5-2015

GameOn: P2p gaming on public transport

Nairan ZHANG
University of Wisconsin-Madison

Youngki LEE
Singapore Management University, YOUNGKILEE@smu.edu.sg

Meera RADHAKRISHNAN
University of Wisconsin-Madison

Rajesh Krishna BALAN
Singapore Management University, rajesh@smu.edu.sg

DOI: https://doi.org/10.1145/2742647.2742660

GameOn: p2p Gaming On Public Transport

Nairan Zhang[‡*], Youngki Lee[†], Meera Radhakrishnan[†], and Rajesh Krishna Balan[†]
[‡]Department of Electrical and Computer Engineering, University of Wisconsin-Madison,
[†]School of Information Systems, Singapore Management University

ABSTRACT

Mobile games, and especially multiplayer games are a very popular daily distraction for many users. We hypothesise that commuters travelling on public buses or trains would enjoy being able to play multiplayer games with their fellow commuters to alleviate the commute burden and boredom. We present quantitative data to show that the typical one-way commute time is fairly long (at least 25 minutes on average) as well as survey results indicating that commuters are willing to play multiplayer games with other random commuters. In this paper, we present *GameOn*, a system that allows commuters to participate in multiplayer games with each other using p2p networking techniques that reduces the need to use high latency and possibly expensive cellular data connections. We show how *GameOn* uses a cloud-based matchmaking server to eliminate the overheads of discovery as well as show why *GameOn* uses Wi-Fi Direct over Bluetooth as the p2p networking medium. We describe the various system components of *GameOn* and their implementation. Finally, we present numerous results collected by using *GameOn*, with three real games, on many different public trains and buses with up to four human players in each game play.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication; C.3 [**Special-Purpose and Application-based Systems**]: Real-time and embedded systems; C.5.3 [**Microcomputers**]: Portable devices

General Terms

Design, Experimentation, Performance, Measurement

Keywords

p2p Games; Mobile Gaming; Public Transportation

[*]This work was done when he was a Ph.D. intern at Singapore Management University.

MobiSys'15, May 18–22, 2015, Florence, Italy.

ACM 978-1-4503-3494-5/15/05 ...$15.00.
http://dx.doi.org/10.1145/2742647.2742660.

1. INTRODUCTION

Games remain the most popular application category on both the iOS and Android ecosystems [14] in terms of downloads, usage, and revenue earned. In particular, multiplayer games are becoming increasingly popular to both game players and developers. Players find that the unpredictability that arises from playing against human opponents keeps them engaged for much longer periods while game developers find that more engaged users generate a lot more ads and in-game sales revenues than players who play for a few minutes and then leave. Indeed some of the most lucrative and popular games on both the mobile [11] and console markets [37] tend to be multiplayer-focused games such as Brave Frontier, Clash of Clans, and Pirate Kings on mobile devices and FIFA 15, Destiny, and Call of Duty: Advanced Warfare on consoles. In these games, players usually play against random strangers either individually or in groups where each group is made up of friends or even random strangers.

Multiplayer games encourage players to spend tens of minutes or longer for every game session (especially for games where the other human players are also interacting in real time) unlike casual single player games such as Candy Crush or Angry Birds that are optimised for one to five minutes of short game sessions. As such, these types of longer multiplayer games tend to be played during lunch breaks or after work/school is over.

However, there is also another opportunity to play longer multiplayer games. In many dense crowded urban cities (which are common in Asia and Europe, and include some US cities like New York City and San Francisco), the cost of driving tends to be quite high in terms of traffic, time taken, aggravation, and parking availability, etc. As such, a significant fraction of the population in these cities take public transport for their daily commute. These commutes also tend to be long – for example, as we show in Section 3, the average one-way commute time in Singapore is about 26 minutes with larger times reported for other urban cities (e.g., 40 minutes for New York City, 66 minutes for Tokyo, and 97 minutes for Beijing).

This commute period is a natural "down time" where the commuter can be engaged. Currently, many commuters spend the time by sleeping, reading something, or using their phones to check email, browse the web, chat with friends, watch videos, listen to music, or play games. We also observed that an increasingly large fraction has access to smartphones (87% smartphone penetration in Singapore and Hong Kong and rapidly rising in other Asian cities [28]) that have the performance and networking capabilities required for mobile game playing. We thus hypothesise that these commuters could benefit from playing spontaneous multiplayer games, to ease the commute boredom, if the functionality was available.

Thus, we present *GameOn*, a system for allowing public transport commuters to engage in multiplayer games with fellow commuters travelling on the same bus or train. The key technical challenges we overcame were:

High 3G/LTE latencies. This causes serious lag and playability issues in multiplayer games (especially in the near real-time games like shooting or racing games). We overcame this by using p2p networking solutions for the actual game plays.

Identifying the appropriate p2p networking mechanisms. As we show in Section 4.4, Bluetooth [6] does not work well for this use case. Instead, we used Wi-Fi Direct [39], a relatively new Wi-Fi mode optimised for p2p communications, which is now a standard feature on new smartphones, as our communication medium. We showed performance characteristics of both Bluetooth and Wi-Fi Direct in various p2p game scenarios.

Matching game players in an efficient way. A drawback of pure p2p solutions is that neighbour discovery can take a long time. We overcame this by using the insight that all the passengers still have Internet connectivity via cellular connections — albeit with high latency and low bandwidth. We leveraged this and used a central server to perform the matchmaking of players. This also allowed us to match players based on various pre-collected player information such as skill levels, travel times, and other preferences.

Working with minimal modifications to existing systems. For *GameOn* to be successful, it should be as backward compatible as possible. As such, we designed it to run as a normal application (root access is not needed) with minimal changes needed to existing applications. For instance, we retained the existing client-server models used by most existing games to minimise code changes. Thus, one smartphone will have to serve as both the master server as well as a client — we show that the energy overhead for the server phone is still quite acceptable. We also intentionally kept *GameOn* as simple as possible to make it easier to deploy, debug, and explain to end users.

Overall, we make the following contributions:

- A detailed analysis of the public transport travel times in Singapore. In particular, we show how long passengers are co-located on the same train or bus (which is the shared time when they can play a game together). We also present summary results for 11 other cities.
- A detailed comparison of the efficacy of Bluetooth and Wi-Fi Direct as a communication medium for playing multiplayer games. We show results from both in-lab synthetic experiments as well as real-world experiments conducted by playing games on actual public trains at various times of the day.
- A detailed description of the design and implementation of *GameOn*. This includes a discussion of how the *GameOn* matchmaker can be extended to support many more metrics (such as co-location times and connection stability) to enable spontaneous p2p games, beyond ping times commonly used by existing matchmakers.
- An in-depth evaluation of *GameOn* that comprises of both micro benchmarks involving synthetic evaluations of various system components as well as real-world tests involving actual game play, using three different popular games, on a public train (at various times of the day). The games chosen were *OpenArena* [34], *Racer* [27], and *2048* [30], which represent the shooter, car racing, and casual game genres, respectively. We have posted videos of *GameOn* being used with these games on commuter trains and buses at **http://tinyurl.com/gameon-videos**.

2. MOTIVATING SCENARIOS

Jill is heading to school and her regular commute involves a 25 minute train ride. She boards the train and settles in for the somewhat long journey. She starts using her smartphone to do her regular routine — check emails, browse news articles, facebook posts, and videos tagged by friends. However, she quickly finishes all of these and realises that she is still 20 minutes away from her station and she is getting bored.

Fortunately, she remembers about that new application, called GameOn*, that her friend asked her to install. She starts* GameOn *and sees that 3 people around her are interested in playing Quake III multiplayer (which is setup to require at least 4 people). She expresses her interest in playing the game. Within seconds,* GameOn *starts a server on one of the 4 phones, and automatically connects all the 4 game players (using their anonymous in-game IDs) to the server using Wi-Fi Direct and the game starts. 10 minutes later, the game concludes as some of the participants get off the train. Jill is happy with her performance and wonders who she was playing with (that info is not revealed).*

She realises that she still has $\approx$ 10 minutes left and she decides to see if a quick round of 2048 (a puzzle game) is possible. She re-starts GameOn *and specifies that she is looking for one other person to play 2048 with. Within seconds, she is connected with another anonymous player (on the same train) and the game starts. This continues until Jill reaches her train stop at which point she ends the game, gets off, and goes to her classes happily.*

The above scenario motivates the entire design of *GameOn*. In Section 3, we first show that passengers spend sufficient shared time on public buses and trains. We then present the design, implementation, and evaluation of *GameOn* in the remaining sections.

3. IS GAMEON EVEN PRACTICAL?

To support the above scenarios, we require a number of preconditions to be true as follows:

1 First and most importantly, commuters must be on the same train or bus long enough for a shared game session to be feasible. Prior work [29] has published the minimum game length at about 10 minutes. Accounting for the overheads of settling onto the bus/train and allowing for time to finish reading emails, news sites, etc., we pessimistically need commuters to be co-located with a large number of other commuters on the same train or bus for at least 20 to 25 minutes for a *GameOn*-like system to be plausible.
We present rigorous analysis of Singapore's transportation system data (Section 3.1) along with summary data from other countries (Section 3.2), and show that these shared commute times are very achievable in practice.

2 Second, commuters need to have a smartphone that is capable of supporting a wide range of multiplayer games. Fortunately, industry progress has solved this issues and modern smartphones (any smartphone bought from 2013 onwards) have the CPU, GPU, memory, and networking capabilities to support many different types of multiplayer games. Indeed the ability to play many types of games is a key selling point for smartphones in some countries.
Note: It is possible for our solution to be adapted to work on feature phones (using Bluetooth instead of Wi-Fi Direct – albeit with worse performance in some cases). However, we did not do that as a) the number of games available on feature phones is limited and, b) the smartphone penetration rate has been growing rapidly even in developing countries

(such as Indonesia, Thailand, and India) due to decreasing phone costs and increasing prosperity [12].

3 Third and finally, commuters must have the interest to play games while on buses and trains with others, probably random commuters. Fortunately, statistics [14] show that games are the most popular applications downloaded from any app store and this popularity increases as the population gets younger. In addition, multiplayer games tend to be the most engaging of all game types. Thus, we believe that the desire to play multiplayer games is present in a large fraction of the commuting population. We also present results from a self-reported survey in Section 3.1, showing that a majority of commuters are interested in playing mobile multiplayer games even with random commuters.

3.1 Public Commute Times in Singapore

In this section, we present a detailed analysis of the commute times observed in Singapore.

