Network programming with Intel® Edison

1 Secure Shell (SSH)

Secure Shell (SSH)[1] is a cryptographic network protocol for operating network services securely over an unsecured network. The best known example application is for remote login to computer systems by users. SSH provides a secure channel over an unsecured network in a client-server architecture, connecting an SSH client application with an SSH server. Common applications include remote command-line login and remote command execution, but any network service can be secured with SSH.

The standard TCP port **22** has been assigned for contacting a SSH server. Setting a device password on an Intel® Edison board will enable SSH.

1.1 Setting a password

1. Connect to the Edison via the **Console** port.
2. On Edison, run **configure_edison --password**.
3. Follow the onscreen prompts and set password as **pass1234**.
4. Turn off the module before unplugging.
5. Connect and login again to make sure the password has been set.

1.2 Edison's Ethernet over USB

The Edison provides a Virtual Ethernet connection via the OTG port. By default, the address of this device is **192.168.2.15/24**.

1. Connect to the Edison via the **OTG** port.
2. Look for a network adapter with the label **RNDIS**, assign an IP address with the same subnet as the Edison.
3. Ping **192.168.2.15** to make sure it is properly connected.

1.3 Creating a SSH session with TeraTerm

1. Launch TeraTerm.
2. Choose TCP/IP, service: SSH, TCP port: 22, Host: 192.168.2.15. Click OK.
3. Enter User name: root, Passphrase: pass1234. Choose *Use plain password to login.* Click OK.

[1] https://tools.ietf.org/html/rfc4251

You may create as many SSH sessions as you want. Besides TeraTerm, you should investigate on using **WinSCP** in order to transfer files easily via SSH.

2 Problem 1: Implement an UDP echo client/server

Echo server is an application which sends back an identical copy of the data it received. In this problem, there are 2 tasks to complete:

1. Implement an UDP echo server.
2. Implement an UDP echo client that continuously read input from STDIN, send to the server and receive response from the server.

The skeleton code of the server and the client is provided in **UDPEchoServer.c** and **UDPEchoClient.c**. The job of creating sockets has been done, please add your code to TODO sections to complete the code. For more information about socket programming in C, please visit here.

Submit: source code, a makefile, information about CPU & Memory usage. Your program should provide the same output as Figure 1.

```
root@TrucNDT-Edison:~/Network# ./UDPEchoClient 127.0.0.1

Input: IoTLab
UDPEchoClient: sent 6 bytes to 127.0.0.1
Receive: IoTLab

Input: Intel
UDPEchoClient: sent 5 bytes to 127.0.0.1
Receive: Intel

Input: Edison
UDPEchoClient: sent 6 bytes to 127.0.0.1
Receive: Edison

Input:
```

(a) Client

```
root@TrucNDT-Edison:~/Network# ./UDPEchoServer

UDPEchoServer: waiting to recvfrom...
UDPEchoServer: got packet from 127.0.0.1
UDPEchoServer: packet is 6 bytes long
UDPEchoServer: packet contains "IoTLab"
UDPEchoServer: send packet back to client

UDPEchoServer: waiting to recvfrom...
UDPEchoServer: got packet from 127.0.0.1
UDPEchoServer: packet is 5 bytes long
UDPEchoServer: packet contains "Intel"
UDPEchoServer: send packet back to client

UDPEchoServer: waiting to recvfrom...
UDPEchoServer: got packet from 127.0.0.1
UDPEchoServer: packet is 6 bytes long
UDPEchoServer: packet contains "Edison"
UDPEchoServer: send packet back to client

UDPEchoServer: waiting to recvfrom...
```

(b) Server

Figure 1: UDP echo client/server

Hints: You should try using Packet Sender, a powerful tool to send/receive network packets, at https://packetsender.com/.

3 Problem 2: Implement a TCP echo client/server

In this problem, there are 3 tasks to complete:

1. Implement a TCP echo server.
2. Implement a TCP echo client that continuously read input from STDIN, send to the server and receive response from the server.

3. Implement a TCP echo server which is able to handle multiple clients at the same time (using multithreading or I/O multiplexing technique).

Submit: source code, a makefile, information about CPU & Memory usage.