آزمایشگاه شبکه

آزمایش ۵: کنترل ازدحام، TCP و UDP

در این آزمایش، مقدمات لازم برای راه‌اندازی و پیکربندیِ نشست‌های ارتباطی UDP و TCP را فراهم می‌کنیم. در آزمایش‌های بعدی، مکانیزم کنترل ازدحام TCP را مورد بررسی قرار می‌دهیم.

الف) روال آزمایش

برنامه‌های مورد نیاز برای اجرای این آزمایش از فولدر lab5 در دسترس هستند. این فولدر شامل موارد زیر است:

a. فولدر lab5 حاوی اسکریپت‌های مورد نیاز برای ساخت توپولوژی‌های مورد نظر می‌باشد.

b. فولدرهای lab5/tcp و lab5/udp حاوی کلاینت‌ها و سِروِرهای TCP و UDP هستند که برای اندازه‌گیری goodput[1] هر جریان مورد استفاده قرار خواهند گرفت.

ماشین مجازی را راه‌اندازی کرده و در صورت لزوم، با ورود به دایرکتوری lab5، با اجرای دستورِ زیر، برنامه‌ها را به حالت «اجرایی» درآورید:

```
chmod +x tcp/tcpclient tcp/tcpserver udp/udpclient udp/udpserver
```

مکانیزم کنترل ازدحام در ماشین مجازی را بررسی کنید. روی یک ماشین لینوکس، می‌توانید با تایپ دستور زیر بررسی کنید که چه مکانیزمی مورد استفاده است:

```
cat /proc/sys/net/ipv4/tcp_congestion_control
```

در این آزمایش، ما TCP را ملزم می‌کنیم که از الگوریتم کنترل ازدحام reno استفاده کند. در صورتی که پیشاپیش، الگوریتم مورد نظر از نوع reno نباشد، شما می‌توانید با تایپ دستور زیر در پنجرهٔ ترمینال، آن را به reno تغییر دهید (البته تا reboot بعدی):

```
sudo bash -c 'echo reno >/proc/sys/net/ipv4/tcp_congestion_control'
```

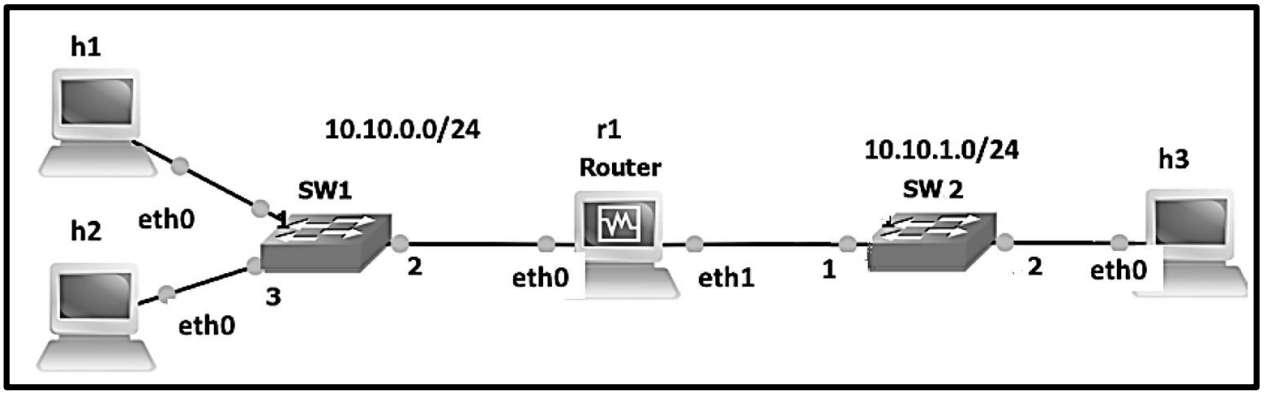
راهکار دیگر این است که مستقیماً محتوای فایل فوق را به reno تغییر دهید.

