موسسه بابان

انتشارات بابان و انتشارات راهیان ارشد

درس و کنکور ارشد

شبکه‌های کامپیوتری

(حل تشریحی سوالات دولتی ۱۳۹۷)

ویژه‌ی داوطلبان کنکور کارشناسی ارشد مهندسی کامپیوتر و IT

براساس کتب مرجع

کراس راس و لئون گارسیا

ارسطو خلیلی‌فر

تست‌های فصل چهارم

۷۷- شبکه‌ای را که در آن امکان برخورد (collision) بسته ها وجود دارد و پروتکل CSMA/CD فعال است را در نظر بگیرید. در این شبکه زمان انتشار (propagation) بین نود A و نود B یک میلی‌ثانیه (mSec) است. در لحظهٔ $t = ۰$ نود A بسته ای را با نرخ ۴ مگابیت بر ثانیه ارسال می‌کند و در لحظهٔ $t = ۰/۸ \, m\,sec$ نود B بسته‌ای را با نرخ ۴ مگابیت بر ثانیه ارسال می‌کند. به‌ترتیب از راست به چپ حداقل اندازهٔ بستهٔ A چند بایت باشدکه A متوجه برخورد شود و حداقل اندازهٔ بستهٔ B چند بایت باشدکه B متوجه برخورد شود؟

۱) ۱۰۰۰ _ ۱۰۰۰

۲) ۶۰۰ _ ۱۴۰۰

۳) ۱۰۰ _ ۹۰۰

۴) ۶۴ _ ۶۴

۷۷- گزینه (۳) صحیح است.

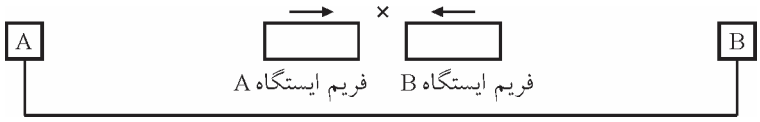
داده‌های مسئله به صورت زیر است:

$$T_{P(AB)} = \frac{D_{AB}}{V} = ۱ms,\quad R = ۴Mbps = ۴\times۱۰^{۶}\ bps,\quad T_{Send(A)} = ۰\ ms,\quad T_{Send(B)} = ۰/۸\ ms$$

انواع برخورد

انواع برخورد بر سه حالت می‌باشد:

حالت اول



ایستگاه A به خط گوش می‌دهد و آن را آزاد می‌یابد و فریم خود را روی خط قرار می‌دهد. اندکی بعد، قبل از اینکه به دلیل پدیده تأخیر انتشار (T_P) فریم بر روی کانال، این فریم به ایستگاه B برسد، ایستگاه B نیز به خط گوش می‌دهد و او نیز خط را آزاد می‌یابد و فریم خود را روی کانال قرار می‌دهد و تصادم پیش می‌آید. بنابراین دلیل اصلی تصادم حالت اول، پدیده تأخیر انتشار، در کانال است.

شرط کشف تصادم در فریم در حال ارسال

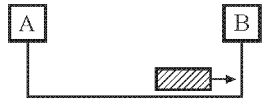
مدار کشف تصادم از هنگام شروع ارسال بیت اول فریم تا پایان ارسال بیت آخر فریم به کانال گوش می‌دهد و چنانچه توان مشاهده شده بر روی کانال بیش از توان سیگنال ارسالی خودش باشد، متوجه وجود سیگنال دیگری بر روی کانال می‌شود که نشانه تصادم است.

توجه: اما مشکل اینجاست که ممکن است تصادمی پیش آید و موج حاصل از تصادم با فریم ایستگاه مقابل، بعد از اتمام انتقال فریم موردنظر بر روی کانال انتقال به ایستگاه مربوطه برسد. در این صورت مدار کشف تصادم ایستگاه مربوطه متوجه وقوع تصادم نخواهد شد.

توجه: وقتی یک فریم در زمان T_F، به طور کامل بر روی کانال انتقال قرار گرفت، دیگر جلوی آن را نمی‌توان گرفت و شروع به حرکت در کانال انتقال می‌کند. هنگامی که کودکی بر روی سُرسُره بازی قرار می‌گیرد دیگر سُر می‌خورد و نمی‌توان جلوی آن را گرفت، فرصت برای ممانعت از سُر خوردن کودک تا زمانی وجود دارد که هنوز کودک به طور کامل بر روی سُرسُره قرار نگرفته است.

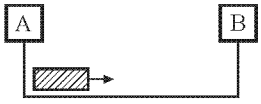
شرط کشف تصادم برای فریم در حال انتقال بر روی کانال انتقال چیست؟

به شکل زیر توجه کنید:

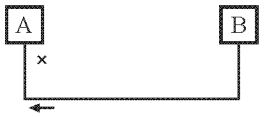


فریم ارسالی از A تقریباً در لحظه T_P به B می‌رسد.

(ب)

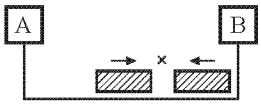


(الف)



نویز انفجاری در لحظه T_P به A برمی‌گردد.

(د)



(ج)

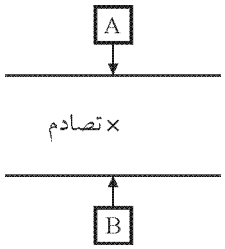
توجه: بنابراین زمان انتقال فریم بعدی بر روی کانال انتقال باید بیشتر از زمان رفت و برگشت حاصل از تأخیر انتشار باشد تا ایستگاه فرستنده بتواند نویز انفجاری ایجاد شده در نزدیکی ایستگاه B را احساس کند تا ادامه انتقال فریم بعدی را متوقف کند. «**به قول معروف جلوی ضرر از هر جا گرفته شود، منفعت است!**» به عبارت دیگر، شرط کشف تصادم در فریم در حال ارسال به صورت زیر است:

$$T_F \geq 2T_p \quad \Rightarrow \quad \frac{L}{R} \geq 2\frac{D}{v}$$

توجه: بازه تشخیص تصادم حداقل $2T_p$ است. این زمان را زمان رفت و برگشت یا RTT (Round Trip Time) نیز می‌نامند.

حالت دوم

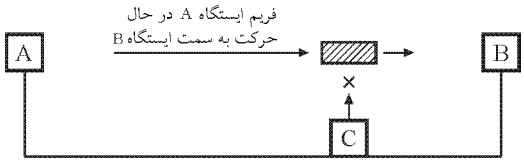
دو ایستگاه A و B همزمان به کانال گوش می‌دهند و هر دو آن را آزاد می‌یابند و با هم فریم خود را بر روی کانال قرار می‌دهند و تصادم رخ می‌دهد.



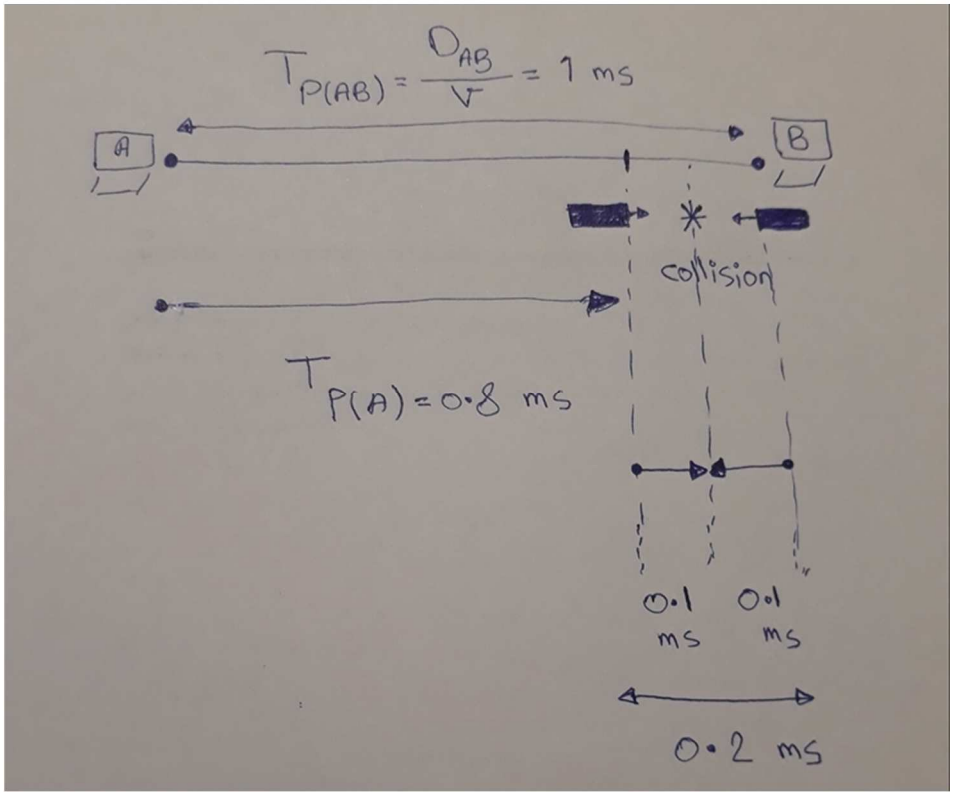
حالت سوم

فضای جلوی ایستگاه C توسط فریمی که از مدت‌ها قبل توسط ایستگاه A ارسال شده است اشغال است (فریم ایستگاه A در حال عبور از جلوی ایستگاه C می‌باشد) حال اگر ایستگاه C

بدون بررسی کانال انتقال (عدم شنود کانال) اقدام به ارسال فریم کند، تصادم رخ می‌دهد.



