Mobile Games Through the Nets: a Cross-Layer Architecture for Seamless Playing

Vittorio Ghini
Department of Computer Science
University of Bologna
Mura Anteo Zamboni 7
40127, Bologna, Italy
ghini@cs.unibo.it

Stefano Ferretti
Department of Computer Science
University of Bologna
Mura Anteo Zamboni 7
40127, Bologna, Italy
sferrett@cs.unibo.it

Fabio Panzieri
Department of Computer Science
University of Bologna
Mura Anteo Zamboni 7
40127, Bologna, Italy
panzieri@cs.unibo.it

ABSTRACT
Mobile games are characterized by the need to exploit responsive and reliable communication protocols, in order to guarantee that players can perceive a fluid and dynamic game-play evolution. Current approaches to the provision of support for this kind of real-time applications are generally based on the use of a single wireless network, as the transit from a network to another (or even changing the network service provider, while using the same network interface) causes a temporary reconfiguration of both the operating system and the network interface it is exploiting. This indeed results in a pause of game events processing at the application (game) level, which may give rise to a jerky game evolution. In order to overcome this problem, in this paper we propose the use of a cross-layer architecture which allows mobile devices to concurrently exploit different network interfaces so as to offer seamless and uninterrupted communications among the entities involved in the game. A smart adaptation algorithm balances the load between different network interfaces, based on the performances these are offering. The proposed approach can be viably implemented in client/server as well as in mirrored game server architectures. Experimental results conducted on a real field confirm both the adequacy and the effectiveness of our proposal.

Categories and Subject Descriptors
C.2.1 [**Computer Systems Organization**]: COMPUTER-COMMUNICATION NETWORKS – Network Architecture and Design, Wireless communication; K.8.0 [**Computing Milieux**]: PERSONAL COMPUTING – General, Games

General Terms
Algorithms, Design, Experimentation.

DISIO 2010, March 15, 2010, Torremolinos, Malaga, Spain.

Keywords
Mobile gaming, wireless networks, cross-layer architectures.

1. INTRODUCTION
Online games have currently concentrated on the production of novel distributed solutions that enable players to communicate and interact in a responsive and scalable way [5, 8, 16]. Concurrently, new generations of hand-held game consoles have been developed that support multi-player gaming over wireless networks. Current proposals for the development of mobile games are usually based on the use of a single, pre-determined wireless network. For instance, in most cases it is assumed that the hand-held terminal utilizes Wi-Fi or cellular networks [2, 15]. Recent advances in the design of mobile terminals allow users to select the wireless network to use in order to connect to Internet services. Such choice may well be made at game configuration time, before starting the playing session. However, it is reasonable to assume that during such session the communication flows only through a single net.

The use of a single net is characteristic, from a certain point of view, makes sense, as responsiveness is the main requirement of all real-time games, and a single net enables the development of simple techniques that meet that requirement [1, 2, 4, 5, 17]. Once the game has started, players must perceive its evolution without any interruptions, as if the game were executed locally. Hence, the application requires that adequate distributed infrastructures be available that meet effectively application specific network bandwidth and latency requirements. Specifically, game updates should be delivered to players within some strict time deadlines; missing these deadlines entails that the game is not properly supported. From this standpoint, it is clear that changing the communication network while playing may introduce a temporary disconnection that can jeopardize the game-play. This problem can be overcome by exploiting concurrently all the available networks, rather than a single one, so as to i) select the "best available" (i.e. lowest latency, lowest error rate) one to disseminate game events to the game server/other players; ii) balance the load among different networks.

In view of the above observation, this paper presents a cross-layer architecture which allows to use simultaneously all the available

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8656
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8656

network interface cards a mobile terminal is equipped with. This architecture forwards each given game event generated at the mobile node through the best available network interface. Same holds for messages coming from the game server towards the mobile node. Policies for load balancing and recovery purposes can be devised. This architecture, that we term Robust Wireless Multi-Path Channel (RWMPC) is constructed out of two proxies, i.e. a client proxy installed on the mobile terminal, and a server proxy, external to the wireless access networks, which represents the mobile node to the game server (and other game entities); it is responsible for maintaining the communication continuity with the mobile node and integrates the necessary functionalities to overcome firewalls.

The architecture described in the paper enables the seamless dissemination of UDP datagrams, which are typically exploited in online games to distribute game events [4, 5, 14, 15]. However, this architecture can be also deployed in order to support TCP flows, when these are used in online games [6, 11]. In addition, our architecture can be exploited for other real-time multimedia applications, e.g. VoIP [9]. In this paper, we show how our architecture can proficiently manage mobile game traffic. Specifically, this architecture is meant to support online games that require an access to the Internet, in order to reach a game server or other nodes involved in the game session. The architecture is not meant for games deployed over mobile ad-hoc networks (MANETs), which exploit a single wireless network typology (e.g. Wi-Fi), and where peers are organized as an overlay, based on the geographical location of nodes themselves.

Another consideration worth of mention is that nodes involved in the online game are not specifically required to adopt the approach we are proposing. This mechanism is an additional service (provided with the game) that gamers may think to exploit, when they are playing through their hand-held terminal, while moving. Fixed nodes may continue to employ the classical communication methods, typical of real-time applications.

