

**IMMUNE: an Infrastructure for Mobile Machines
in an Unplanned Network for Epidemiology**

A Technical Report
in STS 402

Presented to

The Faculty of the
School of Engineering and Applied Science
University of Virginia

In Partial Fulfillment

Of the Requirements for the Degree

Bachelor of Science in Computer Science

By

Michael Dietz

William Ashford

April 29, 2008

On my honor as a University student, on this assignment I have neither given nor received unauthorized aid as defined by the Honor Guidelines for Papers in STS Courses.

Signed _____

Approved _____ Date _____

Marty Humphrey, Department of Computer Science

Approved _____ Date _____

Bryan Pfaffenberger, Department of Science, Technology, and Society

1 Introduction

IMMUNE is a network infrastructure supporting unplanned communications between devices using short range Bluetooth radio communication. It is intended to provide information on the real world interactions of people without imposing restrictions such as direct observation or infection on those people. To demonstrate one practical application for a network of this type we implement an infection simulation model that gathers data on the spread of an airborne disease (Influenza in our prototype).

IMMUNE operates by periodically searching for and communicating with nearby participating devices. Devices exchange information relevant to the application running on IMMUNE, such as Influenza infection status in our prototype application. To improve the quality of information exchanged we developed a system for approximating physical location without using GPS devices. IMMUNE is general enough, however, for additional means of gathering location data to be added.

Previous research in the capabilities and unique abilities of Bluetooth networks has only focused on stationary devices [7, 5, 3]. IMMUNE exploits the capabilities of mobile devices, particularly those devices whose movements mirror that of a person. While the set of problems leveraging these capabilities is small, we believe they merit consideration.

Section 2 discusses the experimental environment including the devices used to build our prototype of IMMUNE. In Section 3 we describe the unplanned network, particularly common difficulties and their solutions. We go on to describe our overlaid infection model in Section 4. Section 5 discusses gathering location data through 802.11 WiFi and its applications to IMMUNE. In Section 6 we discuss several difficulties encountered in the development process that shaped our design. Section 7 evaluates the prototype IMMUNE system. Section 8 discusses areas of improvement and Section 9 offers our conclusions on our work to date.

1.1 Motivation

More than half of all individuals in the United States own cell phones [4]. It is difficult to walk down the street without spotting someone holding a one sided conversation with their cell phone or Blackberry. Many of these people make use of the Bluetooth technology in their cell phone and connect a Bluetooth headset to it to make their conversation easier. The number and ubiquity of Bluetooth enabled mobile devices sparked our interest in pursuing a project that would use these innocuous devices to answer a difficult question.

Another unique attribute of cell phones that interested us was that cell phones and other mobile devices occupy a special niche in that they are often carried in pockets or bags and taken everywhere by their owners while powered on. We postulated that code executing on mobile devices carried by users and communicating with similar devices can gather data about the interactions between the device owners without requiring the owners to deviate from their normal, day to day behavior. This obviously contributes to the convenience for the owner: that person does not need to go anywhere or do anything special to participate in a study. This also improves the quality of the gathered data because no additional restrictions or impositions are made on the participant.

As an example application, a distributed Bluetooth network coupled with location-based data from mobile devices would provide the perfect framework for a system designed to simulate an Influenza outbreak. The threat of a devastating Influenza outbreak in the coming years provides a social motivation to gather data on human interactions now [6]. This and other studies that utilize the mundane day-to-day interactions of people in their daily lives motivated the research and development of this project.

1.2 Problem Statement

Existing Ad-Hoc Bluetooth networks focus on stationary devices that use Bluetooth to facilitate communication over sensor networks. Other research projects have demonstrated the usefulness of location information gathered from mobile devices to create better models of the world [1]. We developed IMMUNE to demonstrate that a network of mobile Bluetooth devices can be combined with wireless localization technology to simulate real world interactions, such as the spread of the Influenza virus throughout a population.