3.1.1 Singapore's Public Transportation Network

Singapore is a small country of about 715 square kilometres (≈60% the size of New York City) with about 5.3 million inhabitants. It has a modern integrated public transportation network of trains, buses, and taxis (not considered for this analysis). The bus network uses about about 360 bus routes to serve over 4,800 bus stops while the train network comprises of over 120 stations across 5 main lines. In total, the buses and trains handle over 6 million trips per day [18].

Singapore uses a NFC-based store value card system to pay for bus and train rides that requires every commuter to tap their NFC cards at both entry and exit before the actual fare is computed based on the distance travelled. This is different from fixed rate systems used elsewhere, such as the New York City and Paris subways, which only require a tap on entry. This requirement to tap in and out makes it possible for data analysts to know exactly when a particular NFC card has entered or exited a bus or train station (even though the owner of the card is unknown).

3.1.2 Singapore Transportation Data Set

We used three months of bus and train entry and exit data (from November 2011 to January 2012) obtained from the Land Transport Authority of Singapore [16]. For every public bus, we had the time and location (bus stop number) where every passenger boarded and alighted. For trains, we have, for every train station, the exact time when a commuter entered and left that train station. With these two sets of data along with the publicly available train/bus timings and route maps, we can quantitatively determine the average commute time needed to reach any destination in Singapore. Table 1 summarises the data used for this analysis.

For the purpose of this analysis, we used our university campus as the final destination and calculate the commute time statistics needed to reach our campus from any location in Singapore. Note 1: because our university campus is located down town, it is very well connected and served by 3 different train stations and 43 different bus routes across 7 different bus stops. Note 2: when performing our analysis, we only considered the most direct routes to our campus that did not require switching between trains to buses and vice versa.

3.1.3 Quantitative Analysis Results

Table 2 shows the results of our data-driven analysis for both trains and buses across all 5 weekdays for both peak hours (7.30 a.m. - 9.30 a.m.) and off-peak hours (9.31 a.m. - 5.59 p.m.). Note that we only consider the morning peak period as the evening peak period will not have too many people coming to campus.

Bus Data	Nov. 2011	Dec. 2011	Jan. 2012
Total # of Records	100,521,633	100,732,193	105,449,970
Unique Bus Routes	353	353	353
Unique Bus Stops	4873	4873	4873
Unique Commuters	3,910,636	4,364,309	4,202,792

Train Data	Nov. 2011	Dec. 2011	Jan. 2012
Total # of Records	62,272,880	63,655,069	63,092,608
Unique Train Stations	127	127	127
Unique Commuters	4,210,625	4,051,357	4,384,240

Table 1: Summary of Public Transportation Data

	Commute Time (mins)			
	Bus		Train	
	Peak	Off-Peak	Peak	Off-Peak
Mon	17.4 (9.2)	18.5 (10.6)	24.1 (11.1)	23.3 (13.4)
Tue	16.5 (11.8)	17.1 (9.8)	27.5 (12.9)	21.1 (13.7)
Wed	17.0 (10.4)	17.9 (10.1)	25.6 (11.2)	20.5 (13.3)
Thu	16.9 (11.0)	17.1 (10.3)	27.9 (12.9)	20.9 (13.6)
Fri	17.1 (10.7)	17.4 (10.2)	25.6 (11.2)	21.1 (13.6)
All	16.9 (10.8)	17.6 (10.2)	26.5 (12.1)	21.4 (13.5)

Numbers in parenthesis are the standard deviations

Table 2: Average Commute Times for Buses and Trains

The data shows that the average time spent on a bus is about 17 minutes with a fairly high standard deviation (numbers in parenthesis). For trains, the average time is about 26 minutes with a reasonably large standard deviation as well. This matches well with reported data [18] that states that trains are the preferred option for longer routes. However, even though these numbers look low, many commuters experience higher commute times as they need to take more indirect routes that involve multiple trains/buses for their commute. We show this through a survey in the following section where the majority of respondents reported high commute times with more than one transfer.

3.1.4 Qualitative Survey Results

In addition to the data driven analysis presented above, which is completely game agnostic, we also surveyed a large number of undergraduates along with a few working professionals to obtained their self-reported commute times and willingness to play multiplayer games while commuting.

We send out an online survey (with 20 questions) to various school mailing lists, and 118 participants voluntarily responded to the survey. To avoid biasing the answers, we did not provide any details about *GameOn* in the survey, and participants had no idea what the purpose of the survey was (except that it was a public transportation survey). All user-centric experiments reported in this paper were conducted after obtaining an appropriate IRB approval. We did not provide any form of compensation for doing the survey.

Out of 118 participants, 85 (72%) were males and 33 (28%) were females, across various age groups: 18-20 - 15 (13%), 21-25 - 69 (58%), 26-30 - 20 (17%), 31+ - 4 (4%). 91 (77%) were students with 24 (20%) working professionals and 3 (3%) others. 69 (58%) participants used an Android smartphone, 41 (35%) an iPhone, with just 8 (7%) others. The full set of survey responses can be obtained at **http://tinyurl.com/gameon-responses** with the raw data at **http://tinyurl.com/gameon-responses-raw**.

City	Average One-Way Commute Time (minutes)
London	39.5 [36]
New York	40.0 [1]
Montreal	38.0 [25]
Toronto	39.5 [25]
Tokyo	66.0 [4]
Seoul	53.0 [33]
Hong Kong	46.0 [8]
Taipei	37.5 [26]
Beijing	97.0 [7]
Delhi	42.3 [38]
Mumbai	47.3 [38]

Table 3: Average Commute Times for Other Cities

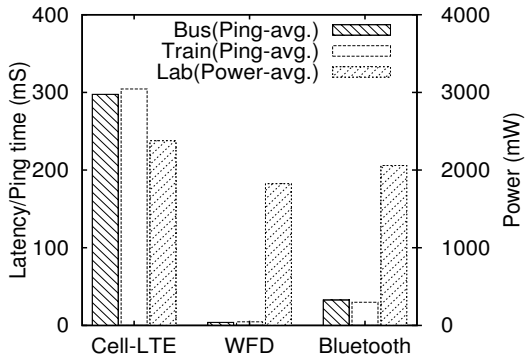
The key survey results were the following: 90% (106 participants) used public transport daily with 67 (57%) taking buses and 86 (73%) taking trains (note: some participants take both trains and buses). The average one-way commute time for the participants (time spent only on buses/trains/cars going from home to work/school excluding any walking time) was "Less than 10 minutes" - 4 (3%), "10-20 minutes" - 17 (14%), "20-30 minutes" - 27 (23%), "30-40 minutes" - 27 (23%), "40-50 minutes" - 15 (13%), "50-60 minutes" - 13 (11%), "Other" - 15 (13%). In particular, 83% of the participants had commute times $>$ 20 minutes. In addition, 61% of the participants took more than one bus/train during commuting.

In addition, 90 participants (76%) stated that they played mobile games on their phones with 67 (57%) saying that they played mobile games while commuting. The most popular type of game played were Casual games (56 responses (47%)), with Puzzle Games (37 (31%)), and Strategy Games (30 (25%)) close behind.

Finally, 64 participants (54%) answered yes to the question "Are you interested in playing multiplayer games with other commuters travelling in the same bus/train/car?". When asked why they wanted to play these games, the answers provided were "Ease the boredom during the commute" - 44 (37%), "Potential to meet more people who share similar interests" - 34 (28%), "Thrill of competitive challenge inherent in multiplayer gaming" - 33 (28%), "Other" - 2 (2%).

For the 54 participants (46%) who were against the idea, the most common reason offered (via a free form text box) was the unwillingness to pay for 3G/LTE bandwidth just to play a game on the train – "Dataplan consumption and slow / connectivity issue when in train". They also felt that the 3G/LTE speeds were not good enough for gaming – "You need a solid connection when playing such games during commuting. Singapore's telco isn't able to provide that solid connection underground". Another strong opinion raised was the fear that playing with nearby strangers would impact their real world comfort levels – "Do you really think we are that open to play with strangers standing right next to us? Its okay when we do it over the internet because we don't know who that is". Finally, some participants feared that the game experience would be bad due to poor player quality or players leaving abruptly.

Overall, the survey results indicate that there is potential for *GameOn* to be successful. However, to become even more accepted, *GameOn* must reduce the use of 3G/LTE bandwidth that a) may have high usage charges in some countries, and b) might have connectivity issues in certain parts of the transport network. *GameOn* overcomes this by using completely local bandwidth provided by Wi-Fi Direct to support the various games. Thus, it does not incur any charges and is much less likely to have connectivity issues. In addition, the survey shows that the matchmaking component also needs to take into account the physical proximity of people when making a match – we might want to avoid matching people who are physically too close to each other and also take into account the expected trip length for each person to avoid game interruptions caused by people leaving.



All experiments were conducted with a Galaxy S3 & S5 Android smartphones. The latencies were computed by pinging a common in-lab server. The power consumption was measured in our lab using a Monsoon power monitor [24]. WFD == Wi-Fi Direct.

Figure 1: Latency & Power Consumption Comparison

3.2 Commute Times in Other Major Cities

We additionally describe analysis of the commute times observed in other urban cities (from prior work and online sources). Table 3 shows our findings. What we observe is that commute times in other cities tends to be higher (some quite significantly!) than those in Singapore (which were computed using raw data). Hence, *GameOn* might also prove to be useful in other cities.

4. DESIGN GOALS & ASSUMPTIONS

In this section, we present the design goals for *GameOn* along with our assumptions.

4.1 Design Goals

The main design goals for *GameOn* were:

- **Provide a smooth gameplay experience**: This is the most important design goal and it permeates all the other goals below. In a nutshell, *GameOn* should add as little overhead as possible to both game players and game developers.
- **Low latency networking with sufficient bandwidth**: A key cause of discontent in multiplayer games is lag caused by network issues. Thus, *GameOn* should not introduce any user noticeable lag or bandwidth artefacts when games are being played. We compared the client to server latencies and energy consumption of LTE, Bluetooth, and Wi-Fi Direct (results shown in Figure 1) and found Wi-Fi Direct to have the lowest latencies and the lowest energy consumption. *GameOn* thus uses Wi-Fi Direct for the actual game plays while using the cellular Internet connectivity only for the matchmaking process (a low bandwidth latency tolerant task that requires history tracking)
- **Easy and effective matchmaking**: Commuters should be able to easily express their game interests and also easily find games that they can join. The matchmaker should also ensure that the players in the game do not leave abruptly and that any skill, demographics, or other factors are also factored in, where necessary, when performing the matchmaking. For example, even though *GameOn* enables playing multiplayer games with fellow passengers in close proximity, some players may not want to be matched with players located next

to them on the bus/train as they may not want their physical identities to be easily discovered. To support this, we use a centralised matchmaking service, that can track historical performance etc., located in the cloud.

