ABSTRACT

RASHID, MD. AHSANUR. Fault Tolerance in Mobile Outdoor Wi-Fi Games. (Under the direction of Dr. Michael Devetsikiotis.)

Mobile games are a fast growing genre. Over 55% gamers have played games on their mobile or handheld devices. One of the newest trends in mobile games is mobile outdoor Wi-Fi games, especially exer games. Games that used to exist in gaming consoles or in real life, e.g. Pac-Man or treasure hunt, can now be played on a mobile device. These games can be played on campus or in a wireless LAN setting. However, due to various factors like unreliability of wireless internet, different kinds of interference, areas not covered by Wi-Fi, Wi-Fi blind spots etc., it is difficult to maintain a continuous synchronous connection. We propose two fault tolerant algorithms pertinent to mobile games over Wi-Fi that reduce the data loss and make communication transparent to the end user and the game developer. We also propose differentiating the data packets according to their roles. In our algorithms, important packets are buffered and re-transmitted until they are received by the destination. In our first algorithm, data packets are buffered at the local device so that data loss is minimized in case the source is not connected to Wi-Fi. However the loss would still be high in case the destination is not connected to Wi-Fi. Our second algorithm is an extension to the first algorithm in which if the source is connected to Wi-Fi and destination is not connected to Wi-Fi, the source can send the important data packets to the central server (Custody Handover) and the destination will receive those data packets from the central server whenever it connects to Wi-Fi. Using both simulation and implementation, we show that our algorithms have a lower data packet loss as compared to the regular (no algorithm) approach.
Fault Tolerance in Mobile Outdoor Wi-Fi Games

by

Md. Ahsanur Rashid

A thesis submitted to the Graduate Faculty of North Carolina State University in partial fulfillment of the requirements for the Degree of Master of Science

Computer Science

Raleigh, North Carolina

2012

APPROVED BY:

Dr. Laurie Williams

Dr. David L. Roberts

Dr. Michael Devetsikiotis
Chair of Advisory Committee
DEDICATION

To Ma, Baba, Apu, Bhaiya.
Md. Ahsanur Rashid was born on October 22, 1987 in Dhaka, Bangladesh. He holds a Bachelor's degree in Computer Science and Engineering, awarded in 2009, by Bangladesh University of Engineering & Technology (BUET), Dhaka, Bangladesh. He has been a Masters student at North Carolina State University, in Department of Computer Science, since August 2009.
ACKNOWLEDGEMENTS

I thank my parents and my sister & brother-in-law for their unwavering support and rock-steady faith in me. I would not have made it without their constant encouragement and motivation.

I am deeply grateful to my adviser Dr. Michael Devetsikiotis for his supervision. In addition to his many useful suggestions and constant guidance, I cherish the trust he placed in me, and the freedom I was allowed during the course of this work. I would like to thank Dr. Laurie Williams and Dr. David L. Roberts for graciously agreeing to be in the thesis committee and providing guidance from time to time.

I would like to thank my friends for life - Ratan, Mahtab, Faisal, Mamnun, Sajib, Raihan, Tawseef, Towhid, Atiq, Tonmoy, Tanzir, Omees, Tasnuva, Jinan and Nazia. Thanks for being with me through all the ups and downs and twists and turns.

Thanks Shalini Chauhan for being the constant source of help, motivation, guidance and company throughout my stay at Raleigh. I truly enjoyed working on coursework and research with you. Thanks Mohammad Shamsul Araf in for your constant help and motivation. Thanks Mahfuzur R. Chy for your tireless help with my statistical problems despite your busy schedule.

I realize that this work would not have completed without all your contributions, and I am glad to have received this chance of expressing my gratitude. Thanks a lot!
# TABLE OF CONTENTS

List of Figures ................................................. vii

Chapter 1 Introduction ........................................ 1
  1.1 Networked Games and Network Unavailability ......... 1
  1.2 Technical Challenges and Motivation ................. 3
  1.3 Our Proposition ....................................... 7
  1.4 Thesis Structure .................................... 7

Chapter 2 Related work ........................................ 9
  2.1 Delay Tolerant Networks ............................... 9
  2.2 Fault Tolerant Middleware for Mobile Computing .... 12

Chapter 3 Proposal ............................................. 14
  3.1 Assumptions ......................................... 14
  3.2 Packet Types ........................................ 15
  3.3 Terminologies Used ................................. 16
  3.4 Flow Charts .......................................... 17
  3.5 Algorithms ......................................... 20
  3.6 Description ......................................... 20

Chapter 4 Design and Simulation .............................. 22
  4.1 Simulation Design .................................... 23
  4.2 Experimental Setup .................................. 24
  4.3 Simulation for Experimental Level .................... 26

Chapter 5 Data Collection and Analysis ....................... 27
  5.1 Wi-Fi Distribution 1 .................................. 27
  5.2 Wi-Fi Distribution 2 .................................. 30
  5.3 Wi-Fi Distribution 3 .................................. 33
  5.4 Experimental Data .................................... 36
  5.5 Simulation Data for Experimental Level .............. 37
  5.6 Simulation Data vs. Experimental Data for Experimental Level .................. 38

