به نام خداوند بخشنده‌ی مهربان

تمرین برنامه‌نویسی اول[1]

شبکه‌های کامپیوتری

مدرس: مهدی خرازی پاییز ۹۱

اهداف:

- آشنایی با چارچوب کاربری پرتو(PARTOV)
- آشنایی با سرآیندها و سوکت‌های خام
- آشنایی با مفاهیم ابتدایی گره‌های شبکه (NAT به طور خاص)
- آشنایی با مفاهیم مدیریت ترافیک (Token Bucket به طور خاص)

۱. مقدمه

با پیشرفت‌های فراوان در فناوری و صنعت، تعداد دستگاه‌های متصل به شبکه‌ی اینترنت روند صعودی شدیدی را طی کرد. به‌طوریکه در چند سال اخیر با اتمام آدرس‌های اینترنتی معتبر مواجه شده‌ایم. از این رو به منظور تامین نیازهای ارتباطی تعداد بسیار زیاد دستگاه‌ها چاره‌اندیشی‌هایی شده است. رایج ترین این روش‌ها استفاده از آدرس‌های اینترنتی غیر معتبر در شبکه‌های داخلی و ترجمه‌ی آن‌ها به آدرس(های) معتبر در شبکه‌ی اصلی است که به آن "ترجمه‌ی آدرس‌های شبکه" یا Network Address Translation به اختصار NAT گفته می‌شود. راه‌کار همیشگی استفاده از آدرس‌های اینترنتی نسخه‌ی ۶ که ۱۲۸ بیت برای آدرس‌های اینترنتی در نظر گرفته است، می‌باشد و امروزه در حال توسعه است (چندی پیش به طور رسمی مورد استفاده قرار گرفت).

NAT فقط بدین منظور مورد استفاده نیست. استفاده‌های امنیتی، انعطاف‌پذیری در تغییر تامین‌کننده‌ی ارتباط خارجی (بدون تغییر در ساختار داخلی شبکه)، کنترل و تغییرات آدرس‌ها و پورت‌ها و به طور کلی تغییرات شفاف از دید داخل و خارج از دیگر کاربردهای ترجمه‌ی آدرس‌هاست. در این تمرین تمرکز بر روی کاربرد اصلی یعنی تغییر آدرس‌های اینترنتی معتبر و ترجمه‌ی آن‌ها به آدرس‌های شبکه‌ی داخلی است. این وسیله به طور کلی در مرزهای شبکه‌ها و به عنوان گذرگاه (gateway) مورد استفاده است و به طور معمول در مسیریاب‌ها یا دیواره‌های آتش پیاده‌سازی می‌شود و به عنوان دستگاه جداگانه به‌کار گرفته نمی‌شود.

کاربران و کاربردهای مختلف شبکه و نیازهای آن‌ها منجر به پیدایش نیاز به مدیریت پهنای باند و ترافیک مورد استفاده، شده است. به عنوان نمونه، کاربردهای real time نیاز به کیفیت ارتباط، پهنای باند به نسبت بالا و تضمین شده دارند، در حالی که دریافت یک فایل از شبکه می‌تواند تحت پروتکل‌های کنترلی مانند TCP، و با قبول پهنای باند و کیفیت متغیر انجام گیرد. بدین منظور روش‌های مدیریت

[1] با تشکر از بهنام مومنی، مهدی احمدی‌نژاد، کامیار اللهوردی، سجاد فولادی، علی‌محمد ربانی، روزبه کتابی و رامتین رطبی

پارامترهای ارتباطی شبکه مطرح می‌شوند. در اینجا تاکید بر روی مدیریت ترافیک شبکه‌ی داخلی و تعامل آن با خارج است. یکی از روش‌های رایج برای کنترل میزان بسته‌های اینترنتی ارسالی و دریافتی استفاده از مفهوم Token Bucket است. در این روش هر میزبان داخلی با نرخ خاصی Token دریافت می‌کند و از آن برای فرستادن بسته‌هایش استفاده می‌کند. بدین ترتیب هم نمی‌تواند بیش از حد بسته ارسال کند و هم هنگامی که نیاز به استفاده از پهنای باند بیشتری دارد (به اصطلاح traffic burst)، می‌تواند نیاز خود را تامین کند و سپس به حالت عادی بازگردد.

