هو الحق

دانشکده‌ی مهندسی کامپیوتر

CE-40443 پاییز ۱۳۹۶

شبکه‌های کامپیوتری ـ تمرین صفرم

استاد: مهدی جعفری

۱ مقدمه

هدف از این تمرین، آماده‌سازی و دست‌گرمی برای تمرین‌های آینده و آشنایی با سیستم ثبت و نمره‌دهی خودکار درس است. در این تمرین شما یک نرم‌افزار chat تحت ترمینال را به عنوان مثال ساده‌ای از برنامه‌نویسی سوکت و با استفاده از امکانات پایه‌ای کار با سوکت در زبان‌های برنامه‌نویسی پیاده‌سازی خواهید کرد. برای دانشجویانی که تجربه کار با سوکت‌ها را داشته‌باشند، حل این تمرین بسیار سرراست و سریع خواهدبود.

۲ پیش‌زمینه

هدف این بخش یادآوری مقدمات شبکه، ماهیت سوکت‌ها و مفاهیم اولیه پروتکل TCP است. در صورتی که این اطلاعات برای شما بدیهی‌است، می‌توانید این قسمت را به‌طور کامل رد کرده و از آماده‌سازی ادامه‌دهید.

۱.۲ سوکت چیست؟

سوکت‌ها را در شبکه می‌توان به صورت نقطه‌های پایانی ارسال و دریافت داده در داخل یک کامپیوتر و یا بین کامپیوترهای مختلف تعریف کرد. اکثر سیستم‌عامل‌های متداول از سوکت‌های پشتیبانی کرده و API های مشخصی را برای استفاده از آنها تعریف می‌کنند. [۱]
به کمک سوکت‌ها، کاربران و پردازه‌های [۲] درون سیستم‌عامل می‌توانند به سادگی از امکانات شبکه برای ارسال و دریافت داده استفاده کنند، بدون این که درگیر جزئیات پیچیده نحوه ارسال شبکه و پروتکل‌های سطح پایین‌تر بینابین بشوند.

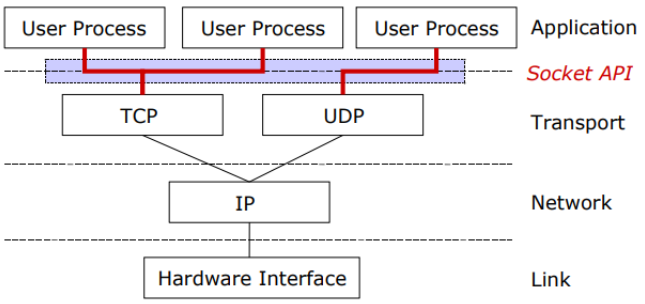
۲.۲ پارامترهای سوکت

- پروتکل
  - برای مثال، UDP TCP

Parts of the material and the graphics in this assignment are taken from Eleftherios Kosmas' CS556 slides (May 2012), Stanford's CS244A slides by Clay Collier (Jan 2007) and *Introduction to Socket Programming* intellectual property of University of California, Riverside

[۱] این API ها متشکل از فراخوانی های سیستمی (system call) لازم برای باز کردن و بستن سوکت‌ها و نوشتن و خواندن از طریق آنهاست. از این نظر API سوکت‌ها شباهت زیادی به API فایل‌سیستم دارد، با این تفاوت که در شبکه مفهوم client و server نیز بایستی لحاظ شود.

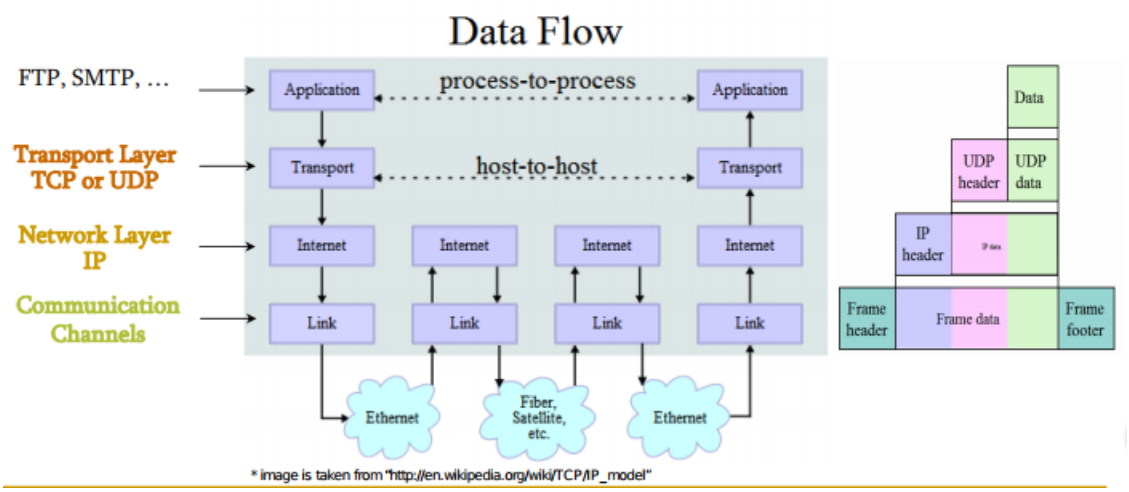
[۲] Process



- آدرس local و remote
  - مثلاً `192.168.1.102` و `127.0.0.1`
- شماره پورت local و remote
  - برای مشخص کردن برنامه‌ای که سوکت برای ارتباط با آن ایجاد می‌شود، استفاده می‌شود
  - برخی پورت‌ها به‌صورت قراردادی رزرو شده هستند (برای مثال پورت ۸۰ برای HTTP) [۳]
  - خواندن از (یا به عبارت دیگر گوش‌کردن روی) پورت‌های زیر ۱۰۲۴ در Unix نیاز به دسترسی root دارد

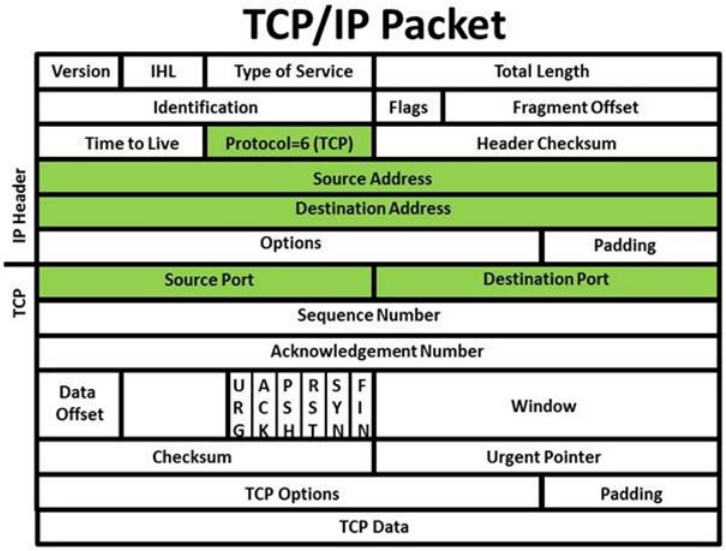
۳.۲ پروتکل TCP

شما در این تمرین از پروتکل TCP/IP استفاده خواهیدکرد. TCP/IP ، ارتباط بین دو گره از شبکه را با تعریف‌کردن نحوه ساختاردهی (format) ، آدرس‌دهی ، ارسال ، مسیریابی و دریافت بسته‌های داده میسر می‌کند. این پروتکل به صورت گسترده در وب و اینترنت استفاده می‌شود.
پروتکل‌های شبکه به صورت *لایه‌ای* سازمان‌دهی می‌شوند، به این صورت که در هر لایه، امکانات سطح بالاتر به کمک ابزارهایی که لایه پایین‌تر در اختیار می‌گذارد پیاده‌سازی می‌شود.



هرکدام از لایه‌های یادشده، داده‌ها ساختار و شکل خاص خود را دارند و معمولاً در هر لایه یک سرآیند (header) و مقداری داده (data/payload) داریم. [۴]

[۳] در سیستم‌عامل‌های unix-based می‌توانید لیست کاملی از این پورت ها را در فایل /etc/services مشاهده کنید
[۴] برای مثال کل سرآیند و داده‌های TCP در داخل داده های بسته IP قرار می‌گیرد و کل بسته IP (شامل سرآیند و داده) در داخل داده‌های فریم ethernet

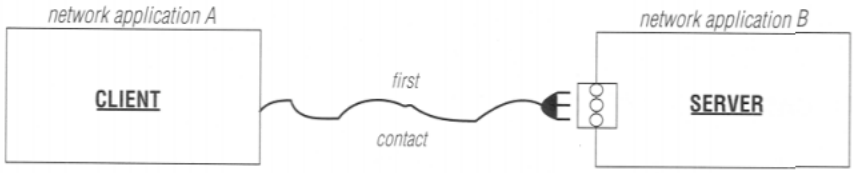


شکل ۱ : Image belongs to Frank Yue's 2012 article *Application is More than Header*

خوشبختانه اکثر زبان‌های برنامه‌نویسی سازوکار ساده‌ای برای کار کردن با سوکت‌های TCP/IP ارائه می‌دهند و تمام جزئیات پیچیده بسته‌بندی، ارسال، دریافت و سرهم‌کردن داده‌ها توسط سیستم‌عامل و کتابخانه‌های Library شبکه انجام می‌شود. بنابراین کاربر می‌تواند سوکت TCP را مانند یک جریان دوطرفه داده فرض کند.
پروتکل TCP یک پروتکل دارای connection است و از مدل client-server تبعیت می‌کند.

۴.۲ مدل client-server

اگرچه این امکان‌پذیر است که دو طرف یک ارتباط شبکه هم‌زمان شروع به مخابره کنند تا ارتباط برقرار شود، در عمل چنین سازوکاری برای بسیاری از کاربردها دشوار است. بنابراین یک ایده خوب برای طراحی سازوکار شبکه این است که دو طرف ارتباط، به صورت مکمل یکدیگر عمل کرده و عملیات خود را متوالی انجام دهند.
در مدل کارخواه‌ـ کارگزار (client-server) ، کارگزار [۵] اول اجرا شده و منتظر تماس کارخواه [۶] می‌شود. پس از آن، کارخواه اجرا شده و اولین بسته مخابره را به کارگزار ارسال می‌کند. پس از ارتباط اولیه، کارخواه و کارگزار هردو می‌توانند داده ارسال یا دریافت کنند و یا مخابره را به‌صورت یک‌طرفه یا دوطرفه پایان دهند.



A client initiates communications to a server.

(یا پروتکل دیگر سطح link استفاده‌شده) قرار می‌گیرد. در واقعیت این موضوع می‌تواند پیچیده‌تر شود. برای مثال ممکن است یک بسته TCP در چندین بسته IP ارسال شود.

[۵] Server

[۶] Client

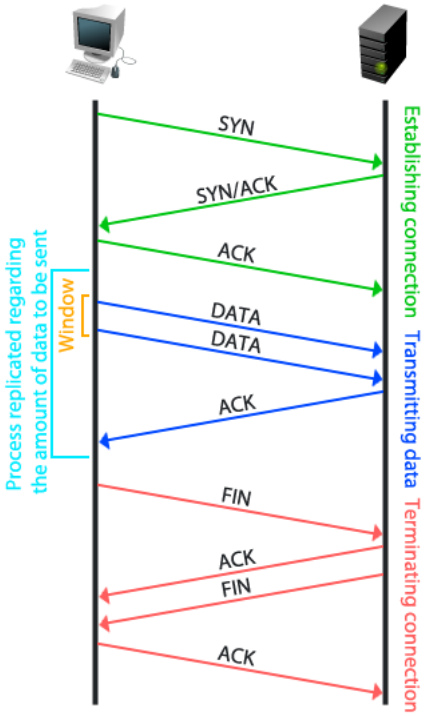
۵.۲ پروتکل connection-oriented

به‌طور کلی پروتکل‌های شبکه را می‌توان به دو نوع connection-less و connection-oriented تقسیم کرد:

- **Connection-oriented (stream)**: دارای مفهوم session هستند که می‌توان آن را هم‌ارز یک تماس تلفنی یا یک ارتباط نیمه‌مانا فرض‌کرد. یعنی ابتدا بایستی یک ارتباط بین دو طرف مخابره برقرار شود و سپس از طریق آن داده‌ها ارسال شود، به این ترتیب ارتباط دارای حالت (state) می‌باشد. در داخل یک session می‌توان با مکانیسم های مختلف از رسیدن داده‌ها به مقصد بدون تغییر و با ترتیب درست اطمینان حاصل کرد.
- **connection-less (datagram)**: فاقد مفهوم session هستند. IP و UDP از این دسته محسوب می‌شوند. فرستنده صرفاً بسته‌های داده را به مقصد ارسال می‌کند و اتصال بین مبدأ و مقصد فاقد حالت (state-less) بوده و برای سرویس های کوچک و ارتباط های غیر متدوام مناسب است.

۶.۲ ارتباط «مطمئن»

هر ارتباط TCP با ایجاد یک session آغاز می‌شود. به این منظور کارخواه یک پیام از نوع SYN به کارگزار ارسال کرده و sequence number آغازین خود را به کارگزار اعلام می‌کند. کارگزار با یک بسته از نوع SYN-ACK پاسخ کارخواه را می‌دهد که در آن sequence number اولیه خود را اعلام می‌کند. پس از آن کارخواه با یک بسته از نوع ACK دریافت بسته اولیه کارگزار را تأیید می‌کند و ارتباط آغاز می‌شود.



پس از آماده‌سازی ارتباط، تفاوت کمی بین کارخواه و کارگزار وجود دارد. هر دو طرف مخابره می‌توانند داده های خود را از طریق بسته‌های نوع DATA به طرف دیگر ارسال کنند. هر یک از طرفین، در پاسخ بسته‌های طرف دیگر بسته‌های ACK ارسال می‌کنند تا همتای خود را از دریافت بسته‌ها آگاه کنند و طرف دیگر بسته‌های بعدی را ارسال کند.
به این ترتیب در TCP می‌توان اطمینان داشت بسته‌ها در مقصد به درستی و با ترتیب صحیح دریافت شده‌اند. این مکانیسم توسط سیستم‌عامل و router های میانی کنترل می‌شود.

۳ آماده‌سازی

استثنائاً در این تمرین برای درک بهتر سازوکار شبکه شما موظف به استفاده از زبان C هستید. از آنجا که فراخوانی‌های سیستمی مربوط به شبکه شدیداً وابسته به بستر و سیستم‌عامل هستند، شما برای راه‌اندازی کد خود از Docker استفاده خواهیدکرد. [۷] در صورتی که توضیحات این بخش برای شما کافی نیست، آموزش آن در کلاس حل تمرین ارائه خواهدشد.

به این‌منظور یک `Dockerfile` از قبل برای شما طراحی شده‌است که کد C شما را در بستر لینوکس کامپایل و اجرا می‌کند. در ابتدا شما می‌بایست اسکلت پروژه را از مخزن [۸] Git شخصی خود دریافت کنید. اگر تا کنون مخزن خود را `clone` نکرده‌اید، با نام‌کاربری و رمزعبور خود [۹] در سرور گیت درس (`http://compnets.imakbari.com`) وارد شده و مخزن شخصی خود را انتخاب کنید. سپس از بالای صفحه آدرس SSH یا HTTPS [۱۰] آن را کپی کنید، در یک ترمینال به دایرکتوری موردنظر خود رفته و دستورهای زیر را وارد کنید:

```
git clone http://compnets.imakbari.com/students-40443-961/<YOUR STUDENT ID>.git
cd <YOUR STUDENT ID>
cd HW0
```

در اینجا فایلی به‌نام `Dockerfile` وجود دارد که حاوی تنظیمات Docker برای اجرای کد C شما در بستر مشخص می‌باشد. فایلی به نام `code/main.c` نیز وجود دارد که بایستی تابع `main()` کد شما درون آن قرار داشته باشد.
پس از نصب Docker می‌توانید از درستی کارکرد آن را با اجرای `./build.sh` برای build کردن کد تحت Docker و سپس اجرای `./runclient.sh` یا `./runserver.sh` برای اجرای کد کارخواه و کارگزار تحت Docker اطمینان حاصل کنید. در صورتی که کد بدون مشکل کامپایل و اجرا شود، جمله hello-world چاپ خواهد شد.
توجه داشته‌باشید کدهای شما پس از ارسال به صورت خودکار نمره‌دهی خواهندشد. دریافت و ارسال تمرین تماماً از طریق Git انجام خواهدشد.

