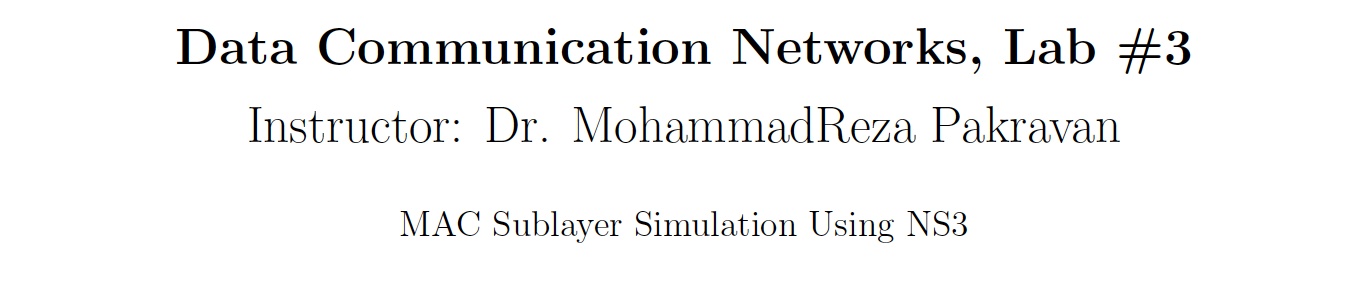
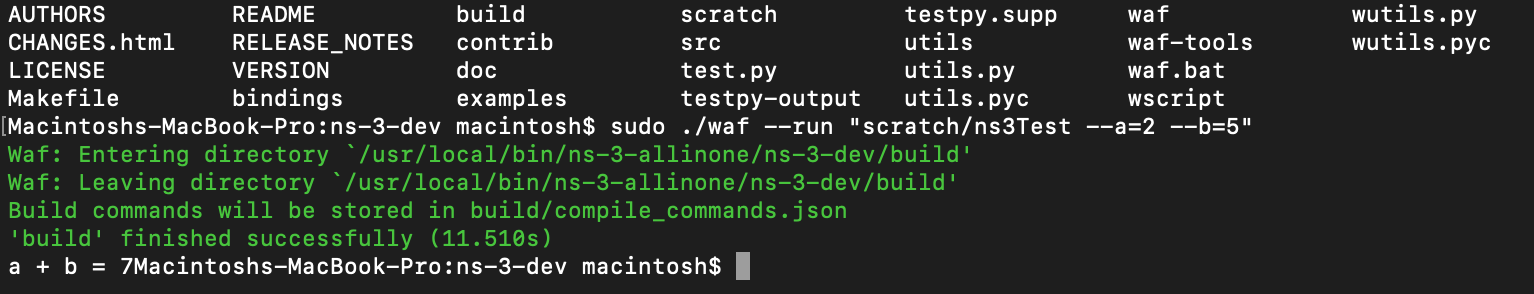
به نام خدا



علی فتحی ۹۴۱۰۹۲۰۵

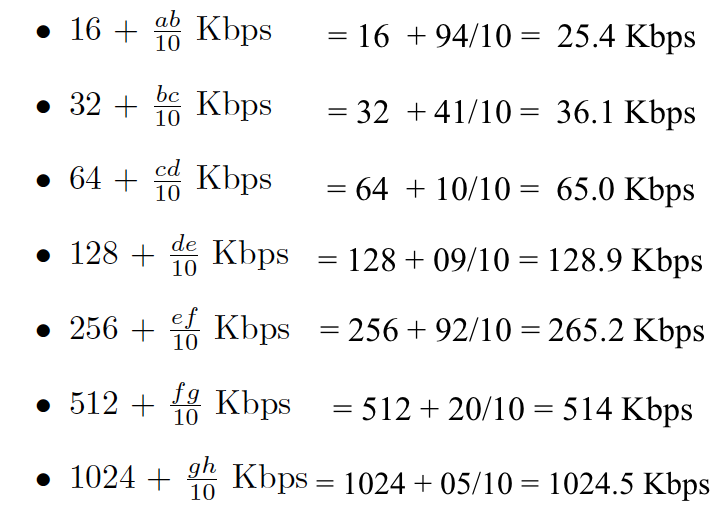
مقدمه:

بخش۲.۱: نمونه اجرای کد نوشته در صورت سوال، در کامپیوتر من بدین صورت است:



چهار رقم سمت راست شماره دانشجویی من، ۹۲۰۵ است لذا Base Address را برابر 19.20.5.0 قرار می‌دهم.

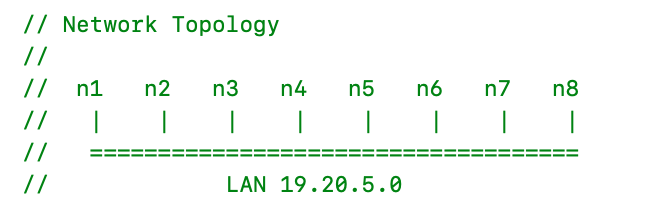
همچنین نرخ های ترافیک بر اساس شماره دانشجویی من بدین صورت خواهد بود:



توضیح API کد:

1. توپولوژی:

بر اساس خواسته سوال، توپولوژی به شکل زیر است:



که در واقع همان CSMA است (چیزی که از این پروتکل استفاده می‌کند. تفاوتی در دستورات ندارد).

