# Computer Network

AmirReza Azari
99101087

# تمرین دوم

## سوال اول

**سؤال ۱ (۱۵ نمره)** فرض کنید به‌جای ارتباط *Three-way Handshake*، از یک ارتباط دوگانه (*Two-way Handshake*) استفاده کنیم. آیا در این روش مشکلی وجود خواهد داشت؟ در صورتی که مشکلی وجود دارد مثال بزنید و در غیر این صورت نشان دهید مشکلی وجود ندارد.

## پاسخ سوال اول:

مشکل وجود دارد.

فرض کنید میزبان دوم برای شروع ارتباط یک بسته با شماره ترتیبی برابر x به میزبان اول ارسال می‌کند و میزبان اول آن را تایید می‌کند و به همراه آن شماره ترتیب خودش را برای میزبان دوم ارسال می‌کند. در این زمان ارتباط از سمت میزبان اول باز شده است و منتظر بسته بعدی که همان x+1 است، می‌باشد. حال اگر این ACK با تاخیر بیشتر از timeout به میزبان دوم برسد، میزبان دوم مجددا برای شروع ارتباط یک بسته جدید با شماره ترتیبی برابر مثلا u به میزبان اول ارسال می‌کند. ارتباطِ بازشده در میزبان اول چون منتظر بسته x+1 است این بسته را نادیده می‌گیرد. وقتی تایید بسته اولیه با تاخیر به میزبان دوم می‌رسد، میزبان دوم از روی شماره تایید (ack number)متوجه می‌شود که این بستهٔ تایید تقاضایی که اخیرا فرستاده نیست و آن را ندیده می‌گیرد، اما همچنان منتظر رسیدن تایید تقاضای جدیدش می‌ماند که هرگز به دستش نخواهد رسید. وقتی دوباره timeout شود، میزبان دوم یک تقاضای SYN جدید می‌فرستد و این هم به سرنوشت قبلی دچار می‌شود. این روند می‌تواند تا ابد ادامه یابد و به نوعی ددلاک رخ دهد.

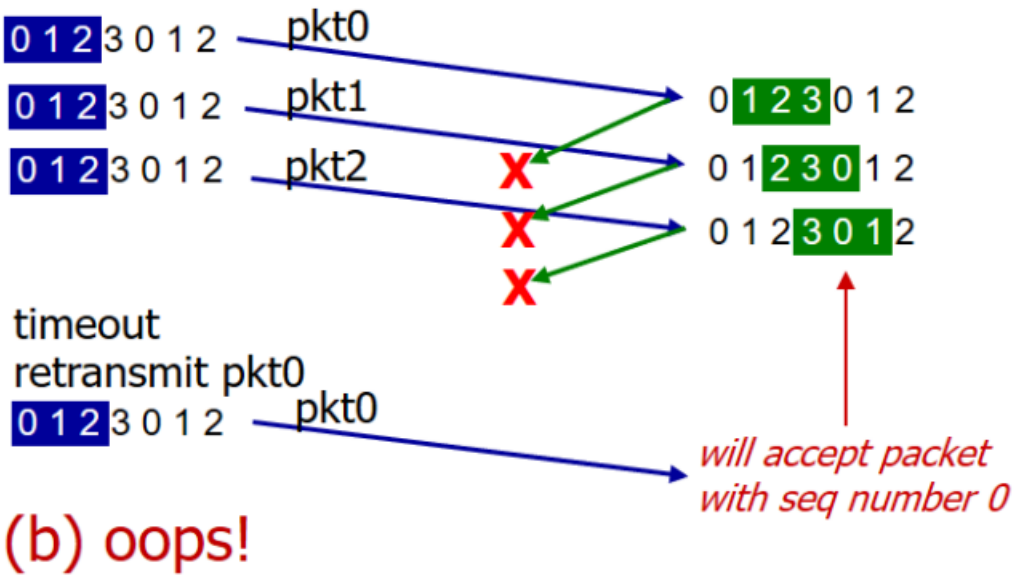
اگر دست‌تکانی دومرحله‌ای باشد، میزبان اول باید بعد از ارسال SYN-ACK یک زمان‌سنج روشن کند که اگر در مدت مشخصی میزبان دوم بستهٔ دیگری متعلق به همین اتصال TCP ( با همان شماره ترتیب مورد انتظار) نفرستاد، اتصال را ببندد.

## سوال دوم

**سؤال ۲ (۱۵ نمره)** اگر *sequence number* در *Selective Repeat* در بازه $[0, n]$ و *window size* نیز $w$ باشد. نشان دهید اگر $n$ کمتر $2w$ باشد، پروتکل دچار مشکل می‌شود.

## پاسخ سوال دوم:

مانند مثال اسلاید، در نظر بگیرید که seq numberها از 0 تا 3 باشند و اندازه window برابر 3 باشد. هنگامی که بسته‌های 0 تا 2 ارسال می‌شوند، توسط گیرنده دریافت می‌شوند و گیرنده به ازای هر پکتی که دریافت می‌کند، هم ack آن را می‌فرستد و هم پنجره خود را جلو می‌برد. در حالتی که پکت‌های 0 تا 2 برسند، پنجره گیرنده 3 خانه جلو می‌رود و اکنون به دلیل اینکه 4 عدد seq number مختلف داشتیم، به ترتیب منتظر 2 1 0 3 خواهد بود. اما اگر ackهای مرحله قبل در بین راه از دست بروند و فرستنده آن‌ها را دریافت نکند، بعد از گذشت timeout مجددا همان پکت‌های ابتدایی با seq numberهای 0 تا 2 را ارسال خواهد کرد. در این هنگام وقتی گیرنده پکتی با seq number = 0 دریافت می‌کند، خیال می‌کند پکت جدید که با شماره 0 منتظر آن بوده را دریافت کرده است که این مشکل دارد. این مشکل برای تعداد seq number از 0 تا 4 ( 2 برابر سایز پنجره منهای یک) نیز برقرار است. به شکلی که در پنجره جدید بعد از دریافت پکت‌های 0 تا 2 از فرستنده، پنجره جدید از سمت گیرنده به ترتیب منتظر 0 4 3 خواهد بود و اگر همان مشکل برای ackهای قبلی پیش بیاید، برای پکت 0 به مشکل خواهیم خورد. اما اگر همین مثال تعداد seq numberها برابر 6 یا همان 2 برابر سایز پنجره بود، دیگر همچین مشکلی را نداشتیم. بنابراین اگر تعداد seq numberها کمتر از 2w باشد، پروتکل دچار مشکل می‌شود.




## سوال سوم

**سؤال ۳ (۱۰ نمره)** می‌دانیم که *TCP* دارای مکانیزم *flow control* است توضیح دهید که این مکانیزم چیست و فیلد *window size* در هدر سگمنت *TCP* چه کارکردی در این مکانیزم دارد.

## پاسخ سوال سوم:

مکانیزم flow control در TCP برای مدیریت جریان اطلاعات بین فرستنده و گیرنده استفاده می‌شود تا اطمینان حاصل شود که گیرنده قادر به پردازش و دریافت داده‌ها با سرعتی که فرستنده آن‌ها را ارسال می‌کند، است.

در TCP، مکانیزم flow control به کمک یک فیلد به نام "window size" کار می‌کند که در هدر سگمنت TCP قرار دارد. این فیلد نشان دهنده حداکثر حجم داده‌ای است که گیرنده می‌تواند بدون درخواست از فرستنده دریافت کند. به عبارت دیگر، این فیلد نشان می‌دهد که گیرنده چقدر فضای خالی در برای دریافت داده‌ها دارد. وقتی که گیرنده اطلاعات را دریافت می‌کند، آن‌ها را به فرستنده اعلام می‌کند که فضای خالی در بافر خود دارد (که معمولاً با مقدار window size نشان داده می‌شود). ارسال کننده با توجه به این اطلاعات، می‌تواند تعداد بسته‌های داده‌ای که برای ارسال در صف قرار دارند را تنظیم کند، تا جلوگیری از ارسال داده‌های بیش از حدی که گیرنده قادر به پردازش آنها نیست، شود. به این ترتیب، مکانیزم flow control اجازه می‌دهد تا ارسال کننده و گیرنده با هماهنگی بیشتری عمل کنند و از ایجاد اشباع شبکه جلوگیری کنند.

## سوال چهارم

**سؤال ۴ (۲۰ نمره)** میزبان A می‌خواهد یک فایل 6000 بایتی را از میزبان B دریافت کند. در پروتوکل مورد استفاده، $MSS = 4byte$ و داریم $timeout = 3RTT$. همچنین از *cumulative ack* استفاده نمی‌شود و هر بسته باید جداگانه *ack* شود.

۱) با فرض اینکه بسته‌های ۱م و ۷م، از دست بروند، مشخص کنید هریک از پروتکل‌های انتقال مطمئن زیر، چقدر زمان نیاز دارند تا میزبان B از انتقال مطمئن به میزبان A آگاه شود. (فرض کنید هیچ بسته دیگری گم نمی شود.)

- *Stop and Wait*
- *Go-Back-N* (اندازه پنجره ارسال، ۳ MSS می‌باشد.)
- *Selective Repeat* (اندازه پنجره ارسال، ۳ MSS می‌باشد.)
- *Selective Repeat* (اندازه پنجره ارسال، ۵ MSS می‌باشد.)

۲) اکنون فرض کنید فقط بسته‌ی ۱۰۰ام گم می‌شود. با فرض اینکه از فرستنده‌ای با *FSM* داده شده در *Figure 3.51* کتاب درس استفاده کنیم، این مدت زمان چقدر خواهد شد؟

## پاسخ سوال چهارم:

۱) می‌دانیم $\frac{6000}{4} = 1500$ بسته داریم.

- Stop and wait

در این پروتکل، ابتدا در یک RTT بسته اول را می‌فرستیم و منتظر ACK خواهیم ماند. بعد از timeout = 3RTT، این بسته را مجددا ارسال می‌کنیم و یک RTT دیگر زمان می‌گذاریم. حال بسته‌های 1تا 6 را به ترتیب در 6RTT ارسال می‌کنیم. در یک RTT دیگر، بسته 7ام را ارسال می‌نماییم و به اندازه timeout = 3RTT صبر می‌کنیم و مجددا بسته را ارسال می‌کنیم. از اینجا به بعد چون بسته دیگری از دست نمی‌رود، می‌توانیم بگوییم بسته‌های 7 تا 1500 به ترتیب در 1494RTT ارسال خواهند شد. بنابراین داریم:

$$1\ timeout + 6\ RTT + 1\ timeoout + 1494\ RTT = 3 + 6 + 3 + 1494 = 1506\ RTT$$

- GBN

اندازه پنجره ارسال برابر 3 می‌باشد. ابتدا بسته‌های 1 تا 3 را ارسال می‌کنیم. چون بسته اول از دست می‌رود، بعد از timeout مجددا این 3 بسته را ارسال می‌نماییم. حال بسته‌های 4 تا 6 را در یک RTT ارسال می‌کنیم. سپس به بسته‌های 7 تا 9 می‌رسیم که مشابه حالت 1تا 3، بعد از timeout مجددا این بسته‌ها را ارسال می‌کنیم. درواقع قبل ارسال مجدد 7 تا 9، بسته‌های 1 تا 6 ارسال شده‌اند. 4149 بسته باقی مانده‌اند که در $\frac{1494}{3}RTT$ ارسال خواهند شد. بنابراین داریم:

$$1\ timeout + 2\ RTT + 1\ timeout + \frac{1494}{3} RTT = 3 + 2 + 3 + 498$$
$$= 506\ RTT$$

- SR (MSS = 3)

ابتدا بسته‌های 1 تا 3 را ارسال می‌کند. پاسخ بسته‌های 2 و3 را دریافت می‌کند اما به دلیل اینکه پاسخ بسته اول را نگرفته است، نمی‌تواند پنجره را جلو ببرد. بنابراین به اندازه timeout صبر می‌کند و سپس در یک RTT تنها بسته اول را مجدد می‌فرستد. سپس بسته‌های 4 تا 6 را در یک RTT دیگر ارسال می‌کند. حال نوبت به بسته‌های 7 تا 9 می‌رسد. مشابه حالت قبلی، به اندازه یک timeout صبر می‌کند و سپس در یک RTT بسته 7ام را ارسال می‌کند. تا اینجا بسته‌های 1 تا 9 ارسال شده‌اند. 1491 بسته باقی مانده‌اند که در $\frac{1491}{3} RTT$ ارسال خواهند شد. داریم:

$$1\ timeout + 1\ RTT + 1\ RTT + 1\ timeout + 1\ RTT + \frac{1491}{3} RTT$$
$$= 3 + 1 + 1 + 3 + 1 + 497 = 506\ RTT$$