To the best of our knowledge, this is the first proposal that employs cross-layer techniques, and multi-path communication in general, for the support of online games. Results coming from an experimental scenario confirm the adequacy of our proposal. In particular, we show that our approach allows mobile devices to seamlessly pass though different networks, e.g. Wi-Fi and UMTS, exploiting them all, while keeping active the end-to-end communication with their game server. This augments the reliability of the communication and reduces the delivery times in case of network congestion. The actual implementation of the developed prototype exploits modifications of the (linux) operating system kernel. We are currently exploring the possibility of implementing a prototype on Symbian-based devices.

The remainder of the paper is organized as follows. Section 2 introduces the architecture we propose. Section 3 discusses the experimental assessment we have performed and the results we obtained. Finally, some concluding remarks are discussed in Section 4.

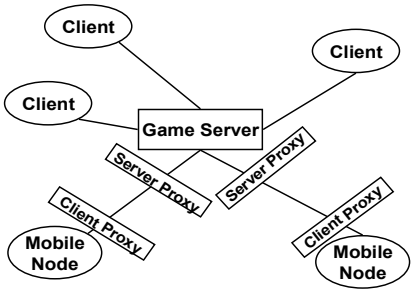

Figure 1. Client/server architecture.

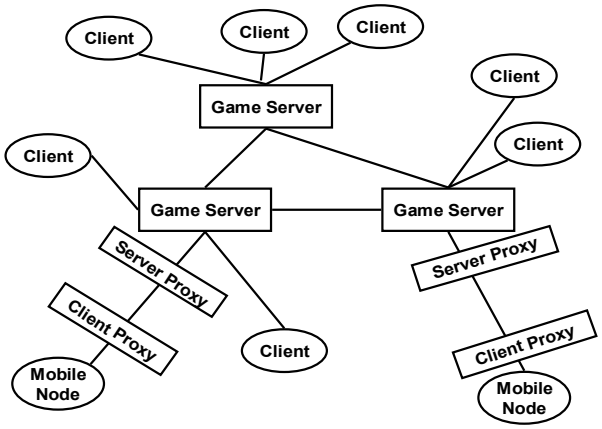

Figure 2. Mirrored game server architecture.

2. SYSTEM ARCHITECTURE

Our architecture provides mobile nodes, executing a distributed game application, with a seamless communication service, ensuring uninterrupted availability of communications and high interactivity. The architecture is located at the ISO/OSI session level, but utilizes a cross-layer approach that involves the lowest levels of the network stack. The basic idea is to exploit the multiplicity and heterogeneity of the network interfaces which may be available on each mobile node.

An overview of the software structuring of the vast majority of game applications is depicted in Figures 1 and 2. These figures show that, in essence, the software of a distributed game application can be structured as either i) a conventional classic client/server application (Figure 1) or ii) a so-called "mirrored game server" application (Figure 2). With the former structuring, a central server maintains a unique copy of the game state; then, clients send game events generated by players to the server that collects them all, computes a novel game state and sends the update to the clients. The latter structuring is a generalization of the client/server approach, which guarantees higher scalability and fault-tolerance. In essence, in "mirrored game server" applications several game servers are distributed over the net, which maintain a replicated game state. Clients connect with the nearest game server and communicate with it.

Servers employ synchronization algorithms to maintain the consistency of the game state [12, 14]. Our RWMPC architecture

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8656
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8656

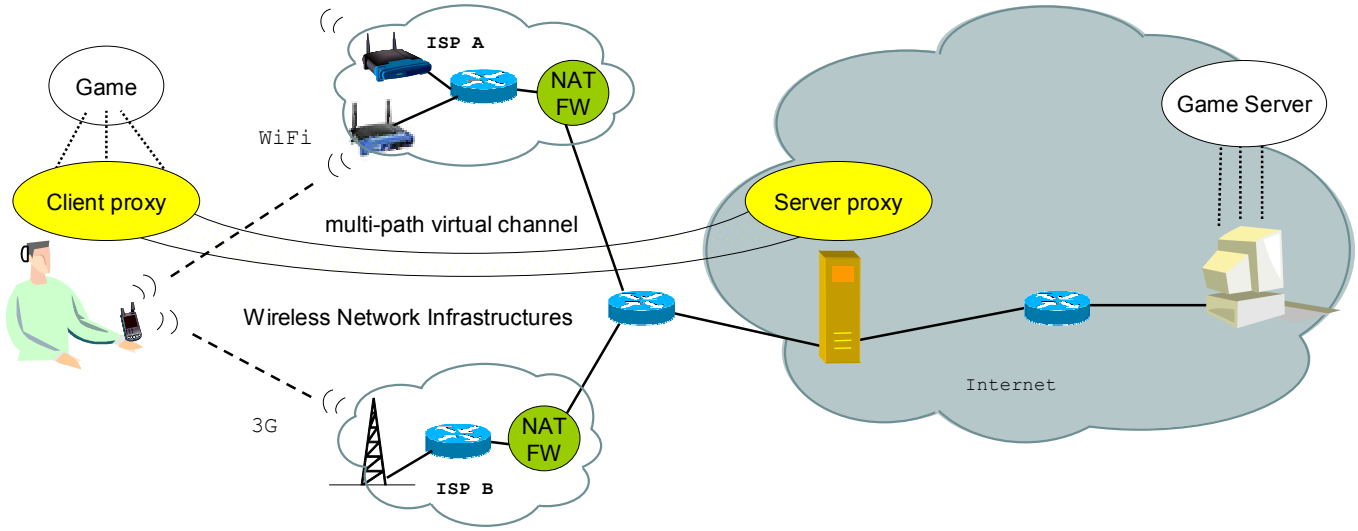

Figure 3. Path between the mobile node and its own game server.

