Distribution Management System Using Computer Networks (DMSCN)

Hamza Abu Ajamia, Salman Qabaja
Department of Electrical and Computer Engineering
Palestine Polytechnic University
Palestine
hamza a a@student.ppu.edu,
sqabaja@student.ppu.edu

Eng. Ayman Wazwaz
Department of Electrical and Computer Engineering
Palestine Polytechnic University
Palestine
aymanw@ppu.edu

Abstract -- DMSCN is a Network programming course project that aims to manage control and data acquisition from distributed nodes using computer networks. The system contains a group of clients (nodes) that are connected to a central server that collects data from clients. Each client manages and controls a set of hardware devices locally, and is presented as either a TCP Client (TCPC) or UDP client (UDPC). TCPCs are securely connected to the server, while UDPCs are not. Moreover, the central server has a backup server that serves as replica whenever the main server fails the backup will work without any losses.

I. Introduction

DMS's is a collection of applications which shall monitor & control the entire distribution network efficiently and reliably. It acts as a decision support system to assist the control room and field operating personnel with the monitoring and control of the electric distribution system, effectively Improving the reliability and quality of service in terms of reducing systems outages, minimizing failures, and maintaining proper levels of service as the key deliverables of a DMS[1] . Far from the complications of distribution management systems this project comes to prove the ability of Computer networks of handling this problem. Each client in the system is connected to a set hardware (a collection of sensors and electrical machines), where if for example the data (temperature, CO2, and Pressure) that client collects exceeds certain levels; the client sends may send an SMS message or an email to the system administrator.

Where each client has control over its own hardware, clients can send their data to the server for propose of storage and data sharing. The central server accepts connection using either TCP or UDP protocol. TCP connections are established when TCPC requests a connection. Client then can exchange data with server containing user name, text message, and the data collected from hardware (Sensors). Since UDP is connectionless protocol UDPC's works in different way of TCP's. When UDPC user needs to connect to the server client broadcasts a Message on affixed port carries a request to the server. Only the intended server responded to this message with a message that contains a multicast group IP and list of clients that are connected to the server. In turn the client user has the choice to join the group. If the user chooses to join the group client sends a message that carries user name. When server receive the message it adds the new user to the users list and sends update carry the new user information (user name, IP, and port) to the group of connected clients. client then can send data to server as Unicast (Peer to Peer), to selected client as Unicast (Peer to Peer), or to the hole group of server and connected clients as multicast (peer to group). The system contain backup server for the UDP. When backup server runs, it sends request to the main server. In turn TCP session established between the main server and the backup server, where main Server sends every data and controls received to the backup server. If the Main server fails the backup will immediately take its place without any losses. The TCP connections between TCPC's

and server are secured using JSLOC protocol. Both server and client in log's every incoming and outgoing message.

II. Methods

In this section will introduce the protocols that used and design for the purpose of this project starting from client server connections, message format, UDP connection protocol, backup protocol, security protocol, hardware connection, mailing and SMS protocol, logging method.

A. Used protocols

1. TCP

The will known TCP protocol Used for connecting TCPC's to the server.

2. UDP

Used as TCP for connecting UDPC's to server. It's also used for multicast and broadcast.

B. Designed Protocols

1- Message format

The message that the clients exchange with each other and with the server have affixed format. The following figure 1 shows a sample massage format

Temperature value	CO2 value	Pressure value	Text Message	User Name

Figure 1 Sample Data Format

2- UDPC's connection Protocol

This protocol describes the way that UDPC's connects to the server and how they exchange data. The following figure describes the scenario of UDPC's protocol. And the following procedures describe the process in more detailed manner. As shown in figure 2.

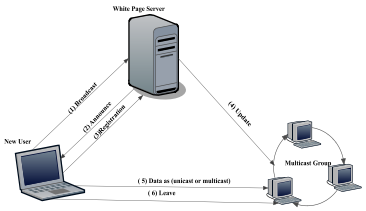

Figure 2 UDPC's connection Protocol

I. Broadcast

When a new user wants to join a chat group, the user sends broadcast searching for (white page server) that provides the registration service.

