دانشکده‌ی مهندسی کامپیوتر

شبکه‌های کامپیوتری

نیم‌سال دوم ۰۴-۱۴۰۳

استاد: امیرمهدی صادق‌زاده

تمرین چهارم

پاسخ‌دهنده: معین آعلی - ۴۰۱۱۰۵۵۶۱

فهرست مسائل

پاسخ مسئله‌ی ۱.

الف

	$Dc(e)$	$Dc(d)$	$Dc(c)$	$Dc(b)$	$Dc(a)$
$t=۱$	∞	۰	∞	∞	∞
$t=۲$	∞	۲	۰	۶	∞
$t=۳$	۵	۲	۰	۵	۷
$t=۴$	۵	۲	۰	۵	۶

ب

در پروتکل‌های مسیریابی بردار فاصله، مشکل شمارش تا بی‌نهایت تنها زمانی رخ می‌دهد که هزینه یک لینک افزایش یابد یا لینکی به طور کامل قطع شود. در شرایط دیگر، این مشکل بروز نمی‌کند.

حالت اول: کاهش هزینه یک لینک

گره‌های مجاور به سرعت هزینه جدید و کمتر را از طریق تبادل اطلاعات دریافت می‌کنند. این خبر خوب به سرعت در شبکه منتشر می‌شود. هر گره، با دریافت آپدیت جدید، جدول مسیریابی خود را به‌روزرسانی کرده و مسیر بهینه‌تر را انتخاب می‌کند. این فرآیند منجر به همگرایی سریع شبکه به یک حالت پایدار جدید می‌شود و لوپ بی‌نهایت ایجاد نمی‌شود.

حالت دوم: اتصال دو گره جدید

برقراری یک لینک جدید، مسیرهای جدید و کوتاه‌تری را به شبکه معرفی می‌کند. این خبر خوب توسط گره‌های دو سر لینک به سرعت در شبکه منتشر می‌شود. سایر گره‌ها با دریافت این اطلاعات، جداول مسیریابی خود را به‌روزرسانی کرده و مسیرهای بهینه‌تر جدید را کشف می‌کنند. این فرآیند نیز منجر به همگرایی سریع شده و از ایجاد لوپ بی‌نهایت جلوگیری می‌کند.

ج

الگوریتم‌های متمرکز (Centralized)

- ویژگی‌ها: در این مدل، تمام اطلاعات توپولوژی و وضعیت لینک‌ها به یک گره مرکزی (کنترل‌کننده مسیر) ارسال می‌شود. این گره دیدی کامل و سراسری از شبکه دارد و تمام مسیرها را از این دیدگاه جامع محاسبه می‌کند.
- مزایا و معایب: مزیت اصلی، توانایی محاسبه مسیرهای کاملاً بهینه است. اما این ساختار یک Single Point of Failure ایجاد می‌کند؛ اگر کنترلر از کار بیفتد، کل فرآیند مسیریابی مختل می‌شود. همچنین با بزرگ شدن شبکه، با چالش‌های مقیاس‌پذیری روبرو می‌شود.
- مثال: معماری شبکه‌های نرم‌افزارمحور (SDN) که در آن یک کنترلر مرکزی مسئولیت تصمیم‌گیری و مدیریت جریان ترافیک را بر عهده دارد.

الگوریتم‌های توزیع‌شده (Distributed)

- ویژگی‌ها: منطق مسیریابی بین تمام روترها تقسیم می‌شود. هر گره مسیرهای خود را بر اساس اطلاعات محلی و داده‌هایی که به صورت تکرارشونده از همسایگان مستقیم خود دریافت می‌کند، محاسبه می‌نماید. هیچ گرهی نقشه کامل شبکه را در اختیار ندارد.
- مزایا و معایب: این رویکرد به دلیل نداشتن نقطه شکست واحد، بسیار **مقاوم و مقیاس‌پذیر** است. با این حال، چون تصمیمات با اطلاعات ناقص گرفته می‌شود، فرآیند همگرایی پس از تغییرات شبکه کندتر است و ممکن است به مشکلات موقتی مانند حلقه‌های مسیریابی منجر شود.
- مثال: پروتکل‌های خانواده بردار فاصله (Distance-Vector) مانند RIP و پروتکل‌های Link-State مانند OSPF که در آن هر روتر به طور مستقل الگوریتم دایجسترا را اجرا می‌کند.

پاسخ مسئله‌ی ۲.