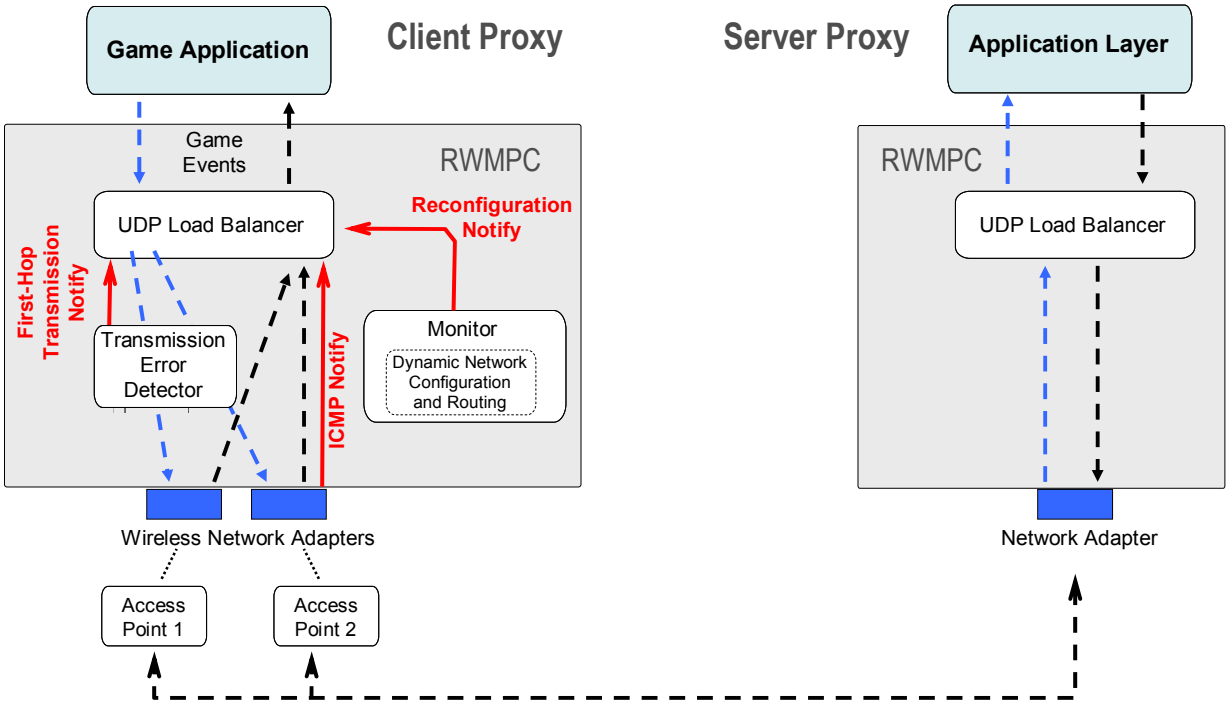

Figure 4. The overall RWMPC middleware system.

allows clients to be fixed nodes as well as mobile nodes. The peculiar aspect is that the interactions between the mobile node and its own game server pass through two proxies, one installed on the mobile terminal, while the other is assumed to be external to the wireless access network (and not shadowed by a firewall).

Figure 3 illustrates the path between a mobile node and its own game server in the RWMPC architecture. The client proxy is a local communication manager that maintains the communication, through the different network interface cards, with the server proxy. The server proxy represents the mobile node to the game server. It maintains the communication continuity with the mobile node, and integrates the necessary functionalities to overcome the presence of firewalls. On the mobile node, a multi-path virtual channel between the client and the server proxies is maintained.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8656
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8656

A characteristic of our approach is that it differs from the common Always Best Connected (ABC) model; in this model, the best available network is identified and used until its performance degrades [11]. In contrast, in our approach all the available Network Interface Cards (NICs) are used simultaneously, differentiating the choice of a network from datagram to datagram, and introducing the ability to switch to the most suitable path depending on the characteristics of the datagram [9]. This enables to create policies for load balancing and recovery in order to maximize the available bandwidth and decrease the loss rate.

It is worth mentioning that due to the possible use of different wireless networks, it is likely that game events generated at the same node, by passing through different networks, may be delivered out of order. However, UDP-based multiplayer online games typically deal with issues of event ordering at the (game) application level. It is in fact current practice to add timestamps to game events in online games. Each game entity (server, client) is thus able to understand whether it has not received some message from a specific node or whether there are late events. Depending on the game semantics and on the game management rules, the server may decide to wait for late messages, rather than considering them as obsolete, and drop them [5].