II. Announce

The server sends announce message that contains the group multicast address and a list of available users to the requestor user that sends broadcast message.

III. Registration

the user send here/his user name to the server.

IV. Update

When a new user register to the server the server sends a multicast message to the group to update available user list in each client.

V. Data

The new user can exchange data with any user within the group as a Unicast message exchange, or with the whole group as multicast message exchange.

VI. Leave

When user leaves the chat room it must send a multicast message to inform the group. "The server is member of the multicast group".

3- Backup Protocol:

A part of the UDPC's connection protocol where Backup server sends message to the main server that contains the backup server IP. The Main server sends start TCP session send the list of connected client users and then continually update the backup server. If the main server fails because of any rezone it the control is transferred to the backup server in order to fill main server place. Figure 3 describes how the backup server works.

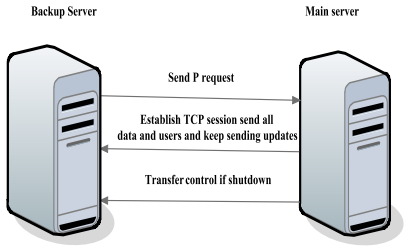

Figure 3 Back Server Implementation

4- Security JSLOC(J Secure Line of Communication)

This protocol is a mini version of the SSL (Secure Socket Layer) protocol [2] . This protocol uses hybrid cryptographic algorithms where it uses symmetric [3] and asymmetric[4] cryptography. It works as follow, when client connects to the server it requests server public key (A Symmetric Key), in turn server returns its public key to the client. The client then generates session key (Symmetric Key), encrypt it with server public key and sends it bake to the server. In turn server decrypts the encrypted session key and start the secure session with the client. The following figure describes the JSLOC working scenario.

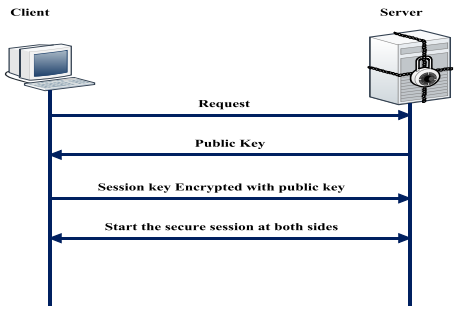

Figure 4 JSCLOC working scenario

5- Logger

Special type of logging mechanism that used for logging all the incoming and outgoing messages and data from servers and clients.

6- Hardwar connection protocol

In real time implementation of this system every client is connected to its own hardware. For simplicity in this project connection designed in different manner. Hardware is connected directly to central server. Clients connected to the hardware server to get the sensors values.

7- Mailing and SMS

Used in the interface with hardware. If one of the collected data crosses a predefined threshold the client sends both SMS and Email for the system administrator. SMS is done using GSM modem. While Emails sends using JAVA Mailing service.

III. Implementation

This project was implemented in both JAVA and C-Sharp programming languages. C-Sharp was used specially for interfacing with hardware. JAVA was the basic programming language for implementing the system components both on widows and Android operating systems. That means a mobile version of the project was produced especially for the TCP connections.

1- TCP Connections

Using JAVA Sockets and Server Sockets[5] that are specialized for TCP connections, this part of the project was accomplished. Sockets used in the client part for requesting connection to server in turn Server Sockets used to accept the connections with the client sockets. Data exchanged between the communicated parties using input and output streams [6] . The operation of reading and writing to I/O Streams is similar to File read write operations.

2- UDPC's Connection Protocol

Using JAVA Datagram Sockets and Datagram Packets [7] data traveled between sender and receiver. The data shape during transfer is as follow:

The protocol Frame structure:

Type	Data/Control information

Figure 5 UDPC Connection Protocol

The Type field is 1 byte field size. The protocol support the following types illustrated in table 1

Type	Encoding
Broadcast	'B'
Announcement	'A'
Registration	'R'
Update	'U'
Leave	'L'
Data	'D'
Backup	'P'

Table 1 Protocol data types

Data Formatting:

A. Broadcast format will contain only the type flag without any additional data

'B'	Empty

Figure 6 Broadcast Format

B. Announcement: will contain the group multicast address and a list of the available users

'A'	Multicast Address	Users List

Figure 7 Announcement Format

C. Registration: contains the user information (IP, port, username)

'R'	IP	Port	User Name

Figure 8 Registeration Format

D. Update :contains the IP Address and the port of the new User

'U'	IP	Port

Figure 9 Update Format

E. Leave :contains only the leave flag without any additional data

'B'	Empty d

Figure 10 Leave Message Format

F. Data: contains the flag and the data to send

For Unicast:

'D'	Data

Figure 11 Data Format

G. Backup: contains the flag and the backup server IP

'P'	Backup Server IP

Figure 12 Backup message Format

As seen in the backup state is supported by this protocol.

3- Backup Server

Backup uses both TCP and UDP protocols. The data must be transferred to backup server in reliable manner, because of that TCP protocol was chosen to serve this part of the project. UDP protocol was not adequate for this mission because unreliability in transferring data, so data transfer using UDP could be risky.

4- Security Protocol (JSLOC)

At the start of JSLOC protocol handshaking between client and server occur to exchange security credentials (server public key and the encrypted client session key). The server's public key traveled via network encoded using special type of encoding. And the client part client reconstruct the public key and use it to encrypt the generate session key which is transferred as normal encrypted text. Server receives the encrypted session key, decrypt it with its private key and use it to encrypt data to be send and decrypt received data. The JSLOC protocol is designed only TCP connections, because of the handshake part of the protocol that must be accomplished in reliable manner.

5- Logger

The logger is mechanism designed to log all incoming and outgoing data. The logger logs the start of the session (when server run) and the end of the session (when server closed). The data logged in the file system in the following format:

Source IP	Source Port	Protocol type	Direction of data	Date	Data

Figure 13 Logger row format

Where:

A. Source IP: is the sender IP.
B. Source Port: is the port of the sender.
C. Protocol Type: is the protocol user for transferring data on transport layer TCP or UDP.
D. Direction: is the direction of the data in or out.
E. Date: is the date when data logged.

F. Data: the incoming data or the outgoing data with the message format defined previously.

6- Hardware Connection protocol

The connection to the central hardware server is accomplished using TCP protocol. The server is connected to GSM modem (for sending SMS) and to a kit that contains a set of sensors controlled by PIC microcontroller. The interface between the GSM modem, microcontroller, and the computer is done using C-Sharp programming language via USB port. The microcontroller collects data from sensors and transfers them to C-Sharp program. When TCPC's and UDPC's request this data from the server the server sends the data to them. If the data exceeds certain limits the client will sends email to the administrator, and sends message to the central server. In turn the central server set a proper control (if temperature is high run the air conditioner) and sends SMS to the administrator.

7- Mobile Version of the project

A simplified version of this project was developed to run on android mobiles. The application functionality restricted to connect to TCP server and exchange data with it.

IV. Results

At the end of this project the following functionalities where been implemented successfully:

1- Main TCP/UDP server module, implemented in java and illustrated in the following figure:

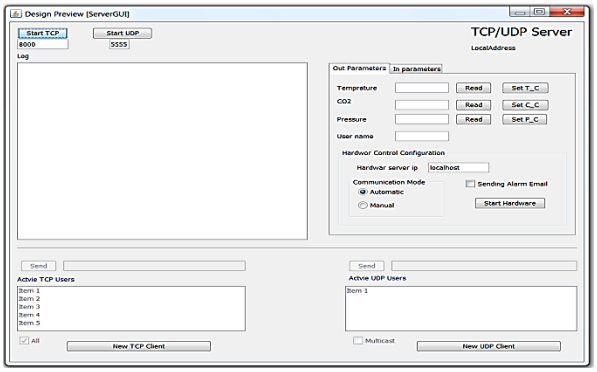

Figure 14 Java TCP/UDP Server

2- TCP and UDP client Modules, classes built to facilitate integration with other projects, implemented in java and C# to insure system compatibility the .Net and the Java platform. Testing these clients' modules is illustrated in figure 15, figure 16 and 17.

