به نام خدا

پروژه‌ی سوم شبکه

پروتکل TCP

زمستان و بهار ۹۷-۹۶

تحویل: جمعه ۷ اردیبهشت، ساعت ۲۳:۵۵

هدف پروژه

در این پروژه به قصد آشنایی بیشتر با پروتکل TCP Reno (و تا حدودی TCP Tahoe) می‌خواهیم قسمت‌هایی از آن را پیاده‌سازی کنیم. برای ارسال بسته‌ها به شکل خام از پیاده‌سازی پروتکل غیرمطمئن[1] و بدون اتصال[2] UDP در زبان جاوا استفاده می‌کنیم.

ارسال داده‌ها به شکل خام

برای ارسال داده‌ها به شکل خام از پیاده‌سازی پروتکل UDP در زبان جاوا استفاده می‌کنیم. البته همانطور که می‌دانید پروتکل UDP خود دارای checksum است و نیازی به بررسی درستی داده‌های یک بسته توسط شما نیست. در واقع در این پروتکل هیچ تضمینی برای ارسال یک بسته وجود ندارد ولی در صورت دریافت در سمت گیرنده، داده‌ها سالم هستند (اگر از Packet Injection صرف نظر کنیم).

---

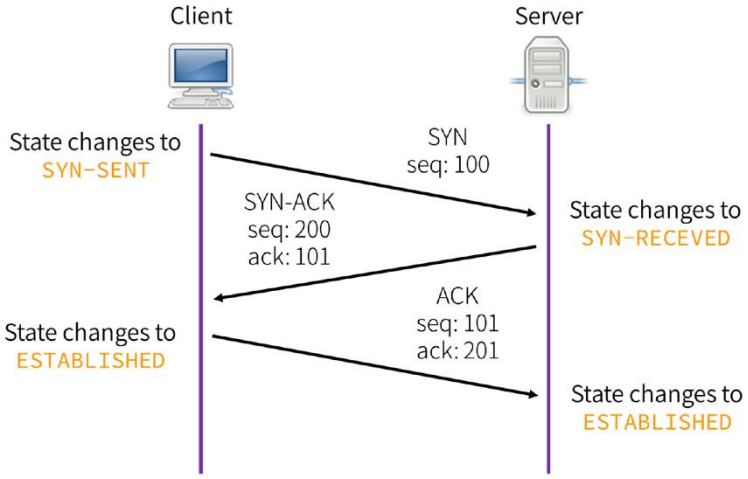
[1] Unreliable
[2] Connectionless

پروتکل TCP

در این پروژه ویژگی‌های زیر از پروتکل TCP مدنظر هستند:

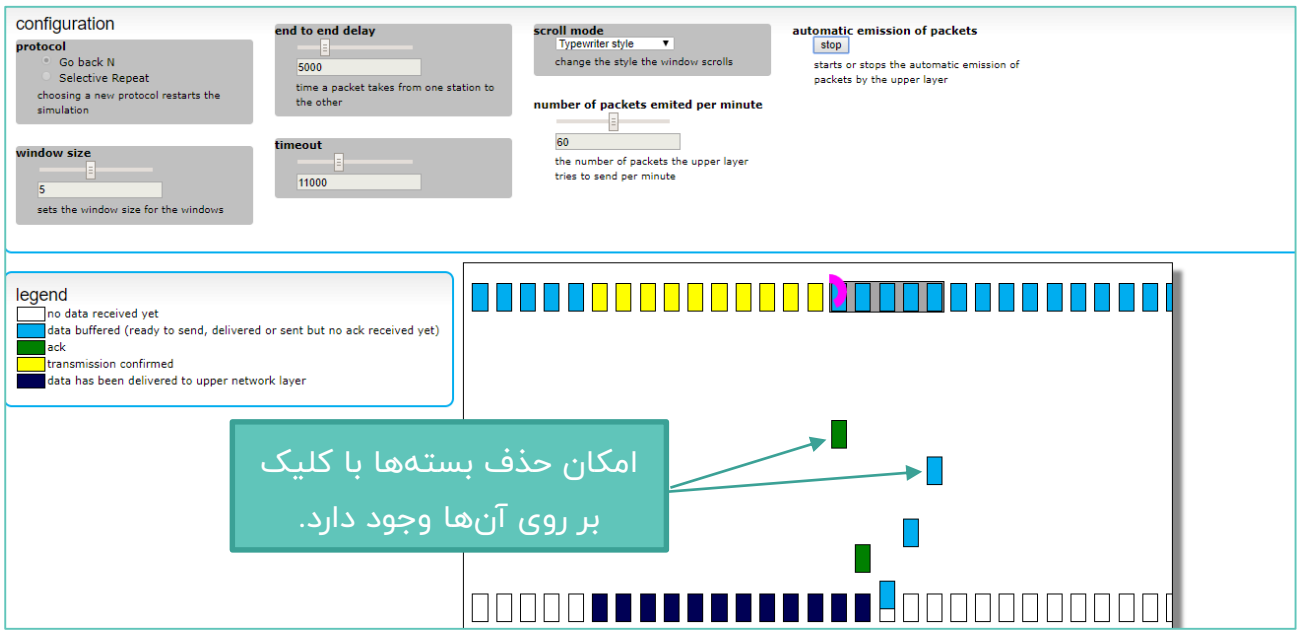
۱. شروع اتصال[3] (3-Way Handshake)

برای شروع اتصال باید فرآیند SYN → SYN-ACK → ACK توسط کلاینت و سرور طی شود:



۲. ارسال مطمئن و پایپ‌لاین داده‌ها[4] توسط پروتکل Go-Back-N

در صورت عدم دریافت بسته‌ها در سمت گیرنده، باید بر اساس قواعد موجود در پروتکل Go-Back-N بسته‌ها دوباره فرستاده شوند. همچنین بر اساس همین پروتکل امکان ارسال بسته‌ها بدون دریافت تصدیق[5] بسته‌های قبلی وجود دارد. شدیداً پیشنهاد می‌شود قبل از پیاده‌سازی این لینک را مشاهده کنید:

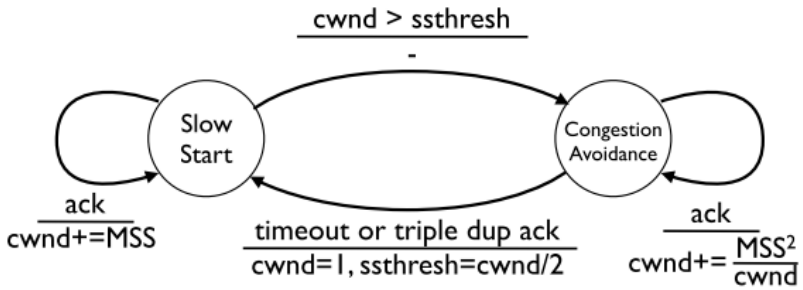


---

[3] Connection
[4] Pipelined Reliable Data Transfer
[5] Acknowledge

۳. کنترل ازدحام[6]

به وسیله‌ی طول پنجره‌ی مورد استفاده برای پایپ‌لاین داده می‌توان نرخ ارسال و ازدحام شبکه را در یک اتصال کنترل کرد. در این قسمت به جهت سهولت، به جای پیاده‌سازی کنترل ازدحام TCP Reno، کنترل ازدحام TCP Tahoe را پیاده‌سازی می‌کنیم که ماشین حالت آن به شکل زیر است:



جزئیات ماشین حالت بالا در این لینک قابل مشاهده است (فایل PDF آن نیز در کنار صورت پروژه قرار داده شده‌است).

برخی از ویژگی‌هایی که شما **نباید** پیاده‌سازی کنید:

۱. تسهیم[7] (امکان اتصال چندین کلاینت به یک سرور)

۲. پروتکل اتمام اتصال (فرآیند FIN → ACK)

استفاده از کلاس EnhancedDatagramSocket

برای شبیه‌سازی برخی از ویژگی‌های لینک و انجام بررسی‌هایی توسط دستیاران یک کلاس با نام EnhancedDatagramSocket به شما داده شده‌است که در واقع از کلاس DatagramSocket ارث‌بری کرده‌است؛ در نتیجه از تمام متدهای این کلاس از جمله receive، send، close و ... پشتیبانی می‌کند. در این پروژه شما باید از این کلاس برای ارسال پیام‌ها استفاده کنید و با تغییر پارامترهای آن می‌توانید برخی از ویژگی‌های لینک را نیز شبیه‌سازی کنید.

---

[6] Congestion Control
[7] Multiplexing/Demultiplexing

کلاس‌های نهایی

در این پروژه شما باید کلاس‌های زیر را توسط ارث‌بری[8] توسعه[9] دهید. یک نمونه پیاده‌سازی‌های این کلاس‌ها (کلاس‌های TCPServerSocketImpl و TCPSocketImpl) به همراه نحوه‌ی استفاده از آن‌ها (کلاس‌های Sender و Receiver) در اختیار شما قرار داده شده‌است.

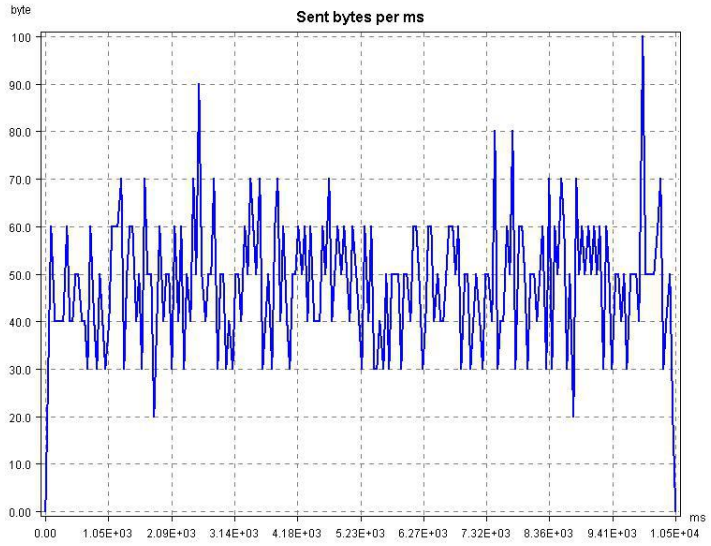
کلاس TCPServerSocket			
متد	ورودی	خروجی	توضیحات
constructor	port:int	-	روی پورت مشخص‌شده یک سوکت دیتاگرام باز می‌کند.
accept	void	TCPSocket	به صورت Blocking منتظر اتصال یک کلاینت می‌ماند و در صورت اتصال، یک شیء از نوع TCPSocket برای ارتباط با آن کلاینت باز می‌گرداند. (می‌توانید فرض کنید که هیچگاه دو کلاینت به یک TCPServerSocket متصل نمی‌شوند)
close	void	void	باید سوکت دیتاگرام مربوطه بسته شود. (پروتکل بسته‌شدن اتصال TCP را **نباید** پیاده‌سازی کنید)

کلاس TCPSocket			
متد	ورودی	خروجی	توضیحات
constructor	ip:String port:int	-	شروع به برقراری اتصال به سوکت سرور که بر روی آی‌پی و پورت ورودی مشخص شده‌است می‌کند.
send	pathToFile:String	void	فایلی که آدرس آن در ورودی دریافت شده‌است را به مقصد می‌فرستد. پس از ارسال کامل فایل اجرای این متد به اتمام می‌رسد.
receive	pathToFile:String	void	داده‌های دریافتی را بر روی فایلی که آدرس آن در ورودی نوشته شده‌است می‌نویسد و پس از اتمام دریافت کامل آن اجرای این متد به اتمام می‌رسد.
getWindowSize	void	long	طول پنجره‌ی کنونی را بر می‌گرداند.

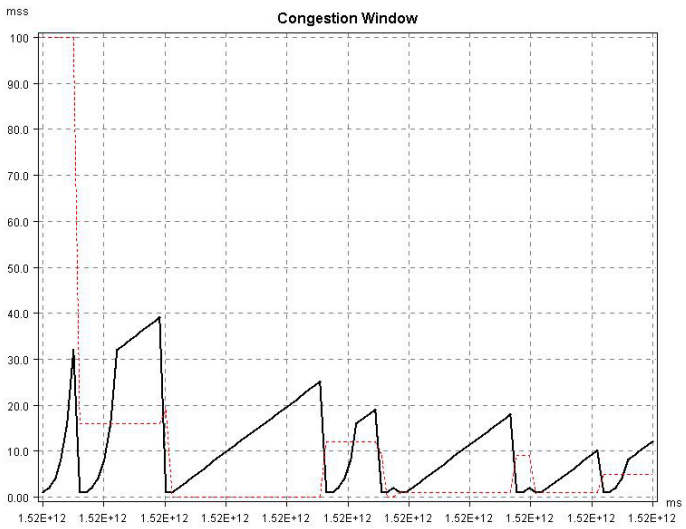
[8] Inheritance
[9] Extend

getSSThreshold	void	long	مقدار آستانه‌ی حالت Slow Start را بر می‌گرداند.
onWindowChange	void	void	این متد هنگام تغییر طول پنجره یا آستانه‌ی حالت Slow Start **باید** فراخوانی شود. در واقع این کار باعث رسم نمودار طول پنجره و آستانه‌ی Slow Start نسبت به زمان می‌شود.
close	void	void	باید سوکت دیتاگرام مربوطه بسته شود. (پروتکل بسته‌شدن اتصال TCP را **نباید** پیاده‌سازی کنید)

در کدهای قرار داده شده دو نمودار به طور خودکار همراه برنامه‌ی شما رسم می‌شوند. اولین نمودار تعداد بایت‌های ارسال شده توسط برنامه‌ی شما در واحد زمان است:



دومین نمودار طول پنجره (رنگ مشکی) و آستانه‌ی حالت Slow Start (رنگ قرمز) بر حسب زمان را مشخص می‌کند. برای رسم این نمودار کافی است متدهای getWindowSize و getSSThreshold را پیاده‌سازی کنید و در هنگام تغییر دو متغیر مربوطه متد onWindowChange را فراخوانی کنید:



گزارش کار

در گزارش کار کافی است مواردی که پیاده‌سازی کرده‌اید را مشخص کنید. همچنین در هر مورد کلاس‌ها و متدهای درگیر به همراه ساختمان داده‌ی استفاده شده را به طور خلاصه بیان کنید.

برای مثال:

- **ارسال داده به صورت پایپ‌لاین توسط پروتکل Go-Back-N:** کلاس‌های ReceivingWindow و SendingWindow. استفاده از ساختمان داده‌ی ArrayList و BlockingQueue جاوا.
- **پیاده‌سازی 3-Way Handshake:** کلاس‌های TCPServerSocket، TCPSocket، ReceivingSegment و SendingSegment.

نکات تکمیلی

- در صورت هر گونه کپی‌برداری (به صورت جزئی یا کلی) برای هر دو طرف نمره‌ی ۱۰۰- در نظر گرفته می‌شود. در صورتی که از کدهای گروه‌های سال‌های پیش نیز استفاده شود برای شخص کپی کننده نمره‌ی ۱۰۰- در نظر گرفته می‌شود. توجه نمایید که همه‌ی کدها (با کدهای امسال و سال‌های قبل) توسط ابزار Moss بررسی می‌شوند.
- پروژه دو نفره است و نمره‌ی افراد لزوماً یکی نیست.
- برای ارسال داده‌ها تحت شبکه تنها امکان استفاده از کلاس EnhancedDatagramSocket وجود دارد و استفاده از کلاس‌های دیگر همانند DatagramSocket، Socket، ServerSocket و ... تقلب محسوب می‌شود.
- توجه نمایید که Payload بسته‌های UDP که توسط EnhancedDatagramSocket فرستاده می‌شوند نباید از ۱۴۰۸ بایت بیشتر شود.
- برای این پروژه تنها می‌توانید از زبان برنامه‌نویسی جاوا استفاده کنید.
- دستیاران آموزشی با تغییر پارامترهای کلاس EnhancedDatagramSocket و بررسی صحت فایل دریافت شده در سمت گیرنده درستی پیاده‌سازی شما را بررسی می‌کنند.
- تمام مقدارهای اولیه‌ی متغیرها همانند SSThreshold، CongestionWindowSize و ... توسط خود شما انتخاب می‌شوند و مناسب است به صورت تجربی بهترین آن‌ها را برای بالا بردن کارایی انتخاب نمایید.