به نام خدا

گزارش پروژه دوم درس شبکه های کامپیوتری

سارا رضائی منش ۸۱۰۱۹۸۵۷۶

برنا توسلی ۸۱۰۱۹۸۳۷۴

عنوان پروژه

آشنایی با ابزار شبیه سازی NS2 و بررسی و تحلیل برخی پارامترهای شبکه در هنگام استفاده از یک شبکه Wireless

ساختار پروژه

ساختار درختی فایل ها به صورت زیر می‌باشد:

```
.
├── tcp-exp.tcl
└── utility
    ├── charter.py
    ├── charterThroughputBySize.py
    └── parser.py

1 directory, 4 files
```

نحوه اجرا

برای اجرای برنامه شبیه ساز از دستور زیر استفاده کنید:

```
ns tcp-exp.tcl
```

با اجرای این برنامه فایل های .nam و فایل تولید پایتون تولید کننده throughput، average end-to-end delay و packet transfer ratio اجرا می‌شوند. نتایج اجرای برنامه پایتون، علاوه بر چاپ شدن در ترمینال، به نتایج قبلی موجود در فایل result.txt اضافه می‌شوند تا در مراحل بعد برای کشیدن نمودار ها مورد استفاده قرار می‌گیرند.

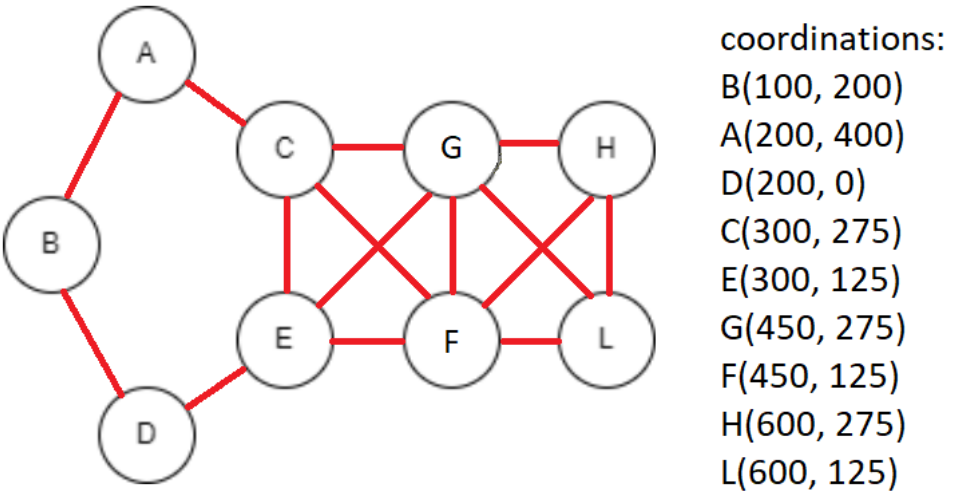
پیشفرض ها:

در شبکه شبیه سازی شده در این پروژه از پروتکل TCP و سیستم ارسال RTS/CTS و ACK استفاده شده است.

نحوه قرار گیری نود ها و نحوه اتصال آنها در بخش بعد آمده است.

تعاریف و ترمینولوژی

توپولوژی: نحوه قرارگیری اجزا شبکه (لایه‌های فیزیکی و مسیر داده) در کنار هم در فضا است. فاصله از پیش تعیین شده بین هر دو node وایرلس در ns برابر با 250 متر است. در ادامه شماتیک همسایگی و فاصله بین node ها آورده شده است.



سیستم ACK و RTS-CTS: برخلاف UDP، در پروتوکل TCP گیرنده با فرستادن packet های acknowledgement رسیدن یک packet را تایید می‌کند. از مکانیزم RTS-CTS می‌توان برای کم کردن frame collision ها (که به خاطر hidden node ها بوجود می‌آیند)، استفاده می‌شود.

منحنی Throughput: منظور از Throughput، نرخ داده دریافت شده در بازه زمانی می‌باشد. منحنی Throughput رسم شده بر حسب اندازه بسته‌های ارسالی یا همان packet size می‌باشد. برحسب تعریف ارائه شده، فرمول محاسبه Throughput به شکل زیر می‌باشد.

$$Throughput = \frac{Size_{recieved}}{Time_{simulation}}$$

اندازه Packet: در این پروژه نمودار Throughput را به ازای سایزهای مختلف بسته‌های ارسالی (از 10 تا 1000 بایت) رسم کرده‌ایم.