Chapter 6 Conclusion ........................................... 42
  6.1 Observation .......................................... 42
  6.2 Other Approaches .................................... 43
## Chapter 7 Summary and Future work

7.1 Summary ................................................. 44
7.2 Future work .............................................. 45

## References ................................................. 46

## Appendices ................................................... 50
Appendix A  No DTN Approach ................................. 51
  A.1 Algorithm .............................................. 51
  A.2 Description ........................................... 51
  A.3 Flow Chart ............................................ 52
Appendix B  Experimental Server Setup ....................... 53
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1</td>
<td>Peer to Peer network architecture in a mobile environment [34]</td>
<td>2</td>
</tr>
<tr>
<td>Figure 1.2</td>
<td>Client Server network architecture in a mobile environment [34]</td>
<td>3</td>
</tr>
<tr>
<td>Figure 1.3</td>
<td>Smartphone Penetration in the US [19]</td>
<td>4</td>
</tr>
<tr>
<td>Figure 1.4</td>
<td>Top 50 Apps by Category (Apple App Store) [4]</td>
<td>5</td>
</tr>
<tr>
<td>Figure 2.1</td>
<td>Store and Forward Message Passing [6]</td>
<td>10</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>The Bundle layer on top of other lower region layers [6]</td>
<td>10</td>
</tr>
<tr>
<td>Figure 2.3</td>
<td>Different nodes in a DTN network [6]</td>
<td>11</td>
</tr>
<tr>
<td>Figure 2.4</td>
<td>Shared Tuple Space in Linda [35]</td>
<td>13</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>Proposed Tolerance Layer (TL)</td>
<td>15</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>Structure of Data Packet</td>
<td>15</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>Flowchart for Basic DTN</td>
<td>18</td>
</tr>
<tr>
<td>Figure 3.4</td>
<td>Flowchart for DTN with CH</td>
<td>19</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>Pac-Man Championship 2007 level [16]</td>
<td>22</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>Pac-Man level used for simulation (based on Pac-Man Championship 2007 level)</td>
<td>23</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>Pac-Man level used for simulation (numbered blocks used for computing where a player is and where it can go next etc.)</td>
<td>24</td>
</tr>
<tr>
<td>Figure 4.4</td>
<td>Android Implementation of Pac-Man-like level-based app, sending and receiving data</td>
<td>25</td>
</tr>
<tr>
<td>Figure 4.5</td>
<td>Level used in experimental data collection (red blocks do not have Wi-Fi coverage, blue blocks have Wi-Fi coverage) [map ©Microsoft, 2011]</td>
<td>26</td>
</tr>
<tr>
<td>Figure 5.1</td>
<td>Wi-Fi Distribution 1 (red blocks do not have Wi-Fi)</td>
<td>27</td>
</tr>
<tr>
<td>Figure 5.2</td>
<td>Packet Loss (%) vs. Local Buffer Size for Wi-Fi distribution 1</td>
<td>28</td>
</tr>
<tr>
<td>Figure 5.3</td>
<td>Packet Loss (%) vs. Server Buffer Size (across different Local Buffer Size) for Wi-Fi distribution 1</td>
<td>29</td>
</tr>
<tr>
<td>Figure 5.4</td>
<td>Wi-Fi Distribution 2 (red blocks do not have Wi-Fi)</td>
<td>30</td>
</tr>
<tr>
<td>Figure 5.5</td>
<td>Packet Loss (%) vs. Local Buffer Size for Wi-Fi distribution 2</td>
<td>31</td>
</tr>
<tr>
<td>Figure 5.6</td>
<td>Packet Loss (%) vs. Server Buffer Size (across different Local Buffer Size) for Wi-Fi distribution 2</td>
<td>32</td>
</tr>
<tr>
<td>Figure 5.7</td>
<td>Wi-Fi Distribution 3 (red blocks do not have Wi-Fi)</td>
<td>33</td>
</tr>
<tr>
<td>Figure 5.8</td>
<td>Packet Loss (%) vs. Local Buffer Size for Wi-Fi distribution 3</td>
<td>34</td>
</tr>
<tr>
<td>Figure 5.9</td>
<td>Packet Loss (%) vs. Server Buffer Size (across different Local Buffer Size) for Wi-Fi distribution 3</td>
<td>35</td>
</tr>
</tbody>
</table>
Figure 5.10 Wi-Fi Distribution 4 (red blocks do not have Wi-Fi, blue blocks have Wi-Fi) [map ©Microsoft, 2011] ........................................ 36

Figure 5.11 Packet Loss (%) vs. Local Buffer Size for Experimental Level . . . 37

Figure 5.12 Packet Loss (%) vs. Local Buffer Size for Simulation on Wi-Fi
distribution 4 Experimental Level ........................................ 38

