دانشگاه شهید بهشتی

دانشکده مهندسی و علوم کامپیوتر

گزارش پروژه سوم درس شبکه پیشرفته

Random Early Detection Gateways for Congestion Avoidance

نگارش

سهیل ضیائی قهنویه

استاد

دکتر مقصود عباسپور

بهمن ۱۳۹۹

مقدمه

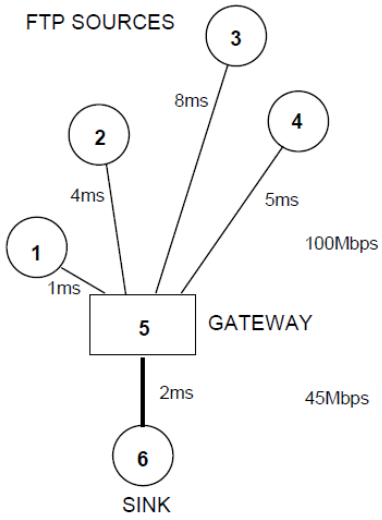
هدف از این پروژه پیاده سازی مقاله [1] به منظور شبیه سازی و مقایسه عمکرد Random Early Detection با Drop Tail Fifo در پروتکل TCP برای کنترل ازدحام می باشد. برای شبیه سازی از ابزار NS3 استفاده شده است و به عنوان راهنما مستندات این ابزار [2] به کار گرفته شده است.

بخش اول: مشاهده کنترل ازدحام از طریق کنترل صف و پنجره ارسال

نحوه پیاده سازی

برای پیاده سازی این بخش مثال red-tests.cc تغییر داده شده است.

توپولوژی شکل ۴ مقاله به صورت زیر بایستی پیاده سازی شود:



برای این کار در فایل part1.cc توپولوژی به صورت زیر پیاده شد:

```
/** Network topology
 *
 *    100Mb/s, 1ms
 * n0---------------|
 *                  |
 *                  |
 *    100Mb/s, 4ms  |
 * n1---------------|
 *                  |   1.5Mbps/s, 20ms        45Mb/s, 2ms
 *                 n4--------------------n5-------------------n6
 *    100Mb/s, 8ms  | <---- Gateway ---->
 * n2---------------|
 *                  |
 *                  |
 *    100Mb/s, 5ms  |
 * n3---------------|
 *
 */
```

همانطور که مشاهده می شود برای Gateway دو نود n4 و n5 اختصاص داده شده است که قرار است در NetDevice های بین این دو نود که bottleneck خواهد شد، صف RED برای کنترل ازدحام نصب شود. در سایر نود ها از صف FIFO استفاده می شود.

عرض باند و تاخیر لینک بین این دو نود Gateway به ترتیب 1.5Mbps/s و 20ms فرض شده است. همچنین از نسخه TcpNewReno به عنوان پروتکل لایه ۴ استفاده می شود:

```
  std::string redLinkDataRate = "1.5Mbps"; // Assumption
  std::string redLinkDelay = "20ms"; // Assumption
  Config::SetDefault ("ns3::TcpL4Protocol::SocketType", StringValue ("ns3::TcpNewReno"));
  // 42 = headers size
  Config::SetDefault ("ns3::TcpSocket::SegmentSize", UintegerValue (1000 - 42));
  Config::SetDefault ("ns3::TcpSocket::DelAckCount", UintegerValue (1));
  GlobalValue::Bind ("ChecksumEnabled", BooleanValue (false));
```

برای صف RED پارامترهای زیر مقداردهی می شود. لازم به ذکر است حداکثر احتمال drop بسته از طریق LInterm مطابق مقاله برابر 1/50 قرارداده شده است:

```
  uint32_t meanPktSize = 500;
  Config::SetDefault ("ns3::RedQueueDisc::MaxSize", StringValue ("1000p"));
  Config::SetDefault ("ns3::RedQueueDisc::MeanPktSize", UintegerValue (meanPktSize));
  Config::SetDefault ("ns3::RedQueueDisc::Wait", BooleanValue (true));
  Config::SetDefault ("ns3::RedQueueDisc::Gentle", BooleanValue (true));
```

```
  Config::SetDefault ("ns3::RedQueueDisc::LInterm", DoubleValue (0.02));
  Config::SetDefault ("ns3::RedQueueDisc::QW", DoubleValue (0.002));
  Config::SetDefault ("ns3::RedQueueDisc::MinTh", DoubleValue (5));
  Config::SetDefault ("ns3::RedQueueDisc::MaxTh", DoubleValue (15));
```

برای شروع پیاده سازی توپولوژی ۷ نود ایجاد می شود:

```
  NodeContainer nodes;
  nodes.Create (7);
```

پشته پروتکل روی نودهای نصب می شود:

```
  InternetStackHelper internet;
  internet.Install (nodes);
```

برای ایجاد صف ها از TrafficControlHelper استفاده می شود:

```
  TrafficControlHelper tchPfifo;
  uint16_t handle = tchPfifo.SetRootQueueDisc ("ns3::PfifoFastQueueDisc");
  tchPfifo.AddInternalQueues (handle, 3, "ns3::DropTailQueue", "MaxSize", StringValue
("1000p"));


  TrafficControlHelper tchRed;
  tchRed.SetRootQueueDisc ("ns3::RedQueueDisc", "LinkBandwidth", StringValue
(redLinkDataRate),
                           "LinkDelay", StringValue (redLinkDelay));
```

