بسمه تعالی

دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

درس شبکه‌های کامپیوتری، نیمسال دوم سال تحصیلی ۱۴۰۴-۱۴۰۳

پاسخ تمرین سری چهارم

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

سوال ۱:

در یک برنامه کاربردی نظیر به نظیر (P2P) توزیع فایل، می‌خواهیم یک فایل ۱ گیگابیتی را بین ۱۰۰ کاربر توزیع کنیم. اگر نرخ دریافت (download) هرکاربر d_i برابر ۱۲۸ کیلوبیت بر ثانیه و نرخ ارسال (upload) هر کاربر u_i برابر ۱۰ کیلوبیت بر ثانیه باشد و نرخ ارسال کاربر اولیه دارنده فایل نرخ ۱۰۰ کیلوبیت بر ثانیه باشد، مطلوبست حداقل زمان لازم برای توزیع این فایل بین ۱۰۰ کاربر مورد نظر.

پاسخ:

$\mathrm{F} = 1\ \mathrm{GB} = 2^{30}\ \mathrm{bits}$

$d_i = 128\ \mathrm{Kbps} = 128 \times 10^3\ \mathrm{bps} \qquad for\ i = 1.2.\cdots.N$

$u_i = 10\ \mathrm{Kbps} \ = 10 \times 10^3\ \mathrm{bps} \qquad for\ i = 1.2.\cdots.N$

$u_s = 100\ \mathrm{Kbps} = 100 \times 10^3\ \mathrm{bps}$

$N = 100$

$$D_{P2P} = max\ \{F/u_s\ .F/d_{min}\ .NF/(u_s + \sum_{i=1}^{N} u_i)\}$$

$$\frac{F}{u_s} = \frac{2^{30}}{100 \times 10^3} = \frac{1.07 \times 10^9}{10^5} = 1.07 \times 10^4\ \mathrm{Sec}$$

$$\frac{F}{d_{min}} = \frac{2^{30}}{128 \times 10^3} = \frac{1.07 \times 10^9}{128 \times 10^3} = 8.39 \times 10^3\ \mathrm{Sec}$$

$$\frac{NF}{u_s + \sum_{i=1}^{N} u_i} = \frac{NF}{u_s + Nu_i} = \frac{100 \times 2^{30}}{100 \times 10^3 + 100 \times 10 \times 10^3} = \frac{1.07 \times 10^{11}}{1.1 \times 10^6} = 9.76 \times 10^4\ \mathrm{Sec}$$

$D_{P2P} = 9.76 \times 10^4\ \mathrm{Sec}$

سوال ۲:

فرض کنید می‌خواهیم فایلی به اندازه ۸ گیگابایت را بین N نظیر (peer) توزیع کنیم. اگر نرخ ارسال (upload) سرویس‌دهنده ۲۰ مگابیت بر ثانیه و نرخ دریافت (download) و نرخ ارسال (upload) هر نظیر به ترتیب ۱۰ مگابیت بر ثانیه و ۵۰ کیلوبیت بر ثانیه باشد، به ازای $N = 10,\ 100$ حداقل زمان توزیع این فایل را در دو معماری نظیر به نظیر (P2P) و سرویس‌دهنده-سرویس‌گیرنده (client-server) بدست آورید. با افزایش تعداد نظیرها حداقل زمان توزیع فایل در این دو معماری چگونه تغییر می‌کند؟

پاسخ:

$F = 8\ \mathrm{GB} = 8 \times\ 2^{30} \times 8 = 2^{36}\ \mathrm{bit}$
$d_i = 10\ \mathrm{Mbps}\ \ for\ i = 1.2. \ldots . N$
$u_i = 50\ \mathrm{Kbps}\ \ for\ i = 1.2. \ldots . N$
$u_S = 20\ \mathrm{Mbps}$

$$D_{cs} = \max\ (\frac{NF}{u_s}.\frac{F}{d_{min}})$$

$$D_{P2P} = \max\left(\frac{F}{u_s}.\frac{F}{d_{min}}.\frac{NF}{u_s + \sum_1^N u_i}\right) = \max(\frac{F}{u_s}.\frac{F}{d_{min}}.\frac{NF}{u_s + Nu_i})$$

		NF/u_s	F/d_{min}	F/u_s	$NF/(u_s+Nu_i)$	D_{cs}	D_{P2P}	D_{cs}/D_{P2P}
N	10	34360	6872	3436	33522	34360	33522	1.02
	100	343600	6872	3436	274878	343600	274878	1.25

همانطور که انتظار می‌رفت حداقل زمان توزیع فایل در معماری P2P از معماری Client-server کمتر است و هرچه تعداد نظیرها بیشتر شود، تفاوت بیشتر نیز می‌شود.

سوال ۳:

الف) مفاهیم زیر در معماری BitTorrent را تعریف کنید.