Figure 5.13 Packet Loss (%) vs. Local Buffer Size for Simulation and Experimental Data on Experimental Level for Basic DTN approach . . . 39

Figure 5.14 Packet Loss (%) vs. Local Buffer Size for Simulation and Experimental Data on Experiment Level for No DTN approach . . . 40

Figure A.1 Flowchart for No DTN ........................................ 52
Chapter 1

Introduction

In this thesis we address the problem of network unavailability (Wi-Fi blind spots, unavailable Wi-Fi coverage, interference etc.) in mobile outdoor Wi-Fi games (human Pac-Man, treasure hunt etc.). Our approach relies on algorithm for a new fault tolerant layer between the application and the transport layer. We design and implement a basic Pac-Man-like level-based app using this strategy. We test the application against a number of situations with limited network availability (both simulation and experimental scenarios) and present the results.

1.1 Networked Games and Network Unavailability

Tristan describes networked games as part of the genre of applications known as Networked Virtual Environment (NVE) [34]. Networked Virtual Environments can be defined as “a software system in which multiple users interacts with each other in real-time, even though those users may be located around the world” [45, 34]. The main distinguishing features of NVEs mentioned by Singhal [45] include:

- “a shared sense of space”
- “a shared sense of presence”
- “a shared sense of time”
- “a way to communicate”
NVEs generally have three parts:

- a database that has the representation of the shared virtual world
- a communication system
- devices to show the shared virtual world

Users (Players) can interact with the virtual world and generate events as a result.

There are two kinds of network topologies that NVEs use either separately or in combination: client-server topology (e.g. MASSIVE) and peer-to-peer (e.g. DIVE).

In today's mobile environment, the two kinds of topologies can be visualized as:

![Peer to Peer network architecture in a mobile environment](image-url)
However, due to unreliable nature of wireless internet, Wi-Fi blind spots, wireless interference etc. it is almost impossible to maintain a synchronous connection. Due to all these factors, there is a need for a middleware that makes the communication between players transparent for both the developer and the end user. This is especially true for games using peer to peer network architecture.

1.2 Technical Challenges and Motivation

In today's world, mobile gaming is a fast growing segment. According to the Entertainment Software Association (ESA), in the year 2011, 55% of gamers played games on their
mobile or handheld devices [8]. This is an increase from 42% in the year 2010 and 20% in the year 2002 [38]. Consumers have spent over $25 billion in the 2010 for games and accessories [7]. The mobile gaming industry was worth $33 billion in 2010 and is expected to reach $54 billion by the year 2015. 70-80% of all mobile downloads are games. Angry Birds has been downloaded over 140 million times [12]. All these mean that there has been a huge boom in the mobile gaming industry and it is poised to grow even further in the coming years.

The penetration rate of smartphones in the US market in 2011 was about 44% compared to only 18% in the year 2009 [19]. In Apple app store, a break out of the top 50 apps (top 25 paid and top 25 free apps) for both the iPhone/iPod Touch and the iPad reveals a large proportion of gaming apps [4]. In the Android market, the entertainment section (which includes games) has the most number of total apps and free apps [13].

Figure 1.3: Smartphone Penetration in the US [19]
The Journal of American Medical Association (JAMA) has recently published an article emphasizing on health focused games (including mobile ones) [11]. Zamzee, a startup, is using a combination of mobile activity monitoring and an online reward system to get youngsters to be more active [11]. Insurance provider Humana has multiple games for children that focus on outdoor activities [15]. Diane mentions about different health games for physical activity including PCGamerBike, NeoRacer, EyeToy, Gamercize, Zyked, Dancetown and Cybex TRAZER [46]. Kiili et al. have designed two multiplayer exer games. The first, Tug of War, can be played between two teams each consisting of one to five players. The second, Diamond Hunter, is designed for one to four players [37].

Now there is a whole new generation of location based multi-player games for users to play [14]. Location-based mobile games use urban environments as game virtual environment and are played with on mobile platforms having localization and wireless internet services. There is a German game that is called FastFoot-Challenge where there is a runner and multiple chaser in real life [17]. Then there is The Game where players have to go treasure hunting in real world for 24 hours. iSpy is another location based scavenger hunt game that uses photography based on the popular game I spy with my little eye... [17]. Urban defender is another location based multi-player game where users roam about in
an urban area and own buildings. The target is to own as many buildings as possible [21]. SCVNGR is a game all about going places, doing challenges and earning points. Instead of riddles for guiding to a place, SCVNGR concentrates on completing challenges at a particular location [22]. Mashables list of the top 5 location based services include Yelp for mobile, Neer, Loopt, SCVNGR and Foursquare [20]. These services allow users to check-in to places, earn badges and share information with friends and many other things.

The technologies used in outdoor multiplayer location aware games are wireless, to allow players to walk around freely. IEEE 802.11, also known as wireless internet, is the most common protocol used [44]. In collaborative team based location aware games, team members communicate with each other using wireless internet [44].

