NS3 Project Update 2

Sihat Afnan

Student ID: 1705098

Paper OverView

$$\delta = \frac{RTT_{sample} - RTT_{previous}}{RTT_{previous}} \quad \text{(Step 1)}$$

$$D = 1 - \frac{1}{F * S} \quad \text{(Step 2)}$$

$$B \leftarrow \max(\delta, D * B) \quad \text{(Step 3)}$$

$$RTT_{\max} = \max(RTT_{sample}, RTT_{previous}) \quad \text{(Step 4)}$$

$$RTO \leftarrow \max(D * RTO, (1 + B) * RTT_{\max}) \quad \text{(Step 5)}$$

$$RTO \leftarrow \max(RTO, RTO_{\min}) \quad \text{(Step 6)}$$

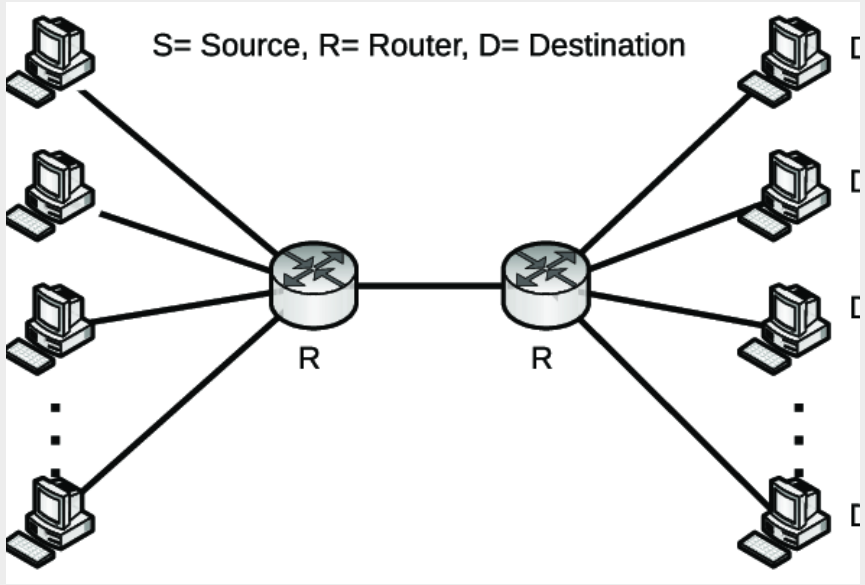
DumbBell Topology

```
NodeContainer leftGate;
leftGate.Create (1);
NS_LOG_INFO ("Number of Nodes After left Gate: " << NodeList::GetNNodes());

NodeContainer rightGate;
rightGate.Create (1);
NS_LOG_INFO ("Number of Nodes After Right Gate: " << NodeList::GetNNodes());

NodeContainer sources;
sources.Create (num_flows);
NS_LOG_INFO ("Totoal Number of Nodes After Sources: " << NodeList::GetNNodes());

NodeContainer sinks;
sinks.Create (num_flows);
NS_LOG_INFO ("Totoal Number of Nodes After Sinks: " << NodeList::GetNNodes());
```



Among Routers

```
PointToPointHelper BottleNeckLink;
BottleNeckLink.SetDeviceAttribute ("DataRate", StringValue (shared_bandwidth));
BottleNeckLink.SetChannelAttribute ("Delay", StringValue (shared_delay));
BottleNeckLink.SetDeviceAttribute ("ReceiveErrorModel", PointerValue (&error_model));
```

Among Routers and Src/Sinks

```
// and the channels between the sources/sinks and the gateways
PointToPointHelper LocalLinkL;
PointToPointHelper LocalLinkR;
for (int i = 0; i < num_flows; i++)
    {
      if ((i%2) == 0) {
        LocalLinkL.SetDeviceAttribute ("DataRate", StringValue (access_bandwidth));
        LocalLinkL.SetChannelAttribute ("Delay", StringValue (access_delay));
        LocalLinkR.SetDeviceAttribute ("DataRate", StringValue (access_bandwidth));
        LocalLinkR.SetChannelAttribute ("Delay", StringValue (access_delay));

        NetDeviceContainer devices;
        devices = LocalLinkL.Install (sources.Get (i), leftGate.Get (0));

        address.NewNetwork ();
        Ipv4InterfaceContainer interfaces = address.Assign (devices);

        devices = LocalLinkR.Install (rightGate.Get (0), sinks.Get (i));
        address.NewNetwork ();
        interfaces = address.Assign (devices);
        sink_interfaces.Add (interfaces.Get (1));
```