- **Use simple user-space mechanisms**: For *GameOn* to be easily deployable, it has to be a user space component (i.e., no rooting of the phone is required) and it should be as simple as possible (making it easier to explain to end users and more robust overall). In addition, we retain the existing client-server models used by almost all multiplayer games. However, this requires us to dynamically host the server on one of the smartphones of the commuters playing that game. The game and player statistics are then uploaded to the matchmaking service after the game ends.
- **Low energy usage**: *GameOn* should not add any significant energy cost beyond the cost of playing the game itself. In particular, the smartphone that has to host the game should not see a large increase in energy usage.

4.2 Assumptions

The assumptions we made when devising a solution that addressed our design goals were the following: 1) We assume that every commuter had access to a smartphone with cellular Internet connectivity. The smartphone was necessary for gameplay while the connectivity was necessary to use a central matchmaker. 2) Some changes to the game interfaces may be needed for *GameOn* to be fully operational. In particular, the game will a) have to report game statistics (in game scores etc.) to *GameOn* so that it can be used during matchmaking and b) have to use the *GameOn* APIs to send data to/from other p2p clients. Indeed, to demonstrate how easy our APIs are to use, for our evaluation, we converted, with minimal effort, an open source *single player version* of a popular game, *2048*, to work as a multiplayer game using *GameOn*. Finally, 3) we assume that the multiplayer games will only be played by a small number of players – 2 to 6 players at most. This system is not designed by larger games that involve 10s or 100s simultaneous players. However, there can be multiple games being played simultaneously in the same area.

4.3 Overall Architecture

To satisfy the design requirements stated in Section 4, *GameOn* was designed to use a hybrid p2p architecture composed of *GameOn* clients interacting with each other using local networking capabilities coupled with a matchmaking service located in the cloud. Figure 2 shows the architecture overview of *GameOn*. We focus our discussion on only a few core modules (the shaded blocks in Figure 2). Overall, *GameOn* comprises of two components:

1. *GameOn* clients: A *GameOn* client supports various multiplayer games that can be played by peers co-located on a train or bus. It has a UI component that allows players to login, specify grouping preferences, and discover co-located peers. When a user starts *GameOn*, peer discovery is started and any discovered peers (along with their performance metrics) is passed to the matchmaker. Upon request, the matchmaker provides the *GameOn* client with the list of playable games and corresponding game hosts. When a peer is already hosting a user's desirable game, the *GameOn* client makes a new game client connection to the peer. Otherwise, it serves as a game host for the user's specified game and waits for other players to join.

The game play is automatically initiated when the required number of players join. During game play, *GameOn* clients form a star topology by default, and all the game packets are relayed through

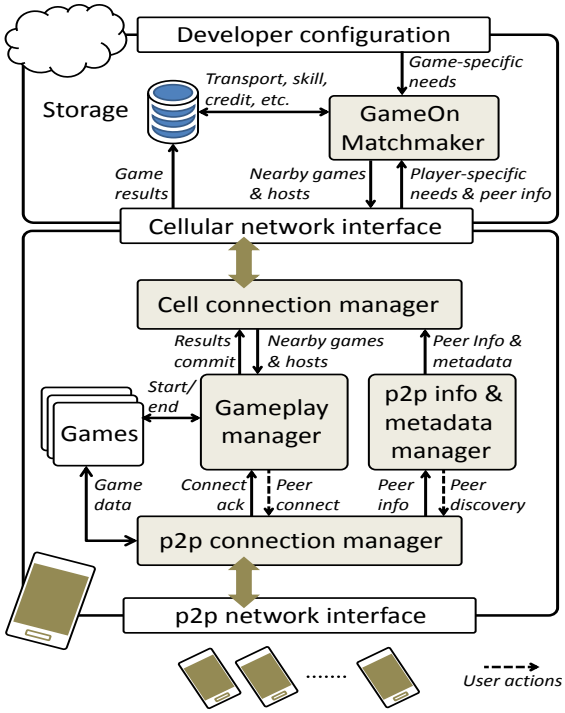


Figure 2: The *GameOn* Architecture

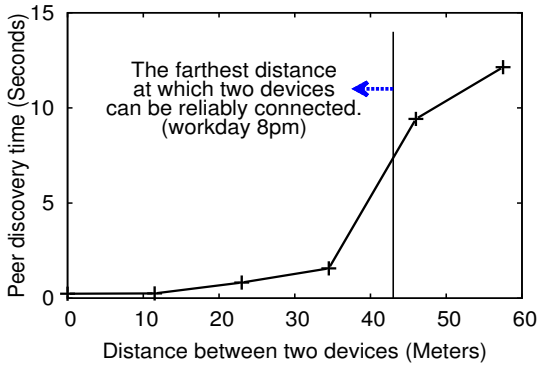
the host device; *GameOn* also supports a multi-hop topology when a host cannot be connected to a client directly due to distance (located on the other side of the train for example). The *Gameplay Manager* component configures the *p2p connection manager* component by specifying the role of the player in a group and the topology of collocated peers. Note: We do not require any changes to the existing game logic to support this type of p2p game play. *GameOn* wraps around the original networking APIs (Section 4.5) used by the multiplayer games and automatically re-routes packets to p2p hosts using either Wi-Fi Direct or Bluetooth. When the game ends, the game results and performance data are reported to the matchmaking server to update its records.

2. The *GameOn* matchmaking server: This server allows *GameOn* clients to find a set of players that are co-located and who will stay on the same bus/train long enough for a satisfying game session. It collects various information required for p2p matchmaking from *GameOn* clients such as the observed signal strength and ping times between peers, as well as the mobility patterns (how long they spend on a specific train etc.) and skill levels (how well they did in previous sessions of a game etc.) of each user (as represented by their mobile phones). We show how the matchmaker can use all available data (mobility history, user preferences, game-specific skill levels, and performance measures) to match the best set of people together for any game request.

4.4 Which p2p Protocol is Best on Trains?

The success of *GameOn* depends on having reliable p2p networking connectivity between peers on a bus or train. However, these are particularly challenging environments due to their movement, layouts, and frequent passenger movements. In this subsection, we present detailed performance results to understand the performance of wireless protocols in these environments.

The first key question we addressed was the choice of network protocol. The two main options were Bluetooth and Wi-Fi Direct. Eventually, we chose Wi-Fi Direct as its overall performance, beyond just the better latencies and power consumption (Figure 1), was better as explained below.



We found that discovery times $>$ 10 seconds, corresponding to a peer distance of about 46 meters (2 carriages away) was a good indicator of a peer that could not be reliably connected to. We then shortened the inter-device distance by a meter at a time (roughly) and were able to reliably connect two devices at a 38 to 43 meter range. We found that even very far devices (70 meters away) could be eventually discovered (taking 108 seconds). We omit these long tail numbers from the plot.

Figure 3: Discovery Time vs. Distance (Wi-Fi Direct)

4.4.1 Experiment Setup

To understand how the protocols behave in realistic environments, we conducted experiments using Galaxy S3 and S5 smartphones on a train during three time periods – when the train was extremely full (6 p.m.), normal load (8 p.m.), and empty (midnight). We used the Galaxy S3 (running Android 4.3) as the stationary peer and moved the S5 (running Android 4.4.2) to different adjacent train carriages (up to 3 carriages away) and measured (on the S3), using both Bluetooth and Wi-Fi Direct, the RSSI signal strengths of the S5 and the ping times to the S5. Each train consisted of 3 carriages [17]. Each carriage was filled with numerous metallic objects (seats, hand rails, guard rails etc.) and was 23 meters in length, 3.2 meters in width, and 2.1 meters in height with a very small (negligible) inter-carriage gap. We repeated each experiment multiple times over different days. We do not report any results for buses as the public buses are shorter in length (each bus is about 12 meters long [32]) than a train carriage. Thus, the train is a more demanding environment.

4.4.2 Peer Discovery, Connectivity & Density

The first step in connecting phones together in a p2p fashion is to discover them. In our preliminary measurements, we also discovered that just because a device can be discovered does not mean that a successful connection can be made to it. A typical Wi-Fi Direct connection starts with scanning, then group owner negotiation, then provisioning, and finally DHCP. When peers are side-by-side, these steps can be done quickly without packet loss. However, as peers are further away and/or in "noisy" environments, these steps can become harder to complete.

To include the effect of people density in this experiment, we performed it during normal hours (when the train was normally crowded). With this level of crowd, we can assume that the density of people increases linearly as we move further away from the discovery node. To perform this experiment, we used one device as the stationary node and moved another device further and further away (in increments of half a train carriage every time). Both devices then tried to discover the other device. In addition to discovering the device, we also tried to connect to the device after it was discovered. We found that even though both devices could eventually discover each other (taking about 10 seconds) even at a two train carriage distance (about 46 meters), they were unable to actually connect to each other. However, at shorter distances, the two devices could discover and connect to each other. Our experiment results are shown in Figure 3.

The *GameOn* matchmaker has to make decisions about peering without being able to actually check the connectivity between those hosts – at best it knows something about inter-peer ping times. As such, a naive host assignment might pair hosts together who can discover each other but cannot actually connect (because one of the steps involved (probably DHCP) fails). What we discovered, for Wi-Fi Direct, was that the discovery time turned out to be a good predictor of connectivity. In particular, as shown in Figure 3, peers that could be discovered within 10 seconds (i.e., before the discovery time shoots up) can be successfully connected to.

However, even a 10 second discovery time can be too long as every scan is costly in terms of battery usage. Thus we reduced the scan time to 5 seconds to strike a balance between power consumption and finding enough nearby connectable peers. Each peer performs a scan every time it requests a peer match list from the matchmaker. This allows the matchmaker to gradually build a client map for a bus / train without needing aggressive client scanning.

The discussion above is solely for Wi-Fi Direct. We also repeated this discovery and connectivity tests for Bluetooth and achieved very disappointing results. We found that Bluetooth was unreliable beyond 20 to 25 meters. We show the difference between Bluetooth and Wi-Fi Direct in terms of RSSI and ping times in Figure 4.

