# Multipong: a Multiplayer Ad-Hoc Version of Pong

Marco Begolo, Sebastiano Valle, Marco Zanella,
Armir Bujari, Ombretta Gaggi, Claudio E. Palazzi
University of Padua
Department of Mathematics
Via Trieste 63, 35123 Padua, Italy
{marco.begolo, sabastiano.valle, marco.zanella}@studenti.unipd.it
{abujari, gaggi, cpalazzi}@math.unipd.it

***Abstract*—Mobile games have come to revolutionize the mobile handset and gaming industry, pushing chip vendors to compete in order to provide better and better graphics capabilities by means of dedicated processors. This capability coupled with the sensing and communication ability offered by smartphones, provides the developers with the building blocks for innovative gaming solutions. In particular, multiplayer mobile games could exploit context and proximity information, providing an added value to the gaming experience. In this context, we present the design and analysis of a modern mobile and multiplayer version of the classic Pong game. In addition to the classic interaction model through remote server(s), players could interact and play locally by exploiting ad-hoc connectivity offered by the Wi-Fi Direct technology.**

***Index Terms*—Games, Multiplayer, Wi-Fi Direct, Proximity.**

## I. Introduction

Mobile games have seen a tremendous increase in adoption and they are expected to gain further ground with respect to traditional ones. Recent statistics show that the global game market reached $99.6 billion and, for the first time, mobile gaming took a larger share than the desktop counterpart with $36.9 billion (up to 21.3% globally), reaching the 37% of the game market [1]. Key factors contributing to this success are surely the possibility to play anytime and anywhere, the ease of use and the social-dimension they embody. Yet, the widespread availability of mobile handheld devices and their computing, sensing and communication capabilities have created the means for building innovative gaming experiences and at the same time have lowered the barrier of entry of individual developers into the gaming market [2], [3].

On the other hand, mobile gaming also represents an interesting challenge due to the resource-constrained nature of their targeted environment [5], [6], [7]. While computational capabilities of mobile devices are continuously increasing, the battery is still lacking behind. Moreover, mobile data transfer is cost-attributed, thus demanding for alternate networking techniques making a parsimonious use of data exchange. Also, the input mechanism mainly relies on the touch sensor (soft keypad) which, if not properly considered, might hinder the gaming experience [8], [9].

It is very interesting to note that, looking at the Google's Play Store, most of the top grossing games are multiplayer online games, and many of them require real-time users interaction. Furthermore, these games are limited to a client-server approach demanding Internet access.

In this paper, we set on a trial to investigate the feasibility of multiplayer gaming exploiting ad-hoc communication. Pursuing this goal, we designed and implemented *Multipong*, a multiplayer version of the classic Pong game whereby users could match against each other without the need of Internet access. The game allows matches amongst two or more players. To achieve this goal, we rely on the Wi-Fi Direct technology capable of connecting user devices without the need of an access point. While quality of experience (*QoE*) is a broad term that is comprehensive of many factors [10], we focus our aim on some technical aspects that affect the gameplay quality. In particular, we focus on network-layer metrics, leaving a user-experience study as a future work.

## II. Background

The Wi-Fi Direct standard also known as Wi-Fi P2P enables devices to connect with each other without requiring the presence a physical wireless access point [11]. Wi-Fi P2P implements a software access point module, capable of host configuration and management. In Wi-Fi Direct terminology a network unit is referred to as a *group* and each group has a *group owner (GO)* whose role is analogous to the one of an access point in infrastructure-mode.

Usually, a device supporting Wi-Fi Direct, in order to create or join a group, starts a discovery session in which it may find other unconnected Wi-Fi Direct devices or GOs. A device can autonomously decide to start the formation of a group, or may ask to join one. During group formation, devices need to negotiate their roles in order to find a peer that assumes the role of a *logical* access point. While the GO negotiation protocol is specified by the standard, applications can implement their own logic of electing a suitable one. Legacy devices on the other side, those that do not support Wi-Fi Direct, may later on decide to connect to the GO and join the group.

