AUTOMATED SYNDROMIC SURVEILLANCE USING INTELLIGENT MOBILE AGENTS

Paul Miller, B.S.

Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

December 2007

APPROVED:

Armin R. Mikler, Major Professor
Tom Jacob, Committee Member
Gary Goodman, Committee Member
Ram Dantu, Committee Member
Krishna Kavi, Chair of the Department of Computer Science and Engineering
Oscar N. Garcia, Dean of the College of Engineering
Sandra L. Terrell, Dean of the Robert B. Toulouse School of Graduate Studies

Miller, Paul. Automated Syndromic Surveillance using Intelligent Mobile Agents. Master of Science (Computer Science), December 2007, 49 pp., 7 tables, 19 illustrations, references, 33 titles.

Current syndromic surveillance systems utilize centralized databases that are neither scalable in storage space nor in computing power. Such systems are limited in the amount of syndromic data that may be collected and analyzed for the early detection of infectious disease outbreaks. However, with the increased prevalence of international travel, public health monitoring must extend beyond the borders of municipalities or states which will require the ability to store vasts amount of data and significant computing power for analyzing the data.

Intelligent mobile agents may be used to create a distributed surveillance system that will utilize the hard drives and computer processing unit (CPU) power of the hosts on the agent network where the syndromic information is located. This thesis proposes the design of a mobile agent-based syndromic surveillance system and an agent decision model for outbreak detection. Simulation results indicate that mobile agents are capable of detecting an outbreak that occurs at all hosts the agent is monitoring. Further study of agent decision models is required to account for localized epidemics and variable agent movement rates.

ACKNOWLEDGEMENTS

I would like to thank Dr. Mikler for helping me through this long and arduous process of writing a thesis. Both my writing skills and thought processes have improved under his guidance.

I would also like to thank my mother for helping me proofread. Her many years of editing proved very useful. And thanks to the rest of my family who put up with my long absences and ”I have to get more work done on my thesis” excuses for not visiting.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

Epidemiology is ”the study of the distribution and determinants of health-related states or events in specified populations, and the application of this study to control of health problems” [23]. In other words, epidemiology seeks to monitor and prevent the spread of disease by studying the causes and the spread of diseases. Unlike clinical medicine which evaluates individual persons, epidemiology focuses on studying diseases within populations.

Epidemiologists require data such as the morbidity rate (rate of infection) and the mortality rate (rate of death due to the disease) of a disease to better understand how the disease spreads. This data is collected from medical professionals reporting cases of the disease to health organizations. Hence epidemiologist have to wait for live cases of the disease. The field of computational epidemiology may help improve data collection techniques or provide the ability to simulate the spread of diseases.

Mathematics and computer science have contributed to many fields of medical science. One prominent example is the human genome project where the technology of high performance computing was used to help solve the large and complex problem of mapping human genes [8]. This is just one of the subfields of computational biology. Other areas include bioinformatics which involves the storage of biological data. Unlike computational biology, computational epidemiology is a recent field that employs techniques and algorithms from computer science to help analyze and predict disease outbreaks. Stochastic cellular automata have been used to simulate the outbreak of a disease using the knowledge of known outbreaks as the basis of the simulation model [31]. By simulating outbreaks, the nature of how a disease spreads can be studied without the need to wait for the next live outbreak. State transition systems have also been used to analyze the control and treatment alternatives of HIV/AIDS [16]. Another aspect of epidemiology that has not been assisted by computational epidemiology is public health surveillance.

1.1. Infectious Disease Surveillance

Outbreak investigation is a field of epidemiology where the spreading of an infectious disease is studied in order to control or prevent the further spreading of the disease. Infectious disease

outbreaks are detected through the surveillance of diagnoses cases of the disease. National surveillance of infectious disease started in 1878 to prevent the introduction of infectious diseases in the US from overseas [10]. Surveillance has expanded to include morbidity and mortality reports from state health organizations for diseases in the annual lists of nationally notifiable diseases published by the Center for Disease Control and Prevention (CDC). However, reporting cases of a disease to the CDC is voluntary [4]. Hence the data collected may not portray an accurate status of the state of public health.

State health organizations require health care providers to report confirmed cases of an infectious disease. The health organizations publish lists of identifiable diseases similar to the CDC including the time frame in which the disease must be reported. Some diseases are required to be reported immediately including anthrax, food born botulism, rebeola (measles), and the plague. Other diseases are required to be reported within one week such as asbestos exposure, chickenpox, gonorrhea, or the mumps.

The CDC and state health organizations analyze trends in the the morbidity and mortality data to determine if an epidemic may be occurring. However, these trends may be imprecise as the data collected may be delayed. The first delay occurs in the time when symptoms manifest and when the infected person first sees a physician. For the cases of inhalation anthrax in the United States in 2001 the median duration between the onset of symptoms to the initial healthcare visitation was 3 days [6]. The second delay occurs in the time when a patient first visits a physician and a diagnosis is made. A physician will not report the disease until the diagnosis has been confirmed through examinations and diagnostics tests.

Misdiagnosis also causes delays as many severe infectious diseases have similar symptoms as the more common influenza [29]. Physicians may also not recognize a disease as symptoms may be similar to a patient's pre-existing conditions. For example, a patient was being treated for congestive heart failure in a Toronto hospital emergency room and was exposed to the severe acute respiratory syndrome (SARS) virus [11]. After being released the patient returned four days later with fever, trouble breathing, and fluid in the lungs which is consistent with congestive heart

failure. However, the first two symptoms are consistent with SARS, and the patient was misdiagnosed with recurrent congestive heart failure. A diagnosis of SARS was never made before the patient died 15 days later.

1.2. Syndromic Surveillance

Disease surveillance monitors the state of public health using diagnosed cases of a disease in order to detect possible outbreaks. Another way to measure public health is to monitor the effects of disease which are discernible before a diagnosis is confirmed [24]. Prior to going to a physician, a person might miss days at work or school, purchase over the counter medications, or purchase other items such as kleenex or juices high in vitamin c. Syndromic surveillance typically will use syndromes, also called chief complaints, or diagnostic tests to search for abnormal clusters or areas where the occurrence of the syndromes are above normal [14]. Time series analysis and other statistical tools are used to locate the abnormal rates of occurrence in syndromic data. By monitoring the effects of a disease, syndromic surveillance systems may be able to detect possible outbreaks earlier than disease surveillance systems as abnormal increases in the rate of occurrence of the effects should be detectable before abnormal increases in the rate of a disease are identified.

1.2.1. Current Surveillance Systems

The Real-time Outbreak and Disease Surveillance (RODS) [30] and the Early Notification of Community-based Epidemics (ESSENCE) system [22] store data collected from participating health systems in centralized databases. Outbreak detection algorithms are executed every 4 hours on the data, and alarms are raise 3 d based on criteria set within the algorithms. These systems have a number of shortcomings. A failure with the database may prevent the surveillance system from performing efficiently or result in the loss of data. The developers of RODS had to deal with such an issue where data being transmitted to the database was lost while the database was offline. To solve this problem data is cached the until until the database is online, but this does not prevent the potential delay in executing the detection algorithm should the database be offline for longer than 4 hours. Catastrophic database failures will also result in the loss of all data if a sufficient

backup is not kept. Even with a backup, time must be taken to restore the database before outbreak detection can continue.

Another issue is scalability both in data storage space and computing power to process the data. Current syndromic surveillance systems typically contain a set amount of storage with a constant amount of computing power which limits the the amount of data the system is able to process. Monitoring larger geographical areas will require large amounts of storage space and more computing power, and a single computer system would incur the cost of upgrading the hard drive and central processing unit (CPU) as the amount of data grows too large. Advances have been made that have increase the computing power of CPUs and being able to store more data on the same sized hard drives. However even a system with the most powerful processor and an array of the largest hard drives is still constrained and thus will limited in the geographical size of syndromic data that can be processed. Distributed computing systems provide the required scalability using multiple computers to create a single computing system. To increase the storage space and computing power more computers are added to the system. An example of a distributed computing system is the intelligent mobile agent system.

1.3. Intelligent Mobile Agents

The mobile agent paradigm is a shift from the traditional client server communication network. In the client server paradigm, a stationary program on the client transfers data to the server where the data is processed or stored. In contrast, programs of a mobile agent systems that process or collect data move to the location of the data [27]. Mobile agents are executed on the computing resources within the agent network making the mobile agent system scalable in terms of computing power and storage space. The mobile agent paradigm has two main goals. The reduction of network bandwidth utilization, and asynchronous interaction with the user [26].

Figure 1.1 depicts a mobile agent network with nodes where information may be found and computing power used to analyze the data. Nodes are connected over data networks through which agents travel and agents are autonomous programs that make decisions about where to move and what actions to take. For example, the agents found in the Nomad eAuction system are used to visit eAuctionHouse sites to place bids on behalf of a user [18]. When a user wishes to participate in

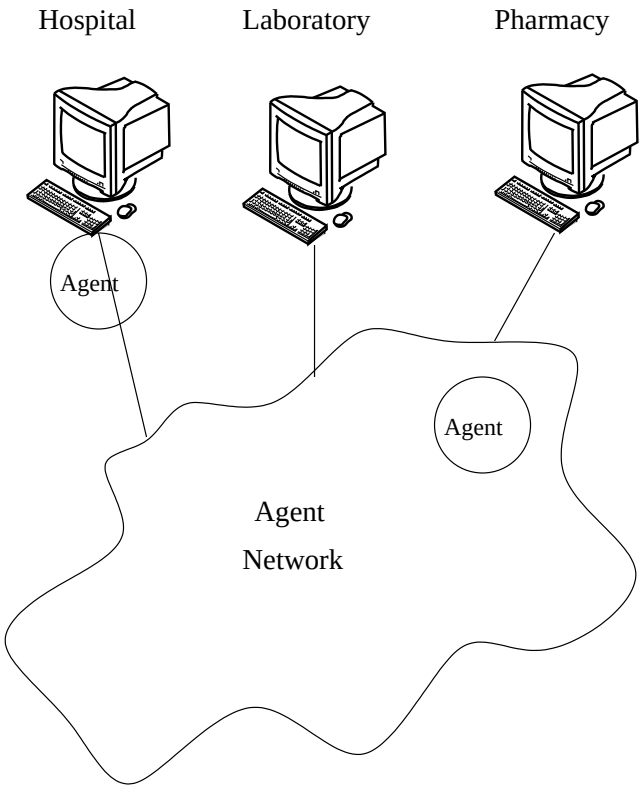

FIGURE 1.1. Example agent network with connected hosts for syndromic surveillance.

an eAuction, a mobile agent is programmed with the user define parameters, and the agent travels via the Internet to eAuctionHouse sites searching for auctions that match the user parameters. The mobile agents are autonomous in that the agents place bids without interaction with the end user. In other words, the mobile agents make decisions using the predefined user parameters and data found at the eAuctionHouse sites.

Mobile agents may be used to control the amount of network bandwidth used for network intensive systems. An example of mobile agents have been used to decrease network overhead is agent-based distance vector routing (ADVR). In this system mobile agents move between routers in a network analyzing and updating the routing table of the routers [2]. Agents take the place of large network packets of traditional dynamic routing protocols that pass a router's entire routing table to neighboring routers. As networks grow in size, the overhead of traditional routing protocols would grow unbounded, but by bounding the number of agents the network overhead of the ADVR system is bounded given the amount of data contained by the agents is not proportionate to the size

of the network. To control the amount of agent data, agents do not carry the entire routing table of all the routers in the network. Rather, the agents carry only the data required to calculate a new routing table at each router. The feature of agent systems can be used for syndromic surveillance systems where agents contain only data required to make decisions regarding possible outbreaks. This is important for protection of patient information.

