Lab3

Last modified: 26/10/2022 10:20

Use UDP to implement the problems from Lab1 and Lab2 and the following problems. Think of a way to write concurrent UDP servers. Use examples bellow as guidance.

Example –UDP

Example – UDP Broadcast

Example SELECT– Multiplexed TCP Client-Server Chat application

Note: Run the server and the client on different hosts. For this you need to know the IP address of the machine where the server runs (ifconfig/ipconfig)

Note: Learn to use Wireshark in order to inspect network traffic.

1. The client sends periodical PING datagrams with a random content to a <server> and <port> specified in command line. The server returns back (echoes) the same packets (content). The client checks the content of the received packets to match what was sent and computes the round trip time and displays it to the user – for each sent packet.
2. Implement the Python concurrent example from lab2 to work with UDP. You no longer need threads to be able to serve multiple clients – so the implementation should be much shorter.
3. Implement the Chat server example (see the link bellow) using UDP and TCP –only this time each client should contact the server just for registration. All communication happens directly between the peers (clients) without passing trough the server. Each client maintains an endpoint (TCP/UDP) with the server just for learning the arrival/departure of other clients. You create a mesh architecture where all clients connect directly between each others.
4. Write an UDP broadcast application that serves as client and server at the same time. The application is started with the network broadcast address (<NBCAST>) as argument in the command line.
    1. Upon launching the application listens on UDP port 7777.
    2. Every 3 seconds the application sends a UDP broadcast message to NBCAST port 7777 with the format: TIMEQUERY\0 (string)
    3. Whenever the application receives a TIMEQUERY demand it answers to the source IP:port with a string message: TIME HH:MM:SS\0 (current time)

using unicast.

4. Every 10 seconds the application sends a UDP broadcast message to NBCAST port 7777 with the format: DATEQUERY\0 (string)
5. Whenever the application receives a DATEQUERY demand it answers to the source IP:port with a string message: DATE DD:MM:YYYY\0 (current date) using unicast.
6. The application will keep a list of peers (that answer to broadcast – IP:portno) and update the information anytime a unicast message is received upon a broadcast.
7. When an entry in a list does not have any answer for 3 consecutive broadcasts it will be removed from the list.
8. The list will be displayed (ip,date, time) on the screen upon each update (using a screen positioning api like conio or by erasing the screen before each update).
9. Every malformed request/response received will be counted and displayed at the end of a screen update. You will have a list of malformed messages displayed with their source IP address. The list should be limited in size and implemented as a queue. Recent messages are added to the head and old messages are moving towards the tail.

Note: Suggestion: Implement the application on Windows, or run it on your laptop in order to be able (all of you simultaneously) to listen on port 7777. Your application should strictly follow the protocol and be able to interact with all applications written by your colleagues.

Note: On Windows in order to have timer like events (periodical events handled) use **timeSetEvent** or a similar function (the newer **CreateTimerQueueTimer**) and set a different callback function for each type of event.

Note: Sending broadcast UDP requires a *setsockopt(sock,SOL_SOCKET,SO_BROADCAST,&broadcast,sizeof(broadcast)* as in example.

Receiving broadcast usually doesn't require any additional effort compared to a normal UDP application. If not able to receive broadcast on windows try to *setsockopt* on the receiving socket as well.

Note (Malformed traffic): To generate malformed traffic one could use the **nc** (network cat) command on a linux like system as it follows:

nc -4 –u <dstip> <dstport>

and type in anything until CTRD+D is pressed ! Anything typed in will be sent to the the remote IP and port using the specified protocol (u=udp)

or

<command> | nc -4 –u 172.30.5.16 7777

Example of displayed results:

172.30.0.4:7777 TIME 16:51:43 DATE 08:04:2014
172.30.118.185:7777 TIME 16:51:47 DATE 08:04:2014
172.30.116.220:7777 TIME 16:51:44 DATE 08:04:2014
172.30.112.50:7777 TIME 16:51:42 DATE 08:04:2014

Malformed Data:
172.30.118.185:7777 - HELLO
172.30.118.185:7777 - HELLO
172.30.118.185:7777 - HELLO
172.30.118.185:7777 - HELLO
172.30.118.185:7777 - HELLO
172.30.116.220:7777 - asdasdDATE 08:04:201
172.30.116.220:7777 - asdasdTIME 16:30:50
172.30.116.220:7777 - asdasdTIME 16:30:52
172.30.116.220:7777 - asdasdTIME 16:30:56
172.30.116.220:7777 - asdasdTIME 16:30:59