. However, the amount of time used as a parameter for local lag generally corresponds to the longest transmission latency experienced by the most unlucky player of the game. In practice, this kind of approach increases game delays and may jeopardize interactivity as in some case the unlucky client may be connected very far away from its server and/or through a slow connection.

Consequently, the efficiency and applicability of the local lag approach strongly depend on the network conditions and on the interactivity degree required by the game. Indeed, especially in the case of a highly interactive MMOG, servers should be optimally located to efficiently serve a large number of customers [76]. Yet, guaranteeing both interactivity and full fairness through local lag can sometimes be achieved only at the cost of impeding the access to some users whose connectivity is irremediably affected by large network delays.

A tradeoff relationship thus exists among scalability (especially in terms of geographical dispersion of the players), interactivity, and fairness. According to this, interactivity and fairness are traditionally seen as incompatible requirements in MMOGs. Conversely, we claim now that upholding interactivity may be useful also to the aim of ensuring fairness. To demonstrate this, we have developed a novel mechanism named Fairness and Interactivity Loss Avoidance (FILA) [164, 165]. Our scheme can be divided into two complementary sub-components. The first one exploits the semantics of the game to drop superseded events and speed up the delivery of game events as seen in Chapter 3. The second one takes advantage of this reduced transmission time to magnify the efficiency of a local lag-type algorithm in ensuring fairness without compromising interactivity.

4.2 Achieving Fairness through Interactivity

FILA can be thought of as comprised of two complementary parts. The first one, enforced among GSSs, takes substantial inspiration from the aforementioned ILA-RED scheme to speed up the delivery of "fresh" game events by dropping some events which have become obsolete. The second part takes advantage of this reduced transmission time to magnify the efficiency of a local lag-type of algorithm to ensure fairness. FILA utilizes (2.4) to determine the visualization time of a game event, thus providing fairness without compromising interactivity.

To calculate the appropriate δ in (2.3), the *OD* should be determined for each player. For this reason, game events are marked at their creation with a generation timestamp and then sent to the destination: hence, they are orderable. Obviously, a global concept of time has to be maintained in the system. This can be achieved through a variety of solutions that enable the synchronization of GSSs' physical clocks [67, 111], or by employing new technological synchronization devices such as GPS. Thanks to this, GSSs are able to monitor the *ODs* of their engaged players and make them available for the FILA algorithm.

More in detail, the first part of FILA drops queued obsolete game events with a certain probability P_d when the average *OD*, namely *avgOD*, value increases putting at risk the interactivity of the system. The discarding probability P_d is directly proportional to *avgOD* and dependent on a constant Pmax. Instead, the value for *avgOD*, at iteration *n*, is computed through the low-pass filter showed in (4.1), where w is a parameter that determines how close the average follows the sample trend:

$$avgOD_n = avgOD_{n-1} + w \times (sample_n - avgOD_{n-1}). \tag{4.1}$$

More in detail, with FILA, all the game events are regularly processed and forwarded while *avgOD* is smaller than an alert threshold named *tmin*. When *avgOD* exceeds *tmin*, the GSSs drop obsolete events with probability P_d, with neither processing nor

forwarding them. Finally, if *avgOD* exceeds the subsequent *tmax* (>*tmin*) threshold, then P_d is set equal to 1 and all obsolete events waiting for being processed are discarded.

This stabilization mechanism succeeds is reducing $OD_i(e)$ by impacting on $q_i(e)$. In fact, the time spent in queue by a certain event is diminished by the spared processing time of preceding obsolete events which have been dropped with neither processing nor forwarding them. Moreover, since only obsolete events are discarded, FILA fully maintains consistency in the game evolution [62, 109].

To explain FILA more in detail we use the clarifying help of Fig 4.1 which provides the graphical definitions for some terms utilized in our explanation: *OD*, *GTD*, and *LHD* (Last Hop Delay).

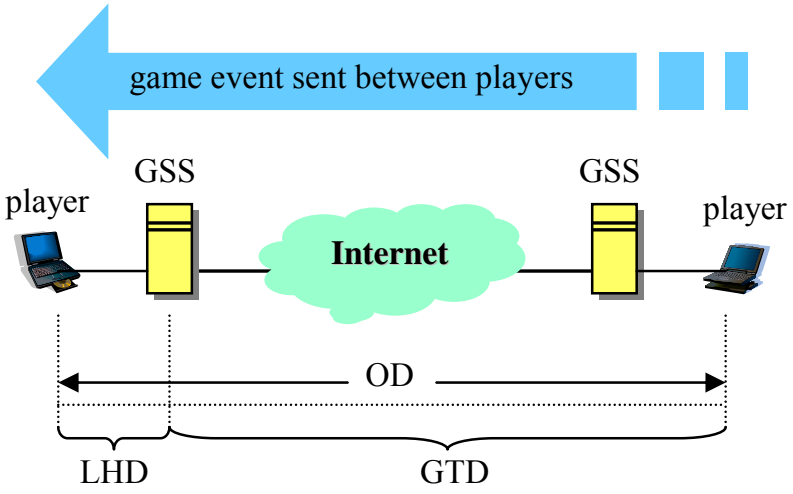


Figure 4.1: Delay definitions.

First of all, it should be noticed that FILA performs its operations on the GSSs. This choice helps us in maintaining a simpler control of the exploited game platform. Under that circumstance, however, for each event *e*, GSSs can compute *GTD(e)* but not *LHD(e)*. An estimation of *LHD(e)* is necessary in order to compute *OD* and utilize it in our algorithm. For this reason, each GSS continuously monitors the latencies to each of its engaged players and maintains a variable named λ_{GSS}. The value of this variable

represents the maximum among the latencies from the considered GSS to each of its connected clients (this set of clients is named *C_GSS*) and is calculated as follows:

$$\lambda_{GSS} = \max_{i \in C\_GSS} \{ LHD_i \}. \tag{4.2}$$

However, we cannot let some irremediably delay-affected client to excessively impact on the calculations performed by our scheme. Utilizing, in FILA, the excessively high λ_{GSS} generated by some player connected very far away from the GSS, in fact, would result in very high sample (and *avgOD*) values with respect to *GIT*. In this case, FILA would increase the aggressiveness of its discarding function as perceived by all the players with no positive results: the "unlucky" player will still not be able to receive game events with delays below the interactivity threshold. For this reason, we need to consider a *Delay Upper Bound* (*DUB*) in order to limit the impact of "unlucky" players on its algorithm. To this aim, (4.3) provides the formula for a fundamental parameter utilized by FILA to handle the impact of *LHD(e)* on its algorithm:

$$\sigma = min\{ \lambda_{GSS}, DUB \}. \tag{4.3}$$

The usage of this parameter depends on the employed version of our scheme; to determine *DUB* we rely on a heuristic that dynamically computes its value based on the general network condition during the game. Its formula is as follows:

$$DUB = GIT - max\{ GTD \}. \tag{4.4}$$

where $\max\{GTD\}$ represents the largest among the *GTDs* experienced over all the connections between each GSS and the players engaged by the other GSSs.

To compute *DUB*, each GSS has hence to periodically determine the *GTD* that features in average the slowest of its connections with players engaged by the other servers. Then, this value has to be communicated back to all the other peers in order to allow a global knowledge of the worst *GTD* endured by each GSS. Finally, the highest

among these maximum *GTDs* can be univocally determined by each of the GSSs and used to determine the global *DUB* in the system as shown in (4.4).

The second part of FILA is simply in charge of equalizing the delay differences among players with a local lag-type scheme that appropriately computes the δ value shown in (2.3) so as to satisfy (2.4) whenever possible. We are going now to empirically demonstrate how the combination of phase one and two is effective in ensuring fairness and interactivity while allowing a scalable number of contemporary players.

We compare the regular local lag (LL) mechanism against FILA. In particular, LL embodies the traditional local lag scheme with no discarding mechanism for obsolete events. Even in this case, however, as for all the other compared protocols, the algorithm is not allowed to introduce artificial delays if this would result in jeopardizing interactivity (i.e. $t^{v(e)}$ cannot be set greater than $t^{g(e)} + GIT$).

Focusing on FILA, we have set $tmax = GIT$ and $tmin < tmax$. Moreover, the estimation of the $LHD_i(e)$ is performed through (4.2) and (4.3) and utilized to calculate the new sample upon every new event arrival as follows:

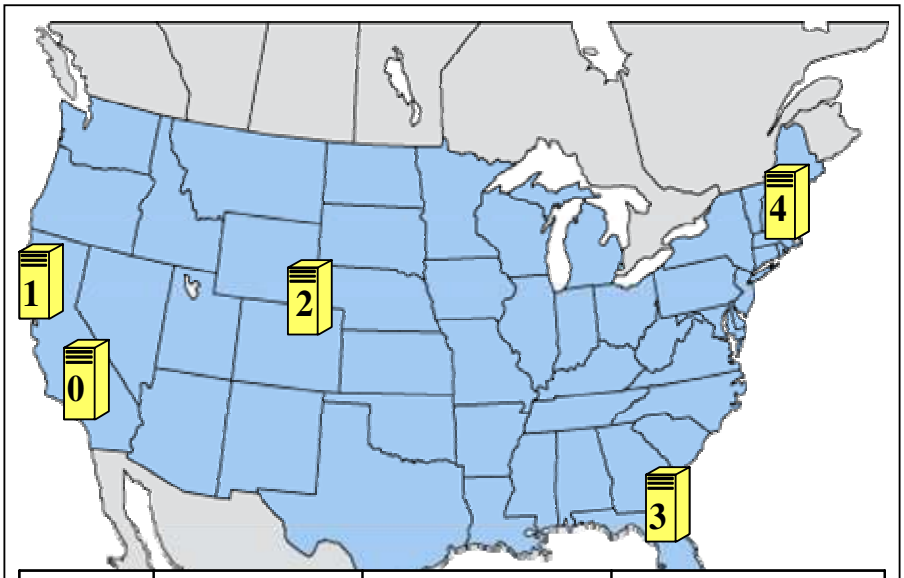
$$sample(e) = GTD(e) + \sigma. \tag{4.5}$$

4.3 Simulation Assessment

It is well known that MMOG service providers should appropriately position their game servers in such a way that their target player market would be located within a circle having 150-180ms of latency diameter [76]. Following this rule and aimed at creating a configuration able to factually support a highly interactive MMOG, we have simulated a constellation of five GSSs deployed across U.S.A. by choosing optimal market locations.

Clients are supposed to be distributed all over the North American continent and connected through various access technologies that provide them with different access delays. We have focused our attention on the event receiving aspect of a single GSS

(GSS_0), pretending that the other GSSs are sending events to it (without any loss of generality).



GSS	Avg latency	City	Road distance
0	0 ms	Los Angeles	0 km
1	20 ms	San Francisco	602 km
2	35 ms	Denver	1650 km
3	50 ms	Orlando	3870 km
4	65 ms	Boston	4896 km

Figure 4.2: Game servers deployment.

Inspired by literature [77], the *GTD* values were chosen based on a lognormal distribution whose approximate average was obtained by means of repeated runs of the ping application. More in detail, game events coming from clients connected to the sending GSSs (i.e. GSS_1–GSS_4) and traveling towards GSS_0 experience average latencies as reported in Fig. 4.2, with a standard deviation of 10ms. Further, several scenarios were considered where the values of $\max_{i \in C\_GSS} \{ LHD_i \}$ were chosen for each GSS within the

following set {25ms, 50ms, 75ms, 100ms, 125ms, 150ms}. This choice is justified by the consideration that clients should be located within a circle having a maximum latency diameter of 150ms. We assumed to have 10 clients connected to each GSS, engaged in a fast-paced game, and generating a new action every 300ms in average. This results in a flow of game events having 30ms of inter-departing time. Finally, the average game event size (200 Bytes) was inspired by literature about games as well [17].

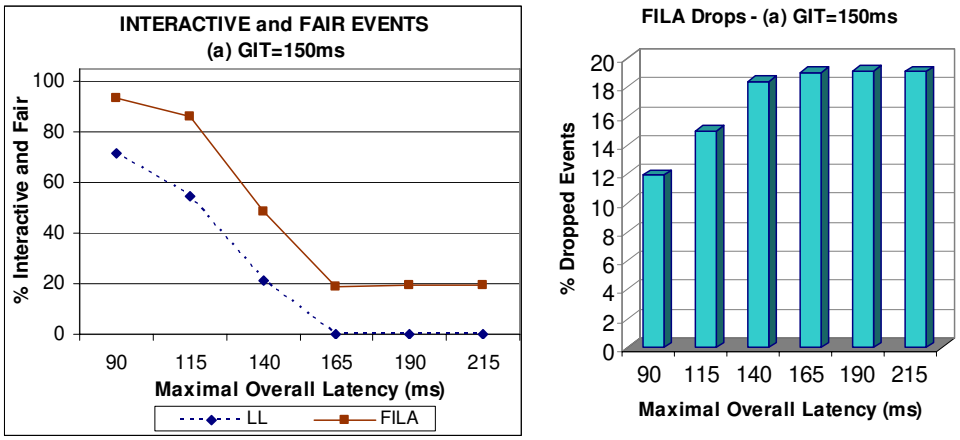


Figure 4.3: Interactivity and fairness improvement (left) and dropped events (right) with GIT=150ms and AIDT=30ms.

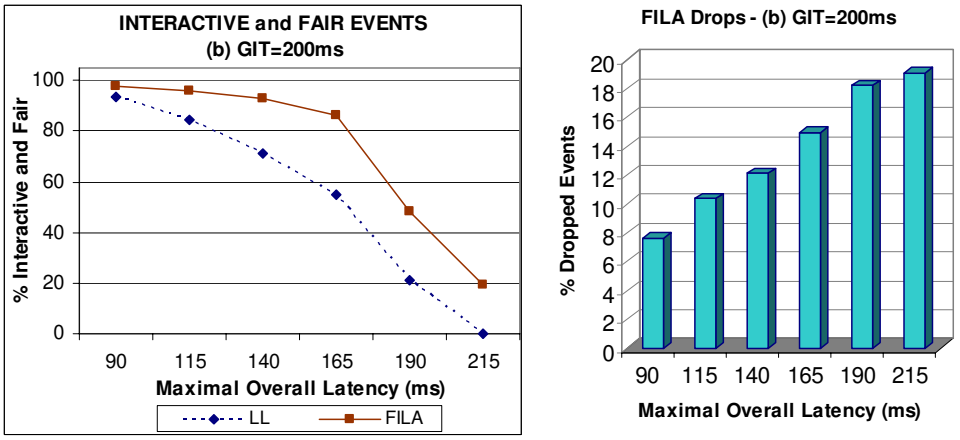


Figure 4.4: Interactivity and fairness improvement (left) and dropped events (right) with GIT=200ms and AIDT=30ms.

Focusing on the parameters in the FILA algorithm, we have set w = 1/8 for all the simulations. The alert threshold *tmin* was equal to *GIT* – 100ms and the probability that an event makes obsolete preceding ones was set to 90%. This represents a realistic scenario for a vast plethora of possible games (e.g. adventure, strategic, vehicle race, flight simulator, etc.), where most of the events are just independent movements.

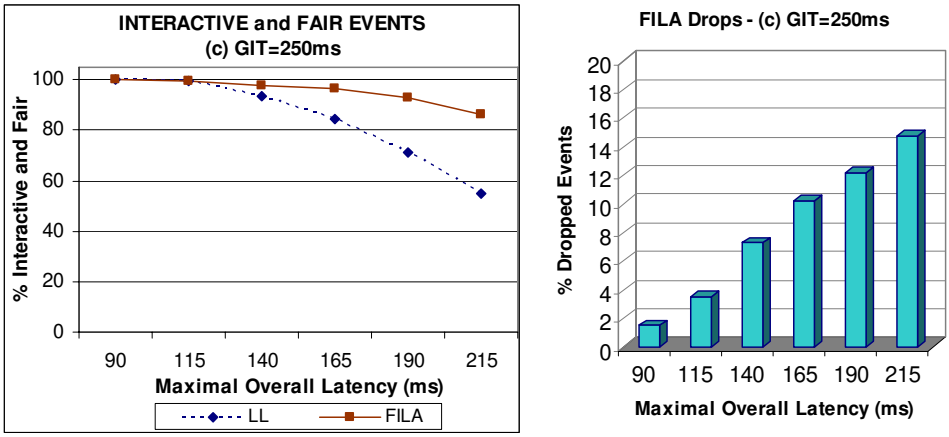


Figure 4.5: Interactivity and fairness improvement (left) and dropped events (right) with GIT=250ms and AIDT=30ms.

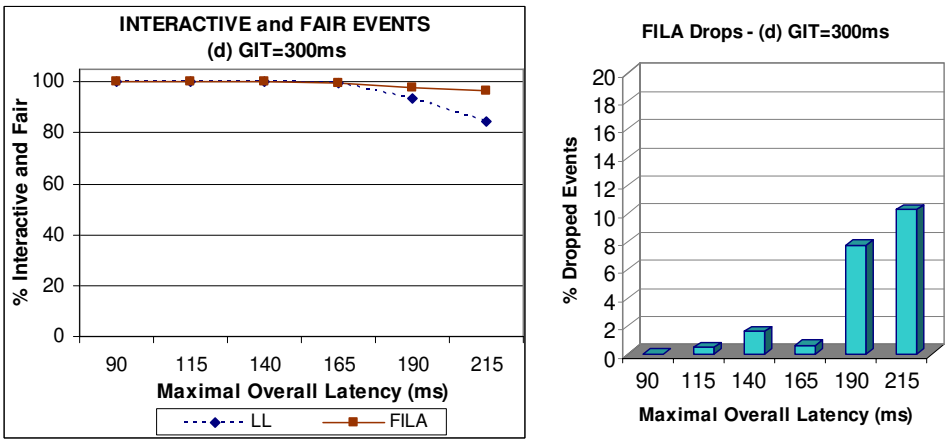


Figure 4.6: Interactivity and fairness improvement (left) and dropped events (right) with GIT=300ms and AIDT=30ms.

Each experiment was identically replicated to compare the outcomes of FILA against the regular LL algorithm. In [104], Zander *et al.* demonstrated that there is a statistically significant difference between the mean kill rates of player groups which are affected from diverse client-server latencies. In essence, lower latencies results in higher mean kill rates and thus in unfairness. Coherently, we have chosen to evaluate as a performance parameter the percentage of events that were delivered by GSS_0 to *all* of its players in time to be contemporary visualized before the *GIT* expiration. We are hence considering the achievement of per-event interactivity and fairness.

4.4 Results

In this section, we demonstrate through results how FILA is able to ensure a higher interactivity and fairness degree if compared to the traditional LL scheme. Moreover, we provide a scalability evaluation of the two mechanisms showing how FILA improves its performance in situation with intense traffic and outperforms regular LL.

4.4.1 Interactivity and Fairness

Fig. 4.3 – 4.6 show, respectively, four different sets of experiments, obtained varying the *GIT* from 150 ms to 300 ms. Each set was comprised of six different experiments and each experiment consisted in the transmission of about 4000 game events which experienced, in the worst case, a maximal overall latency whose value is reported on the x-axis of each provided chart. The maximal overall latency represents the largest average latency experienced on the connection between any two players in the system.

The leftmost graphs of Fig. 4.3 – 4.6 show the percentage of game events that GSS_0 was able to deliver to all of its engaged players in time to be simultaneously delivered with an *OL* lower than *GIT*. It hence represents the amount of events which satisfied condition (2.2) and were thus fairly processed by all the clients.

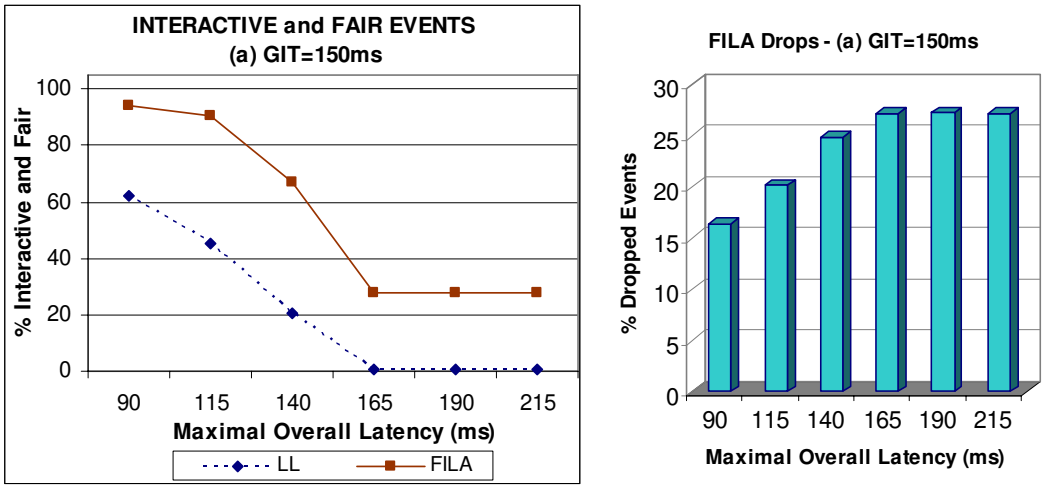


Figure 4.7: Interactivity and fairness improvement (left) and dropped events (right) with GIT=150ms and AIDT=20ms.

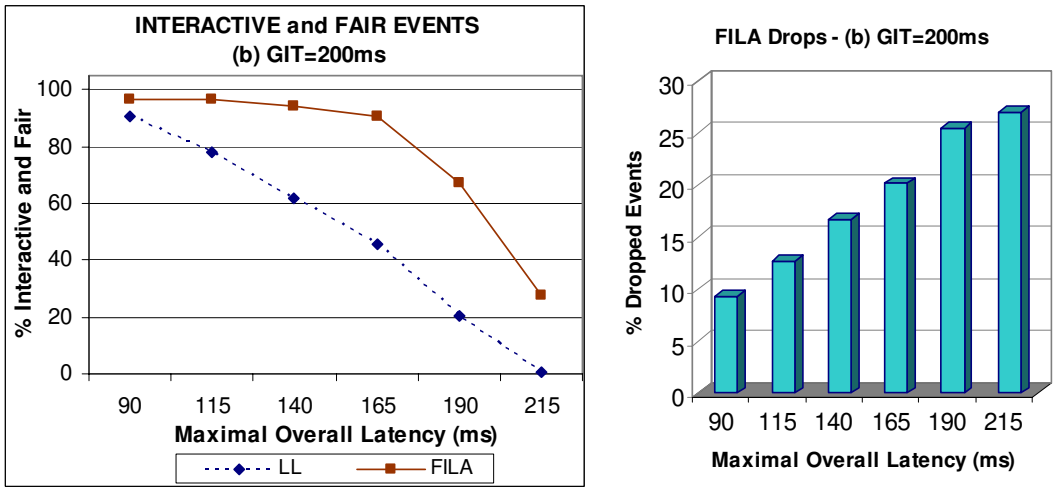


Figure 4.8: Interactivity and fairness improvement (left) and dropped events (right) with GIT=200ms and AIDT=20ms.

As can be seen from these graphs, having a higher *GIT* improves the efficacy of both the evaluated schemes since larger local lags can be utilized. However, regular LL algorithm experiences a premature performance decrease when the maximal overall latency increases even if it is still far from the *GIT*. Instead, FILA ensures a good fairness degree for a larger set of overall latencies.

Results turn out to be even better if we focus only on those cases where the overall latency is not irremediably high with respect to *GIT*. Considering the configurations when the maximal overall latency is lower than *GIT* by 35 ms or more, we find that FILA always guarantees more than 86% of fairly delivered game events with less than 15% of dropped events.

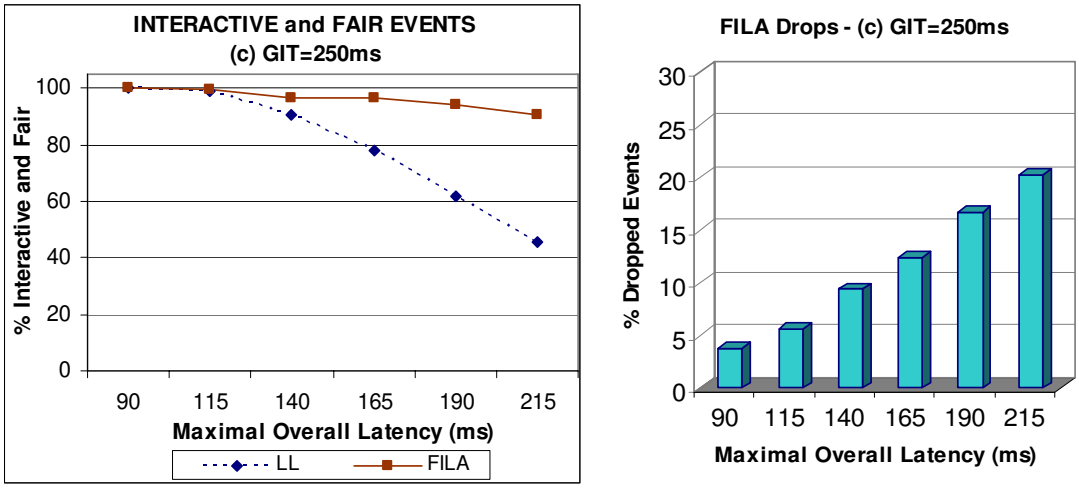


Figure 4.9: Interactivity and fairness improvement (left) and dropped events (right) with GIT=250ms and AIDT=20ms.

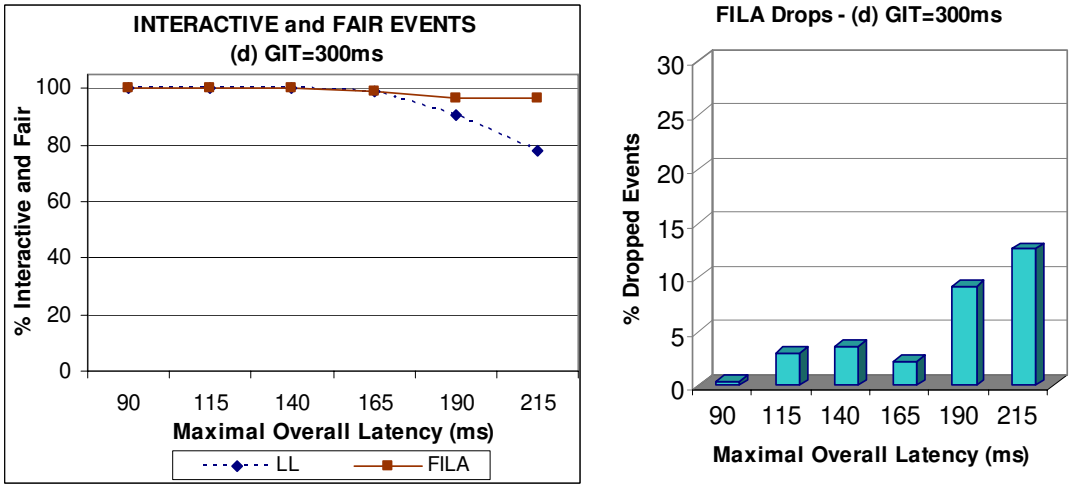


Figure 4.10: Interactivity and fairness improvement (left) and dropped events (right) with GIT=300ms and AIDT=20ms.

Obviously, in those configurations where the maximal overall latency is close to (or surpasses) GIT, both schemes cannot overwhelm network conditions, thus achieving poor fairness (and interactivity). Even in this case, however, FILA behaves better than the regular LL algorithm.

FILA pays these better results with the drops of some obsolete events. Specifically, the rightmost charts of Fig. 4.3 – 4.6 reveal the percentage of game events which were discarded by FILA. In all the considered cases, less than 20% of the game events were dropped and these events were exclusively obsolete ones.

4.4.2 About Scalability

In order to test the scalability of FILA and LL, we have decreased the AIDT to generate scenarios with a higher level of game traffic in the network. In particular, Fig. 4.7 – 4.10 refer to configurations with 20 ms of AIDT, while Fig. 4.11 – 4.14 correspond to the cases where AIDT is equal to 10 ms. Again, in each figure we present the outcomes for four different GIT values: 150 ms, 200 ms, 250 ms, and 300 ms, respectively. The leftmost charts show the percentage of game events that were fairly and interactively delivered to all the clients engaged by GSS_0. Instead, the rightmost ones reveal the percentage of game events which were discarded by FILA.

As one can expect, the higher the game traffic, the lower the interactivity and fairness degree provided by LL. On the contrary, not only is FILA able to manage higher traffic, but its performance actually improves when the AIDT decreases. This surprising result has a simple explanation. Higher rates in game event transmissions result in larger queues at GSSs; these queues contain packets that have not yet been processed. This represenst an insurmountable problem for LL since $q_i(e)$ increases for all clients putting at risk the performance of the system without having any countermeasures. With FILA, instead, a larger queue of game events at a certain GSS represents also a resource. In fact, obsolete game events in queue can be discarded, thus reducing the $q_i(e)$ that a subsequent event e will experience in its traveling towards the various clients i.

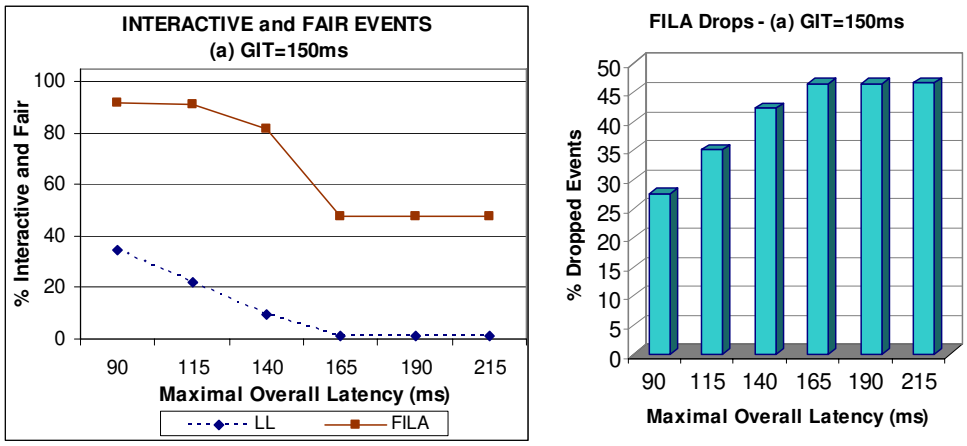


Figure 4.11: Interactivity and fairness improvement (left) and dropped events (right) with GIT=150ms and AIDT=10ms.

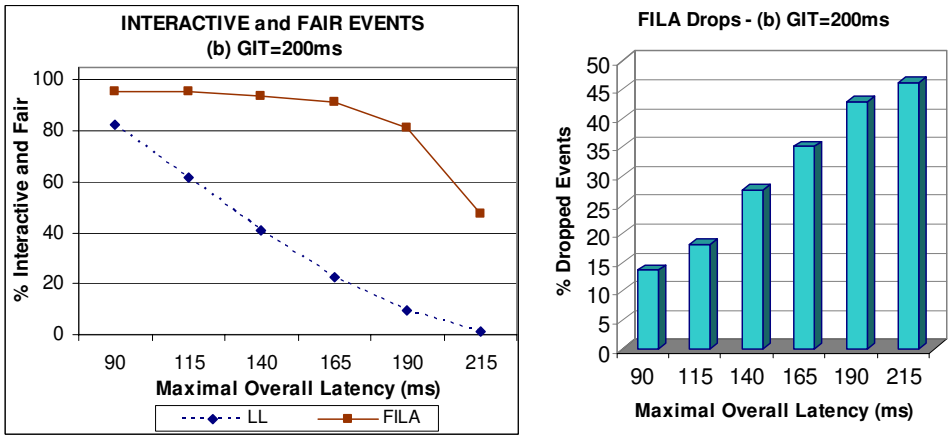


Figure 4.12: Interactivity and fairness improvement (left) and dropped events (right) with GIT=200ms and AIDT=10ms.

As a proof for our rationale, we can notice that the number of obsolete game events dropped by FILA increases when decreasing the AIDT. This is caused by higher *avgOD* values due to the increased traffic, but is also possible thanks to the presence of more game events in queue that FILA can exploit to drop obsolete ones.

Finally, analogously to the scenario with 30ms of AIDT, even when the AIDT is set equal to 20ms or 10ms, the percentage of discarded game events remains still reasonably small.

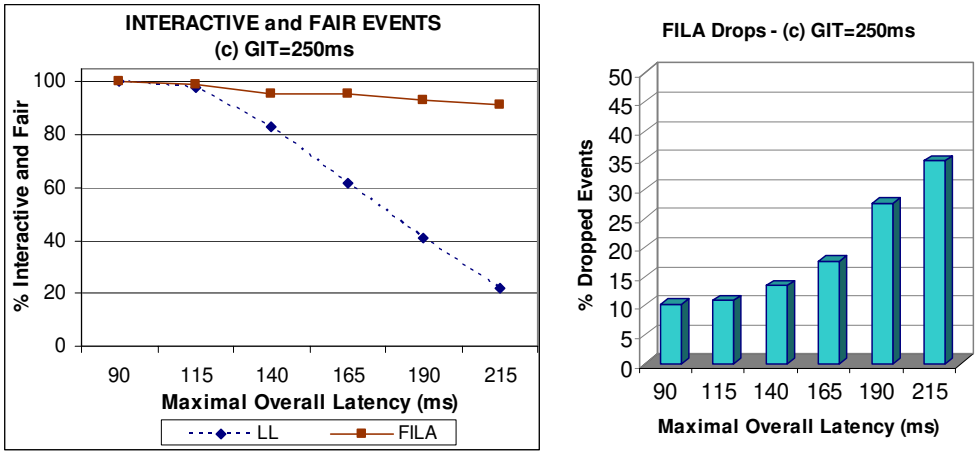


Figure 4.13: Interactivity and fairness improvement (left) and dropped events (right) with GIT=250ms and AIDT=10ms.

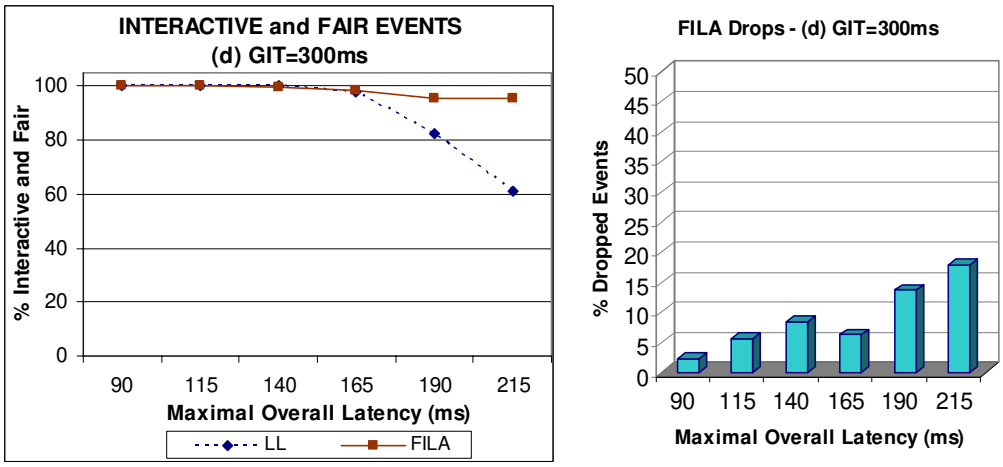


Figure 4.14: Interactivity and fairness improvement (left) and dropped events (right) with GIT=300ms and AIDT=10ms.

