Redes de Computadores II
Aula 18
Pulando para a Camada de Aplicação

Prof. Elias P. Duarte Jr.
Universidade Federal do Paraná (UFPR)
Departamento de Informática
www.inf.ufpr.br/elias/redes

Sumário

- Chegamos ao topo: camada de aplicação
- Definição de sistemas cliente-servidor
- Aplicações sobre TCP Vs. UDP
- Concorrência em sistemas cliente-servidor

Sistemas Cliente-Servidor

- Considere dois processos que vão se comunicar usando TCP/IP ou UDP/IP
- Como começam a comunicação?

Sistemas Cliente-Servidor

- Considere dois processos que vão se comunicar usando TCP/IP ou UDP/IP
- Como começam a comunicação?
- Correria?

Sistemas Cliente-Servidor

- Considere dois processos que vão se comunicar usando TCP/IP ou UDP/IP
- Como começam a comunicação?
- Correria?
- Pode até dar certo no mesmo computador ou mesmo ambiente
  - mas e se os processos estão em lados opostos do mundo?

Sistemas Cliente-Servidor

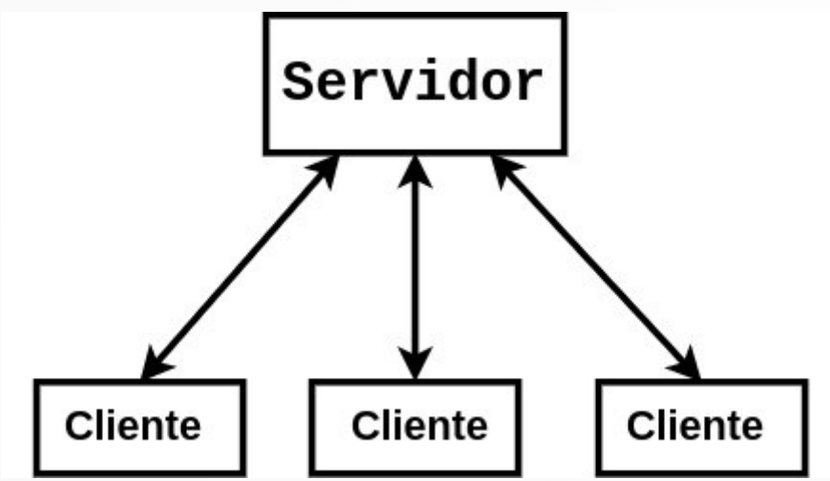
- O modelo Cliente-Servidor resolve exatamente este problema
- Como iniciar a comunicação entre dois processos
- Solução é simples: um dos processos fica já em execução
- Escutando
- Aguardando uma comunicação de outro processo
- Que pode ser iniciada a qualquer momento

Sistemas Cliente-Servidor

- Servidor: o processo que fica em execução, escutando, aguardando a comunicação
- Cliente: o processo que inicia a comunicação, dispara a primeira mensagem para o servidor
- O Cliente oferece uma interface para um usuário (humano ou processo) que é quem demanda a comunicação

