# Interactive Mobile Gaming over Heterogeneous Networks

Claudio E. Palazzi, *Member, IEEE*

***Abstract*—Interactive, mobile online games have recently become popular thus receiving the attention of researchers. Mobile games represent a particularly interesting case because of the proliferation of smart personal devices and the increasing availability of high speed wireless access points. With this vision, we propose a holistic solution that enables a top quality online gaming experience regardless whether the player is wired, wireless, or even mobile. Main components of this solution are: i) a synchronization mechanism based on an Internet server overlay that enables interactivity, fairness, and scalability, and ii) a smart access point able to support efficient coexistence between elastic (download) and real-time (game) traffic.**

***Index Terms*—Interactivity, Mobile Online Games, Wireless Network.**

## I. Introduction

Wireless access points are rapidly increasing in number, providing people with connectivity in almost all buildings they enter (e.g. home, work place, cafeteria, etc.) and all streets they walk and drive in. In this context, mobile games are gaining attention as their users are increasing in number [1], [2], [3], [4]. From a research point of view, they represent a very interesting and challenging topic especially in wireless environments and when involving many players simultaneously sharing the same virtual arena.

In this paper we focus on highly interactive mobile games, i.e., games based on fast action evolution on screen, requiring prompt reactions by the user and played through an handheld device with wireless communication capabilities (e.g., PDA, handheld console, smart cellphone); in particular we discuss their main requirements.

More in detail, in order to ensure optimal performance to players we split the problem into two sub-parts which requires specific solutions applied by different subjects. The first sub-part regards the communication and synchronization among game servers and represents the portion of the total connectivity that can be handled by the online game service provider (either directly, or through ISP-domain managers). The second sub-part, is concerned with the links between game servers and their engaged players, thus including also the last-mile wireless hop, which generally corresponds to the bottleneck of the connection and may be source of large queuing delays.

To obtain a holistic solution we propose to proceed through successive steps and address the two sub-problems independently. Solutions discussed in this paper complement each other since their scopes are detached (even if connected) and, although they generate the best performance results when combined, they produce benefits even if singularly applied.

In particular, for the first sub-part we propose to exploit a hybrid architecture combining both the advantages of client-server and peer-to-peer paradigms. This solution deploys over the network a constellation of communicating replicated *Game State Servers* (GSSs), each of which locally maintains a vision of the game state [5]. It goes without saying that, within this architecture, an efficient event synchronization scheme among GSSs needs to be employed to guarantee a consistent and responsive evolution of the game state. To this aim, the semantics of the game can be put to good use in order to increase the interactivity degree provided to the player: discarding game events that are superseded by others can free network resources thus speeding up the delivery of fresh game events [6].

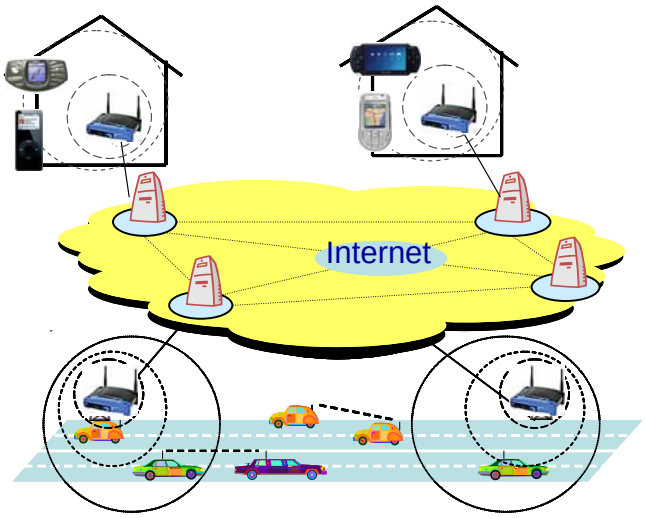


Fig. 1. Hybrid architecture for distributed game entertainment in heterogeneous scenarios with mobile users (even car passengers).

Manuscript received October 22, 2007. This work was supported in part by the Italian MIUR under the DAMASCO initiative.