4.4.3 Effect of Density on Network Latency

We now investigate the effect of people density on wireless performance – in particular the latency of the connection. This is important as games require low latency network connections. To do this, we picked three different times of the day (corresponding to light, normal, and heavy train/bus use) and four different inter-client distances. We measured the inter-client ping times and also measured the RSSI values. Note: the ping times changed when we repeated this experiment across different days. In the rest of this section, we present the ping times for the worst day.

Figure 4 shows how the signal strength and ping times changed as the distance to the peer phone varied. We observe that in all cases, Wi-Fi Direct performs better than Bluetooth. In particular, the second row of results shows the RSSI observed when the stationary phone connects to the moving peer using various protocols. The actual RSSI values are not important (as they fluctuate due to noise etc.). What matters is the pattern and trends.

For Wi-Fi Direct, we could connect using both the 5Ghz and 2.4Ghz spectrums with no clear winner in spectrum choice emerging. We found that, when the train was normally occupied, the maximum distance that a peer could be connected to was 2 carriages away using Wi-Fi Direct. For Bluetooth, the range was just 1 carriage away. When the train was busy, the range of Wi-Fi Direct decreased to just the same train carriage while Bluetooth could only usefully connect to clients very close by (distances greater than 20 meters had very high ping times).

The last row shows the ping times achievable to the connected peer using Wi-Fi Direct and Bluetooth. In all cases, the ping times for Wi-Fi Direct are much lower than Bluetooth. In addition, Bluetooth stops working (the line for the ping graphs stops) at much lower distances than Wi-Fi Direct. For example, on a normal occupancy train, Bluetooth stops receiving pings at about 20 meters while Wi-Fi Direct continues until about 60 meters.

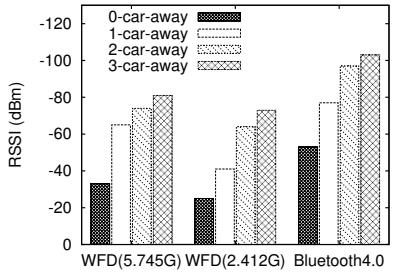
4.4.4 Connectivity Issues at Train Stations

Unfortunately, even with Wi-Fi Direct, we found that if peers were 2 or more carriages apart, on entering a station, the process of

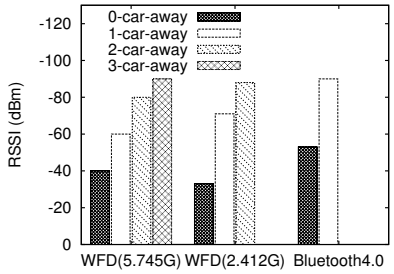
(a) Empty (midnight)

(b) Normal (8 p.m.)

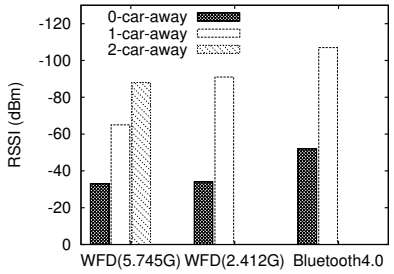
(c) Busy (6 p.m.)



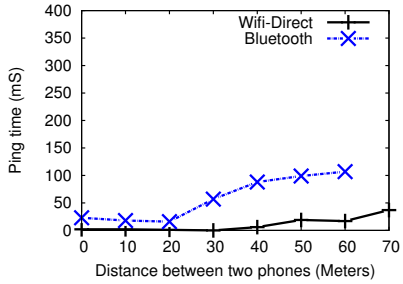
(d) Empty



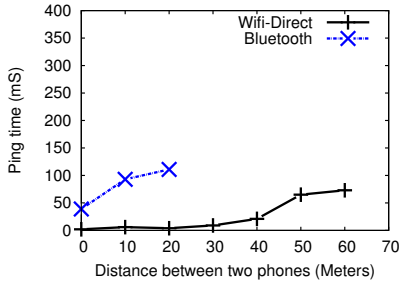
(e) Normal



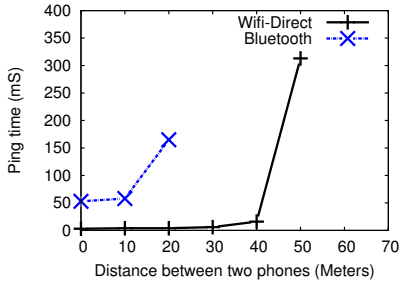
(f) Busy



(g) Empty



(h) Normal



(i) Busy

The top row shows the state of the train (empty, normal, busy) at the time of the measurement, the middle row shows the observed RSSI values of a peer device by a stationary phone (after connecting to that peer) when the peer device was placed further and further away (by up to 3 train carriages). The bottom row shows the observed ping times on one device (to the other) as the other device moved further away. Missing data in the figures indicates that the other phone was not connectable to or pingable at that distance using that protocol. Each result was repeated multiple times over different days and the averages are shown. We omit the error bars to improve readability as these results are presenting trends (actual values are not important).

Figure 4: Comparison of Observed RSSI & Ping Times at Different Times on a Public Train

Games	Multiplayer already?	Language	Lines of code added
OpenArena [34]	Y	C / C++	8
Racer [27]	N	Java	86
2048 [30]	N	Java / JavaScript	14

Table 4: Three Games Modified to Use *GameOn*

opening the doors to let passengers embark and disembark resulted in high latency spikes. Figure 5 shows this where a peer (located 1 carriage away) experiences constant good ping times while another peer (located 2 carriages away) experiences consistent latency spikes which corresponded directly with the train entering a station, stopping, opening its door, and then leaving (the high latency goes away at this point). We have no current solution other than adding a matchmaking heuristic to not match peers more than one carriage away for games that cannot handle brief latency spikes.

4.5 Modifying Games to Work with GameOn

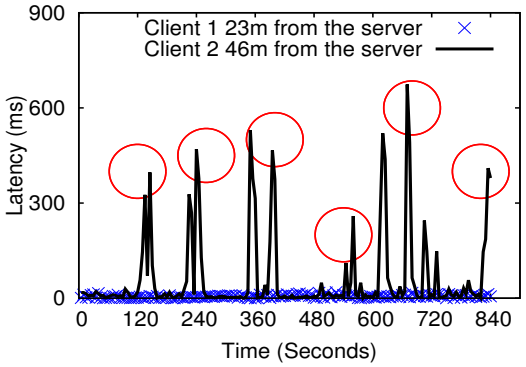
In this section, we describe how *GameOn* support can be added to existing games by making two different modification; 1) support local client-server multiplayer, and b) interface with *GameOn's* networking, matchmaking, and reporting APIs.

4.5.1 Games Used for Evaluation

The first requirement for any game to work with *GameOn* is for the game to support client-server multiplayer. To make a game multiplayer compatible, it requires creating a server component for the game along with changing the UI, where necessary, to display any multiplayer-specific information. In this work, we decided to use both existing multiplayer games as well as support single-player games to understand the complexity inherent in making different types of games work with *GameOn*.

The three games we used are described in Table 4. *OpenArena* was the only game that already multiplayer-enabled with separate client and server components. Even in this case, as shown in Figure 6, we need to modify the game to use a local server (that is running on a peer phone and accessed via Wi-Fi Direct) instead of a server sitting in the cloud that is accessed via a cellular link.

Unlike OpenArena, *Racer* and *2048* were single player games that had no server component. For both games, we created a simple server that basically stored and forwarded packets to other clients. To help developers to extend existing singleplayer games to communicate with a game server, we provide two functions to share game state: *sendCommand(String jsonObjectInString)* is used to



During a 15-minute experiment, the train stopped at 6 stations (times at each station are circled). We conducted this experiment during peak hours.

Figure 5: Latency Spikes at Stations

send client moves periodically, and *updateSnapshot()* is used to receive global game states from the game server.

In all cases, the amount of additional code we had to write was minimal (86 lines for *Racer* and 14 for *2048*). For both games, we did not modify the UI component and just leveraged the existing game code that could already display the output of a secondary player. Currently *GameOn* does not provide any UI modules as these components are very game specific. Instead, *GameOn* focuses on the networking components and provides enough infrastructure (and APIs) to allow game developers to concentrate on the UI and gameplay portions of the game and let *GameOn* handle all the networking bits.

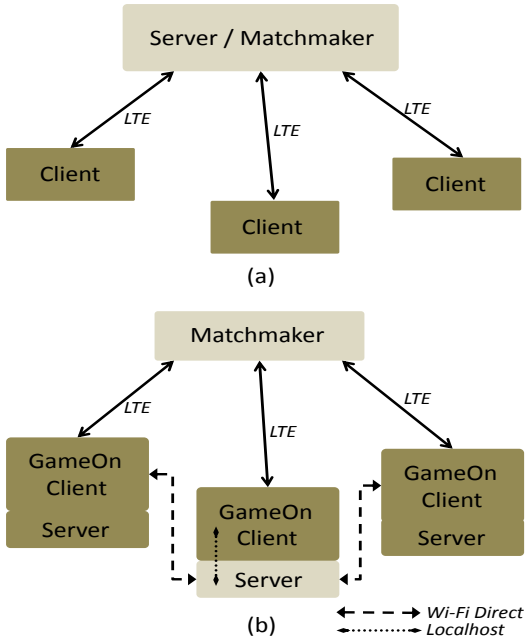
4.5.2 *Using GameOn Libraries*

Next, we had to modify all three games to use the *GameOn* APIs. This required 1) using the *GameOn* matchmaking service, 2) using the *GameOn* networking libraries, and 3) using the *GameOn* game statistics reporting libraries.

The matchmaking service is initiated by the player (in our prototype, the player presses a UI button). This is a single API call in *GameOn* and it sends a request to the matchmaker, using JSON objects, along with the performance measurements from the current client (neighbours discovered, ping times to neighbours etc.). The matchmaker responds with a list of games and hosts. The developer can then use *GameOn's* p2p APIs to initiate a game request with discovered clients. Once the game starts, the state sharing APIs described earlier are used to play the game. Finally, the developer has to use the *GameOn* reporting libraries to commit the game results back to the matchmaker (for use in global statistics and future matchmaking sessions). Note: games don't communicate with the matchmaker directly. That functionality is handled transparently by *GameOn* .

The *GameOn* networking libraries handle most networking requests. Internally, the *GameOn* networking logic uses two layers: a physical Wi-Fi Direct group, and a logical game group. In our current implementation, the physical group is built using legacy Android APIs (for backward compatibility), while the logical group is built using TCP/UDP sockets. All status sharing information is exchanged via the TCP/UDP sockets. For simplicity reasons, when a player initiates a new game, our current implementation makes him or her the game server and owner of the logical group. All subsequent players are clients in the group. This logic can be changed, if necessary, to share the server load among other all players.