Such feature of exploiting multiple networks is even more proficient when we consider the converse path, i.e. game updates which are periodically sent by the game server to mobile nodes. If a mobile node receives a novel game update, without having received some previous messages, these latter may be automatically dropped, since they refer to old views of the game state. (in such a case, dead reckoning and other latency hiding approaches can be exploited so as to avoid jerky renderings [13]).

In the rest of the section, we will describe the multipath virtual channel RWMPC that can be created between the mobile node and the game server.

2.1 Multi-Path Virtual Channel

The RWMPC mechanism uses a cross-layer approach in order to timely propagate, from the MAC layer up to the application layer, the information concerning the transmission status of UDP datagrams generated by the game application. This cross-layer approach allows the session layer to carrying out a timely error recovery procedure, and decide which wireless interface, among those available on the mobile host, is to be used for the transmission of the successive UDP datagrams. RWMPC consists of the following components (see Figure 4):

1) Transmission Error Detector (TED): it operates at the MAC layer. Its principal responsibility is to monitor each single UDP datagram sent over an active wireless link in order to detect whether that datagram has been successfully received by the access point or discarded by the MAC layer. TED notifies the UDP Load Balancer component (see below) of UDP datagram rejections ("First-hop Transmission Notification" in Figure 4).

2) Monitor: it is responsible for monitoring and configuring the wireless interfaces, and reporting which interfaces are actually active. In more detail, the Monitor dynamically configures the wireless network interfaces and the routing rules. Once a wireless network interface has been either fully configured or disabled as a consequence of a communication error, a notification to ULB (see below) is sent ("Reconfiguration Notification" message in Figure 4).

3) UDP Load Balancer (ULB): this component receives game events generated at the application layer, encapsulates them in UDP datagrams, and sends the datagrams to the game server. To this end, ULB uses a UDP socket for each active wireless network interface of the mobile host (each UDP socket is connected to a wireless interface using the *bind* primitive). ULB receives three types of notifications (the non dashed lines in Figure 4): the first type of notification comes from the Monitor component, which informs ULB that either a new network interface is available or a given active interface has been disabled. For each new available interface, ULB generates a UDP socket and binds that socket with the new interface. In contrast, for each interface that has been disabled, ULB closes the corresponding UDP socket and considers as lost the UDP datagrams that have been sent through that socket and for which no "First-hop Transmission Notification" from TED has been received. The second type of notification comes from the ICMP protocol which may notify when the server proxy (or its destination port) is unreachable (through that network interface). The third type of notification is generated by TED to inform ULB that a UDP datagram has been either successfully received by the access point or permanently discarded. If no TED notification arrives at the ULB before 30 msec after a UDP datagram has been sent, the ULB assumes that datagram has been lost.

Based on these three types of notifications, ULB decides if a lost UDP datagram has either to be retransmitted using another wireless network interface among those available on the mobile host, or permanently discarded. In addition, based on both the monitoring notifications of the sent UDP datagrams and the ICMP notifications, ULB selects the wireless network interface to be used for sending successive UDP datagrams. (The algorithm employed to select the specific network is outlined in the next subsection.)

The server proxy is simpler than the client proxy. It consists of a simplified version of the ULB component above that controls only one wired network interface. Its principal activity consists of recording the IP address of the mobile host wireless interface from which it has received the last UDP datagram. That IP address is used for sending successive response UDP datagrams to the mobile host.

2.2 Network Selection

When using certain wireless network interface cards, TED does not produce notifications in case datagrams have been discarded. This is due to the fact that the firmware of some wireless network interface does not report to the MAC layer that a given frame has been discarded; rather, most of them inform the MAC layer when a frame has been successfully delivered to an access point. Hence, ULB assumes that a datagram has been discarded by the wireless link when either a lost datagram notification is received from TED, or a timeout (equal to 30 ms) expires after the datagram has been delivered to the socket without any notification from TED of successful datagram transmission.

ULB maintains a list of datagrams that are to be transmitted, and a global counter of the transmission attempts. This counter is increased at any time the ULB attempts to send a datagram. Each datagram contains a "transmission sequential number" field.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8656
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8656

Before such transmission attempt, the ULB increases the global counter and assigns such value to the "transmission sequential number" field of that datagram. The algorithm that selects the most suitable wireless network interface is as follows.

At the mobile node, if only one NIC is available, which is correctly configured and working, the ULB uses that NIC. Instead, if both the Wi-fi and UMTS NICs are available to be used, the ULB tries to send the datagram using the Wi-fi network interface. If a notification of successful transmission of that datagram arrives from the TED, the ULB assumes the datagram has been sent and removes it from the list of the datagrams to be transmitted. Conversely, if a notification of failed transmission arrives from the TED or a timeout of 30 msec expires after the transmission of that datagram, the ULB assumes the datagram has been lost, retransmits it using the UMTS network interface and removes it from the list of datagrams to be transmitted.

During the communication, the server proxy must be aware of the NIC the client proxy is currently exploiting. At the server proxy, the ULB discovers the IP address of the current most suitable mobile node's NIC by inspecting the "source" field IP address of the most recent datagram received from the client proxy, i.e. the datagram with the highest "transmission sequential number". For this reason, at the client side, if no novel datagram is ready to be transmitted, locally generated by the game application, for a period of 500 msec, then the ULB generates a small keepalive datagram and inserts it in the lists of the datagram to be transmitted.