لینک های ارتباطی بین نودها ایجاد و همانگونه که گفته شد، بین دو نود gateway صف RED و بین سایر نودها صف FIFO نصب می شود:

```
  TrafficControlHelper tchPfifo;
  uint16_t handle = tchPfifo.SetRootQueueDisc ("ns3::PfifoFastQueueDisc");
  tchPfifo.AddInternalQueues (handle, 3, "ns3::DropTailQueue", "MaxSize", StringValue
("1000p"));


  TrafficControlHelper tchRed;
  tchRed.SetRootQueueDisc ("ns3::RedQueueDisc", "LinkBandwidth", StringValue
(redLinkDataRate),
                           "LinkDelay", StringValue (redLinkDelay));


  // Create channels
  PointToPointHelper p2p;


  p2p.SetQueue ("ns3::DropTailQueue");
  p2p.SetDeviceAttribute ("DataRate", StringValue ("100Mbps"));
  p2p.SetChannelAttribute ("Delay", StringValue ("1ms"));
```

```
NetDeviceContainer devn0n4 = p2p.Install (nodes.Get(0), nodes.Get(4));
tchPfifo.Install (devn0n4);

p2p.SetChannelAttribute ("Delay", StringValue ("4ms"));
NetDeviceContainer devn1n4 = p2p.Install (nodes.Get(1), nodes.Get(4));
tchPfifo.Install (devn1n4);

p2p.SetChannelAttribute ("Delay", StringValue ("8ms"));
NetDeviceContainer devn2n4 = p2p.Install (nodes.Get(2), nodes.Get(4));
tchPfifo.Install (devn2n4);

p2p.SetChannelAttribute ("Delay", StringValue ("5ms"));
NetDeviceContainer devn3n4 = p2p.Install (nodes.Get(3), nodes.Get(4));
tchPfifo.Install (devn3n4);

p2p.SetDeviceAttribute ("DataRate", StringValue ("45Mbps"));
p2p.SetChannelAttribute ("Delay", StringValue ("2ms"));
NetDeviceContainer devn5n6 = p2p.Install (nodes.Get(5), nodes.Get(6));
tchPfifo.Install (devn5n6);

p2p.SetQueue ("ns3::DropTailQueue");
p2p.SetDeviceAttribute ("DataRate", StringValue (redLinkDataRate));
p2p.SetChannelAttribute ("Delay", StringValue (redLinkDelay));
NetDeviceContainer devn4n5 = p2p.Install (nodes.Get(4), nodes.Get(5));
// only backbone link has RED queue disc
QueueDiscContainer queueDiscs = tchRed.Install (devn4n5);
```

آدرس دهی IP روی همه اینترفیس های نودها انجام می شود:

```
Ipv4AddressHelper ipv4;

ipv4.SetBase ("10.1.0.0", "255.255.255.0");
Ipv4InterfaceContainer i0i4 = ipv4.Assign (devn0n4);

ipv4.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer i1i4 = ipv4.Assign (devn1n4);

ipv4.SetBase ("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer i2i4 = ipv4.Assign (devn2n4);

ipv4.SetBase ("10.1.3.0", "255.255.255.0");
Ipv4InterfaceContainer i3i4 = ipv4.Assign (devn3n4);
```

```
ipv4.SetBase ("10.1.4.0", "255.255.255.0");
Ipv4InterfaceContainer i4i5 = ipv4.Assign (devn4n5);

ipv4.SetBase ("10.1.5.0", "255.255.255.0");
Ipv4InterfaceContainer i5i6 = ipv4.Assign (devn5n6);
```

جداول مسیر یابی به صورت استاتیک و پیش فرض ایجاد می شود:

```
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
```

اپلیکیشن گیرنده روی نود n6 نصب می شود:

```
uint16_t port = 50000;
Address sinkLocalAddress (InetSocketAddress (Ipv4Address::GetAny (), port));
PacketSinkHelper sinkHelper ("ns3::TcpSocketFactory", sinkLocalAddress);
ApplicationContainer sinkApp = sinkHelper.Install (nodes.Get(6));
```

اپلیکیشن فرستنده از نوع bulkSend (جهت شبیه سازی فرستنده های ftp مطابق مقاله) روی نودهای n0 تا n3 نصب می شود:

```
ApplicationContainer sendApp;
BulkSendHelper bulkSend ("ns3::TcpSocketFactory", Address ());
AddressValue remoteAddress
  (InetSocketAddress (i5i6.GetAddress (1), port));
bulkSend.SetAttribute ("Remote", remoteAddress);
sendApp = bulkSend.Install (nodes.Get(0));
sendApp.Start (Seconds (0.0));
sendApp = bulkSend.Install (nodes.Get(1));
sendApp.Start (Seconds (0.2));
sendApp = bulkSend.Install (nodes.Get(2));
sendApp.Start (Seconds (0.4));
sendApp = bulkSend.Install (nodes.Get(3));
sendApp.Start (Seconds (0.6));
```

در نهایت مدت شبیه سازی ۱۰ ثانیه در نظر گرفته شده و شبیه سازی انجام می شود.

نحوه ضبط نتایج

با توجه به شکل ۳ مقاله به داده های زیر نیاز است:

۱- طول (لحظه ای) صف RED در مقاطع زمانی منظم

۲- طول میانگین صف RED در مقاطع زمانی منظم

۳- تعداد (لحظه ای) بسته های در حال ارسال از طریق هر یک از جریان ها

برای ضبط موارد ۱ و ۲ از متد زیر استفاده می شود که هر 1/100 ثانیه طول صف را چک می کند و به همراه مینگین در فایل های جداگانه ای ذخیره می کند:

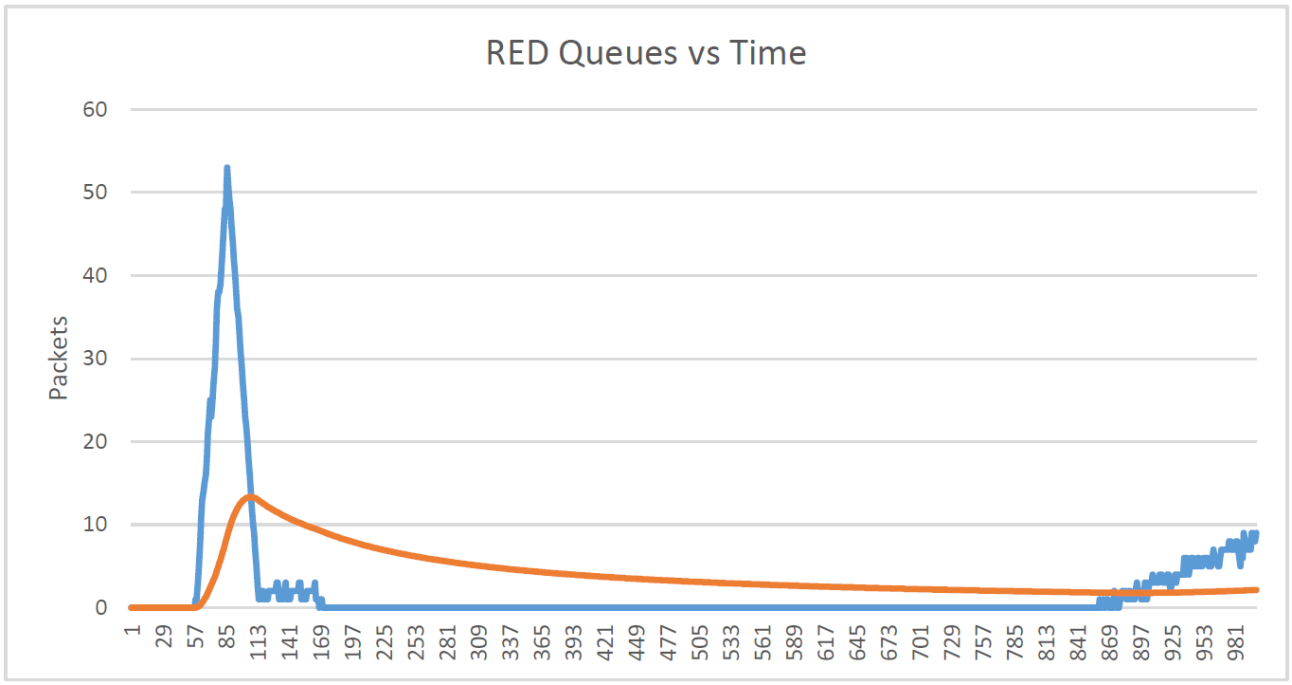
```
void
CheckQueueSize (Ptr<QueueDisc> queue)
{
  uint32_t qSize = queue->GetCurrentSize ().GetValue ();

  avgQueueSize += qSize;
  checkTimes++;

  // check queue size every 1/100 of a second
  Simulator::Schedule (Seconds (0.01), &CheckQueueSize, queue);

  std::ofstream fPlotQueue (filePlotQueue.str ().c_str (), std::ios::out|std::ios::app);
  fPlotQueue << Simulator::Now ().GetSeconds () << " " << qSize << std::endl;
  fPlotQueue.close ();

  std::ofstream fPlotQueueAvg (filePlotQueueAvg.str ().c_str (),
std::ios::out|std::ios::app);
  fPlotQueueAvg << Simulator::Now ().GetSeconds () << " " << avgQueueSize / checkTimes <<
std::endl;
  fPlotQueueAvg.close ();
}
```

نمودار (لحظه ای و میانگین) تعداد بسته های در صف RED در قالب فایل queue-time.pdf از روی فایلهای ذخیره شده توسط نرم افزار Excel تولید می شوند.

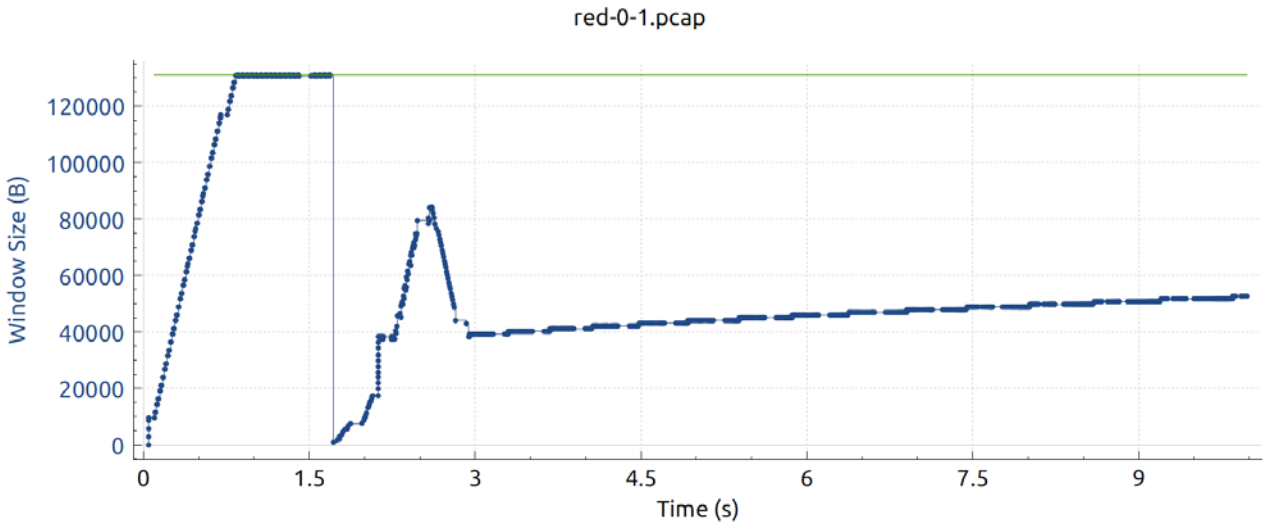
برای مورد ۳، از Pcap استفاده کرده و بسته های ارسال شده از هر نود در فایل های pcap ذخیره می شود. فایل های مذکور با نرم افزار Wireshark باز شده و از طریقه منوی Statistics گزینه TCP Stream Graph نمودارهای پنجره ارسال تولید می شوند.

