

شبکه‌های کامپیوتری

نیم‌سال دوم ۱۴۰۳-۰۴
استاد: امیرمهدی صادق‌زاده



دانشکده‌ی مهندسی کامپیوتر

پاسخ‌دهنده: معین آعلی - ۴۰۱۱۰۵۵۶۱

تمرین اول

فهرست مسائل

۲	مسئله ۱
۲	الف
۲	ب
۳	ج
۴	د
۵	مسئله ۲
۵	الف
۵	ب
۵	ج
۶	د
۶	ه
۷	و
۸	مسئله ۳
۸	الف
۸	ب
۸	ج
۹	مسئله ۴
۹	الف
۹	ب
۱۰	مسئله ۵
۱۰	الف
۱۰	ب
۱۰	ج
۱۰	د
۱۱	مسئله ۶
۱۱	الف
۱۱	ب
۱۲	ج
۱۳	مسئله ۷
۱۳	الف
۱۳	ب
۱۳	ج
۱۴	مسئله ۸

پاسخ مسئله‌ی ۱.

الف

معیار / فناوری	DSL	کابل	FTTH	Wi-Fi	۴G / ۵G
پهنای باند	تا ۱۰۰ Mbps	تا ۱ Gbps	تا ۱۰ Gbps+	تا ۱ Gbps	۴G : ۱ Gbps ۵G : ۱۰ Gbps
تأخیر	۵۰-۲۰ ms	۳۰-۱۰ ms	۱۰-۱ ms	۳۰-۵ ms	۴G : ۵۰-۳۰ ms ۵G : ۱۰-۱ ms
مقیاس‌پذیری	محدود	متوسط	بسیار بالا	محلی	بالا
هزینه	کم	متوسط	بالا	کم	بالا
پوشش	گسترده	شهری خوب	محدود به مناطق مجهز	محیط داخلی	بسیار گسترده

جدول ۱: مقایسه‌ی فناوری‌های شبکه دسترسی از جنبه‌های مختلف

ب

برای هر یک از این مناطق جغرافیایی یک پیشنهاد دارم که در ادامه شرح می‌دهم.

۱. مناطق شهری با تراکم بالا

پیشنهاد: FTTH

دلایل انتخاب:

- کاربران پرمصرف (استریم، بازی، دورکاری) به سرعت بالا و تأخیر کم نیاز دارند.
- تراکم بالا، هزینه سرانه‌ی نصب فیبر را کاهش می‌دهد و توجیه‌پذیر می‌کند.
- مقیاس‌پذیری بالای FTTH امکان رشد آینده را فراهم می‌کند.

۲. مناطق روستایی

پیشنهاد: کابل ترکیبی

دلایل انتخاب:

- هزینه‌ی پایین‌تر نسبت به FTTH در مناطقی با تراکم کمتر.
- امکان استفاده از زیرساخت‌های موجود (در صورت وجود) مانند کابل تلویزیون و تلفن.
- عملکرد قابل قبول برای کاربردهایی مثل ویدئو کنفرانس و استریم با کیفیت متوسط تا بالا.

۳. مناطق دورافتاده

پیشنهاد: اتصال بی‌سیم ۴G و ۵G

دلایل پیشنهاد:

- نصب فیبر یا کابل در این مناطق هزینه‌بر و غیرعملی است.

- دکل‌های سلولی می‌توانند مساحت وسیعی را پوشش دهند.
- با نصب تجهیزات قوی‌تر کاربران می‌توانند به خدمات با کیفیت مناسب دست یابند.

ج

اجزای کلیدی معماری

DSLAM دستگاهی است که سیگنال‌های دیجیتال دریافتی از کاربران DSL را جمع‌آوری و به یک اتصال پرسرعت به سمت شبکه مرکزی (core-network) تبدیل می‌کند. معمولاً در مرکز مخابراتی یا در کابین‌های خیابانی نزدیک به کاربران نصب می‌شود. وظیفه آن تبدیل خطوط تلفن آنالوگ به سیگنال‌های دیجیتال و مدیریت ترافیک چندین کاربر DSL است.

