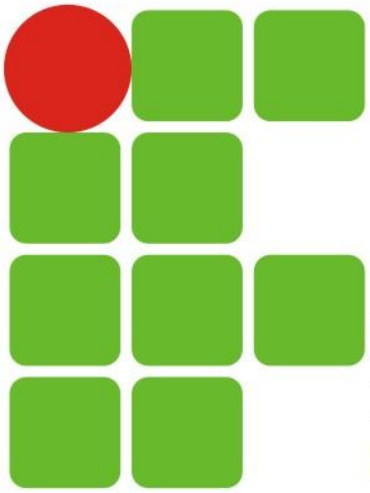


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Curso Superior em Redes de Computadores

Camada de Aplicação

Prof. Sales Filho <salesfilho@cefetrn.br>

Objetivo

- Apresentar os detalhes específicos dos tipos de aplicação
- Apresentar o modelo cliente-servidor
- Apresentar as características da interface *Socket*
- Apresentar os detalhes de projetos de servidores

Introdução

- Camada de aplicação
  - Trata os detalhes específicos de cada tipo de aplicação
    - Mensagens trocadas por cada tipo de aplicação definem um protocolo de aplicação
    - Cada protocolo de aplicação especifica a sintaxe e a semântica de suas mensagens
  - Diversos protocolos de aplicação
    - FTP (*File Tranfer Protocol*)
    - SMTP (*Simple Mail Transfer Protocol*)
    - DNS (*Domain Name System*)
    - HTTP (*HyperText Transfer Protocol*)

Introdução

- Camada de aplicação
  - Implementada usando processos de aplicação
  - Processos interagem usando o modelo cliente-servidor
  - Processos usam os serviços da camada de transporte
    - Processos interagem com as implementações dos protocolos de transporte através de uma API (*Application Programming Interface*)
    - A interface *Socket* é um dos principais exemplos de interface de interação

Modelo cliente-servidor

- Componentes
  - Servidor
    - Processo que oferece um serviço que pode ser requisitado pelos clientes através da rede
    - Comunica-se com o cliente somente após receber requisições
    - Executado continuamente aceitando requisições e respondendo
  - Cliente
    - Processo que requisita um serviço oferecido por um servidor
    - Inicia a interação com o servidor
    - Disponibiliza a interface para o usuário
    - Finaliza a execução após ser utilizado pelo usuário

Modelo cliente-servidor

- Paradigma requisição-resposta
  - Servidor
    - Aceita requisição dos clientes
    - Executa seu serviço realizando o processamento das requisições
    - Retorna o resultado para os respectivos clientes
  - Cliente
    - Envia requisições através da rede para um ou vários servidores
    - Aguarda o recebimento das respectivas respostas

Modelo cliente-servidor

- Identificação de processos
  - Clientes e servidores são identificados por meio das portas
  - Cliente deve conhecer, previamente, a porta usada pelo servidor
  - Servidor não precisa conhecer, previamente, a porta usada pelo cliente
  - Servidor descobre a porta usada pelo cliente somente após receber a requisição

Modelo cliente-servidor

- Identificação de processos
  - Portas são permanentemente reservadas para serviços padronizados e bem conhecidos
    - Porta 53 (DNS)
    - Porta 161 (SNMP)
  - Portas reservadas são utilizadas pelos servidores que implementam os respectivos serviços
  - Demais portas são disponíveis para uso dos clientes

Modelo cliente-servidor

- Negociação de porta
  - Servidor requisita uma porta reservada e bem conhecida, previamente reservada ao seu serviço
    - Servidor informa ao sistema operacional a porta que deseja utilizar e qual protocolo da camada de transporte
  - Cliente requisita uma porta qualquer não reservada
    - Sistema operacional escolhe a porta arbitrária para o cliente