Finally, on both client and server proxies, if no NIC is available and correctly configured, the datagram to be sent is discarded.

3. PERFORMANCE EVALUATION

We performed an evaluation of a real prototype implementation of the presented approach. We considered a mobile user that moves through different coverage networks which are Wi-Fi and UMTS. The cross-layer architecture has been fully implemented; to test it, we simulated the network traffic generated during a multiplayer online game. It is current practice to simulate application data, since such an approach allows to model and vary the type of network traffic, depending on the specific game being played (e.g. first person shooters usually generate events at rates which are higher than, for instance, role playing games) [13, 14]. Hence, the game client application and the game server were simulated by exploiting a specific game traffic generator we built. In this work we concentrate on game events and game state updates produced at the client and server with an inter-departure time with a mean equal to 300 msec. This allows to mimic a player that interacts in a fast-paced game (3/4 moves per second, on average), and receives correspondent updates processed (and distributed) by the game server. Game events were 200 Bytes long, and included into UDP datagrams to mimic a typical game traffic scenario [4, 14].

As mentioned, the client and server proxies are fully implemented systems, able to interact with whatever game application. The prototype of the utilized client proxy has been specifically implemented using as mobile host a Linux laptop, by modifying the kernel version 2.6.30 and the kernel's mac80211 module. Monitor, ULB and TED are separate processes interacting with the operating system kernel and the game application. Detection, configuration and disconnections of Wi-fi network interface cards have been implemented by reusing (and modifying) the open-source wpa_supplicant. Instead, configuration of UMTS NICs has been implemented by modifying the wvdial application and reusing a PPP daemon.

The main objective of the performance study was to show the ability of the system to effectively react to the changes in the availability of the communication resources, and deliver (receive) the game data to (from) its game server within a game delivery threshold. Such threshold is game dependent and corresponds to the limit above which a real-time evolution of the game cannot be guaranteed, with corresponding jerky renderings at the client-side. The value for this game dependent threshold is typically set within the range [150, 300] msec [4, 14].

Real tests were performed within (and in the proximity of) a building (as depicted in Figure 5) where a Wi-Fi network was available; such wireless access point allowed to route the traffic to the Internet via an Italian ADSL Internet Service Provider (ISP). The utilized ISP exploits firewalls and NATs to manage the Internet connections of its customers. This did not represent as an obstacle for RWMPC, as demonstrated by the results. Concurrently, a UMTS cellular network was available in the area where tests were performed. The server proxy and the simulated game server were placed at the Department of Computer Science of the University of Bologna. The average round trip times from the mobile client to the game server were equal to 110 and 259 msec when using the Wi-Fi or UMTS network interface cards, respectively. It is important to point out that these averages have been measured along the path where the tests were made. The signal strength related to each network varied during the path. Of course, the performances of the Wi-Fi network was seriously affected by the distance between the mobile terminal and the access point (and the presence of obstacles between them). In fact, when the signal strength was low, we noticed several packets' retransmissions at the physical level, hence augmenting the final time to transmit each packet at higher levels.

During such tests, a laptop running the simulated game and the client proxy of the RWMPC covered a route of about 120 meters. During its path, the different network access points available to the user were exploited. In particular, the mobile user started its path from a point where both Wi-Fi and UMTS networks were available. Then, he left the building hence causing a disconnection with the Wi-Fi network, while maintaining active its UMTS connection. Finally, he re-entered in the Wi-Fi covered area. The motivation behind these tests was to assess: i) if RWMPC is able to detect which networks are available during the path; ii) if RWMPC is able to automatically decide which network should be exploited when both UMTS and Wi-Fi networks are available (and in case, exploit both networks); iii) the ability of RWMPC to switch from a network to another when some disconnection/reconnection is experienced; iv) measure the network performances obtained when multiple networks are utilized to transmit game events.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8656
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8656