1.4. HIPAA Privacy Rules

Syndromic surveillance systems that collect patient information must comply with the HIPAA privacy rules. The *Health Insurance Portability and Accountability Act* (HIPAA) dictates the development of standards to electronically exchange health information between health providers and health insurers. Included in the act are provisions for the creation of privacy rules to limit the use or exchange of individually identifiable health information by health care providers or health insurance companies. The the privacy rules requires a person's protected health information be kept private while allowing the exchange of de-identified health information to "promote high quality health and to protect the public's health and well being" [9]. De-identified health information includes but is not limited to

- Gender
- Age
- General Location (e.g. zip code)
- List of symptoms
- Diagnosis

Only the de-identified health information is required for syndromic surveillance as information such as a patient's name, social security number, or specific address is not useful in detecting outbreaks. In an agent based syndromic surveillance system this data may be kept in a separate database from the database containing protected patient information.

1.5. Overview

An infrastructure for intelligent mobile agents will be described in Chapter 2. Chapter 3 will show how the intelligent mobile agent infrastructure can be used to perform syndromic surveillance. A simulation of the syndromic surveillance system and experimental results will be discussed in chapter 4, and chapter 5 will summarize this research and specify future work.

CHAPTER 2

AGENT BASED SYSTEMS

Agent based systems are made up of two primary entities: hosts and agents. Hosts in the agent network will require specialized software to receive, initiate, and transmit agents. When transmitting an agent, the host will send the agent code and the state of the agent. The agent must be transfered so as to facilitate the agent's execution on the receiving host in the same state the agent left the transmitting host. The agent state can include the value of the central processing unit (CPU) registers and the execution stack or more simply just the values of the global data structures [20].

One requirement of agent systems is that the receiving host must be able to verify that it is able to run the agent to be received. To run the agent, the receiving host must support the programming language used to implement the agent. Even if the host supports the agent programming language, the agent may still need addition services from the host. The services may include access to specific type of data or specialized code too large to be carried by the agent.

When two agents are located on the same host, the agents must be able to interact. This exchange of data provides agents with additional data without the need to visit all hosts in the network. Complex problems may be divided into a set of less complex sub-problems using different agents for each of the smaller problems. Agents interact and share the results of the individual sub-problems to solve the overall problem in parallel similar to a multithreaded application running on a single host. Consequently, an agent based solution will be able to achieve faster results to the overall problem.

Agent security has an important role in mobile agent system design. Mobile agent systems pose additional issues not seen with traditional server/client systems such agent alteration by a host. The forms of manipulation include removing all agent data or change agent data to make the agent perform actions it would not have normally taken [5]. Agent code can also be modified to add functionality to the agent for carrying out malicious attacks on other hosts or agents.

The issues above will be discussed further in the following sections.

2.1. Agent System Security

Traditional server/client systems must be protected from attacks such as spoofing [17] where an untrusted host claims to be a trusted host. A specific use of spoofing is the ”man in the middle” attack where a malicious host captures data transmitted between two other hosts by spoofing the address of the receiving host. The hosts exchanging the data are not aware of the ”man in the middle” capturing the data. Hosts may also contain vulnerabilities that remote hosts can exploit to gain unauthorized access. After gaining access, the malicious host may collect data off the compromised host or attack other hosts. Just like server/client systems, agent systems are also concerned about data being captured by malicious hosts and should use the same protection mechanisms that the server/client systems use. For example, the data transmitted between two hosts encrypted using either private-key or public-key encryption will protect the data as long as the keys are kept secret or secure key negotiation algorithms are used.

Encryption techniques may also be used to protecting the agent from alteration through the use of cryptographic signatures derived from the agent’s code and data. A simple method is for the transmitting host to create a checksum of the agent code and data using a private key and an asymmetric algorithm. The agent will contain the public key to be used for verifying the checksum on the receiving host.

Another form of attack is for a malicious host to deny an agent its execution after receiving the agent [15]. This form of attack prevents the agent from performing its normal function and may also prevent the agent system from functioning properly. An attack may be prevented by building a trust relationship between hosts. If the transmitting host trusts the receiving host and the agent trusts the transmitting host, an implied trust exists between the receiving host and the agent [13]. Trust between two hosts can be established through authentication between the transmitting and receiving hosts. Alternatively, the receiving host may authenticate itself with the agent before receiving the agent thus constructing a direct trust between the agent and the receiving host.

In a server/client system agent code is stationary, and there is an implied trust between the host and the code as an authenticated user was responsible for installing and running the code. Users of a host are given privileges to run code on the host which may be limited to subset of resources

available on the host. In comparison, mobile agent code does not have the same implied trust as the host will initiate the agent code itself after the agent is transfered from a trusted host. An implied trust could be construed if the receiving host trusts the transmitting host and the transmitting host trusts the agent, but direct trust [21] between the receiving host and the agent is possible by requiring the agent to authenticate with the receiving host. If a malicious agent were able to be transfered to a host, then agent authentication before moving to another host may be able to prevent the agent from traversing the agent network.

2.2. Agent Construction

Conventional network communication involves the only the transmission of data between two hosts, and protocols are designed that specify how hosts will format or interpret the data being transmitted. However, with mobile agents the agent executable code as well as the agent data is transmitted between hosts. Figure 2.1 shows the two main segments of an agent. Separate segments for data and code simplifies the agent delivery system in that the delivery system transparently transmits the data formatted by the agent.

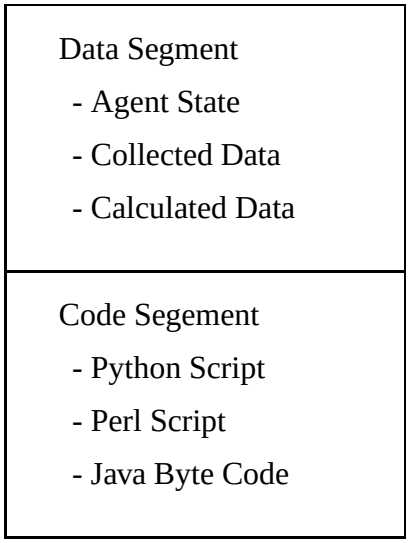

FIGURE 2.1. Segments of an agent.

The code segment contains the agent's executable code. Agents run on a variety of platforms with different types of CPUs and operating systems. Scripting languages such as Perl or Python are best suited for agents as script interpreters exist for most platforms. Java is also a good candidate language as Java Virtual Machine implementations are also available for most platforms. An agent

delivery system will have to provide a mechanism to determine if a destination host supports the language of an agent. This is described further in the next section.

2.3. Agent Transferal

2.3.1. Host Agent API

Agents are transfered to a new host upon an agent's request. In order to support this, hosts must provide an interface to the agents, and a protocol is required between hosts for the agent transfer. The Distributed Agent Delivery System (DADS) [3] describes the agent delivery protocol (ADP) which provides both of these interfaces. An application programmers interface (API) is used by agent programmers to make requests to the agent's operating environment. The ADP API specification includes three methods:

Init(): called by the agent immediately after being executed on a new host to initialize the agents variables.

Move(hostname): called by the agent when the agent wishes to move to a new host. The agent should specify which host it wishes to move to, provide its authentication set, what programming language, and the services it requires

Event(event string): called by the agent to log an event at the local host.

Of this list, only the Move() operation is required. The host should be able to initialize the agent prior to running the agent, thus not relying on individual agent implementations to ensure that Init() is called. Also, agents will require access to local data. However, hosts may not wish to allow global access to all resources. The API should provide an interface for the agent to connect to a resource or service. Then the host API would include the methods described below.

Move(hostname): called by the agent when the agent is ready to move to a new host. The hostname is the host to transfer the agent to.

Connect(service name): called by the agent to attach to a local service. This is to allow the agent access the data on the local host the agent requires to perform its function

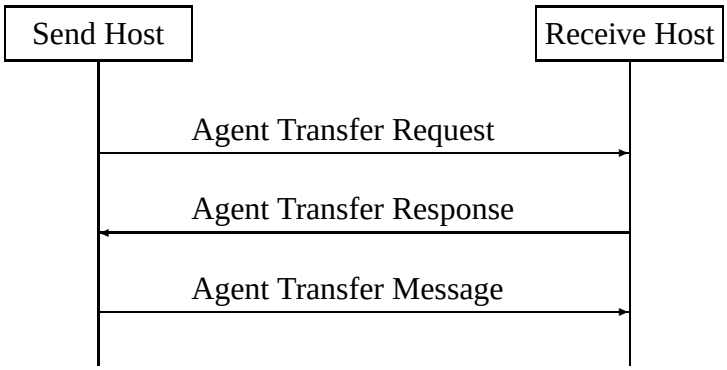

FIGURE 2.2. Sequence of messages for agent transfer.

Hosts will require a well known interface to the agents for agent initialization and activation. The agent API contains the following functions. The exact function prototypes will depend upon the programming language being supported.

Init(data): called by receiving host to initialize the agent variables using the data pointer passed in.

GetData(data): called by sending host to retrieve the agent data. This is the data that will be provided to the Init function on the receiving host.

Execute(): called by the receiving host to run the agent code.

2.3.2. Agent Transfer Protocol

To transfer an agent, hosts use a sequence of messages (see Figure 2.2) which includes three messages. Agent transfer is initiated when the agent calls the Move() host API method. The host will use the agent transfer request to determine if the receiving host is able to receive the agent. Figure 2.3 shows the contents of the agent transfer request message. The message type field should contain a value that uniquely identifies the request message. A request ID is used to correlate

Message Type
Request ID
Agent Language
Agent Authentication Set
Requested Module List Length
Requested Module List

FIGURE 2.3. Agent transfer request message contents.

Message Type
Request ID
Response
Host Authentication Set
Supported Module List length
Supported Module List

FIGURE 2.4. Agent transfer response message contents.

responses to a request as the transmitting host may be sending more than one agent at any time using individual request messages. The agent language field specifies the agent's programming language for the receiving host to verify the language is supported. Authentication sets are used by the receiving host to authenticate the transmitting host. A list of modules is provided by the agent to inform the receiving host which modules the agent requires.

The agent transfer response shown in Figure 2.4 contains a message type field which is set to a value to uniquely identify the response message. The receiving host must use the request ID value from the request message in the response. The response field will indicate if the receiving host is able to receive the agent or specify the reason why the agent cannot be accepted. Possible reasons for rejecting the agent transfer include the agent's programming language is unsupported, authentication failure, or an internal failure of the receiving host. The host authentication set includes the credentials of the receiving host. A list of supported modules is generated from the modules listed in the request message.

After receiving the agent transfer response, the transmitting host should authenticate the receiving host on behalf of the agent. The agent trusts the transmitting host since the host was authenticated by the agent's previous host. Once the receiving host is authenticated, the transmitting host will construct an agent transfer message using the agent's code and data segments. Agent code is already available to the transmitting host and is copied directly into the message. However, the agent data is retrieved by calling the GetData() method which returns a stream of bytes in a format the agent code will understand after the transfer is completed. Java based agents might use Java Object Serialization which calls a Java method of an object for a byte stream that

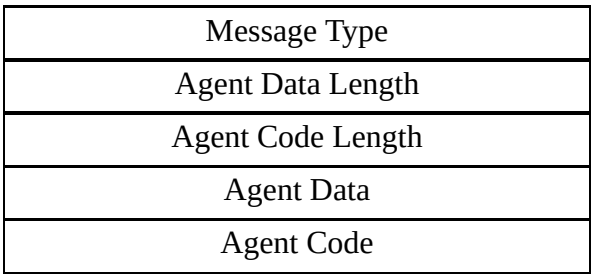

FIGURE 2.5. Agent transfer message contents.

