Geographically Separated WSN Communication Management Project

gfour: Michael Gubbels, Stephen Mkandawire, Scott Robinson, John Tooker

Abstract—Circumstances arise when geographically separated wireless sensor networks need to be joined together as if they were a uniform network. In this paper presents a system that address this issue, enabling wireless sensor networks to communicate via Internet. This system requires wireless sensor networks to register as clients to a centralized server that manages client identification and facilitates communication between clients. Clients can communicate in three ways in the current implementation of this system. A client can (1) relay a message to another client, (2) broadcast a message to all clients, or (3) send a message to the server to store in its database. A client wireless sensor networks must have Internet connectivity. In our implementation, wireless sensor networks connect to the Internet through an Internet gateway, which consists of a sensor mote connected via serial connection to a computer with Internet-connectivity that is running the client software that relays messages from the gateway mote to the server. We refer to the sensor mote that is connected via serial to a computer as a sink mote. Similarly, we refer to the computer that the sink mote is connected to as a sink. A client is an application that is connected via an Internet stream socket to the server being run by some hardware platform. We use the term client platform to refer to the hardware platform running the client application. A client wireless sensor network is a wireless sensor network that includes at least one sink mote that can communicate with a sink.

Index Terms—Wireless Sensor Networks, Server, TelosB, TinyOS

1 Introduction

THIS project links multiple wireless sensor networks (WSN) together via the internet. There are many circumstances where geographically separated WSNs should be viewed as one large network. This project is a server interface that can host multiple WSNs dynamically. Each WSN needs one sink node to connect to the internet and relay information to the server. The server takes care of directing traffic flow similar to the way traffic is directed within the networks themselves. Figure 1 shows multiple WSN connected via central server.

The need to connect and monitor multiple separate networks motivates this design. Take building maintenance for example, suppose there are pressure sensors in the foundation of every building. Within the building, data could be transmitted and processed, but a campus may have many buildings to maintain. If each building's network is connected together, data can be analyzed together, trends can be monitored (i.e. what results an earth quake has on each buildings foundation) and patterns discovered. A building foundation business could monitor hundreds of buildings without physically visiting any of them, saving money.

Using a server also offers many advantages. First, the server can perform processor heavy in-network computation analysis. It can also aggregate and store data. Client applications can receive or monitor data and act upon it when needed without the worry of energy consumption restrictions (like a user interface, or graphics processing software). Another advantage is the ability to implement "virtual networks" as well. These

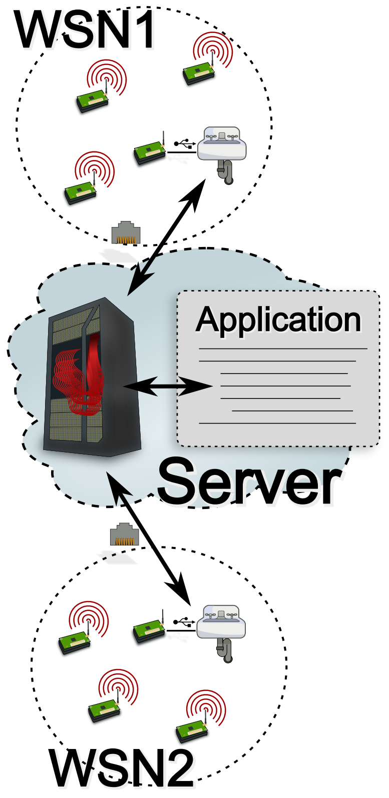

Fig. 1. Multiple WSN and applications Connected via a central host

networks could be used to test the system and are really just a subset of the client applications that could be implemented.

2 Related Work

As wireless sensor networks is a fairly new research area, there has not been a great deal of research done regarding geographically separated sensor networks communicating via a network connected server. Although, since this project covers a very broad concept there are a few other projects that contain related ideas. There are not any other published research projects aimed at using a server to hide the fact that not all nodes connected are in close proximity geographically. Although there have been papers describing the use of an Internet-connected node to allow remote access to a WSN.