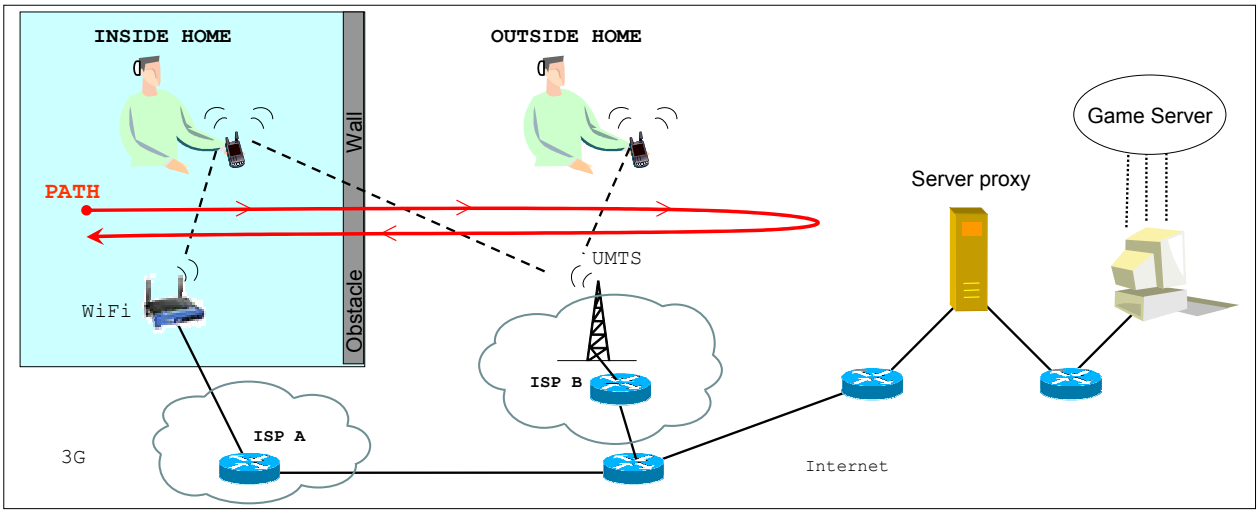

Figure 5. Experimental scenario.

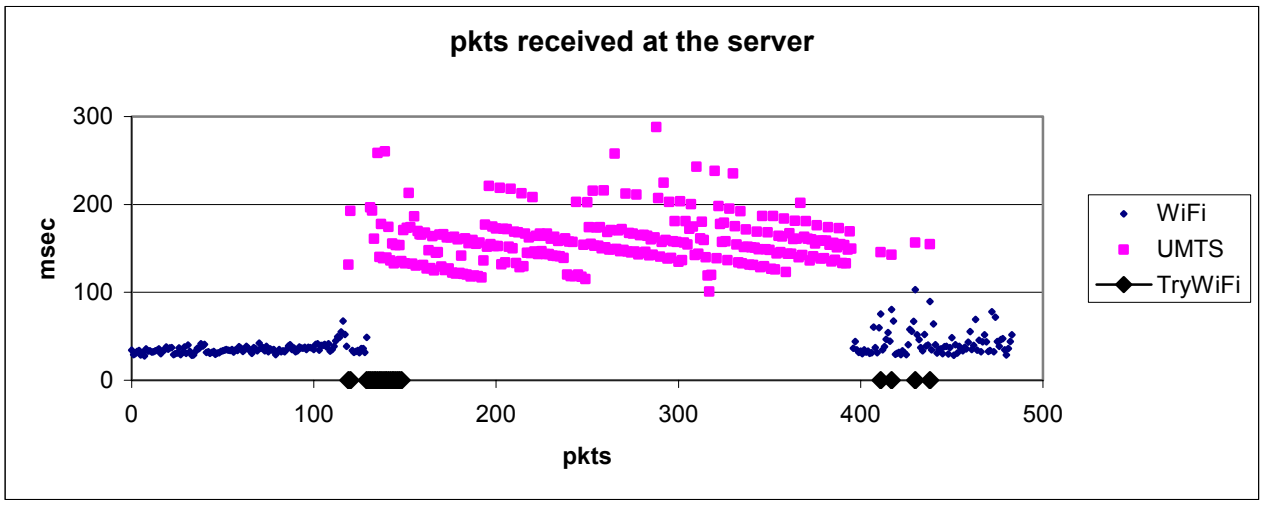

Figure 6. Latencies for game events received at the server-side.

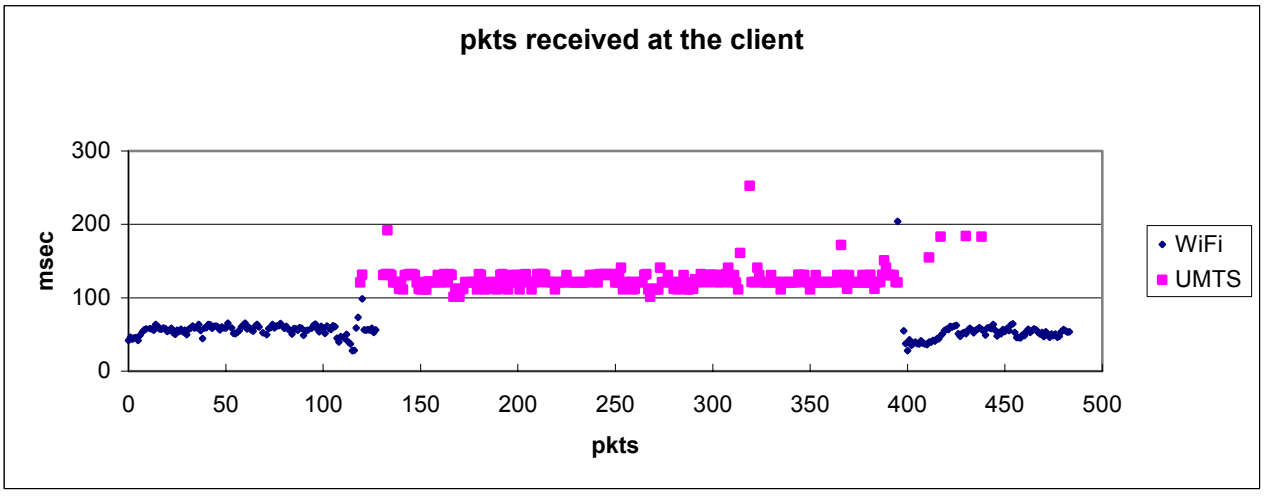

Figure 7. Latencies for game state updates received at the client-side.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8656
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8656

