Camada de Transporte

Camadas de Transporte e Aplicação

PUC Minas Virtual

Agenda

- Introdução
- Primitivas para um Serviço de Transporte Simples
- *User Datagram Protocol* (UDP)
- *Transport Control Protocol* (TCP)

Introdução

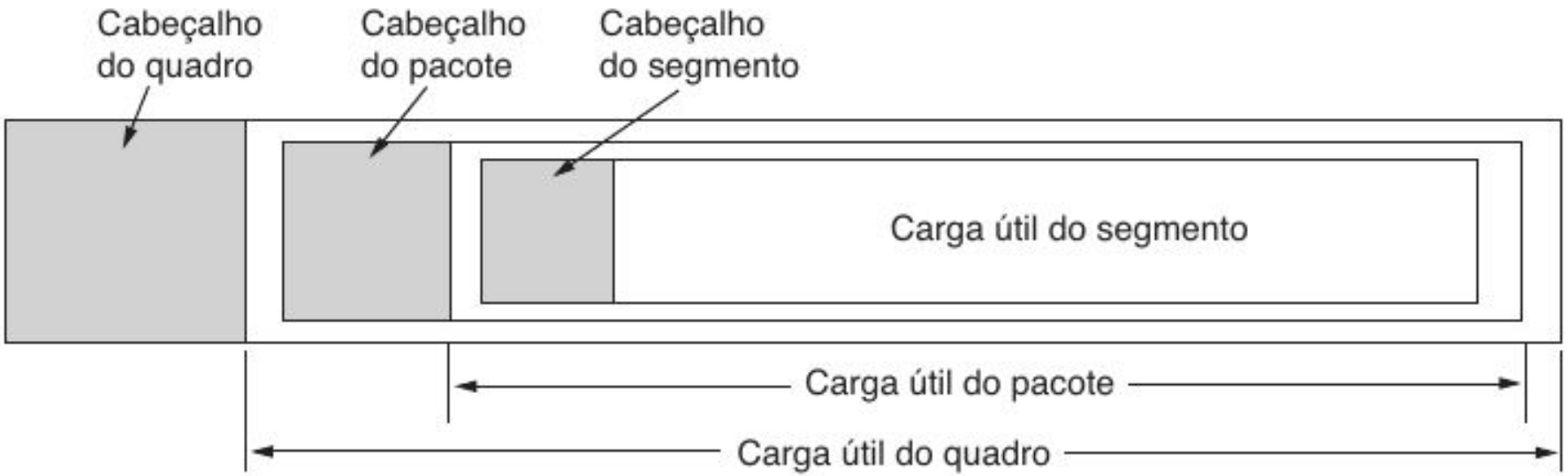
PUC Minas Virtual

Principais Protocolos de Transporte

- *User Datagram Protocol* (UDP)
  - Sem conexões
  - Transferência não confiável de dados

- *Transport Control Protocol* (TCP)
  - Orientado à conexão
  - Transferência confiável de dados
  - Controle de congestionamento
  - Controle de fluxo

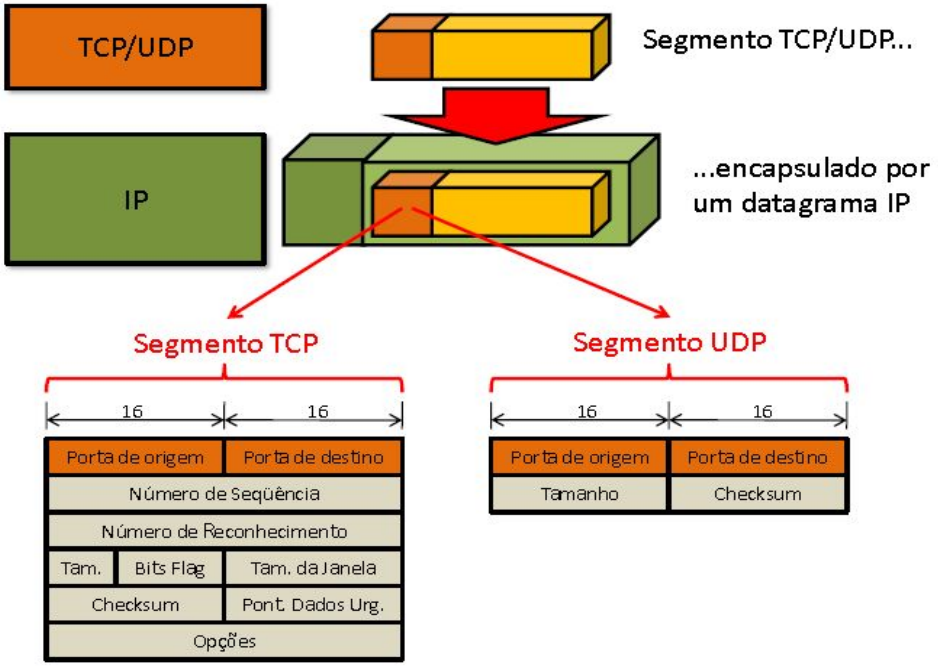
Segmento: Mensagem na Camada de Transporte



Porta

- Número (16 bits) que identifica um processo de transporte em uma máquina
- A porta é o endereço de transporte (e.g, na camada de rede da Internet, o número IP é o endereço de cada máquina na rede)

Formato do Cabeçalho no Segmento TCP/UDP



Números de Porta Bem Conhecidos

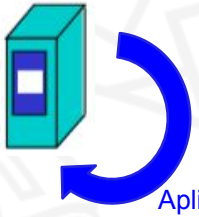
- Reservados para protocolos de aplicação tradicionais (entre 0 e 1023)

Porta	Descrição
20/TCP	FTP - porta de dados
21/TCP	FTP - porta de controle
22/TCP,UDP	SSH
23/TCP,UDP	Telnet
25/TCP,UDP	SMTP
53/TCP,UDP	DNS
80/TCP	HTTP
81/TCP	HTTP Alternativa
110/TCP	POP3
143/TCP,UDP	IMAP4
194/TCP	IRC
366/TCP,UDP	SMTP
989/TCP,UDP	FTP – porta de dados sobre TLS/SSL
990/TCP,UDP	FTP – porta de controle sobre TLS/SSL
993/TCP	IMAP4 sobre SSL
995/TCP	POP3 sobre SSL

Exemplo de Aplicação Telnet (Porta 23)

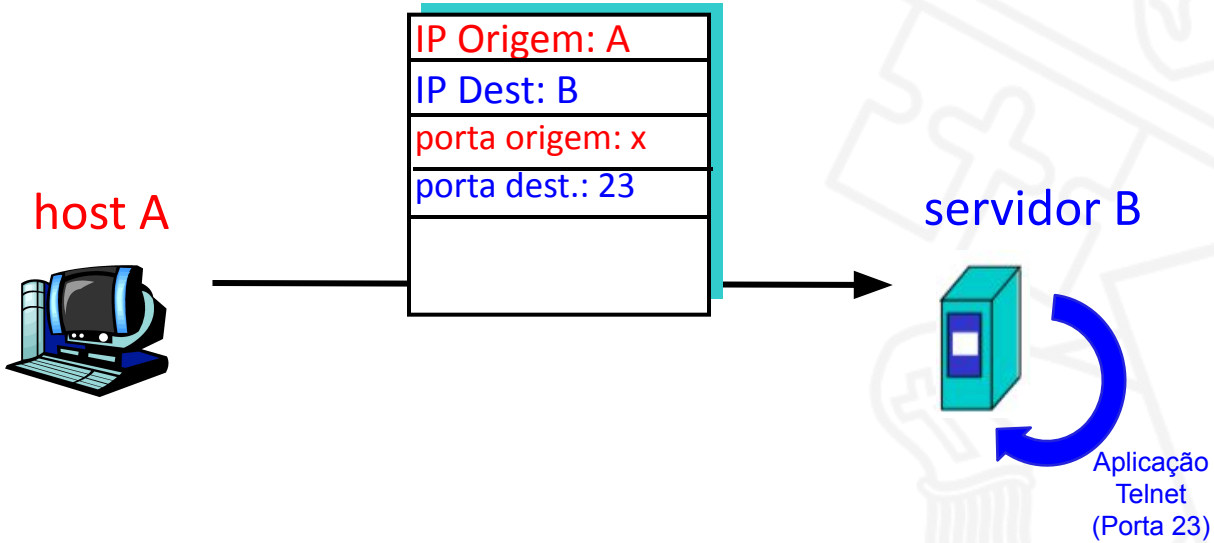
host A

servidor B

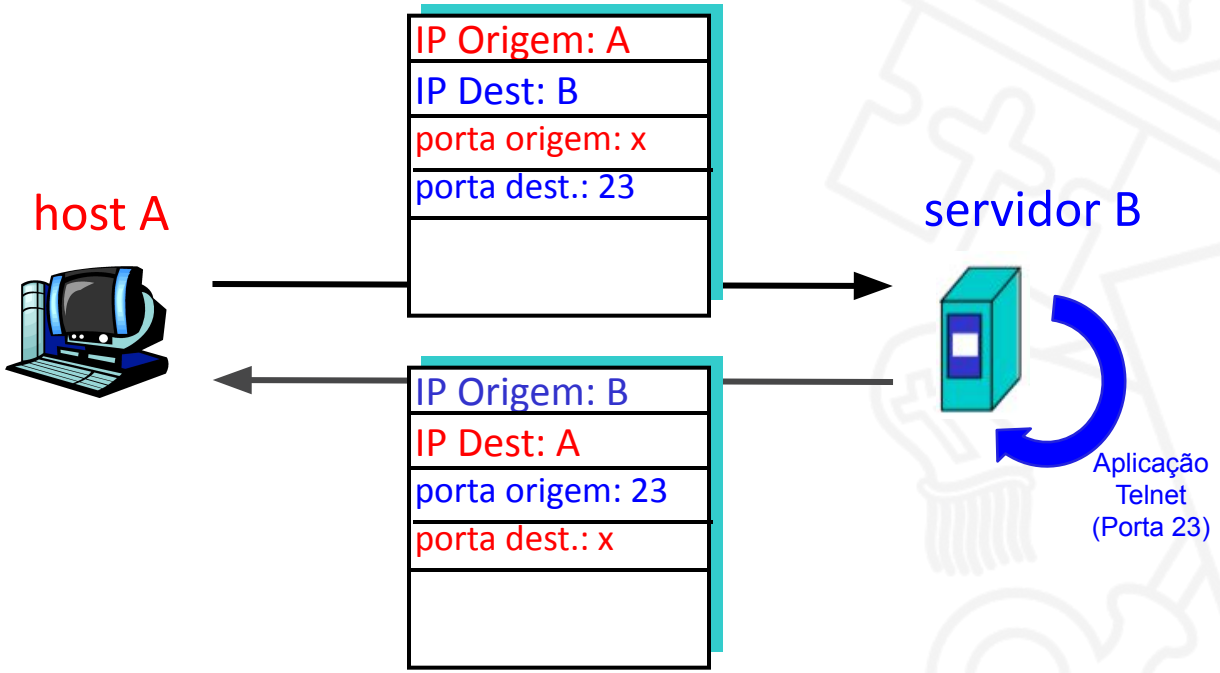


Aplicação
Telnet
(Porta 23)

Exemplo de Aplicação Telnet (Porta 23)



Exemplo de Aplicação Telnet (Porta 23)



Exemplo de Aplicação WEB (Porta 80)

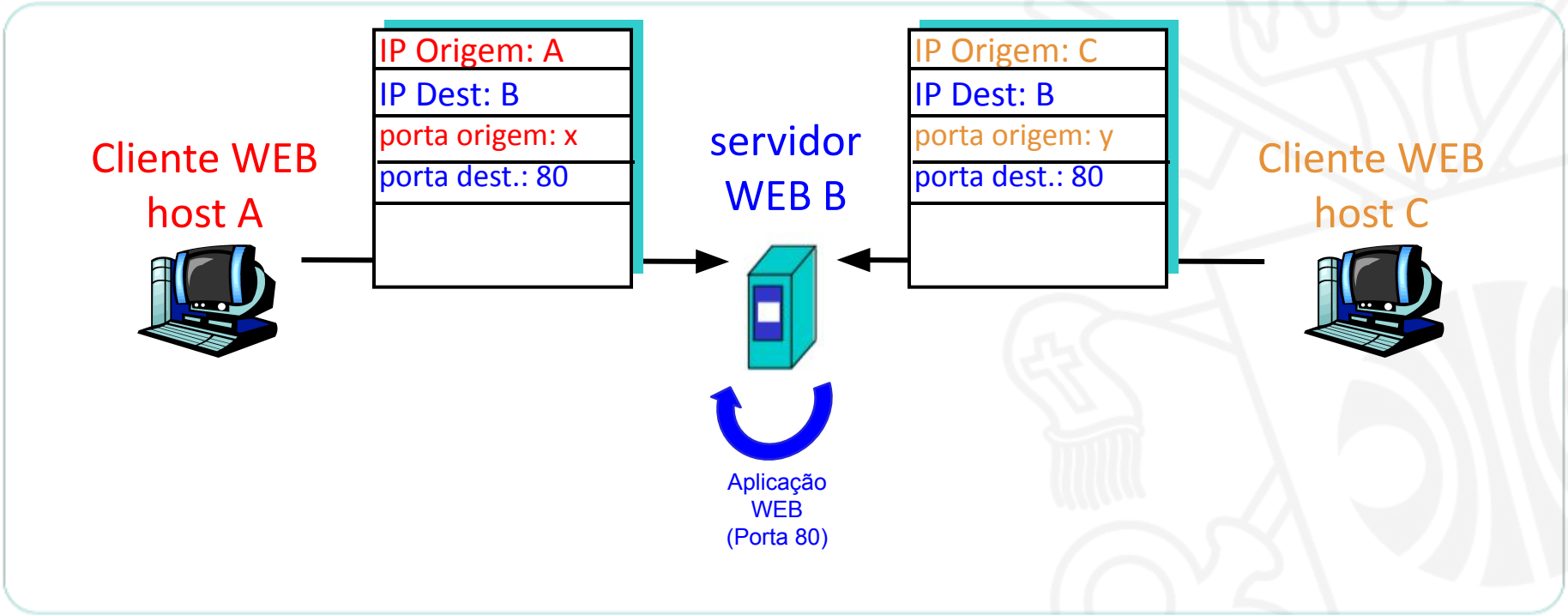
Cliente WEB
host A

servidor
WEB B

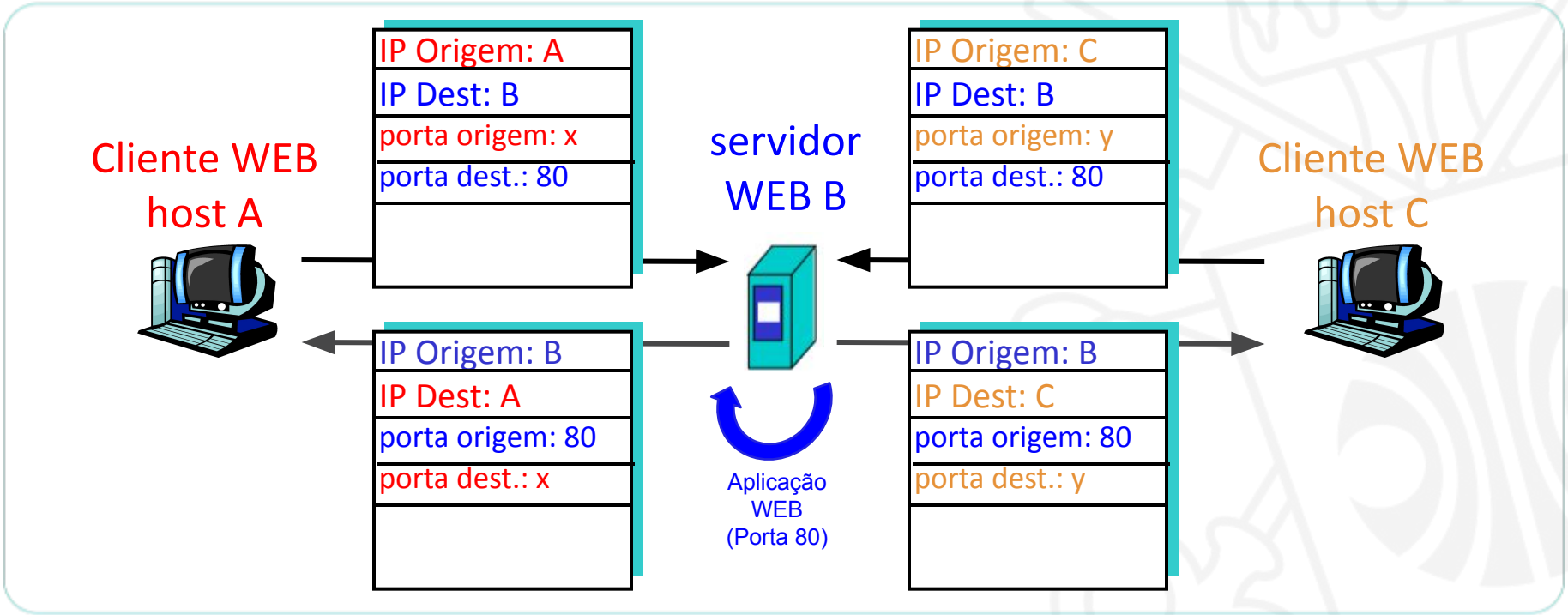
Aplicação
WEB
(Porta 80)

Cliente WEB
host C

Exemplo de Aplicação WEB (Porta 80)



Exemplo de Aplicação WEB (Porta 80)



Camada de Rede vs. de Transporte

- Responsáveis pelo transporte de dados entre origem e destino
- Contudo, têm preocupações distintas:
  - **Camada de rede**: roteamento dos dados
  - **Camada de transporte**: comunicação entre processos que executam na origem e destino. Tais processos executam os protocolos de transporte

Unificar as Camadas de Rede e Transporte?

