# گزارش پروژه دوم درس شبکه‌های کامپیوتری

**دکتر یزدانی**

**بهار ۱۴۰۱**

پرنیان فاضل ۸۱۰۱۹۸۵۱۶ - پریا خوش‌تاب ۸۱۰۱۹۸۳۸۷

---

## اجرای شبیه‌ساز به زبان TCL در محیط NS2

به طور کلی برای ایجاد یک شبیه سازی ns2، باید موارد زیر را انجام دهیم:

1. توپولوژی شبکه را شامل تمام گره‌ها، لینک‌ها و قوانین صف روتر را تعریف می‌کنیم.
2. برخی از اتصالات TCP یا UDP(به دلیل سیگنال ACK که در صورت پروژه گفته شده ما از TCP استفاده کردیم) به نام Agents ایجاد می‌کنیم و آنها را به گره‌ها متصل می‌کنیم.
3. برخی از applicationها را ایجاد می‌کنیم و آنها را به agentها attach می‌کنیم.
4. simulation را شروع می‌کنیم.

حال به جزئیات بیشتر می‌پردازیم:

ابتدا یک فایل WirelessTCP.tcl ایجاد می‌کنیم و سپس مراحل زیر را طی می‌کنیم:

bandwidth و error rate و packet size هر شبیه‌سازی را به عنوان ورودی از ترمینال دریافت می‌کنیم و در متغیرهای مربوطه ذخیره می‌کنیم.

```
Mac/802_11 set bandwidth [lindex $argv 0]
set error_rate [lindex $argv 1]
set packet_size [lindex $argv 2]
```

برای نگه داشتن پارامترهای configuration، یک Tcl object به نام opt می‌سازیم و این پارامترها را به عنوان attributeهای opt قرار می‌دهیم.

```
# Define options
set opt(chan)           Channel/WirelessChannel  ;# channel type
set opt(prop)           Propagation/TwoRayGround ;# radio-propagation model
set opt(netif)          Phy/WirelessPhy          ;# network interface type
set opt(mac)            Mac/802_11               ;# MAC type
set opt(ifq)            Queue/DropTail/PriQueue  ;# interface queue type
set opt(ll)             LL                       ;# link layer type
set opt(ant)            Antenna/OmniAntenna      ;# antenna model
set opt(ifqlen)         50                       ;# max packet in ifq
set opt(nn)             9                        ;# number of mobilenodes
set opt(rp)             AODV                     ;# routig protocol(Ad hoc On-Demand Distance Vector Routing)
set opt(x)              700                      ;# x coordinate of topology
set opt(y)              200                      ;# y coordinate of topology
set opt(finish)         100                      ;# time to stop simulation
```

یک object از کلاس Simulator می‌سازیم که object اصلی شبیه‌سازی در این پروژه است.

```
# Create the simulator object
set ns [new Simulator]
```

traceهای تولید شده برای نمایش انیمیشنی توپولوژی nam و جزئیات مربوط به حرکت کردن بسته‌ها را در فایل‌های tr. و nam. ذخیره می‌کنیم و در ادامه از این فایل tr.برای تجزیه و تحلیل logهای شبکه استفاده خواهیم کرد.

```
# Set up tracing
set tracefd  [open "out_trace.tr" w]
$ns trace-all $tracefd
$ns eventtrace-all

set namtrace [open "trace.nam" w]
$ns namtrace-all-wireless $namtrace $opt(x) $opt(y)
```

یک procedure به نام finish ایجاد می‌کنیم که در انتهای شبیه‌سازی اجرا می‌شود و nam را اجرا می‌کند و در نهایت فایل‌های مربوط به trace بسته می‌شوند.

```
# Define the finish procedure
proc finish {} {
    global ns namtrace tracefd opt
    $ns flush-trace
    # exec nam trace.nam &
    close $namtrace
    close $tracefd
    exit 0
}
```

object و layout مربوط به توپوگرافی را ایجاد می‌کنیم.

```
45  # Create  and define the topography object and layout
46  set topo [new Topography]
47  $topo load_flatgrid $opt(x) $opt(y)
```

نمونه ای از General Operations Director ایجاد می‌کنیم که نودها و قابلیت دسترسی نود به نود را ردیابی می‌کند. در واقع این پارامتر تعداد کل نودها در شبیه سازی است که در اینجا آن را به عدد ۹ ست می‌کنیم.

```
50  # Create an instance of General Operations Director, which keeps track of nodes and
51  # node-to-node reachability. The parameter is the total number of nodes in the simulation.
52  create-god $opt(nn)
```

مرحله بعدی فراخوانی node-config است که بسیاری از پارامترهای opt را به ns می‌دهد و بر ایجاد نود در آینده تاثیر می‌گذارد.

```
55  # General node configuration
56  set chan1 [new $opt(chan)]
57
58  $ns node-config -adhocRouting $opt(rp) \
59                  -llType $opt(ll) \
60                  -macType $opt(mac) \
61                  -ifqType $opt(ifq) \
62                  -ifqLen $opt(ifqlen) \
63                  -antType $opt(ant) \
64                  -propType $opt(prop) \
65                  -phyType $opt(netif) \
66                  -IncomingErrProc UniformErr \
67                  -OutgoingErrProc UniformErr \
68                  -channelType Channel/WirelessChannel \
69                  -topoInstance $topo \
70                  -agentTrace ON \
71                  -routerTrace OFF \
72                  -macTrace ON \
73                  -movementTrace OFF
```

برای بررسی نرخ خطای ارسال با توزیع یکنواخت از تابع زیر استفاده می‌کنیم:

```
76  proc UniformErr {} {
77      global error_rate
78      set err [new ErrorModel]
79      $err unit packet
80      $err set rate_ $error_rate
81      $err ranvar [new RandomVariable/Uniform]
82      return $err
83  }
```

آرایه‌ای از نودها به نام node ایجاد می‌کنیم و سپس مکان این نودها را مطابق توپولوژی داده شده قرار می‌دهیم.

```
# Create nodes as a node array $node()
for {set i 0} {$i < $opt(nn)} {incr i} {
    set node($i) [$ns node]
}


# Set node positions

#A
$node(0) set X_ 200.0
$node(0) set Y_ 400.0
$node(0) set Z_ 0.0

#B
$node(1) set X_ 100.0
$node(1) set Y_ 200.0
$node(1) set Z_ 0.0

#D
$node(2) set X_ 200.0
$node(2) set Y_ 0.0
$node(2) set Z_ 0.0

#C
$node(3) set X_ 400.0
$node(3) set Y_ 300.0
$node(3) set Z_ 0.0

#E
$node(4) set X_ 400.0
$node(4) set Y_ 100.0
$node(4) set Z_ 0.0

#G
$node(5) set X_ 600.0
$node(5) set Y_ 300.0
$node(5) set Z_ 0.0

#F
$node(6) set X_ 600.0
$node(6) set Y_ 100.0
$node(6) set Z_ 0.0

#H
$node(7) set X_ 800.0
$node(7) set Y_ 300.0
$node(7) set Z_ 0.0

#L
$node(8) set X_ 800.0
$node(8) set Y_ 100.0
$node(8) set Z_ 0.0
```

کانکشن‌هایی از نوع TCP با ترافیک Ftp ایجاد می‌کنیم و از این طریق نودهای فرستنده را به نودهای گیرنده متصل می‌کنیم. در واقع به نودهای فرستنده یک TCP Agent و به نودهای گیرنده

یک TCPSink Agent اختصاص می‌دهیم. همچنین فرض می‌کنیم هر دو ترافیک Ftp مختص به نودهای فرستنده شامل مشخصات interval = 0.01 و rate =200kb هستند.

```
# setup TCP connection, using FTP traffic
set sink1 [new Agent/TCPSink]
set sink2 [new Agent/TCPSink]
set src1 [new Agent/TCP]
set src2 [new Agent/TCP]

$ns attach-agent $node(7) $sink1
$ns attach-agent $node(8) $sink2
$ns attach-agent $node(0) $src1
$ns attach-agent $node(2) $src2

$ns connect $src1 $sink1
# $ns connect $src1 $sink2 //for 4 connections
# $ns connect $src2 $sink1
$ns connect $src2 $sink2

set ftp1 [new Application/FTP]
set ftp2 [new Application/FTP]


$ftp1 attach-agent $src1
$ftp1 set rate_ 200kb
$ftp1 set interval_ 0.01
$sink1 set packetSize_ $packet_size

$ftp2 attach-agent $src2
$ftp2 set rate_ 200kb
$ftp2 set interval_ 0.01
$sink2 set packetSize_ $packet_size
```

هر دو ترافیک Ftp را در زمان ۰ شروع می‌کنیم و در زمان ۱۰۰ خاتمه می‌دهیم.