Processor	400 MHz Intel XScale
Memory	64 MB(56 MB user accessible)
Operating System	Microsoft Windows Mobile 2003 Premium
Bluetooth	Bluetooth version 1.1
Bluetooth Chipset	Broadcom
Bluetooth Stack	High Point BTAccess
WiFi	SDIO or CF wireless card
Expansion Slots	SD Slot and CF Slot

Table 1: iPAQ H2210 Specifications

2 Experimental Environment

2.1 iPAQ Mobile Devices

While our motivation and proposal for IMMUNE involved cell phones, we chose to use Personal Digital Assistants in the prototype of IMMUNE. The Personal Digital Assistants that we used allowed us to develop for a homogeneous set of devices. The homogeneity of our development environment let us spend more time developing IMMUNE and less time migrating the client software to work with heterogeneous devices. Second, the Windows Mobile 2003 operating system that comes installed on the PDAs is designed to interface directly with Microsoft Visual Studio. This made the PDAs an ideal platform for our rapid development of the IMMUNE system.

2.2 Bluetooth

Our development PDAs came equipped with an integrated Broadcom Bluetooth chipset. The Broadcom Bluetooth stack that ships with Windows Mobile 2003 Premium does not include an API that allows for developer access to the stack functionality, rendering it unusable for our purposes. As a replacement we found High Point Software’s BTAccess stack. Unfortunately some of the functionality provided by High Point’s stack did not work as specified and because High Point’s BTAccess is not open source we were unable to address the issues present in the stack by modifying the stack source code directly. However BTAccess represents the only Bluetooth stacks that both worked with our mobile devices and allowed for developer interaction with the functionality presented by the stack. Our inability to modify the BTAccess stack impacted some of the design decisions that went into IMMUNE and forced us to create some workarounds to the stack issues, as discussed in Section 6.1 and 7.

3 The Unplanned Bluetooth Network

The most important component of the IMMUNE system is the unplanned Bluetooth network that we construct between distributed mobile devices. There were several technical hurdles that we had to overcome to create the unplanned network. This section describes these problems and the methods used to surmount them.

3.1 Network Creation Overview

To ease the description of the mobile device interactions that go into the creation of an IMMUNE network connection it is useful to define the actors present in the creation of the connection. We refer to the device that initiates communication between one or more devices as $Device_0$, all devices that interact with and respond to the the initiating device are identified as $Device_1$ through $Device_N$. For simplicity we refer to interactions between at most three devices in the description that follows. This is not a limiting factor, as we discuss in Section 7, our system operates with an arbitrary number of devices.

A connection between two devices is initiated by $Device_0$ when it wakes up from a period of sleep and begins to search for Bluetooth devices within communication range (approximately 30 feet for the