دستگاه‌های ارتباطی شبکه که در لایه‌ی ۲ (به بالا) فعالیت می‌کنند هم تحت پروتکل‌های مختلفی در این لایه‌ی شبکه با هم کار می‌کنند. معروف ترین آنها Ethernet است که سرآیند و فیلدهای خاص خود را داراست. این لایه به‌طور ساده شده و حاوی اطلاعات مبدا و مقصد و نوع پروتکل لایه‌ی بالاتر در این تمرین مورد استفاده خواهد بود. ARP (Address Resolution Protocol) یک پروتکل در میان لایه‌ی ۲ و۳ شبکه است و هدف آن دریافت و شناسایی MAC Address (Media Access Control) گره بعدی مسیر با داشتن آدرس‌های اینترنتی است. MAC Address برای هر واسط شبکه همواره یکتا است.

در این تمرین سعی شده در ضمن آشنایی با چارچوب کاربری پرتو، بر آشنایی با مفاهیم ترجمه‌ی آدرس‌های اینترنتی تمرکز داشته باشیم. یک استخر! (pool) از آدرس‌های قابل استفاده در شبکه‌ی درونی در دسترس است که یک یا چند آدرس معتبر به این آدرس‌ها ترجمه می‌شوند (و بر عکس). این گره‌ی شبکه می‌تواند ترافیک شبکه را هم کنترل کند. برای تشکیل بسته‌های تبادلی مورد نیاز به MAC آدرس‌ها احتیاج داریم که برای دستیابی به آنها از ARP بهره می‌بریم. در ادامه جزییات مطرح می‌شوند.

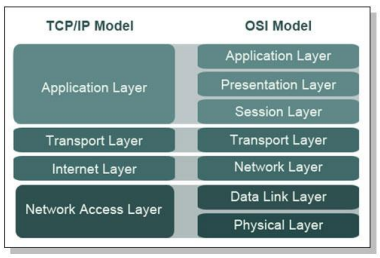
۲. آشنایی با پرتو

پرتو سامانه‌ی شبیه‌ساز شبکه‌های کامپیوتری است که در این تمرین مورد استفاده قرار می‌گیرد. کارگزار پرتو یک توپولوژی شبکه و معماری گره‌ها را برنامه‌ریزی می‌کند و کارخواه‌ها، گره‌های خاص را برنامه‌ریزی می‌کنند. بدین ترتیب کارگزار پرتو بسته‌های شبیه‌سازی شده را به کارخواه‌ها می‌رساند و بسته‌های آن‌ها را دریافت کرده و در اختیار گره‌های مجازی قرار می‌دهد. چارچوب کاربری پرتو، کتابخانه و کلاس‌های نرم افزاری از پیش نوشته شده ایست که نقش شبیه‌سازی دستگاه‌های شبکه را ایفا می‌کند و کاربر با استفاده از آنها و در محیط آن توابع، متدها و دیگر نیازهای برنامه نویسی خود را تامین می‌کند.

در این تمرین، مترجم آدرس‌ها و کنترل کننده‌ی ترافیک بصورت مجتمع به عنوان یک گره‌ی مرزی در این محیط شبیه‌سازی می‌شوند. برای آشنایی بیشتر با چارچوب کاربری و نحوه‌ی استفاده از آن به مستندات مربوط مراجعه کنید.

۳. لایه بندی شبکه و پروتکل ARP

در مدل پیشنهادی OSI شبکه به ۷ لایه تقسیم بندی شده است. در مدل TCP/IP که مدل رایج شبکه‌ی اینترنت است، ۲ لایه‌ی پایینی از ۷ لایه به لایه‌ی لینک (لایه‌ی اول) در این مدل نگاشت می‌شوند.