Modelo cliente-servidor

- Alocação de portas
  - Padronizadas pela IANA (*Internet Assigned Numbers Authority*)
    - Reservada (0 – 1.023)
      - Atribuídas a serviços padronizados
      - Acessados apenas por processos privilegiados
    - Registradas (1.024 – 49.151)
      - Não são reservadas, mas apenas listadas para coordenar o uso para serviços não padronizados
      - Acessadas por qualquer processo
    - Dinâmicas (49.152 – 65.535)
      - Não possuem reserva, podendo ser usadas pelos clientes
      - Acessadas por quaisquer processo

Interface *Socket*

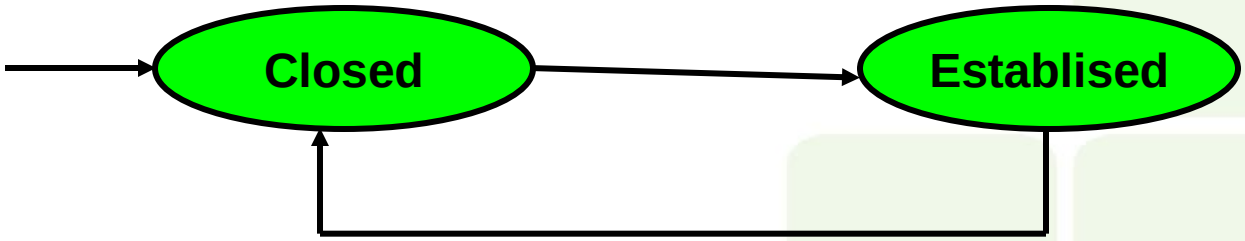
- Características
  - Define interface entre os processos de aplicação e as implementações dos serviços de transporte
  - Originalmente proposta para sistemas UNIX e a linguagem C
  - Amplamente adotada em diversas plataformas e linguagens (em Windows (winsocket); em Java (DatagramSocket - UDP e ServerSocket - TCP)
  - Um *Socket* é uma representação interna do S.O. para ponto de comunicação
    - É identificado pelos *endpoints* local e remoto
    - Cada *endpoint* é representado pelo par (Endereço IP, porta)

Interface *Socket*

- Características
  - Sockets UDP e TCP são gerenciados diferentemente:
  - UDP:
    - Não orientado a conexão;
    - Está sempre pronto para a enviar e receber, exceto quando a fila de Tx e Rx estiver cheia ;
    - O SO não necessita diferentes estados para socket.
  - TCP:
    - Cada conexão com identificação única por par endpoint local e remoto;
    - Não está pronto para enviar e receber, necessita de estabelecer e liberar a conexão;
    - O Socket passa por varios estados

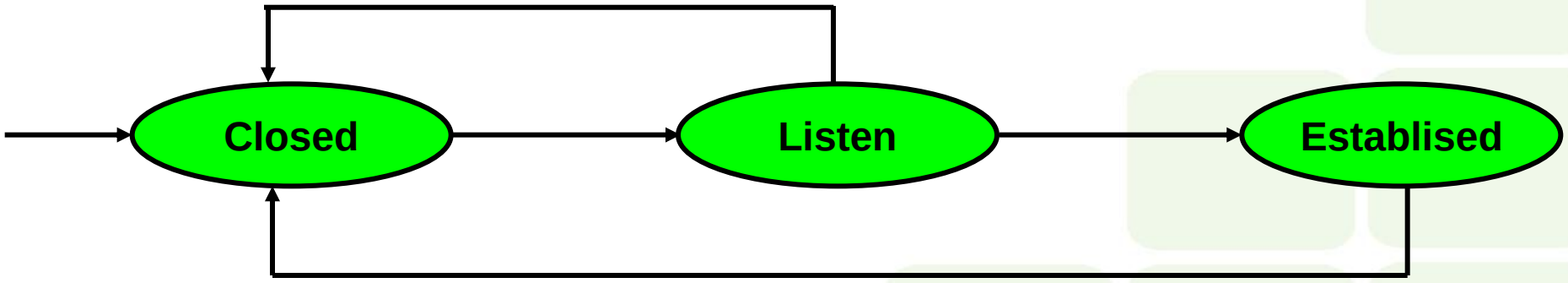
Interface *Socket*

- Estados de um *Socket* TCP
  - *Socket* ativo
    - Usado pelo cliente para ativamente enviar requisições de conexão ao servidor;
    - Após configurado, envia requisição de conexão e passa de Closed para Established.
    - Quando a conexão é feclada, volta à Closed