More in detail, the standard outlines three different group formation techniques, namely standard, persistent and autonomous. The *standard* technique is the most generic group formation technique while the others shortcut some of the phases involved. The procedure starts with nodes first becoming aware of each other either by passive or active scanning of Wi-Fi channels. Once this phase is completed, the GO

negotiation phase takes place, where each device states its own `GroupOwnerIntent`, consisting of a value ranging from 0 (not willing to become the GO) to 15 (highest inclination to become a Group Owner). Successively to the GO negotiation phase, the security and address configuration phases take place in sequence and, if successful, the group is considered as established and nodes can communicate without any infrastructure mediation.

Support for Wi-Fi Direct in Android devices has been rolled out since Android 4.0, enabling P2P connectivity amongst Wi-Fi Direct capable devices. In these settings, a GO is connected to multiple clients in a P2P fashion (hereafter *NGO*s). As discussed, the GO is decided after a negotiation phase between the devices; thus, the same hosts may create an ad-hoc network with different GOs from time to time.

However, the implementation of Wi-Fi Direct in Android presents some issues and limitations: first of all Android does not have native support for multi-group formation and devices must ask the user for the permission to join a group, hindering the automatic creation of Wi-Fi Direct networks [12].

## III. Related Work

According to [13], main requirements for a gaming session are: (i) *good interactivity*, i.e. the delay between the user interaction and the game response should be as short as possible, (ii) *consistency*, i.e. different players should see coherent and admissible game states, (iii) *fairness*, i.e. it should be possible to win a match regardless of different network conditions, (iv) *scalability*, i.e. being able to support a large number of players, and (v) *continuity*, i.e. the present game session should not be interrupted because of disconnections, handoffs, or any other mobility-related issue. Fulfilling these requirements, a lot of research effort has been devoted, spanning from architectural solutions to efficient network-layer proposals [14], [15].

Mobile gaming further exacerbates the issues, also presenting its own challenges in the context of real-time applications: e.g., multiplayer gaming requires Internet connectivity which in the mobile world might be cost-attributed or at least not available anytime, anywhere [16]. As a remedy, one might resort to local gaming sessions whereby a coordinator node hosts the session becoming a potential bottleneck [17]. Pure P2P or hybrid solutions on the side represent an attractive alternative but usually lack of protection from cheaters [18], [19].

MultiPong belongs to the category of *casual games*, i.e, video games which present a simple gameplay and targets a mass audiences [20]. Casual games are designed to be played by users with no special skills and without requiring too much time for both understanding and playing them [21]. A well known example of casual game is the Candy Crash Saga.

Despite being very common among mobile games, casual games were originally played by users through a web browser and a large number of users still play using the web platform, e.g., through social networks: the idea of casual gaming has been indeed mashed up with this recent phenomenon, allowing casual gamers to play with their friends through different platforms and network architectures [22], [23], [24].

Casual games experiments were also undertaken in [25], [26] but, as these studies reported, this type of games has not break through either the academic or the commercial world yet.

## IV. Multipong

In this section we discuss some salient features of the Multipong application. For more details regarding the implementation and related design choices we refer the reader to the public repository available from [27].

### A. Game Description

Multipong is a tribute to the *Pong* game, one of the first arcade videogames, where the players need to prevent a ball from falling out of the screen by using a paddle. The application supports both a *singleplayer* and a *multiplayer* mode.

In the *singleplayer* mode the player scores a point each time the paddle hits the ball, making it bounce upwards until it reaches the top edge and then the ball falls down again. Clearly, the player loses the game when the paddle misses the ball. Otherwise, if the paddle hits the ball, the bounce speed is inversely proportional to the distance $d$ from the center of the paddle for its vertical component, whereas it is proportional to $d$ for its horizontal component. In order to make the game play more enjoyable, we also introduce a random component in the ball bounce speed by adding a random amplification.