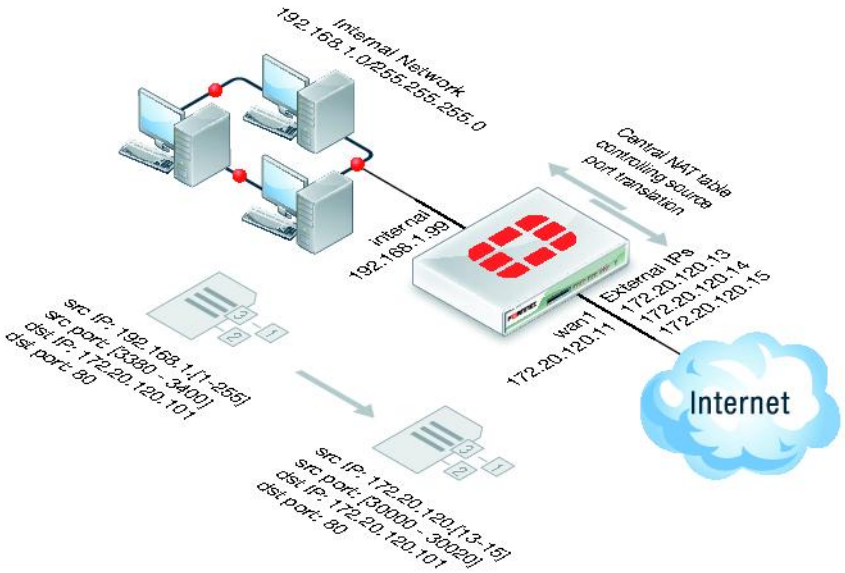
شکل ۱- انواع لایه‌بندی شبکه

ARP در میان لایه‌های ۲و۳ عمل می‌کند و به نوعی نگاشت میان آدرس اینترنتی و آدرس فیزیکی را بیان می‌کند. ترجمه‌ی آدرس‌ها در حالت کلی در لایه‌ی شبکه (لایه‌ی ۳) و لایه‌ی انتقال (لایه‌ی ۴) کار می‌کند.

کنترل ترافیک هم در این تمرین به نوعی در لایه‌ی ۳ انجام می‌شود (زیرا میزبان‌ها و محیط خارج از روی آدرس اینترنتی تمیز داده می‌شوند). در این تمرین در گره‌ی مورد نظر باید برای کشف آدرس‌های فیزیکی، پروتکل ARP را پیاده سازی کنید (باقی گره‌ها به طور خودکار درخواست می‌فرستند و پاسخ می‌دهند).

در این تمرین هدف آشنایی با سرآیندهای بسته‌ها در لایه‌های مختلف است. به طور خاص تمرکز بر روی Ethernet (ساده شده استاندارد IEEE 802.3) برای لایه‌ی ۲، بسته‌های ARP و IP در لایه‌ی ۳ و بسته‌های TCP و UDP در لایه‌ی ۴ خواهد بود. برای آشنایی با سرآیندها به پیوست با عنوان توضیحات در مورد سرآیندها مراجعه کنید. ساختارهای مورد نیاز برای کار تحت این پروتکل‌ها به شما داده می‌شود و شما فقط ملزم به پیاده‌سازی ARP هستید.

۴. ترجمه‌ی آدرس‌های اینترنتی



شکل ۲- نمای کلی کارکرد ترجمه‌ی آدرس‌های اینترنتی

در این قسمت یک مجموعه از آدرس‌های اینترنتی (نامعتبر) در اختیار داریم که آدرس‌های میزبان‌های داخلی زیرمجموعه‌ی آن هستند. همچنین یک Pool از آدرس‌های اینترنتی معتبر در اختیار خواهیم داشت. در این تمرین نگاشت از بیرون به درون ۱ به N فرض می‌شود یعنی هر آدرس اینترنتی خارجی به ۱ یا چند آدرس درونی نگاشته می‌شود اما آدرس‌های داخلی حداکثر به یک آدرس معتبر ترجمه می‌شوند. این نگاشت‌ها پایا هستند و فراموش نمی‌شوند (در هر بار اجرای شبیه‌سازی پس از اینکه نگاشت انجام شد ذخیره شده و مورد استفاده قرار می‌گیرند ولی با اجرای مجدد همه‌ی اطلاعات از میان می‌رود). امکان نگاشت دستی از پیش تعیین شده هم وجود دارد. قواعد و حالت‌های نگاشت به عنوان ورودی به برنامه داده خواهند شد. برنامه می‌بایست مطابق نگاشت‌ها، آدرس‌های مبدا و مقصد بسته‌ها را تغییر داده، IP Checksum و در صورت نیاز TCP Checksumرا در حالت جدید محاسبه کرده و در بسته قرار دهد.

