شبکه های کامپیوتری | تکلیف کامپیوتری اول | ۱۳۹۸/۸/۱۶
یاسمن پرهیزکار | | پریسا قاسمی
۸۱۰۱۹۵۵۵۹ | | ۸۱۰۸۹۴۰۳۷

**بخش ۱:**

تصویر اجرای دستور های ipconfig و ifconfig را در شکل های 10 و 11 مشاهده می کنید.

- باید ابتدایِ دستور مورد نظر، عبارت ip netns exec <my_netnamespace> را نوشته و در ادامه، دستور را با syntax ِ عادی آن بنویسیم.
- با توجه به اینکه دستورِ ping بسته ها را به آدرسِ IP مشخص شده فرستاده و دوباره دریافت می کند، باید آدرس IP موردنظر حتما وجود داشته باشد. در اینجا، پیش از نسبت دادن آدرس 10.0.0.2 از آن استفاده کردیم و به دلیل **عدم وجود** این آدرس، با ارور دسترسی مواجه می شد.
  جهت رفع این خطا باید طبق دستور زیر، آدرس مذکور را به واسطه ای که مقصد دستور ping است، نسبت دهیم:
  ip netns exec <my_netnamespace> ip ifconfig <interface_name> 10.0.0.2
- همانطور که در بالا گفته شد، یا باید از دستور
  ip netns exec <my_netnamespace> ip ifconfig <interfacename> <my_IP_addr>
  استفاده کنیم یا از دستور ip add addr استفاده کنیم.
- با ایجاد یک network namespace یک میزبان مجازی ایجاد میشود که این میزبان می تواند به تعداد دلخواهی واسطه (interface) داشته باشد. یکی از انواع این واسطه ها virtual ethernet (veth) است. هر veth مانند یک کابل LAN مجازی می ماند که در هر دو سر آن، یک واسطه قرار میگیرد و می تواند دو میزبان را به هم متصل کند.
- با دستور ip link add <my_link_name> type veth peer name <my_peer_name>
- با دستور زیر می توان یکی از واسطه های veth را از میزبان پیش فرض به network namespace دلخواه منتقل کرد:
  ip link set <interface_name> netns <my_netnamespace>
- این دستور، تمام آدرس ها (از جمله IPv4) تمامی واسطه های متصل به network namespace ِ ذکر شده در دستور را نمایش می دهد.

**بخش ۲:**

کد این بخش فایل Q2.sh است. با اجرای دستور ping موجود در فایل، میزبان h1 شروع به فرستادن بسته های icmp به میزبان h2 (با آدرس IP نوشته شده) می کند، و طبق قانونِ بسته های icmp، میزبان h2 دقیقا همان بسته را به h1 برمی گرداند.

البته در جدول مسیریابی h1 و h2 نوشته شده که هر بسته ای با هر IP ِ مقصدی را صرفا به سر دیگر کابل متصل به خود (یعنی به سوییچ) بفرستند و سوییچ هر بسته دریافتی را به میزبان مربوطه اش می فرستد.

**بخش ۳:**

کد توپولوژی این سوال، در فایل Q3.py موجود است. همچنین، نتایج nodes و net توپولوژی را در شکل 31 مشاهده می کنید.

- با دستور pingall، تمام میزبان‌های توپولوژی مورد نظر، یکدیگر را ping میکنند. از این دستور میتوان برای اطمینان حاصل کردن از صحت اتصال link های شبکه استفاده کرد. (شکل 32)

- همانطور که انتظار داشتیم آدرس IP ها به شرح زیر است (شکل 33):

h1: 10.0.0.1 ; h2: 10.0.0.2 ; h3: 10.0.0.3 ; h4: 10.0.0.4

**بخش ۴:**

فایل توپولوژی این بخش با نام Q4.py ضمیمه شده است.

- تاثیر Delay لینک ها:

طبق رابطه زیر، اگر تاخیر یک لینک به اندازه x میلی ثانیه زیاد شود، تاخیر کل ping کردن 2x زیاد می شود؛ چون بسته icmp از هر لینک دو بار می گذرد.

$$T_{ping} = TD_{h1} + 2 * [(PD_{l1} + x) + QD_{s1} + TD_{s1} + PD_{l2} + QD_{s2} + TD_{s2} + PD_{l3}] + TD_{h2}$$

شکل 41 و 42

- تاثیر max_queue_size لینک ها:

پارامتر max_queue_size تعداد بیشینه بسته هایی را نشان می دهد که می توانند در آنِ واحد روی یک لینک قرار بگیرند. وقتی max_queue_size را ۱۰۰ بسته گذاشته بودیم، عملا هیچگاه ۱۰۰ بسته همزمان روی یک لینک نداشتیم و این پارامتر تاثیری در زمان ping نداشت. اما وقتی به ۱ بسته کاهش دادیم، گهگاهی پیش می آمد که روی یک لینک دو بسته قرار می گرفت که در این صورت، بسته دوم drop می شد و توسط ping کننده دریافت نمی شد. لذا، هر چند وقت یکبار یک host unreachable داشتیم.

شکل 43 و 44

- تاثیر BandWidth لینک ها:

Bandwidth در تاخیر انتقال تاثیر دارد. طبق فرمول $TD = \frac{L}{R}$ ، هرچه پهنای باند کمتر و سایز بسته بزرگتر باشد، تاخیر انتقال بیشتر می شود که موجب افزایش تاخیر ping می گردد.