- Tracker
- Leecher
- Seed

ب) هدف BitTorrent دریافت (download) همزمان از نظیرها در سریع‌ترین زمان ممکن است. برای این کار با استفاده از استراتژی tit-for-tat، باید مشخص شود که قطعه‌ها (chunks) از کدام نظیرها دریافت شوند. همچنین باید مشخص شود این قطعه‌ها به چه ترتیبی باید دریافت شوند. درباره‌ی استراتژی ترتیب دریافت قطعه‌ها (The piece selection algorithm) به طور کامل توضیح دهید.

ج) انواع حملات به شبکه BitTorrent را نام ببرید و به طور مختصر توضیح دهید.

پاسخ:

الف)

- Tracker: ردیاب مرکزی (Tracker)، فهرستی از آدرس کاربران فعال را در اختیار سایر کاربران قرار می‌دهد. این کاربران سپس با یکدیگر تماس می‌گیرند تا بخش‌هایی از فایل را از یکدیگر دریافت کنند. ردیاب به‌طور مستقیم در انتقال داده‌ها دخالت ندارد. ردیاب و کاربران دانلودکننده از یک پروتکل ساده مبتنی بر HTTP برای تبادل اطلاعات استفاده می‌کنند. ابتدا، کاربر اطلاعاتی مانند فایل مورد نظر برای دانلود، شماره پورت‌هایی که به آن‌ها گوش می‌دهد و... را به ردیاب اعلام می‌کند. پاسخ ردیاب، فهرستی از سایر کاربرانی است که همان فایل را دانلود می‌کنند.
- Leecher: کاربر دانلود کننده که هیچی یا بخش‌هایی از یک فایل را دارد leecher نامیده می‌شود.
- Seed: Seed کاربری است که کل فایل را در اختیار دارد. Seed باید حداقل یک نسخه کامل از فایل را آپلود کند. زمانی که یک نسخه کامل از فایل بین سایر دانلودکنندگان پخش شد، در صورتی که دانلودکننده به تعداد کافی وجود داشته باشد و همه بخش‌های فایل را در دسترس نگه‌دارند، seed می‌تواند آپلود را متوقف کند و دانلود برای همه کاربران ادامه خواهد داشت. برای یک فایل محبوب، ممکن است تنها یک نسخه کامل از طرف seed کافی باشد، اما برای فایلی که محبوبیت کمتری دارد، ممکن است نیاز به آپلود مداوم از سوی seed وجود داشته باشد. نتیجه این است که نیاز به پهنای باند برای ناشر فایل، در صورتی که تعداد زیادی از افراد مشغول به دانلود باشند، کاهش می‌یابد.

ب) بیت‌تورنت از پروتکل TCP استفاده می‌کند و بنابراین بسیار مهم است که فرآیند انتقال داده همواره ادامه داشته باشد؛ در غیر این صورت، نرخ انتقال به دلیل مکانیسم شروع آهسته (slow start) کاهش می‌یابد. قطعه‌ها به زیرقطعه‌هایی با اندازه‌ای معمولاً حدود ۱۶ کیلوبایت تقسیم می‌شوند. پروتکل تضمین می‌کند که همواره چند درخواست (معمولاً پنج عدد) برای زیرقطعه‌ها به‌صورت خط لوله‌ای فعال باشد. به محض دریافت یک زیرقطعه، درخواست جدیدی ارسال می‌شود. زیرقطعه‌ها می‌توانند از کاربران مختلف دریافت شوند.

- سیاست سخت‌گیرانه (Strict Policy): وقتی یک زیرقطعه درخواست داده می‌شود، باقی زیرقطعه‌های همان قطعه باید قبل از رفتن به قطعه‌های دیگر، دریافت شوند. این کار باعث می‌شود یک قطعه کامل هرچه سریع‌تر بدست آید.

- اولویت با کمیاب‌ترین (Rarest First): زمانی که یک همتا می‌خواهد قطعه بعدی را برای دانلود انتخاب کند، قطعه‌ای را انتخاب می‌کند که کمترین تعداد همتا آن را دارند.
- قطعه تصادفی اول (Random First Piece): سیاست rarest first همیشه کارآمدترین نیست. قطعات کمیاب معمولاً کندتر دانلود می‌شوند، چون زیرقطعه‌های آن‌ها فقط از یک یا تعداد کمی همتا قابل دریافت هستند. بنابراین، ابتدا یک قطعه به‌صورت تصادفی انتخاب و دانلود می‌شود. پس از کامل شدن قطعه اول، سپس به سیاست rarest first تغییر داده می‌شود.
- حالت پایان بازی (Endgame Mode): گاهی ممکن است قطعه‌ای با سرعت انتقال پایین دانلود شود، تمام زیرقطعه‌هایی که کاربر هنوز دریافت نکرده، در حال درخواست هستند. همان درخواست به همه همتایان ارسال می‌شود. این کار باعث می‌شود آخرین بخش‌های فایل با بیشترین سرعت ممکن دریافت شوند. وقتی یک زیرقطعه رسید، پیامی برای لغو درخواست به سایر همتایان فرستاده می‌شود تا از ادامه ارسال آن زیرقطعه صرف‌نظر کنند.

