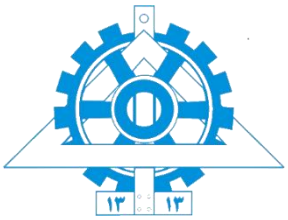
به نام خدای آزادی



تمرین برنامه نویسی شماره ۰۴

عنوان: پروژه برنامه نویسی- شماره ۰۴

درس: شبکه‌های کامپیوتری

استاد راهنما: دکتر ناصر یزدانی[۱]

رشته: مهندسی کامپیوتر

دستیاران آموزشی: اسامه ایراندوست[۲]، محمدرضا ولی[۳]

نیمسال دوم سال تحصیلی ۰۲-۱۴۰۱

[۱] نشانی پست الکترونیکی: yazdani@ut.ac.ir

[۲] نشانی پست الکترونیکی: osameh.irandoust@ut.ac.ir

[۳] نشانی پست الکترونیکی: mvali@ut.ac.ir

فهرست مطالب

عنوان پروژه (پیاده سازی و تحلیل الگوریتم‌های کنترل ازدحام)

در این پروژه شما به پیاده سازی و تحلیل سه الگوریتم کنترل ازدحام Reno، New Reno و BBR می‌پردازید.

۱- مقدمه

در این تمرین می‌خواهیم سه الگوریتم کنترل ازدحام را برای TCP پیاده سازی کنیم؛ TCP یک پروتکل لایه انتقال است که به صورت گسترده مورد استفاده قرار می‌گیرد و مشخصه مهم آن، توانایی برقراری ارتباط قابل اطمینان است.

هدف از انجام این تمرین، دستیابی به درک عمیق‌تر از نحوه برقراری ارتباط TCP و ساز و کارهای مختلفی است که الگوریتم‌های کنترل ازدحام در هنگام مواجهه با ازدحام به کار می‌گیرند.

۲- پیاده سازی الگوریتم Reno

TCP Reno از سه مکانیزم برای شناسایی و واکنش به ازدحام استفاده می‌کند:

۱. Slow start: TCP ابتدا نرخ ارسال خود را به صورت نمایی افزایش می‌دهد تا به آستانه مشخص CWND برسد و پس از رسیدن به حد CWND به حالت Congestion avoidance تغییر فاز می‌دهد.
۲. Congestion avoidance: TCP نرخ ارسال خود را به صورت خطی افزایش می‌دهد و با هر RTT موفق، مقدار CWND را به مقدار ثابتی افزایش می‌دهد؛ اگر Timeout رخ دهد یا ازدحام تشخیص داده شود، به فاز Slow start تغییر فاز می‌دهد.
۳. Fast recovery: اگر بسته‌ای Lost شود، TCP وارد این فاز می‌شود. در این فاز، TCP نرخ ارسال خود را کاهش می‌دهد، CWND را نصف می‌کند و از مکانیزم Fast retransmit برای بازیابی بسته Lost شده بدون انتظار برای Timeout استفاده می‌کند.

۱-۲- تعریف کلاس TCPConnection

کلاسی به نام TCPConnection تعریف کنید که نشان دهنده TCP Connection است. این کلاس باید دارای متغیرهایی private برای ذخیره CWND فعلی، ssthresh و RTT فعلی باشد و مقادیر اولیه به صورت public تعریف می‌شوند:

```
class TCPConnection {
private:
    int cwnd;       // Congestion window
    int ssthresh;   // Slow start threshold
    int rtt;        // Round-trip time

public:
    // Constructor
    TCPConnection() {
        cwnd = 1;       // Initial congestion window size
        ssthresh = 65535;   // Initial slow start threshold
        rtt = 0;        // Initial round-trip time
    }
```

۲-۲- تعریف متد SendData

متدی با عنوان SendData در کلاس TCPConnection تعریف کنید که ارسال داده را روی شبکه شبیه سازی می‌کند. این متد برای محاسبه نرخ ارسال، از مقدار CWND فعلی و ssthresh استفاده می‌کند.

۲-۳- تعریف متد onPacketLoss

متدی با عنوان onPacketLoss در کلاس TCPConnection تعریف کنید که شناسایی بسته گمشده را شبیه سازی می‌کند.

This method should trigger the appropriate congestion control mechanism, such as reducing the cwnd and ssthresh, and entering fast recovery mode.

۲-۴- تعریف متد onRTTUpdate

متدی با عنوان onRTTUpdate در کلاس TCPConnection تعریف کنید که دریافت آپدیت در RTT جاری را شبیه سازی می‌کند.

This method should update the RTT and adjust the cwnd accordingly, based on the current congestion control mode (slow start, congestion avoidance, or fast recovery).

۵-۲- تعریف تابع اصلی

تابعی را تعریف کنید که یک محیط شبکه را شبیه سازی می‌کند. در این تابع، چند نمونه از کلاس TCPConnection را برای نمایش اتصالات مختلف TCP ایجاد کنید و ارسال داده و دریافت آپدیت RTT را برای هر اتصال شبیه سازی کنید.

همچنین شرایطی را شبیه سازی کنید که بسته ای Lost شده و با استفاده از متد onPacketLoss مکانیزم مناسب را فعال کنید.

```
int main() {
    // Instantiate multiple instances of TCPConnection class to represent different TCP connections
    TCPConnection tcpConnection1(10, 5); // Initial cwnd = 10, ssthresh = 5
    TCPConnection tcpConnection2(5, 2);  // Initial cwnd = 5, ssthresh = 2

    // Simulate sending data and receiving updates on RTT for each connection
    tcpConnection1.sendData();
    tcpConnection1.onRTTUpdate(50); // Update RTT to 50 ms

    tcpConnection2.sendData();
    tcpConnection2.onRTTUpdate(100); // Update RTT to 100 ms

    // Simulate packet loss events and trigger congestion control mechanisms
    tcpConnection1.onPacketLoss();
    // This will trigger fast recovery mode in TCPConnection1 and update its cwnd and ssthresh accordingly

    tcpConnection2.onPacketLoss();
    // This will trigger slow start mode in TCPConnection2 and update its cwnd and ssthresh accordingly

    // Print the updated cwnd and ssthresh values for each connection
    std::cout << "TCPConnection1: cwnd = " << tcpConnection1.getCwnd() << ", ssthresh = " << tcpConnection1.getSsthresh() << std::endl;
    std::cout << "TCPConnection2: cwnd = " << tcpConnection2.getCwnd() << ", ssthresh = " << tcpConnection2.getSsthresh() << std::endl;

    return 0;
}
```

توجه کنید که شبه کد بالا صرفا برای درک بهتر نحوه پیاده سازی این تابع اضافه شده است و لازم است شما با خلاقیت خوتان، سناریوهای متفاوتی را در این تابع پیاده سازی کنید.

۶-۲- تست و اجرا

برنامه خود را با سناریوهای مختلف نظیر مقادیر مختلف اولیه برای CWND و ssthresh و نرخ های متفاوت Packet loss تست کنید و الگوریتم پیاده سازی شده را از نظر Performance تحلیل کنید. همچنین لازم است اطلاعات هر TCP Connection شامل CWND، RTT و مکانیزم کنترل ازدحام نمایش داده شوند.

۳- پیاده سازی الگوریتم New Reno

حال که الگوریتم Reno را پیاده سازی کردیم، به سراغ پیاده سازی نسخه جدیدتر این الگوریتم یعنی New Reno می‌رویم.

مراحل پیاده سازی New Reno تا حد بسیار زیادی با مراحل پیاده سازی Reno همپوشانی دارند؛ در ادامه این تمرین با اعمال تغییرات لازم در مراحل بالا، الگوریتم New Reno را پیاده سازی کنید، سپس مرحله 6-2 را برای این الگوریتم تکرار کنید.

راهنمایی:

برای پیاده سازی New Reno لازم است متد onSelectiveAck در کلاس TCPConnection اضافه شود.

This method should update the SACK (Selective Acknowledge) information and adjust the cwnd accordingly, based on the current congestion control mode (slow start, congestion avoidance, or fast recovery with SACK).

۴- پیاده سازی الگوریتم BBR

حال به سراغ الگوریتم سوم یعنی BBR می‌رویم؛ این الگوریتم نسبت به دو الگوریتم ذکر شده جدیدتر است و توسط شرکت Google توسعه داده شده است.

هر چند استفاده از Reno و New Reno در شبکه های امروزی رایج‌تر است ولی روز به روز بر محبوبیت الگوریتم BBR افزوده می‌شود.

فلسفه الگوریتم BBR تا حدی با دو الگوریتم ذکر شده متفاوت است؛ در ادامه این تمرین با اعمال تغییرات لازم در مراحل بالا، الگوریتم BBR را پیاده سازی کنید، سپس مرحله 6-2 را برای این الگوریتم تکرار کنید.