Overall, all these changes were easy to implement. A single grad student, with no game development experience, managed to modify



Unlike traditional approaches, in *GameOn*, each peer can serve as a client and also as a server. *GameOn* selects only one peer to serve as the game server. Game traffic is exchanged with peers using p2p connections (usually Wi-Fi Direct).

Figure 6: Traditional Approach (a) vs. *GameOn* Approach (b)

all three games in less than 2 days each. Most of the time was spent understanding how each of the games maintained its game state (to find the right places to insert the networking and statistics reporting APIs). As shown in Table 4, the amount of code that needed to be created was minimal. *OpenArena*, in particular, needed very little code as it already had discrete client-server components.

The goal of the *GameOn* matchmaker is to find groups of commuters on the same bus / train who can play a game together. In such dynamic environments, these formed groups should be chosen so that they are stable – i.e., members don't abruptly leave. For example, a group is not considered to be stable if the elected game host alights (thus ending the game) very soon after a game session is started.

4.6 Data Used For Matchmaking

To make these matchmaking decisions, the matchmaker can use data from three information sources as shown in Table 5.

4.6.1 *Performance Data*

The first is performance data such as RSSI values and ping times of various nodes (as observed by other nodes). The *GameOn* client periodically updates its discovery results to the matchmaker. With this data, the matchmaker can create a logical map of where each player is situated relative to other players. It can then use the heuristics shown earlier (peers more than 1 carriage apart can experience variable ping times etc.) to match clients together.

In addition to network measurements, we can also use historical predictions about how long a particular client will remain on the train/bus as a key input. These values can be computed using historical data (using techniques similar to Balan et. al [5]). We show in Section 6.3.4 how using predicted trip times can improve the matchmaking performance.

Data	Reason To Use It
Performance	
Detection time	Hint for a robust connection
RSSI	Hint for a robust connection
Ping time	Hint for distance and crowdedness
Pred. Trip time	Games don't end abruptly
Game-specific	
Player Level	Ensure a fair/engaging game
Player Credibility	Ensure no cheating
Min. Player No.	Ensure game is interesting
User-specified	
Only with friends	Guarantee game experience
Nobody close by	Reduce real-world detection probability
Similar interests	Find future friends

Table 5: Data that can be used by the Matchmaker

4.6.2 Game-Specific Data

The next category of matchmaking data is game specific data. This is data that categorises players into different buckets – based on their skill levels, probability of cheating, and other game-specific data. In addition, games can specify minimum and maximum game player numbers to ensure a high game experience. Grouping players according to the skill level is a well-known matchmaking metric. There are a few algorithms that have been employed by commercial video gaming platforms. For example, Xbox Live [23] uses the TrueSkill ranking system [13] that computes the skills of gamers. Unfortunately, our current prototype does not use any of this data or these algorithms as we do not have the game-specific player information to generate this data historical data. However, adding this data into the matchmaking decision process, when the information does become available, is fairly straightforward.

4.6.3 User-Specific Data

The last category of data that can help the matchmaker is user-specific data. This is data that encapsulates a specific user's preferences and interests. For example, a player may not want to play games with nearby people as they are afraid it might lead to a confrontation. On the other hand, another player might want to meet nearby game players – but only if their interests match. Unfortunately, similar to game-specific data, our current prototype does not use this type of data as we have no historical or player records to generate the data from. However, once the data is available, integrating it into the matchmaker is easy.

4.7 Matchmaking Algorithm

In this work, we do not propose any new matchmaking algorithms. Instead, we leverage existing techniques to build a reasonable matchmaking solver. Our current prototype uses a weighted sum of components to determine the final match score of each player relative to every other player. The matchmaker then clusters these matched scores together to group players together who have similar scores. Currently, we use equally weighted normalised forms of co-location time, detection time, and ping time as the data sources for the match. In future work, we plan to investigate more sophisticated algorithms (including dynamic matchers that change their match goals (i.e., weights) based on the current situation) as well as add more data sources to the matching process.

5. IMPLEMENTATION

The *GameOn* Android client was implemented using Android 14 APIs (Android 4.0) as a user space application. It implements two background services that do the following; 1) *Cell connection manager* (225 lines of code) that uses *WebSocket*s to communicate with the cloud-based matchmaker using JSON objects, and 2) *p2p connection manager* (1,027 lines of code) that implements the functions required to support multiple communication mediums (Wi-Fi Direct, Bluetooth, and etc.) as well as support multiple roles (client, server, relay node, and etc.). The *GameOn* client also provides a simple UI (201 lines of code) for the player to sign in, configure which games are available, configure their in-game names (handle), and to select games to play, and accept game requests. All the components are wrapped around a central control core (539 lines of code) that runs in separate threads.

We implemented the matchmaker in Java (188 lines of code) using the Play Framework [31] version 2.3.7. The matchmaker uses *WebSocket* and multiple threads to support multiple *GameOn* clients. All client generated data is stored in a MySQL database. The server also has a web interface for game developers to configure their game requirements and access game and credit records. The code size is small as the matchmaker currently uses "Performance" data only to make its decisions. However, as discussed earlier, the matchmaking logic can be easily modified to support use other data sources as and when they become available.

6. EVALUATION

In this section, we present performance evaluation of *GameOn*. We first experimented *GameOn* 's performance under various real-world use cases. In addition, we present detailed results from micro-benchmark experiments conducted under controlled settings, including overheads of matchmaking and hosting games as a server, performance impacts by various underlying network topologies (star topology vs. multi-hop topology), and impact of co-location time to game plays.

6.1 Experimental Setup

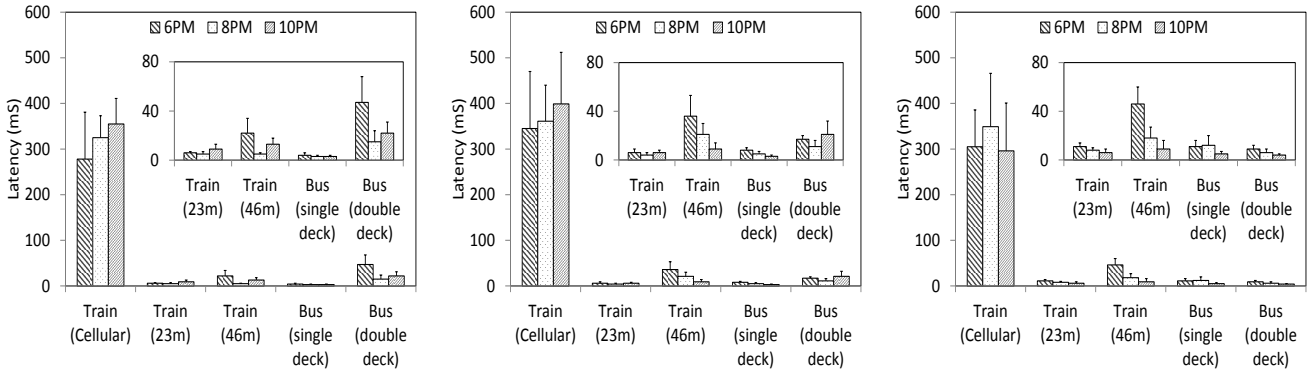
We performed all the experiments using Samsung Galaxy S3 (running Android 4.3) and S5 (running Android 4.4.2) phone. We used the three benchmark games described in Section 4.5 for all our real-world usage results as well as some of our micro-benchmarks. The matchmaker was run on an Ubuntu server with a 3.4GHz 4-core CPU with 32 GB of memory. All power consumption values were measured using a Monsoon power monitor [24].

6.2 GameOn Working in Real Environments

We evaluated the end-to-end real-world performance of *GameOn* by playing three games on real public transports. The main goal was to compare *GameOn*'s performance with that of the game played with *GameOn*. Each experiment was a 10 minute game session conducted by a four person group. After each gameplay session, all group members were asked to report their current phone battery level (which was compared to the reading just before the session started). [1]

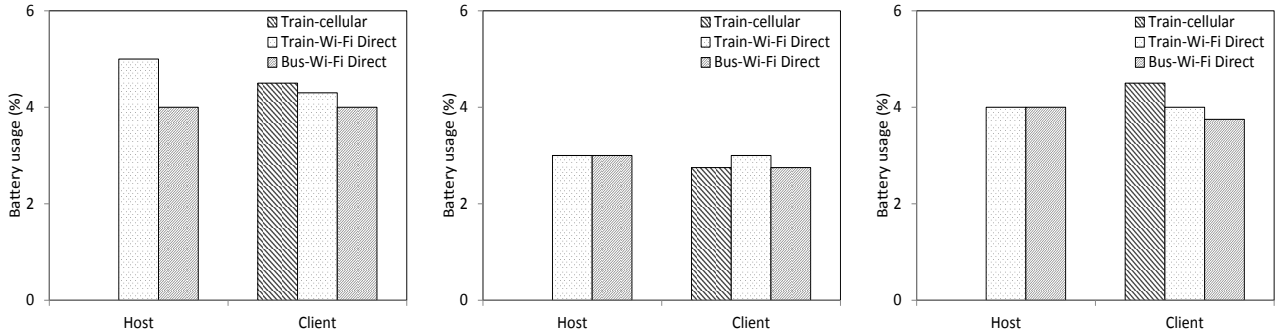
Figure 7 shows the latencies observed when playing the three games under five scenarios across three different times. In the first scenario, the games were hosted on an Internet server that was accessed using a cellular LTE connection. All four players in this scenario played solely as clients. The remaining four scenarios use *GameOn* where one peer device is selected to be the server with all the other peers connecting to it via Wi-Fi Direct. The four scenarios were "All players in the same train carriage, but spread throughout the carriage" (Train-23m), "All players spread across two train carriages (Train-46m), "All players spread across the same single deck

[1] **http://tinyurl.com/gameon-videos** has videos of *GameOn* being used by commuters to play real games on a commuter train



(a) Quake 3 (OpenArena) (b) Racer (c) 2048

Figure 7: Game Latencies across 5 Scenarios and 3 Test Times



(a) Quake 3 (OpenArena) (b) Racer (c) 2048

Figure 8: Battery Usage after 10 Minutes of Game Play

bus" (Bus-Single-Deck), and "All players spread across the same double deck bus with the server on the lower deck" (Bus-Double-Deck). During game play, we periodically logged the ping latencies to the server on each client phone.