Interface *Socket*

- Estados de um *Socket* TCP
  - *Socket* passivo
    - Usado pelo servidor para passivamente aguardar por requisições de conexão



Interface *Socket*

- *Endpoint* local (TCP ou UDP)
  - Criado por default com endereço IP especial 0.0.0.0 e uma porta arbitrária selecionada pelo sistema operacional
    - O IP especial 0.0.0.0, indica que datagrama UDP ou requisição de conexão TCP será aceita quando destinado a qualquer IP das interfaces do sistema
  - Pode ser atribuído um endereço IP e uma porta específica
    - Endereço IP específico deve ser evitado em sistemas Multihomend, exceto por questões de segurança
    - Servidor deve configurar uma porta específica
    - Cliente usa a porta selecionada pelo sistema operacional

Interface *Socket*

- *Endpoint* remoto (TCP e UDP)
  - Criado por default com endereço IP especial 0.0.0.0 e porta "*".
    - Indica, em Servidor, que datagramas UDP e requisições de conexões TCP serão aceitas quando recebidas de qualquer endereço IP e porta remota.
  - Pode ser atribuído um endereço IP e uma porta específica
    - Cliente UDP ou TCP deve especificar o endereço IP e a porta do servidor ao endpoint remoto do seu socket;
    - Servidor UDP pode configurar um endereço IP e porta específica
      - Deve ser evitado em sistemas Multihomend, exceto por questões de segurança
    - Servidor TCP usa associação default

Interface *Socket*

- *Endpoint* local e remoto
  - Vários *sockets* podem utilizar o mesmo número de porta local, desde que os seus respectivos *endpoints* local e remotos sejam diferentes

```
> netstat -tan
Proto  Recv-Q  Send-Q  Local Address       Foreign Address   State
udp         0       0  192.168.1.33:161    0.0.0.0:*
udp         0       0  192.168.1.65:161    0.0.0.0:*
udp         0       0  0.0.0.0:161         0.0.0.0:*
```

```
> netstat -tan
Proto  Recv-Q  Send-Q  Local Address       Foreign Address   State
tcp         0       0  192.168.1.65:25     0.0.0.0:*         LISTEN
tcp         0       0  192.168.1.129:25    0.0.0.0:*         LISTEN
tcp         0       0  0.0.0.0:25          0.0.0.0:*         LISTEN
```