همان‌طور که از متن سؤال مشخص است ایستگاه‌های A و B توسط یک کانال با هم ارتباط دارند(در لایه‌ی پیوند داده گره‌ها توسط یک تک یال به‌هم متصل می‌شوند). برای بدست آوردن زمان تصادم باید بررسی کرد که در زمان ارسال ایستگاه B، فریم داده ارسالی ایستگاه A چه مسافتی را طی کرده است و یا به عبارتی دیگر با ایستگاه B چه فاصله‌ای دارد. در پروتکل CSMA/CD، از توپولوژی باس برای شبکه استفاده می‌شود و برطبق توپولوژی باس، هر کامپیوتر داده‌های ارسالی خود را در دو جهت کانال یعنی به سمت ایستگاه بعدی و هم به سمت ایستگاه قبلی ارسال می‌کند، در صورتیکه دو فریم داده در خلاف جهت یکدیگر و با یک سرعت حرکت کنند، بعد از گذراندن بخشی از مسیر به یکدیگر می‌رسند. بنابراین می‌توان اینگونه بیان کرد که بعد از گذراندن بخشی از مسیر توسط فریم‌های داده برخورد رخ می‌دهد، مطابق فرض سوال ایستگاه A در لحظه‌ی صفر فریم خود را ارسال می‌کند یعنی $T_{Send(A)} = ۰\ ms$ و ایستگاه B در لحظه‌ی ۰/۸ میلی‌ثانیه فریم خود را ارسال می‌کند یعنی $T_{Send(B)} = ۰/۸\ ms$، همچنین مطابق فرض سوال زمان انتشار(Propagation) بین نود A و نود B یک میلی‌ثانیه است، یعنی $T_{P(AB)} = \frac{D_{AB}}{V} = ۱ms$، پس در لحظه‌ی ۰/۸ میلی‌ثانیه، یعنی لحظه‌ی ارسال فریم ایستگاه B، فریم ایستگاه A مقدار ۰/۸ میلی‌ثانیه از زمان کل یک میلی‌ثانیه مابین نود A و B را سپری کرده‌است، بنابراین زمان باقی‌مانده مابین نود A و B برابر مقدار ۰/۲ میلی‌ثانیه است، پس می‌بایست زمان باقی‌مانده مابین نود A و B که برابر مقدار ۰/۲ میلی‌ثانیه است، به دو تقسیم شود، شکل زیر گویای مطلب است:



فریم داده ارسالی توسط ایستگاه A تا زمانی که ایستگاه B ارسال را آغاز کند، یعنی زمان ۰/۸ میلی‌ثانیه، همان زمان ۰/۸ میلی‌ثانیه را سپری می‌کند.

در زمان ۰/۸ میلی‌ثانیه ایستگاه B نیز اقدام به ارسال می‌کند و فریم‌های ارسالی A و B به سمت یکدیگر حرکت می‌کنند، فاصله زمانی مابین فریم ارسالی A با فریم ارسالی B در زمان ۰/۸ میلی‌ثانیه (زمان تولید فریم داده B) برابر با مقدار ۰/۲ میلی‌ثانیه می‌باشد، بنابراین:

$$T_{collision(AB)} = ۰/۸\,ms + \frac{۰/۲\,ms}{۲} = ۰/۸\ ms + ۰/۱\ ms = ۰/۹\ ms$$

زمان **تشخیص** تصادم در CSMA/CD برای ایستگاه A برابر با $۲\times T_{P(A)}$ می‌باشد.

$$T_{collision(A)} = ۲\times\frac{D_{A(collision)}}{V} = ۲\times T_{P(A)} = ۲\times۰/۹ = ۱/۸\ ms = ۱/۸\times۱۰^{-۳}\ s$$

تعداد بیت‌های ارسال شده تا زمانی که نتیجه‌ی وقوع برخورد به دست ایستگاه A برسد، یعنی تعداد بیتی که در مدت $۲\times T_{P(A)}$ ارسال می‌شود از رابطه زیر بدست می‌آید:

$$L_{min(A)} = T_{collision(A)}\times R = ۱/۸\times۱۰^{-۳}\ s\times۴\times۱۰^{۶}\ bps = ۷۲۰۰\,bit = \frac{۷۲۰۰\,bit}{۸} = ۹۰۰\ Byte$$

به بیان دیگر داریم:

زمان نرخ انتقال

$$\left.\begin{matrix} 1s & R = 4\times10^{6}\ bit \\ T_{collision(A)} = 1/8\times10^{-3}\ s & L_{min(A)} \end{matrix}\right\} \Rightarrow$$

$$L_{min(A)} = T_{collision(A)} \times R = 1/8\times10^{-3}\ s \times 4\times10^{6}\ bps = 7200\ bit = \frac{7200\ bit}{8} = 900\ Byte$$

در یک بیان دیگر داریم:
مطابق رابطه‌ی شرط کشف تصادم در پروتکل CSMA/CD برای ایستگاه A داریم:

$$T_{F(A)} \geq 2\times T_{P(A)}$$

$$\frac{L_{min(A)}}{R} \geq 2\times \frac{D_{A(collision)}}{V}$$

$$\frac{L_{min(A)}}{R} \geq 2\times T_{P(A)}$$

بنابراین مطابق این رابطه، می‌توان حداقل اندازه‌ی فریم $L_{min(A)}$ را برای برقرای شرط کشف تصادم محاسبه نمود:

$$\frac{L_{min(A)}}{R} \geq 2\times T_{P(A)} \rightarrow\ L_{min(A)} = 2\times T_{P(A)} \times R =$$

$$\rightarrow 2\times 0/9\times10^{-3}\ s \times 4\times10^{6}\ bps = 7200\ bit = \frac{7200\ bit}{8} = 900\ Byte$$

همچنین زمان **تشخیص** تصادم در CSMA/CD برای ایستگاه B برابر با $2\times T_{P(B)}$ می‌باشد.

$$T_{collision(B)} = 2\times \frac{D_{B(collision)}}{V} = 2\times T_{P(B)} = 2\times 0/1 = 0/2\ ms = 0/2\times10^{-3}\ s$$

تعداد بیت‌های ارسال شده تا زمانی که نتیجه‌ی وقوع برخورد به دست ایستگاه B برسد یعنی تعداد بیتی که در مدت $2\times T_{P(B)}$ ارسال می‌شود از رابطه‌ی زیر بدست می‌آید:

$$L_{min(B)} = T_{collision(B)} \times R = 0/2\times10^{-3}\ s \times 4\times10^{6}\ bps = 800\ bit = \frac{800\ bit}{8} = 100\ Byte$$

به بیان دیگر داریم:

زمان نرخ انتقال

$$\left.\begin{matrix} 1s & R = 4\times10^{6}\ bit \\ T_{collision(B)} = 0/2\times10^{-3}\ s & L_{min(B)} \end{matrix}\right\} \Rightarrow$$

$$L_{min(B)} = T_{collision(B)} \times R = 0/2\times10^{-3}\ s \times 4\times10^{6}\ bps = 800\ bit = \frac{800\ bit}{8} = 100\ Byte$$

در یک بیان دیگر داریم:
مطابق رابطه‌ی شرط کشف تصادم در پروتکل CSMA/CD برای ایستگاه B داریم:

$$T_{F(B)} \geq ۲\times T_{P(B)}$$

$$\frac{L_{min(B)}}{R} \geq ۲\times \frac{D_{B(collision)}}{V}$$

$$\frac{L_{min(B)}}{R} \geq ۲\times T_{P(B)}$$

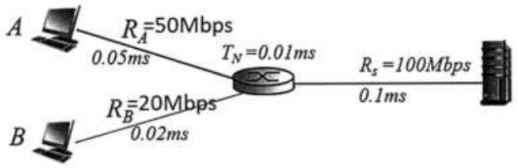
بنابراین مطابق این رابطه، می‌توان حداقل اندازه‌ی فریم $L_{min(B)}$ را برای برقرای شرط کشف تصادم محاسبه نمود:

$$\frac{L_{min(B)}}{R} \geq ۲\times T_{P(B)} \rightarrow L_{min(B)} = ۲\times T_{P(B)} \times R =$$

$$\rightarrow ۲\times ۰/۱\times ۱۰^{-۳}\ s\times ۴\times ۱۰^{۶}\ bps = ۸۰۰bit = \frac{۸۰۰bit}{۸} = ۱۰۰Byte$$

تست‌های فصل پنجم

۷۸- در شبکه زیر، سرور ۱۰۰ بسته به کامپیوتر A و ۱۰۰ بستهٔ دیگر به کامپیوتر B ارسال می‌کند. سرور بسته‌ها را یک در میان برای کامپیوتر A و سپس برای کامپیوتر B ارسال می‌کند. به‌عبارت دیگر، ابتدا یک بسته به کامپیوتر A ارسال شده سپس یک بسته به کامپیوتر B ارسال می‌شود و کار تا ارسال ۱۰۰ بسته برای A و ۱۰۰ بسته برای B ادامه می‌یابد. مسیریاب برای هر بسته زمان $TN = ۰/۰۱ms$ را صرف مسیریابی و سویچینگ می‌کند. اندازه هر بسته ۱۰۰۰ بایت است. آخرین بستهٔ ارسالی برای کامپیوتر B در صف مسیریاب چند میلی‌ثانیه معطل می‌ماند؟ (مقادیری که زیر هر لینک نوشته شده است زمان انتشار **(propagation)** بر حسب میلی‌ثانیه است.)



۱) ۲۱/۱۳msec
۲) ۲۳/۷۶msec
۳) ۲۴/۱۳msec
۴) ۲۵/۱msec

۷۸- گزینه (۲) صحیح است.

