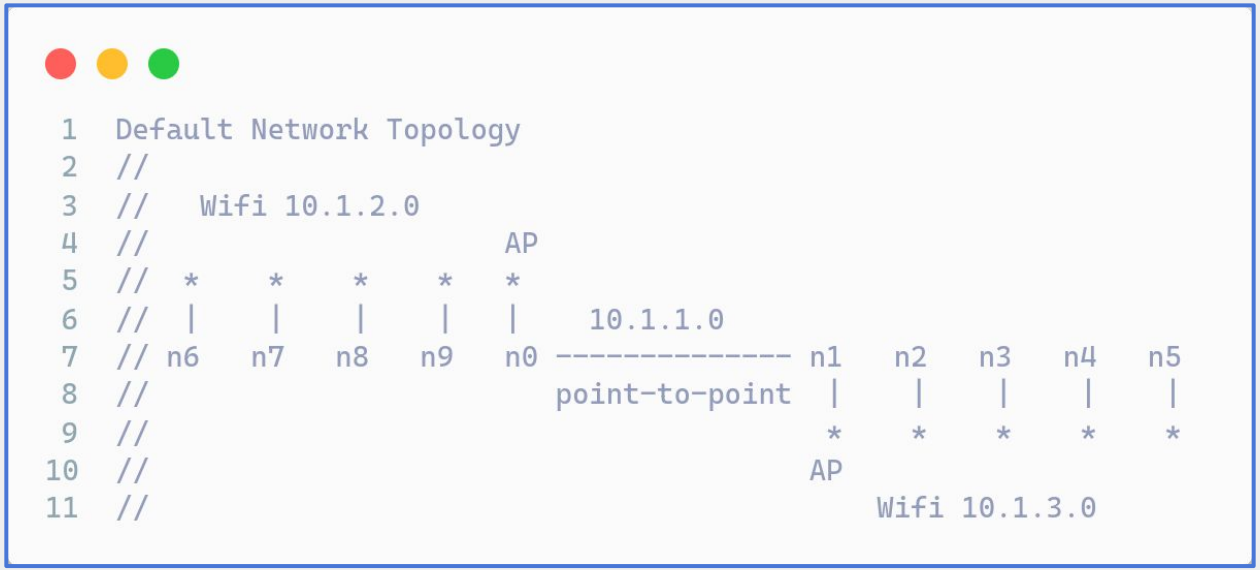
NS3 Baseline Update

1705036 - Ayan Antik Khan

Topology

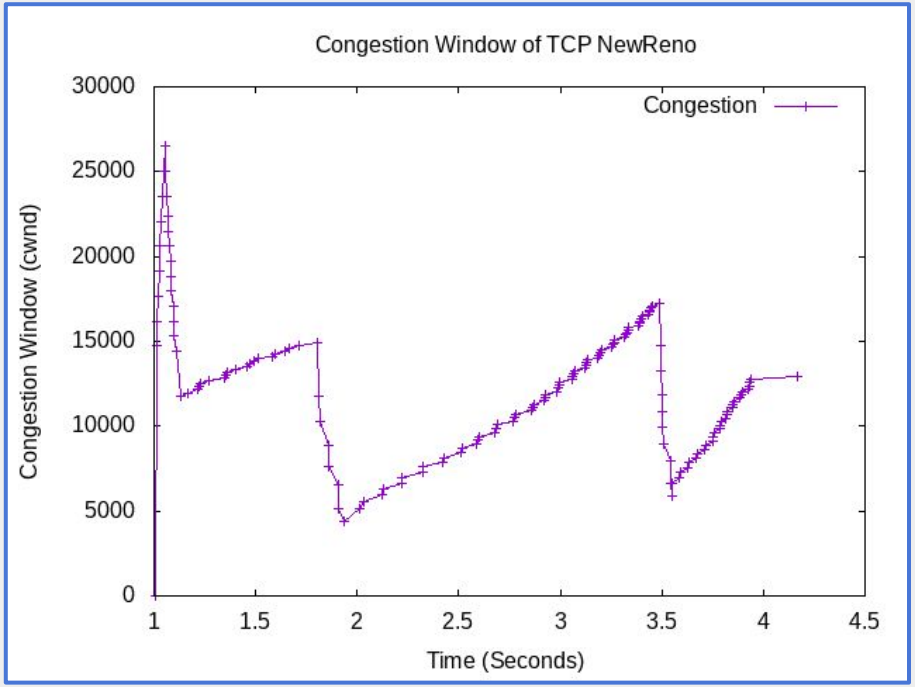
- Network Type: (Wireless) WIFI_STANDARD_80211a
- Total Nodes = 10
- Total Flows = 8