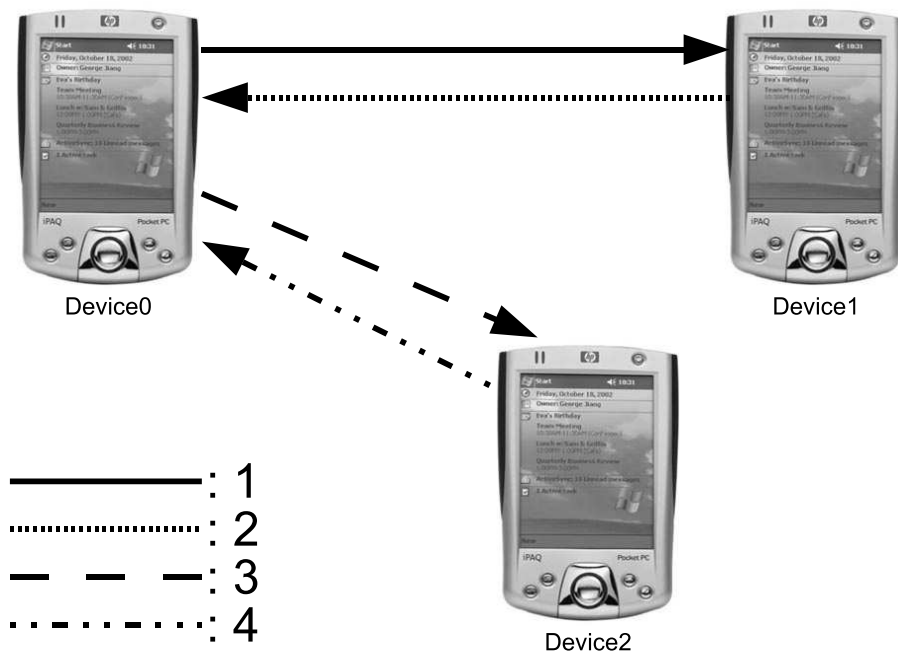


Figure 1: Unplanned network creation

most common class of Bluetooth radio). As devices are discovered Device₀ checks the newly found device's Bluetooth address against a whitelist of devices participating in a given IMMUNE experiment. If the device's address is found in the whitelist, Device₀ adds the device's name and Bluetooth address to a list of discovered devices. This list is subsequently used as a queue for actual connections.

Upon completion of the search operation Device₀ has two possible courses of action. If no devices were discovered then Device₀ sleeps for a period of time (between 30 and 90 seconds in our implementation) and repeats the search operation upon waking. During this sleep cycle Device₀ does not search for new devices or initiate any outgoing connections. Device₀ will, however, respond to connections initiated by Device_N. Otherwise Device₀ attempts to connect to the first device on the discovery queue, hereafter referred to as Device₁.

When Device₀ attempts to instantiate the connection, Device₁ is notified of this incoming connection and attempts to reciprocate by connecting to Device₀. Once the two way connection has been created, both devices determine which device should write data first and which device should read data first. The devices then exchange messages and disconnect from each other, completing a message transaction. After the transaction between Device₀ and Device₁ is complete, Device₀ then initiates connections to the remaining devices in the discovered queue and performs message transactions with them as well. When no more devices are in the discovered queue, Device₀ sleeps.

In order to reduce the number of message transactions required to propagate data between devices, we add devices that have recently initiated a connection to the local device to a temporary blacklist. We do this so that the local device does not try to initiate a connection to a remote device that it has exchanged data with recently. This design decision also reduces the time required to process the discovered devices list on some devices and reduces the number of network collisions, as explained by Figure 3 and the following section.

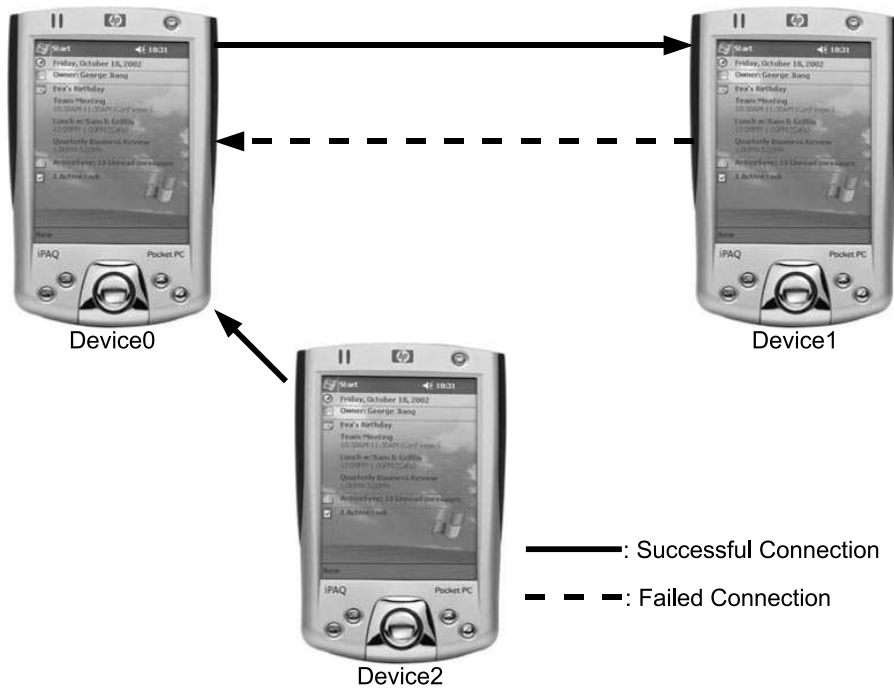


Figure 2: Connection Collision

3.2 Connection Collisions

One problem that arises from using a synchronous communication protocol is the potential for race conditions that can cause non-reciprocated connections. Figure 2 shows an example of an connection collision that has resulted in a non-reciprocal connection between $Device_0$ and $Device_1$. This scenario can occur if $Device_2$ attempts a connection with $Device_0$ while $Device_0$ is attempting to connect to $Device_1$.

