MAPNEXUS: REAPING THE BENEFITS OF A CONTEXT-AWARE 802.11 POSITIONING SYSTEM

AUTHOR: JEAN B. ALMONORD
ADVISOR: TED SELKER

ABSTRACT

Recently, there has been a considerable amount of development in the field of wireless position sensing based on the 802.11 protocol. The concept of location based sensing is not a new concept; prior to 1999, a great deal had been done with infrared (IR) sensors, cell phones, and the Global Positioning Systems (GPS). Previous work in this field includes the RADAR project at Microsoft, which was accurate within five meters 75% of the time, and the Nibble project at UCLA, which was accurate up to ten feet. Similar to most of these projects, the Mapnexus project uses 802.11 position-based sensing to determine a user's location. It was not the aim of this project to pinpoint exact locations, but rather a reasonably good approximation to an area (30-40 feet) or part of a building and as opposed to a specific room. A good approximation was achieved was achieved that was correct about seventy percent of the time.

The primary aim of this project is to develop an 802.11 context-aware self-evolving position-sensing interface. The interface will not only alert users with given information such as their location, but also with pertinent information based upon the user's location, time, date, and season of the year. The interface was specifically designed with a reasonably sized community in mind such as a medium-sized American school campus—in this case the MIT campus. The interface changes dynamically based upon known MIT student's patterns of behavior. For example, the interface changes in terms of most of its content and look and feel based upon whether or not it is a weekday or not.

Acknowledgement

No man is an island, entire of itself; every man is a piece of the continent, a part of the main.
---John Donne (1572-1631), For Whom the Bell tolls

It is with great pleasure that I give thanks to everyone that has helped in one form or another in developing this project. My sincere thanks goes to Professor Ted Selker and Napier Sandford Fuller, without their help, advice, and counseling this project would have not gotten off the ground. I would like to also thank Professor Henry Holtzman, Vadim Gerasimov, Sunil Vemuri, Ernesto Arroyo, Taly Sharon, Will Glesnes, and Sam Davies.

Introduction

A. Motivation for this Project

Often times, while walking around the MIT campus in search of a building or a location, one must return to a lobby with a campus map in order to locate exactly where a given building is. Worse yet, not all of the public bathrooms at MIT are handicapped-accessible bathrooms. A Handicapped person doesn't have an easy way of telling directly which bathrooms are handicapped-accessible without actively getting to the bathroom first. Many times, one misses interesting events, seminars, and or even lectures simply because he/she failed to realize or was not aware that such an event was occurring. The following project looks at a contemporary method of solving such problems altogether.

This project deals with the creation of an integrated, context-aware, intention-based interface. It is comprised of two major components: a server and a small client software package. The server comprises of an Apache-Tomcat server along with a database and a dynamically changing front-end web page. The client software package constantly updates the server with certain information such as the host's Internet protocol (IP) address and a listing of the machine address code (MAC) addresses along with signal strengths and signal to noise ratios of the Access Points (AP) that it is currently receiving signals from. The server serves as a way to forward the information from the web page to

the database and a place where the backend algorithms are processed. The database is used to store the bulk of the information. The web page is used as a front-end interface connecting the users to the back-end (server and database). The client piece does the backend procedure involving the gathering of the relevant information such as the MAC addresses and corresponding signal strengths of the AP. This information is relayed to the server every 5 seconds.

The front-end website tells the user a great deal of information. Such information includes pinpointing the user's location in relation to the MIT campus, and finding on the blueprint of the buildings the user's floor and approximate room location—room location based upon a quadrant or a ninth of the floor depending on area of the building. The site also contains information such as the weather, the next two events that are occurring at MIT, information about the nearest eateries/cafeterias, nearest elevators, bathrooms, and sports facilities along with their operating hours.

Currently there exist the MIT event calendar, the online MIT campus-map, and there is talk about the creation of a website for the tracking the locations of the MIT Saferide buses. If people are interested in finding out about their environments, why not let them search for such information themselves? The point of this project is precisely to integrate all of the above MIT services into one that is context-aware. Such a context-aware project will only bother updating users with pertinent information that is relevant to their environment. In a way, the purpose of it is to update the user in a useful way of what he/she would have found interesting.

