Cliente-servidor com Sockets TCP

Paulo Sérgio Almeida

Grupo de Sistemas Distribuídos
Departamento de Informática
Universidade do Minho

2006/2007

Comunicação orientada à conexão via TCP/IP

- Servidor fica à espera de ligações.
- Cliente liga-se ao servidor, sendo estabelecida conexão.
- Conexão é um canal fiável para comunicação bidireccional.
- Um socket representa um extremo de uma conexão.
- Uma conexão é caracterizada por um socket-pair.

Características de uma conexão TCP

- Fiabilidade:
  - quando são enviados dados é esperado um acknowledgment;
  - se não vier, os dados são retransmitidos.
- Ordem:
  - os dados são partidos em segmentos numerados;
  - se chegarem fora de ordem são reordenados;
  - se chegar em duplicado é descartado.
- Controlo de fluxo:
  - cada lado diz quantos bytes está disposto a receber: janela;
  - janela representa tamanho de buffer disponível no receptor;
  - pode aumentar ou diminuir conforme dados são lidos pela aplicação ou chegam;
  - se receptor estiver lento, emissor tem que esperar.
- Full-duplex:
  - uma conexão é bidireccional, para enviar e receber dados.

Endereçamento

- Cada máquina (host) tem (pelo menos) um endereço ip.
  Exemplo: 193.136.19.96
- Podem existir vários serviços em cada máquina.
- Serviços de uma máquina são distinguidos por portos.
- Um servidor especifica em que porto escuta.
- Um cliente especifica onde se ligar com um par (ip,porto).

Gamas de portos

- Um porto é um número de 16 bits.
- Serviços internet com números de 0–1023, standardizados pela IANA (Internet Assigned Numbers Authority); e.g.:
  - echo: 7
  - ftp: 21
  - ssh: 22
  - telnet: 23
  - www: 80
- Portos registadas, de 1024–49151; registadas na IANA como conveniência à comunidade.
- Portos dinâmicos, de 49152-65535:
  - são atribuídos a sockets de clientes para distinguir conexões;
  - podem ser usados por servidores que usam um serviço de localização.
- Ver lista em http://www.iana.org/assignments/port-numbers.

Cliente-servidor com TCP – esqueleto

Cliente:

```
socket()
connect()
while ()
  write()
  read()
close()
```

Servidor

```
socket()
bind()
listen()
while ()
  accept()
  while ()
    read()
    write()
  close()
```

Cliente

Etapas de um cliente:

- Criar socket.
- Connect:
  - é especificado o ip e porto do servidor;
  - inicia o three-way handshake para estabelecer conexão com o servidor.
- Troca de dados escrevendo e lendo do connected socket.
- Close: fecha conexão. (Pode ser cliente ou servidor.)

Servidor

Etapas de um servidor

- Criar socket.
- Bind: atribui endereço local ao socket:
  - tipicamente apenas o porto;
  - pode ser especificado qual dos ip locais no caso de uma máquina com vários ip.
- Listen:
  - coloca o socket no modo *listening*, para poder aceitar pedidos de connect a ele dirigidos;
  - permite especificar o backlog: quantas conexões são guardadas pelo kernel antes de serem retiradas pelo accept.
- Accept:
  - devolve *connected socket* da lista de conexões estabelecidas;
  - bloqueia caso ainda não exista conexão estabelecida.
- Troca de dados lendo e escrevendo do connected socket.
- Close: fecha conexão. (Pode ser cliente ou servidor.)

Estabelecimento de conexão

- O servidor tem que estar preparado para aceitar conexões, fazendo socket, bind, listen (um chamado *passive open*).
- Um accept que seja feito bloqueia.
- É efectuado um three-way handshake:
  - O cliente faz um *active open* com connect; é enviado um SYN.
  - O servidor toma nota de conexão a ser estabelecida; faz ACK ao SYN e envia o seu SYN; quando segmento chega, o connect retorna no cliente.
  - O cliente faz ACK ao SYN do servidor; quando segmento chega, a conexão está estabelecida e um accept pode retornar.

Estabelecimento de conexão

- Entre o momento em que uma conexão começa e acaba de ser estabelecida passa um *round-trip time* (RTT);
- Durante este tempo, a conexão está numa fila de conexões a ser estabelecidas.
- Se forem chegando muitos pedidos de conexão, esta fila pode crescer.
- O backlog especifica a soma dos tamanhos das filas de conexões a ser estabelecidas ou já estabelecidas e ainda não obtidas pelo accept.
- É importante que o backlog seja alto para servidores com muita carga em redes com elevada latência.

Terminação de conexão

- São efectuados 4 passos:
  1. Um lado (cliente ou servidor) faz close (*active close*), sendo enviado um FIN.
  2. O outro lado que recebe o FIN faz um *passive close*, sendo enviado um ACK do FIN; é provocado um EOF na leitura.
  3. Mais tarde a aplicação que recebeu o EOF faz close do socket; é enviado um FIN.
  4. O lado que fez o active close faz ACK a este FIN.
- Entre os passos 2 e 3, é possível serem enviados dados para quem fez o active close; neste caso temos um *half-close*.
- Um half-close pode ser obtido com *shutdown*, para permitir a parte activa receber dados.