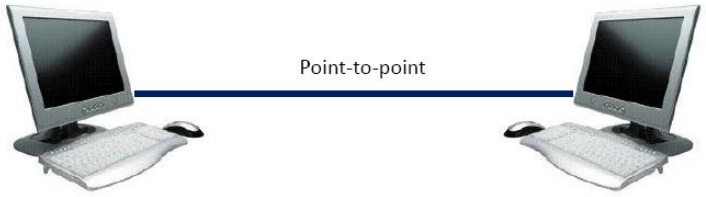
CCN Lab Experiments

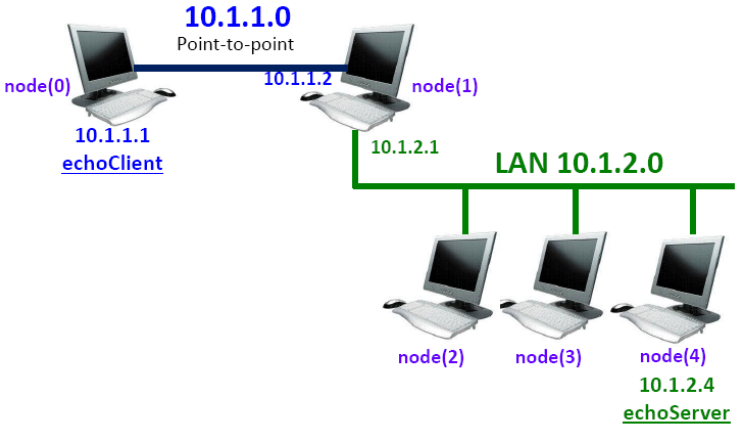
Cycle-1

1. Design a Point-to-Point Network (Figure-1)for Client and Server Echo Application Using Python and capture ECHO packets transmitted from N1 to N2 and Analyze it using Wireshark

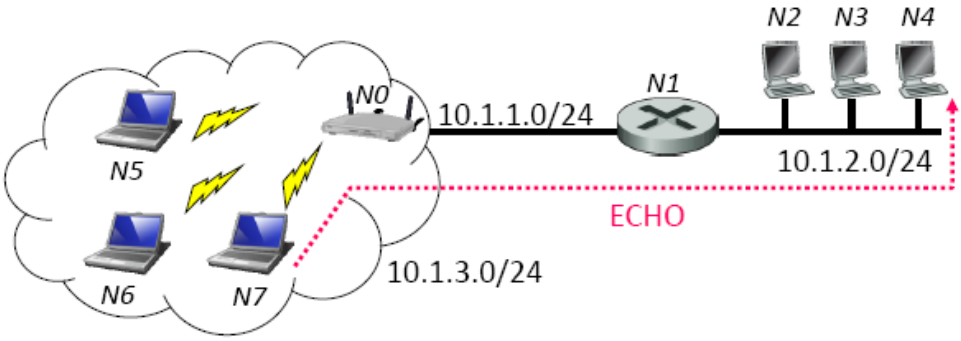


   a. Create a simple topology of two nodes (Node1, Node2) separated by a point-to-point link.
   b. Setup a UdpClient on one Node1 and a UdpServer on Node2. Let it be of a fixed data rate Rate1.
   c. Start the client application, and measure end to end throughput whilst varying the latency of the link.
   d. Now add another client application to Node1 and a server instance to Node2. What do you need to configure to ensure that there is no conflict?
   e. Repeat step 3 with the extra client and server application instances. Show screenshots of pcap traces, which indicate that delivery, is made to the appropriate server instance.

2. Build an Client and Server Echo Application by designing Bus Network Topology (Figure-2) Using Python and capture ECHO packets transmitted from Node-1 to Node-4 and Analyze it using Wireshark



3. Model the network topology below using python and capture ECHO packets transmitted from N7 to N4 and Analyze using Wireshark



4. Create a simple dumbbell topology, two client Node1 and Node2 on the left side of the dumbbell and server nodes Node3 and Node4 on the right side of the dumbbell. Let Node5 and Node6 form the bridge of the dumbbell. Use point-to-point links.
   a. Install a TCP socket instance on Node1 that will connect to Node3.
   b. Install a UDP socket instance on Node2 that will connect to Node4.
   c. Start the TCP application at time 1s.
   d. Start the UDP application at time 20s at rate Rate1 such that it clogs half the dumbbell bridge's link capacity.
   e. Increase the UDP application's rate at time 30s to rate Rate2 such that it clogs the whole of the dumbbell bridge's capacity.
   f. Use the ns-3 tracing mechanism to record changes in congestion window size of the TCP instance over time. Use gnuplot/matplotlib to visualise plots of cwnd vs time.
   g. Mark points of fast recovery and slow start in the graphs.
   h. Perform the above experiment for TCP variants Tahoe, Reno and New Reno, all of which are available with ns-3.
5. Create a simple dumbbell topology, two client Node1 and Node2 on the left side of the dumbbell and server nodes Node3 and Node4 on the right side of the dumbbell. Let Node5 and Node6 form the bridge of the dumbbell. Use point to point links.
   a. Add drop tail queues of size QueueSize5 and QueueSize6 to Node5 and Node6, respectively.
   b. Install a TCP socket instance on Node1 that will connect to Node3.
   c. Install a TCP socket instance on Node2 that will connect to Node3.

d. Install a TCP socket instance on Node2 that will connect to Node4.
e. Start Node1--Node3 flow at time 1s, then measure it's throughput. How long does it take to fill link's entire capacity?
f. Start Node2--Node3 and Node2--Node4 flows at time 15s, measure their throughput.
g. Measure packet loss and cwnd size, and plot graphs throughput/time, cwnd/time and packet loss/time for each of the flows.
h. Plot graph throughput/cwnd and packet loss/cwnd for the first flow. Is there an optimal value for cwnd?
i. Vary QueueSize5 and QueueSize6. Which one has immediate effect on cwnd size of the first flow? Explain why.

Cycle-2

1. Write a Python program to ask the operating system (Linux, Mac OS, and Windows) for resolving the hostname (www.google.com.org.) into IP address using the particular network service, called the Domain Name System. i.e (Hostname into an IP Address);
2. Write a Python Program to communicate to google search simple socket. i.e. sending a raw text message across the Internet and receiving a bundle of text in return.
3. Write a Python Program to implement the UDP Server and Client that make use of the localhost IP address.
4. Write a Python Program to implement the UDP Server and Client on Different Machines (Note : Instead of always answering to client requests, The server should randomly chooses to answer only half of the requests coming in from clients, which will let you see how to build reliability into your client code without waiting what might be hours for a real dropped packet to occur on your network).
5. Write a Python Program to Send a Large UDP Packet from Server to Client using Fragmentation
6. Write a Python Program to implement Simple TCP Server and Client
7. Write a Python Program to implement Client Server example using Pipes, FIFOs, Message Queues, Shared Memory
8. Write a Python Program to implement to Concurrent Server, Multiprotocol Server, Internet Super Server, Chat Server and Mail Server.