الف

مرحله	N'	$D(t)$	$D(u)$	$D(v)$	$D(w)$	$D(x)$	$D(y)$	$D(z)$
۰	$\{\}$	۰	∞	∞	∞	∞	∞	∞
۱	$\{t\}$	۰	۲	۴	∞	∞	۷	∞
۲	$\{t,u\}$	۰	۲	۴	۵	∞	۷	∞
۳	$\{t,u,v\}$	۰	۲	۴	۵	۷	۷	∞
۴	$\{t,u,v,w\}$	۰	۲	۴	۵	۷	۷	∞
۵	$\{t,u,v,w,y\}$	۰	۲	۴	۵	۷	۷	۱۹
۶	$\{t,u,v,w,y,x\}$	۰	۲	۴	۵	۷	۷	۱۵
۷	$\{t,u,v,w,y,x,z\}$	۰	۲	۴	۵	۷	۷	۱۵

ب

مرحله	N'	$D(t)$	$D(u)$	$D(v)$	$D(w)$	$D(x)$	$D(y)$	$D(z)$
۰	$\{\}$	∞	۰	∞	∞	∞	∞	∞
۱	$\{u\}$	۲	۰	۳	۳	∞	∞	∞
۲	$\{u,t\}$	۲	۰	۳	۳	∞	۹	∞
۳	$\{u,t,v\}$	۲	۰	۳	۳	۶	۹	∞
۴	$\{u,t,v,w\}$	۲	۰	۳	۳	۶	۹	∞
۵	$\{u,t,v,w,x\}$	۲	۰	۳	۳	۶	۹	۱۴
۶	$\{u,t,v,w,x,y\}$	۲	۰	۳	۳	۶	۹	۱۴
۷	$\{u,t,v,w,x,y,z\}$	۲	۰	۳	۳	۶	۹	۱۴

پاسخ مسئله‌ی ۳.

الف

اینترنت به دلیل مقیاس عظیم و نیاز به استقلال مدیریتی، به یک ساختار سلسله‌مراتبی از سیستم‌های خودمختار تقسیم شده است. این تفکیک، ایجاب می‌کند که از دو نوع پروتکل مسیریابی با اهداف کاملاً متفاوت استفاده شود:

- مقیاس‌پذیری و کنترل سربار:
یک پروتکل واحد برای کل اینترنت غیرممکن است. اگر تمام روترهای جهان در یک دامنه مسیریابی قرار داشتند، حجم اطلاعات مسیریابی (تغییرات لینک، آپدیت‌ها) به حدی زیاد می‌شد که هیچ روتری توان پردازش آن را نداشت و شبکه به دلیل سربار ارتباطی فلج می‌شد. تقسیم شبکه به AS ها باعث می‌شود که پروتکل‌های درون ناحیه‌ای IGP تنها اطلاعات مربوط به شبکه داخلی خود را مدیریت کنند و از انتشار جزئیات غیرضروری به کل اینترنت جلوگیری نمایند.
- استقلال مدیریتی و اعمال سیاست:
هر AS توسط یک سازمان مستقل مدیریت می‌شود و می‌خواهد کنترل کاملی بر ترافیک ورودی و خروجی خود داشته باشد. پروتکل‌های بین ناحیه‌ای، برای اعمال سیاست‌های پیچیده طراحی شده‌اند. این سیاست‌ها می‌توانند بر اساس توافقات تجاری، مسائل امنیتی یا هزینه‌ها باشند. در مقابل، پروتکل‌های درون ناحیه‌ای، هیچ ابزاری برای درک یا اعمال چنین سیاست‌هایی ندارند.
- اهداف متفاوت بهینه‌سازی:
هدف در مسیریابی درون ناحیه‌ای، معمولاً یک هدف فنی است: یافتن سریع‌ترین یا کوتاه‌ترین مسیر در داخل یک شبکه. اما در سطح بین ناحیه‌ای، بهترین مسیر لزوماً سریع‌ترین مسیر نیست. بهترین مسیر ممکن است مسیری باشد که هزینه کمتری دارد، از طریق یک شریک تجاری معتبر عبور می‌کند، یا از عبور از شبکه یک رقیب اجتناب می‌کند. هدف در اینجا بهینه‌سازی بر اساس سیاست است، نه عملکرد فنی.

ب

خیر، یک روتر BGP همیشه مسیری با کوتاه‌ترین طول AS-PATH را انتخاب نمی‌کند.
هدف اصلی BGP یافتن سریع‌ترین یا کوتاه‌ترین مسیر نیست، بلکه اعمال سیاست‌های مدیریتی و تجاری است. برای این منظور، BGP از یک الگوریتم تصمیم‌گیری مرحله به مرحله استفاده می‌کند که در آن چندین ویژگی به ترتیب اولویت بررسی می‌شوند. طول مسیر AS-PATH تنها یکی از این ویژگی‌هاست و در مراحل اولیه قرار ندارد.
فرآیند کلی انتخاب مسیر به صورت زیر است:

۱. بررسی ویژگی‌های با اولویت بالا: روتر ابتدا ویژگی‌هایی مانند LOCAL_PREF را بررسی می‌کند. یک مسیر با LOCAL_PREF بالاتر، صرف‌نظر از طول AS-PATH آن، همیشه برنده خواهد بود. این ویژگی ابزار اصلی برای اعمال سیاست در یک سیستم خودمختار است.

