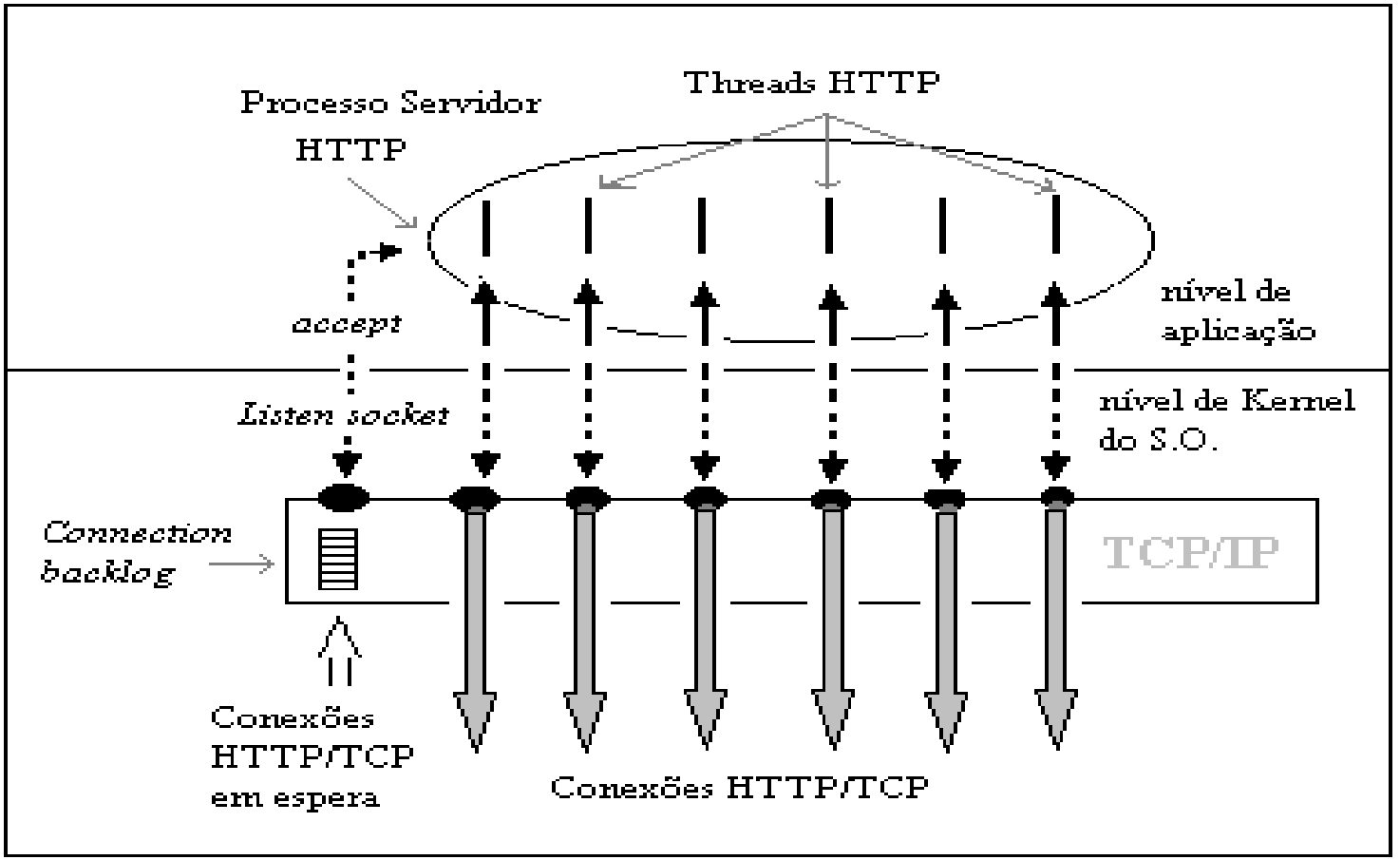
Modelo de servidores de alta performance : caso Web

UNIX Network Programming – Richard Stevens – cap. 27

Servidor Web : funcionamento



Modelo de servidores de alta performance : caso Web

- Forked (*classical UNIX mode*l)
- Pre-forked (NCSA, Apache 1.3.x)
- Multiprocess
- Event-driven (Squid, Zeus,...)
- Single process Multithreads (Alta vista *front end*)

Exemplo Forked

```
#include <...>

//processo filho
void Child(void* arg)
{   char line[100];
    int bytes_read;
    int client = *(int *)arg;

    do
    {
        bytes_read = recv(client, line, sizeof(line), 0);
           if ( bytes_read < 0 )
                 perror("Read socket");
        send(client, line, bytes_read, 0);
    }
    while (strncmp(line, "bye\r", 4) != 0  ||  bytes_read < 0
);
    close(client);
    exit(0);
}
```

