Professor Controller
Wireless Personal Communication Systems
14:332:426

Aviram Shapira
Frank Shao
Andy Wang
Kerry Wong
Lal Bharwani
Introduction

A large class size can seriously intrude upon a student's learning environment as the student may feel overwhelmed by the presence of his peers. No matter how experienced or well prepared the instructor is, managing a big classroom requires some creativity. This disadvantage of being in a big class can lead to a less valuable learning experience. It takes a lot of courage to question the teacher in a big class. The Professor Controller is a system that allows semi-anonymous wireless interaction between students and an instructor in a classroom setting. The system utilizes the 802.11b wireless protocol and requires a LAN connection and a wireless access point. Students and the instructor are assumed to be equipped with laptops that run this application. The following diagram describes the interaction of the application:

Figure 1
Feasibility

In developing this system, there were a number of theoretical issues that needed to be considered which could affect the performance of the application. This section examines some of the issues that were taken into consideration during the implementation of the project.

When using a wireless network, the range of this network becomes an important issue. In general, access points have a range of about 100 ft indoors and 300 ft in open space. However, these ranges are affected by interferences such as microwaves and other radio interferences, as well as interference for structures such as walls and doors.[15]

<table>
<thead>
<tr>
<th>Obstruction</th>
<th>Degree of Attenuation</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Space</td>
<td>None</td>
<td>Campus Center</td>
</tr>
<tr>
<td>Wood</td>
<td>Low</td>
<td>Floor, Partition, Door</td>
</tr>
<tr>
<td>Plaster</td>
<td>Low</td>
<td>Inner Wall</td>
</tr>
<tr>
<td>Glass</td>
<td>Low</td>
<td>Non-tinted window</td>
</tr>
<tr>
<td>Metal Tinted glass</td>
<td>Medium</td>
<td>Tinted Window</td>
</tr>
<tr>
<td>Human Body</td>
<td>Medium</td>
<td>Groups of people</td>
</tr>
<tr>
<td>Bricks</td>
<td>Medium</td>
<td>Outer Wall, Floor</td>
</tr>
<tr>
<td>Paper</td>
<td>High</td>
<td>Stacks of paper</td>
</tr>
<tr>
<td>Silvering</td>
<td>Very High</td>
<td>Mirror</td>
</tr>
<tr>
<td>Metal</td>
<td>Very High</td>
<td>Desk, Reinforced concrete</td>
</tr>
</tbody>
</table>

Table 1, (taken from [15])

The access point has a data rate of 11Mbps, but this data rate does not resemble actual throughput. When data is being transferred via wireless networking, it requires a
certain amount of overhead that is also included in the data rate [15]. The output power of wireless LAN systems is very low, much less than that of a hand-held cellular phone. Since radio waves fade rapidly over distance, very little exposure to RF energy is provided to those in the area of a wireless LAN system. Wireless LANs must meet stringent government and industry regulations for safety. No adverse health affects have ever been attributed to wireless LANs [15].

**802.11b Standard**

The IEEE 802.11, or more commonly known as WiFi, is a set of standards for the wireless local area network (LAN) that was developed by work group 11 of the IEEE 802 LAN/MAN Standards Committee (LMSC) [19]. The 802.11b uses carrier frequencies in the microwave range, specifically in the range from 2.4 GHz to 2.483 GHz [20]. This range of frequencies is also called the unlicensed industrial, scientific, and medical (ISM) band. During recent years, many other technologies have found an interest in using this band as well for high speed data transfers [21]. The 802.11b protocol can transfer up to a maximum speed of 11Mbytes/sec. However, this is only possible with several encoding schemes that increase data throughput in a channel, which in this case is air. However, in the event of noisy conditions, the IEEE 802.11b defined the dynamic rate shifting as the automatic adjustment of data rates in said conditions, so that the 802.11b device will transmit in slower speeds in noisy events, and shift back to higher speeds when conditions return to normal [23].