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

حال ابتدا مکان اولیه هر نود را با استفاده از تابع initial_node_pos ست می‌کنیم و بعد در زمان ۱۰۰ تابع finish را برای اتمام شبیه‌سازی فراخوانی می‌کنیم.

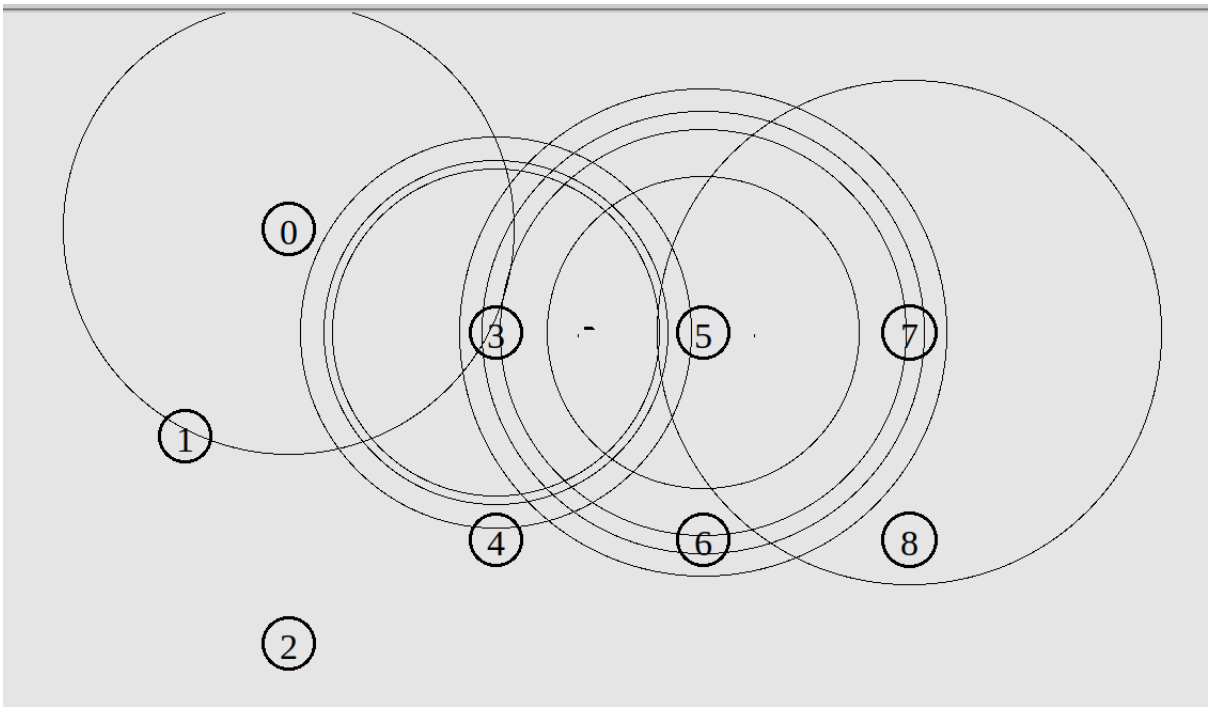
```
for {set i 0} {$i < $opt(nn)} {incr i} {
    $ns initial_node_pos $node($i) 50
    $ns at 100.0 "$node($i) reset";
}

$ns at $opt(finish) "finish"
```

در نهایت با استفاده از دستور run شبیه‌سازی را شروع می‌کنیم.

```
184    $ns run
```

یک نمونه از خروجی nam به صورت زیر می‌باشد:




## تجزیه و تحلیل فایل‌های خروجی مرحله شبیه‌سازی

یک فایل نوتبوک پایتون به نام evaluation.ipynb ایجاد می‌کنیم. این فایل شامل توابع زیر می‌باشد که به شرح آن‌ها می‌پردازیم:

- parse_file: این تابع فایل tr. را به عنوان ورودی دریافت می‌کند و سپس این فایل را به خط‌های آن تجزیه می‌کند.
- get_num_of_recv_bytes: این تابع تعداد بایت‌های دریافت شده توسط نودهای مقصد را محاسبه می‌کند. به این صورت که روی فایل تجزیه شده پیمایش می‌کنیم و در صورتی که

داشته باشیم ACTION=r و LAYER=MAC و TYPE=tcp و WHERE=(H or L) سایز بسته در لایه فعلی را به تعداد کل بایت‌های دریافت شده اضافه می‌کنیم.

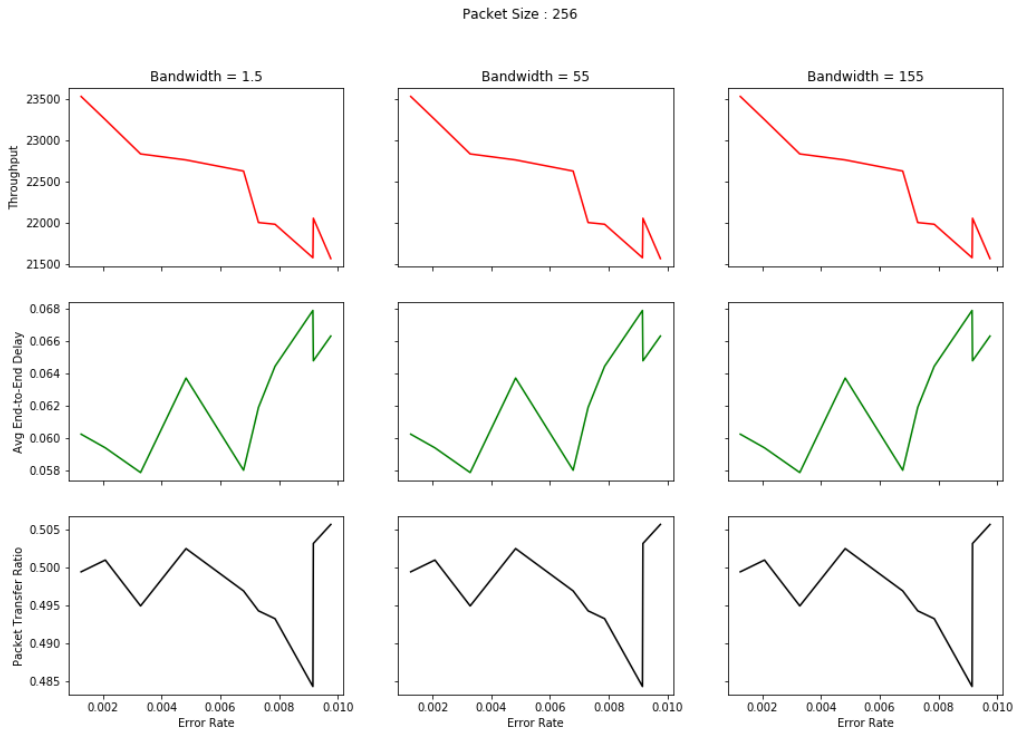
- get_throughput: این تابع مقدار throughput را از فرمول $Throughput = \frac{TransferSize}{TransferTime}$ محاسبه میکند و ریترن می‌کند. TransferSize در واقع همان تعداد بیت‌های دریافت‌شده توسط نودهای مقصد می‌باشد که توسط تابع recv_bits محاسبه شده است و TransferTime همان مدت زمان شبیه‌سازی داده شده در صورت پروژه یعنی ۱۰۰ ثانیه می‌باشد. دقت شود که واحد throughput بازگشتی $Bytes/second$ می‌باشد.
- get_num_of_sent_packets: این تابع تعداد بسته‌های فرستاده شده توسط نودهای مبدا را محاسبه می‌کند. به این صورت که روی فایل تجزیه شده پیمایش می‌کنیم و در صورتی که داشته باشیم ACTION=s و TYPE=tcp و WHERE=(A or D) تعداد کل بسته‌های فرستاده شده را یک واحد اضافه می‌کنیم.
- get_num_of_recv_packets: این تابع تعداد بسته‌های دریافت شده توسط نودهای مقصد را محاسبه می‌کند. به این صورت که روی فایل تجزیه شده پیمایش می‌کنیم و در صورتی که داشته باشیم ACTION=r و LAYER=MAC و TYPE=tcp و WHERE=(H or L) تعداد کل بسته‌های دریافت شده را یک واحد اضافه می‌کنیم.
- get_packet_transfer_ratio: این تابع مقدار Packet Transfer Ratio را از تقسیم تعداد بسته های دریافت شده در مقصد به تعداد بسته های ارسال شده از مبدا محاسبه

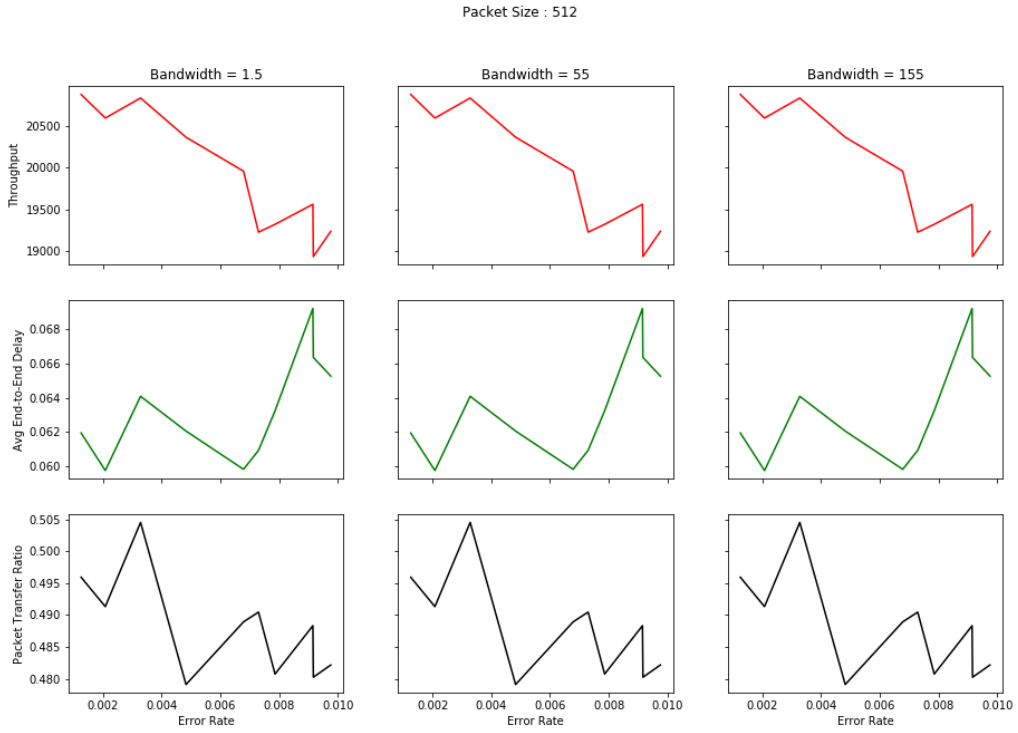
میکند و ریترن می‌کند. برای این منظور تنها کافی است get_num_of_recv_packets را بر get_num_of_sent_packets تقسیم کنیم.

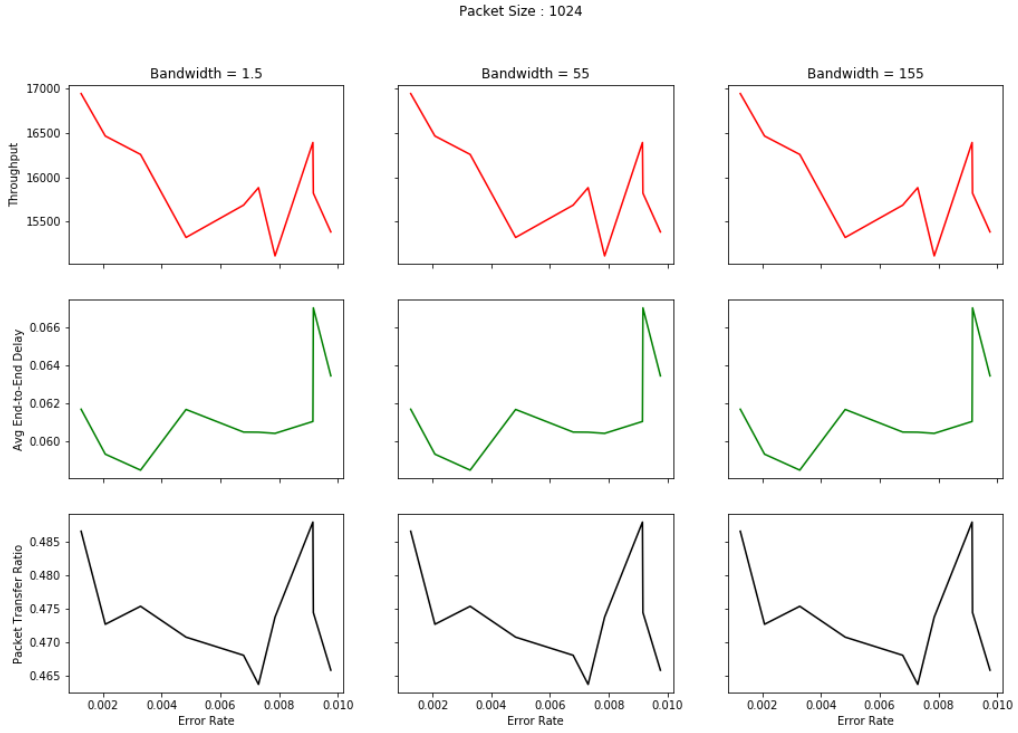
- get_send_time_dict: با استفاده از این تابع یک دیکشنری می‌سازیم که هر پکت را به زمان فرستادن آن توسط یکی از ۲ گره A یا D مپ می‌کند. از این تابع برای محاسبه تاخیر end to end استفاده می‌کنیم.
- get_end_to_end_delay_dict: این تابع یک دیکشنری به عنوان خروجی برمی‌گرداند که با استفاده از تابع get_send_time_dict، هر پکت را به مقدار تاخیر end to end آن مپ می‌کند. تأخیر انتها به انتها یا تأخیر یک طرفه به زمان صرف شده برای انتقال یک بسته در یک شبکه از مبدا به مقصد اشاره دارد. بنابراین زمان فرستادن پکت را با شرط‌های TYPE=tcp، ACTION=s و WHERE=(A or D) به دست آورده و با زمان رسیدن به مقصد با شرط‌های TYPE=tcp، ACTION=r و WHERE=(H or L) و LAYER=MAC به دست آورده و برای هر پکت از هم کم می‌کنیم.
- get_e2e: این تابع مقدار میانگین تاخیر end to end را محاسبه کرده و آن را برمیگرداند. برای محاسبه از دیکشنری که با تابع get_end_to_end_delay_dict ایجاد شد، مجموع تاخیرها را بر تعداد پکت‌های receive تقسیم کرده را مقدار میانگین را به دست آوریم.
- Run: در این تابع، شبیه‌سازی را به ازای bandwidth و error rate و packet sizeهای مختلف داده شده در صورت پروژه (مجموعا ۹۰ بار) اجرا می‌کنیم و به ازای هر بار اجرا

مقادیر Throughput و Packet Transfer Ratio و Average End-to-End Delay را در یک فایل به نام evaluation.txt ذخیره می‌کنیم.

- get_inputs: در این تابع فایل trace را خوانده و پارس می‌کنیم و اطلاعات تجزیه شده در یک لیست ذخیره می‌کنیم.
- draw_plot_by_packet: در این تابع با سایز پکت ثابت، خروجی‌ها را روی نمودار رسم می‌کنیم. خروجی‌ها برای ۳ مقدار ۲۵۶ و ۵۱۲ و ۱۰۲۴ برای سایز پکت:



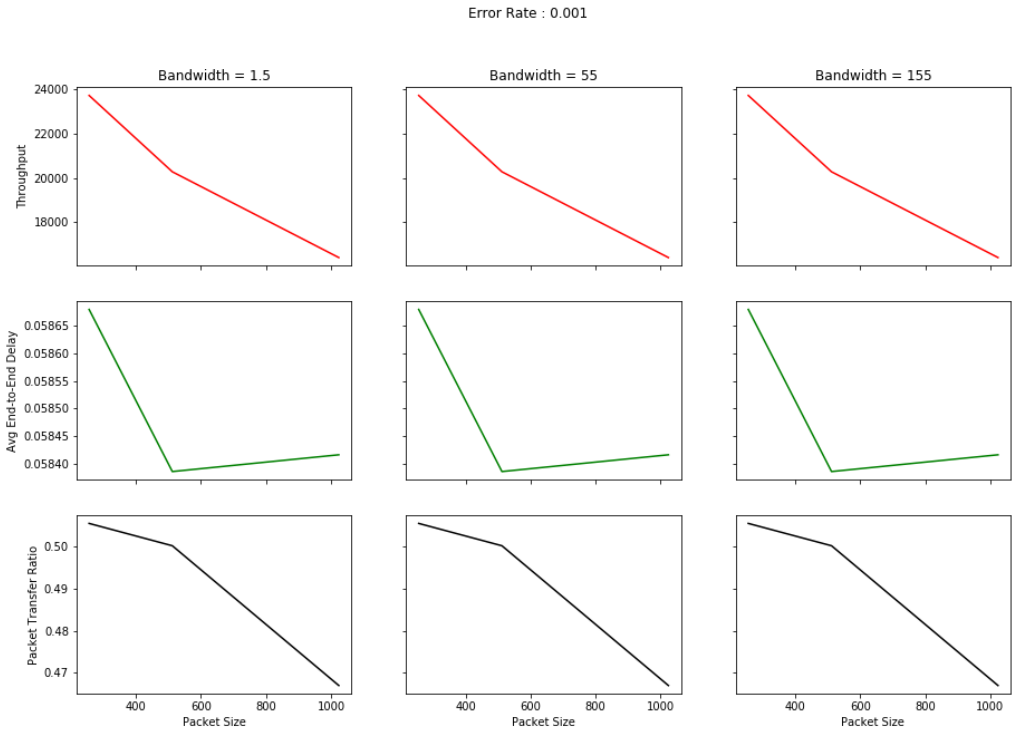



همانطور که در نمودارهای بالا دیده می‌شود به ازای سایز بسته ثابت، مقدار Throughput و

Packet Transfer Ratio و Average End-to-End Delay به ازای bandwidthهای مختلف یکسان میباشد. همچنین مقدار Average End-to-End Delay با افزایش نرخ ارور، افزایش میابد (درواقع چون نرخ ارور افزایش می‌یابد بنابراین پکت‌ها بیشتر در صف منتظر مانده و بنابراین تاخیر انتها به انتها افزایش یافته است) و Packet Transfer Ratio با افزایش سایز بسته ها کاهش میابد زیرا افزایش سایز بسته ها احتمال خرابی هم بیشتر می شود.

- draw_plot_by_error_rate: در این تابع با نرخ خطای ثابت، خروجی را روی نمودار رسم می‌کنیم. نمودار بر اساس نرخ خطای 0.001:



همانطور که در نمودارهای بالا دیده میشود به ازای نرخ ارور ثابت، مقدار Throughput و Packet Transfer Ratio و Average End-to-End Delay به ازای bandwidthهای مختلف با افزایش سایز بسته ها، کاهش میابد.

## منابع:


- https://www.cs.ucf.edu/~czou/CDA6530-12/NS2-tutorial.pdf
- http://intronetworks.cs.luc.edu/current/html/ns2.html
- https://www.nsnam.com/2010/11/how-to-interpret-ns2-tracefile-manually.html?view=sidebar
- Computer Networks: A Systems Approach 6th ed.