Exemplo Forked

```
//processo pai (master)

int main(void)
{   int sd;
...
    if ( (sd = socket(PF_INET, SOCK_STREAM, 0)) < 0 )

    addr.sin_family = AF_INET;
    addr.sin_port = htons(9999);
    addr.sin_addr.s_addr = INADDR_ANY;
    if ( bind(sd, (struct sockaddr*)&addr, sizeof(addr)) != 0)
        ...
    if ( listen(sd, 20) != 0 )
        ...
    while (1)
    {   int client, addr_size = sizeof(addr);
        client = accept(sd, (struct sockaddr*)&addr,
addr_size);
           if ( client < 0 )perror("Accept");
           else
           {printf("Conectado por: %s:%d\n", inet_ntoa
(addr.sin_addr), ntohs(addr.sin_port));
         if ( fork() ) close(client);
              else{ close(sd); Child(&client); exit(0);}
           }
    }
    return 0;
}
```

Modelo de servidores de alta performance : caso Web

- Servidor Pre-forked

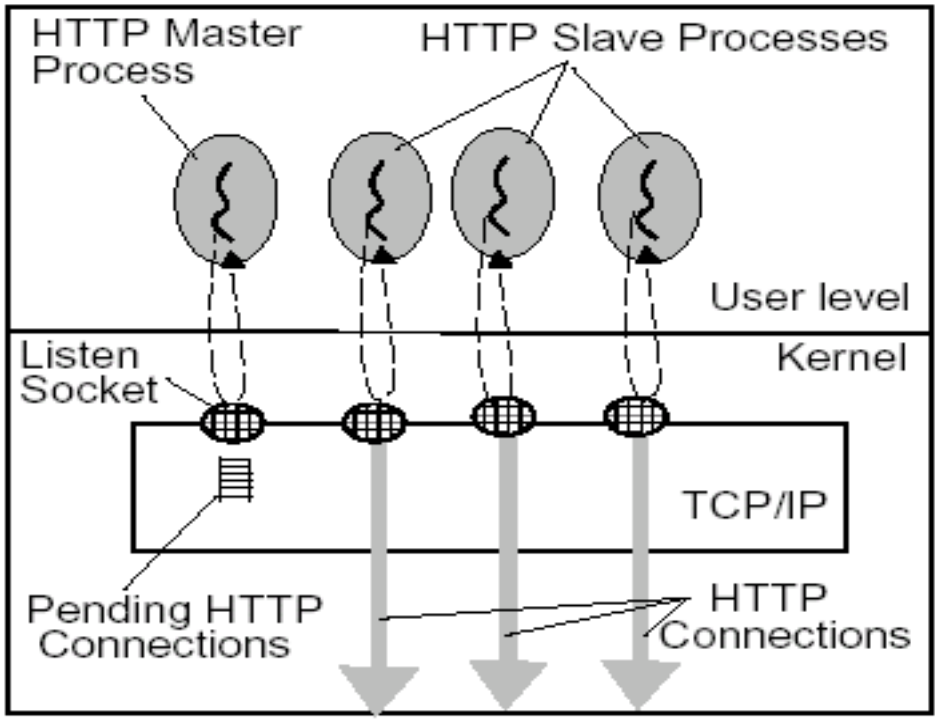


Fig. 1: A process-per connection HTTP server with a master process.

Modelo de servidores de alta performance : caso Web

- Servidor event-driven

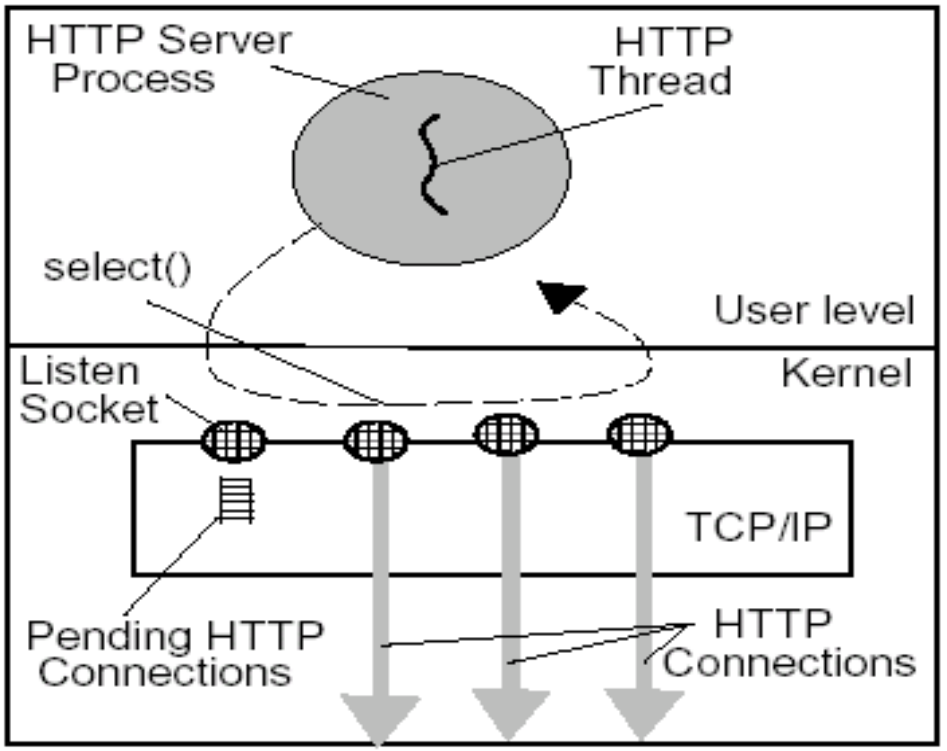


Fig. 2: A single-process event-driven server.