In the *multiplayer* mode, several players connect their devices forming an ad-hoc network. The device who created the gaming session has to make the first move and when this happens, the ball is transferred to the next player's screen as if their gameboards were joint. When a player misses the ball, the player loses a life and the ball is thrown out randomly to the next player's screen. If a player runs out of lives, the player will not be able to play for the rest of the game and the game takes place between the remaining players. The last standing player is the winner of the gaming session.

Through the rest of the paper we will focus on the *multiplayer* mode, namely network formation and the gameplay, since the *singleplayer* mode does not rise any important issue.

### B. Overall architecture

Multipong architecture (Figure 1) is mainly comprised of two layers: (i) the*networking-layer* which handles group formation and communication among peers and (ii) the *game-layer* which handles the game application logic. This loose coupling between layers was intentional, allowing for the reuse of the network-layer for future potential scenarios e.g., in the context of geo-localized partecipatory sensing and collaborative video annotation [28], [29], [4]. In specific, the *network-layer* is able to figure out whether the device is acting as the GO or not, retrieve the IPs of the other peers (through the `Discovery` component), bind those addresses to logical, application-level identifiers (through `NameResolution`

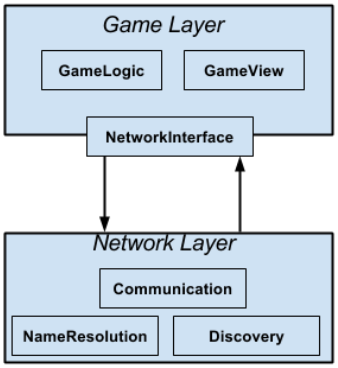


Fig. 1. Multipong architecture

component) and the `Communication` component which manages data exchanging amongst devices.

The game layer is accountable for the application logic (`GameLogic` component) and for displaying that info to the user as well as catching players interactions (through the `GameView` component). Figure 1 depicts the overall architecture of the game: the two layers talk to each other through the `NetworkingInterface`, which is a component that provides a mid-level abstraction.

### C. Game Initialization

Hereafter we assume that the players correctly configured a Wi-Fi Direct ad-hoc network with their Android devices, i.e. that they can communicate with each other within this network.

Since in the initialization (game formation) phase there are no strict requirements to meet in terms of real-time information delivery, we decided to use TCP as a transport protocol. Also, this choice will provide us an element of comparison to the solution we adopted for the gameplay. Raw data sent over sockets are formatted as JSON objects, giving us the ability to distinguish more easily the requests from one peer to another.

Any device in the Wi-Fi Direct group is capable of and allowed to host a gaming session, so even if the game formation has one host and several participants, we can not *a priori* assume that the host will coincide with the GO. Moreover, during the association phase the NGOs will obtain an address from the GO, whereas the GO will only be notified about the *presence* of peers, without obtaining any concrete reference to communicate with them. This means that the networking logic present in our application has to take into account the fact that the GO needs to retrieve somehow the IPs addresses attributed to the NGOs.

Before describing the protocol we employed in the game initialization phase, we want to briefly describe how data exchange between peers is implemented: each peer has two threads, respectively for receiving and sending data. The communication is performed in an asynchronous fashion, so that messages can be exchanged without blocking device while waiting the response. However, we implemented also

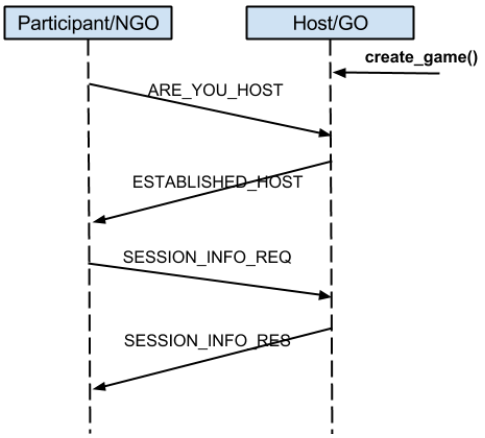


Fig. 2. Communication between a GO-host and an NGO-participant

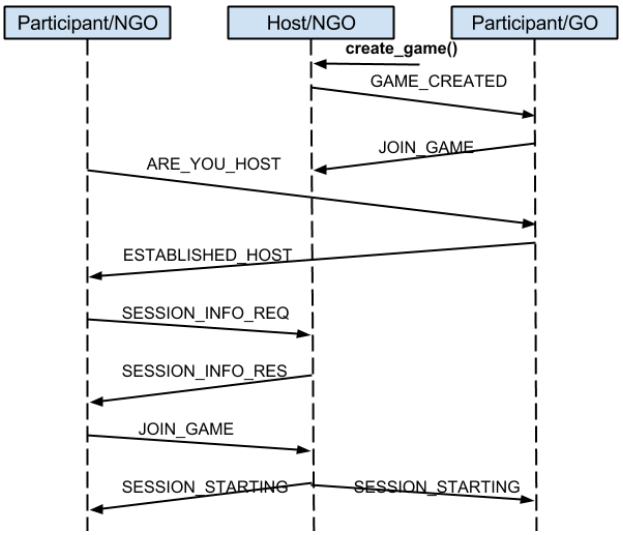


Fig. 3. Communication between an NGO-host, a GO-participant and an NGO-participant

the semantics for synchronizing on certain operations and we guarantee FIFO ordering for the data sent out of a device.

In order to create a match among several players, applications on different devices follow a common protocol. First, at Wi-Fi P2P discovery time, new NGOs need to ask the GO whether it is hosting a game or not by means of a `ARE_YOU_HOST` message. If so, the host (also the GO) replies with a `GAME_CREATED` message (refer to Figure 2), otherwise this peer (which is therefore a participant) replies with a `ESTABLISHED_HOST` message, in which it announces the host (refer to Figure 3).

At this point, the participant(s) surely knows the host IP address (if any) and it is able to send a `SESSION_INFO_REQ` message to the host. This message causes the host to reply with an `SESSION_INFO_RES` message in which currently joined players and game attributes are listed. After this step, the participant can join the game play by sending a `JOIN_GAME` message to the host. Finally, a player can cancel a subscription by sending a `CANCEL_PARTICIPATION` message and re-

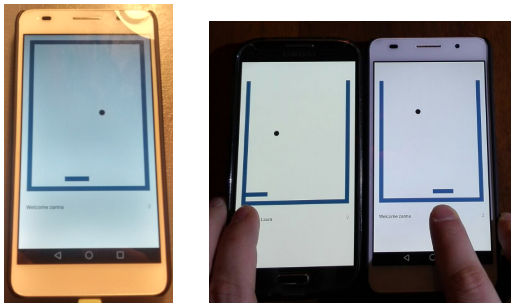

Fig. 4. Singleplayer and multiplayer match

ceiving back an `SESSION_INFO_RES` serving a cancellation confirmation.

If the conditions for starting the gaming session have been satisfied, the host announces its creation by sending a `SESSION_STARTING` message to all of the participants. The trigger used to decide whether the session can start is specified by the host player through a dedicated configuration menu and the options are as follows: (i) maximum number of players reached (ii) time required for joining the game expired. This last interaction among the host and participant devices is synchronous, the host is blocked until all the other peers sent a confirmation reply. If some player, for any reason, does not reply to the message, the player is excluded by the match as described in the following subsection.

### D. Gaming Session

In Figure IV-D) we show examples of gameplay. Differently from the game formation phase, during the gameplay we employ the UDP protocol given that messages have to be delivered quickly. Increasing reliability, we decided to employ application-level acked UDP transmission: after sending an UDP packet, a peer waits for a short ACK packet to come back within a short time frame; if it does not, the peer retries the transmission up to four times in total (a configuration parameter).

