SenseSpace: Sensor Network Namespaces

Gurpreet Singh
gurpreet@andrew.cmu.edu

Carnegie Mellon University

Thesis Committee
Advisor: Prof. Raj Rajkumar
Reader: Anthony Rowe

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Information Networking (MSIN)

ABSTRACT

SenseSpace: Sensor Network Namespaces

by Gurpreet Singh

Advisor:	Prof. Raj Rajkumar
Reader:	Anthony Rowe

A sensor network consists of a variety of spatially distributed autonomous devices which utilize sensors to gather data about their environment. This data is typically transmitted to a remote location for storage and future processing. Sensor networks can be deployed in various physical environments, including those which are unsafe for humans to enter. We present a scheme for management of sensor nodes at a large scale through the use of namespaces. A namespace is an abstraction that captures various attributes and relationships between devices. In the sensor networking domain, this provides a means to locate and categorize different properties associated with sensor nodes like their location and what sensors the node provides.

In this thesis, we create a secure and highly distributed namespace for sensor networks. We provide support for application specific views of the network as well as access control for data protection. To provide scalability, the namespace exists on multiple physically distributed machines. Access to different regions of the namespace is arbitrated through hand-offs eventually connecting the client to the appropriate sensing resource.

Acknowledgements

I would like to thank my advisor Professor Rajkumar for his guidance and motivational support throughout the duration of my research. I would also like to express deep gratitude towards my reader Anthony Rowe for being patient enough in investing countless hours for discussing the design and implementation of the system with me right from this project's inception.

Table of Contents

List of Figures

1 Introduction

As technology continues to develop, we are able to pack more computing power in progressively smaller devices and at rapidly decreasing costs. These devices can be augmented with radios that consume low power, which can last for long periods of time in all kinds of environment. These devices provide information from places that would otherwise be inaccessible.

An example of such a network is Carnegie Mellon's Sensor Andrew, developed under the CenSCIR, *Center for Sensed Critical Infrastructure Research*. Deploying sensor networks is challenging because they are resource constrained, use a lossy wireless medium, and need to support heterogeneous devices and applications. In this thesis, we propose SenseSpace, a management scheme for sensor networks based on "namespaces."

1.1 Motivation

The scales at which sensor networks can be deployed make node addressing and attribute management a challenge. There is a strong need for some sort of abstraction that will make addressing in such a system more convenient. The main motivation behind designing SenseSpace is to come up with an abstraction through which different applications can intuitively access various distributed resources.

1.2 Goals

A namespace for sensor networks needs to be highly scalable and expandable to support newly added sensors. The system would have to allow for simple addressing in a manner that is easily usable in the context of multiple diverse sensor applications. It should provide an intuitive interface for human operators. This abstraction should be semi-aware of the underlying architecture of deployment of the network, so as to more efficiently return the correct results to the user. All these interactions would have to be secure to keep the sensed data private. The system would have to provide support for different types of users, through access control lists. Finally, this system would need to be aware of the attributes available at each sensor, in order to let the user directly access the information in real-time.

1.3 Related Work

This work draws from earlier contributions in namespaces such as file systems, the Domain Name System, and low level naming in sensor networks.

1.3.1 Directed Diffusion with Attribute Tuples

In [1], Heidemann et al, show a naming scheme that relies on the network being "task-aware." The nodes store and interpret results that pass through them instead of blindly forwarding the information through the network without processing the packet at each hop.

Each sensor node receives a sensor request in the form of a query. This query is used to set up a gradient which directs the information towards the parts of the network with values of interest. After setting up the gradient, the node redistributes the query to its neighbours. If a node can infer who the potential sources for the query may be, it can forward the query to that subset of neighbours. If the sources are unknown neighbours, it does a broadcast to all its neighbours. When a node that matches the requested query is found, it activates its local sensors, to collect data. This data is then propagated back along the network. As it propagates, the data is cached at the various nodes. Cached data is useful for multiple purposes, although its primary use is to prevent looping of requests. The cache can also be used for in-network processing, specific to the application being considered. This cache saves energy by allowing for in-network data reduction.

Directed diffusion shows us quite a few features of how the naming scheme and data access is different from traditional networking. All communication is data-centric, where queries are used to specify the data that each node requires. The communication that takes place on the network is hop-by-hop from one neighbour to the next. This can be justified given the nature of most sensing tasks.

In our system, we don't directly deal with sensor to sensor communication. A query traverses through the namespace on the name servers before finally reaching the sensor node which has the data required.

1.3.2 File Systems

[2] Shows an approach where the wireless sensor network is represented as a file system. This is similar to the concepts introduced in Plan 9 [3], an operating system characterized by its heavy use of a file system abstraction.

