Lab 1 - Networks and protocols

TI602

In this lab we will reproduce our tutorial architecture evolution in order to visualize and understand what happens at different levels. We use *Packet Tracer*, a simulation tool from Cisco that allows creating architectures, configuring devices and visualize message exchanges.

You need first to download and install *Packet Tracer* from the Cisco Netacad website using your account or from any other source.

the work is to be done individually, a report, containing screenshots of your work as well as your analysis/discussion for the different questions, is requested at the end of the work and must be uploaded a on Moodle before April 9th.

Case study

We are interested in an internal online shooter video game platform that runs on a local network infrastructure. In this network, each player accesses the game on a machine on which their client application is running. A central server manages the synchronization of the game's views and actions: a client application sends data to the servers at a fairly high frequency about the position of the character and the actions executed by the latter (in particular on the other players characters). The server after receiving this information, made a computation based the actions and movements done (received) and distributes the updates to other machines so that everyone can see everyone's status almost in real time locally on their machine. In this kind of games, the speed of the messages as well as the optimization of resources is very important in order to guarantee an optimal gaming experience (maximum data transmitted correctly and low latency for a view close to real time)

Simulation using Packet Tracer

1. Open Packet Tracer and create a first simple network corresponding to the first topology seen in tutorial 1 (Figure 1). To do, use the panel in the bottom left (drag and drop) to instantiate end computers (end device category), a Hub (network device category), and copper strait-through cables (connection category).

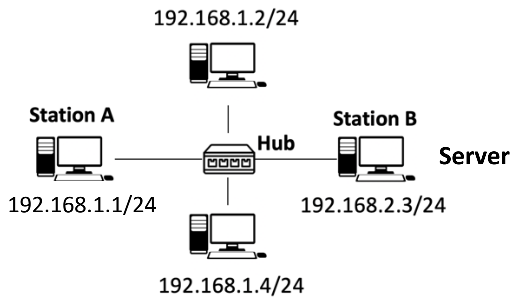

Figure 1

2. Configure the different computers with the given IP addresses and mask on the corresponding network cards: click on the computer → config → Interface → FastEthernet.
3. The simulation tool helps visualizing the transmission of messages, the simple PDU tool (in the top left panel) test the connectivity between a source and a destination by sending a back-and-forth layer 3 message (ICMP/ping).
    a. Test the transmission of a simple PDU from station A to station B (by clicking first on the station A to define it as source and then click on B to define it as a destination).
    b. Follow the process using the simulation panel in the bottom right, what happens?
    c. Locate to problem by clicking on the different *Events* and by analyzing the operations in the different layers. Correct it and test again.
    d. You can delete simulations using the hidden drawer in the bottom left
4. Simulate a game update messages using complex PDU simulation from all clients to the server and from the server to all the clients: set the application to ping (back-and-forth message) and the frequency to 1 second (you can control the execution speed using the slider). What happens?
5. Replace the Hub with a Switch of type "2960". Replay the same simulation. Is there any difference?
6. Extend the network as shown in the figure 2.

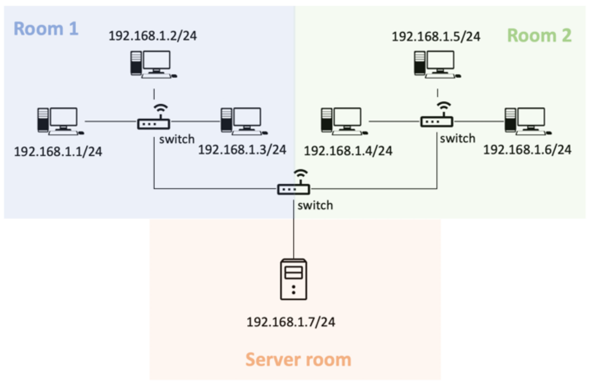

Figure 2

7. Test the connectivity and analyze the messages transfer.
8. Send a broadcast PDU using the address 255.255.255.255 from any client. What happens?
9. Divide a network into different subnetworks (broadcast domains) by attributing addresses to the computers using the address plan bellow (keep the same topology). Test the connectivity.