Most multi player mobile games need to send continuous data to other players or the server and as such using Wi-Fi is the preferred way. There can be a variety of reasons for a sudden disruption in the Wi-Fi connection. Wi-Fi is required to accept interference, including interference that causes disruption of service [23, 3]. There are many electronic items (microwaves, cordless phones, Bluetooth networks etc.) that use the same frequency range as that of Wi-Fi causing interference [1]. Wi-Fi networks have a limited range - about 45 meters indoors and about 90 meters outdoors [23]. Physical barriers can cause the range to be even more limited. Signal strength may not be sufficiently strong everywhere [42]. Again, there may be interference due to multiple networks in overlapping areas [2]. Intervention of closed Wi-Fi points with open Wi-Fi points might prevent users from accessing the open points [23, 3].

If there is network degradation or unavailability it could have an impact on mobile outdoor Wi-Fi games. For example, in a mobile version of the Human Pac-Man [26, 50], both the ghost(s) and the Pac-Man needs to share their coordinates continuously along with the score. In a treasure hunt scenario [47, 49], players not only need to share their coordinates, but also need to update points, have team chats, get instructions etc. Some of these information are transmitted continuously (e.g. GPS coordinates) and can be dropped if there is no network. However, in case of instructions or group messages, the information cannot be dropped.

To sum up, there is an amazing opportunity for mobile outdoor Wi-Fi games. How-
ever, to make sure that the user experience with these games do not degrade and that important communications do not get lost due to unreliable wireless network, there is the need for a middleware, that is delay tolerant and to a certain extent guarantees delivery of important communication between players.

1.3 Our Proposition

We propose a Tolerance Layer (TL) between the application and transport layers. This layer will make the data communication transparent for the applications. As such, apps do not need to be concerned about temporary network failure or unavailability etc. The Tolerance Layer will have all the packets of specific types that are temporarily undeliverable (either because the destination or the source is not within a network). If it is unable to deliver the packets it buffers the Data Packets until it is able to send them. Alternately, it can also hand over custody of certain Data Packets to a Central Server (CS) as soon as it gets connected to the Wi-Fi network. The Central Server is then responsible for handing over the packets to the destination. This is called Custody Handover (CH)

In this thesis, we developed a basic Pac-Man-like level-based app on the Android platform. We then tested it using different Wi-Fi distribution scenario.

The TL with CH provides the ability for reduced Packet Loss in comparison to the TL without CH (Basic). My thesis is to design a Tolerance Layer that reduces packet loss and make the whole communication paradigm transparent to the user/developer.

1.4 Thesis Structure

This thesis is organized as follows. Chapter 2 presents an overview of related work, discussing Delay Tolerance Networks and Fault Tolerant Middleware for Mobile Computing. Chapter 3 details our solution and related algorithms. Chapter 4 presents the design and simulation of a basic Pac-Man-like level-based app on the Android platform. Chapter 5 discusses the analysis of collected data. Finally, Chapter 6 provides a summary and
gives pointers for future works.
Chapter 2

Related work

To the best of our knowledge, there has been no research on delay tolerant architecture for mobile outdoor Wi-Fi games. There has also been some research on Delay Tolerant Network (DTN) architectures and fault tolerant middleware for mobile computing. These are described in details below.

2.1 Delay Tolerant Networks

Existing internet protocols are built on some basic assumptions which include:

- that an end-to-end path between source and destination exists for the duration of a communication session [5]
- that end-to-end loss is relatively small [5]
- that applications need not worry about communication performance [5]

However, these assumptions do not hold true all the time. Especially in sensor networks, inter-planetary networks or military networks. As such, the Internet Engineering Task Force (IETF) has worked on architecture for Delay Tolerant Network (DTN). DTN networks differ from regular networks: connectivity is not always persistent, there can be long (variable) delays before there is a response back, data rate is not symmetric, error rates are usually higher [6].
DTNs solve the above mentioned problems by using Store and Forward message switching. Messages are moved from node to node along the Source-Destination path until they reach their destination [6].

DTNs use a new protocol layer - Bundle Layer- to implement Store and Forward message switching. This new layer is added on top of the lower region specific layers [6].

Figure 2.1: Store and Forward Message Passing [6]

Figure 2.2: The Bundle layer on top of other lower region layers [6]
Bundles contain:

- user data from the source application
- control data pertaining to usage of the data at the destination application
- bundle header [6]

Bundles can be arbitrarily long. The bundle layer is not very conversational in nature, given the fact that connectivity is not very persistent [6].

In a DTN, there are host, router or gateway nodes- all of which are entities with the bundle layer but differentiated based on their activity [6].

![Different nodes in a DTN network](image)

Using this architecture as a base, there have different approaches to solve the problems assumed in a DTN.

The KioskNet project at University of Waterloo was developed with focus on the rural areas. It used vehicles to ferry data to and from the rural areas [32]. At KTH Royal Institute of Technology, peoples phone (known as Byte Walla) has been used as
the medium for data interchange [41].