The file system representation of a network helps to capture the structure dependency of the network. It provides logical attributes that help in aggregation and grouping. The root of the file system abstraction, or the root directory, is the entry point into the entire network. Each sub-directory represents a cluster which can have sub-directories for more clusters, going all the way down to the sensor level. The sensors directory would provide direct access to the sensor itself, with one directory per sensor.

The task of locating and getting sensor values, in such an abstraction, is reduced to finding a path to the corresponding file in the namespace. The low level operations involved in reading from a file, are hidden away from the user by the more familiar file system interface. This allows applications which know how to access files to seamlessly interface with the network.

We build upon this scheme for our namespace. The namespace traversal is similar to traversing a directory structure spread across many servers. Depending on the location of a particular node, communication is set up with the corresponding name server.

1.3.3 Domain Name System

The domain name system (DNS) [4] is probably one of the most famous namespaces today. It consists of a tree of hostnames of machines on a large scale network such as the internet. The most basic task of the DNS is to translate these hostnames into IP addresses which makes it easier for computers to identify and talk to each other on the network. DNS allows us to assign universal names to entities on the network independent of the physical network upon which the entities are connected.

The domain name system consists of a hierarchical set of DNS servers. Each domain or sub-domain can have one or more DNS servers that publish information about that domain, and any domains beneath it. The DNS queries are handled by DNS resolvers. A resolver communicates with the name servers by sending DNS queries and receiving DNS responses.

Our system draws upon the lessons from a DNS based namespace. We have a hierarchical set of servers which resolve the queries from the user, and eventually connect them directly to a sensor node. We do not, however, utilize separate resolvers, but have the name servers perform the resolution of the address.

2 Architecture

The architecture of SenseSpace can be classified into two main components. These are the *Name Servers* that make up the main intelligence of the namespace, and the *Clients*, or the user nodes which contact the system to obtain information from the sensor network. Fig 1 shows the overall system architecture.

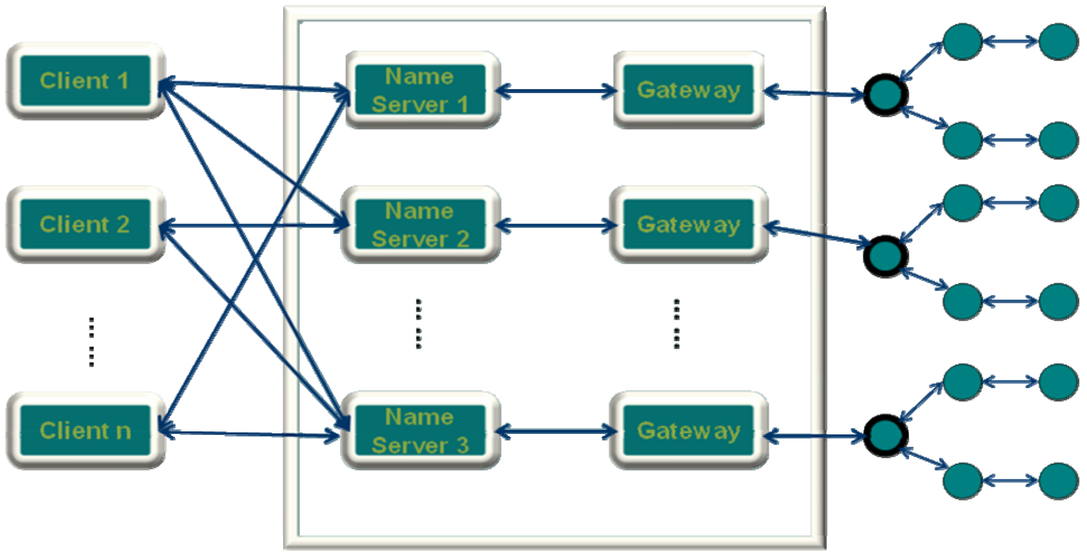


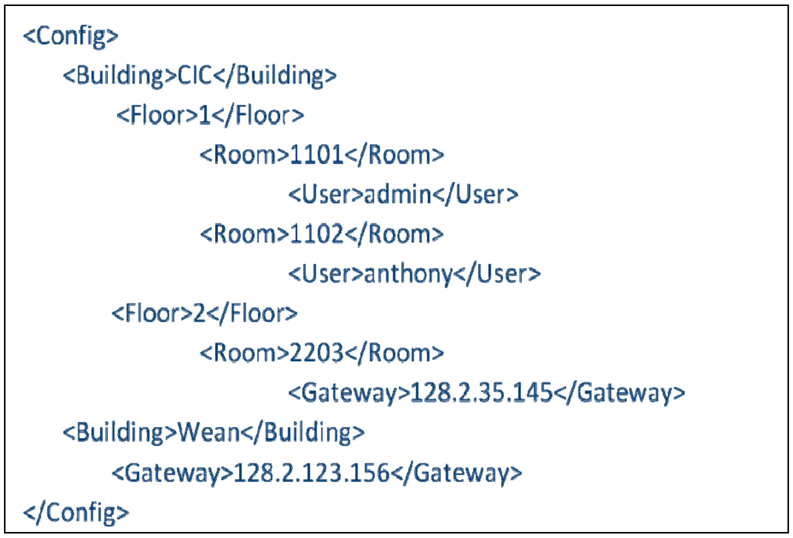
Figure 1: SenseSpace System Architecture