CMTS در سمت ارائه‌دهنده خدمات قرار دارد و داده‌ها را بین اینترنت و مودم‌های کابلی در منازل کاربران رد و بدل می‌کند. در مرکز عملیات شبکه (Headend) نصب می‌شود. وظیفه آن مدیریت اتصالات کاربران، اختصاص پهنای باند، کنترل ترافیک اینترنت کابلی است.

OLT نقطه مرکزی شبکه فیبر نوری است که ارتباط بین اینترنت و چندین ONU/ONT (دستگاه‌های کاربران) را مدیریت می‌کند. در مرکز سرویس‌دهی اپراتور نصب می‌شود. وظیفه آن تبدیل سیگنال‌های نوری به الکتریکی و بالعکس، و ارسال/دریافت داده‌ها از کاربران از طریق شبکه GPON یا EPON است.

اثرات خرابی

اگر DSLAM خراب شود:

- همه کاربران DSL متصل به آن دستگاه اتصال اینترنت خود را از دست می‌دهند.
- ممکن است خطوط تلفن ثابت نیز دچار اختلال شوند (در صورت استفاده‌ی مشترک از تجهیزات).
- اختلال فقط روی محدوده‌ای مشخص از کاربران تأثیر می‌گذارد (وابسته به پوشش آن DSLAM).

اگر CMTS یا OLT خراب شود:

- CMTS : قطع ارتباط برای همه کاربران کابل متصل به آن بخش.
- OLT : از کار افتادن کل زیرشبکه فیبر که به آن OLT وصل است (صدها کاربر ممکن است تحت تأثیر قرار بگیرند).

جلوگیری از خرابی

۱. طراحی با Redundancy :

- استفاده از CMTS DSLAM یا OLT اضافی (Backup) برای پشتیبانی در صورت خرابی دستگاه اصلی.
- ایجاد مسیرهای متنوع برای اتصال کاربران به بیش از یک تجهیز اصلی.

۲. معماری توزیع شده

- استفاده از چند DSLAM یا OLT با پوشش‌های کوچک‌تر به جای یک دستگاه مرکزی بزرگ.
- کاهش تعداد کاربرانی که در صورت خرابی یک تجهیز دچار اختلال می‌شوند.

۳. دسترسی چندگانه:

- در برخی مناطق، ارائه دسترسی ترکیبی (مثلاً هم Wi-Fi و هم DSL) برای سوئیچ خودکار به مسیر پشتیبان در صورت قطعی.

د

مناطق پرتراکم - شبکه سیمی (Wi-Fi + FTTH/LAN)

برای ساختمان‌ها از FTTB یا Ethernet استفاده شود و در فضاهای داخلی پرتدد از Wi-Fi.

توجه:

- فیبر/سیمی: ارائه‌ی سرعت بسیار بالا برای استریم، آپلود فایل‌های حجیم، و کلاس‌های آنلاین.
- Wi-Fi: اتصال موبایل و لپ‌تاپ دانشجویان با سهولت و پوشش بالا در فضاهای عمومی مثل کتابخانه یا سلف.

مناطق کم‌تراکم - اتصال بی‌سیم (Outdoor-Wi-Fi / FWA)

- سیم‌کشی به این مناطق (مثل زمین‌های ورزشی یا مراکز تحقیقاتی دورافتاده) پرهزینه و غیرعملی است.
- با نصب آنتن‌های بی‌سیم نقطه به نقطه یا سلولی، دسترسی به اینترنت با تأخیر کم و هزینه پایین حاصل می‌شود.

ارتباط بین تجهیزات اصلی

- فیبر نوری Backbone برای اتصال همه بخش‌ها به دیتاسنتر یا شبکه اصلی دانشگاه.
- سوئیچ‌های توزیع و هاب‌های محلی برای مدیریت ترافیک.

پاسخ مسئله‌ی ۲.

الف

۱.