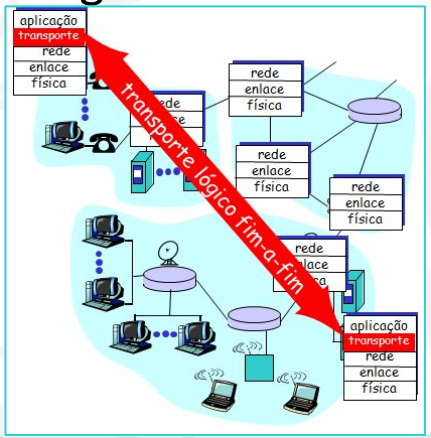
- Se as camadas de rede e transporte oferecem serviços semelhante, as duas não deveriam ser unificadas?
  - Resposta sutil, mas crucial
  - Os protocolos de transporte funcionam integralmente nas máquinas de origem e destino e os de rede, principalmente, nos roteadores

Camada de Rede vs. de Transporte

- Camada de rede
  - Executa o transporte fim a fim usando datagramas ou circuitos virtuais
- Camada de transporte
  - Utiliza a camada de rede
  - Provê as abstrações necessárias para que a aplicação use rede

Comunicação Lógica Fim a Fim

- Significa que uma entidade no nó emissor se comunica diretamente com sua entidade-par do *host* destinatário
- A camada de transporte é a primeira a fazer comunicação lógica fim a fim
- Roteadores, *hubs* e *switches* não precisam analisar os cabeçalhos de transporte para executar suas funções nativas



Camada de Rede vs. de Transporte

Camada de Rede	Camada de Transporte
Transferência de dados entre sistemas finais	Transferência de dados entre processos
Funciona principalmente nos roteadores	Funciona inteiramente nas máquinas dos usuários
Identificam as máquinas nas redes (e.g, número IP)	Identificam os processos nas máquinas (e.g., porta)

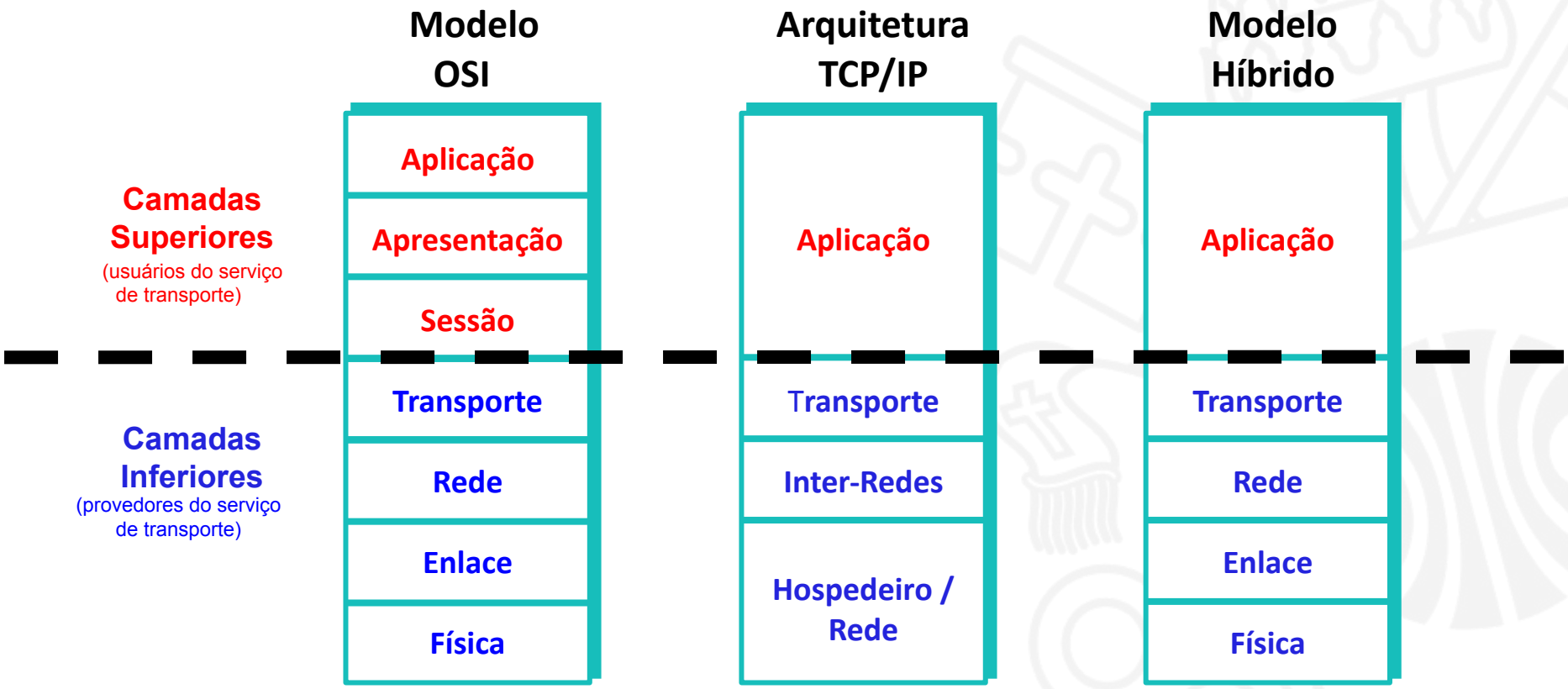
Serviços Oferecidos pela Camada de Transporte

- Normalmente são processos pertencentes à camada de aplicação
- Devem ser confiáveis, eficientes e econômicos
- Devem abstrair os problemas da rede para a aplicação
    - Problemas acontecem: rede não perfeita, heterogênea e dinâmica
    - Usuários não possuem qualquer controle real sobre a camada de rede

Serviços Orientados ou não à Conexão

	Rede orientada a conexão	Rede não orientada a conexão
Camada de Transporte orientado a conexão		**Exemplo: TCP/IP**
Camada de Transporte não orientado à conexão	Combinação normalmente inexistente porque estabeleceria uma conexão para enviar um único pacote	**Exemplo: UDP/IP**

Distinção entre Camadas: Superiores e Inferiores

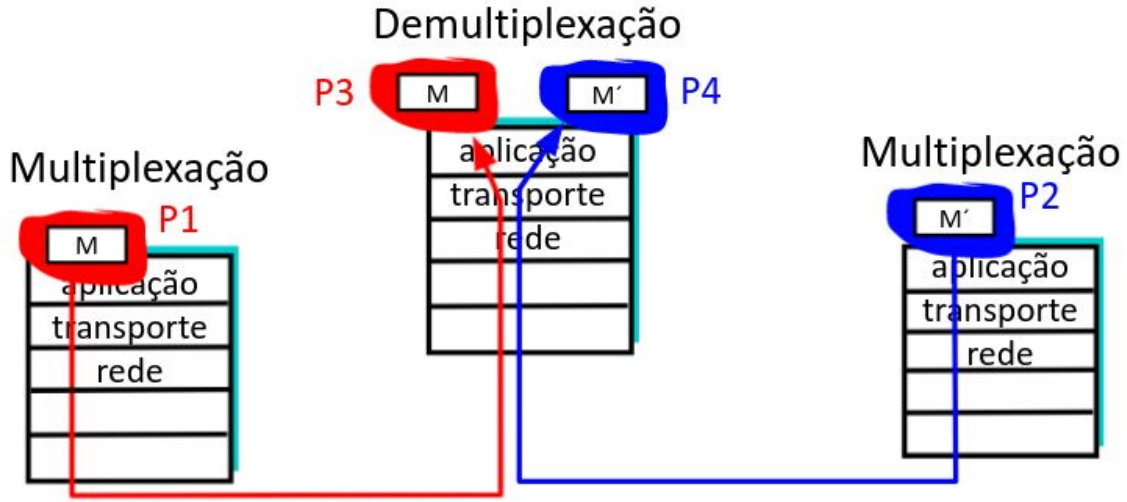


Função da Camada de Transporte

- Dar sentido à pilha de protocolos:
  - Tornando as camadas superiores imunes à tecnologia, projeto e imperfeições da rede
  - Efetuando retransmissões ou restabelecendo conexões

Multiplexação e Demultiplexação de Aplicações

- Permite a comunicação entre processos executando em máquinas distintas



Multiplexação e Demultiplexação de Aplicações

- **Multiplexação** (origem): Reunir os dados provenientes de diferentes processos de aplicação

- **Demultiplexação** (destino): Entrega dos dados contidos em um segmento da camada de transporte ao processo de aplicação correto

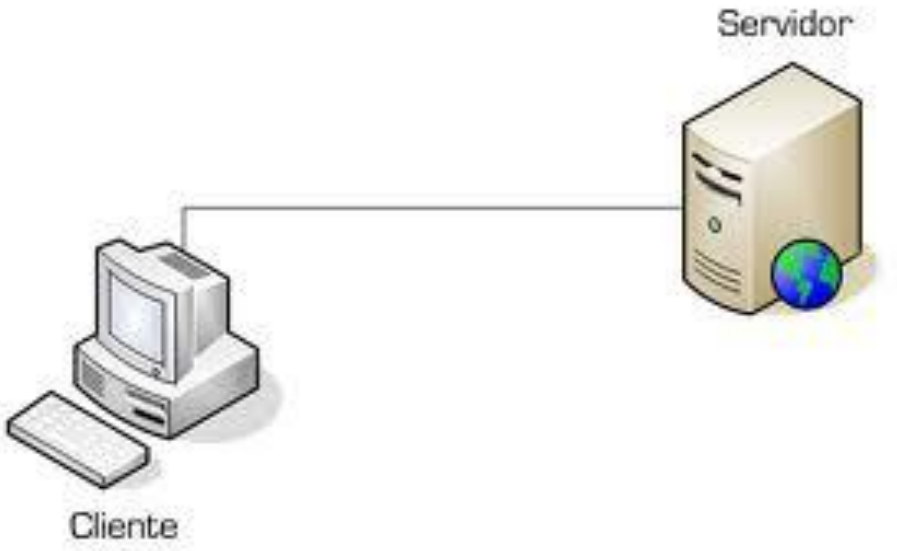
Primitivas para um Serviço de Transporte Simples

PUC Minas Virtual

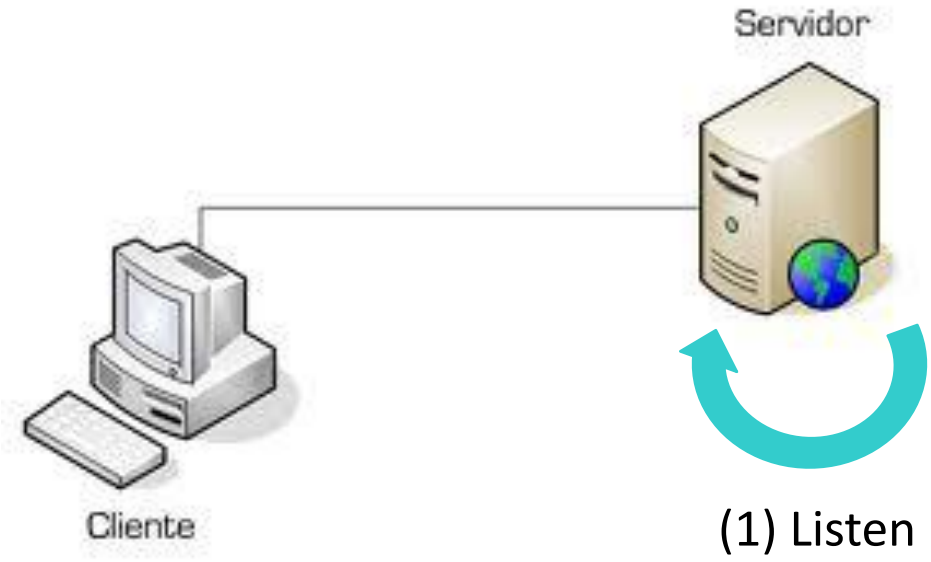
Primitivas

Primitiva	Pacote enviado	Significado
LISTEN	(nenhum)	Bloqueia até algum processo tentar conectar
CONNECT	CONNECTION REQ.	Tenta ativamente estabelecer uma conexão
SEND	DATA	Envia informação
RECEIVE	(nenhum)	Bloqueia até que um pacote de dados chegue
DISCONNECT	DISCONNECTION REQ.	Solicita uma liberação da conexão