B. The 802.11 Specification

Nowadays, 802.11 is at least in the technological field synonymous with wireless. Although there are other series of wireless specifications such as Bluetooth and others, 802.11 is predominant in the wireless world, because of its simplicity and efficiency. 802.11 is a member of the IEEE 802 family, which is a series of specifications for local area network (LAN) technologies [1]. Figure 1 shows the components of the 802 family and how they relate to the OSI model.

Every 802.11 network has a name associated with it. That name is usually referred to as the service set identifier (SSID). There are "only twelve radio-link channels" that can be used in the United States [2]. The SSID typically indicate which logical network is on which channel, thus allowing a receiver to use the strongest available signal. Access Points (AP) usually send out beacon packets that consist of the SSID, a maximum transfer rate, and the MAC address of that access point.

Clients wishing to join a network must first have a network interface card—also known as a wireless card. The wireless card locates the beacon, and then sends out a probe request packet. If access is allowed, the AP will respond to the request with a probe response packet, which contains the network's SSID. Once the client is on the network, the AP serves has the bridge between the wireless networks and wired network

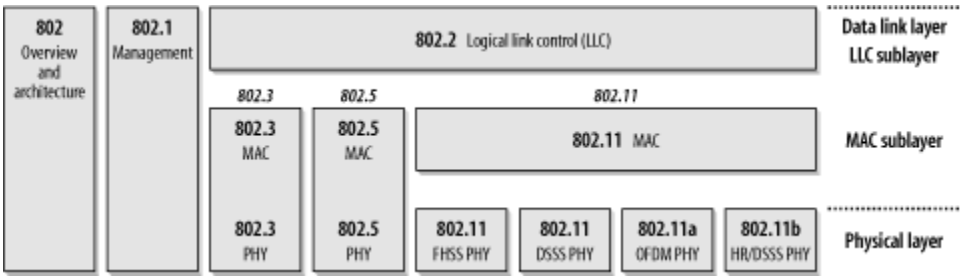


Figure 1. The 802.11 family and it's relation to the OSI model [1]

C. Previous work with 802.11 Position Sensing

Recently, there has been a great deal of research in regard to wireless position sensing. Many office buildings and research buildings are equipped with wireless access points, and using the wireless model of position sensing requires very little extra hardware. Once one is able to obtain the preliminary data from the wireless access cards such as the MAC addresses, signal strengths, and the signal to noise ratio, the bulk of the work is done primarily with software—i.e the algorithms to determine position.

Previous work with 802.11 position-based sensing include: the RADAR project at Microsoft and the Nibble project at the University of California of Los Angeles (UCLA). Whereas both projects dealt with position sensing for exactly one floor of a building, it is noteworthy that the projects varied in how accurate it was to deciphering one's location.

The Microsoft RADAR project created a mapping of radio frequencies of data originating from the signal strengths along with how the user is oriented at seventy different locations on a floor of a building. Whereas locating stationary users was very complex, locating mobile users was one of the hardest tasks of this research. A history-based method – i.e. history of the past locations where the user have been along with some algorithms that will calculate the likelihood of his/her next path— was used to

handle such cases. The RADAR project was only able to accurately determine a person's location within 16 feet seventy-five percent of the time [3].

The Nibble location based sensor, on the other hand used a Bayesian network— a network that applies Bayes's theories of probability to sense location based upon given information about adjacent neighbors— to determine the user's location. The Nibble project was able to determine location accurately within 10 feet [4]. It is important to note that the difference in accuracy between the latter project and the Microsoft RADAR project is approximately six feet.

An accuracy of 10 feet or 16 feet will both work for the Mapnexus project. This project is not overly concerned with pinpointing exactly someone's location but however a general area in the range of 30 to 40 square feet is acceptable. Once such a location has been determined, one can easily gather much about the surrounding of such a location. One can easily determine if there are bathrooms or handicapped-accessible bathrooms in the vicinity, whether or not there are conference rooms, lecture halls, auditorium and when they are occupied, and whether or not there are eateries, bookstores, and libraries nearby and when they are open.