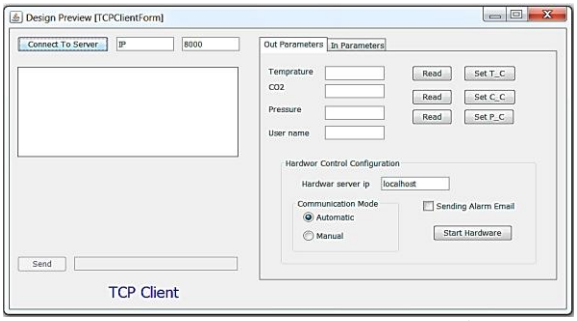

Figure 15 Java TCP Client Testing Interface

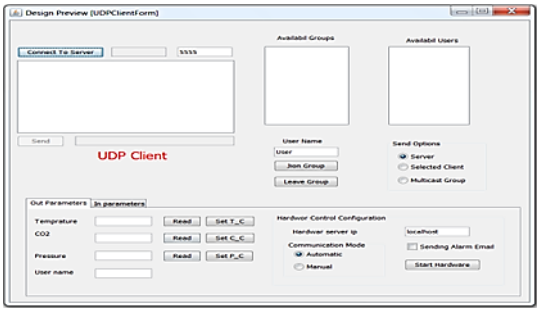

Figure 16 Java UDP Client Testing Interface

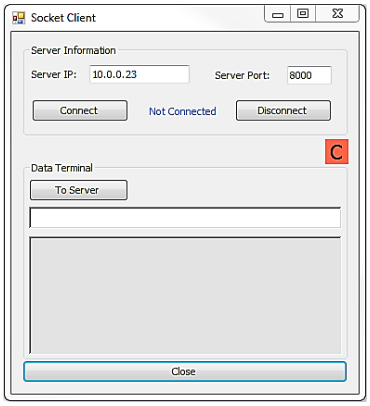

Figure 17 C# TCP Client Testing Interface

3- Backup Server, which provides backup features over UDP connections:

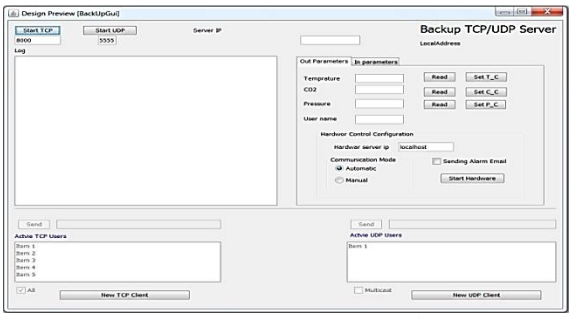

Figure 18 Backup Server Interface

4- JSLOC Secure communication implementation over TCP. The figure below illustrates the data in its encrypted and unencrypted form.

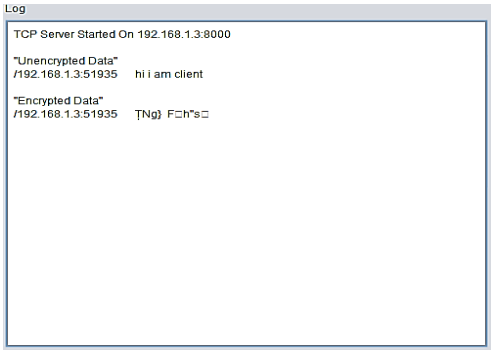

Figure 19 JSLOC Log.

5- Hardware client control panel, where it collects data from various sensors, and controls a number of electrical nodes. Moreover, it contains a module responsible for sending SMS's should the server require so.

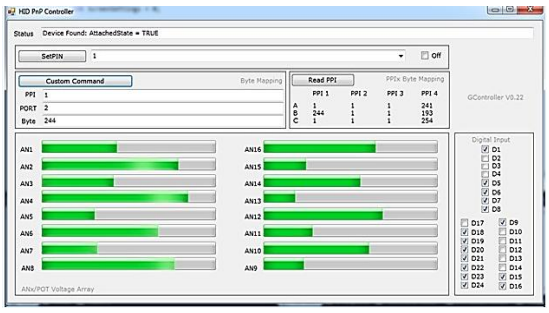

Figure 20 Hardware Control Panel/ TCP Client.