CHAPTER 5

Wireless Home Scenario

The market is currently heading toward houses where all the devices (e.g., computers, televisions, phones, intelligent appliances, etc.) will be wirelessly connected to the home network and possibly controlled by a single hub. This convergence point might be represented by the media center which will expand its features becoming, within few years, the engine of the home network and control the whole home connectivity.

In this context, take, for example, a mid-class American household where a family of four people lives: two teenage kids and the hardworking parents. Each family member presumably owns several networked personal portable devices such as PDAs, MP3 players, game consoles and digital cameras; all these being also connected to the home network.

Based on the market trends, we consider that all those devices are wirelessly connected to a media center that controls the in-house media distribution and provides access to the Internet as well as to the cable television and companies providing external services (e.g., the alarm company).

Moreover, we also assume that several family members will be accessing the household network at the same time according to their work or leisure needs. In particular, for the sake of our study, we consider the following family scenario:

i) one teenager is watching a movie, streaming it from the media entertainment center;

ii) the other one is playing with his latest MMOG against a crowd of buddies across the Internet;

iii) the father is having a conversation through an IP based video-chat;

iv) the mother is downloading the last U2 greatest hit compilation from the Apple iTunes music store.

In the above everyday life picture it is worth noticing that each of the aforementioned employed applications features different requirements in terms of network performance, as well as suffer from very specific problems all due to the best effort nature of the Internet transport protocols.

These are as follows:

- **Video-Streaming**. Streaming applications are affected by the jitter phenomenon while are resilient to some packet loss; a network designed mainly for video-streaming should minimize the jitter. Buffering techniques can be applied to minimize the impact of variable network conditions.

- **Video-Chat and Massive Multiplayer Online Games**. Both this applications require a high degree of interactivity, they greatly suffer from delays and packet jitter while may tolerate some packet loss, as we have seen for the case of superseded events.

- **iTunes Music download**. A music download activity is typically performed using TCP, hence this type of application is resilient to jitter and delays but decreases the sending rate in presence of losses: it hence does not tolerate any error losses (losses that do not depend from congestion).

5.1 Queuing Delay

As represented by (2.5), several delay components determine the final delivery time of each game events and a significant one is represented by queuing time. Queues are built up along the path from sender to receiver when the arriving rate of events at a certain node is superior to the serving rate featuring that node. For instance, there could be routers along the path that receives more packets per unit time than the transmitting rate available on the outgoing link on which those packets have to be forwarded. Another

example is represented by Game servers that receive more game events per unit time than the rate at which they are capable of processing them.

However, as recently demonstrated by measurements on a real OC48 link, the capacity of the Internet is generally larger than the aggregate bandwidth utilized by transiting flows [156]. Moreover, more and more providers are offering today *guaranteed high speed connectivity* to home customers [150, 151, 152, 153, 154, 155]. In essence, tools are offered to customers to verify that their connection is actually able to support as much traffic as the declared bandwidth. This implies that the bottleneck of the connection is generally located at the edge of the path connecting sender and receiver.

Focusing on MMOG deployment, we can further support this assumption. In fact, revenues for MMOG providers come from the subscription payments of many satisfied customers. Therefore, every commercial MMOG is generally supported by adequate resources in terms of connectivity speed among, number and capability of their servers [97]. Moreover, we have just demonstrated in Chapter 3 and Chapter 4 how efficient synchronization policies among GSS can be put to good use to improve interactivity and fairness degree.

However, even when the game network platform is able to bring game events to our house with delivery times within the *GIT*, still problems may arise in the last hop, which represents the bottleneck in terms of the available capacity for the connection. In fact, it might be the case when the Access Point (AP) receives packets at higher rates that it can forward them to destination. This can happen for several reasons as, for instance, the fact that the wireless medium allows the transmission of only one packet at a time and is not full-duplex as wired links.

Moreover, interference, errors, fading, and mobility may cause packet losses which are handled by the MAC protocol through local retransmissions. These local retransmissions hide error losses to the TCP and are useful to increment the reliability of the connection. Without them, the TCP would misinterpret error losses as congestion evidences and reduce its sending rate decreasing its performance. On the other hand, retransmissions follows the well known back off mechanism by which an increasing amount of time is

utilized to determine whether a packet has been lost and hence retransmit it. The 802.11 MAC protocol performs up to seven retransmissions of short packets (i.e., RTS/CTS, acks) and four retransmissions of long packets (i.e., data packets) [142]. This means that following packets have to wait in queue until the preceding one or one of its retransmissions finally reaches the receiver and the corresponding ack is successfully sent back.

Finally, the same wireless connection might be shared by several devices and applications that increase the congestion level causing queuing. As it is well known, TCP connections have an aggressive behavior and continuously probe the channel for more bandwidth until queues are fully utilized and overflowed. In presence of persistent TCP connections it is hence very likely to happen that queues are steadily fully utilized, thus periodically slowing down the delivery time of each packet, and deteriorating the performance of time-sensitive applications such as MMOGs.

At the same time large buffers helps TCP-based flows in keeping a high sending rate. This happens for several reasons but the most important ones are: i) the link successive to the buffer remains fully utilized for longer periods of time since there are (almost) always packets in queue, ready to be sent as soon as possible, and ii) traffic bursts can be more easily accommodated thus reducing packet losses and maintaining higher sending rates for longer periods of time. In essence, a tradeoff relationship exists among the per-packet delay and the total goodput achieved. The solution for this tradeoff depends on the queue size and on its utilization.

As we demonstrate in Section 5.4.1, delay increments caused by TCP-based traffic could hit also tens of milliseconds, which represent a huge waste of time when trying to deliver game events within a *GIT* of 150ms.

5.2 Proposed Solutions

To solve the aforementioned problem we propose different possible solutions. In particular, we may divide them into two main classes depending on the correspondent

networking layer where they are implemented: transport layer solutions and MAC layer solutions.

The former mainly regards exploiting some of the existing features of regular TCP or employing an alternative TCP version. The solutions belonging to this class does not necessary rely on the fact that the last link will be a wireless one. The latter, instead, involves modifications of the 802.11 MAC layer and thereby it is specifically intended for the wireless media [166].

For all the proposed solutions we investigate both their efficiency and factual deployability to expose pros and cons and conclude with a winner.

5.2.1 IEEE 802.11 Parameters Setting

The first proposal regards the utilization of more appropriate setting for parameters of the IEEE 802.11 MAC protocol. Parameters such as maximum number of retransmissions and buffer size were, in fact, decided in a period when the TCP-based traffic was largely predominant in the Internet. The main concerns for designers were hence reliability and high throughputs.

Nowadays, UDP-based real time applications are becoming more and more popular hence demanding for low delays in packet delivery. This kind of applications are resilient to some packet loss whilst cannot tolerate delays in packets delivery. For this reason, it is preferable to drop a packet than to waist time in retransmissions.

This obviously contradicts the initial assumption that reliability is the most important issue over wireless links. Therefore, 802.11 parameters should be modified to make it more sensitive towards real-time application needs. In particular, the number of local retransmissions could be diminished in order to find an efficient solution between reliability and low delays in packet delivery.

In the same way, large buffer sizes at the AP help TCP connection to maintain large sending rate for longer periods and diminish the impact of burst traffic. On the other hand, to a larger buffer corresponds a longer queuing time experienced when the buffer is full, thus jeopardizing the performance achieved by time-sensitive applications. By

adjusting the buffer size to an appropriate value we can again try to find an optimal compromise between the needs of TCP-based applications and real-time applications.

5.2.2 TCP Vegas

TCP Vegas embodies one of the most cited alternatives to regular TCP New Reno in scientific papers. Its main point of interest is represented by the fact that TCP Vegas tries to avoid congestion instead of blindly increase its sending window until a packet loss occurs; it hence perfectly fits our needs for low buffer utilization.

```
for every RTT
{
    if (cwnd/RTT_min - cwnd/RTT) < α   then cwnd++
    if (cwnd/RTT_min - cwnd/RTT) > β   then cwnd--
}

for every loss
    cwnd := cwnd/2
```

Figure 5.1: Pseudocode of TCP Vegas congestion control.

Indeed, while TCP New Reno utilizes packet loss to determine network congestion, TCP Vegas is sensitive to end-to-end queuing delay. With TCP Vegas the sender monitors the difference between its expected rate and the actually achieved one. The difference is compared to a couple of parameters, namely α and β, to determine whether the congestion window has to be incremented or decremented (by 1) in the next RTT [143, 144]. The parameter α and β determine the amount of buffer each flow is permitted to occupy.

More in detail, each TCP Vegas source estimates the number of its own packets in the buffer by monitoring the RTT. It then accordingly adjusts its congestion window to maintain its estimated rate between the two predetermined parameters α and β. If the difference between the expected rate and the achieved one is smaller than α, then the congestion window is increased by 1. If this difference surpasses β, the congestion

window is decremented by 1. Finally, if the difference is comprised between α and β then the sender rate is in its stability region and no particular operation is performed.

The pseudocode of the congestion avoidance algorithm of TCP Vegas is reported in Fig. 5.1 where cwnd is the congestion window. The pseudocode also shows that, in case of a packet loss, the congestion window is halved.

If the buffer at the bottleneck is large enough then TCP Vegas reaches equilibrium. In this case, TCP Vegas flows should experience zero packet losses, a stable congestion window, and a queue which is proportional to the number of TCP Vegas flows present. Otherwise, it oscillates like TCP New Reno flows. Finally, TCP Vegas has been proven to fairly share the link with other TCP Vegas flows but it behaves too conservatively in presence of regular TCP flows (i.e., New Reno, Sack). Regular TCP, in fact, fully exploits the available buffer space and TCP Vegas interprets the consequent RTT trend as an indicator of excessive congestion, thus progressively reducing its sending rate to very low values [148, 196].

TCP Vegas represents a very appealing transport protocol with interesting features such as the possibility to have some control on the amount of buffer utilized. This, in fact, results a fundamental property when having to efficiently deal with contemporary real time traffic. On the other hand, the dramatic efficiency decrease experienced when competing with regular TCP traffic makes unfeasible its factual deployment.

5.2.3 Limited Advertised Window

We are aiming at finding the best solution to the tradeoff relationship existing between TCP throughput and real time application delays. Moreover, the two types of traffic should be able to coexist without interfering each other and the employed solution should be easily and factually deployable.

Starting from the last point, i.e. the deployability, it is evident how a technique that would exploit existing features of the already utilized protocols could be easily implemented in a real scenario. Moreover, we deem that an optimal tradeoff between throughput and low delays could be achieved by maintaining the sending rate of the TCP

flows high enough to efficiently utilize the available bandwidth but, at the same time, limited in its growth so as to not utilize buffers.

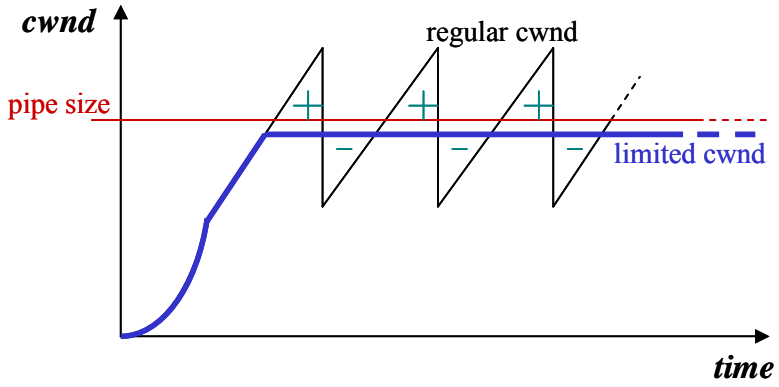


Figure 5.2: Comparison between regular and limited sending windows.

In this way, in fact, the throughput is maximized by the absence of packet losses which would halve the congestion window, while the delay is minimized by the absence of queues.

To better understand how limiting the congestion window could guarantee the same or even a higher throughput with respect to utilizing regular TCP, we show in Fig. 5.2 a typical saw tooth shaped sending window of a regular TCP and overlap it with a limited one. As it is evident, the latter is more stable since it does not use the buffer at the bottleneck link and consequently experiences no losses. The minus signs in the chart represent situations in which the regular congestion windows provides TCP with a sending rate which is inferior to that guaranteed by the limited congestion window, while the plus signs represent the inverse situation (generally accompanied by having packets queuing on the buffer corresponding to the bottleneck link). If the upper bound for the congestion window is appropriately chosen, the balance between the plus and minus signs will guarantee to the limited congestion window an equal or even superior final throughput with respect to the regular congestion window, whilst avoiding queuing delays.

To achieve this desirable result we need first to address two important issues: how to determine an appropriate upper bound and how to apply it in practice to the sending window.

Regarding the first point, the most appropriate formula can be derived from the two main goals we want to achieve: i) full utilization of the available bandwidth and ii) no queue delays. Real time traffic generally exploits UDP and this transport protocol has no congestion control mechanism. Some smart applications, however, implements some sort of congestion control at the application layer [158, 159]. In any case, to avoid queue delays, the aggregate bandwidth utilized by TCP flows cannot exceed the total capacity of the bottleneck link diminished by the portion of the channel occupied by the concurrent real time traffic.

In essence, the maximum sending rate for each TCP flows at time *t*, namely *TCPubrate(t)*, is represented by:

$$TCPubrate(t) = \frac{(C - UDPtraffic(t))}{\#TCPflows(t)} \tag{5.1}$$

where *UDPtraffic(t)* represents the amount of bandwidth occupied by UDP-based traffic at time *t*, *#TCPflows(t)* is the concurrent number of TCP flows, and *C* corresponds to the capacity of the bottleneck link.

The second issue we need to address is how to practically employ this formula in order to have it working in a real scenario. This means i) determining an effectively deployable way to utilize it with the current state of the art of the Internet, ii) identifying the location for its implementation, and iii) proposing a method to compute the value of the various variables.

To solve the first issue, we have to limit the required scope of intervention since modifying the whole Internet in order to run our scheme would obviously not be a feasible option. Moreover, it would definitely be better if we could make good use of some feature already present in the regular TCP. For this reason we propose to exploit the existing advertised window to limit the TCP sending rate. In fact, the actual sending

window is determined as the minimum between the congestion window and the advertised window. The advertised window perfectly embodies an upper bound to the congestion window and is already implemented in all TCP versions. By appropriately modifying it, we can achieve both efficiency and low delays.

The advertised window is generally determined at the receiver; however, this could not represent the most suitable place for the modification we need to perform. Indeed, to determine the most appropriate value for the advertised window, we need a comprehensive knowledge about all the flows that are transiting through the bottleneck (i.e., the last hop links). Since all the flows have to pass through the AP, this represents the node where we could be able to implement our scheme. The AP may also coincide with the Media Center in a wireless home and the mechanism can take advantage of this. In particular, by spoofing the transiting traffic at the AP and/or utilizing the information hold by the Media Center, we can obtain all the information required. In any commercial Operating System it is possible to know which kind of connection is in use and which its nominal speed is just by looking at the status of the network interface. Through spoofing the channel or exploiting information known at the Media Center we can also infer the number of active TCP connections and the aggregate amount of current UDP traffic. The AP can hence easily compute the best *TCPubrate(t)* utilizing (5.1) and modify the advertised window included on the transiting acks accordingly.

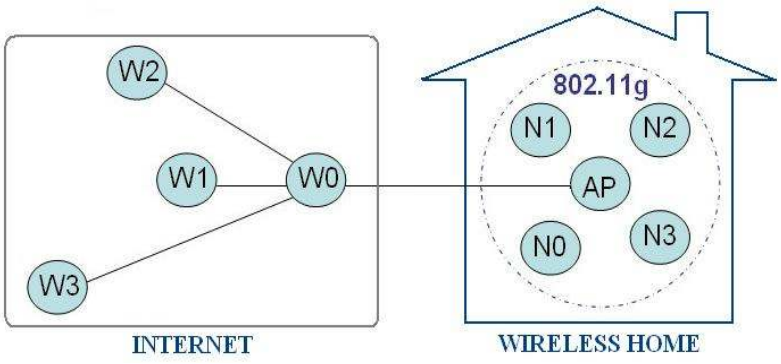


Figure 5.3: Simulated topology.

Table 5.1: Simulation configuration of the wired links.

Node 1	Node 2	Physical Latency	Link Capacity	Queue Size
W1	W0	10ms	100Mbps	140pkts
W2	W0	20ms	100Mbps	140pkts
W3	W0	30ms	100Mbps	140pkts
W0	AP	10ms	100Mbps	140pkts

5.3 Simulation Assessment

In order to analyze in depth our scenario, we have utilized the well known NS-2 network simulator (version ns-2.28) [1]. Our adopted configuration of the nodes and links can be easily visualized by the means of Fig. 5.3. In particular, the house environment is represented by four mobile nodes named N1, N2, N3 and N4, and the Media-Center that incorporates also the AP. The MAC layer parameters have been set accordingly to the IEEE802.11g standard. The simulation outcomes showed us that we were able to reach a maximum achievable bandwidth of circa 20 Mbps. This represents a reasonable value over the declared 54 Mbps even in the real world [174].

Focusing on the wired links, their one-way delays and capacities have been configured as listed in Table 5.1, while their queue sizes have been set equal to 140 packets. This value comes out by multiplying the longest RTT with the smallest link capacity on the path (i.e. the bottleneck) which is represented by the 20Mbps effectively available over the wireless link.

As shown in Table 5.2 and Fig 5.4, several kinds of applications have been run over this network topology. In order to uplift the trustworthiness degree of the simulations, we have exploited real trace files for the video-stream and for the video-chat. Specifically, adopted trace files correspond respectively to high quality MPEG4 *Star Wars IV* for the movie, and two VBR H.263 Lecture Room-Cam for the video-chat, as can be found in [145].

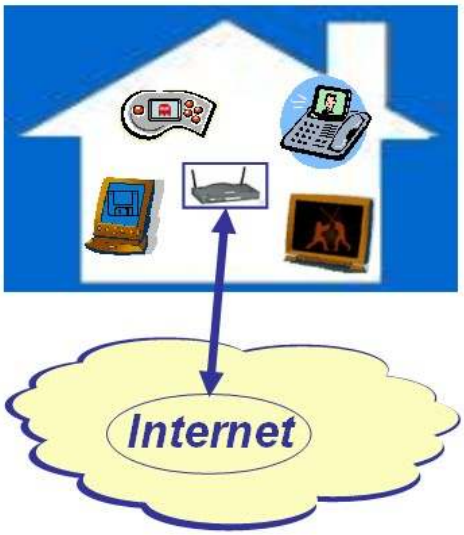


Figure 5.4: In-home wireless scenario.

Table 5.2: Simulated application flows.

From	To	Flow Type	Transp. Prot.	Start	End
AP	N0	Video-Stream	UDP	0s	180s
W1	N1	Online Game	UDP	45s	180s
N1	W1	Online Game	UDP	46s	180s
W2	N2	Video-Chat	UDP	90s	180s
N2	W2	Video-Chat	UDP	91s	180s
W3	N3	FTP	TCP	135s	180s

The parameters characterizing the game-generated traffic have been chosen following the directions provided by scientific literature in this field. Indeed, we can assume that the user in the house is engaged in one of the very popular first person shooter games, e.g. Quake Counter Strike, with other ~25 players, geographically away from each other and connected through the Internet. Hence, to model the traffic generated by this kind of MMOG (packet size and interarrival time), we can use some of the approximations suggested in [146], which are based on real game platform measurements.

In particular, in our simulation game events have been generated at client side every 60ms; while the server was transmitting game state updates every 50ms toward the client.

Moreover, packet size has been set to 42Bytes and 200Bytes, respectively for client and server generated game packets.

Simulation experiments have been replicated to examine the effects generated by differently setting some of the parameters involved in the scenario. Table 5.3 lists all the variable parameters in the simulations; each combination of their possible values has been simulated. In particular we have tried several values for long packet retransmissions going from the regular 4 down to 1. The distance between the AP and the mobile device was either 5 or 10m which represent two realistic values for a normal house. The buffer size at the AP was set either equal to 50 or to 100 packets, as these are the two most common values in commerce.

In particular, it is worth to conclude this discussion by mentioning our experimental choices with respect to the Shadowing Model, which is a realistic and widely utilized signal fading model available in NS-2. We followed the directions provided by the official NS-2 manual to represent a home environment partitioned into several rooms. Specifically, in our simulations, the path loss exponent of the Shadowing Model was always set equal to 4, while the shadowing deviation had alternatively the value of 7 and 9. Transmitted signal attenuation grows with the increase of these parameters; we hence expect to face higher percentage of packet losses over the wireless media when setting the shadowing deviation to 9.

However, where not differently stated, simulations were run utilizing some realistic default values for the simulative parameters listed in Table 5.3. In particular, we had:

- buffer size at the AP = 100 packets;
- distance between the AP and the mobile device = 10m;
- max number of MAC retransmissions for long packets = 4;
- shadowing deviation of the shadowing model = 7.

5.4 Experimental Results

We present here the most relevant results from the extensive set of simulations we have run. In particular, we first demonstrate how concurrent TCP-based traffic can affect the performance of real time applications. We then compare the outcome with those of our proposed solutions.

5.4.1 FTP Impact on Real-Time Entertainment Applications

We have intentionally started the various application flows one after the other in order to notice the progressive impact of the successively incoming and overlapping traffic on the preexisting ones. In particular, we expect to witness increasing delays and jitter in the arrival time of packets as we augment the traffic level.

However, the bandwidth requirement of the first starting applications in our scenario is well below the effectively available capacity of the IEEE802.11g wireless media. We have to wait until the FTP flow takes action, quickly saturating the channel and the queues along the path with its packets, before being able to clearly detect significant variations in the delays and jitter experienced by the various real time flows. This phenomenon is evident in Fig. 5.5 and Fig 5.6 where the mobile device was located at 10 m from a standard IEEE 802.11g AP.

Table 5.3: Changing parameters in the simulated configurations.

Parameter	Values	Comment
MAC data retransmissions	1, 2, 3, 4	default value = 4
shadowing deviation	7, 9	medium, high
user-AP distance (m)	5, 10	same room, different room
MAC queue size (pkts)	50, 100	common default values

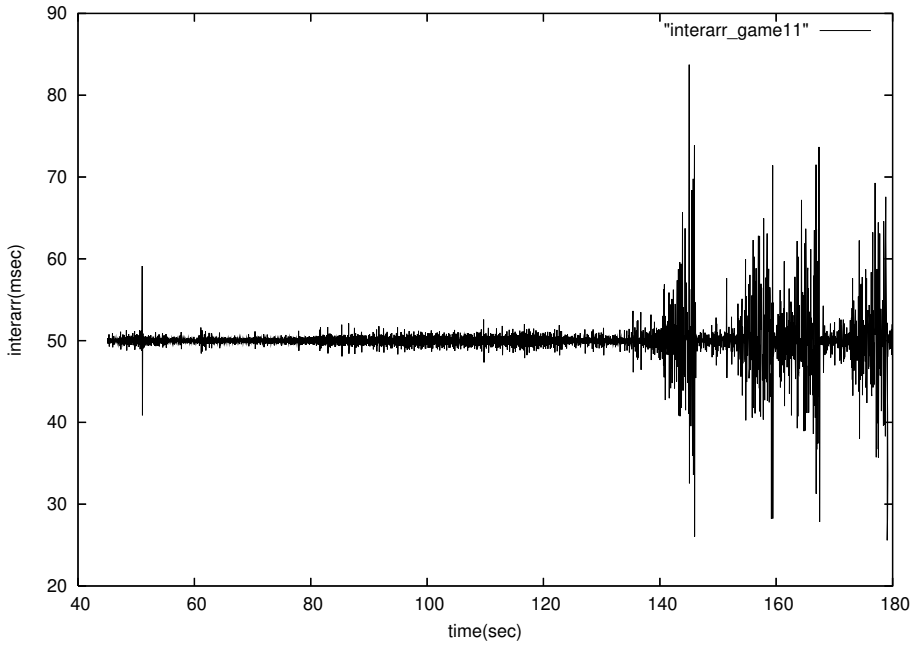


Figure 5.5: Example of online gaming interarrival delays.

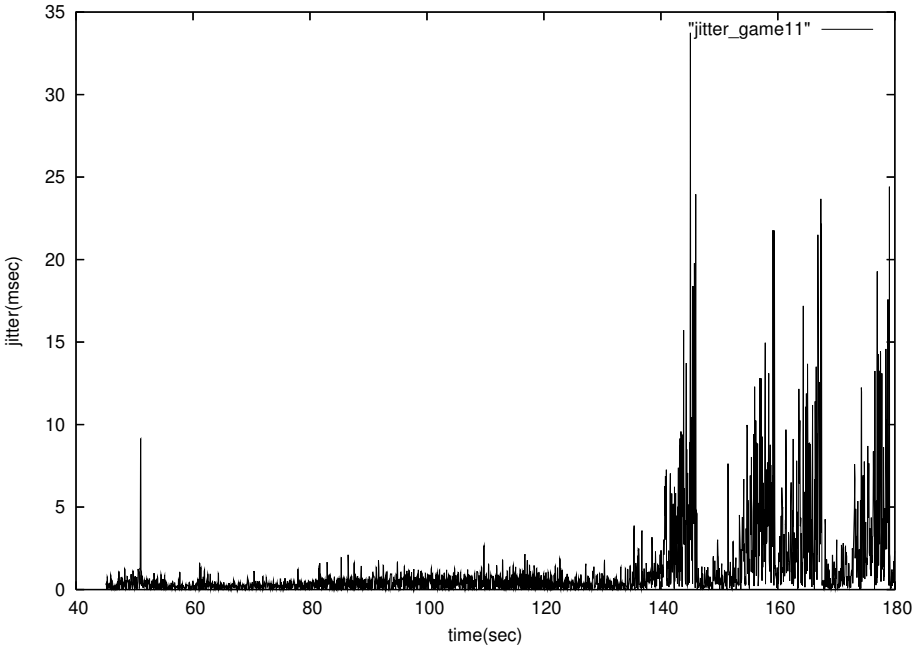


Figure 5.6: Example of online gaming jitter.

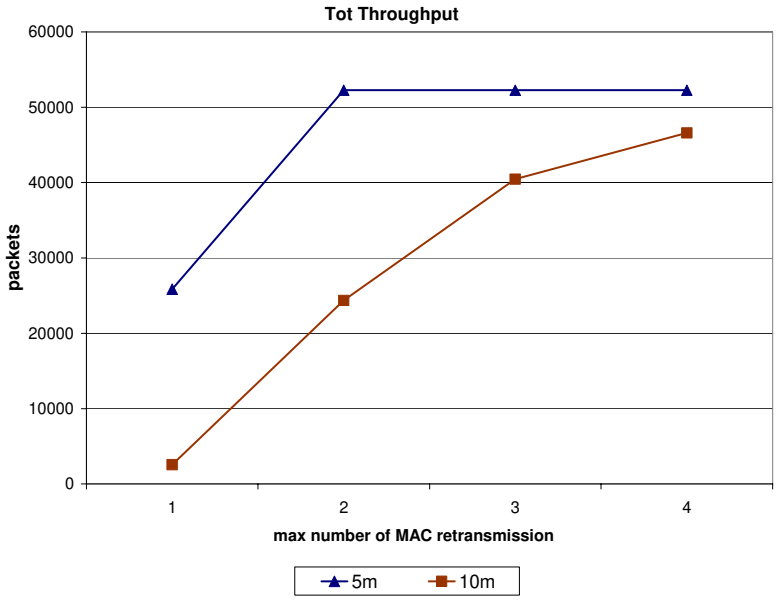


Figure 5.7: FTP total throughput with different user-AP distances; shadowing deviation = 9, MAC queue size = 50pkts.

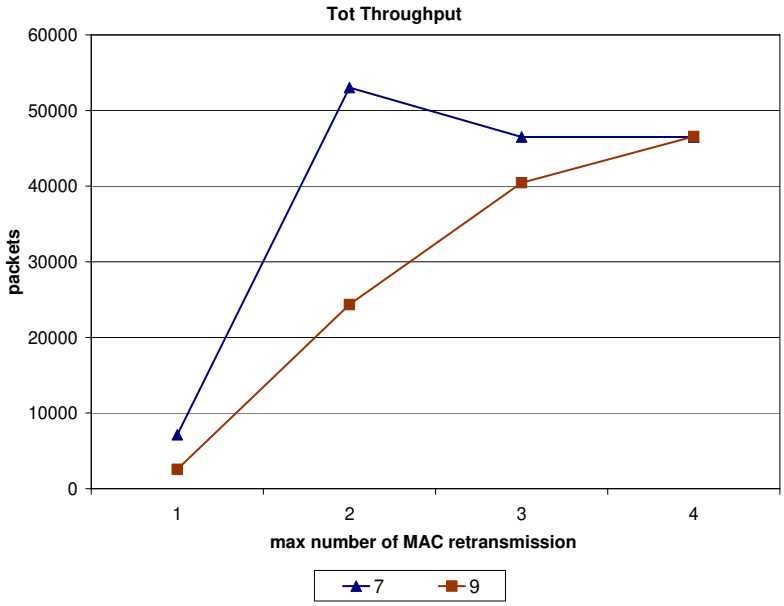


Figure 5.8: FTP total throughput with different shadowing deviation values; user-AP distance = 10m, MAC queue size = 50pkts.

5.4.2 Shadowing and Distance Impact to TCP Throughput

The distance between the AP and the mobile device represents an important factor in determining the transmission quality, especially in a partitioned environment as a house. In fact, Fig. 5.7 shows that, positioning the mobile device at a distance of 5 m from the AP, the maximum throughput achievable by the FTP application is already obtained when utilizing only two retransmissions at the MAC layer. The quality of the transmission signal also depends on the impediments that it may encounter along its path between the AP and the mobile device.

We compare now a house environment constituted by medium level partitions (parameter set to 7), with another one having more unfavorable partitions to wireless transmissions (parameter set to 9). Fig. 5.8 confirms our expectations by showing the throughput gain achievable in the first case. However, Fig. 5.8 also reports a case where having just two retransmissions at the MAC layer produces a higher total throughput than utilizing a greater number.

This apparent paradox has instead a rational explanation. First, we should remember that IEEE802.11 has a limited buffer for transmissions and retransmissions. Second, the game and the video-chat applications generate some reverse traffic that shares path and networking resources with the acknowledgment (ack) packets generated by TCP as transport protocol under the FTP application. When the channel is fully utilized, some acks get lost and may cause timeouts.

As a confirmation of this, Fig. 5.9 and Fig. 5.10 refer to the same configuration of Fig. 5.8 and present the congestion window, the slow start threshold and the bandwidth delay product of the underlying TCP flow when the maximum number of retransmissions at the MAC layer was set equal to 2 and 3, respectively. The two charts show a higher frequency of timeouts when a higher number of retransmissions at the MAC layer helps the traffic to reach a higher level.

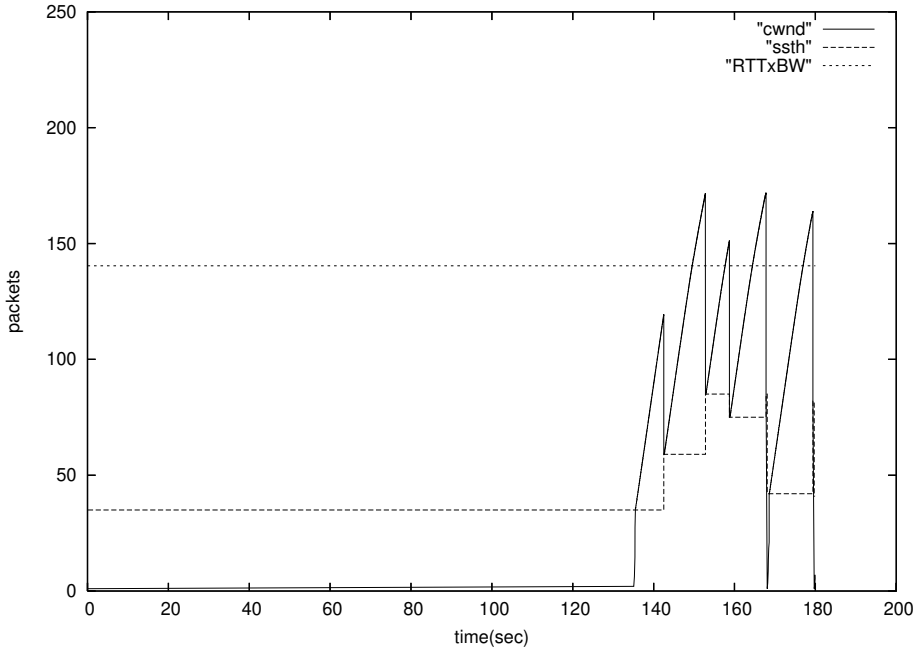


Figure 5.9: Example of TCP congestion window; max MAC retransmissions = 2.

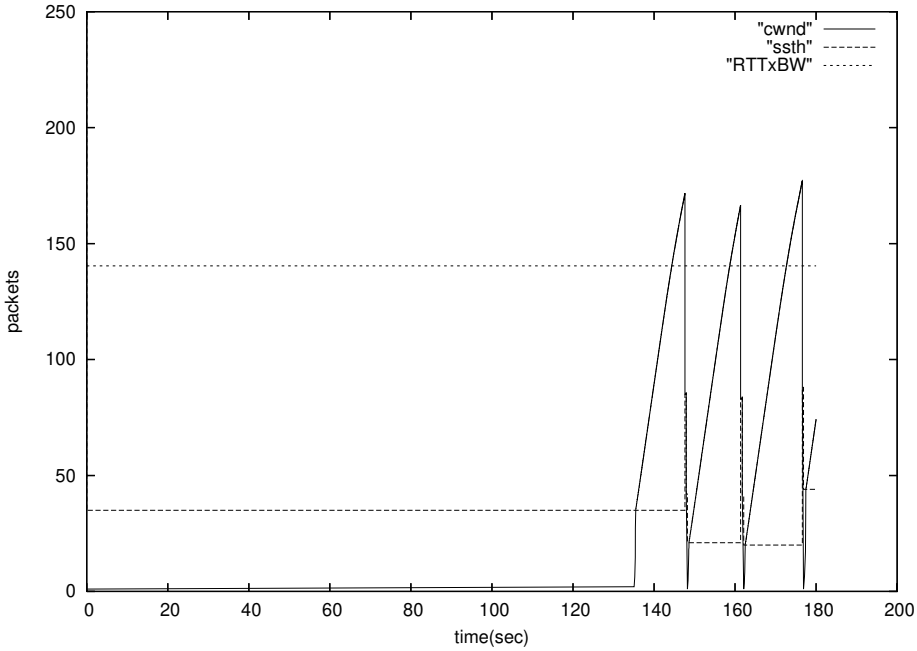


Figure 5.10: Example of TCP congestion window; max MAC retransmissions = 3.