Subnetwork Address	**Usable Host Range**	**Broadcast Address**
192.168.1.0	192.168.1.1 - 192.168.1.62	192.168.1.63
192.168.1.64	192.168.1.65 - 192.168.1.126	192.168.1.127
192.168.1.128	192.168.1.129 - 192.168.1.190	192.168.1.191

10. Replace the switch with a router and assign addresses to its different interfaces, do not forget to turn on the interface (top left in the interface setting panel: the ”on” check box).
11. Create the necessary default gateway entries on the computers (config → settings) using the router address.
12. Test the connectivity between clients from different networks and the server and analyze the messages transfer.
13. Is it possible to send a broadcast message from the server to clients? Test it, what happens?
14. Add two other subnetworks as shown in Figure 3. Propose an address plan that matches this need, configure Router 1 and router 2 routing tables (config → settings → routing → static) and the new computers' IP addresses/mask/gateway.

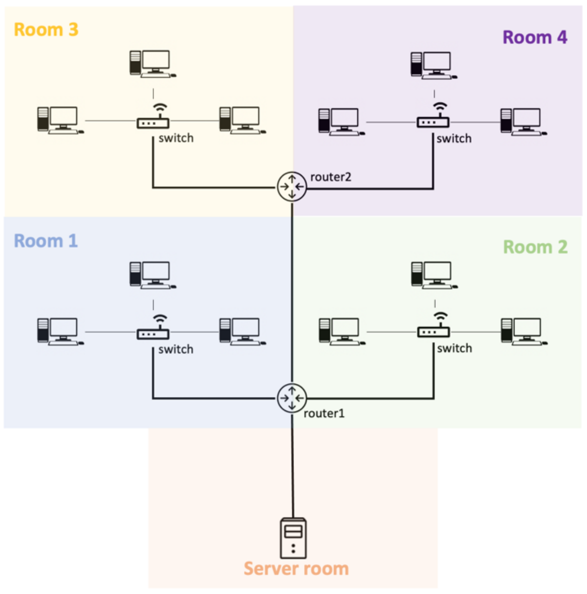

Figure 3

15. Test the final configuration by simulating game update messages with UDP application and port numbers :
    - On the clients (only one client per network)
        - create a program that sends every second a UDP packet to the server: programming → new → template → Python → UDP Socket → open main.py
        - Modify the source port number to 1111, the destination port to 2222 (server UDP port) and the destination address to the server IP address (see the example bellow)
    - On the server
        - create a program that sends every second a UDP packet to the clients: programming → new → template → Python → UDP Socket → open main.py
        - Modify the source port number to 2222
        - Create one socket.send line per client as shown in the example bellow, modify the destination port to 1111 (client UDP port) and the destination address to the corresponding client IP address.
    - Run the different programs, launch and examine the simulation (the different transfer steps and the port multiplexing/demultiplexing process).

Client UDP socket program

```
from udp import *
from time import *

def onUDPReceive(ip,port, data):
        print("received from "
                + ip + ":" + str(port) + ":" + data);

def main():
        socket = UDPSocket()
        socket.onReceive(onUDPReceive)
        print(socket.begin(1111))
                        Source port
        count = 0
        while True:                 Destination
                                     Address :
                                  server address
                count += 1
                socket.send("192.168.1.X", 2222, "hello " + str(count))
                sleep(1)                   Destination
                                           Port :server

if __name__ == "__main__":
        main()
```

Server UDP socket program

```
from udp import *
from time import *

def onUDPReceive(ip,port, data):
        print("received from "
                + ip + ":" + str(port) + ":" + data);

def main():
        socket = UDPSocket()
        socket.onReceive(onUDPReceive)
        print(socket.begin(2222))
                        Source port
        count = 0
        while True:                 Destination
                                     Address :        Destination
                count += 1        client address      Port : client
        # Client 1
                socket.send("192.168.1......", 1111, "hello " + str(count))
        # Client 2
                socket.send("192.168.1......", 1111, "hello " + str(count))
        #same thing for the other clients
                #....
                sleep(1)

if __name__ == "__main__":
        main()
```