۲. بررسی طول AS-PATH : تنها در صورتی که تمام ویژگی‌های با اولویت بالاتر برای دو یا چند مسیر یکسان باشند، روتر به سراغ بررسی طول AS-PATH می‌رود و در این مرحله، مسیر با طول کوتاه‌تر را انتخاب می‌کند.

۳. سایر معیارها: اگر طول مسیرها نیز یکسان باشد، معیارهای دیگری مانند نوع مسیر و... برای شکستن تساوی به کار می‌روند.

اطمینان از بدون حلقه بودن مسیرها در BGP به صورت مستقل و پیش از فرآیند انتخاب مسیر انجام می‌شود. یک روتر هر آپدیت مسیری را که دریافت می‌کند، بررسی کرده و اگر شماره AS خود را در لیست AS-PATH آن مسیر

ببیند، آن را به طور کامل نادیده می‌گیرد تا از ایجاد حلقه جلوگیری کند. بنابراین، تمام مسیرهایی که وارد فرآیند انتخاب مسیر می‌شوند، از قبل تضمین شده است که بدون حلقه هستند.

نتیجه‌گیری: مسیر انتخابی توسط BGP همواره بدون حلقه است، اما به دلیل اولویت بالاتر سیاست‌ها، این مسیر لزوماً کوتاه‌ترین مسیر از نظر تعداد ASها نیست.

ج

هر روتر بر اساس موقعیت خود در شبکه (داخلی یا مرزی) و نوع پیشوند (داخلی یا خارجی)، از پروتکل متفاوتی برای یادگیری مسیر استفاده می‌کند.

- روتر 3a:
  این روتر با eBGP از روتر 4c یاد می‌گیرد.
- روتر 1c:
  این روتر با eBGP از روتر 4a یاد می‌گیرد.
- روتر 3c:
  این روتر با iBGP از روتر 3a یاد می‌گیرد.

د

رابط روی **I2** تنظیم خواهد شد.

این انتخاب صورت می‌گیرد زیرا الگوریتم BGP در AS1، پس از بررسی سیاست‌های محلی، مسیر از طریق AS3 را به دلیل داشتن طول مسیر AS-PATH کوتاه‌تر یا دیگر ویژگی‌های برتر، بهینه تشخیص داده است.

تصمیم‌گیری بین دو مسیر دریافتی برای پیشوند x (یکی از سمت AS2 و دیگری از سمت AS3)، کاملاً به الگوریتم انتخاب مسیر BGP در AS1 بستگی دارد. این فرآیند سلسله‌مراتبی است:

۱. اولویت با سیاست است: ابتدا، BGP ویژگی‌های سیاستی مانند LOCAL_PREF را بررسی می‌کند. مدیر شبکه AS1 می‌تواند با تنظیم LOCAL_PREF بالاتر برای مسیری که از AS3 می‌آید، تمام روترهای داخلی از جمله d1 را مجبور کند که رابط I2 را انتخاب نمایند، حتی اگر آن مسیر طولانی‌تر باشد.

۲. طول مسیر به عنوان معیار دوم: تنها در صورتی که مقادیر LOCAL_PREF برای هر دو مسیر یکسان باشند، BGP به سراغ معیار بعدی، یعنی طول AS-PATH، می‌رود. در این حالت، هر مسیری که تعداد ASهای کمتری در مسیر خود داشته باشد، به عنوان مسیر بهینه انتخاب خواهد شد.

بنابراین، انتخاب نهایی به تنظیمات و سیاست‌های خاصی که در AS1 پیکربندی شده است، بستگی دارد و انتخاب I2 به معنای برتری آن مسیر بر اساس این معیارهاست.

پاسخ مسئله‌ی ۴.

الف

در پروتکل OpenFlow ، هر ورودی جریان (Flow Entry) مجموعه‌ای از پارامترها و قوانین است که تعیین می‌کند چگونه یک سوئیچ با بسته‌های دریافتی رفتار کند. این ورودی‌ها در جدول‌های جریان (Flow Tables) نگهداری می‌شوند و به عنوان هسته تصمیم‌گیری سوئیچ‌های OpenFlow عمل می‌کنند. ساختار یک ورودی جریان شامل سه بخش اصلی زیر است:

- **Match-Fields**: این فیلدها مشخص می‌کنند که چه ویژگی‌هایی از بسته باید بررسی شود تا مشخص شود که آیا بسته با این جریان مطابقت دارد یا خیر. این ویژگی‌ها می‌توانند شامل آدرس MAC، آدرس IP، شماره پورت، پروتکل لایه انتقال و یا شماره پورت ورودی سوئیچ باشند. این بخش نقش کلیدی در شناسایی جریان‌ها و دسته‌بندی ترافیک ایفا می‌کند.
- **Counters**: این بخش شامل اطلاعات آماری مانند تعداد بسته‌ها یا حجم داده‌هایی است که با این جریان تطابق داشته‌اند. شمارنده‌ها برای پایش عملکرد شبکه و تحلیل ترافیک به کار می‌روند و می‌توانند به کنترلر یا مدیر شبکه در تصمیم‌گیری برای بهینه‌سازی سیاست‌ها کمک کنند.
- **Instructions**: این بخش مشخص می‌سازد که در صورت مطابقت بسته با این جریان، چه اقداماتی باید انجام گیرد. دستورالعمل‌ها می‌توانند شامل ارسال بسته به خروجی خاص، هدایت به جدول جریان دیگر، تغییر برخی از فیلدهای بسته، یا حتی ارسال آن به کنترلر باشند. این بخش، مسئول اجرای سیاست‌های شبکه بر اساس تحلیل و تصمیم‌گیری قبلی است.

ب

در معماری شبکه SDN ، کنترل‌کننده مرکزی دارای ساختاری چندلایه است که هر لایه وظایف خاصی را برای مدیریت هوشمندانه شبکه بر عهده دارد. سه لایه‌ی مهم آن عبارتند از:

۱. **لایه ارتباطات:**
این لایه وظیفه برقراری ارتباط مستقیم با تجهیزات صفحه داده، مانند سوئیچ‌ها و روترها را دارد. از طریق پروتکل‌هایی نظیر OpenFlow ، این لایه اطلاعات کنترلی (مانند قوانین ارسال بسته‌ها) را به دستگاه‌های زیرساخت ارسال کرده و اطلاعات وضعیت (مانند جریان‌های ترافیکی و هشدارها) را از آن‌ها دریافت می‌کند. به‌طور خلاصه، این لایه پل ارتباطی میان کنترل‌کننده و شبکه فیزیکی است.

۲. **لایه مدیریت وضعیت سراسری شبکه:**
این بخش به جمع‌آوری داده‌های شبکه، مانند توپولوژی، وضعیت لینک‌ها، جدول‌های جریان و آمار ترافیکی می‌پردازد. هدف آن ایجاد یک تصویر جامع و به‌روز از وضعیت کل شبکه است. اطلاعاتی که در این لایه ذخیره و پردازش می‌شوند، پایه تصمیم‌گیری برای برنامه‌های کنترلی در لایه‌های بالاتر را فراهم می‌کنند.

۳. **لایه برنامه‌های کنترلی شبکه:**
در این لایه، برنامه‌های کاربردی شبکه اجرا می‌شوند که وظیفه آن‌ها تحلیل وضعیت شبکه و اعمال سیاست‌های مدیریتی است. این برنامه‌ها با استفاده از API هایی که کنترل‌کننده در اختیارشان قرار می‌دهد، می‌توانند عملکرد شبکه را در زمینه‌هایی مانند مسیریابی پویا، مدیریت پهنای باند، امنیت یا کیفیت خدمات بهینه‌سازی کنند. در واقع، این لایه هوش شبکه را تشکیل می‌دهد.

این سه لایه به صورت یکپارچه عمل کرده و به کنترل‌کننده SDN اجازه می‌دهند تا دیدی متمرکز و قابل برنامه‌ریزی از کل شبکه داشته باشد.

ج

در معماری SDN ، کنترلر مرکزی قادر است با استفاده از پروتکل OpenFlow سیاست‌های امنیتی مورد نظر را روی شبکه پیاده‌سازی کند. این کار از طریق تعامل مستقیم با سوئیچ‌های صفحه داده و استفاده از پیام‌های کنترلی مختلف صورت می‌گیرد. فرآیند کلی اجرای یک سیاست امنیتی برای جریان‌های ناشناس جدید به شرح زیر است:

۱. **دریافت اولین بسته توسط سوئیچ:**
زمانی‌که یک بسته‌ی جدید که مربوط به جریانی ناشناخته است وارد سوئیچ می‌شود، سوئیچ نمی‌تواند آن را با هیچ‌کدام از ورودی‌های جریان خود تطبیق دهد. در نتیجه، بر اساس تنظیمات پیش‌فرض، سوئیچ این بسته را در قالب یک پیام Packet-In به کنترلر ارسال می‌کند. این پیام حاوی اطلاعات کلیدی مانند سربرگ بسته، شماره پورت ورودی و مشخصات جریان است.

۲. **تحلیل و تصمیم‌گیری کنترلر:**
کنترلر پس از دریافت پیام Packet-In ، با توجه به سیاست امنیتی تعریف‌شده (برای مثال بررسی لیست‌های مجاز یا غیرمجاز، قوانین فایروال، یا تحلیل ترافیک)، اقدام به تحلیل اطلاعات موجود در بسته می‌کند. سپس تصمیم می‌گیرد که آیا جریان جدید باید مجاز شناخته شود یا مسدود گردد.

۳. **ارسال پیام FlowMod :**
اگر کنترلر تصمیم به پذیرش جریان بگیرد، یک پیام FlowMod به سوئیچ ارسال می‌کند. این پیام شامل ورودی جدیدی برای جدول جریان سوئیچ است که مشخص می‌کند بسته‌های مشابه در آینده چگونه پردازش شوند (مثلاً به کدام پورت ارسال یا حذف شوند).

۴. **ارسال Packet-Out :**
برای جلوگیری از تأخیر در پردازش اولین بسته، کنترلر ممکن است علاوه بر FlowMod ، یک پیام Packet-Out نیز ارسال کند تا همان بسته‌ای که ابتدا دریافت شده بود، فوراً به مسیر مقصد هدایت شود. این کار باعث می‌شود جریان از همان ابتدا بدون وقفه پاسخ بگیرد.

۵. **اعمال سیاست توسط سوئیچ:**
پس از دریافت FlowMod ، سوئیچ ورودی جدید را در جدول خود ذخیره می‌کند. از این پس، بسته‌های بعدی که متعلق به همان جریان باشند، بدون نیاز به تماس با کنترلر، مستقیماً و مطابق با سیاست اعمال‌شده هدایت می‌شوند.

در مجموع، این تعامل مبتنی بر پیام‌های OpenFlow به کنترلر اجازه می‌دهد تا کنترل دقیق و متمرکزی بر امنیت و مدیریت جریان‌های داده در شبکه داشته باشد، بدون آنکه به صورت مداوم درگیر پردازش هر بسته شود.

د

در معماری کنترلر OpenDaylight ، لایه‌ای با نام لایه انتزاع سرویس وجود دارد که نقش کلیدی در ساده‌سازی تعامل میان برنامه‌های کاربردی و پروتکل‌های زیرساختی ایفا می‌کند.

هدف اصلی این لایه، ایجاد یک واسط انتزاعی و یکنواخت برای برنامه‌های کنترلی شبکه است؛ به طوری که توسعه‌دهندگان بتوانند بدون درگیری با پیچیدگی‌های جزئیات پیاده‌سازی پروتکل‌های مختلف نظیر OpenFlow ، NETCONF یا BGP به توسعه منطق شبکه بپردازند.

این لایه مانند یک مترجم بین دو دنیا عمل می‌کند: از یک سو، درخواست‌ها و دستورات صادرشده از سوی برنامه‌های کاربردی را دریافت می‌کند و از سوی دیگر، آن‌ها را به شکل سازگار با پروتکل‌های سطح پایین تبدیل کرده و به تجهیزات شبکه ارسال می‌کند.

این ساختار ماژولار نه تنها توسعه‌ی برنامه‌ها را تسهیل می‌کند، بلکه قابلیت قابل حمل بودن کدها را افزایش داده و امکان استفاده از فناوری‌های مختلف در لایه‌های پایین‌تر را بدون نیاز به بازنویسی برنامه‌های کنترلی فراهم می‌آورد.

۵

در شبکه‌های مبتنی بر SDN ، کنترلر مرکزی نقش کلیدی در مدیریت پویای وضعیت شبکه دارد. در صورت بروز خرابی در یکی از لینک‌ها، زنجیره‌ای از تعاملات میان سوئیچ‌ها، کنترلر و برنامه‌های کنترلی رخ می‌دهد تا مسیرهای جدید تعیین شده و شبکه مجدداً به وضعیت پایدار بازگردد. این فرآیند شامل مراحل زیر است:

۱. **شناسایی خرابی توسط سوئیچ:**
هنگامی که یکی از لینک‌های متصل به یک سوئیچ دچار مشکل شود، سوئیچ با استفاده از مکانیزم‌هایی نظیر LLDP یا مانیتورینگ لایه فیزیکی، خرابی را تشخیص می‌دهد.

۲. **اعلان رویداد به کنترلر:**
سوئیچ با ارسال یک پیام به کنترلر، وقوع خرابی را گزارش می‌دهد. این پیام شامل اطلاعاتی در مورد پورت غیرفعال‌شده و وضعیت آن است.

۳. **به‌روزرسانی وضعیت توپولوژی:**
کنترلر، در لایه مدیریت وضعیت شبکه، وضعیت توپولوژی را به‌روزرسانی می‌کند و لینک معیوب را از نقشه توپولوژی خود حذف می‌نماید. در صورت وجود داده‌های آماری، ممکن است این مرحله شامل بررسی نرخ از دست رفتن بسته نیز باشد.