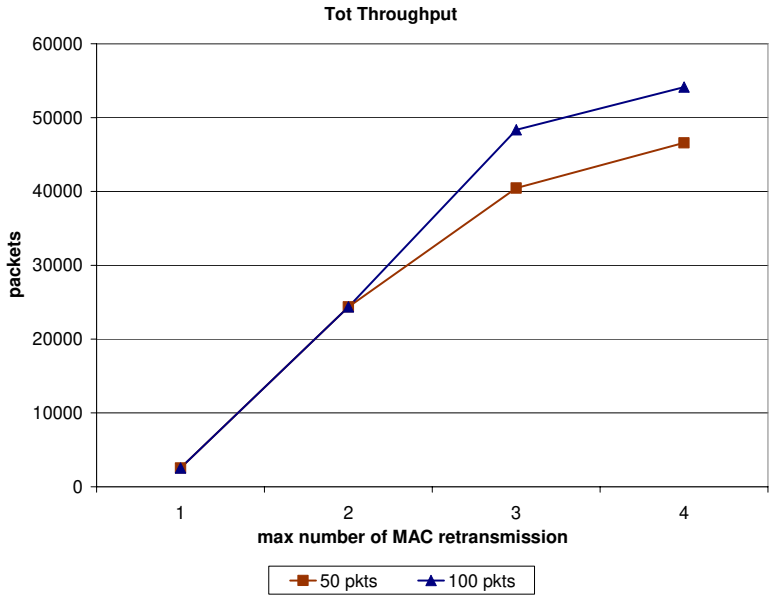


Figure 5.11: FTP total throughput with different MAC queue sizes; user-AP distance = 10m, shadowing deviation = 9.

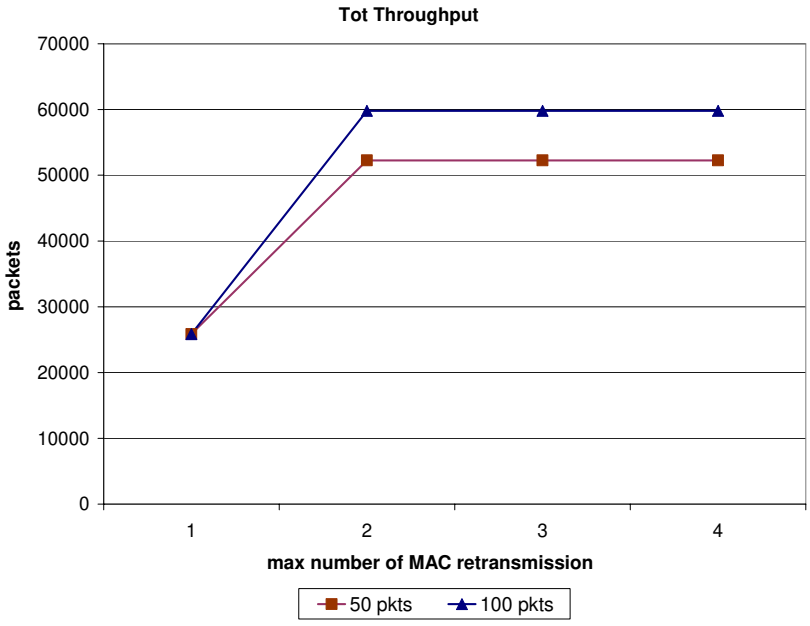


Figure 5.12: FTP total throughput with different MAC queue sizes; user-AP distance = 5m, shadowing deviation = 9.

Table 5.4: Gaming flow jitter statistics; max MAC retransmissions = 4, shadowing deviation = 9.

CONSIDERED PERIOD = [0 – 180]s		
Jitter	**50 pkts**	**100 pkts**
maximum (ms)	33.740	108.36
average (ms)	1.306	2.041
variance	7.360	22.079
pkts received	2658	2658
CONSIDERED PERIOD = [135 – 180]s		
Jitter	**50 pkts**	**100 pkts**
maximum (ms)	33.740	108.36
average (ms)	3.056	5.229
variance	16.665	49.470
pkts received	899	899

Table 5.5: Gaming flow jitter statistics; max MAC retransmissions = 3, shadowing deviation = 9.

CONSIDERED PERIOD = [0 – 180]s		
Jitter	**50 pkts**	**100 pkts**
maximum (ms)	31.091	44.632
average (ms)	1.045	1.566
variance	4.833	11.034
pkts received	2654	2655
CONSIDERED PERIOD = [135 – 180]s		
Jitter	**50 pkts**	**100 pkts**
maximum (ms)	31.091	44.632
average (ms)	2.292	3.835
variance	11.502	24.431
pkts received	896	897

5.4.3 Appropriately Setting MAC Layer Parameters

Finally we intend to highlight the impact of having different queue sizes and maximum number of retransmissions at the MAC layer on the performance of the various types of

traffic present in our considered scenario. Starting with the first parameter, Fig. 5.11 and Fig. 5.12 confirm that having larger queue size helps TCP in achieving higher throughputs. However, there is no difference in the achieved throughput when wireless losses, not recovered via MAC retransmissions, are frequent enough to keep the TCP transmission rates low and hence never have the possibility to utilize more than 50 queue slots.

On the other hand, having large queues along the path may augment the total delay time experienced by packets. In fact, each packet waits in queue for a time which proportionally grows with the number of anterior packets already present in the same queue at its arrival. In case of intense traffic, queues tend to be congested and hence queuing delays may become a significant component of the global delays experienced by each packet. At the same time, having larger queue size on a link also spreads the range of possible queuing delays that packets may experience while traveling on that link (depending on the filling level of the queue). The resulting jitter strongly impact on the un performance achieved by real time applications and, in particular, by highly interactive applications as video-chats and MMOGs. Statistics of the aforementioned game flow jitter permit a clearer understanding of the performance disparity generated by diverse queue sizes. In particular, the upper part of Table 5.4 refers to the whole simulated duration of the MMOG application, while the rest of it considers only the period when the FTP application is running (from second 135 to second 180).

In any case, the worst jitter is experienced when queues are steadily filled up by the FTP flow. To limit this problem we should find a way to barter part of the FTP throughput with lower queuing delays. This can be attained also by simply reducing the maximum number of retransmission at the MAC layer as it is evident by comparing Table 5.5 with Table 5.4

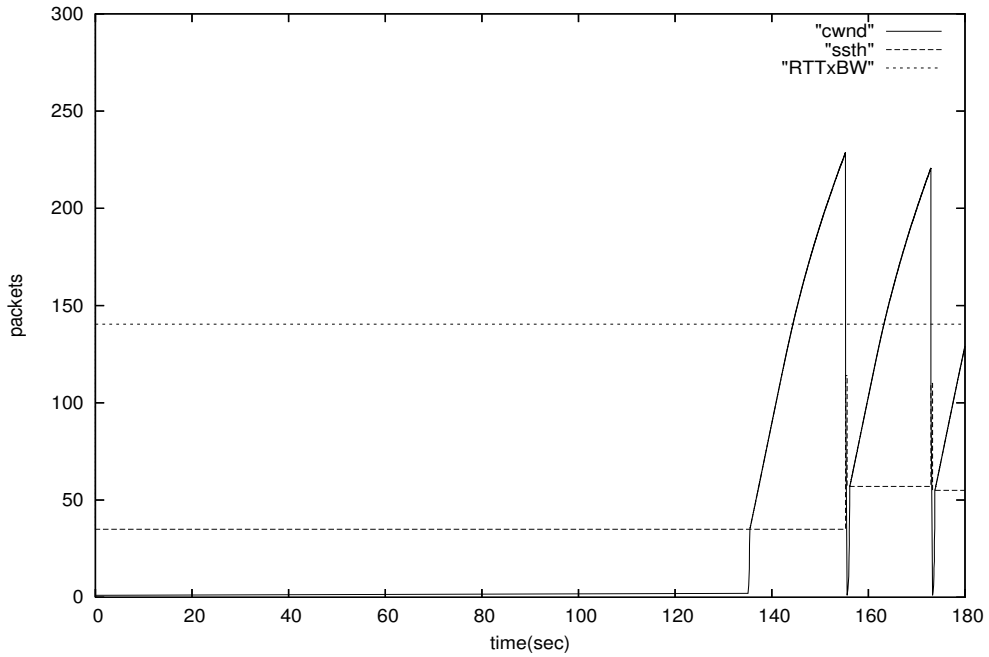


Figure 5.13: TCP congestion window; MAC max retransmissions = 4, buffer size = 50 packets.

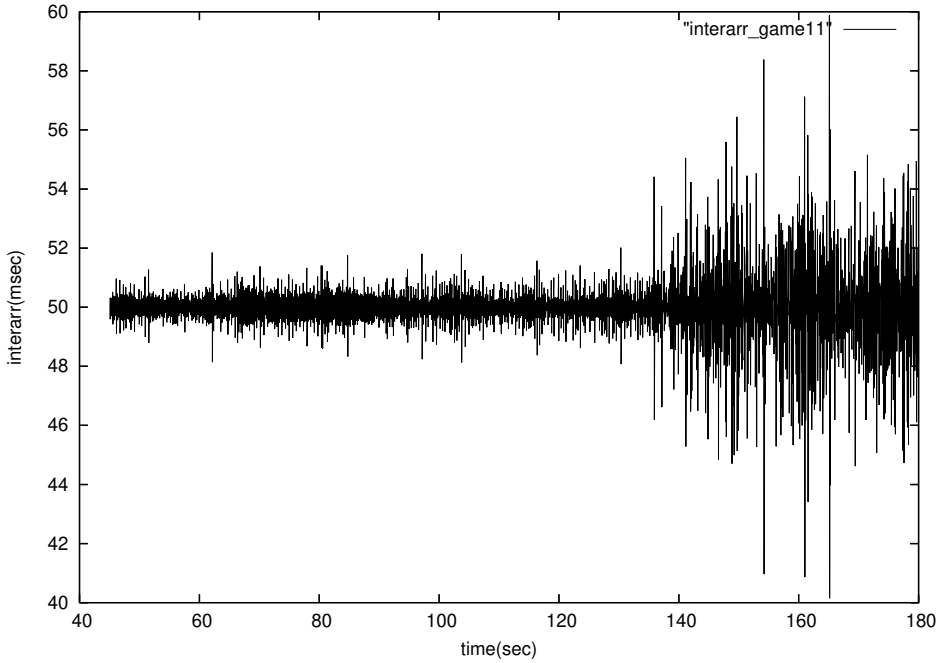


Figure 5.14: Online game interarrival time; MAC max retransmissions = 4, buffer size = 50 packets.

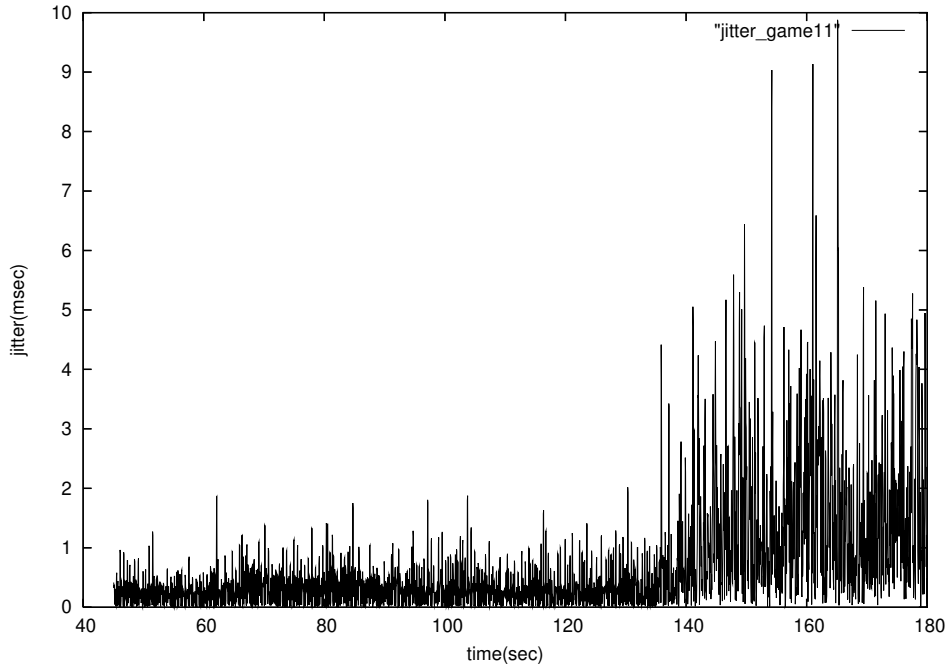


Figure 5.15: Online game jitter; MAC max retransmissions = 4, buffer size = 50 packets.

Even better jitter average and variance can be gained further diminishing the maximum number of MAC retransmission to 2. However, we advice against this choice, unless placing the device hosting the FTP application closer to the AP or in a house with a better shadowing deviation of the transmitted signal. Otherwise, the FTP throughput descends significantly as can be observed in Fig. 5.7, Fig. 5.8, Fig. 5.11, and Fig. 5.12.

Summarizing, we can say that a more appropriate configuration of the IEEE802.11g than the traditional one would probably make use of a maximum number of 3 retransmissions, thus guaranteeing a high FTP throughput whilst maintaining a low per-packet delay and jitter. Moreover, when a unique queue is maintained for all the traffic flows, a small size (50 packets) should be preferred.

In particular, for a scenario where the shadowing deviation is set to 7 and we have 10m of distance between the AP and the mobile device, this configuration of the MAC layer parameters allows a TCP goodput of 55371 packets during the 45 seconds when the

FTP application was running. The corresponding congestion window, slow start threshold and bandwidth-RTT product is shown in Fig. 5.13. The interarrival packet time and the jitter for the online game flow traveling from the server to the client are reported in Fig. 5.14 and Fig. 5.15, respectively.

5.4.4 Utilizing TCP Vegas in Place of TCP New Reno

The ability of TCP Vegas in detecting queues and anticipating their further growth is evident in Fig. 5.16, which shows the congestion window when parameters are set as follows: $\alpha = 1$, $\beta = 3$, $\gamma = 2$. This parameter setting corresponds to utilize a very small amount of buffer at the bottleneck. Consequently, both the queuing time and the achieved goodput will be reduced. The congestion window, in fact, results evidently limited by TCP Vegas algorithm and reaches very low values. The final goodput is obviously affected as well: 43950 packets acknowledged in 45 s instead of 46580 and 54137 packets for a buffer size of 50 and 100 packets, respectively, for the regular configuration with TCP New Reno (as seen in Fig. 5.11).

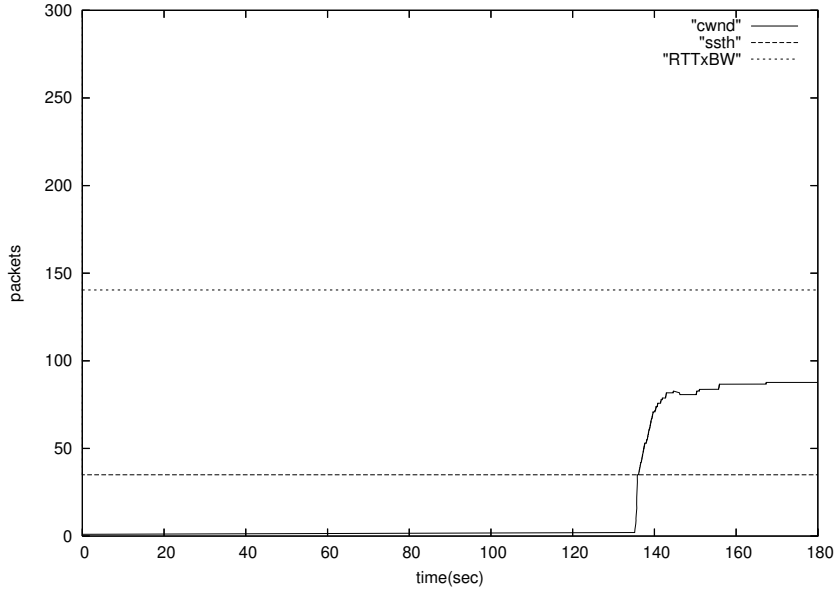


Figure 5.16: TCP Vegas congestion window; $\alpha = 1$, $\beta = 3$, $\gamma = 2$.

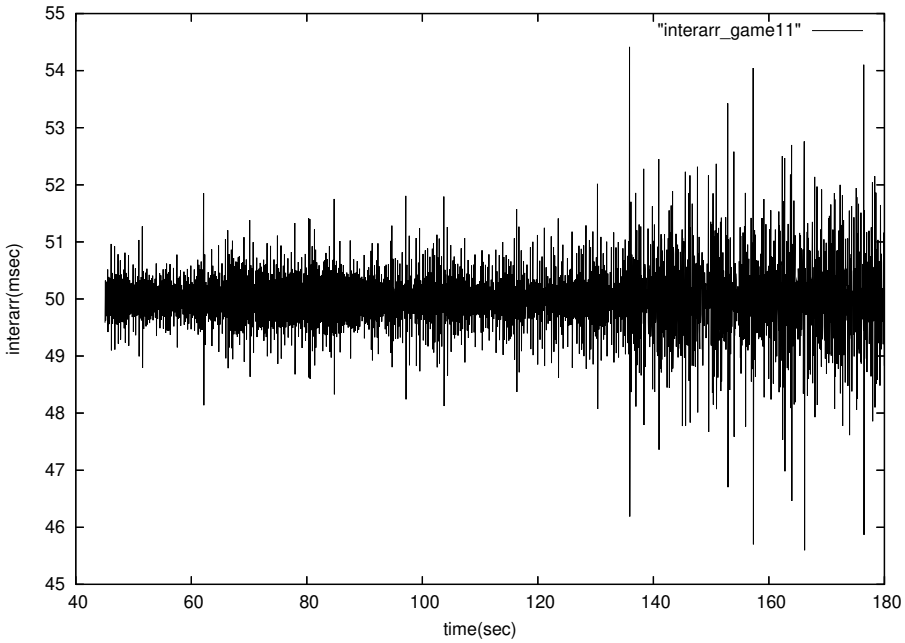


Figure 5.17: Online game interarrival time with concurrent TCP Vegas; $\alpha = 1$, $\beta = 3$, $\gamma = 2$.

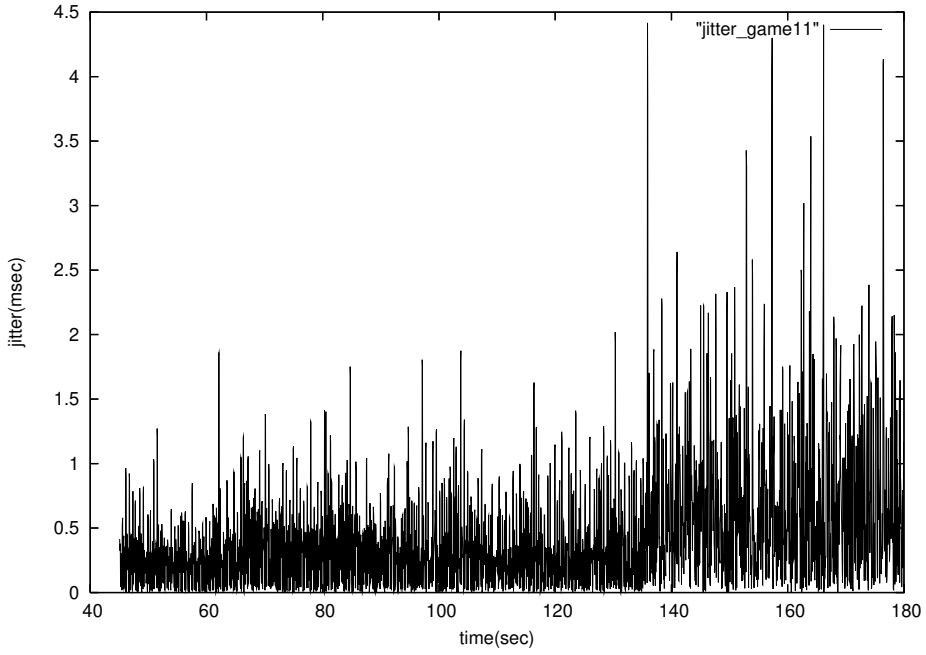


Figure 5.18: Online game jitter with concurrent TCP Vegas; $\alpha = 1$, $\beta = 3$, $\gamma = 2$.

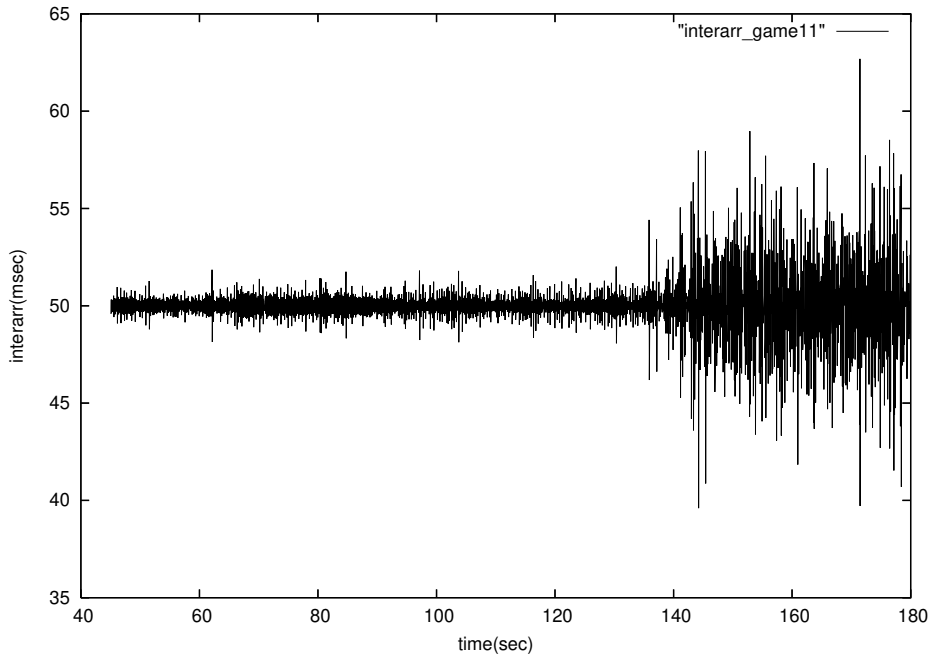


Figure 5.19: Online game interarrival time with concurrent TCP Vegas: $\alpha = 5$, $\beta = 10$, $\gamma = 8$.

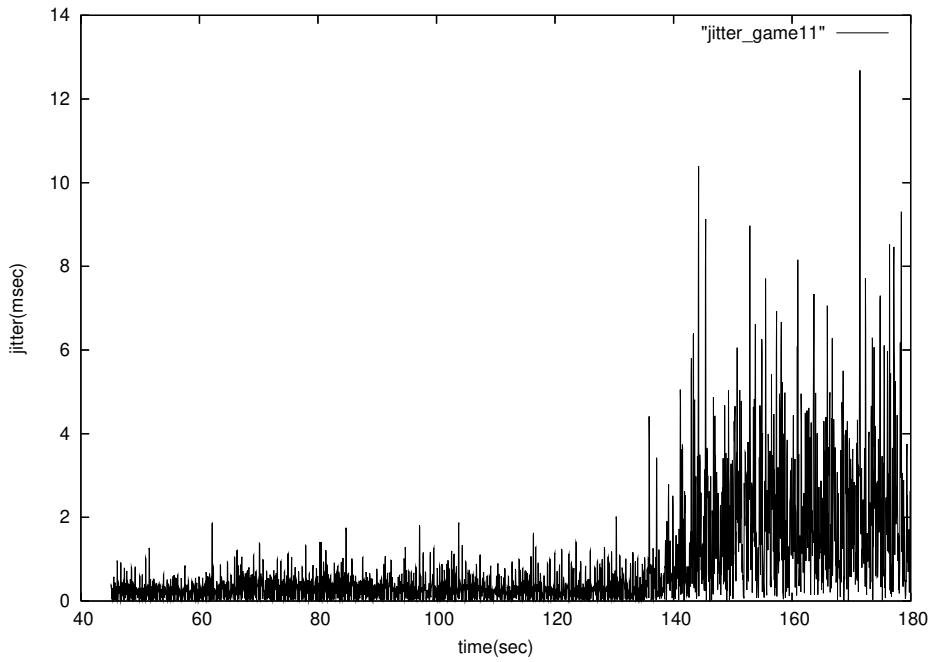


Figure 5.20: Online game jitter with concurrent TCP Vegas; $\alpha = 5$, $\beta = 10$, $\gamma = 8$.

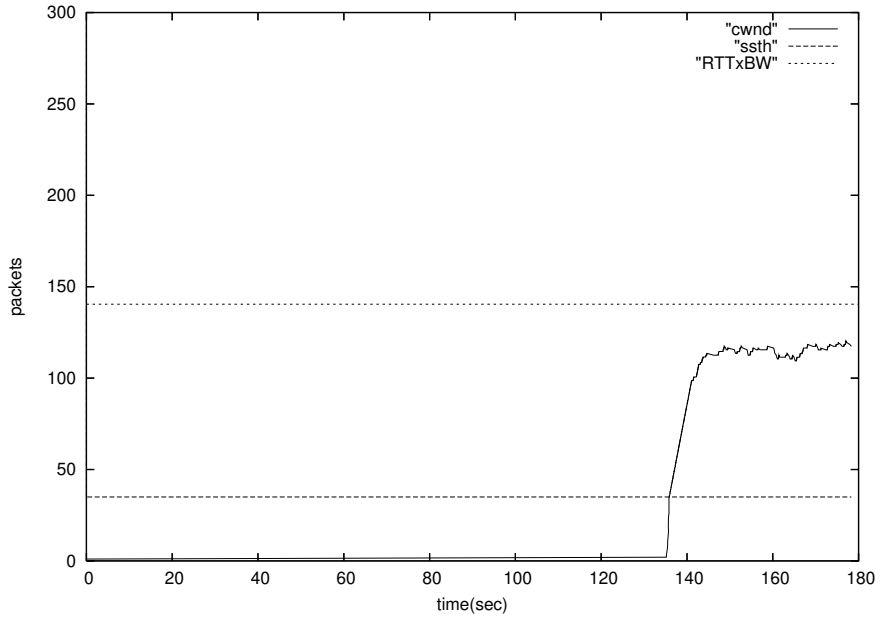


Figure 5.21: TCP Vegas congestion window; $\alpha = 1$, $\beta = 3$, $\gamma = 2$.

On the other hand, having no packets in queue helps real time traffic in reducing its per-packet delay. In particular, in Fig. 5.17 and Fig. 5.18 we show the packet interarrival time and the jitter for the online game flow going from the server to the client. The tremendous reduction for these values results evident from the charts: the jitter, for instance, never reaches 4.5 ms.

With different values of its parameters, TCP Vegas becomes able to make use of more buffer space at the AP. This results in higher delays but, at the same time, higher goodputs. Indeed, the tradeoff relationship between the per-packet delay and the total goodput highlighted in Section 5.1 is perfectly embodied in TCP Vegas parameters. By setting the queue target size, those parameters could be seen as knobs able to move that tradeoff towards one direction or the other. As an example, if we set $\alpha = 5$, $\beta = 10$, and $\gamma = 8$, then the total goodput raises to 59999 packets acknowledged in 45s. On the other hand, the packet interarrival time and the jitter for the online game flow going from the server to the client result increased as shown by Fig. 5.19 and Fig. 5.20.

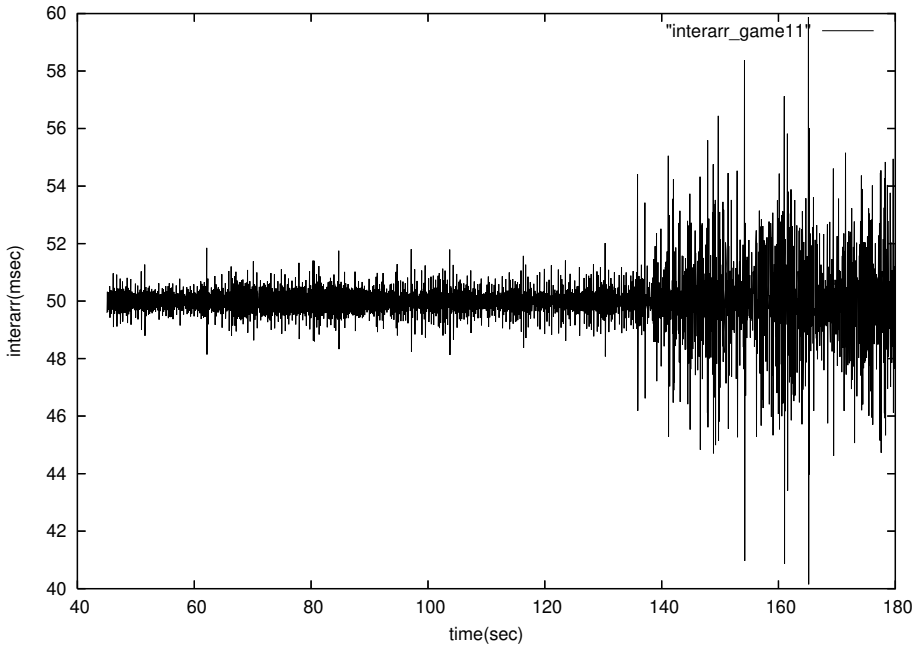


Figure 5.22: Online game interarrival time with concurrent TCP Vegas: $\alpha = 3$, $\beta = 7$, $\gamma = 5$.

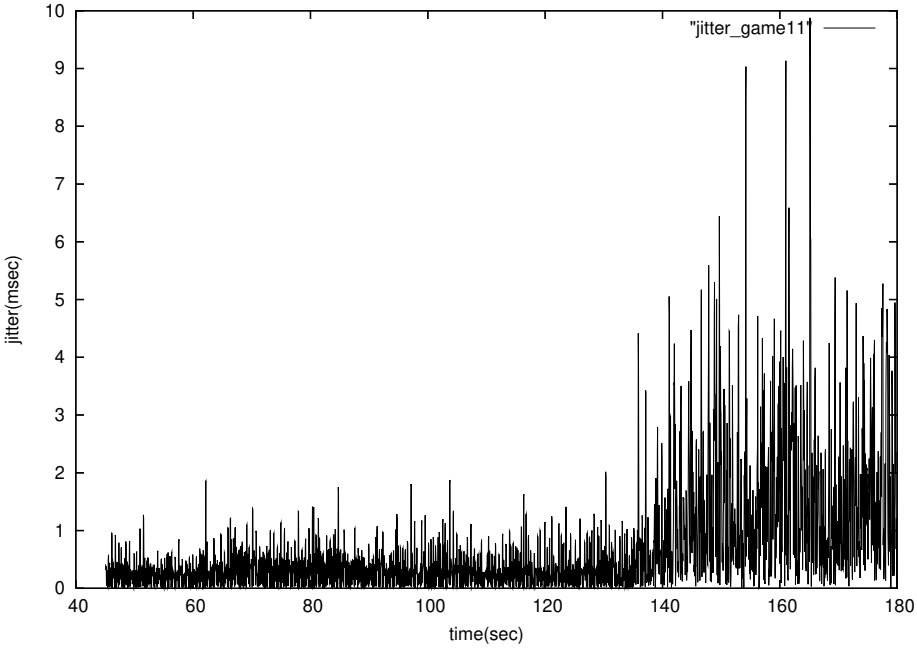


Figure 5.23: Online game jitter with concurrent TCP Vegas; $\alpha = 3$, $\beta = 7$, $\gamma = 5$.

The best tradeoff between goodput and real-time traffic jitter could be obtained for this configuration by using $\alpha = 3$, $\beta = 7$, and $\gamma = 5$. For this set of parameters, in fact, the achieved goodput is still high: 57171. At the same time, the interarrival time and the jitter of real time traffic achieves very good results. Specifically, we can see the trend of the congestion window in Fig. 5.21, while the interarrival time and the jitter of online game packets going from the server to the client is depicted in Fig. 5.22 and Fig. 5.23, respectively.

Unfortunately, TCP Vegas suffers from three main drawbacks which are well known in the scientific community. First, setting its parameters is not a trivial task and depends on many factors such as the buffer size at the bottleneck and the number of flows sharing that link. In particular, the last factor continuously changes and it is not possible to continuously adapt α, β, and γ. Second, TCP Vegas has been shown to be unstable since the contemporary presence of two poles [160]. Third, TCP Vegas behaves very poorly in terms of throughput when competing with the traditional TCP New Reno and TCP Sack (which support the large majority of the data flows in the current Internet) [148]. In presence of congestion, in fact, queues will be fully exploited by traditional TCP, while TCP Vegas will shrink its congestion window as it will sense continuous queue utilization.

Since these problems and, in particular, the last one, TCP Vegas is not factually deployable in the Internet. We need hence to find an alternative solution that could be practically employed.

5.4.5 Limiting TCP's Advertised Window

The link capacity actually achieved by a wireless connection is usually less than half of the nominal one. From tests run in our lab with a real 802.11b antenna we were able to get only a maximum of about 5Mbps over a nominal rate of 11Mbps. However this was the result of a tuning of the connectivity obtained by choosing the best positioning for the AP and the mobile device. Under regular circumstances, the factual transmission rate would be even less than that. We obtained coherent results also by simulating an 802.11b link on NS-2.

We have then simulated an 802.11g link on NS-2 and measured a maximum effectively achievable transmission rate of circa 20Mbps. We have enhanced our scenario by enabling the AP to modify the advertised window of returning acks accordingly with the bandwidth left available by the UDP-based flows as determined in (5.1). In particular, the average UDP-based aggregate traffic was computed through a simple low-pass filter and the new advertised window was determined every 200ms.

In this configuration, various values for the parameter C in (5.1) have been tested and results are reported in Fig. 5.24. In this chart, we can see the average, the standard deviation and the maximum value for the delays experienced by the gaming application in its flows directed from the server to the client. Moreover, it also contains the throughput trend of the concurrent TCP flow. We do not report here the equivalent charts corresponding to all the other real time traffic flows since their results are coherent with those presented in Fig. 5.24 and do not need further explanation.

As clearly shown, both the average and the standard deviation of the online game flow increase when we utilize higher values for C. This is obviously due to the fact that higher C values decrease the resilience to TCP bursts thus leading to some queuing at the AP. However, both the average and the standard deviation are very low for all C values and we could have the wrong impression that the online game flow always experience good performance. Unfortunately, this is not true as can be noticed by looking at the curve representing the maximum delay value experienced by packets traveling through the AP.

Finally, Fig. 5.24 also demonstrates how the throughput decreases when C is set too low. Instead, if C is set higher than the maximum achievable throughput on the channel (in this case, 20 Mbps), then the sender will be allowed to send more packets than those bearable by the bottleneck link causing queuing delays. Moreover, at a certain point, some packets will overflow the buffer and the consequent packet loss will cause the reduction of the sending window and average throughput.