From the figure, we observe that LTE latencies are about 10 times longer latencies than *GameOn* and that *GameOn* has very low latencies even across different types of transport and at different times (peak hours, normal etc.)

Figure 8 shows the battery usage of the phone when playing those three games. Since these experiments were done on buses and trains, we could not connect a hardware power monitor to the phones. Instead, we just used the Android battery levels as a gauge. The "Host" values is the power consumption of the phone that was chosen to host the server while "Client" values are the power consumption of the other client-only phones. Note: the "Host" phone serves as both a server *and* a client.

From the figure, we observe that hosting a server is not that expensive – power wise. Indeed, the power consumption for Hosts and Clients are quite similar and within the margin of error. Across the protocols, the power consumption is also somewhat similar.

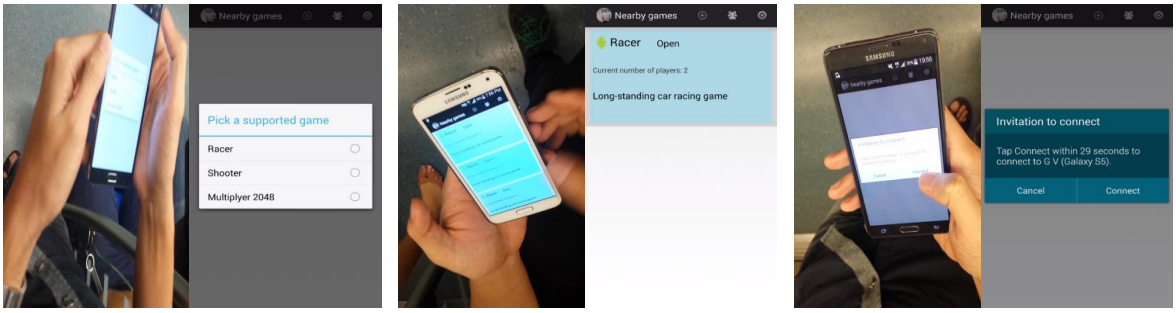
The measured latency and energy values show that *GameOn* is capable of providing good local multiplayer game experience even in different types of train and bus environments. However, does it impact the user experience in some subtle way? To verify this, we asked each of the 4 game players to answer two self-reported questions on whether they felt the game was playable. To calibrate each member, they were asked, before doing the experiment, to play each game in a lab setting with no *GameOn* modifications to understand what the unmodified game felt like under perfect conditions. The two self reported questions were 1) "The game experience is the same as the one in the lab" and 2) "The phone feels

Figure 9: *GameOn* Being Used on a Real Public Train

hotter than it did in the lab". For both questions, the members had to answer using a 5-point Likert scale (1 – Strongly Agree to 5 – Strongly Disagree).

The final score was that all 4 game players strongly agreed that the modified game had the same experience as the in-lab unmodified version. In addition, all 4 players also strongly disagreed that the phone felt hotter than it did in the lab. However, they also mentioned that one of the games, *2048*, was not the easiest to play in a multiplayer fashion due to some UI limitations. However, this bug was not introduced by *GameOn* and was beyond our ability to fix.

Figure 9 demonstrates two players playing a game on the same train using *GameOn*. In this use case, the two players are a half-carriage away from each other – one sitting and one standing. Figure 10 shows the matchmaking process to start the game session. At step (a), the player 1 starts a new game session using the *GameOn*



(a) Player 1 Starts a New Game (b) Player 2 Searches for a Game to Play (c) Player 2 Joins Player 1's Game

Figure 10: The Demonstration of the Bootstrap of a *GameOn* Game Play

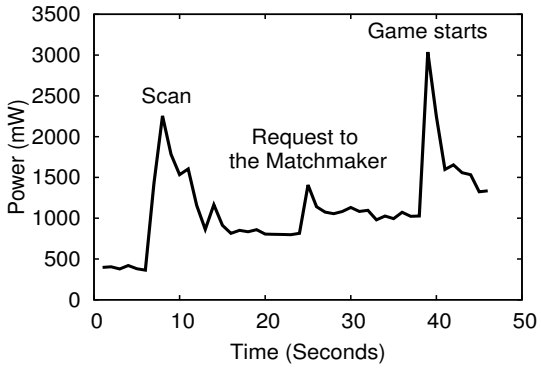
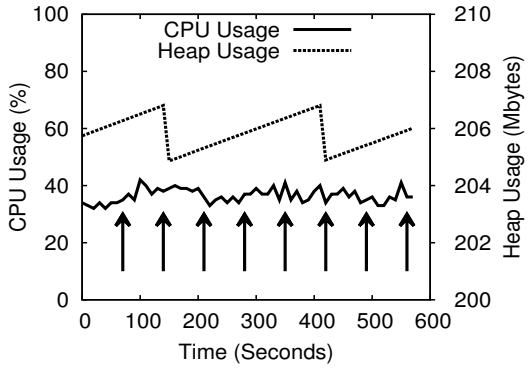


Figure 11: Energy Overhead of Scanning



Up to 8 clients join the group gradually. The arrows represent the time at which one more client joins. The saw tooth decrease in heap size is when the Java garbage collector activates.

Figure 12: CPU & Memory Overhead

UI, by selecting a game to host. At step (b), player 2 starts *GameOn* and queries the *GameOn* matchmaker to find the available games in their vicinity. Player 2 then picks one of the available games through the *GameOn* UI – he can only host a game if there are no suitable games available. Finally, at step (c), player 1 accepts the join request from player 2, and the game starts.

6.3 Micro-benchmarks

We now present micro-benchmarks results:

6.3.1 GameOn Overheads

We evaluated the overheads of two key operations: peer discovery and requesting a suitable game group from the matchmaker. Figure 11 shows the energy cost of scanning for nearby players via Wi-Fi Direct and that of sending a request for a game group. By themselves, both actions cause reasonably high spikes in the power consumption. However, compared with the power spike when the game itself starts, the scanning and requesting costs are acceptable.

6.3.2 Resource Usage and Group Scalability

GameOn selects a player to host the game server and all the other players will connect to this server. As shown earlier, this does not increase the energy cost of the server device. However, what about the scalability of the device? Can it support multiple game clients without any performance degradation?

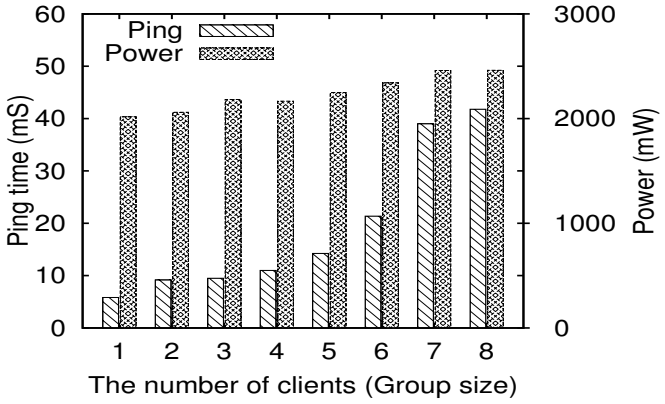
To understand this, we scheduled 8 clients to join a particular server one after the other at fixed intervals over a 10-minute period. On the server device, we logged its resource usage, including CPU utilisation, heap usage, and network traffic using Wi-Fi Direct, every 10 seconds. Figure 12 shows the CPU and heap usage plots. Each plot starts from a single client case where the server phone is connected to itself with new clients (up to a max of 8) periodically connecting. The heap usage shows a zigzag curve due to memory being reclaimed by the Android garbage collector at regular intervals. From the figure, we observe that the CPU and heap usage do not substantially increase even when the server device is hosting all 8 client players.

We next investigate how large p2p groups can become before performance (in terms of server ping times) and server energy consumption become factors. Figure 13 shows the ping times and power consumption when the group size scales up. When connected to 8 players, the power consumption of the server increases 23.7% compared to hosting just 1 player. Thus, hosting a game does not add a very large overhead to the phone's energy usage. However, we found that the ping latencies increase quite fast as more and more clients are added. In particular, we observed a large latency rise when the 7th client was connected. Thus, we find that a current modern smartphone can comfortably serve as the server for up to 6 clients. After this point, the ping latencies start to increase significantly which could result in gameplay issues.

Figure 14 shows the network usage of the server in terms of the number of bytes exchanged over the Wi-Fi Direct link. We observe that both the received and transmitted traffic grows quadratically as the group size increases. A transmitted packet from the server usually includes a snapshot of the whole group state, while a received packet usually includes only a single client command or update. Thus, on the server, the amount of data received is usually much lower than the amount sent.

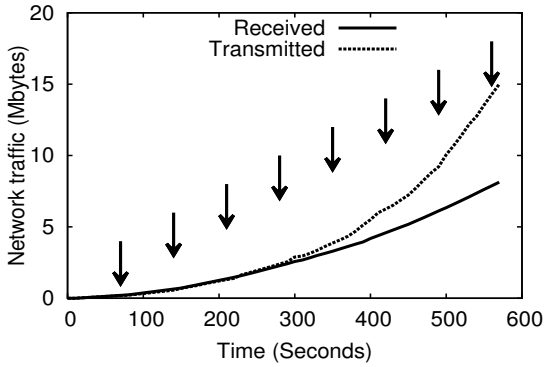
6.3.3 Support for Other Topologies

In all previous experiments, we have used a star topology where every client is connected directly to the server. However, in some cases, a client may not be able to connect directly to the server (when the client is at the other end of a crowded train for example). For example, Figure 15 shows a scenario where the four players are



The ping latencies and power consumption of the server device when hosting a game. The ping time at the client side increases quickly while the power consumption at the host side rises linearly. The result proves a modern smartphone can modestly support a group of 1-8 players.

Figure 13: Power & Latency Overhead

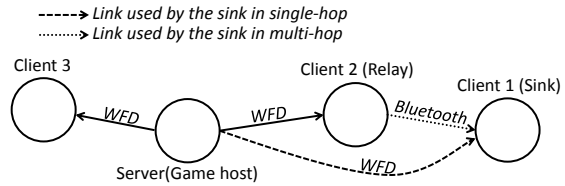


Up to 8 clients join the group gradually. The arrows represent the time at which one more client joins. As the group size becomes larger, both the transmitted and received traffic increases quadratically.