در حالت پیشرفته علاوه بر آدرس‌ها پورت‌های TCP و UDP هم ترجمه می‌شوند. فرض کنید ما آدرس 1.2.3.4 (معتبر) را به آدرس 192.168.1.1 و 192.168.1.2 نگاشته ایم (سناریوی رایج منازل خود شما!) حال فرض کنید دو رایانه مرورگر فایرفاکس را اجرا کرده‌اند که از پورت ۱۲۳۴۵ به پورت ۸۰ دو کارگزار وب، درخواست دو صفحه‌ی اینترنتی را می‌دهند. حال در پاسخ کارگزار جواب را به آدرس 1.2.3.4:12345 می‌فرستد، اگر از پورتmapping استفاده نکنیم دچار ابهام می‌شویم که کدام درخواست را به کدام میزبان داخلی باید بدهیم. برای رفع این مشکل از ترجمه‌ی پورت در کنار IP استفاده می‌کنیم. بدین ترتیب که از نگاشت (IP, Port) به (IP, Port) استفاده می‌کنیم. با توجه به اینکه تعداد پورت‌های خالی زیاد است ، این راه‌کار تا حد قابل قبولی مشکل را حل می‌کند. در این تمرین با توجه به ۱ به N بودن نگاشت از این روش استفاده می‌کنیم. واضح است که پس از تغییرات در لایه‌ی انتقال (عدد پورت) TCP Checksum هم باید دوباره حساب شود (احتیاجی به اعتبار سنجی Checksum بسته‌ی دریافتی نیست، همچنین در این تمرین نیازی به محاسبه‌ی Checksum برای UDP ندارید).

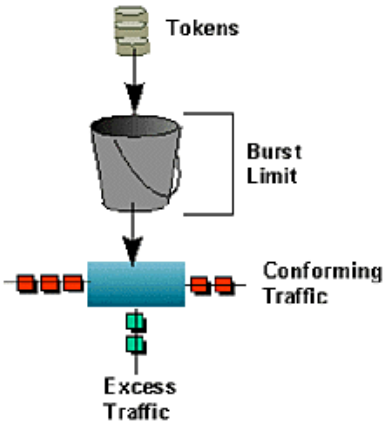
دقت کنید محاسبه‌ی Checksum از فرمول‌های خاصی پیروی می‌کند و تنها جمع ساده‌ی بایت‌ها نیست. محدوده‌ی تحت پوشش هم بین IP و TCP تفاوت دارد. در اولی فقط سرآیند و در دومی تمام بسته از شروع سرآیند TCP تا پایان بسته، مورد بررسی قرار می‌گیرد.

برای اطلاعات بیشتر به لینک RFC زیر مراجعه کنید:

http://tools.ietf.org/html/rfc2663

۵. کنترل ترافیک

در این تمرین هر میزبان داخلی، یک نرخ دریافت token و یک بیشینه‌ی ظرفیت خواهد داشت که در ابتدا به برنامه داده می‌شود (در اطلاعات دلخواه[۲] توسط کارگزار شبیه‌ساز ارسال می‌گردد). گره‌ی مورد برنامه‌ریزی عهده‌دار نگه‌داری و کنترل این اطلاعات است. در فواصل زمانی منظم مقدار مورد نظر برای هر میزبان را به آن اضافه می‌کند (از بیشینه ظرفیت بیشتر نمی‌شود) و هنگامی‌که می‌خواهد بسته ای از میزبان داخلی به خارج بفرستد با استفاده از tokenهای آن میزبان - در صورتی که دارای مقدار کافی باشد - بسته را می‌فرستد (و به تعداد بایت‌های بسته از تعداد tokenها مصرف می‌کند/می‌کاهد) در غیر این صورت با توجه به سیاست‌هایی که در ادامه توضیح داده شده عمل می‌کند. برای تمامی ترافیک ورودی به شبکه هم یک ظرف معین در نظر گرفته شده که به عنوان ورودی (از طریق custom information) داده می‌شود.