---

[1] برای یک جریان از داده‌ها، goodput عبارتست از نرخ داده‌های application (یعنی، داده‌های مفید) که به طور موفقیت‌آمیز ارسال می‌شوند. برای سؤالاتی که در بخش‌های آتی مواجه می‌شوید، باید در نظر بگیرید که بسته‌ها علاوه بر داده‌های application دارای سرآیند (هدر) هم هستند.

ب) جریان‌های TCP و UDP

اسکریپت lab5_network.py موجود در فولدر lab5، توپولوژی نمایش داده شده در شکل ۱ را می‌سازد.



شکل ۱- توپولوژیِ متشکل از سه PC و یک روتر

فایل lab5_network.py را طوری تکمیل کنید که جداول مسیریابی و آدرس‌های IP طبق روش آدرس‌دهی زیر پیکربندی شوند:

- **سابنتِ ماشین‌های h1، h2 و روترِ r1 دارای آدرس 10.10.0.0/24 است. آدرس‌های h1، h2 و روتر هم به ترتیب به صورت 10.10.0.1، 10.10.0.2 و 10.10.0.10 می‌باشد.**
- **سابنتِ ماشین h3 و روتر دارای آدرس 10.10.1.0/24 است. آدرس‌های h3 و روتر هم به ترتیب، 10.10.1.3 و 10.10.1.10 می‌باشد.**

یک ترمینال در ماشین خود باز کنید و اسکریپت lab5_network.py را اجرا نمایید. پس از اجرای توپولوژی، با استفاده از دستور pingall، پیکربندی خود را آزمایش نمایید.

ب-۱) تست اتصال بین کلاینت و سرورهای TCP و UDP

دایرکتوری lab5/udp حاوی دو برنامه است: udpserver و udpclient. نحوهٔ استفاده از این برنامه‌ها به صورت زیر است:

```
# ./udpserver PORT
# ./udpclient IP_SERVER PORT RATE
```

برای سِرور، PORT همان شمارهٔ پورتی است که روی آن سِروِر مشغول گوش کردن می‌باشد. برای کلاینت هم IP_SERVER و PORT به ترتیب، آدرس IP و شماره پورت ماشینی را مشخص می‌کند که بسته‌ها به مقصد آن ارسال می‌شوند و RATE هم نرخی را مشخص می‌کند که با آن داده‌ها توسط کلاینت‌ها ارسال می‌شوند (برحسب کیلوبیت در

ثانیه). اگر نرخ کمتر از 50kbps باشد، کلاینت‌ها بسته‌های به اندازهٔ 125 بایتی ارسال می‌کنند وگرنه بسته‌های به اندازهٔ 1000 بایت می‌فرستند.

خروجی کلاینت UDP دارای فرمت زیر است[۲]:

```
4.0s - sent:  503 pkts, 1000.0 kbits/s
5.0s - sent:  629 pkts, 1000.6 kbits/s
```

به عنوان مثال، مقادیر خط دوم را می‌توان به این نحو تفسیر کرد که:

- 5.0 عبارتست از تعداد ثانیه‌های سپری شده از لحظهٔ شروع بکار کلاینت
- 629 یعنی تعداد کل بسته‌های ارسالی توسط کلاینت
- 1000.6 نرخ ارسال طی آخرین ثانیه است (برحسب kbit/s).

خروجی سِرورِ UDP دارای فرمت زیر است:

```
169.5s - received:  723/ sent:  741 pkts (loss 2.429%), 959.6 kbit/s
170.5s - received:  843/ sent:  867 pkts (loss 2.768%), 957.7 kbit/s
```

به عنوان مثال، مقادیر خط دوم را می‌توان به این نحو تفسیر کرد که:

- 170.5 زمانی است که از راه‌اندازی سِرور می‌گذرد.
- 843 یعنی تعداد کل بسته‌های دریافتی توسط سِرورِ
- 867 یعنی تعداد کل بسته‌های ارسالی توسط کلاینت
- 2.768 یعنی درصد بسته‌هایی که از بین رفته‌اند
- 957.7 نرخی است که طی آخرین ثانیه، بسته‌ها با آن دریافت شده‌اند. به این مقدار، اصطلاحاً goodput| طی آخرین ثانیه گفته می‌شود.

یک سِرورِUDP در h3 راه‌اندازی کنید که روی پورت 10000 گوش می‌کند.

* **نکته:** با توجه به اینکه Mininet یک امولاتور است، اگر یک آزمایش را چندین مرتبه اجرا کنید، نتایج هر بار می‌تواند متفاوت باشد. به همین مناسبت، توصیه می‌شود که هر آزمایش را چندین مرتبه اجرا نمایید (مثلاً: ۵ بار) و مقادیر میانگین را به عنوان جواب ارائه دهید.

[۲] برای کلیه موارد این آزمایش، مقادیر نرخ چاپ شده، از نوع داده‌های application هستند و حجم سرآیندها (هِدِرها) در آنها لحاظ نشده است.

سؤال ۱: روی ماشین h1 یک کلاینت UDP اجرا کنید که داده‌ها را برای سِروِر h3 با نرخ 100kbps ارسال می‌کند. احتمال loss و goodputای که در h3 مشاهده می‌شود، چقدر است؟

سؤال ۲: آزمایش را با نرخ‌های 1Mbps، 10Mbps، 100Mbps و 1Gbps تکرار کنید. مقادیر goodput و احتمال‌های loss چقدر است؟ به ازای چه نرخی، احتمال loss بالاتر از 1% است؟ آیا می‌توانید نتایج را با توجه به کُد برنامه lab5_network.py توجیه نمایید؟

دایرکتوری lab5/tcp حاوی دو برنامه است: tcpserver و tcpclient. نحوهٔ استفاده از آنها مشابهِ udpserver و udpclient است به جز اینکه ما نرخی را برای کلاینت مقرّر نمی‌نماییم. در واقع، فرض می‌شود که کلاینت دارای حجم نامتناهی داده برای ارسال است و از مکانیزم کنترل ازدحام TCP برای کنترل نرخ ارسال داده‌ها به سوی سِرور استفاده می‌شود.

```
# ./tcpserver PORT
# ./tcpclient IP_SERVER PORT
```

خروجی کلاینت TCP چیزی شبیه به شکل زیر است:

```
6.3:  854.0kbps avg ( 944.5[inst], 926.5[mov.avg]) cwnd  9 rtt 83.9ms
7.3:  862.4kbps avg ( 914.6[inst], 925.3[mov.avg]) cwnd  9 rtt 86.8ms
```

به عنوان مثال، مقادیر خط دوم را می‌توان به صورت زیر تفسیر کرد:

- 7.3 نمایانگر زمان است
- 862.4 میانگین نرخ کلاینت است: یعنی، کل حجم داده‌ای که به طور موفقیت‌آمیزی توسط کلاینت منتقل شده است تقسیم بر کل زمان (مقدار حاصل را می‌توان به عنوان مقدار میانگین goodput برای TCP نیز تعبیر نمود).
- 914.6 همان نرخ آنی است (تقریباً طی ثانیهٔ آخر).
- 925.3 نیز مقدار میانگین متحرک (moving average) نرخ است.
- مقدار 9 هم اندازهٔ «پنجرهٔ ازدحام» TCP را نشان می‌دهد.
- 86.8 یعنی RTT اندازه‌گیری شده توسط الگوریتم کنترل ازدحام TCP.

- **یک سِروِر TCP روی ماشین h3 راه‌اندازی کنید که روی پورت 10001 گوش می‌دهد[۳].**

سؤال ۳: یک کلاینت TCP روی ماشین h1 اجرا نمایید که برای سِروِر h3 داده ارسال می‌کند. Goodput ارتباط چقدر است؟

[۳] قبل از راه‌اندازی هر سِرور و کلاینت جدید، کلیه کلاینت‌ها و سِرورهای قبلی را kill کنید. این کار باعث می‌شود تا مقادیر متوسطی که چاپ می‌شوند، reset شوند.

* **نکته:** برای تمامی آزمایش‌ها باید صبر کنید تا مقادیر چاپ شده به حالت پایداری برسند. این امر به خصوص برای TCP خیلی اهمیت دارد. نرخ ارسال داده‌ها در TCP به وقوع lossها ربط دارد که آنها هم به صورت تصادفی رخ می‌دهند. به همین دلیل، برای دستیابی به مقادیر قطعی، باید صبر کنید تا نرخ متوسط به پایداری برسد (برای اغلب سناریوها حدود ۲ دقیقه کفایت می‌کند). ضمن اینکه هر آزمایش ترجیحاً باید چند بار تکرار شود.

ب-۲) محدودسازی پهنای باند روتر

برای انجام آزمایشاتی که در آنها کارایی شبکه بابت ظرفیت برخی لینک‌ها دچار محدودیت است، در این بخش، پهنای باند بعضی از اینترفیس‌ها را بیشتر محدود می‌نماییم.

مثلاً، به عنوان یک راهکار می‌توان از امکانات کلاس TCLink در Mininet و پارامترهایی که برای محدودسازی پهنای باند در اختیار می‌گذارد، بهره گرفت. به طور مشخص، دستور زیر:

```
net.addLink( h1, h2, bw=5, delay='2ms', max_queue_size=1000, loss=1)
```

یک لینک دو طرفه بین ماشین‌های h1 و h2 با پهنای باندِ 5Mbps، تأخیرِ 2ms، احتمال loss برابر با 1% و حداکثر سایز صفِ 1000 بسته ایجاد می‌کند. پارامتر bw عددی را بر حسب Mbps مشخص می‌کند؛ تأخیر به صورت یک رشتهٔ با واحد (مثل: '5ms' یا '100us' یا '1s') بیان می‌شود؛ احتمال loss هم برحسب درصد (بین ۰ تا ۱۰۰) معیّن می‌گردد و max_queue_size هم برحسب تعداد بسته‌ها تعیین می‌شود. برای اطلاعات بیشتر می‌توانید به آدرسِ http://mininet.org/api/classmininet_1_1link_1_1TCLink.html مراجعه نمایید.

به عنوان راهکار دیگر، در اسکریپتِ lab5_network.py که در بخش قبلی آن را تکمیل و اجرا نمودید، دستوری به صورت زیر وجود دارد:

```
link_r1sw2.intf1.config( bw=10 )
```

این دستور صرفاً پهنای باند اینترفیس eth1 از روتر را روی 10Mbps تنظیم می‌کند (به جای اینکه پهنای باندِ کل لینک را محدود سازد). با این وجود، حداکثر ترافیکی که قابل انتقال خواهد بود، مشابهِ زمانی است که شما پهنایِ باند کل لینک را به 10Mbps تنظیم نمایید (مثلاً: از طریق دستورِ link_r1sw2=net.addLink(r1,sw2,bw=10) که در زمان اجرای یک لینک قابل استفاده است).

- **پهنای باند اینترفیس eth1 از روتر را به 3 Mbps محدود سازید.**

* **توجه:** برای محقق شدن تغییر پهنای باند، باید از Mininet خارج شده، توپولوژیِ قبلی را clean-up کرده و سپس مجدداً اسکریپت lab5_network.py را اجرا نمایید.

* **توجه:** محدودیت 3 Mbps هم برای بسته‌های وارده و هم خارجه از اینترفیس eth1 روترِ اعمال خواهد شد چراکه کلاس TCLink، اینترفیس‌های متقارن ایجاد می‌کند. در آزمایش‌های آتی، توضیح خواهیم داد که چگونه با استفاده از دستورات لینوکس می‌توان محدودیت‌های پهنای باند را به صورت نامتقارن بر مبنای زمان‌بندی CBQ[4] ایجاد کرد.

ب-۲-۱) تست UDP

یه یاد داریم که کلاینت UDP در شرایطی که RATE را روی مقداری بزرگتر از 50 kbps تنظیم کنیم، بسته‌هایی هرکدام به اندازهٔ 1000 بایت ارسال خواهد نمود.

سؤال ۴: به لحاظ تئوری، انتظار داریم اندازهٔ (بر حسب بایت) فریم‌های Ethernetی که برای ارسال داده-های کلاینت استفاده می‌شوند، به صورت زیر باشد:

هدِر (سرآیند) UDP به میزان ۸ بایت، هدِر IP به مقدار ۲۰ بایت، هدِر Ethernet به میزان ۱۴ بایت و اندازهٔ داده‌های Application هم که ۱۰۰۰ بایت؛ پس، مجموعاً: ۱۰۴۲ بایت.

با استفاده از WireShark همخوانی واقعیت با این مقدار تئوری را کنترل کنید.

سؤال ۵: پس از اِعمال محدودیت 3 Mbps در پهنای باند، روتر در توپولوژیِ شکل ۱ تبدیل به گلوگاه (bottleneck) شبکه می‌شود. به لحاظ تئوری، حداکثر مقدار قابل دستیابی برای گذردهی داده‌های کاربردی (همان goodput) چقدر خواهد بود؟ محاسبه کنید.

- **یک سِروِر UDP روی ماشین h3 راه‌اندازی کنید که روی پورت 10000 گوش می‌کند.**

سؤال ۶: یک کلاینت UDP روی ماشین h1 راه‌اندازی نمایید که داده‌ها را با نرخ 100 kbps ارسال می‌کند. مقدار احتمال loss و همچنین goodput مشاهده شده در h3 چقدر است؟

سؤال ۷: عملیات مورد نظر در سؤال ۶ را برای نرخ‌های 3Mbps و 10Mbps انجام دهید. مقادیر goodput و احتمالات loss چقدر می‌شود؟ مقادیر حاصل برای goodput در همه این موارد را با مقداری که انتظار دارید (و در سؤال ۵ محاسبه کردید)، مقایسه نمایید.

ب-۲-۲) تست TCP

سؤال ۸: انتظار داریم اندازه (برحسب بایت) فریم‌های اترنتی که برای ارسال داده‌ها توسط ارتباط TCP استفاده می‌شوند برابر با 1514 بایت باشد چراکه: با توجه به MTU، داده‌های برنامهٔ کاربردی 1448 بایت، هدِر IP 20

[4] Class-Based Queuing (CBQ)

بایت، هِدِر اترنت 14 بایت و هِدِر TCP برابر با 32 بایت است. این فرضیه را از طریق گوش دادن به بسته‌ها در سمت سِروِر بررسی کنید.

* توجه: اگر قابلیت TCP Large Segment Offload در یک ماشین فعال باشد (که گاهاً به آن، TSO یا LSO نیز گفته می‌شود)، سیستم‌عامل بسته‌های بزرگتر از MTU به کارت شبکه می‌دهد و درایور کارت شبکه نیز برای جا دادن بسته‌ها در MTU، آنها را خُرد می‌نماید. اگر بتوان بسته‌ها را مستقیماً از رسانه شنود کرد (به جای اینکه از endpoint ارتباط شنود کنیم)، خواهیم دید که بسته‌ها با اندازهٔ درست (همان 1514 بایت) ارسال می‌شوند. قابلیت TSO نوعی بهبود روی عملکرد TCP است ولی امکان غیرفعال کردن آن نیز وجود دارد تا سیستم‌عامل دیگر فریم‌های oversized تولید نکند. برای این منظور در h1 دستور زیر را اجرا نمایید:

```
ethtool -K h1-eth0 tx off sg off tso off
```

سؤال ۹: به لحاظ تئوری، حداکثر مقدار مورد انتظار برای goodput داده‌های کاربردی چقدر است؟ نحوهٔ محاسبهٔ خود را تشریح کنید.

- **یک سِروِر TCP روی ماشین h3 راه‌اندازی کنید که روی پورت 10001 گوش می‌کند.**

سؤال ۱۰: یک کلاینت TCP روی ماشین h1 راه‌اندازی کنید که داده‌هایی را برای سِروِر واقع در h3 می‌فرستد. Goodput ارتباط چقدر است؟ آن را با مقدار تئوری مورد سؤالِ ۹ مقایسه نمایید.