ج)

- Index Poisoning : index به کاربران اجازه می‌دهد تا آدرس IP کاربرانی که محتوای مورد نظر را دارند، پیدا کنند. این نوع حمله، فرآیند جست‌وجو برای یافتن کاربران را دشوار می‌سازد. مهاجم مقدار زیادی اطلاعات نامعتبر را وارد index می‌کند تا مانع از یافتن اطلاعات صحیح توسط کاربران شود. ایده این است که با مجبور کردن همتا برای دانلود بخش‌هایی از فایل از منابع نامعتبر، روند دانلود را کند کنند.
- Decoy Insertion: در این نوع حمله، نسخه‌های خراب یا ناقص از یک فایل در شبکه منتشر می‌شود. تصور کنید ۵۰۰ نسخه از یک فایل وجود داشته باشد و فقط ۲ نسخه واقعی باشند؛ این موضوع باعث می‌شود که کاربران نتوانند فایل واقعی را پیدا کنند. بیشتر وب‌سایت‌هایی که لیست تورنت‌ها را نمایش می‌دهند، دارای سیستم رأی‌دهی هستند. این ویژگی باعث کاهش اثر این نوع حمله می‌شود، چرا که فایل‌های سالم در صدر نتایج جست‌وجو قرار می‌گیرند.

سوال ۴:

الف) توضیح دهید که پروتکل DASH چگونه عمل می‌کند و به چه صورت فایل‌های ویدیویی به تکه‌های کوچک تقسیم می‌شوند. چگونه DASH با استفاده از پهنای باند موجود کاربر، کیفیت ویدیو را تنظیم می‌کند و جریان‌سازی ویدیو را بهینه می‌سازد؟

ب) توضیح دهید که چگونه فایل‌های آشکارساز در DASH مورد استفاده قرار می‌گیرند و چه نقشی در انتخاب نرخ کدگذاری مناسب برای کاربر ایفا می‌کنند.

ج) جریان‌سازی ویدیو ذخیره شده چگونه عمل می‌کند و چه تفاوت‌هایی با جریان‌سازی DASH دارد؟

پاسخ:

الف) پروتکل DASH (Dynamic Adaptive Streaming over HTTP) برای جریان‌سازی ویدئو به صورت پویا و تطبیقی عمل می‌کند. در این روش، فایل ویدئویی به چندین تکه کوچک (segments) با طول‌های مشخص (برای مثال ۲ تا ۱۰ ثانیه) تقسیم می‌شود. هر تکه از ویدئو در چندین نرخ bit rate و کیفیت مختلف ذخیره می‌شود. این پروتکل بر اساس پهنای باند موجود کاربر عمل می‌کند؛ به این صورت که اگر پهنای باند بالاست، ویدئو با کیفیت بالا پخش می‌شود و اگر پهنای باند کم باشد، کیفیت کاهش می‌یابد. DASH به این ترتیب با توجه به شرایط شبکه، کیفیت ویدئو را تنظیم می‌کند تا جریان‌سازی ویدیو بدون وقفه انجام شود.

ب) در پروتکل DASH، فایل‌های آشکارساز (Manifest files) مانند MPD (Media Presentation Description) استفاده می‌شوند. این فایل‌ها اطلاعات مربوط به تمام تکه‌های ویدئویی، نرخ بیت مختلف و مکان قرارگیری هر تکه را در بر دارند. نقش اصلی فایل آشکارساز این است که اطلاعاتی درباره کیفیت‌های موجود را به پخش‌کننده بدهد تا بر اساس پهنای باند و توان پردازشی کاربر، به‌ترین نرخ کدگذاری (Bitrate) را انتخاب کند.

ج) در جریان‌سازی ویدیو ذخیره‌شده (HTTP Streaming)، ویدیو به‌صورت یک فایل معمولی با URL مشخص روی یک سرویس‌دهنده HTTP ذخیره می‌شود. وقتی که کاربر بخواهد ویدیو را ببیند، مرورگر یا اپلیکیشن او به عنوان سرویس‌گیرنده با سرویس‌دهنده ارتباط برقرار می‌کند و یک درخواست HTTP GET برای فایل ویدیو می‌فرستد. سرویس‌دهنده، ویدیو را به‌صورت پیوسته برای سرویس‌گیرنده می‌فرستد و این داده‌ها

در بافر سمت سرویس‌گیرنده ذخیره می‌شود. زمانی که حجم داده‌های دریافت‌شده از یک حد مشخص فراتر رود، پخش آغاز می‌شود. با این‌که این روش ساده است، اما با چالش‌هایی مواجه است، مانند تغییرات پهنای باند به‌دلیل ازدحام در بخش‌های مختلف شبکه، افت کیفیت تصویر یا افزایش تأخیر به‌دلیل از دست رفتن بسته‌ها و ازدحام در شبکه، نبود امکان تطبیق کیفیت ویدیو با شرایط شبکه (مثلاً در اتصال‌های موبایلی)، و ناتوانی در تعاملات سرویس‌گیرنده مانند جلو بردن سریع یا عقب بردن و پرش در ویدیو.