Packet Transfer Ratio: نسبت تعداد packetها دریافت شده به فرستاده شده، می‌باشد.

$$Packet\ Transfer\ Ratio = \frac{Packets\ received}{Packets\ sent}$$

Average End-to-End delay: میانگین زمانی که طول می‌کشد یه packet دریافت شود.

$$Avg_{end-to-end}delay = \frac{\sum receive\ time}{packets\ received}$$

پهنای باند: در این پروژه نمودارهای مختلف به ازای پهنای باند 1.5، 55 و 155 مگابیت بر ثانیه‌ای رسم شده‌اند.

نرخ خطا: نرخ خطا یا همان Packet Error Rate کارایی یک ترمینال گیرنده را می‌سنجد. با افزایش این نرخ، تعداد packet های drop شده نیز افزایش می‌یابد. در این پروژه، شبیه سازی و نمودارهای مختلف بر اساس 10 نرخ خطا متفاوت در بازه 0.000001 تا 0.00001 کشیده شده است.

توضیحات فایل ها

فایل tcp-exp.tcl

در این فایل فرآیند ساخت توپولوژی شبکه و شبیه سازی آن انجام می‌شود.

در ابتدای فایل یکسری متغیر ها که در ادامه برای تعریف شبکه از آنها استفاده می‌شود را تعریف می‌کنیم.

```
set opt(chan)           Channel/WirelessChannel  ;
set opt(prop)           Propagation/TwoRayGround ;
set opt(netif)          Phy/WirelessPhy          ;
set opt(mac)            Mac/802_11               ;
set opt(ifq)            Queue/DropTail/PriQueue  ;
set opt(ll)             LL                       ;
set opt(ant)            Antenna/OmniAntenna      ;
set opt(ifqlen)         50                       ;
set opt(bottomrow)      9                        ;
set opt(adhocRouting)   AODV                     ;
set opt(x)              700                      ;
set opt(y)              700                      ;
set opt(finish)         100                      ;
```

در متغیر های بالا bottomrow نشان دهنده تعداد node های شبکه و finish نشاندهنده زمان پایان شبیه سازی می‌باشد.

در مرحله بعد خصوصیاتی مانند bandwidth و RTSThreshold را تعریف می‌کنیم.مقدار RTSThreshold را به این دلیل تعیین می‌کنیم که در پروتکل 802_11 در صورتی که این پارامتر از یک مقدار مشخصی بیشتر باشد، از RTS/CTS استفاده می‌شود.

```
set bandwidth 155Mb
$opt(mac) set RTSThreshold_ 5000;
Mac/802_11 set dataRate_  $bandwidth;
```

در پایان یک شبیه ساز جدید ساخته می‌شود.

```
set ns [new Simulator]
```

خط های بعد برای تولید فایل های tr و nam هستند.

```
set name [lindex [split [info script] "."] 0]
$ns use-newtrace
set tracefd  [open $name.tr w]
set namtrace [open $name.nam w]
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $opt(x) $opt(y)
```

در سه خط بعد ابتدا یک توپوگرافی جدید ایجاد می‌کنیم و برای آن طول و عرض تعریف می‌کنیم. مقدار x و y در آرایه opt در ابتدای برنامه مشخص شده است. در پایان یک آبجکت god ساخته می‌شود که برای نگه داشتن مکان نود ها در توپولوژی مورد استفاده قرار می‌گیرد.(این آبجکت در شبکه هایی که نود های متحرک دارند کارایی بیشتری دارد.)

```
set topo [new Topography]
$topo load_flatgrid $opt(x) $opt(y)
create-god [expr $opt(bottomrow)]
```

در خطوط بعد، خواص استانداردی را برای نودها تعریف می‌کنیم که در ساخت آنها توسط شبیه ساز تاثیرگذار است. در این خصوصیات اضافه شده IncomingErrProc و OutgoingErrProc برای تعیین نرخ خطای ارسالی نودها مورد استفاده قرار می‌گیرد که در تابع UniformErr مشخص می‌شود.