2.1 Name Servers

The SenseSpace name servers are where the bulk of the system's intelligence lies. These servers keep the information of the namespace, and use this information to service requests made by clients.

Given the large scale of sensor networks that can exist in the world, it would be infeasible to have every sensor node's availability information stored on every name server in the system. This poses many fundamental problems. A centralized system becomes extremely hard to manage.

Our system design relies on the underlying characteristic of the distributed nature of the system. Each name server keeps information only about parts of the system hierarchy which are closest in proximity. The namespace is built up from an XML based configuration file that is stored on each name server. These XML files are easily extendable and can easily scale. Fig 2 shows an example configuration file that is stored on one of the many name servers in SenseSpace.



```
<Config>
    <Building>CIC</Building>
        <Floor>1</Floor>
                <Room>1101</Room>
                        <User>admin</User>
                <Room>1102</Room>
                        <User>anthony</User>
        <Floor>2</Floor>
                <Room>2203</Room>
                        <Gateway>128.2.35.145</Gateway>
    <Building>Wean</Building>
        <Gateway>128.2.123.156</Gateway>
</Config>
```

Figure 2: Sample XML Configuration File

As can be seen from this configuration file, the name server which loads this will be aware of entities that exist in rooms *1101*, and *1102* of the first floor of *CIC*. The name server also stores information about the other name servers which would be aware of the other parts of the system. In this example, there are IP addresses which indicate how a request for *Wean* or for the *second floor of CIC* would have to be serviced if it reaches this name server.

Once the configuration file is loaded into the system a hierarchical tree is built and stored in the memory of the name server. Due to the distributed nature of the system, directory information at each node is small enough to fit in memory. Also, by keeping the namespace in memory, the name servers are able to rapidly handle the requests and service them.

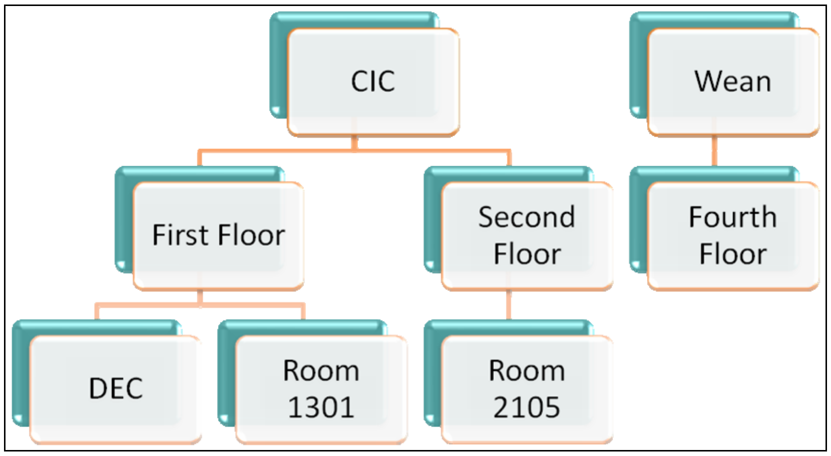


Figure 3: Internal Representation of the name space.

2.2 The Client

The other component in the system is the requesting client that is accessing data. The client is a lightweight front-end used for interacting with the servers. The clients will handoff connections to other servers based on the response it receives from the current name server.

3 Implementation

We chose to implement this system in C to make it portable and efficient.

3.1 Overview

For a namespace that will span a large scale campus wide sensor network, security is always a major concern. As a result, the system was developed taking into account security and access control. All communication that takes places in the namespace is done on an SSL channel. The certificates used for the establishing SSL were self signed for the time of this deployment.

When a client connects to a server, it is prompted to authenticate itself. Currently, we support three levels of access control in the system:

1. Administrator
2. Guest
3. Privileged User

Once the user is authenticated, they can begin sending requests to the system. Each request is processed at the name server, and if the query is a part of the subspace that the name server is aware of, a response is sent back. If further details are required, then the name server tells the client to continue with the request on another name server. This causes the client to break the connection with the first name server, connect to the new name server, and silently authenticate with this new server, before resending the request. All this happens in the background, and is seamless to the user. Figure 4 gives a diagrammatic representation of this sequence of events.

