



# Redes de Computadores

## Parte V: Camada de Transporte

**Professor:** Reinaldo Gomes  
reinaldo@dsc.ufcg.edu.br



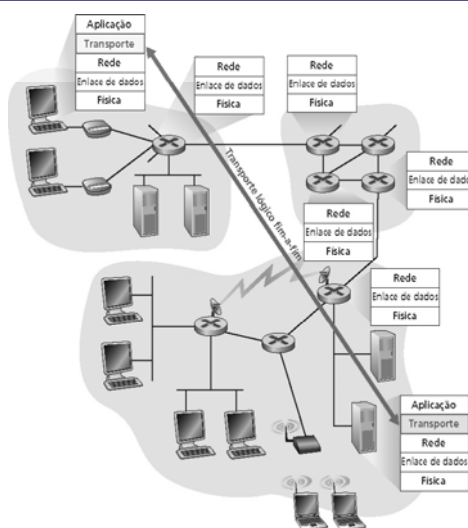
## Camada de transporte

- 3.1 Serviços da camada de transporte
- 3.2 Multiplexação e demultiplexação
- 3.3 Transporte não orientado à conexão: UDP
- 3.4 Princípios de transferência confiável de dados
- 3.5 Transporte orientado à conexão: TCP
  - Estrutura do segmento
  - Transferência confiável de dados
  - Controle de fluxo
  - Gerenciamento de conexão
- 3.6 Princípios de controle de congestionamento
- 3.7 Controle de congestionamento do TCP



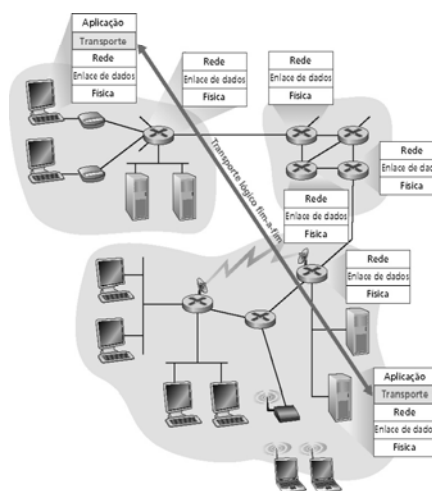
## Protocolos e serviços de transporte

- Fornecem **comunicação lógica entre processos** de aplicação em **diferentes hospedeiros**
- Os protocolos de transporte são **executados nos sistemas finais**
  - **Lado emissor**: quebra as mensagens da aplicação em segmentos e envia para a camada de rede
  - **Lado receptor**: remonta os segmentos em mensagens e passa para a camada de aplicação
- Há **mais de um protocolo de transporte** disponível para as aplicações
  - Internet: **TCP e UDP**



## Protocolos da camada de transporte da Internet

- **TCP**:
  - Confiável garante ordem de entrega
  - Controle de congestionamento
  - Controle de fluxo
  - Orientado à conexão
- **UDP**:
  - Não confiável, não garante ordem na entrega
  - Extensão do "melhor esforço" do IP
- **Serviços não disponíveis**:
  - Garantia a atrasos
  - Garantia de banda
  - Por quê?





## Principais funções

- Controle de erros
- Controle de fluxo
- Multiplexação de aplicações
- **Nem todas as camadas de transporte implementam o mesmo conjunto de serviços**
  - TCP - Controle de congestionamento
  - UDP - apenas multiplexação



## Camada de transporte

- 3.1 Serviços da camada de transporte
- 3.2 Multiplexação e demultiplexação
- 3.3 Transporte não orientado à conexão: UDP
- 3.4 Princípios de transferência confiável de dados
- 3.5 Transporte orientado à conexão: TCP
  - Estrutura do segmento
  - Transferência confiável de dados
  - Controle de fluxo
  - Gerenciamento de conexão
- 3.6 Princípios de controle de congestionamento
- 3.7 Controle de congestionamento do TCP

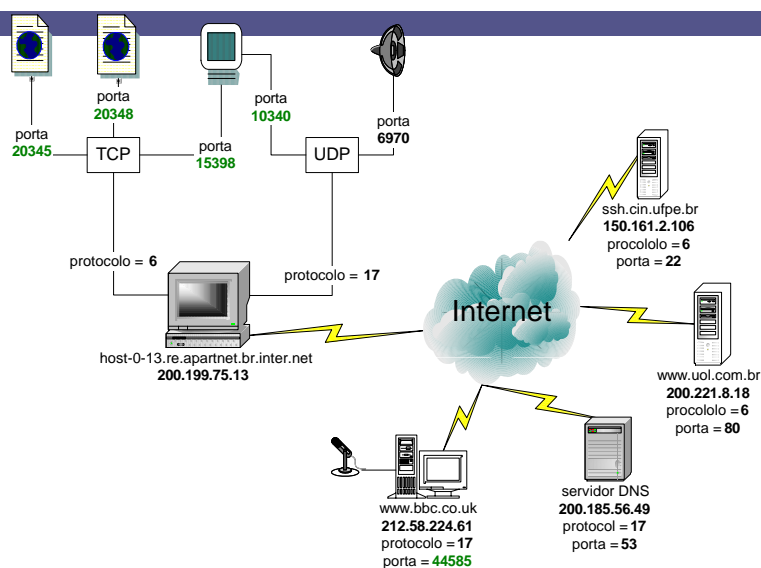


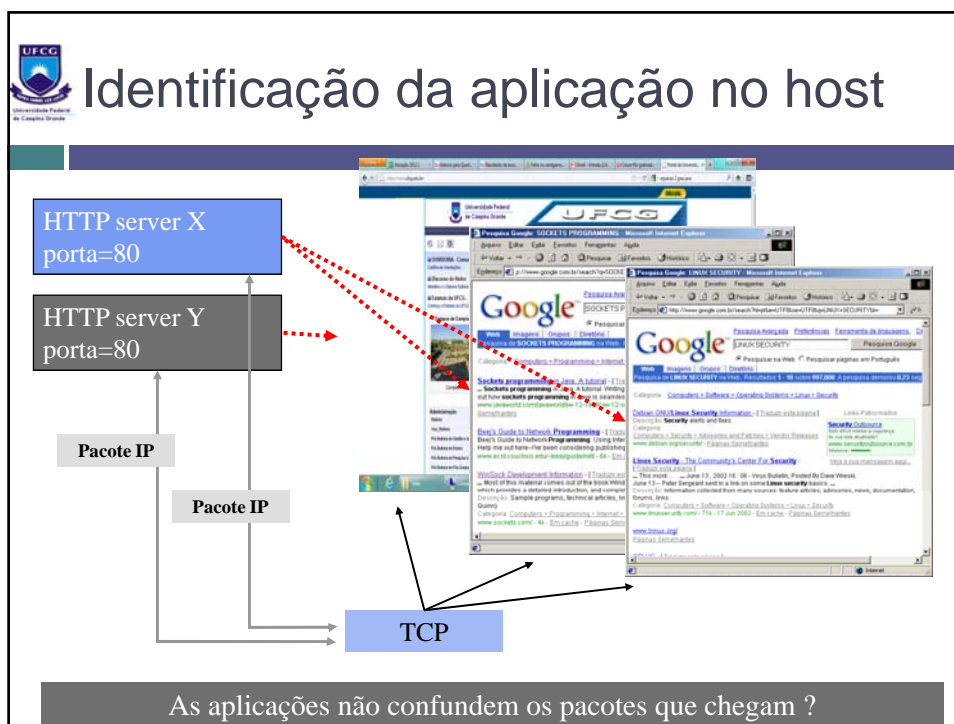
## Identificação da aplicação no host

- Como cada máquina é identificada unicamente na Internet?
- Como a entidade de rede (IP) identifica qual o protocolo de transporte está sendo utilizado ?
- Dentro do host, como a entidade de transporte (TCP,UDP) sabe para qual aplicação entregar o pacote ?
- Como uma aplicação do cliente sabe qual é a aplicação dentro do servidor remoto para poder enviar pacotes?