۴. **اطلاع‌رسانی به برنامه‌های کنترلی:**
تغییر توپولوژی به اطلاع برنامه‌هایی مانند **مسیریابی دینامیک**، تعادل بار یا کنترل کیفیت خدمات می‌رسد. این برنامه‌ها از وضعیت جدید شبکه مطلع شده و وارد عمل می‌شوند.

۵. **محاسبه مسیرهای جایگزین:**
برنامه‌های کنترلی با استفاده از توپولوژی به‌روز شده، مسیرهای جدیدی برای جریان‌های فعلی که از لینک آسیب‌دیده عبور می‌کردند محاسبه می‌کنند. این مسیرها ممکن است بر مبنای معیارهایی چون کمترین تأخیر یا کمترین بار تعیین شوند.

۶. **اعمال تغییرات در سوئیچ‌ها:**
کنترلر با استفاده از پیام‌های FlowMod ، ورودی‌های جریان جدید را به سوئیچ‌های مربوطه ارسال می‌کند تا بسته‌ها از مسیرهای تازه هدایت شوند.

۷. **حذف قوانین قبلی:**
در کنار ارسال قوانین جدید، کنترلر می‌تواند پیام‌هایی برای حذف قوانین قدیمی مرتبط با مسیر قبلی نیز ارسال کند تا از بار اضافی بر روی سوئیچ‌ها جلوگیری شود.

۸. **بازگشت به وضعیت پایدار:**
پس از به‌روزرسانی ورودی‌های جریان در سوئیچ‌ها، مسیرهای جدید فعال شده و ترافیک مجدداً به صورت روان منتقل می‌شود. بدین ترتیب، شبکه به وضعیت پایدار جدیدی بازمی‌گردد.

نقش اجزای مختلف در این فرآیند:

- **سوئیچ‌ها:** وظیفه تشخیص خطای لینک و ارسال گزارش به کنترلر، و اجرای قوانین جدید از سوی کنترلر را برعهده دارند.
- **کنترلر (لایه ارتباطات و وضعیت):** دریافت پیام‌های رویداد، به‌روزرسانی توپولوژی شبکه، و ارسال Flow-Mod به سوئیچ‌ها را مدیریت می‌کند.
- **برنامه‌های کنترلی شبکه:** بر مبنای توپولوژی جدید، تصمیم‌های مدیریتی (مانند بازتنظیم مسیرها) اتخاذ می‌کنند و به کنترلر فرمان می‌دهند.

پاسخ مسئله‌ی ۵.

****

پاسخ مسئله‌ی ۶.

تحقیق و مبانی نظری ICMP

پروتکل ICMP یک پروتکل در لایه شبکه (لایه سوم مدل OSI) است که برای ارسال پیام‌های خطا، کنترل و اطلاعات بین دستگاه‌ها در شبکه استفاده می‌شود. ICMP به کمک پیام‌هایی که تولید می‌کند، به دستگاه‌ها اطلاع می‌دهد که آیا بسته‌های داده به مقصد رسیده‌اند یا مشکلی در مسیر وجود دارد.

پیام‌های Echo Request و Echo Reply

- Request Echo : پیامی است که یک دستگاه به دستگاه دیگر در شبکه می‌فرستد تا از در دسترس بودن آن اطمینان حاصل کند. این پیام همان درخواست پاسخ است.
- Reply Echo : پاسخی است که دستگاه مقصد در جواب Echo Request می‌فرستد تا نشان دهد که فعال و در دسترس است.

این دو پیام اساس ابزار ping هستند که برای تست اتصال شبکه استفاده می‌شود.

Type و Code در ICMP

هر پیام ICMP شامل دو فیلد مهم است:

- Type : نوع پیام را مشخص می‌کند (مثلاً Echo Request یا Echo Reply).
- Code : جزئیات بیشتری درباره نوع پیام ارائه می‌دهد.

برای پیام‌های Echo Request مقدار Type=8 و Code=0 است، همچنین برای پیام‌های Echo Reply مقدار Type=0 و Code=0 است.