C. E. Palazzi is with the Dipartimento di Matematica Pura e Applicata of Università degli Studi di Padova, Via Trieste 63, 35121 Padova, Italy, phone: +39-049-827-1426; fax: +39-049-827-1499; e-mail: cpalazzi@math.unipd.it.

However, even if this scheme is proficient in maintaining a high degree of responsiveness among game servers, still problems may arise at the edges of the considered topology (see Fig. 1), where users in their homes or cars may be engaged in an online gaming through an access point (AP).

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 walking/driving along a road. The wireless medium, in fact, is naturally prone to be shared by several contemporary users who may interfere with each other. In addition to the problem discussed above, it has been recently demonstrated how TCP-based elastic flows (e.g., download) can harm the performance of UDP-based real-time flows (e.g., online gaming) as TCP continuously probes the channel for more bandwidth, thus eventually generating queues (delays) on the connection [7].

To address this issue, a *smart AP* can take advantage of its knowledge about available wireless network resources and the on-going traffic in order to appropriately limit TCP's advertised windows so as to smoothen the network traffic progression and avoid queuing delays that would jeopardize the interactivity of online game applications and, in general, of any real-time application [8].

The remainder of the paper is organized as follows. Section II describes main networking issues related to mobile gaming. In Section III we discuss our proposed architecture to support interactivity. Section IV presents an experimental evaluation of the proposed architecture. Finally, Section V concludes this paper.

## II. Mobile Gaming: Beyond the Fun

In this section we discuss and provide a real case example of main networking challenges related to mobile gaming.

### A. Main Networking Requirements

Under a networking point of view, online games are characterized by five main requirements which are intrinsically correlated: interactivity, consistency, fairness, scalability, and continuity. As discussed in the following, one of them, i.e., interactivity, is particularly important also for its influence on the other requirements.

**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. In order to assure 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 [9]. Unfortunately, variable congestion conditions in Internet may suddenly slow down the game fluency on the screen. Moreover, players in the same virtual arena could be so numerous that some game server may experience impulsive computational load and loose interactivity. These problems are obviously amplified when plunged into a wireless, mobile scenario.

**Consistency** regards the simultaneous uniformity of the game state view in all nodes belonging to the system. 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 the agents before moving toward the next round, for sure grants absolute consistency but, on the other hand, impairs the system interactivity. A trade-off between these two attributes needs thus to be found in order to develop a proficient game platform.

**Fairness**, or (**networking fairness**), is related to provide every player with the same chances of winning the match, 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 recently been demonstrated how increasing the interactivity degree of the game platform may lead also to improved fairness [10].

**Scalability** regards the capability of the system in providing efficient support to a large community of players. Indeed, it is primary interest of game companies to have huge 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 [11]. 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.

**Continuity** is concerned with having game sessions not interrupted by disconnections, handoffs, or any other mobility-related issue. Indeed, players would be very frustrated by having their match continuously interrupted and re-started (maybe after a while) with new players. To this aim, the best solution is certainly that of designing games featured with short game sessions. Moreover, quick/smart handoff mechanisms could be applied to smoothen the effects of mobility [12].

### B. Delay Matters: A Practical Example

As a practical example of problems related to (different) delays in online games, we present in Fig. 2 the frame evolution of an Armagetron game session [13]. 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 degree angles. The goal of the game is that of having all other players 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. 2, there are subsequent frames as seen by a player, named *Cla* (blue cycle and wall), who connected to a certain game server with a 180 ms of round-trip time (RTT). Instead, frames on the right column correspond to the game as seen by another player, named *Eu* (green cycle and wall), who is connected to the same server with only 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 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.