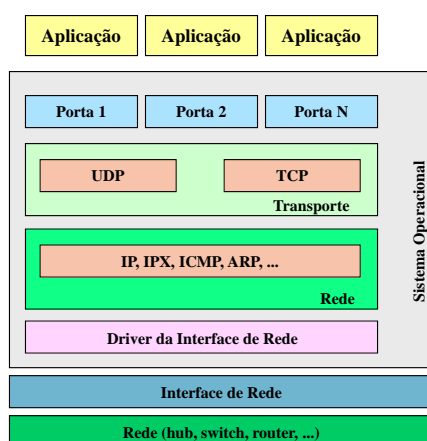


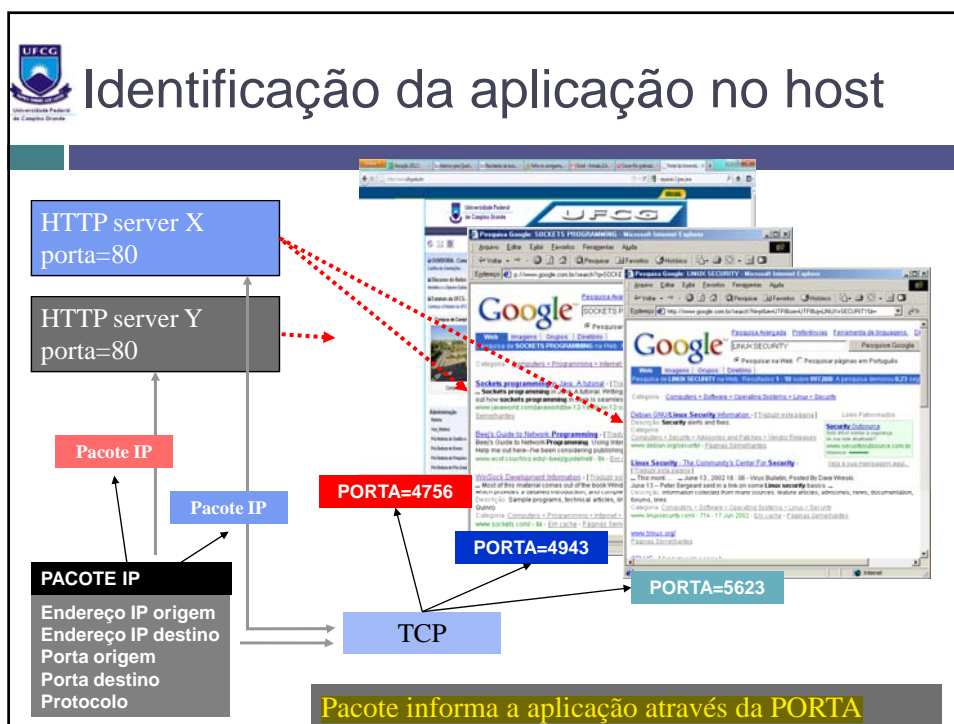
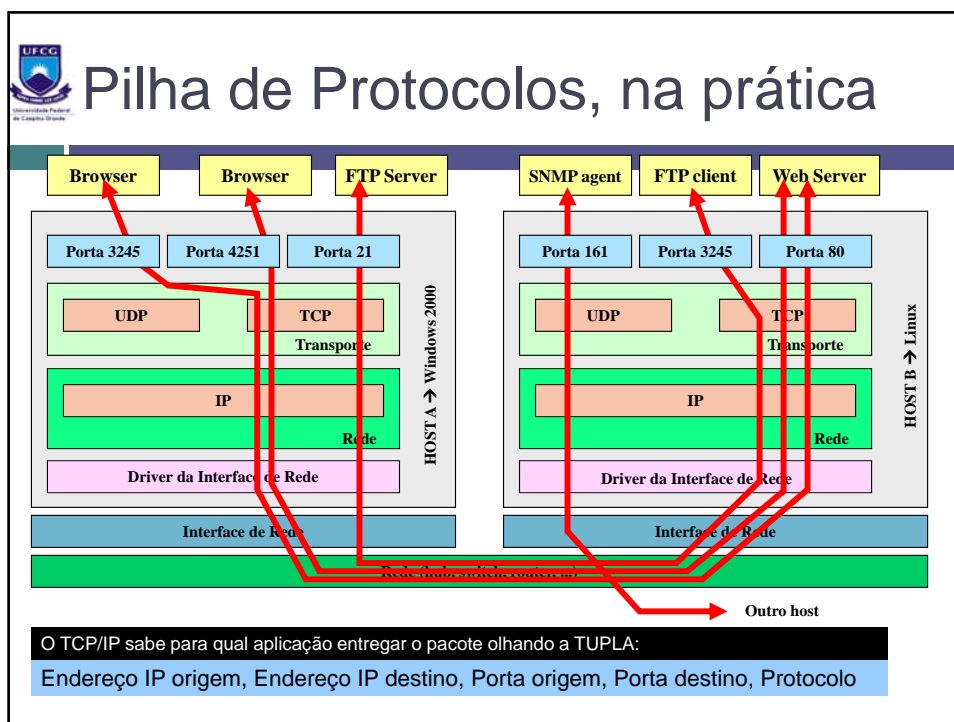
## Identificação da aplicação no host





## Pilha de Protocolos, na prática







## Identificação de aplicações

- Como cada máquina é identificada unicamente na Internet ?
  - Número IP
- Como a entidade de rede (IP) identifica qual o protocolo de transporte está sendo utilizado ?
  - Tipo de protocolo está indicado no cabeçalho IP
- Dentro do host, como a entidade de transporte identifica qual aplicação está sendo utilizada ?
  - Cada aplicação tem uma “Porta” única no host
  - Porta é identificada no pacote IP
- Como uma aplicação cliente sabe qual a porta de uma aplicação servidora para poder enviar pacotes?
  - Alguns serviços têm números de portas já convencionadas (portas “bem conhecidas”)



## Números de portas

- |             |   |
|-------------|---|
| □ 1-255     | reservadas para serviços padrão<br>portas “bem conhecidas”                |
| □ 256-1023  | reservado para serviços Unix  |
| □ 1-1023    | Somente podem ser usadas<br>por usuários privilegiados<br>(super-usuário) |
| □ 1024-4999 | Usadas por processos de<br>sistema e de usuário                           |
| □ 5000-     | Usadas somente por processos<br>de usuário                                |



## Algumas Portas Bem Conhecidas

- 21 → FTP
- 22 → SSH
- 23 → Telnet
- 25 → SMTP
- 53 → DNS
- 69 → TFTP
- 79 → Finger
- 80 → HTTP
- 88 → KERBEROS
- 110 → POP3
- 135-139 → NetBIOS Services
- 161-162 → SNMP
- 443 → HTTPS (HTTP+SSL)
- 995 → POP3S (POP3+SSL)
- 1433 → MS-SQL Server
- 2049 → NFS
- 3006 → MySQL
- 6000 → X Windows

Detalhes em [www.iana.org/assignments/port-numbers](http://www.iana.org/assignments/port-numbers)

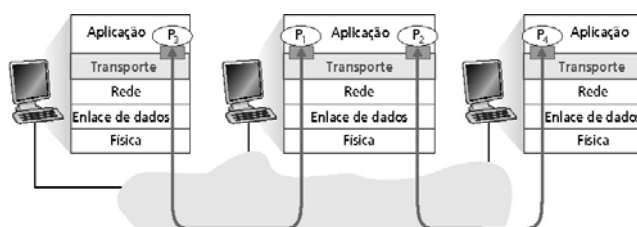


## Demultiplexação/Multiplexação

Demultiplexação no hospedeiro receptor Multiplexação no hospedeiro emissor:

entrega os segmentos  
recebidos ao socket correto

coleta dados de múltiplos sockets,  
envelopa os dados com cabeçalho  
(usado depois para  
demultiplexação)



Legenda:

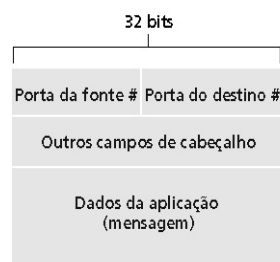
○ Processo ■ Socket





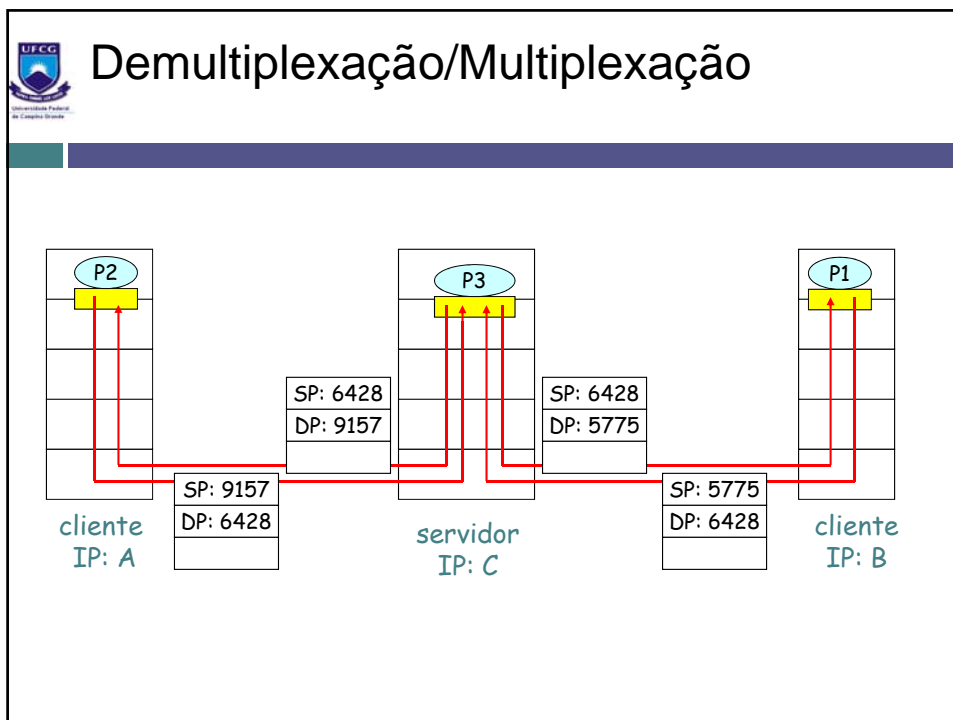
## Demultiplexação/Multiplexação

- Computador recebe datagramas IP
  - Cada datagrama possui endereço IP de origem e IP de destino
  - Cada datagrama carrega 1 segmento da camada de transporte
  - Cada segmento possui números de porta de origem e destino (lembre-se: números de porta bem conhecidos para aplicações específicas)
- O hospedeiro usa endereços IP e números de porta para direcionar o segmento ao socket apropriado