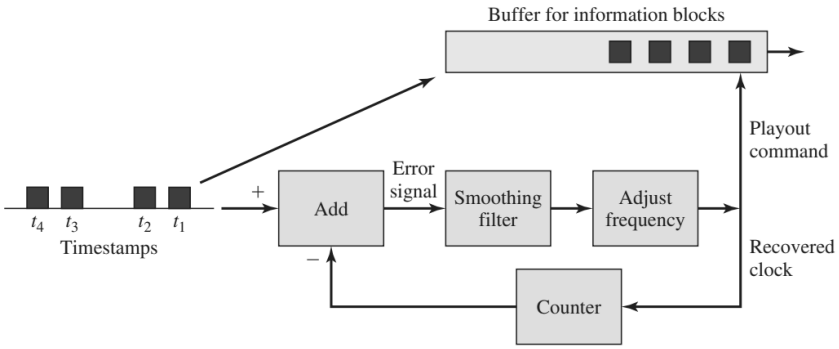
جریان‌سازی تطبیقی پویا بر بستر HTTP یا همان DASH (Dynamic Adaptive Streaming over HTTP) این مشکلات را تا حد زیادی حل کرده است. در DASH، ویدیو ابتدا به چند نسخه با نرخ بیت مختلف (و کیفیت‌های متفاوت) کدگذاری و تقسیم به تکه‌هایی (chunks) چندثانیه‌ای می‌شود و سرویس‌دهنده یک فایل manifest ارائه می‌دهد که شامل URL و اطلاعات نرخ بیت هر نسخه است. سپس سرویس‌گیرنده ابتدا این فایل را دریافت کرده و سپس بسته به شرایط لحظه‌ای پهنای باند، مناسب‌ترین تکه از نسخه‌ای خاص را با یک HTTP GET درخواست می‌کند. اگر پهنای باند زیاد باشد، تکه‌هایی با کیفیت بالا انتخاب می‌شود و اگر کم باشد، تکه‌هایی با کیفیت پایین‌تر انتخاب می‌شوند.

ویژگی	DASH	HTTP Streaming
نوع ویدیو	چند نسخه با نرخ بیت مختلف	یک نسخه ثابت
سازگاری با تغییر پهنای باند	دارد	ندارد
هوشمندی در سمت سرویس‌گیرنده	دارد (تحلیل پهنای باند، انتخاب نرخ مناسب)	ندارد
کنترل تأخیر و بافر	پیشرفته‌تر، با استفاده از الگوریتم‌های تطبیقی	محدود
تعامل کاربر	روان‌تر، قابل تطبیق با موقعیت شبکه	محدود و وابسته به پخش‌کننده
کیفیت ویدیو	متناسب با کیفیت اتصال کاربر	یکسان برای همه

سوال ۵:

الف) تصویر زیر نشان‌دهنده چه مشکلی در انتقال اطلاعات در شبکه است؟ در چه کاربردهایی این رخداد اهمیت دارد؟

ب) مکانیزم زیر چگونه در حل این مشکل کمک می‌کند؟



ج) برای هر کدام از بخش‌های smoothing filter و adjust frequency، یک الگوریتم را مثال بزنید و نحوه کارکرد آن را توضیح دهید.

پاسخ:

الف) اطلاعات ویدیویی و صوتی به صورت همزمان و دوره‌ای تولید می‌شوند. برای بازسازی سیگنال در سمت گیرنده، اطلاعات باید با همان نرخی به decoder داده شود که در encoder تولید شده است. تصویر اثر انتقال اطلاعات از طریق شبکه را به رابطه زمانی بین بلوک‌های اولیه نشان می‌دهد. به طور کلی، شبکه‌ها باعث ایجاد تغییرهای متغیر می‌شوند و برخی بلوک‌ها نسبت به سایرین زودتر یا دیرتر می‌رسند. این پدیده به نام لرزش زمانی یا جیتر (timing jitter) شناخته می‌شود.

ب) شکل، سیستمی را نشان می‌دهد که در آن دنباله ای از timestampها برای بازیابی ساعت (clock recovery) استفاده می‌شود. این timestampها با نمونه‌برداری(sampling) از یک شمارنده که توسط ساعت فرستنده تولید شده، گرفته می‌شوند. سیستم گیرنده دارای یک شمارنده است که تلاش می‌کند با ساعت فرستنده هماهنگ شود. مقادیر timestamp ها با شمارنده گیرنده مقایسه می‌شود و یک سیگنال خطا از این مقایسه تولید می‌شود. با توجه به این سیگنال، فرکانس ساعت محلی تنظیم می‌شود. از ساعت تنظیم‌شده برای کنترل پخش داده‌ها از بافر استفاده می‌شود.