$Device_0$ begins to resolve this situation by identifying that it has an outgoing connection to $Device_1$ but an incoming connection from $Device_2$. To reset the network structure $Device_0$ terminates its outgoing connection to $Device_1$. $Device_0$ then waits until it has zero incoming connections before attempting to reconnect to $Device_1$. $Device_0$'s number of incoming connections will return to zero when $Device_2$ times out while waiting for an incoming connection from $Device_0$. At this point $Device_0$ can attempt to create a new connection and $Device_2$ will enter another sleep cycle before searching for new devices.

Connection collisions are a particularly difficult problem to address because our mobile devices have no way of communicating in order to coordinate a network structure without first setting up a reciprocal communication channel. This communication restriction also means there is no way to inform a device that has created a connection conflict that it should disconnect and allow the network to reform. We therefore designed IMMUNE to recognize and remedy network collisions through time-outs and communication expectations rather than attempting to avoid them by using explicit communication.

Our failure model for connections is to abort the effort and sleep again. This does not have a negative impact on the exchange of information between devices because it reduces to the handshake problem. The handshake problem of number theory deals with a number of people that meet each other and shake hands. If everyone wants to shake hands, one possible scenario is described by Figure 3. In this scenario, the person represented by the blue circle does not need to initiate any handshakes in order to shake hands with the people represented by the other four circles.

Our synchronous communication system works in much the same way. A device may not have to initiate any network connections in order to gather the data from other devices within its communication range. Even if the device that causes the collisions never successfully initiates a connection with a remote device it

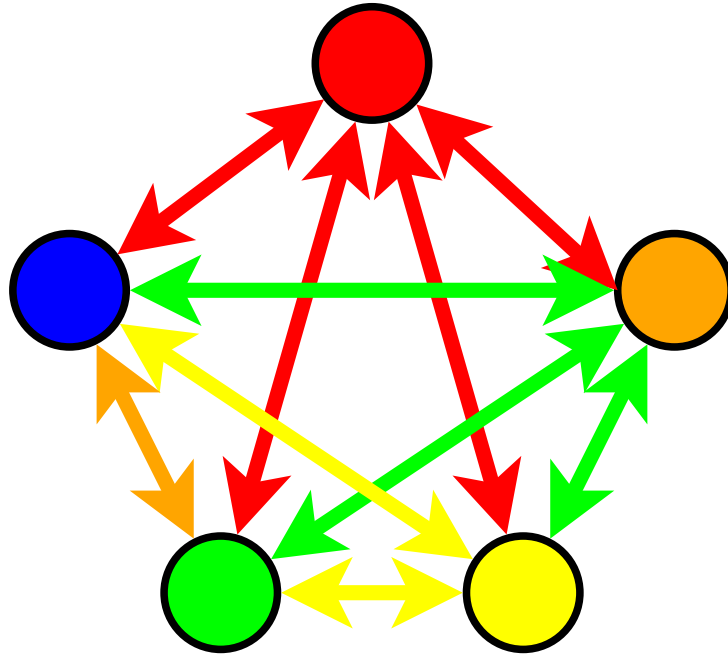


Figure 3: The Handshake problem

will still receive the remote device’s data by passively waiting for other devices to initiate connections to it. While synchronous communication streams make dissemination of information over the unplanned network simpler by reducing the number of connections that are needed to communicate with every device it also produces an unbalanced workload among the devices. An evaluation of the scalability of our synchronous communication protocol is presented in Section ??.