During the gameplay we want to ensure both consistency and low latency of gaming events. For this reason, the two layers of the architecture have different roles: one layer is concerned with coordinating the peers and the other one deals with the multiplayer game logic on top of the coordination layer.

The multiplayer game logic layer has to manage the local state of the game and part of the global state. Also, when the player hits the ball with the paddle, it also has to compute the ball exit point from the screen and send this information to the GO as soon as it is available, so that it can spread this message to all the active participants of the game. We chose to compute and send the ball exit position in advance in order to reduce the network latency perceived by the player.

We stated that a player sends the game event massage to the GO: in fact, this entity serves as a global coordinator of the game. We made this decision so that a reduced amount of traffic has to flow through the network if only the GO has to care about whether the currently active player is still alive and reachable. Moreover, the GO is the main entity in the group and therefore is the natural choice for such role of acting coordinator.

Clearly, the GO represents a single point of failure during the game phase, and a network failure such a crash of the coordinator would make the whole ad-hoc network collapse. Moreover, recreating the topology is an operation that, at the moment, is quite difficult to carry out in a short amount of time (and unfortunately things go even worse when done programmatically). Therefore, we thought that the game of an NGO should end if it is unable to reach the GO, because restoring a coherent game state amongst all involved players can require even more time than starting a new one from scratch.

While GO failures are fatal for the game, the game is able to cope with NGOs failures: in fact, if an NGO appears to be non responsive the GO, it is removed from the game after the GO asked a few times if it is alive or not without getting any response. In that case, the game continues with the GO telling the other players that a certain peer is not active anymore.

A game always starts from the host device and the queue of players is decided before the beginning of the match. Even if an NGO failure happens, the order of the players' queue does not change but the player that is experiencing networking problems is simply removed from the queue.

## V. Experimental evaluation

This section discusses the experimental testbed and the evaluation strategy carried out. The testbed we employed is comprised of: 5 Samsung S5, 2 Samsung S3, 1 Motorola Moto G 2013, 1 Motorola Moto X+1 2014, 1 Huawei P8 Lite.

### A. Network Traffic Measurements

As stated previously we employed TCP for message exchange during the game initialization phase. Hence, an interesting metric lies in application packet's RTT and payload size.

We computed the RTT by measuring, each time a peer sends a message, the delta elapsed between the start of sending and an application-level confirmation of delivery. Even if this measurement is done at the application level, we believe that this approach yields a realistic approximation of the RTT. However, this method is affected by the variability of the network and interferences, so we decided to take several measures and then average them. Also, there is an overhead due to the time spent by the operating system to manage the communication among devices.

For the TCP payload evaluation, we generated some logs in the application to get the exact size of each type of message. Messages always carry the same data structure and length, so in this case it is not necessary to calculate a mean value.

We adopted a similar approach for the UDP-based exchange.

### B. Adopted method

The application was designed to minimize message exchange among peers, so we were interested in measuring how much traffic our application generates. To determine that, we decided to sum up the sizes of the packets peers exchanged among themselves. Though this approach works at the application layer, it is able to provide interesting insights into the amount of data exchanged by peers. While considering the Wi-Fi frame size on-air would have yielded a more accurate measurement, we were unable to perform this measurement due to constraints posed by the Android operating system. Indeed, we tried several solutions but they were not suitable for our purposes for the following reasons:

- 3GWatchdog and similar applications do not provide accurate measures compared to our solutions;
- tPacketCapture is an application that sets up a VPN to monitor traffic; after setting up the VPN, we were not able to make the devices communicate to each other even during the match initialization phase;
- tcpdump requires rooting the smartphone in order to use it, voiding the warranty as a consequence;
- despite we were able to form a Wi-Fi Direct network from a computer and connect the smartphones to it, we did not manage to sniff the communication by using Wireshark since we were able to capture *only* the packets exchanged between the PC and the GO;
- we did not have dedicated hardware (i.e. a wireless network controller such as Alfa One) to sniff traffic directly by the phone.