ج) Smoothing filter: این بخش مانع از پاسخ سریع سیستم به نوسانات موقتی سیگنال خطا می‌شود و صرفا خطای تغییرات پایدار و واقعی که نشان دهنده تغییر فرکانس هستند، نشان داده شود. برای این کار میتوانیم از الگوریتم میانگین متحرک نمایی(exponential moving average) استفاده کنیم:

$$New\ Filtered\ Signal = (Smoothing\ Factor - 1) \times Old\ Filtered\ Signal + Smoothing\ Factor \times Error\ Signal$$

Adjust Frequency: این بخش با استفاده از مقدار فیلترشده سیگنال خطا، فرکانس ساعت محلی را تنظیم می‌کند. اگر ساعت عقب باشد، سرعت افزایش می‌یابد و اگر جلوتر باشد، سرعت کاهش می‌یابد. برای پیاده‌سازی این بخش می‌توانیم از الگوریتم Digital Phase-Locked Loop استفاده کنیم:

$$New\ Frequency = Filtered\ Signal \times Proportional\ Gain + Old\ Frequency$$

سوال ۶:

الف) تفاوت دو فلسفه "Enter Deep" و "Bring Home" در CDN چیست؟

ب) استراتژی‌هایی که برای انتخاب بهترین سرویس‌دهنده CDN وجود دارند را نام برده و به اختصار شرح دهید.

ج) CDN چگونه از DNS برای سد کردن و تغییر مسیر درخواست بهره می‌برد؟

پاسخ:

الف) در فلسفه Enter Deep سعی بر این است که سرویس‌دهنده‌های CDN در عمق شبکه‌های دسترسی ISPهای مختلف در سراسر جهان وارد شوند و خوشه‌های سرویس‌دهنده را در آنجا قرار داده شوند. هدف این روش، نزدیکی بیشتر به کاربران نهایی برای کاهش تأخیر و افزایش سرعت بازیابی محتواست. با کم کردن تعداد لینک‌ها و روترها بین کاربر و سرویس‌دهنده CDN، مسیر بین کاربر و سرویس‌دهنده کاهش می‌یابد که منجر به کاهش تاخیر می‌شود. این روش به دلیل تعداد زیاد سرویس‌دهنده‌ها در مکان‌های مختلف، مدیریت و نگهداری پیچیده‌تری دارد.

در فلسفه Bring Home سرویس‌دهنده‌ها به جای قرار گرفتن در شبکه‌های دسترسیISP، در نقاط تبادل اینترنت (IXPs) قرار داده می‌شوند. در واقع این خوشه‌های سرویس‌دهنده‌های CDN در تعداد کمتری ساخته می‌شوند ولی در مکان‌های کلیدی مستقر می‌شوند. این روش معمولاً هزینه‌های نگهداری و مدیریت کمتری دارد، اما ممکن است به دلیل افزایش تأخیر و کاهش پهنای باند، throughput پایین‌تری داشته باشد.

ب) دو استراتژی متداول عبارتند از:

- انتخاب نزدیک‌ترین سرویس‌دهنده جغرافیایی: براساس پایگاه‌داده‌های موقعیت‌یابی مانند MaxMind وQuova ، نزدیک‌ترین خوشه‌ی CDN انتخاب می‌شود.
- انتخاب بر اساس شرایط لحظه‌ای شبکه: با اندازه‌گیری تأخیر و پهنای باند در لحظه، CDN بهترین سرویس‌دهنده را از نظر کیفیت ارتباط انتخاب می‌کند.

ج) فرآیند تغییر مسیر درخواست کاربر توسط CDN به وسیله DNS به این صورت است:

۱– کاربر با کلیک بر روی لینک ویدیو درخواست خود را شروع می‌کند.

۲– دستگاه کاربر یک درخواست DNS برای دریافت IP سرویس‌دهنده ارسال می‌کند.

۳– سرویس‌دهنده DNS محلی درخواست کاربر را به سرویس‌دهنده DNS تولیدکننده محتوی ارسال می‌کند.

۴- سرویس‌دهنده DNS تولیدکننده محتوی با توجه به نام زیردامنه، به جای ارائه IP، با برگرداندن hostname درخواست را به زیرساخت DNS اختصاصی CDN ارجاع می‌دهد.

۵- زیرساخت DNS متعلق به CDN بر اساس معیارهایی مانند موقعیت جغرافیایی، بار ترافیکی سرویس‌دهنده‌ها و... یک سرویس‌دهنده مناسب را انتخاب کرده و IP آن سرویس‌دهنده را برمی‌گرداند.

