

به نام خدا



درس برنامه سازی پیشرفته

شبکه

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

نیم سال دوم ۹۹-۹۸

اساتید:

مهدی مصطفی زاده، ایمان عیسی زاده، امیر ملک زاده، علی چکاه

نگارش و تهیه محتوا:

علیرضا ضیایی، سجاد ریحانی، محسن دهقان کار

تنظیم داک:

امیر مهدی نامجو

فهرست

۲	شبکه چیست؟
۲	لایه‌های شبکه
۷	معماری شبکه
۸	معماری Client - Server
۹	شبکه در جاوا
۱۰	معماری Peer - To - Peer
۱۰	Authentication Token



شبکه چیست؟

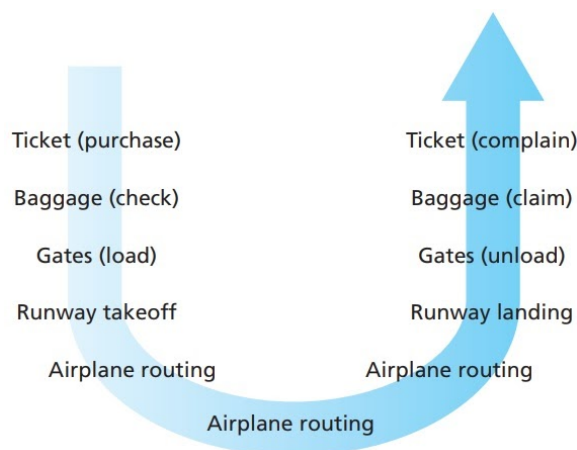
به مجموعه‌ای از کامپیوترهای متصل به هم و زیر ساخت‌های ارتباطی (مانند Switch، Firewall، Router و ...) آن‌ها یک شبکه گفته می‌شود. بزرگ‌ترین شبکه کامپیوتری در جهان همین اینترنت است.

لایه‌های شبکه

شبکه‌های کامپیوتری (مثل اینترنت) پیچیدگی‌های زیادی دارند، به همین خاطر برای توصیف آن‌ها از لایه بندی شبکه استفاده می‌شود. به این صورت که شبکه و پروتکل‌های آن را به صورت یک سری لایه بررسی می‌کنند که هر لایه از سرویسی که لایه زیرینش فراهم می‌کند استفاده کرده و سرویسی را به لایه‌ی بالاتر خود ارائه می‌دهد.

برای روشن شدن لایه‌های شبکه به این مثال توجه کنید. ما روزانه با سیستم‌های پیچیده‌ای سروکار داریم. فرض کنید فردی از شما می‌خواهد که برایش سیستم حمل و نقل هوایی را توصیف کنید. یکی از شیوه‌های توصیف این سیستم می‌تواند این گونه باشد که کارهایی را که یک فرد تا رفتن به مقصد باید انجام دهد را لیست کنیم.

مثلاً ابتدا بلیط تهیه می‌کنیم، چمدان‌هایمان چک می‌شوند، به گیت می‌رویم و نهایتاً سوار هواپیما می‌شویم. سپس هواپیما پرواز می‌کند و مسیرش را تا مقصد می‌پیماید. بعد از فرود آمدن هواپیما، به گیت می‌رویم، چمدان‌هایمان را می‌گیریم. این سناریو در تصویر زیر خلاصه می‌شود.



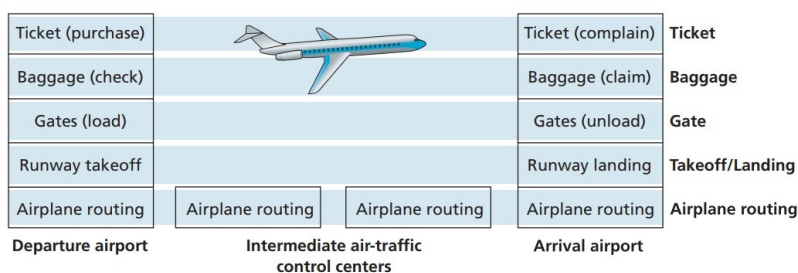
اتفاقی که در شبکه می‌افتد این است که تعدادی بسته اطلاعات (packets data) قرار است از کامپیوتر مبدا به مقصد بروند. همان طور که مشهود است، این کار شباهت‌هایی با سفر با هواپیما دارد.