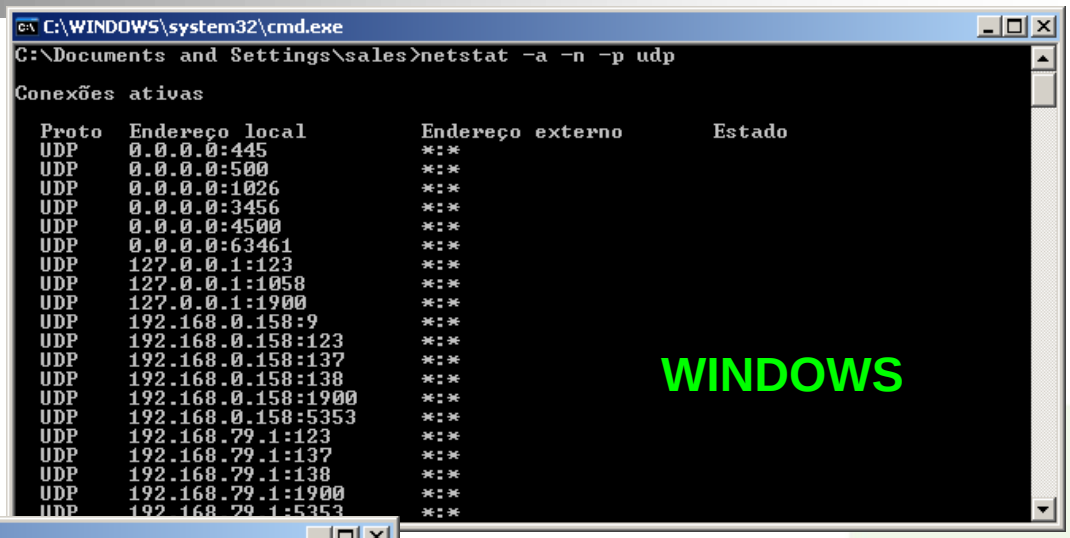
Interface *Socket*

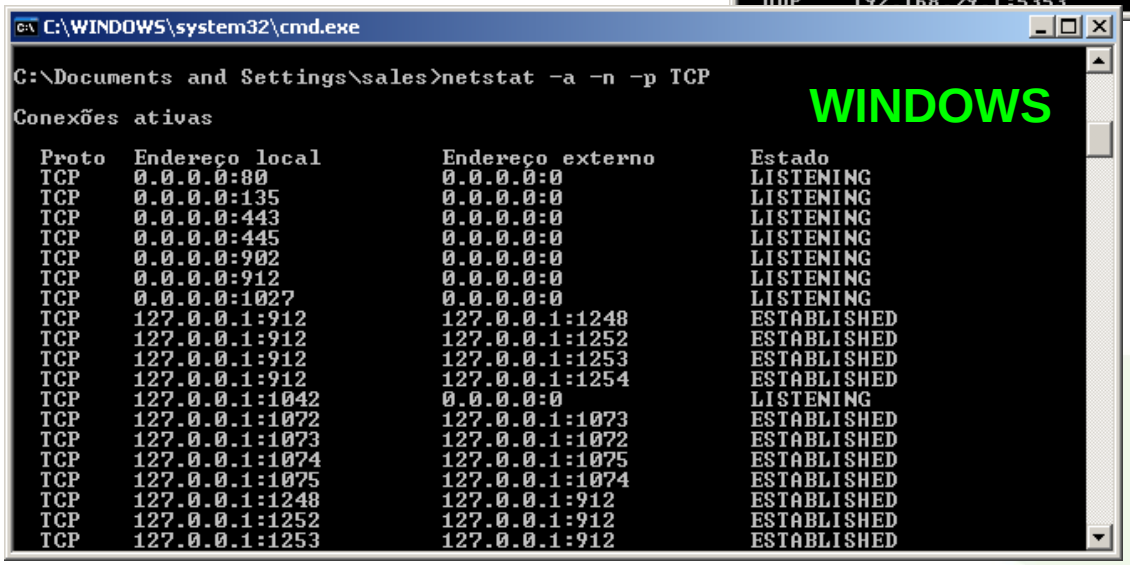
- *Endpoint* local e remoto
  - Vários *sockets* podem utilizar o mesmo número de porta local, desde que os seus respectivos *endpoints* local e remotos sejam diferentes

```
root@ubuntu:~# netstat -anlpu
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
udp        0      0 0.0.0.0:68              0.0.0.0:*
4091/dhclient3
udp        0      0 127.0.0.1:52203         127.0.0.1:52203         ESTABLISHED
4256/postgres
root@ubuntu:~#
root@ubuntu:~# netstat -anlpt
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
4361/apache2
tcp        0      0 0.0.0.0:5432            0.0.0.0:*               LISTEN
4256/postgres
tcp6       0      0 :::22                   :::*                    LISTEN
4231/sshd
tcp6       0      0 :::5432                 :::*                    LISTEN
4256/postgres
root@ubuntu:~# _
```