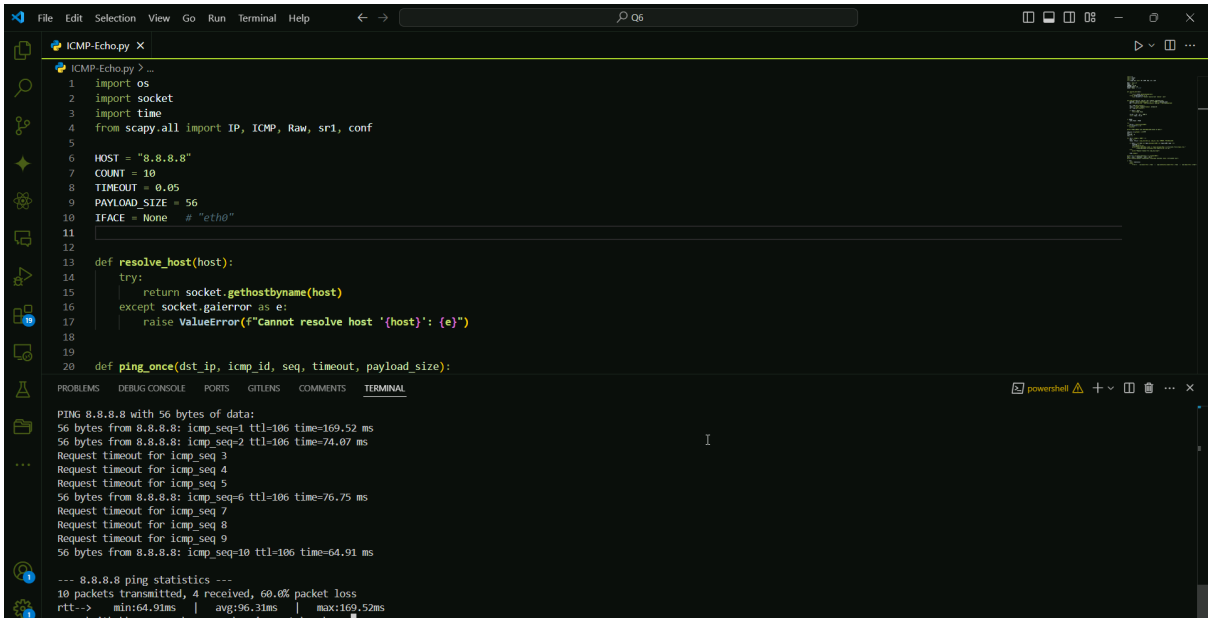
جدول مهم‌ترین Type و Code های ICMP

Type	Code	توضیح
۰	۰	Reply Echo (پاسخ به پینگ)
۳	۰-۱۵	Unreachable Destination (مقصد در دسترس نیست)
۵	۰-۳	Message Redirect (تغییر مسیر)
۸	۰	Request Echo (درخواست پینگ)
۹	۰	Advertisement Router
۱۰	۰	Solicitation Router
۱۱	۰-۱	Exceeded Time (زمان سپری شد)
۱۲	۰-۲	Problem Parameter (مشکل پارامترها)

پیاده‌سازی عملیاتی

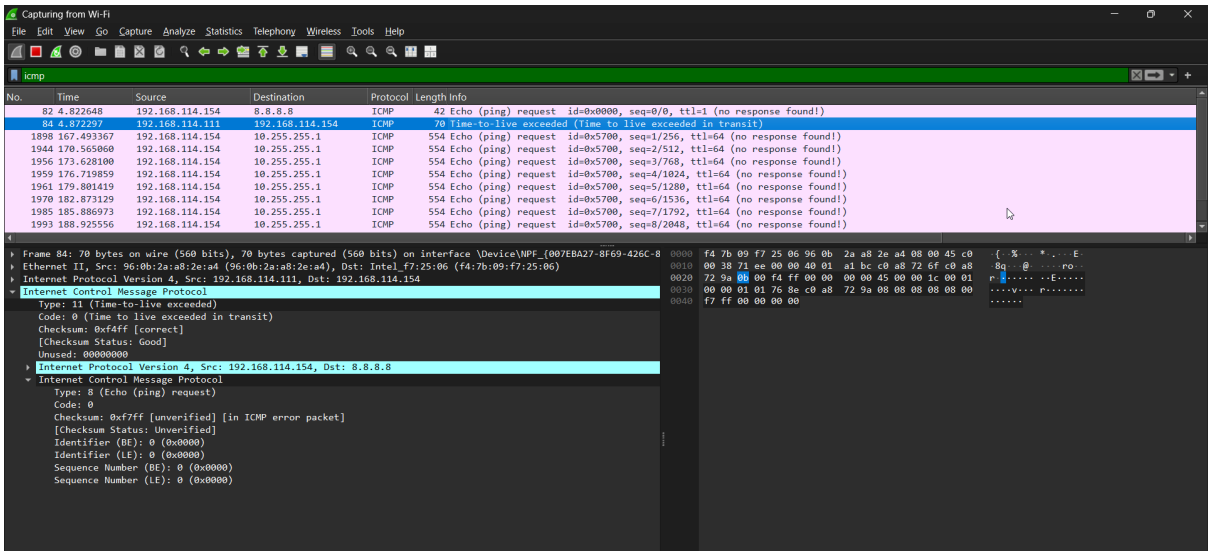
کد پیاده‌سازی شده داخل فایل ICMP-Echo.py است. در ابتدای کد، ادرس مقصد را به عنوان مثال 8.8.8.8 ست می‌کنیم و Timeout و تعداد پکت و سایز دیتا را ست می‌کنیم.

