Laboratório de Redes de Computadores - Aula 05 - Atividade sobre protocolos de transporte

Descrição

O objetivo dessa atividade é praticar o uso de sockets TCP e analisar seu funcionamento através da análise dos pacotes usando Wireshark. Descreva a interpretação dos resultados.

Exercícios

1. Obtenha o código fonte de exemplo de sockets TCP disponível no Moodle, descompacte o pacote e compile. Juntamente com os exemplos, são disponibilizados *makefiles* para automatizar o processo de compilação.

   ```
   $ tar -zxvf simplesocket_c.tar.gz
   $ cd simplesocket/tcp
   $ make
   ```

2. Abra dois terminais no Linux. Em um deles execute o programa servidor e no outro o programa cliente. Faça alguns testes para entender o seu funcionamento. Em um segundo momento, utilize duas máquinas diferentes ao invés de apenas terminais na mesma máquina.

3. Abra um terceiro terminal e execute uma nova cópia do programa servidor. O programa cliente deve ser capaz se comunicar com os dois servidores, para isso utilize portas diferentes para cada servidor. Faça também um teste utilizando a mesma porta nos dois servidores para observar o resultado.

4. Execute o Wireshark para monitorar o trafego TCP gerado pelo programa. IIdentifique os pacotes TCP que estão sendo enviados para cada um dos servidores. Identifique cada uma das seguintes fases da execução do programa:

   a) Estabelecimento da conexão;

   b) Cliente envia mensagem ao servidor;

   c) Servidor envia a mensagem de volta ao cliente; e

   d) Fechamento da conexão.

   Em especial, atente para as flags e números de sequência e acknowledgement dos segmentos TCP. Dica: utilize a ferramenta "Flow Graph" do Wireshark disponível no menu "Statistics → Flow Graph" e selecione a opção "TCP flow".

5. Modifique os programas utilizados nas atividades anteriores de forma que o servidor envie uma mensagem de resposta ao cliente para cada mensagem recebida.

6. Modifique os programas utilizados nas atividades anteriores e crie um protocolo simples que permita finalizar o programa servidor remotamente. Utilize para a isso a palavra FIM. Isto é, quando o cliente enviar a palavra FIM, o servidor deve identificar que essa é uma palavra reservada e parar de receber mensagens de clientes. Caso contrário, o servidor deve enviar a mensagem de volta normalmente.

7. Obtenha o código fonte de exemplo de sockets com timeout disponível no Moodle, descompacte o pacote e compile. Existem duas versões, uma em linguagem C (*timeout_udp_c.tar.gz*) e outra em Java (*timeout_udp_java.tar.gz*), juntamente com *makefiles* para automatizar o processo de compilação. Como o mecanismo de timeout pode ser útil em uma aplicação que utiliza sockets?

```
$ tar -zxvf timeout_udp_c.tar.gz
$ cd timeout_udp_c
$ make
```

8. Obtenha o código fonte de exemplo de sockets com timeout disponível no Moodle, descompacte o pacote e compile. Existem três versões, uma em linguagem C (*serialization_udp_c.tar.gz*) e outras duas em Java (*serialization_udp_java.tar.gz* e *serialization_tcp_java.tar.gz*), juntamente com *makefiles* para automatizar o processo de compilação. Analise o processo de serialização de dados sobre sockets, e como esse mecanismo pode ser utilizado para troca de mensagens em nível de aplicação.

```
$ tar -zxvf serialization_udp_c.tar.gz
$ cd serialization_udp_c
$ make
```