توجه: در شبکه‌های کامپیوتری چهار نوع تأخیر داریم:

تأخیر انتقال (T_F)، تأخیر انتشار (T_{Prop})، تأخیر صف (T_{queue})، تأخیر پردازش $(T_{process})$.

توجه: تأخیر صف‌بندی داخل گره‌ها، یک تأخیر متغیر است که به حجم ترافیک لحظه عبـور از آن گره بستگی دارد. به عبارت دیگر تاخیر صف در طول زمان نوسان دارد. پس تأخیری که از ابتدا به انتها ایجاد می‌شود، متغیر است و از قبل قابل پیش‌بینی نیست.

مثال: مثلاً دسترسی به سیستم آموزشی (پرتال)

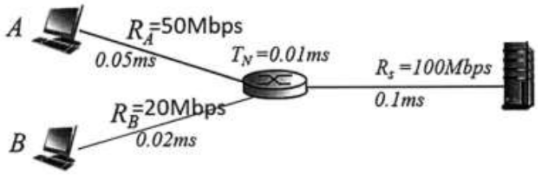
۱- دیدن پرتال از داخل دانشگاه از طریق شبکه محلی (تأخیر در حد ۱ ms)

۲- دیدن پرتال از خانه از طریق اینترنت (هنوز در شبکه داخل کشور) (تأخیر در حد ۱۰ms)

۳- دیدن پرتال از اروپا (تأخیر در حد ۱۰۰ms)

در صورت سوال گفته شده است که

در شبکه زیر، سرور ۱۰۰ بسته به کامپیوتر A و ۱۰۰ بستهٔ دیگر به کامپیوتر B ارسال می‌کند. سرور بسـته‌هـا را یک در میان برای کامپیوتر A و سپس برای کامپیوتر B ارسال می‌کند. به‌عبارت دیگر، ابتدا یک بسته به کامپیوتر A ارسال شده سپس یک بسته به کامپیوتر B ارسال می‌شود و کار تا ارسال ۱۰۰ بسته برای A و ۱۰۰ بسته برای B ادامه می‌یابد. مسیریاب برای هر بسته زمان $TN = 0/01ms$ را صرف مسیریابی و سویچینگ می‌کند. اندازه هر بسـته ۱۰۰۰ بایت است. آخرین بستهٔ ارسالی برای کامپیوتر B در صف مسـیریاب چنـد میلـی‌ثانیـه معطـل مـی‌مانـد؟ (مقادیری که زیر هر لینک نوشته شده است زمان انتشار (propagation) بر حسب میلی‌ثانیه است.)



به طور کلی حداقل زمان لازم برای انتقال بسته‌ها مابین دو گره انتهایی از رابطه زیر محاسبه می‌گرد:

$$T_{Total\ Delay} = [T_{F1}] + T_{Prop1} + [T_{process1} + T_{F2}] + T_{Prop2} + T_{queue}$$

T_F از رابطه زیر بدست می‌آید:

$$T_F = \frac{L}{R}$$

T_F، زمان انتقال بسته به داخل کانال انتقال است.

که L برابر اندازه بسته و R برابر نرخ انتقال می‌باشد.

T_{Prop} از رابطه زیر بدست می‌آید:

$$T_{Prop} = \frac{D}{V}$$

T_{Prop}، زمان تأخیر انتشار است.

که D برابر طول کانال و V برابر سرعت انتشار می‌باشد.

$T_{Process}$ از رابطه زیر بدست می‌آید:

$$T_{Proces} = \frac{b}{R}$$

$T_{Process}$، زمان پردازش موجود در مسیریاب (گره میانی) مربوط به کنترل خطای فریم، احیانا قطعه قطعه شدن بسته و مسیریابی بسته است.

که b برابر تعداد بیت لازم برای پردازش و R برابر نرخ انتقال می‌باشد.

T_{queue} از رابطه زیر بدست می‌آید:

$$T_{queue} = (N-1)\times\left(\frac{L}{\min(R_1, R_2, R_3)}\right)$$

T_{queue}، زمان تأخیر صف است.

که L برابر اندازه بسته، R برابر نرخ انتقال و N برابر تعداد بسته‌ها می‌باشد.

توجه: صف در جایی ایجاد می‌شود که پایین‌ترین نرخ انتقال را دارد یعنی $\min(R_1, R_2, R_3)$ که در این حالت گلوگاه (bottleneck) در آن محل ایجاد شده است.

توجه: مطابق فلش موجود در نمودار صورت سوال حرکت بسته‌ها از سمت راست به چپ یعنی از سمت سرور به سمت کلاینت است.

همچنین داده‌های مسئله به صورت زیر است:

$$L = 1000 \text{ Byte}$$
$$T_{Prop1} = 0/1 \text{ ms}, \; T_{Prop2} = 0/02 \text{ ms}$$
$$T_{Process1} = 0/01$$
$$R_1 = 100 \text{Mbps}, \; R_2 = 20 \text{Mbps}$$

همانطور که گفتیم به طور کلی حداقل زمان لازم برای انتقال بسته‌ها مابین دو گره انتهایی از رابطه زیر محاسبه می‌گرد: (البته در پورت B از مسیریاب که ۱۰۰ بسته قرار می‌گیرند.)

$$T_{Total\ Delay} = [T_{F1}] + T_{Prop1} + \left[T_{process1} + T_{F2}\right] + T_{Prop2} + T_{queue}$$

که پس از جایگذاری اولیه رابطه زیر را خواهیم داشت:

$$T_{Total\ Delay} = \left[\frac{L}{R_1}\right] + 0/1 + \left[0/01 + \frac{L}{R_2}\right] + 0/02 + (N-1)\times\left(\frac{L}{\min(R_1, R_2)}\right)$$

پس از جایگذاری نهایی رابطه زیر را خواهیم داشت:

$$T_{Total\ Delay} = \left[\frac{1000\times8}{100\times10^6}\times10^3\right] + 0/1 + \left[0/01 + \frac{1000\times8}{20\times10^6}\times10^3\right] + 0/02 + (99)\times\left(\frac{1000\times8}{20\times10^6}\times10^3\right)$$

در نهایت داریم:

$$T_{\text{Total Delay}} = [۰/۰۸]+۰/۱+[۰/۰۱+۰/۴]+۰/۰۲+(۹۹)\times(۰/۴)$$

$$T_{\text{Total Delay}} = [۰/۰۸]+۰/۱+[۰/۰۱+۰/۴]+۰/۰۲+۳۹/۶=۰/۶۱+۳۹/۶=۴۰/۲۱\ \text{ms}$$

دقت کنید که مقدار $۴۰/۲۱\ \text{ms}$ از لحظه‌ی صفر محاسبه شده است، اما شروع ارسال بسته‌های B بعد از انتقال بسته $A_۱$ آغاز شده است، بنابراین حداقل زمان لازم برای انتقال کل بسته‌های $B_۱$ تا $B_{۱۰۰}$ مابین دو گره انتهایی به صورت زیر خواهد بود:

$$T_{\text{Total Delay}} = \left[\frac{۱۰۰۰\times۸}{۱۰۰\times۱۰^{۶}}\times۱۰^{۳}\right]+۴۰/۲۱=۰/۰۸+۴۰/۲۱=۴۰/۲۹\ \text{ms}$$

اما در صورت سوال حداقل زمان لازم برای انتقال کل بسته‌ها مابین دو گره انتهایی خواسته نشده است، بلکه پرسیده شده است که بسته $B_{۱۰۰}$ در مسیریاب در صف خروجی پورت B، چند میلی ثانیه انتظار می‌کشد تا لحظه‌ی خروج آن از مسیریاب فرا برسد.

لحظه‌ی قرار گرفتن بسته $B_۱$ در صف خروجی پورت B به صورت زیر محاسبه می‌شود:

$$T_{\text{Total Delay(port B(۱))}} = \left[۱\times\left(۲\times T_{F۱}\right)\right]+T_{Prop۱}+T_{process۱}=\left[۱\times\left(۲\times۰/۰۸\right)\right]+۰/۱+۰/۰۱=۰/۲۷\ \text{ms}$$

لحظه‌ی قرار گرفتن بسته $B_۲$ در صف خروجی پورت B به صورت زیر محاسبه می‌شود:

$$T_{\text{Total Delay(port B(۲))}} = \left[۲\times\left(۲\times T_{F۱}\right)\right]+T_{Prop۱}+T_{process۱}=\left[۲\times\left(۲\times۰/۰۸\right)\right]+۰/۱+۰/۰۱=۰/۴۳\ \text{ms}$$

لحظه‌ی قرار گرفتن بسته $B_{۱۰۰}$ در صف خروجی پورت B به صورت زیر محاسبه می‌شود:

$$T_{\text{Total Delay(port B(۱۰۰))}} = \left[۱۰۰\times\left(۲\times T_{F۱}\right)\right]+T_{Prop۱}+T_{process۱}=\left[۱۰۰\times\left(۲\times۰/۰۸\right)\right]+۰/۱+۰/۰۱=۱۶/۱۱\ \text{ms}$$

