آزمایشگاه شبکه

آزمایش ۷: بررسی رفتار جریان‌های TCP و UDP

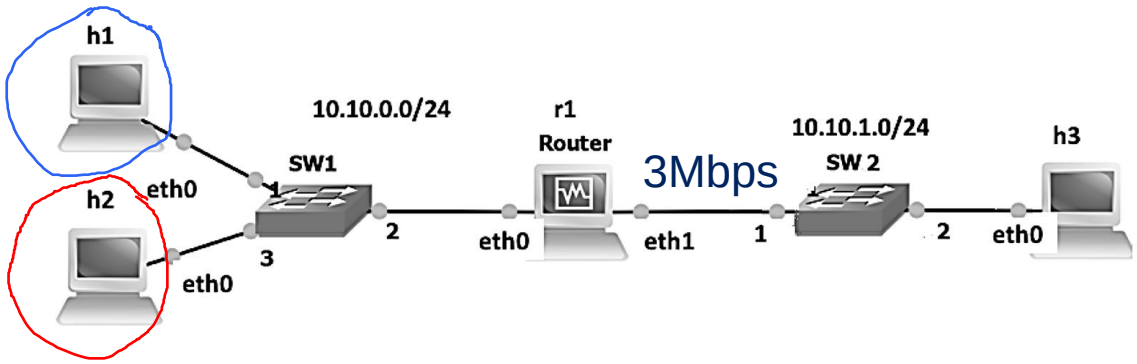
در این آزمایش، به بررسی رفتار جریان‌های ترافیک TCP و UDP در سناریوهای مختلف می‌پردازیم.

الف) رقابت جریان‌های UDP با یکدیگر

بار دیگر، توپولوژیِ آزمایش قبل را تجسم کنید (شکل ۱). پیش از انجام آزمایش، قصد داریم که در سه سناریوی مختلف، مقدار داده‌های تبادل‌شده را پیش‌بینی کنیم.

جدول ۱- سه سناریوی ترافیکی مختلف مبتنی بر UDP

سناریو	نرخ h1 (UDP)	نرخ h2 (UDP)
۱	1 Mbps	1 Mbps
۲	1 Mbps	2 Mbps
۳	1 Mbps	4.5 Mbps



شکل ۱- توپولوژیِ متشکل از سه PC و یک روتر

در هر سناریو از جدول ۱، بر اساس تحلیل نظری، مقادیر goodput و احتمالات loss را می‌توان به این شرح محاسبه نمود: X را برابر با نرخ h1 و Y را نرخ h2 در نظر بگیرید. داریم:

$$goodput_{h1} = \min\left(\left(\frac{X}{X+Y}\right) \times \frac{1000}{1042} \times 3, X\right) Mbps$$

$$goodput_{h2} = \min\left(\left(\frac{Y}{X+Y}\right) \times \frac{1000}{1042} \times 3, Y\right) Mbps$$

بر اساس این محاسبات، جدول ۲، goodput و احتمال loss را برای سه سناریوی مندرج در جدول ۱ نشان می‌دهد.

جدول ۲- مقادیر نظریِ goodput و احتمال loss برای سه سناریوی ترافیکی مختلف مبتنی بر UDP

سناریو	مقدار goodput برای h1	احتمال loss برای h1	مقدار goodput برای h2	احتمال loss برای h2
(۱)	1 Mbps	0%	1 Mbps	0%
(۲)	0.9596 Mbps	4%	1.919 Mbps	4%
(۳)	0.523 Mbps	48%	2.36 Mbps	48%

حال، می‌خواهیم که نتایج تحلیلی فوق را با انجام آزمایش، مورد بررسی قرار دهیم. مشابه قبل، پهنای باند اینترفیس eth1 از روتر به مقدار 3Mbps محدود شده است.

- **برای هر سناریو، دو سِرورِ UDP روی h3 اجرا نمایید که روی پورت‌های 10001 و 10002 گوش می‌دهند. از دستور xterm h3 در پنجره ترمینال mininet استفاده کنید تا یک ترمینال جدید برای h3 باز شود. در انجام آزمایش، دقت کنید که این دو پنجره را به اشتباه نگیرید. سپس، یک کلاینت UDP روی h1 باز کنید که با نرخ 1 Mbps برای h3 داده ارسال می‌کند و یک کلاینتِ UDP هم روی h2 باز کنید که (بسته به سناریو) با نرخ‌های 1 Mbps، 2 Mbps و 4.5 Mbps برای سِرورِ h3 داده می‌فرستد.**

سؤال ۱: مقادیر goodput و احتمالات loss مورد مشاهده در سناریوهای (۱)، (۲) و (۳) چقدر است؟