۵- سوالات

اکنون که این سه الگوریتم را پیاده سازی کردید و به درک عمیق تری از مفاهیم هر یک از آن‌ها پی بردید به سوالات زیر پاسخ دهید:

- تفاوت کنترل ازدحام و کنترل جریان را صورت خلاصه بیان کنید.
- الگوریتم New Reno را به صورت خلاصه شرح دهید.
- الگوریتم BBR را به صورت خلاصه شرح دهید.
- تفاوت این سه الگوریتم را به صورت خلاصه بیان کنید.
- بر اساس خروجی های مرحله ۶ پیاده سازی برای هر الگوریتم، در شرایط یکسان، این سه الگوریتم را تحلیل و با هم مقایسه کنید. کدام الگوریتم بهترین و کدام بدترین کارایی را داشت؟ آیا به صورت کلی می توان گفت یکی از این سه الگوریتم همیشه بر دو الگوریتم دیگر برتری دارد؟ دلیل خود را با بررسی حالت های مختلف، تغییر معیار های مختلف و با ذکر مقادیر خروجی کد مربوطه تحلیل کنید.
- تعدادی از الگوریتم های کنترل ازدحام را که در این تمرین به آن‌ها اشاره‌ای نشده به صورت مختصر بررسی کنید؛ یکی از این الگوریتم ها را که از نظر شما حداقل در یک معیار بر الگوریتم های Reno، New Reno و BBR برتری دارد را نام ببرید و دلیل خود را توجیه کنید.

۶- جمع بندی و نکات پایانی

- مهلت تحویل: **۲۶ خرداد**
- پروژه در گروه‌های ۲ نفره انجام می‌شود. (گروه بندی در سامانه ایلرن نیز انجام می‌شود و تحویل تمرین به صورت گروهی خواهد بود)
- هر ۲ نفر می بایست کار را تقسیم کنند. همچنین از Git برای ساختن branch و تقسیم issue ها استفاده نمایید. (با استفاده از commit ها و تعیین issue ها میزان مشارکت هر نفر مشخص می‌شود). بعد از انجام این کار کدها را در یک repository به نام CN_CHomeworks_4 در اکانت‌های GitHub/GitLab خود قرار دهید(به صورت private). همچنین در یک فایل README.md می‌توانید report و داکیومنت خود را کامل کنید و در کنار repository قرار دهید. در نهایت لینک این repository را در محل پاسخ تمرین قرار دهید. (از فرستادن فایل به صورت زیپ جدا خودداری نمایید.) اکانت تی ای‌های این تمرین رو به Repo خودتون به عنوان Maintainer به پروژه اضافه کنید.

Github ID: UT-CNs02

- برای پیاده سازی این تمرین از C یا C++ استفاده کنید.
- سیستم عامل ترجیحا Linux باشد.
- دقت کنید گزارش نهایی شما می‌بایست همانند یک Document باشد و شامل توضیح کد و ساختار کد، همچنین نتیجه نهایی اجرای کد، اسکرین شات‌های دقیق از تمام مراحل و پاسخ سوالات باشد. (در فایل Readme.md کنار فایل اصلی خود و در Repo مربوطه قرار دهید.) این نکته حائز اهمیت است که فایل PDF به هیچ عنوان مورد پذیرش قرار نخواهد گرفت.
- ساختار صحیح و تمیزی کد برنامه، بخشی از نمره‌ی این پروژه شما خواهد بود. بنابراین در طراحی ساختار برنامه دقت به خرج دهید.
- برای هر قسمت کد، گزارش دقیق و شفاف بنویسید. کدهای ضمیمه شده بدون گزارش مربوطه نمره‌ای نخواهند داشت.
- هدف این تمرین یادگیری شماست. لطفا تمرین را خودتان انجام دهید. در صورت مشاهده‌ی مشابهت بین کدهای دو گروه، مطابقت سیاست درس با گروه متقلب و تقلب دهنده برخورد خواهد شد.
- سؤالات خود را تا حد ممکن در فروم درس مطرح کنید تا سایر دانشجویان نیز از پاسخ آن بهره‌مند شوند. در صورتی که قصد مطرح کردن سؤال خاص‌تری دارید، از طریق ایمیل زیر ارتباط برقرار کنید. توجه داشته باشید که سایر شبکه‌های اجتماعی راه ارتباطی رسمی با دستیاران آموزشی نیست و دستیاران آموزشی مؤظف به پاسخگویی در محیط‌های غیررسمی نیستند.
  - mvali@ut.ac.ir

موفق باشید