وقتی بسته $B_{۱۰۰}$ در لحظه‌ی $۱۶/۱۱\ \text{ms}$ تازه در صف خروجی پورت B قرار می‌گیرد، در این لحظه تعدادی از بسته‌های جلویی از مسیریاب خارج شده‌اند، پس در ادامه باید محاسبه کنیم که در لحظه‌ی $۱۶/۱۱\ \text{ms}$ چند بسته از مسیریاب خارج شده‌اند و دیگر در صف پورت B و جلوی بسته $B_{۱۰۰}$ نیستند، دقت کنید که بسته‌های جمع‌شده در صف خروجی پورت B در هر $\left[\frac{۱۰۰۰\times۸}{۲۰\times۱۰^{۶}}\times۱۰^{۳}\right]$ یعنی $[۰/۴\text{ms}]$ از صف خروجی پورت B خارج می‌شوند، بنابراین حاصل تفاضل لحظه‌ی قرارگرفتن بسته $B_{۱۰۰}$ در صف خروجی پورت B یعنی $[۱۶/۱۱\text{ms}]$ و لحظه‌ی شکل‌گیری صف خروجی پورت B یعنی لحظه‌ی قرارگرفتن بسته $B_۱$ در صف خروجی پورت B یعنی $[۰/۲۷\text{ms}]$ تقسیم بر $[۰/۴\text{ms}]$ می‌شود، تعداد بسته‌های خارج شده از صف خروجی پورت B در لحظه‌ی $۱۶/۱۱\ \text{ms}$، به صورت زیر:

$$\text{Cardinality}_{\text{output}} = \left\lfloor\frac{۱۶/۱۱-۰/۲۷}{۰/۴}\right\rfloor=\left\lfloor\frac{۱۵/۸۴}{۰/۴}\right\rfloor=\lfloor۳۹/۶\rfloor=۳۹$$

دقت کنید که از عدد $۳۹/۶$ مقدار $۰/۶$ از زمان گذشته است و مقدار $۰/۴$ هنوز مانده است. پس در لحظه‌ی $[۱۶/۱۱]$ دقیقا ۳۹ بسته به طور **کامل** از صف خروجی پورت B خارج شده‌اند و از بسته ۴۰ نیز مقدار $۰/۶$ آن از صف خروجی پورت B خارج شده‌است ولی هنوز مقدار $۰/۴$ آن باقی‌مانده است. پس آنچه هنوز جلوی بسته $B_{۱۰۰}$ در صف خروجی پورت B قرار دارد و باقی مانده است، **بخشی** از بسته ۴۰ است و بسته‌های $B_{۴۱}$ تا $B_{۹۹}$ که تعداد آن می‌شود ۵۹ بسته که

همانطور که گفتیم بسته‌های موجود در صف خروجی پورت B در هر $\left[\frac{۱۰۰۰\times۸}{۲۰\times۱۰^{۶}}\times۱۰^{۳}\right]$ یعنی $[۰/۴]$ از صف خروجی پورت B خارج می‌شوند. بنابراین میزان انتظار بسته $B_{۱۰۰}$ در صف خروجی پورت B، تا لحظه‌ی خروج آن از مسیریاب فرا برسد از لحظه‌ی $[۱۶/۱۱]$ به بعد یعنی دقیقا لحظه‌ی قرار گرفتن بسته $B_{۱۰۰}$ در صف پورت B، به صورت زیر محاسبه می‌شود:

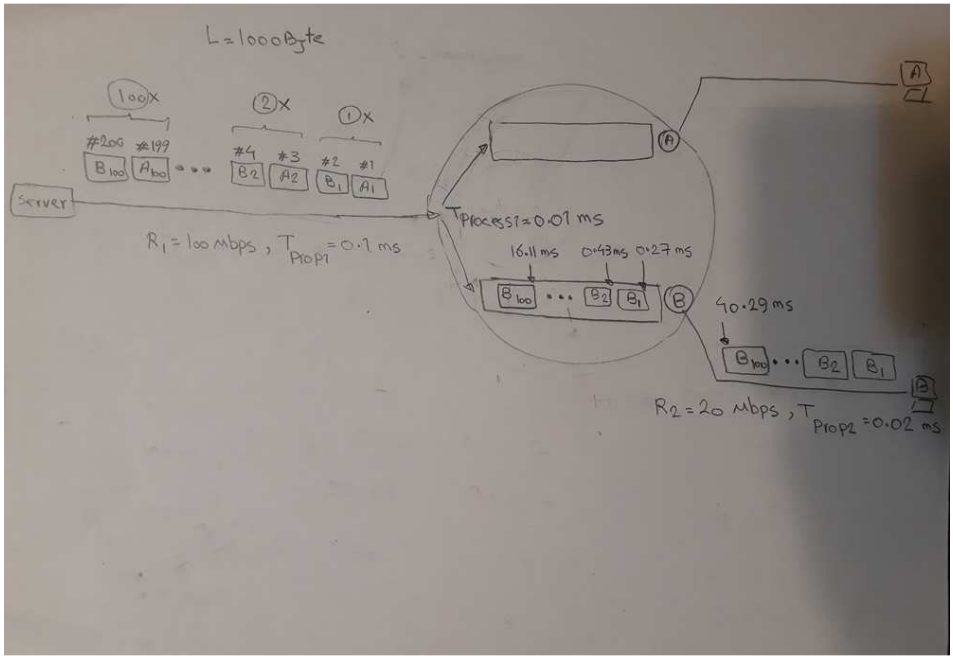
$$T_{queue(wait(B_{۱۰۰}))}=\left[T_{queue(wait(B_{۴۰}))}\right]+\left[T_{queue(wait(B_{۴۱}..B_{۹۹}))}\right]=$$

$$T_{queue(wait(B_{۱۰۰}))}=\left[Cardinality_{B_{۴۰}}\times\frac{L}{R_{۲}}\right]+\left[Cardinality_{B_{۴۱}..B_{۹۹}}\times\frac{L}{R_{۲}}\right]=$$

$$T_{queue(wait(B_{۱۰۰}))}=\left[۰/۴\times\frac{۱۰۰۰\times۸}{۲۰\times۱۰^{۶}}\times۱۰^{۳}\right]+\left[(۹۹-۴۱+۱)\times\frac{۱۰۰۰\times۸}{۲۰\times۱۰^{۶}}\times۱۰^{۳}\right]=$$

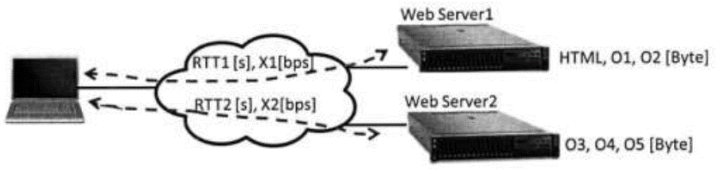
$$\rightarrow[۰/۴\times۰/۴]+[۵۹\times۰/۴]=[۰/۱۶]+[۲۳/۶]=۲۳/۷۶\ ms$$

شکل زیر گویای مطلب است:



تست‌های فصل هفتم

۷۹- یک صفحهٔ وب شامل یک فایل HTML و ۵ ابجکت است. فایل HTML = ۵۰۰۰Byte و ابجکت‌های O۱ = ۵۰۰۰Byte و O۲ = ۷۰۰۰Byte روی وب سرور۱ و آبجکت‌های $O_۳$ = ۱۰۰۰Byte ، O۴ = ۳۰۰۰Byte و O۵ = ۲۰۰۰Byte روی وب سرور۲ قراردارند. کاربری مایل است این صفحهٔ وب را ببیند. زمان رفت و برگشت بین کامپیوترِکاربر و سرور۱ به اندازه RTT۱ = ۰/۰۱s است. زمان رفت و برگشت بین کامپیوترِکاربر و سرور۲ به اندازه RTT۲ = ۰/۰۰۶s است. متوسط گذردهی ارتباط بین کامپیوترِکاربر و وب‌سرور۱ برابر با X۱ = ۸۰۰۰۰ بیت برثانیه است. گذردهی ارتباط بین کامپیوترِکاربر و وب سرور۲ برابر با X۲ = ۶۰۰۰۰ بیت برثانیه است.
چنانچه http1.1 در کامپیوترِکاربر و دو وب سرور فعال باشد، از لحظه‌ای که کاربر http GET را برای دریافت صفحهٔ وب ارسال می‌کند تا زمانی که صفحهٔ وب را کاملا دریافت می‌کند چند میلی‌ثانیه زمان صرف می‌شود؟
(توجه داشته باشید که http1.1 به‌صورت persistent و pipeline کار می‌کند.)



۱) ۱/۷۳
۲) ۱/۹۵
۳) ۲/۳۷
۴) ۲/۴۱

پاسخ‌های فصل هفتم

۷۹- گزینه (۱) صحیح است.

پروتکل HTTP در لایه کاربرد

به برنامه کاربردی که روی اینترنت نوشته شده است، world wide web یا شبکه جهانی وب گفته می‌شود. زیرا documentهایی داریم که Linkها را به هم متصل می‌کند، پروتکلی که برای آن طراحی شده است، پروتکل HTTP (HyperText Transfer Protocol) نام دارد.

کاری که HTTP انجام می‌دهد این است که clientها، objectها را به web server، request می‌دهند و web server هم objectها را می‌آورد.

objectها می‌توانند یک فایل HTML با یک تصویر JPEG و ... باشند که توسط این پروتکل می‌توانند منتقل شوند.

هر object ای در محیط عملیاتی اینترنت با یک آدرس منحصر به فرد معرفی می‌شود که به آن URL گفته می‌شود. URL سرواژه عبارت Uniform Resource Locator می‌باشد.

مثال:

www.iust.ac.ir/index.htm
/home/logo.jpg
/home/Header.jpg

در صفحه اول دانشگاه ممکن است n تا object وجود داشته باشد.

