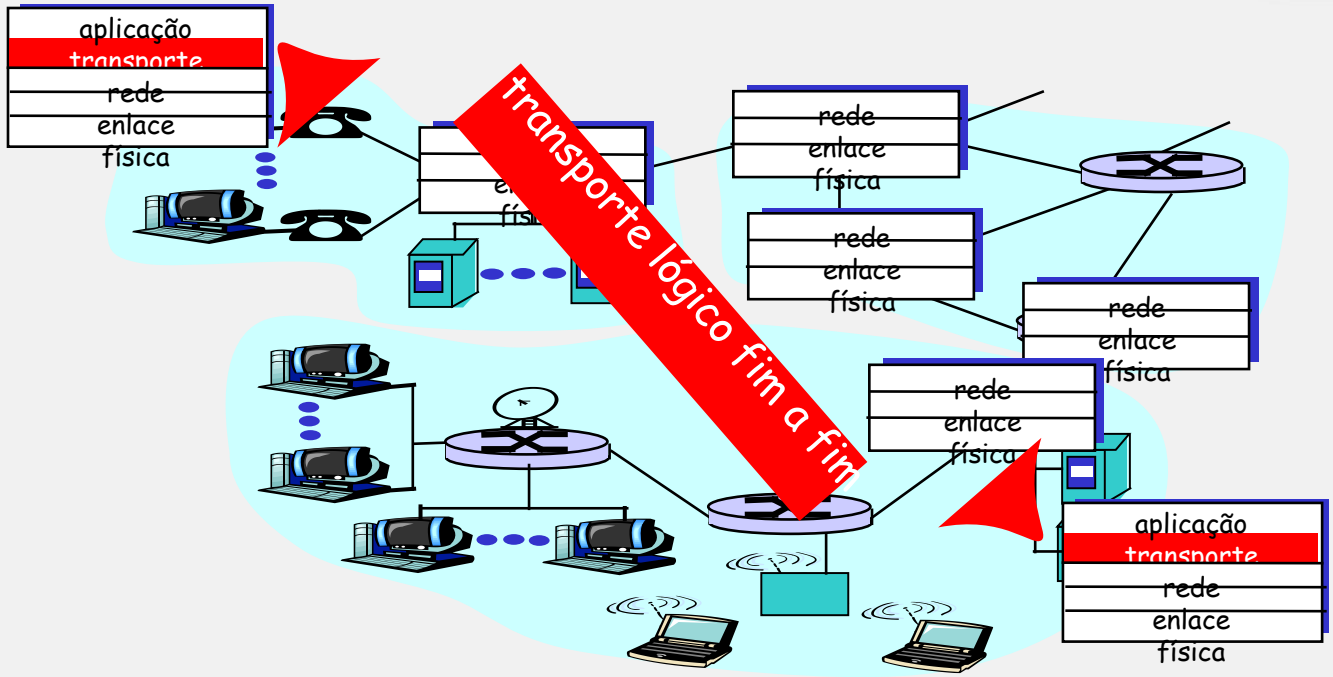
PUCPR
GRUPO MARISTA
ESCOLA
POLITÉCNICA

Camada de Transporte. Protocolos TCP e UDP.

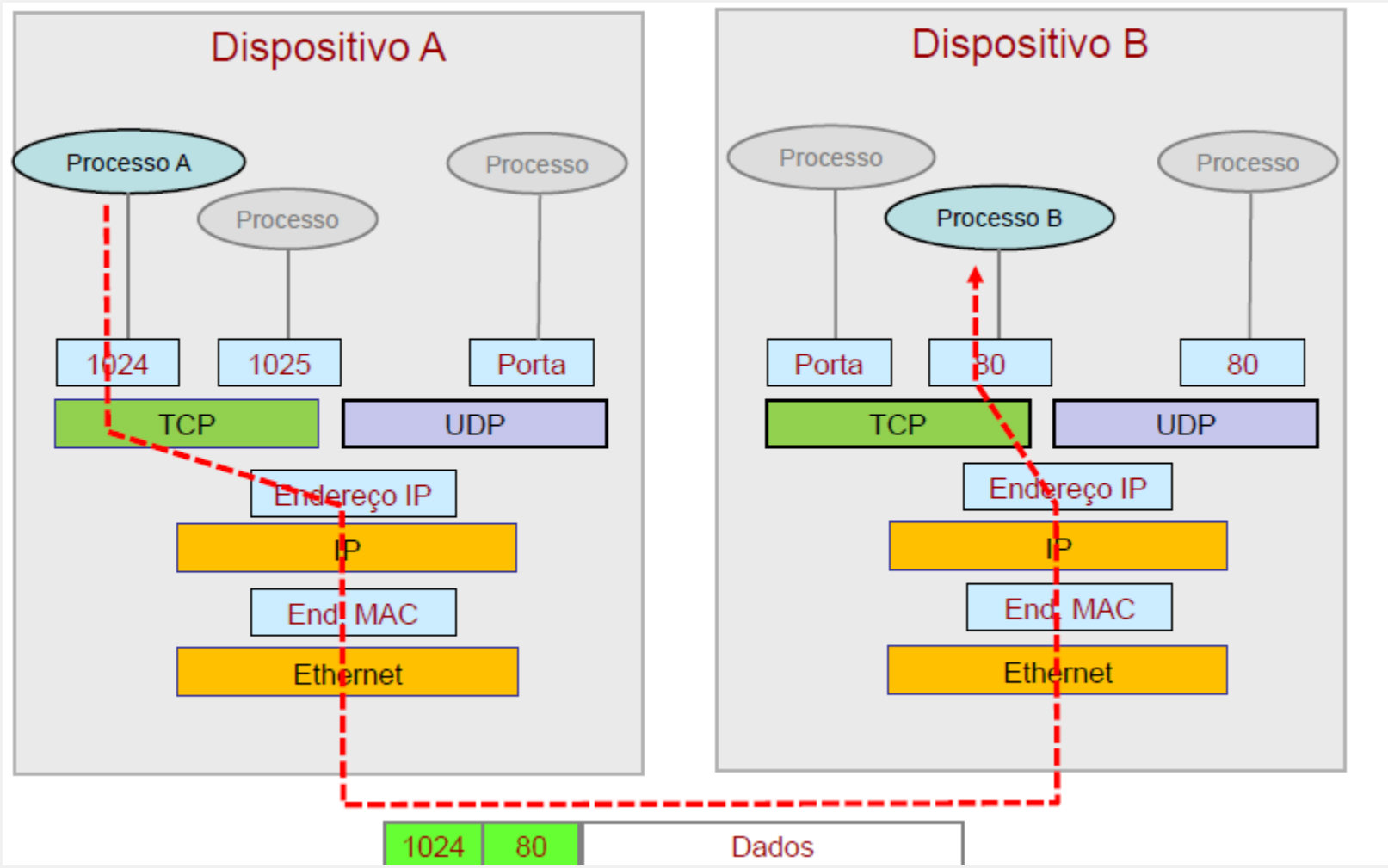
Serviços e protocolos de transporte



- provê *comunicação lógica* entre processos de aplicação
- protocolos de transporte executam em sistemas hosts
- na origem: msgs da app são quebradas em *segmentos* e passadas para a camada de rede enviar
- no destino: remontagem dos segmentos em msgs e envio para a camada de aplicação