Multipong generates different amounts of traffic during the initialization and the gameplay, hence we decided to measure these two phases separately. To get better insights on the performance of our solution, we performed measurements with an increasing number of multiplayer sessions: 2-, 3- and 4-players matches. The comparison among these groups reveals the influence of the pairing phase to the traffic. Avoiding the bias of the human player, we implemented an AI that plays a match, losing exactly after ten turns.

## VI. Results

In this section we discuss the experimentation outcome regarding network traffic analysis. To this end, we quantify the amount of traffic exchanged among peers involved in the gaming session with varying number of participants. We then proceed by comparing the various phases, game formation and game play, interactivity by measuring the average Round Trip Time (RTT) of packets.

### A. Traffic Analysis

In Table I we report the messages peers exchange during Multipong's gameplay along with their size. The message flow exchanged among the devices follows the one detailed in Figure 3. All of the above quantities refer to the application-layer data packet and the actual (on air) frame size differs from those reported in Table I. While we consider only application-layer quantities in our experiment, this does not impact the trend of the experiment outcome.

TABLE I
Messages size

| | *Message* | *Size [B]* |
|---|---|---|
| TCP | ARE_YOU_HOST | 60 |
| | SESSION_INFO_REQ | 82 |
| | SESSION_INFO_RES | 100 |
| | CANCEL_PARTICIPATION | 56 |
| | JOIN_GAME | 54 |
| | ESTABLISHED_HOST | 90 |
| | SESSION_STARTING | 187 |
| | GAME_CREATED | 57 |
| UDP | TURN_INFO | 143 |
| | ARE_YOU_ALIVE | 66 |
| | ACK | 3 |

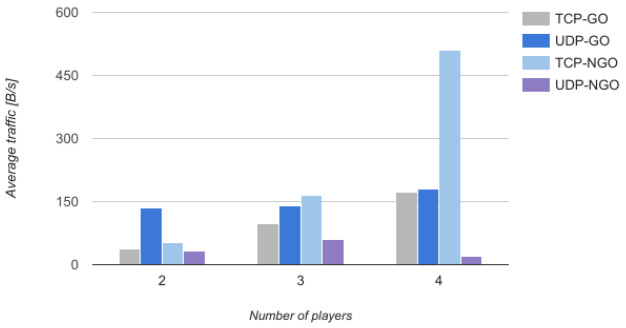


Fig. 5. GO/NGO comparison for traffic

Concerning the game formation phase, the extent to which traffic grows with the number of players is striking (see Figure 5). On the other hand, during the gameplay phase, the amount of bytes per second is lower since there are significant dead times (i.e. times where the ball is not in the player's screen) in which communication is almost null. It's interesting to notice how much the overhead the GO incurs is limited as the number of players rises (see Figure 5); this fact leads us to suppose that we were not able to play matches with a high number of players due to the precariousness of Wi-Fi direct, rather than the overhead caused by the communication among the peers (which is indeed acceptable for a real-time application).

### B. TCP and UDP comparison

In Figure 6, we can see the difference between the mean RTTs of TCP and acked UDP transmissions, which were measured as described in section V-A.

As it can be seen, acked UDP is very efficient compared to TCP and therefore we can acknowledge that acked UDP is the most suitable choice (between the two) in the gameplay phase, whilst TCP is fit for the game formation phase, when we do not need low latency but more reliable packet delivery.