در این تابع ابتدا یک ErrorModel درست می‌کنیم و ریت و واحد خطا را مشخص می‌کنیم.(مثلا در حالت زیر واحد خطا packet است یعنی نرخ خطا همان PER است و مشخص می‌کند که چه مقدار از پکت ها drop شوند.)

```
proc UniformErr {} {
   set err [new ErrorModel]
   $err set rate_ 0.01
   $err unit packet
   return $err
}

$ns node-config -adhocRouting $opt(adhocRouting) \
   -llType $opt(ll) \
   -macType $opt(mac) \
   -ifqType $opt(ifq) \
   -ifqLen $opt(ifqlen) \
   -antType $opt(ant) \
```

```
   -propType $opt(prop) \
   -phyType $opt(netif) \
   -channel $chan1 \
   -topoInstance $topo \
   -IncomingErrProc UniformErr \
   -OutgoingErrProc UniformErr \
   -wiredRouting OFF \
   -agentTrace ON \
   -routerTrace ON \
   -macTrace OFF
```

سپس نود های شبکه را تعریف می‌کنیم.

```
set A [$ns node]
set D [$ns node]
set B [$ns node]
set C [$ns node]
set E [$ns node]
set F [$ns node]
set G [$ns node]
set H [$ns node]
set L [$ns node]
```

و سپس موقعیت آن ها را در توپولوژی مشخص می‌کنیم. مثلا برای نود A داریم:

```
$A set X_ 200.0
$A set Y_ 400.0
$A set Z_ 0.0
```

برای اینکه نود ها دریافت کننده و فرستنده را مشخص کنیم لازم است به آنها agent متصل کنیم. به این صورت که به نود های فرستنده agent های tcp و به نود های دریافت کننده agent های sink متصل می‌شود. Agent های sink وظیفه فرستان پیام ACK به agent های tcp دارند و هر ایجنت سینک فقط با یک ایجنت تی سی پی می‌تواند در ارتباط باشد. پس برای اینکه هر دو نود فرستنده بتوانند به هر دو نود گیرنده داده ارسال کنند باید به هر فرستنده دو ایجنت تی سی پی و به هر گیرنده دو ایجنت سینک متصل باشند. همچنین به هر ایجنت تی سی پی یک traffic generator هم متصل وظیفه آن تولید ترافیک ارسالی برای ایجنت تی سی پی می‌باشد. برای agent های tcp از traffic generator های ftp استفاده می‌شود. به عنوان مثال برای اتصال یک tcp agent به همراه ftp traffic generator به نود A داریم:

```
set tcp1 [new Agent/TCP]
$ns attach-agent $A $tcp1
$tcp1 set packetSize_ $packetSize
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
```

و برای اتصال یک sink agent به نود H داریم:

```
set sink1 [new Agent/TCPSink]
$ns attach-agent $H $sink1
```

در پایان باید هر tcp به یک sink متصل شود:

```
$ns connect $tcp1 $sink1
```

```
$ns connect $tcp11 $sink2
$ns connect $tcp2 $sink11
$ns connect $tcp22 $sink22
```

در خطوط بعد مشخص می‌کنیم که traffic generator ها در چه زمانی شروع به تولید ترافیک کنند و در چه این فعالیت را خاتمه دهند:

```
$ns at 0 "$ftp1 start"
$ns at 0 "$ftp2 start"
$ns at 0 "$ftp11 start"
$ns at 0 "$ftp22 start"
$ns at $opt(finish) "$ftp1 stop"
$ns at $opt(finish) "$ftp2 stop"
$ns at $opt(finish) "$ftp11 stop"
$ns at $opt(finish) "$ftp22 stop"
```

همچنین باید برای هر نود، شعاع آن را در صفحه مشخص کنیم. مثلا برای نود A داریم:

```
$ns initial_node_pos $A 20.0
```

خطوط بعدی رنگ نود ها را در شبیه سازی مشخص می‌کنند. در این پروژه نود ها فرستنده آبی و نود های گیرنده قرمز هستند.

```
$A color blue
$ns at 0.0 "$A color blue"
$D color blue
$ns at 0.0 "$D color blue"
$L color red
$ns at 0.0 "$L color red"
$H color red
$ns at 0.0 "$H color red"
```

در پایان زمان پایان شبیه سازی و تابعی که باید در انتهای شبیه سازی فراخوانی شود تعریف می‌شوند. در تابع finish نود ها برای شبیه سازی ها بعد ریست می‌شوند و فایل پایتون برای خروجی دادن مقادیر مطرح شده در صورت پروژه و فایل .nam برای نمایش تصویری شبیه سازی باز می‌شوند.

```
$ns at $opt(finish) "finish"
proc finish {} {
   global ns tracefd namtrace
   $ns flush-trace
   close $tracefd
   close $namtrace
   exec nam tcp-exp.nam &
   exit 0
}
```

فایل parser.py

از این فایل برای خواندن فایل تریس و چاپ مقادیر end-to-end delay، throughput و packet transfer ratio استفاده می‌شود.