1. ساختن Node ها

بصورت زیر، به کمک NodeContainer، ۸ نقطه برای استفاده از این شبکه تعریف می‌کنیم:

  uint32\_t nCsma = 8;

NodeContainer csmaNodes;

  csmaNodes.Create (nCsma);

1. ساختن Channel

به کمک CsmaHelper، کانال را تعریف کرده و ویژگی‌های تاخیر و DataRate آن را مشخص می‌کنیم:

  CsmaHelper csma;

  csma.SetChannelAttribute ("DataRate", StringValue ("1024Kbps"));

  csma.SetChannelAttribute ("Delay", StringValue ("2ms"));

1. ساختن دستگاه‌ها و Install کردن نقاط روی کانال

به کمک NetDeviceContainer، این کار را انجام می‌دهیم:

  NetDeviceContainer csmaDevices;

  csmaDevices = csma.Install (csmaNodes);

1. ساختن پروتکل اینترنت

پروتکل اینترنت را به کمک InternetStackHelper، روی نقاط شبکه قرار می‌دهیم:

  InternetStackHelper stack;

  stack.Install (csmaNodes);

1. ساختن IPv4 برای ارتباط نقاط شبکه

در اینجا Net Mask (آدرس‌های معتبر در این شبکه) و Base Address را به کمک Ipv4AddressHelper تعیین می‌کنیم. سپس آن را به لایه پایین تر (Device ها) assign می‌کنیم.

  Ipv4AddressHelper address;

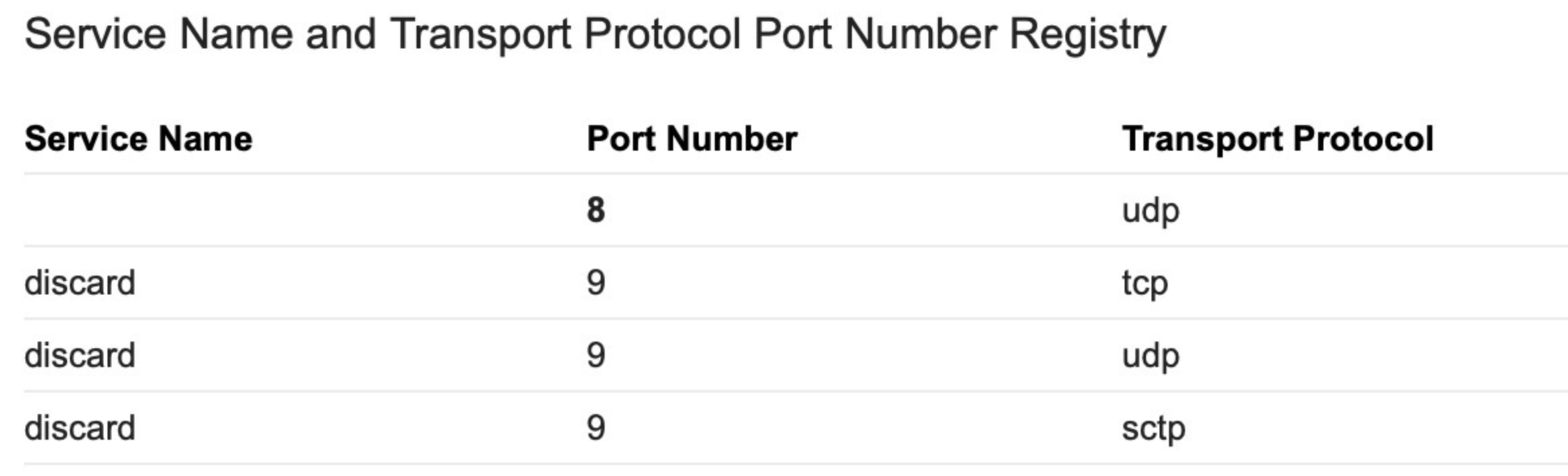
  address.SetBase ("19.20.5.0", "255.255.255.0");

  Ipv4InterfaceContainer csmaInterfaces;

  csmaInterfaces = address.Assign (csmaDevices);

1. آماده کردن Server ها برای دریافت بسته‌ها

از Echo UDP استفاده می‌کنیم. برای ارتباط‌ها، همگی از Port شماره ۹ استفاده می‌کنیم (پورت شماره ۹، درواقع ویژگی زیر را دارد:



و Discard Protocol در واقع یک سرویس روی Internet Protocol تعریف شده در RFC 863برای اعمال testing ، debugging ، measurement و host management purpose است).

سپس سرور را روی اپلیکیشن (ApplicationContainer) راه‌اندازی می‌نماییم. زمان فعالیت سرور را نیز از زمان ۱ تا ۱۵ ثانیه در نظر می‌گیریم.

  endTime = Seconds (15.0);

UdpEchoServerHelper echoServer5 (9);

  UdpEchoServerHelper echoServer6 (9);

  UdpEchoServerHelper echoServer3 (9);

  UdpEchoServerHelper echoServer4 (9);

  ApplicationContainer serverApps5 = echoServer5.Install (csmaNodes.Get (4));

  serverApps5.Start (Seconds (1.0));

  serverApps5.Stop (endTime);

  ApplicationContainer serverApps6 = echoServer6.Install (csmaNodes.Get (5));

  serverApps6.Start (Seconds (1.0));

  serverApps6.Stop (endTime);

  ApplicationContainer serverApps3 = echoServer3.Install (csmaNodes.Get (2));

  serverApps3.Start (Seconds (1.0));

  serverApps3.Stop (endTime);

  ApplicationContainer serverApps4 = echoServer4.Install (csmaNodes.Get (3));

  serverApps4.Start (Seconds (1.0));

  serverApps4.Stop (endTime);

1. تعیین Client ها برای ارسال بسته‌ها

اینجا هم از Echo UDP استفاده می‌کنیم و آن را روی اپلیکیشن (ApplicationContainer) نصب می‌نماییم. پورت‌ها در اینجا نیز شماره ۹ هستند (دقت شود شماره متغیر ها یکی کمتر از عدد واقعی نقاط است زیرا اندیس‌ها از ۰ شروع می‌شوند).

  maxPackets = UintegerValue (2000);

  interval = TimeValue (Seconds (0.315));

  UdpEchoClientHelper echoClient1 (csmaInterfaces.GetAddress (4), 9);

  echoClient1.SetAttribute ("MaxPackets", maxPackets);

  echoClient1.SetAttribute ("Interval", interval);

  echoClient1.SetAttribute ("PacketSize", UintegerValue (1024));

  UdpEchoClientHelper echoClient2 (csmaInterfaces.GetAddress (5), 9);

  echoClient2.SetAttribute ("MaxPackets", maxPackets);

  echoClient2.SetAttribute ("Interval", interval);

  echoClient2.SetAttribute ("PacketSize", UintegerValue (1024));

  UdpEchoClientHelper echoClient7 (csmaInterfaces.GetAddress (2), 9);

  echoClient7.SetAttribute ("MaxPackets", maxPackets);

  echoClient7.SetAttribute ("Interval", interval);

  echoClient7.SetAttribute ("PacketSize", UintegerValue (1024));

  UdpEchoClientHelper echoClient8 (csmaInterfaces.GetAddress (3), 9);

  echoClient8.SetAttribute ("MaxPackets", maxPackets);

  echoClient8.SetAttribute ("Interval", interval);

  echoClient8.SetAttribute ("PacketSize", UintegerValue (1024));

  ApplicationContainer clientApps1 = echoClient1.Install (csmaNodes.Get (0));

  clientApps1.Start (Seconds (2.0));

  clientApps1.Stop (endTime);

  ApplicationContainer clientApps2 = echoClient2.Install (csmaNodes.Get (1));

  clientApps2.Start (Seconds (2.0));

  clientApps2.Stop (endTime);

  ApplicationContainer clientApps7 = echoClient7.Install (csmaNodes.Get (6));

  clientApps7.Start (Seconds (2.0));

  clientApps7.Stop (endTime);

  ApplicationContainer clientApps8 = echoClient8.Install (csmaNodes.Get (7));

  clientApps8.Start (Seconds (2.0));

  clientApps8.Stop (endTime);

در اینجا چند نکته درباره Attribute ها وجود دارد. نرخ ترافیک به عنوان ورودی به برنامه داده می‌شود. برای آن که مدل استاندارد باشد، سایز packet ها را برابر و همه را 1024 Byte یا 1 KByte می‌گیریم (زیرا اندازه packet ها معمولا بر اساس استاندارد تعیین می‌شود و بسته به نرخ ترافیک تغییر نمی‌کند). در نتیجه Interval باید بر اساس نرخ ترافیک تعیین شود. ترافیک هر Client ، مستقلا باید برابر با عدد خواسته شده در صورت سوال باشد (که در مقدمه محاسبه شده است). این عدد برابر نسبت سایز packet به Interval است. پس Interval از تقسیم 1024 Byte یا معادلات 8 Kbit به BitRate بدست می‌آید. تا سه رقم، این مقدار برای ترافیک‌های من بدین صورت است:

{[Traffic, Interval] = [25.4Kbps, 314.96ms], [36.1Kbps, 221.6ms], [65.0Kbps, 123.077ms], [128.9Kbps, 62.06ms], [265.2Kbps, 30.166ms], [514.0Kbps, 15.56ms], [1024.5Kbps, 7.809ms]}

پس، maxPackets را مناسب است عددی بگیریم که همه این مقادیر، در بازه ۱۵ ثانیه‌ای ارسال جا شوند، و مقدار ۲۰۰۰ برای این مهم خوب است (حدود ۱۹۰۰ بسته در ۱۵ ثانیه، در بازه‌های ۷.۸۰۹ میلی ثانیه‌ای ارسال می‌شوند). نکته: این که در کدام نرخ ترافیک هستیم، در ورودی با یک عدد بین ۱ تا ۷ مشخص می‌شود. در این باره توضیح داده می‌شود.

1. تعیین فایل‌های خروجی

در انتها، سه نوع خروجی نیز برای کد محاسبه می‌گردد؛ .tr ، .pcap و دو نوع فایل .xml . کاربری هر کدام مشخص است و در بخش خروجی‌ها نتایجشان مرور می‌شود.

std::string modeStr;

  modeStr = std::to\_string(modeNumber);

  std::string fileName = "HW3/HW3\_94109205\_Mode\_" + modeStr;

  csma.EnablePcapAll (fileName);

  AsciiTraceHelper asciiTraceHelper;

  Ptr<OutputStreamWrapper> stream = asciiTraceHelper.CreateFileStream(fileName+".tr");

  csma.EnableAsciiAll(stream);

  AnimationInterface animationInterface(fileName+".xml");

  animationInterface.SetConstantPosition (csmaNodes.Get (0), 50, 200);

  animationInterface.SetConstantPosition (csmaNodes.Get (1), 100, 200);

  animationInterface.SetConstantPosition (csmaNodes.Get (2), 150, 200);

  animationInterface.SetConstantPosition (csmaNodes.Get (3), 200, 200);

  animationInterface.SetConstantPosition (csmaNodes.Get (4), 250, 200);

  animationInterface.SetConstantPosition (csmaNodes.Get (5), 300, 200);

  animationInterface.SetConstantPosition (csmaNodes.Get (6), 350, 200);

  animationInterface.SetConstantPosition (csmaNodes.Get (7), 400, 200);

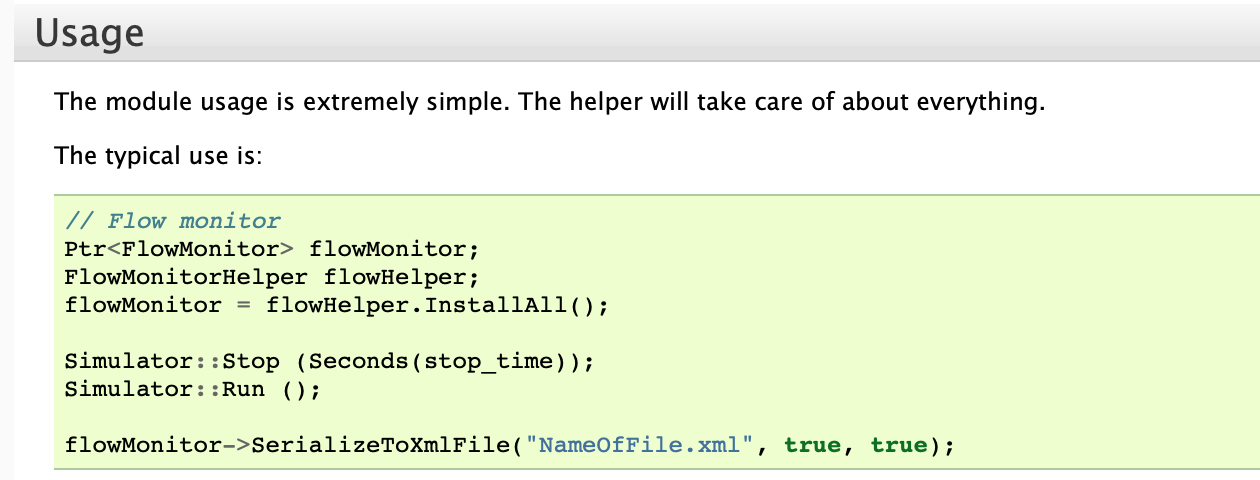
  animationInterface.EnableIpv4RouteTracking(fileName+".xml",

                                              Time (Seconds (1.0)),

                                              Time (endTime),

                                              Time (Seconds (5.0)));

یک نقطه لازم به ذکر است، برای انجام محاسبات delay و throughput ، از یک کتابخانه دیگر به نام FlowMonitorHelper استفاده شده است‌ (ایده انجام این کار توسط بقیه افراد کلاس پیدا شده که با این کلاس، محاسبات دستی ساده‌تر می‌شود. همچنین در کدهای معمول نوشته شده در سایت‌های مختلف نیز برای محاسبه delay و throughput از همین کتابخانه استفاده شده است.) نحوه استفاده از آن نیز در زیر آمده است:



درباره ورودی دادن و اجرا:

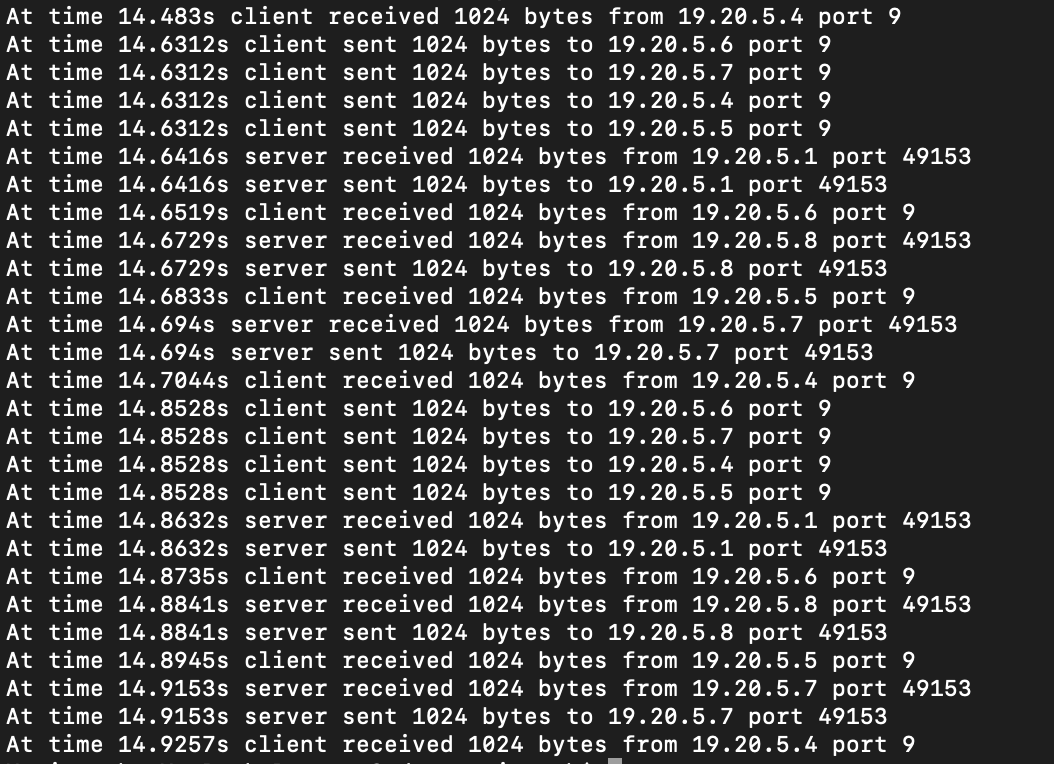
کدها با دستور زیر اجرا می‌شوند:



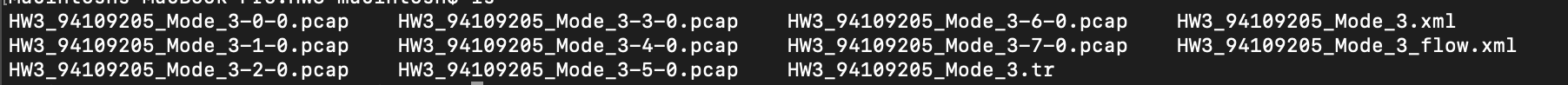
و می‌گوید برنامه با کدام نرخ ارسال کار کند؛ که عددی بین ۱ متناظر با نرخ 25.4Kbps و ۷ متناظر با نرخ 1024.5Kbps است. برای تکلیف نیز با هر ۷ مقدار برنامه اجرا شده و نتایج در فولدر خروجی آمده است.

درباره خروجی کد:

نمونه اعداد چاپ شده در terminal به صورت زیر است:



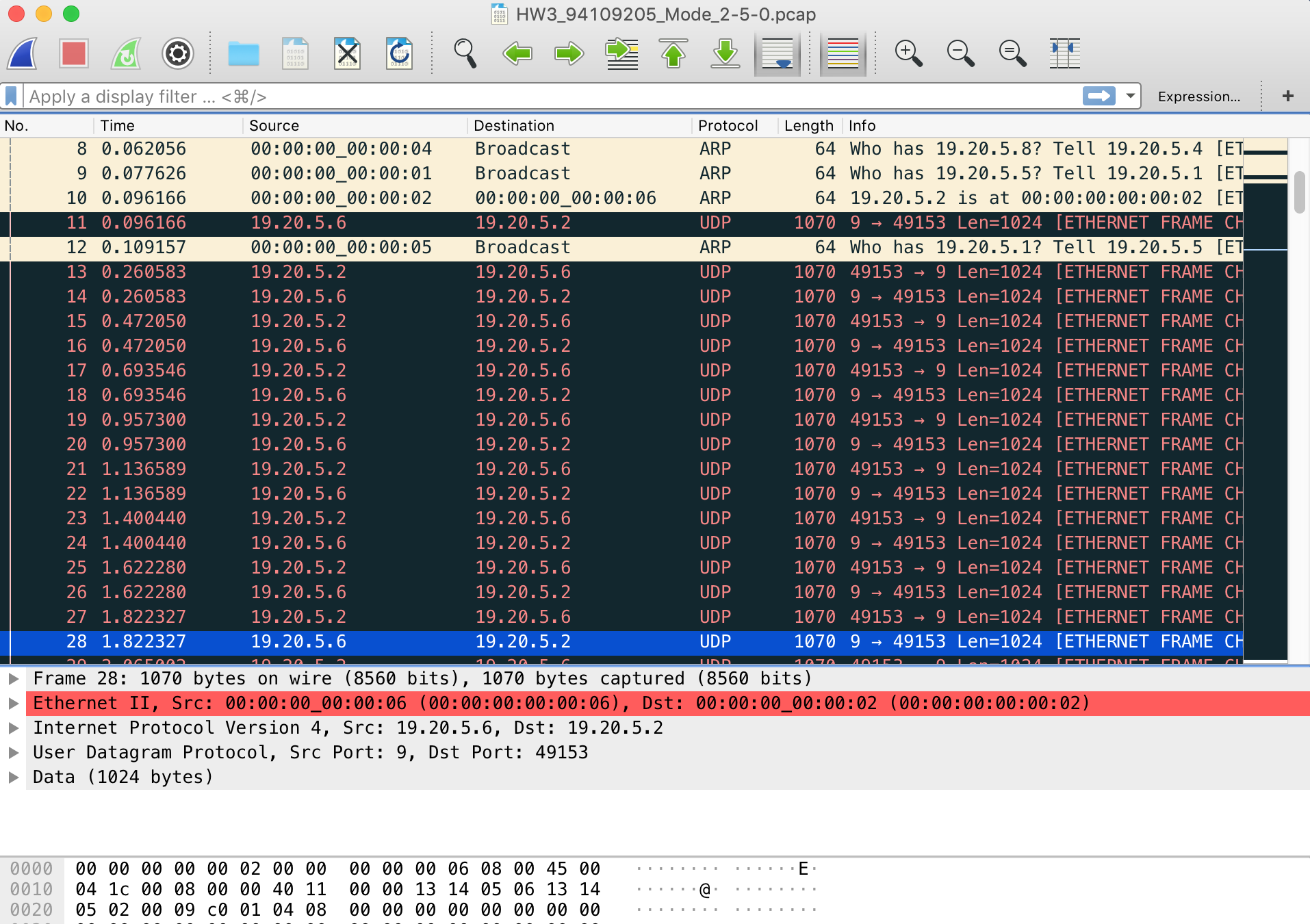
فایل‌های خروجی نمونه برای اجرای modeNumber=3 نیز برای مثال، به فرمت‌های زیر هستند:



که کاربری آنها از پسوندشان مشخص است.

نمونه خروجی Pcap:

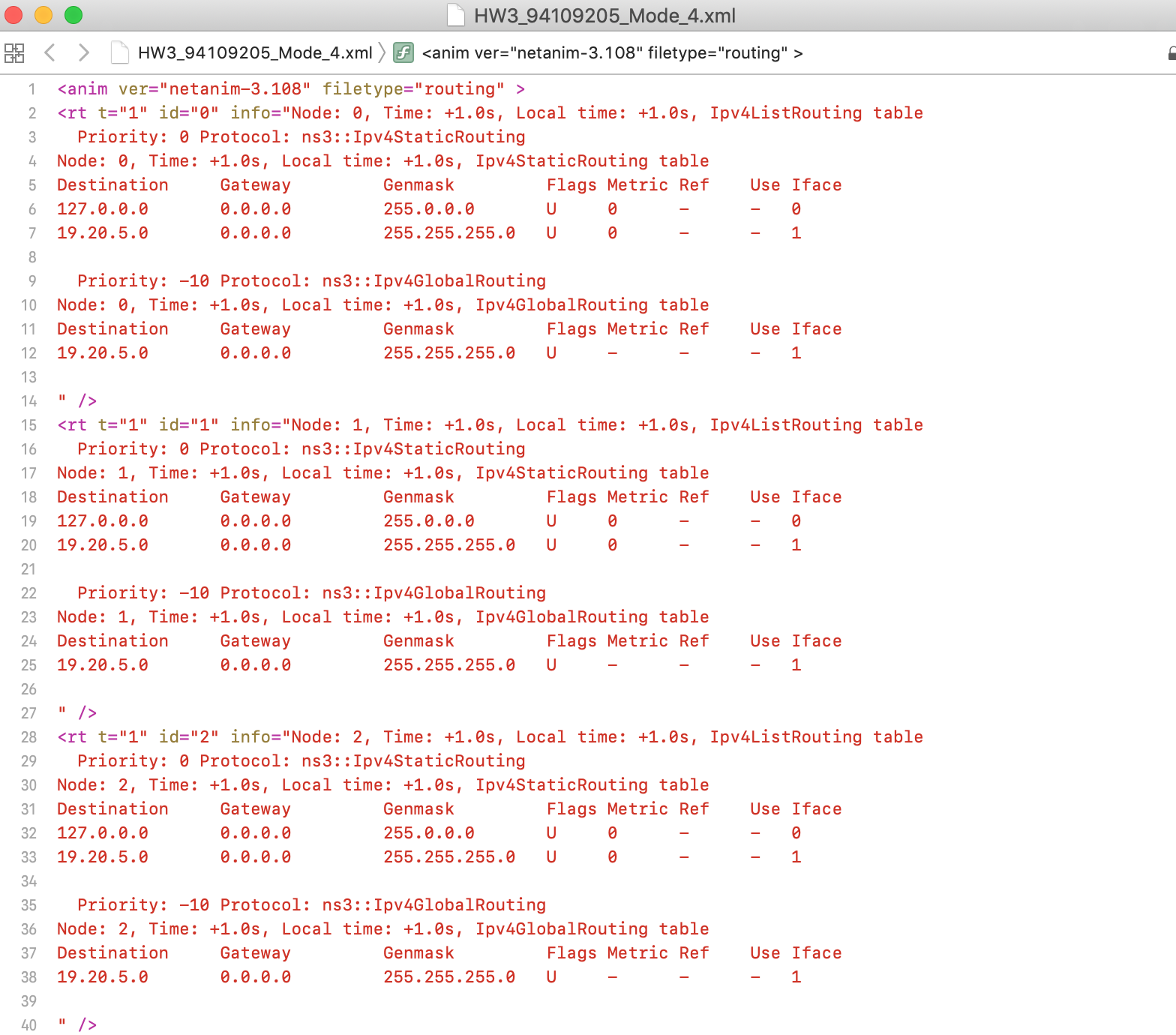
یک نمونه از این خروجی را با Wireshark باز کرده و نمایش می‌دهیم:



که ارتباط نقاط ۲ و ۶ را نشان می‌دهد (ملاحظه می‌شود در بسته‌ها information خاصی وجود ندارد) .

نمونه خروجی xml ( برای AnimationInterface):

ابتدای یک نمونه از این خروجی به فرمت زیر است:



نمونه خروجی xml ( برای FlowMonitor):

ابتدای یک نمونه از این خروجی به فرمت زیر است:



محاسبات رسم نمودار:

همانطور که اشاره شد، از یک روش ساده‌تر برای بدست آوردن نمودار استفاده می‌شود. به طور کلی سه راه وجود دارد:

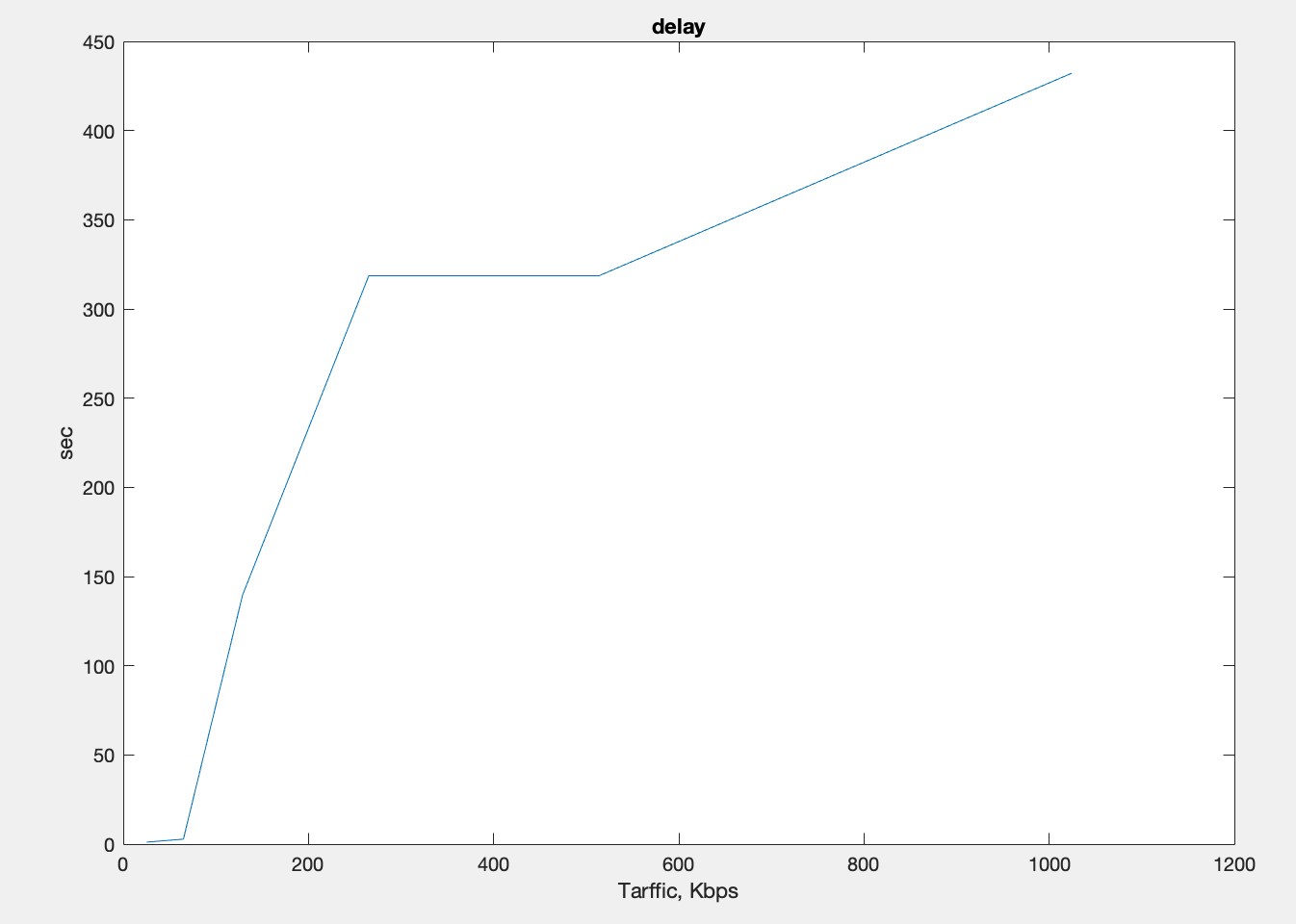
1. مشاهده بسته‌های ارسالی و زمان آنها در Wireshark و محاسبه دستی مقادیر خواسته شده
2. نوشتن یک کد متلب برای استخراج داده‌ها از فایل .tr و انجام محاسبات روی آن و بدست آوردن مقادیر خواسته شده
3. استفاده از مقادیر delaySum (برای تاخیر) و rxPackets (برای throughput ) که فایل .xml در FlowMinitor محاسبه‌کرده است.

که از روش سوم استفاده می‌کنیم و با نوشتن کدی ساده نمودار‌های مربوطه را رسم می‌کنیم که در ادامه می‌آیند. این کد در فولدر تمرین آمده است (با script به نام dn\_plotter و تابعی به نام stc2atr ).

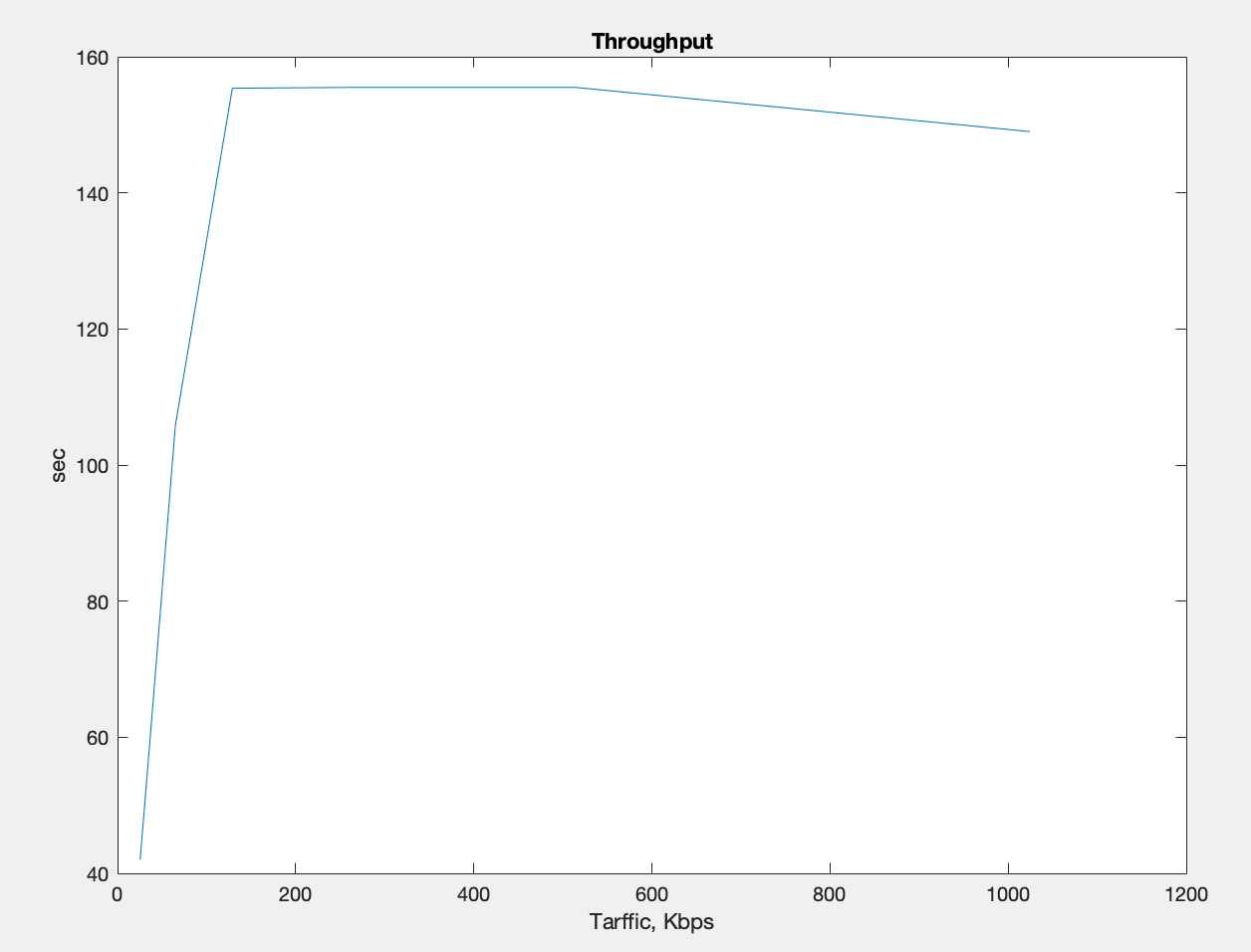
رسم نمودار و تحلیل آن:

نمودار ها به شکل زیر اند:

**تاخیر**



**Throughput**

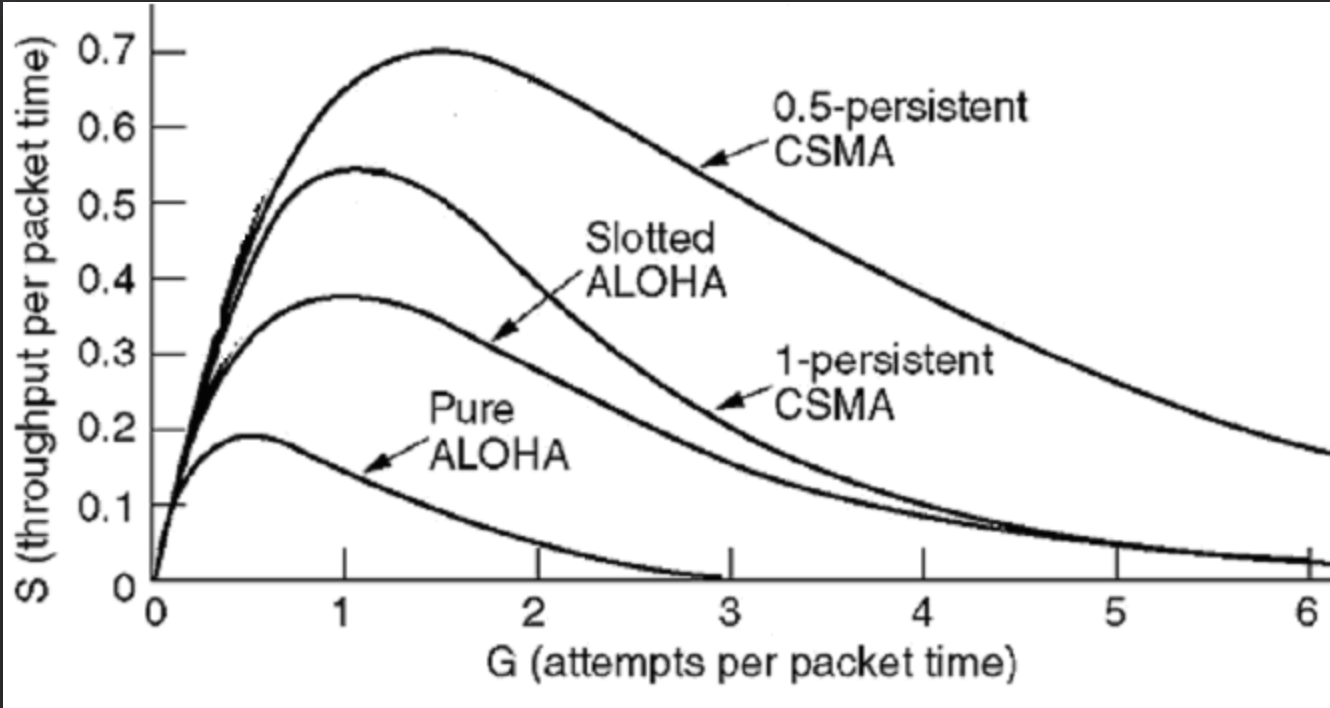


درمورد تاخیر انتظار داریم با شلوغ شدن کانال بیشتر شود. رابطه تاخیر ALOHA بصورت زیر است:

و اگر بصورت خیلی حدودی بگوییم به CSMA شبیه است، مشخص است که با افزایش بار شبکه فرم افزایشی دارد. در اینجا البته هم داده کم است و هم تعداد نقاط قابل توجه نیست پس رابطه ما دقیقا بصورت نمایی ظاهر نمی‌شود؛ اما فرم صعودی خود را دارد.

در مورد throughput نیز انتظار داریم در ابتدا افزایش و سپس کاهش یابد زیرا اگر شباهت آن با ALOHA را در نظر بگیریم، برای آن داریم:

و در واقع CSMA یک 1-persistent است:



که به دلیل محدودیت‌های گفته شده، نمودار خیلی دقیق نیست اما به خوبی رفتار بالا را نشان می‌دهد، و بهترین عملکر و بازدهی آن در حدود مقدار 128Kbps است.

\*\*\*\*\*\*\*\* پایان گزارش \*\*\*\*\*\*\*\*