Interface *Socket*

- *Endpoint* local e remoto



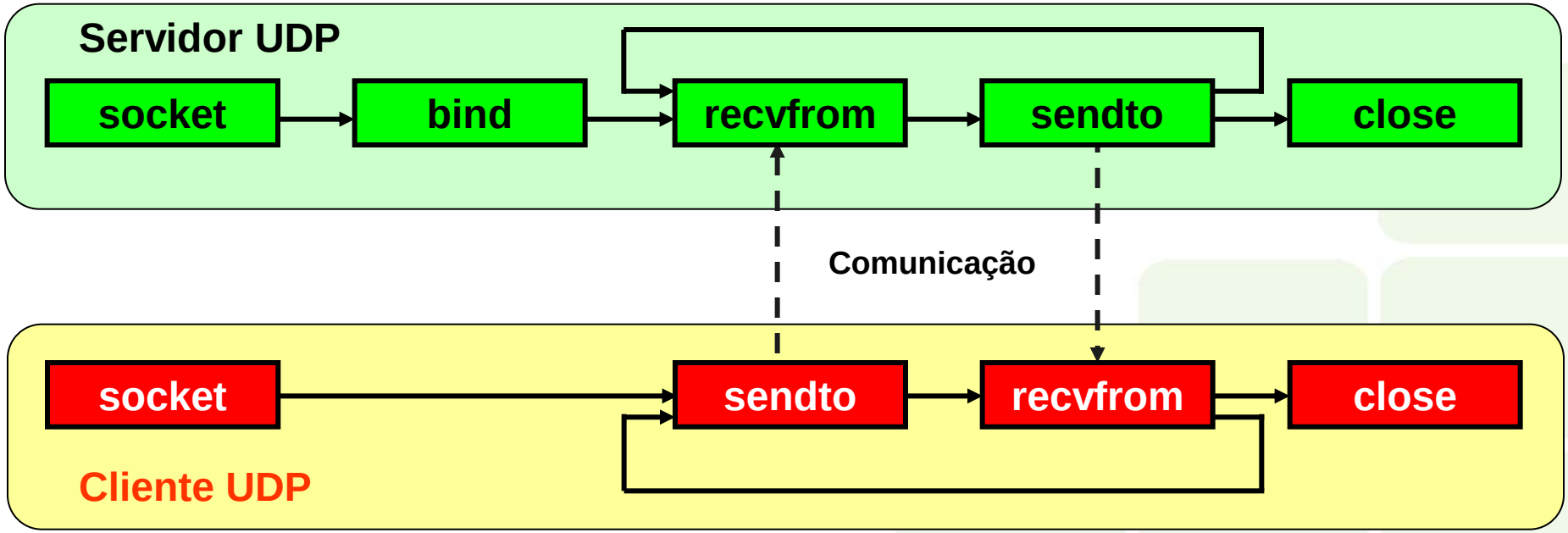


Interface *Socket*

- Modelo de programação
  - Explora chamadas ao sistema operacional
  - Adota o modelo de processo, que é baseado no paradigma **abrir - ler/escrever - fechar**
    - Principais funções
      - Socket (Cria um ponto de comunicação - socket)
      - Bind (Associa o endpoint local do socket a uma porta/IP)
      - Listen (Indica ao Socket que está aguardando conexões)
      - Accept (Bloqueia o processo até receber uma solicitação de conexão)
      - Connect (Realiza um pedido de conexão a um endpoint remoto)
      - Read (TCP) / recvfrom (UDP) (Recebe dados)
      - Write (TCP) / sendto (UDP) (Envia dados)