At the University of Nebraska-Lincoln, a project was done on Heterogeneous Sensor Networks [1]. It focused on the interoperation of multiple remote sensor networks via the Internet as well as creating a new cross-layer MAC, routing and transport protocol supporting next-generation hardware. Their main focus was on the protocol and only briefly explained how the sink nodes would interface the wireless sensor network to an Internet-connected sink.

Previous work and research has been done to find efficient ways to collect data from a WSN and compile it for viewing on an embedded web server for remote users to view and interact with. This was demonstrated in a paper by Dejan Raskovic, Venkatramana Revuri, David Giessel and Aleksander Milenkovic [2]. They presented an implementation of a platform-independent embedded web server and its integration in to a network of wireless sensor nodes. Some services provided by their server includes remote monitoring, email alerts about critical issues in the WSN, secure access to modules that change the operation of the WSN and the ability to shut down sensor nodes. This project can be related to this one in that it integrates a way to communicate to the internet via a server and directs data between the server as well as the network..

From the lack of documented research in this area it can be seen that there is much to be discovered in terms of actual applications and concept implementation. As an emerging field, there is still much work that can be done in wireless sensor networks, and specifically sensor network communication over IP. This project explores the possibilities of Internet-connected WSNs and what can be done to effectively and efficiently communicate between each network.

1. Akin, Haccius, Schemm, Viets, Wang (2008) Heterogeneous Sensor Networks, *University of Nebraska-Lincoln*

2. Raskovic, Revuri, Giessel and Milenkovic (2007) Embedded Web Server for Wireless Sensor Networks, *University of Alaska Fairbanks, University of Alabama in Huntsville*

3 Project Description

The major milestones of the project are:

1) Establish communication between multiple, geographically separated WSNs
2) Send messages between WSNs
    a) Broadcasting messages to all connected clients
    b) Sending client-to-client messages through the centralized server (e.g., from one WSN to another)
    c) Sending messages client-to-server for storage in a centralized database (i.e., a client sends a message containing data to the server that the server stores in a database)
3) Create a simple client application
4) Create light shows and control the lights outside the Schorr Center as requested by WSN or a client application
5) Create a server that will facilitate the milestones listed above

There are three major components of this project: the server, sink (with mote), and the wireless nodes themselves. They all work together so that any single node can communicate with every other node, regardless of whether they are in the same physical wireless network. The server and its relation to each network are the main focuses of the project's design.

The server will be exploited as its computational power is much greater than the network nodes or even the network sinks. The server's first job is to manage the different wireless networks attached. It will also serve to process data via applications that will act like a network themselves.

The sinks will be the connection between the server and the wireless sensor networks, each WSN will require a sink. Each sink will have a special mote attached to it as well. The sink's mote will receive a wireless message and forward this message to the sink, which will forward the message to the server. Messages may go the other way: from the server to the sink to the mote to be transmitted to the WSN (using protocols and designs outside the scope of this project).

The wireless nodes themselves will act like any other wireless network. They will take measurements, process a little data and relay information to and from the source. The issues involved in isolated WSN will not be the focus of this project, other than a single node can transmit to any other node, regardless of its physical location.

To demonstrate the project's functionality, motes will detect a light source's intensity in multiple WSNs. A threshold will be set, if the sensed light's brightness crosses a that threshold, data will be sent from the sensor mote. That message will be picked up by the sink, transmitted to the server and the server will relay the message to a sink in another WSN or store that message in a database. A special WSN will contain the Light Gateway system used to control the lights on the side of the Schorr Center. A special, client application will also read the database, send messages and display information to the user in a graphics manner.

4 Project Details

4.1 Server

The server will direct network traffic, provide additional computation abilities and improve accessibility. The server may be transparent to the nodes themselves, just serving as a tool to transfer a message from one network to another, or the server may communicate with the nodes directly, both requesting and receiving information.