حال برنامه را ران می‌کنیم تا بسته‌ها را ارسال کند.



پس از ارسال بسته‌ها، گزارش آماری از زمان سپری شده و تعداد بسته‌های سالم پرینت می‌شود که در تصویر فوق قابل مشاهده است.

برای تست کردن خطاهای ICMP به صورت هاردکد TTL را کم در نظر می‌گیریم تا خطای تایپ ۱۱ بگیریم. خطا در Wireshark به این صورت نشان داده شده است:



در ادامه توابع مهم و جزئیات پیاده‌سازی آن را مشاهده می‌کنیم:

```python
from scapy.all import IP, ICMP, Raw, sr1, conf

def ping_once(dst_ip, icmp_id, seq, timeout, payload_size):
    payload = bytes((payload_size // 4) * b"PING")[:payload_size]
    pkt = IP(dst=dst_ip) / ICMP(id=icmp_id, seq=seq) / Raw(load=payload)

    t1 = time.perf_counter()
    reply = sr1(pkt, timeout=timeout, verbose=0)
    t2 = time.perf_counter()

    if reply is None:
        return None, None

    rtt_ms = (t2 - t1) * 1000.0
    return reply, rtt_ms
```

تابع فوق که از متدهای اماده کتابخانه scapy استفاده کرده است، یک پینگ به ادرس داده شده ارسال می‌کند و RTT آن را برمی‌گرداند.

```python
try:
    dst_ip = resolve_host(HOST)
except ValueError as e:
    print(e)

print(f"PING {HOST} with {PAYLOAD_SIZE} bytes of data:")

icmp_id = os.getpid() & 0xFFFF
sent = 0
received = 0
rtts = []

for seq in range(1, COUNT + 1):
    sent += 1
    reply, rtt_ms = ping_once(dst_ip, icmp_id, seq, TIMEOUT, PAYLOAD_SIZE)

    if reply is not None and reply.haslayer(ICMP) and reply[ICMP].type == 0:
        received += 1
        rtts.append(rtt_ms)
        print(f"{len(reply[Raw].load) if reply.haslayer(Raw) else 0} bytes from {reply.src}: "
              f"icmp_seq={seq} ttl={reply.ttl} time={rtt_ms:.2f} ms")
    else:
        print(f"Request timeout for icmp_seq {seq}")

    time.sleep(1)

print("\n--- {} ping statistics ---".format(HOST))
loss = (sent - received) / sent * 100.0
print(f"{sent} packets transmitted, {received} received, {loss:.1f}% packet loss")

if rtts:
    import statistics
    print(
        f"rtt-->    min:{min(rtts):.2f}ms   |   avg:{statistics.mean(rtts):.2f}ms   |   max:{max(rtts):.2f}ms")
```

بخش اصلی اسکریپت ما به صورت فوق است. به تعداد در نظر گرفته شده پینگ می‌کند و RTT ها را ذخیره می‌کند و در نهایت نتایج را پرینت می‌کند.

پیاده‌سازی برنامه Traceroute

****

سوال اول - چطور روترها شناسایی می‌شوند؟

وقتی TTL بسته شما به صفر می‌رسد، روتر بسته را دور می‌اندازد و یک ICMP Time Exceeded برای مبدا می‌فرستد. داخل این پیام، هدر IP + ۸ بایت اول از دیتای بسته اصلی وجود دارد. از روی این اطلاعات، مبدا می‌فهمد که کدام بسته در کدام روتر افتاده و بنابراین آدرس روتر میانی را چاپ می‌کند. به این ترتیب با TTL=1 روتر اول، با TTL=2 روتر دوم و ... شناسایی می‌شوند.

سوال دوم - چطور بعضی هاپ‌ها * می‌شوند؟

این اتفاق دلایل مختلفی می‌تواند داشته باشد:

- روتر پاسخ ICMP را فیلتر می‌کند یا اصلا ICMP تولید نمی‌کند.
- بسته پاسخ در مسیر برگشت گم می‌شود یا Timeout می‌خورد.
- فایروال در سیستم شما یا در مسیر، بسته را بلوکه می‌کند.
- روتر آن‌قدر مشغول است که به Traceroute پاسخ نمی‌دهد.

در نتیجه در این موضوع لزومی ندارد مشکل شبکه ما و مقصد باشد؛ بسیاری از اپراتورها عمداً پاسخ به ICMP را محدود می‌کنند.