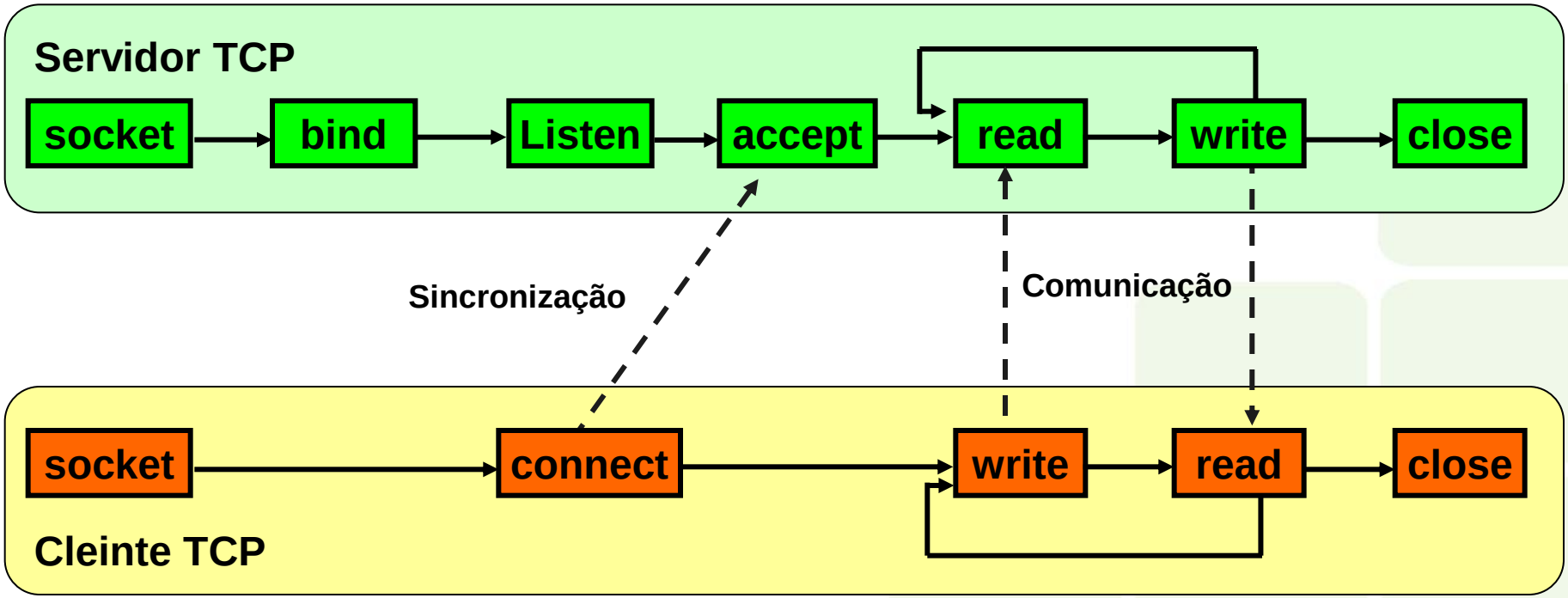
Interface *Socket*

- Clientes e servidores UDP
  - Modelo de implementação



Interface *Socket*

- Clientes e servidores TCP
  - Modelo de implementação

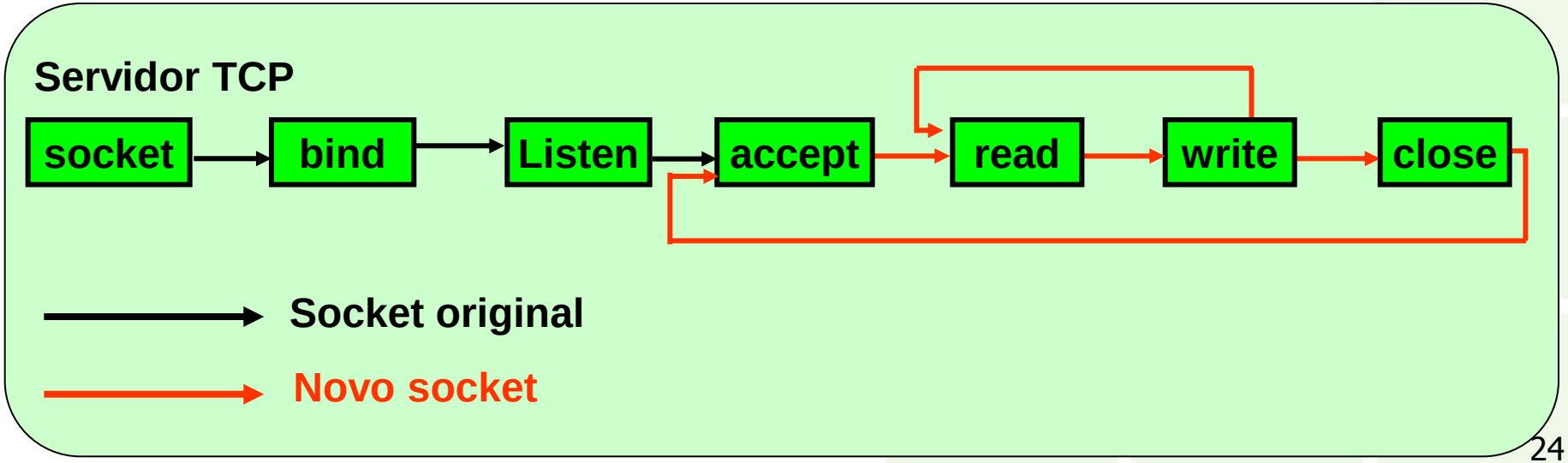


Projeto de servidores

- Tratamento de requisição
  - Servidor iterativo (*single threaded*)
    - Trata requisição de um único cliente a cada instante
    - Implementado como um único processo
  - Servidor concorrente (*multi-threaded*)
    - Trata simultaneamente requisições de vários clientes
    - Implementado como vários processos ou *threads* independentes
    - Cada processo ou *thread* trata individualmente as requisições de um determinado cliente

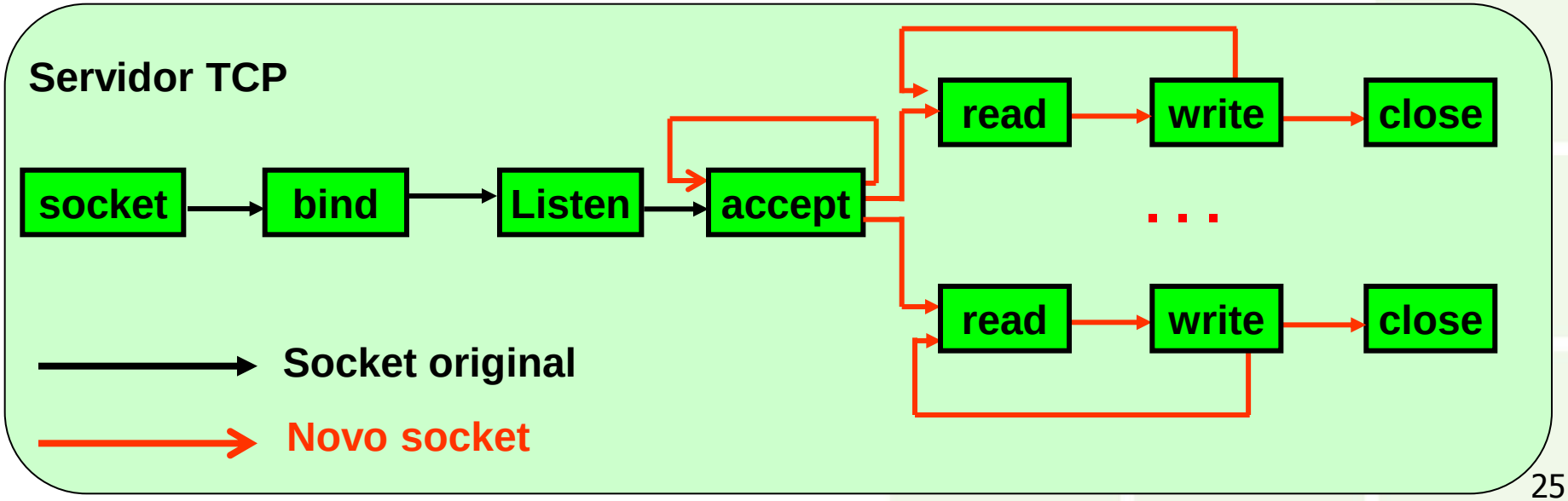
Projeto de servidores

- Tratamento de requisições
  - Servidor Iterativo
    - Adequado para serviços com reduzida taxa de requisição
    - Requisições com baixa carga de processamento



Projeto de servidores

- Tratamento de requisições
  - Servidor Concorrente
    - Adequado para serviços com elevada taxa de requisição
    - Requisições com alta carga de processamento



Projeto de servidores

- Compartilhamento de portas
  - Servidor concorrente
    - *Socket* mestre
      - Está sempre no estado *Listen*
      - *Endpoint* local possui o endereço especial **0.0.0.0** e a porta específica do servidor
      - *Endpoint* remoto possui o endereço especial **0.0.0.0** e porta "*"
    - *Sockets* dos escravos
      - Estão sempre no estado *established*
      - *Endpoints* locais possuem endereço IP da estação e a porta específica do servidor
      - *Endpoints* remotos possuem os endereços IP e as portas dos respectivos clientes

Projeto de servidores

```
root@ubuntu:~# netstat -anlpt
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
4361/apache2
tcp        0      0 0.0.0.0:5432            0.0.0.0:*               LISTEN
4256/postgres
tcp        0      0 192.168.0.146:42755     200.160.2.8:21          ESTABLISHED
4602/wget
tcp        0      0 192.168.0.146:80        192.168.0.158:1525      ESTABLISHED
4383/apache2
tcp        0      0 192.168.0.146:35258     200.160.2.8:56705       ESTABLISHED
4602/wget
tcp6       0      0 :::22                   :::*                    LISTEN
4231/sshd
tcp6       0      0 :::5432                 :::*                    LISTEN
4256/postgres
root@ubuntu:~# _
```

Compartilhamento de portas

- Servidor concorrente
  - Existe um único *socket* no estado *Listen, por Porta*
    - Somente processa segmentos SYN
  - Podem existir diversos *sockets* no estado *established*
    - Somente processam segmentos de dados
    - Endpoints local e remoto desses sockets devem ser diferentes

```
> netstat -tan
Proto  Recv-Q  Send-Q  Local Address      Foreign Address      State
tcp         0       0  0.0.0.0:25         0.0.0.0:*            LISTEN
tcp         0       0  150.1.20.1:25      192.10.1.50:57568    ESTABLISHED
tcp         0       0  150.1.20.1:25      200.50.2.10:58461    ESTABLISHED
tcp         0       0  150.1.20.1:25      150.10.1.20:60496    ESTABLISHED
```

Projeto de servidores

- Gerenciamento de conexões TCP
  - Evita perda de requisições por indisponibilidade de processamento momentâneo do Servidor
  - O SO cria Fila de Requisições
    - Associada a *sockets* no estado de *Listen*
    - Armazena requisições de conexões que já foram aceitas pela entidade TCP
      - Processamento da requisição é realizado pela entidade TCP e não pelo servidor
      - Requisição presente na fila ainda não são aceitas pelo servidor
      - A requisição somente é aceita e removida da fila quando o servidor ativa a função *accept*
    - Possui capacidade fixa, mas pode ser configurada pelo servidor com a função *Listen*

Projeto de servidores

- Multiplicidade do serviço
  - Servidor de um único serviço
    - Aguarda requisições em um único *socket,* que possui uma única porta associada ao *endpoint* local
    - Implementação baseada na função *accept*
  - Servidor de múltiplos serviços
    - Aguarda requisições em diversos *sockets*, que possuem diferentes portas associadas aos respectivos *endpoints* locais
    - Para reduzir o número de processos em execução, a interface socket permite que um único processo aguarde requisições de conexão em diversos sockets simultaneamente.
    - Implementação baseada na função *select*
      - Exemplo: xinetd (Unix/Linux)

Referências

- Comer, Douglas E., Interligação de Redes Com Tcp/ip
- James F. Kurose, Redes de Computadores e a Internet
- Escola Superior de Redes, Arquitetura e Protocolos de Redes TCP/IP