2

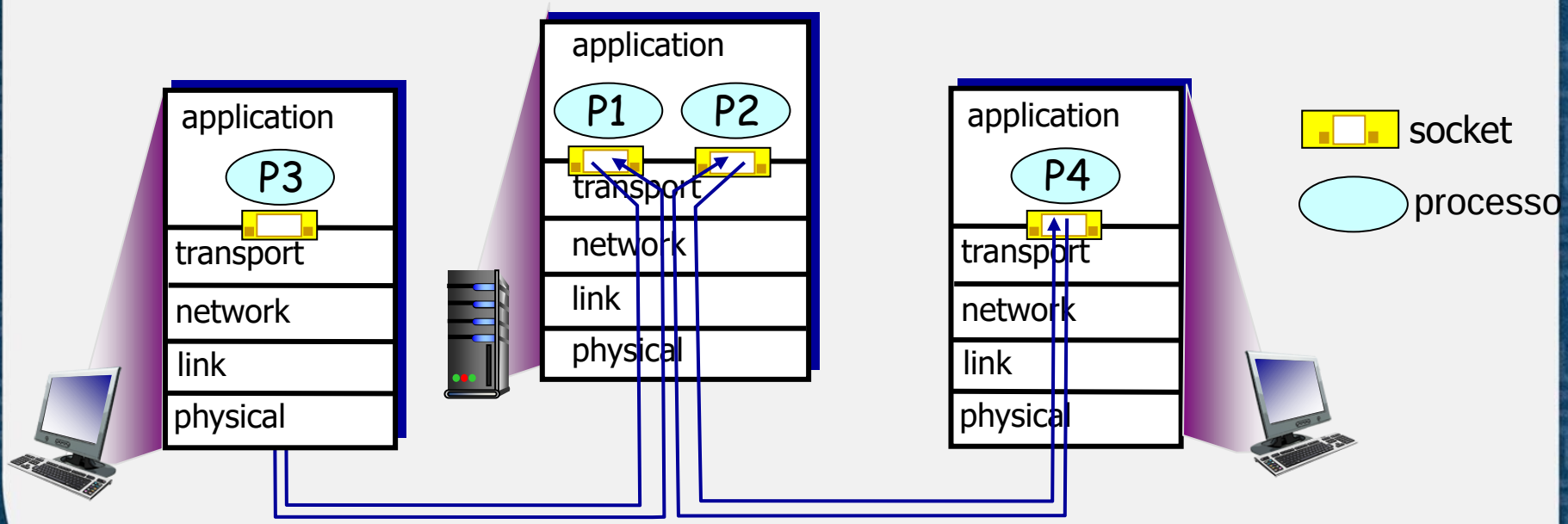
Serviços e protocolos de transporte



3

Multiplexação/demultiplexação

- Fluxos de comunicação entre processos
- Vários sockets ativos



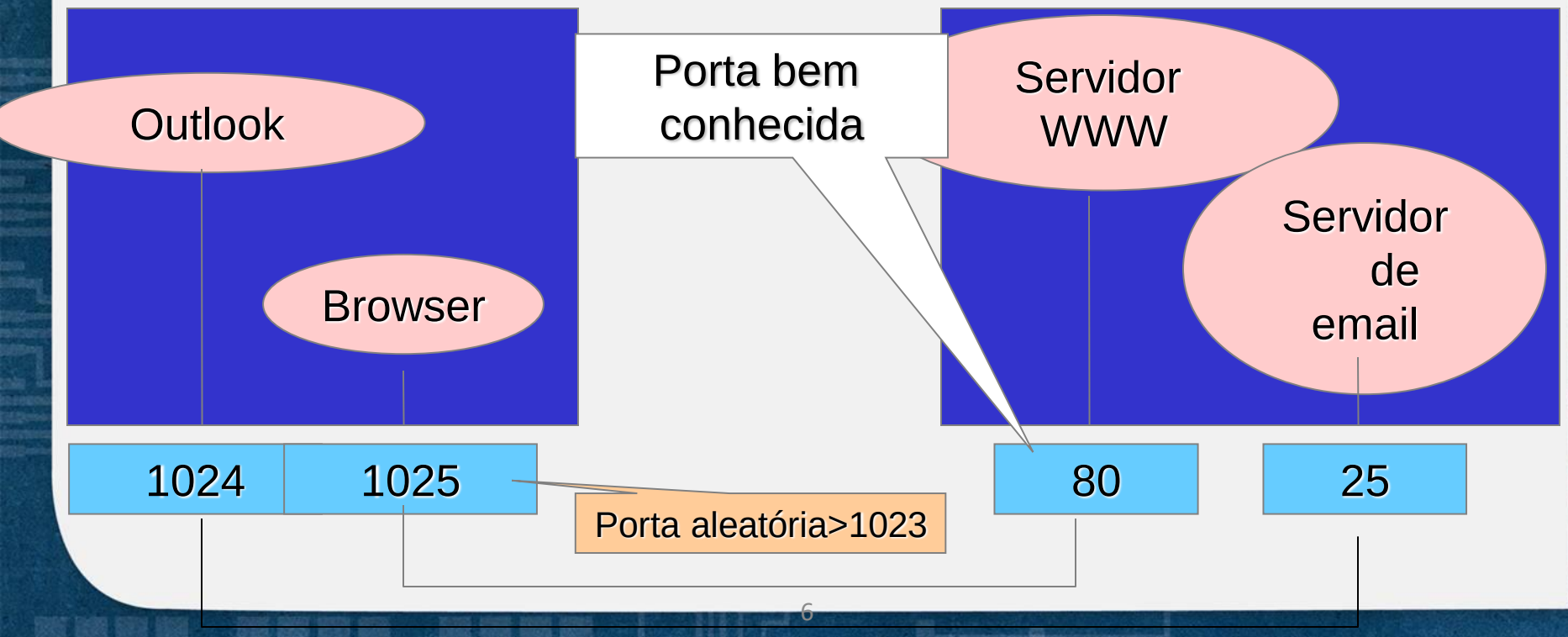
Protocolos da Camada de Transporte

- TCP
  - comunicação confiável e entrega em ordem
  - controle de congestionamento
  - controle de fluxo
  - estabelecimento de conexão
- UDP
  - comunicação não confiável e entrega sem controle de ordem
  - uma extensão do modelo "best-effort" do IP
- O que não tem
  - garantia de atraso
  - garantia de bandwidth

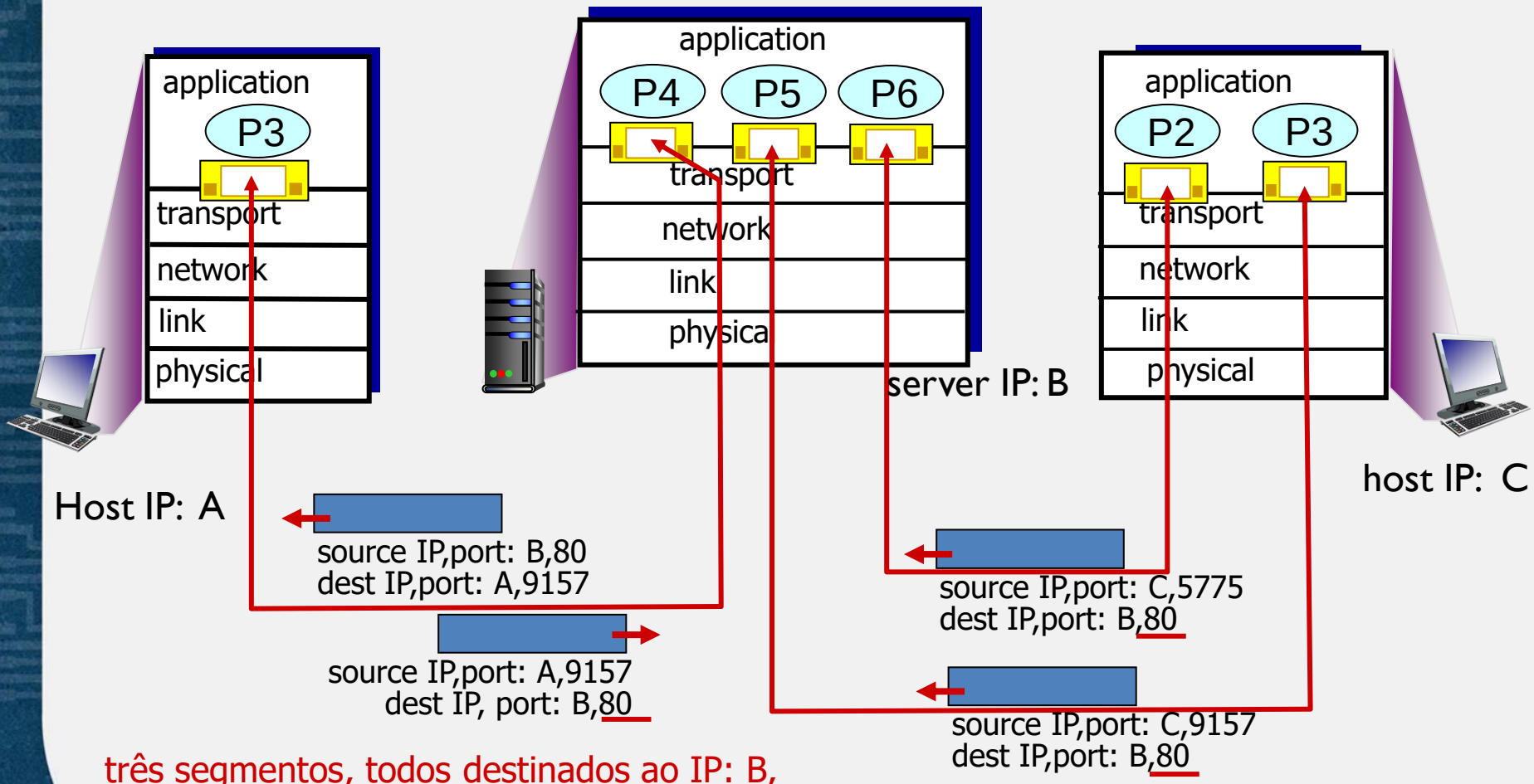
TCP	UDP
Orientado a Conexão Identifica uma sequencia de pacotes com pertencentes a uma mesma transmissão	Não Orientado a Conexão Cada pacote é uma transmissão independente
Transmissão por Fluxo Segmentação e Remontagem feita pelo S.O.	Transmissão por Datagramas: Segmentação e Remontagem feita pela aplicação.
Transmissão Confiável Confirma recebimento e retransmite pacotes perdidos	Não confiável Cabe a aplicação implementar os mecanismos de retransmissão

Portas identificam os processos

Porta Origem	Porta Destino	Dados



Exemplo



três segmentos, todos destinados ao IP: B,
e na porta: 80

Protocolo UDP

- "melhor esforço", segmentos UDP podem ser:
  - perdidos
  - entregues fora de ordem
- *sem conexão:*
  - cada segmento UDP tratado independente

- exemplo de uso:
  - streaming multimídia (tolerante a perdas)
  - DNS
  - SNMP
- se precisar de confiabilidade na comunicação com UDP:
  - adicione na camada de cima ou na aplicação

Formato do segmento UDP

PUCPR GRUPO MARISTA | ESCOLA POLITÉCNICA

por que usar UDP?