در این برنامه ابتدا همه خط های فایل تریس در یک لیست خوانده می‌شوند و سپس هر خط از جاهایی که space دارد جدا می‌شود.

در فایل trace حرف s در ابتدای خط نشان دهنده send و حرف r نشان دهنده receive است. همچنین بعد از فلگ Ni شماره نودی می‌آید که عمل فرستادن، گرفتن، دراپ کردن و یا فوروارد کردن را انجام داده است. از آنجایی که برای محاسبه مقادیر بالا احتیاجی به داده های مربوط به نود های میانی نداریم، برای محاسبه throughput و تعداد پکت های ارسال شده تنها سایز پکت هایی را در نظر می‌گیریم که فرستنده آنها نودهای ۰ و ۱ بوده اند که در شبیه سازی ما همان نود های تولید کننده packet هستند. همچنین برای محاسبه تعداد پکت هایی که به مقصد رسیده اند نیز تنها receive هایی که توسط نود ها ۷ و ۸ که در شبیه سازی ما نود های دریافت کننده هستند را مدنظر قرار می‌دهیم. از آنجایی که پکت های ACK نیز بین نود ها در مسیر برعکس پکت های اصلی جا به جا می‌شوند، تنها پکت هایی که نوع آنها tcp است را در محاسبات لحاظ می‌کنیم. نوع پکت در فایل تریس بعد از فلگ It مشخص شده است.

```
for i, line in enumerate(lines):
   if line[34] == "ack":
       continue
   if line[0] == "s" and (line[8] == "0" or line[8] == "1"):
       sent += 1
       packet_start_time[line[40]] = float(line[2])

   if line[0] == "r" and (line[8] == "7" or line[8] == "8") and "tcp" in line:
       fsize += float(line[36])
       received += 1
       sum_e2e_delay += float(line[2]) - packet_start_time[line[40]]
```

در پایان از فرمول های زیر برای محاسبه سه مقدار ذکر شده استفاده می‌شود:

```
throughput = (fsize * 8) / sim_time
packet_transfer_ratio = received / sent
avg_e2e_delay = (sum_e2e_delay) / received
```

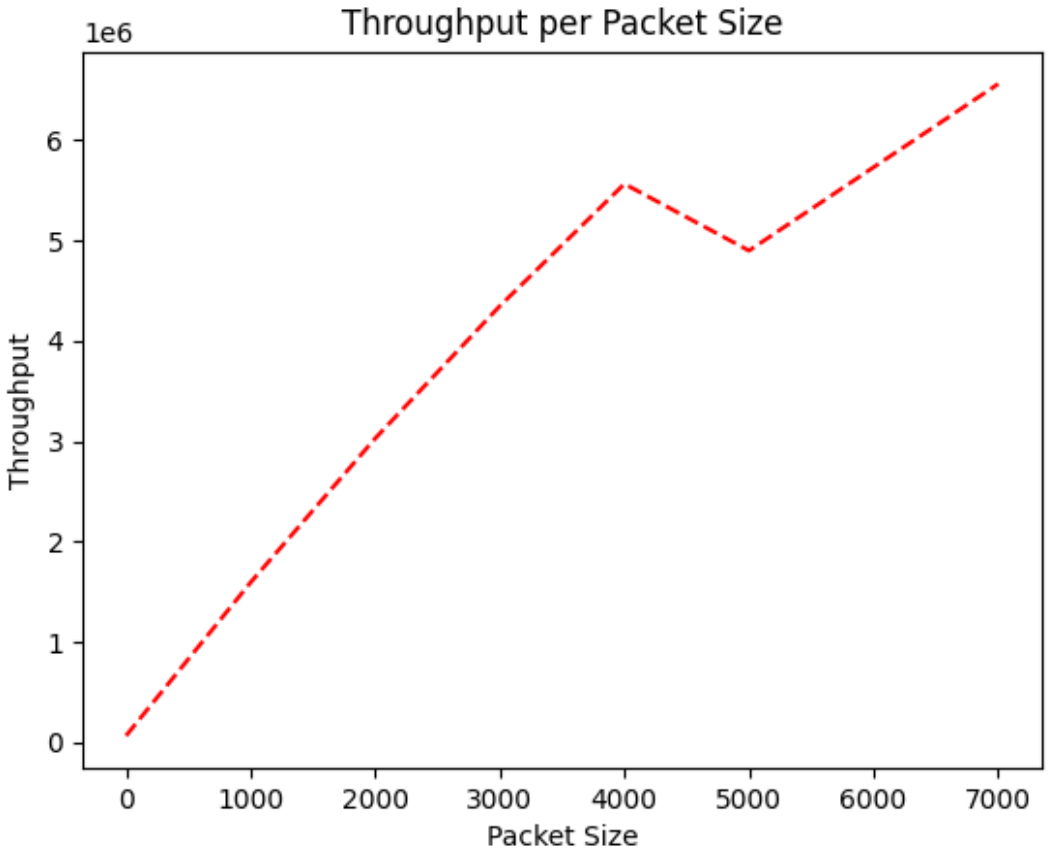
در انتها مقادیر محاسبه شده به فایل result.txt اضافه می‌شوند.

```
f = open("result.txt", "a")
f.write(str(throughput) + ' ' + str(packet_transfer_ratio) + ' ' + str(avg_e2e_delay) +
'\n')
f.close()
```