## Demultiplexação/Multiplexação

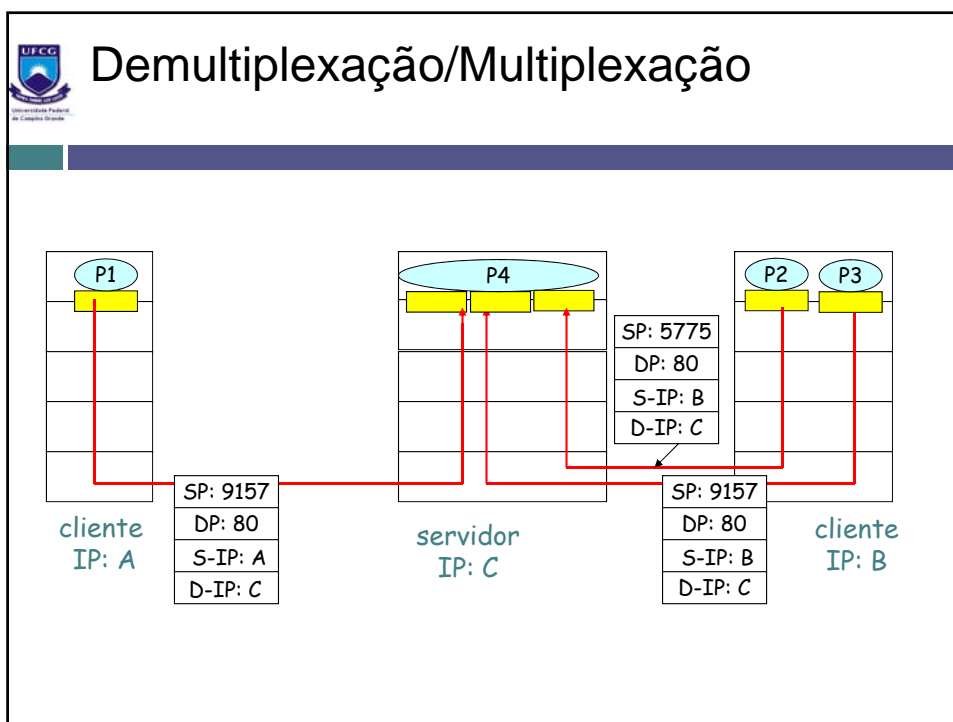
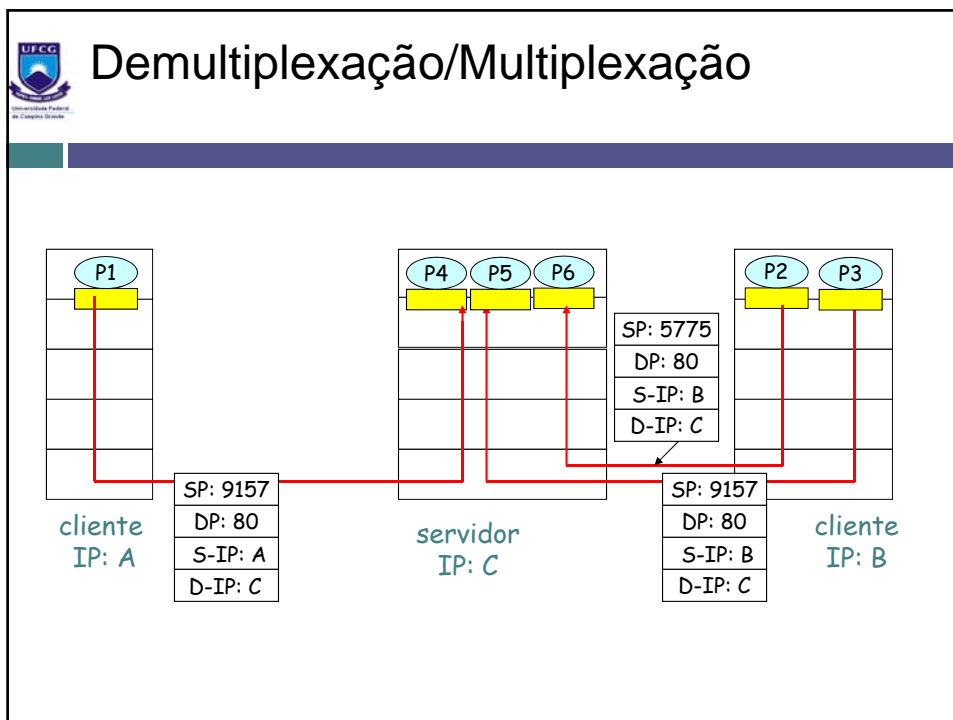
- Socket UDP identificado por dois valores: (endereço IP de destino, número da porta de destino)
- Quando o hospedeiro recebe o segmento UDP:
  - Verifica o número da porta de destino no segmento
  - Direciona o segmento UDP para o socket com este número de porta
- Datagramas com IP de origem diferentes e/ou portas de origem diferentes são direcionados para o mesmo socket



**UFCE**  
Universidade Federal  
de Campina Grande

## Demultiplexação/Multiplexação

- **Socket TCP** identificado por 4 valores:
  - Endereço IP de origem
  - End. porta de origem
  - Endereço IP de destino
  - End. porta de destino
- **Hospedeiro receptor** usa os quatro valores para direcionar o segmento ao socket apropriado
- **Hospedeiro servidor** pode suportar vários sockets TCP simultâneos:
  - Cada socket é identificado pelos seus próprios 4 valores
  - Servidores possuem sockets diferentes para cada cliente conectado





## Camada de transporte

- 3.1 Serviços da camada de transporte
- 3.2 Multiplexação e demultiplexação
- 3.3 Transporte não orientado à conexão: UDP
- 3.4 Princípios de transferência confiável de dados
- 3.5 Transporte orientado à conexão: TCP
  - Estrutura do segmento
  - Transferência confiável de dados
  - Controle de fluxo
  - Gerenciamento de conexão
- 3.6 Princípios de controle de congestionamento
- 3.7 Controle de congestionamento do TCP



## UDP: User Datagram Protocol

- Protocolo de transporte da Internet “sem gorduras”, “sem frescuras”
- Serviço “best effort”, **segmentos UDP** podem ser:
  - Perdidos
  - Entregues fora de ordem para a aplicação
- **Sem conexão:**
  - Não há apresentação entre o UDP transmissor e o receptor
  - **Cada segmento UDP** é tratado de forma independente dos outros

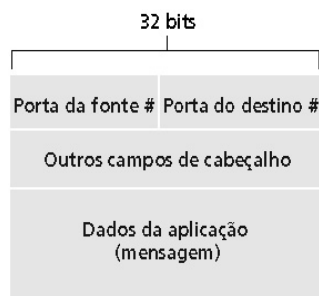
### Por que existe um UDP?

- Não há **estabelecimento de conexão** (que possa resultar em atrasos)
- Simples: **não há estado de conexão** nem no transmissor, nem no receptor
- **Cabeçalho de segmento reduzido**
- **Não há controle de congestionamento:** UDP pode enviar segmentos tão rápido quanto desejado (e possível)



# UDP: User Datagram Protocol

- Muito **usado** por aplicações de **multimídia contínua (streaming)**
  - Tolerantes à perda**
  - Sensíveis à taxa**
- Outros usos do UDP (por quê?):
  - DNS
  - SNMP
- Transferência confiável sobre UDP: **acrescentar confiabilidade na camada de aplicação**
  - Recuperação de erro específica de cada aplicação



## UDP Checksum

**Objetivo:** detectar "erros" (ex.: bits trocados) **no segmento transmitido**

**Transmissor:**

- Trata o **conteúdo do segmento como sequência de inteiros de 16 bits**
- Checksum: soma (complemento de 1 da soma) **do conteúdo do segmento**
- Transmissor **coloca o valor do checksum no campo de checksum do UDP**

**Receptor:**

- Computa o checksum do segmento recebido**
- Verifica** se o checksum **calculado é igual ao valor do campo** checksum:
  - NÃO - erro detectado
  - SIM - não há erros. Mas talvez haja erros apesar disso? Mas depois...