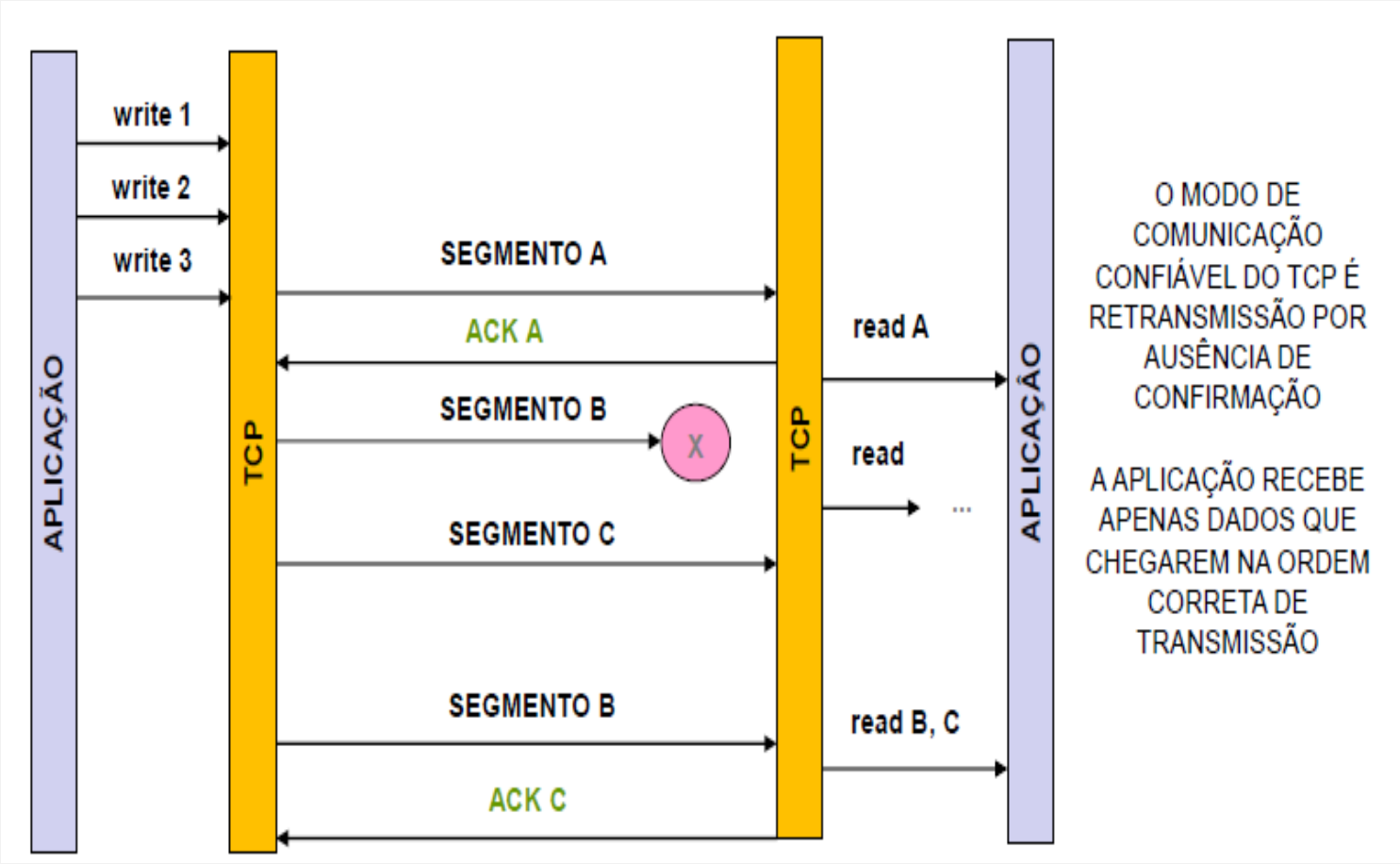
- não tem tempo gasto estabelecendo e mantendo a conexão
- mais simples: não precisa manter estado
- header menor
- sem controle de congestionamento: UDP “vai” na taxa que a aplicação quer

Protocolo TCP

- point-to-point:
  - um host de origem, um de destino
- comunicação confiável, em ordem do streaming de *bytes da app*
- controle de congestionamento

- full duplex:
  - fluxo bi-direcional na conexão
- orientado à conexão
- controle fluxo:
  - uma app não sobrecarrega a outra