Primitivas

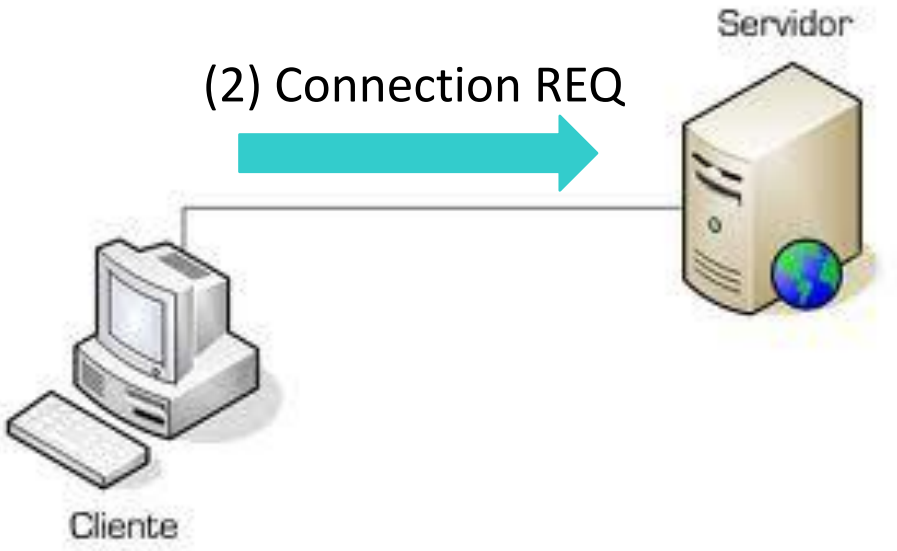


Primitivas

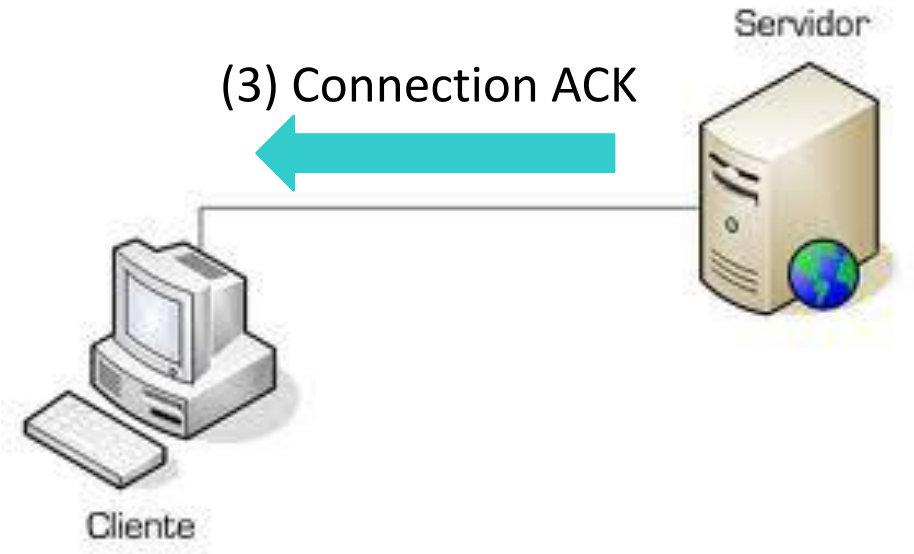


(1) Listen

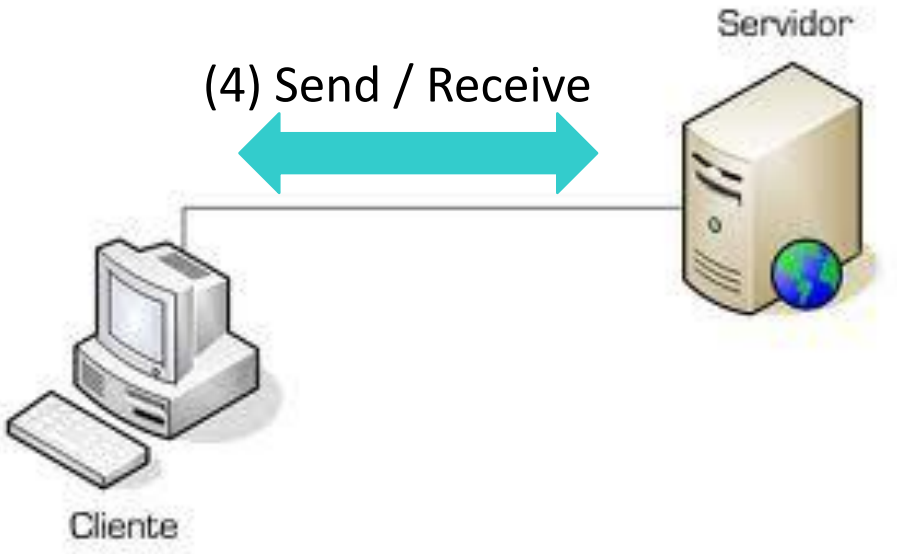
Primitivas



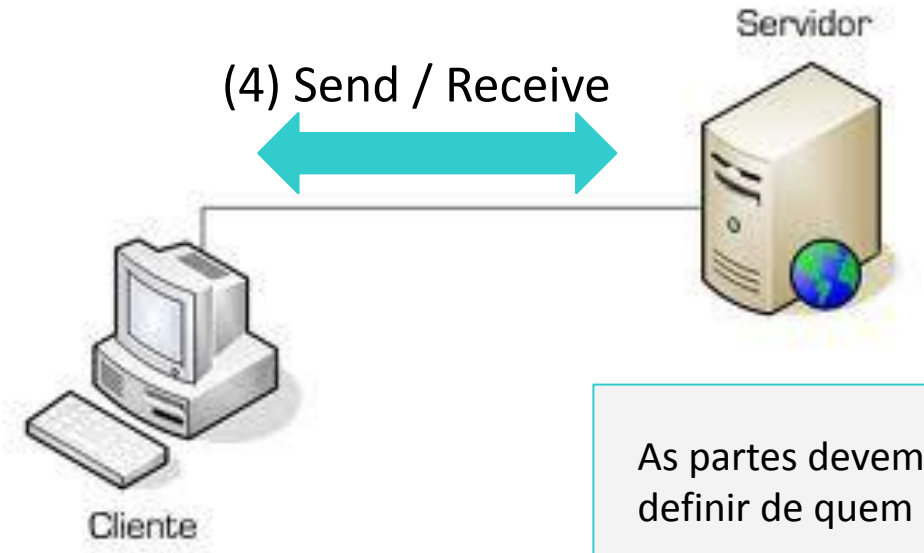
Primitivas



Primitivas

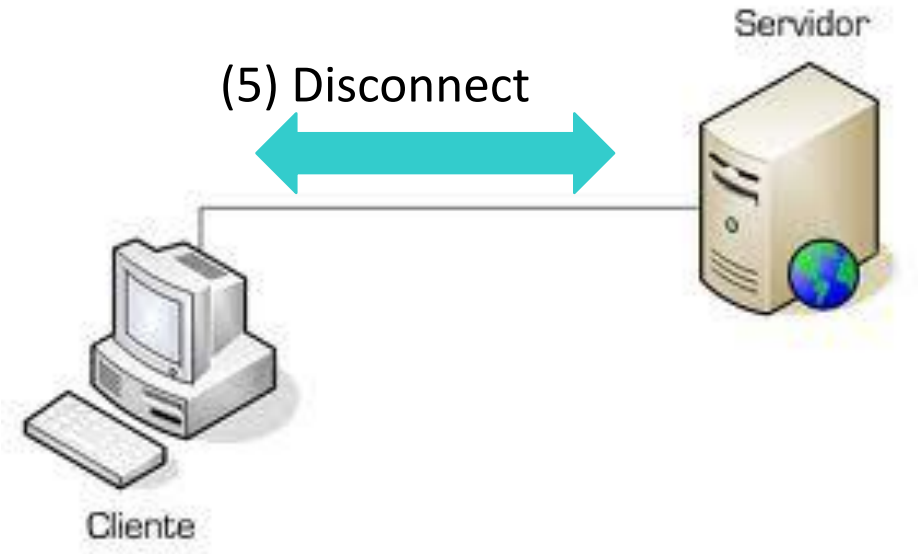


Primitivas

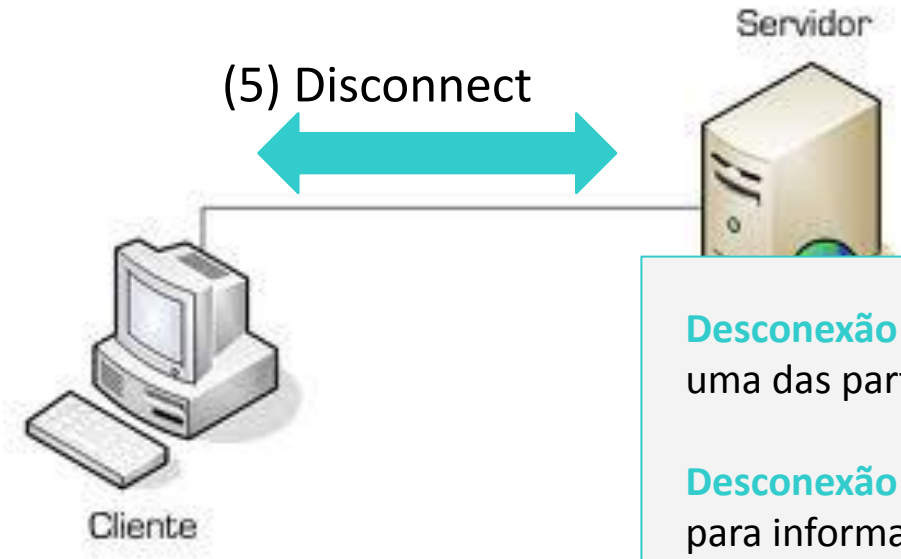


As partes devem ter uma política para definir de quem é a vez de enviar dados

Primitivas



Primitivas



Desconexão assimétrica: a conexão termina quando uma das partes solicita

Desconexão simétrica: cada parte faz sua desconexão para informar que não enviará mais dados

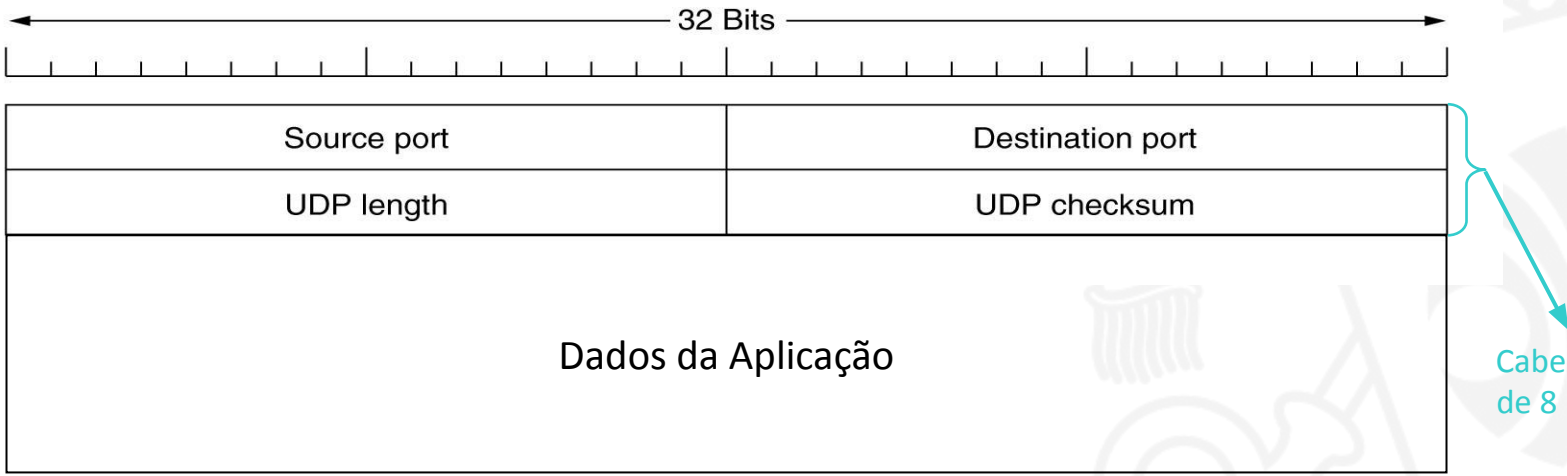
User Datagram Protocol (UDP)

RFC 768

UDP

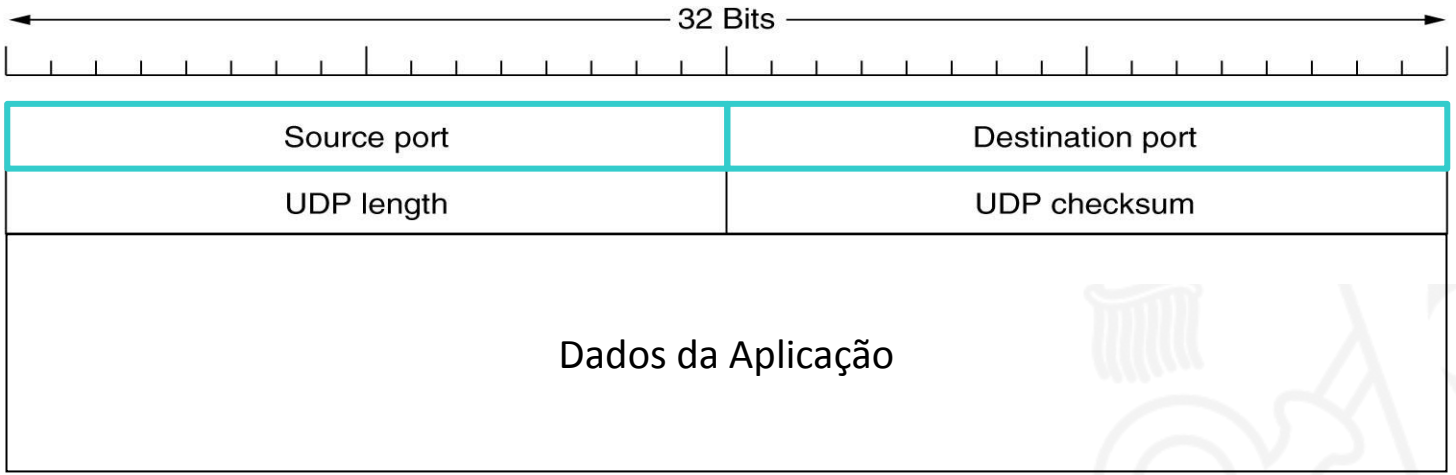
- Protocolo de transporte não confiável
- Não estabelece conexão
  - Não há apresentação entre o UDP emissor e o receptor
  - Cada segmento UDP é tratado de forma independente dos outros
- Faz um serviço *best-effort*, sendo que seus segmentos podem ser:
  - Perdidos
  - Entregues fora de ordem para a aplicação

Segmento UDP



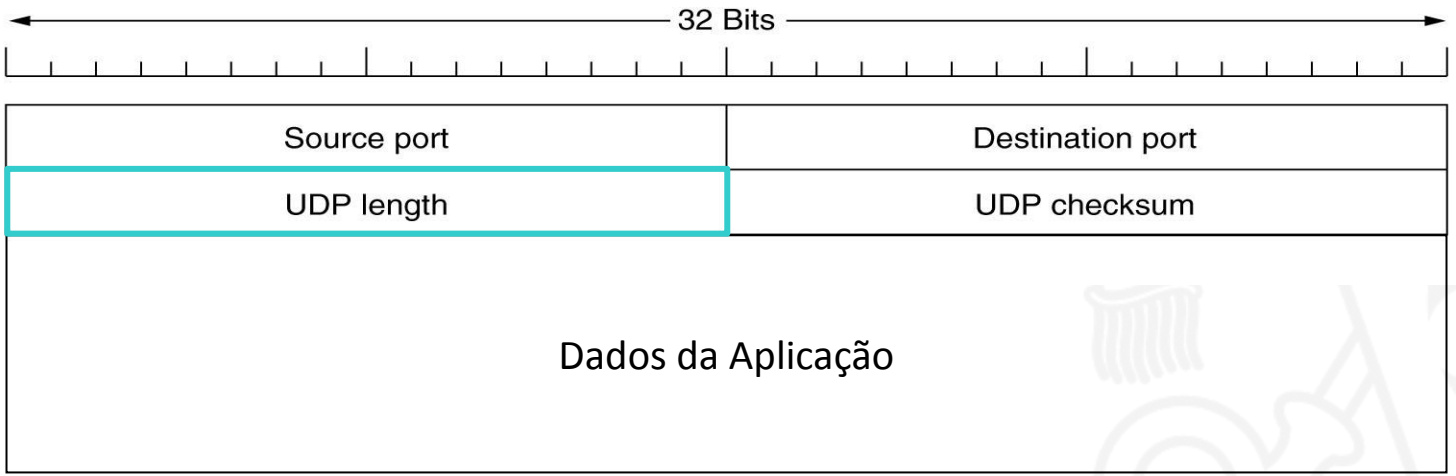
Source Port e *Destination Port* (16 bits, cada)

- Identificam os processos na origem e destino. Quando um pacote UDP chega, sua carga útil é entregue ao processo associado à porta destino



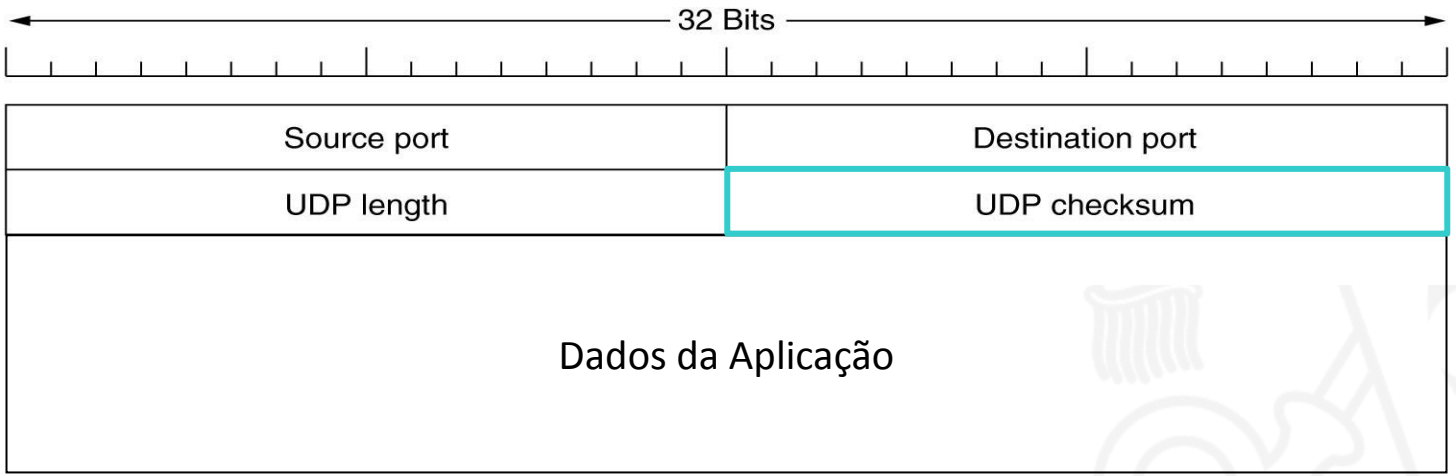
UDP length (16 bits)

- Tamanho total do segmento (incluindo o cabeçalho) em bytes (entre 8 de 65515)



UDP checksum (16 bits)

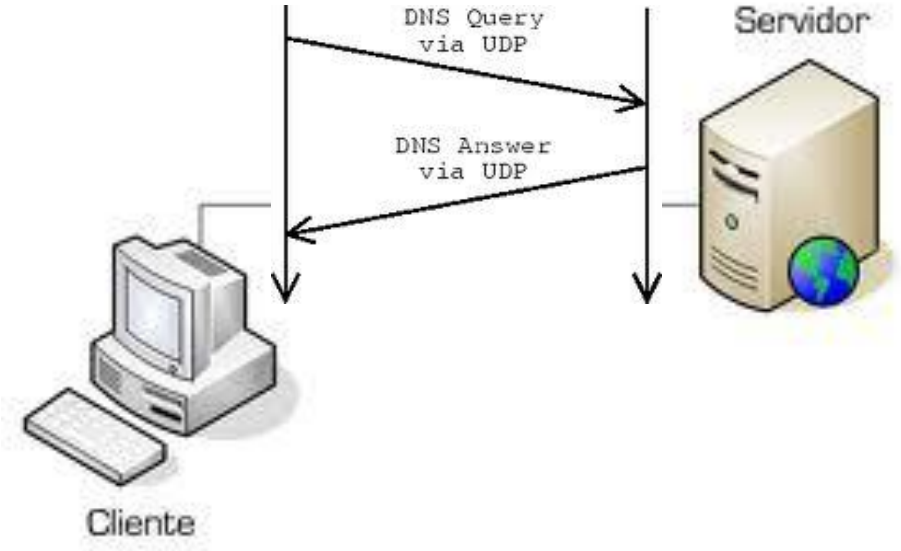
- Campo opcional, armazenado como 0 se não for calculado



Funcionamento Básico do UDP

- Cliente envia uma solicitação curta para o servidor
- Cliente aguarda por uma resposta curta
- Quando o cliente não recebe a resposta, ele aguarda um *timeout* e tenta novamente

DNS: Um Exemplo de Aplicação UDP



Ação do UDP

- Fornece uma interface para o protocolo de rede com o recurso adicional de multiplexação / demultiplexação de vários processos que utilizam as portas

Ações que o UDP Não Realiza

- Controle de fluxo
- Controle de congestionamento
- Controle de erros
- Retransmissão após a recepção de um segmento incorreto

Por que UDP?