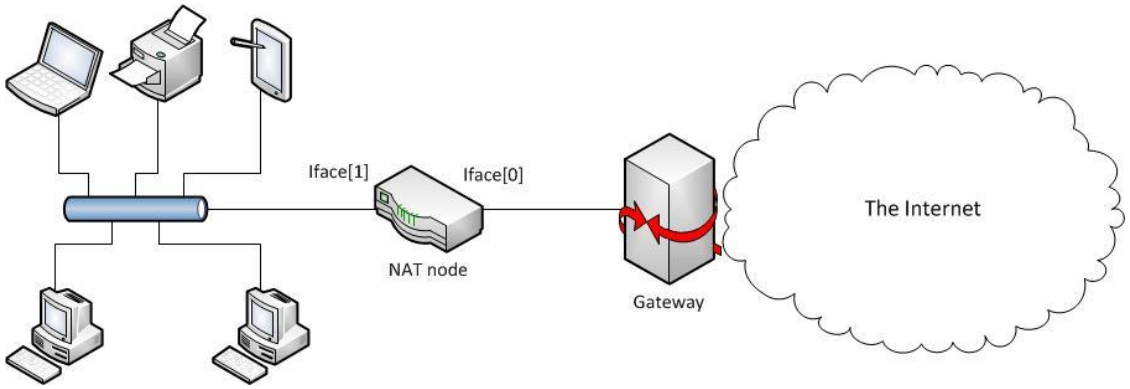


شکل ۳- نحوه‌ی کارکرد Token Bucket

سیاست‌های کنترل ترافیک به دو حالت Shaping و Policing خواهد بود. در حالت Policing در صورتی که نتوانیم بسته را بفرستیم آن را می‌اندازیم. در حالت Shaping بسته‌ها در حافظه نگه‌داری شده و هنگامی‌که token کافی برای ارسال وجود داشت ارسال می‌شوند. بسته‌ها در صف FIFO نگه‌داری می‌شوند. در این حالت بیشینه طول صف ورودی را برابر ماکرو IN_Q_LENGTH و بیشینه طول صف خروجی را OUT_Q_LENGTH در نظر بگیرید (مقدار اولیه‌ی هر کدام را به ترتیب برابر ۵۰ و ۱۰ در نظر بگیرید، این اعداد بیانگر تعداد بسته‌ها هستند). این دو حالت کارکرد به عنوان ورودی به برنامه داده می‌شوند (هنگامی‌که به تعداد tokenها می‌افزاییم اگر صف خالی نبود تا جایی که می‌توانیم بسته‌ها را می‌فرستیم).

[۲] اطلاعات دلخواه (custom information) اطلاعاتی است که توسط کارگزار شبیه‌ساز به کارخواه داده می‌شود و بسته به مواردی چون توپولوژی شبکه، تغییر می‌کند. در ادامه با چگونگی نمایش این اطلاعات بیشتر آشنا خواهید شد.

۶. توپولوژی نمونه



شکل ۴ – توپولوژی نمونه و ارتباط واسط‌ها

گره‌ای که پیاده سازی می‌کنید باید در چنین توپولوژی‌ای کاربرد داشته باشد.

۷. نحوه‌ی کارکرد برنامه

ورودی‌های برنامه به فرمت زیر داده می‌شوند:

./tfmng.out <static mapping file address> <opmode> <policy>

برای مثال:

./tfmng.out Map_sample.txt 1 shaping

./tfmng.out Map_sample.txt 0

- Static mapping file address (string, NULL for no static map file)

  آدرس فایل حاوی نگاشت‌های از پیش تعیین شده. در زیر فرمت فایل نگاشت های دستی نشان داده شده است:

Inside IP1	Inside Port1	Outside IP1	Outside Port1
Inside IP2	Inside Port2	Outside IP2	Outside Port2

- Opmode (integer, enables traffic management mode, 1=enable, 0=disable)

  نحوه‌ی کارکرد گره. در صورتی که برابر صفر باشد هیچ گونه مدیریت ترافیکی انجام نخواهد گرفت.

- policy (string, {“policing”,”shaping”}, case sensitive)

قواعد مدیریت ترافیک. فقط می تواند دو مقدار policing و یا shaping را داشته باشد که معادل سیاست‌های کنترل ترافیک - که در بالا توضیح داده شده - است. دقت کنید در صورتی که Opmode برابر ۰ باشد، policy مفهومی نداشته و نباید به برنامه داده شود.