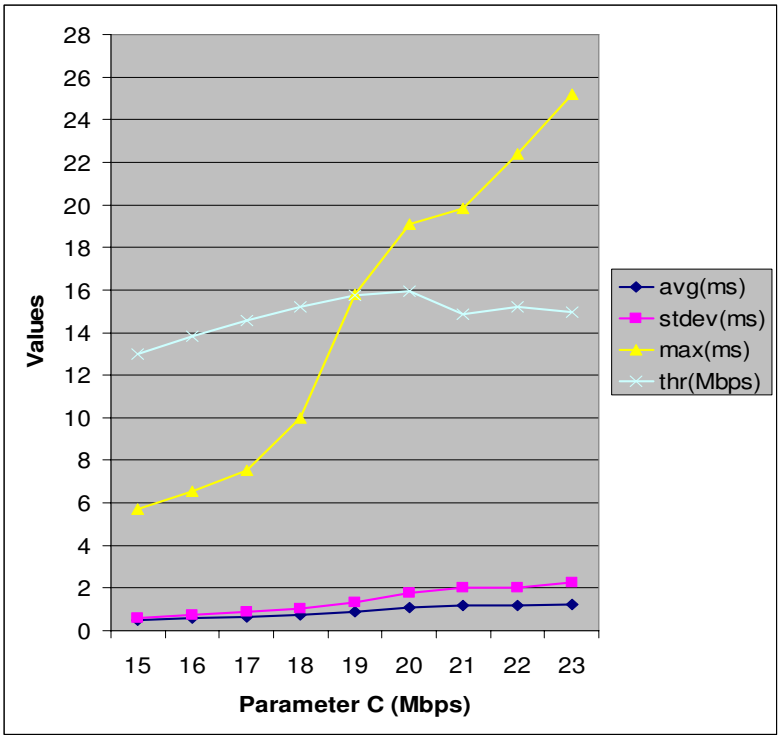


Figure 5.24: Statistical values when employing limited advertised window.

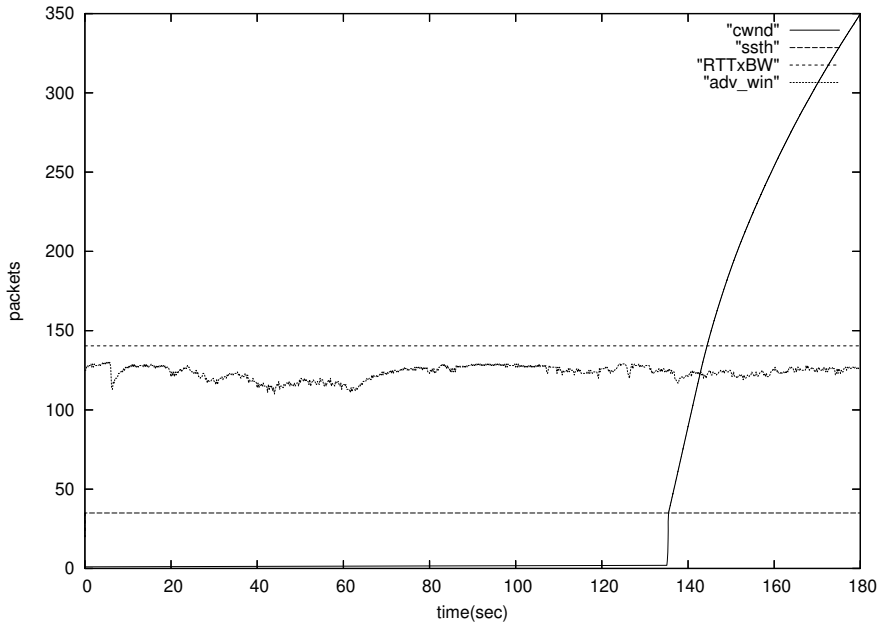


Figure 5.25: TCP behavior with limited advertised window and C = 18Mbps.

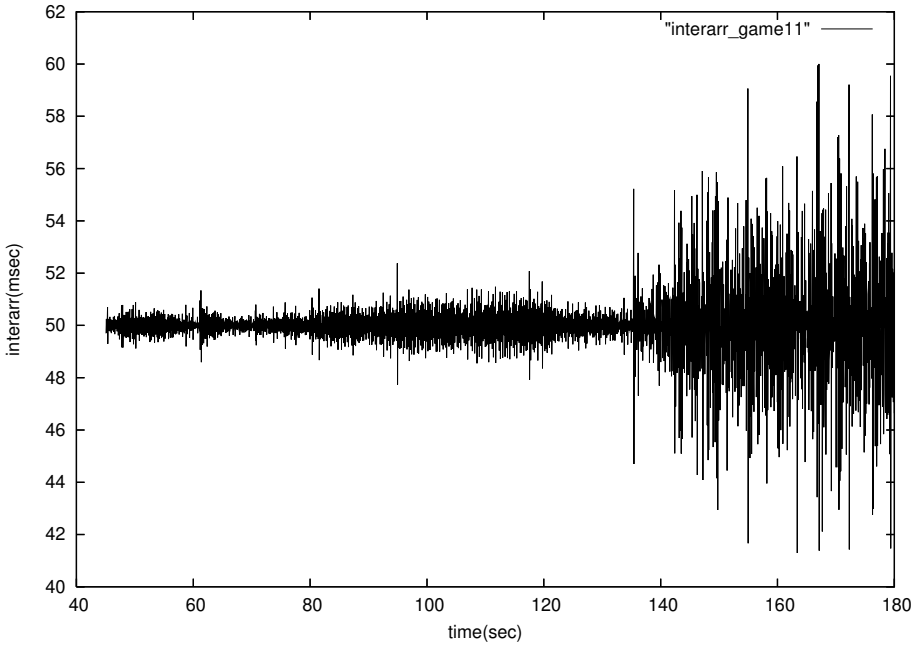


Figure 5.26: Online game interarrival time with concurrent TCP, limited advertised window, C = 18Mbps.

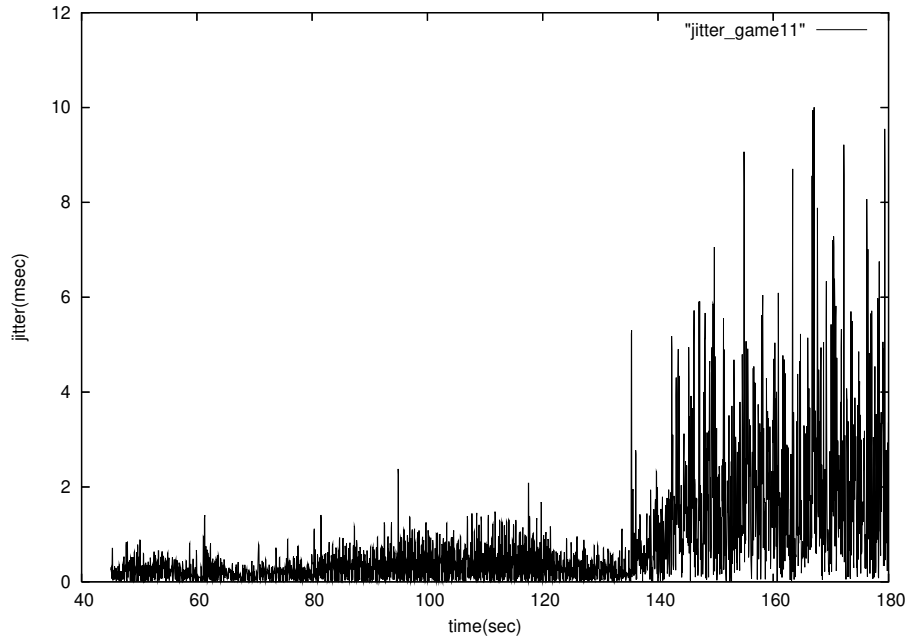


Figure 5.27: Online game jitter with concurrent TCP, limited advertised window, C = 18Mbps.

Setting C as 18Mbps (i.e., the 90% of the maximum achievable bandwidth) seems to be an appropriate choice able to guarantee both low queuing delays and high TCP efficiency. The advertised window exploited by the TCP flow is evident in Fig. 5.25, which also reports the congestion window, the slow start threshold, and the bandwidth-RTT. We have to keep in mind that the TCP flow starts at second 135 of the simulation time and that the actual sending window is determined as the minimum between the advertised window and the congestion window. Said that, we can appreciate from the chart how the AP is able to keep track of the concurrent real time traffic and determine the most appropriate advertised window. In particular, for this configuration, the final goodput in terms of acknowledged packets over 45 s hits 58677, while the interarrival time and the jitter experienced by online game packets are maintained low as demonstrated by Fig. 5.26 and Fig. 5.27, respectively.

Another interesting outcome shown by Fig. 5.25 is the proximity of the advertised window curve and the bandwidth-RTT product ones. The advertised window is prudently set close to the bandwidth-RTT product (i.e., the link pipe size) minus the aggregate UDP based traffic, and this difference also represents an estimate of the amount of real time traffic present on the channel. We have to remind that the real time traffic was simulated considering three simultaneously running applications of various type and exploiting real traffic traces. As it is evident from the chart, the difference between the two curves is relatively small if compared to the whole channel capacity thus demonstrating that real time applications generally do not have to face bandwidth shortage in an 802.11g wireless home, whilst they still have to deal with high and variable delays.

5.4.6 Summarizing Results

In order to compare the various proposed scheme we have summarized in Fig. 5.28 statistical results obtained by: i) utilizing regular TCP New Reno on a standard IEEE 802.11g MAC configuration (Regular) ii) appropriately setting the MAC layer parameters (MAC-Setting), iii) utilizing TCP Vegas in place of regular TCP (TCP-

Vegas), and iv) Smart Access Point with Limited Advertised Window (SAP-LAW) implemented at the AP.

Specifically, the MAC layer parameters for MAC-Setting were set with a maximum number of retransmissions equal to 3 and a buffer size of 50 packets at the AP. Instead, TCP Vegas was configured with $\alpha = 3$, $\beta = 7$, $\gamma = 5$ and SAP-LAW considered $C = 18$.

The compared statistical parameters are the average, the standard deviation and the maximum value of jitter experienced by online game packets entering the house (thus going from the server to the client) via the AP. Results obtained from the other real time applications running in the simulated scenario (i.e. video-stream and video-chat) are coherent with the showed ones; we hence skip to present their charts. Rather, we also show the average throughput achieved by the concurrent TCP connection.

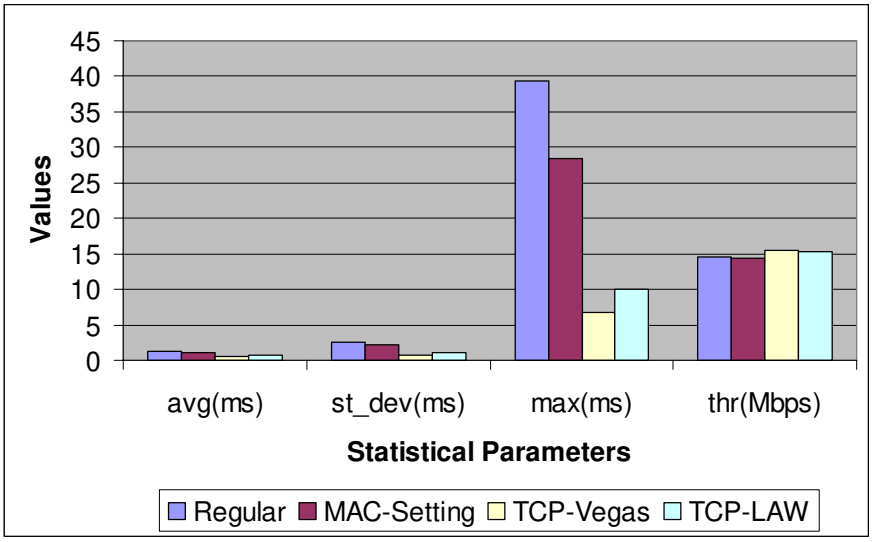


Figure 5.28: Statistical values for the various schemes.

As it is evident, employing TCP Vegas to support FTP traffic is the solution that would guarantee the best performance both in terms of lowest per packet delay and highest throughput. However, as anticipated in Section 5.2.2, TCP Vegas cannot be actually deployed in the Internet since it is not able to efficiently coexist with regular TCP flows [148].

Conversely, SAP-LAW could be easily implemented as it only requires the presence of slightly “smarter” APs. The modifications to the AP are very limited thus minimally impacting on their cost and, at the same time, our scheme can perfectly coexist with the current Internet and its employed protocols. Considering this and the remarkable results achieved, SAP-LAW represents the optimal candidate for enhancing wireless home scenarios.

CHAPTER 6

Infrastructured Vehicular Scenario

Wireless vehicular networks are soon going to be a reality, thanks to the factual interest shown by many Governments and to the market responsiveness when some new technology, both for work and entertainment, is offered today on cars. This suggest that in future we may have city streets and highways mostly covered by a web of APs specifically developed to provide connectivity between passengers in vehicles and the Internet.

Applications run by users will be both regular ones that are every day already utilized from home/office PCs (e.g., email, web-surfing, chatting, online gaming, video streaming) and new ones specifically designed for the vehicular context (e.g., traffic safety, driving directions, location based data collection, parking lot payment).

Clearly, such a scenario presents many novel issues that deserves scientific investigation. As our aim is that of supporting online gaming even for passengers traveling in cars, we are interested in evaluating how this scenario affects the performance of the considered application. Similar to the wireless home scenario presented in the previous chapter, we investigate the coexistence between elastic and real-time applications in infrastructured vehicular networks. We demonstrate how even in this challenging scenario, SAP-LAW represent a valid solution to find the best tradeoff solution between the throughput achieved by elastic applications and the per-packet delay of real-time ones.

6.1 Simulation Assessment

We tested SAP-LAW through the NS-2 simulator in two different vehicular scenarios: urban and highway [122]. In both cases we collected results for a vehicle that was transiting under four AP, one after the other. The scenario was made more realistic by having other heterogeneous traffic sharing each considered AP. Results were collected both for the mobile node and for the other nodes that generate traffic in the network.

More in detail, both in urban and highway configurations we considered a portion of road comprising four APs (AP_0, AP_1, AP_2, AP_3), seven wired nodes (W_0, W_1, …, W_7), and seven wireless ones (N_0, N_1, …, N_7). Among the wireless nodes, N_0 is the one we monitored while it was driving under the various APs. For a comprehensive description of the simulative configurations, Fig. 6.1 shows the network topology for the urban scenario, whereas Fig. 6.2 regards the highway one. The former simulates a block or a group of blocks, with 1000 m between consecutive corners, around which a car is traveling. Instead, the latter corresponds to a strip-shaped highway with APs placed every 1000 m.

For both topologies, wired links have a 100 Mbps capacity, wireless ones have a variable capacity (depending on channel interferences) of circa 19 Mbps, all wired links directly connecting two APs have a 1 ms of propagation delay (they are only 1000 m far from each other), whereas all other wired links have a propagation delay of 20 ms. Buffers in the wired connections are set equal to 70 packets, which corresponds to the pipe size, i.e. the bandwidth-RTT product. The TCP's advertised window was initially set to a very high value, 550 packets and stays there when regular TCP New Reno and AP are employed, whereas SAP-LAW dynamically and continuously set it to a value that corresponds to the maximum share of the bandwidth available for each TCP session.

The vehicular wireless standard, the IEEE 802.11p, is still only a draft. It is hence not possible to create a NS-2 module that exactly replicates it. Therefore, as already done by other researchers [114, 190, 191], we have modified the IEEE 802.11 module available for NS-2 to behave following IEEE 802.11p's specifications. In particular, the maximum (factually) achievable bandwidth was around 20 Mbps and the transmission range was

extended to reach up to 750 m. MAC layer buffers on the APs are set as equal to 1000 packets as this is one of the most common value in off-the-shelf APs.

In the simulates scenarios, wireless nodes continuously transmit and/or receive data through certain APs; the distance between these wireless nodes and their engaged APs is 100 m. We preferred to have fixed wireless nodes connected to the various APs so as to have them continuously utilized by a predetermined background traffic. This allowed us to better appreciate the characteristics of SAP-LAW with respect to employing regular protocols and APs. Simulated flows are described in Table 6.1.

Simultaneously, a mobile node (N_0) is traveling along the road passing by the coverage area of each of the APs. In the urban scenario N_0 has a speed of 14 m/s (about 50 Km/s and 32 Mph), whereas in the highway scenario its speed is 33 m/s (about 120 Km/s and 75 Mph). When N_0 moves into the coverage area of a new AP, it connects with the new antenna and continues its operations through Mobile IP's packet redirection. Needless to say, when SAP-LAW is employed, the new traffic conditions experienced on the considered AP will be taken into account by the algorithm. In different simulations, N_0 runs different applications, specifically: a TCP-based FTP or a UDP-based online game. In the former case the data flow is mostly unidirectional, from a server in the Internet to N_0 (plus ACKs on the returning path), whereas in the latter case the data flow is bidirectional, game events go from N_0 to a game server in the Internet (e.g., a GSS) and game updates go from the server to N_0. A summary of the configuration for the various simulations is reported in Table 6.2.

Similar to the wireless home scenario (see Chapter 5), applications are simulated to be as close as possible to the real world ones. Indeed, even in this case the video streaming corresponded to the real trace file of the movie *Star Wars IV* in high quality MPEG4 format [145]; frames of different sizes are hence sent with a 25 fps frequency. Online gaming traffic is inspired by real traces of the popular Counter Strike action game, and had i) a server-to-client flow characterized by an inter-departing time of game updates of 200 Bytes every 50 ms and ii) a client-to-server flow characterized by an inter-departing time of game events of 42 Bytes every 60ms [146].

As for parameter C in (5.1), three different values were tested: 18, 19, and 20 (Mbps). Clearly C value has an impact only on SAP-LAW's performances, whereas it is not employed when regular protocols and APs are utilized. In the next sections, we identify the case with regular protocols and AP with the name *TCP regular*. Finally, for the sake of a better comprehension of simulation outcomes, we report in Table 6.3 movement details of node N_0 in the urban scenario.

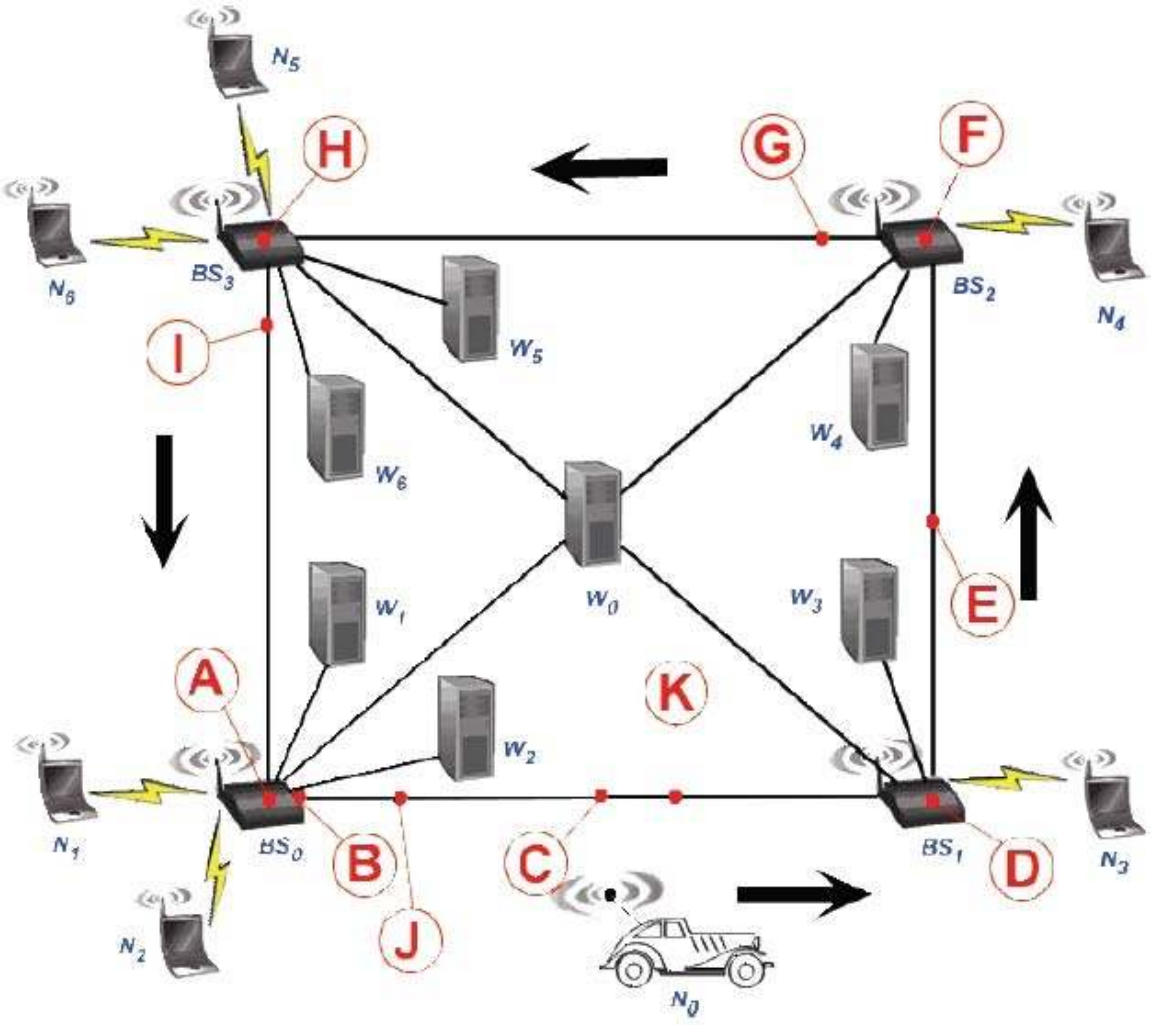


Figure 6.1: Urban vehicular scenario [122].

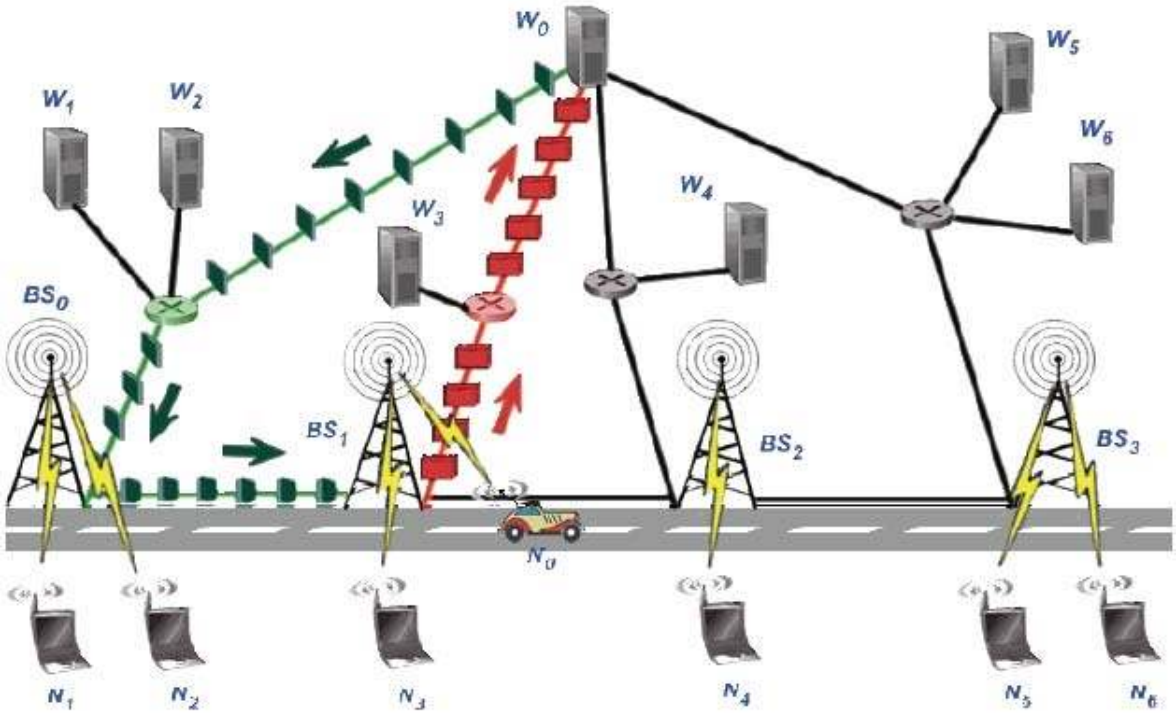


Figure 6.2: Highway vehicular scenario [122].

Table 6.1: Simulated flows.

From	To	Home Agent (AP)	Flow Type	Transport Protocol
W_1	N_1	BS_0	FTP	TCP New Reno
N_2	W_2	BS_0	Online gaming	UDP
W_2	N_2	BS_0	Online gaming	UDP
N_3	W_3	BS_1	Online gaming	UDP
W_3	N_3	BS_1	Online gaming	UDP
W_4	N_4	BS_2	Video streaming	UDP
W_5	N_5	BS_3	FTP	TCP New Reno
W_6	N_6	BS_3	Video streaming	UDP

Table 6.2: Simulated configuration.

Simulation ID	Scenario	Vehicle's Speed	From	To	Application on Vehicle	Transport Protocol
U-ftp	Urban	14 m/s	W_0	N_0	FTP	TCP New Reno
U-game	Urban	14 m/s	N_0/W_0	W_0/N_0	Online gaming	UDP
H-ftp	Highway	33 m/s	W_0	N_0	FTP	TCP New Reno
H-game	Highway	33 m/s	N_0/W_0	W_0/N_0	Online gaming	UDP

Table 6.3: Movement details for the traveling node.

Time	Location on Fig. 6.1	Distance Traveled	Engaged AP	Description
0 s	A	0 m	BS_0	Simulation start
3 s	B	42 m	BS_0	Data transmission start
36.8 s	C	515 m	BS_0	Handoff BS_0-BS_1
71.4 s	D	1000 m	BS_1	Min distance from BS_1
107.5 s	E	1505 m	BS_1	Handoff BS_1-BS_2
142.8 s	F	2000 m	BS_2	Min distance from BS_2
153.7 s	G	2152 m	BS_2	Handoff BS_2-BS_3
214.3 s	H	3000 m	BS_3	Min distance from BS_3
224.6 s	I	3144 m	BS_3	Handoff BS_3-BS_0
285.7 s	A	4000 m	BS_0	Min distance from BS_0
300 s	J	4200 m	BS_0	Simulation end

6.2 Experimental Evaluation

In this section we assess the performance improvement achieved by employing SAP-LAW in a scenario involving vehicular communications through infrastructure. Similar to

the wireless home scenario, the considered metrics are the final goodput achieved by elastic flows (FTP/TCP) and the jitter experienced by real-time ones (gaming/UDP).

6.2.1 Elastic Flow Evaluation

First, we show the case with the mobile node downloading a file through FTP/TCP while driving in the urban scenario. In particular, we present in Fig. 6.3 the congestion window for a TCP regular in the urban scenario. The well known saw-tooth shape is evident in the picture; those peaks often corresponds to a packet loss due to congestion. Yet, before loosing a packet, others where queued at the bottleneck buffer, thus generating queuing delays that affected simultaneous real-time applications.

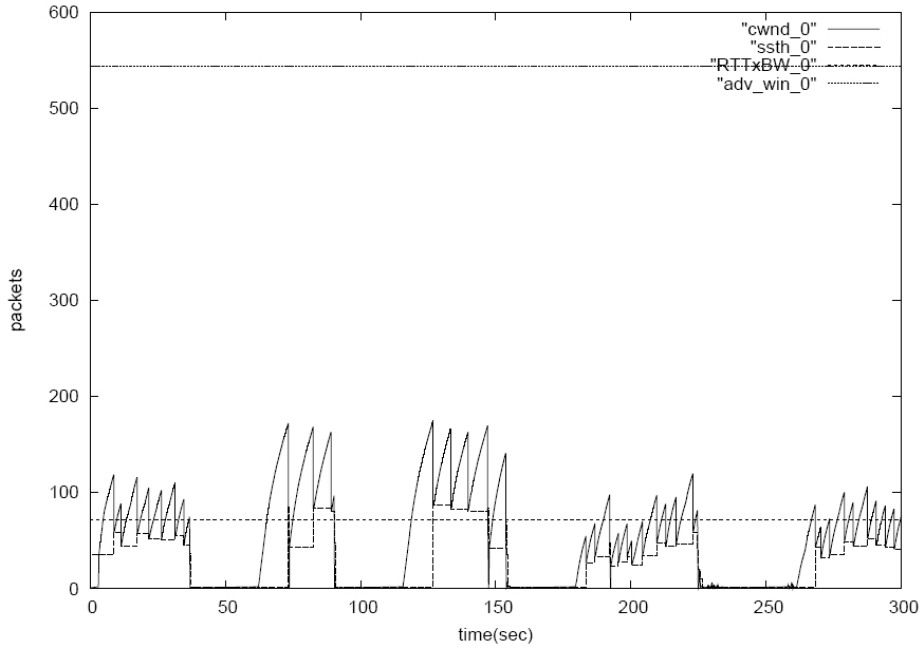


Figure 6.3: Congestion window for an FTP from W_0 to N_0 in the urban scenario; TCP regular employed.

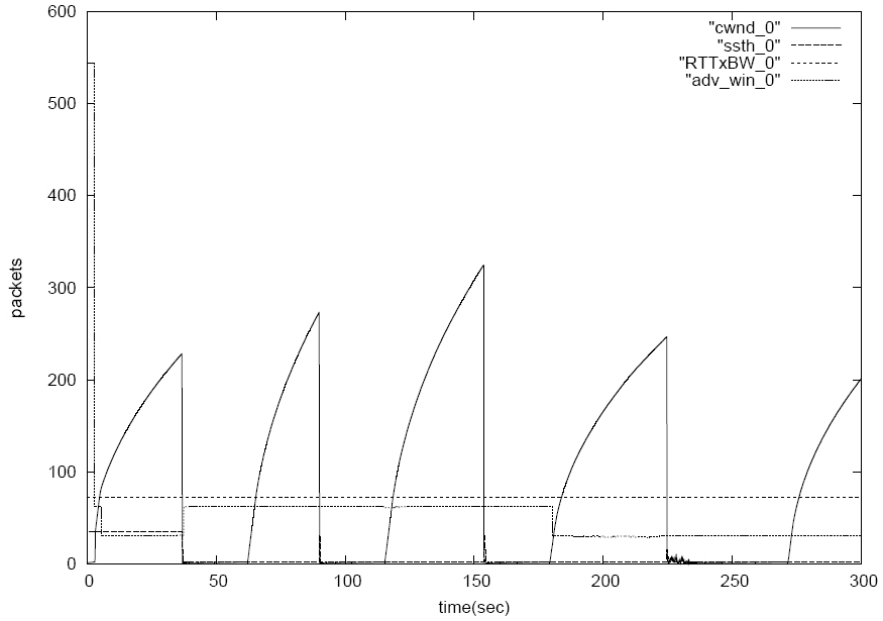


Figure 6.4: Congestion window for an FTP from W_0 to N_0 in the urban scenario; SAP-LAW employed, C = 18.

The same simulative configuration has been utilized also for Fig. 6.4; in this case, however, SAP-LAW was employed on all the APs. As a consequence, the advertised window is not static; rather, it adapts to the traffic condition through the currently engaged AP. The advertised window trend clearly shows when the N_0 switch from BS_0 to BS_1 and from BS_2 to BS_3. In fact, since both BS_0 and BS_3 are also utilized by another FTP/TCP flow each, the advertised window (representing the available share of the bandwidth for a FTP/TCP flow) is doubled in the former case and halves in the latter.

Moreover, it is evident how SAP-LAW is able to avoid congestion and losses. Indeed, the congestion window of the TCP flow keeps growing until a disconnection occurs. This does not mean that also the sending rate increased: the actual sending rate is always the minimum between the congestion window and the advertised window. By appropriately limiting the sending window, SAP-LAW is able to avoid queuing at buffers while maintaining a high utilization of the available bandwidth. The latter assertion is confirmed by Fig. 6.5 that presents the total goodput achieved by an FTP application run

on the mobile node N_0. Indeed, the goodput achieved by SAP-LAW is slightly worse that TCP regular's one, especially when considering cases C = 19 and C = 20.

Even better results are attainable by static wireless nodes. For instance, Fig. 6.6 reports the congestion window of TCP regular of a FTP flow through which N_5 is downloading a file from W_5. Two things in this chart are particularly worth of interest. First, there clearly is no sign of disconnection and, second, during the interval of time 180 s – 220 s there is a visible reduction of the height of the congestion window peaks. In that period the same AP was shared also by the traveling N_0 (running itself an FTP application).

The same configuration is represented in Fig. 6.7 with the only difference that SAP-LAW is employed in place or regular protocols and APs. As it is evident by the seamless growth of the congestion window, N_5's FTP/TCP flow does not experience any losses even when N_0 moves to the same area sharing the same wireless connection. This is the result of the employment of SAP-LAW to have flows coexist in an efficient way, without affecting one another. Indeed, the actual sending rate of the TCP flow is the minimum between the congestion window and the advertised window, with the latter continuously adapted by SAP-LAW to traffic conditions in that AP; it is evident from the chart that when N_0 start sharing the channel with N_5, the advertised window of N_5 (depicted in Fig. 6.7) is correctly halved.

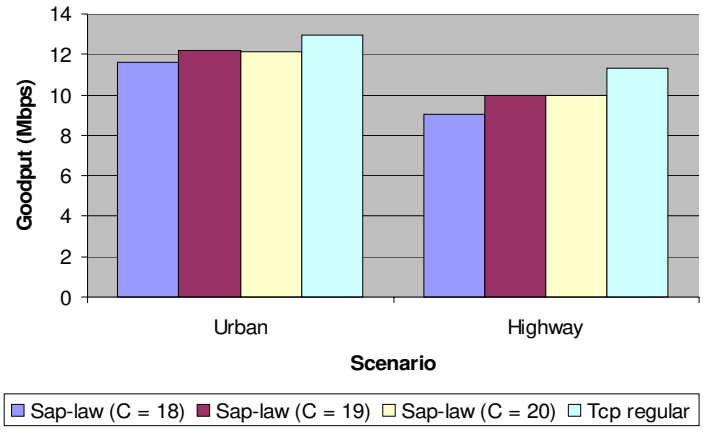


Figure 6.5: Achieved goodput by an FTP flow going from W_0 to N_0.

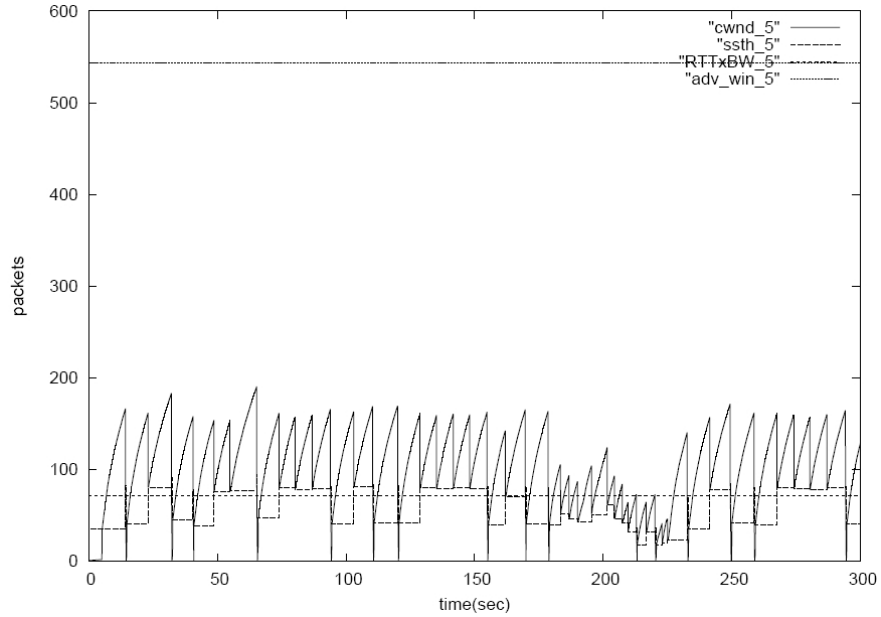


Figure 6.6: Congestion window for an FTP from W_5 to N_5 in the urban scenario; TCP regular employed.