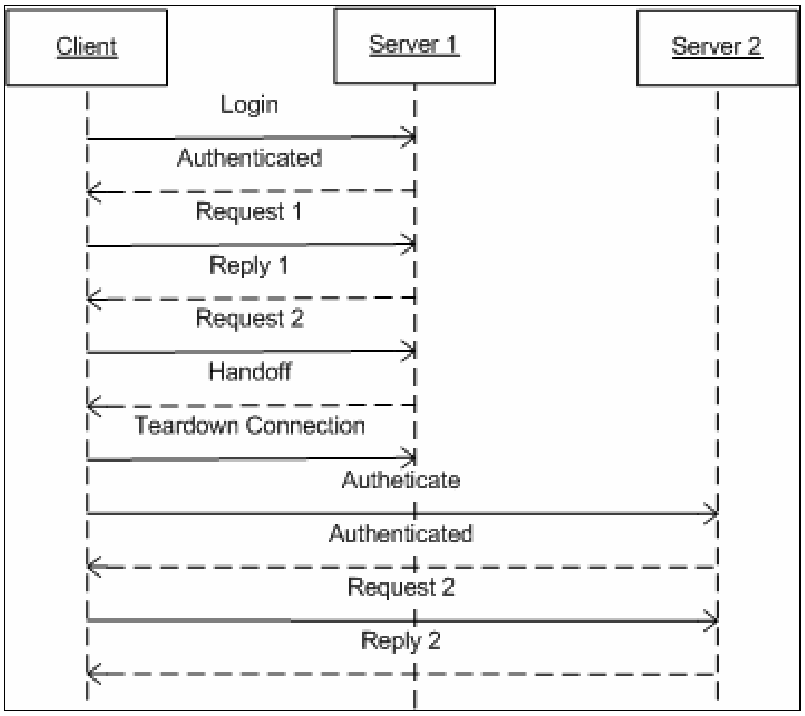


Figure 4: Timing Sequence for SenseSpace interactions

3.2 Name Server

When the system is brought up for the first time, the name servers load a configuration file. Each reads the XML files that store the configuration, and load the information internally in a linked list hierarchical fashion. Once the namespace has been loaded into memory, the name servers are ready to begin servicing requests. The name servers read the certificate file and using that are ready to receive SSL connections from the outside world.

A client that wishes to connect to the name server must then authenticate with the name server. This is done by checking the username and password sent by the user for a match in the password file on the server. For security purposes, the passwords are not stored in plain text, but are stored in a SHA-1 digest form. This ensures that the access control is not compromised if a single node is compromised.

Figure 5 shows how the name server handles and processes all incoming requests. This is done by splitting the request into component parts and traversing the in-memory structure of the namespace looking for matches. Once a match is found, the server determines whether a handoff is required or not. If no handoff is required, the response is sent back to the client. If, however, a handoff is required, then a handoff message is sent back to the client which restarts the process with another name server.

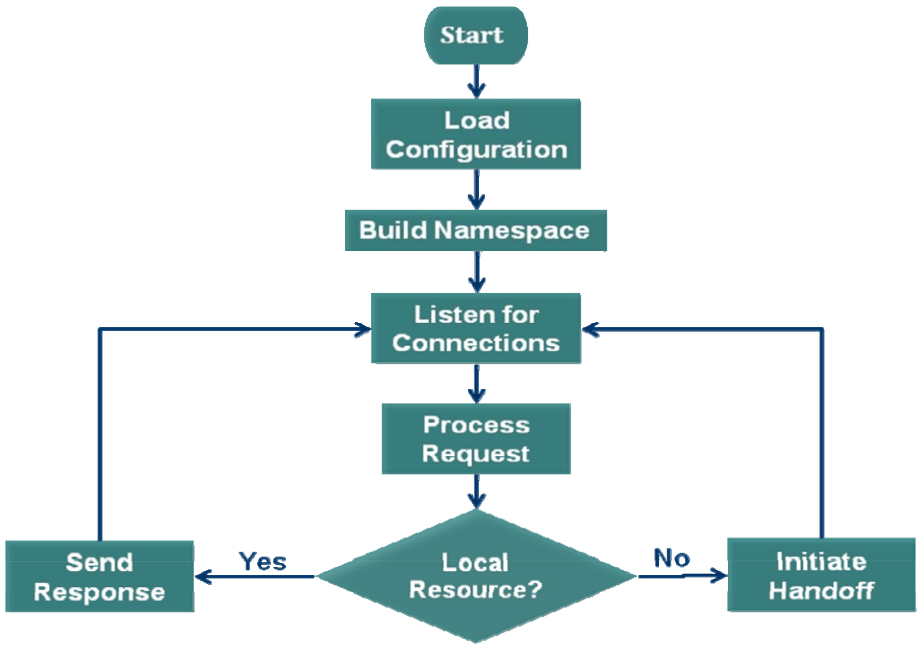


Figure 5: Process Flow for the Name Server

3.3 Client

The client is the part of the system running on the user's computer. This interface allows the user to traverse the namespace. To launch the client, the user enters their username and password. Once they have been authenticated, the client window launches. By clicking on the available parts of the visible namespace, the client builds up a tree like structure which the user can use to navigate the namespace. Every selection of a sub-tree sends a request to the server where it is processed and sent back to the client. If a handoff is required, the handoff is done without the user ever being aware of it.

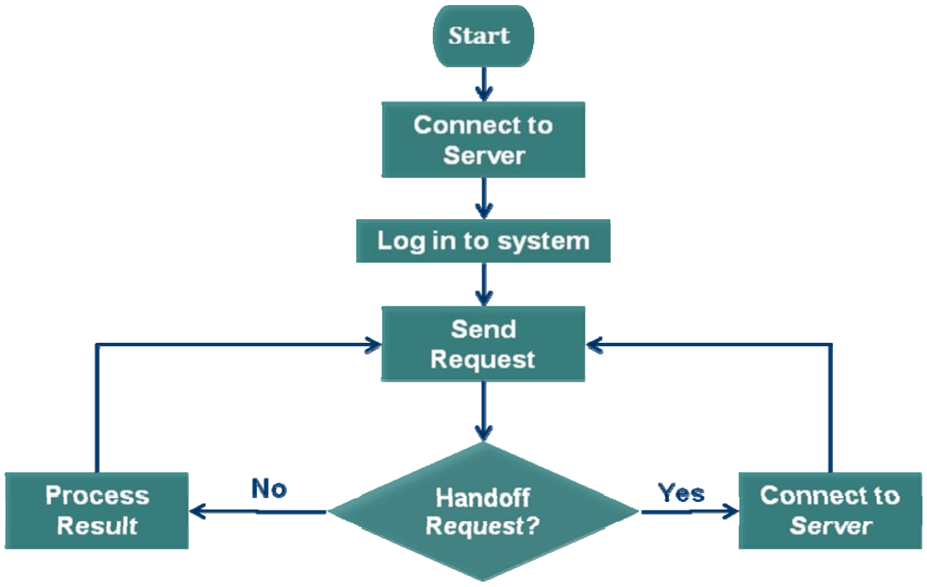


Figure 6: Process flow for the Client

Figure 6 shows the flow of logic at the client, whereas Figure 7 shows a screenshot of a client.

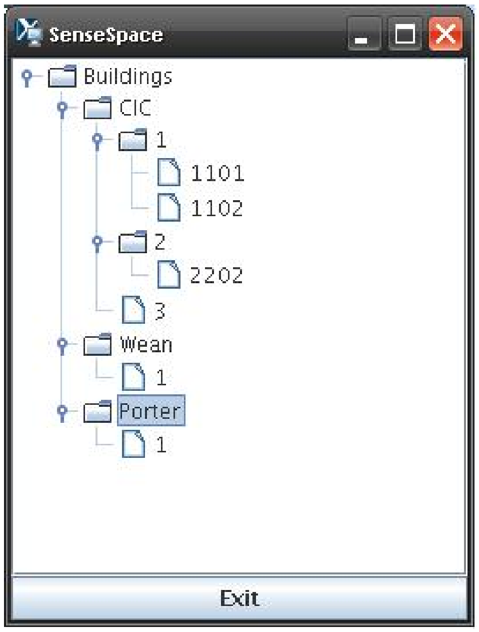


Figure 7: Screenshot of Client interface

3.4 User Access Control

User access control is provided at an individual element granularity through user names. Some rooms in the namespace can be locked in a manner such that only the correct user has access to viewing the portion of the namespace below it in the hierarchy. This provides control over who gets to see what parts of the system. This system implements three broad classes of users as shown in Figure 8 below.

The most privileged user is the administrator. The admin is allowed to view the namespace in its entirety. This is necessary so that he can ensure that every part of the system is working correctly.

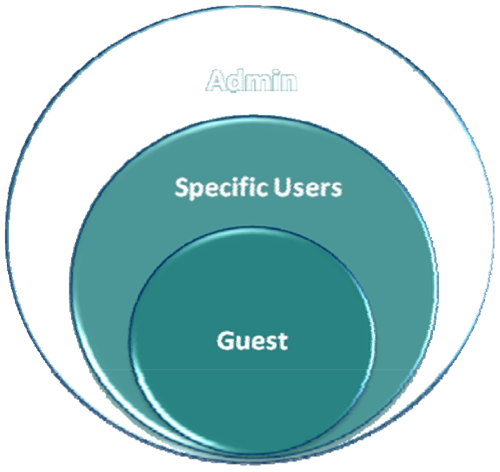


Figure 8: User Access Control

The user with the least privileges is the guest. This account should be used in situations where a person is being introduced to the system. The guest login would permit a user to view all the open components in the system. Sensitive areas and critical pieces of the namespace would be locked away from view.

A slightly more powerful user, but one not as powerful as the administrator is also a part of the system. This user could be an experimenter who wishes to run experiments on the network, but would not want others to view the status of the experiment. This form of user comes from the vision of using Sensor Andrew as a testbed for sensor network research and by using SenseSpace with such user control would permit for an easy transition for other users to be introduced.

3.5 Example

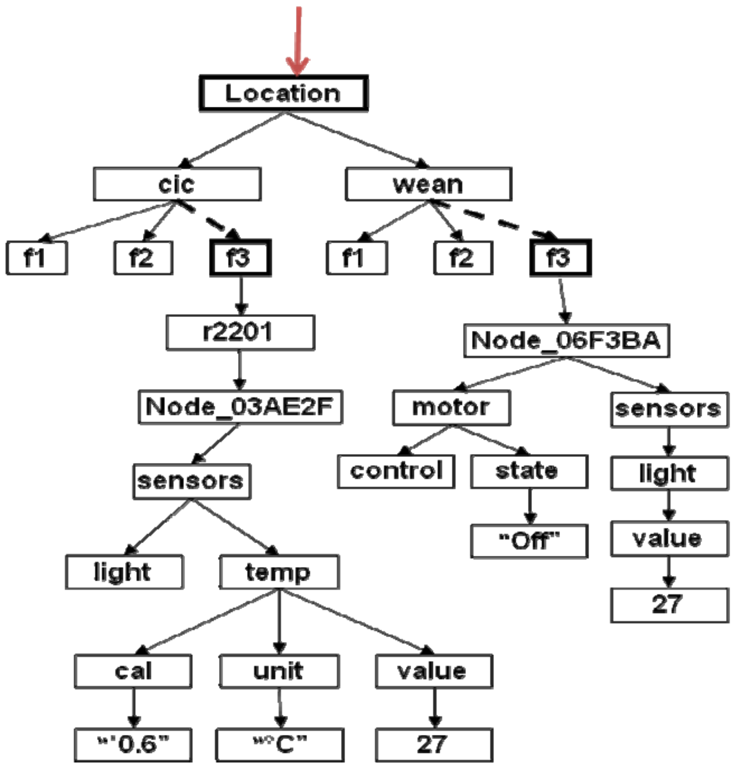


Figure 9: SenseSpace Namespace hierarchy example

Figure 9 shows an example namespace. A dashed line indicates a remote connection and would involve a handoff. If a request for *Location/CIC* was sent by the client, then the request would travel down the tree to *CIC*. The *CIC* node is aware of floors *F1*, *F2*, and *F3* as being one level below it in the tree. The name server would then send a reply stating that the next level to *CIC* was floors *F1*, *F2* and *F3*.

For a more detailed example, suppose a request comes for the state of the motor under *Node_06F3BA*, in *floor 3* of *Wean Hall*. Such a request would appear something like, *Location/Wean/F3/Node_06F3BA/Motor/State*. As earlier, the request would traverse down to *F3* of *Wean*. At this point the name server knows the next level is on another name server. A handoff is initiated, and the client connects to the name server on the third floor of Wean Hall. This name server, then processes the request, and finally returns a value, *OFF*.

4 Evaluation

4.1 Example Case Study

SenseSpace was tested under experimental conditions to obtain various performance metrics. The experiments were conducted using name servers running on the AFS Distributed File System.

Five name servers were set up on five different UNIX hosts on AFS. The assumption was made that all name servers were under similar, lightly loaded configurations. Each name server was responsible for a subset of a sensor network by loading different configuration files into each name server. The client requesting access to the namespace was started on another UNIX machine on AFS. This set up was then used to obtain some performance metrics.

We investigated the following scenarios:

1. Login time
2. Handoff between servers
3. Cascaded handoff between servers

Login time represents the overhead of establishing an SSL connection after which the user enters their username and password. Once logged in, we investigated the time it took to handoff a request from one server to another, as well as a cascading handoff where one request moved through two servers before finally reaching the name server which was aware of the subspace that was relevant to the request.

4.2 Performance Results

In this section we outline the performance aspects of the system.

