Network Support for Mobile Gaming

21

Armir Bujari, Marco Furini, and Claudio E. Palazzi

Contents

Abstract

Mobile gaming has become very popular thanks to its entertainment nature and to the widespread popularity of high-end mobile devices. The game scenario is very challenging as the support of mobile games is not as easy as one may think. In particular, the traffic generated by mobile games has specific network requirements that need to be satisfied; otherwise, the playability of the game might be annoying instead of pleasant. With this vision, we overview the network support solutions for mobile games. In particular, we characterize the mobile game network requirements and present studies that propose a suitable network support for these games.

A. Bujari (✉) • C.E. Palazzi
Department of Mathematics, University of Padua, Padua, Italy
e-mail: abujari@math.unipd.it; cpalazzi@math.unipd.it

M. Furini
University of Modena and Reggio Emilia, Modena, Italy
e-mail: marco.furini@unimore.it

R. Nakatsu et al. (eds.), *Handbook of Digital Games and Entertainment Technologies*,
DOI 10.1007/978-981-4560-50-4_24

Finally, we overview an interesting case study where mobile games can be played in an environment composed of fixed and mobile networks.

Keywords
Interactive games • Wireless network • Access point • Broadcast message • Energy consumption • MAC layer retransmissions • MANETs • Network changes • SAP

Introduction

The mobile gaming market continues to expand quickly and is equaling the revenues of the console game market. It follows that mobile games have become one of the most important digital platforms for gamers and publishers alike: as of 2014, mobile games account for over one third of monthly spending among digital gamers in the United States alone, and it reached the amount of US$21 billion revenue (Newzoo 2014).

Different actors are contributing to the success of this market: giants of the game industry, like Electronic Arts and Ubisoft, are transforming console game into mobile game; novel companies, like King and Zynga, are releasing more and more titles for the mobile gaming market; and cell phone industries are producing devices with more and more appealing multimedia features. As a result, the mobile gaming scenario is filled with a wide range of games: from very simple graphic games to cutting-edge 3D graphics and from single- to multiplayer games (Furini 2008; Palazzi and Maggiorini 2011).

The most common mobile gaming scenario is composed of players (from a few to hundreds of thousands) who use their own device (e.g., a cell phone, a game console) to play a game, of a game server (or of a series of game servers) that is in charge of running the game properly, and of a network that connects devices and server(s). To play the game, each player must interact with the server (and/or with the other players), and the server has to process each player's input, has to compute the new game state, and has to communicate this state to each player. Indeed, the game normally proceeds through states, and to ensure that all players will perceive the same game evolution, the state of the game should be the same in the whole scenario. Unfortunately, the support of mobile games is not easy as one may think, and the variation of the network latency may cause players to experience a game state inconsistent with the one of the game server or with the one of other players. Needless to say, this is a critical issue.

Indeed, mobile games have peculiar characteristics (interactivity, consistency, fairness, scalability, and continuity) that require special services from the network (e.g., very low end-to-end delays). For instance, to provide interactivity, 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 be kept under a certain interactivity threshold; consistency requires all the players to view the same game state; fairness requires the network to not affect the chances of winning; and continuity requires the network to avoid interrupted game sessions usually caused by

disconnections, handoffs, or any other wireless-/mobility-related issue. If the network does not support these peculiar characteristics, the performance of the mobile games might result very poor.

In essence, the traffic produced by mobile games has timing constraints (i.e., it is time sensitive), and the network is required to meet these constraints in order to well support these games. Unfortunately, in a mobile scenario, the service provided by the network might be not sufficient to meet the mobile game requirements. For instance, in an infrastructure-based scenario, the Access Point used to reach the Internet could be employed for heterogeneous flows and applications, both elastic and real time, thus introducing delays that would jeopardize the performance of interactive games. Similarly, if a node moves out of the AP transmission range, it loses all the ongoing sessions. The problems are exacerbated in infrastructure-less networks (e.g., ad hoc, mobile, and vehicular) that are prone to disconnection, energy consumption, etc. For these reasons, the game scenario has been studied by researchers and practitioners in the attempt to provide an effective and efficient supporting scenario (Furini 2007; Griwodz 2002).

In this chapter we focus on network support for mobile games, as depending on the networking performance, the user may experience a pleasant mobile game or an annoying one that will discourage him/her from playing again. In particular, we present what are the mobile game network requirements and what are the current studies that aim at providing a suitable network support for these games. Finally, we describe a particular architecture that seems to be a promising solution to support games in a mobile scenario.

The remainder of the chapter is organized as follows. Section "Mobile Game Network Requirements" describes the typical network requirements of a mobile game, and section "Network Support for Mobile Games" presents the services provided by current wireless networks. Section "Network Architectures" presents the characteristics of the most used architectures able to support mobile games, and section "Smart Architecture: A Case Study" presents a promising holistic solution for the support of mobile games in the wireless scenario. Finally, section "Conclusions" concludes this chapter.

Mobile Game Network Requirements

The mobile game scenario is composed of games that require special services from the network, as the delay, the jitter, and the transmission rate directly affect the quality perceived by players. Before detailing the network requirements of online games, it is worth describing a practical example of problems that might arise when playing online games. Figure 1 shows the frame evolution of an Armagetron game session (Armagetron 2014), where user Cla (blue cycle and wall) plays against user Eu (green cycle and wall). The game's rules are simple: each player controls a light cycle that leaves a wall behind it wherever the cycle goes; the cycle cannot stop and can turn only at 90° angles. The goal of the game is that of having all other players