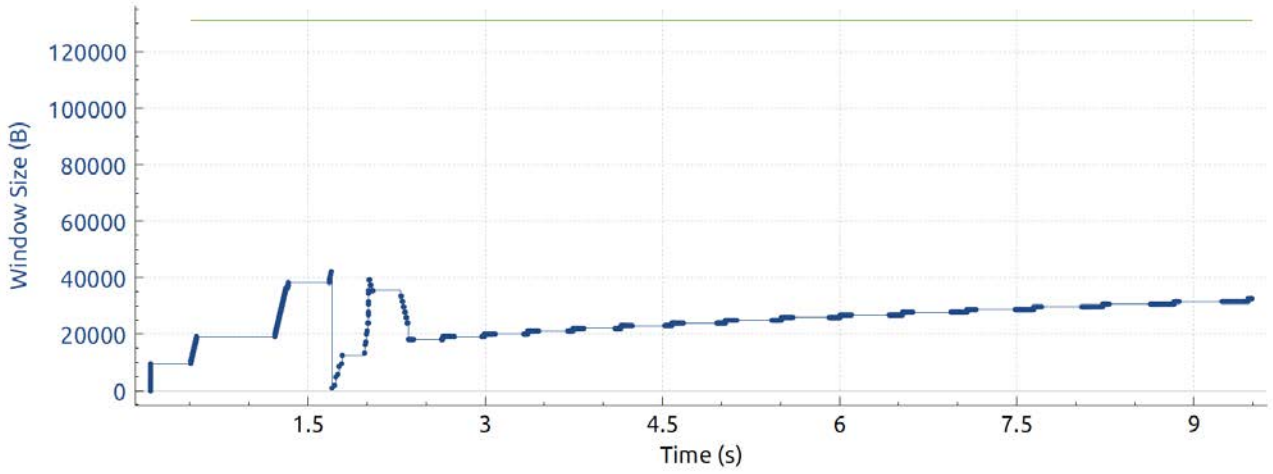
نتایج

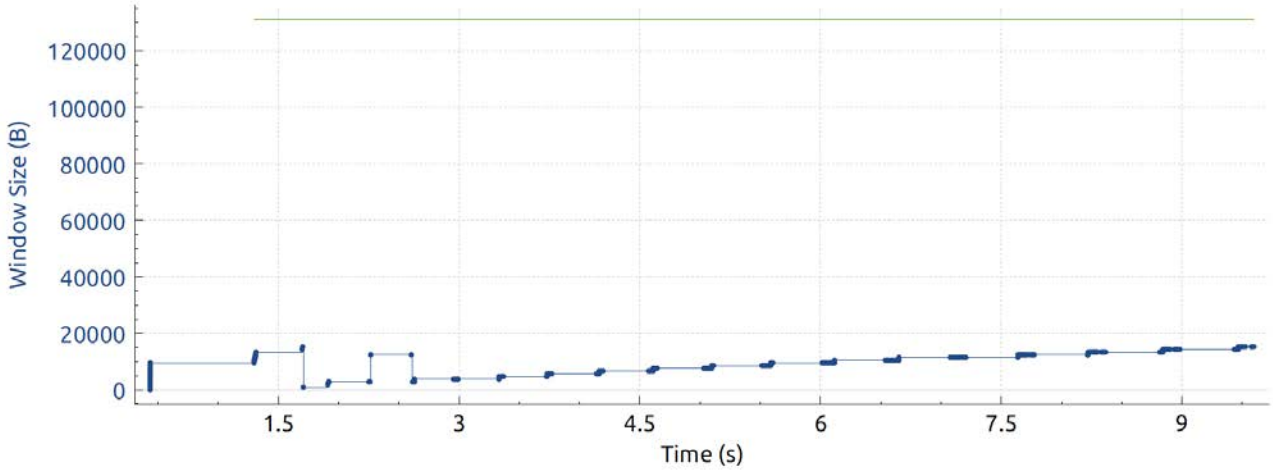
نمودار (لحظه ای و میانگین) تعداد بسته های در صف RED بر حسب زمان:

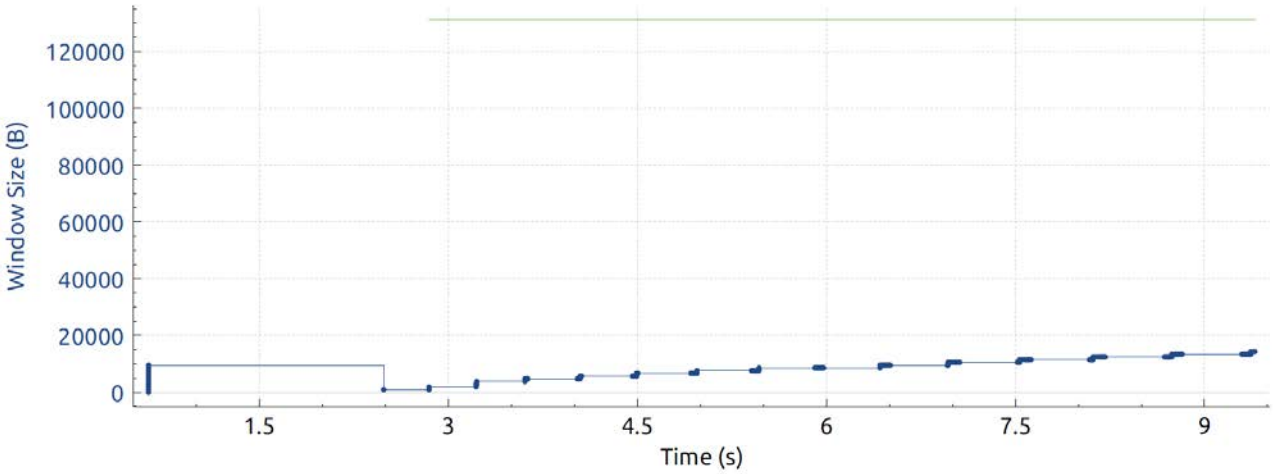


نمودار پنجره ارسال نودهای n0 تا n3 بر حسب زمان:




Window Size (B)
120000
100000
80000
60000
40000
20000
0
1.5
3
4.5
6
7.5
9
Time (s)

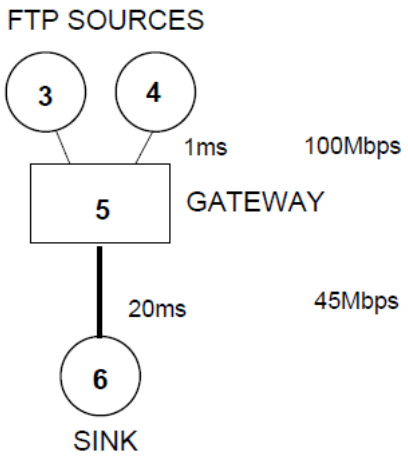

Window Size (B)
120000
100000
80000
60000
40000
20000
0
1.5
3
4.5
6
7.5
9
Time (s)


Window Size (B)
120000
100000
80000
60000
40000
20000
0
1.5
3
4.5
6
7.5
9
Time (s)

بخش دوم: مقایسه میانگین طول صف RED و صف FIFO بر حسب Throughput

نحوه پیاده سازی

توپولوژی شکل ۶ مقاله به صورت زیر است:



همانند بخش قبل، برای Gateway دو نود در نظر گرفته می شود:

```
/** Network topology
 *
 *    100Mb/s, 1ms
 * n0---------------|
 *                  |   1.5Mbps/s, 20ms        45Mb/s, 20ms
 *                 n2-------------------n3-------------------n4
 *    100Mb/s, 1ms  | <---- Gateway ---->
 * n1---------------|
 *
 */
```