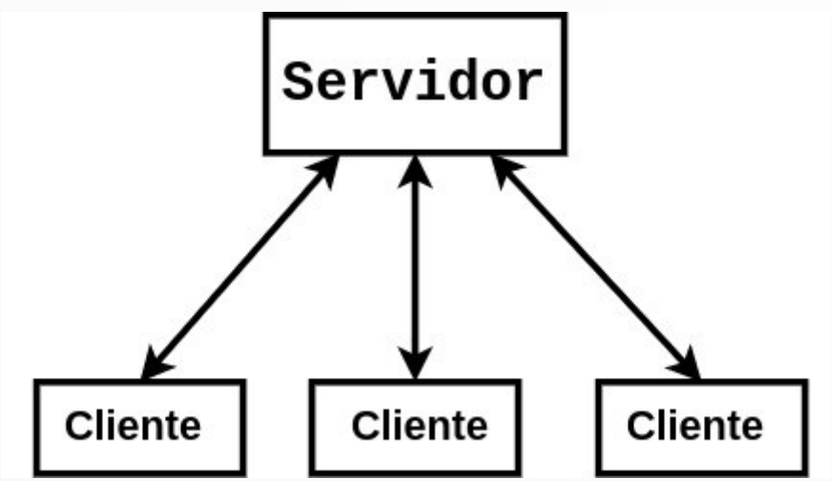
Sistemas Cliente-Servidor

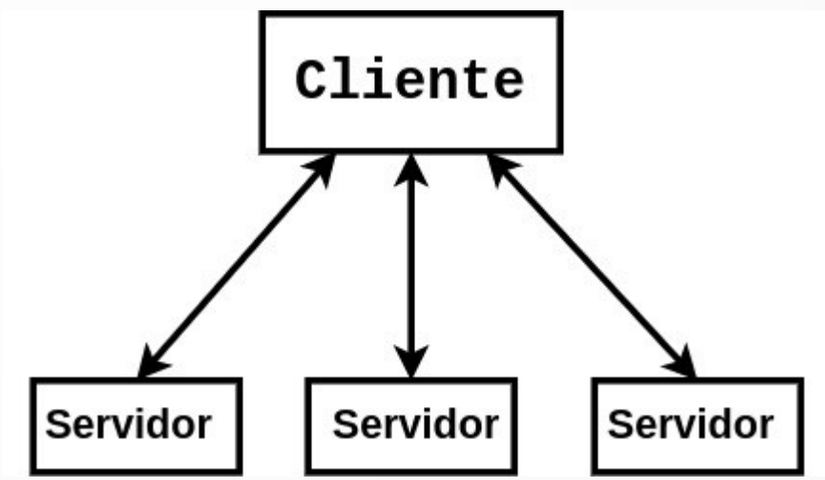
- Bancos de Dados X Redes de Computadores



Sistemas Cliente-Servidor

- Bancos de Dados X Redes de Computadores





Sistemas Cliente-Servidor

- Importante perceber: tem mais a ver com o modelo de comunicação do que com o processo propriamente dito (mais leve/pesado etc.)
- Intermediário: *peer – ao mesmo tempo cliente e servidor*
- O modelo P2P (*Peer-to-Peer*) pode ser visto como alternativo ao modelo Cliente-Servidor
- Mas pode ser visto como parte do modelo: citar exemplo do DNS

Sobre TCP e Sobre UDP

- Dois tipos de sistemas Cliente-Servidor na Internet
- Sobre TCP/IP: Orientados à Conexão
  - tem que abrir a conexão antes de comunicar, depois fechar a conexão
- Sobre UDP/IP: Não Orientado à Conexão

Concorrência Cliente-Servidor

- Definição de processos concorrentes

Concorrência Cliente-Servidor

- Definição de processos concorrentes
- Processos que executam “ao mesmo tempo”