آنچه در توصیف هواپیما گفتیم را به‌خاطر بیاورید، با توجه به شکل زیر این فرایند را بگونه‌ای دیگر نمایش می‌دهیم. دقت کنید که:

- برای سوار شدن هواپیما باید فرآیند خرید بلیط را انجام دهیم (ticketing function) سپس از طرف دیگر هنگام پیاده شدن از هواپیما در مقصد نیز بلیط چک می‌شود (ticketing function).

- قبل از سوار شدن در هواپیما (مبدا) ، چمدان‌هایمان را تحویل می‌دهیم (baggage function) و از سوی دیگر در شهر مقصد چمدان‌هایمان را تحویل می‌گیریم (baggage function)

... •





با توجه به این شکل، سفر با هواپیما ساختار خاصی پیدا می‌کند و ما می‌توانیم کارهای مختلفی را که انجام می‌دهیم (مثلا کارهای مربوط به تهیه و نشان دادن بلیط) به شکل افقی یا لایه‌ای توصیف کنیم. مثلا کارهای مربوط به بلیط یک لایه هستند و کارهای مربوط به baggage یک لایه دیگر محسوب می‌شوند.

اگر دقت کنید، هر لایه همراه با لایه‌های زیرین خود به نوعی یک عملکرد و سرویس را پیاده‌سازی می‌کند. مثلا لایه‌ی بلیط به همراه لایه‌های زیرین، کار انتقال مسافر از پیشخوان یک فرودگاه به پیشخوان فرودگاه دیگر را پیاده‌سازی کرده‌اند و لایه‌ی مربوط به بار (baggage) کار انتقال چمدان‌های مسافران را پیاده‌سازی می‌کند (که به نوعی یک سرویس برای انتقال مسافران است و این سرویس تنها برای مسافرانی است که بلیط دارند). در لایه Airplane Routing عملیات مسیریابی هواپیما پیاده‌سازی شده است.

همان‌طور که دیدیم، به کمک ساختاری لایه‌ای توانستیم این سیستم پیچیده را (تأحد خوبی ساده‌تر) توصیف کنیم. در شبکه کامپیوتری نیز ارتباط بین راس‌های مختلف شبکه (node) را با ساختاری لایه‌ای توصیف می‌کنیم. بطور کلی پروتکل‌های شبکه را می‌توان به ۵ لایه‌ی زیر تقسیم کرد:

Application
Transport
Network
Link
Physical

Five-layer
Internet
protocol stack

فرض کنید دو برنامه داریم که بر روی دو کامپیوتر جدا از طریق شبکه با هم در ارتباط هستند:

۱. **Application layer**: این لایه در واقع API یا واسطی است که به وسیله‌ی آن این دو برنامه حرف همدیگر را می‌فهمند. مثلا برنامه نویس قرارداد (protocol) می‌کند که برنامه اول به برنامه دوم در ابتدا یک رشته "salam" بفرستد. سپس برنامه دوم در



جواب این رشته بتواند یکی از دو رشته‌ی "Aleik" و یا "Bye" را بفرستد و اگر برنامه دوم "Aleik" را فرستاده بود برنامه اول ...
مثلا HTTP از پروتکل‌های لایه اپلیکیشن است که در وب بسیار کاربرد دارد.