3.3 Getting the Data Out

Our original plan was to use the Chandy-Lamport Snapshot Algorithm to track the state of the model at a given time quanta. By transmitting state information along with the snapshot markers we would be able to gather the data from the application overlaid on IMMUNE without requiring each device to travel to any sort of polling location. As we developed IMMUNE, however, we also developed a location data gathering mechanism utilizing 802.11 under the assumption that some proper subset of the IMMUNE devices have 802.11 capabilities. With the assumption that some subset of non-zero size has access to the Internet via some WiFi connection we are able to have each device report its state to an 802.11 enabled node for transmission. In the event that no devices in the IMMUNE network have WiFi capabilities our system falls back to the original snapshot method for data extraction.

4 Infection Model

We represent the infection status of a given device as one of four possible states corresponding to the standard SEIR model used in epidemiology: susceptible, exposed, infected, and removed. When a device is in the susceptible state it is equivalent to a healthy human that is susceptible to the disease in question but has not yet been infected. The exposed state represents a human that has become infected with a disease but is not yet contagious. The infected state represents a human that is infected with the disease and is also contagious. The removed state represents a human that is somehow not involved with this disease, whether because that human is at least temporarily immune to this particular disease,, because the disease has killed that person, or some similar reason.

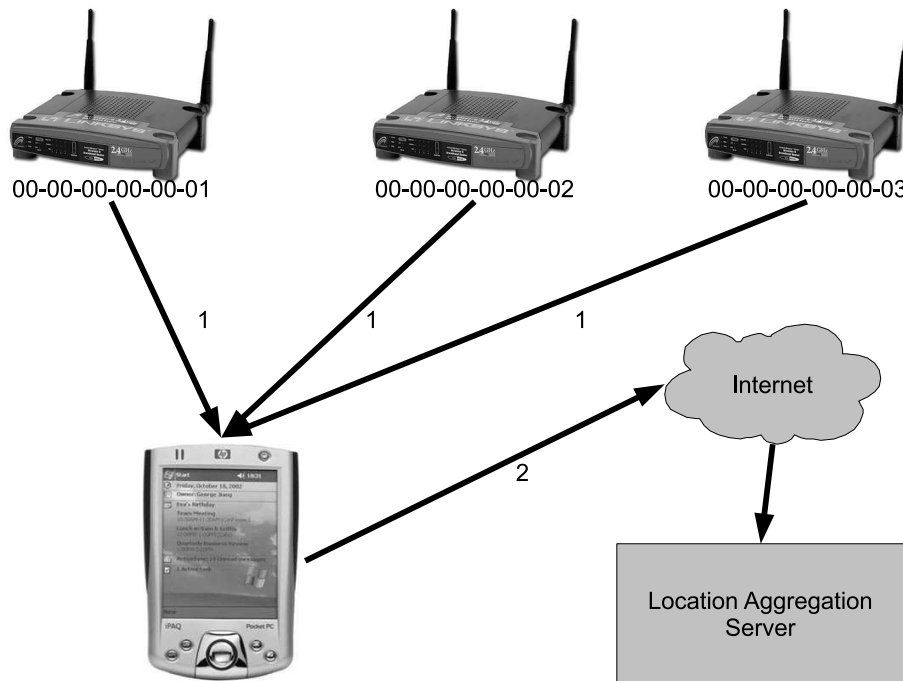


Figure 4: Wireless location example: 1. The mobile device gathers the MAC addresses of all the APs that it can see; 2. The mobile device transmits the MAC addresses to the central aggregation server for location look up and storage

When devices communicate using IMMUNE they exchange their current infection status. Based on the duration of exposure to contagious devices and the probability of infection (specific to the disease being modelled) a device may transition to the exposed state. The length of this period varies according to the disease being modelled, some especially contagious diseases may cause immediate transitions to the infected state.

Once the device has transitioned to the infected state there is a probability of recovery as a function of time and exposure to other contagious devices. Should a device recover from infection it transitions to the removed status for a period of time before returning to not infected. This period of time in the removed state represents a developed immunity and may last zero to infinite time quanta. While infected there is also a probability of death or removal from the simulation for other reasons (e.g., to simulate quarantine or staying home while sick). While recovering from death is rare, removal from quarantine or home confinement could be added as a trivial extension.

While many of our design decisions in IMMUNE were modelled on the behavior of Influenza we believe this disease framework general enough to support most airborne diseases.