۶- سیستم کاربر آدرس IP را دریافت کرده و با سرویس‌دهنده CDN اتصال برقرار می‌کند و محتوی را دریافت می‌کند.

سوال ۷:

با توجه به برنامه‌نویسی سوکت برای سوکت‌های UDP و TCP در سمت سرویس‌گیرنده (client) و سرویس‌دهنده (server)، به سوالات زیر پاسخ دهید:

الف) چرا در برنامه UDPServer.py از حلقه بی‌پایان while True: استفاده شده؟

ب) چه اطلاعاتی در بسته UDP به عنوان آدرس مقصد قرار می‌گیرد؟

ج) در برنامه TCPClient.py چرا از دستور close() برای سوکت استفاده می‌شود؟

د) در برنامه TCPClient.py، متد connect() چه نقشی دارد؟

ه) چرا در برنامه سرویس‌دهنده TCP از متد accept() استفاده می‌شود؟

و) فرض کنید ابتدا TCPClient را اجرا کنید و سپس TCPServer را اجرا کنید. چه اتفاقی می‌افتد؟ چرا؟

ز) فرض کنید ابتدا UDPClient را اجرا کنید و سپس UDPServer را اجرا کنید. چه اتفاقی می‌افتد؟ چرا؟

پاسخ:

الف) در برنامه UDPServer.py، از حلقه بی‌پایان while True: برای این استفاده می‌شود که سرویس‌دهنده بتواند به طور مداوم منتظر دریافت پیام‌ها از سرویس‌گیرنده‌ها باشد و پس از دریافت هر پیام، آن را پردازش کرده و پاسخ دهد. بدون این حلقه، سرویس‌دهنده پس از اولین دریافت بسته، متوقف می‌شود.

ب) آدرس مقصد در بسته UDP شامل دو قسمت اصلی است:

آدرس IP میزبان مقصد: این آدرس مشخص می‌کند که بسته باید به کدام میزبان ارسال شود.

شماره پورت سوکت مقصد: این شماره پورت مشخص می‌کند که بسته به کدام فرآیند (سوکت) در میزبان مقصد ارسال شود.

ج) دستور close() برای بستن سوکت پس از اتمام ارتباط استفاده می‌شود. پس از ارسال پیام به سرویس‌دهنده و دریافت پاسخ، سرویس‌گیرنده نیازی به استفاده از سوکت ندارد، بنابراین سوکت بسته می‌شود تا منابع سیستم آزاد شوند و فرآیند به پایان برسد.

د) متد connect() در برنامه TCPClient.py برای برقراری اتصال TCP بین سرویس‌گیرنده و سرویس‌دهنده استفاده می‌شود. با استفاده از این متد، سرویس‌گیرنده آدرس سرویس‌دهنده (آدرس IP و شماره پورت) را به سرویس‌دهنده اعلام می‌کند و سه مرحله دست دادن انجام می‌شود تا اتصال TCP برقرار شود.

ه) متد accept()در برنامه سرویس‌دهنده پس از آنکه سرویس‌گیرنده به سرویس‌دهنده متصل شد، یک سوکت جدید برای ارتباط با آن سرویس‌گیرنده ایجاد می‌کند. این سوکت جدید (که به آن Connection Socket گفته می‌شود) به طور اختصاصی برای تعامل با سرویس‌گیرنده مورد استفاده قرار می‌گیرد.

و) اگر ابتدا TCPClient را اجرا کنید، به محض اجرا سرویس‌گیرنده تلاش خواهد کرد تا یک اتصال TCP با سرویس‌دهنده و سرویسی که وجود ندارد برقرار کند. بنابراین در این حالت که سرویس‌دهنده TCP در حال اجرا نباشد ، اتصال TCP برقرار نخواهد شد.

ز) UDPClient نیازی به برقراری اتصال TCP با سرویس‌دهنده ندارد زیرا سرویس‌گیرنده بلافاصله پس از اجرا اتصال برقرار نمی‌کند. بنابراین، اگر ابتدا UDPClient را اجرا کنید و بعد از آن UDPServer را اجرا کرده و بعد مقداری ورودی از طریق صفحه‌کلید وارد کنید، همه چیز به‌درستی کار خواهد کرد.

سوال ۸:

الف) Multiplexing و Demultiplexing در لایه انتقال چه نقشی دارند و چرا وجود شماره پورت (port number) برای پیاده‌سازی آن‌ها ضروری است؟

ب) فرض کنید در یک میزان، چندین فرایند مختلف بر بستر UDP فعالیت می‌کنند و هرکدام شماره پورت خاص خود را دارند. اگر این میزبان تنها یک کارت شبکه فیزیکی داشته باشد، توضیح دهید لایه انتقال چگونه بسته‌های دریافتی را به فرایند مربوطه تحویل می‌دهد.

