شبکه‌های کامپیوتری
پاییز ۱۴۰۴

تمرین دوم

دکتر مهدی دولتی

سوال ۱

صحیح یا غلط بودن موارد زیر را مشخص کنید. در موارد غلط، جمله را اصلاح و یا دلیل غلط بودن جمله را بیان کنید. (۱۰ نمره)

۱. TCP برای demultiplexing از یک tuple چهارتایی شامل آدرس‌های IP مبدأ و مقصد و شماره پورت‌ها استفاده می‌کند.

۲. به دلیل نداشتن congestion control، پروتکل UDP می‌تواند داده‌ها را با هر سرعتی که برنامه اجازه دهد ارسال کند.

۳. برنامه‌هایی که از UDP استفاده می‌کنند می‌توانند به این پروتکل برای بافر کردن و مرتب‌سازی دوباره‌ی datagramها اعتماد کنند.

۴. در TCP، برای برقراری کامل اتصال، فرستادن SYN از طرف کلاینت کافی است.

۵. مکانیزم TCP Fast Retransmit تنها زمانی فعال می‌شود که فرستنده سه ACK تکراری دریافت کند، زیرا این وضعیت به‌طور قوی نشان‌دهنده‌ی از دست رفتن یک segment است.

۶. فقط ارسال‌های مجدد ضروری بر throughput تأثیر دارند؛ بسته‌های تکراری غیرضروری (unneeded duplicates) توسط TCP نادیده گرفته می‌شوند و اثری ندارند.

۷. زمانی که یک segment خارج از ترتیب دریافت شود (نشان‌دهنده‌ی وجود شکاف در جریان داده)، TCP بلافاصله یک duplicate ACK ارسال می‌کند که شامل شماره دنباله‌ی بایت بعدی مورد انتظار است تا احتمال از دست رفتن داده را اعلام کند.

۸. در TCP Reno، نرخ ارسال (یا اندازه‌ی congestion window) زمانی که از طریق سه ACK تکراری از دست رفتن بسته تشخیص داده شود، به نصف کاهش می‌یابد؛ اما وقتی از طریق timeout تشخیص داده شود، به اندازه‌ی یک MSS بازنشانی می‌شود — این رفتار مطابق با اصل multiplicative decrease است.

پاسخ:

۱. صحیح

۲. صحیح

۳. غلط، زیرا UDP هیچ مکانیزمی برای بافر کردن یا مرتب‌سازی دوباره‌ی datagramها ندارد؛ هر بسته به‌صورت مستقل تحویل داده می‌شود و اگر برنامه نیاز به حفظ ترتیب داشته باشد، باید خودش در سطح application آن را پیاده‌سازی کند.

۴. غلط، زیرا در TCP، برقراری کامل اتصال نیازمند سه مرحله (3-way handshake) است که در آن هر دو طرف باید SYN ارسال کنند و در نهایت ACK دریافت شود؛ تنها ارسال SYN از طرف کلاینت اتصال را برقرار نمی‌کند.

۵. صحیح

۶. غلط، زیرا حتی ارسال‌های مجدد غیرضروری (unneeded duplicates) نیز بر throughput تأثیر منفی دارند؛ این بسته‌ها هرچند ممکن است در سمت گیرنده نادیده گرفته شوند، اما پهنای باند شبکه را اشغال می‌کنند و باعث هدررفت ظرفیت مؤثر لینک و کاهش کارایی کلی انتقال داده می‌شوند.

۷. صحیح

۸. صحیح

سوال ۲

به سوالات زیر پاسخ کوتاه دهید. (۱۰ نمره)

آ) با توجه به ساختار headerهای TCP، کاربرد فلگ‌های (C, E, U, P) را بنویسید.

ب) اگر داخل یک ساختار خط لولهٔ ۵تایی، اندازهٔ هر پکت ۱۰۰۰ بایت، نرخ ارسال ۱ Mb/s و مقدار RTT برابر با ۱۲۰ میلی‌ثانیه باشد، مقدار بهره‌وری خط لوله را حساب کنید.

ج) هزینهٔ ارسال مجدد در Go-Back-N و Stop-and-Wait وقتی نرخ خطای بیتی زیاد می‌شود را مقایسه کنید. همچنین تأثیر این موضوع را بر throughput شبکه‌ای با ازدحام توضیح دهید.

د) در الگوریتم Selective Repeat با sequence number دو بیتی و window size = 3 چه خطایی ممکن است رخ دهد؟ کوتاه توضیح دهید چگونه گیرنده ممکن است پکت‌ها را اشتباه دریافت و تفسیر کند و چگونه می‌توان این مشکل را رفع کرد.

ه) در TCP وقتی فرستنده پکت‌های SYN یا FIN را با Sequence Number = x ارسال می‌کند، گیرنده پکت ACK با Acknowledgment Number = x + 1 ارسال می‌کند. این یعنی هر دو پکت SYN و FIN هر کدام یک Sequence Number مصرف می‌کنند. برای هرکدام توضیح دهید آیا مصرف یک Sequence Number ضروری است یا خیر. اگر بله است، یک سناریو ارائه دهید که در آن افزایش ندادن باعث ابهام شود.

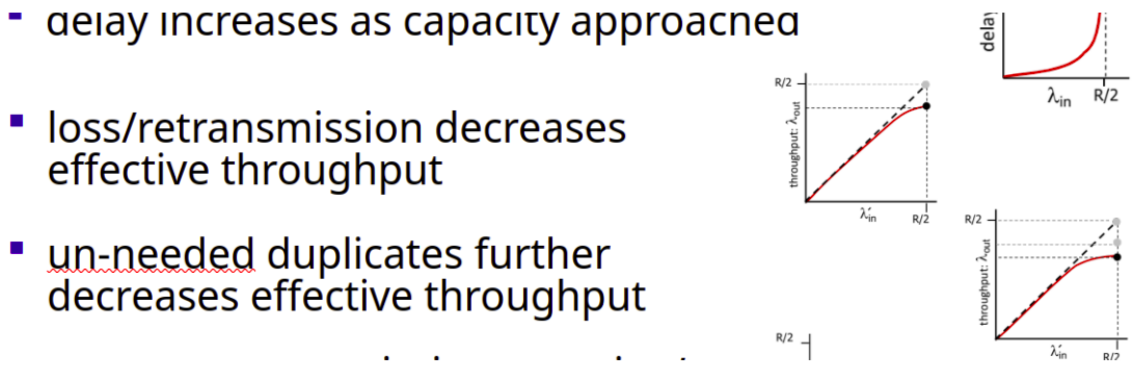
پاسخ:

آ)
- CWR (Congestion Window Reduced): زمانی استفاده می‌شود که فرستنده پیامی مبنی بر وقوع ازدحام شبکه (ECN) دریافت کرده و نرخ ارسال خود را کاهش داده است.
- ECE (ECN Echo): برای اعلام وقوع ازدحام شبکه به فرستنده به‌کار می‌رود، در حالتی که قابلیت ECN فعال باشد.
- URG (Urgent): نشان می‌دهد بخشی از داده‌ها فوری و با اولویت بالا هستند و باید سریع‌تر پردازش شوند. در این حالت فیلد Urgent Pointer معتبر است.
- PSH (Push): به گیرنده اطلاع می‌دهد داده‌ها را بدون انتظار برای پر شدن بافر، بلافاصله به برنامهٔ بالادست تحویل دهد (مثلاً در ترافیک تعاملی مانند SSH یا Telnet).

ب)

$$U = \frac{N \times L/R}{RTT + (L/R)} = \frac{5 \times (1000 \times 8/10^6)}{0.12 + (1000 \times 8/10^6)} = 0.3125$$

ج) در Go-Back-N چون با هر خطا نیاز به ارسال مجدد تمام پکت‌های بعدی وجود دارد، هزینهٔ ارسال مجدد بسیار بیشتر از Selective Repeat است. با افزایش نرخ خطا (p)، امید ریاضی تعداد ارسال مجدد زیاد می‌شود و با توجه به رابطه‌ی $1-(1-p)^L \approx pL$، احتمال خطا در طول بسته افزایش می‌یابد. در شبکه‌های دارای ازدحام، این ارسال‌های اضافی باعث افزایش unneeded duplication و در نتیجه کاهش محسوس throughput می‌شود.

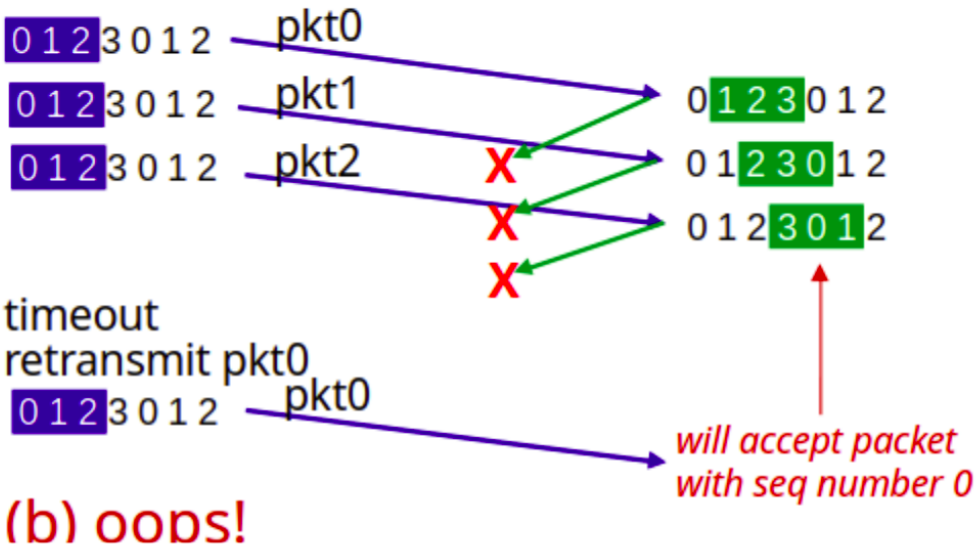


شکل ۱: تاثیر Retransmit و Un-needed Duplication بر شبکه

د) به دلیل محدود بودن فضای شمارهٔ دنباله (فقط چهار مقدار ممکن)، ممکن است گیرنده پکت‌های retransmit شده را به‌عنوان پکت جدید در نظر بگیرد. برای رفع این مشکل، باید شرط زیر رعایت شود:

$$\text{Window Size} \leq \frac{\text{Sequence Space}}{2}$$

در نتیجه اندازهٔ پنجره باید حداکثر برابر با نصف تعداد مقادیر قابل شمارش sequence number باشد.

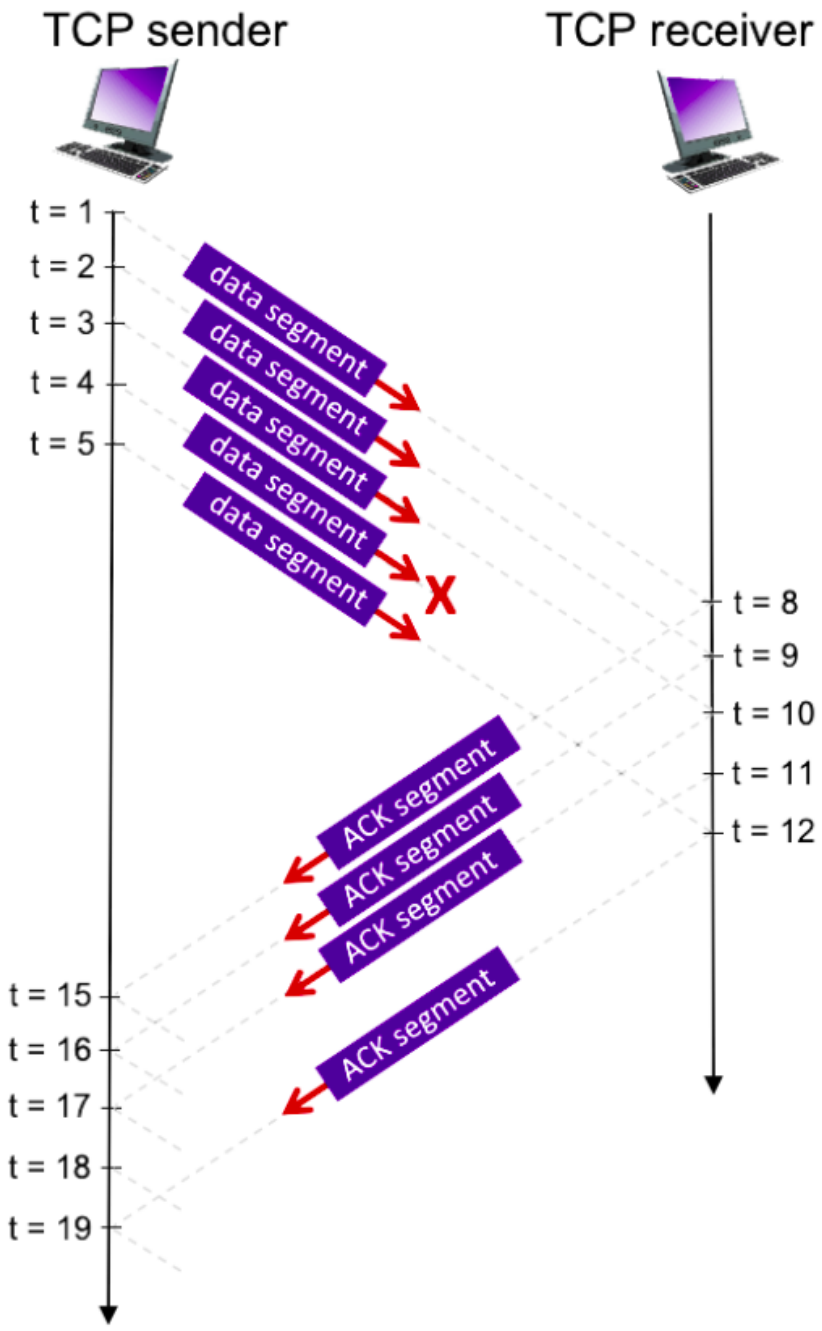


شکل ۲

ه)
- FIN: افزایش Sequence Number ضروری است، زیرا در غیر این صورت گیرنده نمی‌تواند تشخیص دهد که ACK مربوط به بستهٔ دادهٔ آخر است یا مربوط به پیام پایان ارتباط (FIN). این امر می‌تواند منجر به تفسیر اشتباه بسته‌ها شود.
- SYN: افزایش Sequence Number از نظر فنی ضروری نیست، زیرا در ابتدای ارتباط هیچ داده‌ای وجود ندارد و مقدار Sequence Number از ابتدا تصادفی انتخاب می‌شود. اما برای جلوگیری از پیچیدگی و حفظ سازگاری بین دو حالت، در استاندارد TCP هر دو پکت SYN و FIN یک Sequence Number مصرف می‌کنند.

سوال ۳

آ) شکل زیر را در نظر بگیرید:



شکل ۳

فرض کنید مقدار Sequence Number در ابتدا برابر با ۱۶۶ است و هر سگمنت دارای ۱۹۹ بایت داده می‌باشد. مقدار ACK و Sequence Number را برای تمامی پکت‌ها بنویسید.

ب) فرض کنید مقادیر تخمینی فعلی TCP برای زمان رفت‌وبرگشت (EstimatedRTT) و انحراف در زمان رفت‌وبرگشت (DevRTT) به‌ترتیب برابر با ۳۱۰ میلی‌ثانیه و ۴۲ میلی‌ثانیه هستند. سه مقدار اندازه‌گیری‌شده‌ی بعدی برای RTT به‌ترتیب برابرند با: ۳۴۰، ۳۱۰ و ۳۲۰ میلی‌ثانیه. مقادیر جدید DevRTT، EstimatedRTT، و Timeout Interval را پس از هر سه اندازه‌گیری محاسبه کنید. از مقادیر $\alpha = 0.125$ و $\beta = 0.25$ استفاده کنید و پاسخ‌ها را تا دو رقم اعشار گرد نمایید.

پاسخ:

آ) مقادیر Sequence Number و ACK Number برای هر پکت در جدول زیر آمده است:

شماره پکت (T)	Number Seq	Number ACK
۱	۱۶۶	۳۶۵
۲	۳۶۵	۵۶۴
۳	۵۶۴	۷۶۳
۴	۷۶۳	X
۵	۹۶۲	۷۶۳

جدول ۱: مقادیر Seq و ACK برای پکت‌های متوالی

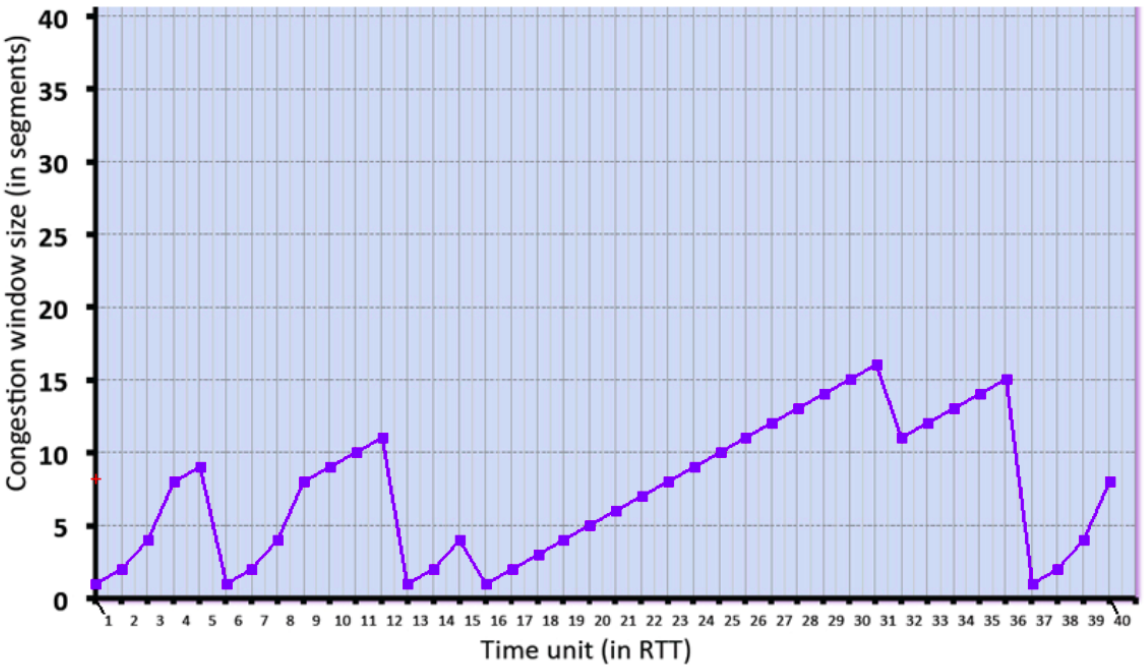
ب) برای محاسبات از روابط استاندارد TCP RTT Estimation استفاده شده است. (به اسلایدها مراجعه کنید) نتایج به‌دست‌آمده در جدول زیر آمده است:

#	**EstimatedRTT (ms)**	**DevRTT (ms)**	**TimeoutInterval (ms)**
1	313.75	39.00	469.75
2	313.28	30.19	434.03
3	314.12	24.32	411.40

جدول ۲: محاسبهٔ مقادیر جدید EstimatedRTT، DevRTT و TimeoutInterval

سوال ۴

شکل زیر را در نظر بگیرید:



شکل ۴

این شکل تغییرات اندازه‌ی پنجره‌ی ازدحام (congestion window) در پروتکل TCP را در ابتدای هر واحد زمانی نشان می‌دهد (هر واحد زمان برابر با یک RTT است). در مدل انتزاعی این مسئله فرض می‌شود که TCP در ابتدای هر واحد زمانی، یک پنجره کامل از بسته‌ها (flight) به اندازه‌ی مقدار cwnd ارسال می‌کند. نتیجه‌ی ارسال این مجموعه از بسته‌ها می‌تواند یکی از حالت‌های زیر باشد:

- همه‌ی بسته‌ها در پایان واحد زمانی تأیید (ACK) می‌شوند.
- برای اولین بسته، انقضای زمان (timeout) رخ می‌دهد.
- برای اولین بسته، سه تأیید تکراری (triple duplicate ACK) دریافت می‌شود.

در این مسئله خواسته شده است دنباله‌ی رویدادها (ACKها و ازدست‌رفتن بسته‌ها) را بازسازی کنید که باعث تغییرات مشاهده‌شده در مقدار cwnd در نمودار شده‌اند. مقدار اولیه‌ی cwnd = 1 و مقدار اولیه‌ی ssthresh = 8 (که با علامت + قرمز نشان داده شده است) در نظر گرفته می‌شود.

آ) بازه‌های مربوط به slow start را مشخص کنید.

ب) بازه‌های مربوط به congestion avoidance را مشخص کنید.

ج) بازه‌های مربوط به fast recovery را نشان دهید.

د) در کدام لحظه(ها) timeout رخ داده است؟

ه) در کدام لحظه(ها) triple duplicate ACK رخ داده است؟

و) در کدام لحظه‌ها مقدار ssthresh تغییر کرده است؟

پاسخ:

آ) بازه‌های مربوط به slow start:

$$[1, 2, 3, 6, 7, 8, 13, 14, 15, 16, 37, 38, 39]$$

ب) بازه‌های مربوط به congestion avoidance:

$$[4, 5, 9, 10, 11, 12, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 33, 34, 35, 36, 40]$$

ج) بازه‌های مربوط به fast recovery:

$$[32]$$

د) لحظه‌های وقوع timeout:

$$[5, 12, 15, 36]$$

ه) لحظه‌های وقوع triple duplicate ACK:

$$[31]$$

و) لحظه‌هایی که مقدار ssthresh تغییر کرده است:

$$[5, 12, 15, 31, 36]$$

سوال ۵:

میزبان A پس از handshake می‌خواهد یک فایل ۵۰۰۰ بایتی را از میزبان B دریافت کند. در پروتکل مورد استفاده $\text{MSS} = 4\ \text{Byte}$ و $\text{Timeout} = 4\ \text{RTT}$ است. همچنین از Cumulative Acknowledgement استفاده نمی‌شود و هر بسته باید جداگانه تایید شود. می‌دانیم در فرآیند انتقال فقط اولین بسته ارسالی از B گم می‌شود. با فرض این‌که ظرفیت ارسال نامحدود، RTT ثابت و زمان پردازش بسته‌ها در هر کدام از میزبان‌ها قابل صرف‌نظر باشد، در هر کدام از روش‌های زیر برای انتقال مطمئن (Reliable) داده، چند RTT زمان لازم است تا میزبان B از دریافت کامل فایل توسط میزبان A اطمینان حاصل کند. راه‌حل خود را برای رسیدن به جواب ذکر کنید. (۱۰ نمره)

(آ) stop-and-wait

(ب) Go-back-N در صورتی که اندازه‌ی پنجره‌ی ارسال 3MSS باشد.

(ج) Selective repeat در صورتی که اندازه‌ی پنجره‌ی ارسال 3MSS باشد.

(د) Selective repeat در صورتی که اندازه‌ی پنجره‌ی ارسال 5MSS باشد.

پاسخ:

تعداد کل بسته‌هایی که باید ارسال شود برابر است با: $\frac{5000}{4} = 1250$

(آ) در این روش پس از handshake بستهٔ اول ارسال می‌شود. طبق صورت سؤال این بسته گم می‌شود و تأیید آن به میزبان B نمی‌رسد. پس از یک timeout بستهٔ اول دوباره ارسال می‌شود و پس از آن در هر RTT یک بسته توسط میزبان A دریافت و تایید آن به B می‌رسد. در نتیجه در کل طول می‌کشد تا میزبان B از دریافت کامل فایل اطمینان حاصل کند.

$$1\,\text{Timeout} + 1250\,\text{RTT} = 1254\,\text{RTT}$$

(ب) در این روش ابتدا میزبان B با توجه به اندازهٔ پنجرهٔ ارسال، سه بسته را ارسال می‌کند و پس از رخ دادن timeout با وجود این‌که بسته‌های دوم و سوم گم نشده است هر سه بسته را دوباره ارسال می‌کند. پس از آن با صرف نظر کردن از زمان پردازش بسته‌ها، در هر RTT سه بسته توسط میزبان A دریافت و تأیید آن‌ها به B می‌رسد. در نتیجه زمان کل برابر است با

$$1\,\text{Timeout} + \lceil 1250/3 \rceil\,\text{RTT} = 4 + 417 = 421\,\text{RTT}$$

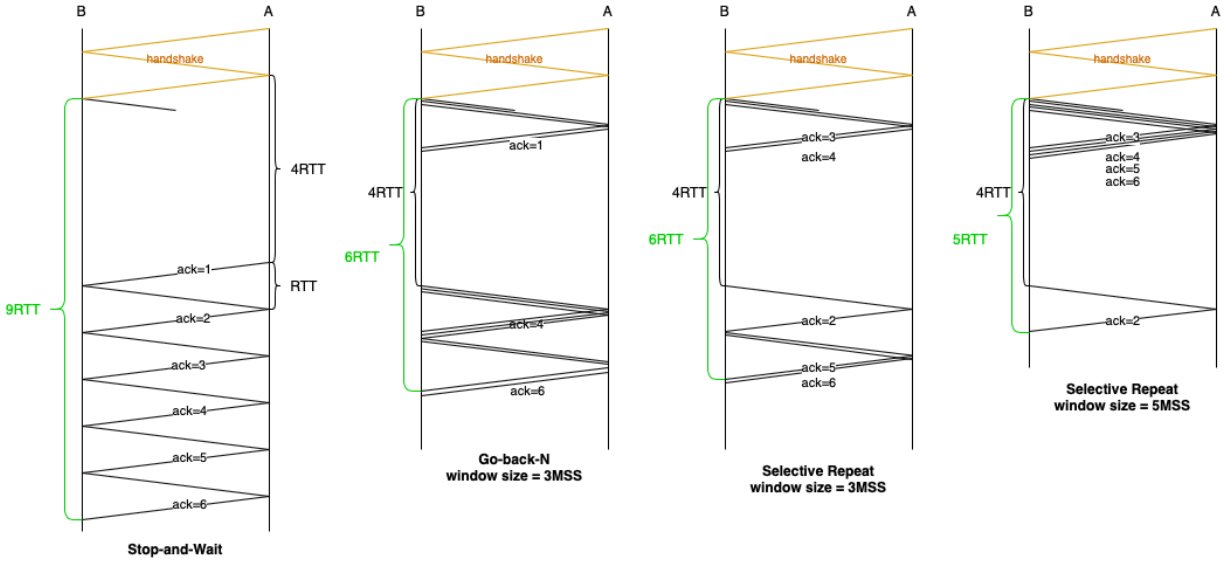
(ج) در این روش میزبان B سه بسته را ارسال می‌کند و پس از رخ دادن timeout فقط بستهٔ اول را دوباره ارسال می‌کند. اما چون اندازهٔ پنجرهٔ ارسال برابر سه است و تأیید اولین بسته دریافت نشده نمی‌تواند بستهٔ جدیدی ارسال کند. پس از یک RTT تأیید بستهٔ اول به B می‌رسد. تأییدهای بستهٔ دوم و سوم نیز پیش از این به B رسیده است. در نتیجه سه بستهٔ بعدی را ارسال می‌کند و از آن به بعد در هر RTT سه بسته توسط میزبان A دریافت و تأیید آن‌ها به B می‌رسد. زمان کل برابر است با

$$1\,\text{Timeout} + 1\,\text{RTT} + \lceil 1247/3 \rceil\,\text{RTT} = 4 + 1 + 416 = 421\,\text{RTT}$$

(د) این قسمت مشابه قسمت قبل خواهد بود با این تفاوت که اندازهٔ پنجرهٔ ارسال پنج است و در نتیجه میزبان پنج بسته را ارسال می‌کند. زمان کل برابر با مقدار زیر خواهد بود:

$$1\,\text{Timeout} + 1\,\text{RTT} + \lceil 1245/5 \rceil\,\text{RTT} = 4 + 1 + 249 = 254\,\text{RTT}$$

در شکل زیر می‌توانید نمودار ارسال برای نسخهٔ کوچک‌شدهٔ مسئله را ببینید. میزبان B قصد ارسال ۵ بسته به میزبان A دارد و برای سادگی فرض شده است که اندازهٔ هر بسته ۱ بایت است و به‌صورت ترتیبی ۱، ۲، ۳... شروع می‌شود.



شکل ۵: نمودار ارسال و تأیید بسته‌ها

سوال ۱ عملی

همانطوری که در پیوست سوالات مشاهده کردید، ۳ فایل با پسوند pcap آماده شده:

آ) درباره این نوع فایل‌ها تحقیق کنید و به صورت کوتاه توضیح دهید، سپس با استفاده از ابزار wireshark (یا هر ابزار دیگری برای آنالیز بسته‌های شبکه) بسته‌ها را باز کنید و برای هر فایل موارد زیر را برای پکت SYN مشخص کنید:

- Seq Number (مقدار تصادفی که در شروع ارتباط انتخاب می‌شود)
- Source Port
- Destination Port
- Window

هدف این سوال مقدمه‌ای بر ابزارهای آنالیز بسته‌های شبکه و بررسی بسته‌ها در لایه Transport است. این داده‌ها مربوط به دو برنامه هستند که روی یک ارتباط لایه Transport داده‌های مختلف ارسال و دریافت می‌کنند.

هر فایل مربوط به داده‌های دریافت روی یک interface معیوب است. ایرادات این interfaceها ممکن است هرکدام از ایرادات زیر باشد (هر interface فقط یک ایراد از موارد زیر را دارد و یکی از ایرادات زیر اضافی است):

(آ) Reorder: بسته‌های شبکه را با ترتیب‌های رندوم دریافت می‌کند.

(ب) High Miss & Latency: بسته‌های شبکه در این حالت احتمال Miss شدن نسبتا بالاتری دارد و نسبت به دو فایل دیگر دارای latency بیشتری هستند.

(ج) Duplication: بسته‌ها به صورت تصادفی به صورت duplicated به دست دریافت‌کننده می‌رسد.

(د) Corruption: داده‌های این بسته‌ها به صورت تصادفی flip شده‌اند، بنابراین این بسته‌ها به صورت تصادفی دچار corruption می‌شوند.

ب) به صورت تئوری کوتاه بررسی کنید درصورت بروز چنین ایراداتی چه اتفاقاتی در لایه Transport رخ می‌دهد؟

ج) از ابزار آنالیز خود استفاده کنید تا نمودارهای مربوط به Throughput، RTT، Window Scaling، Time Sequence را برای هر فایل بکشید و تحلیل کنید. (داخل Wireshark به بخش Statistics مراجعه کنید.)

د) از ابزار آنالیز خود استفاده کنید تا اتفاقات لایه Transport را بررسی کنید. سپس مشخص کنید هر فایل مربوط به کدام ایراد است؟ (در ابزار Wireshark به بخش Analyze مراجعه کنید.)

پاسخ:

آ) برای بسته‌ها به ترتیب شماره نوشته شده:

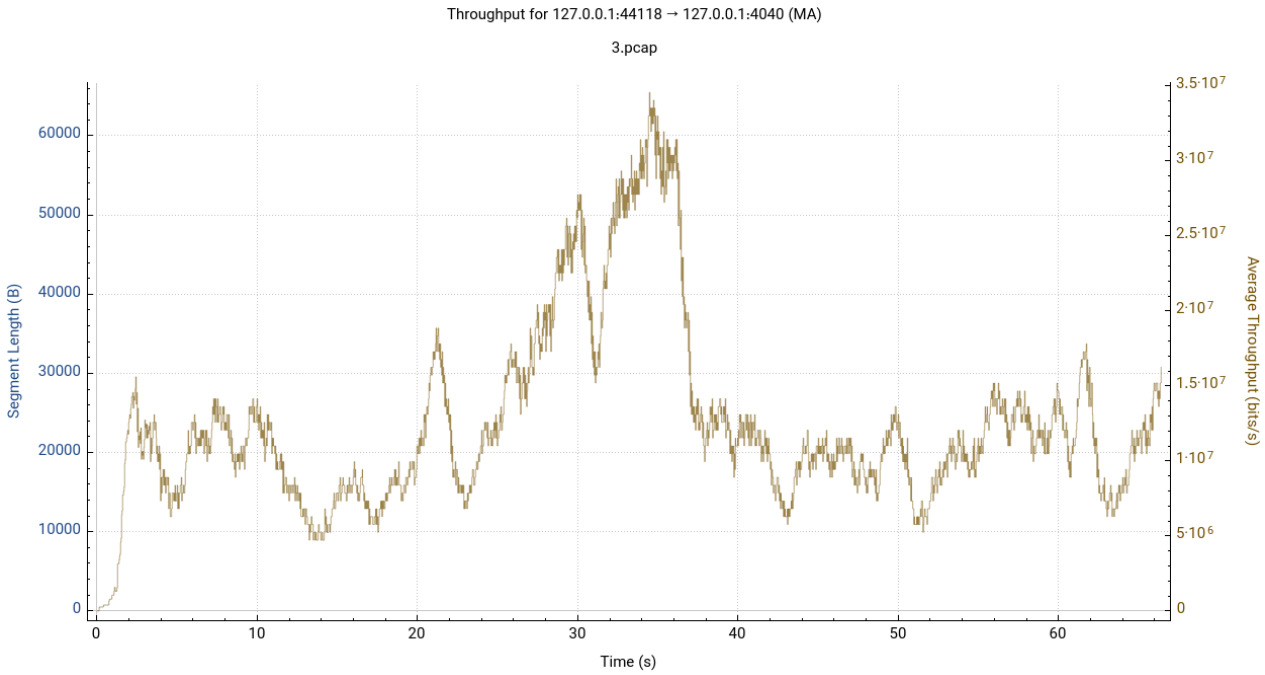
- Seq Number: ۹۵۹۱۱۴۴۲
- Source Port: ۳۹۵۹۰
- Destination Port: ۲۰۲۰
- Window: ۶۵۴۹۵

- Seq Number: ۲۲۹۸۵۱۲۱۲۱
- Source Port: ۵۶۰۷۸
- Destination Port: ۳۰۳۰
- Window: ۶۵۴۹۵

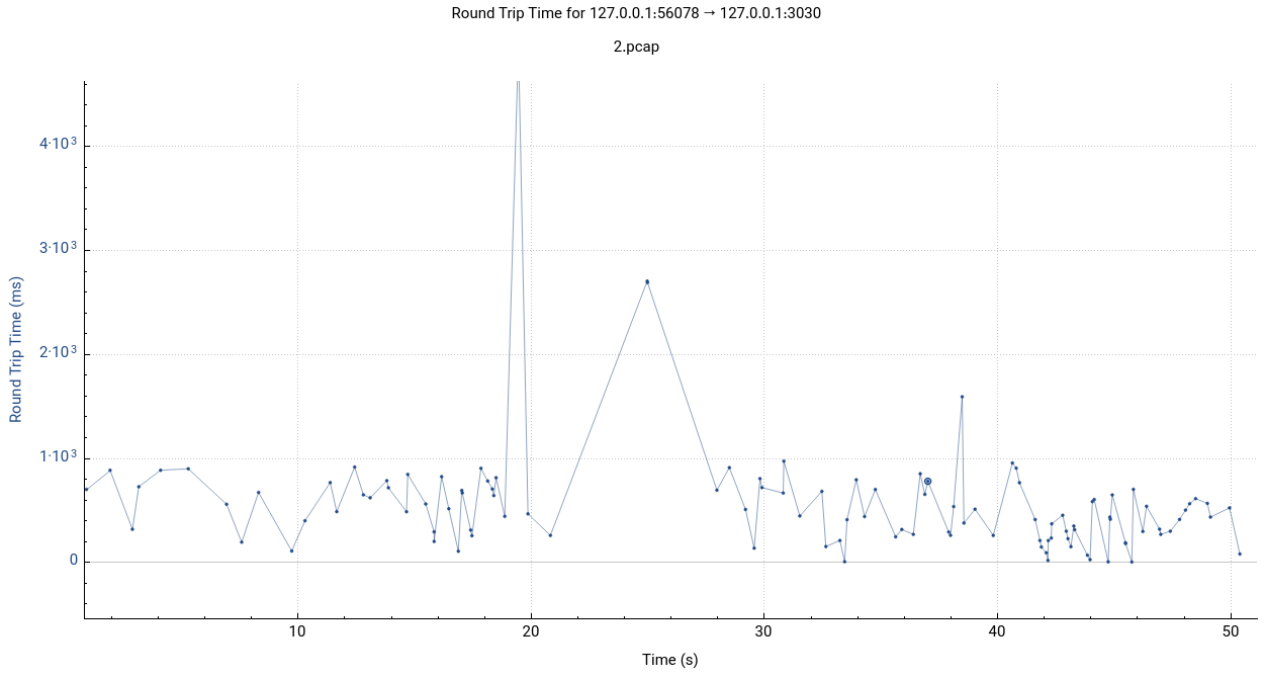
- Seq Number: ۱۳۷۹۵۷۳۷۰۹
- Source Port: ۴۴۱۱۸
- Destination Port: ۴۰۴۰
- Window: ۶۵۴۹۵