فایل charterThroughputBySize.py

از این فایل برای رسم نمودار throughput بر اساس packet size های مختلف استفاده می‌شود.

این برنامه مقدار throughput را بر اساس ۸ پکت سایز مختلف رسم میکند.



مشاهده می‌شود که تقریبا throughput با packet size رابطه مستقیم دارد. یعنی با افزایش packet size، میزان throughput هم افزایش پیدا می‌کند. البته اگر اندازه پکت از پهنای باند شبکه بیشتر شود، throughput به شدت کاهش پیدا می‌کند. چراکه شبکه نمی‌تواند هیچ کدام از پکت ها را عبور دهد و فقط overhead می‌فرستد. اما در صورتی که کوچکتر باشند، افزایش سایز پکت باعث استفاده مفیدتر از پهنای باند و ارسال بیشتر داده می‌شود.

ضعف استفاده از packet size های خیلی بزرگ این است که اگر یک پکت drop شود، بخش زیادی از داده از دست می‌رود. همچنین در برخی از پروتکل ها مقدار overhead متناسب با سایز پکت افزایش پیدا می‌کند و در نتیجه مجبور می‌شویم حجم زیادی از داده ها به همراه داده اصلی به عنوان overhead بفرستیم.

فایل charter.py(بخش امتیازی)

این فایل برای رسم نمودارهای end-to-end delay، throughput و packet transfer ratio بر حسب تغییرات error rate و bandwidth مورد استفاده قرار می‌گیرد.

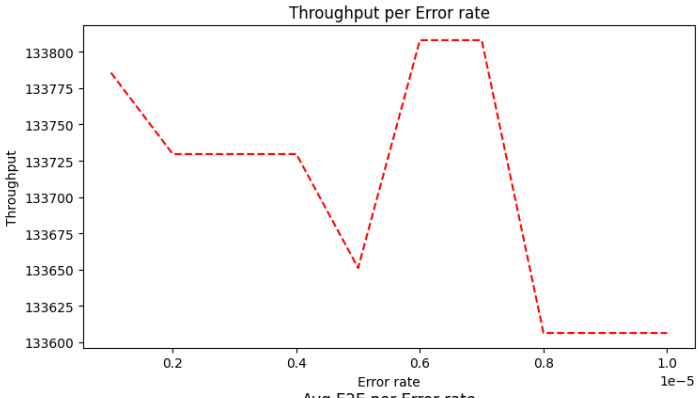
این برنامه ابتدا ده مقدار وارد شده به ازای هر یک از سه معیارهای بالا را از فایل result.txt می‌خواند. سپس نمودار آنها را بر اساس ده مقدار متفاوت نرخ خطا رسم می‌کند.