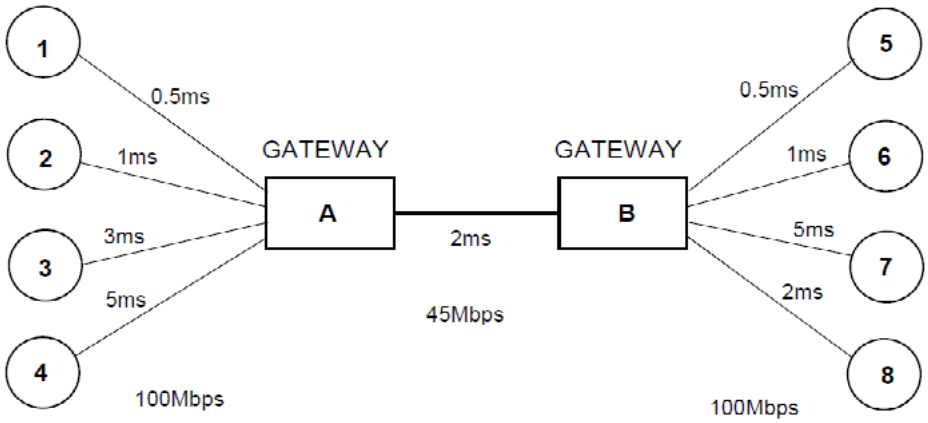
برای لینک بین n2 و n3 که تبدیل به bottleneck شبکه خواهد شد، یک بار صف RED و یک بار صف FIFO تخصیص داده می شود. شبکه برای هر یک از این حالات، در Throughput های مختلف شبیه سازی شده و طول میانگین صف ضبط می شود. برای تولید throughput های مختلف برای صف FIFO از افزایش بافر و در RED از افزایش و کاهش Min_{th} و Max_{th} (حد پایین و بالای تعیین وضعیت بسته برای ورود به صف) استفاده می شود.

سایر مشخصات پیاده سازی همانند بخش قبل است؛ اپلیکیشن فرستنده روی n0 و n1 و اپلیکیشن گیرنده روی n4 نصب می شود.

بخش سوم: مشاهده طول صف RED و تعداد بسته در ترافیک دو طرفه و پر ازدحام

نحوه پیاده سازی

توپولوژی شکل ۱۰ مقاله به صورت زیر است:



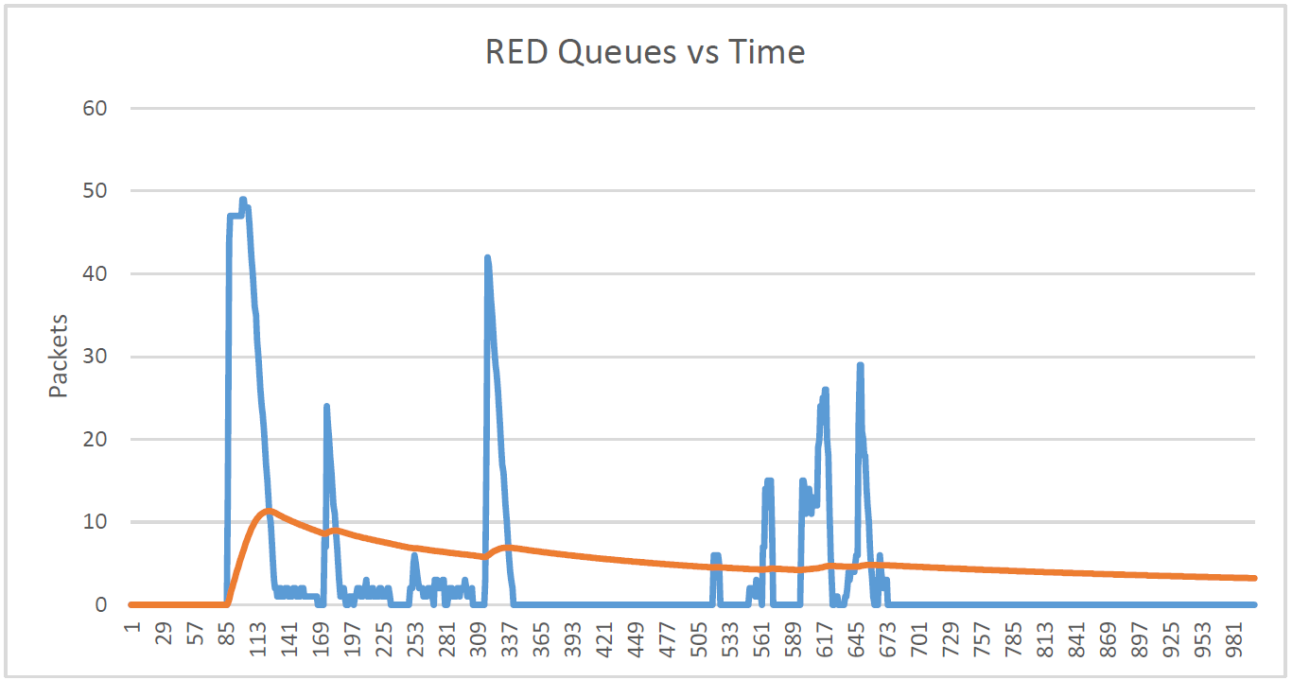
همانند بخش های قبل، برای هر یک از Gateway ها دو نود در نظر گرفته می شود:

```
/** Network topology
 *
 *    100Mb/s, 0.5ms                                                  100Mb/s, 0.5ms
 * n0---------------|                                              |---------------n8
 *                  |                                              |
 *                  |                                              |
 *    100Mb/s, 1ms  |                                              | 100Mb/s, 1ms
 * n1---------------|                                              |---------------n9
 *                  |   1.5Mbps/s, 20ms       45Mb/s, 2ms      1.5Mbps/s, 20ms    |
 *                 n4-------------------n5-------------------n6-------------------n7
 *    100Mb/s, 3ms  | <--- Gateway A ----->                   <--- Gateway B ----> | 100Mb/s, 5ms
 * n2---------------|                                              |---------------n10
 *                  |                                              |
 *                  |                                              |
 *    100Mb/s, 5ms  |                                              | 100Mb/s, 2ms
 * n3---------------|                                              |---------------n11
 *
 */
```