**DSSS**

The encoding protocol used in 802.11b is the DSSS (direct-sequence spread-spectrum) technology. DSSS is a Code Division Multiple Access scheme that works by taking a data stream of zeros and ones and modulating it with a second pattern, called the chipping
sequence [20], which is also known as the pseudo-code. Originally, in 802.11, the sequence used is the Barker code of 11-bit sequence (10110111000). This follows the mathematical equation of

\[ \left| \sum_{i=1}^{l-k} a_i a_{i+k} \right| \leq 1 \]

and were used for pulse compression of radar signals[24]. The digitized signal stream is exclusive OR'ed or added with the pseudo-code to generate a series of data objects called chips. Therefore, each signal bit is encoded by a set of 11-bit Barker Code.

**CCK**

However, in starting from 1998, the Barker Code was replaced in favor of the complementary code keying modulation scheme. CCK is much more complicated than the Barker Code, and it can hold 64 unique code words, so that up to 6 bits can be represented by any particular code word, as compared to the one bit represented by the Barker symbol [20]. The Complementary Code Keying came be combined with the 2 Mbytes/sec modulation scheme, the QPSK technology.

**QPSK**

Another scheme that is used to increase the throughput of the 802.11b is the QPSK. Quadrature Phase Shift Keying (QPSK) is used for the higher data rate signal modulation. Instead of Binary Phase Shift Keying (BPSK) where there is one phase shift for each bit sent, the QPSK uses four orthogonal phase shifts (0, 90, 180 and 270 degrees) to encode 2 bits of what is being sent using what would have been the same space of the encoding for one BPSK [20, p1]. The trade-off is that you must increase the amount of power in transmitting the signal or decrease the range of how far a certain transmitter can go to maintain signal quality. Since the FCC regulates how much output power of portable radios can transmit, 1 watt EIRP (equivalent isotropically radiated
power)[20], range is the only plausible remaining factor that can change. That is why that on 802.11 devices, as you move away from the access point, the radio (AP) readjusts itself and uses a less complex (and slower) encoding mechanism to send data[20, p1].

Through the use of the QPSK and CCK modulation schemes in tandem, it is much more difficult to discern which of the 64 code words is being transmitted across the airwaves, mainly due to the complex encoding[20,p2]. In addition to all this, the radio receiver design is significantly more difficult. In fact, while a 1-Mbps or 2-Mbps radio has need of only one correlator, which is the device responsible for lining up the various signals bouncing around and turning them into a bitstream, the 11-Mbps radio must have 64 correlator devices.

**CSMA/CA**

In the 802.11b device, a network control protocol called the carrier sense multiple access with collision avoidance is used to figure out when to transmit[25]. First the host tries to determine if there are others transmitting at that time. Once the host has determined that channel, or medium, has been idle for some time period, known as DIFS (DCF [Distributed Coordination Function] Inter-Frame Spacing), it may transmit a packet [20, pg 2]. If the medium is busy, the transmitter must wait for a time equal to DIFS, plus some generated random number.

As each station listens to the network, the first device to finish its allocated number of random count down, begins transmitting. If another station hears the first station talk, its own count down is frozen until the network is idle again [20, p2], then it resumes the countdown. More than just the basic back-off algorithm, 802.11 also includes a back-off timer that makes sure of fairness. Each device starts a random back-off timer when waiting for the open window of time for it to transmit. This timer goes all
the way down to zero while waiting in the for that time. Each node gets a new random timer only when it wants to transmit[20, pg 2].

When the node does get to transfer, the signal is not a continuous stream of data, but rather, small packets that are filled with a lot of header. The station, sends a packet out, waits for an acknowledgement from the receiver. If the acknowledgement is not received within a certain time, then the station must re-transmit the packet [25, pg 8].