سؤال ۲: آیا تفاوتی میان این مقادیر تجربی با مقادیر تحلیلی مشاهده می‌کنید؟ اگر بلی، فکر می‌کنید این تفاوت‌ها ناشی از چیست؟

همیشه ثابت نیستند و تغییر میکنند
ترتیب وارد شدن به صف ها هم شانسی است
کدام را اول انجام می دهیم و کدام را دوم

$$goodput_{h1} = \min\left(\left(\frac{X}{X+Y}\right) \times 3 \times \frac{1000}{1042}, X\right) Mbps$$

$$goodput_{h2} = \min\left(\left(\frac{Y}{X+Y}\right) \times 3 \times \frac{1000}{1042}, Y\right) Mbps$$

ب) رقابت جریان TCP با جریان‌های UDP

در این بخش، سناریوی مشابهِ بخش (الف) را مد نظر قرار می‌دهیم. جدول ۳ را ملاحظه نمایید. ماشین h1 جریان داده‌های UDP را با نرخ 1 Mbps به سوی h3 ارسال می‌کند و ماشین h2 نیز جریان UDP دیگری (مثلاً: ویدئو) با نرخ‌های 1 Mbps، 2 Mbps و 4 Mbps برای h3 می‌فرستد. علاوه بر اینها، ماشین h2 یک ارتباط TCP با h3 باز کرده تا مثلاً فایلی را روی این سِرورِ بارگذاری نماید.

جدول ۳- سه سناریوی ترافیکی مختلف با ترکیب جریان‌های TCP و UDP

سناریو	نرخ h1 (UDP)	نرخ h2 (UDP)
(۱)	1 Mbps	1 Mbps
(۲)	1 Mbps	2 Mbps
(۳)	1 Mbps	4.5 Mbps

در هر سناریو از جدول ۳، بر اساس تحلیل نظری، مقادیر goodput و احتمالات loss را می‌توان به این شرح محاسبه نمود: X را برابر با نرخ جریان UDP در h1، Y را نرخ جریان UDP در h2 و Z را هم نرخ جریان TCP در h2 در نظر بگیرید. داریم:

$$Z = 3 - X - Y \; Mbps$$

$$goodput_{h1,UDP} = \min\left(\left(\frac{X}{X+Y}\right) \times \frac{1000}{1042} \times 3, X\right) Mbps$$

$$goodput_{h2,UDP} = \min\left(\left(\frac{Y}{X+Y}\right) \times \frac{1000}{1042} \times 3, Y\right) Mbps$$

$$goodput_{h2,TCP} = Z \times \frac{1448}{1514} \; Mbps$$

pure data

overhead + pure data

* در واقع، به طور دقیق‌تر، در مواردی که $X + Y < 3$، خواهیم داشت که:

$$Z = 3 - \frac{1042 \times (X+Y)}{1000},$$

$$goodput_{h1,UDP} = X, \;\; goodput_{h2,UDP} = Y, \qquad goodput_{h2,TCP} = Z \times \frac{1448}{1514}$$

بر اساس محاسبات فوق، جدول ۴، مقادیر goodput را برای سه سناریوی مندرج در جدول ۳ نشان می‌دهد.

جدول ۴- مقادیر نظریِ goodput برای سه سناریوی ترافیکی مختلف مبتنی بر UDP و TCP

سناریو	مقدار goodput برای جریان UDP در h1	مقدار goodput برای جریان UDP در h2	مقدار goodput برای جریان TCP در h2
(۱)	1 Mbps	1 Mbps	0.876 Mbps
(۲)	0.959 Mbps	1.919 Mbps	0 Mbps
(۳)	0.523 Mbps	2.36 Mbps	0 Mbps

سؤال ۳: سناریوهای جدول ۳ را مورد آزمایش تجربی قرار دهید. آیا تفاوتی میان این مقادیر تجربی با مقادیر تحلیلی مشاهده می‌کنید؟ اگر بلی، فکر می‌کنید این تفاوت‌ها ناشی از چیست؟

ج) بررسی تأثیر مکانیزم «اخطار صریح ازدحام[1]» بر RTT و «پنجرهٔ ازدحام[2]»

قابلیت ECN این امکان را فراهم می‌کند که بدون drop شدن بسته‌ها، فرستنده‌ها را از رویداد ازدحام قریب‌الوقوع مطلع ساخت. به طور سنتی، شبکه‌های TCP/IP با drop کردن بسته‌ها وقوع ازدحام را به طور ضمنی اطلاع می‌دهند. امّا وقتی قابلیت ECN فعال شود، یک روترِ مجهز به این قابلیت، می‌تواند به جای drop کردن بسته، یک نشانه در هدرِ IP بگذارد تا از احتمال وقوع ازدحام خبر دهد. گیرندهٔ بسته هم، اخطار ازدحام را به اطلاع فرستندهٔ نظیرش می‌رساند تا در نهایت، با کاهش نرخ ارسال واکنش نشان دهد.

ACK

small cwnd

در این بخش، تأثیر فعال‌سازیِ قابلیت ECN روی RTT و «پنجرهٔ ازدحام» را بررسی خواهیم نمود. برای این منظور، از همان توپولوژیِ شکل ۱ استفاده می‌کنیم.

- **پس از خروج از Mininet و پاک‌کردن توپولوژیِ پیشین، اسکریپت lab6.py را طوری تغییر دهید که طول صف روتر ۱ (با تنظیم پارامتر max_queue_size) به مقدار ۱۰۰۰ بسته کاهش یابد. بعلاوه، پهنای باند اینترفیس eth1 از این روتر را روی 5 Mbps تنظیم کرده و ویژگیِ enable_ecn را نیز برابر False قرار دهید. یک سِروِر TCP روی ماشین h3 اجرا نموده و یک کلاینت TCP هم روی ماشین h1 بالا بیاورید. منتظر بمانید تا نرخ‌ها پایدار شوند.**

حالت سنتی

link_r1sw2.intf1.config(bw=5, max_queue_size=1000, enable_ecn=False)

سؤال ۴: مقدار نرخ ماشین منبع (یعنی h1) چقدر است؟ حدود مقادیر RTT و نیز محدودهٔ مقادیر «پنجرهٔ ازدحام» را مشخص نمایید.

حالت مدرن تر

- **حال، با دستکاریِ پیکربندیِ eth1 از روتر ۱، قابلیت ECN در آن را فعال نمایید، به صورت زیر:**

```
link_r1sw2.intf1.config(bw=5, max_queue_size=1000, enable_ecn=True)
```

ویژگیِ enable_ecn به صورت پیش‌فرض با مقدار False تنظیم شده است که اگر این مقدار را به True تغییر دهید، عملاً ECN فعال می‌شود. حال، از Mininet خارج شده و پس از پاک کردن توپولوژیِ قبلی، اسکریپت اصلاح شدهٔ lab6.py را اجرا نمایید. یک سِروِر TCP روی ماشین h3 بالا آورده و یک کلاینت TCP هم روی h1 اجرا کنید. منتظر بمانید تا نرخ‌ها پایدار شوند.

---

[1] Explicit Congestion Notification (ECN)
[2] Congestion Window (cwnd)

سؤال ۵: مقدار نرخ منبع (یعنی h1) چقدر است؟ محدودهٔ مقادیر RTT و حدود مقادیر «پنجرهٔ ازدحام» را بیان کنید.

سؤال ۶: با مقایسهٔ مقادیر مشاهده شده در سؤال ۵ با مقادیری که نظیر حالتِ ECN غیرفعال هستند (سؤال ۴)، چه نتیجه‌ای می‌توان گرفت؟

الکی بسته های بیخود که فقط ذراپ میشن فرستاده نمیشه و درنتیجه شبکه شلوغ نمیشه و صف تشکیل نمیشه و سرعت بالا میره و تاخیر صف خیلی کمتر میشه

د) عدالت در TCP و تأثیر RTT

الگوریتم کنترل ازدحام TCP تضمین می‌کند که منابع شبکه به طور عادلانه میان ارتباطات مختلف به اشتراک گذاشته شود. در این بخش، بررسی می‌کنیم که در شرایطی که چندین گلوگاه (bottleneck) در شبکه وجود دارد، الگوریتم RENO چگونه پهنای باند را به اشتراک می‌گذارد. مشخصاً، وجود دو ویژگی را در RENO بررسی می‌کنیم: اینکه RENO برای هر جریان (عادلانه) fair است و اینکه نسبت به تأخیر، حساس است.

- **برای بخش‌های (د-۱) و (د-۲)، همان توپولوژیِ شکل ۱ را مبنا قرار می‌دهیم. همچنین، پهنای باند روتر را به 3 Mbps محدود می‌کنیم و طول صف را به ۱۰۰۰ بسته.**
- **برای آزمایش‌های این بخش، اهمیت ویژه‌ای دارد که صبر کنید تا goodputهای حاصل، پایدار شوند (حدود ۵ دقیقه). همچنین، وقتی پهنای باند روتر را محدود می‌نماییم، برای اینکه همگرایی بهتری عاید شود و کمتر تحت تأثیرِ تأخیر صف باشیم (که روی RTT اثر می‌گذارد)، باید مشابه بخش قبل، قابلیت ECN را فعال کنیم.**

د-۱) افزودن تأخیر به یک اینترفیس

به منظور اینکه شرایط آزمایش را واقع‌بینانه‌تر کنیم، قدری تأخیر به شبکه می‌افزاییم. برای این منظور، از ماجول netem از نرم‌افزار کنترل ترافیکی که در لینوکس موجود است، استفاده می‌کنیم. می‌توان از دستور tc برای تعریف یک قانون جهت افزودن تأخیر به یک اینترفیس بهره گرفت (جهت آشنایی بیشتر با سازوکار دستورات tc و ماجول netem به وِب‌سایت http://www.linuxfoundation.org/collaborate/workgroups/networking/netem رجوع نمایید). برای مثال، دستور زیر به میزان 300ms تأخیر به کلیهٔ بسته‌های خارج‌شونده از اینترفیس eth0 اضافه می‌کند (توجه کنید که افزایش تأخیر فقط در یک جهت انجام می‌شود و نه به بسته‌های وارد شونده):

```
# tc qdisc add dev eth0 root netem delay 300ms
```

برای اِعمال تغییر در این قانون، می‌توان دستوری مثل زیر را تایپ نمود:

```
# tc qdisc change dev eth0 root netem delay 400ms
```

همچنین، برای حذف یک قانون، دستور زیر می‌تواند استفاده شود:

```
# tc qdisc del dev eth0 root
```

سؤال ۷: به مقدار 300ms تأخیر به اینترفیس eth0 از h3 اضافه نمایید. سپس، از سویِ h1، ماشین h3 را پینگ کنید. مقدار RTT مشاهده شده چقدر است؟

* اگر جدول ARP را flush کرده باشید (مثلاً با استفاده از ifconfig eth0 down و ifconfig eth0 up) و سپس، netem را برای اضافه کردن تأخیرِ 300ms استفاده کنید، RTT اولین بسته باید بزگتر از RTT بسته دوم باشد (به خاطر درخواست ARP).

clear

د-۲) عدالت میان جریان‌های TCP و تأثیر تأخیر بر آن

TCP نوعی اشتراک‌گذاریِ عادلانه میان جریان‌های ترافیک فراهم می‌کند به این معناکه ماشینی که چندین ارتباط TCP باز می‌کند، پهنای باند بیشتری هم عایدش خواهد شد. برای بررسی این موضوع، سناریویی را در نظر می‌گیریم که در آن،

- یک مقدار تأخیر اضافهٔ 300ms روی اینترفیس eth0 از ماشین h3 وجود دارد ولی هیچ تأخیر اضافه‌ای روی ماشین‌های h1 یا h2 نیست.
- ماشین h1 یک ارتباط TCP به سوی h3 باز می‌کند و ماشین h2 هم سه ارتباط TCP به سوی h3 می‌گشاید.

بر مبنایِ تحلیل نظری، کل goodputی که ماشین‌های h1 و h2 در سناریوی فوق بدست می‌آورند، به صورت زیر قابل محاسبه است:

فرض کنید که سهم جریان h1 را با x نشان دهیم. در اینصورت، سهم h2 برابر است با 3x. از طرفی، مجموع این دو جریان یعنی 4x باید حداکثر برابر با ۳ بشود. پس، x=3/4. در نهایت، می‌توان سهم goodput ماشین h1 را به صورتِ 3/4*1448/1514 محاسبه نمود که برابر می‌شود با: 717 kbps. در خصوص سهم کلی h2 هم این مقدار معادلِ 2152 kbps خواهد شد.

- **سه کلاینتِ TCP روی ماشین h2 اجرا کنید و تنها یک کلاینت TCP روی h1. صبر کنید نرخ‌ها پایدار شوند.**

سؤال ۸: مقدار goodput هر جریان از ماشین‌های h1 و h2 چقدر است؟

سؤال ۹: آیا مقادیر حاصل از آزمایش با مقادیر نظری همخوانی دارند؟

سؤال ۱۰: آیا می‌توانید متوجه شوید که اساساً تأخیر صف هم داریم یا خیر؟ بله

نرخ ندهد افزایش میابدloss قابل مشاهده است تا جایی کهcwnd همانطور که از اندازه کاهش میابد که نشاندهندهی تاخیر صف استloss و پس از

ping --> rtt: 300ms
now --> rtt: 330-372 ms
yes we have queuing delay
queuing delay = 30-72 ms
reason: delay in output port(before sending delay) router cause queue when packets arrive