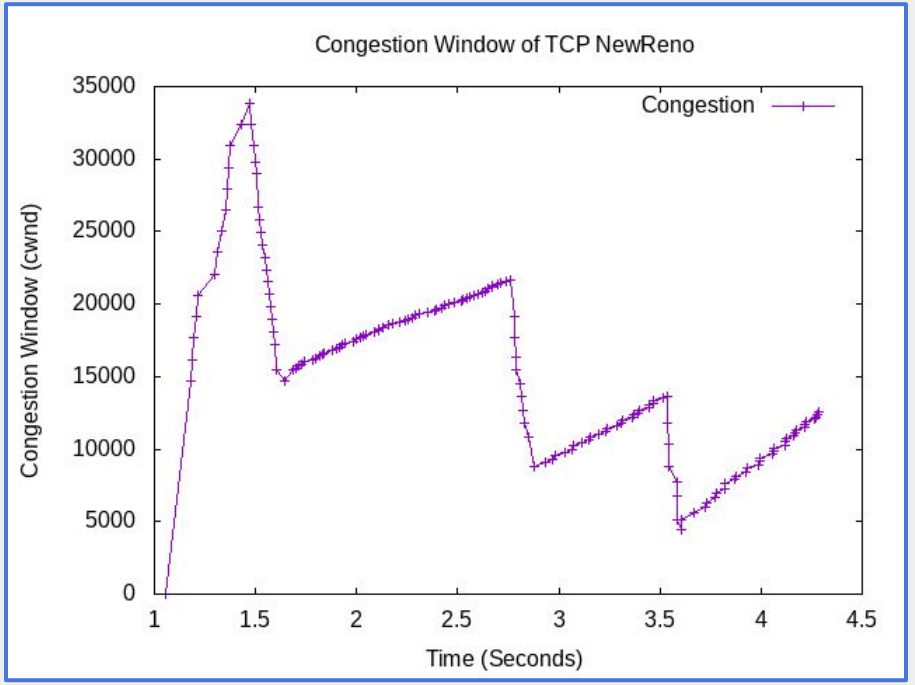


Baseline Algorithm: TCP NewReno

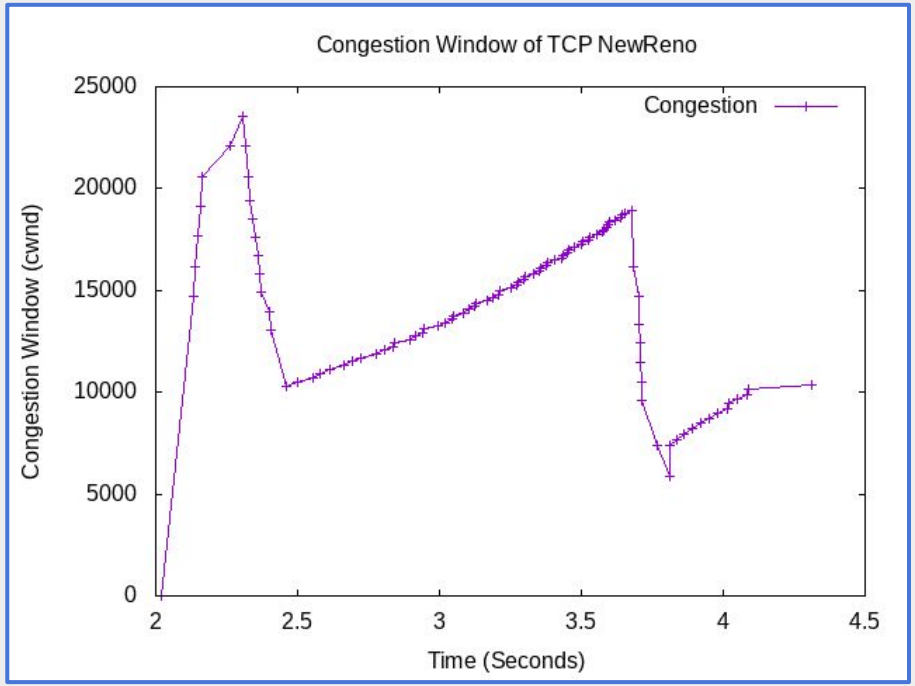
```
1  uint payloadSize = 1472;                       /* Transport layer payload size in bytes. */
2      std::string dataRate = "100Mbps";               /* Application layer datarate. */
3      std::string tcpVariant = "ns3::TcpNewReno";          /* TCP variant type. */
4      // std::string phyRate = "HtMcs7";                  /* Physical layer bitrate. */
5      double simulationTime = 5;                      /* Simulation time in seconds. */
6      // bool pcapTracing = false;
7      uint nPackets = 20000;
8
9      Config::SetDefault ("ns3::TcpL4Protocol::SocketType", StringValue (tcpVariant));
10     Config::SetDefault ("ns3::TcpSocket::SegmentSize", UintegerValue (payloadSize));
```