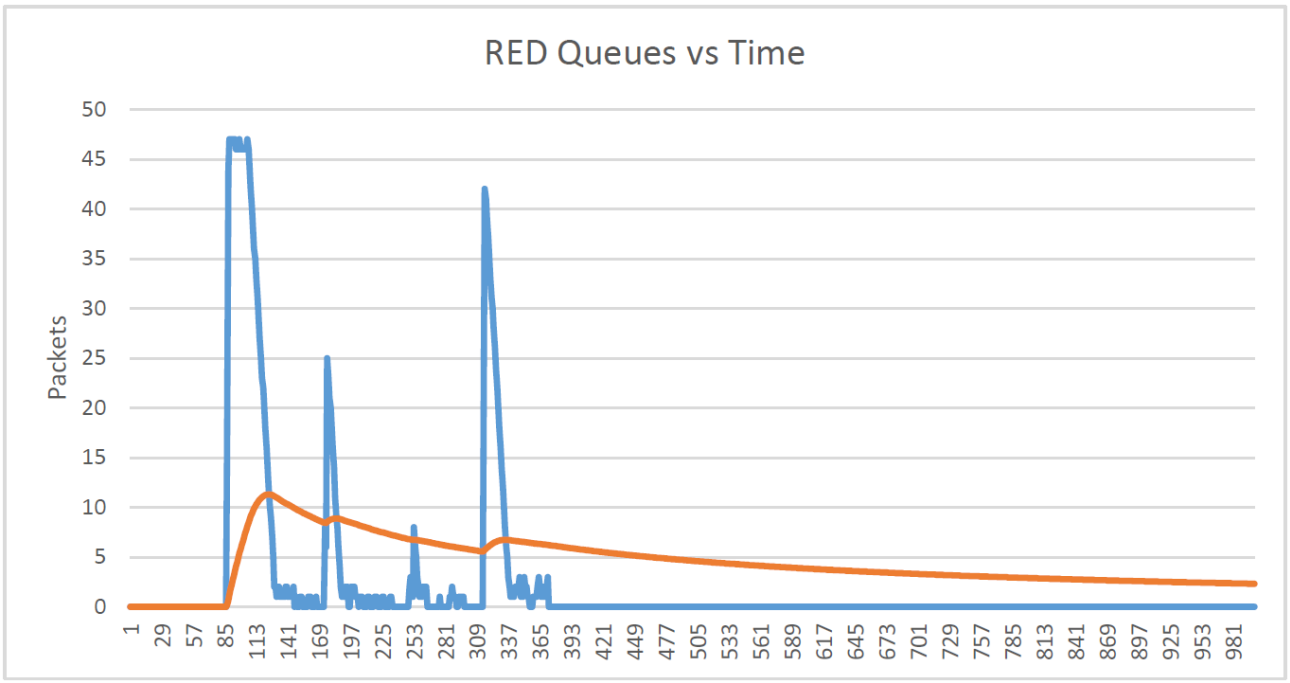
سایر مشخصات پیاده سازی همانند بخش های قبل است؛ با این تفاوت که اپلیکیشن های فرستنده و گیرنده روی همه نودهای غیر Gateway نصب می شود.

نتایج

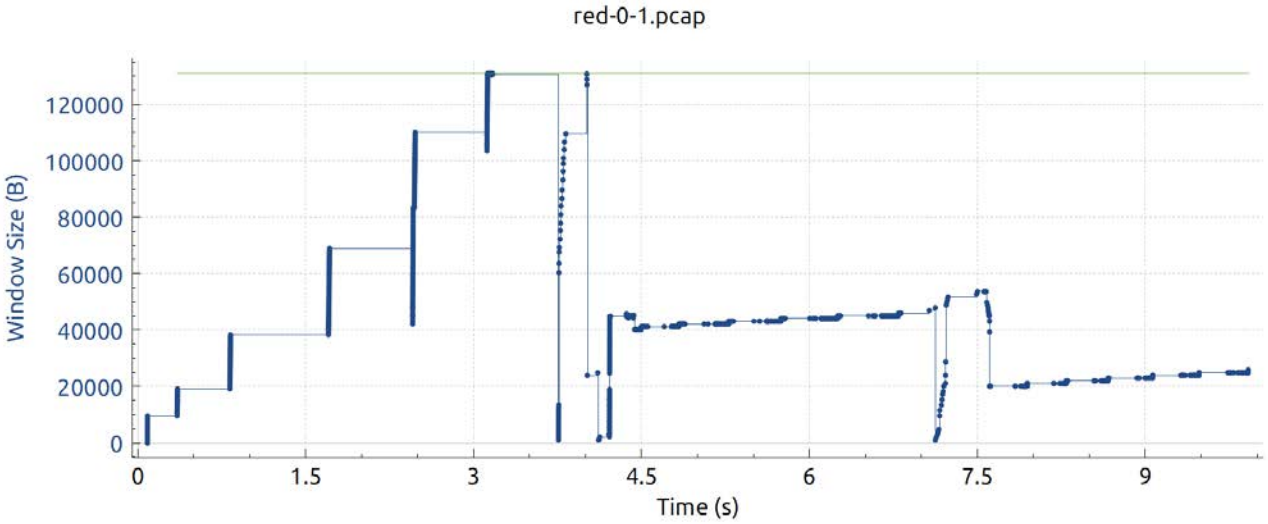
نمودار (لحظه ای و میانگین) تعداد بسته های در صف Gateway A بر حسب زمان:

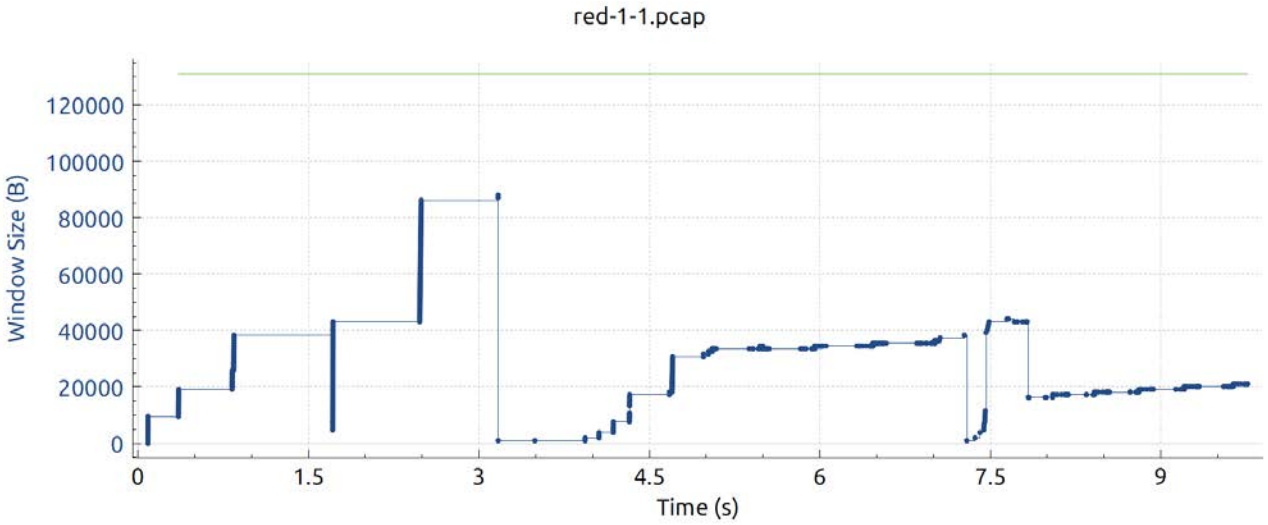


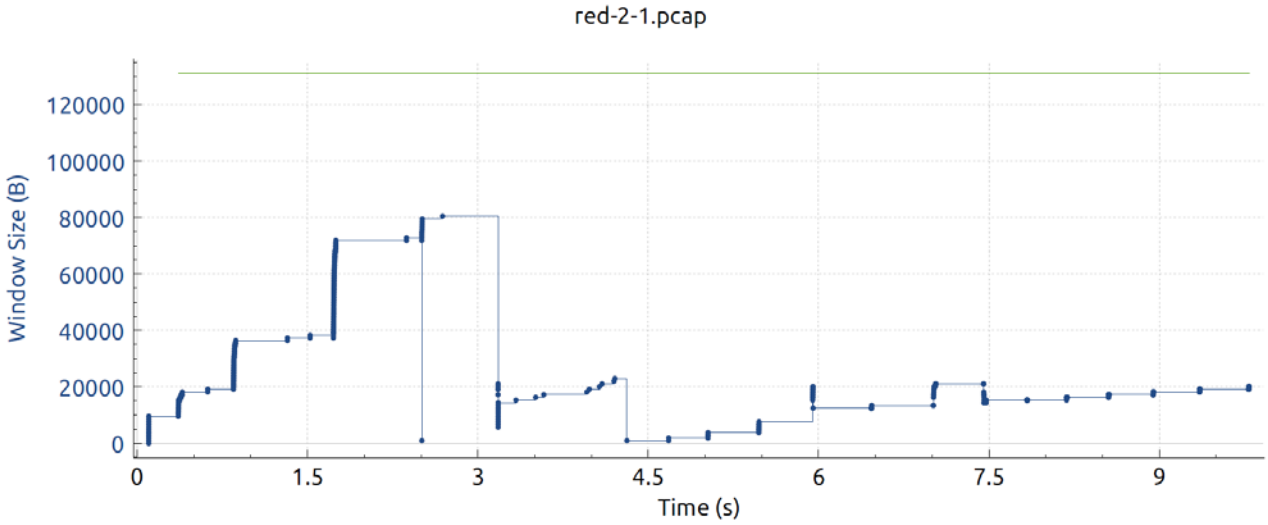
نمودار (لحظه ای و میانگین) تعداد بسته های در صف Gateway B بر حسب زمان:

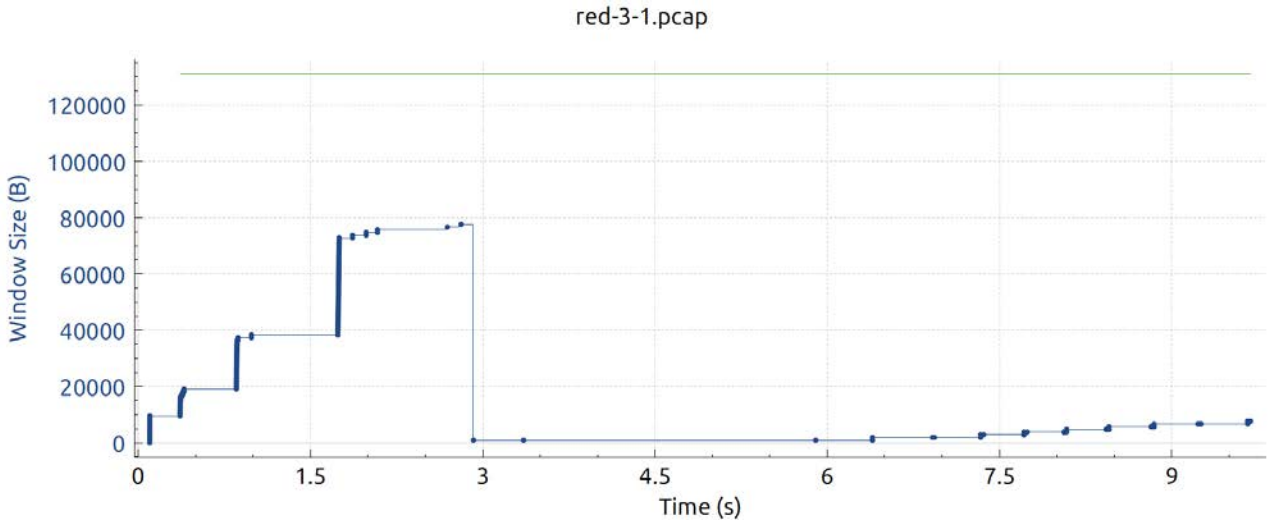


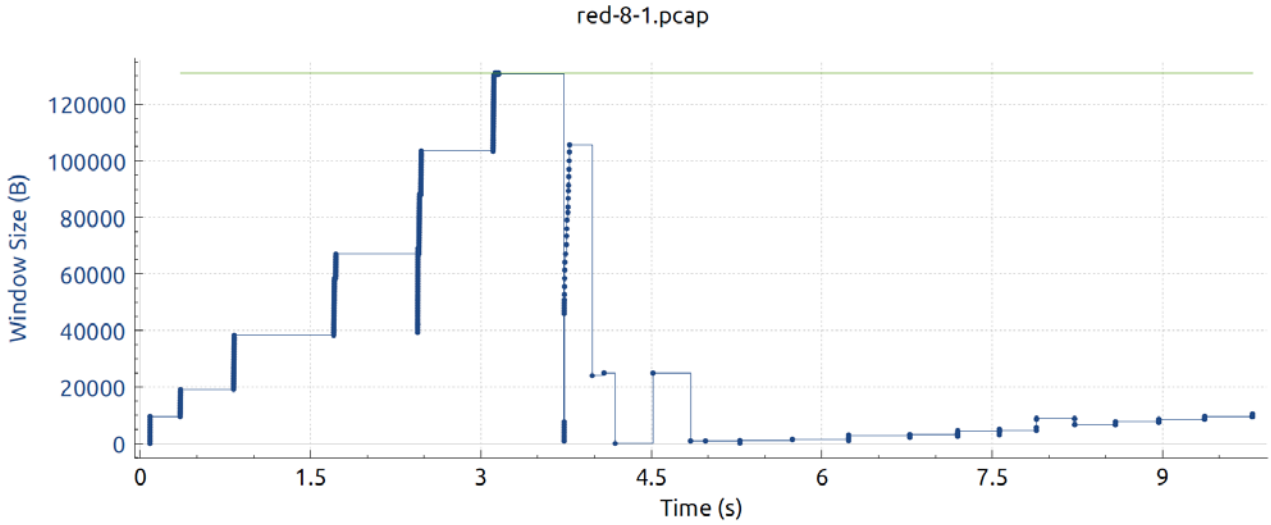
نمودار پنجره ارسال نودهای n0 تا n3 و n8 تا n11 بر حسب زمان:

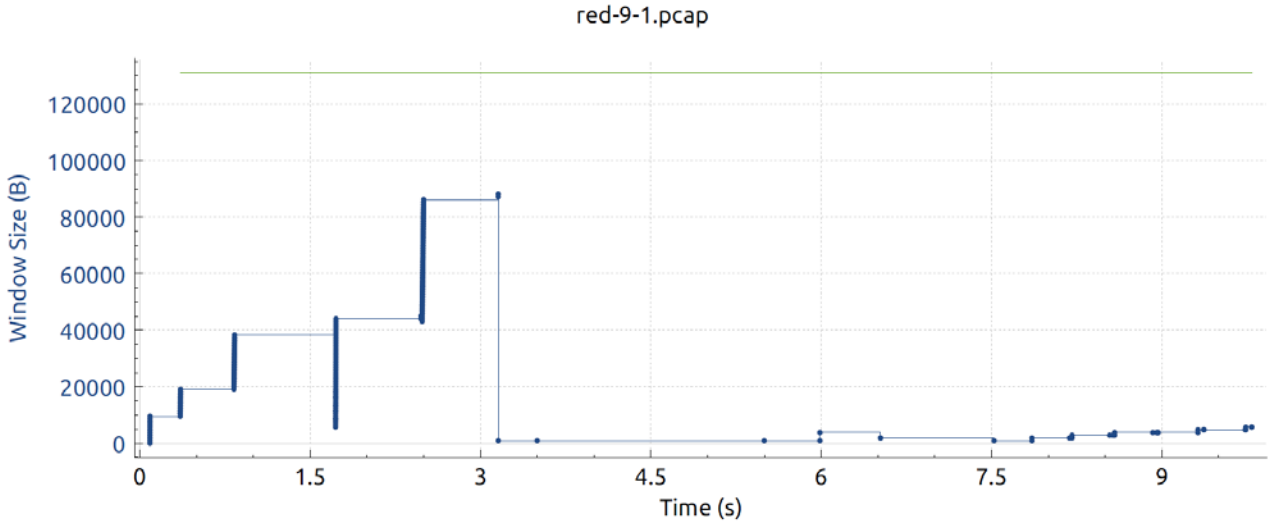

red-0-1.pcap
Window Size (B)
120000
100000
80000
60000
40000
20000
0
0
1.5
3
4.5
6
7.5
9
Time (s)

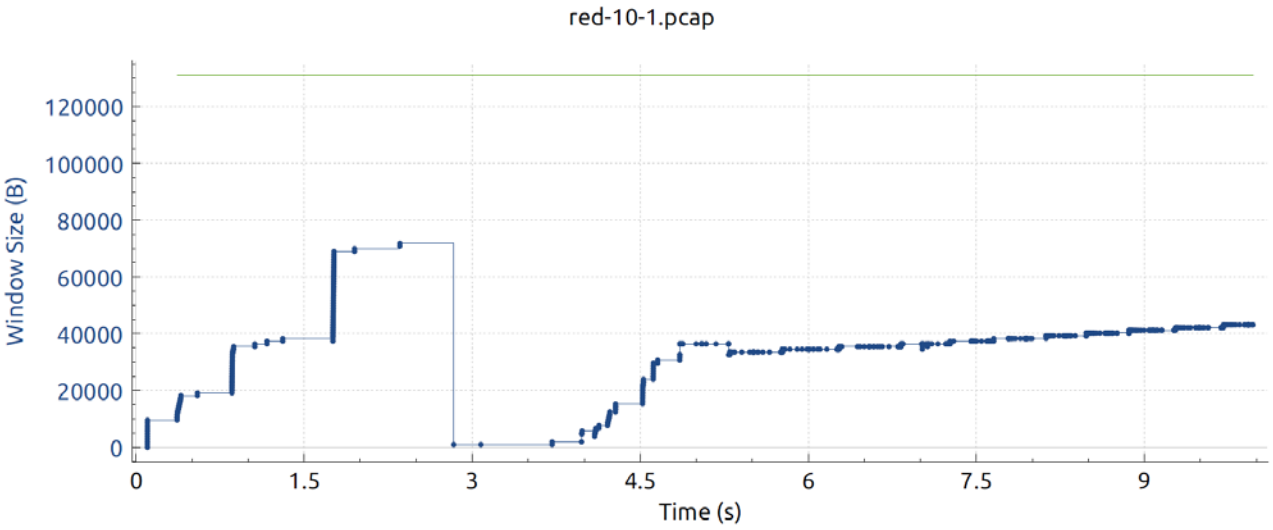

red-1-1.pcap
Window Size (B)
120000
100000
80000
60000
40000
20000
0
0
1.5
3
4.5
6
7.5
9
Time (s)


red-2-1.pcap
Window Size (B)
120000
100000
80000
60000
40000
20000
0
0
1.5
3
4.5
6
7.5
9
Time (s)


red-3-1.pcap
Window Size (B)
120000
100000
80000
60000
40000
20000
0
0
1.5
3
4.5
6
7.5
9
Time (s)


red-8-1.pcap
Window Size (B)
120000
100000
80000
60000
40000
20000
0
0
1.5
3
4.5
6
7.5
9
Time (s)


red-9-1.pcap
Window Size (B)
120000
100000
80000
60000
40000
20000
0
0
1.5
3
4.5
6
7.5
9
Time (s)



مراجع

[1] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM TRANSACTIONS ON NETWORKING.*, vol. 1, 1993.

[2] [Online]. Available: https://www.nsnam.org/doxygen/index.html.