میزان تاخیر از مبدا تا اولین سویچ:

$$delay = \frac{L}{R} = \frac{10^6}{5 \times 10^6} = 0.2s$$

۲.

هر سویچ بسته را به طور کامل دریافت و سپس ارسال میکند، پس تا مقصد نهایی باید ۳ مرتبه تاخیر مرتبه قبل را حساب کنیم:

$$delay = 3 \times \frac{L}{R} = \frac{3 \times 10^6}{5 \times 10^6} = 0.6s$$

۳.

اگر N ایستگاه در مسیر داشته باشیم، پس ۱ + N مسیر داریم، پس در نهایت هم ۱ + N تاخیر داریم:

$$delay = (N + 1) \times \frac{L}{R} = \frac{(N + 1) \times 10^6}{5 \times 10^6} = (N + 1) \times 0.2s$$

ب

۱.

میزان تاخیر اولین بسته به این صورت است:

$$delay_{packet_1} = \frac{L}{R} = \frac{10^4}{5 \times 10^6} = 2ms$$

۲.

همزمان با دریافت کامل بسته اول، بسته دوم ارسال میشود که آن هم ۲ میلی ثانیه بعد به اولین سویچ میرسد. پس:

$$delay_{packet_2} = 2 \times delay_{packet_1} = 4ms$$

۳.

به تعمیم بخش قبل، زمان رسیدن بسته K ام به اولین سویچ برابر است با:

$$delay_{packet_K} = K \times delay_{packet_1} = 2K ms$$

ج

۱.

میدانیم پکت آخر در $t = 200ms$ در اولین سویچ است، پس $4ms$ بعد در زمان $t = 204ms$ به مقصد میرسد. همچنین بدون شکستن پیام، نیاز به $600ms$ داشتیم.

این اتفاق به این دلیل است که بخشی از انتقال پیام‌ها به صورت موازی جلو می‌رود. هنگامی که پکت اول در حال انتقال از سویچ دوم به سوم است، پکت دوم از سویچ اول به دوم منتقل می‌شود و ...
واضحا هرچه تعداد سویچ‌های در میان راه بیشتر باشد، این اختلاف زمانی بیشتر خواهد شد.

۲.

در صورتی که سائز بسته‌ها متفاوت باشد، زمان ارسال آن‌ها هم متفاوت خواهد بود و بر روی زمان کل تاخیر اورهد دارد. چون بسته‌های بزرگتر زمان بیشتری صرف ارسال می‌کنند و باعث کم شدن موازی سازی بسته‌های کوچک تر هم می‌شوند. نیاز به صف و تاخیر صف هم خواهیم داشت در این صورت. اما باز هم این حالت بهتر از حالت اول است که کل پیغام را یکجا ارسال کردیم!

د

۱.

در این حالت یکی از مسیرها زمان بیشتری از $s/2$ نیاز دارد، پس:

$$delay = \frac{L}{R_1} + 2 \times \frac{L}{R_2} = 0.5 + 2 \times 0.2 = 0.9s = 900ms$$

در حالت تقسیم بسته هم نیاز به $510ms$ زمان داریم.

$$delay = \frac{1000}{4 \times 1.6} = 0.15625s = 156ms \rightarrow 100 \times 5 + 2 \times 5 = 510ms$$

۲.

اگر بافر ما محدود باشد آنوقت تعدادی از بسته‌ها دراپ می‌شوند و نیاز است دوباره ارسال شوند که باعث افزایش تاخیر می‌شود.

ه

۱.

در حالت تقسیم پیام، به هر ۱۰۰ بسته یک هدر ۱۰۰ بیتی اضافه می‌شود پس کل زمان ارسال برابر است با:

$$delay = (100 + 2) \times \frac{100 + 1000}{5 \times 1.6} = 20604ms$$

در حالت ارسال بدون تقسیم بندی، این سربار زمانی به مراتب کمتر و ناچیز خواهد بود:

$$delay = 3 \times \frac{100 + 1000000}{5 \times 1.6} = 3 \times 200002ms = 600006ms$$

۲.