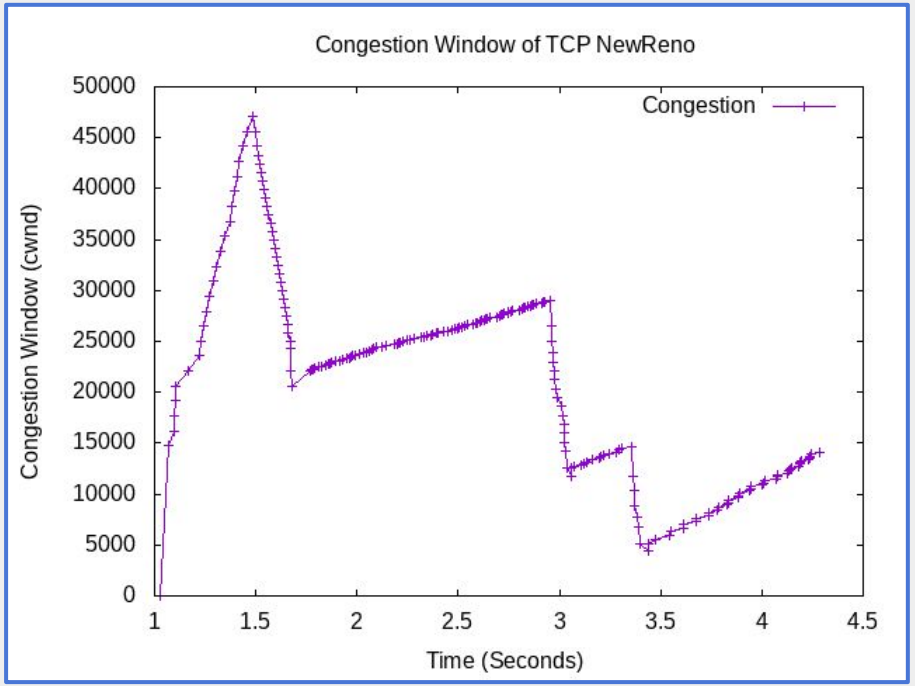
TCP NewReno Graphs



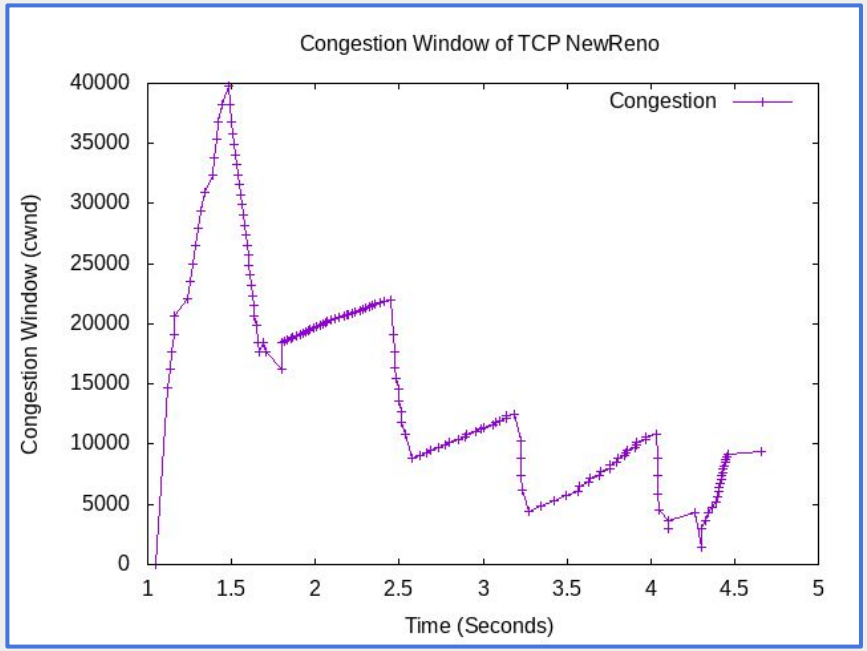


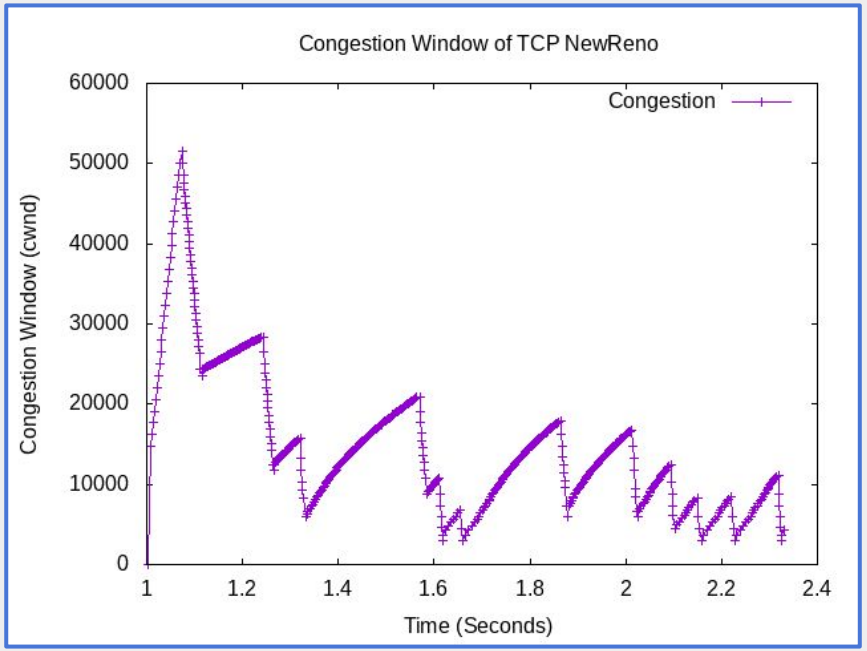
TCP NewReno Graphs





TCP NewReno Graphs





Generating Flows & Use of FlowMonitor

Use of FlowMonitor

```
Ptr<Ipv4FlowClassifier> classifier = DynamicCast <Ipv4FlowClassifier> (fmhelp.GetClassifier());
FlowMonitor::FlowStatsContainer stats = monitor->GetFlowStats();
```