Custom information[3]**:**

192.168.1.0/31 [Inside hosts range]

2 [Number of hosts being configured]

192.168.1.0 [host address]

50[token per second]

500[bucket capacity]

192.168.1.1 [host address]

34[token per second]

600[bucket capacity]

213.233.168.1 [GATEWAY IP ADDRESS]

110[token per second]

1400[incoming bucket capacity]

213.233.169.0/8 [CIDR subnet address for pool]

با توجه به اینکه نگاشت‌های دستی احتیاج به آگاهی از pool دارند توصیه می‌شود ابتدا اطلاعات دلخواه پردازش شود و سپس فایل نگاشت‌های دستی خوانده شود. همچنین دقت کنید ممکن است تعداد میزبان‌های داخلی که برای مدیریت ترافیک تنظیم می‌شوند از تعداد کل میزبان‌های در دسترس با توجه به محدوده‌ی آدرس کمتر باشند. در این حالت، باقی میزبان‌ها حق هیچ گونه ارسال و دریافت بسته به/ از خارج شبکه را ندارند.

۸. توصیه‌ها

نکات زیر در انجام هر چه بهتر این تمرین به شما کمک می‌کند (الزامی‌در رعایت آن **نیست**):

۱- ابزارهای مورد نیاز خود را در کنار فایل سرآیندها که به شما داده می‌شود تهیه کنید. این ابزارها شامل تابع‌ها، کلاس‌ها، ساختارها و ماکروهای مورد نیاز برای کار بر روی بسته‌های شبکه و قابلیت نمایش اطلاعات به صورت قابل فهم برای انسان است.

۲- تا جای ممکن تعداد ریسه‌ها در برنامه خود را کم نگه دارید. در زمینه‌ی همگام سازی نهایت تلاش خود را انجام دهید.

۳- خطاها، بسته‌ها و مکان flow در هنگام اجرای برنامه را چاپ نمایید. به عنوان نمونه پیغامی مبنی بر ارسال بسته با این داده‌ها به فرمت hex (به آن packet hex dump گفته می‌شود) پس از ارسال چاپ نمایید.

۴- دقت کنید بسته‌هایی که در محیط شبیه‌ساز دریافت می‌کنید مال شما نیستند![4] هم چنین به constructor و copy constructorها توجه کنید تا از مشکلات مربوط به کار با حافظه تا جای ممکن به دور باشید.

[3] اطلاعات دلخواه. این اطلاعات با استفاده از چارچوب کاربری پرتو، با استفاده از متد getCutomInformation() قابل دسترسی است.

۵- دید Event oriented داشته باشید!

۶- در هنگام استفاده از داده‌ساختارها و کتابخانه‌های زبان C++ به ویژه STL، بسیار دقت کنید. در مورد چگونگی کارکرد آنها اطمینان حاصل کنید و از ساختارهای پیچیده تا جای ممکن پرهیز کنید (اگر چندین نمونه شی را با استفاده از templateها به صورت تودرتو استفاده می‌کنید حتما دقت کنید وگرنه با مشکلات زیادی روبه‌رو خواهید شد).

۷- با توجه به اینکه گره‌ی مورد نظر در حالت‌های مختلفی کار می‌کند این حالت‌ها را از طریق متغیرهایی ذخیره کنید و با توجه به آنها جریان برنامه را کنترل کنید.

۸- به نظم نمایش اعداد توجه کنید و تا جای ممکن در یک نظم کار کنید (Byte Endianness).

۹- روند زیر برای پیاده‌سازی توصیه می‌شود (در هر مرحله از صحت مرحله‌ی قبل اطمینان حاصل کنید):

تهیه‌ی ابزارها < شناسایی ورودی و مقدمات و ایجاد داده ساختارها < ARP < NAT < Checksum < Token Bucket

۹. نکات ضروری

- در صورتی که هر مشکل یا پرسشی داشتید که فکر می‌کنید پاسخ آن برای همه مفید خواهد بود، آن را به گروه پستی درس ارسال کنید.
- از فرستادن جواب تمرین به گروه پستی خودداری کنید.
- فرستادن کل یا قسمتی از برنامه‌تان برای افراد دیگر، یا استفاده از کل یا قسمتی از برنامه‌ی فرد دیگری به نام خود، تقلب محسوب می‌شود.
- پس از اتمام کارتان لازم است که پوشه‌ی user را به همراه Makefile فشرده کرده و بر روی سیستم خودکار داوری[۵] upload کنید.