Fig 10 shows the average time taken to log in to the system by a client. As can be seen, the time required to actually get into the system is extremely small (of the order of thirty milliseconds). The major portion of this time (mean = 34.65 ms, standard deviation = 0.00412) is used in setting up the SSL connection from the server to the client. A fractional amount of the total time (mean = 1.23 ms, standard deviation = 0.00125) is used for the actual authentication into the system. This authentication determines the user access control, as well as determines if the user has privilege to enter the system.

Figure 11 shows the time taken to handoff from one server to another. This time (mean = 69.76 ms, standard deviation = 0.00142) is fairly consistent over multiple runs, and is roughly 70 milliseconds. This is double the time taken to log into the system, but is still within the threshold within which it is an imperceptible delay. This time is double the time needed to actually log into the system, but this can be explained in the way the system was designed. By having the client tear down its SSL connection to the server before making a new one with the server which it is being handed off to, we go through two sets of SSL handshakes, as opposed to just the one for logging in to the system.

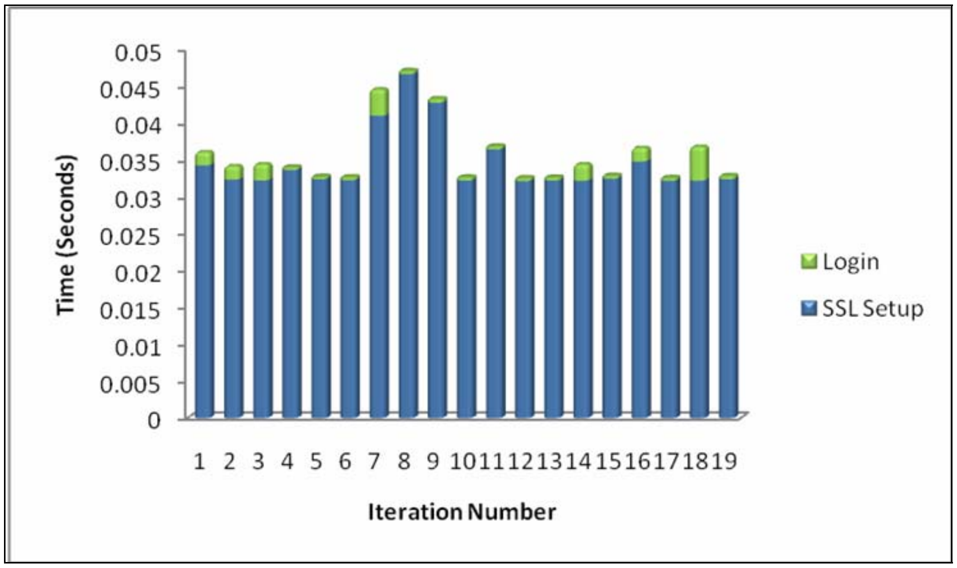


Figure 10: Login timings

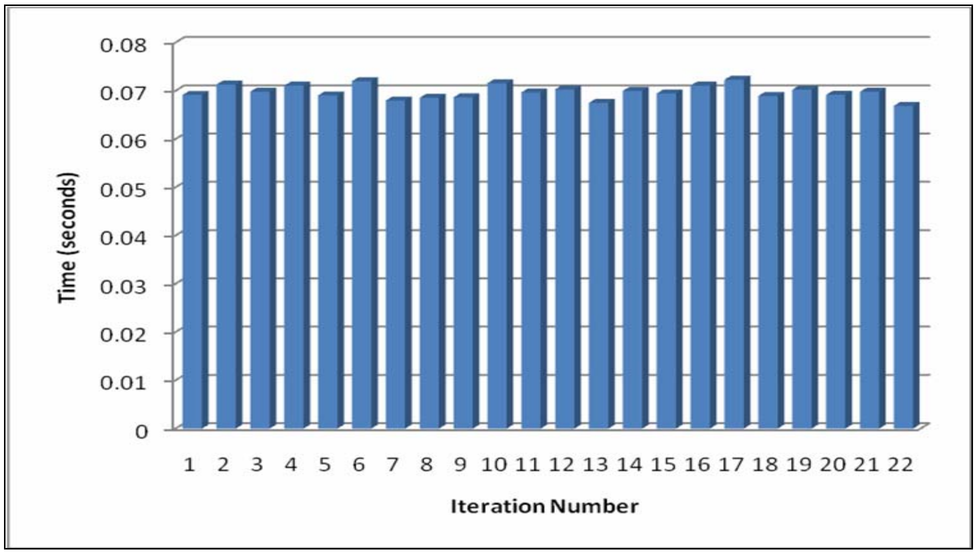


Figure 11: Distribution of the time to handoff between servers

Finally, we measured the time taken for cascaded handoffs (mean = 143.15 ms, standard deviation = 0.00632) in the system as shown in Figure 12. They were seen to take twice as much time as a single handoff. This shows that the system scales linearly. We see no unwarranted increase in time as the requests propagate through the system. Caching optimizations could be made to more efficiently handoff connections based on previous queries.