Generating Multiple Flows

```
for(uint i = 0; i< nFlow; i++){

    uint16_t port = 8000+i;
    PacketSinkHelper sinkHelper ("ns3::TcpSocketFactory", InetSocketAddress (Ipv4Address::GetAny (), port));
    ApplicationContainer sinkApp = sinkHelper.Install (wifiStaNodesR.Get(i));
    Ptr<PacketSink> packetSink = StaticCast<PacketSink> (sinkApp.Get(0));
    sinkApp.Start (Seconds (0.));
    sinkApp.Stop (Seconds(simulationTime));

    Ptr<Socket> ns3TcpSocket = Socket::CreateSocket (wifiStaNodesL.Get (i), TcpSocketFactory::GetTypeId ());
    Ptr<MyApp> app = CreateObject<MyApp> ();
    app->Setup (ns3TcpSocket, InetSocketAddress(staInterfacesR.GetAddress(i), port), payloadSize, nPackets, DataRate (dataRate));
    wifiApNodeL.Get (0)->AddApplication (app);
    app->SetStartTime (Seconds (1.));
    app->SetStopTime (Seconds (simulationTime));

    AsciiTraceHelper asciiTraceHelper;
    Ptr<OutputStreamWrapper> stream = asciiTraceHelper.CreateFileStream ("Congestions/customWiFi" + std::to_string(i)+".cwnd");

    ns3TcpSocket->TraceConnectWithoutContext ("CongestionWindow", MakeBoundCallback (&CwndChange, stream));

}
```

Performance Metrics

```
Flow ID: 1
Src Addr: 10.1.2.2   Destination Addr: 10.1.3.2
Sent Packets: 295
Received Packets: 291
Packet Delivery Ratio: 98.6441%
Packet Drop Ratio: 1.35593%
End to end Delay: +2.98079e+10ns
Average Throughput: 1049Kbps
```

```
Flow ID: 2
Src Addr: 10.1.2.3   Destination Addr: 10.1.3.3
Sent Packets: 192
Received Packets: 189
Packet Delivery Ratio: 98.4375%
Packet Drop Ratio: 1.5625%
End to end Delay: +1.70035e+10ns
Average Throughput: 716Kbps
```

```
Flow ID: 3
Src Addr: 10.1.2.4   Destination Addr: 10.1.3.4
Sent Packets: 390
Received Packets: 386
Packet Delivery Ratio: 98.9744%
Packet Drop Ratio: 1.02564%
End to end Delay: +4.13094e+10ns
Average Throughput: 1393Kbps
```

```
Flow ID: 4
Src Addr: 10.1.2.5   Destination Addr: 10.1.3.5
Sent Packets: 299
Received Packets: 293
Packet Delivery Ratio: 97.9933%
Packet Drop Ratio: 2.00669%
End to end Delay: +2.87866e+10ns
Average Throughput: 1002Kbps
```

Performance Metrics

```
Flow ID: 5
Src Addr: 10.1.1.1   Destination Addr: 10.1.3.1
Sent Packets: 232
Received Packets: 227
Packet Delivery Ratio: 97.8448%
Packet Drop Ratio: 2.15517%
End to end Delay: +1.68511e+10ns
Average Throughput: 792Kbps
```

```
Flow ID: 6
Src Addr: 10.1.3.1   Destination Addr: 10.1.1.1
Sent Packets: 146
Received Packets: 146
Packet Delivery Ratio: 100%
Packet Drop Ratio: 0%
End to end Delay: +3.05389e+08ns
Average Throughput: 17Kbps
```

```
Total Stats:
Sent Packets: 2257
Received Packets: 2235
Received Packets: 22
Packet Delivery Ratio: 99.0253%
Packet Drop Ratio: 0.974745%
Average Throughput: 505Kbps
Average Throughput without Acknowledge Flows: 4969Kbps
```

Thank You!