Figures 6 and 7 show the delivery times experienced to transmit game events from the client to the game server and vice versa, respectively. They refer to a single experiment. We repeated such experiment several times (15 tests), obtaining similar results. (Actually, qualitative similar outcomes were obtained in other real scenarios of assessment of the architecture, not reported in this paper.) Each chart reports, for each message, its delivery time and which network has been exploited. As expected, we may appreciate different behaviours, depending on the available networks. In fact, when both Wi-Fi and UMTS are available, the communication primarily flows through the Wi-Fi, which provides better communication performances. Conversely, when the Wi-Fi is no longer available the communication flows through the UMTS network. The approach allows to dynamically switch from a preferred network to another without causing interruptions in the communication. A consideration worth of mention is that, even if UDP has been exploited as the transport communication protocol, a negligible amount of game messages has been lost, lower than 1%. Neither gaps in the events' transmission have been experienced, due to network (re)configurations.

Such low percentage of lost messages is due to the use of our cross-layer architecture. In Figure 6, points marked as TryWifi (black points lying on the abscissa line) refer to those packets that have been firstly sent through the Wi-Fi network, when the connection with the Wi-Fi access point is being lost, but the client is still not aware that the Wi-Fi signal strength is deteriorating. In a classic scenario, where no proxies are available (and UDP is exploited at the transport layer), such packets would have been lost. Conversely, using RWMPC, the client proxy detects the problem and retransmits the packets through the UMTS network, which are finally received at the server-side. This is a demonstration that RWMPC prevents communication holes due to the time needed to detect and recover network connections, typical of a "single connection" based approach, e.g. when a single network is exploited, or when an ABC mechanism is employed (which usually performs an automatic network reconfiguration).

It is interesting to observe that after the second Wi-Fi reconnection, the mobile terminal transmits some game events using both networks. This is due to the fact that the ULB probes these nets, trying to understand which one provides better performances. This may be appreciated by looking at those game events which are delivered to the server proxy two times, through both networks (of course, experiencing different network delays). Similar outcomes are obtained also on the converse path, from the server (proxy) to the client (proxy).

As to end-to-end latencies experienced using RWMPC, these obviously depend on the specific network infrastructure being exploited to carry game data between the hosts involved in the communication. If we set the game dependent threshold for network latencies equal to 200 msec, by looking at Figure 7 we observe that experienced latencies are generally lower than that threshold. Only ~5% of the game messages have higher latencies when UMTS is active. However, worse performances would have been obtained by resorting to a single network. Indeed, using Wi-Fi the communication would have not been possible in the portion of the path without Wi-Fi signal coverage. Conversely, higher average network latencies would have been observed when resorting to UMTS.

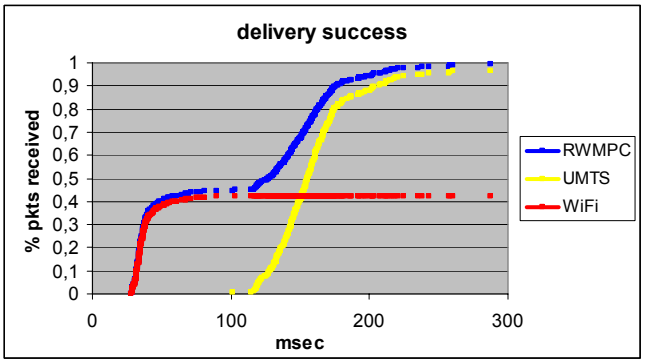

Figure 8. Percentage of delivery success.

This last consideration is also demonstrated by results shown in Figure 8, which reports the percentage of game events (with respect to the whole event trace) that experienced a delivery time lower that the value specified on the x-axis (i.e. a sort of cumulative probability of having a specific delivery time). The three curves relate to i) RWMPC; ii) results obtained for messages sent through Wi-Fi; iii) results obtained for messages sent through UMTS. It is possible to observe that Wi-Fi has a higher percentage of messages delivered in shorter times, with respect to the other approaches. However, such network does not provide a full reliability, since it was not possible to transmit the whole game event trace (i.e. in the reported experiment, only less than the 45% of game events). Conversely, UMTS was able to deliver the 100% of the generated event trace. However, latencies are really higher. RWMPC provides full reliability (the whole trace has been delivered), while guaranteeing low delivery times.

4. CONCLUSIONS

We presented a cross-layer architecture which allows mobile devices to concurrently exploit different network interfaces to offer a seamless communication among entities involved in multiplayer online games. The approach is based on the idea of monitoring all the network interfaces available at the mobile terminal; all these networks are concurrently exploited, while the game data is transmitted through these nets to obtain the best performances.

The use of client and server proxies allows to surmount all problems due to the presence of NAT and firewalls, typically exploited by Internet service providers. This has been confirmed by results obtained from a real testbed we have set. More specifically, outcomes from this performance assessment demonstrate that our approach enables to seamlessy exploiting all the active network interfaces, thus providing better performances and offering seamless and uninterrupted communications.