2.2 Fault Tolerant Middleware for Mobile Computing

Fault tolerance has been an important factor in the adoption of middleware in both traditional and mobile distributed systems. Connectivity has been an important aspect of fault tolerance being investigated by middleware researchers. Mascolo et al. has described different research approach to fault tolerant middleware for mobile computing [39]. In ALICE [33], hand held devices are used to support client-server architecture in nomadic conditions. IIOP, essential for CORBA, has been adapted for mobile environment in DOLMEN project [18], using buffering and acknowledgements to counter unstable wireless networks. Message based middleware JMS (Java Messaging Server) [10] supports both point to point messaging as well the publish-subscribe model. Here, a device can communicate with a single other device or subscribe to a topic & get notified of all the messages for that topic.

Mobile and wireless environments favor decoupled and opportunistic approach to communication. There has been some research along this line using queues [36] or events [48]. In Linda (a language for parallel programming), shared tuple space is the basis of its communication model [30]. The shared tuple space is like a big storage area where tuples are stored. Tuples are a data structure. Producers produce tuples and put them in the shared space. Consumers get the tuples and process them. A lot of research has been done on tuple based middleware, e.g. Tspaces [51], JavaSpaces [9], Lime [40] etc.
Paolo et al. have worked on a Mobile Agent based middleware for mobile computing [24]. In this approach, there is no need for continuous network connectivity. It needs to be long enough for the mobile node to be able to inject the Mobile Agent onto the fixed network. At the end of processing, the Mobile Agent will send back the necessary data to the mobile node. If there is no available connection, it will do so once the connection is restored.

Chien-Liang et al., have worked on Limone - a coordination middleware based on the Linda model [27]. Here, physical mobile hosts have virtual agents with their own acquaintance list. Coordination activities are restricted to those agents on the acquaintance list. Bruno et al., have worked on a multi-client multi-server application middleware that provides transparent communication for different applications [43]. Applications have to register with the middleware at both the client and server side. Communication faults (loss of connectivity, sending data etc.) are handled by the middleware. There are also multiple servers that can act as gateways for the mobile clients.
Chapter 3

Proposal

3.1 Assumptions

In the mobile outdoor Wi-Fi games scenario that we envision, we make a few assumptions:

- Low amount of continuous data is being sent/received by the players
  - Packet size of 1kb or less
  - Data may include GPS coordinates, score, instruction message, chat message etc.

- Disconnections can happen many times
  - Depends on the pattern of the Wi-Fi and no Wi-Fi regions
  - In simulation, we got up to 95% disconnections for our scenarios

- Players are continuously on the move

We propose an extra layer between the Application and the Network layers. We call this layer the Tolerance Layer. Tolerance Layer will store data sent from the application and send it whenever it connects to the Wi-Fi. Depending on the algorithm, it either stores the data until it can send it to the destination or sends the data to the Central Server if the destination cannot be connected with.
3.2 Packet Types

We can define Packet Types as:

- (type 1) Discard after initial try (e.g. GPS coordinates)
- (type 2) Discard after initial retry (e.g. Score)
- (type 3) Keep until sent or buffer overflow (e.g. Game Instruction, Group Chat etc.)
3.3 Terminologies Used

- **Server (or Central Server)**
  - Server or Central Server is a device with significantly better computing power (and significantly higher memory availability) than a mobile device (e.g. a desktop computer or laptop). A server has a reliable internet connection (wired or wireless). It also has a Database Server for storing game and player specific information etc.

- **Local Buffer Length**
  - The length of the buffer used at the device for buffering packets of type 2 or 3. There are separate buffers for type 2 and type 3. Buffer length 2 means that 2 Data Packets can be stored in the buffer.

- **Server Buffer Length**
  - The length of the buffer used at the server for buffering packets of types 3. There are separate buffers for each player. Buffer length 2 means that 2 Data Packets can be stored in the buffer.

- **Custody Handover (CH)**
  - Custody Handover means handing over the custody of packets meant for a device to the Central Server. This happens for DTN w CH algorithm when the source has Wi-Fi and the destination doesn’t. Whenever the destination reconnects to the Wi-Fi, it queries the Central Server and retrieves the packets.