پس اولین کاری که می‌کنیم تا یک صفحه web بیاید این است که یک request از سمت Client به Server بدهیم بدون این که چیزی مشخص کنیم. از آنجاییکه پروتکل http به دلیل دغدغه صحت داشتن با پروتکل TCP در لایه انتقال کار می‌کند، در ادامه ابتدا TCP درخواست Clint به سمت Server را معوق می‌کند تا یک TCP Connection مابین فرستنده و گیرنده برای درخواست و دریافت فایل پایه HTML ایجاد کند. این TCP Connection در سه گام یعنی (۱) فاز برقراری اتصال (3–way handshaking)، (۲) فاز تبادل داده و (۳) فاز رهاسازی اتصال انجام می‌گردد. که در ادامه به بررسی فاز برقراری اتصال (3–way handshaking) می‌پردازیم:

فاز برقراری اتصال (3–way handshaking)

برای ایجاد TCP Connection ، سه پیغام TCP رد و بدل می‌شود که به آن 3–way handshaking (دست‌تکاندهی سه طرفه) نیز گفته می‌شود. مراحل فاز برقراری اتصال به صورت زیر است:

۱) ابتدا Clinet، درخواست برقراری Connection را به Server می‌دهد.(SYN=1)

۲) Server یک ACK به Client ارسال می‌کند یعنی می‌پذیرد که Connection سمت Client به سمت Server باز شود. همچنین Server علاوه بر ACK یک درخواست ایجاد Connection از سمت Server به Client هم می‌فرستد.(ACK=1, SYN=1)

توجه: Server ACK و درخواست ایجاد Connection هر دو با هم از طرف Server در قالب یک پیام به سمت Client ارسال می‌گردد.

توجه: وقتی Client، ACK را از Server گرفت، Connection سمت Client به Server باز می‌شود، پس Client می‌تواند داده و درخواست بفرستد. Client این اختیار را دارد که همراه ACK، داده و درخواست هم بفرستد.

۳) Client یک ACK به Server ارسال می‌کند یعنی می‌پذیرد که Connection سمت Server به سمت Client باز شود. (ACK=1)

توجه: وقتی Server، ACK را از Client گرفت، Connection سمت Server به Client باز می‌شود، پس Server می‌تواند داده و درخواست بفرستد.

توجه: TCP، Connectionهایش دو طرفه است، یعنی هم از سمت Client به سمت Server یک Connection ایجاد می‌کند و هم از سمت Server به سمت Client یک Connection ایجاد می‌کند.

توجه: تا این سه پیغام رد و بدل نشوند. Connection بین Client و Server ایجاد نشده است، به این سه پیغام در TCP اصطلاحاً 3-way handshaking گفته می‌شود. به معنی دست‌تکان‌دهی سه طرفه، در واقع با این کار، دو گره دارند عمل خوشامدگویی انجام می‌دهند و سپس Connection به شکل دو طرفه برقرار می‌شود.

مثال: مثلاً شما وقتی دوستتان را ببینید برای باز کردن سر صحبت یک سری تعارفات اولیه انجام می‌دهید: سلام، ... ، دست دادن ... این‌ها که گفتیم برای فاز برقراری اتصال بود.

توجه: پس حداقل یک زمان رفت و برگشت طول می‌کشد تا Client بتواند یک request مربوط به درخواست و دریافت فایل پایه html را بدهد. البته اگر request اش را همراه ACK بدهد، که معمولا به این صورت است. به این زمان رفت و برگشت اصطلاحا RTT یا Round Trip Time گفته می‌شود.

توجه: این تأخیر RTT از موقعی که Client یک request به Server می‌دهد تا ACK آن را دریافت کند یعنی Connection برقرار شود، یا از موقعی که یک پیغام می‌دهد تا جواب آن را بگیرد، شامل تمام تأخیرهای شبکه است، تأخیر انتقال (T_F)، تأخیر انتشار (T_{Prop})، تأخیر صف (T_{queue})، تأخیر پردازش $(T_{process})$.

توجه: RFC ای که برای HTTP وجود دارد RFC۱۹۵۴ و RFC۲۶۱۶ است.

توجه: تمام پروتکل‌هایی که در شبکه‌ی اینترنت وجود دارند، دارای RFC هستند، برای مثال بـرای دیدن جزئیات آن ها باید RFCشان را بگیریم و مطالعه کنیم یا اگـر بخـواهیم آن‌هـا را پیاده‌سـازی کنیم باید RFC آنها را تهیه کنیم. RFC مانند کتاب قانون است، قوانینی دارد که می‌گوید:

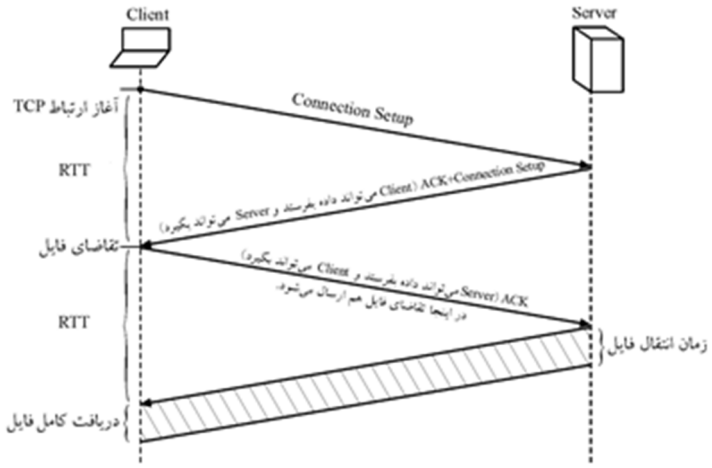
۱- اول این کار را انجام بده

۲- این پیغام را دریافت کردی، بعد این کار را انجام بده و ...

RFC یک Reference برای پیاده سازی بدون ابهام است.

توجه: شرح RFC ها در سایت IETF.ORG قرار دارد.

توجه: پس از آنکه فاز برقـراری اتصـال (3-way handshaking) انجـام شـد، یعنـی Connection سمت Client به Server باز شد. آنگاه نوبت به ارسال request به معنی درخواست و دریافت فایل پایه HTML از سمت Client به Server می‌رسد، این صفحه‌ی اصلی یعنی فایل پایـه HTML بـه فرمت HTML می‌آید، در فایل پایه HTML گفته شده است که در آن چنـد object وجـود دارد و بعد browser شـما objectهـا را بـه آن شـکلی کـه هسـت نشـان می‌دهـد. در ایـن حالـت نقشـه درخواست و دریافت objectها در فایل پایه HTML مشخص شده است. شکل زیر گویای مطلـب می‌باشد:



به طور کلی زمان دستیابی به یک صفحه وب به طور کامل از رابطه زیر محاسبه می‌گردد:

$$T_{Access\ (Website)} = T_{Translate\ (Domain\ to\ IP)} + T_{Destination} = T_{DNS\ LOOK\ UP} + T_{HTTP}$$

توجه: فرض کنید در مرورگر وب خود برای دریافت یک صفحه وب به طور کامل بـر روی یـک لینک کلیک می‌کنید و آدرس IP مربوط به این URL در میزبان محلی **ذخیره نشده** است، در نتیجه برای به دست آوردن آدرس IP به یک DNS LOOK UP نیاز اسـت. فـرض کنیـد بـرای دریافـت آدرس IP از طریق سرویس DNS، n سرور DNS ملاقات می‌شوند و تاخیر زمان رفت و برگشـت

معادل RTT1 تا RTTn باشد. بنابراین بدون در نظر گرفتن زمان مربوط بـه درخواسـت و دریافـت فایل پایه html و objectهای موجود در آن، رابطه زیر را خواهیم داشت:

$$T_{Access\ (Website)} = T_{Translate\ (Domain\ to\ IP)} + T_{Destination} = T_{DNS\ LOOK\ UP} + T_{HTTP} = \sum_{i=1}^{n} RTT_i + T_{HTTP}$$

حال در ادامه به نحوه‌ی محاسبه T_{HTTP} در شرایط مختلف می‌پردازیم:

توجه: از پروتکل HTTP به دو حالت می‌توان استفاده کرد:

(۱) non-persistent http : ناپایدار و (۲) persistent http : پایدار

Non-persistent http ناپایدار(غیرمصر یا غیرمداوم)

در حالت Non-persistent http یک TCP connection مابین فرستنده و گیرنده برای درخواست و دریافت فایل پایه HTML ایجاد می‌گردد و در انتها Connection بسته می‌شود. در ادامه نیـز بـرای درخواست و دریافت objectها به طور مستقل Connection باز و بسته می‌شود.

