School of Electrical Engineering and Telecommunications

ELEC4633: Real Time Engineering

Laboratory Module 3 – Supervisory Control Over a Network

Aim

The aim of this exercise is to:

- set-up client-server communication over a network;
- extend the real-time motor control software system to include supervisory control over a network.

Exercise 1 - Client Server (Network) Communication

You are to set up a simple network communication loop between two computers in the lab. This will be done with the construction of a server program on one computer and client program on the other. The communication (or client request) could be very simple. That is, the client may simply request a simple message in return from the server, and the server may simply "serve" the one client by waiting for a single request.

Simple example programs will be provided, server.c and client.c, which do initiate very simple server-client communications. You should set up the server program first which listens for requests from the client. On the server side, you will be required to have some understanding of, and use, the following functions:

```
sockfd = socket(int socket_family, int socket_type, int protocol);

void *memset(void *s, int c, size_t n);

uint32_t htonl(uint32_t hostlong);

uint16_t htons(uint16_t hostshort);

int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);

int listen(int sockfd, int backlog);

int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);

int close(int fd);

ssize_t write(int fd, const void *buf, size_t count);

ssize_t read(int fd, void *buf, size_t count);
```

For the client, you will need to have some understanding of, and use, the following functions:

```
int inet_pton(int af, const char *src, void *dst);

int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);

ssize_t write(int fd, const void *buf, size_t count);

ssize_t read(int fd, void *buf, size_t count);
```

1. Using the manual pages, ensure you have sufficient understanding of what each of the functions above are used for.
2. Compile the provided server and client programs using gcc: gcc -o server server.c and gcc -o client client.c.
3. Using a free computer in the lab, run the server-client example. You will need to note the IP address of the server machine, as this address will be needed by the client program. The client program is run by running the command ./client "¡IP address of server¿".
4. Modify the server/client programs so that the server listens indefinitely (not just for one connection). Show your working code to a demonstrator.

Checkpoint #1 (2 marks): Obtain a signature from your demonstrator.

Exercise 2 - Server-Client Communication in real-time control loop

In this exercise, you are to use checkpoint 1 above, and extend the real-time control software developed in Lab 2.

In this sense, you will have a computer which is connected to the motor process, responsible for its closed-loop control. This is the target computer. Within this computer, you will be required to develop a server program which waits for requests from a client program. On a different free computer in the lab, you will be required to develop a client which communicates with the target computer. This is the remote computer. From the remote computer, you will be able to send set-point information to the target computer, and also receive signal information from the target machine on request, so that the motor output can be displayed and monitored on the remote computer.

In essence, the display and set-point clients developed in Lab 2, are being moved to a remote machine, connected via the network instead of message passing channels.

1. Modify the client and server programs from Lab 2 to set up the network communication between the target and remote computers.
2. Modify the server and client code such that the remote computer is capable of requesting motor signal data, as well as requesting a change to the motor control loop set-point.
3. Show your working system to a demonstrator.

Checkpoint #2 (2 marks): Obtain a signature from your demonstrator.

Exercise 3 - Communication Fault Detection

In this exercise, you are to put in place a fault detection mechanism, whereby failure of the communication link between the remote and target machine will result in "sensible" shutdown of the controller. This detection would need to occur not just when the client(s) is trying to request a service from the server. That is, there would need to be some periodic check of the communication link.

You will need to give some though as to how the detection of the "broken" link will occur, as well as how you will then subsequently shutdown the controller sensibly.

Checkpoint #3 (2 marks): Obtain a signature from your demonstrator.