- **Packet Loss**
  - Packets can be of type 1, 2 or 3. Packets with type 3 are considered as important packets. Whenever an important packet is discarded, its a packet loss. A packet might need to be discarded if the buffer for that type of packet is full and there is a new packet that needs to be put in the buffer. The packet to be discarded can be any packet.
3.4 Flow Charts

The flow charts for both the Basic DTN and the DTN with CH approach are given on the next page(s).
Figure 3.3: Flowchart for Basic DTN
Figure 3.4: Flowchart for DTN with CH
3.5 Algorithms

Algorithm (Basic DTN)

1. Send data directly
2. If it fails and packet type is 1, discard and wait for next packet
3. For packet types 2 and 3, retry after specific time interval or when there is an Wi-Fi
4. If it fails and packet type=2, discard and wait for next packet
5. Store all the remaining packets and try to send them (these are type 3 packets that must be sent) whenever there is a connection to the destination

Algorithm (DTN with CH)

1. Send data directly
2. If it fails and packet type is 1, discard and wait for next Data Packet
3. If there is Wi-Fi, send data packet of type=3 to server
4. Retry whenever there is a Wi-Fi to connect or after specific interval
5. If it fails and packet type is 2, discard and wait for next Data Packet
6. Store all the remaining packets (type 3) and try to send them to server whenever there is Wi-Fi (Custody Handover)

3.6 Description

The whole mechanism can be divided into a few parts:

Sending
The sending part will try to immediately send whenever there is a packet to be sent. If the packet is sent successfully, it will wait for the next packet. If it is not successful,
the packet is discarded if its type=1, or saved in a buffer for retry if its type=2 or 3. Now, the sending part will wait till it connects to a Wi-Fi again. If it fails to send again, packets of type 2 are discarded. Type 3 packets are never discarded (unless there is buffer overflow) and retried whenever there is Wi-Fi connection. If the number of stored packets in the buffer crosses a specific limit, the older packets are discarded.

**Receiving**

The receiving part always waits for incoming data. Whenever it receives any data, it sends the data to the app/game.

**Server Updating**

Whenever the device reconnects to Wi-Fi, the Server Updating part updates the server about its new IP address. The server updates the layer with the new IP addresses of the other devices and with any stored data packet.
Chapter 4

Design and Simulation

We designed and ran simulation for the Basic DTN algorithm and DTN with CH algorithm, as well as the No DTN algorithm. Our simulation was based on the Pac-Man game. The level that we used resembles the level used for Pac-Man Championship 2007 [28]. We used multiple Wi-Fi distribution scenarios. We had four players (1 Pac-Man and 3 ghosts). We used Java to write the simulation program.

![Figure 4.1: Pac-Man Championship 2007 level [16]](image)

Initially, each player is randomly assigned a block from the ones that have Wi-Fi. At each step of the simulation, every players adjacent blocks are computed and one of them is randomly selected as the next block for the player. Similarly, for data packet types, it is always randomly chosen. Discarding packets of type 1 and 2 increases the packets discarded count, while discarding packets of type 3 increases the packets lost count.
In the figure above, we show a representation of the Pac-Man level. In chapter 5, we show more detailed Pac-Man level with Wi-Fi and no Wi-Fi blocks.

4.1 Simulation Design

There are three algorithms that we simulate here. The No DTN algorithm basically discards data whenever there is no connection with the destination. This algorithm is explained further in Appendix A. The other two algorithms Basic DTN algorithm and DTN w CH algorithm have been described in chapter 3.

At the start of the simulation, starting positions of each of the four players are chosen randomly among the Wi-Fi enabled blocks. At each step of the simulation, the next block for a player is randomly chosen from among all the adjacent blocks of the players current position. Also, each player sends Data Packets to all the other players. In a single simulation, each of the four players sends a total of 1200 packets to each of the other 3 players. This simulation was repeated 2000 times.

For the No DTN simulation, there are no buffers. For Basic DTN, simulation the
buffer size used was 2, 5, 10 and 20. For DTN w CH, the local buffer size used was 2, 5, 10 and 20. For server buffer size 2,5,10 and 20 were used. For the 3rd Wi-Fi scenario, both the local and server buffer size used was 2, 5, 10, 20, 50, 100, 200, 300 and 400.

Figure 4.3: Pac-Man level used for simulation (numbered blocks used for computing where a player is and where it can go next etc.)

In the figure above, the blue boxes are the walls or boundaries or buildings that players cannot go into. The white numbered blocks are the place where players can go. In chapter 5, we show more detailed Pac-Man level with Wi-Fi and no Wi-Fi blocks.

### 4.2 Experimental Setup

We also collected experimental data using android phones. There are 2 players - a Pac-Man and a ghost in our human Pac-Man game type scenario. We fixed a level on campus for this purpose. We ran both the No DTN and Basic DTN algorithms here. Both are similar to the computer simulation. In this case, since the IP address of a player might change when it reconnects to the Wi-Fi again, a database table is maintained at the central server which keeps track of the different IP addresses of the players. Whenever a player comes to a Wi-Fi enabled block from a non-Wi-Fi enabled block, it queries the
central server providing its own status/role and IP address. The central server sends back a list of all the IP addresses of the other players. We ran both the algorithms 15-25 times each for each approach and in case of Basic DTN algorithm, for local buffer lengths of 5, 10 and 20.

Figure 4.4: Android Implementation of Pac-Man-like level-based app, sending and receiving data
Figure 4.5: Level used in experimental data collection (red blocks do not have Wi-Fi coverage, blue blocks have Wi-Fi coverage) [map ©Microsoft, 2011]

4.3 Simulation for Experimental Level

Using a level similar to the experimental setup, we ran a computer simulation. We had two players (one Pac-Man and one ghost). We ran the simulation for 2000 times.
Chapter 5

Data Collection and Analysis

5.1 Wi-Fi Distribution 1

Figure 5.1: Wi-Fi Distribution 1 (red blocks do not have Wi-Fi)
In Fig 5.2, we notice that both the Basic DTN and the DTN w CH approach have a lower packet loss than the No DTN approach. Also, as the size of the local buffer increases, the packet loss decreases for both Basic DTN and DTN w CH approach.

We also run simulation to compute the packet loss for different server buffer size while keeping the local buffer size the same.
In Fig 5.3, we notice that the packet loss decreases as we increase the local buffer size. Also, as the server buffer size increases, the packet loss decreases.
5.2 Wi-Fi Distribution 2

![Figure 5.4: Wi-Fi Distribution 2 (red blocks do not have Wi-Fi)](image)

30
In Fig 5.5, we notice that both the Basic DTN and the DTN w CH approach have a lower packet loss than the No DTN approach. Also, as the size of the local buffer increases, the packet loss decreases for both Basic DTN and DTN w CH approach.

We also run simulation to compute the packet loss for different server buffer size while keeping the local buffer size the same.
Figure 5.6: Packet Loss (%) vs. Server Buffer Size (across different Local Buffer Size) for Wi-Fi distribution 2

In Fig 5.6, we notice that the packet loss decreases as we increase the local buffer size. Also, as the server buffer size increases, the packet loss decreases.
5.3 Wi-Fi Distribution 3

![Figure 5.7: Wi-Fi Distribution 3 (red blocks do not have Wi-Fi)](image-url)

---

Figure 5.7: Wi-Fi Distribution 3 (red blocks do not have Wi-Fi)
In Fig 5.8, we notice that both the Basic DTN and the DTN w CH approach have a lower packet loss than the No DTN approach. Also, as the size of the local buffer increases, the packet loss decreases for both Basic DTN and DTN w CH approach.

We also run simulation to compute the packet loss for different server buffer size while keeping the local buffer size the same.

Figure 5.8: Packet Loss (%) vs. Local Buffer Size for Wi-Fi distribution 3
In Fig 5.9, we notice that the packet loss decreases as we increase the local buffer size.
size. Also, as the server buffer size increases, the packet loss decreases.

5.4 Experimental Data

Figure 5.10: Wi-Fi Distribution 4 (red blocks do not have Wi-Fi, blue blocks have Wi-Fi) [map ©Microsoft, 2011]
Figure 5.11: Packet Loss (%) vs. Local Buffer Size for Experimental Level

In Fig 5.11, we notice that the Basic DTN approach has a lower packet loss than the No DTN approach. Also, as the size of the local buffer increases, the packet loss decreases for the Basic DTN approach.

5.5 Simulation Data for Experimental Level

We have also done a simulation for the same level in order to compare against our experimental data. From our simulation we get the following data:
Figure 5.12: Packet Loss (%) vs. Local Buffer Size for Simulation on Wi-Fi distribution

4 Experimental Level

Fig 5.12 is comparable to what we got for our experimental data in Fig 5.11.

5.6 Simulation Data vs. Experimental Data for Experimental Level

Comparing specific approaches of the simulation and experimental data, we get the following graphs:
Here we can see that the experimental data is a little above (has more packet loss) the simulation data for both buffer sizes 5 and 10. For buffer size 20, they are almost similar. The main reason behind this difference is that in real life we have more interference, Wi-Fi is unreliable, devices don’t work seamlessly etc. All these things have an adverse effect on the packet loss especially when the buffer size is small. With increasing buffer size, data packets that we cannot send are stored with more ease for future retry.

For buffer size 5, we see that there is some difference between the plots for the simulation and experimental data. The difference decreases for buffer size 10. Now for buffer size 20, simulation data loss (0.005) is more than experimental data loss (0.0). This can be explained: 0.005 loss means 5 packets lost per thousand iterations. We only did 15 iterations. As such, while experimental packet loss seems to be less than simulation
packet loss (for buffer size of 20), for a bigger data set we could have shown that they are more or less similar.

Figure 5.14: Packet Loss (%) vs. Local Buffer Size for Simulation and Experimental Data on Experiment Level for No DTN approach

For the No DTN approach, we also get a considerable difference between the simulation and experimental data. This is because, similar to the Basic DTN approach, there is interference in real life, Wi-Fi is unreliable, devices might not connect properly, Wi-Fi signal fades out etc. Also, the difference is a somewhat larger than what we got for the Basic DTN approach. This is because; in No DTN approach we do not have any buffers. As such, at any time, if Wi-Fi is not available then packet losses happen.