The prototype we have implemented facilitates a real deployment of multiplayer online games over mobile networks. Our architecture is transparent to the game developer, since we provide the pair of client and server proxies that may be exploited when developing the game. Finally, we are currently implementing a prototype on Symbian devices.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8656
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8656

5. REFERENCES

[1] Chen, K., Huang, P., and Lei, C. 2006. Game traffic analysis: an MMORPG perspective. *Comput. Netw.* 50, 16 (Nov. 2006), 3002-3023.

[2] Claypool, M. 2005. On the 802.11 turbulence of nintendo DS and sony PSP hand-held network games. In *Proceedings of 4th ACM SIGCOMM Workshop on Network and System Support For Games* (Hawthorne, NY, October 10 - 11, 2005). NetGames '05. ACM, New York, NY, 1-9.

[3] Kao, Y., Peng, P., Hsieh, S., and Yuan, S. 2007. A Client Framework for Massively Multiplayer Online Games on Mobile Devices. In *Proceedings of the 2007 international Conference on Convergence information Technology* (November 21 - 23, 2007). ICCIT. IEEE Computer Society, Washington, DC, 48-53.

[4] Färber, J. 2002. Network game traffic modelling. In *Proceedings of the 1st Workshop on Network and System Support For Games* (Braunschweig, Germany, April 16 - 17, 2002). NetGames '02. ACM, New York, NY, 53-57.

[5] Ferretti, S. 2008. A synchronization protocol for supporting peer-to-peer multiplayer online games in overlay networks. In *Proceedings of the Second international Conference on Distributed Event-Based Systems* (Rome, Italy, July 01 - 04, 2008). DEBS '08, vol. 332. ACM, New York, NY, 83-94.

[6] Ferretti, S. and Ghini, V. 2009. A Web 2.0, Location-Based Architecture for a Seamless Discovery of Points of Interests. In *Proceedings of the 2009 Fifth Advanced international Conference on Telecommunications* (Venice, Italy, May 24 - 28, 2009). AICT. IEEE Computer Society, Washington, DC, 226-231.

[7] Ferretti, S., Palazzi, C. E., Roccetti, M., Pau, G., and Gerla, M. 2006. FILA in gameland, a holistic approach to a problem of many dimensions. *Comput. Entertain.* 4, 4 (Oct. 2006), 8.

[8] Ferretti, S. and Roccetti, M. 2005. Fast delivery of game events with an optimistic synchronization mechanism in massive multiplayer online games. In *Proceedings of the 2005 ACM SIGCHI international Conference on Advances in Computer Entertainment Technology* (Valencia, Spain, June 15 - 17, 2005). ACE '05, vol. 265. ACM, New York, NY, 405-412.

[9] Ghini, V., Lodi, G., & Panzieri, F. (2009). Always Best Packet Switching: the Mobile VoIP Case Study. *Journal Of Communications,* Academy Publisher, *4*(9).

[10] Griwodz, C. and Halvorsen, P. 2006. The fun of using TCP for an MMORPG. In *Proceedings of the 2006 international Workshop on Network and Operating Systems Support For Digital Audio and Video* (Newport, Rhode Island, November 22 - 23, 2006). NOSSDAV '06. ACM, New York, NY, 1-7.

[11] Gustafsson, E. and Jonsson, A. (2003) 'Always best connected', *IEEE Wireless Communications Magazine*, Vol. 10, No. 1, pp. 49-55.

[12] Mauve, M., Fischer, S., and Widmer, J. 2002. A generic proxy system for networked computer games. In *Proceedings of the 1st Workshop on Network and System Support For Games* (Braunschweig, Germany, April 16 - 17, 2002). NetGames '02. ACM, New York, NY, 25-28.

[13] Palant, W., Griwodz, C., and Halvorsen, P. 2006. Evaluating dead reckoning variations with a multi-player game simulator. In *Proceedings of the 2006 international Workshop on Network and Operating Systems Support For Digital Audio and Video* (Newport, Rhode Island, November 22 - 23, 2006). NOSSDAV '06. ACM, New York, NY, 1-6.bo

[14] Palazzi, C. E., Ferretti, S., Cacciaguerra, S., and Roccetti, M., 2006. Interactivity-loss avoidance in event delivery synchronization for mirrored game architectures. *IEEE Transactions on Multimedia*, 8(4):874--879, 2006.

[15] Palazzi, C. E., Roccetti, M., Ferretti, S., and Frizzoli, S. 2008. How to let gamers play in infrastructure-based vehicular networks. In *Proceedings of the 2008 international Conference on Advances in Computer Entertainment Technology* (Yokohama, Japan, December 03 - 05, 2008). ACE '08, vol. 352. ACM, New York, NY, 95-98.

[16] Roccetti, M., Ferretti, S., and Palazzi, C. E. 2008. The Brave New World of Multiplayer Online Games: Synchronization Issues with Smart Solutions. In *Proceedings of the 2008 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing* (May 05 - 07, 2008). ISORC. IEEE Computer Society, Washington, DC, 587-592

[17] Zander, S. and Armitage, G. 2005. A traffic model for the Xbox game Halo 2. In *Proceedings of the international Workshop on Network and Operating Systems Support For Digital Audio and Video* (Stevenson, Washington, USA, June 13 - 14, 2005). NOSSDAV '05. ACM, New York, NY, 13-18.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8656
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8656