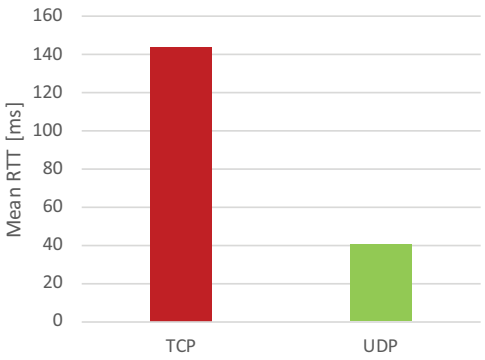


Fig. 6. TCP vs. UDP

## VII. Conclusion

Thanks to the wide-spread availability of smartphones and their sensing and communication capabilities, the toolset available to game developers provides a wide range of opportunities. In this paper, we presented *Multipong*, a multiplayer version of the old arcade game Pong, whereby players could match against one another when in proximity. Multipong exploits the Wi-Fi Direct technology making the game play available to users anytime, anywhere.

While standardized and available in the smartphone market for sometime, Wi-Fi Direct is really cumbersome: in fact, group formation setup is not user transparent and very often it takes too much time. Yet, another problem is managing the unreachability of the GO in a Wi-Fi Direct network: at the current state-of-the-art, the best solution is the one we undertook, i.e. interrupting the current match, because of the impossibility to recreate the network in a short amount of time and automatically.

As a future work we plan to address the automatic group re-formation process, that is when the GO leaves the network for some reason. Complementing the current assessment of the solution, we plan to perform more accurate measurements of network traffic and battery expenditure following an approach similar to [30].

## Acknowledgment

This work has been partially funded by the University of Padua, through the projects PRAT CPDA137314 and PRAT CPDA151221.