When a node (laptop) wants to join an existing Base Station (AP), it needs to get synchronization information to the access point [25, pg 10]. There are two ways to obtain the information. First, passive scanning is when the station or node is waiting for the synchronization information in the form of a Beacon Frame that is periodically sent out from the access point. Second is the active scanning, where the node sends out Probe Request Frames and awaits a response from the receiver.

**IP Addressing**

In networking theory, the concept of the IP address is a fundamental one to understand. Basically, an IP address is an identifier for a device or computer on a TCP/IP network. By connecting to the IP address of the destination, a computer can route messages to another computer on the network. Every host interface is represented by a 32-bit address composed of numbers that is written as a set of four 8-bit numbers separated by periods. For example, an IP address could be 1.160.10.240 [2].

The four numbers within an IP address are used differently to identify a specific network and a host on that network. It is easier to represent these numbers in decimal notation, rather than binary. However, it is in binary where one can find out which class of network the IP address belongs to. There are three classes of IP addresses: Class A, Class B, and Class C. Class A supports 16 million hosts on each of 128 networks and
includes the binary addresses that begin with 0, which means that the first decimal number can be in the range 0 to 127. The network can be identified by looking at the first 8 bits while the following 24 bits indicate the host within the network. For example, a Class A IP address, 102.168.212.226 will mean that "102" identifies the network and "168.212.226" identifies the host on that network [3]. Class B supports 65,000 hosts on each of 16,384 networks and includes the binary addresses that begin with 10, which means that the address range is from 128 to 191. The network can be identified by observing the first 16 bits while the host within the network can be found from the remaining 16 bits. A possible Class B IP address can be 168.212.226.204 where "168.212" identifies the network and "226.204" identifies the host on that network [3].

Lastly, Class C contains 2 million networks, each consisting of 256 addresses. It includes the binary addresses that begin with 110 and so the address range is 192 to 223. To give an example, a Class C IP address number might be 200.168.212.226. The first portion, "200.168.212", is the identifier of the network while the second portion, "226", is the identifier of the host on that network [3].

The original rationale behind these three address classes is that the address block assignments are based on the size of the network. Class A should be assigned for address blocks when considering organizations with large networks. For Class B, address blocks for organizations with medium-sized networks should be assigned. Class C address blocks are mainly used for organizations with small networks [2].

In this project, an isolated network was implemented. The only condition in an isolated network is for the IP addresses of each computer to be unique. The classroom software can be accessed without actually getting onto the Internet. Thus, there is no need to use registered IP addresses or Internet addresses.

**Subnetting**
Subnetting is a networking technique whose function is to provide the capability for a single IP address to span numerous physical networks. The supporters of subnetting should be the hosts of an IP address. Subnetting is implemented by utilizing a few bits from the host portion of an IP address to identify the physical network. These bits of the network identifier are found out by using the subnet mask. Every IP address consists of a subnet number and host ID. The purpose of the subnet number is to allow addressing to a particular LAN. Each subnet number is associated with a subnet mask, which gives the number of bits associated with the subnet number. Routing between LANs is based on the subnet number of the destination address. The host ID indicates the specific host interface on the subnet. The requirement is that all hosts on the identical network should have identical subnet masks. To find the subnet number, the subnet mask and the IP address undergo the AND operation [4].

For example, in the project, there are several subnets or separate networks that are interconnected by a switch, which is the Linksys wireless router. Behind one of the subnets lies the professor laptop. Behind another subnet lies the student laptop. Each router connected to the switch and each computer located behind each subnet has a unique IP address. The fundamental question is: How are clients identified on the network? Ultimately, the server, which is also connected to the switch, will foresee the operations within these multiple networks and give out commands. If a student laptop (for example, a Class B network 128.10.0.0) wants to communicate with the professor’s laptop, then the student’s designated IP address must be able to communicate with the IP address of the professor laptop. This is achieved by allowing the student IP address to reach the server and then from there, the IP address will be routed to the professor laptop, which is the final destination. Once this has been established, the student can now communicate with the professor by submitting questions. Technically, the student IP can
be subnetted using the first 8-bits of the host part of the IP address and is allowed to span different multiple physical networks, 254 per sec. The subnet mask could possibly be 255.255.255.0. In addition, the subnetworks could possibly be: 128.10.1.0, 128.10.2.0, …, 128.10.254.0. Furthermore, each of the subnetworks has a capacity of up to 254 distinct hosts, 128.10.XXX.1, 128.10.XXX.2, …, 128.10.XXX.254 [4].