Modelo de servidores de alta performance : caso Web

- Servidor single process multi-threaded

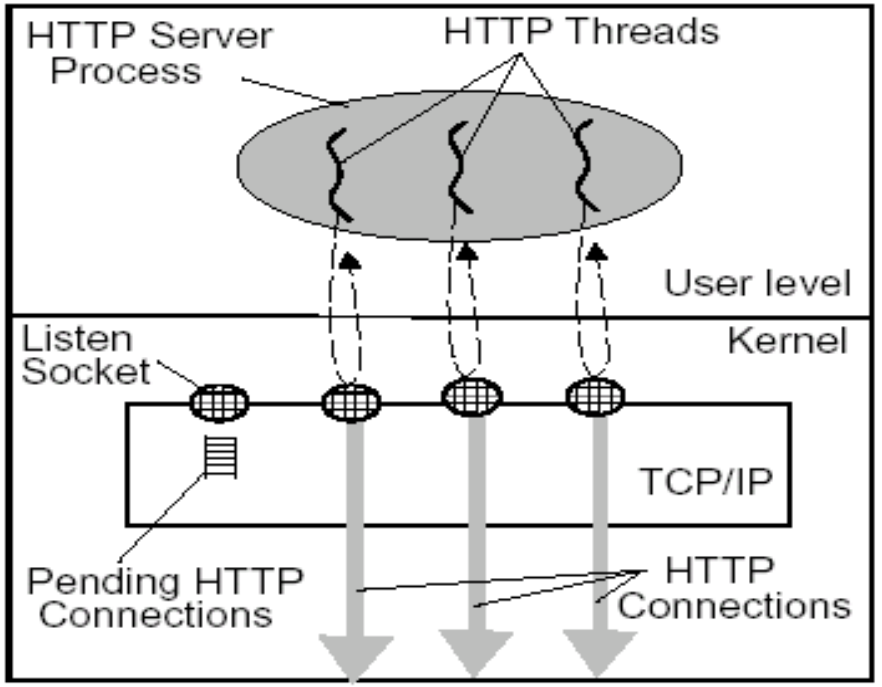


Fig. 3: A single-process multi-threaded server.

Exemplo servidor multithread Java

```
import java.net.*;
public class Server{
public Server( ) {
try {s = new ServerSocket (5155);}
      catch (java.io.IOException e) {System.out.println(e);  System.exit(1);}
try {while (true) {client = s.accept( ); c = new Connection(client); c.start();}}
     catch (java.io.IOException e){System.out.println(e);}}
public static void main(String args[ ] )
 {Server timeOfDayServer = new Server ( );}

private ServerSocket s;
private Socket   client;
private Connection c;
            }
```

Accept → novo *thread*

Exemplo servidor multithread Java

```
import java.net.*;
import java.io.*;
public class Connection extends Thread{
public Connection(Socket s) {outputLine = s;}
public void run() {
try {PrintWriter pout = new
    PrintWriter(outputLine.getOutputStream(),true);
    pout.println("The Date and Time is " + new
    java.util.Date().toString());     outputLine.close();}
catch (java.io.IOException e) {System.out.println(e);}
                }
    private Socket outputLine;}
```

Exercício

- Construir um servidor WEB *multithread*
  - Apenas o método GET precisa ser implementado
  - O servidor deverá:
    - Fazer o parsing para obter request cliente
    - abrir o arquivo (objeto) requisitado pelo cliente fazer a leitura e enviar via *socket*, porta 80
  - OBS:. Não é necessário lidar com requisitos de segurança

Exercício : dimensionamento APACHE 1.3.x (*pre-forked process*)

- Estação servidora:
  - 128 MB RAM
  - 8MB ~ Kernel
- Cada processo servidor
  - ~10MB, sendo 7MB compartilhados com os demais *pre-forked process*
  - Qtde. Máxima de processos servidores = 120-7/3 = 36
- Parâmetros:
  - Max/MinSpareServers → número máximo e mínimo que o Apache deve manter em aberto
  - MaxClient → no. Máximo de clientes que serão servidos a cada momento
  - StartServers → no. máx . De servidores que o Apache deve inicializar
- Valores Otimizado?
  - Memória X custo *fork ( )*
  - Parâmetro de *backlog* ?