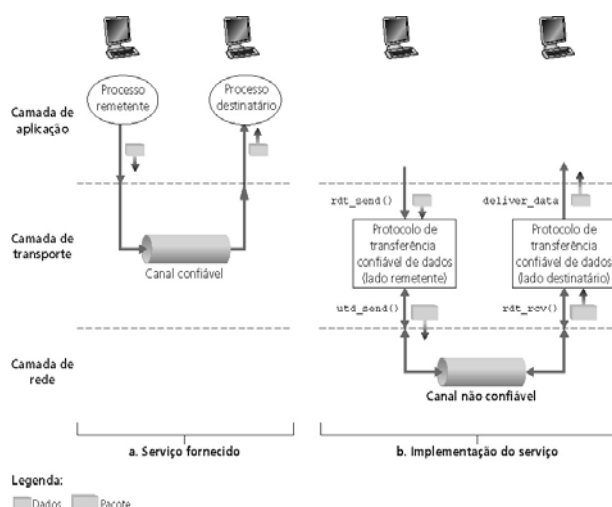
## Camada de transporte


- 3.1 Serviços da camada de transporte
- 3.2 Multiplexação e demultiplexação
- 3.3 Transporte não orientado à conexão: UDP
- 3.4 Princípios de transferência confiável de dados
- 3.5 Transporte orientado à conexão: TCP
  - Estrutura do segmento
  - Transferência confiável de dados
  - Controle de fluxo
  - Gerenciamento de conexão
- 3.6 Princípios de controle de congestionamento
- 3.7 Controle de congestionamento do TCP



## Princípios de transferência confiável de dados


- Importante nas camadas de aplicação, transporte e enlace
- Características dos canais não confiáveis determinarão a complexidade dos protocolos confiáveis de transferência de dados (rdt)





## Transferência confiável usando um canal com erro de bits

- Canal subjacente pode trocar valores dos bits num pacote
  - Checksum para detectar erros de bits
- A questão: **como recuperar esses erros:**
  - **Reconhecimentos (ACKs)**: receptor avisa explicitamente ao transmissor que o pacote foi **recebido corretamente**
  - **Reconhecimentos negativos (NAKs)**: receptor avisa explicitamente ao transmissor que o **pacote tem erros**
  - **Transmissor reenvia o pacote** quando da recepção de um NAK
- Mecanismos necessários:
  - **Detecção de erros**
  - Retorno do receptor: **mensagens de controle** (ACK, NAK) rcvr->sender



## Transferência confiável usando um canal com erro de bits e perdas

O que acontece se o **ACK/NAK é corrompido ou perdido?**

- **Transmissor não sabe o que aconteceu no receptor!**
- Transmissor deve **esperar durante um tempo** razoável pelo ACK e se não recebe-lo deve **retransmitir a informação**
  - Não pode apenas retransmitir: **possível duplicata**

**Tratando duplicatas:**

- **Transmissor acrescenta** número de sequência em cada pacote
- **Transmissor reenvia o último pacote** se ACK/NAK for **perdido**
- **Receptor descarta** (não passa para a aplicação) pacotes **duplicados**



## Estratégias de Retransmissão

- Conhecidos como algoritmos ou protocolos **Automatic Repeat Request (ARQ)**
- Questões de projeto:
  - ▣ Como o receptor requisita uma retransmissão?
  - ▣ Como a fonte sabe quando retransmitir?
- Desempenho e exatidão
- Para simplificar as explicações assumiremos comunicações do **tipo ponto-a-ponto**



## Estratégias de Retransmissão

Mesmas soluções adotadas pela camada de enlace:

- **Verificação de erros** (*checksum*)
- **Retransmissão**
- **Temporização**
- **Número de sequência**
- **Pipelining**
- **Janela deslizante**





## Camada de transporte

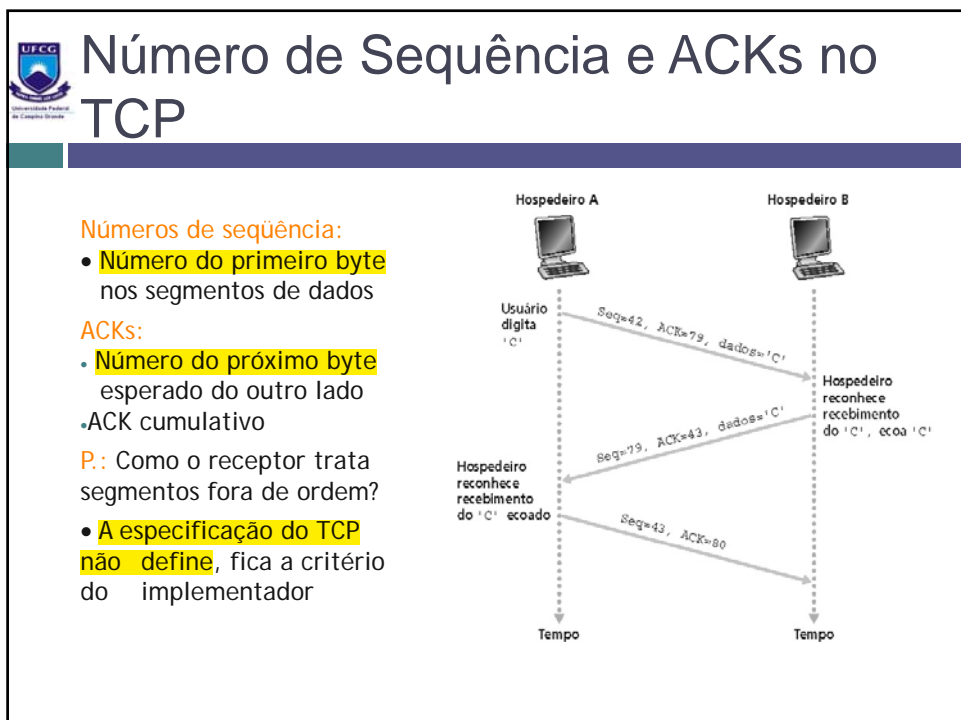
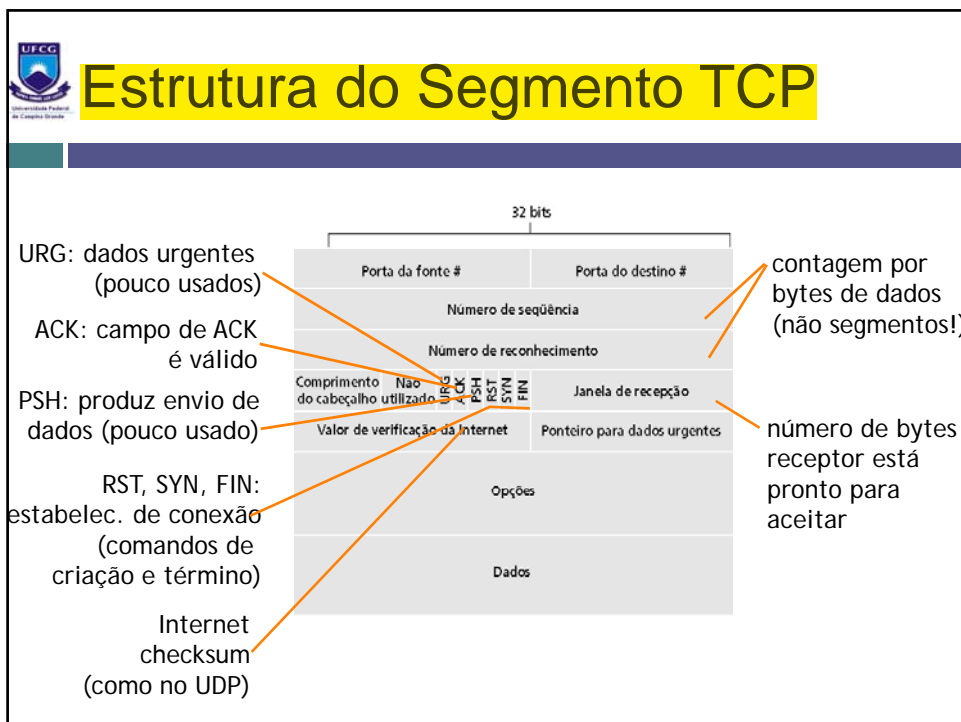
- Serviços da camada de transporte
- Multiplexação e demultiplexação
- Transporte não-orientado à conexão: UDP
- Princípios de transferência confiável de dados
- Transporte orientado à conexão: TCP
  - Estrutura do segmento
  - Transferência confiável de dados
  - Controle de fluxo
  - Gerenciamento de conexão
- Controle de congestionamento do TCP



## O Protocolo TCP

- **Ponto-a-ponto:**
  - Um transmissor, um receptor
- **Confiável, sequencial byte stream:**
  - Não há contornos de mensagens
- **Pipelined:** (transmissão de vários pacotes sem confirmação)
- **Controle de congestionamento e de fluxo** definem tamanho da janela
- **Buffers** de transmissão e de recepção
- **Dados full-duplex:**
  - Transmissão bidirecional na mesma conexão
- MSS: maximum segment size
- **Orientado à conexão:**
  - **Apresentação** (troca de mensagens de controle) inicia o estado do transmissor e do receptor **antes da troca de dados**
- **Controle de fluxo:**
  - Transmissor não esgota a capacidade do receptor







## TCP Round Trip Time e temporização

P.: como escolher o valor da temporização do TCP?

