Atividade 5 - Implementação de Aplicação Cliente/Servidor com Sockets TCP

Gabriel Hidasy Rezende

October 4, 2015

Questão 1:

cliente.c

As funções interessantes da parte de redes são:

- `int socket(int domain, int type, int protocol)`

Requisita um socket para o sistema operacional, retorna um descritor de arquivo (o socket nada mais é do que um arquivo em sistemas unix), retorna o socket ou -1 em caso de erro

- `int connect(int sockfd, const struct sockaddr *addr,socklen_t addrlen)`

Usa a system call connect para, como o nome diz, conectar o socket a um endereço (especificado na estrutura sockaddr). No caso desse programa (que usa sockets do tipo stream) isso significa realmente realizar uma conexão TCP, realizando o three way handshake com o servidor especificado em sockaddr. Se o socket fosse do tipo DGRAM ela iria especificar para que endereço enviar (e de que endereço aceitar receber) datagramas. retorna 0 em caso de sucesso e -1 em caso de erro

- `int read(int sockfd, char *data, int buffer_size)`

Lê dados do socket e os armazena em `=data=`, o valor de retorno indica quantos bytes foram lidos do socket especificado.

- `int inet_pton(int af, const char *src, void *dst)`

Essa função converte uma string de caracteres (src) em uma estrutura do tipo especificado em af (AF_{INET} ou AF_{INET6}), e copia esse endereço para dst Retorna 1 em caso de sucesso, 0 se src não contem um endereço do tipo especificado em af, e -1 se af for de um tipo não suportado

- `uint16_t htons(uint16_t hostname)`

Essa função converte um unsigned short int de host-byte-order para network-byte-order, preparando o endereço para ser usado na rede

0.1 servidor.c

- `int bind(int sockfd, struct sockaddr * servaddr, int size)`

Liga um socket a uma porta (especificada em servaddr.sin_{port}), é usada no servidor para permitir que ele ouça em uma porta especifica por clientes.

- `int listen(int sockfd, int backlog)`

Marca um socket como um socket como passivo, o que significa que ele será usado para receber conexões, backlog diz o tamanho da fila de conexões pendentes que será aceito. Retorna 0 em caso de sucesso, e -1 em caso de erro

- `int accept(int sockfd, struct sockaddr * servaddr, int size)`

Aceita conexões, retorna um file descriptor para um socket ligado ao cliente. A thread que roda essa chamada fica bloqueada até que um cliente se conecte.

- `uint32_t htonl(uint32_t hostname)`

Faz o mesmo que a htons, mas com inteiros de 32 bits São simples conversores de big-ending-little-ending

Questão 2:

A função bind no servidor retorna um erro pois ela está tentando ligar o socket a porta 13, que é uma porta baixa (por padrão apenas o super usuário pode usar bind em portas abaixo de 1024), mudando o servidor para usar uma porta alta (ex. 12345) resolve o problema. Obviamente nesse caso precisamos modificar o cliente para que ele se conecte na porta correta. Feitas as modificações e executando ambos os comandos na mesma maquina, e passando o endereço local para o cliente (./cliente 127.0.0.1), temos como saída a data e hora de hoje (Mon Sep 28 22:20:53 2015). O servidor não imprime nada

Questão 3

Executando em maquinas diferentes tive o mesmo resultado, só precisei usar o ip da outra maquina (./cliente 177.220.85.187). (Na outra maquina pude user a porta 13)

Questão 4

Usando o tcpdump chequei o trafego passando pelo meu notebook, mais especificamente com o comando `sudo tcpdump -A | grep Mon`, e a cada requisição do cliente o tcpdump registrava mais um pacote com a data de hoje sendo enviada. Também é possivel usar telnet ou netcat como cliente (ambas as ferramentas não geram a saida com a data por si só, o que significa que a string de data tem que ter vindo do servidor)

Questão 5

No código

Questão 6

No código, após as modificações e rodando em duas maquinas diferentes tive a saída que segue:

```
./servidor
Remote socket 192.168.1.212

./cliente 192.168.1.147
local IP is 0.0.0.0
Sun Oct  4 20:21:49 2015
```

A resposta com IP local zerada era esperada, pois getsockename retorna com que IP o socket está ligado (bound), e um socket de cliente em geral não está bound a nenhum IP especifico (0.0.0.0 significa "Qualquer endereço" - `INADDR_ANY`)

Questão 7:

A maquina servidor em estado `TIME_WAIT`, que é o ultimo estado em que a maquina que iniciou a desconexão fica, isso acontece pois o servidor é a

primeira maquina a fechar a conexão (`close(connfd)`), em geral não é uma boa ideia deixar o servidor em `TIME_WAIT` pois ele é mais sujeito (por ter vários clientes) a ficar sem recursos com múltiplas conexões nesse estado.

Questão 8

Sim, telnet pode ser usado para no lugar do cliente, basta rodar o comando telnet especificando a porta correta (no meu caso 12344).

```
telnet 192.168.1.147 12344
Trying 192.168.1.212...
Connected to 192.168.1.212.
Escape character is '^]'.
Mon Oct  5 00:00:38 2015
Connection closed by foreign host.

./servidor
Remote socket 192.168.1.125
```

Para o servidor não faz diferença qual o cliente usado.

Uma modificação no servidor que impediria o uso de comando telnet seria modificar o mesmo para codificar as mensagens de alguma forma (por exemplo, transmitindo os dados em Base 64, ou comprimir os dados de alguma forma, ou cifrar eles, ou no caso do protocolo usado (que envia datas) enviar apenas o epoch e deixar a tradução para uma data por conta do cliente. Todas as modificações exigiriam uma mudança adequada do lado do cliente (telnet ou netcat ainda conseguiriam receber a string do servidor, mas a saída seria incompreensível)