حالت Non-persistent http خود به سه روش ترتیبی، موازی نامحـدود و مـوازی محـدود وجـود دارد، که روابط آن به صورت زیر است:

روش ناپایدار ترتیبی:

$$\left[(\underbrace{RTT}_{\text{درخواست و برقراری ارتباط}} + \underbrace{RTT}_{\text{درخواست و دریافت فایل اصلی}} + \underbrace{T_{Base\ HTML}}_{\text{زمان انتقال فایل اصلی}})\right] + \left[\underbrace{n}_{\text{تعداد درخواست‌ها}} \times (\underbrace{RTT}_{\text{درخواست و برقراری ارتباط}} + \underbrace{RTT}_{\text{درخواست و دریافت objectها}}) + \sum_{i=1}^{n} \underbrace{T_{object}(i)}_{\text{زمان انتقال object}}\right]$$

روش ناپایدار موازی نامحدود:

$$\left[(\underbrace{RTT}_{\text{درخواست و برقراری ارتباط}} + \underbrace{RTT}_{\text{درخواست و دریافت فایل اصلی}} + \underbrace{T_{Base\ HTML}}_{\text{زمان انتقال فایل اصلی}})\right] + \left[\underbrace{n}_{\text{یکی}} \times (\underbrace{RTT}_{\text{درخواست و برقراری ارتباط}} + \underbrace{RTT}_{\text{درخواست موازی و دریافت موازی}}) + \sum_{i=1}^{n} \underbrace{T_{object}(i)}_{\text{زمان انتقال object}}\right]$$

روش ناپایدار موازی محدود:

$$\left[(RTT+RTT+T_{Base\ HTML})\right]+\left[n\times(RTT+RTT)+\sum_{i=1}^{n}T_{object}(i)\right]$$

(درخواست و برقراری ارتباط؛ زمان انتقال فایل اصلی؛ درخواست و دریافت فایل اصلی؛ درخواست و برقراری ارتباط؛ تعداد درخواست‌ها؛ درخواست موازی و دریافت موازی؛ زمان انتقال object)

persistent http پایدار(مصر یا مداوم)

در حالت persistent http یک TCP connection مابین فرستنده و گیرنده برای درخواست و دریافت فایل پایه HTML ایجاد می‌گردد و در انتها Connection باز می‌ماند. در ادامه نیز برای درخواست و دریافت object ها همان TCP connection اولیه مورد استفاده قرار می‌گیرد.

حالت persistent http خود به سه روش ترتیبی، موازی نامحدود و موازی محدود وجود دارد، که روابط آن به صورت زیر است:

روش پایدار ترتیبی:

$$\left[(RTT+RTT+T_{Base\ HTML})\right]+\left[n\times(\cancel{RTT}+RTT)+\sum_{i=1}^{n}T_{object}(i)\right]$$

(درخواست و برقراری ارتباط؛ زمان انتقال فایل اصلی؛ درخواست و دریافت فایل اصلی؛ درخواست و برقراری ارتباط؛ صفر؛ تعداد درخواست‌ها؛ درخواست و دریافت object؛ زمان انتقال object)

پایدار موازی نامحدود:

$$\left[(RTT+RTT+T_{Base\ HTML})\right]+\left[n\times(\cancel{RTT}+RTT)+\sum_{i=1}^{n}T_{object}(i)\right]$$

(درخواست و برقراری ارتباط؛ زمان انتقال فایل اصلی؛ درخواست و دریافت فایل اصلی؛ درخواست و برقراری ارتباط؛ صفر؛ یک؛ درخواست موازی و دریافت موازی object؛ زمان انتقال object)

روش پایدار موازی محدود:

$$\left[(RTT+RTT+T_{Base\ HTML})\right]+\left[n\times(\cancel{RTT}+RTT)+\sum_{i=1}^{n}T_{object}(i)\right]$$

درخواست و برقراری ارتباط — زمان انتقال فایل اصلی — درخواست و دریافت فایل اصلی — درخواست و برقراری ارتباط — تعداد درخواست‌ها — صفر — درخواست موازی و دریافت موازی object — زمان انتقال object

توجه: اگر برای مدتی روی Connection ، request ای نیاید، server آن را می‌بندد.

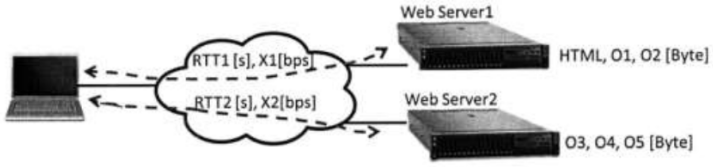
توجه: بستگی به برنامه کاربردی دارد Persistent یا Non persistent را انتخاب کند. پروتکل HTTP به هر دو اجازه می‌دهد.

در صورت سوال گفته شده است که

یک صفحهٔ وب شامل یک فایل HTML و ۵ آبجکت است. فایل HTML = ۵۰۰۰Byte و آبجکت‌های $O_1 = 5000\,Byte$ و $O_2 = 7000\,Byte$ روی وب سرور۱ و آبجکت‌های $O_3 = 1000\,Byte$, $O_4 = 3000\,Byte$ و $O_5 = 2000\,Byte$ روی وب سرور۲ قراردارند. کاربری مایل است این صفحهٔ وب را ببیند. زمان رفت و برگشت بین کامپیوترِکاربر و سرور۱ به اندازه $RTT_1 = 0.01s$ است. زمان رفت و برگشت بین کامپیوترِکاربر و سرور۲ به اندازه $RTT_2 = 0.006s$ است. متوسط گذردهی ارتباط بین کامپیوترِکاربر و وب‌سرور۱ برابر با $X_1 = 80000$ بیت بر ثانیه است. گذردهی ارتباط بین کامپیوترِکاربر و وب سرور۲ برابر با $X_2 = 60000$ بیت بر ثانیه است.

چنانچه http1.1 در کامپیوترِکاربر و دو وب سرور فعال باشد، از لحظه‌ای که کاربر http GET را برای دریافت صفحهٔ وب ارسال می‌کند تا زمانی که صفحهٔ وب را کاملا دریافت می‌کند چند میلی‌ثانیه زمان صرف می‌شود؟

(توجه داشته باشید که http1.1 به‌صورت persistent و pipeline کار می‌کند.)



داده‌های مسئله به صورت زیر است:

$L_{Base\ HTML} = 5000\ Byte$

$L_{Object1} = 5000\ Byte\ ,\ L_{Object2} = 7000\ Byte$

$L_{Object3} = 1000\ Byte\ ,\ L_{Object4} = 3000\ Byte\ ,\ L_{Object5} = 2000\ Byte$

$R_{TOTAL\ CHANNEL(X1)} = 80000\ bps,\ RTT_1 = 0.01$

$R_{TOTAL\ CHANNEL(X2)} = 60000\ bps,\ RTT_2 = 0.006$

$Cardinality(Object) = 5,\ T_{DNS\ LOOK\ UP} = 0$

توجه: در صورت سوال محدودیت توازی مطرح نشده است، پس توازی نامحدود را در نظر می‌گیریم.

رابطه روش پایدار(persistent) موازی(pipeline) نامحدود به صورت زیر است:

$$\left[(\overset{\uparrow}{RTT}+\underset{\downarrow}{RTT}+\overset{\uparrow}{T_{Base\ HTML}})\right]+\left[\underset{\downarrow}{n}\times(\overset{\uparrow}{\cancel{RTT}}+\underset{\downarrow}{RTT})+\sum_{i=1}^{n}\overset{\uparrow}{T_{object}}(i)\right]$$

درخواست و برقراری ارتباط — زمان انتقال فایل اصلی — درخواست و دریافت فایل اصلی — یک — صفر — درخواست موازی و دریافت موازی object — زمان انتقال object

اما رابطه روش ناپایدار موازی نامحدود با در نظر گرفتن دو ارتباط به webserver1 و webserver2 به صورت زیر است:

$$\left[(\overset{\uparrow}{RTT}+\underset{\downarrow}{RTT}+\overset{\uparrow}{T_{Base\ HTML}})\right]+\left[Max\left(\left[T_{webserver1(object1,object2)}\right],\left[T_{webserver2(object3,object4,object5)}\right]\right)\right]$$

درخواست و برقراری ارتباط — زمان انتقال فایل اصلی — درخواست و دریافت فایل اصلی

همچنین رابطه روش ناپایدار موازی نامحدود با در نظر گرفتن دو ارتباط به webserver1 و webserver2 به فرم دقیق‌تر به صورت زیر است:

$$\left[(\overset{\uparrow}{RTT_1}+\underset{\downarrow}{RTT_1}+\overset{\uparrow}{T_{Base\ HTML}})\right]+\rightarrow$$

درخواست و برقراری ارتباط — زمان انتقال فایل اصلی — درخواست و دریافت فایل اصلی

$$T_{webserver1(object1,object2)}=\left[\underset{\downarrow}{n}\times(\overset{\uparrow}{\cancel{RTT_1}}+\underset{\downarrow}{RTT_1})+\frac{\overset{\uparrow}{L_{Object1}}}{R_{TOTAL\ CHANNEL(X1)}}+\frac{\overset{\uparrow}{L_{Object2}}}{R_{TOTAL\ CHANNEL(X1)}}\right]$$

درخواست و برقراری ارتباط — یک — صفر — درخواست موازی و دریافت موازی object — زمان انتقال object1 — زمان انتقال object2

$$T_{webserver\,2(object3,object4,object5)} = \left[\underset{\downarrow}{n}\times(RTT_2 + RTT_2) + \frac{L_{Object3}}{R_{TOTAL\,CHANNEL(X2)}} + \frac{L_{Object4}}{R_{TOTAL\,CHANNEL(X2)}} + \frac{L_{Object5}}{R_{TOTAL\,CHANNEL(X2)}} + \right]$$

n: یک
RTT_2: درخواست موازی و دریافت موازی object
RTT_2: درخواست و برقراری ارتباط
$L_{Object3}$: زمان انتقال object3
$L_{Object4}$: زمان انتقال object4
$L_{Object5}$: زمان انتقال object5

$T_{Base\,HTML}$ از رابطه زیر بدست می‌آید:

$$T_{Base\,HTML} = \frac{L_{Base\,HTML}}{R_{TOTAL\,CHANNEL(X1)}}$$

$T_{Base\,HTML}$، زمان انتقال فایل پایه html به داخل کانال انتقال است.

که $L_{Base\,HTML}$ برابر اندازه فایل پایه html و $R_{TOTAL\,CHANNEL}$ برابر نرخ انتقال کل کانال می‌باشد.