Based on both the experimental data and simulation data discussed and compared
above, we can say that, the new approach works quite well in reducing packet loss for mobile outdoor Wi-Fi games.
Chapter 6

Conclusion

6.1 Observation

From our experiments in Chapter 4 and 5, we can make the following observations:

- Our approach works worst when there are a lot of non-Wi-Fi blocks especially continuous non-Wi-Fi blocks. This is true for low local buffer size and any amount of server buffer size.

- We have to increase the size of the local buffer to quite high (in comparison to other scenarios) for the packet loss percentage to come down significantly.

- In most other scenarios, our approach works well with relatively low amount of local buffer size and/or server buffer size.

- Local buffer size has a high effect on decreasing packet loss percentage. However, server buffer size’s effect on decreasing packet loss percentage is not as significant.

- Our approach is highly scalable due to the above factor. We only need to have some storage capacity in each user’s device. The central server needs limited storage capacity to facilitate our DTN w CH approach.

- This approach uses existing network architecture and hence there’s no need for any overhead for extra routing or ad-hoc protocols.
6.2 Other Approaches

There are two other approaches that we consider:

- Using Ad-hoc communication instead of internet:
  Using ad-hoc communication instead of using internet has its merits and demerits. While, this approach doesn’t depend on internet, there is more overhead for ad-hoc communication. We need to maintain a routing table. The devices must be in a specific range for ad-hoc communication to work. If the devices (sender and receiver) are far and there’s no other device in between, the message can’t be sent.

- Buffering on different user devices instead of a central server:
  Buffering on user devices instead of a central server is another approach that we considered. Its a nice approach because this allows us to avoid relying on a central server (for the DTN w CH approach). However, there are a number of demerits for this approach: User devices will need more storage to store data packets from different other players, Since, the data storage for each device is really distributed, there will be lots of communication overhead and delays, packets may be out of synch etc. Some packets can even get lost if the storing device fails to meet the receiver in a timely manner.
Chapter 7

Summary and Future work

7.1 Summary

In this thesis:

- We proposed a solution for network unavailability for mobile outdoor Wi-Fi games based on the Delay Tolerant Network (DTN) architecture.

- We proposed a priority based Store and Forward algorithm especially suited for mobile outdoor Wi-Fi games that reduces packet loss.

- Our proposal included two different algorithms: the Basic DTN algorithm and the DTN with CH algorithm. We showed that the DTN with CH algorithm provided a better packet loss reduction than the Basic DTN algorithm.

- We tested the algorithms (including a No DTN algorithm, described in Appendix A, for the sake of comparison) using simulation in a Pac-Man-like level-based app. We also collected experimental data for the algorithms. We showed that for both simulation and experimental setup, our algorithms provided a much reduced packet loss in comparison to the No DTN algorithm.

- We also compared simulation and experimental data for the same experimental level and explained that while simulation data is shows lower packet loss than experimental data, this is expected due to real life interference, device issues etc.
7.2 Future work

There is scope for improving on this proposition. If there are multiple data packets of similar types, then the data can be aggregated into one packet, whenever possible. For example, if a data packet has geo-coordinates and another one has some text message, these two packets can be aggregated into one. Furthermore, there can be fragmentation. We can send all the data at the same time. Or, send it after fragmentation. Fragmenting will ensure faster and better delivery of data in a mobile environment. We have ignored the importance of authentication in this proposition. However, in real life, authentication is a major issue and as such should be addressed in the future. And, we can also add acknowledgements to ensure end-to-end data delivery.
REFERENCES


[15] Our Solutions Checkout how we can help you play your way to better health, December 2011.


Appendix A

No DTN Approach

A.1 Algorithm

1. Wait for data to send
2. Try to send data directly
3. On failure, discard data and increment appropriate count

A.2 Description

The whole mechanism can be divided into a few parts:

**Sending**
The sending part will try to immediately send whenever there is a packet to be sent. If the packet is sent successfully, then it will wait for the next packet. If its not successful, the packet will be discarded and depending on the type either the packet discarded count (type 1, 2) or packet lost count (type 3) will increase.

**Receiving**
The receiving part always waits for incoming data. Whenever it receives any data, it sends the data to the game/app.
Server Updating
Whenever the device reconnects to the Wi-Fi, it connects with the central server, sends its IP address and retrieves the IP addresses of the other players.

A.3 Flow Chart

![Flowchart for No DTN](image)

Figure A.1: Flowchart for No DTN
Appendix B

Experimental Server Setup

For the real life experiment, we setup a central server. We used a free web hosting provider, www.000webhost.com, which provides subdomains in the form of *.webatu.com. We setup a MySQL database on the server. The database had the following fields:

1. Player (Pac-Man or Ghost)
2. IP Address (Current IP Address of the Player)

We had a PHP script at the server which would update the IP address of a player and return the IP address of the other player if provided with the required information. A typical link to the PHP page looked like this:


Here, we are giving the role of the player along with its IP address to the PHP script on the server. The PHP script updated the IP address of the player in the database and returned the IP address of the other player.

For this setup, instead of creating a new account with www.000webhost.com, we used the existing account of one of our colleagues, Shalini Chauhan.