Connect Source to destination

```
for (uint16_t i = 0; i < sources.GetN (); i++)
  {
    AddressValue remoteAddress (InetSocketAddress (sink_interfaces.GetAddress (i, 0), port));
    Config::SetDefault ("ns3::TcpSocket::SegmentSize", UintegerValue (tcp_adu_size));
    BulkSendHelper ftp ("ns3::TcpSocketFactory", Address ());
    ftp.SetAttribute ("Remote", remoteAddress);
    ftp.SetAttribute ("SendSize", UintegerValue (tcp_adu_size));
    ftp.SetAttribute ("MaxBytes", UintegerValue (data_mbytes * 1000000));

    ApplicationContainer sourceApp = ftp.Install (sources.Get (i));
    sourceApp.Start (Seconds (start_time * i));
    sourceApp.Stop (Seconds (stop_time - 3));

    sinkHelper.SetAttribute ("Protocol", TypeIdValue (TcpSocketFactory::GetTypeId ()));
    ApplicationContainer sinkApp = sinkHelper.Install (sinks.Get (i));
    sinkApp.Start (Seconds (start_time * i));
    sinkApp.Stop (Seconds (stop_time));
  }
```

uint32_t tcp_adu_size

Tracing

```
static void
RttTracer (Time oldval, Time newval)
{
  if (firstRtt)
    {
      *rttStream->GetStream () << "0.0 " << oldval.GetSeconds () << std::endl;
      firstRtt = false;
    }
  *rttStream->GetStream () << Simulator::Now ().GetSeconds () << " " << newval.GetSeconds () << std::endl;
}

static void
RtoTracer (Time oldval, Time newval)
{
  if (firstRto)
    {
      *rtoStream->GetStream () << "0.0 " << oldval.GetSeconds () << std::endl;
      firstRto = false;
    }
  *rtoStream->GetStream () << Simulator::Now ().GetSeconds () << " " << newval.GetSeconds () << std::endl;
}
```