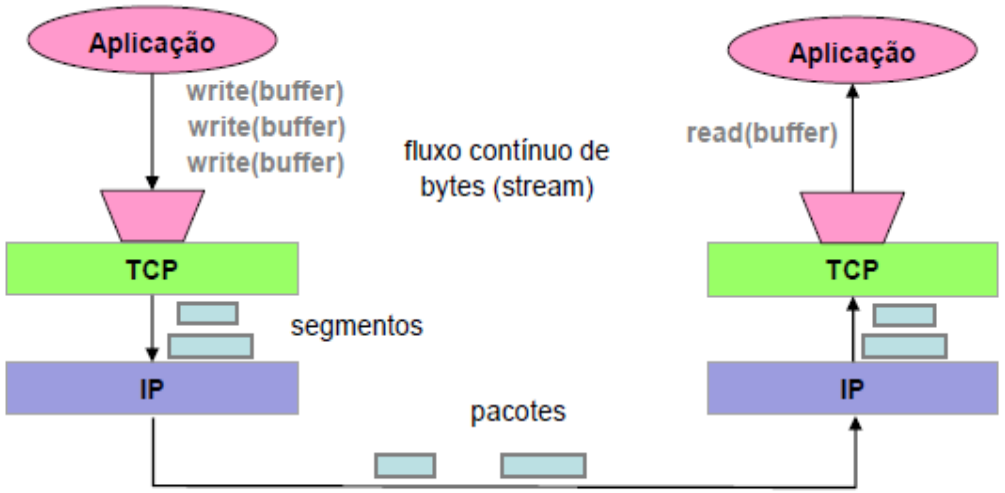
Protocolo TCP: comunicação confiável



O MODO DE COMUNICAÇÃO CONFIÁVEL DO TCP É RETRANSMISSÃO POR AUSÊNCIA DE CONFIRMAÇÃO

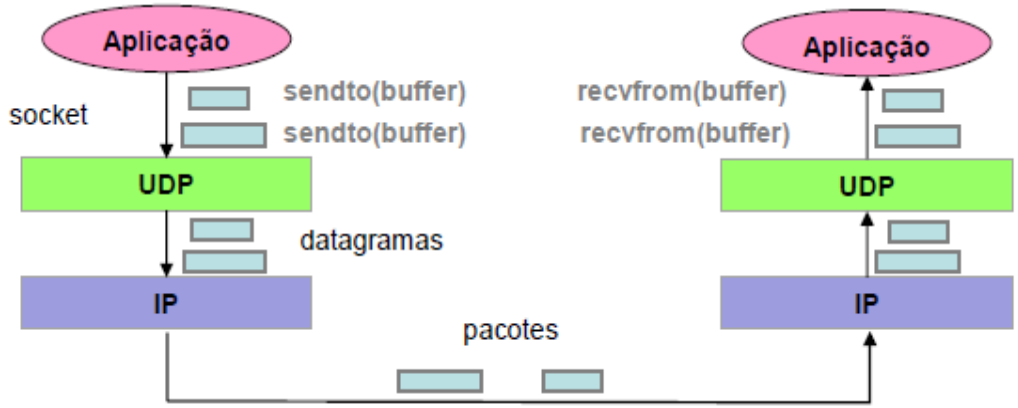
A APLICAÇÃO RECEBE APENAS DADOS QUE CHEGAREM NA ORDEM CORRETA DE TRANSMISSÃO

Modelo de comunicaçao: TCPxUDP



NA TRANSMISSÃO POR FLUXO A APLICAÇÂO NÃO CONTROLA O TAMANHO DO PACOTE

OS DADOS RECEBIDOS SÃO VISTOS PELA APLICAÇÃO RECEPTORA COMO UM FLUXO CONTÍNUO DE BYTES



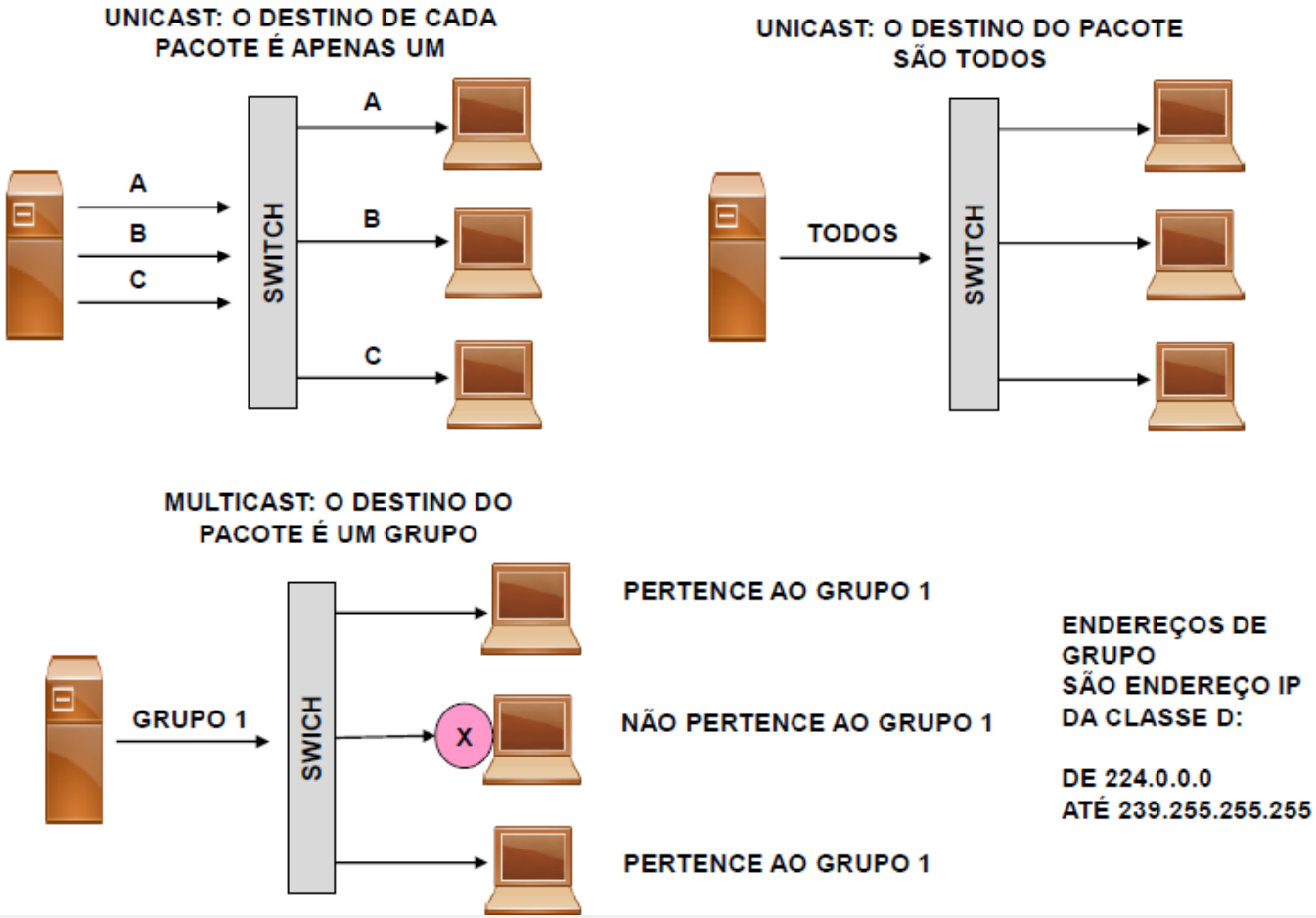
NA TRANSMISSÃO POR DATAGRAMA A APLICAÇÃO DEFINE O TAMANHO DOS PACOTES

A MENSAGEM RECEBIDA É LIDA POR INTEIRO (recvfrom)

Modelo de comunicaçao: TCPxUDP

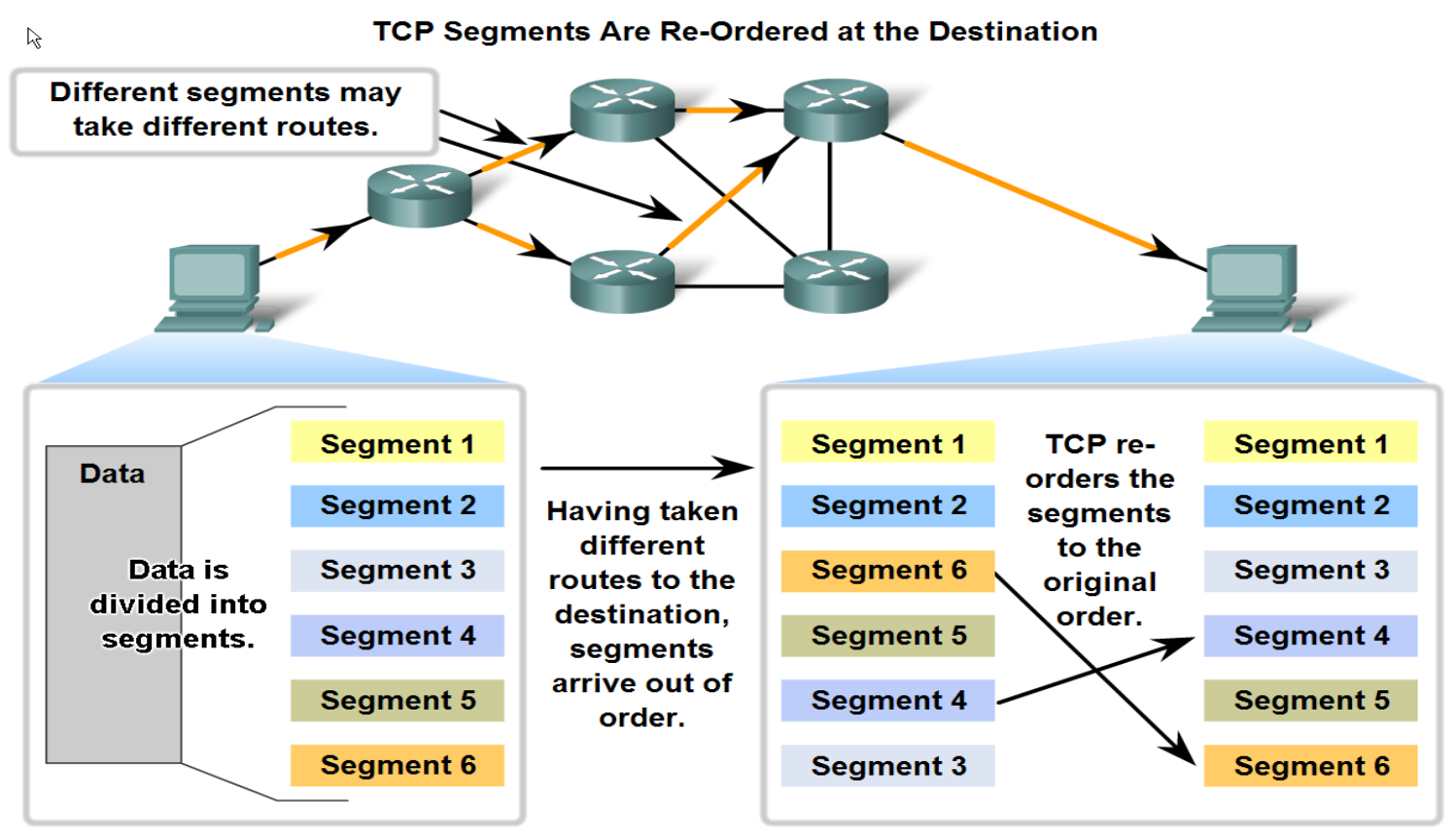
TCP	UDP
Controle de Congestionamento Perda de pacotes fazem o transmissor reduzir sua taxa de transmissão entre confirmações	Sem controle Perda de pacotes não interfere na velocidade de transmissão
Controle de Fluxo O espaço livre no buffer do S.O. do receptor é informado ao transmissor a cada confirmação	Sem controle: O transmissor não recebe nenhuma informação sobre o buffer do receptor.
Apenas Unicast O destino de um pacote é apenas um dispositivo	Unicast, Broadcast e Multicast O destino de um pacote pode ser um ou mais dispositivos

