Pontifícia Universidade Católica do Rio Grande do Sul
Escola Politécnica
Laboratório de Redes

Primeiro Trabalho de Laboratório de Redes
Cliente/Servidor com sockets TCP e avaliação de controle de congestionamento

Objetivo

Implemente um mini-serviço de **transferência confiável de arquivos,** com **listagem remota de arquivos e atendimento concorrente**. O objetivo é gerar tráfego TCP realista para que você **analise o comportamento dos mecanismos do protocolo TCP**.

Este trabalho tem por objetivo:

- o desenvolvimento de uma aplicação cliente/servidor transportada sobre **TCP/IPv4**;
- a criação de um **protocolo de aplicação** para o serviço;
- tratar **leituras/escritas** e erros;
- implementar **concorrência** no servidor;
- coletar métricas TCP;
- montar um **ambiente de experimentos** com modelagem de link (tc/netem).

Descrição

1) Módulos Cliente/Servidor

A aplicação a ser desenvolvida deve ter os seguintes módulos:

- **Cliente**: responsável por interpretar os comandos do usuário, executar a comunicação com o servidor e exibir informações da aplicação ao usuário;
- **Servidor**: módulo gerenciador da aplicação, que recebe e armazena arquivos enviados pelos clientes de forma concorrente, ou seja, pode atender mais de um cliente por vez.

2) Protocolo de aplicação

O protocolo de aplicação deve prever a formação de um frame contendo informações de controle e dados da aplicação. Assim sugere-se que:

- **Cada mensagem** contenha um campo informando a operação, um campo com o tamanho do *payload* de dados transportado no pacote e um campo com o *payload* de dados;
- Sejam suportadas as seguintes **operações:**
    - LIST → servidor retorna lista de arquivos;

    - PUT <nome_arquivo> → upload do arquivo do cliente para o servidor;
    - QUIT → encerra sessão.
- Seja implementado o **tratamento de erros** para o envio de arquivos e armazenamento de arquivos com o mesmo nome.
- O **servidor** implemente as seguintes ações:
    - **Escuta** em <IP>:<PORTA>, aceitando N clientes simultâneos (concorrência obrigatória);
    - **Armazena** arquivos em um diretório.

3) Interface do Cliente

A interface do cliente deve ser em linha de comando e deve também:

- Suportar os seguintes comandos:
    - list
    - put <arquivo>
    - quit

- Suportar as seguintes opções, para que o servidor possa ser especificado:
    - Host: identificação do IP do servidor
    - Port: identificação da porta da aplicação

4) Instrumentação para análise

Para análise do desempenho das transmissões, devem ser armazenados alguns dados, no cliente. Então:

- **Crie** um *log* por conexão: *timestamps de início e fim da conexão*, *bytes* enviados/recebidos, duração da conexão, taxa aproximada de bytes por segundo que foram transmitidos na conexão.
- **Utilize** o Wireshark para captura e análise do tráfego. Gere Salve os arquivos com a monitoração para posterior análise.
- **Opcional para ambiente Linux: Colete**, no cliente, informações da opção *tcp_info* via *getsockopt(TCP_INFO)* e registre RTT, cwnd, ssthresh, retransmissões. Em geral, a função *getsockopt* recupera o valor de uma opção associada a um *socket*, e no caso da opção TCP_INFO, recupera informações específicas da pilha TCP, o que a torna útil para qualquer situação onde seja preciso entender a saúde e as métricas de uma conexão TCP. Deve ser utilizada, enquanto o socket está ativo.

5) Ambiente de experimentos

A topologia mínima necessária para o trabalho é:

- Cliente e servidor (máquinas físicas, VMs, WSL2 ou containers Docker).
- Bottleneck**:** usar tc/netem para limitar banda, latência e perda numa interface (requer root). Tc/netem são opções para ambiente Linux.

Exemplo (Linux) para criar gargalo de 10 Mbit/s, 50 ms e 0,1% de perda:

```
# substitua IFACE pela interface (ex.: eth0)
sudo tc qdisc add dev IFACE root handle 1: htb default 10
sudo tc class add dev IFACE parent 1: classid 1:10 htb rate
10mbit ceil 10mbit
sudo tc qdisc add dev IFACE parent 1:10 handle 10: netem delay
50ms loss 0.1%

# Remover:
# sudo tc qdisc del dev IFACE root
```

6) Cenários de teste

Execute os cenários propostos, colete as informações solicitadas anteriormente, e compare os resultados obtidos nos seguintes cenários:

1. **Cenário 1 -** execute uma instância da aplicação cliente sem qualquer alteração da interface de rede. Colete as informações para enviar um arquivo de 200 MB.
2. **Cenário 2 -** execute de duas a quatro instâncias de aplicação cliente, de forma concorrente, sem qualquer alteração da interface de rede. Colete as informações para enviar um arquivo de 200 MB.
3. **Cenário 3 -** repita as operações previstas em a) com alterações de interface de rede do cliente, uma de cada vez e colete os resultados:
    i. Configure a interface de rede da máquina para incluir perda de pacotes
    ii. Configurar a interface de rede da máquina para incluir latência variável.
4. **Cenário 4 –** repita as operações previstas em b) com alterações de interface de rede do cliente, uma de cada vez e colete os resultados:
    i. Configure a interface de rede da máquina para incluir perda de pacotes
    ii. Configurar a interface de rede da máquina para incluir latência variável.

Observações:

1) No Linux: Utilize o módulo *netem* para fazer alterações no funcionamento da interface de rede. Com esse módulo é possível adicionar atrasos, perda, duplicação, corrupção, reordenação de pacotes, entre outros. O funcionamento do netem está descrito no site: https://srtlab.github.io/srt-cookbook/how-to-articles/using-netem-to-emulate-networks.html ou https://netbeez.net/blog/how-to-use-the-linux-traffic-control/
2) No Windows: Baixe e configure o clumsy no endereço: https://github.com/jagt/clumsy.
3) No Mac: Baixe o Network Link Conditioner. Consulte esta página: https://nshipster.com/network-link-conditioner/

7) Roteiro para análise

a) Utilize o Wireshark, aplicando os seguintes filtros, e analise o resultado:
   1. As retransmissões ocorridas: *tcp.analysis.retransmission*
   2. As retransmissões por timeout:
      tcp.analysis.retransmission && !tcp.analysis.fast_retransmission
   3. As retransmissões por por fast retransmit (3 dupACKs):
      tcp.analysis.fast_retransmission
   4. Os ACKs duplicados: *tcp.analysis.duplicate_ack*
   5. Os segmentos considerados perdidos: *tcp.analysis.lost_segment*

b) Utilize o Wireshark, e gere os seguintes gráficos para análise:
   1. Time-Sequence Graph (tcptrace)
      i. Menu: *Statistics → TCP Stream Graphs → Time-Sequence (tcptrace)*
      ii. Permite observar o crescimento da janela (slow start e congestion avoidance) e quedas após eventos de perda.
   2. I/O Graph com tcp.analysis.bytes_in_flight
      i. Menu: Statistics → I/O Graphs
      ii. Configure o eixo Y como MAX e o campo como tcp.analysis.bytes_in_flight.
      iii. resultado mostra a evolução da janela de congestionamento e suas quedas após perdas.

Observações:

1) Adicione a coluna **"Delta time displayed"** para observar o intervalo entre eventos. Timeouts normalmente apresentam uma pausa maior antes do reenvio.

2) Também é útil adicionar a coluna **"TCP Bytes in Flight"** (*Preferences → Columns → Add → Field name: tcp.analysis.bytes_in_flight*). Quedas abruptas neste valor indicam redução da janela de congestionamento.

8) Resultado esperado

Deve ser apresentado no relatório:

- Os dados coletados no cliente e no servidor, analisando e comparando o desempenho obtido nos 4 cenários.
- Capturas de tela com o filtro aplicado e os eventos destacados.
- Gráfico de bytes_in_flight mostrando a queda após a perda.
- Comentário breve explicando:
  - O tipo de perda (fast retransmit ou timeout)
  - A reação do TCP (queda de cwnd/ssthresh)
  - O processo de recuperação (crescimento linear ou slow start)

Resultados e Entrega

Grupos: Até 3 componentes.

Data Entrega e apresentação:
Obs.: Todos os participantes devem estar presentes

Entrega final no Moodle:
- Relatório da aplicação implementada e da análise de tráfego.
- Código fonte comentado.

IMPORTANTE: Não serão aceitos trabalhos entregues fora do prazo. Trabalhos que não compilam ou que não executam não serão avaliados. Todos os trabalhos serão analisados e comparados. Caso seja identificada cópia de trabalhos, todos os trabalhos envolvidos receberão nota ZERO.