ب)

ج) دو مورد به عوان مثال آمده:



شکل ۶



شکل ۷

د) ● یک مربوط به dupication است

- دو مربوط به loss و delay high است
- سه مربوط به reorder است

سوال ۲ عملی

پیاده‌سازی **Go-Back-N** روی **UDP**

(۱) هدف

در این تمرین قصد داریم یک پروتکل انتقال داده‌ی قابل اعتماد (Reliable Data Transfer) را با استفاده از پروتکل UDP پیاده‌سازی کنیم. UDP ذاتاً غیرقابل اعتماد است و هیچ تضمینی برای رسیدن، به‌ترتیب یا عدم تکرار بسته ارائه نمی‌دهد. هدف این تمرین پیاده‌سازی یک لایه‌ی انتقال قابل اعتماد بر روی UDP است که رفتار آن مشابه پروتکل Go-Back-N (GBN) باشد.

(۲) مشخصات پیاده‌سازی

(۱.۲) ساختار کلی

پروژه شامل دو بخش client و server است.

(۲.۲) عملکرد

- فرستنده داده‌ها را به بسته‌های UDP تقسیم می‌کند و برای هر بسته Sequence Number اختصاص می‌دهد.
- گیرنده در GBN فقط بزرگ‌ترین شماره‌ی توالی دریافت‌شده‌ی in-order را نگه می‌دارد و برای آن cumulative ACK ارسال می‌کند؛ بسته‌های Out-of-Order را discard می‌کند (ذخیره‌سازی در بافر سمت گیرنده وجود ندارد).
- فرستنده یک Send Window از حداکثر N بسته‌ی ارسال‌شده ولی تأییدنشده نگه می‌دارد.
- فرستنده حداقل یک تایمر برای قدیمی‌ترین بسته‌ی ارسال‌شده و تأییدنشده نگه می‌دارد؛ در صورت انقضای تایمر (Timeout) بدون دریافت ACK مربوط، همان بسته و همه‌ی بسته‌های بعد از آن در

پنجره مجدداً ارسال می‌شوند (go-back-n).

- با دریافت cumulative ACK معتبر (مثلاً ACK(k) یعنی تا k تحویل in-order شده)، تمام بسته‌ها با شماره کوچک‌تر و مساوی k از پنجره حذف و پنجره به جلو حرکت می‌کند.

(۳.۲) ویژگی‌ها

- Pipelining (ارسال هم‌زمان چند بسته تا سقف Window)
- Cumulative ACK در گیرنده
- عدم نگه‌داری Out-of-Order در گیرنده (discard)
- تشخیص و جبران Packet Loss با Retransmission تجمعی پس از Timeout
- امکان شبیه‌سازی Packet Loss و Reordering برای تست عملکرد

(۴.۲) محدودیت‌ها

- استفاده از هیچ کتابخانه‌ی خارجی برای مدیریت GBN مجاز نیست (فقط socket، threading، time، random و کتابخانه‌های استاندارد پایتون مجاز است).
- ارسال پیام‌ها باید صرفاً از طریق UDP انجام شود (نه TCP).

(۳) تحویل نهایی

(۳.۱) **README**

شامل موارد زیر:

- نام، نام‌خانوادگی، شماره دانشجویی
- توضیح مختصر درباره‌ی طراحی و ساختار کد (معماری Sender/Receiver، ساختار Packet، نحوه‌ی مدیریت Window و Timeout)
- دستور اجرای سرور و کلاینت (نمونه دستورات)
- توضیح سناریوهای تست، نحوه‌ی اجرای تست‌ها و نتایج به‌دست‌آمده (بخش ۴ را ببینید)

Makefile (۳.۲)

شامل دستورات زیر:

- make run-server
- make run-client
- make test (اجرای اسکریپت تست توضیح‌داده‌شده در بخش ۴)

(۳.۳) پوشه‌ها

- client شامل کد فرستنده، server شامل کد گیرنده ، tests شامل فایل‌ها و اسکریپت‌های تست

۴) تست

۴.۱) طراحی تست‌ها

مجموعه‌ای از تست‌ها برای ارزیابی عملکرد طراحی و خلاصه‌ی نتایج را در README گزارش دهید.
تست‌ها باید کیس‌های مختلف از جمله packet loss و reordering را پوشش دهند.
می‌توانید خطاهای شبکه (Loss/Reorder/Delay) را در کد کلاینت یا سرور، یا با استفاده از یک پروکسی مانند tests/netem_proxy.py شبیه‌سازی کنید (هر روش صحیح دیگری نیز پذیرفته است).

۴.۲) اجرای خودکار

با اجرای دستور `make test` تمام تست‌ها باید اجرا شوند و خروجی شامل Digest پیام‌ها (ارسالی و دریافتی) باشد.