۲. **Transport layer**: این لایه پایین‌تر از لایه Application قرار دارد (در واقع ارتباط بین این دو لایه از طریق سوکت (socket) برقرار می‌شود که در مورد آن صحبت خواهد شد). همان طور که می‌دانیم، در کامپیوتر ما چندین برنامه در حال ارتباط با شبکه هستند. پس اگر برنامه اول (از کامپیوتر اول) پیامی را به سمت کامپیوتر دوم می‌فرستد، این کامپیوتر باید به نحوی بفهمد که این پیام برای کدام برنامه است. این عملیات در این لایه پیاده‌سازی شده است. در واقع به هر برنامه (process) که منتظر دریافت پیام از شبکه است (listening)، یک عدد به اسم پورت نسبت داده می‌شود که با استفاده از آن برنامه مقصد بطور یکتا در کامپیوتر مشخص می‌شود. TCP و UDP، دو پروتکل این لایه محسوب می‌شوند.

۳. **Network Layer**: حال فرض کنید برنامه اول پیامش (packet) را فرستاد. این پیام چگونه باید به کامپیوتر مقصد برسد؟ فرایند مسیریابی (routing) در شبکه توسط این لایه پیاده‌سازی شده است. در واقع هر میزبان (host)، یک آدرس یکتایی خواهد داشت به اسم IP address. برای فرستادن یک بسته، بر روی آن IP مقصد برچسب می‌خورد و با سرویس‌هایی که این لایه فراهم می‌کند، این بسته به مقصد می‌رسد.

بیشتر بدانید:

در حال حاضر دو نسخه از IP پیاده‌شده است که به IP های رایج که به فرمت X.X.X.X بوده و ۳۲ بیتی هستند IPv4 گفته شده و به IP هایی که به فرمت X:X:X:X:X:X:X:X و ۱۲۸ بیتی هستند IPv6 می‌گویند.



- رنج‌های IP:

- IP های محلی: این بازه از IP ها فقط درون همان شبکه معتبر و یکتا است و عملاً خارج از اسکوپ (scope) آن شبکه اعتبار ندارد و دیگر یکتا نیست.

- ۱. 10.0.0.0 تا 10.255.255.255

- ۲. 172.16.0.0 تا 172.31.255.255

- ۳. 192.168.0.0 تا 192.168.255.255

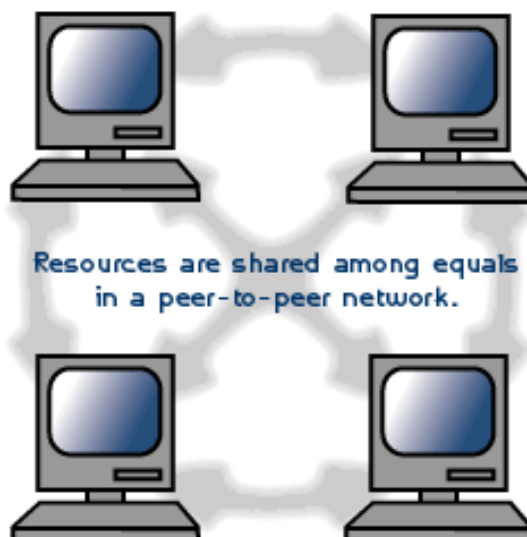
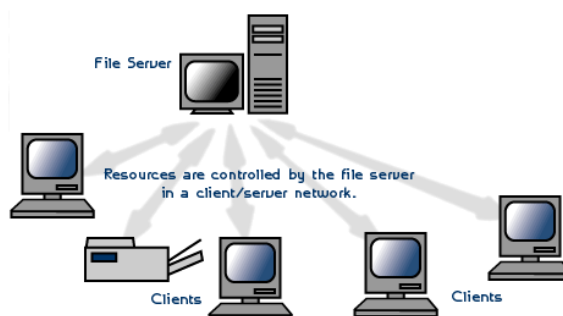
- IP های عمومی: تمامی IP ها به جز IP محلی از این نوع هستند.