Concorrência Cliente-Servidor

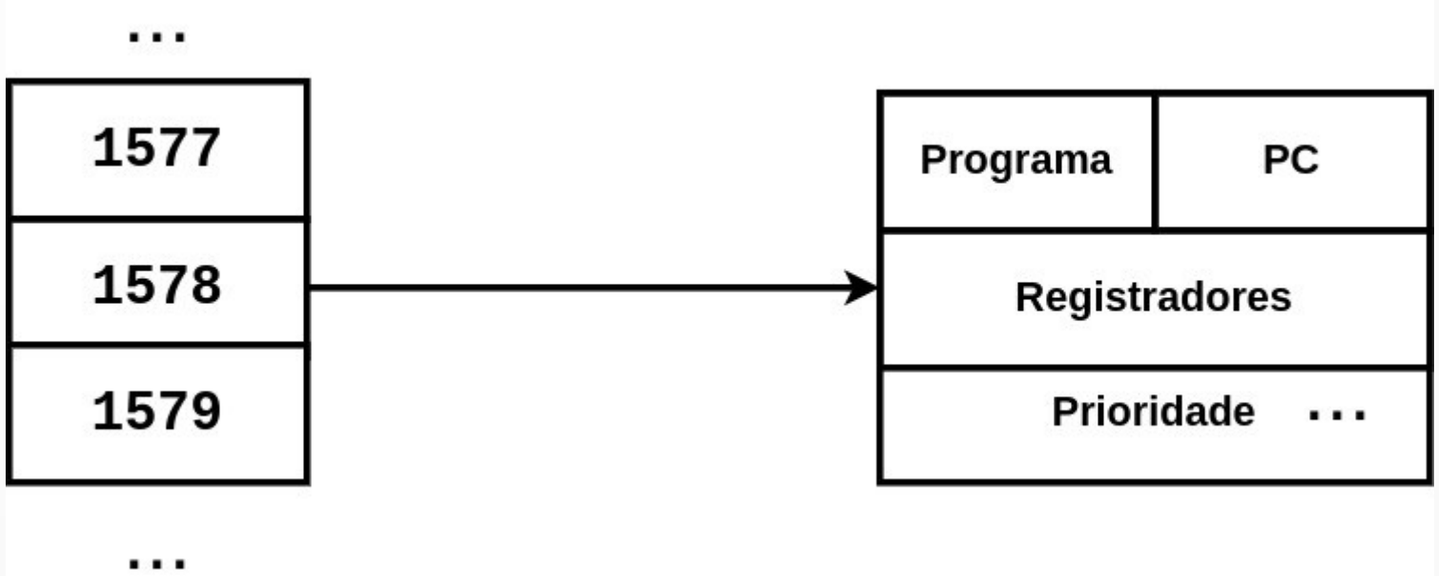
- Definição de processos concorrentes
- Processos que executam “ao mesmo tempo”
- Em contraposição: processos paralelos executam ao mesmo tempo
- Execução de processos concorrentes em um único processador (CPU)
  - em um instante específico de tempo apenas 1 processo está em execução
  - aparência para seres humanos: execução simultânea
  - arquiteturas modernas com muito paralelismo: +nebuloso!

Concorrência Cliente-Servidor

- Concorrência dos clientes: suporte do Sistema Operacional
- Concorrência dos servidores: deve ser programada
- Servidores concorrentes: comunicam com mais de um cliente “ao mesmo tempo”

Programação de Concorrência

- Processo: programa em execução
- Cada processo tem um identificador no Sistema Operacional: *pid*
- Índice de uma tabela do S.O. com informações sobre o processo (figura prox slide)
- Fácil criar uma cópia



Um Programa Concorrente

```
/* um programa concorrente simples */

int soma; /* variável global */
int main() {
  int i;
  soma = 0;

  for (i=1; i<=3; i++) {
    printf("i = %d\n", i);
    soma = soma + i;
  }
  printf("soma = %d\n", soma);
  return 0;
}
```

A Saída Deste Programa

```
i = 1
i = 2
i = 3
soma = 6
```

Um Programa Concorrente

```
/* um programa concorrente simples */
/* agora sim concorrente: fork()! /

int soma; /* variável global */
int main() {
   int i;
   soma = 0;
   fork();
   for (i=1; i<=3; i++) {
      printf("i = %d\n", i);
      soma = soma + i;
   }
   printf("soma = %d\n", soma);
   return 0;
}
```

A Saída Deste Programa

```
i = 1
i = 2
i = 3
soma = 6
i = 1
i = 2
i = 3
soma = 6
```

O *System Call fork()*

- O fork() é uma chamada de sistema que replica o processo em execução
- Cria um chamado “processo filho”
- O processo original é chamado “processo pai”
- Após o fork(): ambos os processos executam a instrução seguinte ao fork()

O *System Call fork()*

- Os dois processos são praticamente iguais
- Exceto pelo retorno do fork()
- Pai: pid do filho
- Filho: 0 (zero) (eu Elias acho isso pouco intuitivo ;)

```
int retorno_fork;
retorno_fork = fork();
if (retorno_fork != 0)
  {processo pai aqui}
else {processo filho aqui}
```

Servidor Concorrente

- Como programar um servidor concorrente?
- Quando chega uma requisição de um novo cliente
- Abre um processo filho para atender aquele novo cliente
- Diversos processos filhos podem estar atendendo diversos clientes “ao mesmo tempo”

Conclusão

- Hoje terminamos o TCP
  - Algoritmo de Nagle
  - Solução de Clark
  - Encerramento da Conexão
- Começamos Cliente-Servidor
  - Definição do modelo
  - Concorrência

Obrigado!

Lembrando: a página da disciplina é:
https://www.inf.ufpr.br/elias/redes