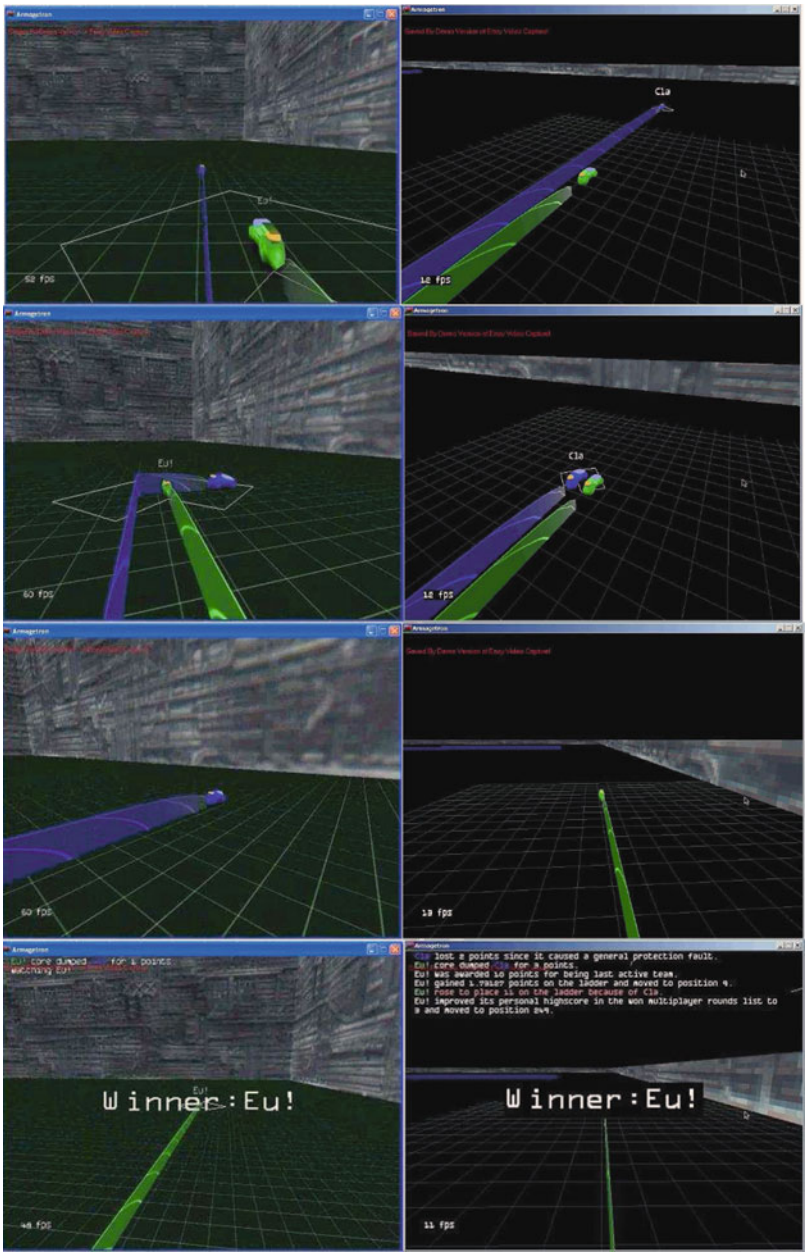

Fig. 1 Frame sequence of an Armagetron game session: *Cla*'s view (*left column*, blue player) versus *Eu*'s view (*right column*, green player); evident lag differences generate inconsistencies and unfairness

Fig. 2 Frame sequence of a game session as seen by a client: other participants are positioned in the game arena with a lag that depends on the distance between clients and server

crashing 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. 1, there are subsequent frames as seen by the player Cla, who is connected to a certain game server with 180 ms of round-trip time (RTT). Instead, frames on the right column correspond to the game as seen by the player Eu, who is connected to the same server with only 20 ms of RTT. Frames on the same row relate 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 row of frames, player Cla believes he has just won (in the left frame of the second row, Cla sees Eu hitting his wall), whereas after a couple of seconds (last frame row), he realizes that the server has 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. Even if the game were 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 and embodies an example of how the network may affect the playability of a mobile game. Indeed, the real game evolution is the server's view, 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 evolving in advance with respect to Cla, as the former is much closer to the game server than the latter. To clarify this aspect, Fig. 2 presents another practical example on how network delays might affect

consistency. The picture presents both the position of a certain participant as seen by another playing client (the full figure avatar) and the position as seen by the server (the human-shaped light boxes). It can be noted that transmission lag creates a difference in the avatars' positions as perceived by the server and by each of the various clients.

Inconsistency and unfairness are only two among the various problems that a network may cause to a mobile game. Indeed, in general, online games require the network to support five main features that are intrinsically correlated: interactivity, consistency, fairness, scalability, and continuity.

Interactivity: also known as responsiveness, it 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. In order to ensure an enjoyable playability to the final user, 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 be kept under a certain interactivity threshold (Armitage 2003; Panten and Wolf 2002). Unfortunately, variable network congestion conditions may suddenly slow down the game fluency on the screen. Moreover, players in the same virtual arena could be so numerous that some game servers may experience impulsive computational load and lose interactivity. These problems are obviously amplified when plunged into a wireless, mobile scenario. In fact, wireless characteristics and node mobility generate a high variability in experienced delays and network traffic and also generate a high churn rate that worsens the aforementioned problems (Nandan et al. 2005a).

Consistency: it refers to the simultaneous uniformity of the game state view in all nodes belonging to the system. Also in this case, the network response might affect the consistency and playability of the game. The easiest way to guarantee absolute consistency would be that of making the game proceed through discrete locksteps. Having a single move allowed for each player and synchronizing all agents before moving toward the next round surely grants absolute consistency but, on the other hand, impairs the system interactivity. A trade-off issue between consistency and interactivity needs thus to be solved in order to develop a proficient game platform.

Fairness: every player should have the same chances of winning the game session, regardless of different network conditions. 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 participants. To this aim, it has been demonstrated how increasing the interactivity degree of the game platform may lead also to improved fairness (Ferretti et al. 2005). However, it is known that the wireless environment generates peculiar delays and related unfairness problems. These problems came from the nature of the shared wireless medium and from location dependency. In fact, if we view a node and its interfering nodes to form a "neighborhood," the aggregate of local queues at these nodes represents the distributed queue for this neighborhood. This aggregate queue is not FIFO served, as flows sharing the queue have different, dynamically changing priorities determined by the topology and traffic patterns. Thus, they get different feedback in terms of packet loss rate and packet delay when congestion occurs (Xu et al. 2005).

Scalability: it regards the capability of the system in providing efficient support to a large community of players. Indeed, it is of primary interest for game companies to have revenues generated by a very high number of customers. Besides, humans are social beings that enjoy the presence of others and the competition against real adversaries. Yet, especially in the case of fast-paced games, when the interactivity threshold cannot be met, scalability is sometimes sacrificed by denying the access to some users depending on their experienced delays (Rakion 2014). Therefore, by sustaining interactivity, one can also provide a higher scalability degree in terms of both the number and the geographic dispersion of players allowed to participate to the same virtual arena. Discussing scalability, it is particularly interesting to notice that wireless connectivity provides means to widen the set of potential players as it is no longer required to be wired to the Internet to engage online games. Think of the possibility to establish outdoor online gaming sessions through 3G, WiMAX, vehicular IEEE 802.11p, ad hoc networks, mesh networks, etc. Clearly, this brings novel issues into the scenario, such as disconnections and energy consumption (Marfia and Roccetti 2010; Roccetti et al. 2010).

Continuity: it is concerned with having game sessions not interrupted by disconnections, handoffs, or any other wireless-/mobility-related issue. Indeed, players would be very frustrated by having their game session continuously interrupted and restarted (maybe after a while). This problem may happen also when trying to exploit the new wireless capabilities of popular smartphones to create proximity-based gaming sessions. Indeed, we can imagine players forming an ad hoc network to engage in an outdoor multiplayer game based on the connectivity means of their smartphones. Yet, players' movements and different energy consumption among devices may create detached cluster of nodes (Kokkinos et al. 2011; Yu et al. 2003). To this aim, proposed solutions regard games featured with short game sessions, having quick/smart handoff mechanisms, and considering route and server migration (Chen et al. 2006; Palazzi et al. 2010a; Zhu et al. 2011).

Network Support for Mobile Games

To ensure a good playability in the mobile game scenario, the network should provide a service able to guarantee the five features described in the previous section. To this aim, many researchers have focused their studies on the issues encountered in a wireless environment. In the following, we analyze problems and solutions that have emerged while supporting mobile games through infrastructure-based and infrastructure-less wireless networks.

Infrastructure-Based Wireless Network

In this type of network, there is an Access Point (AP) that has access to the Internet, and all the other mobile nodes connect to the AP to have Internet access. In this

scenario, the AP offers to all engaged mobile nodes the same functionalities. An example of this network can be found in the home scenario (but also in public areas), where a Wi-Fi network can be used to establish an interactive gaming session or to connect to remote players or game servers. There are two main drawbacks in this scenario: (i) if a mobile node moves out of the transmission range of an AP, it loses its connectivity to the Internet and therefore the mobile game stops working, and (ii) the AP could be employed to manage the traffic produced by both elastic and real-time applications, thus introducing delays that would compromise the playability of interactive games (Palazzi et al. 2007).

To mitigate these problems, researchers proposed to extend the range of a wireless infrastructure with the employment of mesh networks, which represent a way to create wireless chains among APs. In this scenario, if the moving node enters in an area covered by a new AP, it might be able to connect to the Internet again. However, it has to deal with handoffs that probably terminate its ongoing sessions. This problem can be solved through a smooth handoff that seamlessly transfers the connectivity from the old to the new AP before disconnection. A well-known protocol to perform this task is represented by Mobile IP that transfers packets from the old AP to a new one through routing triangulation (Perkins 2002). Therefore, this network topology can hypothetically support infrastructure-based wireless mobile games where players are located and moving in an area wider than one single AP's transmission range, but the problems due to the management of heterogeneous traffic flows remain.

To mitigate these problems, many research papers focused on IEEE 802.11 and presented analysis, problems, and solutions. Unfortunately, the vast majority of them focused on a throughput/losses point of view (Bianchi 2000; Bottiliengo et al. 2004; Heusse et al. 2003), and therefore they cannot be applied to the mobile game scenario. In fact, the performance of mobile games (and of real-time applications in general) depends on the measured per-packet delay and jitter and not on the achieved throughput/losses (Beigbeder et al. 2004; Conti et al. 2002). For this reason, different studies focused onMAC layer retransmissions, showing that MAC retransmissions can be wasteful and potentially harmful for time-sensitive applications as they introduce delays (Palazzi et al. 2005b); however, other studies showed that without retransmissions implemented at the link layer, the loss rate becomes unacceptable for any application (Nam et al. 2003; Xylomenos and Polyzos 1999). Similarly, another important issue is that persistent TCP-based flows (e.g., downloads) are responsible for performance deterioration of concurrent UDP-based flows (e.g., interactive games). Indeed, the continuous search for more bandwidth performed by TCP's congestion control algorithm creates queues at buffers, thus augmenting the per-packet delay of any flow sharing the same channel. With the aim of ensuring low per-packet delays, while preserving downloading throughput, Palazzi et al. (2006b) proposed a solution based on a smart AP (SAP). The idea is to monitor the entire traffic passing through and, for each TCP flow, to compute the maximum transmission rate so as to not exceed the channel capacity and accumulate packets in queue. Only standard features and protocols are used to facilitate deployment: the maximum transmission rate for each flow is enforced by modifying on the fly the advertised window in TCP ACK packets. We observe that an

insightful analysis of this issue with respect to online games is still missing, as well as efficient solutions aimed at reducing queuing delay over wireless links (Hole and Tobagi 2004; Palazzi et al. 2005b).

Infrastructure-Less Wireless Network

In an infrastructure-less network (aka ad hoc networks) wireless nodes can communicate with any other node in its transmission range without the need of any AP. Transmissions can happen both directly between two nodes that are close to each other and through multi-hop when a node needs to send a message to another node that is out of its range. Ad hoc networks have received the attention of researchers and practitioners thanks to 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, etc. In this scenario, Bononi et al. (2009) showed that ad hoc networks can be used to support both real-time and non-real-time applications provided that each node manages the outgoing traffic with specific transmission scheduling and with (at least) two different queues: one dedicated to the traffic generated by elastic applications and another one dedicated to the traffic coupled with timing constraints.

A particular type of ad hoc network is the one where nodes move (Corson and Macken 1999). This scenario is known as mobile ad hoc networks (MANETs) and represents a typical outdoor gaming scenario, where players' devices dynamically connect to each other creating an ad hoc network to support their multiplayer games. This scenario is very challenging for two main reasons: disconnections and energy consumption. MANETs are prone to disconnections. Due to mobility, one or more nodes may get out of range of the original ad hoc network, becoming unable to reach the game server node and continue playing. Even worse, the server node itself may get out of range for the rest of the network, thus interrupting the game session for everybody.

Energy consumption might be a major concern when considering outdoor games based on the connectivity of small devices because of the limited amount of energy stored in their batteries. This limited energy can be quickly consumed by game-related computation, visualization, and communication. Clearly, the worst energy consumption is experienced by the node that is also the server of the game session: that node will experience a much faster decline of its energy reserve as it will have to receive all game events from other players, compute game state updates, and transmit these updates back to the other players. Many solutions have been proposed to ensure data transmissions over MANETs while aiming at low energy consumption (Yu et al. 2003). These solutions are generally based on having nodes alternating sleep/awake modes, and therefore they cannot be exploited for interactive mobile games due to the timing constraints of the produced traffic. Other solutions exist that aim at saving nodes' energy through smart routing in a MANET (Kokkinos et al. 2011; Yu et al. 2003). Yet, they do not consider any delivery delay options, thus

not necessarily ensuring interactivity. Moreover, the scenario with one of the nodes also embodying the game server is not considered. In this scenario, one single node (i.e., the game server) receives and generates most of the traffic. The application itself and the network architecture impose unbalanced energy consumption. Thereby, the server (and player) node will run out of energy much before the other player nodes, yet interrupting the gaming session for every node. To address this issue, Palazzi et al. (2010a) proposed to utilize three kinds of nodes in the game network: active servers, backup servers, and players. In essence, the traffic produced by mobile games has timing constraints (i.e., it is time sensitive), and the network is required to meet these constraints in order to well support these games. In this solution, all servers collect game events sent by other players and continuously update their game state view. However, only the active server forwards the current game state to all the players (including the backup servers); this limits the energy consumption of backup servers until they are called to become active.

Another particular type of ad hoc network is the one where nodes move at high speed. In this case, the network is named vehicular ad hoc networks (VANETs) and represents a scenario where mobile games will probably play a successful role (Palazzi et al. 2010b). Similar to the MANET scenario, in a VANET, players' devices dynamically connect to each other creating an ad hoc network to support the game. In particular, transmissions can follow either a pull or a push model (Nandan et al. 2005b). With the former, vehicles explicitly ask to receive certain data from other vehicles, whereas with the latter, messages are proactively broadcast to every node in a certain area of interest. Both models can involve multi-hop transmissions. Since the shared nature of the wireless channel, communications involving several nodes in the same VANET are clearly more efficient if performed through broadcasting (i.e., push model). Having many connected players moving fast in their cars certainly represents one of the most challenging and interesting context for infrastructure-less mobile games. Fundamental issues for mobile games such as the quick propagation of game events among players are exacerbated in this context and require special attention (Beigbeder et al. 2004; Biswas et al. 2006); similarly, interference and sudden congestion may be exacerbated by the very high mobility of vehicles causing the management of these networks to be tougher than the one of MANET (Casteigts et al. 2011). Moreover, this scenario is very challenging for two main reasons: management of network changes and the broadcast of a message.

Network changes. The management of network changes in a fast mobility scenario represents a tough issue both for ensuring connectivity and for the high variability in concurrent network traffic (Zhu et al. 2011). Indeed, since the topology of a vehicular network (vehicles around a certain player and their concurrent data traffic) may continuously and quickly change, the traffic may experience sudden delays that will likely jeopardize the interactivity of the gaming session (Palazzi et al. 2010b).

Broadcast of a message. The broadcast of a message also represents a tough issue (Yang et al. 2004), as its delivery is not fast as it should be. Experts report that main reasons behind a slow broadcast delivery are due to a nonoptimal 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 (Biswas et al 2006; Fasolo et al. 2005; Korkmax et al. 2004; Yang et al. 2004). To tackle this problem a theoretically optimal broadcast algorithm has been proposed (Zanella et al. 2004). However, there are practical difficulties in the implementation of this algorithm as it would require a complete and continuously updated knowledge of the network topology. For instance, with N cars, the algorithm employs as many as O(N log N) control messages (Wan et al. 2002). It goes without saying that this is not a scalable solution. A backoff mechanism that reduces the frequency of message retransmissions when congestion is causing collisions is proposed in Yang et al. (2004). In Biswas et al. (2006), 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. The solution presented in Bononi and Di Felice (2006) utilizes a distributed proactive clustering scheme to dynamically create an efficient virtual backbone infrastructure in the vehicular network. The backbone formation process takes into consideration both the current distance among candidate backbone vehicles and the estimated lifetime of the wireless connection among neighbor backbone members. The attempt is that of improving the robustness and lifetime of connections among backbone members even in a highly mobile scenario as a vehicular network. To minimize the number of hops, Korkmax et al. (2004) proposed to individuate 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 mobile game scenario. Fasolo et al. (2005) proposed to assign different contention windows to each car to have different waiting times before propagating the broadcast message. Nodes set their respective contention windows with an inverse proportion of the distance from the sender, thus needing less forwarders (and transmissions) to cover a certain area. Yet, this scheme assumes that there is a unique and constant transmission range for all the cars in every moment; this is obviously not realistic in a VANET because of its high and fast mobility. Other solutions have hence been devised to solve this shortcoming. In particular, similar automatic transmission range estimators are proposed in Palazzi et al. (2007) and Roccetti et al. (2010) to assess the actual transmission range for every car in the platoon. More in detail, the former exploits this information to have the farthest vehicle (i.e., the farthest relay) in the sender's transmission range becoming the next forwarder of the message. Instead, the latter uses this information to assign the forwarding task to the farthest spanning relay. In both cases, the computed transmission range estimation is used to support a multi-hop broadcasting scheme for

message exchange able to dynamically adapt to the different (transmission range) conditions a vehicular network may encounter.

Network Architectures

To support mobile games, many researchers focused on the design of network architectures to ensure a good playability. According to their characteristics, these proposals might be grouped into three different categories (centralized client–server, fully distributed, and mirrored game server). In the following, we present the main advantages and disadvantages of these categories when applied to a mobile game scenario.

Centralized client–server. It is an architecture composed of a single authoritative point (the server) which is responsible to run the main logic of the game, 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, render it on the screen, and forward player's commands. The server can be both a single computer and cluster of computers in order to increase the performance of the system (Butterfly 2014). The centralized client–server architecture represents the simplest solution for authentication procedures, security issues, and consistency maintenance (Gautier and Diot 1998; Quake 2014; Ultima 2014). This architecture, assuming to have N simultaneous players, generates a number of messages in the order of O(N), but the presence of a single authoritative point represents a unique bottleneck that limits its efficiency and scalability.

Fully distributed. It is an architecture that well represents the peer-to-peer paradigm: all the involved nodes share the same intelligence and are equally responsible for running the whole logic of the system; each client has to autonomously update the game state view based on its player's commands and on game actions received from other players. 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 (Gautier and Diot 1998; Safaei et al. 2005). However, this approach requires terminals endowed with higher computational capabilities, and identical copies of the current game state need to be stored at each node. Therefore, it is necessary to introduce some complex coordination mechanism 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. The exchanged messages could hence rise to the order of $O(N^2)$, where N is the number of simultaneous players. Finally, authentication, cheating, and general consensus among all the peers are harder to be addressed than when a centralized architecture is employed. However, the fully distributed architecture is generally preferred for infrastructure-less networks such as MANETs and VANETs because of its ability to deal with high mobility of nodes that continuously changes the topology of the network (Palazzi et al. 2010a).

Mirrored game server. It is an architecture that represents a hybrid solution able to efficiently embrace all the positive aspects of both centralized client–server and

fully distributed architectures (Palazzi et al. 2006a). When employing this architecture, the game state servers (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. There are three main advantages in employing a mirrored game server architecture: (i) the absence of a single point of failure, (ii) the networking complexity that is maintained at the server side, and (iii) 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 simultaneous 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. The presence of multiple high-performance GSSs helps in distributing the traffic over the system and reduces the processing burden at each node (Safaei et al. 2005). 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 (Jehaes 2003). Indeed, 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, through this architecture, it becomes simpler to adopt efficient solutions for the trade-off among the five main features of a mobile game (i.e., interactivity, consistency, fairness, scalability, and continuity). For instance, Palazzi et al. (2005a) suggested that during a game session some events can lose their significance as time passes, and therefore discarding superseded events for processing fresher ones may be of great help for delay-affected GSSs, achieving high interactivity degree without compromising consistency. Furthermore, for very fast-paced games, little inconsistencies are not highly deleterious for players' fun. In these cases, even some non-superseded game event could be dropped when dropping all superseded ones is not yet sufficient to maintain an adequate level of responsiveness.

Given the available architectural solutions, we can hence infer that when considering a (moderately) mobile network of players supported by infrastructure and by Access Points, the hybrid solution of mirrored servers probably represents the best choice. Conversely, in case of no available infrastructure or very high mobility of players (e.g., cars' passengers), then a fully distributed solution may be the only feasible option.

Smart Architecture: A Case Study

In this section we review the Smart Architecture, an interesting case study where mobile games can be played in an environment composed of fixed and mobile networks (see Fig. 3). The Smart Architecture embraces the positive aspects of the

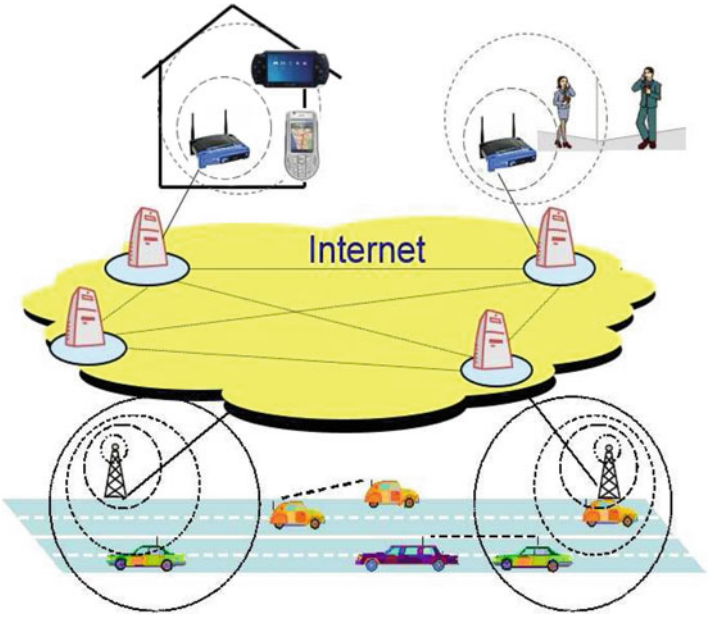

Fig. 3 Hybrid architecture for distributed game entertainment in heterogeneous scenarios with mobile users (even car passengers)

centralized client–server and of the fully distributed architectures, and therefore it can be considered a hybrid between the two architectures (Cronin et al. 2004). Experiments showed the benefits of using the hybrid architecture when supporting mobile games. In the following, we review the two main components of the architecture: (i) a smart synchronization mechanism for mirrored game servers and (ii) a smart AP able to avoid last-mile queuing delays that would jeopardize interactivity just one step before delivering.

Smart Synchronization Mechanism Among GSS

To speed up the synchronization among GSSs, the Smart Architecture employs a smart synchronization mechanism (SSM) that exploits the semantics of the game to discard few game packets in order to preempt interactivity loss when intense network traffic or excessive computational load is slowing down some GSS (Palazzi et al. 2006a; Safaei et al. 2005). The idea behind the SSM takes inspiration from the active queue management approaches employed in RED (Floyd and Jacobson 1993) and RIO (Clark and Fang 1998) and utilizes a uniformly distributed dropping function. Yet, the parameter taken under control by GSSs is the time elapsed from the generation of the game event, which is named Game Time Delivery (GTD). In fact, upon each packet arrival, each GSS determines the GTD of the arrived event, namely, sample_GTD, and feeds a low-pass filter to compute

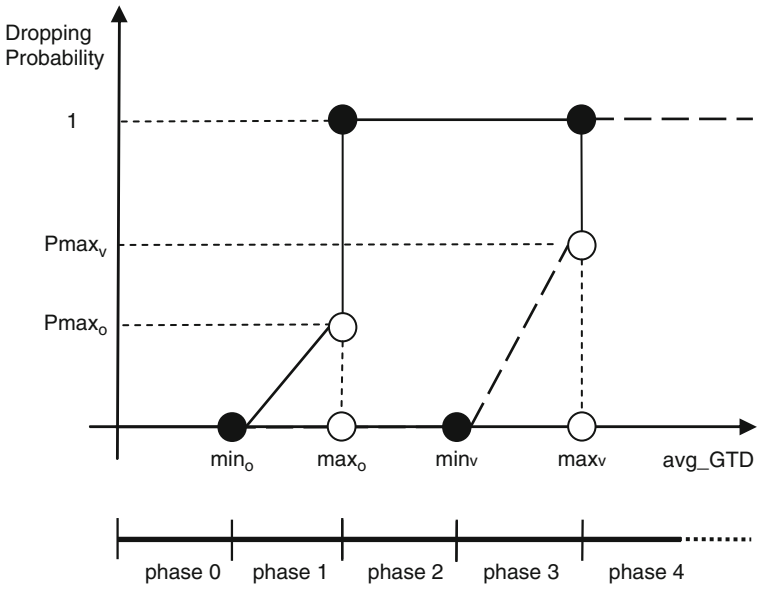

Fig. 4 Discarding probability functions

the updated average GTD, namely, avg_GTD. When avg_GTD exceeds a certain threshold, the GSS drops superseded events with a certain probability p, without processing them. If avg_GTD exceeds a subsequent limit, p is set equal to 1, and all superseded events waiting for being processed are discarded.