Unicast, Broadcast e Multicast



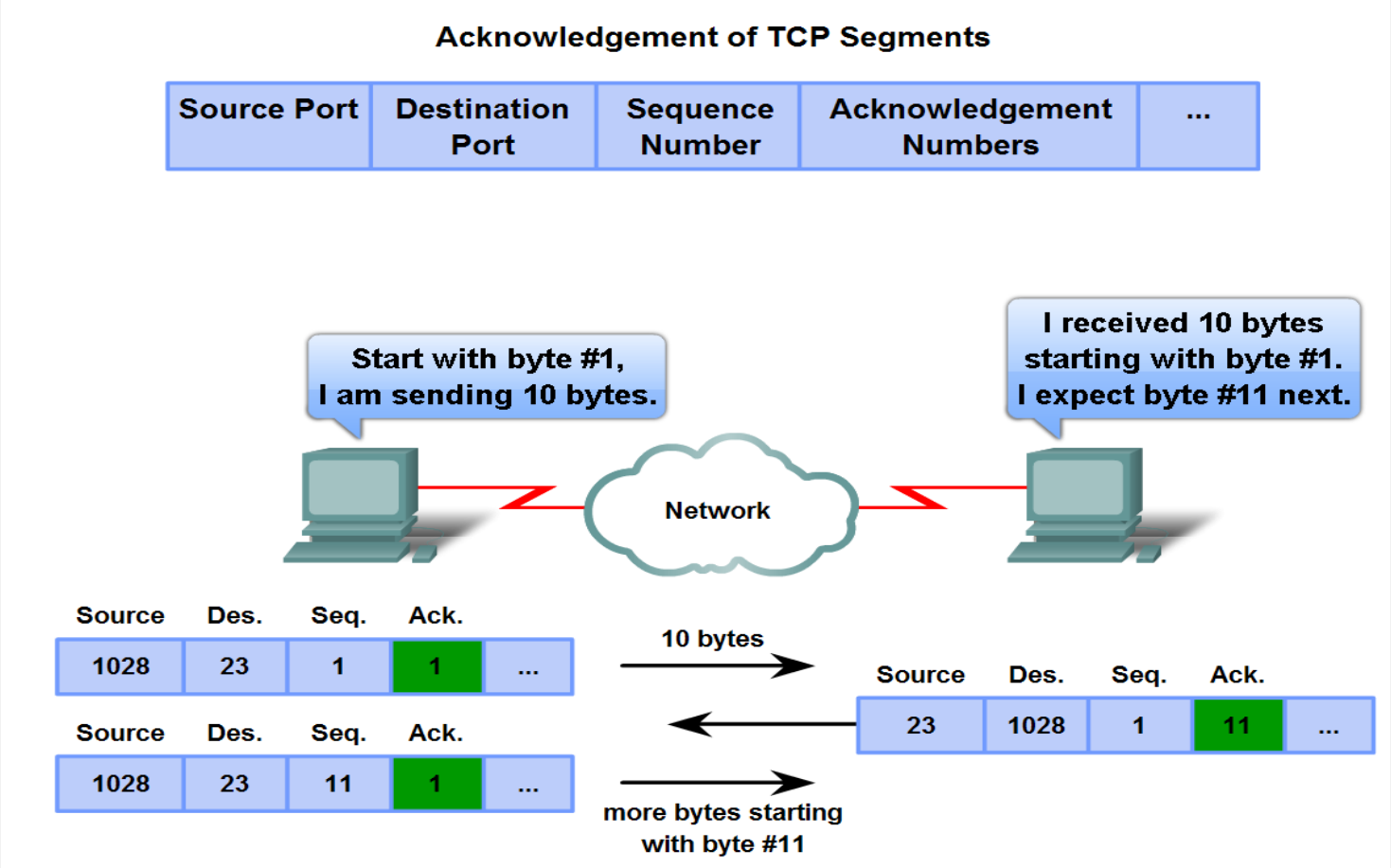
Sessão TCP

- Números de sequência são usados para reconstruir os segmentos



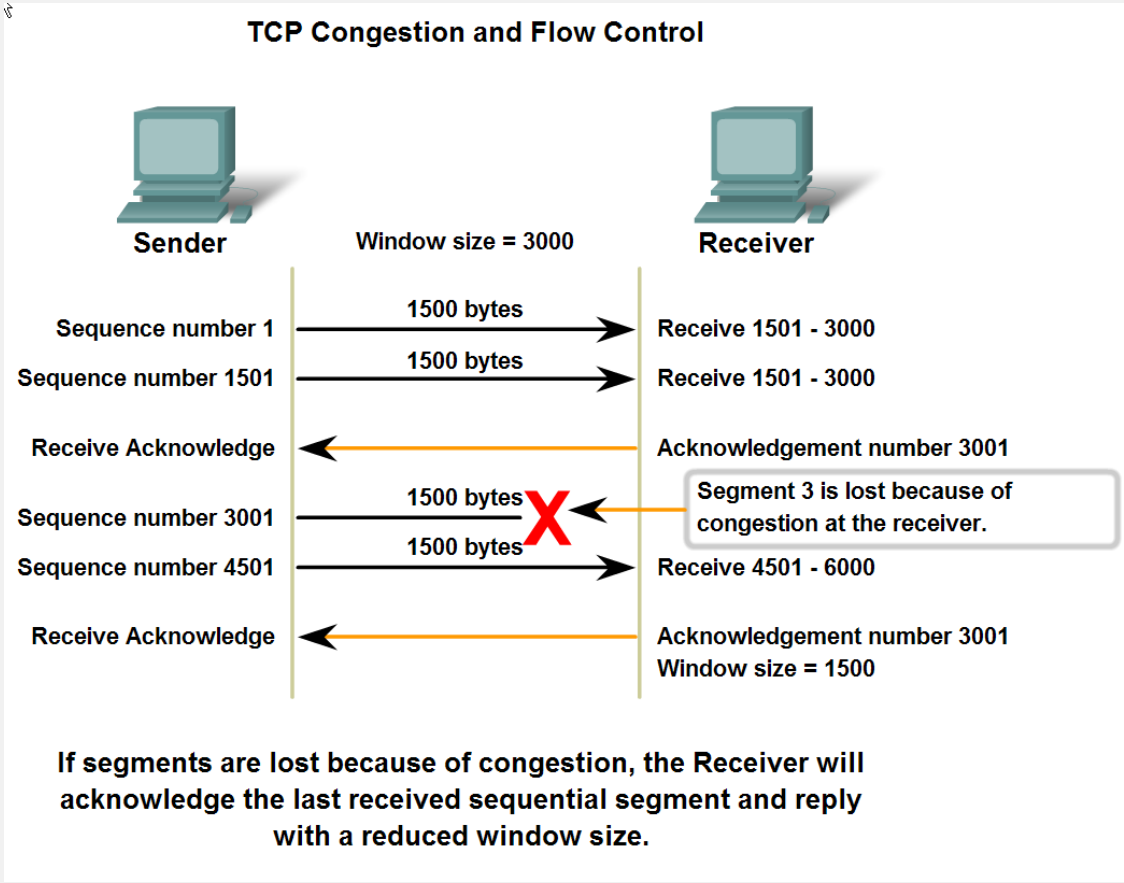
Sessões TCP

Números de sequência são usados para reconstruir os segmentos



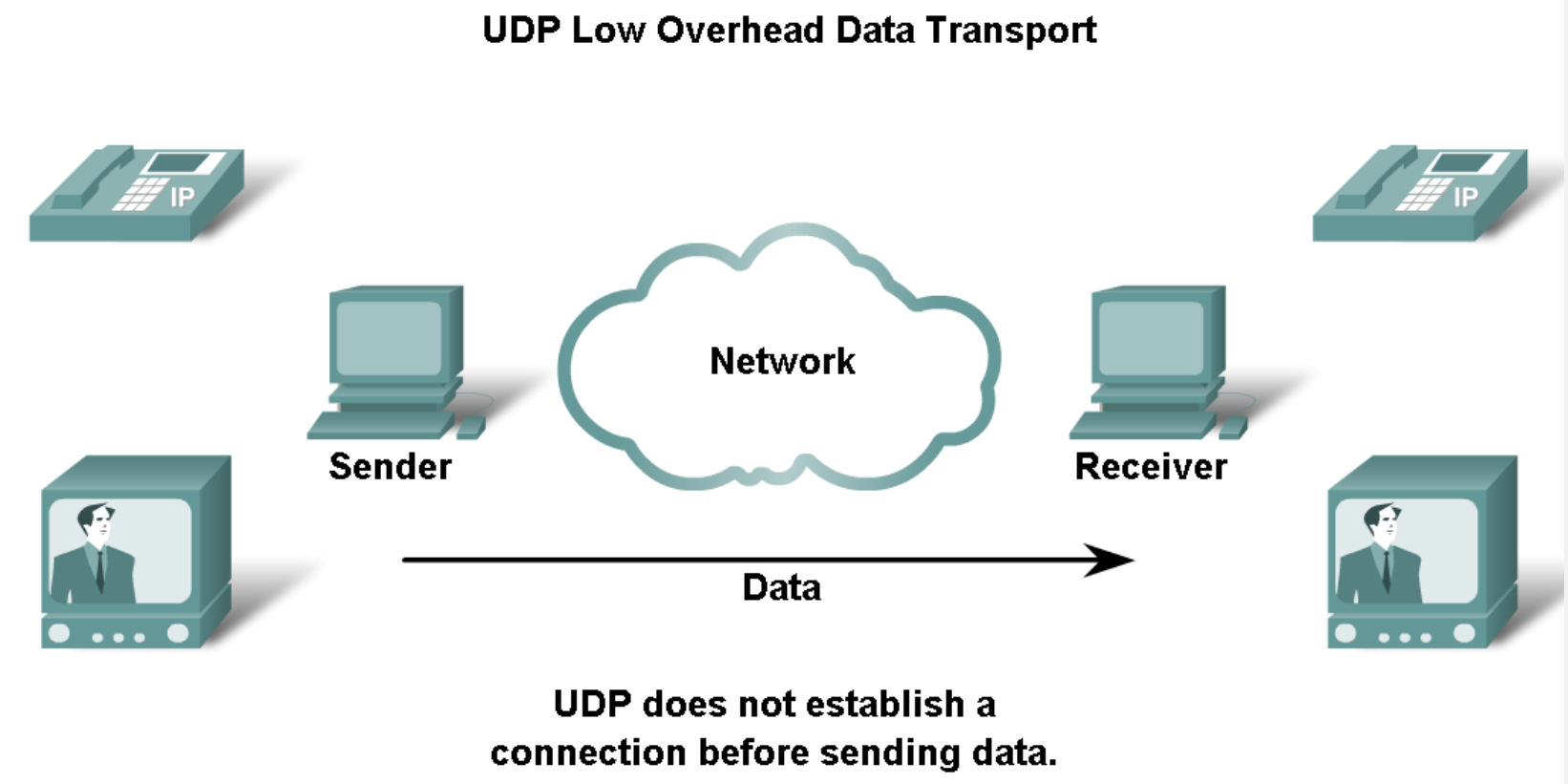
Sessão TCP