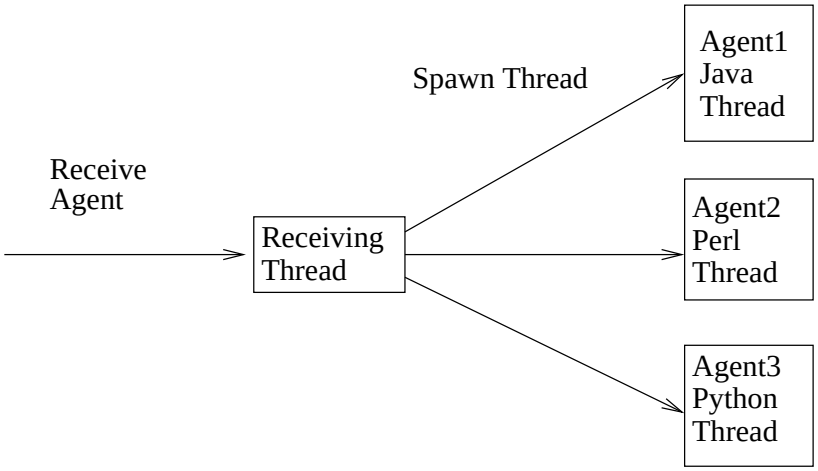

FIGURE 2.6. Thread model for running multiple agents.

represents the object's state. The transmitting host should encrypt the agent code and agent data before sending the agent code and data to protect the agent from being intercepted by a malicious host. The transmitting and receiving hosts would use a key negotiation algorithm to determine the encryption key.

2.4. Agent Execution

Once an agent has been transfered the receiving host decrypts the agent's code and data segments. The agent code is passed to the appropriate interpreter for script language based agents or the Java Virtual Machine for Java based agents. In order to support the execution of multiple agents the host should spawn separate threads of execution for each agent (see Figure 2.6). The new thread is responsible for initializing the agent data using the agent's *Init()* method and executing the agent using the agent's *Execute()* method.

In order to better protect the host from rogue agents, the host should run the agent in a sandbox environment which is an environment that restricts access to local resources. Unix based hosts

typically include the *chroot* utility which limits an application's access to the host's file system an hardware devices. For each agent that arrives, the host launches the interpretor or Java Virtual Machine using the *chroot* utility.

2.4.1. Agent Injection

Agents are injected into the agent network by software written that reacts to user input. For example, agents are created by the Nomad eAuction software when a user selects an item they wish to find and how much they are willing to spend. User applications are not the only trigger for agent creation. Operating system or hardware events on the host such as a hard drive failure could cause and agent to be created to check the health of hard drive on various hosts in the network.

To inject a new agent into the system, a trusted host should be used that supports the agent transfer protocol. Like any other transmitting host, the injecting host should call the agent's Init() method but should not pass in any data. This requires that agent data structures have default values set by the Init() method to ensure stable and robust agents. Then the injecting host calls the agent's Execute() method to have the agent determine the first host to move to and call the host API Move() method.

2.5. Agent Access to Local Data

A primary goal of agents is to access or store data stored at a host and additionally alter the running parameters of the host. One example is agent-based distance vector routing where agents access the host's routing table, calculate the shortest path and update the host's routing table. Agent delivery systems will not know what access will be required by all agent types. Agent based solutions will require the capability to extend the functionality of the host's delivery system. An agent based solution will define the agent types and behaviors as well as the required host modules the agent will use. The modules are included in the agent's runtime environment, or more precise the modules are contained in the script interpretor environment or Java runtime environment. Hence modules must be written in the same programming language as the agents.

2.6. Inter Agent Communication

Inter agent communication increases the flow of information through an agent network by sharing information that agents will have collected visited different sets of hosts. Various forms of agent communication are available from storing data at a hosts for other agents to find, collaboration using the knowledge of the location of other agents, or direct agent interaction when agents are collocated on the same host.

Agent-based distance vector routing includes an example of agents exchanging information without the need for direct interaction. Two agents will have routing table data based on various sets of routers within the network. When one agent visits a router, the agent will update the routing table of the router with its knowledge of the network, and when the second agent visits the same router the agent will incorporate the router's routing table data that contains the first agent's data. Thus individual agent information is adjusted with routing data from other routers in the network without having to have visited all of the routers.

On some mobile agent systems, agents that are not collocated on the same host are capable of exchanging information. Concordia provides a mechanism for agents to communicate or coordinate with a group of agents. In other words, agents receive events that group members send to a central group manager object [32]. Interagent communication between hosts requires more complex agent delivery systems that include agent location management. Location management depends upon a central host that agents register with to receive messages or events from other agents. Remote agent communication also contrary to one of the main benefits of mobile agent systems which is reduced network utilization [25].

Agents will define the type and format of the information to be exchanged similar to agent data segments transfered between hosts. Hence the agent delivery system does not need to know the details of the data agents exchange. To facilitate interagent communication, a list of agents present at the host must be available which includes a unique agent identifier assigned by the host upon agent arrival. An agent type may also be provided by the agent for other agents to use in their decision to initiate communication. An agent would then iterate the list of agents to determine which agents to communicate with.

Interagent communication is initiated by an agent sending a message to another agent via the host using the agent id provided. Prior to leaving a host, agents should query the agent delivery system for any messages sent to the agent. When the host delivers a message to an agent, the agent ID of the sending agent must be provided to the receiving agent which will establish the means for the receiving agent to send a response message to the originating agent. Once both agents have the agent ID of the other agent, messages may be exchanged until the agent interaction is completed.

2.6.1. White Board

Another form of interagent communication uses the concept of a whiteboard where information may be stored future use. Each host will provide a whiteboard API interface for agents to store information and access the next time the agent visits the host. In fact, any agent visiting the host should have access to information left by other agents which will facilitate the ability for agents to exchange data without being collocated on the same host. To organize large amounts of data, multiple whiteboards may be used each with a name that uniquely identifies the whiteboard. Agents will require the ability to quickly find information stored on a whiteboard. To quickly find information in a database, keys are used to differentiate the records of the database. This approach may be used to store and locate whiteboard information using a unique identifier for a particular set of information.

The name of whiteboards and data identifiers should be completely controlled by the agents to accommodate a versatile environment for storing different types of data that may be used by agents that perform unrelated functions. This is realized by additions to the host API.

CreateWhiteBoard(wName): creates a whiteboard with the name if one does not already exist

StoreData(wName, dIdentifier, data): stores the data on the whiteboard wName using identifier as a key

FindData(wName, dIdentifier): return the data stored on the whiteboard wName with the key dIdentifier

Changes to the agent API are not necessary as the agent initiates all interactions with the host regarding whiteboard access.

CHAPTER 3

AGENT BASED SYNDROMIC SURVEILLANCE AND SYSTEM DESIGN

An agent based syndromic surveillance system will use agents to collect and process information from various sources of syndromic data to analyze the trends in the data and make decisions on whether an alarm should be raised. This chapter will show how the concepts described in the previous chapter may be used to construct the surveillance system.

3.1. Data Sources

Data used by the agents will come in many forms including patient data from hospitals and clinics, medication sales from pharmacies, or tests ordered at independent laboratories. Patient information is usually written on paper forms and will require data entry into digital formats for agents to collect. The digitized patient data can be free form text as used by the Early Notification of Community-based Epidemics (ESSENCE) II and Real-time Outbreak and Disease Surveillance (RODS) syndromic surveillance systems. However, free form text requires filtering software to detect the syndromes being monitored. The *International Classification of Diseases, 10th Revision,* (ICD-10) and the *International Classification of Diseases, Ninth Revision, Clinical Modification* (ICD-9-CM) are used by the National Center for Health Statistics to collect morbidity and mortality data from health care providers. ICD codes provide a standard for classifying diseases and symptoms to be used by the health care providers. Studies have shown that ICD codes can be used for syndromic surveillance systems to detect outbreaks. One study performed at the University of Pittsburgh gathered ICD codes for 669 patients to determine the ability to detect acute respiratory illnesses [12]. This study showed the accuracy to be lower than expected with a sensitivity of 44%. The primary issue is the accuracy of assigning ICD codes by health care providers. An additional issue with ICD codes is that they may not be recorded in patient records until days or weeks later [24]. Another study compared the accuracy of three methodologies for syndrome detection: Naive Bayes classifier on free form text, bigram Bayes classifier on free form text, and ICD coded emergency department diagnosis classifiers [19]. This study showed that the Naive Bayes classifier had the best sensitivity of the three with 69%. As in these studies, a mobile agent based system must use either ICD codes or free form text. An agent based system will have to choose how to detect

syndromes in the same manner as standard surveillance systems. The sensitivity percentages of the studies indicate that free form text is better at detecting the syndromes.

Syndromic data needs to be in a digital format for collection by an agent. For some diseases, a large delay may prevent the system from detecting possible outbreaks. Cases of inhalation anthrax in 2001 showed an average of one to three days between when a patient first sought health care and when the patient was admitted to a hospital for inhalation anthrax [6]. Syndromic surveillance systems have a small window of time to detect an outbreak in order to provide the benefit of early detection. To improve the amount of time to make data available, syndromic surveillance systems such as the resource reservation protocol (RSVP) [33] and Lightweight Epidemiological Advanced Detection Emergency Response System (LEADERS) [28] have been developed using web-based or hand-held devices. The latter provides the best opportunity for ensuring a timely insertion of data, and if the hand held devices were wireless, patient data entered by physicians could be transmitted to a central database as the data is entered. The devices would also improve the accuracy of ICD codes as an interface would be presented for a physician to select appropriate symptoms, enter a diagnosis, and order tests.

For non-patient related data such as the sales of over the counter medication, the universal product codes (UPC) assigned to these products can be used as unique classifiers. Sales data is typically transmitted by cash registers to a central inventory database that contains the number of sales for items in the store. Prescription medication sales will require a different for of identification as they do not have UPCs like over the counter medicine. The name of the medication will be sufficient as a unique identifier as each medication including generics have unique name. The issues of transcription errors evident with patient information are not a concern with medication sales as the identification and counts are generated by the cash registers and inventory database.

Syndromic surveillance systems may use census information to determine how wide spread an outbreak is within a population area. Census information is unique in that the data is updated every 10 years in the USA. It would not make sense for agents to travel to special hosts on a regular basis to retrieve the data. Agents that will utilize census data as a part of their decision model could be configured to retrieve the data from specialized hosts every 10 years or as often as needed. Another

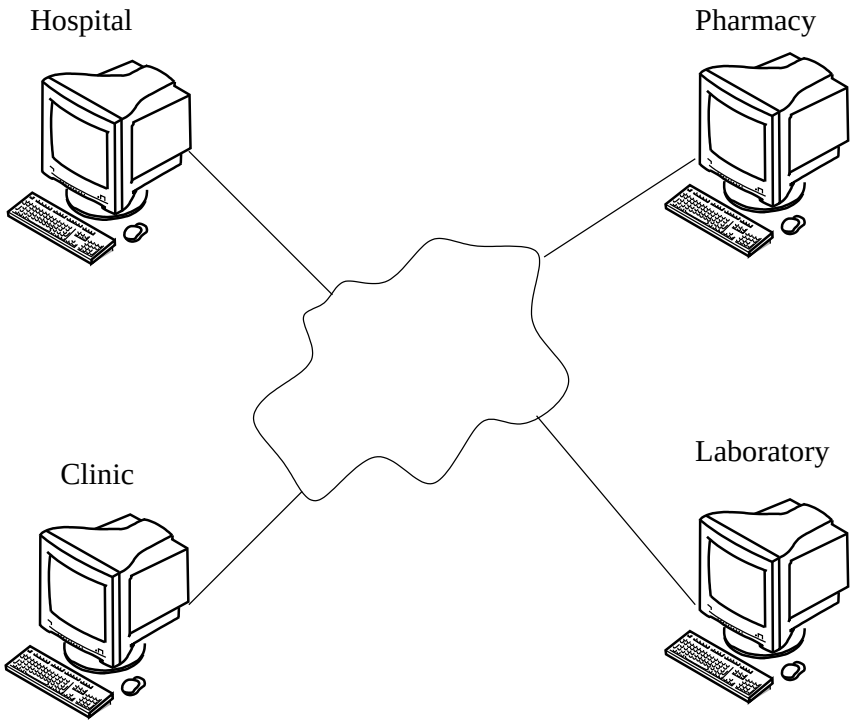

FIGURE 3.1. Hosts connected via the Internet.

approach would be to use another type of agent (Census Agent) that is responsible to disseminate census data to the other agents in the network.