در شکل 40 و 41 می بینیم که کاهش پهنای باند از 10Mbps به 1Mbps تغییر چندانی در تاخیر ping نداد؛ چراکه سایز بسته ها آنقدر کوچک بود که در هر دو حالت تاخیر انتقال تقریبا صفر بوده و تاخیر ping بیشتر ناشی از تاخیر انتشار و صف است.

اما در شکل 45 می بینیم که با کاهش پهنای باند به 0.01Mbps تاخیر ping به طرزی نمایی افزایش پیدا کرد و با هر ping جدیدی، همچنان در حال افزایش بود؛ دلیل این پدیده، این است که تاخیر انتقال آنقدر زیاد شده که در سوییچ ها صف تشکیل می شود و تاخیر صف هم اضافه می گردد. بعلاوه، با هر ping جدید، طول صف ها افزایش می یابد.

- تاثیر افزایش تعداد سوییچ ها:

طبعا افزایش تعداد سوییچ ها منجر به افزایش تعداد لینک ها (⟵ تاخیر انتشار)، تعداد تاخیر های انتقال (به خاطر store&forward بودن سوییچ ها) و تاخیرهای صف می شود. در نتیجه تاخیر ping افزایش می یابد.

شکل های 46 و 41 که تاخیر انتشار لینک ها در آنها 20ms است، تاثیر افزایش تعداد سوییچ ها در تاخیر انتشار را نشان می دهند و شکل های 47 و 48 (که تاخیر انتشار آنها بسیار کم است)، تاثیر افزایش تعداد سوییچ ها در تاخیر انتقال را نشان می دهند.

- تاثیر تاخیر ping در تعداد بسته هایی که h1 پیش از دریافت پاسخ می فرستد:

کلا h1 هر بسته icmp را در دوره زمانی ثابت می فرستد (تقریبا هر یک ثانیه، یک بار ping می کند). حال اگر به هر دلیلی، تاخیر یک ping از یک ثانیه بیشتر شود، بسته های بعدی پیش از دریافت پاسخِ بسته یِ اول فرستاده می شوند.

در شکل 49 ، تاخیر ping کمتر از یک ثانیه است. درنتیجه، پاسخ هر بسته ابتدا دریافت شده و سپس، بسته بعدی فرستاده می شود. ولی در شکل 410، تاخیر ping حدود ۳ ثانیه است. درنتیجه، سه بسته پیش از دریافت اولین پاسخ فرستاده می شوند.

**بخش ۵:**

← اختلاف این دو زمان، نشان دهنده **تاخیر انتقال** بسته بزرگتر است (از تاخیر انتقال بسته کوچک صرف نظر کردیم)؛ چون تاخیر انتشار در هر دو ping وجود داشته و حذف می شود و تاخیر صف هم به دلیل بالا بودن پهنای باند ها اتفاق نمی افتد.

← توپولوژی این سوال را می توانید در فایل Q5.py چک کنید. پس از run کردن این توپولوژی در mininet فایل SimpleHTTPServer را (که یکی از مثال های آماده ی mininet است) روی هر ۴ میزبان اجرا کردیم.

سپس، h1 را مشتری در نظر گرفته و هر ۴ میزبان (از جمله خود h1) را ping کردیم. از آنجایی که در این توپولوژی برای لینک ها، تاخیر انتشار درنظر نگرفتیم، تاخیر انتشار آنها صفر است و هر تاخیری که می بینیم صرفا مربوط به تاخیر انتقال می باشد.

وقتی h1 خودش را ping میکند، تاخیر انتقال تقریبا صفر است؛ چراکه هیچ بسته ای عملا در لینکی منتقل نشده (شکل 51).

اما وقتی h2 را ping میکند، تاخیر انتقال کل حدود ۲۰۵ میلی ثانیه است که مرکب از $TD_{h2} + 2*TD_{s1} + TD_{h1}$ می باشد (شکل 52)

وقتی h3 یا h4 را ping می کند، تاخیر انتقال کل به شدت بالا می رود که دلیل آن، پهنای باند نسبتا کمتر لینک بین سوییچ ها می باشد (شکل 53 و 54)

نتایج کامل این آزمایش را می توانید در فایل Q2.txt و عکس های 51 تا 59 مشاهده کنید.

همین آزمایش را با h3 بعنوان مشتری تکرار کرده و نتایج مشابهی گرفتیم (شکل های 55 تا 59).

← همین کار ها با نظارت wireshark نیز انجام دادیم. درتصویر 510 می بینیم که با ping شدن h1 توسط خودش، هیچ بسته ی icmp در هیچ لینکی فرستاده نمی شود. در تصویر 511 می بینم که h1 ، h2 را ping کرده و سیر بسته ها را در شبکه مشاهده می کنیم. مسیر بسته ها طبق انتظار است، اما مدت زمانی که در wireshark برای ping شدن گرفته می شود، مطابق نتایج mininet نیست. در wireshark پنج بار ping شدن، حدود ۴ ثانیه طول کشیده، درحالی که در آزمایش قبلی، هر ping کمتر از 0.1ms زمان می گرفت و جمعا کمتر از 0.5ms می شد.

دلیل این موضوع این است که wireshark زمان واقعی ارسال/دریافت بسته ها را ثبت می کند، اما mininet زمان شبیه سازی شده با توجه به پارامتر ها را.

و چون در واقعیت، اجرای دستور های شبیه سازی overhead زمانی دارد، به جای 0.5ms ، چهارثانیه وقت صرف می شود.

چند بار دیگر هم آزمایش ping با wireshark را انجام دادیم که نتایج آن را در شکل های 512 تا 516 می بینید.