مکانیزم‌های زیادی وجود دارد، به عنوان مثال اگر یکی از پکت‌ها دچار مشکل شود و خراب ارسال شود، نیاز نیست کل پکت‌ها مجدداً ارسال شوند و ما می‌توانیم پکت خراب را شناسایی کنیم و فقط آن را دریافت کنیم. می‌توان در هر پکت اطلاعاتی اضافی قرار بدیم تا در صورت نیاز خطای آن را اصلاح کرد! البته هر دوی این روش‌ها مقداری سربار دارند و ما داریم اطلاعات اضافی در هر پکت قرار می‌دهیم، اما به طور کلی باعث بهبود اوضاع می‌شود و مفید است.

و

.۱

اگر یکی از لینک‌های شبکه دارای پهنای باندی متغیر بین ۲ تا ۵ مگابیت بر ثانیه باشد، در حالتی که پیام به صورت یکپارچه و بدون تقسیم ارسال شود، ممکن است کل پیام در زمانی منتقل شود که لینک در کمترین پهنای باند خود قرار دارد. این موضوع باعث افزایش چشم‌گیر زمان انتقال می‌شود.

اما اگر پیام به بخش‌های کوچکتر تقسیم شود، این امکان به وجود می‌آید که بسته‌ها در زمان‌هایی که پهنای باند بالاتر است، منتقل شوند. در نتیجه، بخشی از داده‌ها سریع‌تر جابجا شده و زمان کلی انتقال کاهش می‌یابد.

.۲

در چنین شرایطی، استفاده از Adaptive-Segmentation و تنظیم پویا اندازه بسته‌ها بر اساس شرایط لحظه‌ای شبکه، می‌تواند بهره‌وری انتقال را به شکل قابل توجهی افزایش دهد و عملکرد کلی شبکه را بهبود بخشد. به این صورت که در زمان ازدحام، بسته‌های کوچک تری ارسال می‌شود.

پاسخ مسئله‌ی ۳.

زمان مورد نیاز برای جست‌وجوی DNS به این صورت است:

$$RTT_1 + RTT_2 + RTT_3 + RTT_4 = 250 + 150 + 100 + 50 = 500ms$$

الف

در این حالت به ازای هر فایل رو کانکشن TCP تشکیل داده و فایل را دریافت میکنیم. به ازای هر فایل هم به ۲ RTT نیاز داریم. یکی برای تشکیل کانکشن TCP و یکی هم برای دریافت فایل. پس در مجموع:

$$DNS + 3 \times (2 \times RTT_1) = 500 + 6 \times 300 = 2300ms$$

ب

در این حالت هر سه فایل موازی و با یک کانکشن مخصوص به خود دریافت میشوند، پس برای هر سه فایل نیاز به ۲ RTT داریم. زمان DNS هم که همچنان نیاز است، پس:

$$DNS + (RTT_1 + RTT_1) = 300 + 300 + 500 = 1100ms$$

ج

در این حالت فقط ۱ کانکشن TCP داریم که همه فایل ها یکی یکی از همین دانلود میشوند. پس یک DNS داریم و یک RTT برای اتصال TCP و به ازای هر فایل هم یک RTT. پس:

$$DNS + (RTT_1 + RTT_1 + RTT_1 + RTT_1) = 500 + 1200 = 1700ms$$

پاسخ مسئله‌ی ۴.

پهنای باند اختصاص داده شده به هر اتصال:

$$r = \frac{500}{N} \text{ bps}$$

زمان ارسال یک بسته کنترلی:

$$t_{ctrl} = \frac{500 \text{ bit}}{\frac{500}{N} \text{ bps}} = Ns$$

زمان ارسال یک بسته داده کوچک:

$$t_{100k} = \frac{100000}{\frac{500}{N}} = 200Ns$$

زمان ارسال یک بسته داده بزرگ:

$$t_{300k} = \frac{300000}{\frac{500}{N}} = 600Ns$$

الف

زمان‌هایی که در این حالت باید بررسی کنیم عبارتند از:

- سه بسته کنترل برای Handshaking TCP
- یک بسته کنترل برای GET + تاخیر پردازش
- ارسال داده
- بسته کنترلی برای ACK نهایی

به هر یک از تاخیرهای بالا یک تاخیر صف هم اضافه خواهد شد.

$$T = 3 \times (t_{ctrl} + d_{queue}) + (t_{ctrl} + d_{queue} + d_{proc}) + (t_{data} + d_{queue}) + (t_{ctrl} + d_{queue}) = 5N + 0.1 + 6 \times 0.05 + t_{data}$$

با توجه به اینکه نسبت تعداد بسته‌های کوچک و بزرگ برابر است، پس بین طول آن‌ها میانگین گرفته و فرض میکنیم

$$t_{200k} = \frac{200000}{\frac{500}{N}} = 400Ns \text{ پس زمان ارسال آن‌ها برابر است با: } 400Ns$$

پس در نهایت زمان ما برای یک شی برابر است با:

$$5N + 0.1 + 6 \times 0.05 + t_{data} = (40.5N + 0.4)s$$

ما در مجموع ۲۱ شی داریم، یکی فایل html و ۲۰ فایل ارجاع شده. و چون N اتصال موازی هستند، پس زمان نهایی برابر است با:

$$T_{total} = \frac{21 \times (40.5N + 0.4)}{N} = 850.5 + \frac{84}{N}$$

با توجه به سربار اتصال‌های موازی و ثابت بودن زمان ۸۵۰.۵ ثانیه‌ای، افزایش N عملاً کمکی به کاهش زمان کل دانلود نمی‌کند. پس این کار خیلی منطقی نیست.

ب

به صورت شهودی واضح است که بهبود قابل توجهی حاصل نخواهد شد. چون بخش بیشتر زمان دانلود به دلیل حجم بالای داده‌ها و لینک بسیار کم سرعت است. با HTTP پایا مقدار کمی بهبود سرعت داریم اما در برابر زمان انتقال داده‌ها بسیار ناچیز است. پس با این کار زمان دانلود مقدار کمی بهبود می‌یابد، اما آنقدر قابل توجه نیست. برای بهبود سرعت یا باید ظرفیت لینک را افزایش داد یا حجم داده‌ها را کم کرد.

پاسخ مسئله‌ی ۵.

فرض میکنیم که ریکوئست‌ها به صورت نرمال در ۲ دقیقه زده شده‌اند. پس بین هر ۲ ریکوئست ۱۲ ثانیه فاصله است.

الف

پرسش به ریشه و دریافت ارجاع به TLD : $50ms$

پرسش به سرور TLD و دریافت ارجاع به Authoritative : $40ms$

پرسش نهایی برای آدرس `www.example.com` و دریافت پاسخ نهایی : $30ms$

بازگشت پاسخ نهایی به کلاینت : $10ms$

پس در مجموع $130ms$ برای دریافت نتیجه زمان لازم است.

ب

در این صورت درخواست اول و ششم کش ندارند و باید $130ms$ منتظر بمانیم. اما باقی دستورات در DNS محلی کش شده‌اند و هر کدام $10ms$ زمان نیاز دارند. پس:

$$2 \times 130 + 8 \times 10 = 340ms$$

ج

در این صورت درخواست‌های اول و چهارم و هفتم و دهم کش ندارند و هر کدام $130ms$ زمان نیاز دارند. باقی درخواست‌ها هر کدام $10ms$ نیاز دارند. پس:

$$4 \times 130 + 6 \times 10 = 580ms$$

د

افزایش TTL در Authoritative-DNS

در این صورت مدت زمان بیشتر آدرس کش میشود و نیاز نیست به DNS Root ریکوئست زده شود. البته از معایب این کار هم دیرتر اپدیت شدن تغییر ip است. البته میتوان با یک عدد معقول حداکثر بهره را از کش کردن برد.