Indeed, 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.

To ensure an adequate playability degree even to fast and furious class of games, a further dropping probability function is provided in order to discard even non-superseded game events when dropping all the superseded ones is not yet sufficient to maintain an adequate level of responsiveness.

Figure 4 depicts the two discarding functions of the SSM. Three parameters (and three phases) characterize each of the twin algorithms: $\min_o$, $\max_o$, and Pmax_o, for superseded events, and $\min_v$, $\max_v$, and Pmax_v for non-superseded ones.

A Smart AP

Even if SSM coupled with a mirrored game server architecture is proficient in maintaining a high degree of responsiveness among GSSs, still problems may arise at the edges of the considered topology, where users in their homes or along a street may be engaged in an online game through an AP (see Fig. 3). Concurrent traffic

may generate queues that build up at the AP, thus delaying the game event delivery and wasting all the interactivity patrimony created by the SSM. In particular, as previously mentioned, some applications may be particularly harmful toward online gaming traffic as they may increase queuing delays to such an extent that interactivity may be completely compromised (Palazzi et al 2005b).

The smart AP aims at achieving the best performance for both elastic and real-time applications by appropriately limiting the advertised window for TCP flows (Palazzi et al 2006b). An optimal trade-off between throughput and low delays could be achieved by maintaining the sending rate of TCP flows high enough to efficiently utilize all available bandwidth but, at the same time, limited in its growth so as to not utilize buffers. As a result, the throughput is maximized by the absence of packet loss, while the delay is limited by the absence of queuing.

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

$$TCPubrate\ \ (t) = \frac{(C - UDPtraffic\ \ (t))}{\#TCPflows\ \ (t)} \tag{1}$$

where *UDPtraffic (t)* is the amount of bandwidth occupied by UDP-based traffic at time t, *#TCPflows (t)* is the concurrent number of TCP flows, and C is the total capacity of the bottleneck link. This upper bound can be enforced to all TCP flows sharing the same wireless link by having the corresponding AP appropriately modifying the advertised window of passing-through TCP flows.

To compute (1), a comprehensive knowledge of the flows transiting through the bottleneck (i.e., the last hop links) is needed: the total capacity of the channel, the aggregate amount of current UDP traffic, and the number of TCP flows currently active on the wireless link. This information is indeed possessed by the AP as all flows have to pass through it.

A mathematical model evaluated the Smart Architecture. In particular, the model considered a lognormal distribution for the DTD game traffic (Park and Willinger 2002), and the simulated scenario was the one of Fig. 3: seven interconnected GSSs with a network latency between the two farthest GSSs of 90 ms, with GSS that transmits to the other GSS game updates related to their engaged clients every 30 ms.

Results were gathered collecting the total latency experienced by game events reaching one of the clients connected through an IEEE 802.11g AP to one of the GSS. The same AP was also in charge of handling traffic coming from other applications run on different devices that were simultaneously sharing that wireless link: a UDP-based video stream, a UDP-based live video chat, and a TCP-based downloading session. The video stream and video chat applications were simulated by injecting in NS-2 real traces corresponding to high-quality MPEG4 Star Wars IV and VBR H.263 Lecture Room-Cam, respectively, as available in Movie (2014).

In the following charts, REG represents the case where a regular synchronization scheme is adopted by GSSs, whereas SMA represents the case employing the Smart Architecture. Finally, GIT (Game Interactivity Threshold) represents the maximum

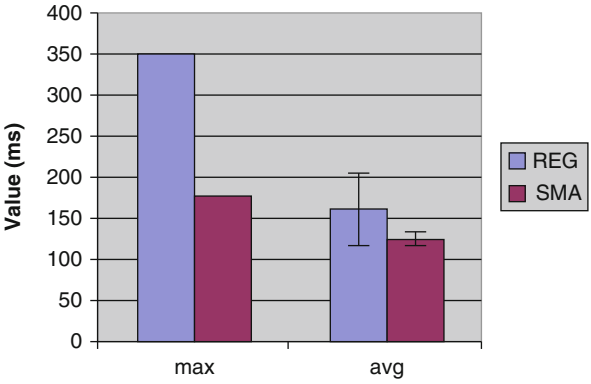

Fig. 5 SSM evaluation: statistical values for the delivery delay of game events (packets) from their generation to the GSS engaging the considered player

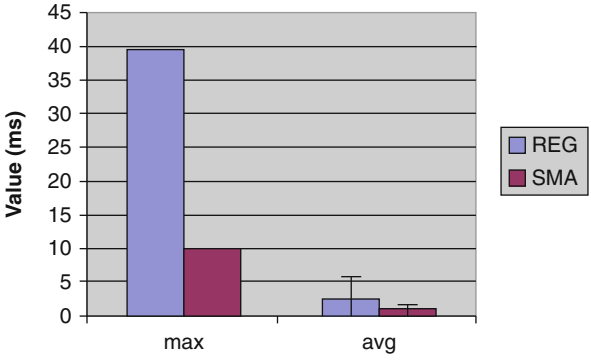

Fig. 6 Smart AP's evaluation: statistical values for the delivery delay of game events (packets) from the AP to the considered player

delay that a game event can experience from its generation to its delivery to the gaming device to preserve interactivity. As a reference, the threshold of 150 ms was considered based on related scientific literature for interactive online games (Pantel and Wolf 2002).

Focusing on the interactivity benefits provided by the first component of the Smart Architecture, i.e., the SSM coupled with a mirrored game server architecture, Fig. 5 shows the maximum and the average with the standard deviation bars of the delivery time that game events experience to reach the GSS that supports the considered player. The chart reports statistical values about the time elapsed from the generation of a game event and its delivery to the GSS that will then forward it to the considered player. Clearly, SMA outperforms REG, which demonstrates the effectiveness of FS in quickly synchronizing GSSs.