T_{Object} از رابطه زیر بدست می‌آید:

$$T_{Object} = \frac{L_{Object}}{R_{TOTAL\,CHANNEL}}$$

T_{Object}، زمان انتقال Object به داخل کانال انتقال است.

که L_{Object} برابر اندازه Object و $R_{TOTAL\,CHANNEL}$ برابر نرخ انتقال کل کانال می‌باشد.

با توجه به شرایط ذکر شده در صورت سؤال، مطابق آنچه گفتیم ابتدا بایستی فایل پایه HTML را دریافت کرد و سپس ۵ فایل object دیگر را دریافت کرد و حالا با توجه به نوع ارتباط که **پایدار موازی نامحدود** است بایستی اینگونه عمل کنیم:

ابتدا یک RTT صرف درخواست و برقراری ارتباط با webserver1 می‌شود، سپس یک RTT دیگر صرف درخواست و دریافت فایل پایه HTML می‌شود که پس از دریافت فایل پایه HTML مشخص می‌شود که فایل‌های object1 و object2 در webserver1 قرار دارند و همچنین فایل‌های object3، object4 و object5 در webserver2 هستند و پس از آن به دلیل پایدار بودن ارتباط، ارتباط میان client و webserver1 قطع نمی‌شود و پایدار می‌ماند، به صورت زیر:

$$\left[(RTT_1 + RTT_1 + T_{Base\,HTML})\right] = \left[(0.01 + 0.01 + \frac{5000\times 8}{80000})\right] = \left[(0.01 + 0.01 + 0.5)\right] = 0.52$$

بعد از دریافت فایل پایه HTML ، به دلیل وجـود **کانکشن پایدار موازی نامحدود میان client و webserver1** از قبل، نیاز به درخواست و برقراری ارتبـاط مجـدد میـان client و webserver1 در یک RTT نمی‌باشد. اما یک RTT دیگر صـرف درخواسـت مـوازی و دریافـت مـوازی فایل‌هـای object1 و object2 می‌شود. بنابراین مقدار n در رابطه $T_{webserver1}$ باعنوان تعداد درخواسـت‌ها برابـر ۱ خواهد بود، زیرا فقط یک کانکشن برای درخواست موازی و دریافت موازی فایل‌هـای object1 و object2 ایجاد کردیم.

که پس از جایگذاری اولیه رابطه زیر را خواهیم داشت:

$$T_{webserver1(object1,object2)} = \left[\underset{\text{یک}}{n} \times (\underset{\text{صفر}}{\cancel{RTT_1}} + \underset{\substack{\text{درخواست موازی و دریافت موازی object}}}{RTT_1}) + \frac{L_{Object1}}{R_{TOTAL\ CHANNEL(X1)}} + \frac{L_{Object2}}{R_{TOTAL\ CHANNEL(X1)}} \right]$$

درخواست و برقراری ارتباط — زمان انتقال object1 — زمان انتقال object2

$$T_{webserver1(object1,object2)} = \left[1 \times (0 + 0.01) + \frac{5000 \times 8}{80000} + \frac{7000 \times 8}{80000} \right]$$

پس از جایگذاری نهایی رابطه زیر را خواهیم داشت:

$$T_{webserver1(object1,object2)} = \left[1 \times (0 + 0.01) + 0.5 + 0.7 \right] = 1.21s$$

بعد از دریافت فایل پایه HTML از webserver1 دیگر نیاز به دریافت مجـدد آن نـداریم، زیـرا از فایل‌های object1، object2، object3، object4 و object5 آگاه و باخبرشدیم. اما می‌بایسـت میـان client و webserver2 یک ارتباط پایدار موازی نامحدود برقرار شود.

بنابراین ابتدا یک RTT صرف درخواست و برقراری ارتباط بـا webserver2 می‌شـود، سـپس یـک RTT دیگر صرف درخواست مـوازی و دریافـت مـوازی فایل‌هـای object3، object4 و object5 می‌شود. بنابراین مقدار n در رابطه $T_{webserver2}$ باعنوان تعداد درخواست‌ها برابر ۱ خواهد بـود، زیـرا فقط یـک کانکشـن بـرای درخواسـت مـوازی و دریافـت مـوازی فایل‌هـای object3، object4 و object5 ایجاد کردیم.

که پس از جایگذاری اولیه رابطه زیر را خواهیم داشت:

$$T_{webserver\,2(object3,object4,object5)}=\left[\underset{\text{یک}}{n}\times(RTT_2+\underset{\text{درخواست موازی و دریافت موازی object}}{RTT_2})+\underset{\text{زمان انتقال object3}}{\frac{L_{Object3}}{R_{TOTAL\,CHANNEL(X2)}}}+\underset{\text{زمان انتقال object4}}{\frac{L_{Object4}}{R_{TOTAL\,CHANNEL(X2)}}}+\underset{\text{زمان انتقال object5}}{\frac{L_{Object5}}{R_{TOTAL\,CHANNEL(X2)}}}+\right]$$

$$T_{webserver\,2(object3,object4,object5)}=\left[1\times(0.006+0.006)+\frac{1000\times8}{60000}+\frac{3000\times8}{60000}+\frac{2000\times8}{60000}\right]$$

پس از جایگذاری نهایی رابطه زیر را خواهیم داشت:

$$T_{webserver\,2(object3,object4,object5)}=\left[1\times(0.006+0.006)+0.8\right]=0.812\,s$$

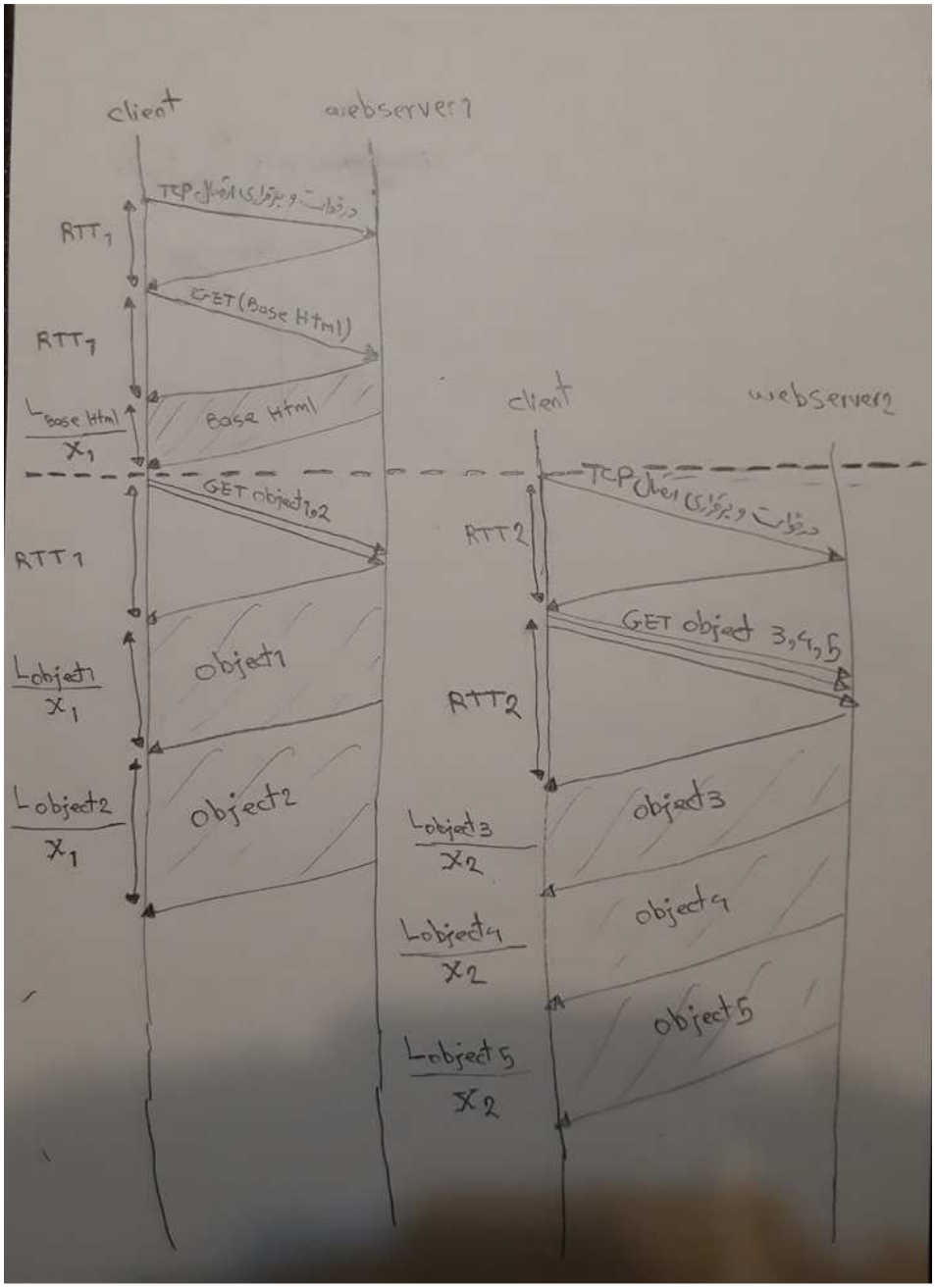
همانطور که گفتیم رابطه روش ناپایدار موازی نامحدود با در نظر گرفتن دو ارتباط به webserver1 و webserver2 به صورت زیر است:

$$\left[(RTT_1+RTT_1+T_{Base\,HTML})\right]+\left[\,Max\left(\left[T_{webserver1(object1,object2)}\right],\left[T_{webserver\,2(object3,object4,object5)}\right]\right)\right]$$

$$\left[(0.52)\right]+\left[\,Max\left([1.21\,s],[0.812\,s]\right)\right]=\left[(0.52\,s)\right]+\left[1.21\,s\right]=1.73s$$

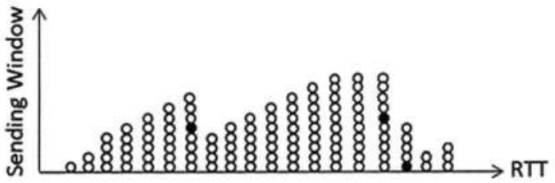
توجه: عبارت چند **میلی‌ثانیه** زمان صرف می‌شود در صورت سوال باید به چند **ثانیه** زمان صرف می‌شود اصلاح گردد.