- SR (MSS = 5)

ابتدا بسته‌های 1 تا 5 را ارسال می‌کنیم و مشابه بخش قبل، چون بسته اول نمی‌رسد، بعد timeout در یک RTT مجددا بسته اول را ارسال می‌کنیم. حال بسته‌های 6 تا 10 را ارسال خواهیم کرد. در این بخش بعد از یک RTT پاسخ بسته ششم می‌رسد و در همان زمان می‌توانیم بسته 11ام را ارسال کنیم. پاسخ بسته‌های دیگر را نیز داریم اما چون پاسخ بسته 7ام نرسیده است، نمی‌توانیم بسته جدید را ارسال کنیم. پاسخ بسته 11ام در RTT بعدی می‌آید اما چون همچنان timeout نگذشته است، تا پایان آن صبر می‌کنیم و بعد از آن تنها بسته 7 را مجددا ارسال می‌کنیم. حال بسته‌های 1 تا 11 را ارسال کرده‌ایم و 1489 بسته باقی مانده‌اند. داریم:

$$1\ timeout + 1\ RTT + 1\ timeout + 1\ RTT + ceil\left(\frac{1489}{5}\right)$$
$$= 3 + 1 + 3 + 1 + 298 = 306\ RTT$$

**نکته: دقت شود در حل مسائل بالا، از handshake صرف نظر شده است یا به گونه‌ای، بعد از handshake مسئله را حل کرده‌ایم. برای در نظر گرفتن handshake کافی است یک RTT به تمامی پاسخ‌ها اضافه نماییم.**

2 )

برای این بخش مطابق FSM مطرح شده، و با توجه به اینکه بسته 100 را از دست داده‌ایم، داریم:

$$1\ RTT(1) + 1\ RTT(2,3) + 1\ RTT(4-7) + 1\ RTT(8-15) + 1\ RTT(16-31) + 1\ RTT(32-63) = 6\ RTT$$

حال در مرحله بعد، 64 تا 127 را خواهیم فرستاد که بعد از timeout، مقدار cwind مجددا 1 خواهد شد. داریم:

$$1\ RTT(64-127) + 1\ RTT(128-226) + 1\ RTT(227-325) = timeout = 3\ RTT$$

پس تا قبل timeout بسته‌های 1 تا 325 به جز 100 ارسال شده‌اند. بنابراین $1500 - 324 = 1176$ بسته باقی مانده‌اند. چون از این به بعد در هر مرحله بدون مشکل 2 برابر می‌شود از log_2 کمک می‌گیریم. در نهایت داریم:

$$6\ RTT + 3\ RTT + \text{ceil}(\log_2 1176) = 6 + 3 + 11 = 20\ RTT$$

البته برای حل این بخش به گفته صورت سوال فرض کردیم که ack ها تجمعی نباشند. اگر برخلاف صورت سوال، ackها را تجمعی در نظر بگیریم، داریم:

مانند بخش قبل از 1 تا 63 را در 6 RTT ارسال می‌کنیم. بعد سراغ 64 تا 127 می‌رویم. 64 تا 99 ارسال می‌شوند و CWIND برابر 100 می‌شود. حال تا بسته 199 ارسال خواهد شد. اکنون 3 تا dup ack برای 101 تا 103 خواهد آمد. در این حالت sstresh برابر 50 و cwind برابر 53 خواهد شد. حال سگمنت 100 دوباره ارسال می‌شود و بسته‌های 104 تا 199 هم داخل شدن و اکنون cwind = 149 می‌شود. حال برای بسته 100 ack میاد و چون تجمعی است، مقدار ack تا 199 می‌رود. حال به حالت congestion avoidance می‌رویم و cwind = 50. حال در این استیت، مقدار بسته‌ها خطی زیاد می‌شوند.

تا الان $6 + 1 + 1 = 8\ RTT$ پیش رفته‌ایم. اندازه cwind هم برابر 50 است. $1500 - 199 = 1301$ بسته باقی مانده‌اند. داریم:

$$50(1) + 51(2) + 52(3) + 53(4) + \cdots + 71(22)$$

در cwind = 71، تمام بسته‌ها را ارسال کرده‌ایم. بنابراین $22 + 8 = 30RTT$ به دست آمد.

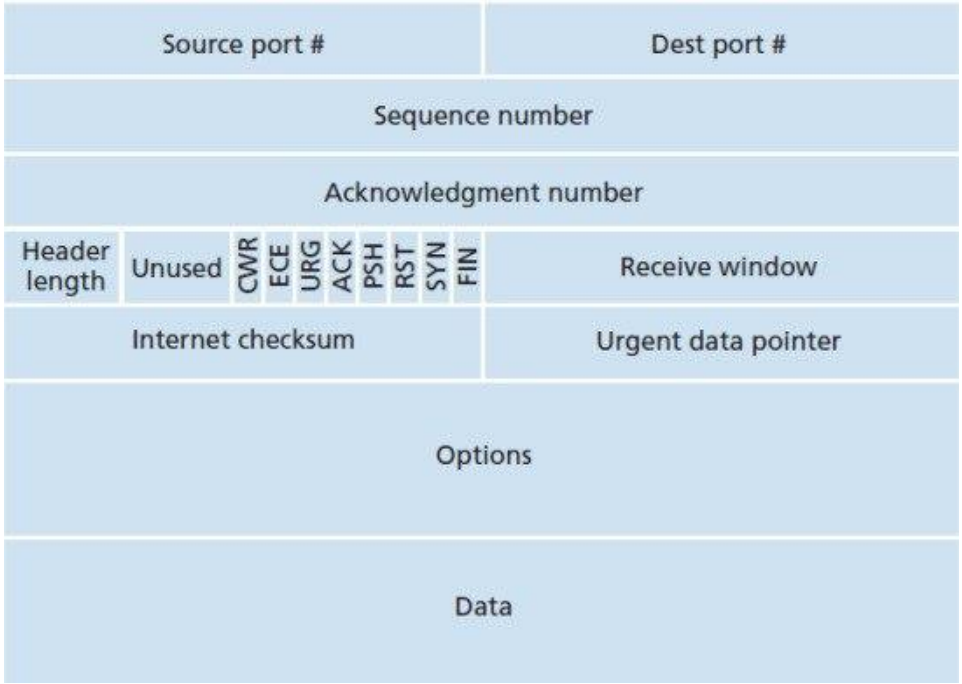
## سوال پنجم

**سؤال ۵ (۱۵ نمره)** بسته‌ای توسط پروتکل *TCP* دریافت شده‌است، و اطلاعات مربوط به سرآیند آن در جدول زیر آمده است. با توجه به آن، موارد خواسته شده را مشخص کنید.

| | | | |
|---|---|---|---|
| $C4$ | 57 | $AB$ | $5B$ |
| $6B$ | $DB$ | $CF$ | $9E$ |
| 83 | 47 | $2D$ | 97 |
| 50 | 11 | 01 | $F5$ |
| 72 | $4D$ | 00 | 00 |

۱ ) پورت مبدا و مقصد را مشخص کنید .
۲ ) این بسته چندمین بایت را *Ack* می‌کند؟ (*Base Ack* = 2202479963) آیا این مقدار معتبر است؟
۳ ) این بسته مربوط به کدام قسمت اتصال *TCP* است؟
۴ ) شماره‌ی ترتیب (*sequence number*) این بسته چند است؟ این عدد نشان‌دهنده‌ی چیست؟

## پاسخ سوال پنجم:




( 1

$$\text{source port number} = C457(\text{hex}) = 50263$$

$$\text{dest port number} = AB5B(\text{hex}) = 43867$$

( 2

$$ack\ number = 83472D97(hex) = \ 2202480023$$

$$base\ ack = \ 2202479963$$

$$2202480023 - \ 2202479963 = 60$$

پس 60امین بایت را ack می‌کند.

با توجه به اینکه بیت ack برابر 1 می‌باشد، بله معتبر است.