- Não há estabelecimento de conexão e, portanto, não introduz atrasos
- Não há estado de conexão no emissor nem no receptor
- Pequeno *overhead* no cabeçalho do pacote
- Taxa de envio não regulada (não há controle de congestionamento)

Aplicações *One Shot*: Onde Usamos UDP

- Aplicações cliente-servidor com apenas uma requisição e uma resposta
- O custo para estabelecer uma conexão é alto quando comparado com a transferência de dado
- Aplicações de multimídia contínua (*streaming*)
  - Tolerantes à perda
  - Sensíveis à taxa

Exemplo de Aplicações UDP e TCP

Aplicação	Protocolo de camada de aplicação	Protocolo de transporte
Correio eletrônico	SMTP	TCP
Acesso a terminal remoto	Telnet	TCP
Web	HTTP	TCP
Transferência de arquivo	FTP	TCP
Servidor remoto de arquivo	NFS	UDP
Recepção de multimídia	proprietário	UDP
Telefonia por Internet	proprietário	UDP
Gerenciamento de rede	SNMP	UDP
Protocolo de roteamento	RIP	UDP
Tradução de nome	DNS	UDP

Transport Control Protocol (TCP)

RFCs: 793, 1122, 1323, 2018, 2581

Agenda

- Introdução
- Primitivas para um Serviço de Transporte Simples
- *User Datagram Protocol* (UDP)
- ***Transport Control Protocol* (TCP)**

  - Introdução
  - Primitivas de Soquetes para TCP
  - Segmento TCP
  - Serviços do TCP

Agenda

- Introdução
- Primitivas para um Serviço de Transporte Simples
- *User Datagram Protocol* (UDP)
- ***Transport Control Protocol* (TCP)**

- **Introdução**
- Primitivas de Soquetes para TCP
- Segmento TCP
- Serviços do TCP

TCP

- Fornece um fluxo de bytes fim a fim confiável em uma sub-rede não confiável
- Projetado para se adaptar dinamicamente às propriedades da sub-rede e ser robusto diante dos diversos tipos de falhas que podem ocorrer
- Principal protocolo de transporte na arquitetura TCP/IP

Características do TCP

- **Transferência confiável de dados**: garante que os dados serão entregues da forma em que foram enviados
- **Orientado à conexão**: conexões gerenciadas nos sistemas finais
- **Controle de fluxo**: o emissor não esgota a capacidade do receptor
- **Controle de congestionamento**: Emissor não esgota os recursos da rede
- **Gerenciamento de temporizadores**: baseado em temporizadores

Modelo de Serviço do TCP

- Aceita fluxos de dados das aplicações e os divide em segmentos de no máximo 65495 bytes
- Fluxo de bytes é diferente de fluxo de mensagens
- Normalmente, usa 1460 bytes de dados fazendo com que cada segmento caiba em um único quadro Ethernet

Modelo de Serviço do TCP

- Envia cada segmento para a camada de rede
- Na máquina do destino, os dados são entregues à entidade TCP e essa restaura os fluxos originais

Modelo de Serviço do TCP

- Entidade TCP receptor retorna um segmento ACK com um número de confirmação igual ao próximo número de sequência que espera receber

- Quando o emissor não recebe o ACK de um segmento cujo *timeout* expirou, ele retransmite esse segmento

Modelo de Serviço do TCP

- Provê um serviço de entrega de dados confiável para as aplicações
- Trata perdas e atrasos sem sobrecarregar a rede e seus roteadores

Modelo de Serviço do TCP

- Obtido quando o emissor e receptor criam processos chamados **soquetes** (em inglês, *sockets*):
  - Identificação do soquete = endereço IP + número de porta
  - Portas bem conhecidas (*well-known ports*) são reservadas para serviços tradicionais

Port	Protocol	Use
21	FTP	File transfer
23	Telnet	Remote login
25	SMTP	E-mail
69	TFTP	Trivial file transfer protocol
79	Finger	Lookup information about a user
80	HTTP	World Wide Web
110	POP-3	Remote e-mail access
119	NNTP	USENET news

Modelo de Serviço do TCP

- Baseado na existência de uma conexão entre um soquete na origem e outro no destino

- Fases do TCP:
  - Estabelecimento da conexão
  - Transferência de dados
  - Término da conexão

Modelo de Serviço do TCP

- As conexões são identificadas pelos soquetes da origem e do destino, ou seja, cada conexão é identificada por:
  - Endereço IP origem
  - Porta origem
  - Endereço IP destino
  - Porta destino

- Um soquete pode ser utilizado por várias conexões ao mesmo tempo (duas ou mais conexões podem terminar no mesmo soquete)

Modelo de Serviço do TCP

- Todas as conexões TCP são *full-duplex* e ponto a ponto
  - Tráfego pode ser feito em ambas as direções ao mesmo tempo
  - Cada conexão possui exatamente dois pontos terminais
  - TCP não admite multidifusão e difusão

Agenda

- Introdução
- Primitivas para um Serviço de Transporte Simples
- *User Datagram Protocol* (UDP)
- ***Transport Control Protocol* (TCP)**

  - Introdução
  - **Primitivas de Soquetes para TCP**
  - Segmento TCP
  - Serviços do TCP

Primitivas de Soquetes para TCP

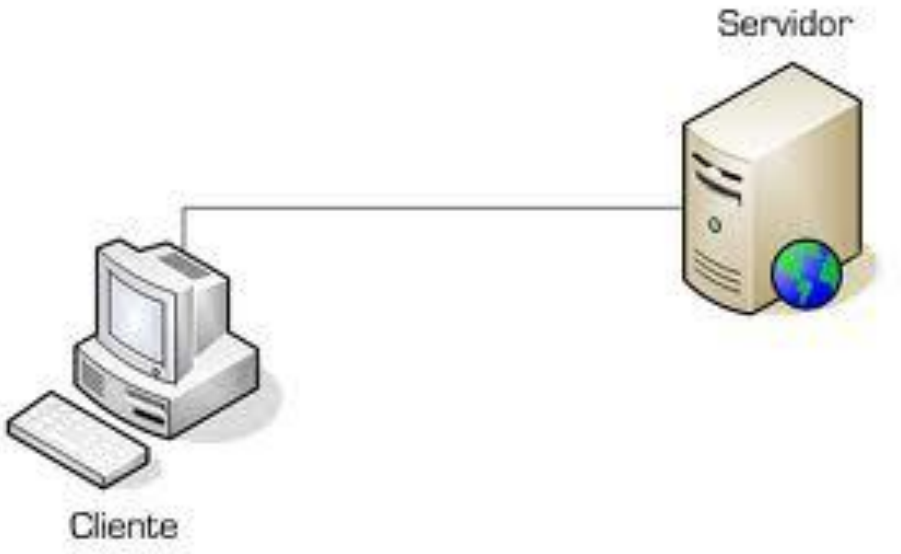
Primitiva	Significado
SOCKET	Criar um novo ponto final de comunicação
BIND	Anexar um endereço local a um soquete
LISTEN	Anunciar a disposição para aceitar conexões; mostrar o tamanho da fila
ACCEPT	Bloquear o responsável pela chamada até uma tentativa de conexão ser recebida
CONNECT	Tentar estabelecer uma conexão ativamente
SEND	Enviar alguns dados através da conexão
RECEIVE	Receber alguns dados da conexão
CLOSE	Encerrar a conexão

Primitivas de Soquetes para TCP

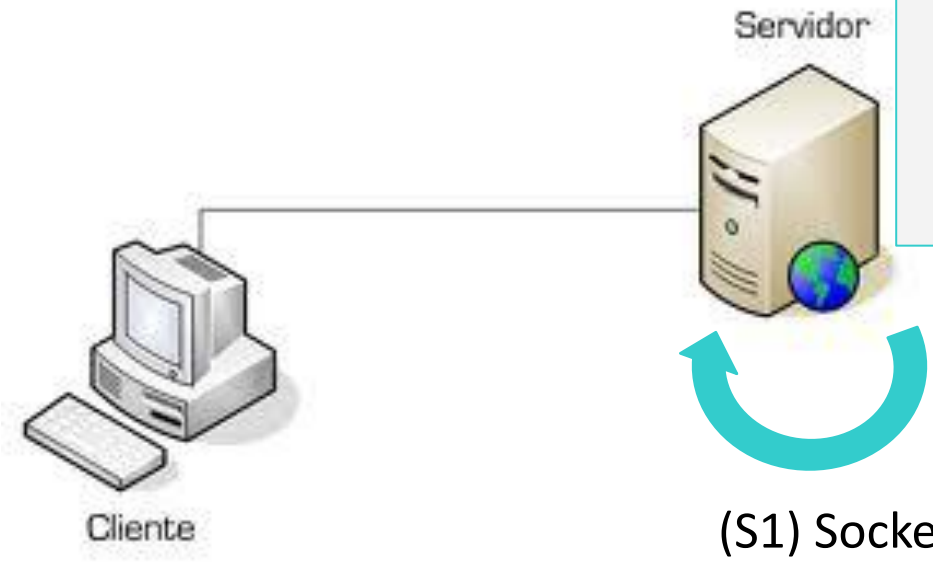
Primitiva	Significado
SOCKET	Criar um novo ponto final de comunicação
BIND	Anexar um endereço local a um soquete
LISTEN	Anunciar a disposição para aceitar conexões; mostrar o tamanho da fila
ACCEPT	Bloquear o responsável pela chamada até uma tentativa de conexão ser recebida
CONNECT	Tentar estabelecer uma conexão ativamente
SEND	Enviar alguns dados através da conexão
RECEIVE	Receber alguns dados da conexão
CLOSE	Encerrar a conexão

Essas primitivas são executadas nesta mesma pelo servidor

Primitivas de Soquetes para TCP

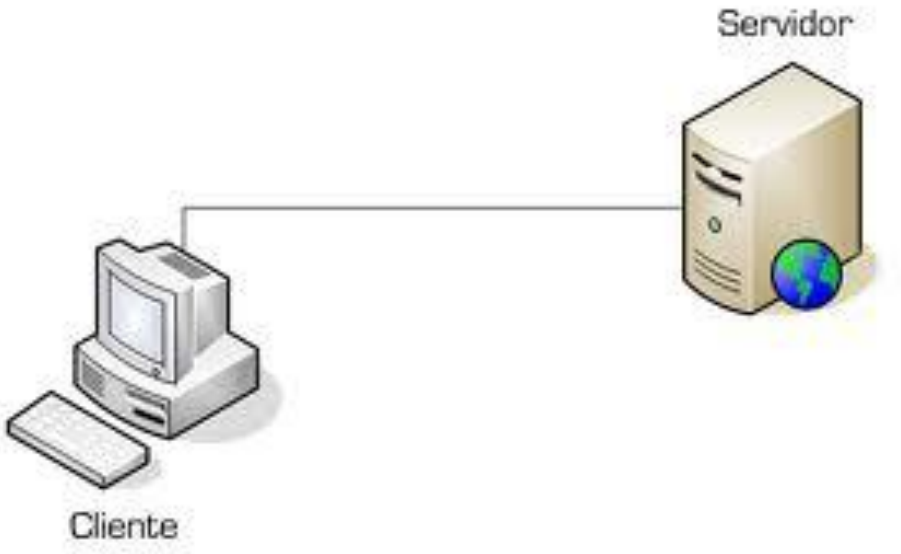


Primitivas de Soquetes para TCP

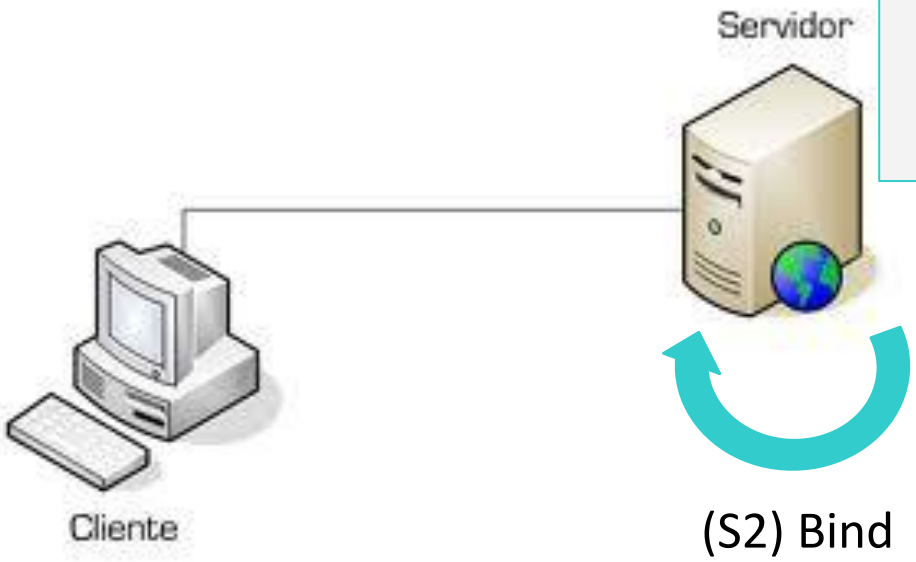


Especifica o formato do endereço, tipo de serviço (e.g., fluxo de bytes confiável) e protocolo

Primitivas de Soquetes para TCP



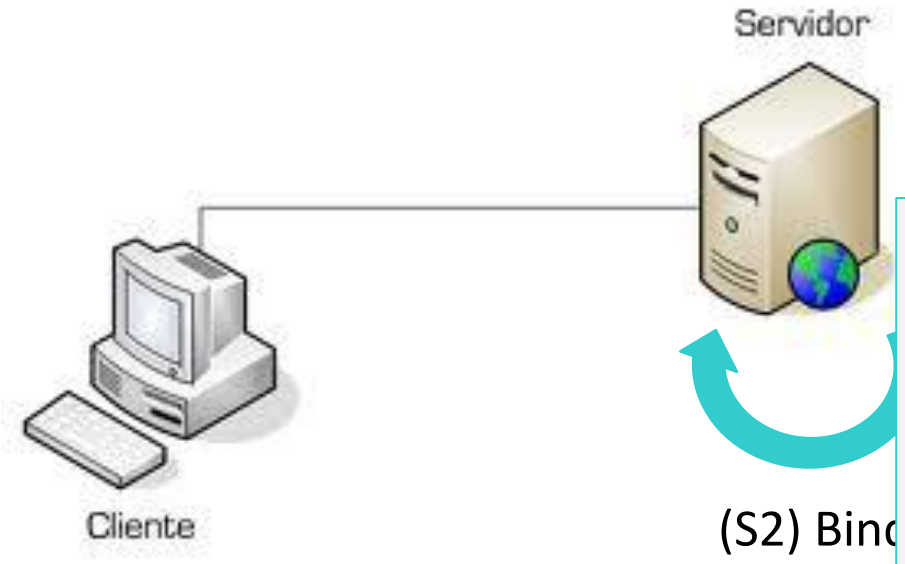
Primitivas de Soquetes para TCP



Atribui um endereço de rede ao soquete

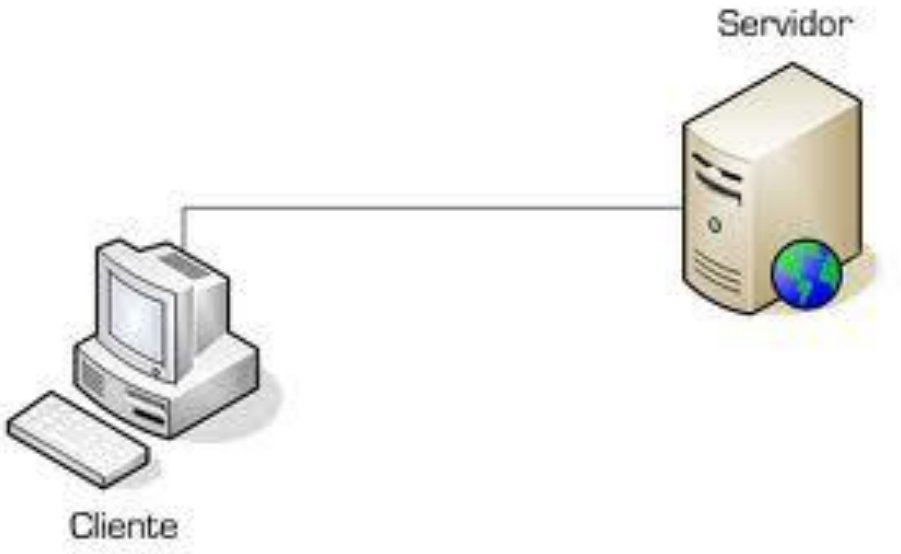
(S2) Bind

Primitivas de Soquetes para TCP

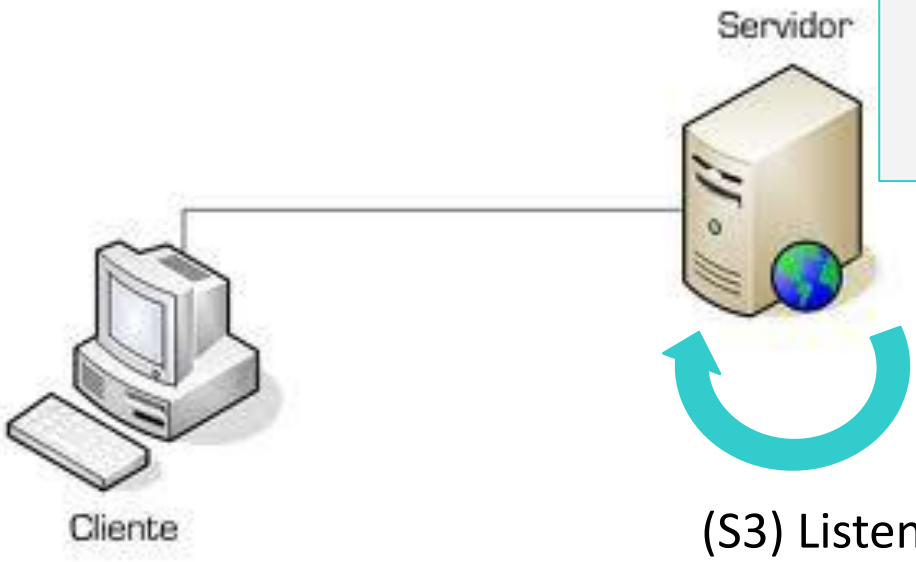


Observação: a primitiva socket não atribui diretamente um endereço porque alguns processos utilizam endereços anteriormente definidos

Primitivas de Soquetes para TCP



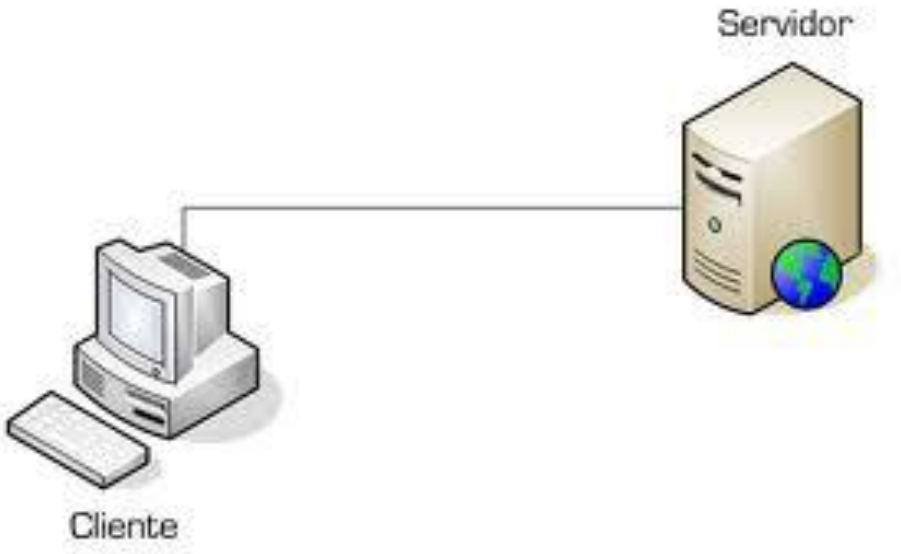
Primitivas de Soquetes para TCP



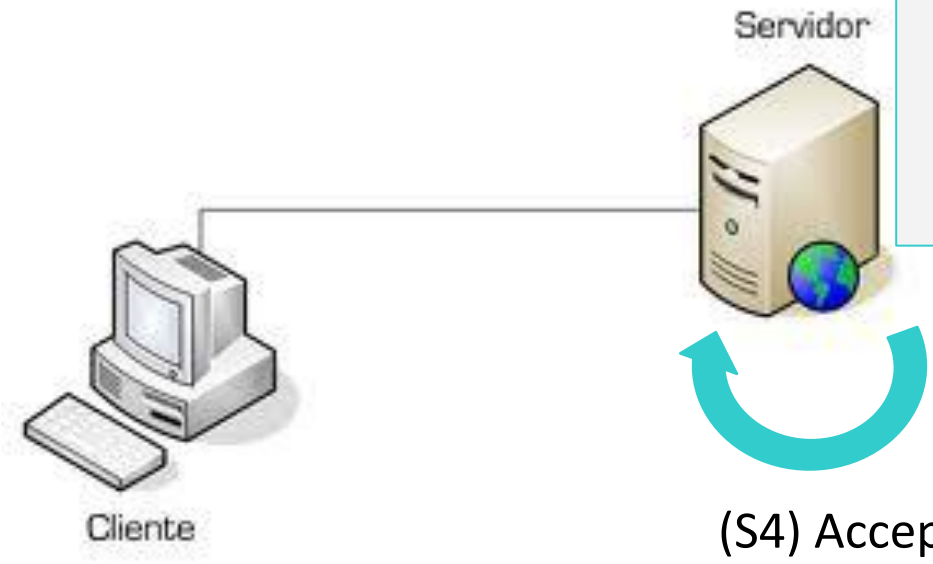
Aloca espaço para a fila de chamadas recebidas

(S3) Listen

Primitivas de Soquetes para TCP



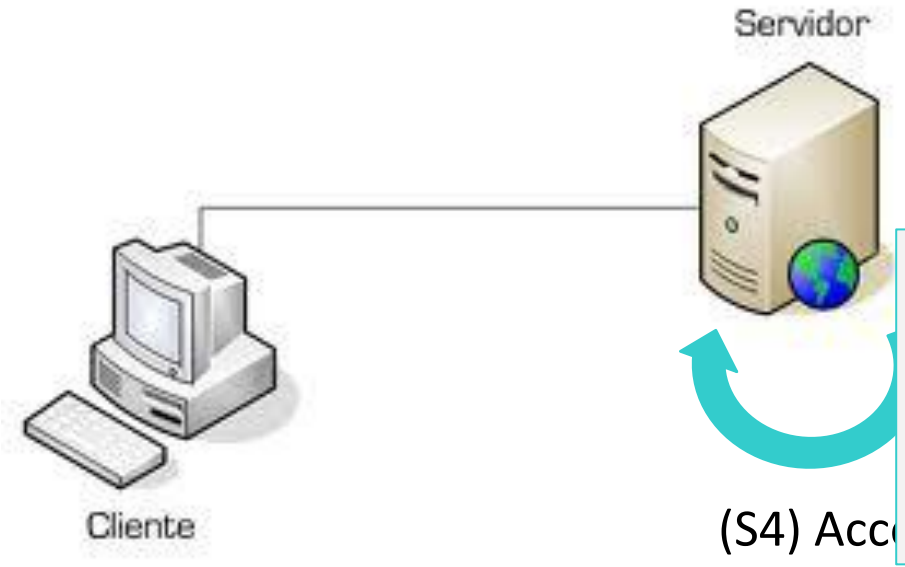
Primitivas de Soquetes para TCP



Bloqueia o responsável pela chamada até que o servidor receba uma tentativa de conexão

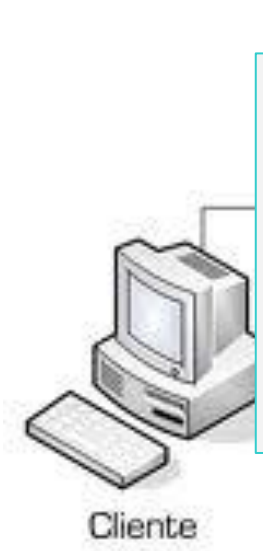
(S4) Accept

Primitivas de Soquetes para TCP



Quando o servidor recebe uma conexão, ele pode desviá-la para outro processo (*thread*) e voltar à espera por uma nova conexão

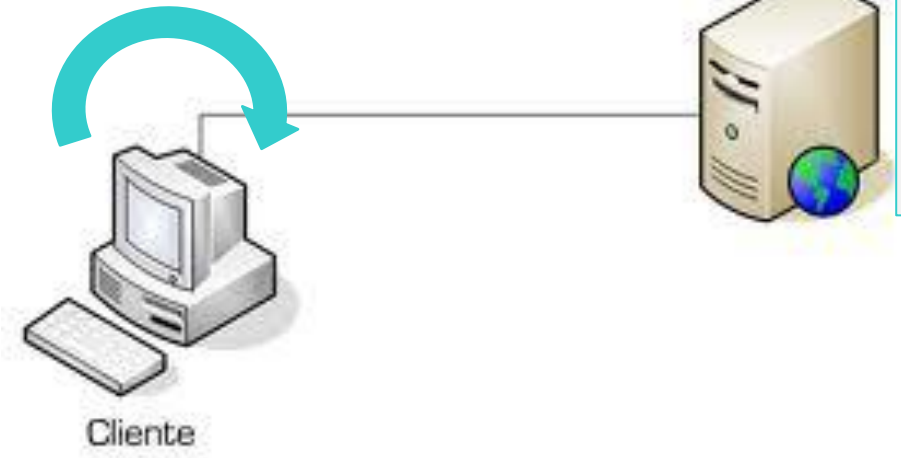
Primitivas de Soquetes para TCP



Vamos para o lado cliente...

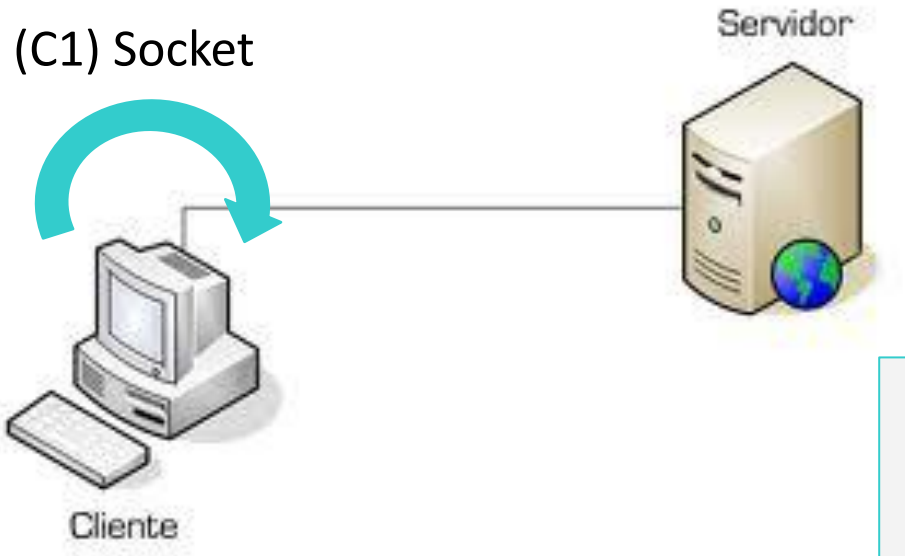
Primitivas de Soquetes para TCP

(C1) Socket



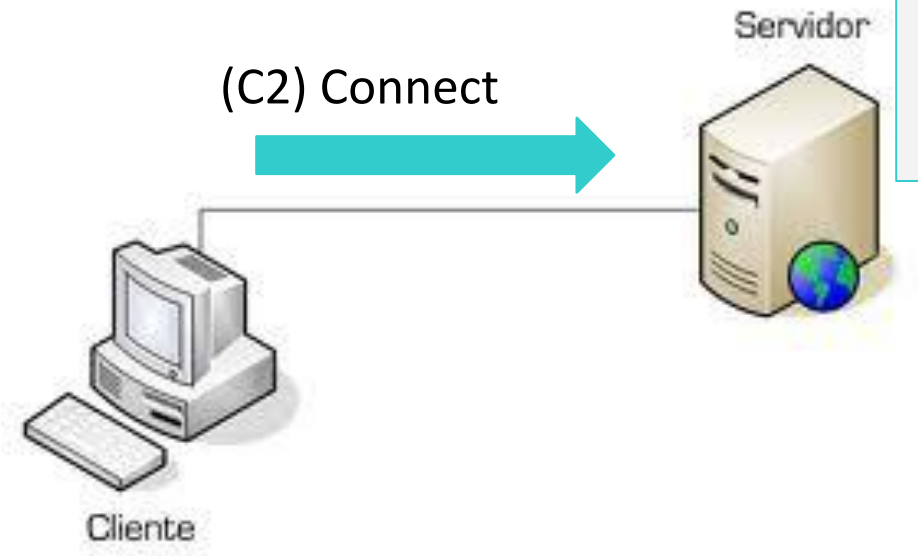
Como no servidor, esta primitiva especifica o formato do endereço, tipo de serviço (e.g., fluxo de bytes confiável) e protocolo

Primitivas de Soquetes para TCP



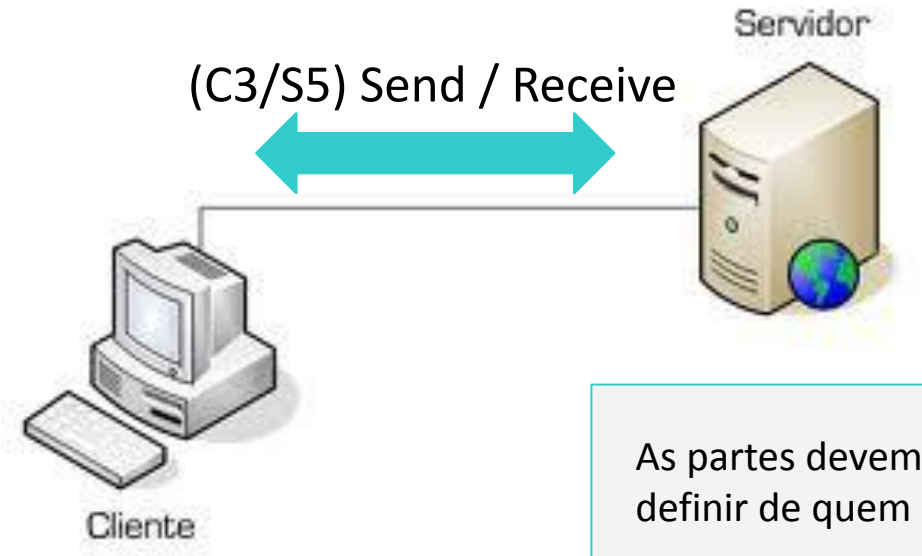
A primitiva BIND não é necessária no cliente, pois seu endereço é irrelevante para o servidor

Primitivas de Soquetes para TCP



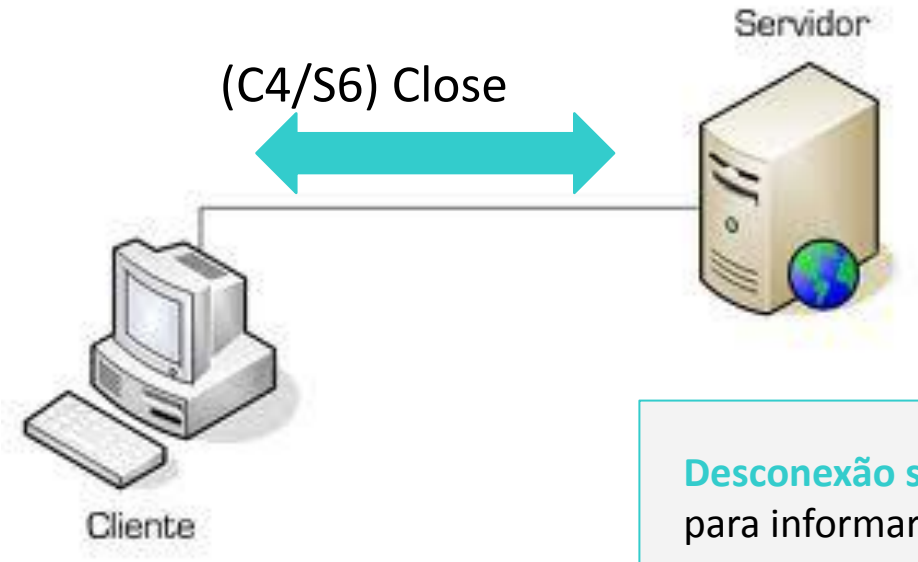
Tenta estabelecer uma conexão com o servidor

Primitivas de Soquetes para TCP



As partes devem ter uma política para definir de quem é a vez de enviar dados

Primitivas de Soquetes para TCP



Desconexão simétrica: cada parte faz sua desconexão para informar que não enviará mais dados

Agenda

- Introdução
- Primitivas para um Serviço de Transporte Simples
- *User Datagram Protocol* (UDP)
- ***Transport Control Protocol* (TCP)**

  - Introdução
  - Primitivas de Soquetes para TCP
  - **Segmento TCP**
  - Serviços do TCP

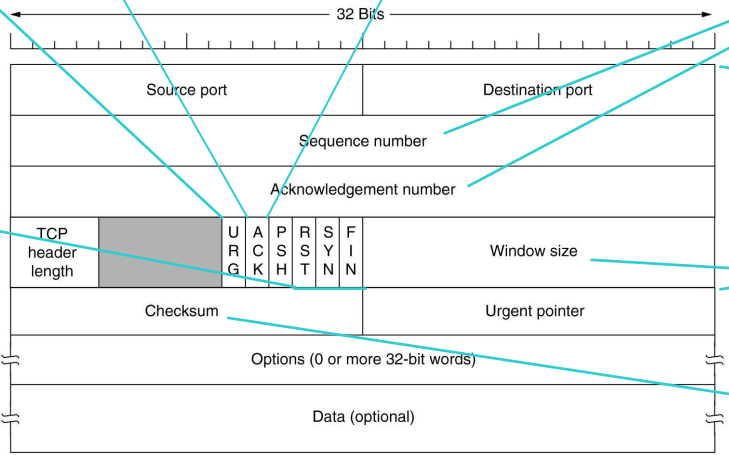
Segmento TCP

URG: dados urgentes (pouco usado)

PSH: produz envio de dados (pouco usado)

ACK: campo de ACK é válido

contagem por bytes de dados (não segmentos!)



20 bytes

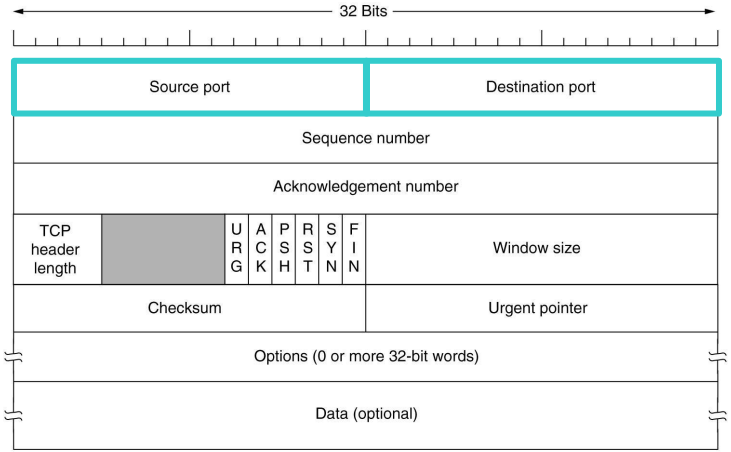
RST, SYN, FIN: estabelecimento de conexão (comandos de criação e término)

número de bytes que o receptor está pronto para aceitar

Total de verificação do segmento

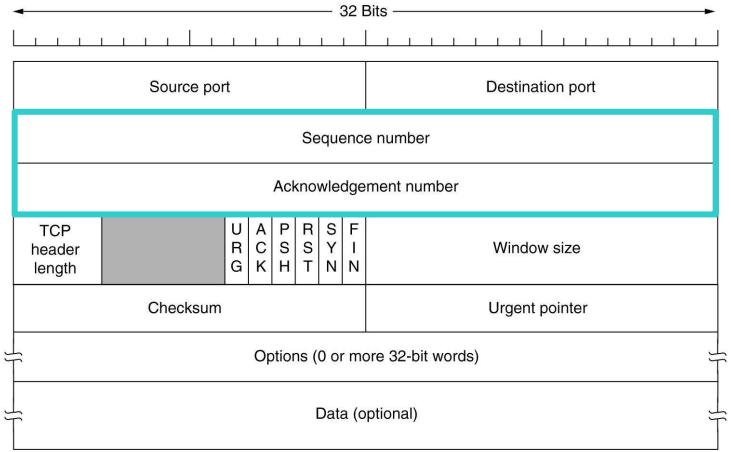
Source Port e *Destination Port* (16 bits, cada)

- Identificam os processos na origem e destino



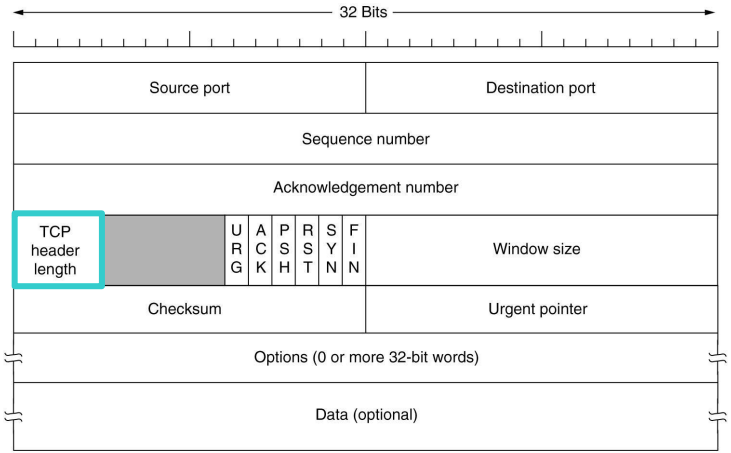
Sequence e *Acknowledgement numbers* (32 bits, cada)

- Atuam na transferência confiável de dados

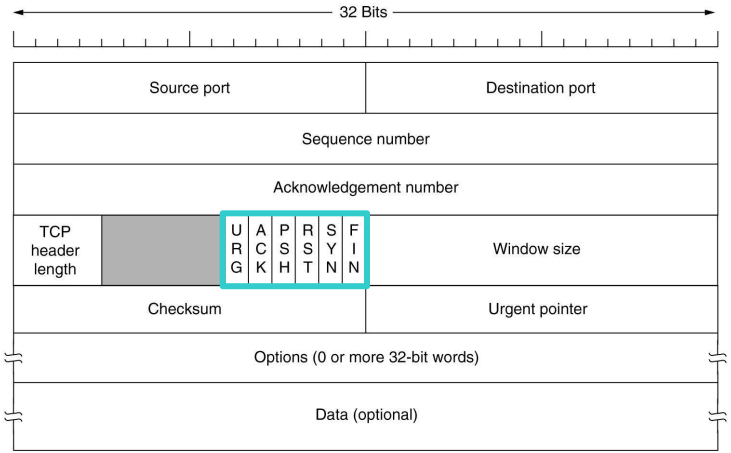


TCP header length (4 bits)

- Informa quantas palavras de 32 bits existem no cabeçalho TCP

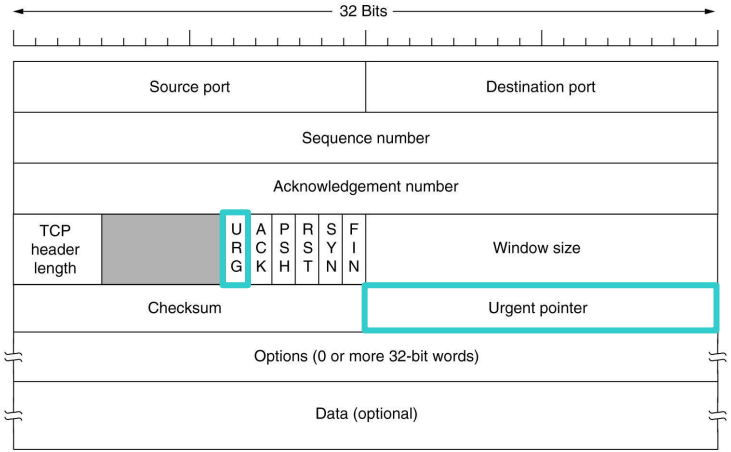


Flags URG, ACK, PSH, PST, SYN e FIN (1 bit, cada)



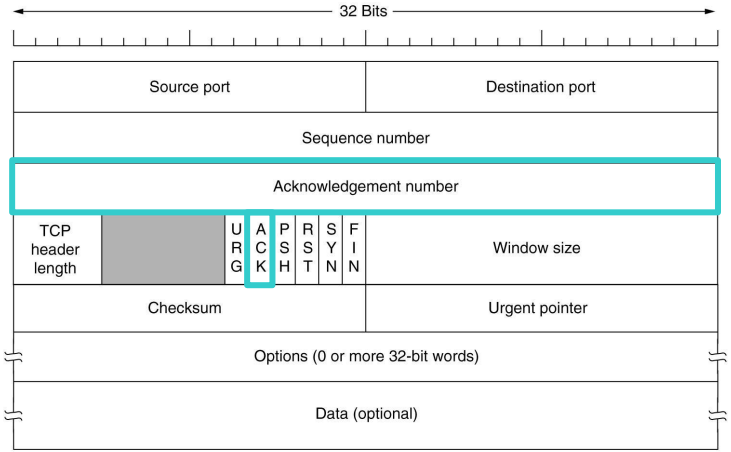
Flag URG (1 bit) e Urgent pointer (16 bits)

- Se URG = 1, Urgent pointer indica o deslocamento de bytes para os dados urgentes (mensagens de interrupção)



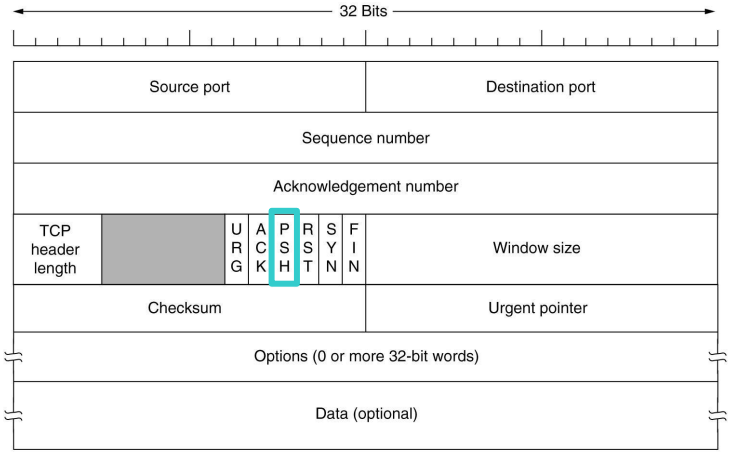
Flag ACK (1 bit) e *Acknowledgement number* (32 bits)

- Se ACK = 1, o campo *Acknowledgement number* é válido, senão, o segmento não contém uma confirmação



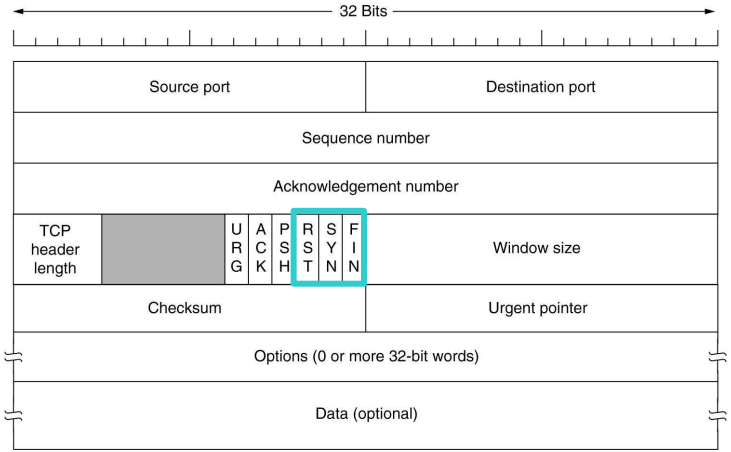
Flag PSH (1 bit)

- Informa ao TCP para não retardar a transmissão
- O receptor é solicitado a entregar os dados à aplicação imediatamente
- Pouco utilizado



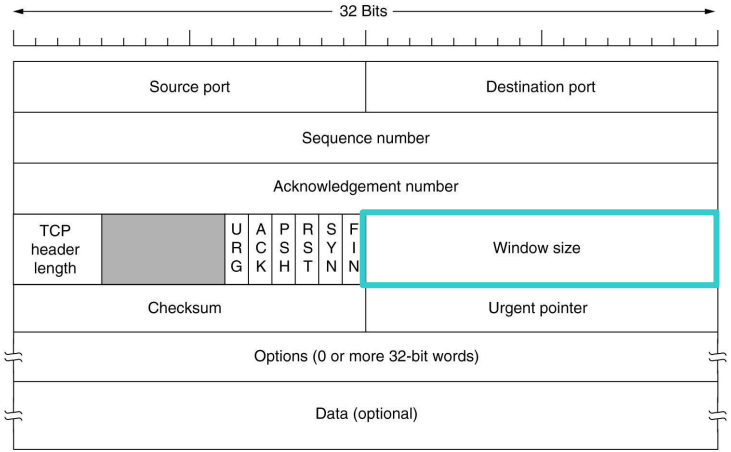
Flags RST, SYN e FIN (1 bit, cada)

- Gerenciamento de conexões



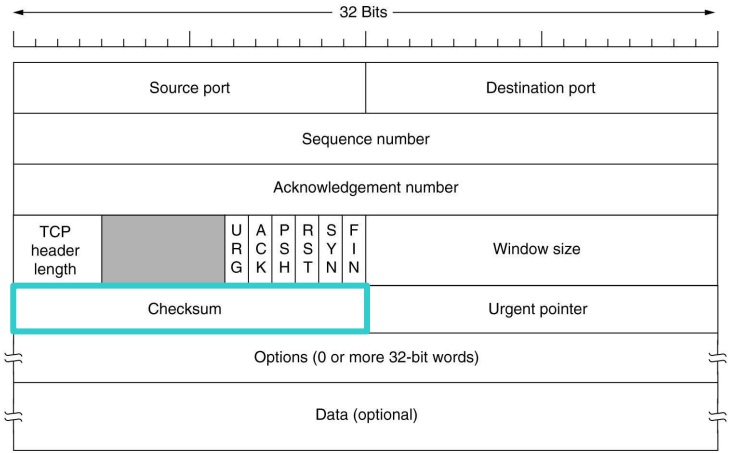
Window size (16 bits)

- Controle de fluxo



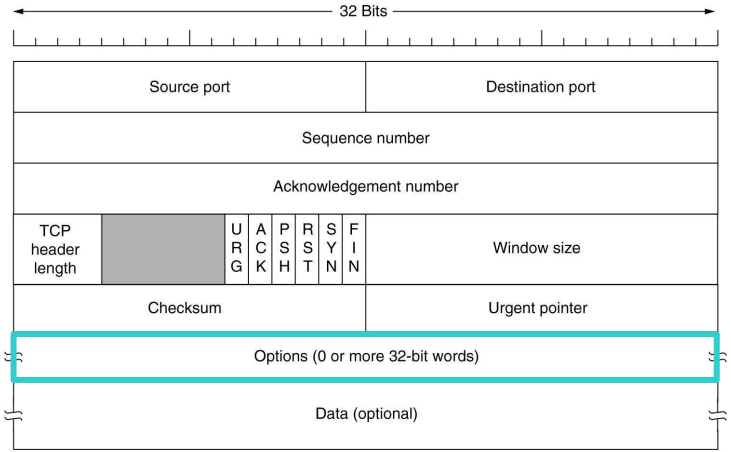
Checksum (16 bits)

- Verificação do segmento



Options (máximo, 40 bytes)

- Permite recursos não previstos no cabeçalho comum (e.g., negociação entre *hosts* para estipular o máximo de carga útil do TCP)



Agenda

- Introdução
- Primitivas para um Serviço de Transporte Simples
- *User Datagram Protocol* (UDP)
- ***Transport Control Protocol* (TCP)**

  - Introdução
  - Primitivas de Soquetes para TCP
  - Segmento TCP
  - **Serviços do TCP**

Serviços do TCP

- Transferência Confiável de Dados
- Gerenciamento da Conexão TCP
- Controle de Fluxo
- Controle de Congestionamento
- Gerenciamento de Temporizadores

Transferência Confiável de Dados

- Garantir que quando um processo destino lê o *buffer* com uma *stream*, essa:
  - Não está corrompida
  - Não tem lacunas
  - Não possui duplicações
  - Está em sequência
  - A cadeia de bytes é exatamente a mesma enviada pela origem

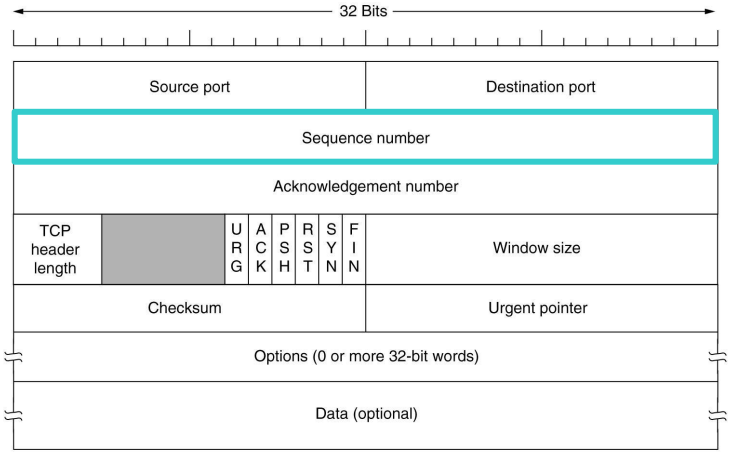
Considerações sobre a Transferência Confiável de Dados

- O TCP vê os dados como uma cadeia de bytes desestruturada, mas ordenada

- Cada byte em uma conexão TCP tem seu próprio número de sequência de 32 bits

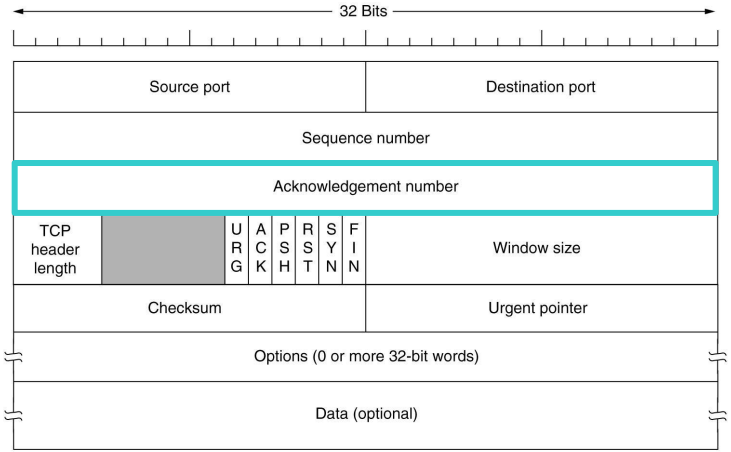
Campo *Sequence number* (32 bits)

- Aplicado sobre a cadeia de bytes transmitidos (não, série de segmentos transmitidos), indica o número do primeiro byte no segmento de dados

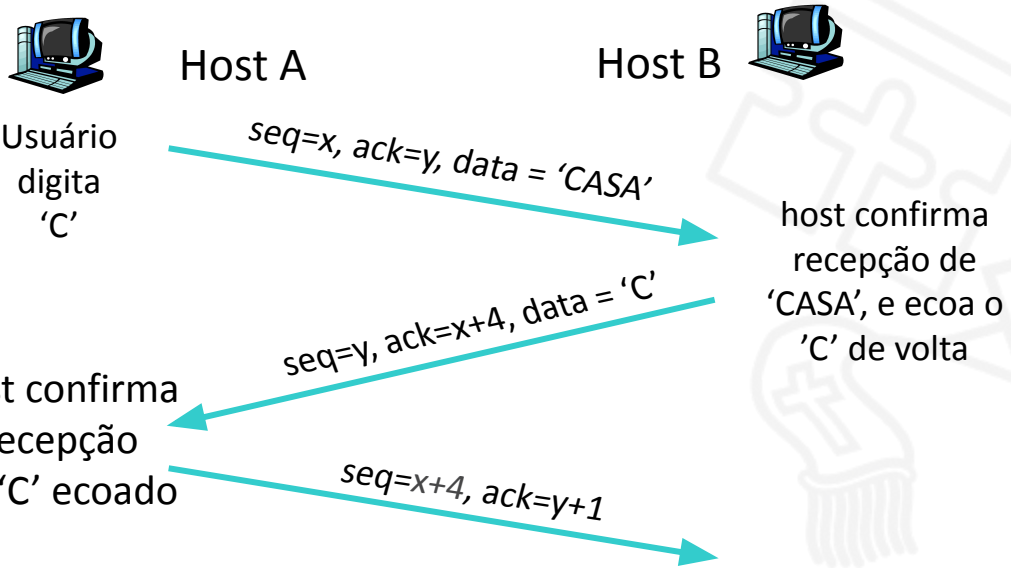


Campo *Acknowledgement number* (32 bits)

- Especifica o número do próximo byte esperado (não o último byte recebido corretamente)



Exemplo em um Cenário Telnet Simples



Gerenciamento da Conexão TCP

- Utiliza os bits RST, SYN e FIN do cabeçalho TCP
  - Bit RST: Recusa uma tentativa de conexão
  - Bit SYN: Estabelece conexões
  - Bit FIN: Finaliza uma conexão

Gerenciamento da Conexão TCP

- O TCP emissor estabelece uma conexão com o receptor antes de trocar segmentos de dados
- As conexões são estabelecidas através do 3-way handshake (3 vias)
- Inicialização de variáveis:
  - Números de sequência
  - Buffers, controle de fluxo

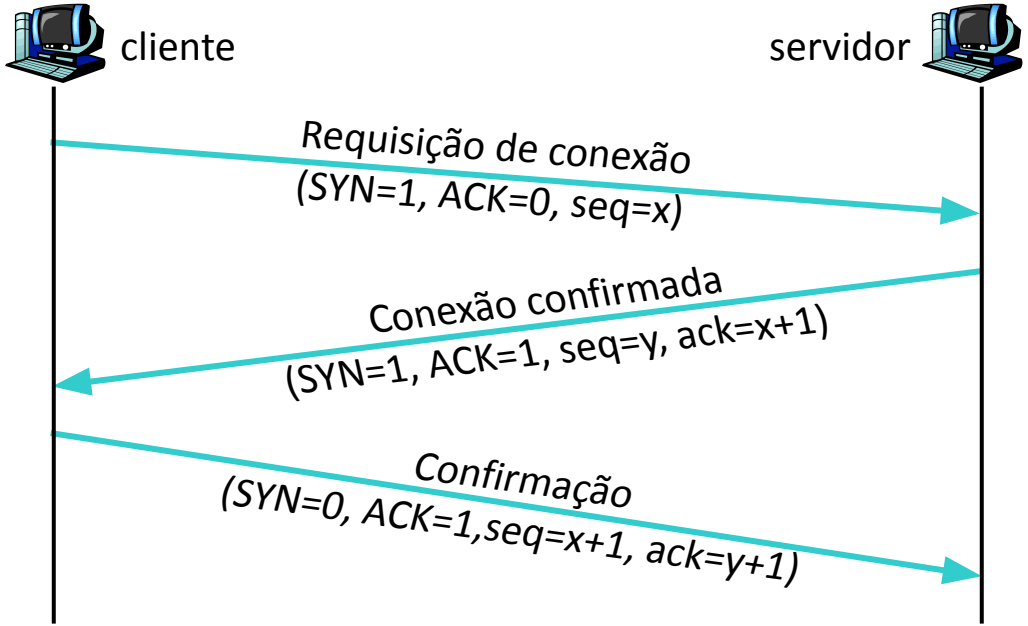
Estabelecimento da conexão TCP

- **Passo 1)** O cliente envia um segmento com SYN=1 e ACK=0 ao servidor para especificar o número inicial da sequência

- **Passo 2)** O servidor responde com o segmento SYNACK (SYN=1, ACK=1), reconhecendo o SYN recebido, alocando *buffers* e especificando o número de inicial de sua sequência

- **Passo 3)** O cliente recebe o SYNACK

Estabelecimento da conexão TCP

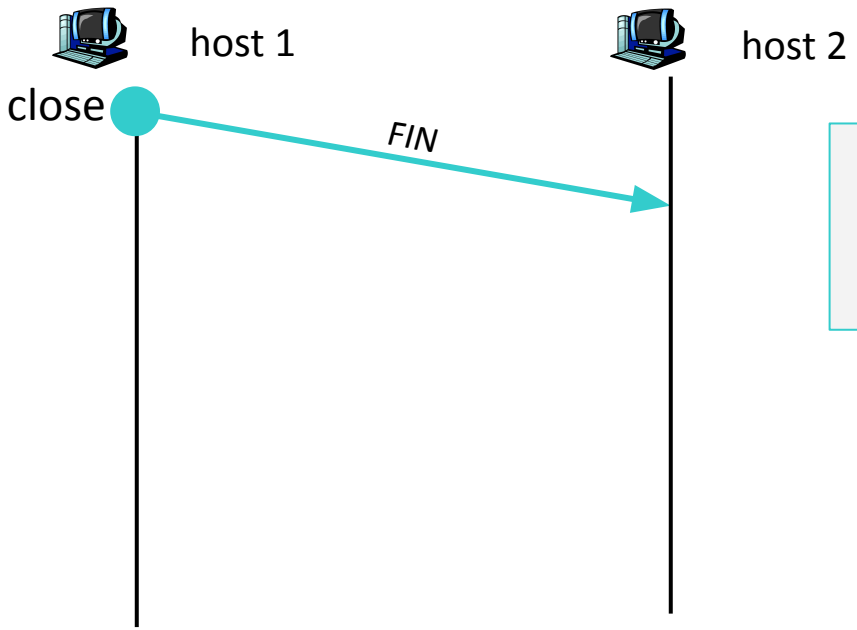


Um segmento SYN consome 1 byte de espaço de sequência

Término da conexão TCP

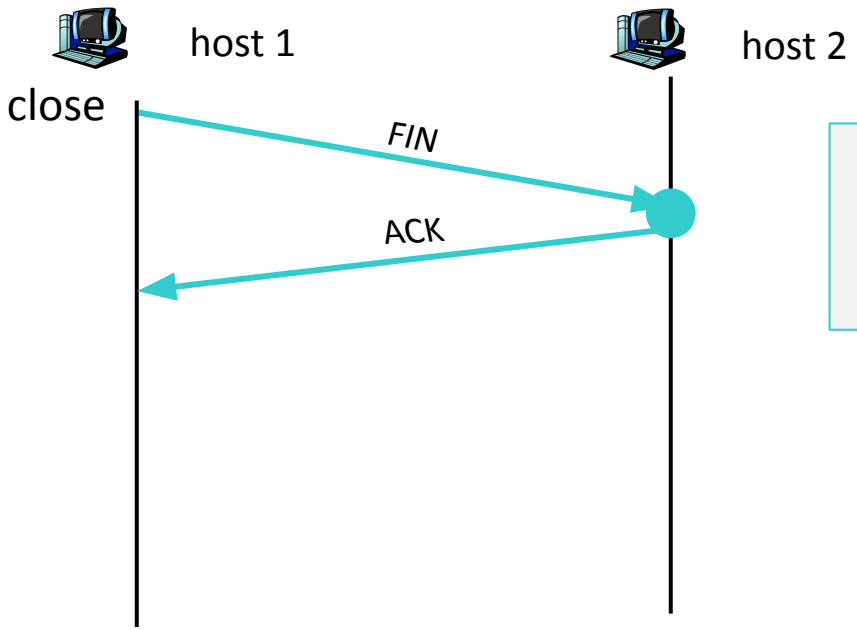
- As conexões TCP são *full-duplex*
- Contudo, é mais fácil compreender seu encerramento como se elas fossem um par de conexões simplex
- Assim, no TCP, cada conexão simplex é encerrada de modo independente de sua parceira

Término da conexão TCP



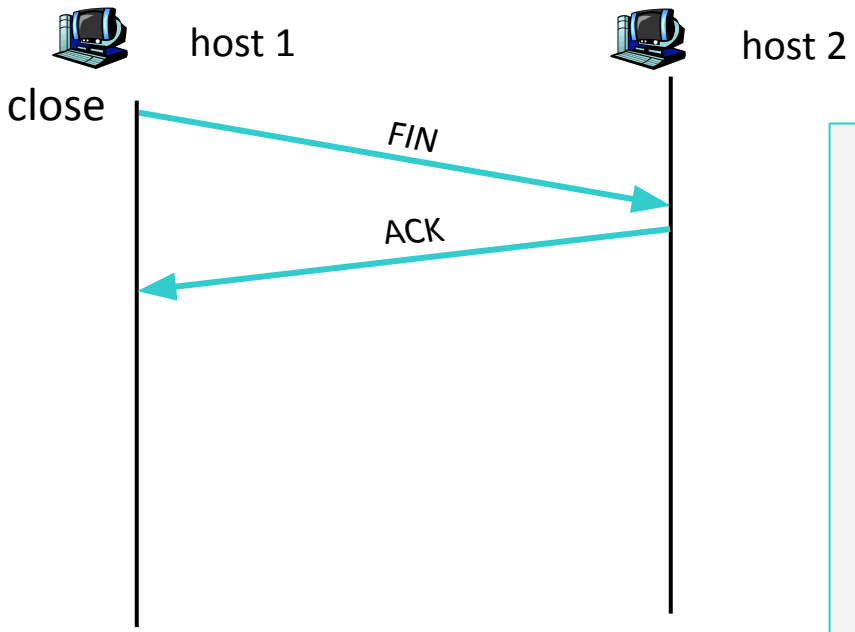
Passo 1) O host 1 envia um segmento TCP FIN para o host 2

Término da conexão TCP



Passo 2) O host 2 recebe o FIN e o responde com um ACK

Término da conexão TCP

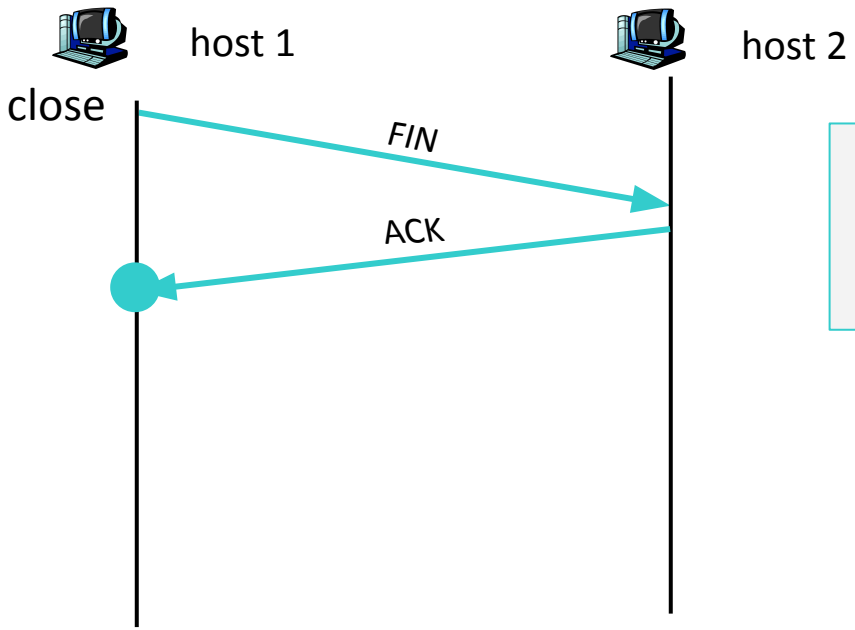


Passo 2) O host 2 recebe o FIN e o responde com um ACK

Em seguida, o host 2 fecha a conexão no sentido host 1 ⇒ host 2

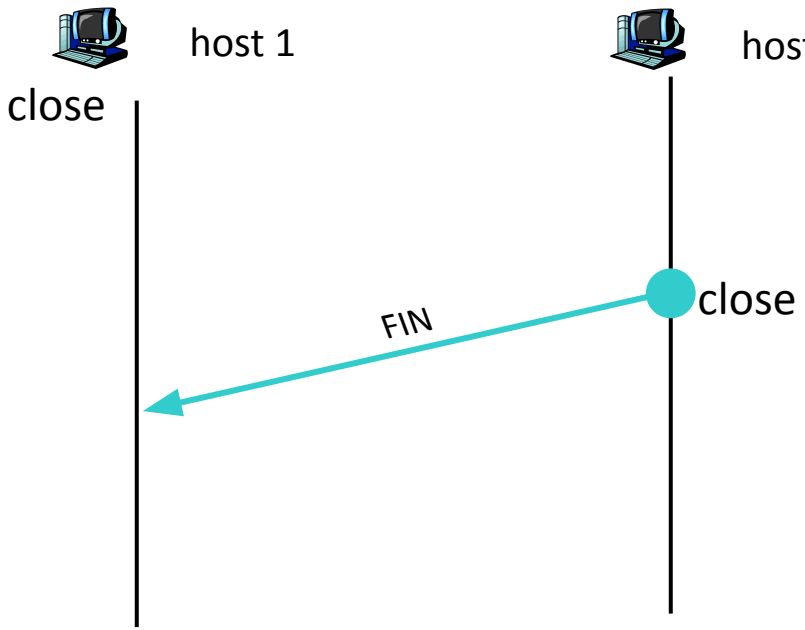
Contudo, o fluxo de dados pode continuar no sentido oposto

Término da conexão TCP



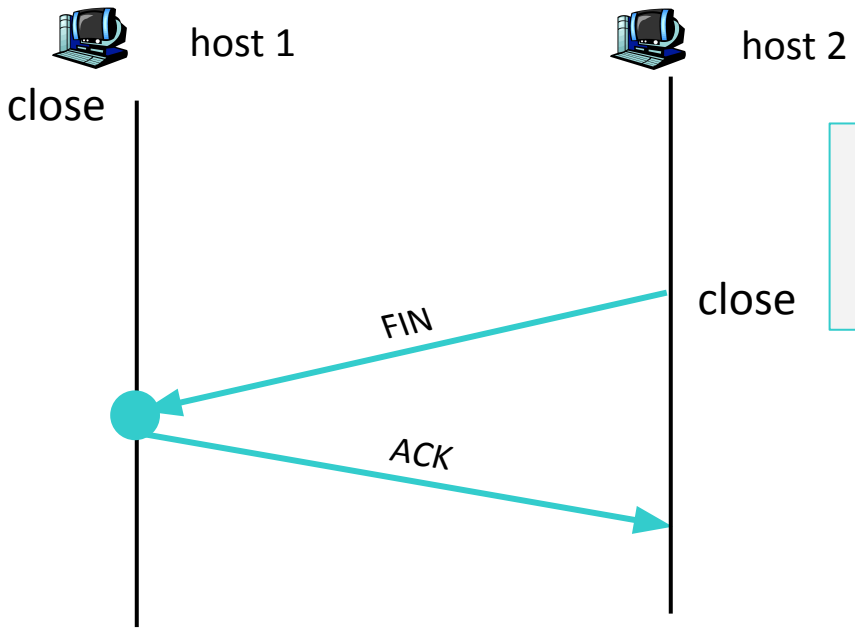
Passo 3) O host 1 recebe o ACK e fecha a conexão no sentido host 1 ⇒ host 2

Término da conexão TCP



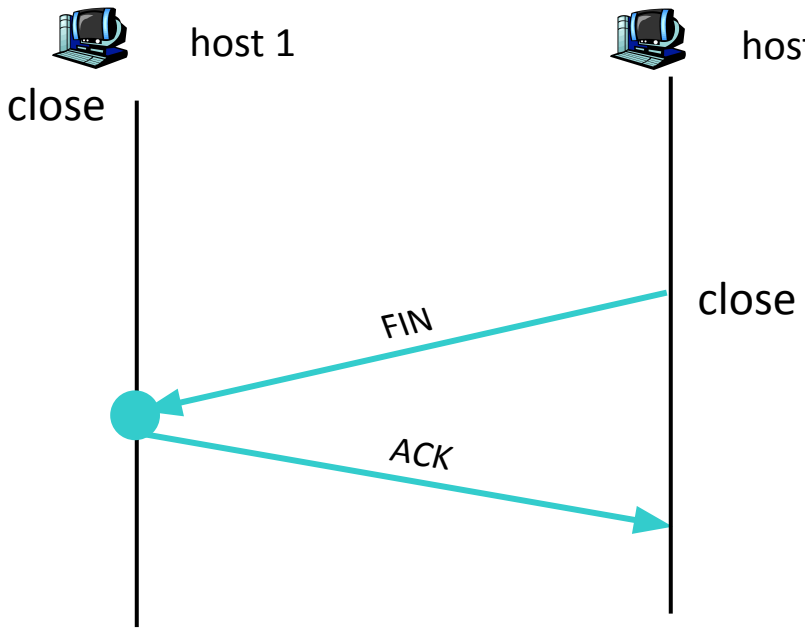
Passo 4) O host 2 envia o segmento TCP FIN para o host 1

Término da conexão TCP



Passo 5) O host 1 recebe o FIN e o responde com um ACK

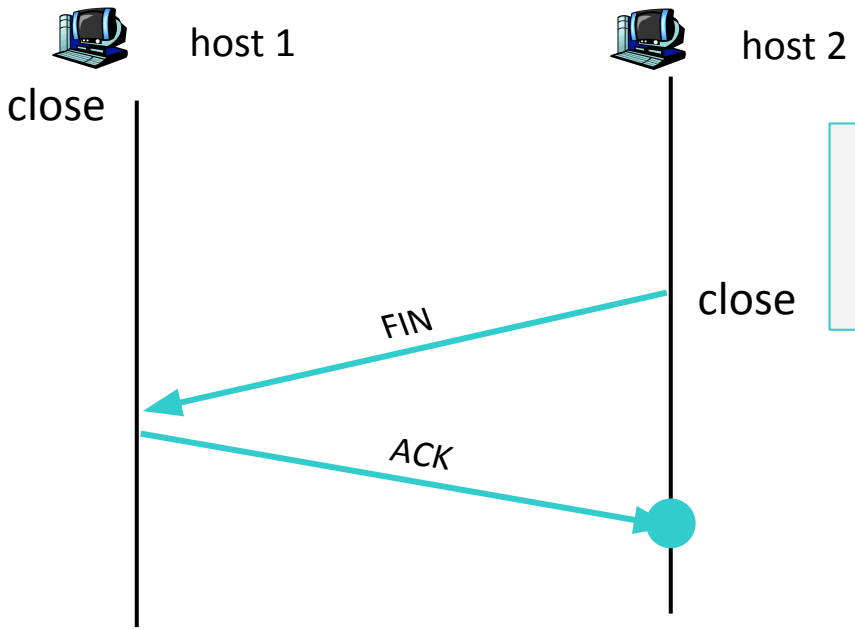
Término da conexão TCP



Passo 5) O host 1 recebe o FIN e o responde com um ACK

Em seguida, o host 1 fecha a conexão no sentido host 2 ⇒ host 1

Término da conexão TCP



Passo 6) O host 2 recebe o ACK e fecha a conexão no sentido host 2 ⇒ host 1

Controle de Fluxo

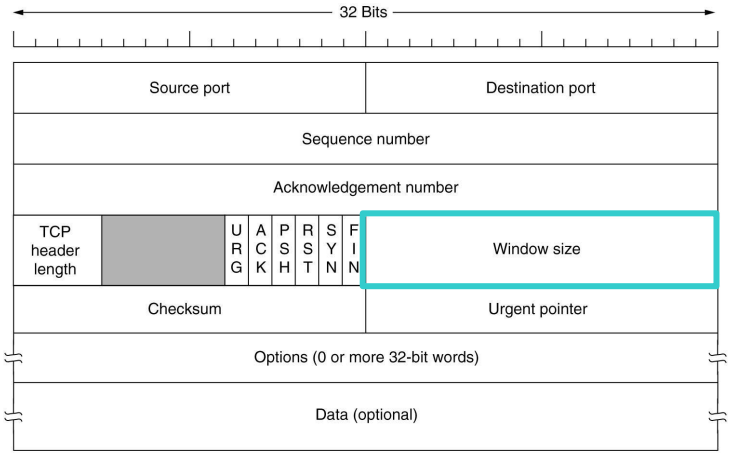
- Evitar que o emissor esgote a capacidade do receptor
- No estabelecimento da conexão, um *buffer* de recepção é alocado sendo tamanho é informado para a entidade par no campo Window size
- Em toda confirmação, envia-se o espaço disponível nesse *buffer* e esse espaço é chamado de janela

Controle de Fluxo

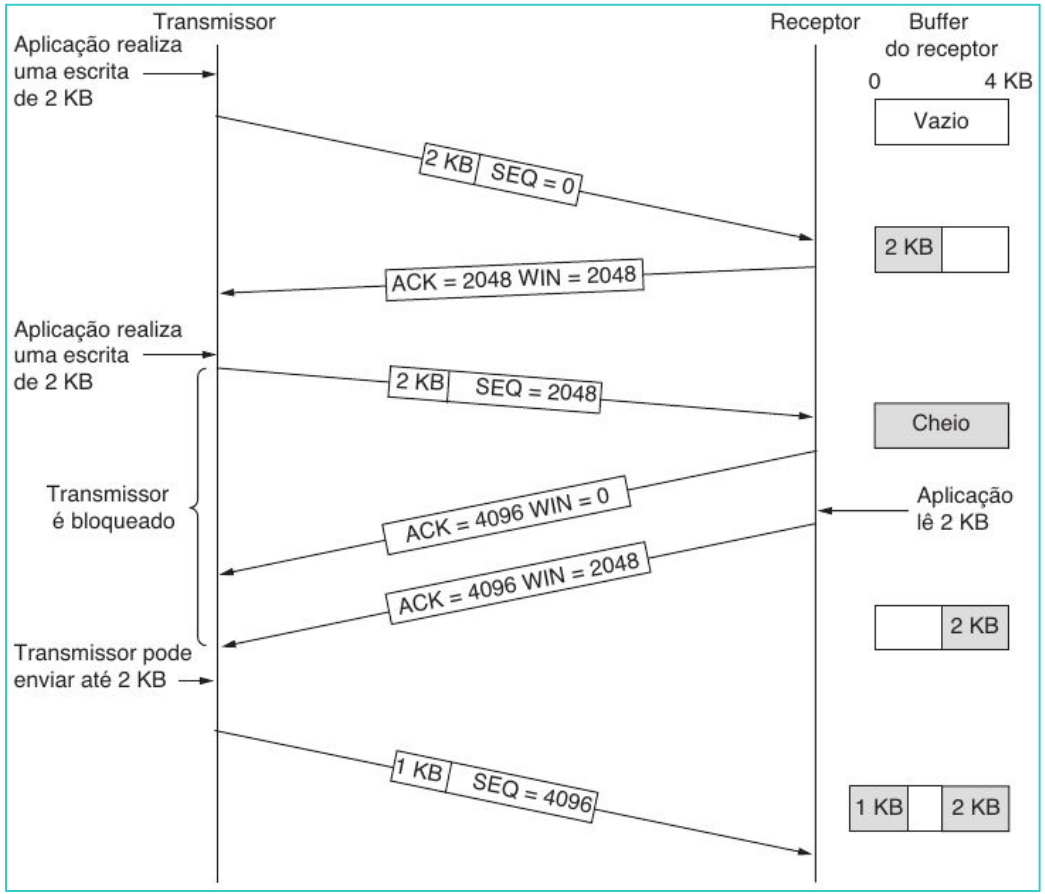
- Emissor não esgota os *buffers* de recepção enviando dados rápido demais
    - Receptor: Informa explicitamente ao emissor qual é a área livre em seu buffer através do campo *Windows size*
    - Emissor: Mantém a quantidade de dados transmitidos mas não reconhecidos menor que o último *Windows size* recebido

Window size (16 bits)

- Indica quantos bytes podem ser enviados a partir do byte confirmado



Gerenciamento de Janelas no TCP



Controle de Congestionamento

- Quando a carga oferecida a qualquer rede é maior que sua capacidade, acontece um congestionamento
- Diferente de controle de fluxo!
- Sintomas:
  - Perda de pacotes (saturação de *buffer* nos roteadores)
  - Atrasos grandes (filas nos *buffers* dos roteadores)
- Um dos 10 problemas mais importantes na Internet!

Considerações sobre o Controle de Congestionamento

- O congestionamento da rede pode ser piorado se a camada de transporte retransmite pacotes que não foram perdidos

- Esse problema pode causar até um colapso da rede

Como detectar congestionamento?

- O TCP usa a quantidade de pacotes perdidos (pacotes com *timeout*) como uma medida de congestionamento

- O TCP reduz a taxa de retransmissão à medida que esse valor aumenta

Considerações sobre o Controle de Congestionamento

- Antigamente, um *timeout* causado por um pacote perdido podia ter sido provocado por
  - Ruído em uma linha de transmissão, ou
  - O pacote foi descartado em um roteador congestionado

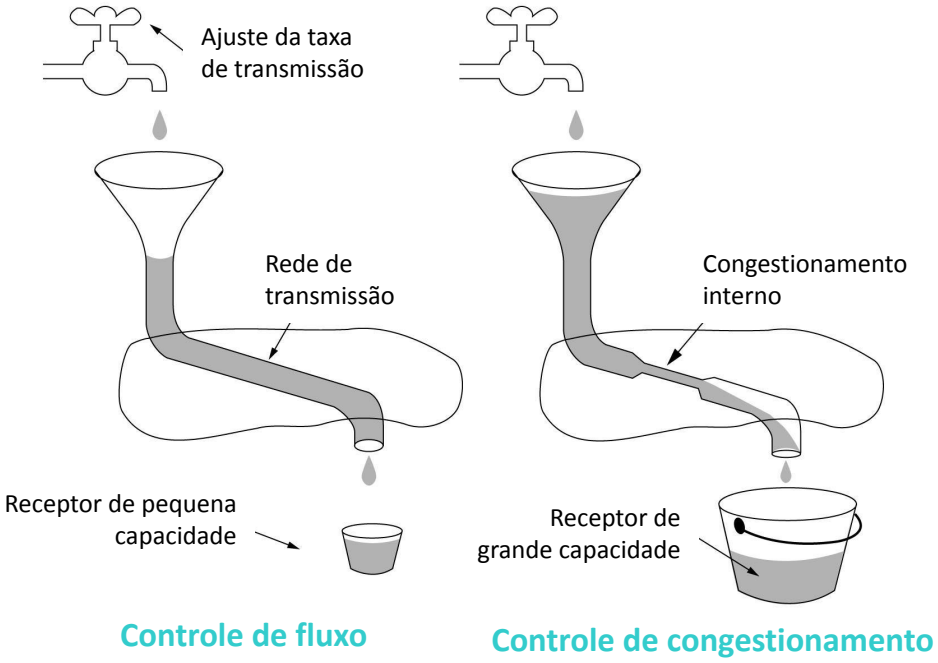
- Hoje em dia, a perda de pacotes devido a erros de transmissão é relativamente rara

- A maioria dos *timeouts* de transmissão na Internet se deve a congestionamentos

Dois Problemas em Potenciais na Internet

- **Capacidade da rede** ⇒ Controle de Congestionamento
- **Capacidade do receptor** ⇒ Controle de Fluxo

Controle de Fluxo vs. de Congestionamento



Número de Bytes que podem ser Transmitidos

- Cada emissor mantém duas janelas:
  - Janela fornecida pelo receptor (controle de fluxo)
  - Janela de congestionamento (controle de congestionamento)
- O número de bytes que podem ser transmitidos é o valor mínimo entre as duas janelas

Algoritmo de Iniciação Lenta

- Quando uma conexão é estabelecida, a janela de congestionamento é igual ao tamanho do segmento máximo em uso na conexão
- Cada rajada confirmada duplica a janela de congestionamento
- O crescimento exponencial da janela de congestionamento acontece até que:
  - Um limiar seja atingido,
  - Aconteça um *timeout*, ou
  - A janela do receptor seja alcançada

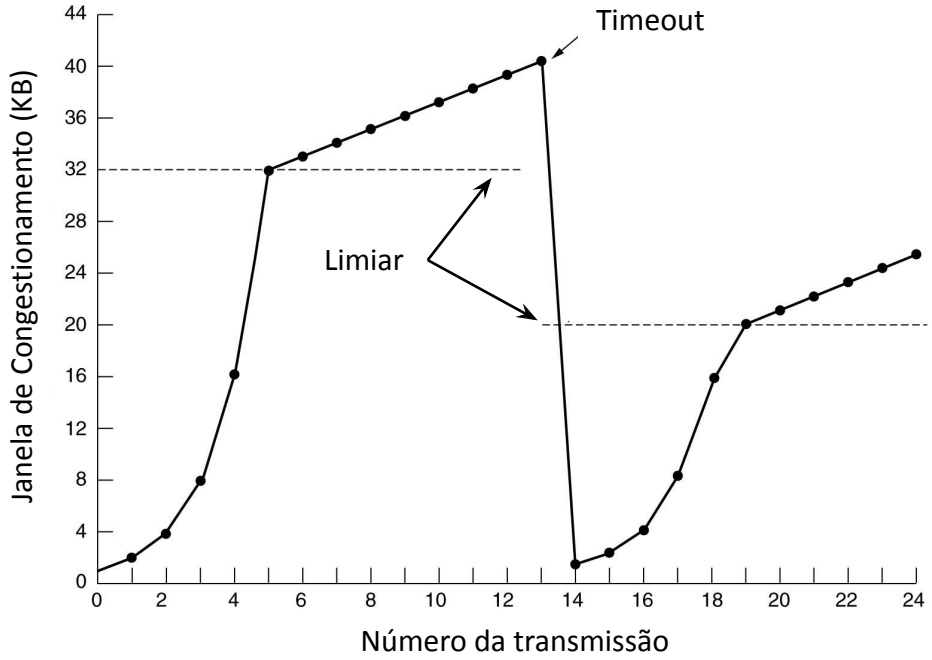
Término do Crescimento Exponencial

- **Um limiar é atingido**: as transmissões bem-sucedidas proporcionam um crescimento linear à janela de congestionamento

- **Acontece um *timeout***:
  - O limiar é definido como a metade da janela de congestionamento atual
  - Janela de congestionamento ⇒ segmento máximo
  - Inicialização lenta é usada

- **Janela de recepção é alcançada**: janela de congestionamento pára de crescer e permanece constante

Término do Crescimento Exponencial



A transmissão de mensagens é feita de forma exponencial até atingir um dado valor, quando passa a aumentar mais lentamente

Gerenciamento de Temporizadores

- Quando um segmento é enviado, um *timer* de retransmissão é ativado
- Se o segmento for confirmado antes do *timer* expirar, ele será interrompido
- Se o *timer* expirar antes da confirmação chegar, o segmento será retransmitido e o *timer* será disparado novamente

Gerenciamento de Temporizadores

- **Tempo de viagem de ida e volta (RTT)**: tempo transcorrido desde o instante em que um segmento é enviado até o instante em que ele é reconhecido

- Para cada conexão, o TCP mantém uma variável, RTT, que é a melhor estimativa no momento para o tempo de percurso de ida e volta até o destino em questão

Como Escolher o Valor do *Timer* do TCP?

- Maior que o RTT (RTT varia)
- Muito curto: Retransmissões desnecessárias, sobrecarregando a Internet com pacotes inúteis
- Muito longo: O desempenho será prejudicado devido ao longo retardo de retransmissão sempre que um pacote se perder

Como Escolher o Valor do *Timer* do TCP?

- O TCP atualiza a variável RTT de acordo com a fórmula:

$$RTT = \alpha RTT + (1- \alpha)M$$

onde:

  - M: último RTT medido
  - α: fator de suavização que determina o peso dado ao antigo valor

Exercícios

Exercício (1)

- Faça um paralelo entre a porta para a camada de transporte e o endereço de rede para a camada de rede.

Exercício (2)

- Por que a camada de transporte dá sentido à pilha de protocolos?

Exercício (3)

- O significa uma aplicação do tipo *one shot*? Os protocolos TCP pode ser utilizado na camada de transporte para esse tipo de aplicação? E o UDP? Justifique suas respostas.

Exercício (4)

- Explique como executamos as primitivas de soquetes para o TCP em uma aplicação do tipo cliente-servidor

Exercício (5)

- Como os campos Sequence e Acknowledgement numbers atuam na transferência confiável de dados do TCP

Exercício (6)

- Explique como os campos SYN, ACK e FIN funcionam no estabelecimento e no término de conexões TCP

Exercício (7)

- Descreva o controle de fluxo no TCP

Exercício (8)

- Descreva o controle de congestionamento no TCP