5 Wireless Location System

As a supplement to the unplanned Bluetooth network we also developed a system that leverages the 802.11 WiFi Access Points distributed around an academic institution (the Grounds at U.Va., for example) to determine the position of our devices. This system allows us to incorporate location based information into the data that we can gather and propagate between devices and provide to potential applications. The WiFi triangulation aspect of IMMUNE was specifically designed to increase the usability and quality of Influenza infection data gathered with IMMUNE, identifying infection hot spots for example.

The first step in the creation of our WiFi triangulation system was to develop a web service end point on

a central server that is globally visible to our mobile devices. We accomplished this by creating a web service that publishes a service that accepts a mobile device Bluetooth address and a list of wireless Access Point Media Access Control (i.e., hardware) addresses. The web service then checks the list of MAC addresses against a database that maps AP MAC addresses to GPS coordinates. We currently use the average of the GPS coordinates of the visible APs to determine the location of the device and store the device's location along with a time stamp. The stored device GPS coordinates and time stamps can then be used to determine the location of a device at a given time as well as a mapping of the devices location over a given period of time.

The second aspect of our WiFi location system is the mobile device portion of IMMUNE that consumes the server side web service. The application instructs the wireless card to search for wireless APs and to record the MAC addresses of any currently visible APs. If the device is able to connect to the Internet via one of these APs then the client side application attempts to consume the server side web service and transmit its currently seen APs to the server side process described above. Similar to the message transaction protocol the client side application then attempts to sleep for an arbitrary period of time before waking and performing another search for wireless APs and attempting to communicate with the server side web service.

A trivial extension to our current protocol would be for devices with 802.11 to simply record visible access points to local storage. Using this extension the requirement for periodic Internet connectivity would be eliminated, as the recorded data log could be converted from lists of visible access points per time stamp to GPS coordinates after the fact. This extension would be ideal for locations with heterogeneous, non-cooperative, or restricted WiFi networks such as metropolitan areas.

It is important to note that we assume that only some proper subset of the devices participating in the IMMUNE network is capable of 802.11 connectivity. In order to provide an approximation of location data to those devices lacking WiFi connectivity we also distribute recent location data as part of each message transaction.

6 Problems Encountered

6.1 Bluetooth Restrictions

We would have liked to utilize the Personal Area Network profile of the Bluetooth communication protocol and use TCP style sockets for communication between devices. However, we found that the High Point BTAccess stack implements the Personal Area Network as a restricted serial port connection. We decided to discard the additional restrictions imposed by the provided PAN profile and utilize the Serial Port Profile directly. Furthermore, the BTAccess stack only provides one Bluetooth serial port per device, restricting us to one concurrent connection per pair of devices. With the BTAccess stack, a consequence of using a serial port over Bluetooth (both via SPP and High Point's PAN) is that each device must actively open a connection to the other, no device may be passive. This forces us to ensure that all incoming connections have a reciprocated outgoing connection (and vice versa) before a transaction can occur.

Due to these restrictions we used a synchronous method for communication between devices. Asynchronous, broadcast style communication was our original intent but the limitation to one serial port constrained us. While asynchronous communication is possible with a single serial port used for reading and writing it is highly likely that data collisions would occur due to remote devices writing to a local device's serial port while the local device is trying to write to a remote device or when two or more remote devices attempt to write to the local device's serial port at the same time. Alternatively, a synchronous communication protocol enforces an ordering of reads and writes on the serial port of a device.

Minimizing the time Bluetooth is used for active connections is beneficial to our distributed connection scheduling. In order to prevent transmission errors and coordinate connections we need to ensure that each connection has exclusive access to the serial port used for communication. Because the stack we utilized only provided a single serial port we must minimize the time spent excluding other potential users of the serial port in order to have any semblance of performance and throughput. The brevity of connections also allows us to communicate with devices that are only in range for short periods of time.