The server's most basic task is to manage the flow of packets received from registered clients. Only registered clients can connect to the server, therefore, a connected client is a registered client. Connected clients can communicate in three ways. They can (1) relay packets to another connected client, (2) broadcast packets to all connected clients, or (3) send packets to the server with data for storage into its database. All packets are 16 bits in length and consist of three fields: the (1) source address, (2) destination address, and (3) payload data. The source address and destination address fields occupy 4 bits, and the data field occupies 8 bits.

```
[source][destination][data]
```

This packet structure imposes functional limitations on the number of clients that can be registered on a system and the amount of data that can be sent in a straightforward manner (an application-level protocol could be devised that allowed arbitrarily amounts of data to be sent serially using multiple packets). In particular, since the source and destination address fields occupy exactly 4 bits, and client IDs are unsigned integers, there are exactly 16 unique client IDs available (and therefore, 16 uniquely identifiable clients). Two of these client IDs are reserved for identifying the server itself and to specify all connected clients simultaneously. These reserved client IDs are shown in Table 1. Our implementation includes four client wireless sensor networks and one client application in addition to the server. Each of these clients were assigned a unique client ID, as shown in Table 1.

Messages may be send in a variety of ways. Intra-network messages will not be passed through the server, as the sink nodes will not send them. Inter-network messages will be passed transparently through the server to other networks. The is also a broadcast address, allowing for a single node to send a message to every other node (should that be required). Messages may also be sent directly to a server application. Each application will have an address, so the structure of the messages is uniform, despite the messages destination. Groups could be created as well, mapping a single address to a specific set of destinations, all handled by the server. Figure 2 shows three message relations.

The server may implement a virtual network or host application. This will allow real motes to send messages

Client ID (base 10)	**Client or *Purpose***	**Purpose or *Function***
0	*Reserved (Server)*	*Stores packet data in server's MySQL database*
1	WSN_1	Sends packets to server to store in its database
2	WSN_2	Sends packets to server to store in its database
3	WSN_3	Requests Lights shows from External Gateway's WSN
4	GUI Application	Reads the server's database, analysis its data, requests shows and provides a GUI to the users
6	WSN_4	Sends a light show request message 'directly' to WSN_3
15	*Reserved (all connected clients)*	*Broadcasts packet*

TABLE 1
Claint and client ID associations for our server, wireless sensor network clients, and application client implementations.

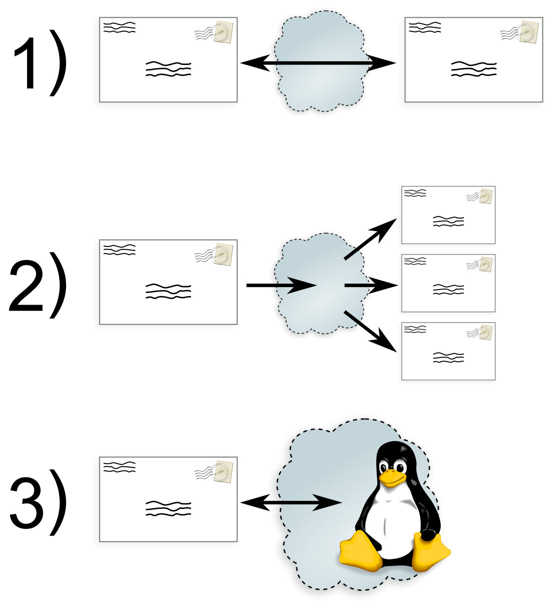

Fig. 2. Multiple Message Destination: 1) Point to Point 2)Broadcast 3) To/From Application.

to an imaginary device's address. Using the faster processor/memory on the server, much more computation can be done. Servers also have much more room to store and log data than nodes in the network using large hard drives and databases.