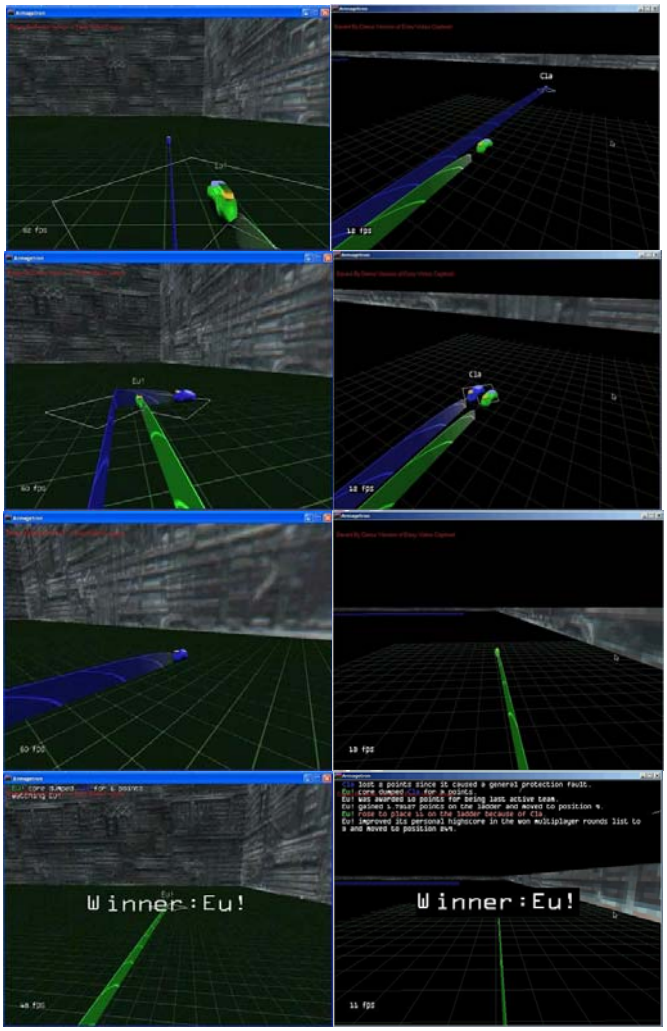


Fig. 2. Frame sequence of an Armagetron game session: *Cla*'s view (left column, blue player) vs *Eu*'s view (right column, green player); evident lag differences generate inconsistencies and unfairness.

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 evolving in advance with respect to *Cla*, as the former is much closer to the game server than the latter.

## III. Smart Architecture: A Holistic Solution

In this section we describe the two components of the proposed holistic solution: 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. For the sake of clarity, we name this holistic solution *Smart Architecture*.

### A. Fast Synchronization over a Hybrid Architecture

Mirrored game server architectures represent a hybrid solution which efficiently embraces all the positive aspects of both centralized client-server and fully distributed architectures [5]. 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 game events of its engaged players, updates the game state, and periodically 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 [14]. 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 experienced delay [15]. 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 various tradeoff among interactivity, consistency, fairness, scalability, and continuity.

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. Moreover smart solution can be devised to speed up this synchronization process. Indeed, taking inspiration from the Active Queue Management approach (e.g., RED [16], RIO [17]) in case of incipient congestion in best effort networks, the synchronization mechanism among GSSs could exploit the semantics of the game to discard few game packets to preempt interactivity loss when intense network traffic or excessive computational load is slowing down some GSS [6], [18].

For the sake of clarity, in the rest of the paper we refer to this synchronization mechanism able to discard game events as *Fast Synchronization* (FS).

Similar to RED/RIO, FS 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 we name *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 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.

Discarding superseded events for processing fresher ones may be of great help for delay-affected GSSs, achieving high interactivity degree without compromising consistency.

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. 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-superseded events can be done without consistency-related consequences only for a category of games where little inconsistencies are not highly deleterious for players' fun (e.g., fast-paced games).

In Fig. 3 we depict the two discarding functions of FS. 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. In the graph, the y-axis represents the dropping probability corresponding to the *avg_GTD* indicated by the x-axis. Focusing on superseded events, for values of *avg_GTD* in [0, $min_o$) the mechanism performs normal operations, with no packet drops, while in [$min_o$, $max_o$) superseded packets are discarded with a computed probability, and finally in [$max_o$, ∞) all superseded packets are thrown away. The intervals [0, $min_v$), [$min_v$, $max_v$) and [$max_v$, ∞) define the corresponding phases for non-superseded game 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 or forwarding them, thus helping in restoring an adequate level of time interaction between servers.