۴ وظیفهٔ شما

یکی از ابتدایی‌ترین مثال های برنامه‌نویسی سوکت، نرم‌افزاری است که بتواند یک ارتباط متنی بین دو کاربر در یک شبکه ایجاد کند. برنامه شما باید بتواند هردو نقش کارخواه و کارگزار را بازی کند.

۱.۴ سمت کارگزار

در صورتی که برنامه شما **بدون آرگومان** ورودی اجرا شود، باید در نقش کارگزار شروع به کار کند. به این صورت که ر **وی پورت TCP 0.0.0.0:1234 گوش فرادهد** و منتظر وصل شدن کارخواه بماند. نکته: فرض بر این است که در هیچ حالتی بیش از یک کارخواه هم‌زمان به کارگزار متصل نخواهد شد، اما پس از اتمام کار کارخواه، کارگزار باید به حالت اولیه برگشته و مجدداً با گوش‌کردن روی پورت یادشده منتظر کارخواه بعدی بماند.

---

[۷] به این‌ترتیب شما می‌توانید در سیستم‌عامل محبوب خودتان برای بستر Linux برنامه‌نویسی کنید. اگرچه برای انجام این تمرین نیاز به دانش زیادی از عملکرد docker ندارید، می‌توانید برای اطلاعات بیشتر از ماهیت آن مستندهای رسمی آن را در `https://www.docker.com/what-docker` و `https://www.docker.com/use-cases` مطالعه کنید یا نحوه استفاده آن را در `https://www.conetix.com.au/blog/docker-basics-practical-starters-guide` مشاهده کنید.

[۸] Repository

[۹] به طور پیش‌فرض نام کاربری و رمز عبور شما همان شماره‌دانشجویی شماست. شما موظف هستید تا تاریخ **یکشنبه ۱۶ مهرماه ۹۶** به سرور Git وارد شده، رمز عبور خود را عوض کنید و اطمینان حاصل کنید مخزن شخصی شما وجود دارد. در صورت بروز هرگونه مشکل، دستیاران آموزشی را مطلع کنید. در غیر این‌صورت، مسئولیت تمامی مسائل یادشده با خود شما خواهدبود.

[۱۰] برای استفاده از SSH می‌بایست Public-key خود را در تنظیمات سرور گیت وارد کنید. در صورت عدم آشنایی با این مقوله، از آدرس HTTPS استفاده کنید. مزیت مهم استفاده از SSH عدم نیاز به واردکردن نام کاربری و رمز عبور در اجرای دستورات غیر محلی git است.

پس از اتصال کارخواه، هر جریانی از داده متنی که از کارخواه دریافت می‌شود باید عیناً در خروجی (STDOUT) چاپ شود و هر متنی که از STDIN دریافت می‌شود عیناً برای کارخواه ارسال شود. [11]

۲.۴ سمت کارخواه

برنامه شما در صورتی که با دو آرگومان اجرا شود، در نقش کارگزار عمل خواهد کرد. آرگومان‌های ورودی همواره یک **آدرس IP (جداشده با نقطه و ده‌دهی، مانند** `127.0.0.1` **)** و پس از آن یک **عدد شماره پورت** است که در آن یک کارگزار در حال گوش‌کردن می‌باشد.
کارخواه می‌بایست ابتدا به کارگزار در آدرس و پورت داده‌شده متصل شود. در صورتی که کارگزاری در این آدرس درحال گوش‌کردن نباشد، بایستی خطای `CAN'T CONNECT` در STDOUT چاپ شده و برنامه متوقف شود.
سپس مشابه عملکرد کارگزار، می‌بایست تمامی ورودی های STDIN از طریق سوکت برای کارگزار ارسال شود و هر داده متنی که از کارگزار دریافت شد در STDOUT عیناً چاپ شود.
در انجام پیاده‌سازی خود **حتماً** به نکات برنامه‌نویسی زیر توجه کنید:

- ورودی‌ها و خروجی با فرمت UTF-8 داده می‌شوند. بنابراین شما باید حالت کاراکترهای طولانی‌تر از یک بایت را نیز درنظر بگیرید. همچنین مفهوم Endianness داده‌ها در حافظه و در سوکت را نیز در نظر داشته‌باشید. به‌طور قراردادی در شبکه معمولاً داده‌ها به صورت Big-endian ارسال می‌شوند و از این رو به Big-endian لفظ Network Endianness نیز اطلاق می‌شود. شما نیز باید داده‌ها را به صورت Big-endian ارسال کنید. این درحالی است که داده‌های حافظه در سیستم‌های x86 به‌صورت Little-endian هستند.
- نیازی نیست که داده‌ها کاراکتر به کاراکتر بلافاصله ارسال شوند. می‌توانید برای ارسال یا چاپ داده‌ها تا line-break (خط جدید) بعدی یا EOF صبر کنید، هرچند این موضوع اجباری نیست.
- جریان داده با کاراکتر EOF تمام می‌شود. در حالت کارخواه، برنامه بایستی با دریافت EOF از STDIN آخرین ارسال خود را نیز انجام داده و سپس اتصال را بسته و متوقف شود. در حالت کارگزار نیز با دریافت EOF از سوکت، کارگزار اتصال را متوقف کرده و منتظر کارخواه بعدی می‌شود.

۳.۴ ارسال

برای ارسال، کافیست کدهای خود را commit و push کنید.

```
git add -A
git commit -m "<Commit Message Here>"
git push
```

از **ساعت ۲۳:۵۹ دوشنبه ۱۷ مهرماه** به بعد، با هر push ، حداکثر تا نیم‌ساعت بعد کد شما به صورت اتوماتیک تست و نمره‌دهی شده و نتیجه آن از طریق ایمیل به شما اطلاع داده می‌شود. در صورتی که پس از نیم‌ساعت ایمیلی دریافت نکردید، TA های درس را مطلع نمایید. از آنجا که منابع سرور نمره‌دهی خودکار محدود است، برای رفاه همکلاسی‌های خود از push های بی‌مورد خودداری کنید. بدیهتاً با هرگونه تلاش برای کند کردن سرور درس شدیداً برخورد خواهد شد.

۴.۴ نکات

- در این تمرین و سایر تمرین‌های درس، با هرگونه تقلب شدیداً برخورد خواهدشد. کدهای ثبت شده به صورت خودکار برای تقلب بررسی می‌شوند.
- از آنجا که مسئله داده‌شده بسیار سرراست و متداول است، طبعاً منابع زیادی برای آن در اینترنت وجود دارد. از این رو، کپی‌کردن کد از اینترنت نیز مصداق تقلب محسوب می‌شود.

[11] به عبارتی برنامه شما همانند دستور netcat یونیکس به صورت دوطرفه عمل می‌کند.

- در انجام تمرین شما فقط مجاز به استفاده از کتابخانه استاندارد C برای Unix هستید که در فایل قالب اولیه نیز include شده‌است. استفاده از منابع و کتابخانه‌های دیگر با هدف تمرین در تناقض بوده و مجاز نمی‌باشد.
- ددلاین ارسال تمرین ساعت **۲۳:۵۹** روز **۲۱ مهرماه ۱۳۹۶** می‌باشد. پس از آن به ازای هرروز، ٪۲۰ از نمره شما کاسته خواهد شد.
- در صورت داشتن هرگونه سوال و مشکل، می‌توانید از طریق piazza و یا ایمیل‌های `imakbari@gmail.com` و `hooman.shaah@gmail.com` آنها را مطرح کنید.
- از ارسال پاسخ در پیاتزا و گروه‌های تلگرام و سایر منابع عمومی خودداری کنید.
- ملاک عملکرد کد، عملکرد آن در container طراحی شده با Dockerfile داده‌شده است. بنابراین صرفاً به‌ازای کارکردن آن در سیستم خودتان (بدون Docker)، در قسمت نمره‌دهی خودکار به شما نمره‌ای تعلق نخواهدگرفت.

موفق باشید