Proyecto: Simulación de control de congestión de TCP en un Enlace Compartido

29 de agosto de 2025

Resumen

Este documento describe los parámetros de una actividad práctica para comparar el rendimiento de los algoritmos de control de congestión TCP: Reno, CUBIC y BBR. La actividad consiste en implementar una simulación en Python para modelar el comportamiento de múltiples clientes que compiten por un ancho de banda limitado en un enlace compartido. El objetivo es analizar cómo cada algoritmo gestiona la ventana de congestión en presencia de pérdidas causadas por la saturación del búfer del enlace.

Índice

1. Introducción **2**

2. Escenario de Simulación en Python **2**
2.1. Parámetros de la Simulación 2

3. Actividad **2**
3.1. Parte 1: Implementación de la Simulación 2
3.2. Parte 2: Análisis Comparativo 3
3.3. Parte 3: Propuesta de Optimización del throughput 3

4. Entregables **3**

1. Introducción

El control de congestión es un mecanismo fundamental en la capa de transporte de TCP, diseñado para que un emisor tome acciones ante eventos que indiquen degradación en la calidad del enlace. A lo largo de los años, se han desarrollado diversos algoritmos para optimizar este proceso. En este proyecto, se implementará una simulación en Python para estudiar el comportamiento de tres algoritmos: Reno, CUBIC y BBR. A diferencia de los simuladores de eventos discretos, esta actividad se centra en una implementación simplificada para resaltar el funcionamiento básico de los algoritmos en un escenario de contención por un recurso compartido.

2. Escenario de Simulación en Python

La simulación se construirá en un entorno de Python, utilizando los siguientes componentes para emular una red simple (se sugiere que sean modelados como clases):

- **Enlace compartido:** Representará el enlace de red compartido. Este componente tendrá un búfer (cola) de tamaño limitado y una capacidad de procesamiento fija. Si el búfer se llena, cualquier paquete adicional será descartado.
- **Cliente:** Representará cada uno de los flujos de datos. Cada cliente estará asociado a un algoritmo de control de congestión específico y gestionará su propia ventana de congestión (`cwnd`), RTT y temporizadores de paquetes.
- **Algoritmos de Congestión:** Se implementarán las lógicas de Reno, CUBIC y BBR. Estos algoritmos modificarán la `cwnd` del cliente basándose en los eventos de la simulación, como la recepción de `ACKs` y la detección de pérdidas.

2.1. Parámetros de la Simulación

- **Capacidad del Enlace:** 10 y 100 paquetes por segundo (p/s).
- **Tamaño del Búfer del Enlace:** 10 y 30 paquetes.
- **Número de Clientes:** 9 clientes en total: 3 con TCP Reno, 3 con TCP CUBIC y 3 con TCP BBR.
- **Flujo de Datos:** Cada cliente intentará enviar un flujo continuo de paquetes durante toda la simulación.
- **Tiempo de Simulación:** 100 segundos.

3. Actividad

3.1. Parte 1: Implementación de la Simulación

1. Implemente el enlace compartido con los parámetros de capacidad y tamaño de búfer especificados.

2. Implemente el código para los algoritmos `TCPReno`, `TCPCUBIC` y `TCPBBR`. Cada algoritmo debe gestionar la lógica de su `cwnd`.

3. Escriba el bucle principal de la simulación, que debe:

   - Simular el envío de paquetes de cada cliente al `enlace compartido` (hasta su `cwnd`).
   - Procesar los paquetes en el `enlace compartido` y detectar si alguno se pierde por desbordamiento del búfer.
   - Simular el retardo de la red (RTT) y la recepción de `ACKs`.
   - Actualizar la `cwnd` de cada cliente basándose en los eventos (ACK o pérdida).

3.2. Parte 2: Análisis Comparativo

1. **Medición de Métricas:** Durante la simulación, registre el *throughput* instantáneo (paquetes por segundo) y el RTT para cada uno de los 9 clientes en cada paso de la simulación.

2. **Visualización y Análisis:**

   - **Gráfico de Throughput:** Genere un gráfico de líneas que muestre el *throughput* de cada cliente a lo largo del tiempo. Analice el comportamiento: ¿Qué algoritmo es más agresivo? ¿Se observa una mayor equidad en el reparto del ancho de banda entre los clientes?
   - **Gráfico de RTT:** Genere un gráfico que muestre el RTT de cada cliente. ¿Cómo se relaciona el RTT con la saturación del búfer del enlace? ¿Se comporta de manera diferente el RTT de BBR en comparación con Reno y CUBIC?

3.3. Parte 3: Propuesta de Optimización del throughput

Crea una nueva técnica, por ejemplo, TCPCustom, que implemente tu propia estrategia de control de congestión. Esta clase debe definir cómo se ajusta la cwnd basándose en las métricas de la simulación (RTT, pérdidas, etc.). Compara el rendimiento de tu técnica propuesta con los otros 3 algoritmos. Escoja un escenario de comparación.

4. Entregables

- Código fuente de la simulación en Python.
- Un informe en formato PDF que debe ser sustentado
  - Los gráficos de *throughput* y RTT generados.
  - Un análisis detallado de los resultados observados.
  - La descripción de la estrategia de optimización propuesta.