Many Class A and B networks do not reach the full capacity for the amount of hosts it can hold, which means there is a lot of address space that can be utilized. The advantage of subnetting is that it allows this address space to be better used by dividing these big networks into smaller ones [4].

**DNS, DHCP, WINS**

Basically, DNS (Domain Name System) is a group of protocols and services in which a registry of names and addresses is provided. This is important because when finding hosts on the Internet, it would not be necessary to remember IP addresses when there is a registry with names that would be automatically retrieved. The advantage to DNS is that it allows interaction between the user’s computer, the network that it belongs to, and the Internet by utilizing user-friendly names that are directly mapped to their respective addresses [7]. These addresses may then be referenced back to these names [6].

A group of protocols and services that is similar to DNS is called WINS (Windows Internet Naming Server). In addition, WINS assigns an IP address to a certain computer in a network. It is mainly for use with computers that are operating with the Windows OS. WINS accesses a database that has automatic updates to the registry of computer names and their respective IP addresses [6].

DHCP (Dynamic Host Configuration Protocol) is a method to dynamically assign IP addresses, instead of manually or statically. It is a slightly more complex process
because each time a client logs on to the network, it must assign it an IP address. The advantage to DHCP is that a user does not have to manually enter a specific IP address for a specific hardware address because DHCP dynamically and automatically assigns IP addresses to clients [6].

For the purpose of the project, it was decided to utilize regular IP addressing rather than using DNS, DHCP, or WINS methods. This is because IP addressing is the method that most people use and thus, would be more convenient for the implementation of the project. It is complicated to deal with a registry of user-friendly names such as in DNS or WINS.

**Sockets**

Sockets originated in Berkeley as a part of the BSD distribution of Unix. Since network communications consists of several layers, sockets are abstract concepts used in software development to access networking hardware. A socket is basically the end point of a connection between two network nodes. However, an adequate understanding of sockets and their use requires a brief discussion on IP addresses and port numbers. As stated earlier in the paper, each node on the network must be assigned a unique IP address. However, in addition to the IP address, a unique port number must be assigned for different services on that node. For example, if two applications were running on one node, and both of those applications were communicating with external applications over the network, every data packet must be delivered to the correct application. By assigning ports to different services, nodes on the network can access specific services on the servicing node.

When a socket is created in two steps – defining its type and binding it to a port number. Since sockets were originally developed under Unix, they can be set to connect to Unix domain addresses with Internet domain addresses (IP). Unix domain addresses
are simply filenames, while Internet addresses are in the familiar 4-byte number format (e.g. 192.168.0.1). Secondly, a socket must be set to communicate using TCP (Transmission Control Protocol) or UDP (User Datagram Protocol). TCP breaks the data into numbered packets for transmit and resequences the data on receive to provide a reliable level of communication. For less critical applications, UDP sends packets but does not resequence them on receive; this responsibility must be used by a higher network layer application if reliable communication is desired using UDP. Once the socket is setup using these two definitive parameters, it is bound to the port number within the service providing application. Any processes on the network looking for the application services would only have to connect to that port number. Figure 3 shows the described process.

In our project, our server application opens a socket using Internet domain addressing and TCP. Since almost all networks use Internet domain addressing, this will ensure compatibility with most existing networks. The application then opens a main port for listening on the dedicated server. It is assumed that on a large system, this server will only be shut down for large network administration purposes. Therefore, as long as the main network is available, any client looking for Wireless Classroom Assistant services can connect to the socket on this main port. After some security verifications the server
application retrieves the correct class port number and returns it to the client. The client then connects to this new port number and joins the class.