## References

[1] Newzoo Games, The Global Games Market Share. April 2016.
[2] M. Furini, "Mobile Games: What to Expect in the Near Future", in Proc. of *GAMEON*, 2007.
[3] C. Prandi, V. Nisi, P. Salomoni and N. J. Nunes, "From Gamification to Pervasive Game in Mapping Urban Accessibility", in Proc. of *ACM Italian SIGCHI Chapter*, 2015, pp. 126-129.
[4] C. Prandi, S. Ferretti, S. Mirri and P. Salomoni, "Trustworthiness in Crowd-sensed and Sourced Georeferenced Data", in Proc. of *IEEE PerCom Workshops*, 2015, pp. 402-407.
[5] M. Ciman, O. Gaggi and N. Gonzo, "Cross-platform mobile development: A Study on Apps with Animations", in Proc. of *ACM SAC*, 2014.
[6] M. Dick, O. Wellnitz, and L. Wolf, "Analysis of Factors affecting Players' Performance and Perception in Multiplayer Games", in Proc. of *ACM SIGCOMM Workshop on Network and System Support for Games*, 2005, pp. 1-7.
[7] S. Möller, S. Schmidt, and J. Beyer, "Gaming Taxonomy: an Overview of Concepts And Evaluation Methods for Computer Gaming QoE", in Proc. of *IEEE QoMEX Workshop*, 2013, pp. 236-241.
[8] K. Chu and C. Y. Wong, "Mobile Input Devices for Gaming Experience", in Proc. of *IEEE i-USEr*, Nov. 2011, pp. 83-88.
[9] S. Cacciaguerra, S. Mirri, P. Salomoni and M. Pracucci, "Wandering About the City, Multi-Playing a Game", in Proc. of *IEEE CCNC*, 2006, pp. 1214-1218.
[10] A. Kaiser, D. Maggiorini, N. Achir and K. Boussetta, "On the Objective Evaluation of Real-Time Networked Games", in Proc. of *IEEE GLOBECOM*, 2009, pp. 1-5.
[11] Wi-Fi Alliance®, *Wi-Fi Peer-to-Peer (P2P) Technical Specification v1.7*. July 2016.
[12] C. Casetti, C. F. Chiasserini, L. C. Pelle, C. D. Valle, Y. Duan and P. Giaccone, "Content-centric Routing in Wi-Fi Direct Multi-group Networks", in Proc. of *IEEE WoWMoM*, 2015, pp. 1-9.
[13] C. E. Palazzi, "Interactive Mobile Gaming over Heterogeneous Networks", in Proc. of *IEEE/ITI International Conference on Information and Communications Technology*, Cairo, Egypt, Dec. 2007.
[14] L. Pantel and L. C. Wolf, "On the Impact of Delay on Real-time Multiplayer Games", in Proc. of *ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video*, May 2002, pp. 23-29.
[15] A. Bujari, M. Massaro and C. E. Palazzi, "Vegas over Access Point: Making Room for Thin Client Game Systems in a Wireless Home", *IEEE Transactions on Circuits and Systems for Video Technology*, 25(12), 2015.
[16] C. E. Palazzi and A. Bujari and G. Marfia and M. Roccetti, "An Overview of Opportunistic Ad hoc Communication in Urban Scenarios", in Proc. of *IFIP MedHocNet*, 2014.
[17] C. E. Palazzi, S. Ferretti, S. Cacciaguerra and M. Roccetti, "On Maintaining Interactivity in Event Delivery Synchronization for Mirrored Game Architectures", in Proc. of *GLOBECOM Workshops*, 2004, pp. 157-165.
[18] T. Fritsch, H. Ritter and J. Schiller, "CAN Mobile Gaming be Improved?", in Proc. of *ACM SIGCOMM Workshop on Network and System Support for Games*, 2006.
[19] T. Fritsch, H. Ritter and J. Schiller, "User Case Study and Network Evolution in the Mobile Phone Sector (A Study on Current Mobile Phone Applications)", in Proc. of *ACM SIGCHI ACE*, 2006.
[20] M. Furini, "An Architecture to Easily Produce Adventure and Movie Games for the Mobile Scenario", *ACM Computers in Entertainment*, 6(2), 2008.
[21] A. Grimes, V. Kantroo and R. E. Grinter, "Let's play!: Mobile Health Games for Adults", in Proc. of *ACM UbiComp*, Sep. 2010, pp. 241-250.
[22] D. Maggiorini, C. Quadri, L. A. Ripamonti, "On the Feasibility of Opportunistic Collaborative Mixed Reality Games in a Real Urban Scenario", in Proc. of *ICCCN*, 2012.
[23] D. Maggiorini, A. Nigro, L. A. Ripamonti, M. Trubian, "Loot Distribution in Massive Online Games: Foreseeing impacts on the players base", in Proc. of *ICCCN*, 2012.
[24] M. Gerla, D. Maggiorini, C. E. Palazzi and A. Bujari, "A Survey on Interactive Games Over Mobile Networks", *Wireless Communications and Mobile Computing*, 13(3), 2013.
[25] J. Paavilainen, K. Annakaisa, K. Jussi, M. Frans, S. Hannamari and N. Johannes, GameSpace: Methods and Evaluation for Casual Mobile Multiplayer Games. Available at: https://tampub.uta.fi/handle/10024/65773, 2009.
[26] K. Li, and S. Counts, "Exploring Social Interactions and Attributes of Casual Multiplayer Mobile Gaming", in Proc. of *ACM Mobility*, 2007, pp. 696-703.
[27] *Multipong Source Code*. https://github.com/snate/multipong
[28] S. Ferretti and S. Mirri and M. Roccetti and P. Salomoni, "Notes for a Collaboration: On the Design of a Wiki-type Educational Video Lecture Annotation System", in Proc. of *IEEE ICSC*, 2007.
[29] C. E. Palazzi, and L. Teodori and M. Roccetti, "Path 2.0: A Participatory System for the Generation of Accessible Routes", in Proc. of *IEEE ICME*, 2010.
[30] M. Ciman and O. Gaggi, "Evaluating the Impact of Cross-platform Frameworks in Energy Consumption of Mobile Applications", in Proc. of *WEBIST*, 2014, pp. 423431