3 )

چون بیت های FIN و ACK این بسته برابر با یک هستند پس این بسته مربوط به اتمام یک اتصال TCP است.
(مرحله close_wait )

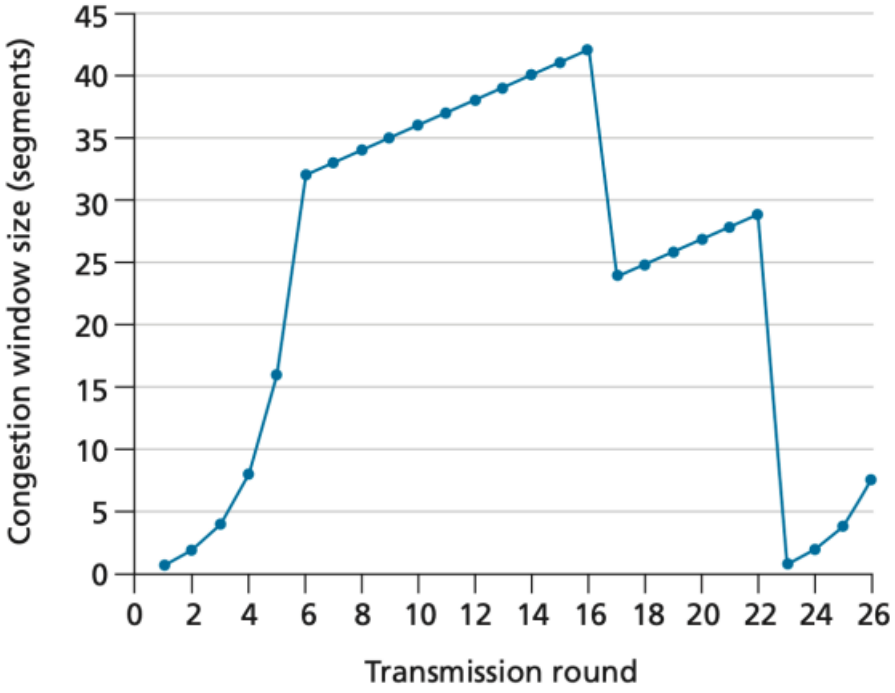
4 )

$$sequence\ number = 6BDBCF9E(hex) = \ 1809567646$$

این عدد نشان‌دهنده شماره اولین بایت فرستاده شده در قسمت data این سگمنت است.

## سوال ششم:

سؤال ۶ (۲۰ نمره)




۱) بازه *slow start* را مشخص کنید.
۲) بازه *congestion control* را مشخص کنید.
۳) در ۱۶ مین دور چه رویدادی رخ داده است؟
۴) مقادیر *ssthresh* و *cwnd* در ۱۷ مین دور چیست؟
۵) *ssthresh* ابتدایی چقدر است؟
۶) در کدام دور *timeout* و در کدام دور ۳ *ack* تکراری دریافت شده است؟
۷) آیا رویدادی که در دور ۱۶ام رخداده است لزوما به معنی *packet drop* است؟

## پاسخ سوال ششم:

1 ) بازه $[23, 26]$ و$[1, 6]$ در حالت slow start هستند چون رشد نمایی دارند.

2 ) بازه $[6, 16]$ و $[17,22]$ اندازه cwind به طور خطی زیاد شده است. پس در حالت congestion control هستیم.

3 ) در 16امین دور، packet loss به وسیله triple duplicate ACK شناسایی شده است. زیرا اگر timeout بود، مقدار cwind برابر با 1 می‌شد.

4 ) در 16امین دور مقدار هر دو برابر 42 است. پس از triple ack، مقدار ssthresh برابر نصف cwind مرحله قبل یعنی 21 می‌شود. مقدار cwind نیز برابر $21 + 3 = 24$ خواهد بود (از congestion به fast recovery رفته‌ایم.)

5 ) برابر 32 است زیرا هنگامی که مقدار cwind به 32 رسید، از حالت slow start به حالت congestion control می‌رویم.

6 ) در دور 16ام مطابق توضیحات بخش 4، سه ack تکراری دریافت شده است.

در دور 22 نیز timeout رخ داده است زیرا در مرحله بعد از آن مقدار cwind برابر 1 شده است.

7 ) خیر؛ لزوما به معنی packet drop نیست و ممکن است به خاطر به‌هم‌ریختگی در ترتیب رخ بدهد و بسته‌های بعدی زودتر از بسته با شماره کمتر برسند.

## سوال هفتم:

با تمرین عملی تحویل داده خواهد شد.