ج) در چه شرایطی ممکن است پورت‌های تصادفی (ephemeral ports) با محدودیت جدی مواجه شوند و آیا این موضوع می‌تواند خللی در عملکرد سرویس‌دهی یا امنیت سیستم ایجاد کند؟ توضیح دهید.

پاسخ:

الف) Multiplexing و Demultiplexing به ترتیب به معنای ترکیب چند جریان داده برای ارسال (به اشتراک‌گذاری سرویس) در مبدأ و جداسازی این جریان‌ها در مقصد است.

در Multiplexing، لایه انتقال در فرستنده، داده‌های چندین فرایند (process) مختلف را دریافت کرده، برای هر کدام یک سرآیند انتقال مناسب (شامل شماره پورت مبدا و مقصد) اضافه می‌کند و همه را به لایه شبکه تحویل می‌دهد

در Demultiplexing، لایه انتقال در گیرنده، بسته‌های دریافتی را بررسی کرده و با توجه به شماره پورت مقصد، بسته را به فرایند صحیح در میزبان تحویل می‌دهد. بنابراین، شماره پورت مانند یک شناسه برای هر فرایند است و بدون آن، لایه انتقال نمی‌داند که بسته باید به کدام برنامه یا سرویس تحویل داده شود.

ب) حتی اگر میزبان فقط یک کارت شبکه فیزیکی داشته باشد، لایه‌های بالاتر از فیزیکی و شبکه (یعنی لایه انتقال) می‌توانند بسته‌ها را بر اساس سرآیند UDP تفکیک کنند.

فرآیند به این صورت انجام می‌شود:

۱– کارت شبکه، بسته را در لایه فیزیکی و سپس لایه شبکه مثلاً (IP) دریافت می‌کند.

۲– لایه شبکه بررسی می‌کند که بسته باید به این میزبان تحویل داده شود.(بر اساس آدرسIP)

۳– لایه انتقال (UDP) پس از دریافت بسته، شماره پورت مقصد را از سرآیند UDP می‌خواند.

۴– با استفاده از این شماره پورت، بسته را به فرایند مشخص‌شده که به آن پورت گوش می‌دهد (listening) ارسال می‌کند.

ج) پورت‌های تصادفی (ephemeral ports) معمولاً برای ایجاد ارتباط‌های موقتی در سمت سرویس‌گیرنده استفاده می‌شوند، مثلاً هنگام ایجاد یک ارتباطTCP از سرویس‌گیرنده به سرویس‌دهنده.

شرایطی که در آن پورت‌های تصادفی ممکن است با محدودیت مواجه شوند:

۱– **بار زیاد ارتباطات همزمان**: اگر سرویس‌گیرنده تعداد زیادی اتصال همزمان برقرار کند (مثلاً هزاران اتصال به سرویس‌دهنده‌های مختلف یا حتی یک سرویس‌دهنده)، ممکن است کل مجموعه پورت‌های تصادفی موجود (معمولاً در محدوده‌ی ۴۹۱۵۲ تا ۶۵۵۳۵) پر شود.

۲– **نشت پورت(Port Leak) :** اگر سیستم اتصال‌های قبلی را درست آزاد نکند یا برنامه‌ها پورت‌ها را باز نگه دارند، پورت‌ها به سرعت مصرف می‌شوند.

۳– **Firewall یا NAT محدودکننده**: در برخی محیط‌ها، NAT یا فایروال ممکن است محدوده خاصی از پورت‌های تصادفی را باز بگذارد. اگر تعداد اتصالات بیشتر از این محدوده شود، اتصال‌های جدید مسدود می‌شوند.

تأثیر بر عملکرد یا امنیت: عدم دسترسی به پورت‌های آزاد می‌تواند باعث شکست اتصال‌های جدید شود. این موضوع به‌ویژه در سرویس‌دهنده‌هایی با تعداد بالای کاربران یا سرویس‌های real-time مانند VoIP بحرانی است .حملات DoS می‌توانند با پر کردن همه پورت‌های موقتی، سیستم را از ایجاد ارتباط‌های جدید باز دارند .همچنین محدود بودن تعداد پورت‌ها می‌تواند الگوریتم‌های تخصیص را قابل پیش‌بینی کند و زمینه‌ای برای حملات مانند port hijacking فراهم کند.

سوال ۹:

الف) در دیتاگرام زیر موارد a تا j را نام ببرید.

ب) مقدار checksum را برای دیتاگرام زیر محاسبه کنید.