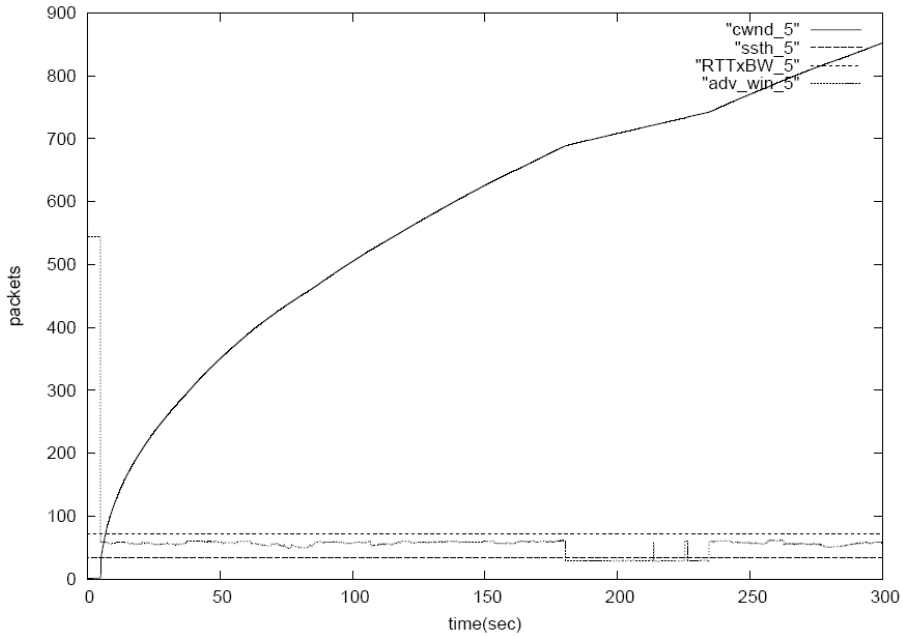


Figure 6.7: Congestion window for an FTP from W_5 to N_5 in the urban scenario; SAP-LAW employed, C = 18.

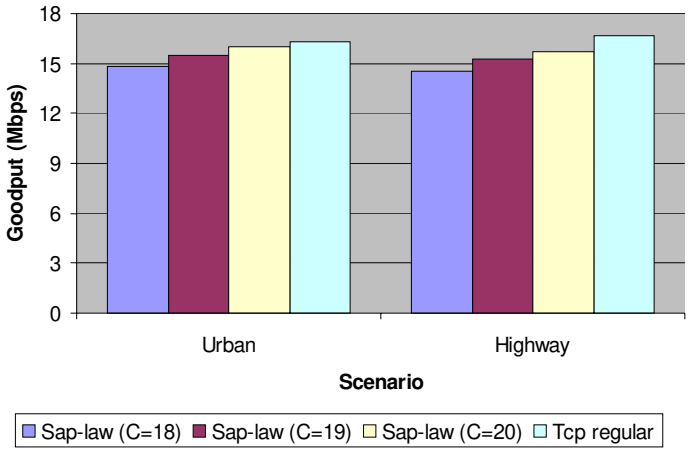


Figure 6.8: Achieved goodput by an FTP flow going from W_5 to N_5.

The resulting goodput attained by the two schemes can be compared with the help of Fig. 6.8. Even in this case, the difference between achieved utilizations of the channel is not significant.

6.2.2 Real-time Flow Evaluation

Focusing on the online game applications, Fig. 6.9 and Fig. 6.10 show the jitter experienced by the game flow directed from server W_2 to the client N_2. As stated in Table 6.1, this game session has to share the same wireless channel (and AP) with a FTP/TCP flow; this allows us to appreciate the different performances achieved by employing TCP regular (Fig. 6.9) or SAP-LAW (Fig. 6.10). Clearly, when competing with a FTP flow based on TCP regular jitter values are consistently higher during the whole simulation period, also achieving peaks of circa 50 ms of jitter, which represent a huge amount of time when trying to deliver game events in less than 150 ms from their generation. Instead, with SAP-LAW, the jitter continuously stay under 15 ms. Only in one case this does not happen: when the mobile node N_0 enters into the coverage area of the same AP engaging N_2. Yet, this single high peak reaches 33 ms whereas with TCP regular we can witness tens of peaks higher than 35 ms.

Even the reverse game flow, from the client to the server, shows a jitter improvement when employing SAP-LAW in place of TCP regular (compare Fig. 6.11 and Fig. 6.12). However, as the direction of this flow is less crowded, the difference is not that wide as it was in the previous case and it is mostly concentrated during the period the channel was shared also with the FTP/TCP flow of mobile node N_0.

In a highly mobile scenario as the one we are considering, the problem of inefficient coexistence among heterogeneous application becomes even more critical. Suddenly, a vehicle generating or receiving a great amount of traffic may move under the same AP engaging other nodes. The sudden increase of the traffic experienced by the AP may cause sudden congestion with consequent losses (particularly deleterious for elastic applications) and queuing delays (particularly deleterious for real-time ones). This problem is evident in Fig. 6.13 that shows the jitter for the online game application between W_3 and N_3 in the urban scenario with TCP regular employed. The AP engaged by N_3 (BS_1) is continuously engaged only by N_3. Therefore, there is neither congestion nor jitter until N_0 passes through the coverage area of BS_1. At that point, the high FTP/TCP traffic transferred to N_0 will suddenly increase congestion and queuing levels in BS_1 causing very high peaks in delay and jitter (up to 54 ms) experienced by the online game.

Instead, the use of SAP-LAW in place of TCP regular greatly improves performances as it is demonstrated by the corresponding Fig. 6.14 where the highest jitter peak is just 6 ms.

Aimed at testing the performance of SAP-LAW with different parameter setting we run simulations with different C values. In particular, we considered the urban scenario where the mobile node N_0 travels at 14 m/s, passing by the various APs, and running an online game applications engaged with the (game) server W_0. We replicated this simulative configuration and tested TCP regular, first, and then SAP-LAW with different C values, specifically: C = 18, C = 19, and C = 20.

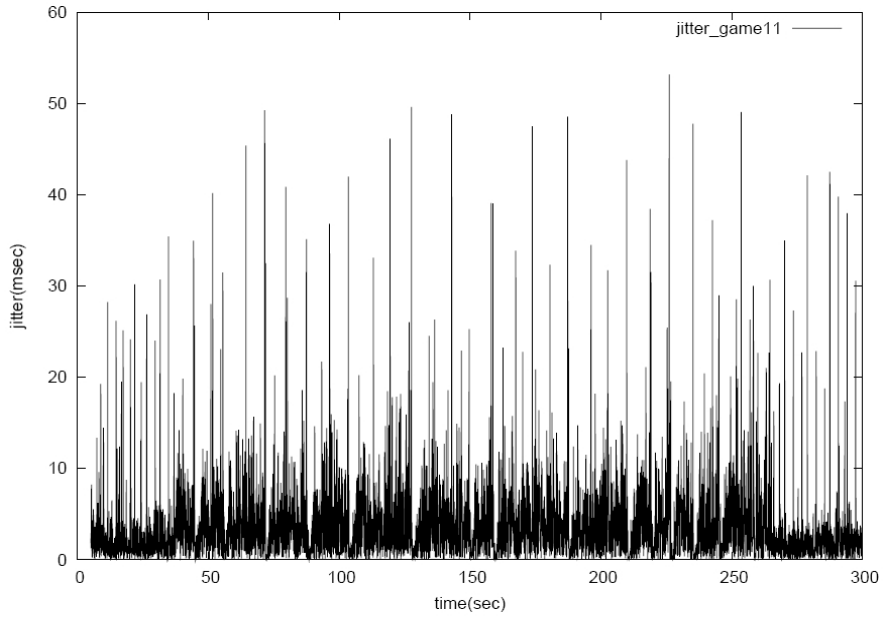


Figure 6.9: Jitter experienced by an online game flow from W_2 to N_2 (from server to client) in the urban scenario; the AP is shared also with a FTP application; TCP regular employed.

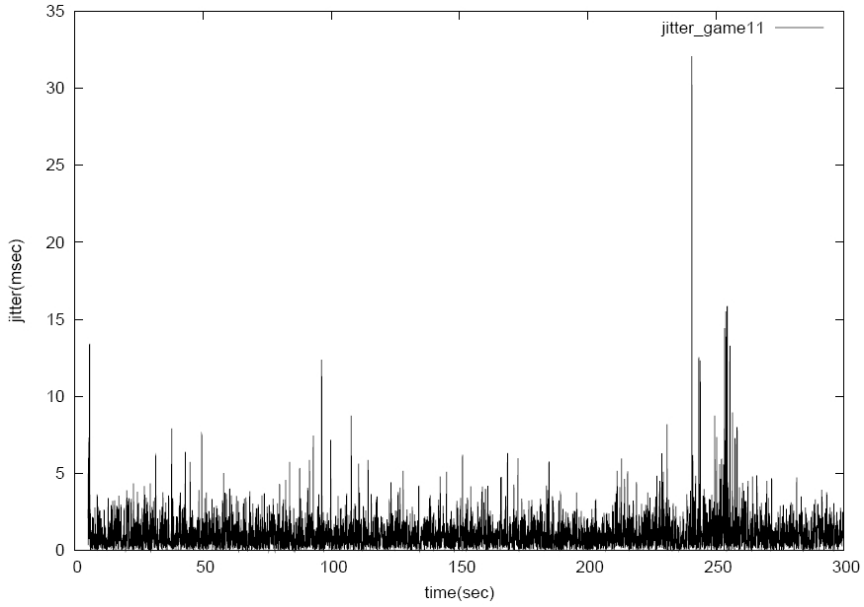


Figure 6.10: Jitter experienced by an online game flow from W_2 to N_2 (from server to client) in the urban scenario; the AP is shared also with a FTP application; SAP-LAW employed, C = 18.

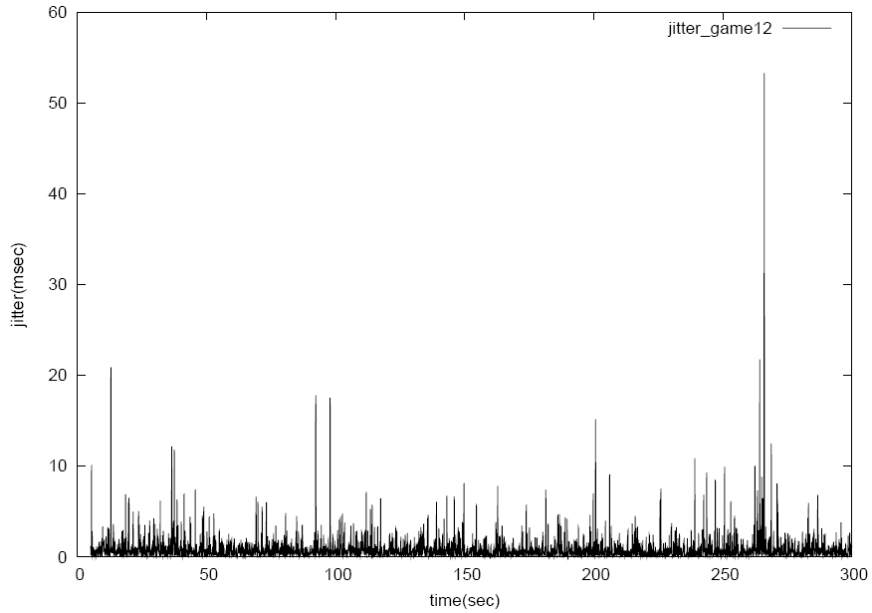


Figure 6.11: Jitter experienced by an online game flow from N_2 to W_2 (from client to server) in the urban scenario; the AP is shared also with a FTP application; TCP regular employed.

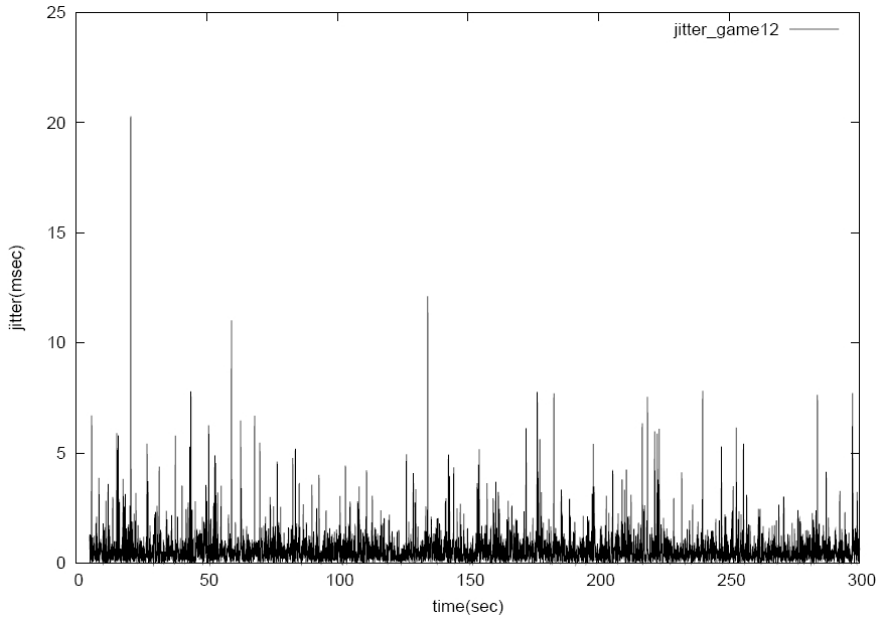


Figure 6.12: Jitter experienced by an online game flow from N_2 to W_2 (from client to server) in the urban scenario; the AP is shared also with a FTP application; SAP-LAW employed, C = 18.

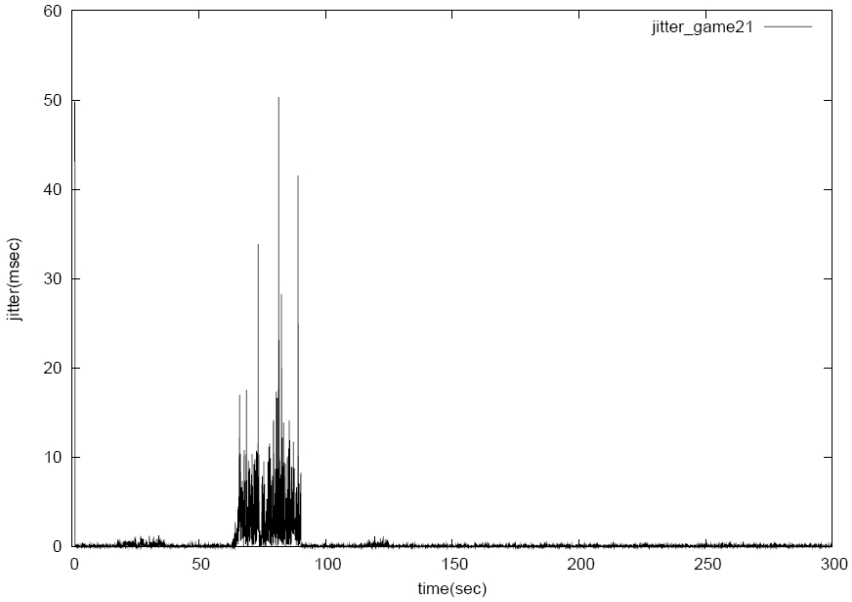


Figure 6.13: Jitter experienced by an online game flow from N_3 to W_3 (from client to server) in the urban scenario; the AP is not shared with other applications (only with N_0 sometimes); TCP regular employed.

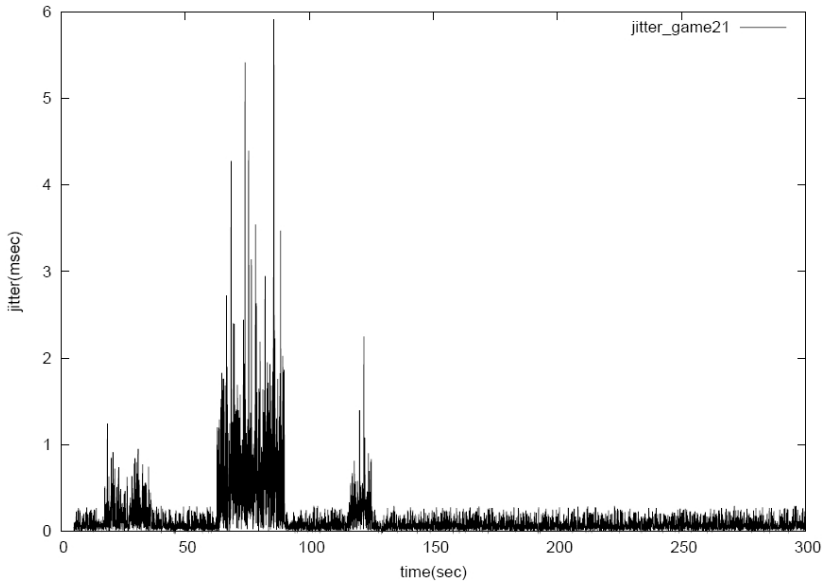


Figure 6.14: Jitter experienced by an online game flow from N_3 to W_3 (from client to server) in the urban scenario; the AP is not shared with other applications (only with N_0 sometimes); SAP-LAW employed, C = 18.

The jitter values for the four cases are shown by Fig. 6.15 – 6.18, respectively. Clearly, SAP-LAW with C = 18, outperforms the other configurations; even if SAP-LAW always shows better jitter values than TCP regular, with C = 19 and C = 20, differences are not that evident. This is due to the fact that bandwidth oscillation on the wireless channel was often closer to 18 Mbps than to higher values (i.e., 19 Mbps or 20 Mbps). Therefore, with C equal or greater than 19 (Mbps), SAP-LAW is not effective in avoiding packet queuing and hence delay jitter.

Outcomes of the highway scenario are analogous to those of the urban context. However, for the sake of completeness, we show the outcome of an experiment run with the highway scenario where the mobile node N_0 is playing to an online game while engaged with the (game) server W_0. Values depicted in Fig. 6.19 and Fig. 6.20 represent the jitter experienced by the online game flow going from the server to the client when utilizing TCP regular or SAP-LAW, respectively. In Fig. 6.19, it is clearly visible how the jitter consistently increases every time the mobile node passes by a congested AP. This effect is greatly smoothened by the ability of SAP-LAW in regulating the bandwidth that elastic applications are factually going to use.

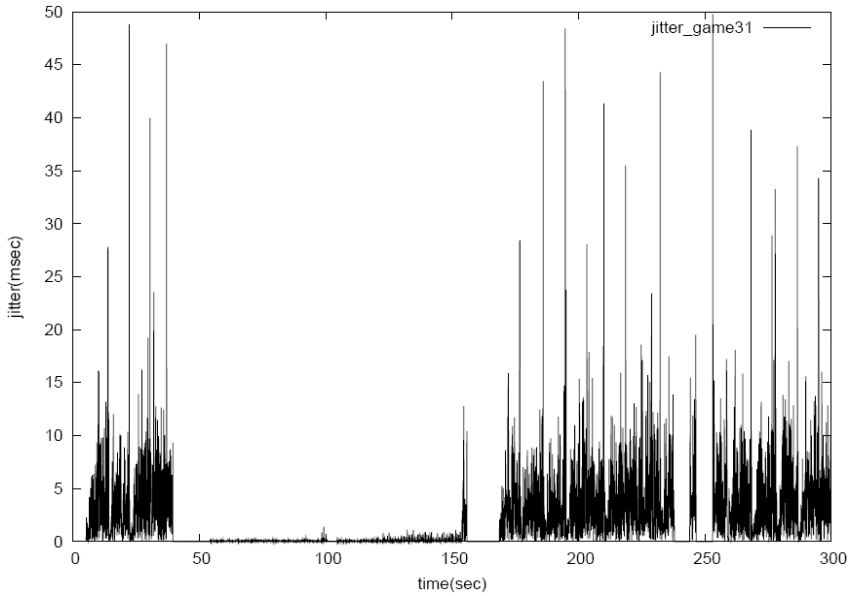


Figure 6.15: Jitter experienced by an online game flow from W_0 to N_0 (from server to client) in the urban scenario; TCP regular employed.

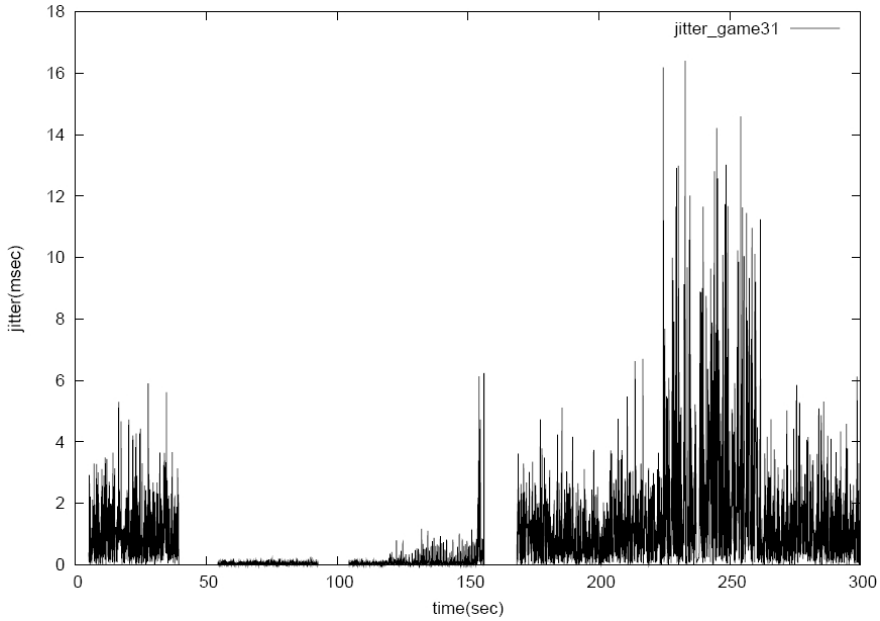


Figure 6.16: Jitter experienced by an online game flow from W_0 to N_0 (from server to client) in the urban scenario; SAP-LAW employed, C = 18.

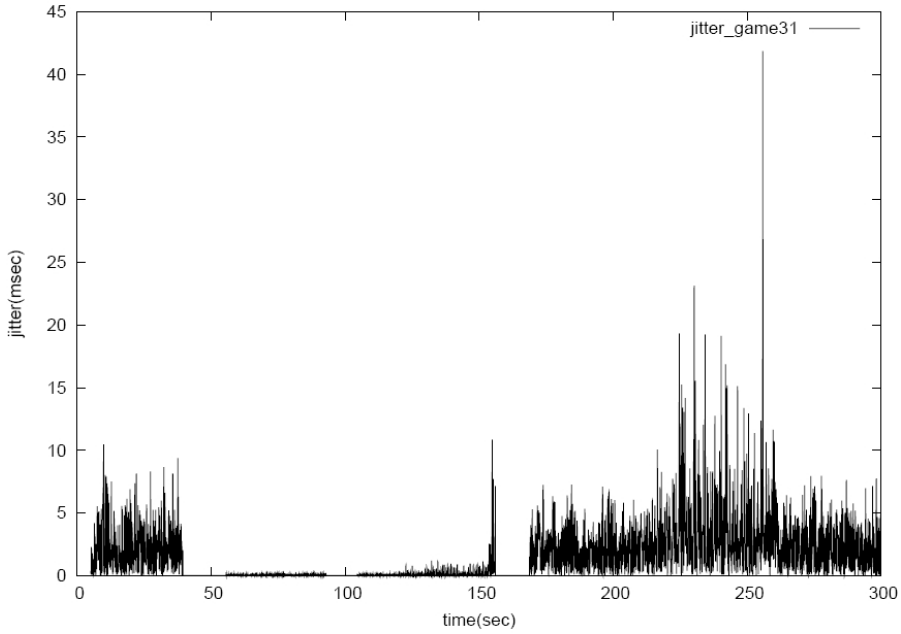


Figure 6.17: Jitter experienced by an online game flow from W_0 to N_0 (from server to client) in the urban scenario; SAP-LAW employed, C = 19.

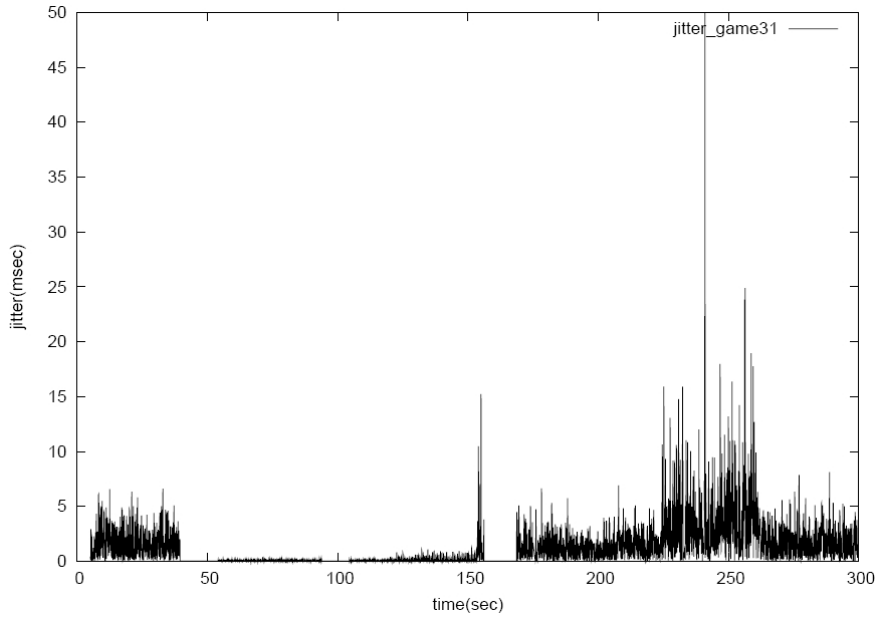


Figure 6.18: Jitter experienced by an online game flow from W_0 to N_0 (from server to client) in the urban scenario; SAP-LAW employed, C = 20.

To evaluate the performance of SAP-LAW with different parameters, also in the highway scenario, we have run simulations with various C values. In particular, we replicated the simulative configuration used for results shown in Fig. 6.20 but using C = 19 and C = 20. The jitter outcome for the former is depicted in Fig. 6.21, whereas the latter is shown in Fig. 6.22. As expected, to a higher value of C correspond a higher jitter, yet, always smaller than 14 ms. These lower values for the jitter, regardless of the employed C parameter, are due to two main reasons: i) the high speed of the mobile node in this scenario did not allow TCP flow (regardless whether considering TCP regular or SAP-LAW) to continuously reach high values of sending windows before moving out of the coverage area of a certain AP and ii) the interferences caused by the linear topology of the highway scenario creates less interferences among APs thus making them offer a higher bandwidth (20 Mbps was a bandwidth effectively often available).

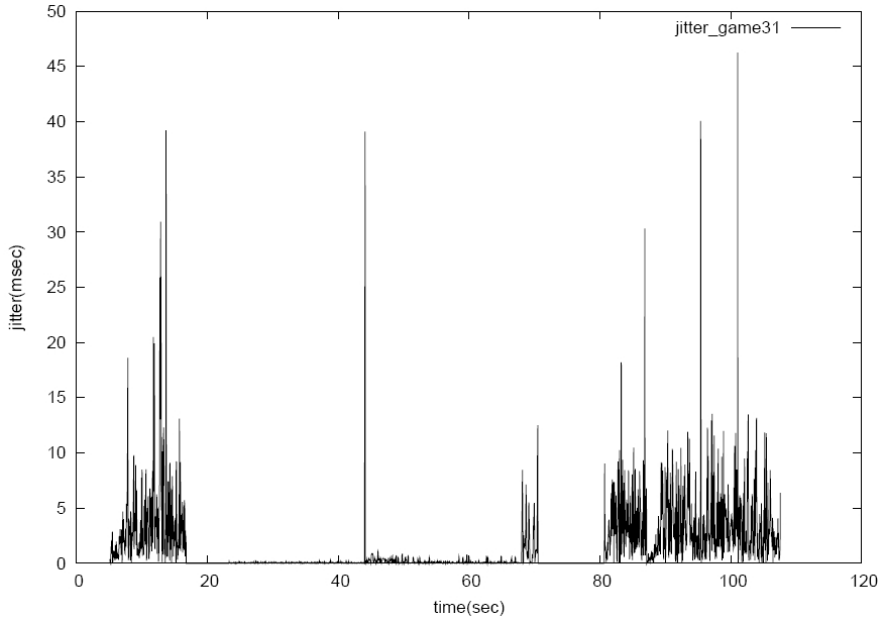


Figure 6.19: Jitter experienced by an online game flow from W_0 to N_0 (from server to client) in the highway scenario; TCP regular employed.

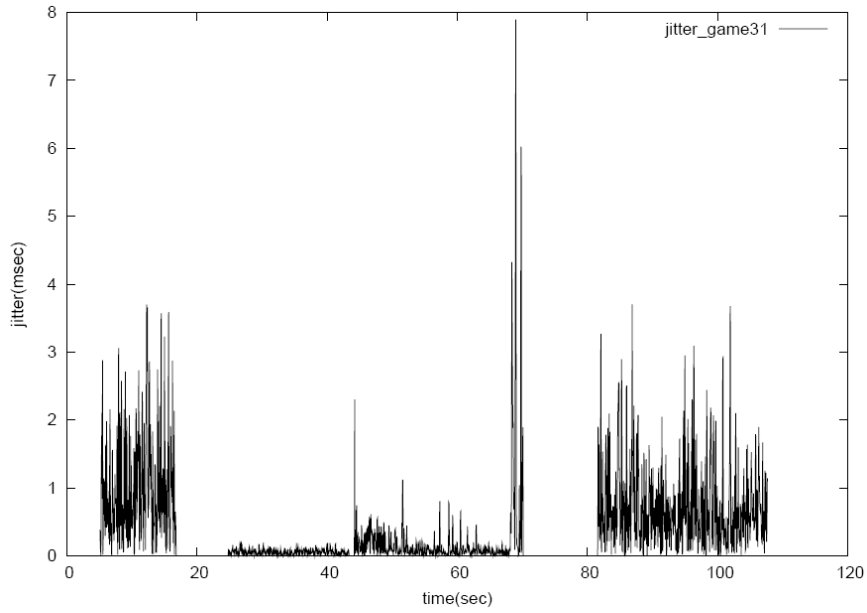


Figure 6.20: Jitter experienced by an online game flow from W_0 to N_0 (from server to client) in the highway scenario; SAP-LAW employed, C = 18.

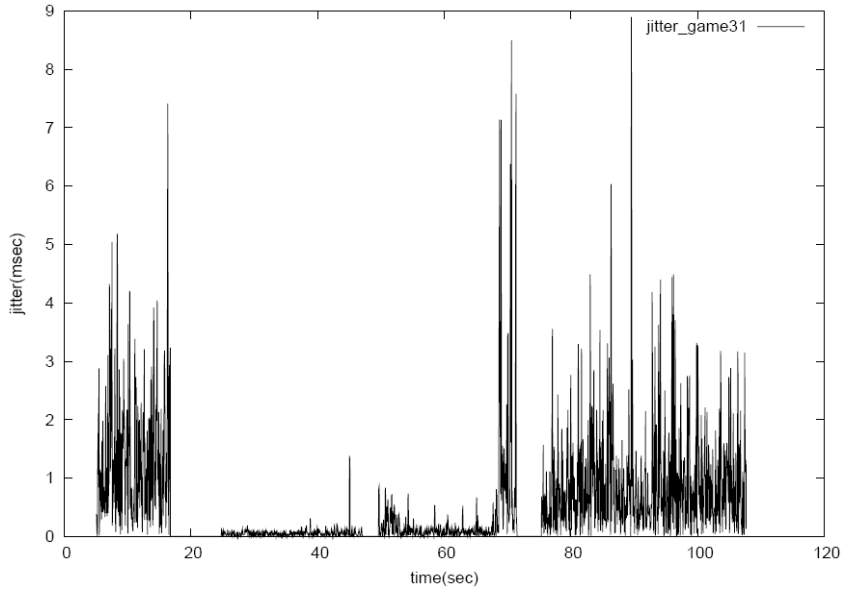


Figure 6.21: Jitter experienced by an online game flow from W_0 to N_0 (from server to client) in the highway scenario; SAP-LAW employed, C = 19.

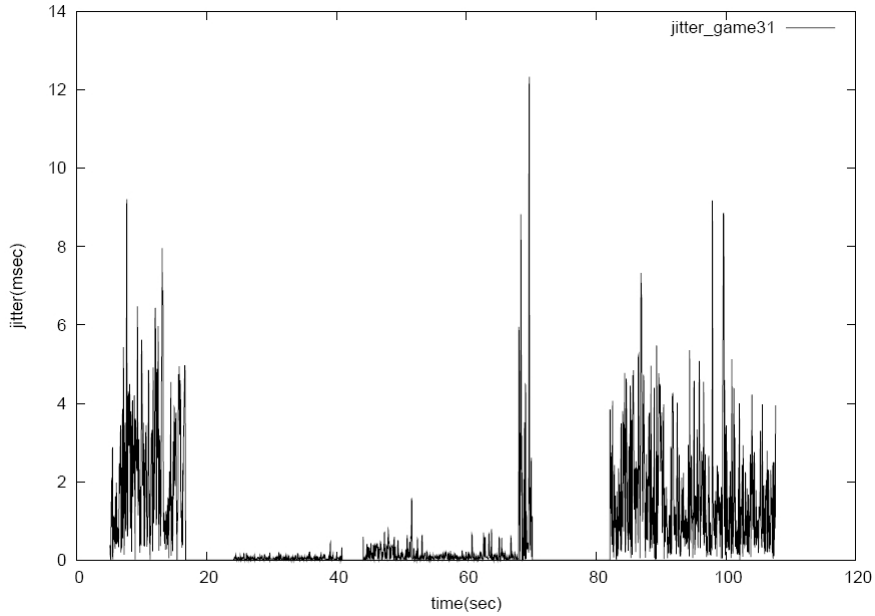


Figure 6.22: Jitter experienced by an online game flow from W_0 to N_0 (from server to client) in the highway scenario; SAP-LAW employed, C = 20.

Statistical values for the jitter of the online game flow going from W_0 to N_0, in the urban scenario and in the highway scenario, are presented in Fig. 6.23 and Fig. 6.24, respectively; in both cases SAP-LAW (with C = 18) and TCP regular are compared. Even if the average is similarly low for both schemes, the variance and the maximum value achieved sensibly differs thus demonstrating how TCP regular would cause continuous interactivity loss for the online game even just considering delays generated on the last link of the connection, whereas SAP-LAW clearly helps in maintain a smooth flow of online game packets between the (game) server and the player.

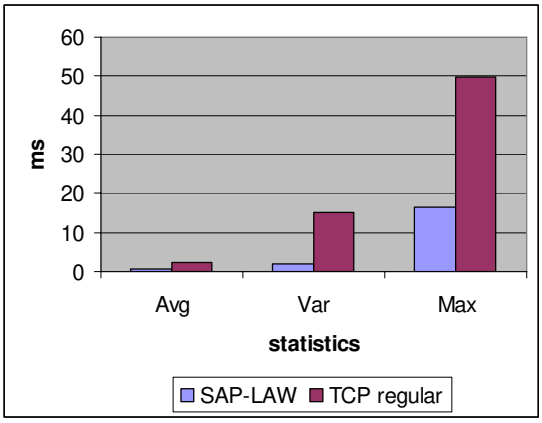


Figure 6.23: Jitter statistics by an online game flow from W_0 to N_0 (from server to client) in the urban scenario.

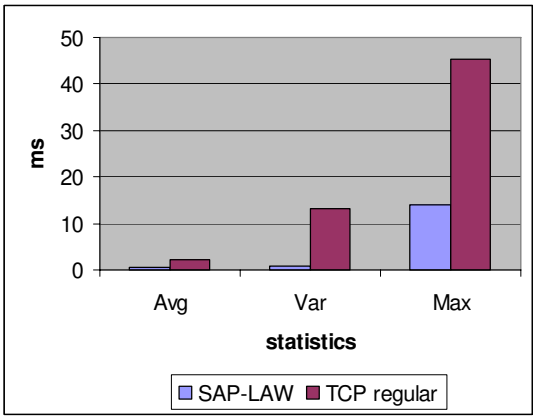


Figure 6.24: Jitter statistics by an online game flow from W_0 to N_0 (from server to client) in the highway scenario.

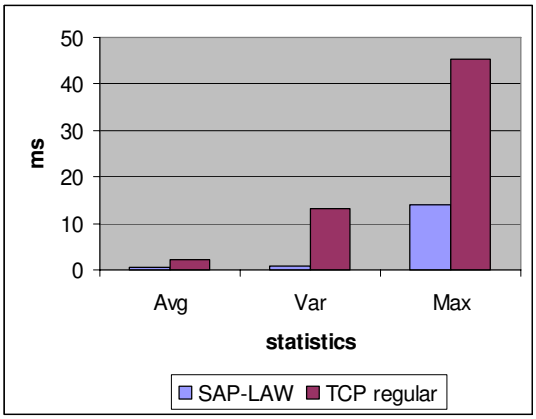


Figure 6.25: Jitter statistics by an online game flow from W_2 to N_2 (from server to client) in the urban scenario; online game application from W_0 to N_0.

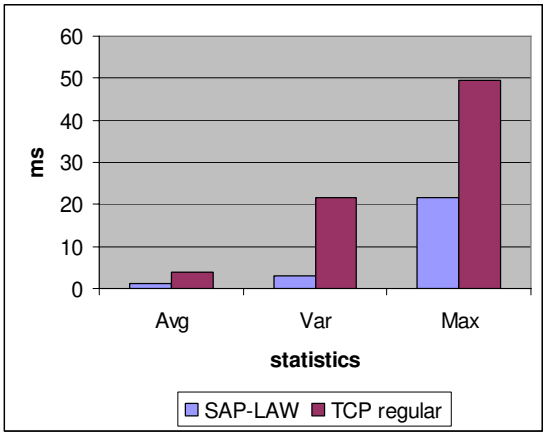


Figure 6.26: Jitter statistics by an online game flow from W_2 to N_2 (from server to client) in the highway scenario; online game application from W_0 to N_0.

We also evaluated jitter statistics for the online game flow going from W_2 to N_2, in urban and highway scenarios, and with different applications running on the mobile node N_0: an online game (Fig. 6.25 and Fig. 6.26) and an FTP (Fig. 6.27 and Fig. 6.28). All results are coherent with the preceding ones and does not need further discussion. However, for the sake of completeness, we show in Fig. 6.27 and Fig. 6.28 also values corresponding to the use of C = 19 and C = 20. Clearly, the approach that provides better

performance to real-time applications in a wider set of scenarios is represented by using SAP-LAW with C = 18. This option should be preferred even if at the cost of a slight reduction of the goodput achieved by concurrent FTP/TCP applications.

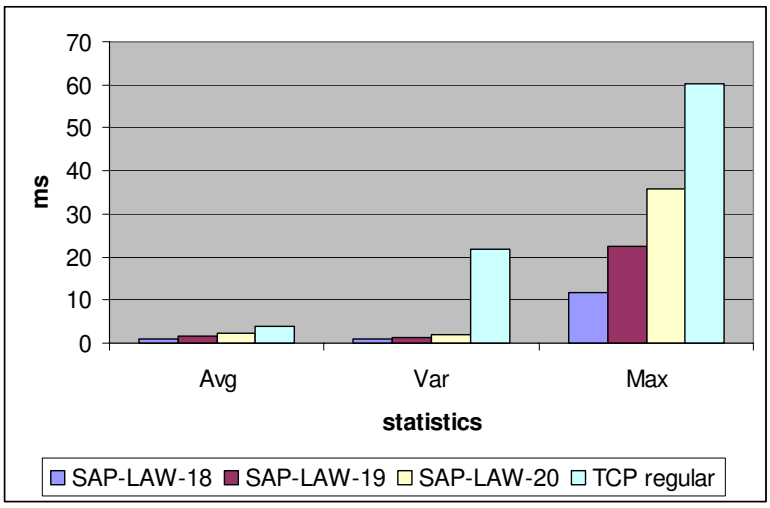


Figure 6.27: Jitter statistics by an online game flow from W_2 to N_2 (from server to client) in the highway scenario; FTP application from W_0 to N_0.

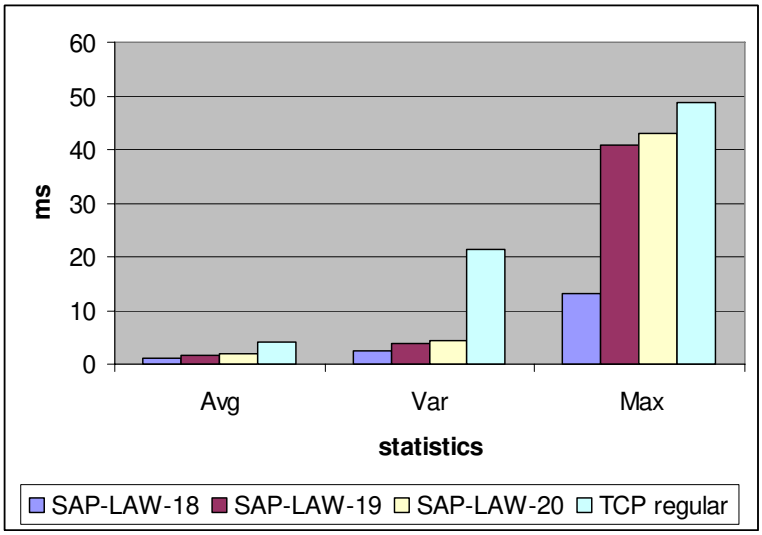


Figure 6.28: Jitter statistics by an online game flow from W_2 to N_2 (from server to client) in the highway scenario; FTP application from W_0 to N_0.

Finally, for a better comprehension of the jitter distribution achieved by the various protocols we show in Fig. 6.29, Fig 6.30, and Fig. 6.31 the cumulative function of the jitter distribution for the compared schemes. The considered scenarios are, in order: i) online gaming flow from W_0 to N_0 (Fig. 6.29), ii) online gaming flow from W_2 to N_2 with also node N_0 involved in online gaming (Fig. 6.30), and iii) online gaming flow from W_2 to N_2 with node N_0 running an FTP application (Fig. 6.31).

The difference among the various scheme is evident, yet we provide quantitative evaluation also A quantitative summary of these cumulative functions shown, in order, in Fig. 6.32, Fig. 6.33, and Fig. 6.34. In these charts the height of the columns corresponds to the jitter value associated to the 95% and 99% of the cumulative function. It is particularly interesting to observe that the 99% of game messages delivered when SAP-LAW is employed experiences very low delay jitter; whereas the same cannot be said for TCP regular.

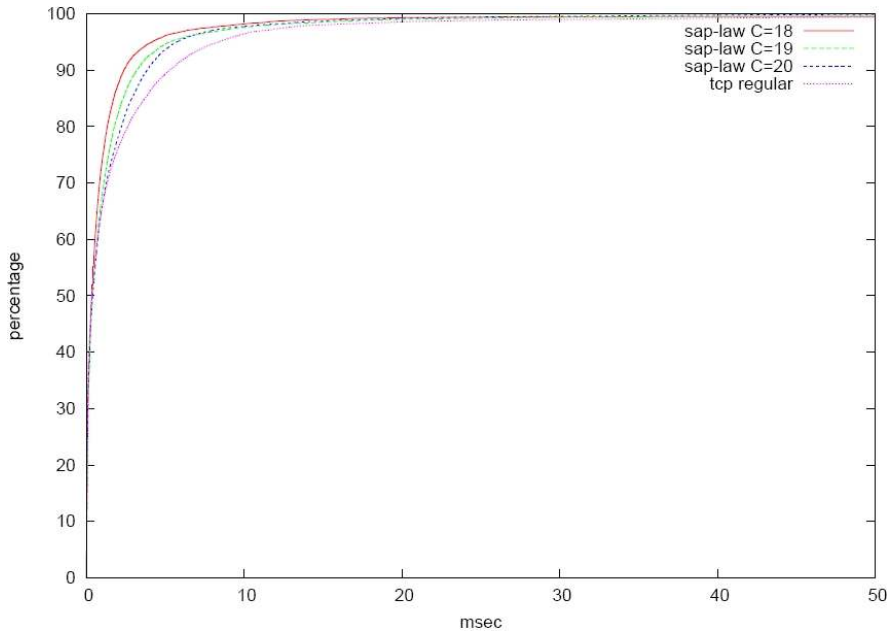


Figure 6.29: Cumulative function of the online game jitter; game flow from W_0 to N_0, urban scenario.

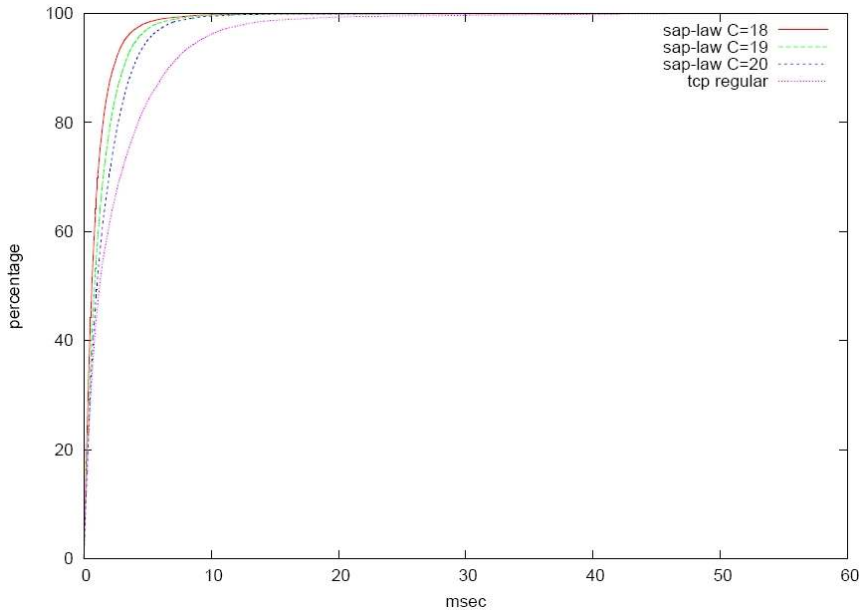


Figure 6.30: Cumulative function of the online game jitter; game flow from W_2 to N_2, urban scenario; online game application from W_0 to N_0.

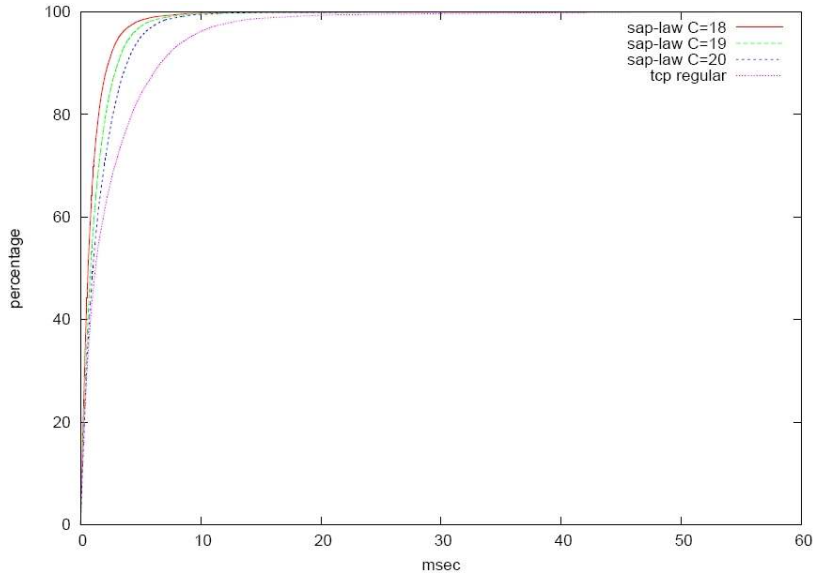


Figure 6.31: Cumulative function of the online game jitter; game flow from W_2 to N_2, urban scenario; FTP application from W_0 to N_0.

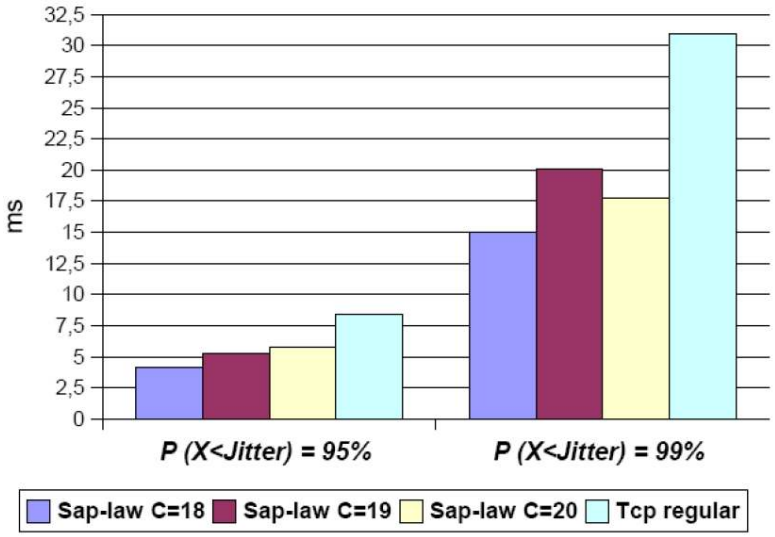


Figure 6.32: Quantitative summary of the online game jitter; game flow from W_0 to N_0, urban scenario.

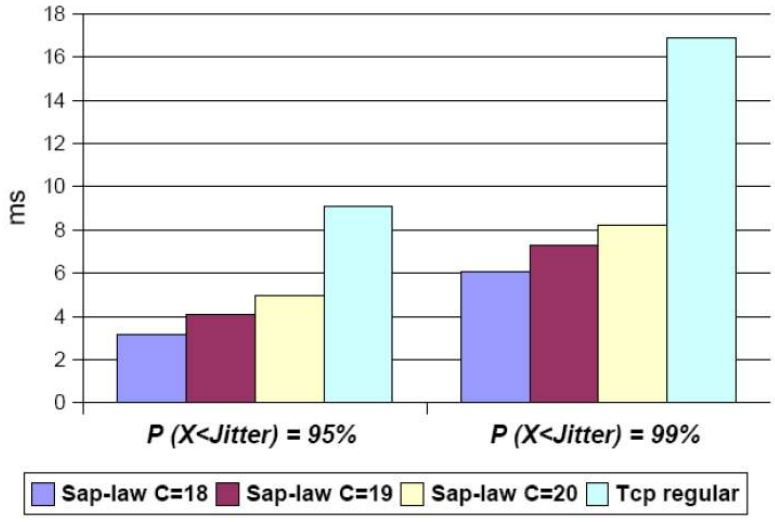


Figure 6.33: Quantitative summary of the cumulative function of the online game jitter; game flow from W_2 to N_2, urban scenario; online game application from W_0 to N_0.

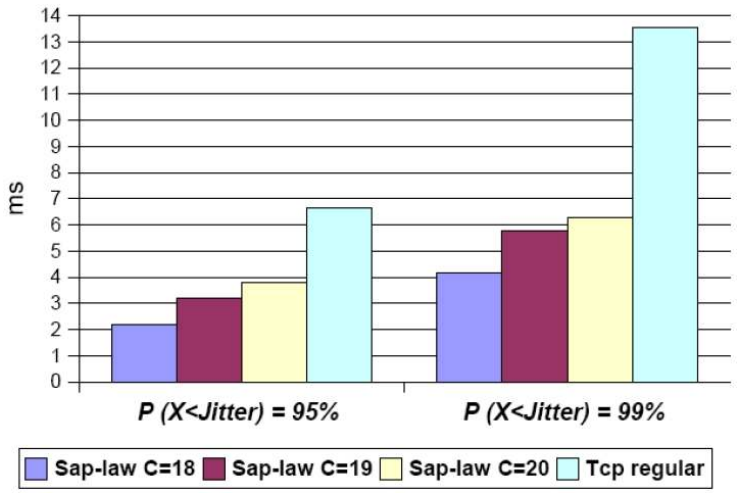


Figure 6.34: Quantitative summary of the cumulative function of the online game jitter; game flow from W_2 to N_2, urban scenario; FTP application from W_0 to N_0.

CHAPTER 7

VANET Scenario

VANETs represent an alternative to infrastructured vehicular communications for cases where an adequate coverage of APs is not present. Moreover, some applications, specifically designed for a vehicular utilization, could be more effective over an ad-hoc networking. This is the case

Every year 6 millions of car accidents happens in United States. For instance, statistics of 2003 reports 230 billions of dollars of cars' damages, almost 3 millions of wounded people and more than 40 thousands of victims [206]. As 90% of the causes are related to humans' errors, it is not surprising that ad-hoc networking technology has been proposed to enhance driving safety. To this aim, we have developed mechanisms able to quickly broadcast, even through multi-hops, alert messages from a vehicle behaving abnormally to all following vehicles in a range of kilometres [200], or video triggering messages to activate a video stream from a camera in a certain geographical location back to a requesting vehicle (e.g., first responders travelling toward an emergency area) [197].

All these schemes made use of hello messages to have each vehicle able to compute its own transmission range and utilize it to reduce the number of hops, the totally transmitted messages, and hence the delay to cover the whole area-of-interest till destination. We show now how to apply these solutions to the case of online gaming; even better, only exploiting regular game packets with no hello message overhead.

7.1 Fast Multi-Broadcast Protocol

Fast Multi-Broadcast Protocol (FMBP) is designed to quickly deliver game events to players in a certain gaming car platoon [123, 124]. Specifically, FMBP is run by all the

vehicle whose passengers are engaged in the online gaming session and its main feature is that of allowing each vehicle to estimate its current transmission range both frontward and backward. Vehicles' current estimations are included in every game event sent by those vehicles so as to have all other vehicles in range aware of them. This way, vehicles receiving broadcast game events can exploit this information to determine their position within the sender's transmission range and to assign themselves a priority in becoming the next forwarder of the received message. Indeed, by putting this information to good use the number of hops (and the delay) that a game event will experience in its trip to destination can be reduced. Needless to say, backward or frontward estimation is used when the game event has to be sent backward or frontward, respectively.

The rationale of this scheme is clearer with the help of Fig. 7.1, which shows a group of cars belonging to the same gaming car platoon. For simplicity, we suppose that cars in the figure are located 200 m apart and that the transmission range along the road is variable due to environmental conditions. Cars move from right to left and each circled area represents the backward transmission range of the leftmost vehicle in that area. Therefore, in Fig. 7.1 we have that car A has a transmission range of about 400 m, thus being able to reach within a single transmission hop cars B and C; then, car C has a transmission range of about 600 m thus being able to be heard directly by cars D, E, and F; and so forth. In this situation, if we pretend that car A sends out a game message that has to reach all vehicles in the gaming car platoon, then the optimal solution would be represented by having (only) cars C, F, and G forwarding it.

However, this optimal solution can be generated only if cars can be aware of their position within the sender's transmission range. Cars C, F, and G have to realize that they probably are the farthest car in the transmission range having heard the last message to realize that they have to take upon themselves the task of being the next forwarders. This is the reason for having a continuously updated transmission range estimation: by including this estimation in game messages.

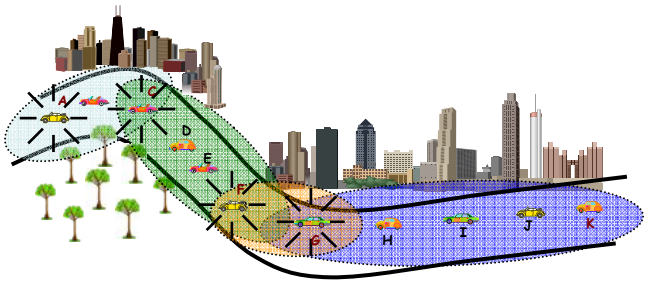


Figure 7.1: Transmission ranges in a gaming car platoon: an example.

7.1.1 Embedding an Efficient Transmission Range Estimator into Regular Game Message Exchange

To compute transmission range estimations, FMBP does not need to generate any overhead message (i.e., *hello messages* [22, 114, 190, 191]); rather, it just rely on regular game event transmissions. In particular, each vehicle includes information about the range of transmissions it has been able to hear and, at the same time, it collects data included in game messages sent by other vehicles. Consequently, a vehicle can update its transmission range upon every time a game message is received from some other vehicle.

More in detail, every game message generated by a vehicle also includes i) its own position, ii) its *backward maximum distance* (BMD) parameter, iii) its *frontward maximum distance* (FMD) parameter, iv) its *backward maximum range* (BMR) estimation, and v) its *frontward maximum range* (FMR) estimation.

Parameters BMD and FMD represent the maximum distance from which another vehicle, backward or frontward respectively, has been heard by the considered one. Supposing that both cars F and K in Fig. 7.1 embodies the farthest cars, with respect to the considered one, from which the game message has been heard, then car G computes FMD = 200 m and BMD = 800 m. Data utilized to determine these parameters are kept by each vehicle only for a certain amount of time, after which they are considered obsolete to participate in the transmission range computation.

Vehicles exploit the information about position, BMD and FMD present in game messages to compute their BMR and FMR. To this aim, both the longest distance from

which another vehicle has been heard sending a game message and the longest maximum distance advertised by heard game messages are employed. Specifically, BMR is obtained by considering only game messages coming from following vehicles; its value is computed as the largest among all their included FMD values and all distances from vehicles that generated them. Instead, FMR utilizes only game messages sent by preceding vehicles; its value corresponds to the largest among distances from preceding vehicles that have sent game messages and the BMDs advertised within.

Indeed, each car can be both a sender and a receiver of game messages. Considering for simplicity only the case where triggering messages are always sent backward (the frontward case is just specular), we have the following purpose for information included by FMBP in each message:

i) Game messages received from the front allow the receiver to compute FMD; its value will then be declared by the receiver in its game messages in order to claim: *"This FMD value represents the farthest distance from which I have been able to hear another car in front of me"*.

ii) Game messages received from the back includes the sender's FMD and position. They hence provide the receiver with information about the hearing capabilities of following cars. This is what the receiver needs to know in order to compute its BMR, which will then be sent along with the triggering messages as it were saying: *"This BMR value is the maximum backward distance at which some car would be able to hear me"*.

Concluding with the example in Fig. 7.1, G will compute a BMR of 800 m and a FMR of 200 m.

7.1.2 Game message propagation by leaps and bounds

During a game session, each player continuously generates game messages the have to be sent from that player's vehicle to all others, very quickly. As discussed, this can be achieved through a reduction of the number of hops a message traverses during its trip. This is exactly what we try to achieve with the use of BMR and FMR.

More in detail, upon players' actions, game messages are broadcast along the gaming

car platoon, both backward and frontward. Each of these messages contains information related to the game evolution also the sender's position and its current BMR and FMR. To avoid cyclic transmissions back and forth of the same game event, each message also includes its propagation direction and a unique identifier.

BMR and FMR represent how far a transmission is expected to go before the signal becomes to weak to be intelligible. Clearly, only BMR is utilized when a message is propagated backward, whereas only FMR is utilized when the message is propagated frontward. Indeed their values are used by vehicles on the message's path to determine which one among them will have to take upon itself the task of forwarding it on the next hop. Since our aim is that of minimizing the number of hops to reduce the propagation delay, we want the farthest possible vehicle from the sending one to perform this task. Therefore, the longer the relative distance of the considered vehicle from the sender with respect to the transmission range estimation, the higher the priority of the considered vehicle in becoming the next forwarder.

In particular, vehicles' priorities to forward a game message are determined by assigning different waiting times from the reception of the message to the time at which they will try to forward it. This waiting time is randomly computed based on a contention window value, as inspired by classical backoff mechanisms in IEEE 802.11 MAC protocol [142].

If, while waiting, some other vehicle closer to the destination already forwarded the game message, all vehicles between the sender and the forwarder abort their countdowns to transmission as the message has already been propagated "over their heads". Instead, all vehicles between the forwarder and the destination will participate to the "forwarding contest" for the next hop. Obviously, the larger the contention window, the more likely somebody else will be faster in forwarding the game message. We also want to point out that BMR and FMR values advertised in the game message are updated at each hop with the value computed by the forwarder. This way, on each hop a proper transmission range estimation is used, based on information related to that specific area.

The contention window of each vehicle is measured in slots and varied between a

minimum value (*CWMin*) and a maximum one (*CWMax*), depending on the distance from the sending/forwarding vehicle (*Dist*) and on the advertised estimated transmission range, which corresponds to BMR if the message is directed backward, or to FMR if the message is directed frontward. The case for BMR is summarized by (5.2).

$$\left\lfloor \left(\frac{BMR - Dist}{BMR} \times (CWMax - CWMin) \right) + CWMin \right\rfloor \tag{5.2}$$

This scheme ensures that the closest vehicle to the destination among those within the transmission range of the sender/forwarder will be statistically privileged in becoming the new forwarder. For instance, considering the setting in Fig. 7.1, if G forwards the triggering message advertising a correct BMR of 800 m, then the contention windows computed by H, I, J, and K based on (5.2), will be 776, 528, 280, and 32 slots, respectively. Consequently, K is more likely to become the next forwarder of the game message and with a high probability the final forwarder-chain will coincide with the optimal solution stated in Section 7.1, i.e., A, C, F, and G.

7.2 Simulation Assessment

We report the outcomes of an extensive campaign of simulative experiments we run to test our FMBP scheme and compare it with other possible schemes inspired by scientific literature.

In these experiments, the length of the vehicular network varies from 1 to 8 Km and vehicles with communicating capabilities are placed in average every 20 m, thus having from 50 to 400 vehicles that are involved in the transmission/forwarding process (even if not directly engaged in the online game session). Note that this does not imply that there were not other vehicles on the road with no communication capabilities or that refuses to act as relay for other vehicles' transmissions; indeed considering the case of a freeway with multiple lanes, several vehicle densities are possible.

Among vehicles with communication capabilities, a number of them, varying from 2 to

50 and uniformly distributed over the vehicular network, are playing among each other. This means that game events are periodically generated in these nodes and broadcast, even through multi-hop, to all other players in the network. Actually, different sending rates are considered to test how the system behaves in presence of different kind of games, from frenetic fast-paced games, to slower strategic games, i.e., 100 ms, 300 ms, 500 ms, whereas the size of each game event was 200 Bytes, constantly [73]. The actual transmission range varies from 300 m to 1000 m in order to test extreme values that have been declared by the IEEE 802.11p developing committee [207].

Focusing on FMBP's parameters, we have set *CWMin* and *CWMax* equal to 32 and 1024 slots, respectively, as inspired by the standard IEEE 802.11 protocol [142]. Two different slot sizes have been compared: 9 μs, which corresponds to the value utilized by IEEE 802.11g [205] and 200 μs, which allows a larger time distribution of contention delays among communicating cars. In particular, in the latter case, we expect to witness a reduced number of collisions among game events even if at the cost of an increased total delivery time.

In all of the experiments reported in Section 7.3, if not differently stated, the default configuration is represented by a vehicular network with 8 km between the first car and the last one, negligible width, about 300 m of factual transmission range (depending on the interference level), 50 simultaneous players sending out game events every 300 ms.

We have compared our FMBP with a scheme that takes inspiration from [189] to work in a scenario with multiple game event transmissions. This scheme is similar to FMBP in that it utilizes (5.2) in the attempt of having the farthest node in the transmission range to become the next-hop forwarder. However, it differs from FMBP because it simply assumes to have the transmission range parameter in (5.2) constantly set to a predetermined value, rather than being able to dynamically compute it according to the factual channel conditions. Specifically, we name this scheme *Static300* if it considers 300 m as the transmission range parameter, and *Static1000* if it uses 1000 m.

Needless to say, Static300 and Static1000 represent the ideal scheme when the factual transmission range is indeed 300 m and 1000 m, respectively. In any other situation, the

utilization of a wrong parameter in (5.2) could result in performance degradation. Indeed, results in Section 7.3 also show that FMBP performs as well as the ideal Static algorithm. Prominent advantage of our approach, this result is achieved without requiring perfect knowledge of the network topology, but just employing the transmission range estimator we designed.

7.3 Experimental Evaluation

To compare the various schemes, we analyze their ability in quickly deliver online game events to players under various conditions. Our tests focus on game interactivity when the system scales in terms of network length, number of players, and game event sending rate.

To this aim, we utilized the following metrics: i) average number of hops that a game event experiences to cover the whole gaming car platoon; ii) average number of slots that a game event cumulatively waits on forwarding vehicles before being transmitted on the next hop; iii) average number of transmissions game events require to cover the gaming car platoon; and iv) average transmission time required by a game event to cover the whole gaming car platoon; and v) the percentage of sent game events that reached all the players in the network. Whereas the first four metrics are related to the game interactivity, the fifth one is concerned with having all players watching a consistent game state evolution.

For the chosen metrics, we considered only game events belonging to the worst case: those sent by the passenger within the car leading the gaming car platoon. Indeed, these messages experience the longest transmissions both in terms of hops and delivery time, as they have to cover the whole platoon to reach all engaged players. Instead, a game event generated by a player located in the middle is simultaneously forwarded frontward and backward through two replica messages that have to cover, in average, just half of the platoon's length.

7.3.1 Network Length Scalability

First, we evaluate the ability of FMBP in ensuring interactivity with diverse lengths of the gaming car platoon. To this aim, Fig. 7.2 – Fig. 7.5 present results for the four aforementioned metrics with respect to different length of the vehicular network and slot duration.

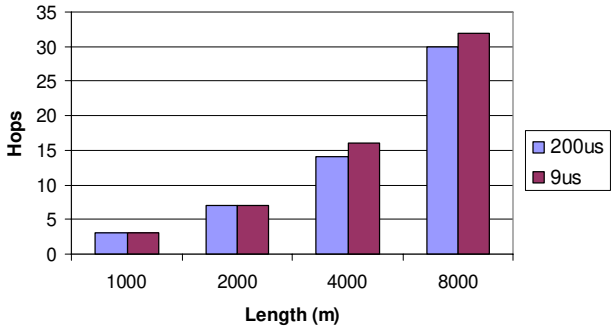


Figure 7.2: Average number of hops required to cover the whole gaming car platoon; 50 simultaneous players utilizing FMBP, 300 ms of message generation interval for each player, 300 m of factual transmission range.

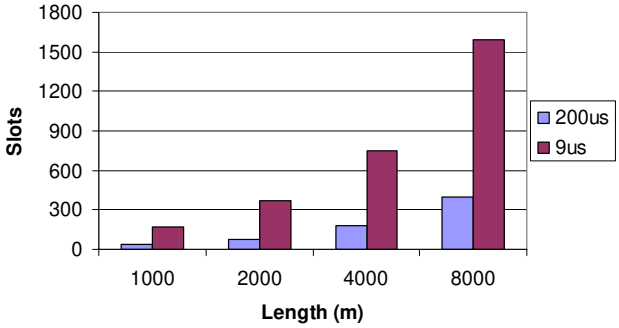


Figure 7.3: Average number of slots cumulatively waited at forwarding vehicles by a message that has to cover the whole gaming car platoon; 50 simultaneous players utilizing FMBP, 300 ms of message generation interval for each player, 300 m of factual transmission range.

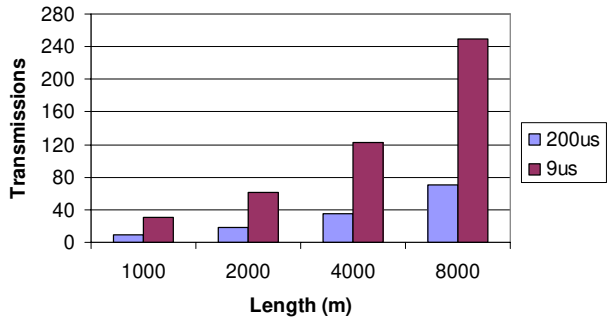


Figure 7.4: Average number of transmissions that a single game event experience while covering the gaming car platoon; 50 simultaneous players utilizing FMBP, 300 ms of message generation interval for each player, 300 m of factual transmission range.

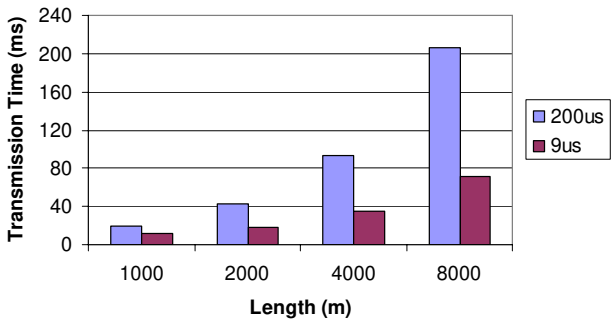


Figure 7.5: Average transmission time required to cover the whole gaming car platoon; 50 simultaneous players utilizing FMBP, 300 ms of message generation interval for each player, 300 m of factual transmission range.

Obviously, the smaller the vehicular network is, the faster game event delivery results. Indeed, outcome values are generally proportional to the length of the vehicular network and Fig. 7.2 shows how the number of hops that game events have to traverse to cover the whole gaming car platoon is about 3-4 hops for each km, regardless of the vehicular network's size and of the slot duration. This is coherent with the fact that the actual transmission range was around 300 m.

The number of slots a game event has to wait before being forwarded on the next hop

also depends on possible collisions that forces vehicles to retransmit the game event even multiple times. This happens more often when the time width of a slot is very small, i.e., 9 μs, as demonstrated by Fig. 7.3 and Fig. 7.4. Indeed, Fig. 7.3 shows that with 9 μs, the number of slots that a game event experiences in its path through the vehicular network is much higher than when 200 μs is employed; while Fig. 7.4 demonstrates that this is due to a higher number of transmissions or, in other words, retransmissions caused by message collisions.