Figure 14: Network Overhead

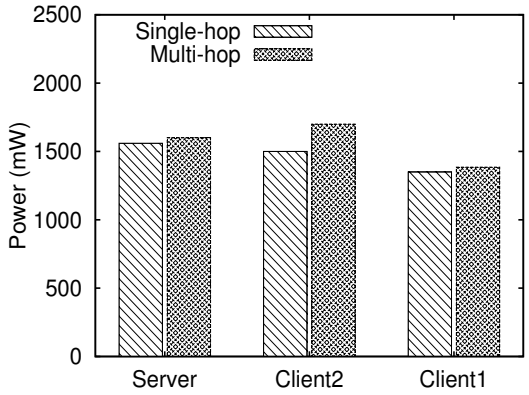
spread out linearly so that the rightmost player does not have a reliable connection with the server client. In these cases, is it possible to leverage intermediate clients as relay nodes to form a multi-hop *linked-list* topology where a node is connected to an intermediate node that connects it to the server? *GameOn* supports multi-hop networks but with some limitations. In particular, joining two Wi-Fi Direct groups (to create a multi-hop network) at the same time is not allowed, even in the latest version of Android. This joining of groups feature is an optional feature in the Wi-Fi Direct standard that has not been implemented in Android. Thus, to create a relay node for a multi-hop environment, we have to use two different networking technologies / radios. In this case, we will have to use Bluetooth together with Wi-Fi Direct with one side of the linked-list using Bluetooth and the other side using Wi-Fi Direct. However, as stated earlier, Bluetooth is not the best protocol for the scenarios *GameOn* is tackling. We re-visit these claims using a three-node scenario as shown in Figure 15.

Figure 16 shows the energy consumption when using three nodes with a star and a linked-list topology. We instrumented the *Racer* game so that it automatically looped the same track to create a repeatable trace. For each experiment, we turned off all background processes and measured the power consumption using the Monsoon power monitor. It should be noted that the absolute numbers



The arrows indicate the server information flow direction. Dotted Wi-Fi Direct link indicates poor connectivity.

Figure 15: Single-hop vs. Multi-hop Topologies



Power consumption of three nodes using different topologies. In the linked-list topology, the relay node (client 2) connects to the source node via a Wi-Fi Direct link, and is connected by the sink node via a Bluetooth link.

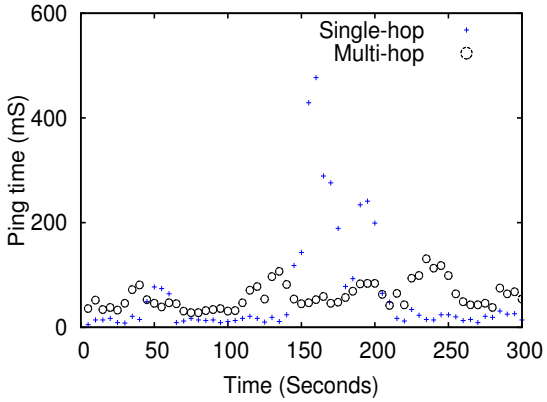
Figure 16: Power Consumption under Various Topologies

are not that interesting as they are device-specific. Instead we focus on the difference between the two topologies.

We observed that in the star topology, the server node (the one sending most of the data) consumes the most power followed by the other two nodes (client 2 and client 1). When using the linked-list topology, the host (server) and sink (client 1) nodes consume similar power to the star topology. However, the relay node (client 2) uses 13% more power in the linked-list case. This is because it has to use two radios (Wi-Fi Direct and Bluetooth) simultaneously to bridge the two sides of the relay.

We now evaluate the effectiveness of the linked-list topology at reducing latency spikes caused by nodes being too far away from each other. To do this, we placed two node (a source and a sink) two train carriages apart from each other on a public train (that was moving and picking up passengers etc.). We then placed a relay node in between the two nodes (i.e., the relay node was 1 carriage away from both the source and the sink). The source and the sink were then connected to each other using Wi-Fi Direct. The source was also connected to the relay node via Wi-Fi Direct while the sink connected to the relay node via Bluetooth. The sink then started pinging the source across both the direct Wi-Fi Direct connection as well as the multi-hop (via the relay) Bluetooth connection.

Figure 17 shows the latency results. We observe that the direct link between the source and the sink (i.e., the star topology) showed variable ping times as the distance was far and the link quality was thus affected by passenger movements etc. However, the link via the relay node showed much more predictable and stable performance. However, the ping times for the Bluetooth link are still high (yet stable) as Bluetooth is not the best protocol (as shown earlier) for this type of environment.



Client-side latencies under two topologies. A client (sink) is connected to the group owner via two different routes at the same time. One is a direct connection via a Wi-Fi Direct link. And the other one is a connection to the relay node via a Bluetooth link.

Figure 17: Latency of Various Topologies

6.3.4 *Improving Matchmaking With Time Predictions*

In this final test, we show that using predictions of how long a particular trip will last for a given individual can have big positive effects on the matchmaking performance. To perform this test, we selected only the trips that started from one of the starting train stations (called *k*) to a specific train station at our university. The starting train station was chosen because the trains start off empty there (so everyone on that train when it leaves got on at station *k*) and it was not an interchange station. I.e., everyone had to swipe their NFC cards at that station itself to get into it. It was not possible for them to enter at some other station and take another train to this station. 2) We picked a specific time (8 a.m.) and day (1st Monday of Nov.) and extracted all the commuters who entered station *k* at that time and day. 3) We then exhaustively created all possible 2, 3, 4, 5, 6 person groups that could be created from the set of people who entered that station. 4) We then computed how long each of these groups actually stayed together (i.e., the minimum co-location time until someone in that group left the train). This result represents a naive matchmaker that just selects people randomly and hopes that they will be together long enough.

Figure 18 shows the results of this test. We observe that the time the entire group was together is quite low and with a very high standard deviation (indicating that some groups were together for much less time). In addition, as the group size increased, the time spent together decreased significantly as the probability of any one person in the group leaving increased. This result shows that just randomly grouping people together can lead to bad outcomes.

However, we found that using predicted individual trip times can result in better estimates. First, we created historical buckets for each user (similar to Balan et. al [5]) that was station, day, and time specific. We then used this history to calculate, for each user, a predicted trip start time (with stdev.) for any station at any time and day. This prediction lets us increase the minimum co-location time for all group sizes as we can cluster passengers by their predicted trip times. For example, our standard deviation for any given trip time and group size dropped to a few % compared to 60-70% with the naive approach. However, this approach can lead to data sparsity issues. For example, only 3,500 of the 15,948 passengers (22%) used to generate Figure 18 had multiple trips from that station from which we could calculate a history. We plan to look at techniques to improve this yield in future work.

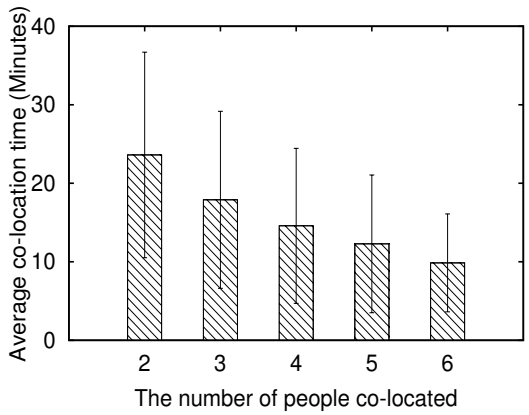


Figure 18: Group Size versus Collocation Time

6.4 Summary

In this section, we showed that *GameOn* works well in the real world with good latencies and end-user experiences with up to 4 players in a variety of bus and train environments. We then showed that the energy, CPU, ping latencies, and memory overheads of hosting a game server are minimal if the number of game clients is kept low (under 8). Next, we should that we could support multi-hop p2p methodologies in addition to he base star topology. Finally, we showed that naively predicting the co-location time can result in very sub-optimal matches.

7. DISCUSSION

The idea of commuters playing games with fellow commuters raises a number of interesting questions. In particular, why would commuters play games on their phones as they drain a lot of energy. Even though, as we showed earlier, *GameOn* itself does not have a large energy overhead, the base energy consumption of a game is already huge – on the order of few Watts in some cases! Thus playing a game for about 20 to 40 minutes will significantly reduce your phone's battery lifetime. So why would people do that?

One reason why might be that they are commuting either to work / school or back home. In both cases, there is an opportunity to charge their phone at the other side (at their work desk, classroom, home). Thus, playing a game with whatever "residual energy" is left in the phone might be okay as a recharge point is available immediately afterwards. However, we have not investigated this willingness in more depth beyond the survey presented earlier (Section 3.1.4).

Another concern raised is that multiplayer games usually result in someone losing. What happens if that person gets angry and starts looking for the person(s) they lost to? Would it be a form of "game-rage" (similar to road rage)? Identifying the people you are playing with will be hard in crowded trains where everyone is awake and using their phone. But what about on longer train journeys where a majority of commuters are sleeping? These types of social phenomena and implications need to be investigated in more details.

7.1 Limitations

The main limitations of the current *GameOn* prototype are: 1) The matchmaker, while supporting many attributes well (as far as we can tell), cannot be completely validated as we do not have data for many of the skill and player-centric attributes. 2) We have built *GameOn* to handle only the system aspects of multiplayer games. Unfortunately, we have no control over the game itself which has a larger say on user satisfaction. For example, *GameOn* can han-

dle cases where users join and leave a game in the middle (as long as the player hosting the server does not leave) and *GameOn* can handle alternate game modes such as "spectator mode" if those modes use standard networking APIs. However, *GameOn*, by itself, can do very little to make a game "fun" which is ultimately the most important criteria. And 3), the multi-hop support needs to use Bluetooth and Wi-Fi Direct for multi-hop settings (thus lowering its range to what Bluetooth supports) as Android does not currently allow multiple Wi-Fi Direct connections. Finally, the survey presented in Section 3.1.4 was conducted mostly with undergraduate students and thus may not be generalisable. In addition, our performance experiments were conducted using only two different models of phones. Thus results may vary with other phone types.

8. RELATED WORK

p2p game matchmaking: Switchboard [22] proposed techniques to predict latencies during a game play using quick pre-game measurements. Htrae [2] predicts inter-player latencies using geolocation data. Ly et al. [21] developed an approach to select the best detour route for game packets. However, these systems and techniques were targeting game consoles or devices connected to the Internet. Our goal is to use p2p networking to connect players on public transport.