An example of the server's code is seen below. The pseudocode segment highlights the decisions that must be made when an incoming message are received. The server code was written in C.

```
void messageReceived(Message msg){
    // see if the address is valid
    // (registered) both the source
    // and destination address
    if (registeredAddress(msg){

        if (msg.destination == 0000){
            // this message goes to
            // he MySQL DB
            dbWrite(msg.payload);

        } else if (msg.dest == 1111){
            // broadcast this message
            // to all WSN
            broadcast(msg.payload);

        } else {
            // message has a specific
            // destination
            forward(msg);
        }
        // could also match the
        // destination to a group ID
        // and forward the message
        // to specific groups.
    }
    // else ignore this message
}
```

Code Segment - Server

User interfaces are also important, the server will be able to serve a webpage interface or provide a socket for a client application. A web interface could be set up allowing messages to be sent directly to and from nodes from a browser. This could be done directly or as part of an interactive application or used by developers to test the network.

4.2 Sink

The sink's role is straight forward. It watches for messages from the server and transmits them to its connected mote via USB (serial) for it to disperse to the network. The sink also waits from messages from its WSN and relays them directly to the server. The sink does not require any additional computation, though it could be used for some in-network processing, the server could do the same processing much quicker. Code Segment - Sink highlights the sink's simple role, the actual code was written in C.

The sink does need to be connected to the internet and supplied with sufficient power. It may not be feasible to have a remote, battery powered sink node, though as solar power and cellular technologies develop, this limitation shrinks.

If a message encoding scheme were implemented, it would be the sink's duty to encrypt and decrypt the messages.

```
while(1){
    // Communication Order
    // 1. Socket -> Serial
    // 2. Serial -> Socket

    msg = checkForMessage(socket);
    if (valid(msg)){
        forward(msg, serial);
    }

    msg = checkForMessage(serial);
    if (valid(msg)){
        forward(msg, socket);
    }
}
```

Code Segment - Sink

4.3 Sink's Mote

The mote attached to the sink will transform wireless messages to messages sent over the USB and vice versa. They will be implemented using MicaZ and TelosB nodes. This mote's only job is to wait for messages and relay them.

Besides any error correction (i.e. BCH code), the mote will not have to decode or interpret the message. It will either transmit it directly to the sink (via serial) or broadcast it to the network preserving the address and data information (via IEEE 802.15.4).

4.4 WSN Motes

As was said before, the mote's role will not be the focus of this project. Simple sensors data will be directly mapped to show requests. In general, each mote would have a unique ID number (address). This project abstracts the actual WSN as a single node attached directly to the sink.

5 Results

Each of the milestone requirements have been met. To demonstrate this, four WSNs were used. One of the networks was 'attached' to the Schorr Center's light controller network; two other networks sent messages to a database on the server, and one network sent show requests directly to the light controller network. The server stored information in a database and a client program periodically queries the server and sends a light show request based on the total lights on (as stored in the database). This client program also offers a user interface. This setup is described in Figure 3.

The server meets the specified requirements. It can receive and relay messages to other networks. For the demonstration, a set of networks sent data directly to the server's database application (which has a message address). Another network sent data through the server directly to the light controller network. A client application queried the database and sent show request

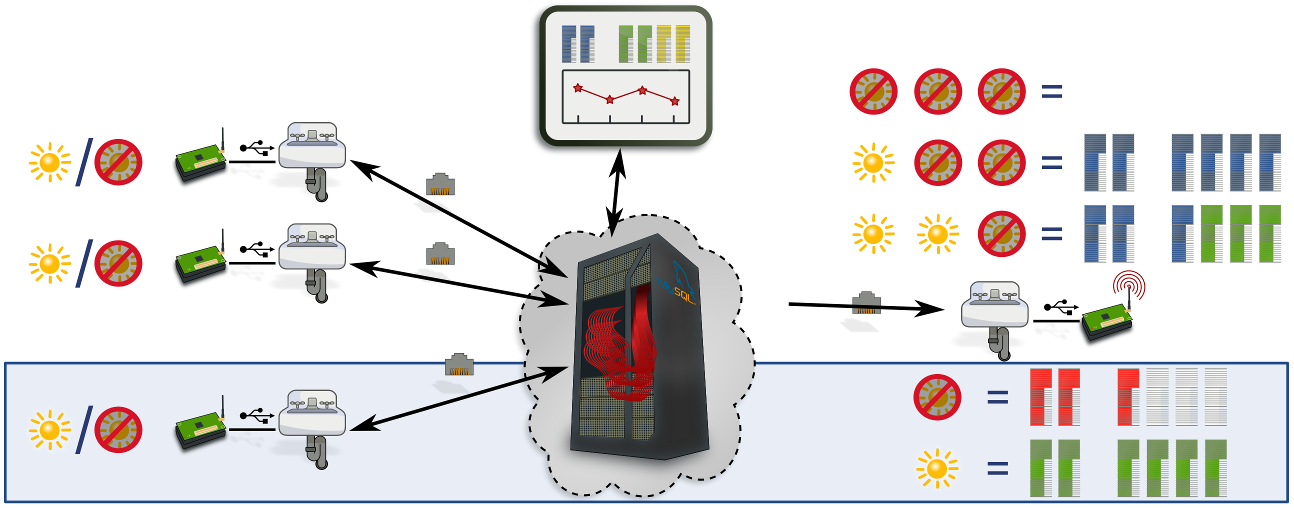

Fig. 3. A figure demonstrating the communication network that demonstrates the project functionality. The upper left WSNs send their light data directly to the server's database, the client program (upper center) displays a user interface and emits a show request message. The lower left WSN sends a show request directly to the light controller network, shown on the right. The light shows in the upper right correspond to zero, one, two and three lights on (via the database) and the bottom right light shows correspond to one of two show requests from the lower left WSN

messages directly to the light controller network at certain intervals. The server was capable of boradcasting messages, though this was not demonstrated. A Linux server was used, but this is not a requirement.

The server also provided a webpage that allowed users to register a network. Messages sent to the server from unknown address were ignored. A more advanced form of security was not implemented, but could have been added without changing the purpose of this project.

TCP socket communication protocols were used to transmit messaged from the sinks in each WSN to the server itself. Messages were transmitted fairly quickly from network to network. It took less than one second for a show request to be sent from one network and appear on the External Light Gateway's console window (in another network); but a comprehensive speed test was not administered.

The WSN themselves consisted of just one sensor mode (TelosB) and one sink (Linux laptop). Each mote sensed the ambient light, when the light crossed a certain threshold, as message was sent. Most of the motes sent a message directly to the database on the server, though one mote sent show requests directly to the light show's network. The bulk of the code is summarized in the Code Segment - Mote, though the actual code was written in nesC:

The client program showed the dynamic functionality of this project. Here another machine ran code that interfaced directly with the server and with another WSN. This program periodically queried the database to see how many networks were currently under bright light. This number was converted to a show request and sent out; a visual representation of the Schorr Center lights was also displayed. The program also keeps track of the

```
event void timer0.fired(){
    //read the light sensor
    call Read.read();
}
event void Read.readDone(data){
    if (data - oldData > THRESHOLD){
        // light changed
        oldData = data;

        // update LEDs for debugging:
        updateLED(data > THRESHOLD);

        // source, destination, payload
        send(MY_ADDR, 0000
                , data > THRESHOLD);
    }
}
```

Code Segment - Mote

ambient light history and displayed this as a graph on the screen. A code sample showing the period checking of the database, sending messages and updating the user interface is described in Code Segment - Application; the original code was implemented in Java.

The lights themselves correspond to one of six show requests. Figure 4 represents the shows available. Shows one through four represent zero, one, two or three WSN networks having bright light. These show requests came from the client application and only represent the WSNs that send their data to the database.

Shows five and six, which correspond to requests from the WSN that sent show requests directly to the light controller.

The lights were not updated too frequently (every

```
void timerThread.run(){
    // the database is on the
    // server, query it via
    // the network.
    bResult = queryDBviaSocket(socket);

    // count the number of lights
    // on as recored in the
    // database query result.
    int lightsOn = 0;
    for each dbResult.entries {
        if (dbResult[i].isOn){
            lightsOn++;
        }
    }

    // update the line graph GUI
    updateGraph(System.currentTime
                , lightsOn);

    // update the light display GUI
    updateLightsImage(lightsOn);

    // actually send a show request
    // message to the server
    sendMessage(lightsOn, socket_info);
}
```

Code Segment - Application

30 seconds) as they seemed to freeze if refreshed more often. The light controller gateway did show that each message was received correctly even if the light controller itself was frozen.

Multiple programming languages were used for each component showing the flexibility of the systems as a whole by using existing networking standards.

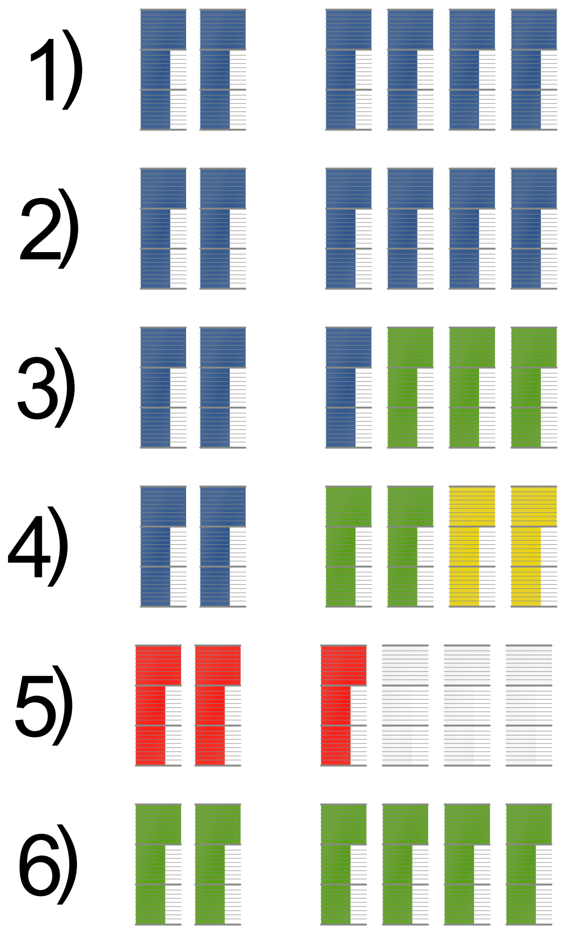

Fig. 4. The different shows available. Shows 1-4 correspond to the database. Shows 5 and 6 correspond to the mote directly communicating to the light controller's WSN.

6 Conclusion

The geographically Separated WSN Communication Management Project succeed in connecting multiple wireless sensor networks together while offering a dynamic communication scheme. The lights on the Schorr Center and client interface show the different ways to communicate and process information, from point to point communication to database storage and remote client processing. As solar energy is harnessed more effectively and cellular coverage grows, many remote WSN cites will be able to take advantage of the connection benefits that geographically separated communication can provide.

7 Future Work

The work presented in this paper could be extended in many ways. In our work, we focused on a designing and implementing a centralized server and client sinks to enable wireless sensor networks to communicate. We are interested in extending this work to develop a system to label the data stored in the server's database according to some characteristic of the data, such as the type of information it represents, or the nature of the data on a conceptual level (e.g., noisy data, weather data, temperature data, geographical, spacial and temporal labels, etc.), or a combination of different kind of labels. The purpose of labeling the data is to allow queries to be submitted to the server by client application using a query language that doesn't require client application developers to know the details of any particular wireless sensor network client. Ideally, client application developers would be able to query the server to return data according to the purpose of the client application and needs of the application's developer. These queries might be made using relational operators or constraints, such data type and spatial or temporal constraints.

Another opportunity for future work is in developing automated services associated with particular wireless sensor networks, for use by either anonymous or registered clients, which might be other wireless sensor networks. For example, a service might provide weather forecasts predicted using some predictive model and data aggregated from multiple wireless sensor networks associated with the geographical location of interest.

One immediate extension that would be very useful would be to expand the client-server application layer protocol used for communication. To facilitate transmission of large data sets (such as results from a database query), the protocol could be extended with standard message fields that correspond to queries and messages for indicating the beginning and end of serialized data being sent. When a large data query is requested, it may be appropriate for the server application to create a separate process that exists for the duration required to send the data. Since the reliability of client WSNs can vary significantly, the protocol used to relay data should be appropriately robust or allow the protocol, error correction, and error handling to be (optionally) specified per query.

Communication could be expanded in a number of other ways. The server could be extended to allow UDP clients that are in environments where constant connection doesn't make sense and data is submitted to the database when possible. A variation of broadcast could be implemented that allows a WSN to broadcast to a subset of the other WSNs. The server and client functionalities could be distributed so each client could function as a combined client and server to enable clients to establish direct connections via Internet sockets and bypass the centralized server and communicate in a decentralized manner. Permissions could also be assigned to clients during the client registration process that could determine the types of communication operations that a client would be allowed to perform.

8 Appendix: Compiling and Running Code

8.1 Prerequisites

The applications written for this project have various prerequisites. A HTTP web server with PHP and MySQL support is required. A MySQL database named *wsnhub* is required with tables named *clients* and *data*. This database and these tables can be created using the script file `'Web Server/deploy_website.sh'`. Also required are the MySQL C API and the Tiny OS C API.

8.2 Server (Client ID = 0)

To compile the server, enter:

```
cd Server
make
```

To run the server, enter:

```
./server
```

This starts the server, which listens for TCP socket connections on port 9034.

8.3 Motes - TelosB (Client ID = 1,2,3, 6)

- **ID = 1, 2, to DB** - Motes that send data to the database directly (can set the ID for each mote)
- **ID = 3, to External Light Gateway** - Mote that talks to the External Gateway Light Controller
- **ID = 6, Point to Point** - Mote that sends a show request directly (through the server) to the light controller mote

You may need to adjust the ttyUSBn based on how many TelosBs are plugged into your computer. To compile the show requester TinyOS nesC code for the TelosB platform and install the compiled application to a TelosB mote accessible on /dev/ttyUSB1, enter:

```
cd "Client WSN Sink - Database Giver"
make telosb install,1 bsf,/dev/ttyUSB1

cd "Client WSN Sink - Show Requester"
make telosb install,3 bsf,/dev/ttyUSB1

cd "Client WSN Sink - Light Remote"
make telosb install,6 bsf,/dev/ttyUSB1
```

8.4 Client Sink

Place code on a Linux server with TinyOS installed (or at least the C serial package portion of TinyOS). A programmed telosB mote needs to be pluggled in order to run.

To compile the sink client application, enter:

```
cd "Client Sink" make
```

To run, enter:

```
./client <device> <rate> <host IP>
<client ID>
```

For example, to start the client sink for the light remote sink mote (Client ID = 6), accessible on /dev/ttyUSB0, enter:

```
./client /dev/ttyUSB0 115200 localhost 6
```

Or, to start the client sink for the show requester sink mote (Client ID = 3), accessible on /dev/ttyUSB1, enter:

```
./client /dev/ttyUSB1 115200 localhost 3
```

8.5 Client Application (Client ID = 4)

The client application requests data directly from the server's MySQL database, which is used to generate show request packets, relayed through the server to the show requester client wireless sensor network (Client ID = 3).

To compile, enter:

```
cd "Client Application" make
```

To run, enter:

```
java ServerApp
```

8.6 Light Shows

The light shows are located in: `./Light_Shows` Each show comprises two files, x.data and x.size where x is the number of the show. Copy each of these files to the group folder on the internal gateway /var/www/4