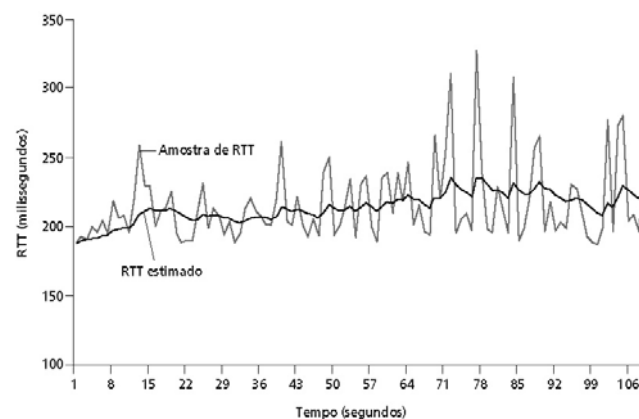
- Maior que o RTT
  - Nota: RTT varia
- Muito curto: temporização prematura
  - Retransmissões desnecessárias
- Muito longo: a reação à perda de segmento fica lenta


P.: Como estimar o RTT?

- **SampleRTT**: tempo medido da transmissão de um segmento até a respectiva confirmação
  - Ignora retransmissões e segmentos reconhecidos de forma cumulativa
- SampleRTT varia de forma rápida, é desejável um amortecedor para a estimativa do RTT
  - Usar várias medidas recentes, não apenas o último SampleRTT obtido




## Exemplos de estimativa do RTT



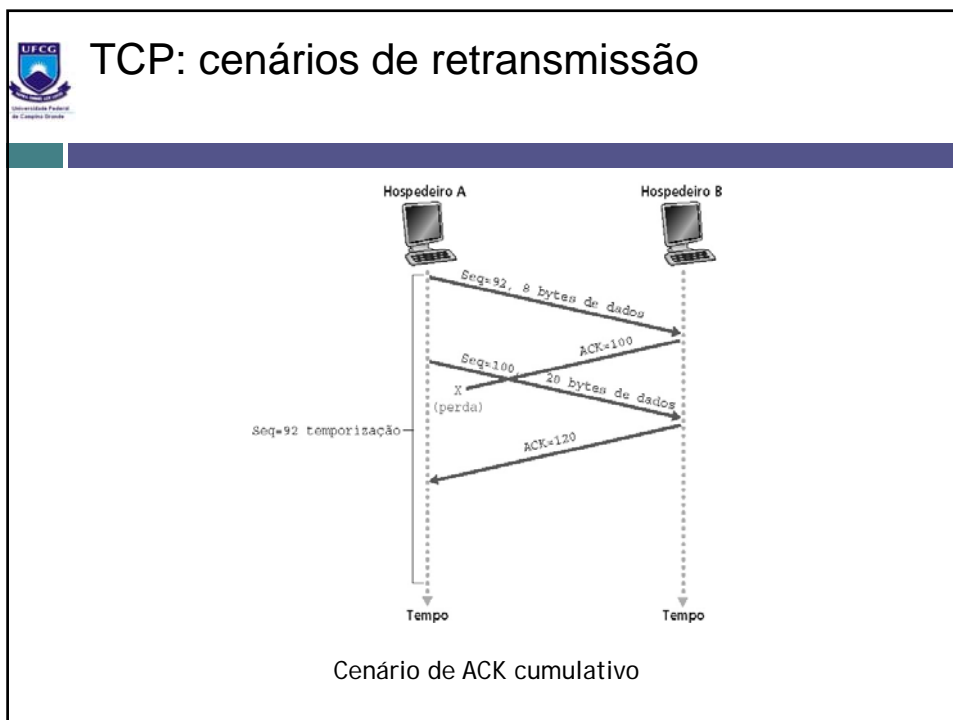
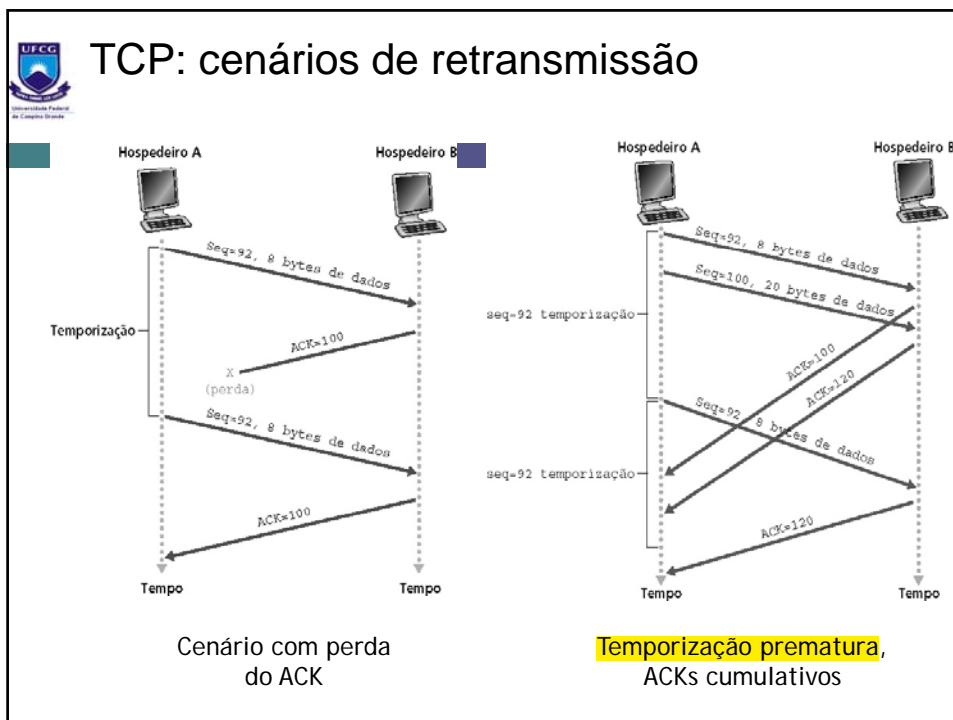


- Serviços da camada de transporte
- Multiplexação e demultiplexação
- Transporte não-orientado à conexão: UDP
- Princípios de transferência confiável de dados
- Transporte orientado à conexão: TCP
  - Estrutura do segmento
  - Transferência confiável de dados
  - Controle de fluxo
  - Gerenciamento de conexão
- Controle de congestionamento do TCP




## TCP: transferência de dados confiável

- TCP cria serviços de transferência confiável de dados em cima do serviço não-confiável do IP
- Transmissão de vários segmentos em paralelo (*Pipelined segments*)
- ACKs cumulativos
- TCP usa tempo de retransmissão simples
- Retransmissões são disparadas por:
  - Eventos de tempo de confirmação
  - ACKs duplicados




Evento no receptor	Ação do receptor TCP
Segmento chega em ordem, não há lacunas, segmentos anteriores já aceitos	ACK retardado. Espera até 500 ms pelo próximo segmento. Se não chegar, envia ACK
Segmento chega em ordem, não há lacunas, um ACK atrasado pendente	Imediatamente envia um ACK cumulativo
Segmento chega fora de ordem, número de sequência chegou maior: gap detectado	Envia ACK duplicado, indicando número de sequência do próximo byte esperado
Chegada de segmento que parcial ou completamente preenche o gap	Reconhece imediatamente se o segmento começa na borda inferior do gap

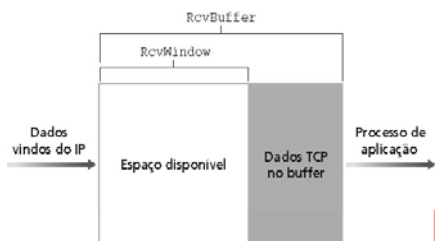
Retransmissão rápida
<ul style="list-style-type: none"> <li>• Com frequência, o tempo de expiração é relativamente longo:             <ul style="list-style-type: none"> <li>• Longo atraso antes de reenviar um pacote perdido</li> </ul> </li> <li>• Detecta segmentos perdidos por meio de ACKs duplicados             <ul style="list-style-type: none"> <li>• Transmissor frequentemente envia muitos segmentos</li> <li>• Se o segmento é perdido, haverá muitos ACKs duplicados</li> </ul> </li> <li>• Se o transmissor recebe 3 ACKs para o mesmo dado, ele supõe que o segmento após o dado confirmado foi perdido:             <ul style="list-style-type: none"> <li>• <b>Retransmissão rápida:</b> reenvia o segmento antes de o temporizador expirar</li> </ul> </li> </ul>

 Universidade Federal do Espírito Santo

- Serviços da camada de transporte
- Multiplexação e demultiplexação
- Transporte não orientado à conexão: UDP
- Princípios de transferência confiável de dados
- Transporte orientado à conexão: TCP
  - Estrutura do segmento
  - Transferência confiável de dados
  - Controle de fluxo
  - Gerenciamento de conexão
- Controle de congestionamento do TCP