Mobile p2p applications: Collaborative smartphone applications have emerged in diverse application domains such as media sharing [15] and context sensing [19][20]. Like *GameOn*, they propose several core techniques to enable in-situ collaboration among co-located smartphones. McNamara et al. [15] devised a scheme to predict remaining co-location duration for stable exchange of multimedia files. CoMon [19] proposed a resource planning mechanism to maximise benefit while achieving fairness. However, building a system for collaborative mobile gaming imposes a set of unique challenges due to the strict gaming latency and power requirements. To address these, we developed a new end-to-end system, *GameOn*, with careful attention to various system components such as network protocols, peer discovery, matchmaking, and low latency game play. Some airlines offer multi-player games among passengers during long-haul flights [35]. However, only a few limited games can be supported through wired entertainment systems embedded in passenger seats whereas *GameOn* can support commodity mobile games on smartphones without any infrastructure support in buses or trains.

Mobile p2p framework: There have been efforts to develop generic platforms to facilitate development of various mobile p2p applications [10]. For example, the well-known open source project Alljoyn [3], aims to provide a set of APIs and runtime to easily build network connections among multiple mobile devices. *GameOn* opens a new application domain of multiplayer gaming by supporting game-specific requirements that Alljoyn does not support. It will be an interesting to test if Alljoyn can work with *GameOn*. There have also been prior work to re-write binaries without source code access. RetroSkeleton [9] presents an app rewriting framework that allows developers to integrate new features into existing apps. *GameOn* did not use re-writing methods initially as we wanted to understand the challenges required to port existing games to use *GameOn*.

9. CONCLUSION

In this paper, we presented *GameOn*, a system for allowing commuters on public transportation to play multiplayer games with each other using Wi-Fi Direct as a p2p communication medium. We motivated the reasons why *GameOn* is useful (long commute times) and then described the various components of *GameOn*. Finally, we presented extensive evaluation results showing that *GameOn* works. Even though *GameOn* is a proof-of-concept idea, it has been implemented and works quite well with the modified games (videos available at **http://tinyurl.com/gameon-videos**). However, this is just step one. Our broader goal is to use *GameOn* as a platform for providing many different types of commuter friendly engagement channels. For example, we plan to extend *GameOn* to allow users who share similar interests (that are discovered through specific types of games) to meet up with each other in the physical world. We are also considering system-level support for spectator-mode; where commuters can join existing games as passive observers instead of active players. Both of these planned extensions should increase the adoption rate of *GameOn*.

10. ACKNOWLEDGEMENTS

We sincerely thank David Chu for shepherding our paper, and the anonymous reviewers for their valuable feedback. We also thank Lixuan Zhai, Xiaonan Guo, Nguyen Minh, Shriguru Nayak, Kazae Quek, Huynh Nguyen, Hai Le Gia, Kasthuri Jayarajah and Kartik Muralidharan for their helps to perform the in-bus and in-train measurements and evaluations. Finally, this work was supported partially by the Singapore Ministry of Education's Academic Research Fund Tier 2 under research grant MOE2011-T2-1001, and partially by the National Research Foundation, Prime Minister's Office, Singapore, under the IDM Futures Funding Initiative. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the granting agencies or Singapore Management University.

11. REFERENCES

[1] Aaron M. Renn, New Geography. *Commuting In New York City, 2000-2010*. http://www.newgeography.com/content/002691-commuting-new-york-city-2000-2010/, 2012. Accessed on 25th March 2015.

[2] Agarwal, S. and Lorch, J. R. Matchmaking for online games and other latency-sensitive p2p systems. *Proceedings of the ACM SIGCOMM conference on Data communication (SIGCOMM)*, 2009.

[3] The Allseen Alliance. https://allseenalliance.org/. Accessed on 25th March 2015.

[4] Bakki, N. *Commuting time of workers in Japan, 2011*. http://nbakki.hatenablog.com/entry/2014/02/20/124942/ , 2014. Accessed on 25th March 2015.

[5] Balan, R. K., Nguyen, K. X., and Jiang, L. Real-time trip information service for a large taxi fleet. *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2011.

[6] Bluetooth SIG. *Learn the Bluetooth(R) technology basics*. http://www.bluetooth.com/Pages/what-is-bluetooth-technology.aspx, 2014. Accessed on 25th March 2015.

[7] chinadaily.com.cn. *Commuting time in capital averages 97 minutes*. http://www.chinadaily.com.cn/china/2014-11/24/content_18967891.htm/, 2014. Accessed on 25th March 2015.

[8] Cox, W. *Hong Kong's Decentralizing Commuting Patterns*. http://www.newgeography.com/content/003300-hong-kong-s-decentralizing-commuting-patterns/, 2012. Accessed on 25th March 2015.

[9] Davis, B. and Chen, H. Retroskeleton: Retrofitting android apps. *Proceeding of the 11th annual international conference on Mobile systems, applications, and services (MobiSys)*, 2013.

[10] Fitzek, F. H. P. and Charaf, H. *Mobile Peer to Peer (P2P): A Tutorial Guide*. Wiley, 2009.

[11] Google. *Top Grossing Games on the Android Marketplace*. https://play.google.com/store/apps/collection/topgrossing?hl=en. Accessed on 3rd December 2014.

[12] GSMA Intelligence. *The Mobile Economy Asia Pacific 2014*. http://asiapacific.gsmamobileeconomy.com/GSMA_ME_APAC_2014.pdf, 2014. Accessed on 6th December 2014.

[13] Herbrich, R., Minka, T., and Graepel, T. Trueskill(tm): A bayesian skill rating system. *Advances in Neural Information Processing Systems 20 (NIPS)*, 2007.

[14] Khalaf, S. *Flurry Five-Year Report: Its an App World. The Web Just Lives in It*. http://www.flurry.com/bid/95723/Flurry-Five-Year-Report-It-s-an-App-World-The-Web-Just-Lives-in-It, 2013. Accessed on 6th December 2014.

[15] L. McNamara, C. M. and Capra, L. Media sharing based on colocation prediction in urban transport. *Proceedings of ACM 14th International Conference on Mobile Computing and Networking (MobiCom)*, 2008.

[16] Land Transport Authority of Singapore. http://www.lta.gov.sg. Accessed on 25th March 2015.

[17] Land Transport Authority of Singapore. *Fast Facts*. http://www.lta.gov.sg/content/ltaweb/en/public-transport/projects/circle-line/fast-facts.html. Accessed on 25th March 2015.

[18] Land Transport Authority of Singapore. *Singapore Land Transport: Statistics In Brief*. http://www.lta.gov.sg/content/ltaweb/en/publications-and-research.html. Accessed on 25th March 2015.

[19] Lee, Y., Ju, Y., Min, C., Kang, S., Hwang, I., and Song, J. Comon: cooperative ambience monitoring platform with continuity and benefit awareness. *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2012.

[20] Lee, Y., Min, C., Hwang, C., Lee, J., Hwang, I., Ju, Y., Yoo, C., Moon, M., Lee, U., and Song, J. Sociophone: Everyday face-to-face interaction monitoring platform using multi-phone sensor fusion. *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2013.

[21] Ly, C., Hsu, C.-H., and Hefeeda, M. Improving online gaming quality using detour paths. *Proceedings of the international conference on Multimedia (MM)*, 2010.

[22] Manweiler, J., Agarwal, S., Zhang, M., Choudhury, R. R., and Bahl, P. Switchboard: a matchmaking system for multiplayer mobile games. *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2011.

[23] Microsoft. *Xbox Live*. http://www.xbox.com/en-US/live.

[24] Monsoon Solutions Inc. *Monsoon Power Moniter*. http://www.msoon.com/LabEquipment/PowerMonitor/.

[25] National Post. *Travel times prove transit's a non-starter*. http://www.financialpost.com/story.html?id=0f49732a-b9d9-477d-828b-01a727147219/. Accessed on 25th March 2015.

[26] National Statistics, Republic of China (Taiwan). *2000 Taiwan Social Development Trends Survey Results*. http://eng.stat.gov.tw/ct.asp?xItem=9378&ctNode=1647, 2000. Accessed on 25th March 2015.

[27] Nicolas Gramlich. *AndEngine game engine*. https://github.com/nicolasgramlich/AndEngine. Accessed on 3rd December 2014.

[28] Nielsen. *The Asian Mobile Consumer Decoded*. http://www.nielsen.com/ph/en/insights/news/2014/asian-mobile-consumers.html, 2014. Accessed on 25th March 2015.

[29] Patro, A., Rayanchu, S., Griepentrog, M., Ma, Y., and Banerjee, S. The anatomy of a large mobile massively multiplayer online game. *Proceedings of the First ACM International Workshop on Mobile Gaming (MobiGames)*, 2012.

[30] Paul Sarbinowski. *2048 Android port*. https://github.com/uberspot/2048-android. Accessed on 3rd December 2014.

[31] Play Framework. https://www.playframework.com/. Accessed on 25th March 2015.

[32] SBS Transit. *Bus Memory Lane*. http://www.sbstransit.com.sg/about/memorylane.aspx. Accessed on 25th March 2015.

[33] Seoul Metropolitan Government. *Seoul Public Transportation*. http://english.seoul.go.kr/wp-content/uploads/2014/06/Seoul-Public-Transportation-English.pdf, 2014. Accessed on 25th March 2015.

[34] Sergii Pylypenko. *Openarena Android port site*. https://play.google.com/store/apps/details?id=ws.openarena.sdl&hl=en. Accessed on 3rd December 2014.

[35] Singapore Airlines. *Flying with us - Games and applications*. http://www.singaporeair.com/en_UK/flying-with-us/games-applicationslanding/games/. Accessed on 25th March 2015.

[36] Trade Union Congress (TUC), UK. *Commute times increase as the UK's transport system gets more crowded*. http://www.tuc.org.uk/economic-issues/labour-market-and-economic-reports/industrial-issues/transport-policy/commute-times/, 2014. Accessed on 25th March 2015.

[37] VGChartz. *Top Selling Console Games for 2014*. http://www.vgchartz.com/yearly/2014/Global/. Accessed on 3rd December 2014.

[38] Vinita Dawra Nangia, The Times of India. *Travel times prove transit's a non-starter*. http://timesofindia.indiatimes.com/life-style/Is-commute-time-taking-over-your-life/articleshow/3508745.cms/, 2008. Accessed on 25th March 2015.

[39] Wi-Fi Alliance. *Portable Wi-Fi(R) that goes with you anywhere*. http://www.wi-fi.org/discover-wi-fi/wi-fi-direct, 2014. Accessed on 25th March 2015.