D. The Significance of this Project

802.11 position-based sensing is not a brnew concept. Many researchers have done interesting researche in the field. The Mapnexus project is significant, however in that it is a context-aware interface that relies upon 802.11 position-based sensing to determine a person's location. Once the location has been determined, the interface can then gather pertinent information about such a location. Hence Mapnexus, is different and significant in the sense that once a person's location has been determined, it can dynamically retrieve information about such a location and make inferences about which information is relevant based upon the time of day, season of the year, and general particular habits of people belonging to a particular community.

The Mapnexus project is also significant because it is designed to deal with a specific community. In this case, the community is the MIT campus. It is obviously the case that most of the MIT populace shares some of the same habits, if not similar patterns of behavior. It is obviously the case that during the final examination periods, most MIT students are studying on campus. Similarly during the Independent Activities Period (IAP), many students either eat out on a regular basis or they cook for themselves. The Mapnexus interface could easily manipulate such information to make recommendations to students—i.e recommendations about nearby restaurants, supermarkets, and the MIT campus dining facilities. It should be noted that this project is different from the Nibble

and RADAR projects because the domain consists of not only one floor of a building but instead of multiple floors of a building.

Design Criteria

A. Efficiency: Tricks of the Trade

There were many design patterns that were used in order to achieve efficiency. Whereas some of the design patterns that were more obvious in some modules versus others, some modules used a combination of design patterns. Some of the design patterns used include: the singleton, the factory method, the decorator, and the flyweight design pattern.

Mapnexus needs to actively retrieve data fairly fast from either the database, or from an html page. If the interface were to retrieve such information on each request, it would be quite redundant. As a result the module that processes such request, html querier, was created as singleton. The fact that it was a singleton ensured that there could only be one instance of this module running at a time. A flyweight design was used as the structural pattern. The flyweight pattern uses sharing to support large numbers of fined-grained objects efficiently [6]. Rather than make html queries for every request, the module stored information about the past requests that it has made and updates them every hour.

The singleton pattern was also used for the Database Access module. There only needs to be one of such a module, as it would be inefficient to have a numerous amount of connection pools to the database. Furthermore, using a singleton assured that the decorator pattern could be used for the database. Decorators provide a flexible alternative to subclassing for extending functionality and they attach additional responsibilities to an object dynamically [5]. The higher-level modules do not need to know what type of a database is storing the files, or what query language is being used to make queries to the database.

B. Modularity

The Mapnexus project was designed from the earliest stage with modularity in mind. Many of the components of the interface can easily be broken down into subcomponents and these subcomponents can be broken down into smaller subcomponents. This is allowed for a rigorous validation strategy for the entire system, as the individual modules could be tested both independently prior to integration and as whole after integration. Figure 1 shows a diagram of the layout of the modules.

The concept of a modular design also proved useful in terms of interoperability. Many of the modules that were specifically designed for the interface, can easily be used with other interfaces or systems. For example the html querier module— the module that goes to certain html pages and retrieves certain information such as a String with time 6:00 PM-- can easily be used with any interface that needs to extract information from an html page.

C. Simplicity

Keep it Simple Stupid.

A chief goal of the Mapnexus project was to design an interface that is not only very simple to use but also very simple for any programmer to extend. The interface was designed with the mindset that each module should be implemented in the simplest way possible--simplest referring not to the lack of complexity but rather to the shortest and easiest way possible.

The original design consisted of many more modules than the currentt design. Many unnecessary modules were either deleted or combined with existing modules. For example, earlier the database access module- module for retrieving and making queries to the database— consisted of two different modules. One module simply created a connection pool to the database, and the second one processed request to execute queries to the database. Whereas the module that was creating the connection pool was a singleton—only one of it's kind can exist—the module processing the requests weren't. In turn, this might have caused problems if two of the upper processing modules are making queries to read or write the same information. As a result the two modules were combined into one: the Database Access module.