استفاده از کش سمت مرورگر کاربر

در حال حاضر فقط سرور DNS محلی کش دارد (به مدت ۶۰ ثانیه). اگر کلاینت یا مرورگر نیز بتواند نتایج resolve شده را در سطح خودش کش کند، دیگر نیازی به تماس مجدد با سرور DNS محلی در هر درخواست نخواهد بود.

استفاده از DNS-Prefetching در مرورگر

مرورگر یا برنامه، دامنه‌هایی که احتمال می‌دهد کاربر به اون‌ها مراجعه کند رو از قبل resolve می‌کنه.

پاسخ مسئله‌ی ۶.

میدانیم که:

$$D_{CS} \geq \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$$

$$D_{P2P} \geq \max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum u_i}\right\}$$

همچنین:

$$F = 4 \text{ Gbit} = 4000 \text{ Mbit}$$

$$d = 2 \text{ Mbps}$$

$$u_s = 20 \text{ Mbps}$$

الف

در حالت Client-Server داریم:

$$D_{CS} \geq \max\left\{\frac{N \times 40 \times 1024}{20}, \frac{40 \times 1024}{2}\right\} = \max\{2048N, 20480\}$$

Delay	N
ms50 + s20480	10
ms50 + s204800	100
ms50 + s2048000	1000

در حالت P2P فرض میکنیم اپلود همگن است، پس: $\sum u_i = N \times u$

پس برای این حالت از این رابطه استفاده میکنیم:

$$D_{P2P} \geq \max\left\{\frac{40960}{20}, \frac{40960}{2}, \frac{40960N}{20+Nu}\right\} = \max\{20480, 2048, \frac{40960N}{20+Nu}\}$$

Delay(s)	u(Mbps)	N
20	0.3	10
20	0.7	10
20	2	10
82	0.3	100
46	0.7	100
20	2	100
128	0.3	1000
57	0.7	1000
20	2	1000

پس در $N = 10$ تاخیر به زمان دانلود محدود میشود، اما در N های بیشتر، حالت P2P نتیجه بهتری دارد.

ب

وقتی ۲۰٪ از همتاها هر ۵ دقیقه از شبکه جدا می‌شوند، بلوک‌هایی که دریافت کرده‌اند در شبکه باقی نمی‌مانند و مجبوریم آن‌ها را مجدداً از سرور یا دیگر همتاها درخواست کنیم. این امر باعث بار اضافه روی سرور و طولانی‌تر شدن توزیع می‌شود.

اگر نرخ آپلود هر همتا از توزیع نرمال با $\sigma = 0.2u$ پیروی کند، برخی پییرها کندتر و برخی سریع‌تر عمل می‌کنند. نبود تعادل، جریان گردش بلوک‌ها را مختل و زمان کل را به‌طور قابل توجهی افزایش می‌دهد.

پهنای باند سرور با دوره‌ی ۱۰ دقیقه و دامنه‌ی $\pm 10\%$ نوسان دارد. در فازهای افت، ظرفیت ارسال کمتر شده و تأمین

بلوک‌های اولیه کند می‌شود که زمان توزیع را بیشتر می‌کند.

RTT حدود $50ms$ بین سرور و همتا و $100ms$ بین همتاها باعث تأخیر در ارسال ACK و درخواست بلوک بعدی می‌شود. این انتظار اضافی نرخ گردش داده را کاهش و توزیع را کندتر می‌کند.

با ترکیب این چهار عامل، توزیع فایل از چند دقیقه‌ای ایده‌آل به ده‌ها دقیقه در سناریوی واقعی کشیده می‌شود.

ج

- ایجاد انگیزه برای ماندگاری: با مکانیزم اعتبار و پاداش دهی، کاربران را تشویق کنید پس از دریافت کامل فایل همچنان فعال بمانند.
- یک یا چند همتای قدرتمند با پهنای باند بالا را همیشه به عنوان seed نگه دارید تا هر زمان churn بالا رفت بتوانند جایگزین بلوک‌های از دست رفته شوند.
- هر همتا پیش از ترک شبکه چند بلوک اضافی برای همتا‌های دیگر دانلود و نگهداری کند تا در دوره‌های خروج انبوه، قطع سرویس نداشته باشیم.
- همتاها را بر اساس تاخیر شبکه گروه‌بندی کنید و در اولویت با همتا‌های نزدیک‌تر یا با RTT کمتر دانلود/آپلود کنید تا مدت زمان انتقال هر بلوک کوتاه‌تر شود.