- این لایه و لایه‌های بعدی بطور کلی مسئولیت مسیریابی و یافتن مسیر بهینه (با توجه به سیاست‌های شبکه) برای رسیدن هر بسته به مقصد مورد نظر با استفاده از تجهیزات (مثل سویچ، روتر و ...) و پروتکل‌هایی که در این لایه و دو لایه بعد وجود دارد (IP) در این لایه و به طور مثال Ethernet در لایه بعدی و ...) را بر عهده دارند که با توجه عدم نیاز به آشنایی با لایه‌های بعدی و پروتکل‌های آن در این پروژه، در مورد لایه‌های بعد صحبتی نمی‌کنیم.

معماری شبکه

معماری شبکه به طراحی فیزیکی و منطقی نرم افزار ، سخت افزار ، پروتکل‌ها و رسانه انتقال داده‌ها می‌پردازد. به زبان ساده‌تر می‌توان گفت که چگونه کامپیوترها ساماندهی شده و با هم ارتباط گرفته و چگونه وظایف به رایانه اختصاص می‌یابد.

معماری شبکه به دو نوع عمده Peer-To-Peer و Client-Server تقسیم می‌شود. شبکه Peer-To-Peer شبکه‌ای است که در آن کلیه رایانه‌ها با امتیاز و مسئولیت برابر برای پردازش داده‌ها بهم متصل می‌شوند.





معماری Client-Server

مدل Client-Server که بسیاری از شبکه‌ها بر مبنای این مدل کار می‌کنند شامل یک برنامه در نقش سرور و چندین برنامه در نقش کلاینت است. منابع و اطلاعات همگی در سمت سرور نگه داشته می‌شوند و کلاینت‌ها بعد از اتصال به سرور با ارسال درخواست اطلاعات را از سرور می‌گیرند و یا اطلاعات جدید را به سرور می‌دهند.

ارتباط کلاینت‌ها و سرور از طریق سوکت برقرار خواهد شد. سوکت بیانگر یک سر از یک ارتباط دوطرفه بین دو برنامه است. هر سوکت اطلاعات مربوط به آی پی و پورت ارتباط را نگه میدارد و دریچه ارسال و دریافت اطلاعات به آن سوی ارتباط نیز هست.

اگر آدرس دهی یک شبکه را با فرایند آدرس دهی یک سیستم که بر پایه اداره پست کار می‌کند، مقایسه کنید، آنگاه متوجه خواهید شد که آدرس آی پی میزبان نقش آدرس یک ساختمان را داشته و درگاه (پورت) شبیه به شماره واحدی است که درون یک ساختمان قرار دارد و سوکت نیز مانند صندوق پست آن واحد است. یک سوکت شامل هر دو گروه آدرس آی پی میزبان و پورت TCP یا UDP مربوط به یک برنامه است که با یک علامت جداکننده این دو مقدار از یکدیگر جدا شده‌اند. (مثلا 172.0.0.1:8000)

سوکت‌ها دو نوع مهم دارند:

۱. سوکت‌های استریم که از پروتکل [TCP](#) برای انتقال داده استفاده می‌کنند.

۲. سوکت‌های دیتاگرام که از پروتکل [UDP](#) برای انتقال داده استفاده می‌کنند.

پروتکل TCP (transmission control protocol) یک پروتکل ارتباط محور (connection oriented) است و عملکرد آن بدین صورت است که برای هر پکت ارسالی توسط کامپیوتر مبدأ، باید یک پکت از سرور مقصد، مبنی بر دریافت صحیح و بدون نقص پکت دریافت کند. اگر طی زمان مشخصی این پیام توسط مبدأ دریافت نگردد، فرایند ارسال پکت مجدداً تکرار خواهد شد و کاربرد آن بیشتر در مواردی است که نیاز به اطمینان از صحت انتقال اطلاعات داریم. (مثل ارتباط اپلیکیشن بانکی با سرور بانک)

پروتکل UDP (User Datagram Protocol) یک پروتکل غیر ارتباط محور (connectionless) است. برخلاف TCP در این پروتکل هیچ گونه پیامی مبنی بر دریافت پکت از سوی سرور ارسال نشده و بیشتر در مواردی مانند انتقال صوت یا ویدئو که پهنای باند در این موارد از



اهمیت بالایی برخوردار است بکار می‌رود زیرا در صورت استفاده از پروتکل TCP، ترافیک هر پیام تایید به ازای دریافت پکت، خود باعث اشغال پهنای باند خواهد شد.