 **TCP: controle de fluxo**

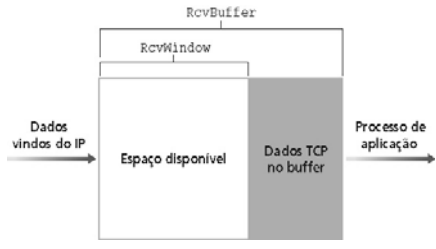
- Lado receptor da conexão TCP possui um buffer de recepção:



**Controle de fluxo**  
Transmissor **não deve esgotar** os buffers de recepção **enviando dados rápido demais**

- Serviço de **speed-matching**: **encontra a taxa de envio adequada à taxa de vazão da aplicação receptora**
- Processos de aplicação podem ser lentos para ler o buffer

**Controle de fluxo TCP: como funciona**



- Receptor informa a área disponível incluindo valor **RcvWindow** nos segmentos
- Transmissor limita os dados não confinados ao **RcvWindow**
- Garantia contra overflow no buffer do receptor

- Serviços da camada de transporte
- Multiplexação e demultiplexação
- Transporte não orientado à conexão: UDP
- Princípios de transferência confiável de dados
- Transporte orientado à conexão: TCP
  - Estrutura do segmento
  - Transferência confiável de dados
  - Controle de fluxo
  - Gerenciamento de conexão
- Controle de congestionamento do TCP





# O Protocolo TCP

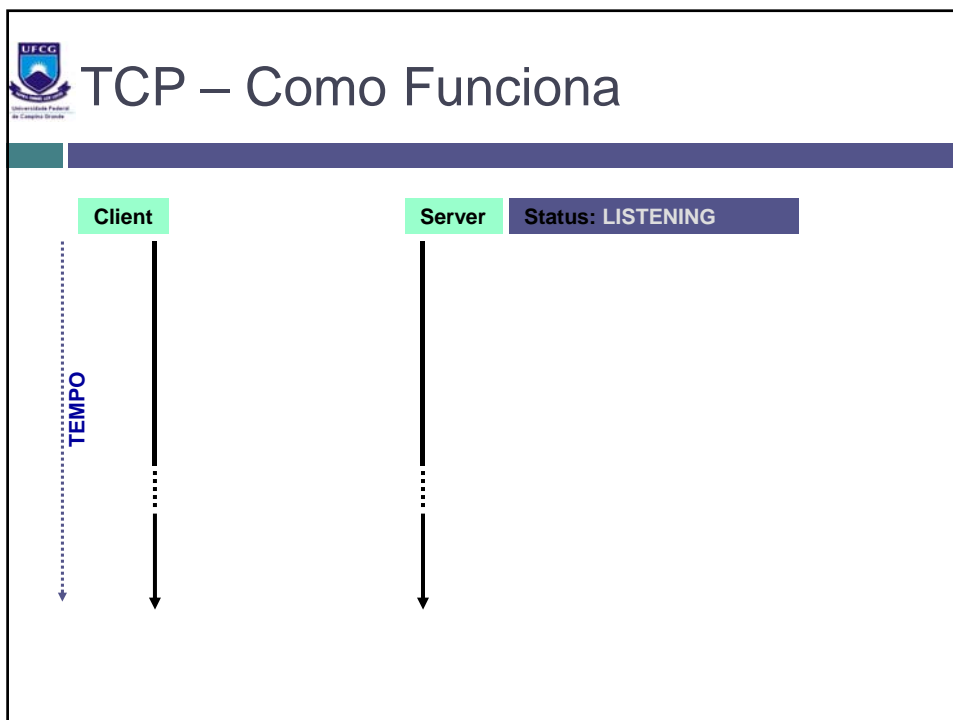
## □ Estabelecimento de Conexão

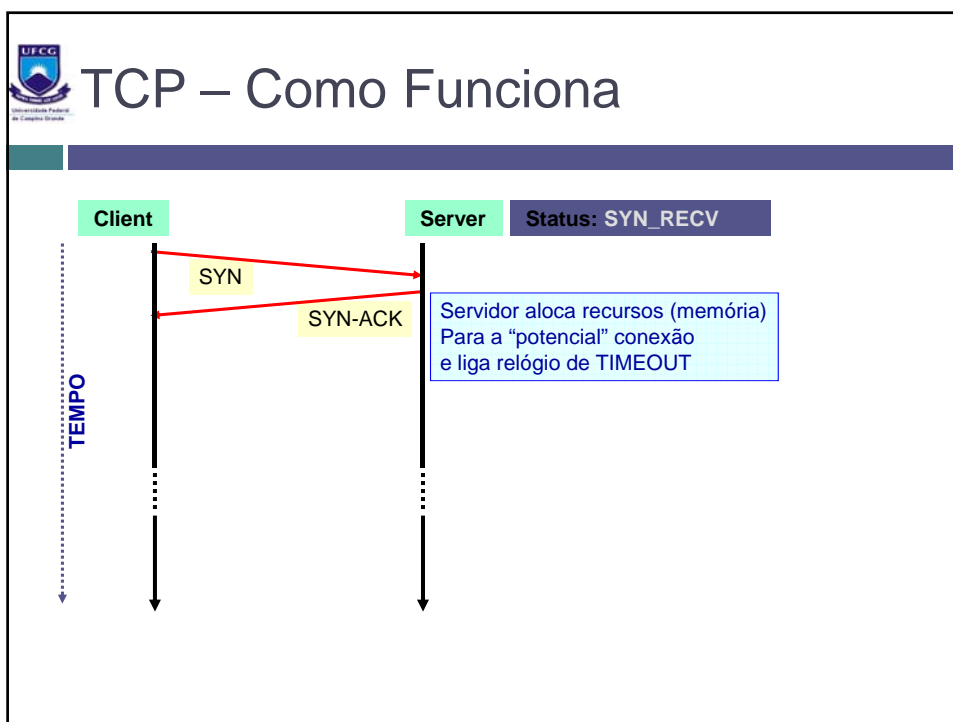
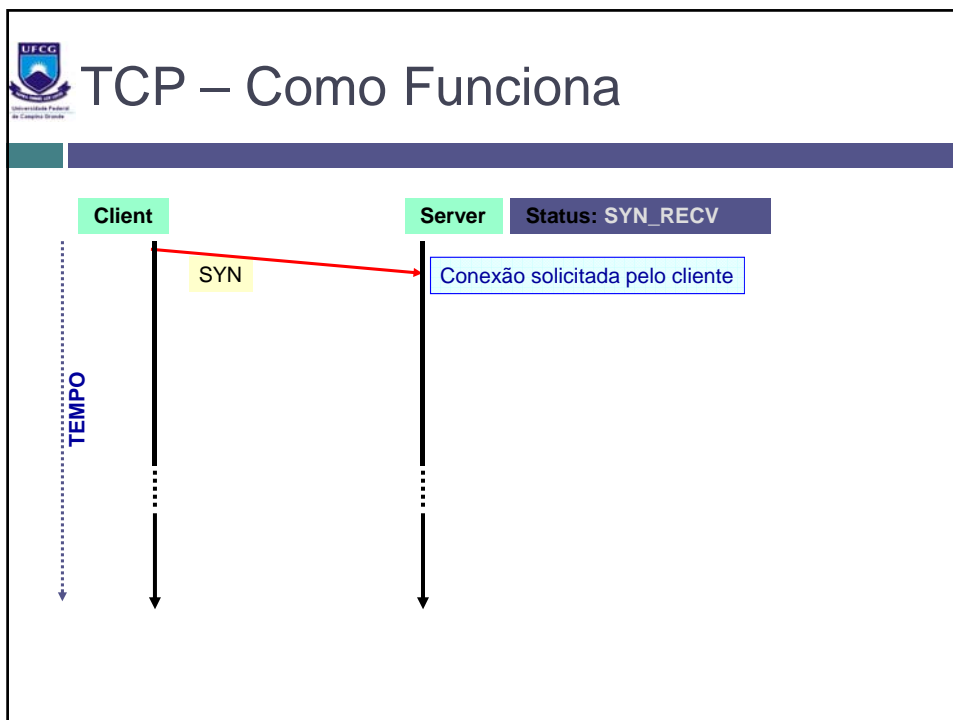
### ▣ Protocolo