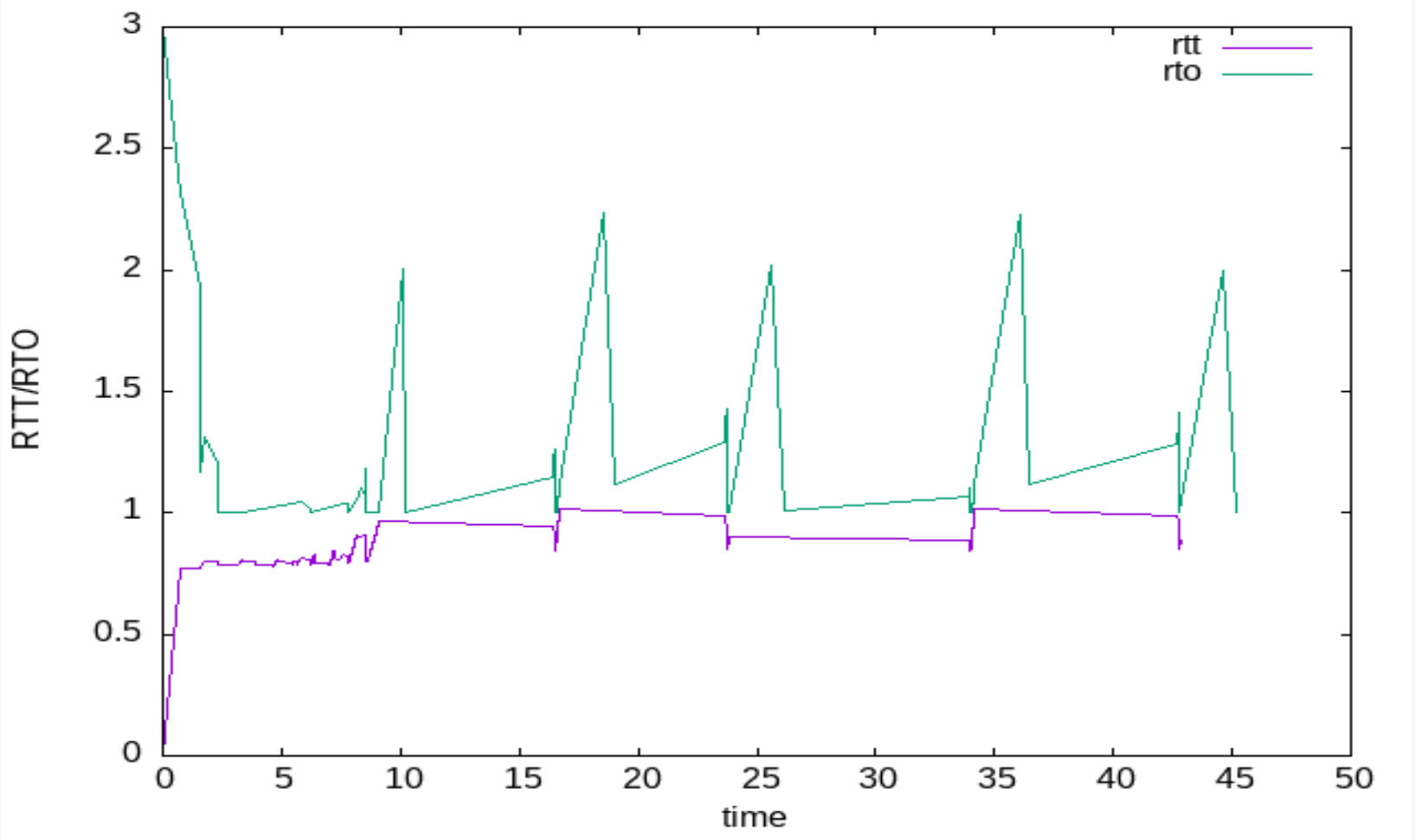
Tracing

```
static void
TraceRtt (std::string rtt_tr_file_name)
{
  AsciiTraceHelper ascii;
  rttStream = ascii.CreateFileStream (rtt_tr_file_name.c_str ());
  Config::ConnectWithoutContext ("/NodeList/2/$ns3::TcpL4Protocol/SocketList/0/RTT", MakeCallback (&RttTra
}

static void
TraceRto (std::string rto_tr_file_name)
{
  AsciiTraceHelper ascii;
  rtoStream = ascii.CreateFileStream (rto_tr_file_name.c_str ());
  Config::ConnectWithoutContext ("/NodeList/2/$ns3::TcpL4Protocol/SocketList/0/RTO", MakeCallback (&RtoTra
}
```

```
  Simulator::Schedule (Seconds (0.1), &TraceRtt, prefix_file_name + "rtt.data");
  Simulator::Schedule (Seconds (0.1), &TraceRto, prefix_file_name + "rto.data");
```

RTT/RTO Graph



Performance metrics : FlowMonitor

```
FlowMonitorHelper flomon;
Ptr<FlowMonitor>monitor;
if(flow_monitor){
  monitor = flomon.InstallAll();
}
Simulator::Stop (Seconds (stop_time));
Simulator::Run ();

if (flow_monitor)
  {
    FlowMonitor::FlowStatsContainer stats = monitor->GetFlowStats();
    for(auto iter = stats.begin(); iter != stats.end(); iter++) {
      NS_LOG_UNCOND("----Flow ID:" <<iter->first);
      NS_LOG_UNCOND("Sent Packets = " << iter->second.txPackets);
      NS_LOG_UNCOND("Received Packets = " << iter->second.rxPackets);
      NS_LOG_UNCOND("Lost Packets = " << iter->second.txPackets - iter->second.rxPackets);
      NS_LOG_UNCOND("Packet loss ratio = " << (iter->second.txPackets-iter->second.rxPackets)
                                              *100.0/iter->second.txPackets << "%");
      NS_LOG_UNCOND("Throughput = " << iter->second.rxBytes * 8.0/(iter->second.timeLastRxPacket
                                       .GetSeconds()-iter->second.timeFirstTxPacket
                                       .GetSeconds()) << " bits/s");
    }
  }

Simulator::Destroy ();
```

Output

```
----Flow ID:1
Sent Packets = 8909
Received Packets = 8852
Lost Packets = 57
Packet loss ratio = 0.639802%
Throughput = 561609 bits/s
----Flow ID:2
Sent Packets = 23900
Received Packets = 23744
Lost Packets = 156
Packet loss ratio = 0.65272%
Throughput = 1.48914e+06 bits/s
----Flow ID:3
Sent Packets = 6589
Received Packets = 6004
Lost Packets = 585
Packet loss ratio = 8.87843%
Throughput = 380627 bits/s
```


Thank You