شبکه در جاوا

کلاس Socket و ServerSocket که در کلاس و حل تمرین نحوه استفاده و پیاده‌سازی آن تدریس شده، از نوع TCP هستند. برای آشنایی بیشتر می‌توانید به [اینجا](#) و برای هندل کردن همزمان چند کلاینت به [اینجا](#) مراجعه کنید.

همچنین برای آشنایی با پیاده‌سازی پروتکل HTTP در جاوا می‌توانید به [این لینک](#) مراجعه کنید.

از آنجایی که در پروژه شما سرعت و حجم انتقال داده‌ها اهمیت چندانی ندارد نیازی به استفاده از سوکت udp ندارید اما برای آشنایی بیشتر با پیاده‌سازی این مدل می‌توانید به لینک‌های زیر مراجعه کنید.

[A Guide To UDP In Java](#)

[TCP vs UDP - Difference and Comparison](#)

به طور خلاصه ارتباط بین سرور و کلاینت (مدل TCP) به طریق زیر صورت می‌گیرد:

۱. سرور یک سوکت بر روی پورت مثلا 6666 می‌سازد و به آن گوش می‌کند:

```
1 //Server side
2 ServerSocket ss=new ServerSocket(6666);
```

۲. کلاینت نیز یک سوکت می‌سازد و روی همان پورت درخواست اتصال می‌دهد:

```
4 //Server side
5 //Client side
6 Socket s=new Socket("localhost",6666) ;
7 //localhost == 127.0.0.1
```

۳. سرور با دریافت درخواست اتصال آن را تایید می‌کند و دو طرف به هم وصل می‌شوند:

```
10 //Server side
11 Socket s=ss.accept();
```

۴. دو طرف از طریق سوکت سمت خودشان شروع به تبادل داده می‌کنند:



```
13 //Client side
14 DataOutputStream dout=new DataOutputStream(s.getOutputStream());
    dout.writeUTF("Hello Server");

15 //Server side
16 DataInputStream dis=new DataInputStream(s.getInputStream());
17 String str=(String)dis.readUTF();
```

دقت کنید که پورتنی که سرور می‌خواهد استفاده کند نباید در اختیار (bind) برنامه دیگری باشد، برای فهمیدن اینکه چه پورت‌هایی توسط سیستم یا برنامه‌های دیگر در حال استفاده هستند می‌توانید به این لینک‌ها مراجعه کنید: [ویندوز](#)، [لینوکس](#)

معماری Peer-To-Peer

برای مطالعه در مورد این معماری (از موارد امتیازی این فاز است) به داک P2P مراجعه کنید.

Authentication Token

Token یک کلید برای ارتباط بین کلاینت و سرور است. برای هر کاربر پس از احراز هویت توسط نام کاربری و رمز عبور یا هر روش دیگر، یک Token از سمت سرور مشخص می‌شود که از آن به بعد، پیام‌های ارسالی توسط کاربر به وسیله‌ی آن Token در سرور شناسایی می‌شوند. این Token ها می‌توانند محدودیت زمانی استفاده داشته باشند و برای رعایت امنیت باید پس از هر بار ورود دوباره‌ی کاربر Token جدیدی به او اختصاص داده شود.

پیاده‌سازی توکن در ارتباط بین کلاینت و سرور فروشگاه از موارد امتیازی این فاز است. برای مطالعه بیشتر در مورد توکن به این لینک مراجعه کنید:

[The Ins and Outs of Token Based Authentication](#)