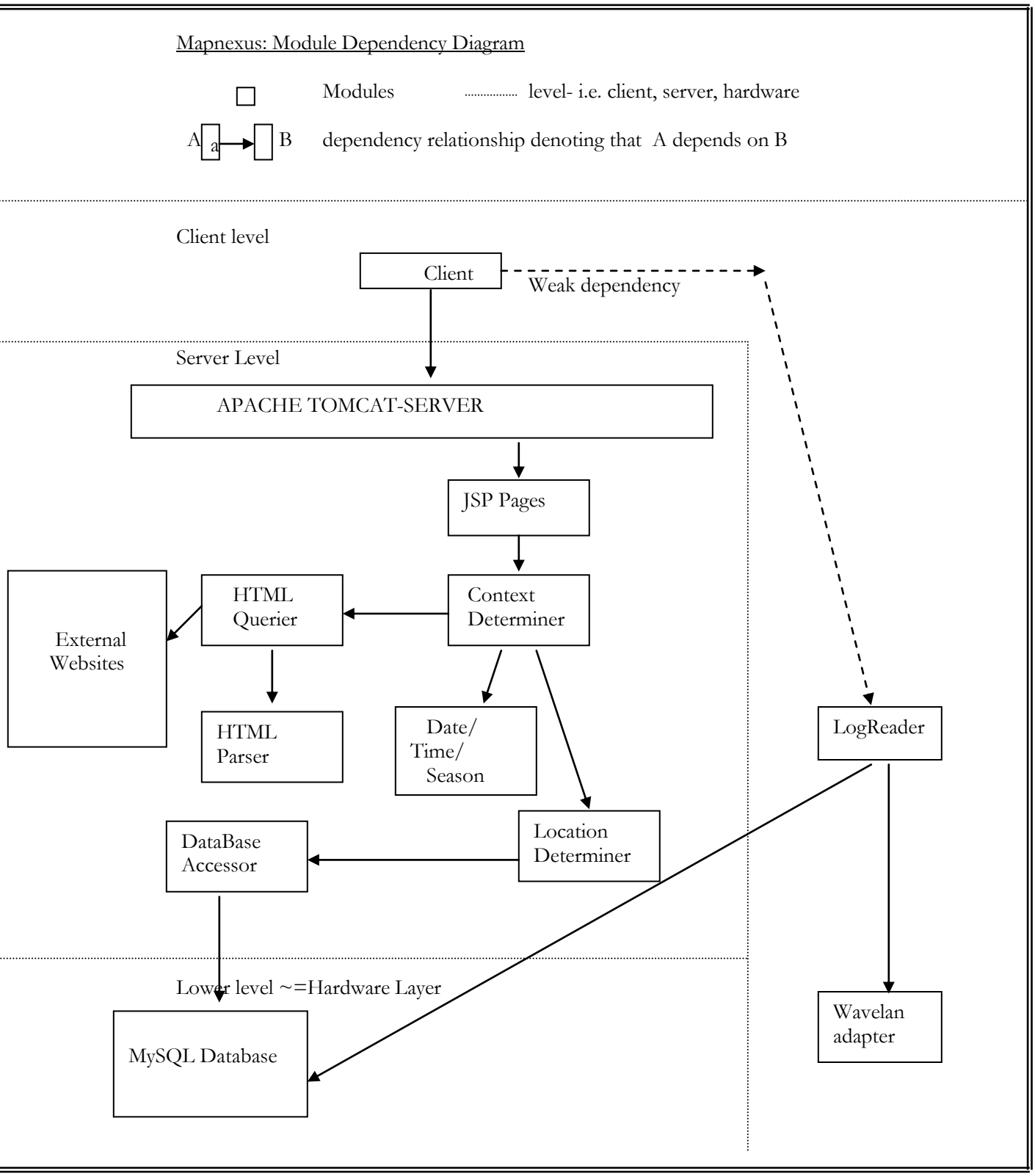


Figure 2. Module Dependency of Mapnexus.