Figure 6 shows the impact of the wireless last hop on the game interactivity. In particular, the chart reports statistical values related to time elapsed from the moment when the AP receives the game update from its GSS and the moment when the considered player actually receives it on his/her mobile device. Basically, Fig. 6 highlights the effectiveness of the smart AP versus a traditional one.

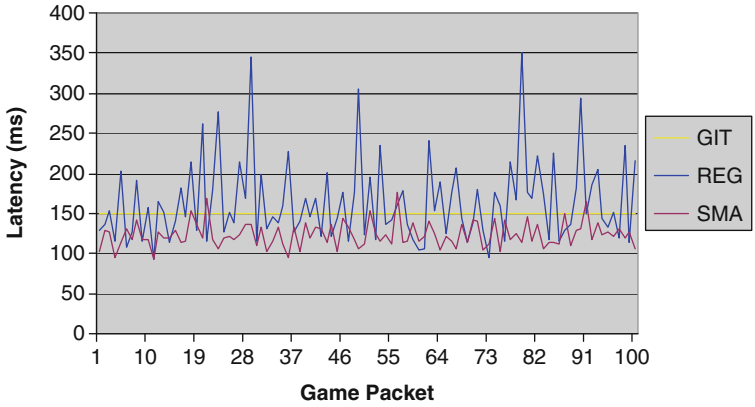

Fig. 7 Smart Architecture's comprehensive evaluation: instantaneous delivery latency of game events (packets) over the whole game platform

Indeed, Fig. 7 confirms that combining the two components of SMA will produce a sum of the positive benefits seen in Fig. 5 and Fig. 4. In particular, Fig. 7 reports the delivery time of 100 subsequent game events through the whole gaming platform. The regular configuration of the game platform (REG) is compared with the configuration including both FS and smart APs (SME). The outcome clearly demonstrates how SMA outperforms REG. Moreover, SMA is able to keep the game event delivery time almost always under the interactivity threshold (GIT).

Conclusions

The mobile gaming scenario is becoming very popular thanks to the widespread availability of mobile devices equipped with powerful networking and multimedia features. In this chapter we highlighted that mobile games produce time-sensitive traffic (i.e., traffic coupled with timing constraints) and require the network to meet these constraints in order to well support their peculiar characteristics: interactivity, consistency, fairness, scalability, and continuity.

Unfortunately, the support of mobile games is not as easy as one may think, as the service provided by the network might be not sufficient to meet the mobile game requirements. For this reason, many researchers focused their studies on this challenging scenario with the goal of finding solutions to support the peculiar characteristics of the mobile games. Throughout this chapter, we overview the network support solutions for mobile games and we focused on an interesting solution specifically designed to support the playability of games in the mobile scenario.

References

Armagetron. A Tron clone in 3D. http://armagetron.sourceforge.net/. Accessed 31 Oct 2014

G. Armitage, An experimental estimation of latency sensitivity in multiplayer Quake 3, in *Proc. of IEEE International Conference on Networks (ICON)* (Sydney, 2003), pp. 137–141

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 2004) (Portland, 2004)

G. Bianchi, Performance analysis of the IEEE 802.11 distributed coordination function. IEEE J. Select Area Commun. **18**(3), 535–547 (2000)

S. Biswas, R. Tatchikou, F. Dion, Vehicle-to-vehicle wireless communication protocols for enhancing highway traffic safety. IEEE Commun. Mag. **44**(1), 74–82 (2006)

L. Bononi, M. Di Felice, A cross-layered MAC and clustering scheme for efficient broadcast in VANETs, in *Proc. of 2-th IEEE International Workshop on Mobile Vehicular Networks (MOVENET 2006)* (Pisa, 2006)

L. Bononi, L. Donatiello, M. Furini, Real-time traffic in ad-hoc sensor networks, in *Proc. of the IEEE International Conference on Communications* (Dresden, 2009)

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, 2004)

Butterfly Grid Solution for Online Games. http://www.butterfly.net, Accessed 31 Oct 2014

A. Casteigts, A. Nayak, I. Stojmenovic, Communication protocols for vehicular ad hoc networks. Wireless Commun. Mobile Comput. John Wiley & Sons, Ltd. **11**(5), 567–582 (2011)

L.-J. Chen, T. Sun, G. Yang, M. Gerla, USHA: a simple and practical seamless vertical handoff solution, in *Proc. of IEEE Consumer Communications and Networking Conference (CCNC 2006)* (Las Vegas, 2006)

D.D. Clark, W. Fang, Explicit allocation of best-effort packet delivery service. IEEE/ACM Trans. Netw. **6**(4), 362–373 (1998)

M. Conti, L. Donatiello, M. Furini, Design and analysis of RT-ring: a protocol for supporting real-time communications. IEEE Trans. Ind. Electron. **49**(6), 1214–1226 (2002)

S. Corson, J. Macker, Mobile ad hoc networking (MANET): routing protocol performance issues and evaluation considerations (IETF RFC 2501, 1999)

E. Cronin, A.R. Kurc, B. Filstrup, S. Jamin, An efficient synchronization mechanism for mirrored game architectures. Multimed. Tools Appl. Springer **23**(1), 7–30 (2004)

E. Fasolo, R. Furiato, A. Zanella, Smart broadcast algorithm for inter-vehicular communication, in *Proc. of Wireless Personal Multimedia Communication (WPMC'05)* (IWS 2005, Aalborg, 2005)

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 ACM GDTW 2005*, (Liverpool, 2005), pp. 68–76

S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance. IEEE/ACM Trans. Netw. **1**(4), 397–413 (1993)

M. Furini, An architecture to easily produce adventure and movie games for the mobile scenario. ACM Comput. Entertain. **6**(2), 19:1–19:16 (2008)

M. Furini, Mobile games: what to expect in the near future, in *Proc. of EuroSis GAMEON Conference on Simulation and AI in Computer Games* (Bologna, 2007)

L. Gautier, C. Diot, Design and evaluation of MiMaze, a multi-player game on the Internet. in *Proc. of IEEE International Conference on Multimedia Computing and Systems (ICMCS'98)* (Austin, 1998)

C. Griwodz, State replication for multiplayer games, in *Proc. of ACM NetGames 2002* (Braunschweig, 2002)

M. Heusse, F. Rousseau, G. Berger-Sabbatel, A. Duda, Performance anomaly of 802.11b. in *Proc. of IEEE INFOCOM 2003* (San Francisco, 2003)

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, 2003)

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 ACM NetGames 2003* (Redwood City, 2003)

P. Kokkinos, C. Papageorgiou, E. Varvarigos, Multi-cost routing for energy and capacity constrained wireless mesh networks. Wireless Commun. Mobile Comput. John Wiley & Sons, Ltd. **11**(3), 424–438 (2011)

G. Korkmax, E. Ekici, F. Ozguner, U. Ozguner, Urban multi-hop broadcast protocol for inter-vehicle communication systems, in *Proc. of the 1st ACM Workshop on Vehicular Ad-hoc Networks (VANET 2004)* (Philadelphia, 2004)

G. Marfia, M. Roccetti, Dealing with wireless links in the era of bandwidth demanding wireless home entertainment, in *Proc. 6th IEEE International Workshop on Networking Issues in Multimedia Entertainment (NIME′10) – 2010 I.E. International Conference on Multimedia and Expo* (ICME′10, Singapore, 2010)

Movie Trace Files. http://www-tkn.ee.tu-berlin.de/research/trace/ltvt.html. Accessed 31 Oct 2014

C.H. Nam, S.C. Liew, C.P. Fu, An experimental study of ARQ protocol in 802.11b wireless LAN, in *Proc. of IEEE Vehicular Technology Conference (VTC 2003)* (Orlando, 2003)

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* (WONS, St. Moritz, 2005a)

A. Nandan, S. Das, B. Zhou, G. Pau, M. Gerla, Ad-torrent: digital billboards for vehicular networks, in *Proc. of IEEE/ACM International Workshop on Vehicle-to-Vehicle Communications*, vol. 1, (V2VCOM, San Diego, 2005b), pp. 286–294

Newzoo: Market Research for the Games Industry Online. http://www.newzoo.com. Accessed 31 Oct 2014

C.E. Palazzi, S. Ferretti, S. Cacciaguerra, M. Roccetti, A RIO-like technique for interactivity loss avoidance in fast-paced multiplayer online games. ACM Comput. Entertain. **3**(2) (2005a)

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 WIRELESSCOM 2005* (Maui, 2005b)

C.E. Palazzi, S. Ferretti, S. Cacciaguerra, M. Roccetti, Interactivity-loss avoidance in event delivery synchronization for mirrored game architectures. IEEE Trans. Mult. **8**(4), 847–879 (2006a)

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 Trans. Consum. Electron. **52**(4), 1280–1288 (2006b)

C.E. Palazzi, M. Roccetti, S. Ferretti, G. Pau, M. Gerla, Online games on wheels: fast game event delivery in vehicular ad-hoc networks, in *Proc. of 3rd IEEE V2VCOM 2007, IEEE Intelligent Vehicles Symposium 2007* (Istanbul, 2007)

C.E. Palazzi, A. Kaiser, N. Achir, K. Boussetta, A preliminary evaluation of backup servers for longer gaming sessions in MANETs, in *Proc. of the ACM International Workshop on DIstributed SImulation & Online gaming (DISIO 2010) – ICST SIMUTools 2010* (Torremolinos, 2010a)

C.E. Palazzi, M. Roccetti, S. Ferretti, An inter-vehicular communication architecture for safety and entertainment. IEEE Trans. Intell. Transp. Syst. **11**(1), 90–99 (2010b)

C.E. Palazzi, D. Maggiorini, From playgrounds to Smartphones: mobile evolution of a kids game, in *Proc. of the 8th IEEE Communications and Networking Conference (CCNC 2011)* (Las Vegas, 2011)

L. Pantel, C.L. Wolf, On the impact of delay on real-time multiplayer games, in *Proc. of the ACM 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video* (Miami, 2002)

K. Park, W. Willinger, Self-Similar Network Traffic and Performance Evaluation. John Wiley & Sons, Inc., New York 2002
C. Perkins, IP Mobility Support for IPv4. IETF RFC 3344 (2002)
Quake Forge Project. http://www.quakeforge.org/. Accessed 31 Oct 2014
Rakion. http://rakion.softnyx.net/. Accessed 31 Oct 2014
M. Roccetti, G. Marfia, A. Amoroso, An optimal 1D vehicular accident warning algorithm for realistic scenarios, in *Proc. IEEE Symposium on Computers and Communications* (ISCC′10) (Riccione, 2010)
F. Safaei, P. Boustead, C.D. Nguyen, J. Brun, M. Dowlatshahi, Latency driven distribution: infrastructure needs of participatory entertainment applications. IEEE Commun. Mag. **43**, 106–112 (2005) (Special Issue on "Entertainment Everywhere: System and Networking Issues in Emerging Network-Centric Entertainments Systems", Part I)
Ultima Online. http://www.uo.com. Accessed 31 Oct 2014
P.J. Wan, K. Alzoubi, O. Frieder, Distributed construction of connected dominating set in wireless ad hoc networks, in *Proc. of IEEE INFOCOM 2002* (New York, 2002)
K. Xu, M. Gerla, L. Qi, Y. Shu, TCP unfairness in ad hoc wireless networks and a neighborhood RED solution. Wireless Netw. ACM. **11**(4), 383–399 (2005)
G. Xylomenos, G.C. Polyzos, TCP and UDP performance over a wireless LAN, in *Proc. of IEEE INFOCOM '99* (New York, 1999)
X. Yang, J. Liu, F. Zhao, N. Vaidya, A vehicle-to-vehicle communication protocol for cooperative collision warning, in *Proc. of ACM 1st Annual International Conf. on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous '04)* (Boston, 2004)
C. Yu, B. Lee, H.Y. Youn, Energy efficient routing protocols for mobile ad hoc networks. Wireless Commun. Mobile Comput. John Wiley & Sons, Inc. **3**(8), 959–973 (2003)
A. Zanella, G. Pierobon, S. Merlin, On the limiting performance of broadcast algorithms over unidimensional ad-hoc radio networks, in *Proc. of IEEE WPMC04* (Abano Terme, 2004)
K. Zhu, D. Niyato, P. Wang, E. Hossain, D.I. Kim, Mobility and handoff management in vehicular networks: a survey. Wireless Commun. Mobile Comput. John Wiley & Sons, Inc. **11**(4), 459–476 (2011)