3.2. Network Hosts

There are two types of hosts within the agent network: data hosts and maintenance hosts. The hosts may be fully connected via the Internet or through a direct connection between two hosts. Connection via the Internet would be the most cost effective as prices for high speed broadband Internet access has decreased significantly over the past few years. However, the Internet also poses a higher security risk as the agent network hosts are exposed to all hosts on the Internet. In other words, any host on the Internet could attempt to crack into the agent network host to gain access to the host's database or attempt to send a malicious agent to the host. The Internet does provide the most opportunity for agent mobility as the agent itinerary is not limited to any specific order. The hosts of the network can also store information about other hosts on the network. The information may include the location of the remote hosts and the type of data contained at the hosts. As an agent travels the network the agent will learn about the other hosts and develop new itineraries with the knowledge gained.

FIGURE 3.2. Networks connected via a direct connection.

Direct connections may be used create a combined agent network from autonomous networks such as private medical centers comprised of a hospital and individual physician offices (Figure 3.2). The exposure of the networks is limited to the gateways which can be used to protect the hosts from attack originating in the other network. Direct connections for hosts are not as cost effective as the Internet as each host would have multiple connections as shown in Figure 3.3 to ensure agent mobility. If a hosts only contains a single direct connection, the agent will have travelled from the remote host and is forced to travel back to the remote host which may not contain a significant amount of new data.

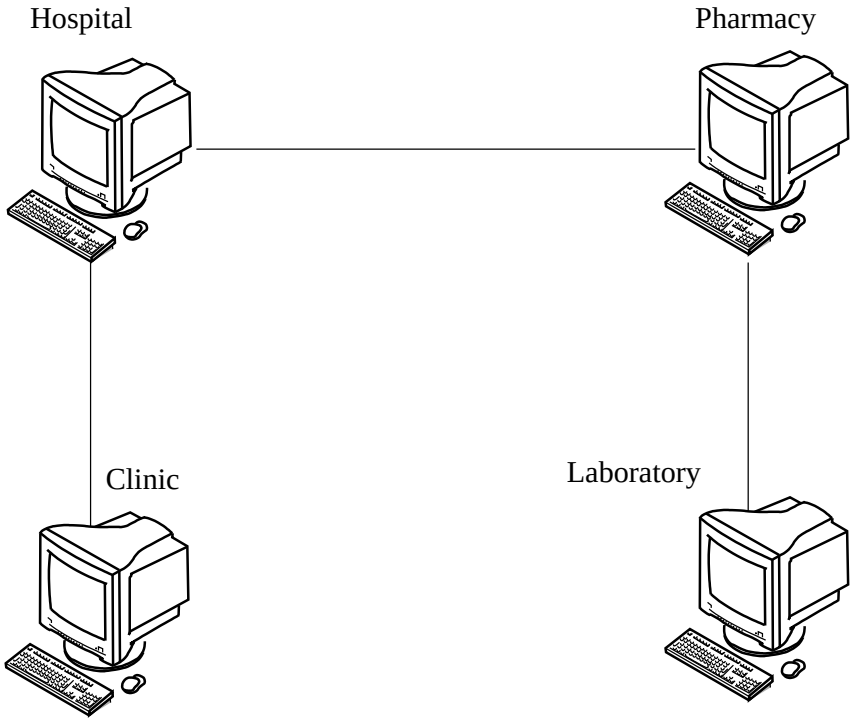

FIGURE 3.3. Hosts connected via multiple direct connections.

3.2.1. Data hosts

Data hosts are hosts located throughout the agent network. Agents will visit the data hosts to find the information needed to perform their functions. These hosts are located at hospitals, clinics, grocery stores, pharmacies, independent laboratories, and many other possible locations. For hosts that contain patient data, the HIPAA protected data should be filtered into a separate agent database as seen in Figure 3.4. Separating the protected patient data from the agent network will also provide more protection from host base attacks against the computer connected to the agent network. While protecting patient data is important, it is not the only reason why filtered data may be presented to the agents. Pharmacies and grocery stores will not want the sales of all items in the store made available. Rather only the sale counts of the items of interest are required to be presented to the agents.

The separation of protected data and filtered data raises the question of how the data is moved to the agent database. The protected database could be configured to send the required data to the agent database in real-time. In other words the required data is sent to the agent database as the protected database receives the data. However this approach would result in large amounts of network traffic. In a pharmacy, the protected database is constantly updated as sales data from cash registers are received, and it is possible for the same item to be updated by different registers at the same time. If the data is sent to the agent data base in real-time, then the same item could be updated more than once in a relatively short period of time. An alternative would be for the agent database to poll the protected database at regular intervals or have the protected database push the required data to the agent database at regular intervals.

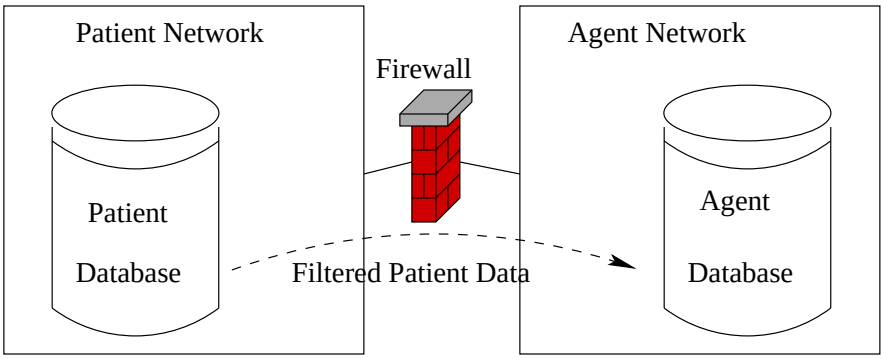

FIGURE 3.4. Separation of protected patient data and filtered patient data.

3.2.2. Maintenance hosts

Maintenance hosts are needed for the maintenance and upkeep of the agent system. There may be multiple maintenance hosts to perform the various tasks, or a single host could be used to perform all tasks. One task is to inject the initial agents into the system when the system is first deployed or inject new agents types after the system has been operating for some time.

Maintenance hosts can also be used to re-program agents. When an agent visits the maintenance host, either the host may detect that the agent is to be upgraded or the agent may detect it is located at a maintenance host and request available upgrades. The former solution provides for simpler and smaller agent code sizes as agent code to detect the maintenance host type is not required. However, this does require the agent types to have a specific signature that the host can detect or the agent would have to provide its type to the host. Having agents provide their own type is the most robust solution as no heuristics would be required, and thus no possibility for error in detecting the agent type. When a maintenance host receives an agent, the host does not run the agent. Rather, the host will move the agent's data to a new agent with upgraded code. The new agent is then injected into the network, and the old agent is discarded.

In a syndromic surveillance system, alarms are raised to alert health officials when a possible outbreak is detected. In an agent based system the agents are responsible for raising these alarms. However, the alarms cannot be raised on any host in the network. Specialized hosts are required that will alert health officials based on the severity of the alarm. These alerts may include one or more of the following.

- Send text messages to pagers or cell phones
- Play a sound file
- Send emails

The chosen alerts(s) would depend on the severity of the alarms from agents. For example a low severity alarm might not require immediate attention and only an email would be sent, but for a high severity alarm that requires immediate attention the host could use all forms if alerts.

A data collection host may be used to track the state the health of a population over time. An agent based system is well suited to gather data from more sources than current surveillance

systems. The agent will perform the processing of data records to derive counts to be delivered to the data collection host which then may be used for visualization or trend analysis. Seeing the trends of disease factors is useful to update the agents and improve the agent's ability to accurately detect outbreaks. Such improvements may decrease the number of false positives and decrease the costs to health organizations in reacting to the false alarms.

3.3. Data Agents

Data agents are agents that travel the network looking for a specific factors of disease. These factors have been previously described as the symptoms presented by patients, over the counter medication sales, laboratory tests ordered, or work absentees. A data agent will collect data one factor such as those patients with a chief complaint of a cough. Statistical analysis is used to detect trends in the data collected by the agent as well as abnormal trends.

When a data agent visits a host, the agent will request access to the host's database will search for data records matching the factor the agent is monitoring. For example an agent looking for the sales of over the counter allergy medicine would search a pharmacy's database for records that include the count of over the counter (OTC) allergy medication sales. The data collected by the agent should be updated with an indication that it has been processed to prevent the agent from double counting. Likewise, other agents can use the indication if only one agent is expected to collect the data

3.3.1. Agent Movement

When an agent is ready to move, it will require a list of hosts to move to. Agents could be configured to know about all hosts in the network when being injected into the network, but this requires that the agents carry large amounts of data for larger networks. A better mechanism is for each host to store a list of remote hosts for the agents to use. The list contains the number of agents that have travelled to the remote host, and the agents will select the remote host that has the smallest agent count and is not the host where the agent travelled from. This scheme will improve the chances that all hosts within the network will be visited. However, this relies on the remote host lists being setup such that all hosts appear on a remote host list. If each host appears the same

number of times on various remote host lists, then each host in the network will have the same probability of being visited.

The following example illustrates how each host in a remote host list will be visited. A host contains a list of three other hosts the agent will select from, and the agent count for a remote host is initialized to zero when added to the list (see initial host counts in Table 3.1). An agent travels from the remote host 1 to the current host. As each host has the same count the agent will choose randomly between the remote hosts 2 and 3. Host 1 is not included as it is the agent's previous host. The agent chooses the second host, and the agent count for that host is incremented (see after first agent in Table 3.1). A second agent visits the host from the remote host 3 and will select from remote hosts 1 and 2. Host 1 will be selected as it has the smallest agent count. After the second agent, remote hosts 1 and 2 will agent counts of 1. Then a third agent moves from a host not in the list. The agent will select from all three remote hosts choosing host 3 with the smallest agent count. As all hosts in list have the same counts, all agent counts in the list should be set to zero .

The movement paradigm just described depicts how an agent will move through out the entire agent network. When an agent begins to detect a possible outbreak the agent's movement should be localized to the host where the first increase in the symptom was found. As outbreaks tend to be localized, the modified agent movement will improve the probability of the agent to detect the outbreak. The agents will require a means for determining which hosts are local to the current host. A host's list of remote hosts should also include the distance to the remote hosts which agents will to ensure it does not move to far away from the initial host. The distance allowed by the agent should be sufficiently large to allow for the fact that the initial host is on the edge of the epidemic area.

3.3.2. Decision Model

To detect an outbreak an agent must learn the normal patient arrival rate, but the daily arrival counts will vary too greatly to be used for outbreak detection. Averages of the daily arrival counts will be used to smooth the counts as the more historical data the average is based on the less the average will be susceptible to large variances in the daily arrival counts. Figure 3.5 shows two averages for a single set of arrival counts. A running average is calculated as

Remote host	Agent Count
Initial host Counts	
1	0
2	0
3	0
After first agent	
1	0
2	1
3	0
After second agent	
1	1
2	1
3	0
After third agent	
1	1
2	1
3	1

TABLE 3.1. Number of agents transfered to a remote host.

$$avg_{run} = TPC/TAR$$

TPC is the total patient count which is the sum of all patients the agent has counted and TAR is the total agent runtime in days. Both values are based on the entire history of the agent and will grow very large the longer the agent collects patient counts. Hence, an implementation of the running average will have to take integer overflow into account as the central processing unit (CPU) registers will not be able to hold a value larger then the maximum integer value for the largest integer type.

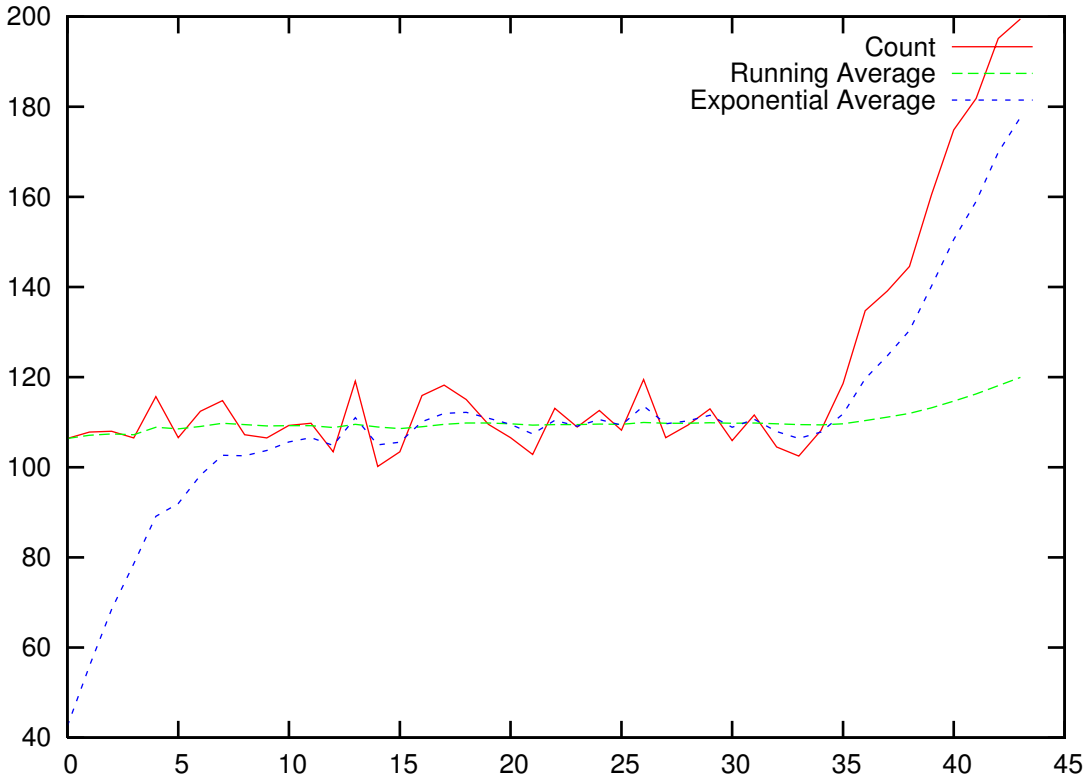

FIGURE 3.5. Comparison of the running and exponential averages.

An exponential average will be more responsive to the changes in the patient counts and is based on the equation

$$avg_{exp}[t] = PC * 0.4 + avg_{exp}[t-1] * 0.3 + avg_{exp}[t-2] * 0.2 + avg_{exp}[t-3] * 0.1$$

where Avg is the exponential average and PC is the patient count at time t.

One caveat with this approach is seen in Figure 3.5 where the exponential average takes time to reach the expected average. Although it is not visible in the graph, the running average also started lower than the expected average but took less time to reach it. During this time, the averages will have a large difference which must not raise any alarms. The agent will detect that the variances of the average has stabilized when the change in both averages fall below the individual thresholds $\frac{davg_{exp}}{dt} < \delta_{exponential}$ and $\frac{davg_{run}}{dt} < \delta_{running}$. However, if the patient arrival counts vary by a large amounts the change in the averages may also fluctuate by large amounts. Table 3.2 shows a large change in the running average of 8.500 preceded by small changes where those small changes may

Count	Running Average	Change in Running Average	Exponential Average	Change in Exponential Average
1031	1031.000		412.400	
1039	1035.000	4.000	539.320	126.920
1040	1036.667	1.667	660.276	120.956
1032	1035.500	1.167	759.987	99.711
1078	1044.000	8.500	845.183	85.196
1032	1042.000	2.000	884.380	39.197
1062	1044.857	2.857	935.149	50.769
1073	1048.375	3.518	971.129	35.990
1036	1047.000	1.375	981.210	10.070
1032	1045.500	1.500	994.906	13.696
1046	1045.545	0.045	1010.228	15.322

TABLE 3.2. Changes in running and exponential averages.

fall below the threshold $\delta_{running}$. Hence it will be necessary to require the change in the average to fall below δ for 2 or 3 consecutive time periods.

When the patient counts are stable, the difference between the averages is small. However when the patient counts begin to increase the exponential average will increase more quickly than the running average. The difference between the two averages will be used to detect a possible outbreak when the difference exceeds the threshold γ. The agent will raise an alarm when $avg_{exp} - avg_{run} > \gamma$.

CHAPTER 4

SIMULATION AND EXPERIMENTAL ANALYSIS

The simulation will model agents moving to data sources that contain patient data. This will require a model to simulate arrival of patients at a physicians office, but before the patient arrival model is described the simulation design will be presented.

4.1. Simulation Design

An event driven simulation framework will be used to simulate the movement of agents and processing of data hosts. The main type of object in this framework are the events to be simulated which are scheduled at a specific time. Hence the simulation does not run in real time, but rather will skip time to the next scheduled event.

4.1.1. Simulation Components

An event driven simulation will require components that represent real world entities. This simulation will contain only three components.

- hosts
- patients
- agents

Host components will contain a list of patient components injected into the simulation, and each patient includes a list of symptoms. The agent components simulate the mobile agents that move between hosts processing the patient records. As the agent component processes the list of patients at a host, the agent will either remove the patient to prevent other agents from counting the patient or mark it to prevent the agent itself from double counting but allow other agents to count the patient.

4.1.2. Simulation Events

There are three event types that will be simulated.

- Inject Patient
- Agent Move
- Process Node

During initialization, the first patient's data is read from a file, and an inject patient event is schedule for the time specified in the file. When this event occurs the patient is added to the list of patients at the assigned host, and the next patient's data is read from the file to schedule the next inject patient event. The patient injection event will not be scheduled after the last patient in the file has been injected.

The agent move event will notify an agent component that it is ready to move to a new host. Agent movement for most experiments will be random amongst the hosts which implies a completely connected network of hosts. After the agent has moved to a new host, a process node event will be scheduled which will notify the agent to process the host's list of patients. Once the list has been process and the running and exponential averages have been updated, an alarm will be raised if the the threshold γ has been exceeded, and a new agent move event will be scheduled again to have the agent move to another host. Unlike patient injection where a file controls then the events are scheduled, agent move and process node events will be scheduled by the agent component at regular intervals of 1 minute.

4.2. Patient Injection

Patients are injected into the simulation using the Poisson distribution which models the probability that n events will occur within a time interval $[t_{k-1}, t_k], k = 1, 2, \ldots$. The inter-arrival time, t, of the events is exponentially distributed where $P(t) = 1 - e^{-\lambda t}$ [7]. A formula (1) may derived from the exponential distribution to randomly generate the inter-arrival times for patients.

$$t = \frac{-\ln(1 - U)}{\lambda} \tag{1}$$

U is a random real number uniformly distributed from 0 to 1, and λ is the average arrival rate of patients. A typical arrival rate at a single host might be 1 patient every thirty minutes or 48 patients per day which gives us $\lambda = 1/30 \approx 0.033$ using minutes at the time unit. To generate patient arrivals across multiple hosts, the value for λ is adjusted by the number of hosts. For example if there are 1000 hosts, then $\lambda = 1/30 * 1000 \approx 33.33$. Patients are then randomly assigned to hosts as they arrive.

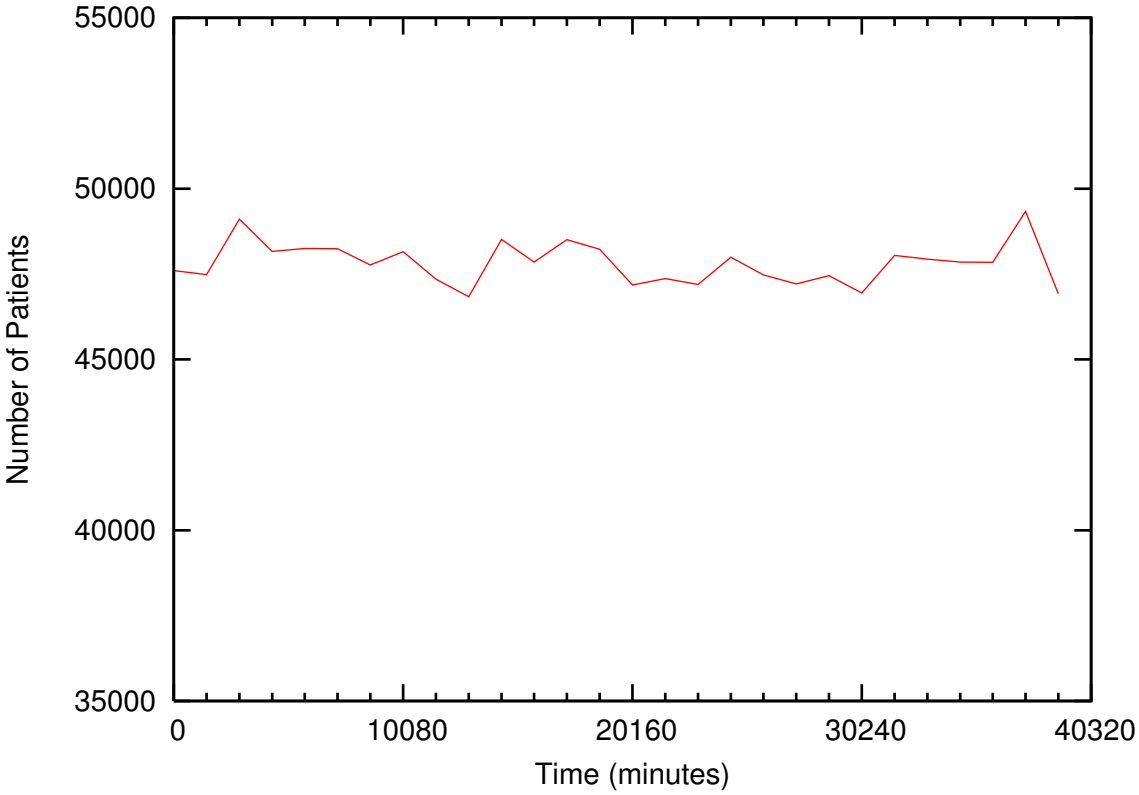

FIGURE 4.1. Daily patient arrival counts.

In a syndromic surveillance system, agents will be searching for patients with a particular symptom. Assuming each patient has a set of symptoms $S = \{s_1, s_2, ...\}$ and s_s is the symptom the agents will be searching for, then $P(s_s \in S) = \alpha$.

Using 33.33 as the value for λ the expected average daily arrival count or λ_{daily} is 48,000 patients. Figure 4.1 shows that the daily patient arrival counts from data generated using equation (1) are centered around the expected λ_{daily} value.

To simulate an epidemic, the value for λ is increased for the hosts where the epidemic is to occur. However the arrival rate during an epidemic does not simply increase to a new value one time. Rather the arrival rate increases over time until a maximum value is reached, and then the average arrival rate will decrease as the epidemic abates. For the purposes of the simulations, the increase of the arrival rate will be simulated as we are interested in how quickly agents detect an increase in the arrival rate. Hence λ will be increased over the span of the simulated epidemic, but will not be decrease. This is accomplished by splitting the epidemic into time intervals, and the

Model	λ_{normal}	Value for c	Interval size	α
1	48 per day	no epidemic	no epidemic	5%
2	48 per day	Increase by 1 patient per day	2 days	5%
3	48 per day	Increase by 0.5 patient per day	1 day	5%
4	48 per day	Increase by 1 patient per day	1 day	5%

TABLE 4.1. Epidemic models.

value for λ is increased for each interval. The arrival rate function during the epidemic is defined as

$$\lambda_i = \lambda_{normal} + 2^i * c$$

where $i = 1...m$, m is the number of intervals, and c is a constant value which controls $d\lambda/dt$. The value of c is an increment value to the arrival rate, the interval factor, 2^i, will cause an exponential increase. The simulations in this chapter will use four epidemic models using two values for c. The time unit used for c should match the time unit used for λ above which is days. To increase the patient arrive by by 1 patient per day $c = 1$. To increase the patient arrival by 0.5 patient per day $c = 0.5$. The four epidemic models are shown in Table 4.1.

Table 4.2 shows the average daily arrival rates that will be used with four simulation models. The first model does not produce an epidemic, and the remaining models produce epidemics with increasing degrees of severity. Daily patient counts of patients generated using the four models are shown in Figure 4.2. Prior to the epidemic starting in week 3, the counts for all models average 48,000 patients per day for all 1000 hosts which equals the expected value $\lambda_{daily} * 1000$.

4.3. No Epidemic Experiment

The first experiment shows how data agents perform with no epidemic (epidemic model 1). The simulation includes 1000 hosts, and each host is connected to all other hosts. As a single agent is able to move freely to and from any host, the agent's expected daily patient count is $48 * 1000 * 0.05 = 2400$ for symptom.

Day	Model 1		Model 2		Model 3		Model 4	
	Rate Inc	λ_{daily}	Rate Inc	λ_{daily}	Rate Inc	λ_{daily}	Rate Inc	λ_{daily}
1 to n	0	48	0	48	0	48	0	48
n + 1	0	48	1	49	0.5	48.5	1	49
n + 2	0	48	1	49	1.0	49.0	2	50
n + 3	0	48	2	50	2.0	50.0	4	52
n + 4	0	48	2	50	4.0	52.0	8	56
n + 5	0	48	4	52	8.0	56.0	16	64
n + 6	0	48	4	52	16.0	64.0	32	80
n + 7	0	48	8	56	32.0	80.0	64	112

TABLE 4.2. Daily patient arrival rates for four epidemic models.

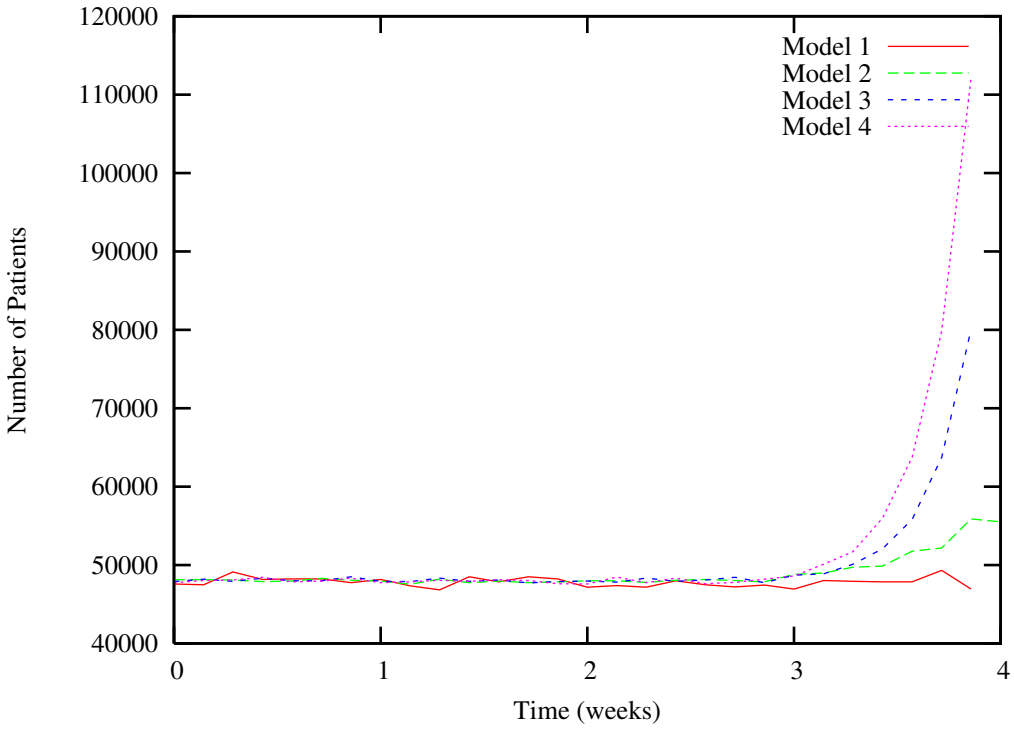

FIGURE 4.2. Daily patient counts for epidemic models.

Figure 4.3 shows the running and exponential averages calculated by the agent. As expected the running average has a small variance but falls short of the expected value 2400 due to the

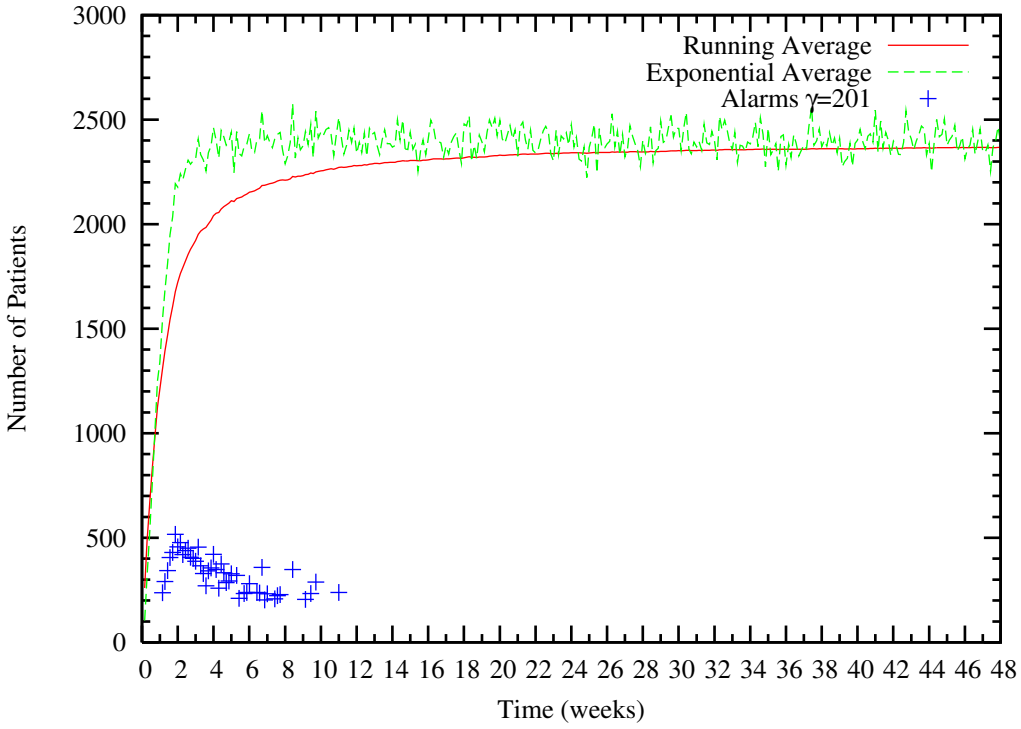

FIGURE 4.3. Patient arrival averages for single agent and no epidemic.

small initial daily counts collected at the beginning of the simulation. These initial values will prevent the running average from reaching the expected value. This is important to note because the exponential average which uses a limited amount of historical information is able to reach the expected value. Hence the small initial values are aged out of the average calculation. The fact that the current daily count is given the most weight causes the larger variance seen in the exponential average.

The averages should increase when an epidemic occurs, but the exponential average should increase at a faster rate than the running average. The threshold for the difference between the averages defined as γ in Section 3.3.2 should be larger than the differences seen when no epidemic is occurring. Thresholds set slightly higher than the highest average difference should be chosen to reduce the risk of false positives. The drawback of higher thresholds is the agent's sensitivity to increases in the average differences will diminish which will result in the agent taking longer to detect possible outbreaks. To properly set the γ threshold requires analyzing the risks of false

Single Symptom Agent		Two Symptom Agents	
Week	Difference	Week	Difference
16	187	20	125
18	195	20	104
37	200	42	104

TABLE 4.3. Top average differences.

positives compared to the risk of longer response times to outbreaks. This analysis is beyond the scope of this Thesis.

Given the top differences in Table 4.3 the threshold for a single agent searching for a symptom would be set to 201. The alarms for this threshold are shown in Figure 4.3. Alarms occur during the beginning of the simulation where the averages may be considered unstable, but the δ thresholds may be used to prevent the false alarms. The $\frac{dx}{dt}$ values for the exponential average during the ramp-up period are not larger that the values later in the simulation. In fact the largest value of 405 .39 was seen day 286 during week 40. This is well beyond the time where the average appears to have stabilized in Figure 4.3. However, it is clear that the running average will reach stability after the exponential average and the largest $\frac{dx}{dt}$ values are seen during the ramp-up period. All $\frac{dx}{dt}$ after day 66 (week 9) are below 10, but there are average differences larger than the chosen γ until week 12. To prevent these false alarms, the value for δ_{run} will be set to 0 and must be met two times which will prevent the agent from raising alarms until day 107 (week 15).

If multiple agents searching for the same symptom are introduced into the system and the agents do not double count the same patient, the daily patient counts of the agents will decrease. Table 4.3 also shows the top average differences for two agents, and based on the differences the threshold for two agents would be set to 126. An appropriate value for δ_{run} will also have to be chosen from the data for both agents.

	Model 2	Model 3	Model 4
One Agent per Symptom			
Agent 0	7 days	5 days	3 days
Two Agents per Symptom			
Agent 0	6 days	5 days	4 days
Agent 1	7 days	5 days	5 days

TABLE 4.4. Agent response times.

4.4. Epidemic Experiments

Using the threshold determined above, six experiments were run with one and two agents using the epidemic models 2, 3, and 4 given in Table 4.2. The simulation includes 1000 hosts, and the epidemic occurs simultaneously at all hosts while the agents move at a rate of 1 minute per host.

Table 4.4 shows the number of days before a single agent exceeded the γ threshold of 120. As expected the agent was able to detect faster growing epidemics more quickly than the slower growing epidemics, but increasing the number of agents did not improve the response time. With two agents moving at the same rate and same probability of moving to any host in the network, it is expected that each agent will count approximately 50% of the patients injected in a day. Thus, neither agent will have an advantage in detecting the outbreak.

A single agent moves from host to host at 1 host per minute or 1440 hosts per day. An agent will visit most if not all of the 1000 hosts and will count most if not all of the patients injected during that day. Hence more agents would not provide any advantage over a single agent. Even if the rate movement is slowed to once every 10 minutes, multiple agents may still not show any improvement. In this case a single agent may visit at most 144 hosts in a day and will not be able to count all of the patients injected into the simulation within a day. However, over time the agent will reach a daily patient count centered around the average rate of patient injection (See Figure 4.4). Assuming the agent visits 144 unique hosts after the first day d_0, there will be 856 hosts that were not visited each containing one days worth of uncounted patients. On day d_1 the agent should visit about 123 of the 856 hosts not visited on day d_0 as each host has a 14.4% chance of being visited

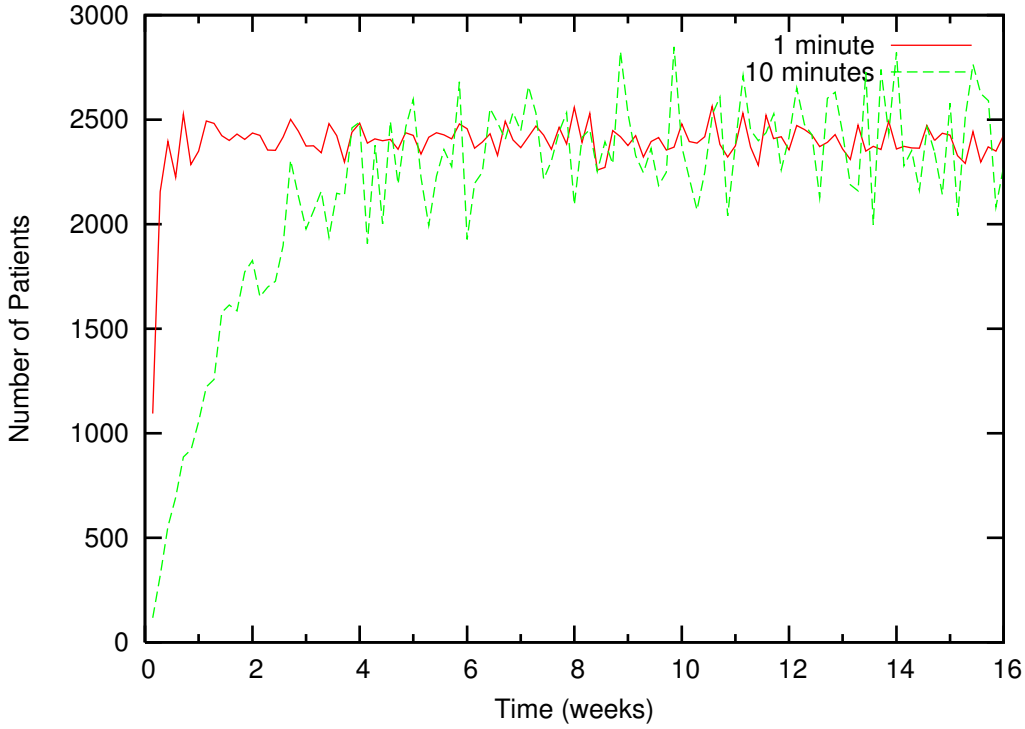

FIGURE 4.4. Daily patient counts for agents moving at different rates.

in a day. For each host the agent will count a full days worth of patients for the previous day as well as patients already injected for the current day. After day d_1 there will be about 733 hosts the agent has not yet visited each with two days worth of uncounted patients. Then on day d_2 the agent should visit about 105 of the hosts not visited in the past two days and will count two days worth of patients or more. This trend will continue until all hosts have been visited which may take up to 37 days. When the agent visits the last unvisited host there will be 37 days worth of patients to count. Each of these larger patient counts continually increase the agent's daily patient count over time until the the last unvisited host is visited where the daily counts should be centered around the expected value.

Daily patient count have a larger variance with 144 hosts per day compared with 1440 hosts per day. If in one day an agent visits more hosts that have not been visited in a long time, the daily count will be high relative to if the agent visits more hosts that have been recently visited. This will lead to larger variances in the exponential average even without an epidemic which will produce larger differences between the averages. Hence, a new value for the threshold γ will be required.

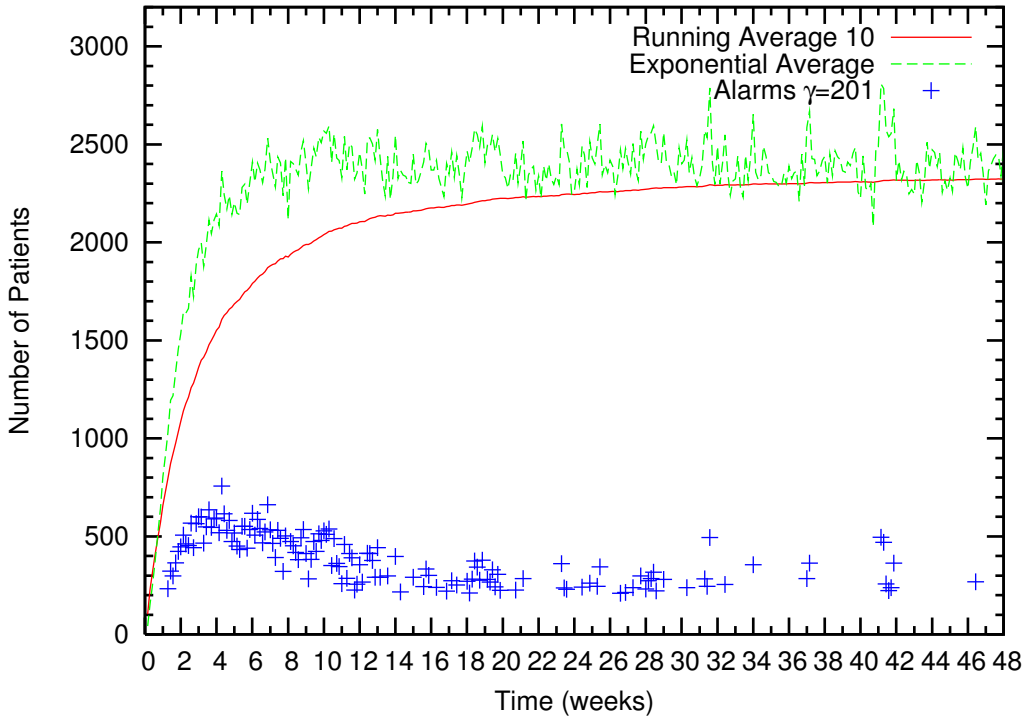

FIGURE 4.5. Averages for model 1 and agents moving every 10 minutes.

Figure 4.5 shows the false alarms if the threshold is kept at the original value of 201. The number of false alarms decrease as the running average increases closer the real average, but there are still too many. Another side effect of the slower agent movement is the running average takes longer to approach to the expected value. When the agent was moving once per minute the running average reached 2300 after about 19 weeks whereas the agent moving once per 10 minutes reached 2300 after about 40 weeks. Many false alarms will be generated with the original γ value especially during the ramp up time of the running average (see Figure 4.5).

4.5. Localized Epidemic

It is reasonable to expect that an epidemic will not occur at all hosts at the same time. Rather, the epidemic will start at a single or a small number of hosts and spread to other locations over time or not at all. In the next experiment, the epidemic will be localized to 50 of the 1000 hosts. The patient arrival rate at the remaining hosts will remain at λ_{normal} during the epidemic and λ will be increased at the 50 epidemic hosts. An additional change is included where multiple agents searching for the same symptom will count the same patients. In other words, patients will not be

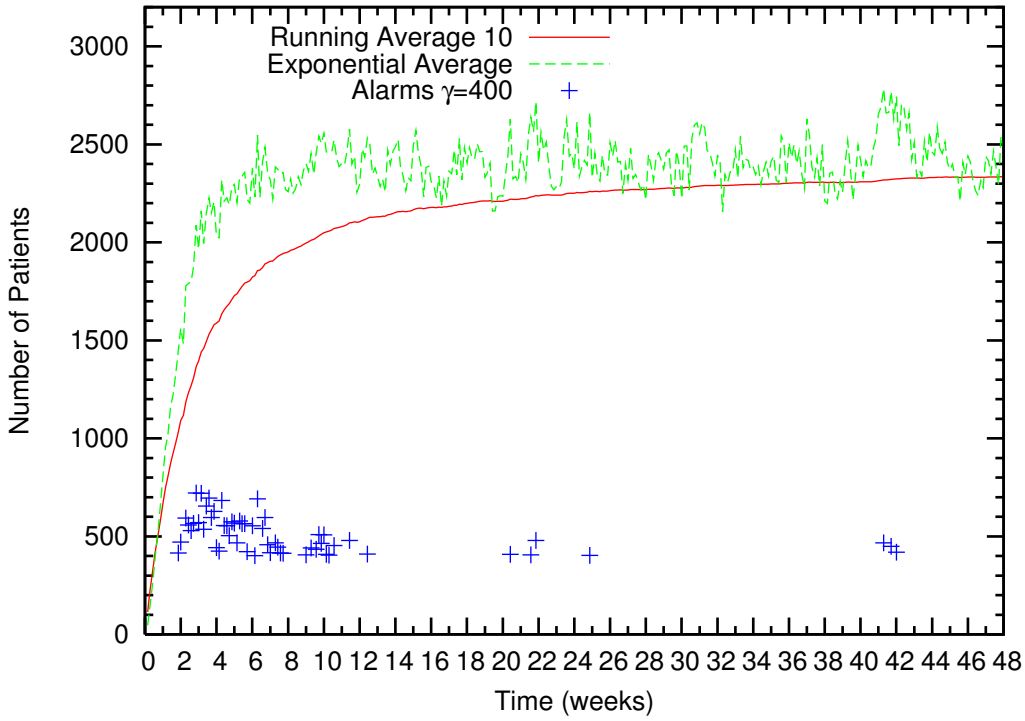

FIGURE 4.6. Averages for model 3 and agents moving every 10 minutes.

marked as counted per symptom which prevents other agents from counting this patient. Rather, the patients will be marked per per agent which will allow each agent to count the patient. If one of the agents were to visit every epidemic host, the agent may be able to detect an increase in the daily arrival counts if the epidemic occurs at a large enough rate or number of locations.

As seen in Table 4.5 a large number of agent will not detect an increase in the difference between the exponential and running average more quickly than a smaller agent population. The largest difference was seen with 500 agents which is a very large agent population for 1000 hosts. To understand why large agent populations do not see a large average difference, one must take into account the probability that a single agent will visit all epidemic hosts one right after the other. Overall there are $1000!$ possible paths an agent may follow, and $50!$ paths amongst the epidemic hosts. The probability an agent will visit the epidemic hosts in succession is $50!/1000!$, and the probability will increase as the agent population increases. However, an extremely large agent population would be required raise the probability high enough that one of the $50!$ paths would be traversed by an agent. Even if an agent were to visit the epidemic hosts, the patient arrival counts,

Number of Agents	Day	Average Difference
10	67	359
20	62	342
30	73	386
50	43	356
100	52	388
500	69	410

TABLE 4.5. Largest average difference for multiple agent population sizes.

although higher than the current averages, may not be large enough to increase the exponential average to where the γ threshold is exceeded. Given an agent will visit 144 hosts per day, the 50 epidemic hosts will only account for about one third of the patient arrival counts collected that day.

When the epidemic is created at all hosts, the exponential average will increase enough to detect a possible outbreak. The same results should be achievable if the agent were to limit its movement to the epidemic hosts during a localized the epidemic. However, the average daily patient count of 2400 is based on all 1000 hosts. Therefor, the current averages must be normalized for the decreased number of hosts to be visited. To accomplish this the agent must know the number of hosts the current averages are based on, n_o, as well as the number of local hosts to be visited, n_l. The normalized average is calculated using the equation $avg_{norm} = \frac{avg_{orig}}{n_o} * n_l$ which may be used to adjust both the running and exponential averages.

The ability to detect when to change the itinerary to localized movement as well as when to resume movement amongst all hosts is left for future work. For our purposes, an epidemic using model 3 will be created at epidemic hosts 1 through 50 during week 40, and when an agents visited one of the epidemic hosts the agent will move randomly between between the 50 hosts. The new expected daily arrival value based on 50 hosts is 120 patient per day.

Testing shows the exponential average does not normalize as expected. Instead the exponential average, on the first day of localized movement, starts with an average larger than the expected

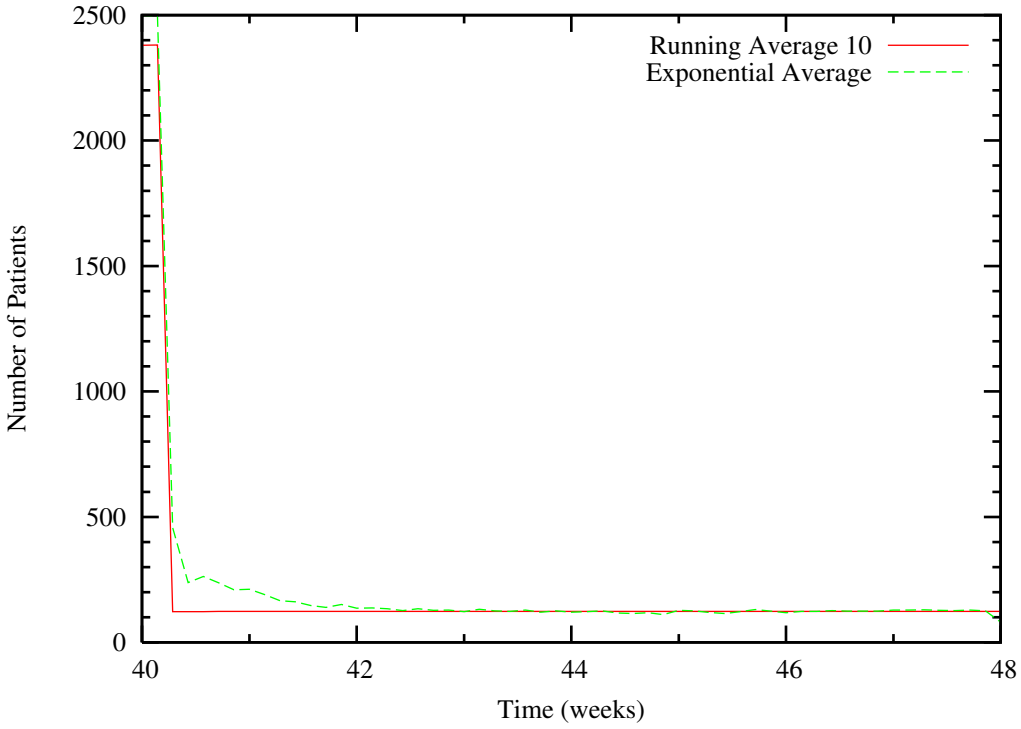

FIGURE 4.7. Local agent movement and epidemic model 3 during week 40.

value and decreases for about 2 week before the expected value is reached (see Figure 4.7). Most likely, an agent will make the decision to begin localized movement in the middle of the day. For the first part of the day where the agent is moving amongst all 1000 hosts, the agent will have collected may daily patient counts larger than the normalized average which contributes to the larger exponential average. A second factor is the local hosts containing large patients counts as these hosts may not have been visited for over 37 days as was describe previously. Each of the local hosts will be visited during the first full day of local movement if not the first day and will collect the large patient counts. Afterwards, all hosts are guaranteed to be visited multiple time every day which is why the exponential average is able to stabilize around the expected value. Also observed in the Figure is the variance of the exponential average is relatively small. The same result was seen in Figure 4.3 where the agents were visited 1440 hosts per day.