شکل زیر گویای مطلب است:



تست‌های فصل ششم

۸۰- در یک ارتباطِ tcp، فایلی از کامپیوتر۱ به کامپیوتر۲ ارسال می‌شود. شکل زیر پنجره‌های ارسال را در حوزه زمان نشان می‌دهد. در این شکل هر دایره یک بسته را نشان می‌دهد. دایره‌های سیاه معرف بسته هایی‌اند که به کامپیوتر۲ نرسیده‌اند. اگر مکانیزم Go-Back-n فعال باشد، چند بسته بیش از یک‌بار به کامپیوتر۲ می‌رسد؟



۱) ۷
۲) ۹
۳) ۱۱
۴) ۱۴

پاسخ‌های فصل ششم

۸۰- گزینه (۱) صحیح است.

روش بازگشت به N سگمنت قبلی (Go-Back-N)

در این روش فرستنده بسته‌های(سگمنت‌های) داده را پشت سرهم و به طور پیوسته و به ترتیب شماره ترتیب (Sequence Number) به سمت گیرنده ارسال می‌کند. گیرنده نیز با دریافت هر کدام از سگمنت‌های داده، یک سگمنت ACK با شماره تصدیق (Acknowledge Number) مربوطه برای فرستنده ارسال می‌کند. فیلد شماره تصدیق (Acknowledge Number) درون سگمنت ACK شماره سگمنت بعدی را که فرستنده باید ارسال کند را به فرستنده می‌گوید. سگمنت ACK ارسال شده از گیرنده به سمت فرستنده برای فرستنده به این معنی است که سگمنت داده قبلی را درست و کامل دریافت کرده‌ام و منتظر دریافت سگمنت بعدی هستم. در صورتی که فرستنده سگمنت ACK مربوط به سگمنت داده ارسالی خود را دریافت نکند در روش Go-Back-N به عقب بر می‌گردد و سگمنت داده‌ای که سگمنت ACK آن توسط فرستنده دریافت نشده است و تمام سگمنت‌های بعد از آن را یک بار دیگر ارسال می‌نماید.

به عبارت دیگر در روش Go-Back-N، اگر سگمنتی در مسیر loss شود، هیچ یک از سگمنت‌های با شماره ترتیب بعد از سگمنت loss شده، در مقصد پذیرش نمی‌شوند و همگی نامعتبر تلقی می‌شوند، بنابراین فرستنده پس از کشف رویداد loss، خود سگمنت loss شده و همه سگمنت‌های با شماره ترتیب بعد از آنرا مجددا ارسال می‌کند.

توجه: هر دایره‌ی مطرح شده روی شکل صورت سوال، نشانه‌ی یک سگمنت است، دایره‌های سیاه معرف سگمنت‌هایی هستند که به کامپیوتر مقصد نرسیده‌اند و دایره‌های سفید معرف سگمنت‌هایی هستند که به کامپیوتر مقصد رسیده‌اند.

توجه: در ارتباط با پروتکل TCP در لایه‌ی انتقال، پروتکل TCP تابع و وظیفه‌ی کنترل ازدحام را دارد که الگوریتم این وظیفه به فرم‌هایی نظیر TCP TAHOE و TCP RENO وجود دارد، با توجه به شکل صورت سوال، الگوریتم کنترل ازدحام ارتباط TCP مطرح شده به صورت TCP RENO است.

دور هفتم فرستنده ۸ سگمنت داده ارسال می‌کند و از آنجا که سگمنت **۳۰** از دست می‌رود (loss) و به دست گیرنده نمی‌رسد، پس گیرنده (۱-۸) یعنی ۷ سگمنت ACK تولید می‌کند که به دست فرستنده می‌رسد.

و در **دور هشتم** فرستنده سگمنت‌های داده **۳۰**، **۳۱**، **۳۲** و **۳۳** را مجددا ارسال می‌کند، یعنی فرستنده ۴ سگمنت داده را مجددا ارسال می‌کند و گیرنده نیز ۴ سگمنت ACK تولید می‌کند که به دست فرستنده می‌رسد.

بنابراین مطابق روابط زیر داریم:

$\text{Segment}_{\text{Data}} = \text{Segment}_{\text{Data:۷}} + \text{Segment}_{\text{Data:۸}} = ۸ + ۴ = ۱۲$

$\text{Segment}_{\text{ACK}} = \text{Segment}_{\text{ACK:۷}} + \text{Segment}_{\text{ACK:۸}} = (۸ - ۱) + ۴ = ۷ + ۴ = ۱۱$

نتیجه اینکه مطابق فرض سوال، سگمنت‌های ۳۱، ۳۲ و ۳۳ در مجموع دور هفتم و هشتم یعنی ۳ سگمنت دوبار ارسال شده‌اند و دوبار هم به مقصد رسیده‌اند یعنی بیش از یک بار به کامپیوتر ۲ می‌رسند.

توجه: دقت کنید که سگمنت‌های loss شده، بیش از یکبار توسط فرستنده ارسال می‌شوند، اما بیش از یکبار توسط گیرنده دریافت نمی‌شوند.

دور شانزدهم فرستنده ۱۰ سگمنت داده ارسال می‌کند و از آنجا که سگمنت ۹۴ از دست می‌رود (loss) و به دست گیرنده نمی‌رسد، پس گیرنده (۱-۱۰) یعنی ۹ سگمنت ACK تولید می‌کند که به دست فرستنده می‌رسد.

و در **دور هفدهم** فرستنده سگمنت‌های داده ۹۴، **۹۵، ۹۶، ۹۷ و ۹۸** را مجددا ارسال می‌کند، یعنی فرستنده ۵ سگمنت داده را مجددا ارسال می‌کند اما بازهم سگمنت ۹۴ از دست می‌رود (loss) و به دست گیرنده نمی‌رسد، پس گیرنده (۱-۵) یعنی ۴ سگمنت ACK تولید می‌کند که به دست فرستنده می‌رسد.

و در **دور هجدهم** فرستنده سگمنت‌های داده **۹۴** و **۹۵** را مجددا ارسال می‌کند، یعنی فرستنده ۲ سگمنت داده را مجددا ارسال می‌کند و گیرنده نیز ۲ سگمنت ACK تولید می‌کند که به دست فرستنده می‌رسد.

و در **دور نوزدهم** فرستنده سگمنت‌های داده **۹۶** و **۹۷ و ۹۸** را مجددا ارسال می‌کند، یعنی فرستنده ۳ سگمنت داده را مجددا ارسال می‌کند و گیرنده نیز ۳ سگمنت ACK تولید می‌کند که به دست فرستنده می‌رسد.

بنابراین مطابق روابط زیر داریم:

$\text{Segment}_{\text{Data}} = \text{Segment}_{\text{Data:۱۶}} + \text{Segment}_{\text{Data:۱۷}} + \text{Segment}_{\text{Data:۱۸}} + \text{Segment}_{\text{Data:۱۹}} =$

$\rightarrow ۱۰ + ۵ + ۲ + ۳ = ۲۰$

$\text{Segment}_{\text{ACK}} = \text{Segment}_{\text{ACK:۱۶}} + \text{Segment}_{\text{ACK:۱۷}} + \text{Segment}_{\text{ACK:۱۸}} + \text{Segment}_{\text{ACK:۱۹}} =$

$\rightarrow (۱۰ - ۱) + (۵ - ۱) + ۲ + ۳ = ۹ + ۴ + ۲ + ۳ = ۱۸$

نتیجه اینکه مطابق فرض سوال، سگمنت‌های ۹۵، ۹۶، ۹۷ و ۹۸ در مجموع دور شانزدهم، هفدهم، هجدهم و نوزدهم یعنی ۴ سگمنت سه بار ارسال شده‌اند و سه‌بار هم به مقصد رسیده‌اند یعنی بیش از یک بار به کامپیوتر ۲ می‌رسند.

توجه: دقت کنید که سگمنت‌های loss شده، بیش از یکبار توسط فرستنده ارسال می‌شوند، اما بیش از یکبار توسط گیرنده دریافت نمی‌شوند.

نتیجه نهایی اینکه سگمنت‌های ۳۱، ۳۲ و ۳۳ در مجموع دور هفتم و هشتم یعنی ۳ سگمنت دوبار ارسال شده‌اند و دوبار هم به مقصد رسیده‌اند یعنی بیش از یک بار به کامپیوتر

۲ می‌رسند. و همچنین سگمنت‌های ۹۵، ۹۶، ۹۷ و ۹۸ در مجموع بار شانزدهم، هفدهم، هجدهم و نوزدهم یعنی ۴ سگمنت سه بار ارسال شده‌اند و سه‌بار هم به مقصد رسیده‌اند یعنی بیش از یک بار به کامپیوتر ۲ می‌رسند.

بنابراین داریم:

$$۳+۴=۷$$

شکل زیر گویای مطلب است: