In the name of God

Second Programming Assignment[1]

COMPUTER NETWORKS

Instructor: Mehdi Kharrazi Spring 2011

In this assignment you will become more familiar with **TCP** and **implement** some parts of it.

1 Introduction

The Transmission Control Protocol (TCP) has been used widely in the Internet for reliable communications. This protocol enhances the host-to-host connection (which is provided by IP) to an **end-to-end** connection between two processes using the **port** concept. It also eliminates data corruption in the channel with high probability using ARQ, ACK, and Checksum mechanisms. Also the use of sequence numbers prevents reordering of packets and generally provides a reliable channel for transferring data for the upper layer. In addition using the congestion detection and avoidance mechanisms, the TCP provides the rate control possibility and thus make equivalent and efficient use of network resources possible for all nodes which are connected to the network.

The goal of this assignment is to become familiar with these mechanisms which are used in the TCP without getting involved with all of their complexities. For this purpose you should start with the code of previous assignments and prepare code of this assignment by changing/developing them.

1Thanks to Behnam Momeni, Ali Fattaholmanan, Farzaneh Moghaddam, Ashkan Nikravesh, Koosha Mirhosseini, Amir Sheikhha, and Hassan Eslami

2 Environment

The Basis of your work in this assignment is in the 4th layer. So like the previous assignment, the Partov[2] system will be used for executing your programs. For this purpose it's required to implement your programs using the Client Framework (CF) of Partov. For more information you could check the CF User Manual document.

3 Topology

The initial topology which will be given to each student is like the illustration 1. Of course your program should not be dependent to any special topology and should not assume any special assumption about the topology (assumptions based on IP ranges like being a private IP address are not acceptable).

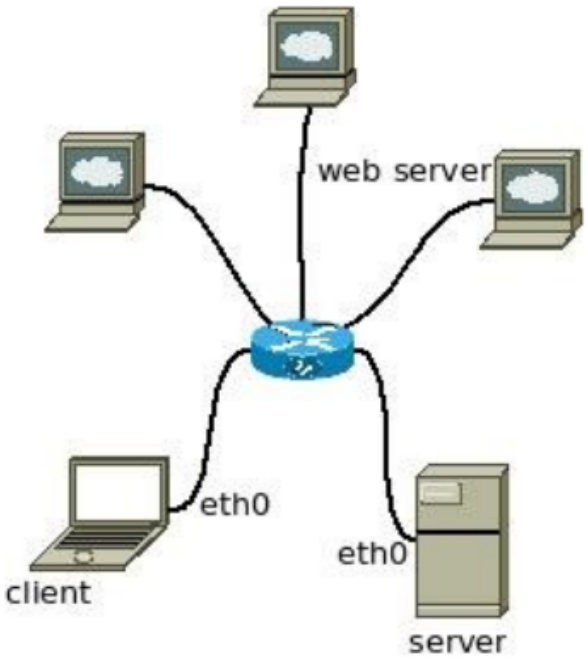

Illustration 1: A typical topology

In this topology you implement the **client** and **server** nodes from the illustration 1.

2 Portable And Reliable Tool fOr Virtualization (Partov)

4 Expectations

You should implement two programs in Partov environment. First program is for the **server** just like your program in PA1. Second program is for the **client** just like your program in PA0 with one difference: Instead of socket programming you should use the Partov to implement your own TCP (like the **server** in PA1). You could change the Makefile as required so both programs could be compiled correctly. Your program must be compiled using **make** command and two executable files (client.out and server.out) should be created.

5 Programs

5.1 Client's program

This program is like the client program in previous assignments. It has an IP and a port in the beginning. The IP address and port number of the server are present in the custom information field of the client as follows (in separated lines). In the next line of custom information, the MAC address of the gateway is placed. Client should send all of its packets through this gateway. Then the windows size which must be used in the TCP packets (the buffer size of client) and the ISN (the initial value of the sequence number in the TCP header) are followed.

Server-IP-Address

Server-Port-Number

MAC-of-the-Gateway

Initial-TCP-Window-Size

Initial-TCP-Sequence-Number

Also the client program has two arguments for **host name** and **file name** which must be downloaded by the client. For example the Icient program may be executed as:

./client.sh www.google.com */intl/en_com/images/srpr/logo1w.png*

The client program could get these two arguments before the initialize method get called, by implementing following method:

```
static void SimulatedMachine::parseArguments(int argc, char *argv[]);
```

The client program must read server IP and port from custom information after execution. Then it should ask about the requested file (which was given in command

line arguments) by sending UDP packets to that IP/Port including file name as UDP packet payload. In response it will read the IP/Port of the server which hosts requested file.

An example of the client custom information is as follows:

192.168.23.17

1234

00:17:20:00:10:A3

128

1128715

Like PA1 there is no limitation about the name which you will use to save the downloaded file name. It's enough for the used file name to not start with special characters like – and finish with correct extension (like .html, …).

After sending UDP request for the file name and getting IP/Port of the web server, like PA0, it must establish a TCP connection with the found IP/Port and then must use the HTTP protocol (above the established TCP connection) to download the requested file. You should not use socket programming in this assignment and you should use your own implemented UDP and TCP connections (like server program in PA1) by direct interaction with protocol headers.

For the TCP connection you are not required to implement all aspects. it's enough to implement the following mechanisms:

1. Three-way TCP handshake,
2. ACK and Sequence Number,
3. Meaningful default values for headers' fields and flags,
4. TCP checksum, Reliability, and retransmission mechanisms,
5. Fast retransmission upon receiving a duplicate ACK,
6. Flow control and window size (not the congestion window),
7. Four-way TCP connection tear down.

5.2 Server's program

The server program (not the web-server) is like the server program in PA1 with this difference that you are not required to implement the NAT anymore. The server program has folowing information in its custom information field:

Port-for-UDP-Requests

MAC-of-the-Gateway

Count-of-Files

File-Address

IP

Port

…..

The server should wait for UDP requests on the given port in the first line of custom information field. After receiving each request, it should search for the requested file name/address in its list of files and then should return the IP and Port of the related web server. An example of custom information for server is as follows:

1234

00:17:20:00:10:A3

2

/favicon.ico

213.233.171.14

8080

/old/index.html

213.233.171.15

2280

For more information about the required protocols, you could check following links:

- http://en.wikipedia.org/wiki/Transmission_Control_Protocol
- http://en.wikipedia.org/wiki/User_Datagram_Protocol

6 Important Notes

First your implementation must not rely on value of IP addresses in anyway. For example you could not tell that if Packet's IP address starts with 213.233 then do that work.

Second change the Makefile as required so two executable files, namely server.out and client.out, were created by *make* command.

Third you could obtain the IP address and netmask of each interface using getMask and getIP public methods of the Interface class.

Forth your program does not require to handle ARP packets. Those packets which are sent between client or server and the gateway nodes, are captured, processed, and handled by the Partov system. So your program does not receive them.

Fifth in case of any ambiguities, feel free to ask your questions in course mailing list.

7 Delivery Method

First test your programs completely on your own computer and when you were sure about its functionality, upload your programs as one file in zip format to the online judge website. This zip file must include the *user* folder and the *Makefile*. "*make clean*" your project before creating the zip file so it does not contain binary files. Also the zip file should not contain additional folders. Just zip the *user* folder and *Makefile*, not the folder containing them.