- Tamanho da janela, perda de dados e o controle de congestionamento



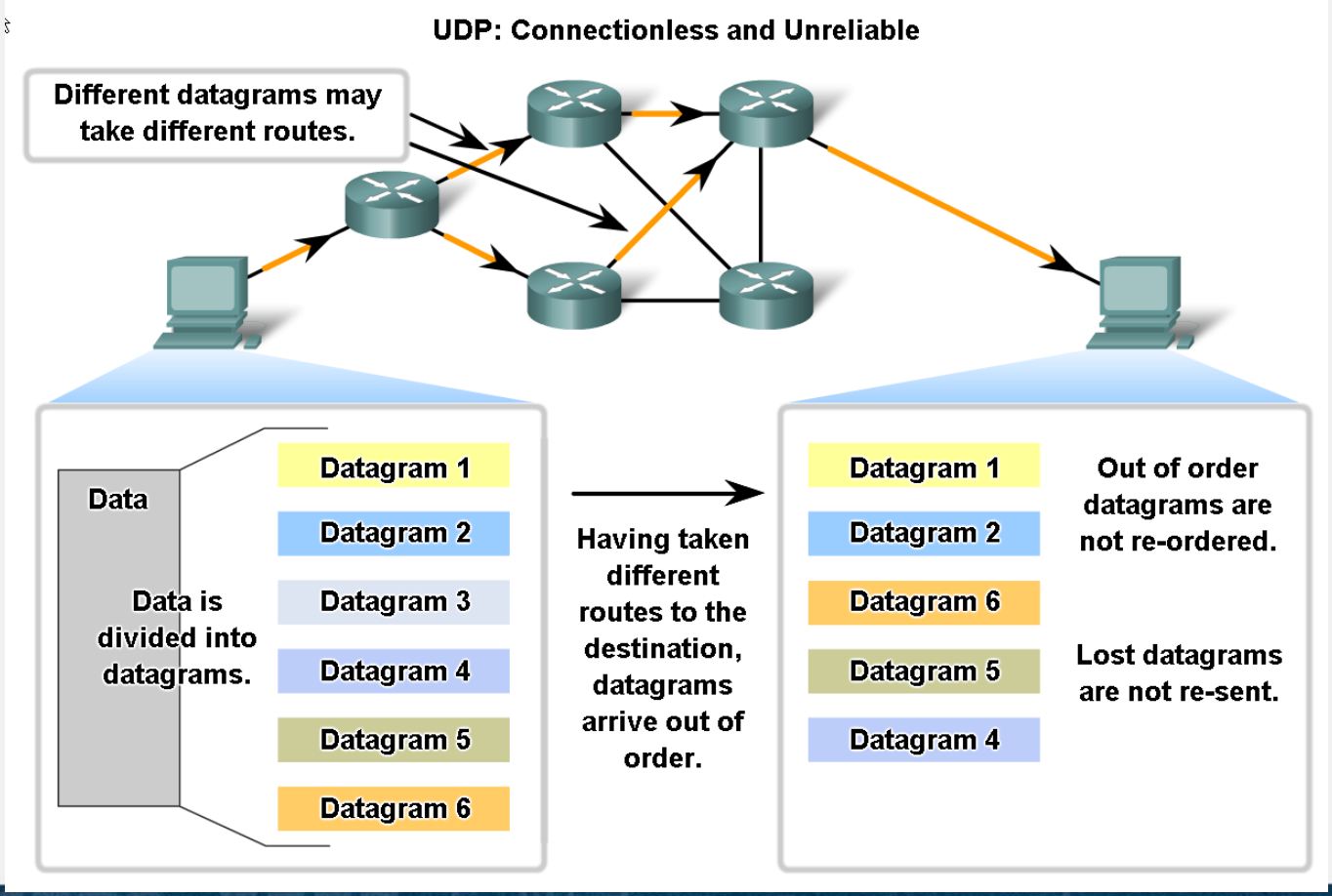
Protocolo UDP

- Não orientado à conexão



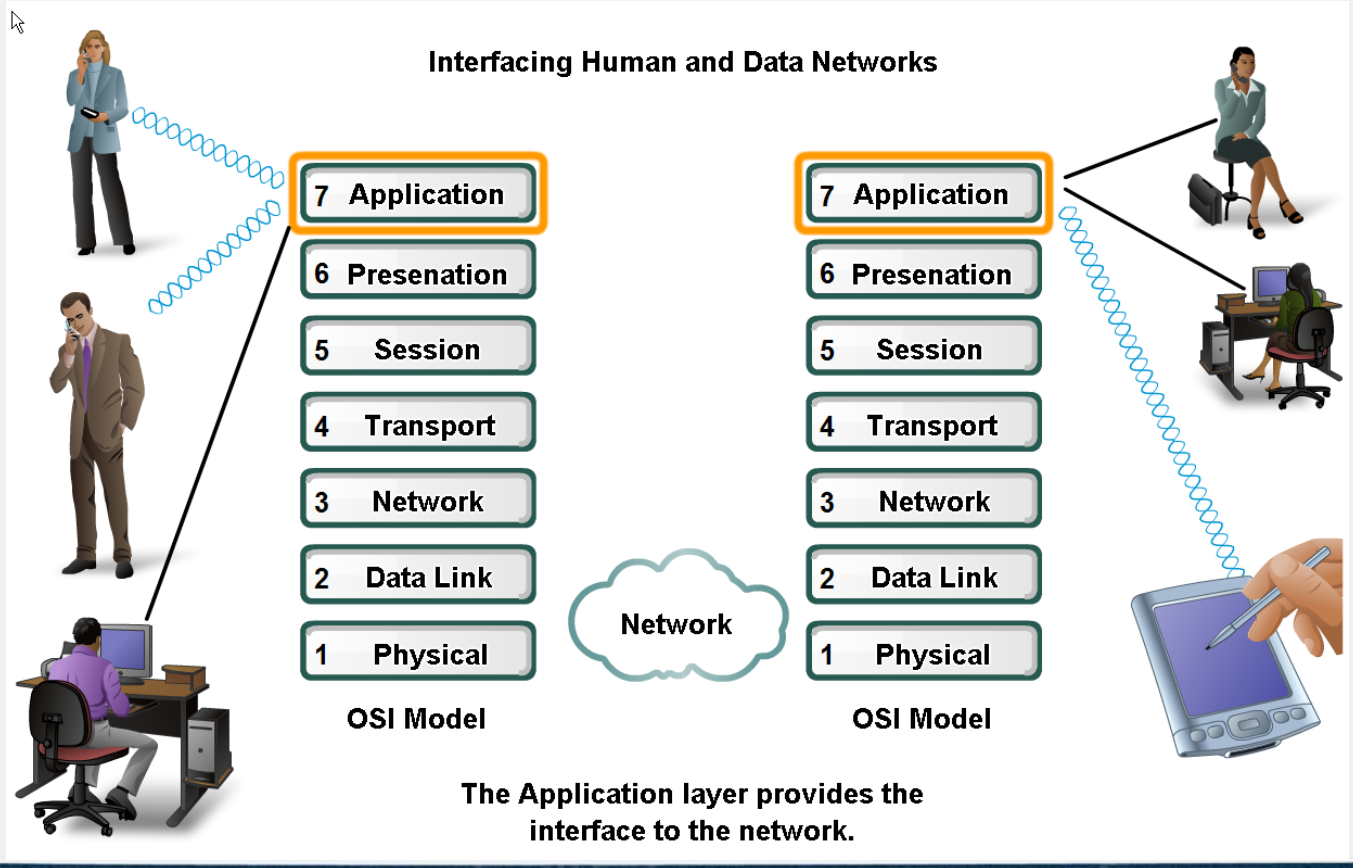
Protocolo UDP

- A informação é dividida em vários datagramas UDP, os quais são transmitidos de modo independente na rede



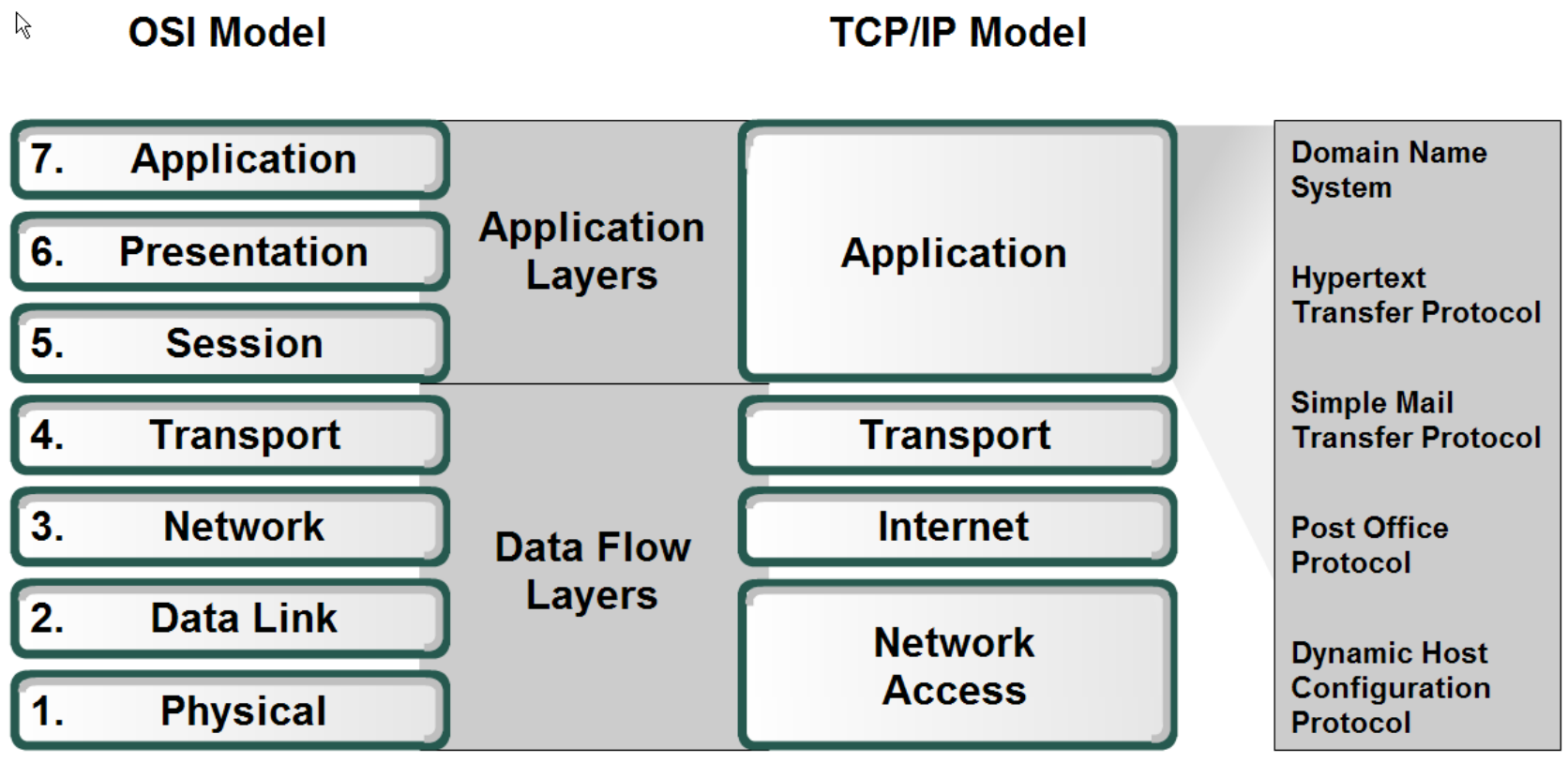
Camada de Aplicação

Camada de Aplicação: A interface entre a Rede e o Usuário

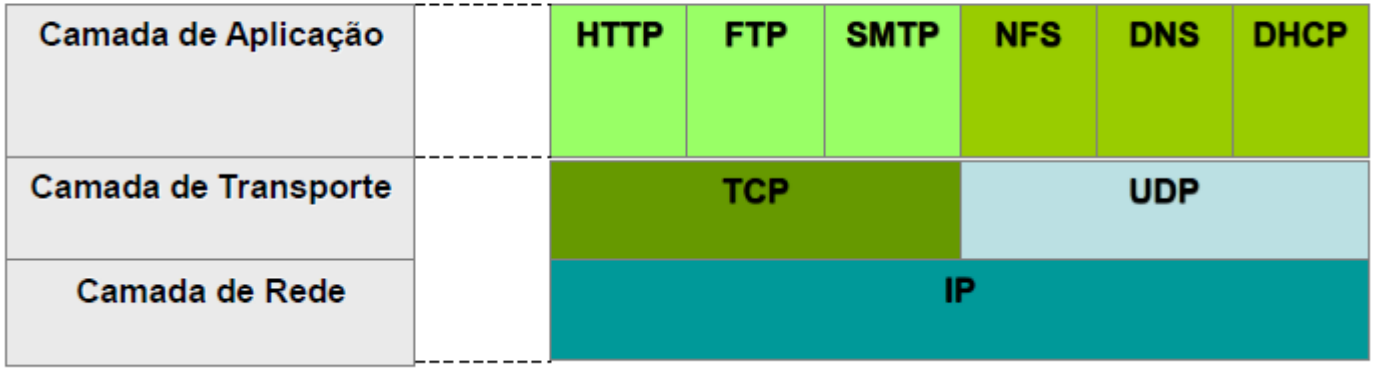


Camada de Aplicação: A interface entre a Rede e o Usuário

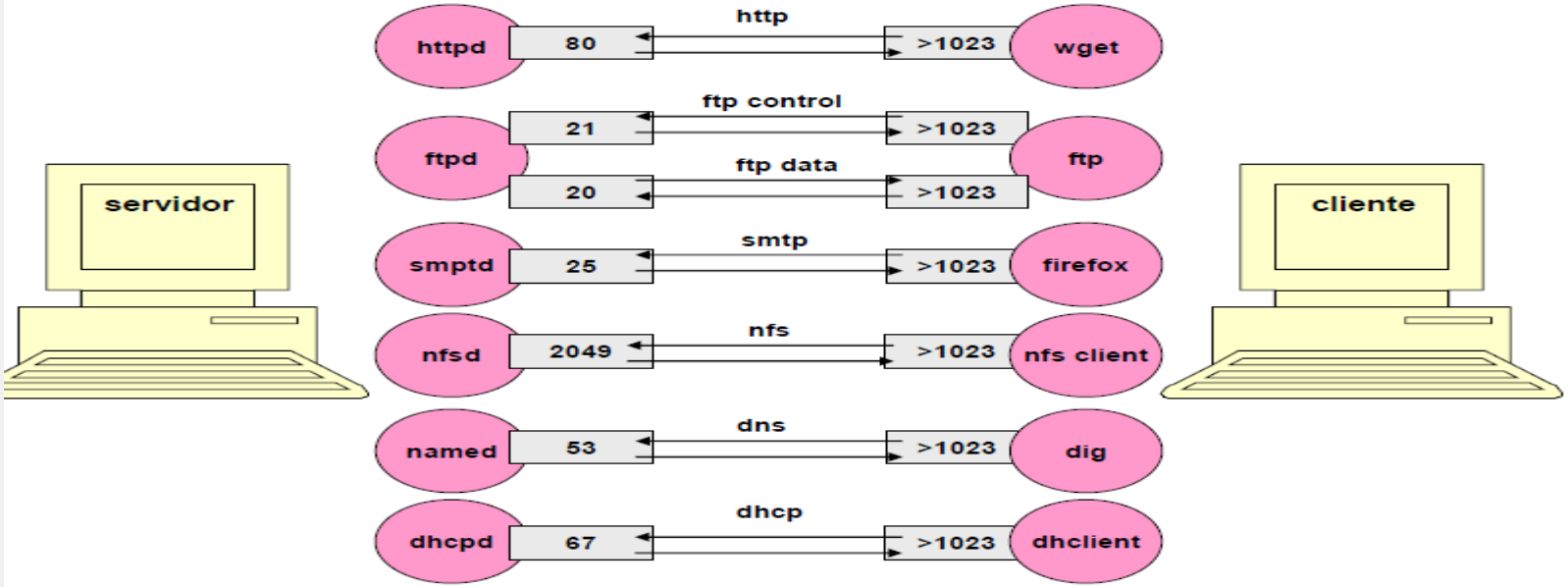
- Serviços e seus protocolos



Protocolos de Aplicação

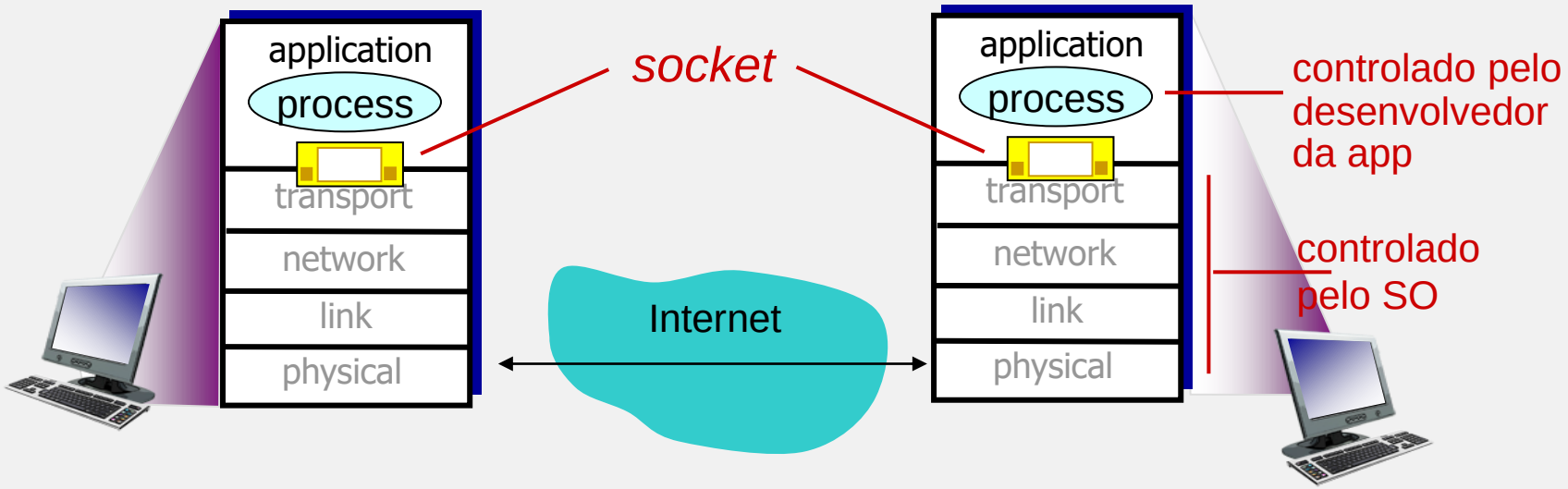


Portas bem Conhecidas



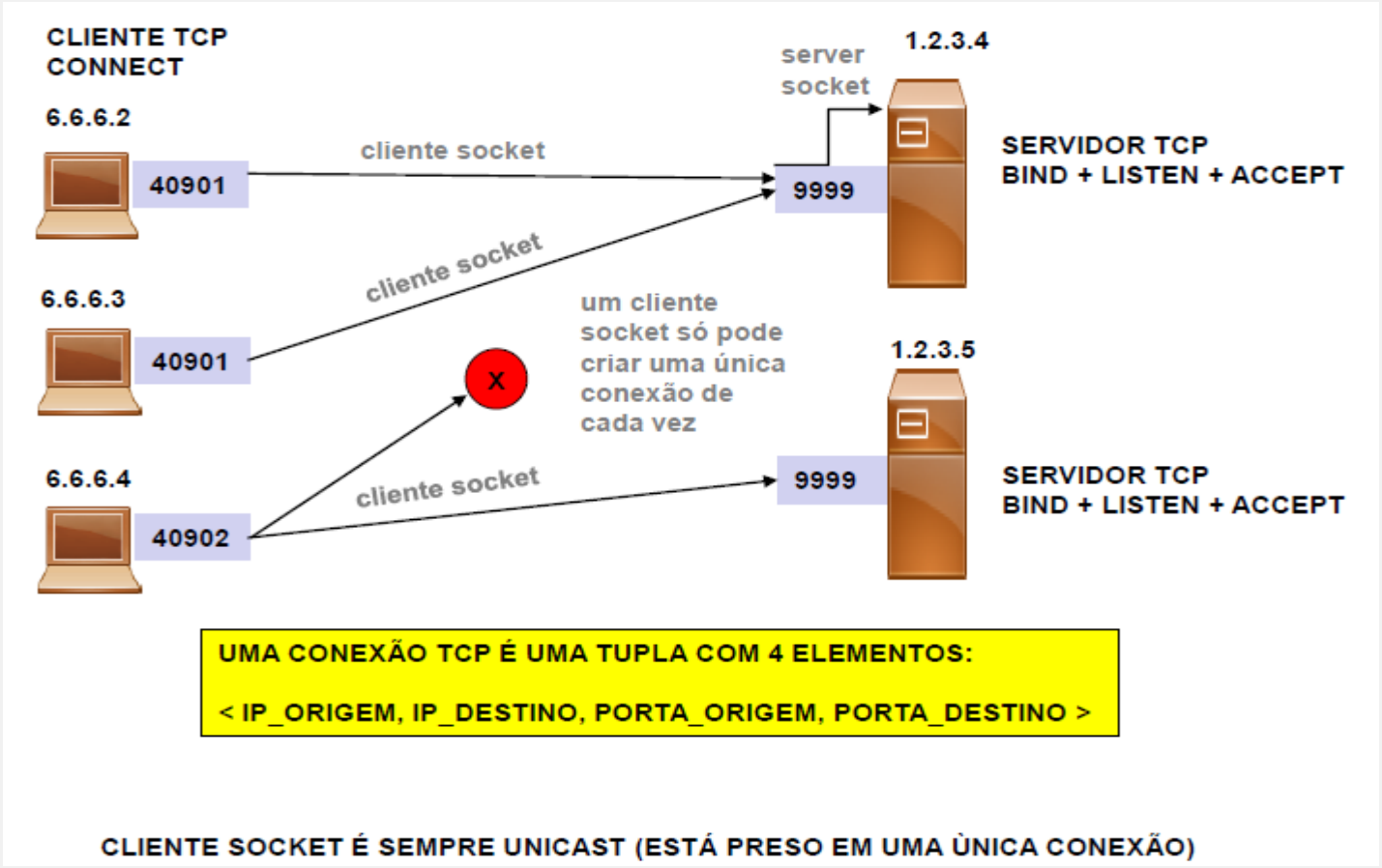
Desenvolvimento de Aplicações

socket: ”porta” entre o processo de uma aplicação e o protocolo de transporte fim-a-fim

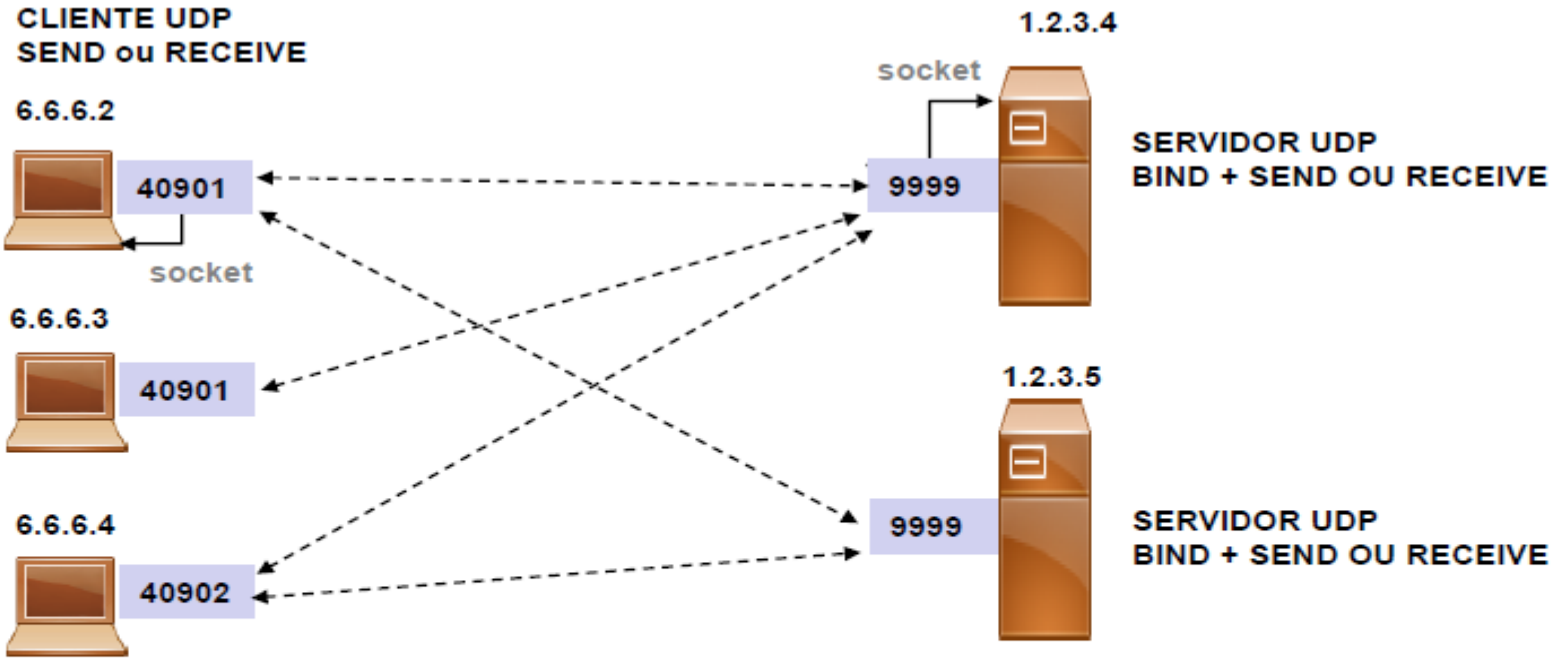


Desenvolvimento de Aplicações

socket: ”porta” entre o processo de uma aplicação e o protocolo de transporte fim-a-fim



Desenvolvimento de Aplicações



UMA FLUXO UDP É UMA TUPLA COM 4 ELEMENTOS:

< IP_ORIGEM, IP_DESTINO, PORTA_ORIGEM, PORTA_DESTINO >

UDP SOCKET PODE TRANSMITIR PARA VÀRIOS ENDEREÇOS UNICAST, BROADCAST E ENDEREÇOS DE GRUPO (MULTICAST)

Programação Socket

Dois tipos:

- *UDP:* datagrama sem comunicação confiável
- *TCP:* confiável, orientado a stream de bytes

Exemplo de aplicação:

1. O Cliente lê uma linha de caracteres (dados) do teclado e envia os dados ao servidor.
2. O servidor recebe os dados e converte os dados para maiúsculo.
3. O servidor envia os dados modificados ao cliente.
4. O cliente recebe os dados modificados e mostra na sua tela.

Programação *com UDP*

UDP: sem "conexão" entre cliente & servidor

UDP: dados transmitidos podem ser perdidos ou recebidos fora de ordem

Do ponto de vista da app:

- O UDP fornece uma transferência de grupos de bytes ("datagramas") com serviço não confiável entre cliente e servidor

Client/server com UDP

servidor (rodando no serverIP)

criar socket, porta= x:

serverSocket = socket(AF_INET,SOCK_DGRAM)

ler datagrama de serverSocket

enviar resp usando serverSocket especificando o endereço do cliente, porta

cliente

criar o socket:

clientSocket = socket(AF_INET,SOCK_DGRAM)

criar o com o server IP e porta=x; enviar o datagrama via clientSocket

ler o datagrama de clientSocket

fechar clientSocket

Exemplo: cliente UDP

Python UDPClient

incluir biblioteca Python socket

```
from socket import *
serverName = 'hostname'
serverPort = 12000
clientSocket = socket(socket.AF_INET,
                      socket.SOCK_DGRAM)
```

criar socket UDP

ler entrada via teclado

```
message = raw_input('Entre com uma frase:')
```

mandar para o server

```
clientSocket.sendto(message,(serverName, serverPort))
```

ler resposta do server

```
modifiedMessage, serverAddress =
                      clientSocket.recvfrom(2048)
```

imprimir e fechar o socket

```
print modifiedMessage
clientSocket.close()
```

Exemplo: servidor UDP

Python UDPServer

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print "Servidor estah pronto"
while 1:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.upper()
    serverSocket.sendto(modifiedMessage, clientAddress)
```

criar socket UDP

associar o socket à porta 12000

loop

Lê msg do socket UDP e carrega em message

envia a msg modificada ao cliente