[۴] در متد processFrame() یک frame به شما داده شده که برای دریافت همه‌ی بسته‌ها به صورت مشترک مورد استفاده است. در صورتی که نیاز به نگه‌داری بسته دارید آن را کپی کنید و در حافظه‌ای که خود در اختیار دارید قرار دهید.

[5] http://partov.sharif.edu/

Ethernet Frame

Preamble	Start of frame delimiter	MAC destination	MAC source	802.1Q tag (optional)	Ethertype (Ethernet II) or length (IEEE 802.3)	Payload	Frame check sequence (32-bit CRC)	Interframe gap
7 octets	1 octet	6 octets	6 octets	(4 octets)	2 octets	42[note 2]–1500 octets	4 octets	12 octets
		64–1522 octets						
72–1530 octets								
84–1542 octets								

شکل فوق بیانگر نحوه‌ی قرارگیری سرآیند و اطلاعات از دید Ethernet است.

تنها سه فیلد زیر (به صورت پشت سر هم) به محیط کاربری داده می شود:

بیت ها	نام	توضیحات
0-47	DST ADDRESS	MAC آدرس مقصد (گره‌ی بعدی)
48-95	SRC ADDRESS	MAC آدرس مبدا (گره‌ی فعلی)
96-111	ETHERTYPE	نوع پروتکل لایه‌ی بالاتر

IP

نسخه‌ی ۴ پروتکل اینترنت مورد استفاده است.

IPv4 Header Format

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP						ECN		Total Length															
4	32	Identification																Flags			Fragment Offset											
8	64	Time To Live								Protocol								Header Checksum															
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															

برای اطلاع بیشتر از سرآیند این پروتکل به آدرس زیر مراجعه کنید:

http://en.wikipedia.org/wiki/IPv4#Packet_structure

ARP

برای ارسال بسته‌ها، لازم است که آدرس MAC برای گره‌ی مقصد را در اختیار داشته باشید (در سرآیند Ethernet باید آدرس MAC گره‌ی «بعدی» قرار گیرد؛ این درحالی است که در سرآیند IP آدرس فرستنده و گیرنده قرار دارد). در صورتی که مقصد بسته شبکه‌ی بیرونی است، این آدرس MAC، آدرس Gateway خواهد بود. برای اینکه بتوانیم از روی IP مقصد، آدرس MAC مقصد را بدست آوریم، از پروتکل ARP کمک می‌گیریم. در این پروتکل با داشتن یک آدرس IP می‌توانیم آدرس MAC هدف را بدست آوریم.

در این تمرین، برای بدست آوردن آدرس‌های MAC باید پروتکل ARP را پیاده‌سازی کنید. می‌توانید برای سادگی یک جدول از زوج‌های IP/MAC نگه‌داری نمایید و برای هر آدرس IP تنها یک‌بار از پروتکل ARP استفاده کنید.

Internet Protocol (IPv4) over Ethernet ARP packet		
bit offset	0 – 7	8 – 15
0	Hardware type (HTYPE)	
16	Protocol type (PTYPE)	
32	Hardware address length (HLEN)	Protocol address length (PLEN)
48	Operation (OPER)	
64	Sender hardware address (SHA) (first 16 bits)	
80	(next 16 bits)	
96	(last 16 bits)	
112	Sender protocol address (SPA) (first 16 bits)	
128	(last 16 bits)	
144	Target hardware address (THA) (first 16 bits)	
160	(next 16 bits)	
176	(last 16 bits)	
192	Target protocol address (TPA) (first 16 bits)	
208	(last 16 bits)	

پروتکل ARP میتنی بر یک سیستم پیام ساده است که هر پیام شامل درخواست یا پاسخ به یک درخواست می‌باشد. اندازه‌ی بسته‌ی ARP وابسته به اندازه‌ی آدرس‌های لایه‌ی پایین‌تر (مثلاً Ethernet) و لایه‌ی بالاتر (مثلاً IPv4) است. سرآیند ARP نوع این لایه‌ها و اندازه‌ی آدرس‌ها را در خود مشخص می‌کند. همچنین یک کد Operation در قسمت سرآیند قرار دارد که مشخص می‌کند بسته درخواست است یا پاسخ به درخواست.