	Two Devices	Three Devices	Four Devices
Run 1	12.90	21.13	30.64
Run 2	14.14	20.52	27.20
Run 3	13.47	22.13	27.36
Run 4	12.87	21.46	28.57
Run 5	13.25	23.04	29.19
Run 6	12.99	23.84	28.61
Run 7	13.42	21.07	28.67
Run 8	14.02	20.99	28.47
Run 9	14.30	20.74	28.36
Average Time (s)	13.48	21.66	28.60

Table 2: Total message transaction time for groups of two, three, and four devices

6.2 (Lack of) Event Driven Connection Notifications

The High Point Bluetooth driver that we used in our implementation of IMMUNE lacks the functionality to inform a device that it has received or lost a Bluetooth connection. This functionality is critical to the creation of our unplanned network as we must receive notification of a new incoming connection so that the connection may be reciprocated for two way communication. Device₀ can only successfully connect to Device₁ when Device₁ recognizes that it has received the incoming connection from Device₀. Because we do not have an event driven notification from the BTAccess stack we must poll the stack to find connection notifications. Polling introduces a small amount of wait time, about half a second, into the connection system so that we can be sure that Device₁ has polled the Bluetooth stack and identified the incoming connection from Device₀.

7 Evaluation

Our evaluation of IMMUNE consisted of several micro-benchmarks and small scale tests that we performed in the final stages of the development cycle. The most important evaluation data that we gathered involves the total time taken to perform Bluetooth device searching and message transactions between two, three, and four devices. The data that we gathered during this test, as shown in Figure 2 shows that the current implementation of IMMUNE could be successfully used to simulate an Influenza infection because our Influenza model uses the amount of time that a device is in the same area as an infected device to determine if a new infection is formed. This model does not require a particularly fast dissemination of information and can work with the current performance of IMMUNE. To simulate more virulent diseases we would need to improve the connection speed and reduce the search time of the system as discussed below.

To analyze the results of our performance findings we first investigated the amount of time that a device spends performing a Bluetooth device search. We discovered that the Bluetooth stack spends ten seconds searching for devices while in most cases all devices within communication range are found within the first five seconds. Further, no other Bluetooth operations may be undertaken while a search is in progress. This means that for connections between two devices, over two thirds of the time needed to communicate information between the two devices is spent searching for devices. With access to the source code we could modify the Bluetooth stack to reduce the time spent searching for devices but that is beyond the scope of this project.

We further analyzed the results in an attempt to discover how to improve upon the speed of message transaction operations. Our findings indicated that a message transaction takes on average 3.84 seconds. Between 1 and 1.5 seconds of the message transaction are consumed by the polling driven connection notification system we discussed in Section 6.2. We found that for the current implementation of IMMUNE, the almost four second message transaction time is bearable for our applications; Influenza is most easily transmitted from close contact for extended periods of time. The probability of an infection from merely passing an infected individual is very low and is reflected in this base communication time. We would like

to either migrate to a stack that uses true event driven notification of connection events or write our own stack in a production version of IMMUNE as reduced message transaction time has very positive effects on scalability.

While evaluating the message transaction performance we stumbled upon a consistent five second wait time that occurs between a disconnection from a device and the initiation of a connection to another device. This wait time is a seemingly arbitrary component of the Bluetooth stack and accounts for about a third of the total time taken to perform the three connections required to disseminate information between four devices. The elimination of the stack's arbitrary wait time between new connections could increase the performance of our system two fold without any major modifications to the base IMMUNE system.

The use of synchronous communication streams has the benefit of decreasing the number of connections needed for every device to exchange data with every other device. However, the number of connections that each device must make to exchange data with all other devices is not balanced among the devices. For example if five devices wish to exchange data using the synchronous communication method the first device to wake from sleep must connect to four other devices to gather the data from each other device. The last device to wake from sleep doesn't need to connect to any other devices to gather the data because all the other devices have connected to it while it was sleeping. While this is currently not a problem with small sets of between five and ten devices it may be an issue as the number of devices scales up. However, we are somewhat insulated from the issues that may arise from managing connections between hundreds of devices because we expect that the upper bound of people that can fit in a 10 meter sphere to be around 20. Therefore we can design IMMUNE to work well for between two and twenty devices and ignore some of the issues associated with scalability.