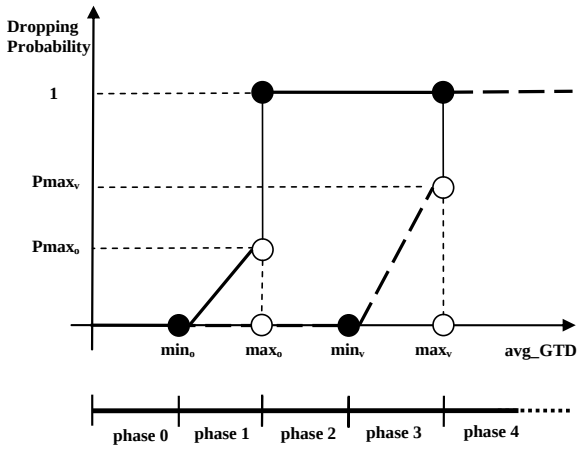


Fig. 3. Discarding probability functions.

### *B. A Smart AP Can Save the Interactivity Patrimony*

Even if FS coupled with a Mirrored Game Server architecture is proficient in maintaining a high degree of responsiveness among game servers (i.e., 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. 1). 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 FS. The applications run in this context may vary and some of these may be particularly harmful toward real-time traffic (online games but also video-streaming, video-chats, etc.). In particular, TCP-based FTP application for downloading files increases queuing delays to such an extent that interactivity may be completely jeopardized [7].

To this aim, to our Smart Architecture we also add a *smart AP* that aims at achieving best performance for both elastic and real-time applications [8] by appropriately limiting the advertised window for TCP flows. This way, a solution to the tradeoff relationship existing between TCP throughput and real-time application delays can be found.

To this aim, it is evident how a technique that exploited existing features of standard protocols could be easily implemented in a real scenario. Moreover, an optimal tradeoff between throughput and low delays could be achieved by maintaining the sending rate (hence, the sending window) 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 would be maximized by the absence of packet loss, while the delay would be minimized by the absence of queuing. This can be achieved through limiting the aggregate bandwidth utilized by TCP flows just below the total capacity of the bottleneck link diminished by the portion of the channel occupied by the simultaneous UDP-based real-time traffic.

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

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

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 exploiting to this aim the TCP's advertised window. Simply, the actual sending window of a TCP flow is determined as the minimum between the congestion window and the advertised window; thereby, having the AP appropriately modifying the advertised window of passing-through TCP flows would limiting the factual sending rate of TCP flows under *TCPubrate(t)*.

To compute (1), a comprehensive knowledge of all the flows that are transiting through the bottleneck (i.e., the last hop links) is needed, i.e., 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. However, this information is possessed by the AP as all application flows have to pass through it.

## IV. Experimental Evaluation

In this section we evaluate through simulations the interactivity improvement ensured by the discussed holistic solution, i.e., the Smart Architecture.

### A. Simulation Assessment

To evaluate the discussed solution, we have generated game traffic through a mathematical model so as to have a lognormal distribution of the GTD values as suggested by [19]. Then we injected this traffic into a simulative environment based on the well known NS-2 simulator [20]. In particular, we have simulated a scenario representing the topology depicted by Fig. 1, considering seven interconnected GSSs with a network latency between the two farthest GSSs of 90ms, i.e., an intra-continental GSS overlay network. To synchronize the respective game states, every 30ms each GSS transmit to the other GSSs game updates related to their engaged clients. Finally, only 90% of simulated game events could become superseded by successively generated game events. This represents a realistic setting for a vast plethora of possible games where critical game events that cannot become superseded 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 [21].

Results were gathered collecting the total latency experienced by game events reaching one of the clients connected through a IEEE802.11g 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 [22].

Interactivity represents the main feature determining the perceived quality of an online game service, even when players employ PDAs or cellphones. Therefore, in the next subsection we report results related to the GTD experienced by game events delivered to the considered mobile gaming device. The smaller the GTD, the higher the perceived interactivity.

In particular, in the following charts, *REG* represents the case where a regular synchronization scheme is adopted by GSSs, whereas *SMA* represents the case employing the discussed Smart Architecture. Finally, *GIT* stands for *Game Interactivity Threshold* and represents the maximum delay that a game event can experience from its generation to its delivery to the gaming device in order to still consider the on-going gaming session as interactive. As a reference, we have taken 150ms as the GIT, since this is the value suggested by related scientific literature for interactive online games [9].