فرمت کلی بسته‌های ARP در جدول زیر آمده است:

بیت‌ها	نام	توضیحات
0 – 15	HTYPE	این فیلد نوع پروتکل لایه‌ی پایین‌تر را مشخص می‌کند. برای نمونه در Ethernet این مقدار برابر با 1 است.
16 – 31	PTYPE	نوع پروتکل لایه‌ی بالاتر توسط این بخش مشخص می‌شود. برای IPv4 این مقدار برابر 0x8000 خواهد بود.
32 – 39	HLEN	طول آدرس لایه‌ی پایین‌تر (آدرس سخت‌افزار) را به بایت مشخص می‌کند. برای نمونه در Ethernet این مقدار برابر با 6 است.
40 – 47	PLEN	طول آدرس لایه‌ی بالاتر (که نوع آن را PTYPE معین کرده‌است). برای نمونه در IPv4 این مقدار 4 است.
48 – 63	OPER	نوع عملیات؛ مقدار ۱ برای درخواست و ۲ برای پاسخ آن.
64 – 111	SHA	آدرس سخت‌افزاری (برای نمونه آدرس MAC برای Ethernet) فرستنده.
112 – 143	SPA	آدرس لایه‌ی بالاتر فرستنده.
144 – 191	THA	آدرس سخت‌افزاری گیرنده. برای درخواست مقدار این فیلد اهمیتی ندارد (بهتر است صفر در نظر گرفته شود).
192 – 224	TPA	آدرس لایه‌ی بالاتر فرستنده. برای درخواست، مقدار این فیلد را برابر آدرسی قرار می‌دهیم که می‌خواهیم MAC آن را به دست آوریم.

دقت کنید که بسته‌ی درخواست باید در شبکه Broadcast شود؛ به این منظور باید MAC address مقصد برابر با ff:ff:ff:ff:ff:ff قرار داده شود.

برای آشنایی بیشتر به آدرس های زیر مراجعه کنید:

http://en.wikipedia.org/wiki/Address_Resolution_Protocol

http://lostintransit.se/2010/07/30/arp-address-resolution-protocol/

TCP & UDP

TCP Header

Offsets Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size														
16	128	Checksum																	Urgent pointer (if URG set)														
20 ...	160 ...	Options (if Data Offset > 5, padded at the end with "0" bytes if necessary) ...																															

برای محاسبه‌ی TCP checksum، شبه‌سرآیند مربوط در نظر گرفته می‌شود:

TCP pseudo-header for checksum computation (IPv4)

Bit offset	0–3	4–7	8–15	16–31
0	Source address			
32	Destination address			
64	Zeros		Protocol	TCP length
96	Source port			Destination port
128	Sequence number			
160	Acknowledgement number			
192	Data offset	Reserved	Flags	Window
224	Checksum			Urgent pointer
256	Options (optional)			
256/288+	Data			

همان طور که در شکل مشخص است، فیلد های زیر به ابتدای سرآیند افزوده شده‌اند:

```
IP Source Address               4 bytes
IP Destination Address          4 bytes
TCP Protocol                    2 bytes
*padded one byte zero to protocol field in IP header
TCP Length                      2 bytes
```

برای محاسبه‌ی فیلد Checksum صفر منظور می‌شود. سپس حاصل جمع تمام بایت‌های این شبه‌سرآیند به همراه داده محاسبه می‌شود. اگر طول داده فرد باشد، یک بایت صفر به عنوان بایت آخر به آن اضافه می‌شود. (این افزونه‌ها و شبه‌سرآیند در شبکه فرستاده نمی‌شوند). اگر در هنگام محاسبه حاصل جمع کلمات ۸ بیتی از ۸ بیت بیشتر شد، آن را مانند یک عدد ۱۶ بیتی در نظر می‌گیریم، سپس ۸ بیت پر ارزش را با ۸ بیت کم ارزش جمع می‌کنیم تا حاصل جمع در یک بایت جای گیرد.

Offset (bits)	Field
0	Source Port Number
16	Destination Port Number
32	Length
48	Checksum
64+	Data

UDP فقط شماره‌ی پورت مقصد و طول بسته را مشخص می‌کند. Checksum و پورت Source در UDP اختیاری است.

این سرآیند ها را در ویکیپدیا به دقت بررسی کنید:

http://en.wikipedia.org/wiki/User_Datagram_Protocol#Packet_structure

http://en.wikipedia.org/wiki/Transmission_Control_Protocol#TCP_segment_structure