پاسخ مسئله‌ی ۷.

الف

پایین آوردن سریع بیت‌ریت در سطح بافر بحرانی به موقع است. با پایین آوردن بیت‌ریت به ۱ Mbps، احتمال پر شدن بافر و جلوگیری از توقف پخش افزایش می‌یابد. استفاده از اطلاعات بیت‌ریت قبلی و پهنای باند جاری کمک می‌کند که بیت‌ریت انتخابی با شرایط واقعی شبکه تطبیق داشته باشد. الگوریتم فقط به وضعیت فعلی نگاه می‌کند (نه تغییرات آتی یا روند پهنای باند). این باعث می‌شود در مواجهه با نوسانات سریع، پیش‌بینی‌پذیری پایین بیاید. همچنین افزایش یا کاهش‌های بیش از حد کیفیت در این الگوریتم به همراه داریم. الگوریتم در بیشتر شرایط پایدار یا نیمه‌پایدار پهنای باند عملکرد خوبی دارد و از Rebuffering جلوگیری می‌کند. اما اگر پهنای باند به طور ناگهانی و سریع کاهش یابد و سطح بافر هنوز بالای ۵ ثانیه باشد، ممکن است الگوریتم به اندازه کافی سریع واکنش نشان ندهد، و خطر Rebuffering به وجود بیاید.

ب

به عنوان مثال:

پهنای باند در دسترس: $B_t = 6 Mbps$

سطح بافر: $L_t = 6s$

بیت‌ریت آخرین قطعه: $R_t = 4 Mbps$

چون $L_t > 5s$ است، شرط اول برقرار می‌شود و بیت‌ریت به پایین‌ترین مقدار کاهش می‌یابد. از آنجا که ما اکنون با بیت‌ریت ۱ شروع کرده‌ایم، حتی اگر بافر کم شود یا پهنای باند بالا باشد، شرط ارتقا هیچ‌گاه فعال نمی‌شود (چون $R_t/2 = 0.5 < 1$) و در نتیجه سیستم در ۱ Mbps قفل می‌ماند.

معمولاً وقتی بافر کم است، باید بیت‌ریت را کاهش دهیم و وقتی بافر بسیار بالاست، می‌توانیم به تدریج افزایش دهیم. اما اینجا برعکس پیاده شده، به محض رسیدن بافر بالاتر از ۵ ثانیه، بیت‌ریت ناگهان به حداقل افت می‌کند، حتی وقتی شبکه خیلی خوب است.

پیشنهاد بهبود: استفاده از میانگین‌گیری از وضعیت پهنای باند و بافر در چند لحظه به جای بررسی فقط یک لحظه‌ی خاص.

ج

به عنوان مثال:

پهنای باند ثابت: $B = 3.5 Mbps$

طول هر قطعه ویدئو: ۲ ثانیه

شروع با بیت‌ریت $R = 2 Mbps$ و بافر اولیه $L = 7s$

سطح بافر دائم بین خالی شدن تا زیر ۵s و پر شدن تا بالای ۱۰s نوسان می‌کند و کیفیت ویدئو پیوسته تغییر می‌کند. پیشنهادات برای رفع این مشکل:

- استفاده از Smoothing-Filter به این صورت که به جای استفاده از B_t لحظه‌ای، از میانگین متحرک استفاده کند.
- یک شرط اضافه کنیم که یک حداکثر تغییراتی را بررسی کند تا ویدئو یک دفعه از بالاترین کیفیت به پایین‌ترین کیفیت نپرد.

پاسخ مسئله‌ی ۸.

گزارش پروژه