### B. Results

Focusing on the interactivity benefits provided by the first component of SMA, i.e., the FS coupled with a Mirrored Game Server architecture, we show in Fig. 4 the maximum, the average, and the standard deviation of the delivery time that game events experience to reach the GSS that supports the considered player. Basically, 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 demonstrate the effectiveness of FS in quickly synchronizing GSSs.

Instead, in Fig. 5 we evaluate 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 her/his mobile device. Basically, Fig. 5 highlights the effectiveness of the smart AP versus a traditional one.

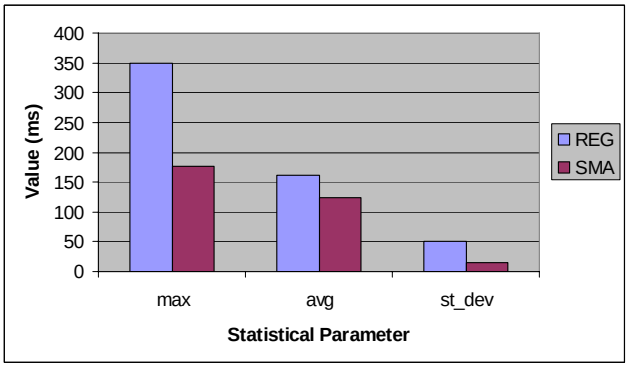


Fig. 4. Fast Synchronization's evaluation: statistical values for the delivery delay of game events (packets) from their generation to the GSS engaging the considered player.

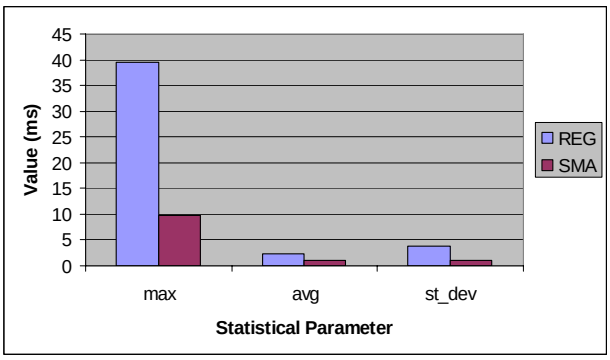


Fig. 5. Smart AP's evaluation: statistical values for the delivery delay of game events (packets) from the AP to the considered player.

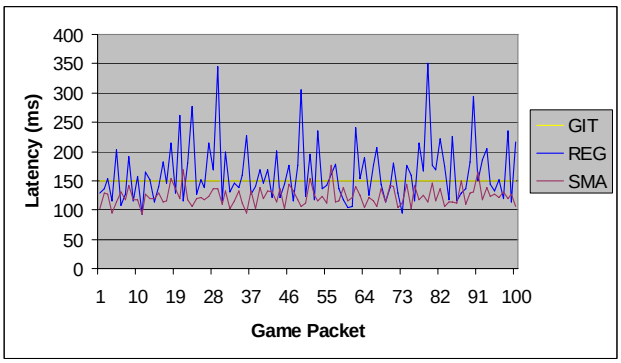


Fig. 6. Smart Architecture's comprehensive evaluation: instantaneous delivery latency of game events (packets) over the whole game platform.

Clearly, when the smart AP is employed, the delay is sensibly smaller with respect to the regular case, thus demonstrating its ability in avoiding queuing delays.

We expect that combining the two components of SMA will produce a sum of the positive benefits seen in Fig. 4 and Fig. 5. Indeed, Fig. 6 confirms this expectation. In particular, Fig. 6 reports the delivery time of 100 subsequent game events that have to be delivered to the considered player 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), exceeding that threshold only sporadically and just slightly. Conversely, with REG, delivery delays frequently and significantly exceed the GIT.

This demonstrates the ability of our Smart Architecture solution in factually ensuring a high interactivity degree to mobile online gamers.

## V. Conclusion

Mobile online games are increasing their popularity also because of the proliferation of personal devices with wireless connectivity (e.g., smart mobile phones with Wi-Fi technology), and the increasing availability of high speed wireless APs. With this vision, we have discussed two solutions that could be employed solo or together to improve the interactivity degree of online gaming, even if the user is playing through a mobile device. These solutions are, respectively: i) a synchronization mechanism, named FS, for game servers of a hybrid architecture that exploits the semantics of the game to support interactivity and ii) a smart AP able to smooth the transmission rate of elastic traffic (download) so as to avoid queuing delays that would disrupt the interactivity patrimony created by FS.

Reported experiments confirm the ability of each of the aforementioned solutions in reducing the game event delivery time. This result is further improved when the two solutions are combined into the proposed Smart Architecture.

Moreover, the Smart Architecture discussed in this paper is proficient also at ensuring scalability and fairness as: i) having several GSSs in the hybrid architecture is helpful to the aim of supporting scalability as discussed in Section III-A; ii) ensuring a higher degree of interactivity may be put to good use to improve also fairness and scalability (see Section II-A)

Therefore, we can claim that Smart Architecture is a holistic solution able to support interactivity, fairness, and scalability. Finally, we suggest to enhance the architecture also with a smart/quick handoff mechanism as the one suggested in [12] to address mobility/continuity when considering games with longer general sessions.

## Acknowledgment

The author of this manuscript wishes to express his deep gratitude towards Prof. Marco Roccetti, Dr. Giovanni Pau, Prof. Mario Gerla, Prof. Stefano Ferretti, and Dr. Stefano Cacciaguerra; part of this manuscript is based on data gathered and/or ideas emerged while collaborating with them at different times.

## References

[1] C. Griwodz, "State Replication for Multiplayer Games", *in Proc. of NetGames2002*, Braunschweig, Germany, 2002.

[2] 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.

[3] 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*, Washington, DC, USA, May 2002.

[4] 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, Turkey, Jun 2007.

[5] 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.

[6] C. E. Palazzi, S. Ferretti, S. Cacciaguerra, M. Roccetti, "Interactivity-Loss Avoidance in Event Delivery Synchronization for Mirrored Game Architectures", *IEEE Transactions on Multimedia*, vol. 8, no. 4, pp. 847-879, Aug 2006.

[7] C. E. Palazzi, G. Pau, M. Roccetti, M. Gerla, "In-Home Online Entertainment: Analyzing the Impact of the Wireless MAC-Transport Protocols Interference", *IEEE WIRELESSCOM2005*, Maui, HI, USA, Jun 2005.

[8] 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*, vol. 52, no. 4, pp. 1280-1288, Nov 2006.

[9] 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.

[10] 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 GDTW 2005*, Liverpool, UK, pp. 68-76, Nov 2005.

[11] Rakion. [Online]. Available: http://www.rakion.com/

[12] L.-J. Chen, T. Sun, G. Yang, M Gerla, "USHA: a Simple and Practical Seamless Vertical Handoff Solution", in Proc. of the 2006 IEEE Consumer Communications and Networking Conference (CCNC 2006), Las Vegas, NV, USA, Jan 2006.

[13] Armagetron: a Tron Clone in 3D. [Online]. Available: http://armagetron.sourceforge.net/

[14] F. Safaei, P. Boustead, C. D. Nguyen, J. Brun, M. Dowlatshahi, "Latency Driven Distribution: Infrastructure Needs of Participatory Entertainment Applications", *IEEE Communications Magazine, Special Issue on "Entertainment Everywhere: System and Networking Issues in Emerging Network-Centric Entertainments Systems"*, Part I, May 2005.

[15] 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, CA, USA, 2003.

[16] S. Floyd, V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397-413, 1993.

[17] 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.

[18] 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.

[19] K. Park, W. Willinger, *Self-Similar Network Traffic and Performance Evaluation*, Wiley-Interscience, 1st Edition, 2000.

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

[21] G. Armitage, "An Experimental Estimation of Latency Sensitivity in Multiplayer Quake 3", *in Proc. of ICON*, pp. 137-141, Sydney, Australia, Sep-Oct 2003.

[22] Movie Trace Files. [Online]. Available: http://www-tkn.ee.tu-berlin.de/research/trace/ltvt.html