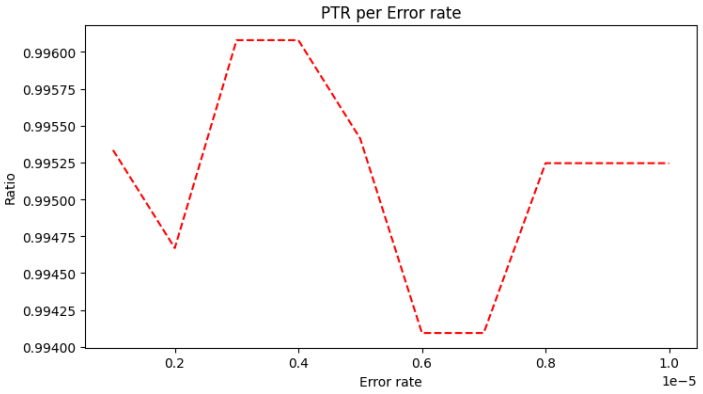
نرخ خطاهای مختلف:

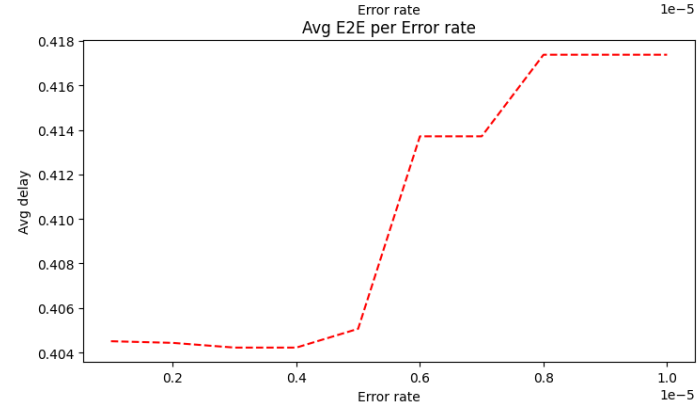
0.000001 0.000002 0.000003 0.000004 0.000005 0.000006 0.000007 0.000008 0.000009 0.00001

نتایج ران کردن این فایل برای سه پهنای باند مختلف به صورت زیر می‌باشد:

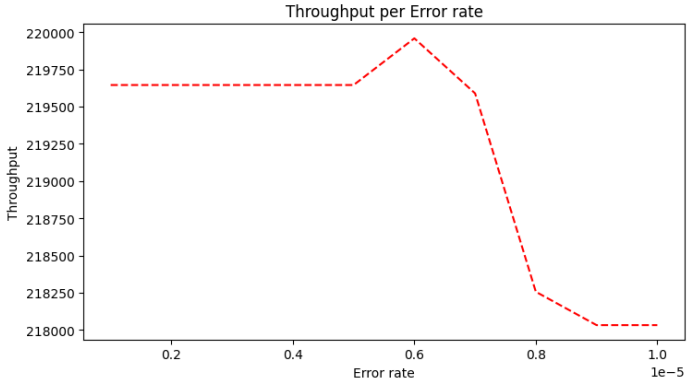
برای bandwidth = 1.5Mb:

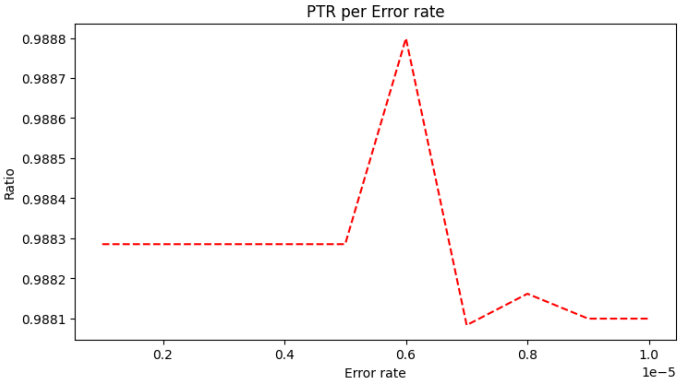


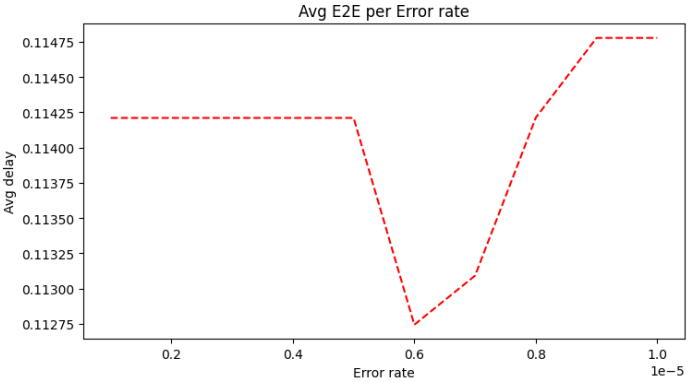




برای bandwidth = 55Mb:







برای bandwidth = 155Mb: