MAC0352 - Redes de Computadores e Sistemas Distribuídos

Daniel Macêdo Batista

IME - USP, 29 de Abril de 2021

Roteiro

Observações importantes

Cliente (daytime)

Servidor (daytime)

Observações importantes

Características do TCP

- Orientado a conexão
  - Cada conexão é identificada por um par de sockets: IP origem, porta origem, IP destino, porta destino

Características do TCP

- A conexão é estabelecida nos dois sentidos
  - É possível tanto o cliente quanto o servidor terem `write` e `read`
  - Os `write` e `read` precisam estar sincronizados

Características do TCP

- O programador não precisa se preocupar em tomar ações caso um segmento não chegue do outro lado ou chegue duplicado
  - Mas pode ser interessante fazer algumas verificações com relação a segurança
  - Ainda assim problemas podem acontecer. Por exemplo, se a rede cair e a temporização do segmento estourar pode ser interessante reenviar a mensagem

Segurança

- Preocupação não só com TCP
- Evitar usar funções que não especifiquem o tamanho dos buffers que armazenarão mensagens recebidas da outra ponta
- Limitar as mensagens que podem ser recebidas

Cliente (daytime)

Passo 0

- Ler a RFC
  - Porta bem definida (daytime $\rightarrow$ 13)
  - Aprender as sequências de reads e writes
  - Descobrir quem fecha a conexão: cliente ou servidor

Passo 1

- Definir quantos sockets serão necessários
- Declarar um inteiro para cada socket (no caso do daytime, 1 único socket)

```
int sockfd
```

Passo 2

- Definir quantas áreas de memória serão necessárias para armazenar informações enviadas e recebidas
- Declarar um buffer para cada área de memória (no caso do daytime, apenas um buffer para informações recebidas)

```
char recvline[MAXLINE + 1]
```

Passo 3

- Declarar uma estrutura de endereçamento para cada socket (no caso do daytime, apenas uma)

  `struct sockaddr_in servaddr`

Passo 4

- Criar cada socket e associar aos descritores do passo 1 (no caso do daytime, há apenas um)
- É preciso definir se o socket vai ser para a Internet e que vai ser TCP. Isso é feito com o `AF_INET` e o `SOCK_STREAM` respectivamente

```
sockfd = socket(AF_INET, SOCK_STREAM, 0)
```

Passo 5

- Preparar a estrutura de endereçamento de cada socket criado no passo 4 (no caso do daytime, há apenas um)
- Lembrar de transformar os endereços de porta e de IP por conta das diferenças na representação interna das máquinas (little endian X big endian)

```
bzero(&servaddr, sizeof(servaddr))
servaddr.sin_family = AF_INET
servaddr.sin_port = htons(13)
inet_pton(AF_INET, argv[1],
&servaddr.sin_addr)
```

Passo 6

- Fazer a conexão no servidor associando o socket com a estrutura de endereço (no caso do daytime, apenas uma conexão)

```
  connect(sockfd, (struct sockaddr *)
&servaddr, sizeof(servaddr))
```

Passo 7

- Implementar o protocolo da camada de aplicação (no caso do daytime, apenas um laço com read)

```
read(sockfd, recvline, MAXLINE)
```

Servidor (daytime)

Passo 1

- Definir quantos sockets são necessários, lembrando que agora há necessidade de no mínimo 2. Um para escutar e um específico para o cliente conectado
- Declarar um inteiro para cada socket (no caso do daytime, 2)

```
int listenfd, connfd
```

Passo 2 e Passo 3

Observações importantes

Cliente (daytime)

Servidor (daytime)

- Áreas de memória e estrutura de endereço similar ao feito no cliente

Passo 4

- Criação do socket similar ao feito no cliente com a diferença de que agora isso deve ser feito para o socket que vai ficar escutando (no caso do daytime, apenas um)

```
listenfd = socket(AF_INET, SOCK_STREAM, 0)
```

Passo 5

- Preparação da estrutura de endereçamento similar ao feito no cliente com a diferença de que agora precisa definir em qual endereço vai escutar (no caso do daytime, vamos escutar em todos os endereços da máquina)
- `servaddr.sin_addr.s_addr = htonl(INADDR_ANY)`

Passo 6

- Informar que o socket vai escutar na porta definida (no caso do daytime, na porta 13 e há apenas 1 socket)

```
bind(listenfd, (struct sockaddr *) &servaddr, sizeof(servaddr))
```

Passo 7

- Informar que esse socket é de um servidor e esperar conexões (no caso do daytime, apenas 1 socket)

  ```
  listen(listenfd, LISTENQ)
  ```

Passo 8

- Aceitar as conexões dos clientes em um laço infinito (no caso do daytime, apenas 1 laço com apenas um comando para aceitar conexões)

```
  connfd = accept(listenfd, (struct sockaddr *)
NULL, NULL)
```

Passo 9

- Implementar o protocolo da camada de aplicação com reads e writes (no caso do daytime, apenas um write e fechar a conexão)

```
write(connfd, buff, strlen(buff))
close(connfd)
```