6- Automatic/Manual System operation, where it can automatically monitor the status of the sensor and send emails and alarms should any critical value be read. Or, it can be operated manually by a system administrator.

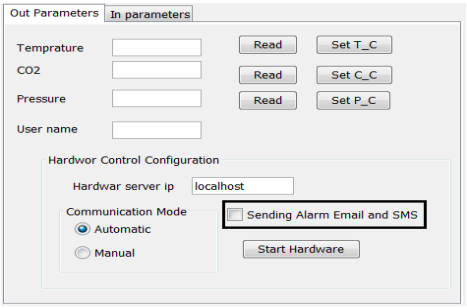

Figure 21 Manual/Automatic Operation.

7- System logging of data exchange for, classified and characterized by clients and sessions.

```
#########Session starts at May 19, 2012 12:12:20 AM#########

192.168.1.3 36332   TCP\in  May 19, 2012 12:12:20 AM              321
192.168.1.3 36332   TCP\out May 19, 2012 12:12:20 AM              kjh
192.168.1.3 36332   TCP\in  May 19, 2012 12:12:20 AM              321
192.168.1.3 36332   TCP\in  May 19, 2012 12:12:20 AM              321
192.168.1.3 36332   TCP\out May 19, 2012 12:12:20 AM              kjh
192.168.1.3 36332   TCP\out May 19, 2012 12:12:20 AM              kjh
#########Session ends at May 19, 2012 12:12:20 AM##########
```
Figure 22 Data Logging.

8- Mobile extensions for Android Devices, the systems reach is extended to mobile devices by migrating the java classes into android devices, as illustrated in the figure below showing message exchange as it appears on the android UI :

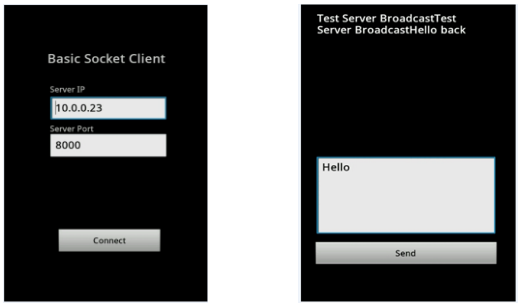

Figure 23 Android TCP Testing Interface.

V. Conclusion

Supervisory Control and Data Acquisition Systems also known as SCADA systems, which contain DMS systems, are in a very active area of research, where control, management and efficiency are the hot topics for research.

During this project, network management, registration, and control was successfully implemented over multiple environments, more specifically Java, C#, and Android. And so, effectively producing objects and modules that can be integrated into various others projects and increase their scope.

Acknowledgment

For Hassan Jaafreh, Mamoun Idies, Samer Iseed, Shamma Rumal, Da'a Asafra, Bayan Halawany, Ahoud Abu Al Seba'a, Sara Halahleh, Lama Sharabati, Afnan Ramadan.

References

[1]R'n'B. Distribution management system. Wikipedia the free encyclopedia. Last modified at 14:06, 16 April 2012. Accessed at 18:10,18 May 2012. http://en.wikipedia.org/w/index.php?title=Distribution_management_system&oldid=487668294

[2] WILLIAM STALLINGS, LAWRIE BROWN. (2008) "Computer security Principals and practice". Pearson Prentice Hall. (PP. 652-656)

[3] WILLIAM STALLINGS, LAWRIE BROWN. (2008) Computer security Principals and practice. Pearson Prentice Hall. (PP. 594-598)

[4] WILLIAM STALLINGS, LAWRIE BROWN. (2008) Computer security Principals and practice. Pearson Prentice Hall. (PP. 626-632)

[5] Esmond Pitt. (2006) Fundamental Networking in Java. Springer Science. (PP. 27-32)

[6] Esmond Pitt. (2006) Fundamental Networking in Java. Springer Science. (PP. 38-48)

[7] David Reilly, Michael Reilly. (2002) Java™ Network Programming and Distributed. Addison Wesley .(PP. 89-96)