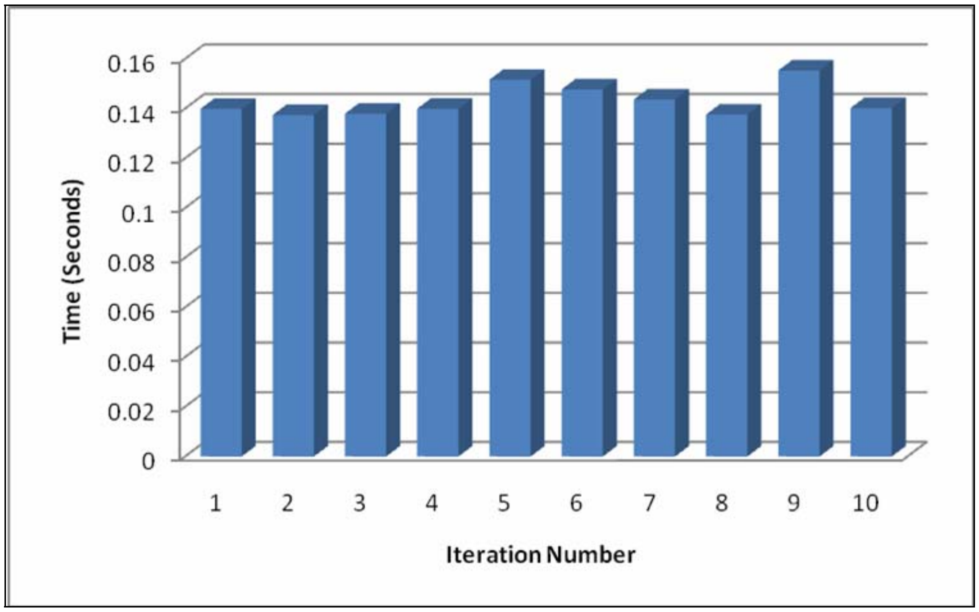


Figure 12: Distribution of the time taken for a cascaded handoff between servers

5 Future Work

This work describes our approach in constructing a secure, distributed namespace for sensor networks. In this section we describe several areas in which this work may be further extended.

Our current implementation provides simple access control; however, a more comprehensive scheme will lead to a more widely usable system. By including a more fine grained set of Access Control Lists (ACL), we allow the flexibility of introducing user groups, as well as extending the number of users who can access particular resources in the namespace.

Currently there is no support for mobility. As a result, if a node moves from one location to another, the namespace will be unaware of this movement, and will redirect the user to the wrong name server for information. This works well with static nodes which can be used for infrastructure monitoring but would need to adapt to include mobile nodes for other applications. This would require dynamically updating the XML structures in memory at run-time.

Finally, our system lacks fault tolerance mechanisms. If one of the name servers crashes, then the subspace of the system under that name server is lost to the user. There is no way the user can access that part of the network unless they had previously resolved the address information. Introducing fault tolerance would make the system much more robust and allow for the users to feel more secure in the ability to get at the information they require at any time.

6 Conclusion

As part of this thesis, we described an approach to design a distributed namespace for sensor networks, looking more particularly at Sensor Andrew, but leaving the implementation to be open for all kinds of networks. We managed to create a completely abstract visualization of the sensor nodes across campus. The namespace was completely distributed with no single point of failure, and by the use of configuration files that subdivided the namespace, we allowed for a completely scalable and easily expandable system. We also managed to add security to the system, to prevent unauthorized access to information, as well as restricting information based on users who log into the system.

7 References

[1] John Heidemann, Fabio Silva, Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, Deepak Ganesan. Building Efficient Wireless Sensor Networks with LowLevel Naming. In *Proceedings of the eighteenth ACM symposium on Operating systems principles*, Banff, Alberta, Canada, 2001.

[2] Sameer Tilak, Bhanu Pisupati, Kenneth Chiu, Geoffrey Brown, Nael Abu-Ghazaleh. A File System Abstraction for Sense and Respond Systems. In *Proceedings of the 2005 workshop on End-to-end, sense-and-respond systems, applications and services*, Seattle, Washington.

[3] Eric Van Hensbergen, Ron Minnich, Grave Robbers from Outer Space Using 9P2000 Under Linux, In *Proceedings of the annual conference on USENIX Annual Technical Conference*, Anaheim, CA, 2005.

[4] Paul V. Mockapetris, Kevin J. Dunlap, Development of the Domain Name System. In *Proceedings of SIGCOMM ‘88, Computer Communication Review Vol. 18, No. 4,* August 1988, pp. 123–133.