The clients in our case consist of either the professor or the students. The client opens up a socket using Internet domain addressing and TCP, and then attempts to connect to the server running our server application. The connection is made using the IP address of the server and the port number of the service provider. Once connection is established, the server assigns a new class port number to the client, and the client connects to this port number to join the class.

While Perl comes with built in support for sockets, there is an IO::Socket module which provides an object oriented approach to creating sockets. Address and port information is used to create the object, and therefore a socket. Although the object supports DNS and WINS addressing, we will be using IP addressing to ensure network compatibility. There is a slight difference in its operation when connections are established compared to using the built in Perl socket functions. After the server application sets up the socket, it begins to listen for connection requests. If connections are accepted, the accept method returns another socket class – that is, another socket handle is returned. This socket handle is used for communication with the connected client. This frees the original socket to continue listening for new connections. In this method, the single socket object in the server application can accept connections from multiple clients – a feature which is essential in our application.

With that in mind, our server application handles the task of multiplexing all the clients, and ensuring that client messages are routed to the correct clients. For example, when different classes attempt to connect to the server, the application only routes messages between clients within the same class. There is also capability for the system to recognize a particular client as a professor, and route professor-specific messages to that
client. This is an important security issue as well, since clients requesting to join a class must send a login request to the professor-client. A general broadcast to all would clearly compromise security.

**Architecture**

When setting up any wireless connection, hierarchy and positioning is very important. The basic scenario for our wireless environment is that we want a client to be able to establish a connection with his professor. Ideally the client would establish a direct connection to the professor, and each would write to a socket on their end of that connection. The challenge is finding the professor's application if it is on a different subnet. One solution is to establish a network as shown in the following diagram.
In the diagram, there is a Broker, a client professor, and two student clients. For the most general situation, each client is in a different subnet. For the student client in Subnet A to find the professor’s computer it should not be difficult. Since the student and the professor are in the same subnet, all the student would need to do is search for computers within his subnet for the IP address of the professor. The problem with connection occurs when a student client in subnet C or B tries to search for the IP of the professor. Since both these clients are in different subnets than the professor, when searching for the professor’s IP address they will not find him because they are in the same hierarchy.

To fix this problem, we needed to have a Broker which is on a higher level than all other clients (professor or student) who connect to the network. By using this method, we are able to give this Broker a static IP address so that every client will be able to find and connect to it.

**Broker Operation**

So the way the network is set up is by having every client connect to the Broker on the same port (say 10254). If a professor logs onto the port with the correct identification (password), he will be able to “create a class”. What happens is that the broker will receive the professor’s request, and assign him a new port (See How ports are assigned). The professor will then log off of the server’s main port and sign onto the newly assigned port.

Once the professor has logged on, any registered (meaning student on the professor’s database) student will be able to log on. They will first log onto the main server just as the professor did (to the main port). They will tell the main server what class they want to sign on to, and their password. The server will read in the class name sent, and send the password and username to the professor teaching that class. The
professor interface will then search the professor’s database to see if the username and password are matched. If there is a match, he will send a message to the main server telling it to accept the student. The server then sends the class port number to the student client, who in turn opens a socket connection on that port. The student is now part of the class, and can ask questions, and listen.

Communication between a student and professor works through the main server. The professor is logged on to the main server (broker) and listens, while students send questions. It goes through a port that is created by the server specifically for that class. All the students obtain the port number, open a socket to that port, and listen to the same port. In this way, the students and the instructor can send and receive questions from the server.

**Port Assignment**

Ports are assigned by the server. The main server starts with an empty array of free ports. Once a professor logs on, the server will first search his array of free ports, if the array is empty, the server will create a new port and add it to an array of used ports. When a new professor logs on and starts a new class, the main server will search the array of free ports, and if it is empty, will create a new port and add it to the array of used ports. When a professor is finished with his class and decides to log off, the port number will go into the array of free ports. The server then deletes this port from the array of used ports. This port is now open for future professors to use. For example, the next professor that logs into the server, the server will search through its array of free ports and find that port. Then this port will be deleted from the array of free ports and get stored into the array of used ports.

**Additional Features**

Since the connections between the professor, student and server all occur through
ports, it is crucial there is a method of identifying certain messages from others so that they will be interpreted differently. Therefore, every time a message is sent, it will begin with a double colon and then a keyword stating what sort of command or message is being send or received. For example, when ::question::what time is it? is send to the professor, the professor knows to parse our the word question, and interpret the rest of the message as a question. If a professor sent a message ::rate to the student, the student would know to interpret the message as a request to be rated, and send the professor a rating.

**Booting Students**

When a class port is created, every student in the class will be listening to the same port. There is therefore no way to send a message to an individual client (professor or student). Since every client in the class will be listening to the same port, a boot cannot be specified to one particular client. This caused an issue when dealing with the boot mechanism. To solve the situation, it is the actual student that does the booting, not the professor. The professor sends the message ::boot::username to all the clients listening to the class port. Each client will then receive the string and compare their own user name to the one sent. If there is a match, the student will boot him/herself off. Since confidentiality is an important part of our environment, a method to find an identify for each student while still keeping it anonymous to the professor was needed. Therefore the professor stores every connected client inside an array. Each student username is stored in the array as they logon, but all that is printed for the professor to see is student #1,2,3 etc. It is important to consider the situation when a student logs off. If this is the case, the professor needs to delete that student from the array so that he is not printed or considered in the class anymore.
Professor Rating system

The algorithm of the rating system is relatively simple. Every time a student logs on, his name is added to the professor’s student array as mentioned before. Every 10 seconds (a timing scheme we chose) the professor will send a message to everyone listening to the port, asking them to send a rate. The student will receive the message ::rate and know send the current position of their slide bar to the professor. The professor will then receive a message from each student: ::profrate::username::#. The professor will then parse out the number received an add it to the running total. This total is then divided by the number of students currently logged on. This average number will then be printed to the professor’s screen.

Conclusion

While it may seem that the Professor Controller is a mere replica of an instant messaging application, it should be noted that the system is not a friendly chat program. The only interaction that takes place is between the student and the instructor. The previously powerless student can now rate the instructor on a real time basis and ask questions they would normally not ask. The professor, on the other hand is able to see his current process through the ratings, as well as view and ‘boot’ students who are abusing their new found power. The instructor’s interface is also a friendly tool in preparing quizzes for students (this is done through a simple template in a text file) and saving the session for future reference.

The simplicity of the system is one of its best features, however, in practice, the algorithm used in networking the professor and the student is made complex by the implementation of the server as the bridging communication means. As mentioned earlier, this approach was adopted to enable students and instructors whom are on
different subnets to communicate to each other. This would not have been possible without the a broker always listening at the server for connections that need to be made. Otherwise, the communications will be limited to the Local Area Network that each group of computer is in. With a broker program operating at the server computer that keeps a port open to listen for classroom connections, gateway and subneting problems are overcome.
Appendix
Using Wireless Classroom Assistant for Dummies

This appendix describes proper usage of our software system.

The files included in this system are:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>instructor.pl</td>
<td>Main professor's interface and functions</td>
</tr>
<tr>
<td>student.pl</td>
<td>Main student interface and functions</td>
</tr>
<tr>
<td>server.pl</td>
<td>Liason application to be run on a well-known network server</td>
</tr>
</tbody>
</table>

Installing Perl
ActiveState distributes a perl package named ActivePerl for Windows platforms. This can be found at:
www.activestate.com
UNIX and Linux distributions of perl are also available
Mac OS X comes with Perl 5

Compiling Perl to win32 exe file
Perl/Tk scripts can also be compiled into an win32 executable using Perl2Exe. This can be found at:
www.indigostar.com/perl2exe.htm

Setting up the server:
If perl is installed on the system, the file "server.pl" can be run simply by typing:

```
perl server.pl
```

The server is then automatically set up to listen for incoming connections and multiplex messages from several clients

Setting up the and using the Professor interface
To run the professor client, run:

```
perl professor.pl
```

Connecting to the server:
1. From the pulldown menu, select Connections⇒New Connections…
2. Enter the professor username, server login password, IP address, and the name of the class to be created. Click Connect.
3. If authentication is successful, server will return acknowledgment to professor.

Other professor features
- The professor can identify unique student questions in the question window via an anonymous student tag.
- If the student becomes unruly, professor can select the student to be booted on the student list window, and click on the boot button. Student will lose access to the system for the remainder of the session.
The professor has the option of preparing a text based file and distribution to all other student clients. Click on the Prepare File button.

If a log file of the session is desired, select File→Save As… from the pulldown menu, and enter in the filename to save the questions and Instructor notes.

**Setting up and using the Student Interface**

To run the student client, type:

```perl
perl student.pl
```

**Connecting to the server**

1. From the pulldown menu, select Connections→New Connections…
2. Enter student ID, associated password, classname, and the IP of the main server.
3. If authentication is successful, server will acknowledge, and student will be logged into class.

**Basic features**

- To ask a question, type in question in the question box and click Submit.
- Student can also choose to send quick general questions using the radio button questions provided on the right. Select the desired radio button and click Submit.
- At any point in time, the student may slide the Professor rating slider bar at the bottom and rate the professor. This rating is polled periodically and reported to the professor.
Project Resources

Equipment Used:  2 Laptops from Winlab: 1. Mobile 52
                2. Mobile 39
                1 Access point from Winlab: 1. AP 28
                1 802.11 computer card from Winlab : 1.

Personal Equipment Used:  2 Laptops w/Access Cards
                2 Wireless Acess Points w/ Cat5 Ethernet Cables

Division of Work:

Lal Bharwani       - GUI for Professor and Student Interfaces
                    - Integrated Professor client functionality with interface
                    - Project paper: Intro, feasibility, conclusion

Frank Shao         - Programmed Student client, and server application
                    - Set up port and socket connections between Student and
                    - Project paper: Sockets section, Appendix, proofing

Aviram Shapira     - Programmed the Professor client communication with the Broker.
                    - Log-In and authentication functionality (excel database interface)
                    - helped integrate the booting system.

Andy Wang          - project paper: IP Addressing, Subnets, DNS, DHCP, WINS
                    - Project Paper Integrator

Kerry Wong         - Debugged Professor client functions
                    - Obtained and setup hardware (wireless access points, addressing)
                    - Project paper: 802.11b, QPSK, CCK, DSSS
References

   http://info.iel.unipi.it/~anastasi/papers/clustercomp04.pdf

   http://www.webopedia.com/TERM/I/IP_address.html


   http://www.oreilly.com/catalog/coreprot/chapter/appb.html


   http://www.webopedia.com/TERM/W/WINS.html


    http://woozle.org/~neale/papers/sockets.html

    http://compnetworking.about.com/library/weekly/aa042400c.htm

http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci212811,00.html


http://www.amk.ca/python/howto/sockets/


http://www.comptechdoc.org/independent/networking/guide/netportsandaddr.html


http://www.unixreview.com/documents/s=8472/sam0307g/


http://www.wdvl.com/Authoring/Languages/Perl/Apache_Perl/apache_perl1_2.html


http://www.networkcomputing.com/1115/1115ws2.html


23. Chung, Jong-Moon, Dr. “Introduction To IEEE 802.11b Wireless LAN.” TCOM 5990 Wireless Communication Lab 1. School of Electrical and Computer Engineering Ver. 0.1 Draft. Director of ACSEL & OCLNB Laboratories, Oklahoma State University.