On the other hand, having time slots about 20 time smaller allows the 9 μs setting to still result in shorter total transmission time for each broadcast game event (see Fig. 7.5). Yet, 200 μs represents a better tradeoff among the requirements for limiting the number of message collisions and for having game events quickly covering the whole gaming car platoon. Indeed, it has to be said that all the reported total delivery times are low enough to be acceptable for many games, e.g., strategic games, which represent the majority of online games played by online gamers today [25]. If considering just fast paced games (e.g., first person shooters, car races), the only case where the average transmission time of the 200 μs configuration surpasses the 150 ms interactivity threshold is when considering long vehicular networks (i.e., 8 km). Therefore, it would be enough to limit the gaming car platoon to a shorter maximum length, for instance 4-6 km to employ the 200 μs time slot.

These outcomes and the need for conciseness encourage us in considering just the case of a 8 km long vehicular network for the tests reported in the rest of the paper.

Finally, as the main advantage and contribution of our scheme is represented by its transmission range estimator, we evaluate in Fig. 7.6 its efficiency in terms of the amount of time required to dynamically compute the factual transmission range that is currently available on each vehicle. As expected, with higher game message generation rates, FMBP needs less time to compute the correct estimation. This is a logic consequence of the fact that the estimation is based on information about vehicles' positions and "hearing" distances included in exchanged messages. Therefore, the more the message, the more the information received, and the quicker the correct estimation can be built.

However, all the evaluated configurations shows very little delay, 20 ms at most, which is a really encouraging value; it proves that our scheme is able to adapt itself extremely rapidly, which is a crucial feature in a highly dynamic scenario such as a vehicular network and with highly delay sensitive application such as online gaming.

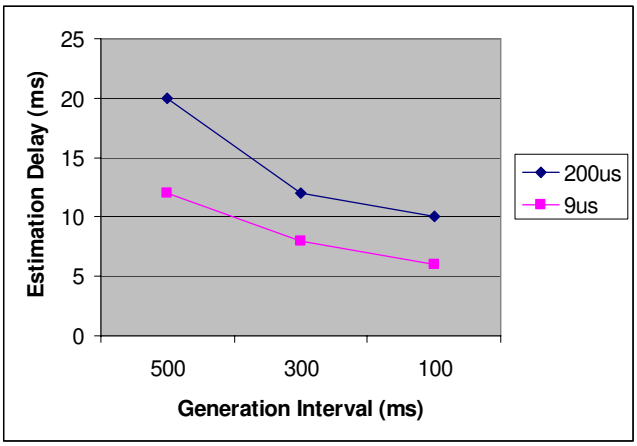


Figure 7.6: Average transmission time required to cover the whole gaming car platoon; 50 simultaneous players utilizing FMBP, 300 m of factual transmission range.

7.3.2 Player Scalability

The simultaneous number of players that a game platform can support is important as it relates to higher revenues for game or network providers or just because humans are social being: "the more we are, the funnier it is". Focusing on this aspect, we report in Fig. 7.7 – Fig. 7.10 results achieved by employing FMBP with a different number of simultaneous players, from just 2 to 50.

As it is evident, results do not change significantly when increasing the number of players that are engaged in the online game. This resilience to a higher number of players, and hence to a more intense traffic on the channel, is obtained thanks to the ability of FMBP in limiting the amount of game events simultaneously "on air". Indeed, each message is forwarded by only few vehicles and is quickly broadcast over the whole network, thus limiting the amount of time a message spend "on air" occupying shared

resources. Analogous results in terms of resilience to player scalability are obtained even when considering different vehicular network's length. We hence omit to present them here as they would not bring any further information for our evaluation.

Considerations expressed in the previous subsection about the different slot durations hold even in this series of tests. Indeed, the case with 50 players in Fig. 7.7 – Fig. 7.10 corresponds to the case with 8 km of network length in Fig. 7.2 – Fig. 7.5, respectively. As an example, we report in Fig. 7.11 transmission times experienced by FMBP in the considered scenario, comparing two possible time slot sizes: 9 μs, and 200 μs.

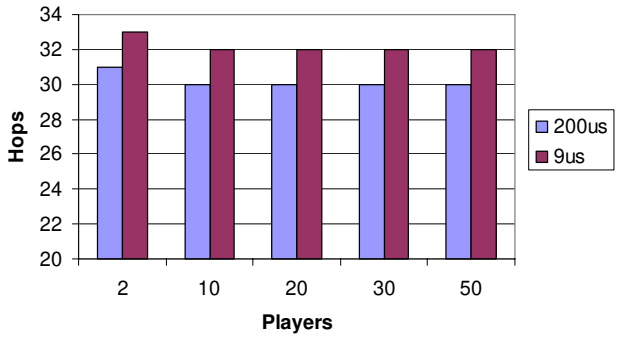


Figure 7.7: Average number of hops required to cover the whole gaming car platoon; FMBP employed, 300 ms of message generation interval for each player, 8 km long vehicular network, 300 m of factual transmission range.

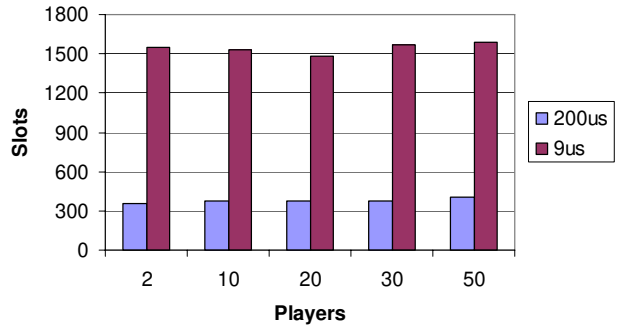


Figure 7.8: Average number of slots cumulatively waited at forwarding vehicles by a message that has to cover the whole gaming car platoon; FMBP employed, 300 ms of message generation interval for each player, 8 km long vehicular network, 300 m of factual transmission range.

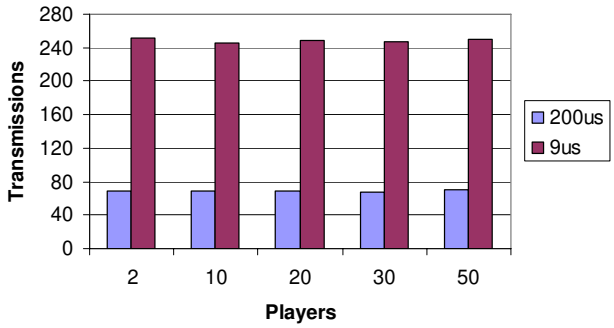


Figure 7.9: Average number of transmissions that a single game event experience while covering the gaming car platoon; FMBP employed, 300 ms of message generation interval for each player, 8 km long vehicular network, 300 m of factual transmission range.

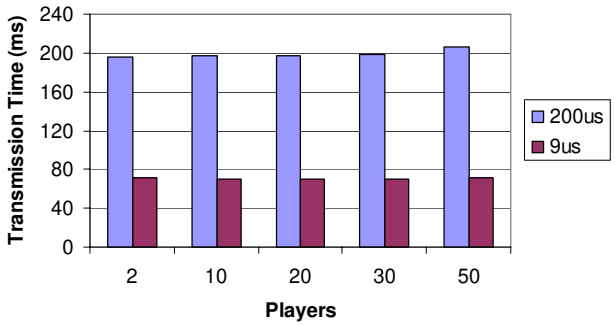


Figure 7.10: Average transmission time required to cover the whole gaming car platoon; FMBP employed, 300 ms of message generation interval for each player, 8 km long vehicular network, 300 m of factual transmission range.

Furthermore, we also evaluated the reliability of the FMBP by verifying the percentage of game messages sent by a vehicle are then received by all the other players in the gaming car platoon. To this aim, Fig. 7.12 shows that with 200 μs slot size, practically all the sent messages are received by all players. Instead, when employing a slot size equal to 9 μs, the reliability of the scheme depends on the amount of traffic in the network; with higher game message generation rate (one every 100 ms) the percentage of messages that are delivered to all players drops to 74%.

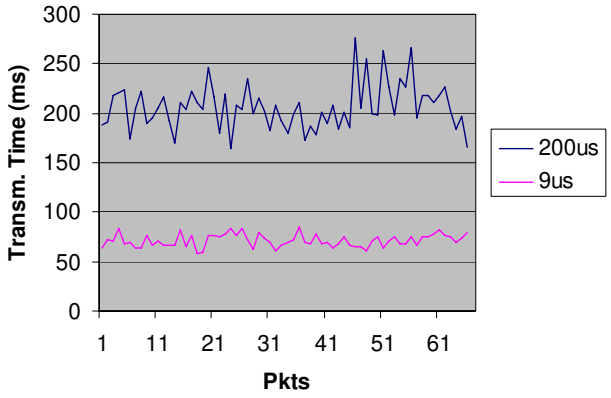


Figure 7.11: Transmission time trend for 50 players by messages that have to cover the whole gaming car platoon; FMBP employed, 300 ms of message generation interval for each player, 8 km long vehicular network, 300 m of factual transmission range.

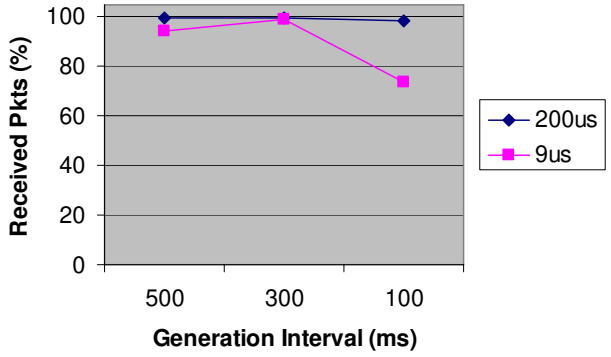


Figure 7.12: Percentage of received game events that succeed in covering the whole gaming car platoon; 50 players with FMBP employed, 300 ms of message generation interval for each player, 8 km long vehicular network, 300 m of factual transmission range.

7.3.3 Comparing FMBP with other Schemes

We are aimed at comparing the performance of FMBP with those achieved by other possible schemes for game event broadcasting. As anticipated, we compare FMBP against Static300 and Static1000. The three schemes are clearly tested under the same

exact conditions: 8 km of vehicular network length, 200 µs of slot duration, 50 simultaneous players, and 300 m of vehicles' transmission range. Different generation rates for game events at each player have been considered. Specifically, game events were generated at each vehicle in the gaming car platoon every 100 ms, 300 ms, or 500 ms.

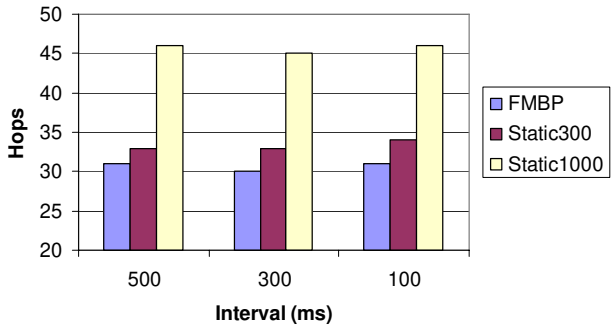


Figure 7.13: Average number of hops required to cover the whole gaming car platoon; 9µs slot, 50 players, 8 km long vehicular network, 300 m of factual transmission range.

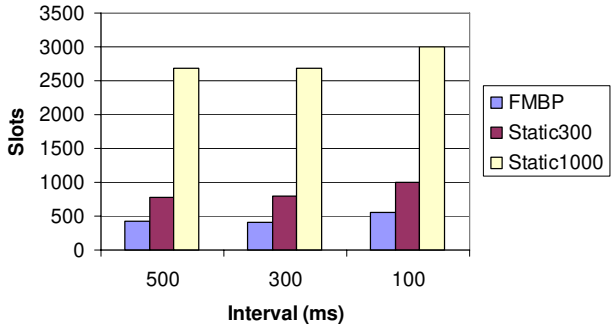


Figure 7.14: Average number of slots cumulatively waited at forwarding vehicles by a message that has to cover the whole gaming car platoon; 9µs slot, 50 players, 8 km long vehicular network, 300 m of factual transmission range.

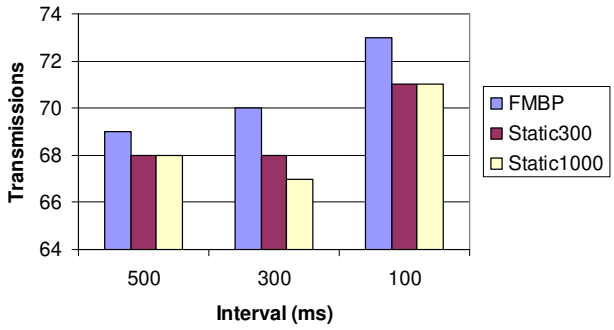


Figure 7.15: Average number of transmissions that a single game event experience while covering the gaming car platoon; 9μs slot, 50 players, 8 km long vehicular network, 300 m of factual transmission range.

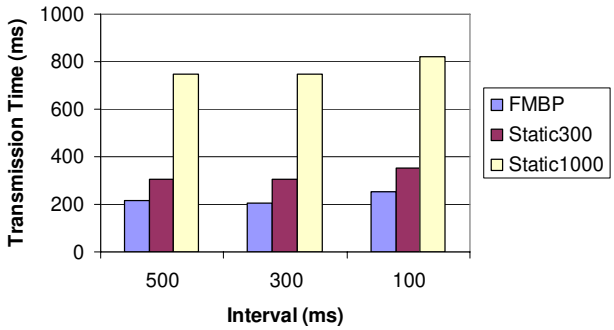


Figure 7.16: Average transmission time required to cover the whole gaming car platoon; 9μs slot, 50 players, 8 km long vehicular network, 300 m of factual transmission range.

Outcomes for the considered metrics are presented in Fig. 7.13 – Fig. 7.16. The first property that emerges is represented by the fact that each of the three schemes achieves similar results with all the three considered message rates. Only with the highest message sending rate (100 ms of inter-departure time) we can observe a little degradation of the performance due to the increased congestion in the network.

The second evident property is that FMBP always obtains better results than the other two schemes, even better than Static300 that is supposed to be the ideal scheme in a scenario with 300 m of transmission range. This exceptional result is not due to any

mistake. Rather, it is due to the fact that even if we have set the transmission range to be 300 m, yet, the adopted wireless model realistically generates interferences as it would happen in real life; these interferences make the factual transmission range oscillate around 300 m. Whereas FMBP dynamically adapts to the changing transmission range conditions to maximize its performance, Static300 is not able to.

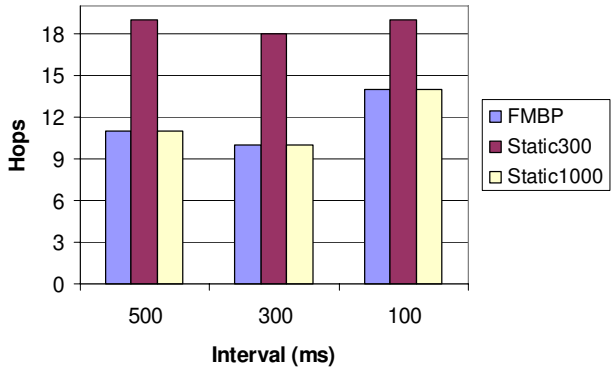


Figure 7.17: Average number of hops required to cover the whole gaming car platoon; 8 km long vehicular network, 50 players, 9µs slot, 1000 m of factual transmission range.

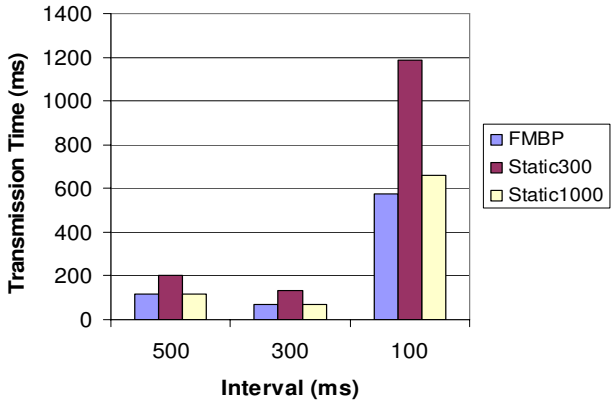


Figure 7.18: Average transmission time required to cover the whole gaming car platoon; 8 km long vehicular network, 50 players, 9µs slot, 1000 m of factual transmission range.

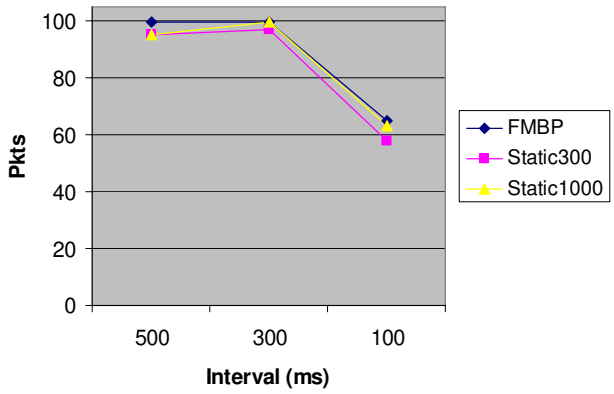


Figure 7.19: Percentage of received game events that succeed in covering the whole gaming car platoon; 8 km long vehicular network, 50 players, 9μs slot, 1000 m of factual transmission range.

To complete our evaluation of FMBP's performance with respect to Static300 and Static1000 we have compared them in a scenario with 1000 m of factual transmission range. The number of hops required to cover all the gaming car platoon is shown in Fig. 7.17. As expected, in this case Static1000 performs much better than Static300; this is due to the fact that with 1000 m of factual transmission range, Static1000 corresponds to the ideal scheme. Yet, without any predetermined knowledge, FMBP succeeds in properly estimating the transmission range for each vehicle and performs as Static1000. The chart also shows that, for all the compared schemes, there is no sensible difference determined by having more or less intense game message generation at each node.

However, if we analyze also the actual transmission time required to propagate a message over the whole gaming car platoon, we notice that if the game application generates a message on each vehicle every 100 ms the total transmission time is considerably higher than the other two considered cases (see Fig. 7.18).

This is clearly due to an excessive increment in the traffic on the wireless channel that causes collisions and time consuming retransmissions. Indeed, Fig. 7.19 confirms this assertion by showing that the case with 100 ms of generation interval for game messages is the only case with a low percentage of game messages delivered to all players.

Finally, we have evaluated the impact of the dispersion of networking vehicles on the

system's performance. Specifically, Fig. 7.20 shows the total transmission time when vehicles that participate in the generation or forwarding of game messages are placed every 20 m (as in all the preceding simulations) or every 80 m. Clearly, FMBP is the only scheme that is not heavily affected by a more dispersed set of networking nodes on which count on to forward the message. This is due to the fact that the simulative transmission range was equal to 300 m, with a factual one often around 280 m. of factual transmission range, having a car every 20 m corresponds to have the farthest car in the factual transmission range exactly at 280 m from the sender; whereas with 80 m of space between successive cars, the farthest one in the factual transmission the transmission will be located at 240 m from the sender. Therefore, Static300 and, in particular way, Static1000 will be utilizing a transmission range parameter wrong with respect to the factual one; whereas FMBP will adapt itself in virtue of its transmission range estimator, correctly computing 240 m.

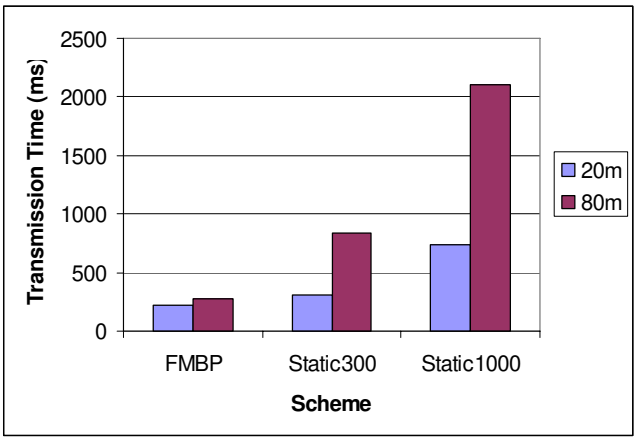


Figure 7.20: Average transmission time required to cover the whole gaming car platoon; 8 km long vehicular network, 50 players, 300 ms of message generation interval, 9 μs slot, 300 m of factual transmission range.

CHAPTER 8

Conclusion

We are living in a world which is faster and faster spinning toward a continuous, ubiquitous, and seamless wireless connectivity that will involve even vehicular scenarios. Customers are more and more attracted by high quality entertainment applications, and mobility represents the next frontier for this kind of technology. On the other hand, real-time applications require continuous connectivity and have stringent delays bounds. MMOG represents an exemplar and very interesting case study among the various entertainment applications that would be soon ubiquitously available.

Several factors concur in creating a pleasant game experience for online players: high responsiveness, uniform view of the game state, and fairness represent the most important of them. We have hence analyzed these issues devoting particular attention to latency issues, since they represent the most crucial prerequisite in developing online games. We have addressed this problem from two standpoints: the server-to-server synchronization and the client-server communication in a congested wireless environment. Both home and vehicular scenarios have been considered and, for the latter, solutions for both infrastructured and ah-hoc connectivity have been proposed and evaluated.

The main contributions of this Thesis can be summarized as follows:

1. Synchronization mechanism for a mirrored game server architecture to support scalability, fairness, and interactivity by exploiting the semantics of the game.
2. Smart AP able to make elastic (e.g., FTP/TCP) and real-time (e.g., games) applications coexist on the same wireless bottleneck without affecting the performance of one another.

3. Fast multi-hop broadcast scheme that exploits a novel transmission range estimator to quickly deliver game events to all players in a certain gaming car platoon.

Through the development of these solutions, other non-technical but even more significant contributions, as they represent general paradigms are:

1. Through our holistic solution for quickly synchronizing game servers, we contradicted the general belief that interactivity and fairness are antithetic objectives. Instead, with the practical example represented by our approach we demonstrated that it is possible to improve (also) fairness by aiming at interactivity.
2. Scientific literature and networking manuals depicts UDP-based flows as harmful toward TCP-based ones as the former generally do not implement any form of congestion control, thus potentially being able to occupy the whole available bandwidth. Instead, we demonstrated that even the viceversa holds: the TCP-based flow can harm the performance of UDP-based ones as TCP continuously probes the channel for more bandwidth, thus eventually generating queues (delays) on the connection.
3. Thanks to our transmission range estimator for VANETs, we have relaxed one of the most common, yet unrealistic, assumptions present in scientific papers related to this issue, which is that the transmission range is a constant for all vehicles known a priori: clearly, it is not.

More in detail, in this Thesis, we have first developed a fast synchronization scheme for a hybrid architecture combining the scalability of peer-to-peer paradigm with the easy management of the client-server one. Our synchronization scheme uses a proactive event discarding mechanism, named ILA, to preserve responsiveness. The proposed mechanism relies on the concept of obsolescence and correlation to drop old game events at GSSs, thus accelerating the processing of fresher ones. We have shown results that demonstrate

the benefits attainable by employing our scheme in place of traditional ones. In particular, with a limited number of obsolete event drops, ILA is able to proactively guarantee a high interactivity degree also maintaining, in the ILA-RED version, full consistency.

Following a holistic approach we have been able to further refine our synchronization mechanism to improve the degree of fairness experienced by players. We have named this upgraded version FILA and we have presented simulation results showing the improvements in terms of game events that were fairly and interactively delivered to destination with respect to traditional Local Lag schemes.

To support engaged players in a wireless and even highly mobile scenario, we have proposed and evaluated solutions able to diminish queuing delays in congested wireless environments. In particular, we have devised SAP-LAW, an easily deployable scheme that exploits enhanced APs and the TCP's advertised window to ensure both goodput efficiency and high responsiveness. The efficacy of our scheme as been evaluated for both a wireless home and an infrastructured vehicular network (urban and highway scenarios).

Finally, as vehicular networks are still far from being supported by an epidemic set of APs specifically deployed to this aim, we have also investigated the scenario of online games played in a VANET. In this context, the best solution to propagate game messages is that of utilizing (multi-hop) broadcast; the longer the broadcast hops, the quicker all players will be reached by each game event.

Our proposal for this issue is that of utilizing priorities so that when a car sends a game event, the farthest car in its transmission range will be the one that will take upon itself the task of forwarding it onto the next hop. We named our scheme FMBP and its more prominent feature is that of employing a novel transmission range estimator for vehicles that works in a distributed way, just exploiting game message exchange, without utilizing control traffic (i.e., hello messages). Through our FMBP, each vehicle becomes aware of how far each game message will go and is hence able to compute its own probability in becoming the next forwarder.

In conclusion, through a holistic approach that covers the game architecture both in the core Internet and in the wireless edges, we have designed and extensively tested a system that is able to maintain a high interactivity degree while achieving also fairness, scalability, and consistency.

8.1 Future Work

Both online gaming and vehicular networks represent interesting and quickly evolving topics. The combination of them certainly provides a fertile soil for the generation of an infinite number of possible extension to this work.

Among the many possibilities, we elaborate on a few potential future studies of prominent interest for the scientific community and that we are already working on.

First, good results in terms of low jitter and per packet delays could be obtained through the utilization of the IEEE 802.11e in place of the regular IEEE 802.11b/g [55, 162]. The IEEE 802.11e defines different classes of traffic and, for each class, diverse configurations of the parameters permit the creation of priorities. These parameters, in fact, determine the various contention delays that packets have to experience before having access to the wireless medium. From the application point of view, video-chats and online games will belong to the highest priority class of traffic. Since video-stream applications could make use of buffering techniques, they could be featured with a medium priority level. Finally, TCP packets could have a low priority level in order to minimize the impact on real time traffic.

It should be said that the IEEE 802.11e needs to have marked packets in order to classify them and it is not clear yet how to implement this functionality in a factually deployable way. Still, it would be interesting to compare the performance of this protocol with those of our SAP-LAW. We deem that not only SAP-LAW is more easily deployable, but it will probably achieve performance comparable with those of IEEE 802.11e.

Automatic IP address configuration in VANETs represents another challenging and almost unexplored issue. The importance of this problem cannot be overemphasized since

any application that involves communication between two or more nodes (even if vehicles) requires the presence of unique identifiers to deliver the data to the right destination. To solve this problem in a vehicular scenario, we have proposed a novel scheme that exploits the topology of vehicular ad-hoc networks and an enhanced DHCP service with dynamically elected leaders to guarantee reliable and fast address configuration [199, 201, 202]. It would now be interesting to evaluate this scheme with an online gaming application running on top of it to evaluate its factual utilization with online games.

Finally, wireless mesh networks represent an interesting evolution of wireless architecture able to merge the quality of the infrastructure with those of ad-hoc connectivity. Mesh networks can be utilized to deliver services for a large variety of applications in personal, local, campus, and metropolitan areas. Unlike WLANs, mesh networks are self-configuring systems where each Access Point (AP) can relay messages on behalf of others, thus increasing the range and the available bandwidth. Key advantages of wireless mesh networks include ease of installation, no cable cost, automatic connection among nodes, network flexibility, discovery of newly added nodes, redundancy, and self-healing reliability. Moreover, they represents a hybrid architecture framework where we could join our SAP-LAW with FMBP and more. We are hence motivated in investigating performances, problems, and solutions that involves online games in this context.

REFERENCES

[1] The Network Simulator, NS-2. http://www.isi.edu/nsnam/ns/

[2] QualNet Network Simulator. http://www.qualnet.com/

[3] C.H. Nam, Soung C. Liew, C.P. Fu, “An Experimental Study of ARQ Protocol in 802.11b Wireless LAN”, in Proc. of IEEE VTC 2003, 2003.

[4] G. Xylomenos and G. C. Polyzos, “TCP and UDP Performance over a Wireless LAN”, in Proc. of IEEE INFOCOM ’99, 1999.

[5] C. E. Palazzi, C. Roseti, M. Luglio, M. Gerla, M. Y. Sanadidi, and J. Stepanek, “Satellite coverage in urban areas using Unmanned Airborne Vehicles (UAVs)”, in Proc. of VTC2004 Spring, Milan, Italy, May 2004.

[6] C. E. Palazzi, C. Roseti, M. Luglio, M. Gerla, M. Y. Sanadidi, and J. Stepanek, “Enhancing Transport Layer Capability in HAPS-Satellite Integrated Architecture”, Wireless Personal Communications International Journal, Special Issue on "High Altitude Platforms: Research and Application Activities", Kluwer Academic Publishers, 2004.

[7] C. Griwodz, “State Replication for Multiplayer Games”, in Proc. of NetGames2002, pp. 29-35, Braunschweig, Germany, 2002.

[8] S. Mascolo, A. Grieco, G. Pau, M. Gerla, C. Casetti, “End-to-End Bandwidth Estimation in TCP to Improve Wireless Link Utilization”, European Wireless Conference, Florence, Italy, Feb 2002.

[9] M. Roccetti, V. Ghini and P. Salomoni, “Distributing Music from IP Networks to UMTS Terminals: an Experimental Study", in Proc. 2002 SCS Euromedia Conference (EUROMEDIA2002), (M. Roccetti Ed.) The Society for Modeling and Simulation International, Modena (Italy), pp. 147-154, Apr 2002.

[10] V. Ghini, G. Pau, M. Roccetti, P. Salomoni, M. Gerla, “For Here or To Go? Downloading Music on the Move with an Ultra Reliable Wireless Internet Application”, IEEE ICC’2004, Paris, 2004.

[11] UMTS Forum. http://www.umts-forum.org

[12] J. Weatherall and A. Jones, “Ubiquitous Networks and their applications”, in IEEE Wireless Communication, Feb 2002.

[13] E. Gustafsson and A. Jonsson, “Always Best Connected”, IEEE Wireless Communications, pp. 49-55, Feb 2003.

[14] G. Huston, “The future for TCP”, The Internet Protocol Journal, vol. 3, n. 3, Sep 2000.

[15] G. Huston, “TCP in a Wireless World”, IEEE Internet Computing, pp. 82 – 84, Mar-Apr 2001.

[16] Mattias Esbjörnsson, Oskar Juhlin and Mattias Östergren, “The Hocman Prototype – Fast Motor Biders and Ad-hoc Networking”, in Proc. of MUM, 2002.

[17] H. Balakrishnan, V. N. Padmanabhan, S. Sehan, and R. H. Katz, "A comparison of mechanism for improving TCP performance over wireless links", IEEE/ACM Trans. Networking, vol. 5, no. 6, pp. 756 – 769, Dec 1997.

[18] M. Mauve, J. Widmer and H. Hartenstein, “A Survey on Position-Based Routing in Mobile Ad Hoc Networks”, IEEE Network, vol. 15, no. 6., pp. 30-39, 2001.

[19] J. Ott and D. Kutscher, “Drive-thru Internet: IEEE 802.11b for Automobile Users”, In Proc. of IEEE INFOCOM 2004, Mar 2004.

[20] A. Nandan, S. Das, G. Pau., M. Gerla, M. Y. Sanadidi, “Co-operative Downloading in Vehicular Ad-hoc Wireless Networks”, in Proc. of WONS 2005, Saint Moritz, Switzerland, Jan 2005.

[21] H. Luo, R. Ramjee, P. Sinha, L. Li, S. Lu, "UCAN: A Unified Cellular and Ad-Hoc Network Architecture", in Proc. of ACM MobiCom 2003, San Diego, CA, Sep 2003.

[22] Q. Xu, T. Mak, J. Ko, R. Sengupta, "Vehicle-to-Vehicle Safety Messaging in DSRC", in Proc. of ACM VANET'04, Philadelphia, PA, Oct 2004.

[23] B. Hughes, R. Meier, R. Cunningham, V. Cahill, "Towards Real-Time Middleware for Vehicular Ad Hoc Networks", in Proc. of ACM VANET'04, Philadelphia, PA, Oct 2004.

[24] Mobile Monday. http://www.mobilemonday.net/mm/story.php?story_id=3748

[25] MMOGCHART. http://www.mmogchart.com/

[26] D. Kushner, Masters of Doom, Random House, New York, NY, 2002.

[27] J. Smed, T. Kaukoranta, H. Hakonen, "Aspects of Networking in Multiplayer Computer Games", in Proc. of International Conference on Application and Development of Computer Games in the 21st Century, pp.74-81, Hong Kong, China, 2001.

[28] M. R. Macedonia, A Network Software Architecture for Large Scale Virtual Environments, Ph.D. Thesis, Naval Postgraduate School, Monterey, CA, 1995.

[29] D. L. Neyland Virtual Combat: A Guide to Distributed Interactive Simulation, Stackpole Books, Mechanicsburg, PA, 1997.

[30] V. Normand, "The COVEN project: Exploring Applicative, Technical, and Usage Dimensions of Collaborative Virtual Environments", Presence, vol.8, no.2, pp. 218-236, 1999.

[31] E. Frécon, M. Stenius, "DIVE: A Scaleable Network Architecture for Distributed Virtual Environments", Distributed Systems Engineering, vol. 5, no. 3, pp. 91-100, 1998.

[32] T. A. Funkhouser, “RING: A Client-Server System for Multi-User Virtual Environments”, in Proc. of the 1995 Symposium on Interactive 3D Graphics, pp. 85-92, Monterey, CA, 1995.

[33] Nindtendo DS. http://www.nintendo.com/ds/index.jsp

[34] Sony PSP. http://www.us.playstation.com/pressreleases.aspx?id=207

[35] Quake Forge Project. http://www.quakeforge.org/

[36] R. M. Mine, J. Shochet, R. Hughston, “Building a Massively Multiplayer Game for the Million: Disney's Toontown Online”, ACM Computers in Entertainment (CIE), vol. 1, no. 1, pp. 15-15, 2003.

[37] P. Mehta and S. Udani. “Voice over IP”, in IEEE Potentials, vol. 20, no. 4, Oct 2001.

[38] W. Cai, P. Xavier, S. J. Turner, B. Lee, “A Scalable Architecture for Supporting Interactive Games on the Internet”, in Proc. of the 16th Workshop on Parallel and Distributed Simulation, pp. 54-61, Washington, DC, 2002.

[39] S. Singhal, M. Zyda, Networked Virtual Environments: Design and Implementation, Addison Wesley, 1999.

[40] T. Kanter and C. Olrog, “VoIP in Applications for Wireless Access”, in IEEE Workshop on Local and Metropolitan Area Networks, Nov 1999.

[41] K. Svanbro, J. Wiorek, and B. Olin, “Voice-over-IP-over-wireless”, in IEEE PIMRC '00, Sep 2000.

[42] C. E. Palazzi, “Buddy-Finder: A Proposal for a Novel Entertainment Application for GSM”, in Proc. of the 1st IEEE International Workshop on Networking Issues in Multimedia Entertainment (NIME'04), GLOBECOM 2004, Dallas, TX, Nov 2004.

[43] C. E. Palazzi, “Residual Capacity Estimator for TCP on wired/wireless links”, in Proc. of WCC2004-Student-Forum IFIP World Computer Congress 2004, Tolouse, France, Aug 2004.

[44] S. Tekinay and B. Jabbari, “Handover and channel assignment in mobile cellular networks”, in IEEE Commununication Magazine, vol. 29, no, 11, pp. 42-46, Nov 1991.

[45] N. Davies, K. Cheverst, A. Friday and K.Mitchell, “Future Wireless Applications for a Networked City: Services for Visitors and Residents”, Feb 2002.

[46] V. Ghini, G. Pau, M. Roccetti, P. Salomoni and M. Gerla, “Smart Download on the Go: A Wireless Internet Application for Music Distribution over heterogeneous Networks”, in Proc. IEEE International Conference on Communications - Access and Home Networks Symposium (ICC'04), accepted for publication, Paris, France, IEEE Communications Society, Jun 2004.

[47] M. Roccetti and P. Salomoni, “The Design and Performance of a Wireless Internet Application for Supporting Multimedia City Guides”, in Proc. of IEEE International Conference on Information Technology: Research and Education (ITRE 2003), Special Session on Multimedia Transport in Heterogeneous Wireless Networks, Newark (USA), Aug 2003.

[48] M. Stemm and R. H. Katz. “Vertical handoffs in wireless overlay networks”, in Mobile Networks and Applications, vol. 3, no. 4, 1998.

[49] K. Wang, S.K. Tripathi, "Mobile-End Transport Protocols: An Alternative to TCP/IP Over Wireless Links", in Proc. IEEE Infocom ‘98, San Francisco, California, USA, pp. 1046-1053, Apr 1998.

[50] M. Stemm and R. H. Katz, "Vertical Handoffs in Wireless Overlay Networks", ACM Mobile Networks and Applications (MONET) Journal, Special Issue on Mobile Networking in the Internet, 1998.

[51] K. Brown, S. Singh, “M-TCP: TCP for Mobile Cellular Networks”, ACM Computer Communication Review, vol. 27, no. 5, Jul 1997.

[52] J. S. Steinman, R. Bagrodia, D. Jefferson, “Breathing Time Warp”, in Proc. of the 1993 Workshop on Parallel and Distributed Simulation, San Diego, CA pp. 109-118, 1993.

[53] D. R. Jefferson, “Virtual Time”, ACM Transaction on Programming Languages and Systems, vol. 7, no. 3, pp. 404-425, 1985.

[54] S. J. Kim, F. Kuester, K. H. Kim “A Global Timestamp-based Scalable Framework for Multi-player Online Games”, in Proc. of the IEEE Fourth International Symposium on Multimedia Software Engineering (MSE’02), 2002.

[55] H. Yoon, J. W. Kim, D. Y. Shin, “Dynamic Admission Control in IEEE 802.11e EDCA-based Wireless Home Network”, in Proc. of IEEE Consumer Communications and Networking Conference (CCNC 2006), Las Vegas, NV, Jan 2006.

[56] M. Gerla, K. Tang, R. Bagrodia, “TCP Performance in Wireless Multi-hop Networks”, in Proc. of the 2nd IEEE Workshop on Mobile Computer Systems and Applications (WMCSA ’99). Washington, DC, USA, p. 41-50, 1999.

[57] M. Heusse, F. Rousseau, G. Berger-Sabbatel, A. Duda, “Performance Anomaly of 802.11b”, in Proc. of IEEE INFOCOM 2003, San Francisco, CA, USA, Apr 2003.

[58] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, M. Claypool, “The Effects of Loss and Latency on User Performance in Unreal Tournament 2003”, in Proc. of ACM Network and System Support for Games Workshop (NetGames), Portland, OG, USA, Sep 2004.

[59] E. Cronin, A. R. Kurc, B. Filstrup, S. Jamin, “An Efficient Synchronization Mechanism for Mirrored Game Architectures”, Multimedia Tools and Applications, vol. 23, no. 1, pp. 7-30, 2004.

[60] D. R. Cheriton, D. Skeen, “Understanding the Limitations of Causal and Totally Ordered Multicast”, in Proc. of the 14th Symposium on Operating System Principles (SOSP '93), pp. 44-57, Asheville, NC, 1993.

[61] X. Défago, A. Schiper, P. Urban, “Totally Ordered Broadcast and Multicast Algorithms: a Comprehensive Study”, Technical Report, DSC/2000/036, Swiss Federal Ecole Politechnique Fédérale de Lausanne, Switzerland, 2000.

[62] C. E. Palazzi, S. Ferretti, S. Cacciaguerra, M. Roccetti, “On Maintaining Interactivity in Event Delivery Synchronization for Mirrored Game Architectures”, in Proc. of 1st IEEE International Workshop on Networking Issues in Multimedia Entertainment (NIME'04), GLOBECOM 2004, Dallas, TX, 2004.

[63] S. Ferretti, M. Roccetti, “A Novel Obsolescence-based Approach to Event Delivery Synchronization in Multiplayer Games”, International Journal of Intelligent Games and Simulation, vol. 3, no. 1, pp. 7-19, 2004.

[64] S. Ferretti, M. Roccetti, S. Cacciaguerra, “On Distributing Interactive Storytelling: Issues of Event Synchronization and a Solution”, in Proc. of the 2nd International Conference on Technologies for Interactive Digital Storytelling and Entertainment (TIDSE 2004), LNCS 3105, Darmstadt, Germany, pp. 219-231, 2004.

[65] R. Drummond, O. Babaoglu, “Low-Cost Clock Synchronization”, Distributed Computing, vol. 6, no. 3, pp. 193-203, 1993.

[66] F. Cristian, “Probabilistic clock synchronization”, Distributed Computing, vol. 3, no. 3, pp. 146-158, 1989.

[67] D. L. Mills, “Internet Time Synchronization: the Network Time Protocol”, IEEE Transactions on Communications, vol. 39, no. 10 pp. 1482-1493, 1991.

[68] S. Floyd, V. Jacobson, “Random Early Detection Gateways for Congestion Avoidance”, IEEE/ACM Transactions on Networking, vol. 1, no. 4, pp. 397-413, 1993.

[69] C. E. Palazzi, S. Ferretti, S. Cacciaguerra, M. Roccetti, “A RIO-like Technique for Interactivity Loss Avoidance in Fast-Paced Multiplayer Online Games: a Preliminary Study”, in Proc. of the 2nd Annual International Workshop in Computer Game Design and Technology (GDTW 2004), Liverpool, UK, 2004.

[70] D. D. Clark, W. Fang, “Explicit Allocation of Best-Effort Packet Delivery Service”, IEEE/ACM Transactions on Networking, vol.6, no.4, pp.362-373, 1998.

[71] M. Mauve, Distributed Interactive Media, PhD thesis, University of Mannheim, ISBN 3-89838-471-3. infix., Berlin, 2000.

[72] K. Park, W. Willinger, Self-Similar Network Traffic and Performance Evaluation, Wiley-Interscience, 1st Edition, 2000.

[73] J. Färber, “Network Game Traffic Modelling” in Proc. of NetGames2002, pp.53-57, Braunschweig, Germany, 2002.

[74] Half-Life. http://counterstrike.sierra.com/

[75] S. Wright, S. Tischer, “Architectural Considerations in Online Game Services over DSL Networks”, in Proc. IEEE International Conference on Communications - (ICC'04), IEEE Communications Society, Paris, France, 2004.

[76] G. Armitage, “An Experimental Estimation of Latency Sensitivity in Multiplayer Quake 3”, in Proc. of ICON, pp. 137-141, Sydney, Australia, 2003.

[77] M. S. Borella, “Source Models for Network Game Traffic”, Computer Communications, vol. 23, no. 4, pp. 403-410, 2000.

[78] L. Pantel, L. C. Wolf, “On the Impact of Delay on Real-Time Multiplayer Games”, in Proc of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video, Miami, FL, May 2002.

[79] T. Henderson, "Latency and User Behaviour on a Multiplayer Game Server", in Proc. of the 3rd International Workshop on Networked Group Communication (NGC01) pp. 1-13, London, UK, Nov 2001.

[80] F. Fitzek, G. Schulte, M. Reisslein, “System Architecture for Billing of Multi-Player Games in a Wireless Environment Using GSM/UMTS and WLAN Services”, in Proc. of NetGames2002, pp. 58-64, Bruanschweig, Germany, Apr 2002.

[81] C. E. Palazzi, S. Ferretti, S. Cacciaguerra, M. Roccetti, “Interactivity-Loss Avoidance in Event Delivery Synchronization for Mirrored Game Architectures”, IEEE Transactions on Multimedia, IEEE Signal Processing Society, vol. 8, no. 4, pp. 847-879, Aug 2006.

[82] Mesh Meets Ad Hoc: the Urban Vehicular Grid. http://www.cs.ucla.edu/ST/

[83] Methods for Subjective Determination of Transmission Quality, ITU-T Recommendation P.800, Aug 1996.

[84] A. Rix, R. Reynolds, M. Hollier, “Perceptual Measurement of End-to-End Speech Quality over Audio and Packet-Based Networks”, AES 106th Convention, Munich, Germany, May 1999.

[85] Methods for the Subjective Assessment of Small Impairments in Audio Systems Including Multichannel Sound Systems, ITU-R Recommendation BS.1116, Jul 1998.

[86] Method for Objective Measurements of Perceived Audio Quality, ITU-R Recommendation BS.1387, Jan 1999.

[87] A. Rix, R. Reynolds, M. Hollier, “Robust Perceptual Assessment of End-to-End Audio Quality”, in Proc. of 1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, NY, Oct 1999.

[88] M. Handley et al., “SIP: Session Initiation Protocol”, IETF RFC 2543, Mar 1999.

[89] H. Hartenstein, B. Bochow, A. Ebner, M. Lott, M. Radimirsch, D. Vollmer, “Position-Aware Ad Hoc Networks for Inter-vehicle Communications: The Fleenet

Project", in ACM Symposium on Mobile Ad Hoc Networking and Computing, MobiHOC, Oct 2001.

[90] Federal Communications Commission, FCC 99-305, FCC Report and Order, Oct 1999.

[91] Federal Communications Commission, FCC 03-024, FCC Report and Order, Feb 2004.

[92] J. Jun and M. L. Sichitiu, "The Nominal Capacity of Wireless Mesh Networks", IEEE Wireless Communications magazine, Oct 2003.

[93] I. Foster and C. Kesselman, editors. The Grid: Blueprint for a New Computing Infrastructure. Morgan-Kaufmann, 1999.

[94] L. Gautier, C. Diot, "Design and Evaluation of MiMaze, a Multi-player Game on the Internet", in Proc. of IEEE Multimedia (ICMCS'98), pp. 234-236 Austin, TX, USA, 1998.

[95] Everquest. http://www.everquest.com

[96] Ultima Online. http://www.uo.com

[97] Butterfly Grid Solution for Online Games. http://www.butterfly.net

[98] J. S. Steinman, "Scalable Parallel and Distributed Military Simulations Using the SPEEDES Framework", in Proc. of 2nd Electronic Simulation Conference (ELECSIM95), Internet, 1995.

[99] A. Balk, M. Gerla, M. Sanadidi, D. Maggiorini "Adaptive Video Streaming: Pre-encoded MPEG-4 with Bandwidth Scaling", Computer Networks vol. 44, no. 4, pp. 415-439, 2004.

[100] C. Boyd, A. Mathuria, "Key Establishment Protocols for Secure Mobile Communications: a Critical Survey", Computer Communications, vol. 23, pp. 575-587, 2000.

[101] P. Mohapatra, J. Li, C. Gui, "QoS in Mobile Ad Hoc Networks", IEEE Wireless Communications magazine, Jun 2003.

[102] IEEE Standard for Information Technology, "Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks - Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment: Medium Access Control (MAC) Quality of Service Enhancements", P802.11e/D13.0, 2005.

[103] S. Cacciaguerra, M. Roccetti, M. Roffilli, A. Lomi, "A Wireless Software Architecture for Fast 3D Rendering of Agent-Based Multimedia Simulations on Portable Devices" in Proc. of the First Consumer Communications and Networking Conference (CCNC), IEEE Communications Society, Las Vegas, NV, Jan 2004.

[104] S. Zander, I. Leeder, G. Armitage, "Achieving Fairness in Multiplayer Network Games through Automated Latency Balancing", in Proc. of ACM SIGCHI ACE2005, pp. 117-124, Valencia, Spain, 2005.

[105] M. Mauve, J. Vogel, V. Hilt, W. Effelsberg, "Local-lag and Timewarp: Providing Consistency for Replicated Continuous Applications", IEEE Transactions on Multimedia, vol. 6, no. 1, pp. 47-57, 2004.

[106] F. Fitzek, G. Schulte, M. Reisslein, "System Architecture for Billing of Multi-Player Games in a Wireless Environment Using GSM/UMTS and WLAN Services", in Proc. of the 1st Workshop on Network and System Support for Games (NetGames2002), ACM SIG MM, pp. 58-64, Braunschweig, Germany, 2002.

[108] Sheldon N., Girard E., Borg S., Claypool M., Agu E. The Effect of Latency on User Performance in Warcraft III. In Proceedings of the 2nd Workshop on Network and System Support for Games (NetGames 2003), ACM SIGCOMM and SIG MM, pp. 3-14, Redwood City, CA, USA, 2003.

[109] S. Ferretti, C. E. Palazzi, M. Roccetti, M. Gerla, G. Pau, "*Buscar el Levante por el Poniente:* In Search of Fairness Through Interactivity in Massively Multiplayer

Online Games", in Proc. of the 2nd IEEE International Workshop on Networking Issues in Multimedia Entertainment (NIME'06), CCNC 2006, Las Vegas, NV, Jan 2006.

[110] T. Jehaes, D. De Vleeschauwer, T. Coppens, B. Van Doorselaer, E. Deckers, W. Naudts, K. Spruyt, R. Smets, "Access Network Delay in Networked Games", in Proc. of the 2nd Workshop on Network and System Support for Games (NetGames 2003), ACM SIGCOMM and SIG MM, pp. 63-71, Redwood City, CA, USA, 2003.

[111] P. Ramanathan, K. G. Shin, R. W. Butler, "Fault Tolerant Clock Synchronization in Distributed Systems", IEEE Computer, vol. 23, no. 10, pp. 33-42, 1990.

[112] F. Safaei, P. Boustead, C. D. Nguyen, J. Brun, M. Dowlatshahi, "Latency Driven Distribution: Infrastructure Needs of Participatory Entertainment Applications", IEEE Communications Magazine, IEEE Communications Society, Special Issue on "Entertainment Everywhere: System and Networking Issues in Emerging Network-Centric Entertainments Systems", Part I, May 2005.

[113] Skype for Pocket PC. http://www.skype.com/products/skype/pocketpc/

[114] A. Nandan, S. Das, B. Zhou, G. Pau, M. Gerla, "AdTorrent: Digital Billboards for Vehicular Networks", in Proc. of IEEE/ACM International Workshop on Vehicle-to-Vehicle Communications (V2VCOM), San Diego, CA, USA, Jul 2005.

[115] A. Nandan, S. Das, G. Pau, M.Y. Sanadidi, M. Gerla "Cooperative Downloading in Vehicular Ad Hoc Wireless Networks", in Proc. of IEEE/IFIP International Conference on Wireless On demand Network Systems and Services, St. Moritz, Switzerland, pp. 32-41, Jan 2005.

[116] S. Das, A. Nandan, G. Pau, M.Y. Sanadidi, M. Gerla "SPAWN: Swarming Protocols for Vehicular Ad Hoc Wireless Networks", in Proc. of the First ACM International Workshop on Vehicular Ad Hoc Networks (VANET 2004), MOBICOM 2004, Berkeley, CA, USA, pp. 93-94, 2004.

[117] S. Moyer, D. Marples, S. Tsang, and A. Ghosh, "Service Portability of Networked Appliances," 2002. [Online]. Available at: http://citeseer.ist.psu.edu/moyer00service.html

[118] A. Mani, H. Sundaram, D. Birchfield, and G. Qian, "The Networked Home as a User-Centric Multimedia System," in NRBC '04: Proceedings of the 2004 ACM Workshop on Next-Generation Residential Broadband Challenges. New York, NY, USA: ACM Press, pp. 19-30, 2004.

[119] G. Bell, J. Gemmell, "A Call for the Home Media Network", Communications of the ACM, vol. 45, no. 7, pp. 71-75, 2002.

[120] M. Tsang, G. Fitzmaurice, G. Kurtenbach, A. Khan, "Game-Like Navigation and Responsiveness in Non-Game Applications", Communications of the ACM, vol. 46, no. 7, pp. 56-61, Jun 2003.

[121] A. E. Cha, "Recalling Iraq's Terrors Through Virtual Reality", Washington Post, Mar 2005. http://www.washingtonpost.com/wp-dyn/articles/A58360-2005Mar22.html

[122] S. Frizzoli, Coesistenza tra Applicazioni Elastiche e Real-time nelle Reti Veicolari con Infrastruttura, Laurea Thesis, Dipartimento di Scienze dell'Informazione, Università di Bologna, Mar 2007.

[123] E. Manca, Broadcast Veloce di Eventi di Gioco nelle Reti Veicolari Utilizzando il Fast Multi-Broadcast Protocol, Laurea Thesis, Dipartimento di Scienze dell'Informazione, Università di Bologna, Mar 2007.

[124] F. Parmeggiani, Fast Multi-Broadcast Protocol: Stima del Raggio Trasmissivo per la Propagazione Veloce di Pacchetti Broadcast, Laurea Thesis, Dipartimento di Scienze dell'Informazione, Università di Bologna, Mar 2007.

[125] S. Ferretti, M. Roccetti, "Event Synchronization for Interactive Cyberdrama Generation on the Web: a Distributed Approach", in Proc. of 13th International

World Wide Web Conference (WWW 2004), vol. WWW2004 Poster Track, W3C/ACM editor, New York, NY, USA, May 2004.

[126] T. M. Rhyne, "Computer Games and Scientific Visualization", Communications of the ACM, vol. 45, no. 7, pp. 40-44, Jul 2002.

[127] J. Brun, P. Boustead, F. Safaei, "Fairness and Playability in Online Multiplayer Games", in Proc. of the 2nd IEEE International Workshop on Networking Issues in Multimedia Entertainment (NIME'06), CCNC 2006, Las Vegas, NV, Jan 2006.

[128] C. Wagner, M. Schill, R. Manner, "Intraocular Surgery on a Virtual Eye", Communications of the ACM, vol. 45, no. 7 pp. 45-49, Jul 2002.

[129] J. Kumagai, "Fighting in the streets", IEEE Spectrum, vol. 38, no. 2, pp. 68-71, Feb 2001.

[130] The Eight Fallacies of Distributed Computing. http://today.java.net/jag/Fallacies.html

[131] M. Mauve, S. Fischer, J. Widmer, "A Generic Proxy System for Networked Computer Games", in Proc. of the 1st Workshop on Network and System Support for Games, pp. 25-28, 2002.

[132] C. Diot, L. Gautier, "A Distributed Architecture for Multiplayer Interactive Applications on the Internet", IEEE Network Magazine, vol. 13, no. 4, Jul/Aug 1999.

[133] K. W. Lee, B. J. Ko, S. Calo, "Adaptive Server Selection for Large Scale Interactive Online Games", in Proc. of the 14th International Workshop on Network and Operating Systems Support for Digital Audio and Video, pp. 152-157. 2004.

[134] K. Alex, S. Taylor, "Using Determinism to Improve the Accuracy of Dead Reckoning Algorithms", in Proc. of Simulation Technologies and Training Conference, Sydney, 2000.

[135] L. Pantel, L. C. Wolf, "On the Suitability of Dead Reckoning Schemes for Games", in Proc. of the 1st Workshop on Network and System Support for Games, pp. 79-84, 2002.

[136] B. Knutsson, H. Lu, W. Xu, B. Hopkins, "Peer-to-Peer Support for Massively Multiplayer Games", in Proc. of the Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004), pp. 96-107, Mar 2004.

[137] F. W. B. Li, L. W. F. Li, R. W. H. Lau, "Supporting Continuous Consistency in Multiplayer Online Games", in Proc. of the 12th Annual ACM International Conference on Multimedia, pp. 388-391, 2004.

[138] J. Postel, "Rfc0793: Transmission Control Protocol," Internet Engineering Task Force (IETF), Tech. Rep., 1981.

[139] V. Jacobson, "Congestion Avoidance and Control", in Proc. of ACM SIGCOMM'88, pp. 314-329, 1988.

[140] S. Floyd, "A Report on Recent Developments in TCP Congestion Control", IEEE Communications, vol. 39, no. 4, pp. 84-90, 2001.

[141] Y. M. Wang, W. Russell, A. Arora, R. K., Jagannathan, J. Xu, "Towards Dependable Home Networking: an Experience Report", in Proc. of IEEE International Conference on Dependable Systems and Networks (DSN 2000), New York, USA, pp. 43-49, Jun 2000.

[142] IEEE, "Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (Phy) Specifications," Specifications, ISO/IEC 8802-11:1999(E), 1999.

[143] S. Low, L. Peterson, L. Wang, "Understanding Vegas: a Duality Model," Journal of ACM, vol. 49, no. 2, pp. 207-235, Mar 2002.

[144] L. Brakmo, S. O'Malley, L. Peterson, “TCP Vegas: New Techniques for Congestion Detection and Avoidance”, in Proc. of the SIGCOMM '94 Symposium, pp. 24-35, Aug 1994.

[145] Movie Trace Files. http://www-tkn.ee.tu-berlin.de/research/trace/ltvt.html

[146] J. Färber, “Traffic Modelling for Fast Action Network Games,” Multimedia Tools and Applications, vol. 23, no. 1, pp. 31-46, 2004.

[147] J. Postel, “User Datagram Protocol”, Internet Engineering Task Force (IETF), RFC 768, Aug 1980.

[148] G. Marfia, C. E. Palazzi, G. Pau, M. Gerla, M. Y. Sanadidi, M. Roccetti, "TCP Libra: Exploring RTT-Fairness for TCP", UCLA CSD Technical Report #TR050037, 2005.

[149] Consumer Electronics - IT Facts. http://www.itfacts.biz/index.php?id=P1861

[150] InternetFrog.com. http://www.internetfrog.com/mypc/speedtest/

[151] Bandwidth Speed Test. http://www.bandwidthplace.com/speedtest/

[152] Broadband Speed Tests. http://www.dslreports.com/stest

[153] Verizon Online DSL. http://www.verizon.com/dsl/

[154] Bellsouth Fastaccess DSL https://www.fastaccess.com/content/splash/

[155] AT&T Worldnet DSL Service. http://www.att.net/dsl/

[156] H. Jiang, C. Dovrolis, “Why is the Internet traffic bursty in short (sub-RTT) time scales?”, in Proc. of ACM SIGMETRICS 2005, Banff, AL, Canada, 2005.

[157] C. E. Palazzi, G. Pau, M. Roccetti, M. Gerla, “Digital Entertainment Delivery in a Wireless House: Time for a MAC Tuning”, China Communications, CIC, vol.4, no.5 Oct 2006, 94-101.

[158] A. Balk, M. Gerla, M. Sanadidi, D. Maggiorini, “Adaptive Video Streaming: Pre-encoded MPEG-4 with Bandwidth Scaling”, Computer Networks: The International Journal of Computer and Telecommunications Networking, vol. 44, no. 4, pp. 415-439, 2004.

[159] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, “Modeling TCP Throughput: A Simple Model and its Empirical Validation”, in Proc. of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, pp. 303-314, Vancouver, B.C., Canada, 1998.

[160] C. Jin, D. Wei, S. H. Low, G. Buhrmaster, J. Bunn, D. H. Choe, R. L. A. Cottrell, J. C. Doyle, W. C. Feng, O. Martin, H. Newman, F. Paganini, S. Ravot, and S. Singh, “Fast TCP: From Background Theory to Experiments,” IEEE Network, vol. 19, no. 1, pp. 4-11, 2005.

[161] D.-Y. Chen, S. Garg, M. Kappes, K. S. Trivedi, “Supporting VBR VoIP Traffic with IEEE 802.11 WLAN in PCF Mode”, in Proc. of OPNETWork 2002, Washington D.C., 2002.

[162] D. Chen, D. Gu, J. Zhang, “Supporting Real-time Traffic with QoS in IEEE 802.11e Based Home Networks”, Mitsubishi Electric Research Laboratory, TR-2004-006, 2004.

[163] C. E. Palazzi, S. Ferretti, S. Cacciaguerra, M. Roccetti, “A RIO-like Technique for Interactivity Loss Avoidance in Fast-Paced Multiplayer Online Games”, ACM Journal of Computers in Entertainment, vol.3, no.2, Apr 2005.

[164] S. Ferretti, C. E. Palazzi, M. Roccetti, G. Pau, M. Gerla, “FILA, a Holistic Approach to Massive Online Gaming: Algorithm Comparison and Performance Analysis”, in Proc. of the 3rd ACM International Conference in Computer Game Design and Technology (GDTW 2005), Liverpool, UK, pp. 68-76, Nov 2005.

[165] S. Ferretti, C. E. Palazzi, M. Roccetti, G. Pau, M. Gerla, "FILA in Gameland, a Holistic Approach to a Problem of Many Dimensions", in ACM Journal of Computer in Entertainment, ACM Press, vol. 4 , no. 4, Oct-Dec 2006.

[166] C. E. Palazzi, G. Pau, M. Roccetti, M. Gerla, "In-Home Online Entertainment: Analyzing the Impact of the Wireless MAC-Transport Protocols Interference", in Proc. of IEEE International Conference on Wireless Networks, Communications and Mobile Computing (WIRELESSCOM 2005), Maui, HI, USA, Jun 2005.

[167] C. Gauthier Dickey, D. Zappala, V. Lo, J. Marr, "Low Latency and Cheat-Proof Event Ordering for Peer-to-Peer Games", in Proc. of ACM International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), Kinsale, County Cork, Ireland, pp. 134-139, Jun 2004.

[168] N. E. Baughman, B. N. Levine, "Cheat-proof Playout for Centralized and Distributed Online Games", in Proc. of IEEE Infocom, Anchorage, AK, USA, pp. 104-113, Apr 2001.

[169] Gamasutra. http://www.gamasutra.com/features/20000724/pritchard_01.htm

[170] B. D. Chen, M. Maheswaran, "A Fair Synchronization Protocol with Cheat Proofing for Decentralized Online Multiplayer Games", in 3rd IEEE Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, pp. 372-375, Aug 2004

[171] W. Bernier, "Latency Compensating Methods in Client/Server In-game Protocol Design and Optimization", in Game Developer Conference (GDC), Mar 2001. http://www.resourcecode.de/stuff/clientsideprediction.pdf

[172] M. Bottigliengo, C. Casetti, C. Chiasserini, M. Meo "Short-Term Fairness for TCP Flows in 802.11b WLANs", in Proc. of IEEE INFOCOM 2004, Hong Kong, China, Mar 2004.

[173] S. Garg, M. Kappes, "An Experimental Study of Throughput for UDP and VoIP Traffic in IEEE 802.11b Networks", IEEE Wireless Communications and Networking Conference (WCNC 2003), New Orleans, LA, USA, pp. 1748-1753, Mar 2003.

[174] A. L. Wijesinha, Y. Song, M. Krishnan, V. Mathur, J. Ahn, V. Shyamasundar, "Throughput Measurement for UDP Traffic in an IEEE 802.11g WLAN", in Proc. of 6th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks (SNPD/SAWN'05), Towson, MD, USA, pp. 220-225, May 2005.

[175] D. P. Hole, F. A. Tobagi, "Capacity of an IEEE 802.11b Wireless LAN supporting VoIP", in Proc. IEEE International Conference on Communications (ICC 2004), Paris, France, pp. 196-201, Jun 2004.

[176] G. Bianchi, "IEEE 802.11-Saturation Throughput Analysis" IEEE Communications Letters, vol. 2, no. 12, pp. 318-320, Dec 1998.

[177] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function", IEEE Journal on Selected Areas in Communications, vol. 18, no. 3, pp.535-547, Mar 2000.

[178] N. Shachtman, "New Army Soldiers: Game Gamers", Wired News, Oct 2001. http://www.wired.com/news/conflict/0,2100,47931,00.html

[179] Rakion. http://www.rakion.com/

[180] S. Ferretti and M. Roccetti. The design and performance of a receiver-initiated event delivery synchronization service for interactive multiplayer games. In S. Natkin Q. Mehdi, N. Gough, editor, Proceedings of the 4th International Conference on Intelligent Games and Simulation (Game-On 2003), London, UK, Nov 2003.

[181] R. M. Fujimoto. Parallel and Distribution Simulation Systems. John Wiley & Sons, Inc., 1999.

[182] M. Mauve. Consistency in replicated continuous interactive media. In Proceedings of the 2000 ACM conference on Computer supported cooperative work, ACM Press, pp. 181-190, 2000.

[183] M. M. Oliveira, T. Henderson, "What online gamers really think of the internet?", in Proc. of the 2nd workshop on Network and system support for games, pp. 185-193, 2003.

[184] S. Ferretti, M. Roccetti, C. E. Palazzi, "An Optimistic Obsolescence-Based Approach To Event Synchronization For Massive Multiplayer Online Games, in International Journal of Computers and Applications, ACTA Press, vol. 29, no. 1, 2007.

[185] S. Ferretti, M. Roccetti, A. La Penna, "Fast Synchronization of Mirrored Game Servers: Outcomes from a Testbed Evaluation", in Proc. of the International Symposium on Intelligence Techniques in Computer Games and Simulations, Shiga, Japan, Mar 2007.

[186] G. Mattew, 802.11 Wireless Networks - Definitive Guide, O'Reilly, 2002

[187] J. Yin, T. El Batt, G. Yeung, B. Ryu, S. Habermas, H. Krishnan, "Performance Evaluation of Safety Applications over DSRC Vehicular Ad Hoc Networks", International Conference on Mobile Computing and Networking, 2004

[188] IEEE 802.11 – Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/IEEE_802.11.

[189] E. Fasolo, R. Furiato, A. Zanella, "Smart Broadcast Algorithm fo Inter-vehicular Communication", in Proc. of Wireless Personal Multimedia Communication (WPMC'05), IWS 2005, Aalborg, DK, Sep 2005.

[190] S. Biswas, R. Tatchikou, F. Dion, “Vehicle-to-Vehicle Wireless Communication Protocols for Enhancing Highway Traffic Safety”, IEEE Communication Magazine, vol. 44, no. 1, Jan 2006, pp. 74-82.

[191] X. Yang, J. Liu, F. Zhao, N. Vaidya, “A Vehicle-to-Vehicle Communication Protocol for Cooperative Collision Warning”, in Proc. of 1st Annual International Conf. on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04), Boston, MA, USA, Aug 2004, pp. 114-123.

[192] G. Korkmax, E. Ekici, F. Ozguner, U. Ozguner, “Urban Multi-hop Bradcast Protocol for Inter-vehicle Communication Sustems”, in Proc. of the 1st ACM Workshop on Vehicular Ad-hoc Networks (VANET 2004), Oct 2004, pp. 76-85.

[193] A. Zanella, G. Pierobon, S. Merlin, “On the Limiting Performance of Broadcast Algorithms over Unidimensional Ad-hoc Radio Networks”, in Proc. of WPMC04, Abano Terme, Italy, Sep 2004.

[194] P. J. Wan, K. Alzoubi, O. Frieder, “Distributed Construction of Connected Dominating Set in Wireless Ad hoc Networks”, in Proc. of IEEE INFOCOM 2002, Jun 2002.

[195] C. Perkins, “IP Mobility Support for IPv4”, IETF RFC 3344, Aug 2002.

[196] G. Marfia, C. E. Palazzi, G. Pau, M. Gerla, M. Y. Sanadidi, M. Roccetti, “TCP Libra: Exploring RTT-Fairness for TCP”, in Proc. of the IFIP/TC6 NETWORKING 2007, Atlanta, GA, USA, May 2007.

[197] M. Roccetti, M. Gerla, C. E. Palazzi, S. Ferretti, G. Pau “First Responders' Crystal Ball: How to Scry the Emergency from a Remote Vehicle”, in of Proc. the 1st IEEE International Workshop on Research Challenges in Next Generation Networks for First Responders and Critical Infrastructures (NetCri 07) - 26th IEEE International Performance Computing and Communications Conference (IPCCC 2007), New Orleans, LA, IEEE Computer Society, Apr 2007.

[198] J. F: Kurose, K. W. Ross, Computer Networking: A Top-Down Approach Featuring the Internet, Addison Wesley Longman, Boston MA, USA, 2001.

[199] M. Fazio, C. E. Palazzi, S. Das, M. Gerla, "Facilitating Real-time Applications in VANETs through Fast Address Auto-configuration", in Proc. of the 3rd IEEE CCNC International Workshop on Networking Issues in Multimedia Entertainment (CCNC/NIME 2007), Las Vegas, NV, USA, IEEE Communications Society, Jan 2007.

[200] C. E. Palazzi, S. Ferretti, M. Roccetti, G. Pau, M. Gerla, "How Do You Quickly Choreograph Inter-Vehicular Communications? A Fast Vehicle-to-Vehicle Multi-Hop Broadcast Algorithm, Explained", in Proc. of the 3rd IEEE CCNC International Workshop on Networking Issues in Multimedia Entertainment (CCNC/NIME 2007), Las Vegas, NV, USA, IEEE Communications Society, Jan 2007.

[201] M. Fazio, C. E. Palazzi, S. Das, M. Gerla, "Vehicular Address Configuration", in Proc. of the 1st IEEE Workshop on Automotive Networking and Applications (AutoNet), GLOBECOM 2006, San Francisco, CA, USA, Dec 2006.

[202] M. Fazio, C. E. Palazzi, S. Das, M. Gerla, "Automatic IP Address Configuration in VANETs", in Proc. of the Third ACM International Workshop on Vehicular Ad Hoc Networks (VANET 2006), MOBICOM 2006, Marina del Rey, Los Angeles, CA, USA, Sep 2006.

[203] C. E. Palazzi, S. Ferretti, M. Roccetti, G. Pau, M. Gerla, "Wireless Home Entertainment Center: Reducing Last Hop Delays for Real-time Applications", in Proc. of ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE 2006), Hollywood, CA, USA, Jun 2006.

[204] C. E. Palazzi, S. Ferretti, M. Roccetti, G. Pau, M. Gerla, "What's in that Magic Box? The Home Entertainment Center's Special Protocol Potion, Revealed", IEEE

Transactions on Consumer Electronics, IEEE Consumer Electronics Society, vol. 52, no. 4, Nov 2006, 1280-1288.

[205] IEEE 802.11g, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, Amendment 4: Further Higher Data Rate Extension in the 2.4GHz Band.

[206] National Center for Statistics and Analysis, "Traffic Safety Facts 2003", Report DOT HS 809 767, Nat'l. Highway Traffic Safety Admin., U.S. DOT, Washington, DC, 2004.

[207] ASTM E2213-03, "Standard Specification for Telecommunications and Information Exchange Between Road-side and Vehicle Systems – 5.9GHz Band Dedicated Short-Range Communications (DSRC) Medium Access Control (MAC) and Physical Layer (PHY) Specifications", ASTM International, Jul 2003.

[208] A. Spagnolli, L. Gamberini, F. Scarpetta, S. Colognesi, "Ironia e ristrutturazione del participant framework: il caso degli ambienti virtuali", Rivista di Psicolinguistica Applicata, vol. 2, pp. 165-176, 2004.

[209] A. Spagnolli, L. Gamberini, P. Cottone, G. Mantovani, "Ergonomics of Virtual Environments for Clinical Applications", Internet and Virtual Reality as Assessment and Rehabilitation Tools for Clinical Psychology and Neuroscience, Amsterdam, IOS Press (Netherlands), pp. 217-230, 2004.

[210] M. Billinghurst, H Kato, "Collaborative Augments Reality", Communications of the ACM, vol. 45, no. 7, pp. 64-70, Jul 2002.

[211] Armagetron: a Tron clone in 3d. http://armagetron.sourceforge.net/.