Communication Networks 2

SS 2021

Assignment 4

Group 06

Name	Mat.Nummer
Juan Aramis Oposich	11701238
Paul Kloker	12034928

June 22, 2021

1 Task description and setup

The task of the last assignment is to model a network in NS-3, which is an open source network simulator, according to the following network diagram (see figure 1). Furthermore, a 10 seconds long simulation with a ping measurement from node 1 to node 3 shall be done and the generated measurement data be discussed.

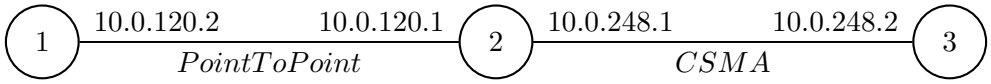

Figure 1: Network topology

For this the latest version of NS-3 was downloaded form `https://www.nsmam.org` and build with the command `$ ./build.py`.

2 NS-3 Model

NS-3 models are C++ based and consist out of network nodes, network devices, communication channels and applications. Most of the code is based on the second tutorial in the example folder. Listing 1 shows the creation of the topology. Because our system consists of one PointToPoint connection and one Carrier Sense Multiple Access (CSMA) network, different kind of nodes were created and the given channel attributes were set by using the helper objects. To finish the topology creation helper objects were used to install the network devices and a IP stack for every node.

```
//P2P
NodeContainer p2pNodes;
p2pNodes.Create (2);

PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate",
                                 StringValue ("11Mbps"));
pointToPoint.SetChannelAttribute ("Delay",
                                  StringValue ("80ms"));

NetDeviceContainer p2pDevices;
p2pDevices = pointToPoint.Install (p2pNodes);

//CSMA
NodeContainer csmaNodes;
```

```
csmaNodes.Add (p2pNodes.Get (0));
csmaNodes.Create (1);

CsmaHelper csma;
csma.SetChannelAttribute ("DataRate",
                          StringValue ("70Mbps"));
csma.SetChannelAttribute ("Delay",
                          TimeValue (MicroSeconds (1000)));

NetDeviceContainer csmaDevices;
csmaDevices = csma.Install (csmaNodes);

InternetStackHelper stack;the
stack.Install (p2pNodes.Get (1));
stack.Install (csmaNodes);
```

Listing 1: Topology creation

In table 1 all nodes are listed with their IP address and the position in the NS-3 node container. The order in which the nodes are created plays a role in the address assignment.

Node No.	**NS-3 Node Container Element**	**IP Address**
1	p2pNodes.Get(1)	10.0.120.2
2	p2pNodes.Get(0)	10.0.120.1
2	csmaNodes.Get(0)	10.0.248.1
3	csmaNodes.Get(1)	10.0.248.2

Table 1: Nodes and the according NS-3 node container element

To create a routing table with the `Ipv4GlobalRoutingHelper`, both subnets first got the right address space assigned to them (see listing 2).

```
Ipv4AddressHelper address;
address.SetBase ("10.0.120.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign (p2pDevices);

address.SetBase ("10.0.248.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces;
csmaInterfaces = address.Assign (csmaDevices);

Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
```

Listing 2: IP address assignment and routing table creation

Listing 3 shows how the ping to the third node was created and installed on the first node. The attribute `Verbose` was set to true to create a ping-style output. The payload was set to 1024 bytes and the interval between the ping messages to 0.2 Seconds. The last two lines of listing 3 define how long the ping application will run during the simulation.

```
V4PingHelper ping = V4PingHelper (
csmaInterfaces.GetAddress (1));
ping.SetAttribute ("Verbose", BooleanValue (true));
ping.SetAttribute ("Size", UintegerValue(1024));
ping.SetAttribute ("Interval", TimeValue(Seconds(0.2)));
ApplicationContainer apps = ping.Install (p2pNodes.Get (1));
apps.Start (Seconds (1.0));
apps.Stop (Seconds (10.0));
```

Listing 3: Ping set-up

Before the simulation was invoked the stop time was specified and packet capturing was enabled for all P2P and CSMA nodes (see listing 4).

```
csma.EnablePcapAll("assignment4", true);
pointToPoint.EnablePcapAll ("assignment4", true);

Simulator::Stop (Seconds (10.0));
Simulator::Run ();
Simulator::Destroy ();
```

Listing 4: pcap and simulation start

The code was then compiled and execute with `waf`. This created the ping measurement and pcap data which will be discussed in the next section.

3 Data analysis

The simulation delivered 45 ping measurements of which 44 had a Round Trip Time (RTT) of exact 163.779 ms. Only the first message deviated from this value with a RRT of 178.881 ms. This is because of the two initial ARP request and response messages which can be seen in the pcap files of node two and three. The Address Resolution Protocol (ARP) is used in IPv4 networks to obtain the MAC address of a node. This is done by broadcasting a ARP request containing the IP address of the node. If a node has this address, it answers with an ARP response containing its MAC address. In the beginning of the Simulation both nodes in the csma network need to obtain the MAC address of the other node to forward the ping message. This leads to a higher RTT in the first ping measurement because after the initial setup the MAC address is saved in the ARP table. In the P2P network ARP is not used because there is just one single communication partner and no need to obtain the MAC address.

After the initial set-up the RTT can be divided into the propagation delay (t_p) of the link which is the time a packet needs to travel between nodes and the transmission delay (t_t) which is the time it takes to transmit a packet from the host to the transmission medium. It can be calculated with the number of bits to be sent (n_{bit}) and the bandwidth of the link (b): $t_t = n_{bit}/b$

$$t_p = 2 \cdot (80\text{ms} + 1\text{ms}) = 162\text{ms}$$

$$t_t = 2 \cdot \left(\frac{1054 \cdot 8}{11\,\text{Mbit}\,\text{s}^{-1}} + \frac{1054 \cdot 8}{70\,\text{Mbit}\,\text{s}^{-1}}\right) = 1.774\text{ms}$$

$$RTT = t_p + t_t = 163.774\text{ms}$$

There are other delay types as well, like processing delay and queuing delay, but they are not considered in the simulation. In reality the RTT is a bit different each time because of the changing network load, and there are sometimes lost packets, but this behavior was not considered in this model.

While analyzing the pcap files with WireShark, it was noticed that the checksums in the IP and ICMP header were not calculated and were always zero. This is due to the fact that NS-3 disables the calculation of the checksum by default, as this would slow down the simulation. Normally the calculation is done with hardware support which speeds up the calculation in reality. There is no need for checksums anyhow because no error behavior like flipping bits was modeled.

4 Conclusion

NS-3 is a powerful tool to easily model different network systems. This is useful to test different set-ups or new protocols, but it has limits. Only the modeled behavior is simulated not more. Therefore, the designer should think about the system constraints and specifications and verify if the model matches the goal. So simulation does not replace the use of the brain.