8 Future Work

8.1 Optimizations

A robust implementation of IMMUNE would naturally be able to improve on some of the areas in which this prototype is lacking. For example, there is a delay in our connection creation mechanism that is not strictly necessary. This delay is due to the polling we used after we discovered that the BTAccess stack does not actually notify our IMMUNE code when an incoming Bluetooth connection is detected despite claiming to provide such a notification.

Searching for new Bluetooth devices takes 10 seconds. Searches occur often, therefore any reduction in searching time would likewise reduce the total time needed to interact with all the devices in communication range. One improvement we considered, but did not implement, was caching of previously seen devices. Such a cache could be alternated or combined with the normal searching in order to increase the performance of the system by reducing the amount of time spent discovering devices.

8.2 Potential Uses and Extensions

Our intention when we set out to build the IMMUNE was to create a system that could simulate an Influenza outbreak on the University of Virginia campus. This is not however the only potential use for the IMMUNE system. The flexibility and peer to peer nature of the unplanned Bluetooth network could be used to distribute data among first responders in emergency situations or allow for distributed communication in disaster areas where traditional data transfer methods are not feasible.

8.3 WiFi Localization

We feel that our WiFi triangulation system could potentially be spun off from the main body of our work on IMMUNE and used to create a WiFi location identification suite similar to the Rice WiFi location suite [2]. We have only created a MAC address to GPS coordinate mapping for a small number of AP centered around Olsson Hall in the Engineering School but would like to see the number of mapped APs grow in the coming years.

9 Conclusions

Unlike prior projects IMMUNE uses communication between non-stationary devices to perform its tasks. This does not lead to a model of general computation but rather one utilizing data gathered from the interactions of these devices. IMMUNE targets the relatively small but important set of problems concerned with these interactions such as epidemiology. The problems in this area can benefit from the ubiquitous cell phones that mirror our movements in our day to day activities. The combination of device to device communication with emerging WiFi localization technology can allow us to use these mundane daily activities to tackle questions about how humans interact and apply our findings to the work of epidemiologist.

References

- [1] CAYFORD, R., AND JOHNSON, T. Operational Parameters Affecting the Use of Anonymous Cell Phone Tracking for Generating Traffic Information. *Institute of transportation studies for the 82th TRB Annual Meeting 1*, 3 (2003), 03–3865.
- [2] HAEBERLEN, A., FLANNERY, E., LADD, A., RUDYS, A., WALLACH, D., AND KAVRAKI, L. Practical robust localization over large-scale 802.11 wireless networks. *Proceedings of the 10th annual international conference on Mobile computing and networking* (2004), 70–84.
- [3] LAW, C., MEHTA, A., AND SIU, K. A New Bluetooth Scatternet Formation Protocol. *Mobile Networks and Applications* 8, 5 (2003), 485–498.
- [4] RICE, R., AND KATZ, J. Comparing internet and mobile phone usage: digital divides of usage, adoption, and dropouts. *Telecommunications Policy* 27, 8-9 (2003), 597–623.
- [5] WANG, Z., THOMAS, R., AND HAAS, Z. Bluenet-a new scatternet formation scheme. *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on* (2002), 9.
- [6] WEBBY, R., AND WEBSTER, R. Are We Ready for Pandemic Influenza? *Science* 302, 5650 (2003), 1519–1522.
- [7] ZARUBA, G., BASAGNI, S., AND CHLAMTAC, I. Bluetrees-scatternet formation to enable Bluetooth-based ad hoc networks. *Communications, 2001. ICC 2001. IEEE International Conference on 1* (2001).

A Distribution of Labor

- William Ashford
 - Unplanned Bluetooth Network
 - * BTAccess Capabilities Analysis
 - * BTAccess Wrangling
 - * Serial Port Communication Semantics
 - * Synchronous Communication Protocol
 - * Snapshot Algorithm Implementation
 - Influenza Infection Model
- Michael Dietz
 - Unplanned Bluetooth Network
 - * Java/Bluetooth Investigations
 - * Network Creation Semantics
 - * Resolution of Connection Collisions
 - * Empirical Data Gathering

- Wireless Location Webservice
- Infection Model Research
- Infection Model Implementation