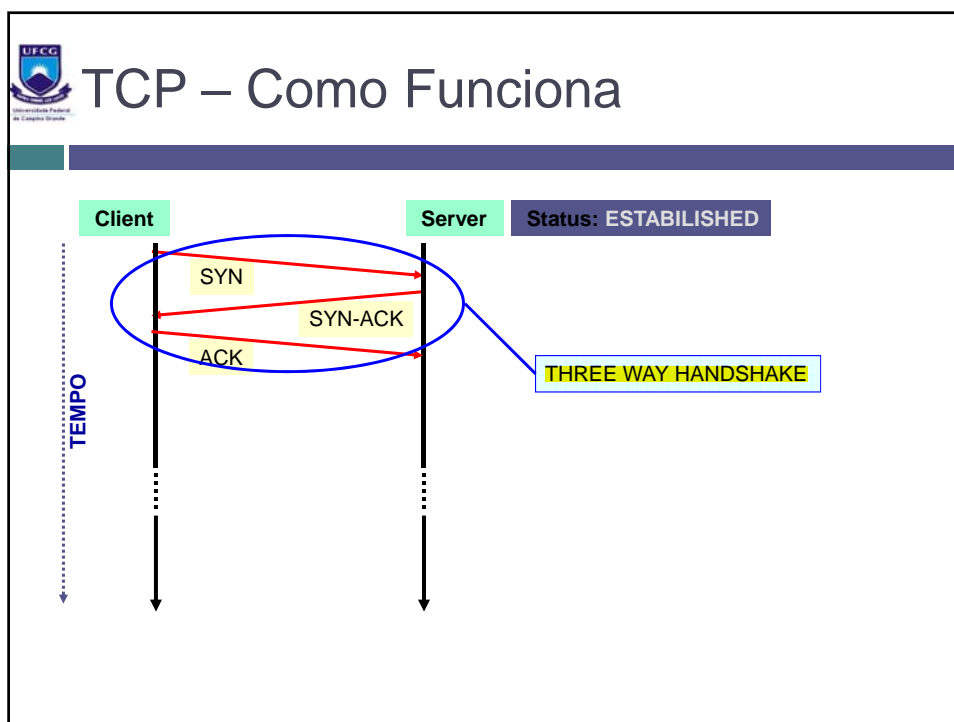
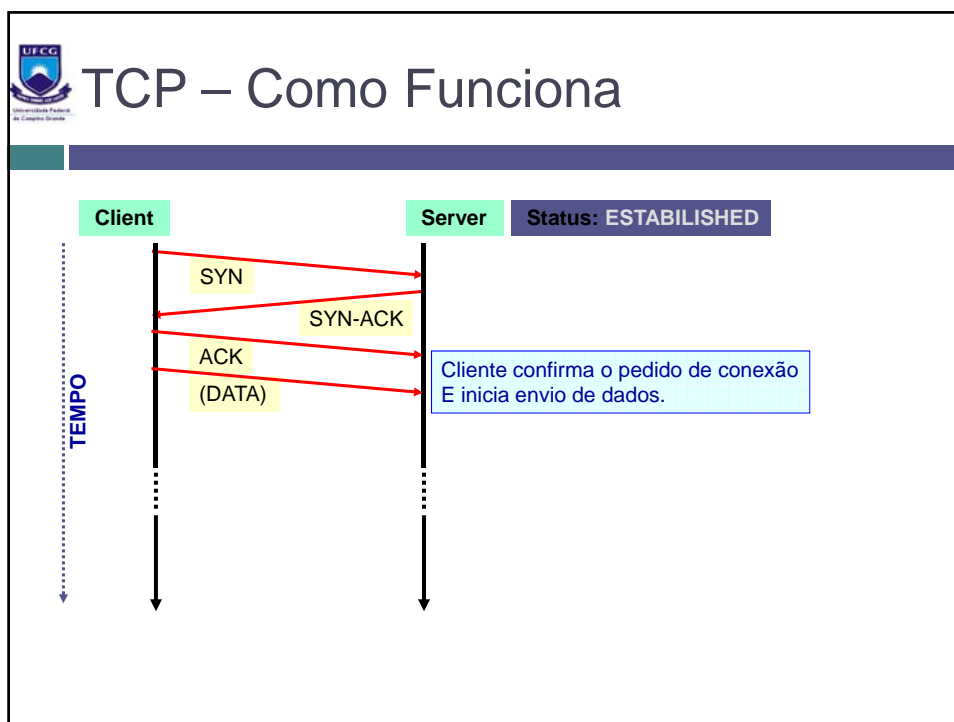
- Passo 1: **o cliente envia** um segmento SYN especificando a porta do servidor ao qual deseja se conectar e seu número de sequência inicial
- Passo 2: o **servidor responde enviando** outro segmento SYN com o ACK do segmento recebido e o seu próprio número de sequência
- Passo 3: **o cliente retorna** um ACK e a conexão se estabelece
- ▣ O tamanho máximo de segmento (MSS) que cada lado se propõe a aceitar também é definido no momento do estabelecimento da conexão
- ▣ Pode acontecer um “half open”



# TCP – Como Funciona



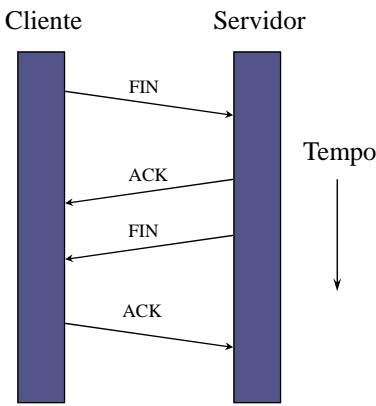




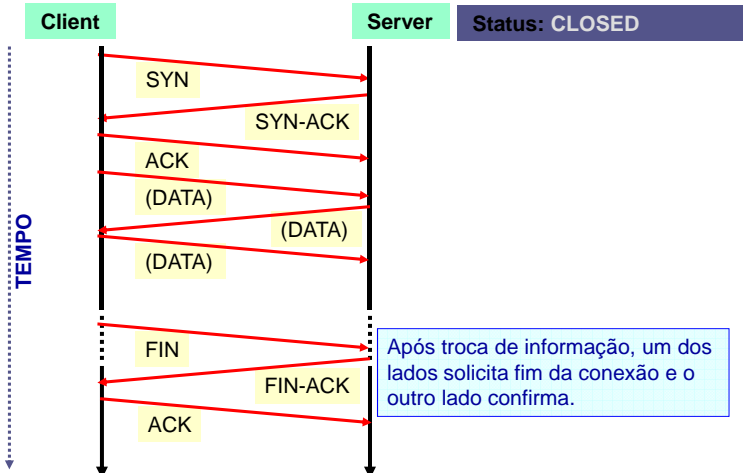
**O Protocolo TCP**

□ **Término de Conexão**

- ▣ Cada direção da conexão é encerrada independentemente
- ▣ Protocolo
  - Passo 1: o cliente envia um segmento FIN
  - Passo 2: o servidor retorna um FIN e um ACK para o cliente
  - Passo 3: o cliente envia um ACK e a conexão se encerra
- ▣ É possível efetuar um “half close”, mantendo-se apenas uma conexão simplex



**TCP – Como Funciona**



Client Server Status: CLOSED

TEMPO

SYN

SYN-ACK

ACK

(DATA)

(DATA)


(DATA)

FIN


FIN-ACK

ACK

Após troca de informação, um dos lados solicita fim da conexão e o outro lado confirma.



- Serviços da camada de transporte
- Multiplexação e demultiplexação
- Transporte não orientado à conexão: UDP
- Princípios de transferência confiável de dados
- Transporte orientado à conexão: TCP
  - Estrutura do segmento
  - Transferência confiável de dados
  - Controle de fluxo
  - Gerenciamento de conexão
- Controle de congestionamento do TCP



## Janela de Congestionamento

- Uma **conexão TCP controla** sua taxa de transmissão **limitando o seu número de segmentos que podem ser transmitidos sem que uma confirmação seja recebida**
- Esse número é chamado o **tamanho da janela** do TCP ( $w$ )
- Uma conexão TCP **começa com um pequeno valor de  $w$  e então o incrementa** arriscando que exista mais largura de banda disponível
- Isso continua a ocorrer **até que algum segmento seja perdido**
- Nesse momento, a **conexão TCP reduz  $w$  para um valor seguro**, e então continua a arriscar o crescimento



## Controle de Congestionamento

- O controle é feito através de duas variáveis adicionadas em cada lado da conexão:

- ▣ **Janela de Congestionamento**

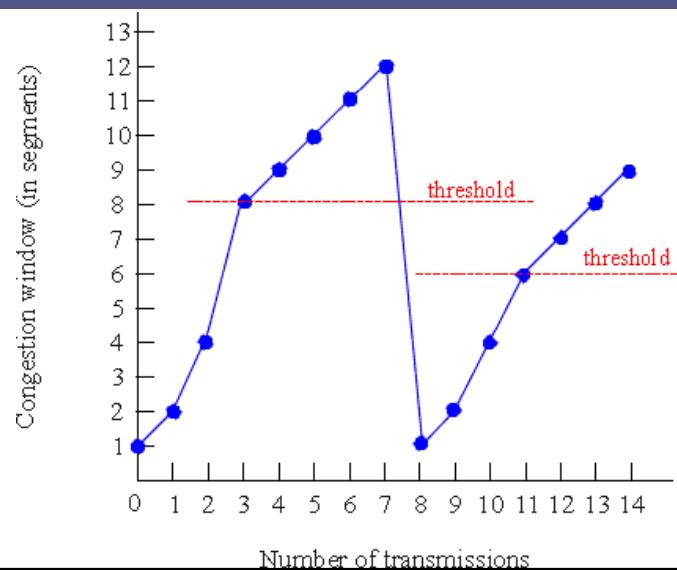
- Janela do TCP explicada anteriormente

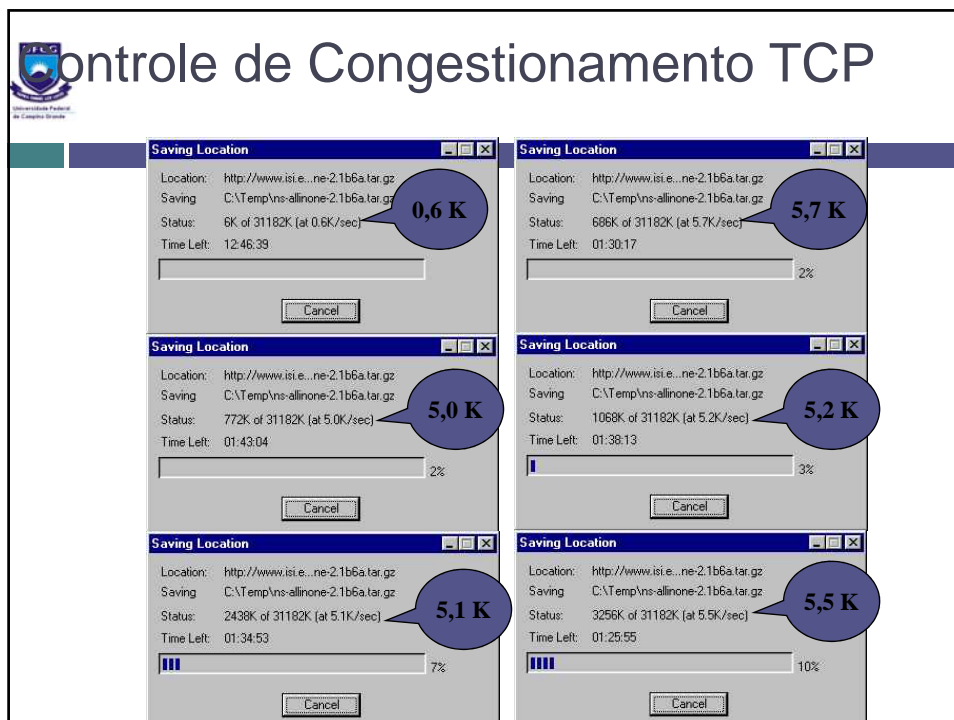
- ▣ **Limiar**

- Serve para controlar o crescimento da janela de congestionamento



## Graficamente ...





## Janela do Receptor

- O número máximo de segmentos não confirmados é dado pelo mínimo entre os tamanhos das janelas de congestionamento e do receptor.
  - ▣ *Ou seja, mesmo que haja mais largura de banda, o receptor também pode ser um gargalo.*



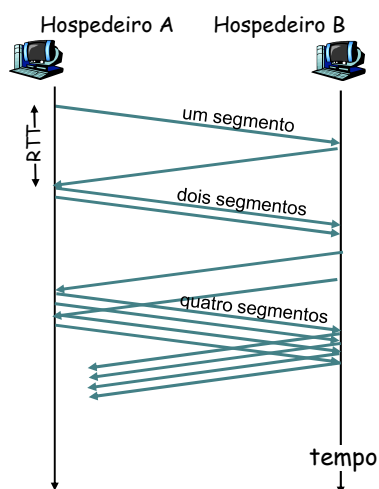
## Evolução de uma Conexão TCP

- No início, a janela de congestionamento **tem o tamanho de um segmento**.
  - ▣ Tal segmento tem o tamanho do maior segmento suportado.
- O primeiro **segmento é enviado e então é esperado seu reconhecimento**.
  - ▣ Se o mesmo chegar antes que ocorra o timeout, o transmissor **duplica** o tamanho da janela de congestionamento e envia **dois** segmentos.
  - ▣ Se esses dois segmentos também forem reconhecidos antes de seus timeouts, o transmissor duplica novamente sua janela, enviando agora **quatro** segmentos.



## Evolução de uma Conexão TCP

- Esse **processo continua até que**:
  - ▣ O tamanho da janela de congestionamento seja maior que o limiar, ou maior que o tamanho da janela do receptor;
  - ▣ Ocorra algum timeout antes da confirmação.





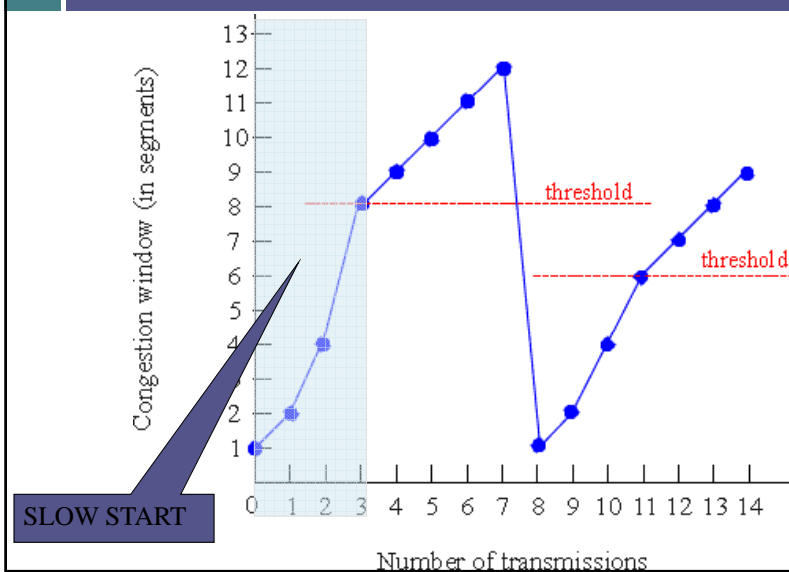


## Duas Fases dessa Evolução

- A primeira fase, em que a janela de congestionamento **cresce exponencialmente é chamada de inicialização lenta (slow start)**, pelo fato de começar com um segmento
  - ▣ **A taxa de transmissão começa pequena porém cresce muito rapidamente**



## Graficamente ...



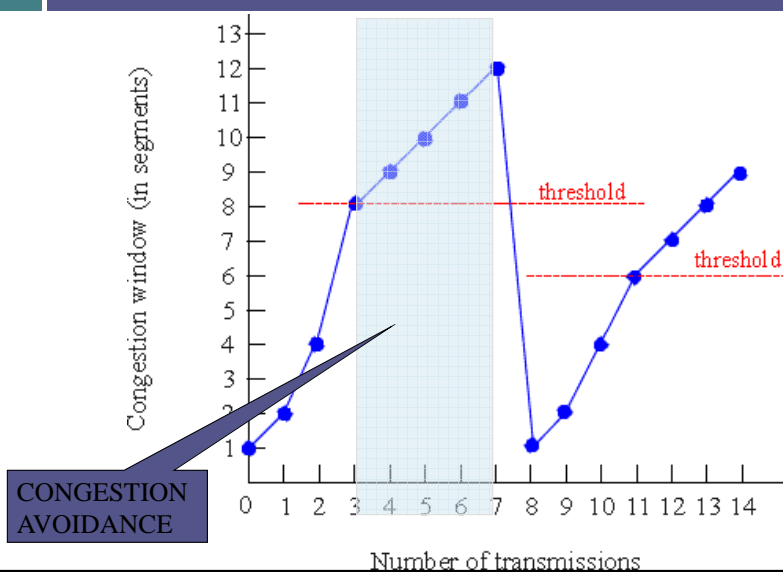



## Duas Fases dessa Evolução


- Uma vez ultrapassado o limiar, e a janela do receptor ainda não seja um limitante, o crescimento da janela passa a ser linear.
- Essa *segunda fase* é chamada de prevenção de congestionamento (congestion avoidance).
  - ▣ Sua duração também depende da não ocorrência timeouts, e da aceitação do fluxo por parte do receptor.




## Graficamente ...




**E quando ocorrer um problema?**




**Evolução de uma Conexão TCP**

- Na **ocorrência de um timeout** o TCP irá configurar:
  - ▣ O valor do limiar passa a ser a metade do tamanho atual da janela de congestionamento
  - ▣ O tamanho da janela de congestionamento volta ser do tamanho de um segmento
  - ▣ O tamanho da janela de congestionamento volta a crescer exponencialmente
- Caso **ocorram 3 ACKs duplicados**:
  - ▣ O valor do limiar é ajustado para metade tamanho atual da janela de congestionamento
  - ▣ O tamanho da janela de congestionamento passa igual ao valor do limiar (metade da janela de congestionamento atual)
  - ▣ O tamanho da janela de congestionamento cresce linearmente



## Resumo

- Quando o tamanho da janela de congestionamento está abaixo do limiar, seu crescimento é exponencial
- Quando este tamanho está acima do limiar, o crescimento é linear
- Todas as vezes que ocorrer um timeout, o limiar é modificado para a metade do tamanho da janela e o tamanho da janela passa a ser 1
  - A rede não consegue entregar nenhum dos pacotes (“congestionamento pesado”)
- Quando ocorrem ACKs repetidos a janela cai pela metade
  - A rede ainda é capaz de entregar alguns pacotes (“congestionamento leve”)



## Graficamente ...