CHAPTER 5

SUMMARY AND FUTURE WORK

5.1. Summary

The proposed mobile agent system design shows how mobile agents may be used to traverse sources of syndromic data. A possible agent decision model that to detect changes in patient arrival rates over a set of patient data sources is also presented. A key design issue for a syndromic surveillance system is compliance with the *Health Insurance Portability and Accountability Act Privacy Rules* which is accomplished in the mobile agent system by the fact that patient data is kept only at the sources of the data and agents contain the results of patient data analysis. The experiments in the previous chapter show that a mobile agent based system is able to detect an increase in the occurrence of a symptom by using daily averages of the number of patients who have the symptom. One factor that may inhibit and agent's ability to detect an outbreak is the speed at which an agent is able to move through the agent network. For example, when agents were limited to visiting 144 of 1000 hosts per day, the variance in the exponential average increased making it more difficult to find a γ threshold that will not cause a large number of false positives. The number of agents searching for the same data did not show any improvement over a single agent in the amount of time when an epidemic began to when the agents raised the first alarm.

The primary contributions of this thesis are

- Design of an Agent Based Syndromic Surveillance system
- Described how an agent system may provide more privacy for HIPAA protected patient information
- Identified syndrome detection issues that may affect the agent based system
- Described an agent movement scheme to ensure that all hosts are visited
- Proposed an agent decision model based on the running average and exponential average of daily patient arrival counts
- Presented design of an event driven simulation for the agent based system
- Described how patients to be injected into the simulation are generated

- Experimented with agent responses to no epidemic and 3 epidemics with increasing degrees of severity
- Described the effects of the ramp-up threshold δ and the epidemic threshold γ
- Compared daily patient counts collected by agents moving at different rates
- Experimented with agent responses to a localized epidemics i.e. an epidemic occurring at a subset of the hosts the agent will visit
- Experimented with changing the agent movement to a sub-set of hosts when a condition is met

5.2. Future Work

5.2.1. Agent Alarm Handling

When an agent detects an alarmable condition in the data, the agent will have to deliver the alarm to a system that is monitored by health organizations. The issue with alarms is not necessarily how to deliver the alarms, but rather how the alarms are handled by the receiving system and the personnel who are monitoring the alarms. A single alarm in and of itself may not be an actionable event requiring the resources of the health organization in conducting an investigation. However, methods of analyzing the rate of the alarms raised by agents monitoring different types of syndromic data may be developed to provide alarm severity levels. A person monitoring the alarms may then make more informed decisions regarding the action to be taken.

5.2.2. Dynamic Agent Populations

Two agents searching for the same symptom and visiting the same set of hosts do not detect an outbreak any faster than a single agent. However when an agent begins to move local relative to a host where an increase of patients was detected, the agent may spawn new agents each with a unique set of hosts local to the originating host to cover a larger area. This will only be useful given cardinality of the host set for each agent is relatively small and each new agent is seeded with the normalized averages discussed in the previous chapter. Host sets may also overlap to counter the possibility that an epidemic area spans two or more disjoint host sets such that each set does not contain enough of the epidemic to cause an alarm.

5.2.3. Disease Agents

Endemic diseases such as influenza are monitored by health organizations to track the various strains of the virus for new strains and to track the efficacy of vaccinations. The agent based system may also track diseases by searching for patients who have been diagnosed with the disease and like the symptom agents can detect irregular increases in the occurrence of the disease and raise alarms when a threshold is exceeded. However not every patient may have been properly diagnosed thus preventing the patient from being included in the disease agent's analysis. The agents, symptom and disease agents alike, are not limited to detecting increases of occurrence. The agents may be programmed with mathematical models to determine the probably of an epidemic for the disease being monitored, and probability thresholds used to decide when to raise alarms. Bayesian networks have been used to build probability models for disease surveillance [1].

5.2.4. Aging of Syndromic Data

Data in the agent databases cannot be stored for an indefinite period of time as the databases size requirements would grow unbounded. Methods should be developed for determining when the data may be removed without adversely affecting the ability of an mobile agent based system to detect possible outbreaks. The simplest method of aging is to remove the data after a set period of time, but it may be possible to remove the data when it is known the data is no longer required. For example, if it is expected that a patient's list of symptoms need only be counted by a single agent, then as each symptom agent type processes the patient record the symptom may be removed the list. Once all symptoms have been removed the last agent may removed the patient record from the database. A new agent type may also be created to visit data nodes and analyze the database records to determine which records will not, when removed, impact the statistical models of the other agents.

5.3. Conclusion

Syndromic surveillance systems collect data from many sources and analyze the data for indications of possible outbreaks. With the continued increase in international travel the need to

analyze syndromic data across larger geographical areas such as whole continents becomes increasingly important. Current syndromic surveillance systems are limited to small geographical areas such as municipalities. One limiting factor is the use of centralized databases which would require considerable amounts of hard drive storage to accommodate health related data from an entire state or country. An agent based system would provide for a decentralized syndromic surveillance system where the health data remains at the sources of data, and the analysis of the health data is performed at the source thus creating a scalable system that can process health data for large geographical areas.

REFERENCES

[1] K. Abbas, A. Mikler et al. “Computational epidemiology: Bayesian disease surveillance” *Proc. of the International Conference on Bioinformatics and its Applications* (Dec 2004)

[2] Kaizar A. Amin, John T. Mayes et al. “Agent-based distance vector routing” *Proceedings of the Third International Workshop on Mobile Agents for Telecommunication Applications* 2164 (2001): 41–50

[3] Hopper Andy, Mikler Armin et al. “Design and implementation of a distributed agent delivery system” *Proceedings of the Third International Workshop on Distributed Communities on the Web* 1830 (2000): 192–201

[4] Ruth Berkelman, Ralph Bryan et al. “Infectious disease surveillance: A crumbling foundation” *Science* 264 (1994): 368–370

[5] N. Borselius “Mobile agent security” *Electronics and Communication Engineering Journal* 14, no. 5 (2002): 211–218

[6] James Buehler, Ruth Berkelman et al. “Syndromic surveillance and bioterrorism-related epidemics” *Emerging Infectious Diseases* 9, no. 10 (2003): 1197–1204

[7] Christos Cassandras *Discrete Event Systems: Modeling and Performance Analysis* Irwin, 1993

[8] Jeremy Cherfas *The Human Genome* Essential Science DK Publishing, 2002

[9] Office for Civil Rights “Summary of the hipaa privacy rule” Available from http://www.hhs.gov/ocr/hipaa/ accessed September 2007

[10] Center for Disease Control “National notifiable diseases surveillance system: History” Available from http://www.cdc.gov/epo/dphsi/nndsshis.htm accessed September 2007

[11] Hy A. Dwosh, Harry H.L. Hong et al. “Identification and containment of an outbreak of sars in a community hospital” *Canadian Medical Association Journal* 168, no. 11 (2003): 1415–1420

[12] J. Espino and M. Wagner “Accuracy of the icd-9-coded chief complaints and diagnosis codes for the detection of acute respiratory illness” *Proceedings AMIA Symposium* (2001): 164–168

[13] WM Farmer, JD Guttman et al. “Security for mobile agents: Issues and requirements” *Proceedings 19th National Information System Security Conference* (October 1996): 591–97

[14] Per Gesteland, Reed Gardner et al. “Automated syndromic surveillance for the 2002 winter olympics” *Journal of the American Medical Informatics Association* 10, no. 6 (2003): 547–554

[15] Michael S Greenberg, Jennifer C Byington et al. “Mobile agents and security” *IEEE Communications* 36, no. 7 (1998): 76–85

[16] T. Habtemariam, B. Tameru et al. “Epidemiologic modelling of hiv and cd4 cellular/molecular population dynamics” *Kybernetes* 31, no. 9/10 (2002): 1369 – 1379

[17] B Harris and R Hunt “Tcp/ip security threats and attack methods” *Computer Communications* 22, no. 10 (1999): 885–897

[18] Sandholm T. Huai “Nomad: mobile agent system for an internet-based auction house” *Internet Computing, IEEE* 4, no. 2 (2000): 80–86

[19] O Ivanov, M Wagner et al. “Accuracy of three classifiers of acute gastrointestinal syndrome for syndromic surveillance” *Proceedings AMIA Symposium* (2002): 345–349

[20] Neeran M. Karnik and Anand R. Tripathi “Design issues in mobile-agent programming systems” *IEEE Concurancy* 6, no. 3 (1998): 52–61

[21] Ching Lin, Vijay Varadharajan et al. “Trust enhanced security for mobile agents” *E-Commerce Technology, 2005. CEC 2005. Seventh IEEE International Conference on* 19, no. 22 (2005): 231– 238

[22] Joseph S. Lombardo, H. Burkom et al. “Essence ii and the framework for evaluating syndromic surveillance systems” *Morbidity and Mortality Weekly Report* 53(Suppl) (2004): 159–165

[23] Last John M, Spasoff Robert A et al. *A Dictionary of Epidemiology* Oxford University Press, 1988

[24] Kenneth Mandl, Marc Overhage et al. “Implementing syndromic surveillance: A practical guide informed by the early experience” *Journal of the American Medical Informatics Association* 11, no. 2 (2004): 141–150

[25] H. Peine and T. Stolpmann “The architecture of the ara platform for mobile agents” *Proceed-ing of the First International Workshop on Mobile Agents* 1219 (1997): 50–61

[26] Vu Anh Pham and Karmouch A “Mobile software agents:an overview” *Communications Magazine, IEEE* 36, no. 7 (1998): 26–37

[27] Subhashini Raghunathan *A Security Model for Mobile Agent Environments using X509 Proxy Certificates* Master’s thesis University of North Texas Denton, TX (2002)

[28] K Schafer “Leaders (lightweight epidemiology advanced detection and emergency response system), presentation w203” Available from http://www.tricare.osd.mil/conferences/2001/agenda.cfm Accessed September 2007

[29] Michael A. Stoto, Matthias Schonlau et al. “Syndromic surveillance: Is it worth the effort?” *Chance* 17, no. 1 (2004): 19–24

[30] Fu-Chiang Tsui, Jeremy U. Espino MD et al. “Technical description of rods: A real-time public health surveillance system” *Journal of America Medical Information Association* 10, no. 5 (2003): 399–408

[31] Sangeeta Venkatachalam and Armin Mikler “Towards computational epidemiology: Using stochastic cellular automata in modeling spread of diseases” *Proceedings of the 4th Annual International Conference on Statistics, Mathematics and Related Fields*

[32] D. Wong, N. Paciorek et al. “Concordia: An infrastructure for collaborating mobile agents” *Proceeding of the First International Workshop on Mobile Agents* 1219 (1997): 86–97

[33] A Zelicoff, J Brillman et al. “The rapid syndrome validation project (rsvp)” *Proceedings AMIA Symposium* (2001): 771–775