<table>
<tr><td colspan="4">153.18.8.105 (a)</td><td rowspan="2">(i)</td></tr>
<tr><td colspan="4">171.2.14.10 (b)</td></tr>
<tr><td colspan="2">All 0s</td><td>17 (c)</td><td>12 (d)</td><td rowspan="3"></td></tr>
<tr><td colspan="2">1087 (e)</td><td>13 (f)</td></tr>
<tr><td colspan="2">15 (g)</td><td>All 0s (h)</td><td rowspan="2">(j)</td></tr>
<tr><td>A</td><td>U</td><td>T</td><td>!</td><td>(k)</td></tr>
</table>

پاسخ:

الف)

a) 32-bit source IP address
b) 32-bit destination IP address
c)8-bit protocol
d) 16-bit UDP total length
e) 16-bit Source port address
f) 16-bit destination port address
g) 16-bit UDP total length
h) 16-bit Checksum
i) pseudo header
j) the UDP header
k) data

ب)

برای محاسبه checksum ابتدا تمام محتوی را تبدیل به مبنای باینری می‌کنیم:

10011001	00010010	00001000	01101001
10101011	00000010	00001110	00001010
00000000	00010001	00000000	00001100
00000100	00111111	00000000	00001101
00000000	00001111	00000000	00000000
01000001	01010101	01010100	00100001

سپس به صورت اعداد ۱۶بیتی این مقادیر را تقسیم می‌کنیم:

1001100100010010	0000100001101001
1010101100000010	0000111000001010
0000000000010001	0000000000001100
0000010000111111	0000000000001101
0000000000001111	0000000000000000
0100000101010101	0101010000100001

می‌توانیم برای راحت شدن محاسبات، آن‌ها را به شکل هگزادسیمال بازنویسی کنیم:

9912	0869
AB02	0E0A
0011	000C
043F	000D
000F	0000
4155	5421

سپس اعداد را به ترتیب باهم جمع می‌کنیم و در صورت به وجود آمدن رقم carry، آن را به اول حاصل جمع اضافه می‌کنیم:

$9912 + 0869 = A17B$
$A17B + AB02 = 14C7D\ (carry) \rightarrow 4C7D + 1 = 4C7E$
$4C7E + 0E0A = 5A88$
$5A88 + 0011 = 5A99$
$5A99 + 000C = 5AA5$
$5AA5 + 043F = 5EE4$
$5EF1 + 000F = 5F00$
$5F00 + 4155 = A055$
$A055 + 5421 = F476$

مکمل اول (1's complement) حاصل جمع را محاسبه می‌کنیم و checksum نهایی را بدست می‌آوریم:

$F476\ (1111010001110110)\ complement \rightarrow \mathbf{0B89}(\mathbf{0000101110001001})$

سوال ۱۰:

الف) هر سه پروتکل IP، TCP و UDP در صورتی که بسته‌ای با خطا در checksum دریافت کنند، بدون اطلاع به فرستنده آن را دور می‌ریزند. دلیل اینکار را توضیح دهید.

ب) چرا در محاسبه checksum از مکمل یک حاصل جمع استفاده می‌شود و از همان حاصل جمع استفاده نمی‌شود؟ اگر از مکمل یک استفاده نشود چه اتفاقی می‌افتد؟

د) آیا امکان دارد خطای یک بیتی وجود داشته باشد که توسط checksum تشخیص داده نشود؟ خطای دو بیتی چطور؟ مثال بزنید.

پاسخ:

الف) پروتکل IP، UDP و TCP با دریافت بسته دارای خطای Checksum، فقط تشخیص می‌دهند که بسته دریافتی دارای خطا است ولی نمی‌توانند تشخیص دهند که کدام یک از فیلدهای اطلاعاتی سرآیند دارای خطا است، در نتیجه قادر به پردازش بسته دارای خطا برای انجام وظایف نخواهند بود و مجبور هستند که بسته دارای خطا را دور بریزند. پروتکل TCP با تشخیص از بین رفتن بسته (Packet Loss) از طریق مکانیزم Timeout یا دریافت سه ACK تکراری، بسته از بین رفته را مجدداً ارسال می‌کند. در مورد UDP اگر لایه کاربرد تحمل‌پذیری خطا (Data Loss) را نداشته باشد، خود لایه کاربرد می‌بایست با استفاده از یک روش کنترل خطا، تصحیح خطا را انجام دهد.

ب) برای تشخیص خطا، گیرنده تمام کلمات ۱۶بیتی (شامل checksum) را جمع می‌کند، اگر نتیجه تماما ارقام یک نباشد، گیرنده متوجه می‌شوند که بسته خطا دارد. این خاصیت تماما یک بودن از مکمل یک بودن checksum نتیجه می‌شود.

د) تمام خطاهای ۱بیتی قابل تشخیص هستند. اما خطاهای ۲بیتی می‌توانند تشخیص داده نشوند. برای مثال: اگر سومین رقم کلمه اول به یک تبدیل شود و سومین رقم کلمه دوم به صفر تبدیل شود:

```
  1 0 0 1          1 0 1 1
+ 1 0 1 1        + 1 0 0 1
---------        ---------
  0 1 0 0          0 1 0 0
+       1        +       1
---------        ---------
  0 1 1 0          0 1 1 0
  1 0 0 1          1 0 0 1
```