Overall Design

A. Position Sensing: Developing a Reasonable Approximation

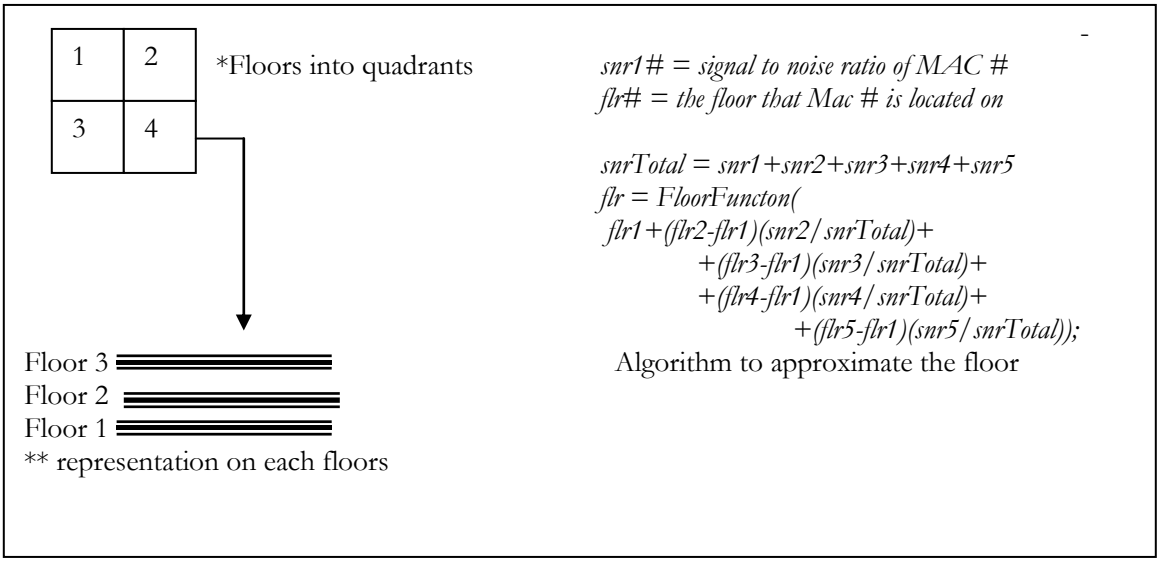
The 802.11specification was discussed in the section titled "The 802.11 Specification."

The important data such as the list of MAC addresses and the corresponding signal strengths and signal to noise ratios of the different APs were directly determined from the wavelan adapters. In order to determine a user's position many factors were taken into account such as the signal strength along with the signal to noise ratio, and the location of the access points. The locations of the APs along with their associated MAC addresses were stored in the database beforehand.

Every floor of the media lab has on average about 4 Access Points, and if each floor is divided into four quadrants the APs usually fall distinctly in those four quadrants. Figure three illustrates this concept. Note that the interface chooses to only work with the five strongest signals. Since there are four quadrants and Access Points, two Access Points must fall into the same quadrants. To determine the quadrant, one can just chose the quadrant that repeats the most among the location of the APs.

In order to make an approximation of the floor using the given information about which floor the access points are on and their signal to Noise Ratio, see the algorithm in figure 3. Using the fact, that the best signal to noise ratio will be the nearest to the user, the interface makes a good approximation as to the user's location. This is done by taking a difference between the floor that the AP with strongest AP is located, which we can call for convenience flr1 and adding it to the difference of the floors that the other APs are located times their respective signal to noise ratio divided by sum of all five signal to noise ratios. Notice that the difference can either be negative or positive depending on whether or not the floors of the APs are above or below.

It is interesting to note that that the algorithm worked about 65% of the time, and when it was off that it was never off by more than one floor. Comparing this algorithm to one in which one chooses the location of the nearest AP, the nearest AP approximation is not as good if one keeps on receiving really strong signal to noise ratios for APs that are far away. The Mapnexus algorithm will give a better approximation in such a case. It is only when such an oddity is not present that the nearest AP algorithm will be more accurate. These oddities are almost always present; In fact, they are what limit how feasible one can use 802.11 to correctly tell location.



Figue 3: Representation of algorithm to determine location

B. Learning about my Environment

In order to fully make this interface context aware there were many things that had to have been done a priori. Some of those things included the creation a list of keywords to associate with each season of the year, time of the day, and month of the year, and dates of the year. For example 12:00 PM – could be associated with noon, midday, lunch, and food.

Having such associations made it very easy to update the user with pertinent information. Every hour, the interface uploads the information about the weather to the user. The interface also uploads information about the events that are going on. For this information, the interface is also active relying on the concept of time to make a decision. In the case of the weather, it is simply the hour. However, in the case of the upcoming events, it deals with both the time and searching through the events calendar at MIT for the events that not only fall on a particular day, but also after a particular time.

The interface also highlights certain things about the environment such as the nearest bathrooms, handicapped-accessible bathrooms, elevators, and stairways. Originally, the goal was to be able to click on these facilities to find out more about them. However, the interface did not get to that point in time for this first release. One can also zoom out of the quadrant to view the full floor plans and so forth for the MIT map. See figure five for an example of such a floor plan.

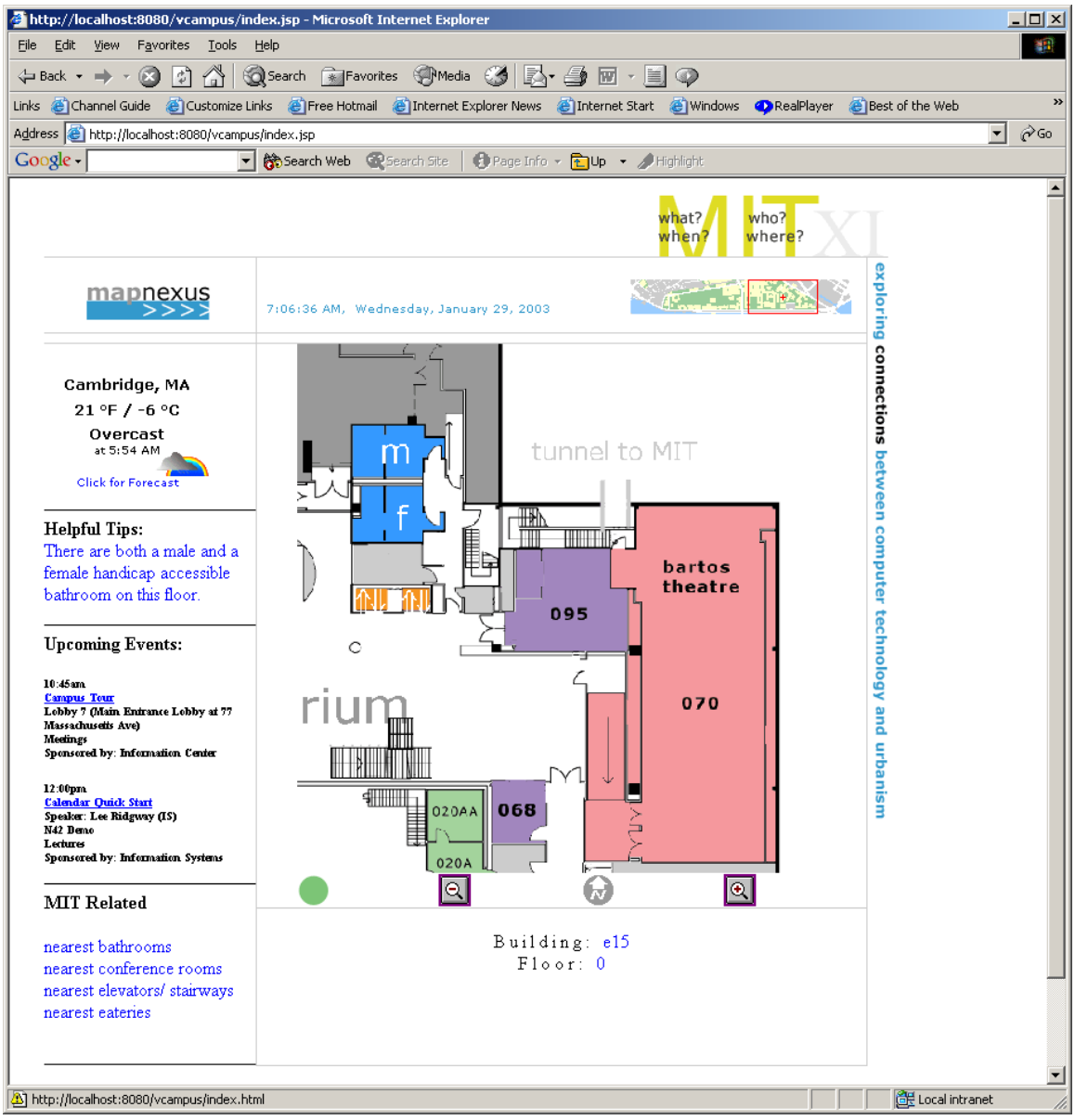


Figure 4:View of Site once the client goes to view it

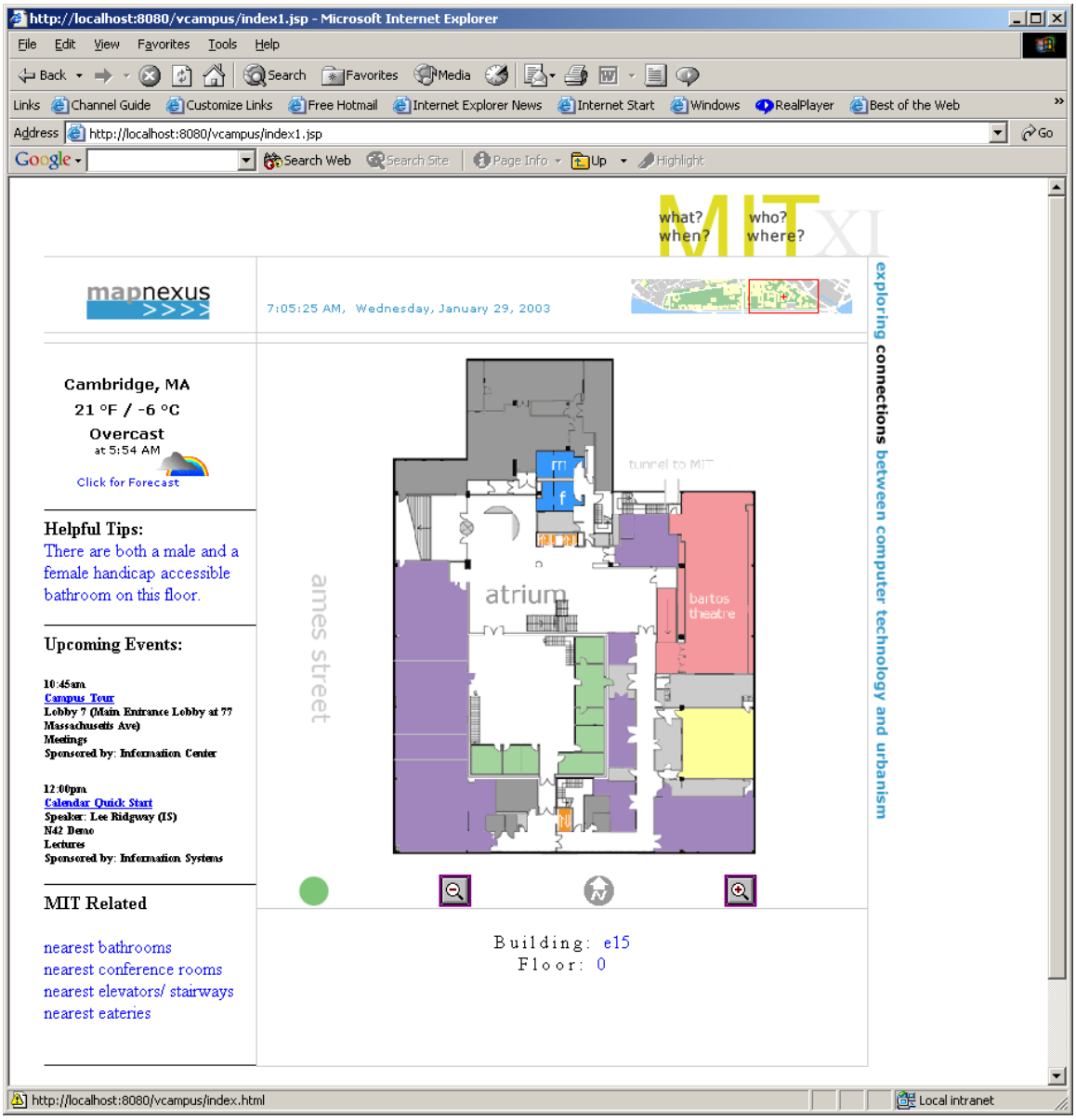


Figure 5: View of the floor plan when the user has zoomed out.

Design Tradeoffs

One of the preliminary design decisions was to implement Mapnexus as a central server using the simple network management protocol (snmp) in order to reduce any work that the client would have to do. The central-server design is similar to the current design with the exception of the fact that the client is no longer updating the server with the information about the access points. The central server would have provided a platform independent way of doing Mapnexus and it would remove any work needing to be done on the client side. However, The central-server proved inefficient in many ways.

It was not scalable. Whereas it would have worked reasonably well, for a small number of users, it could easily have been overloaded when many more users started using it. It was simply doing too much. The server would need to explicitly extract the IP from the http request coming from all clients, and send broadcast packets to every network interface that are in the vicinity of the client's location. Furthermore many networks apply security measures such as MAC address spoofing. The central server would have no way of verifying the MAC addresses and knowing which ones were valid and which ones were not.

The current design avoids these issues altogether by providing the client with a client software package. The client software package can easily update server, which directly reduce a great load from the server. Since the client is a trusted by the Access Points that it is connected to, the client can software package can in turn self-validate the information that it is receiving before sending it out to the server.

Conclusion

The Mapnexus project is a simple context-aware interface that uses 802.11 position-based sensing to determine a user's location. Once that location has been bee determined, the interface dynamically retrieves information that is both associated with the location and relevant to the context of time, day of the week, and/or season of the year. The information can be as simple as the fact that there are handicapped-accessible female bathrooms nearby, or as complex as the fact that there is a seminar occurring during lunchtime at such a place and nearby there is a dining room that is open.

The ultimate goal of the Mapnexus project is to be able to give users detailed information such as the fact that there is a seminar occurring during lunchtime at such a place and on the user's way to such a place there are three popular and inexpensive eateries specializing in these types of cuisines that are now open. Not only would Mapnexus be able to simply know which seminar is occurring at which location and at certain time that falls during lunchtime, it would also associate lunchtime with eateries.

It would only retrieve information about eateries that are on the way to the location where this seminar is taking place coming from the location of the user. Furthermore, the interface would give the user detailed information about whether or not these eateries are open and their specialty—i.e Italian food, Mexican food, Pastry, etc… This would entail a true context-aware interface, an interface so user-friendly that most users would grow attach to it.

The algorithm for determining position based on the access points and the signal strength can also be improved. As it stands, the algorithm works most of the time. It runs into problems when there are APs that are far away and broadcast stronger signals than then the AP's that are closer. Although, this does not typically have any effects upon determining the subsection of a building that a person is on, however it does when it

comes to determining the floor accurately. In these particular situations, the interface usually report a floor above or below the current floor.

Sources

1. Mathew Gast (2002). 802.11® Wireless Networks: The Definitive Guide. O'Reilly Publishing Co.

2. Larry loeb. Roaming charges: Spoofing the war drivers. Wireless Security through obscurity (October 2002).

3. Bahl, Paramvir and Pasmanabhan, Venkata. A Software System for Locating Mobile Users: Design, Evaluation, and Lessons (July 2000) .

4.Castro, Paul. The Nibble Location System (May 2001)

5.Gamma Erich et al (1995). Design Patterns: Elements of Reusable Object-Oriented Software.Addison-Wesley.

6.Hans Bergstein (2001).JavaServer Pages. O'Reilly Publishing Co.

7. Hightower, Jeffery. Location Sensing: A Framework of Techniques and Taxonomy of System Properties (June 2001)

7.Eugene Blanchard (2001). Introduction to Networking and Data Communications. Commandpromt, Inc.