Redes de Computadores

Comunicação fim-a-fim (transporte)

UDP

TCP

RPC

UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Serviço fim-a-fim

- Fornecem um canal (virtual) de comunicação entre aplicações em computadores distintos
- Rede subjacente (IP) opera em *best-effort*
  - pacotes perdidos
  - pacotes fora de ordem
  - pacotes duplicados
  - tamanho limitado
  - atrasos arbitrários

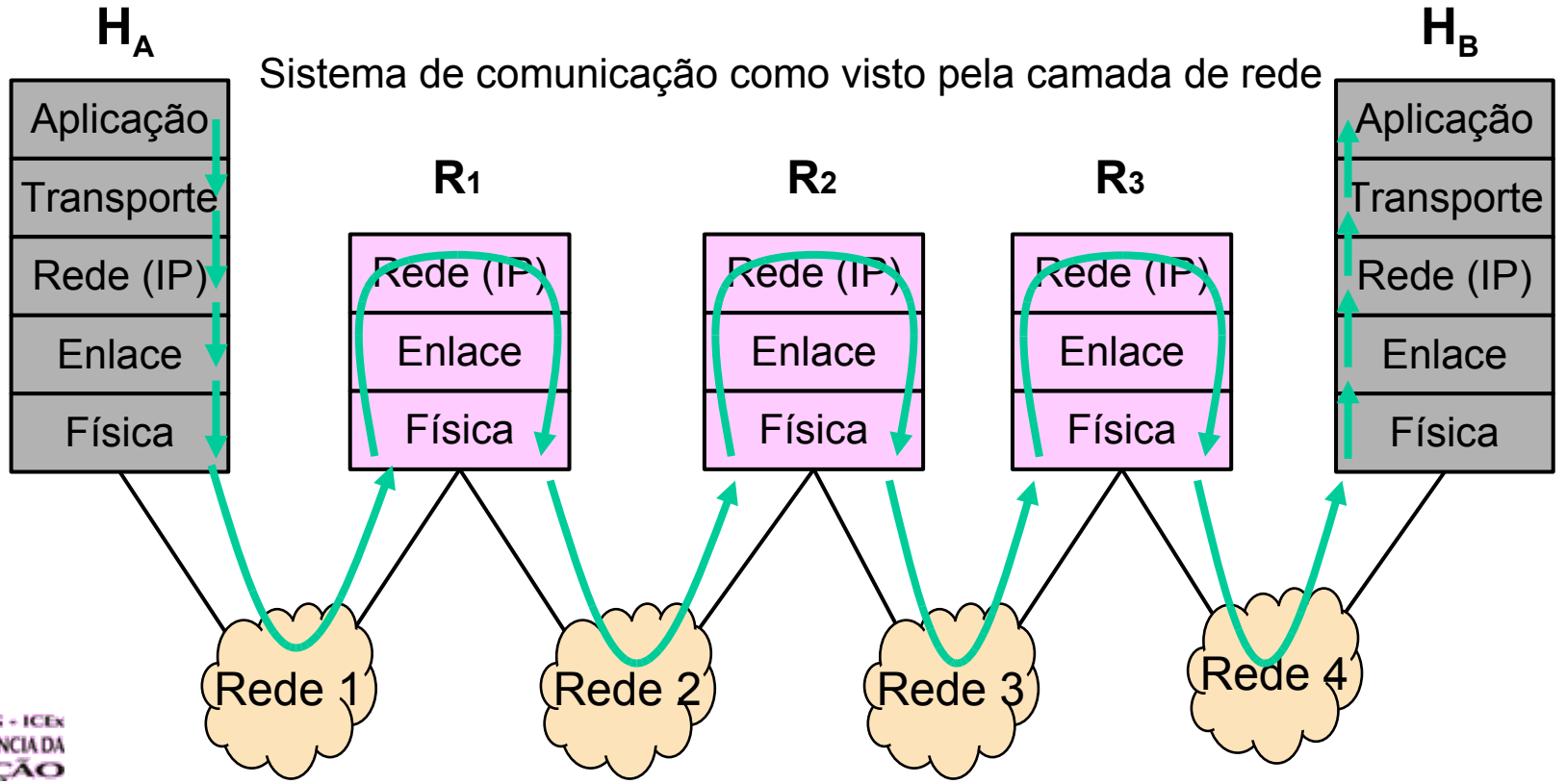
UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Serviço fim-a-fim

- Serviços fim-a-fim usuais
  - entrega garantida das mensagens
  - mensagens entregues em ordem
  - entrega de no máximo uma cópia de cada pacote
  - uso de mensagens de tamanho arbitrário
  - sincronização entre fonte e destino
  - identificação de múltiplas aplicações em um *host*

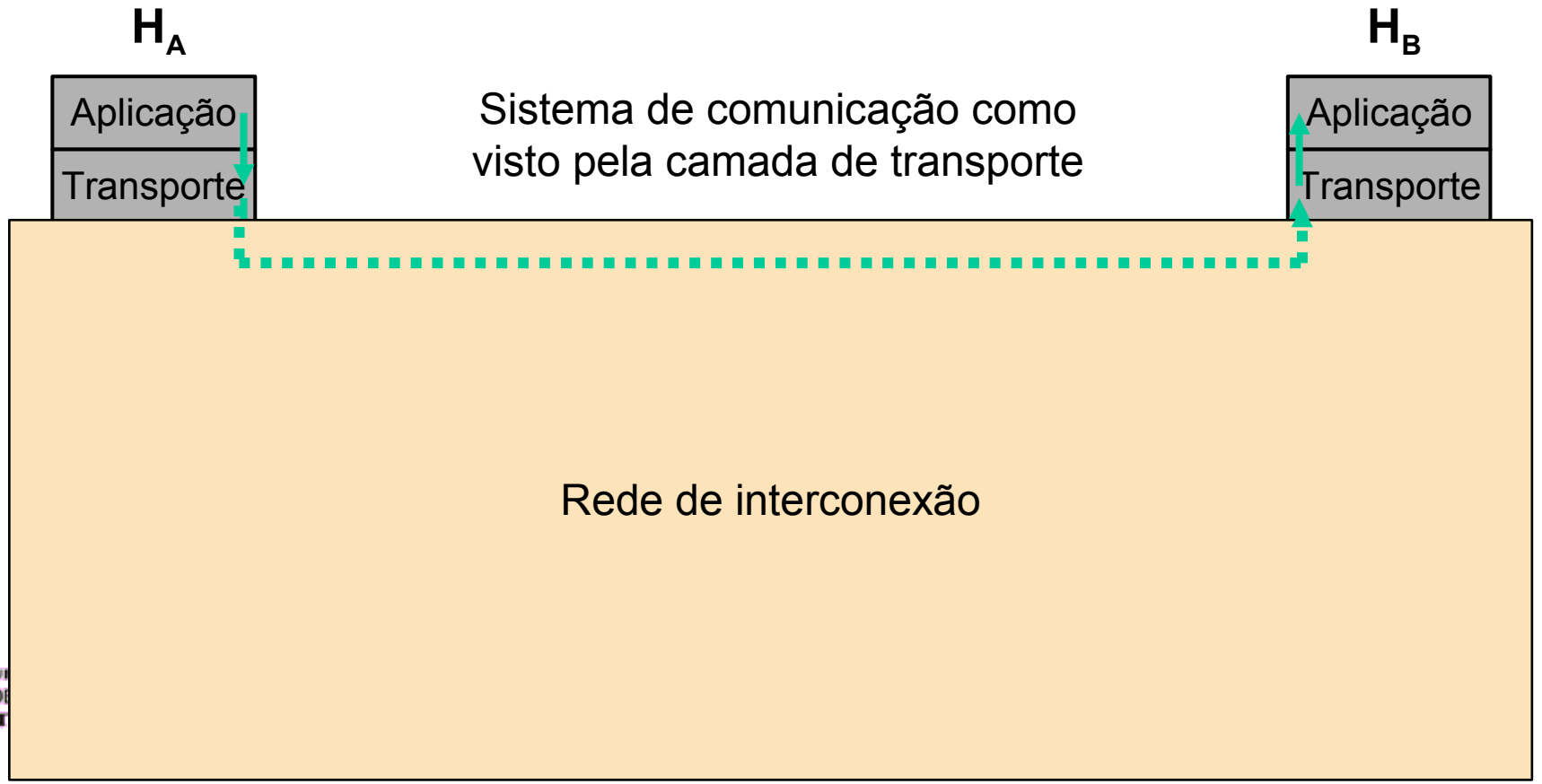
Serviço fim-a-fim

- Canais são implementados em software, já que o sub-sistema de rede não provê tais facilidades
- Mensagens TCP e UDP são encapsuladas em datagramas (pacotes) IP



Serviço fim-a-fim

- Canais são implementados em software, já que o sub-sistema de rede não provê tais facilidades
- Mensagens TCP e UDP são encapsuladas em datagramas (pacotes) IP



Serviços fim-a-fim de interesse

- Um “serviço mínimo”: UDP
- Um fluxo de bytes bi-direcional: TCP
- Requisições e respostas: RPC(s)

User Datagram Protocol (UDP)

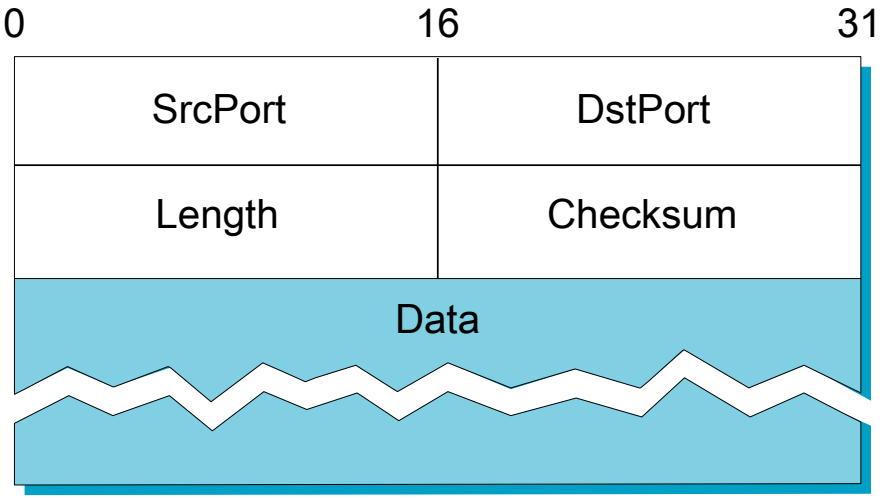
- Serviço sem conexão, não confiável, sem ordem
- Com detecção de erro (opcional)
- Identificação de porto(a) (*"port"*) p/ multiplexação

- Quando usar?
  - Aplicações que implementam entrega confiável
  - Quando a latência deve ser mínima (sem conexão)
  - Aplicações do tipo *request/response*
  - Serviços em que dados podem ser perdidos

UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

User Datagram Protocol (UDP)

- Source/Destination port:
  - Portas usadas pelos processos
  - *Source port* é opcional
- Length:
  - Tamanho do datagrama (octetos)
- Checksum:
  - Verificação de erro no datagrama (opcional)
  - Inclui cabeçalho, dados e *pseudo-header*

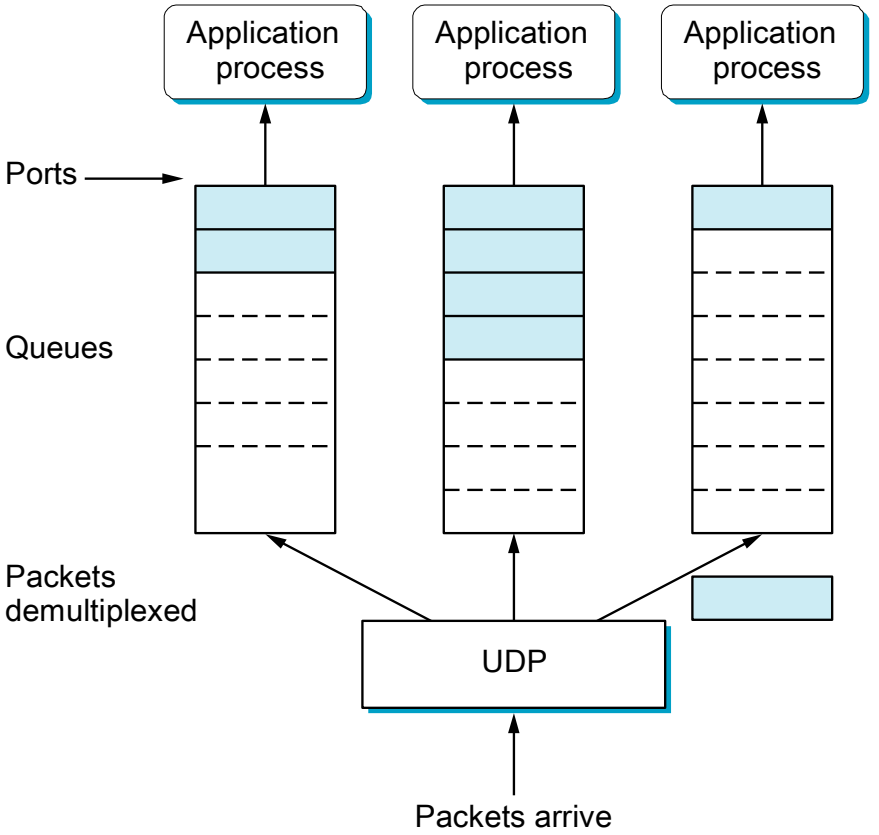


Parte do cabeçalho IP:
endereços de origem/destino

UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Portos

- Identificam os processos de origem e destino
- Sistema operacional oferece interface de acesso
- Pressupõem *buffers* de recepção e transmissão



Portos

- Servidores têm portos bem conhecidos
  - /etc/services (linux), \win*\services (win*)

```
ftp-data        20/tcp          # File Transfer [Default Data]
ftp-data        20/udp          # File Transfer [Default Data]
ftp             21/tcp          # File Transfer [Control]
ssh             22/tcp          # SSH Remote Login Protocol
telnet          23/tcp          # Telnet
smtp            25/tcp mail     # Simple Mail Transfer
domain          53/tcp          # Domain Name Server
http            80/tcp www      # World Wide Web HTTP
pop3            110/tcp         # Post Office Protocol - Ver. 3
nntp            119/tcp         # Network News Transfer Protocol
ntp             123/tcp         # Network Time Protocol
snmp            161/tcp         # SNMP
snmptrap        162/tcp         # SNMPTRAP
```

Serviços fim-a-fim de interesse

- Um “serviço mínimo”: UDP
- Um fluxo de bytes bi-direcional: TCP
- Requisições e respostas: RPC(s)

Transmission Control Protocol (TCP)

- Serviço orientado a conexão (circuito virtual)
- Confiável (detecção e correção de erros)
- Controle automático de *buffers*
- Controle de fluxo por janela deslizante
  - Evita que o transmissor afogue o receptor
- Controle de congestionamento
  - Evita que o transmissor alague a rede
- Identificação de portos
- Sequência de bytes não estruturada (*byte stream*)

TCP: identificação de conexão

- Portas, conexões e endpoints:
  - TCP identifica os dois pontos em comunicação como terminações da conexão (*endpoints*)
  - Um *endpoint* é um par de inteiros da forma:

Host , Port

  - A conexão é identificada por um par de *endpoints*

(128.9.0.32, 1184) (128.10.2.3, 25)

TCP: identificação de conexão

- Portas, conexões e *endpoints*:
  - *Endpoints* permitem que uma determinada porta possa ser compartilhada por múltiplas conexões

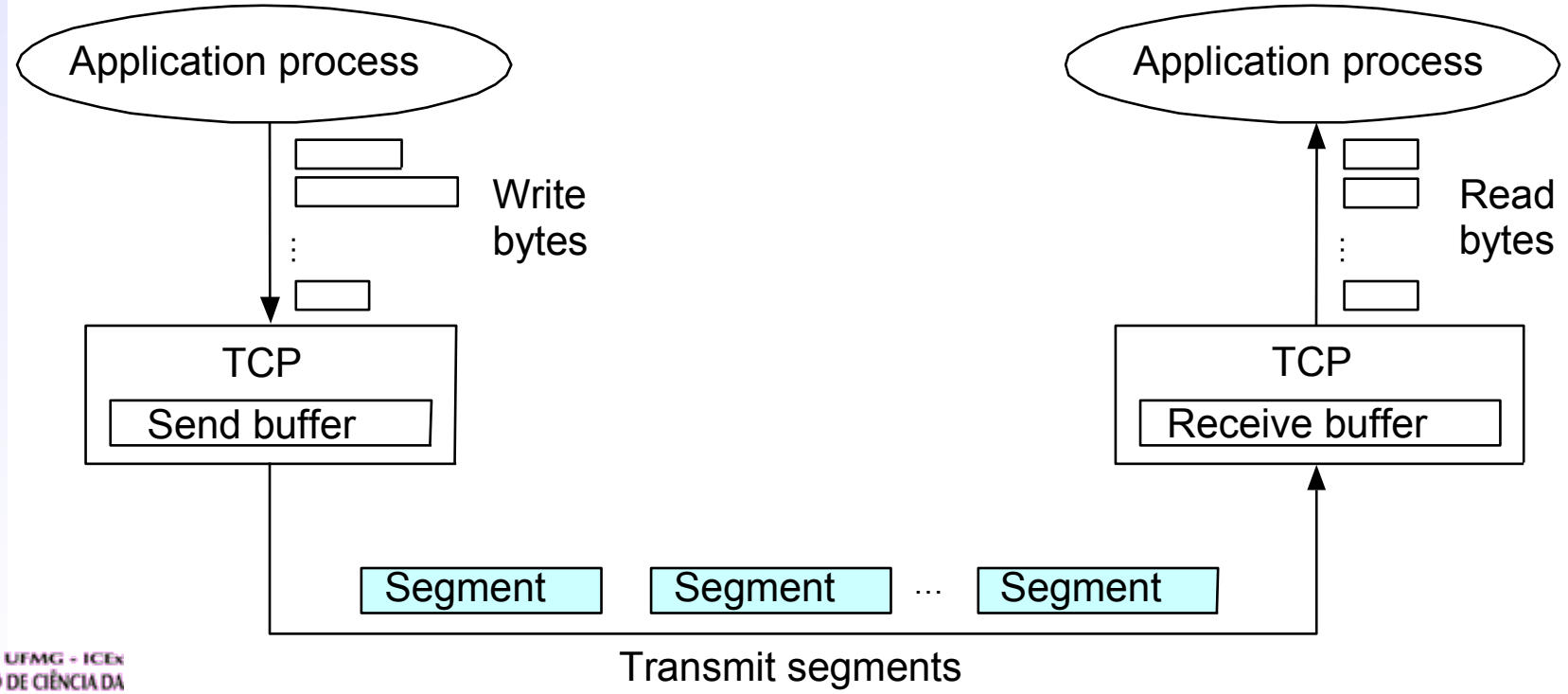
(128.9.0.32, 1184) (128.10.2.3, 25)

(128.2.1.27, 1184) (128.10.2.3, 25)

(128.2.1.27, 2167) (128.10.2.3, 25)

Transmission Control Protocol (TCP)

- Sequência de bytes não estruturada (*byte stream*)
  - aplicação transmissora escreve bytes
  - seqüência é dividida em **segmentos** para o envio
  - aplicação receptora lê bytes

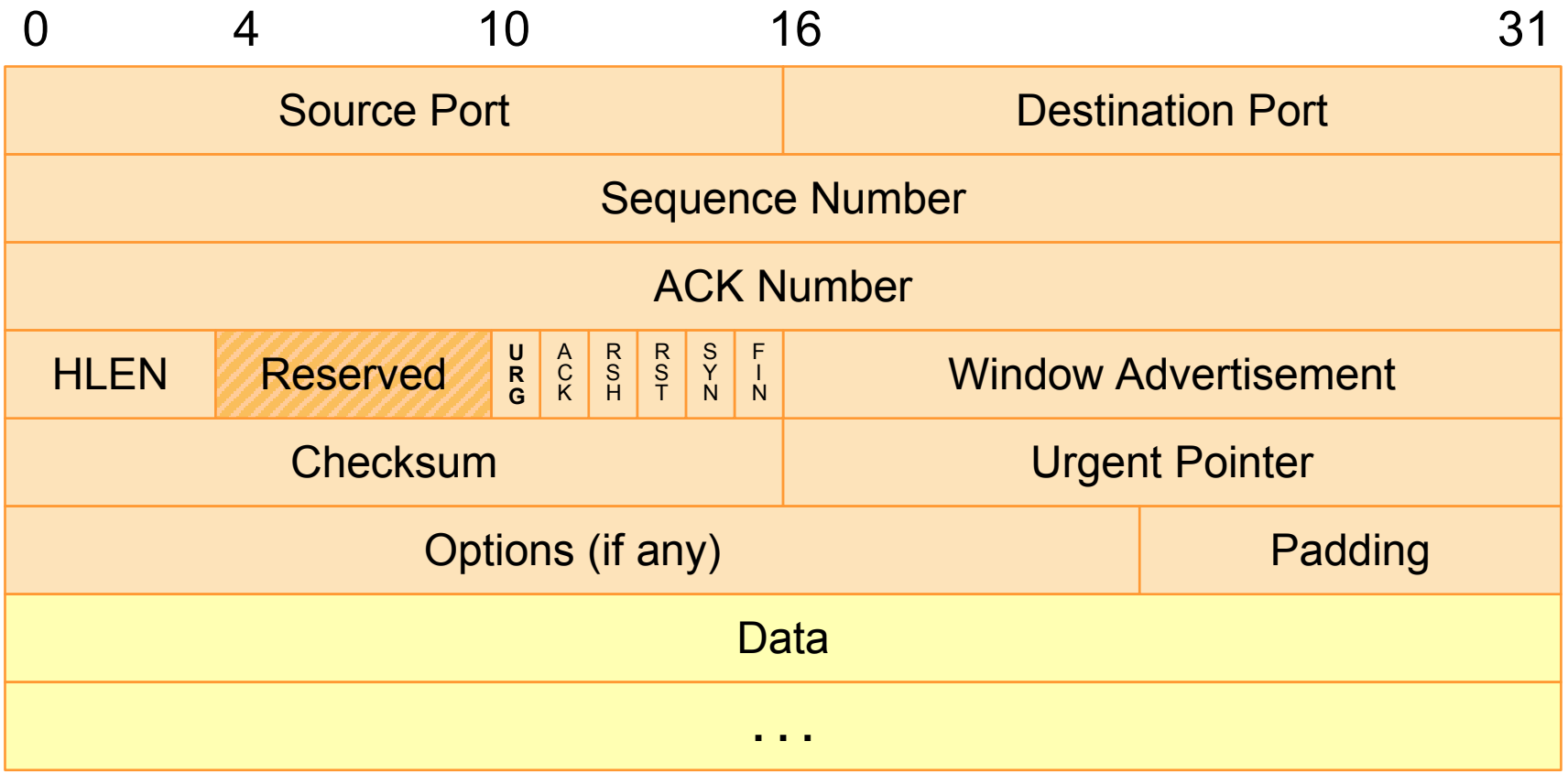


UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Transporte “é mais complicado” que enlace

- Potencialmente conecta muitos hosts diferentes
  - requer estabelecimento e terminação explícita de conexão
- RTTs potencialmente diferentes
  - requer um mecanismo de temporização adaptativa
- Potencialmente atrasos longos na rede
  - precisa se preparar para a chegada de pacotes muito antigos
- Potencialmente destinos podem ter diferentes capacidades
  - precisa acomodar destinos com diferentes capacidades
- Redes com diferentes capacidades
  - precisa ser capaz de lidar com congestionamento na rede

Segmento TCP

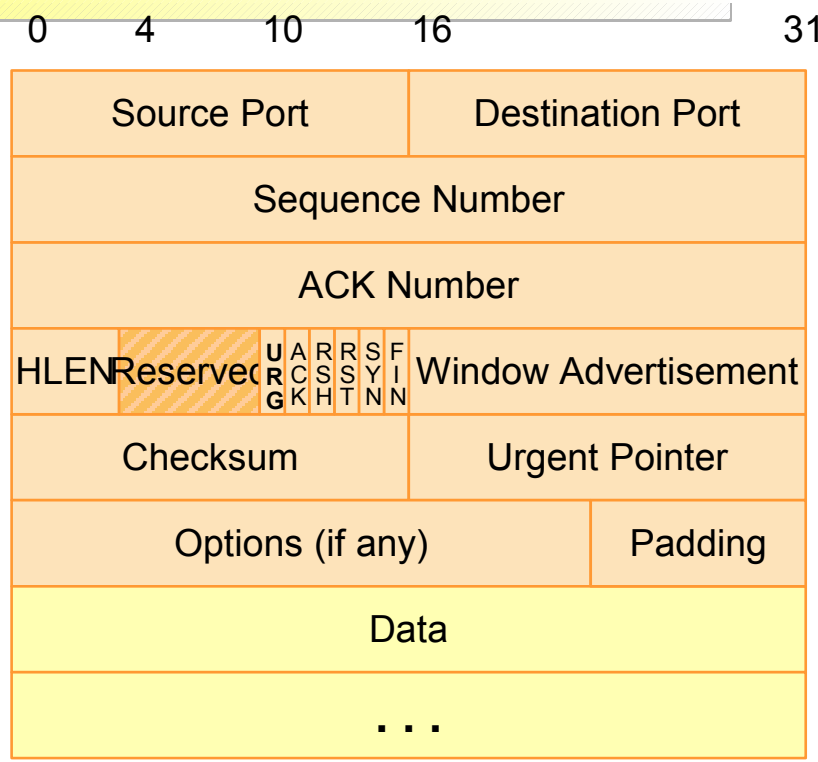


- **Source/Destination Port, checksum**:
  - Mesma utilização de UDP

UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Segmento TCP

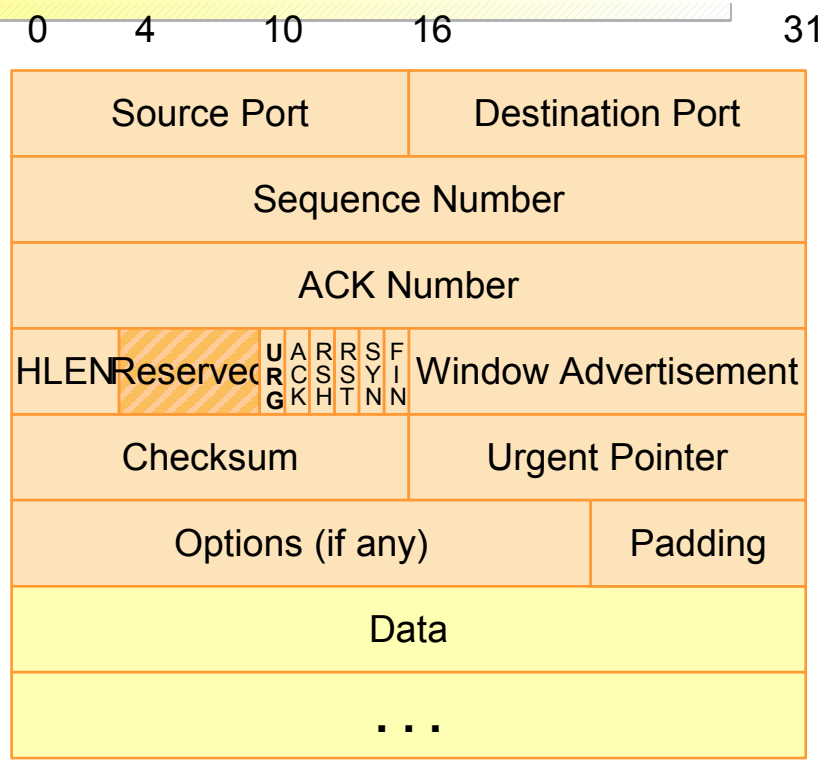
- ***Sequence Number:***
  - Posição do segmento no *stream*
- ***Ack Number:***
  - Número do prox. byte que destino espera receber
  - Pacote que vai em um sentido confirma último pacote recebido no sentido contrário (*piggybacking*)

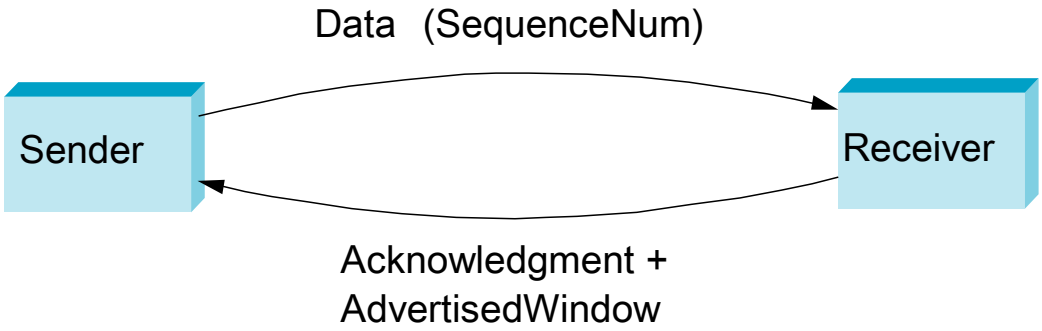


UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Segmento TCP

- ***Window advertisement :***
  - Possibilita o controle de fluxo
  - Evita a sobre-escrita de *buffers*
  - Informa o tamanho da janela disponível no receptor
  - Transmissor não pode enviar dados além do que o receptor pode receber

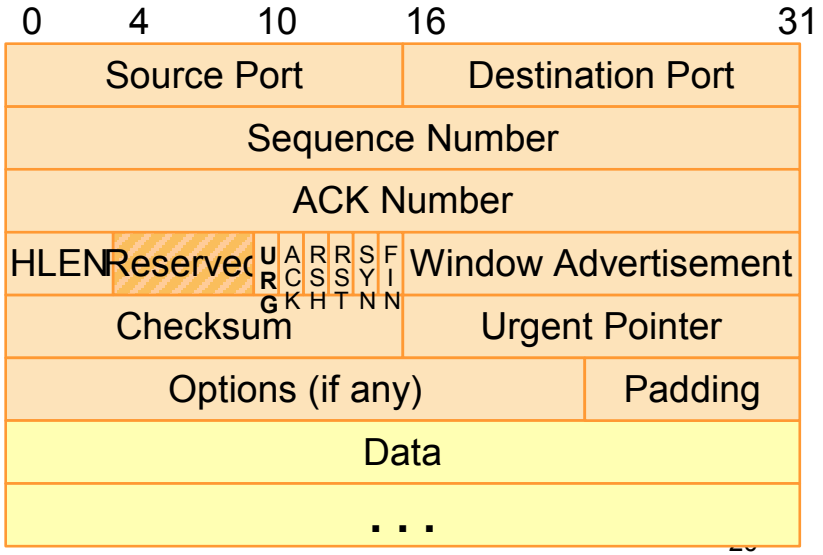




UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

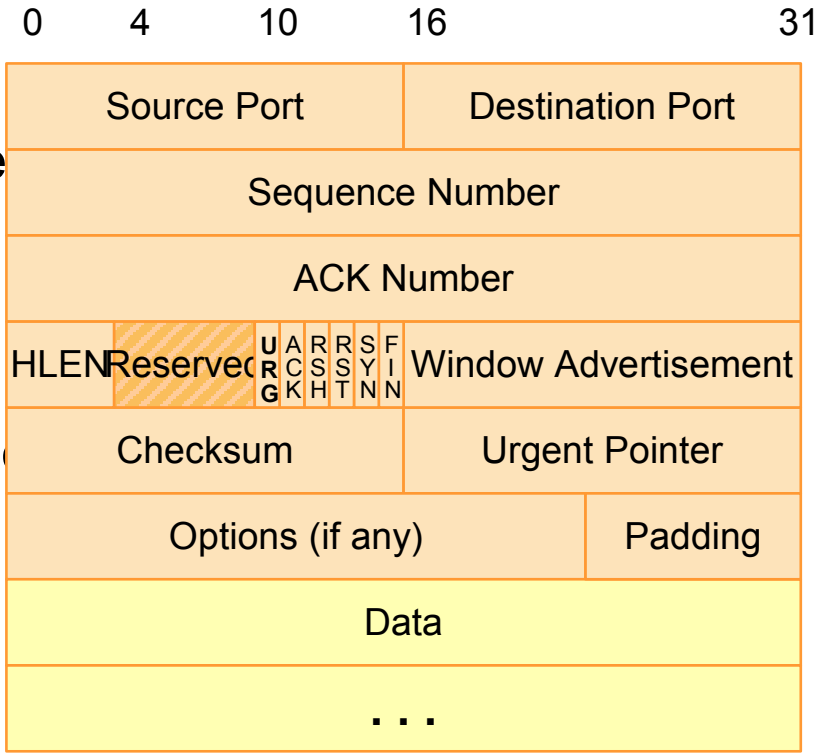
Segmento TCP

- **Code Bits** (flags):
  - **URG**: Campo *Urgent Pointer* válido (p.ex., Ctrl-C)
  - **ACK**: Campo Ack válido
  - **PSH**: Segmento requer entrega imediata
  - **RST**: Término devido a erro
  - **SYN**: Sincroniza números de seqüência (abertura)
  - **FIN**: Origem finalizou envio (fechamento)
- **Urgent Pointer**:
  - Posição de dados urgentes no segmento
    (se existirem)



Segmento TCP

- ***Options***: extensões do protocolo
  - Utilizadas para adicionar funcionalidade sem exigir uma nova versão
- Encontradas em quase todas as implementações modernas
  - Essenciais para se garantir desempenh nas redes de alta capacidade (LFNs)



UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

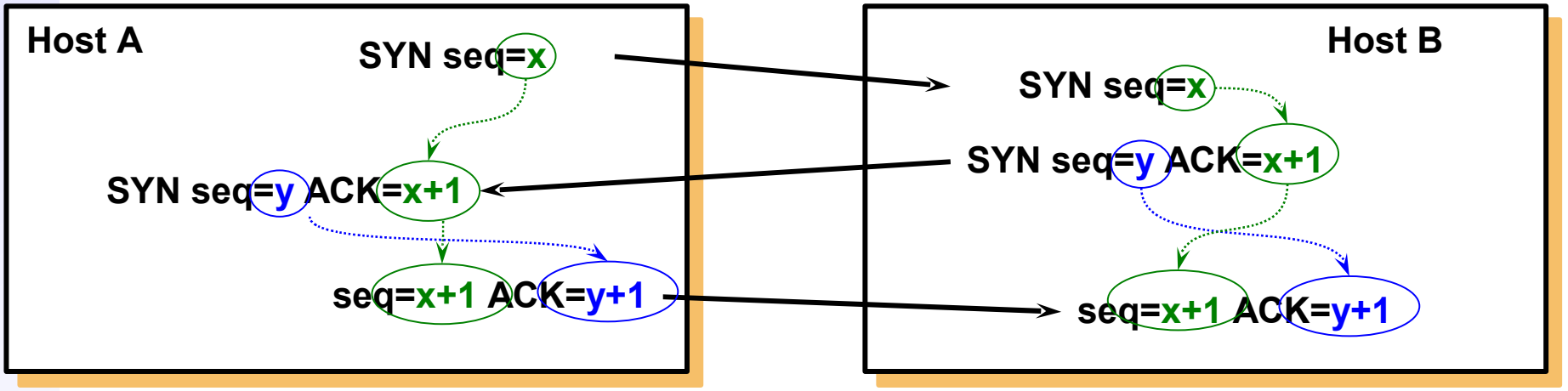
TCP: conexão com três fases

- Estabelecimento
  - TCP requer que aplicações reconheçam a nova conexão como única e inconfundível
  - Pacotes de conexões anteriores não são podem ser tomados como pacotes válidos
- Transferência de dados
  - Comunicação *full-duplex*
  - Interface do serviço é uma seqüência de bytes
- Término da conexão
  - TCP garante a entrega de todos os dados antes de fechar uma conexão a pedido da aplicação
- Fases representadas no diagrama de estados TCP

UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

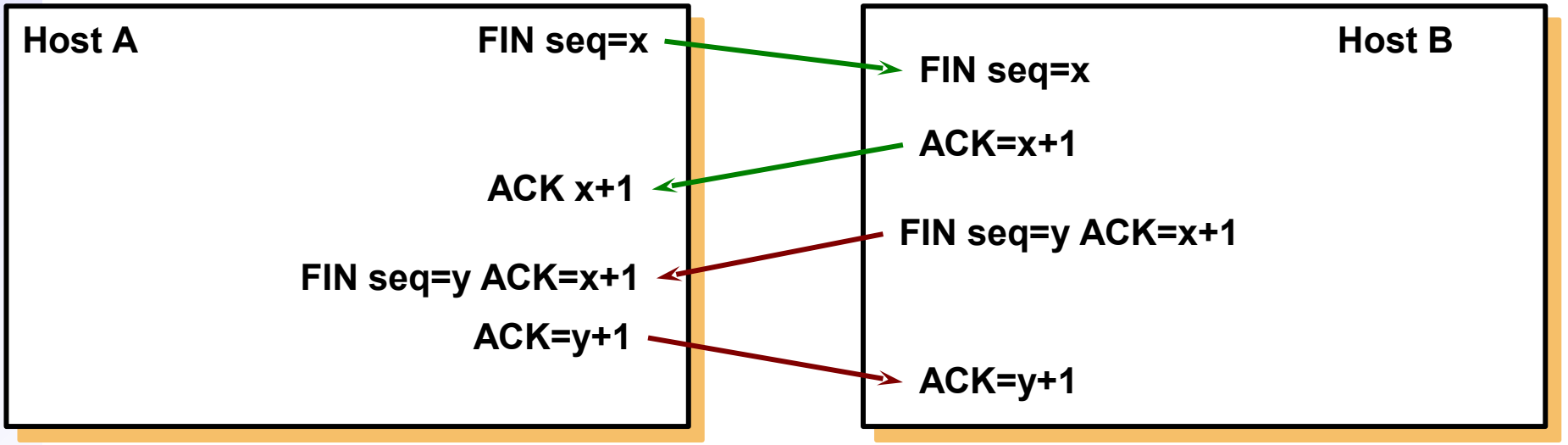
TCP: controle da conexão

- TCP usa segmentos especiais:
  - Segmento de sincronização (SYN *segment*) para descrever mensagens durante a conexão
- Estabelecimento: *three-way handshake*



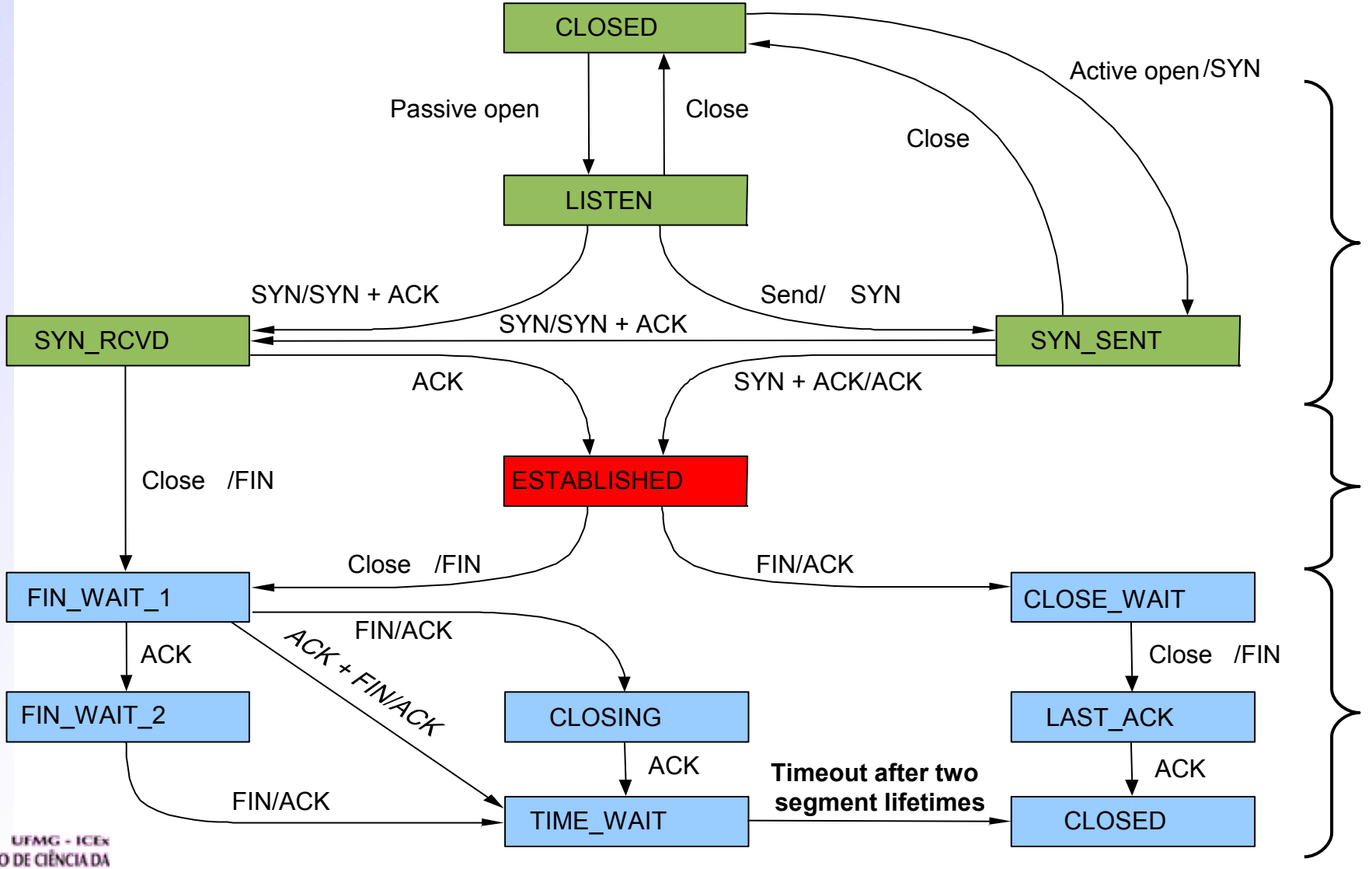
TCP: controle da conexão

- TCP usa segmentos especiais:
  - Segmento de término (FIN (*finish*) *segment*) para descrever mensagens durante a desconexão
- Fechamento: dois *handshakes* independentes



UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

TCP: diagrama de estados



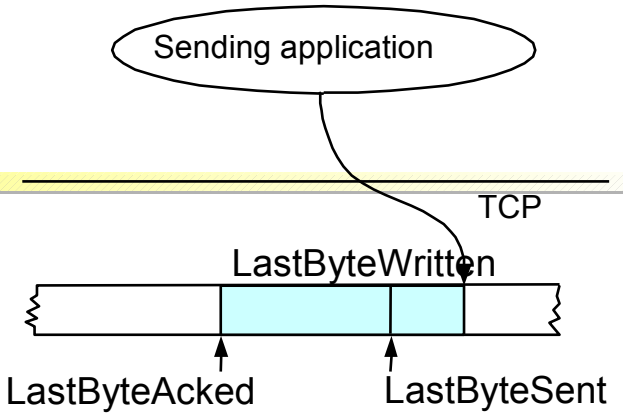
TCP: correção de erros

- Confirmação positiva (*acnowledgement*, ACK)
- Utiliza *piggybacking* no reconhecimento
  - Confirmações enviadas com dados no sentido oposto
- Dados podem ser recebidos fora de ordem
- Qualquer segmento recebido gera uma confirmação do último byte recebido em ordem

TCP: controle de fluxo

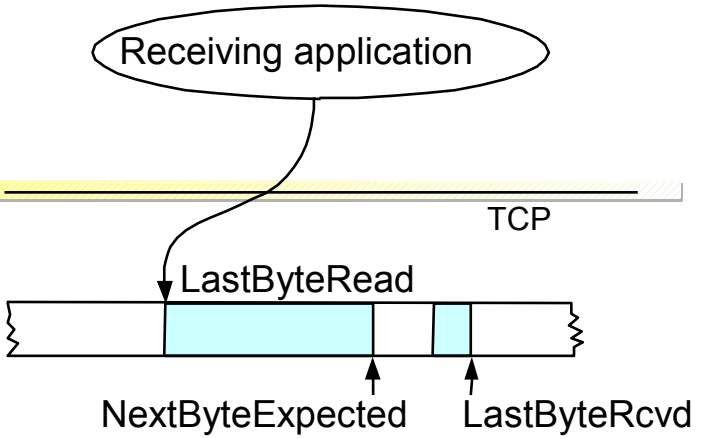
- Mecanismo de janela deslizante variável
- No estabelecimento da conexão, *buffer* é alocado
- Toda confirmação (desde o *three-way handshake*) informa o espaço disponível nesse *buffer*
  - A notificação que contém esse dado é chamada anúncio da janela (*window advertisement*)

TCP: controle de fluxo



- No transmissor
  - LastByteAcked <= LastByteSent
  - LastByteSent <= LastByteWritten
  - LastByteWritten – LastByteAcked <= MaxSndBuffer
  - LastByteSent - LastByteAcked < AdvWindow
  - buffer: [LastByteAcked . . LastByteWritten]
  - EffectiveWindow = AdvWindow – (LastByteSent-LastByteAcked)
  - Bloqueia o transmissor se
    - LastByteWritten – LastByteAcked + newBytes > MaxSndBuffer
  - Se AdvWindow = 0, persiste na transmissão

TCP: controle de fluxo

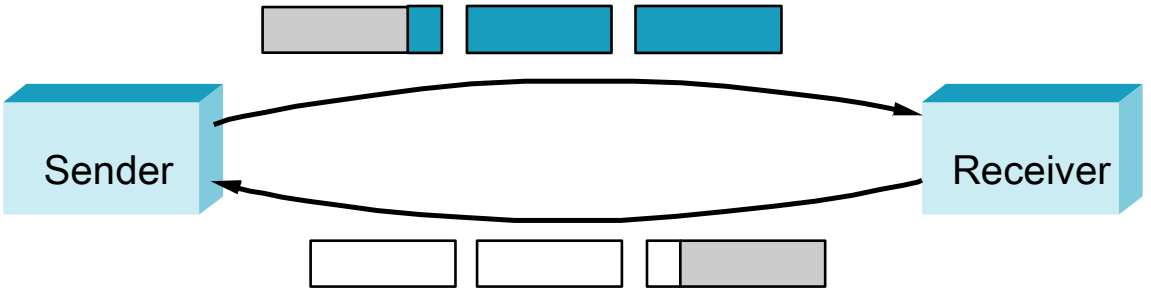


- No receptor:
  - LastByteRead < NextByteExpected
  - NextByteExpected <= LastByteRcvd + 1
    - janela deslizante > 1
  - BytesNotRead = NextByteExpected - LastByteRead
  - buffer: [LastByteRead . . LastByteRcvd]
  - AdvertisedWindow = MaxRcvBuffer – BytesNotRead
  - Sempre envia ACK para um segmento recebido

UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Silly Window Syndrome

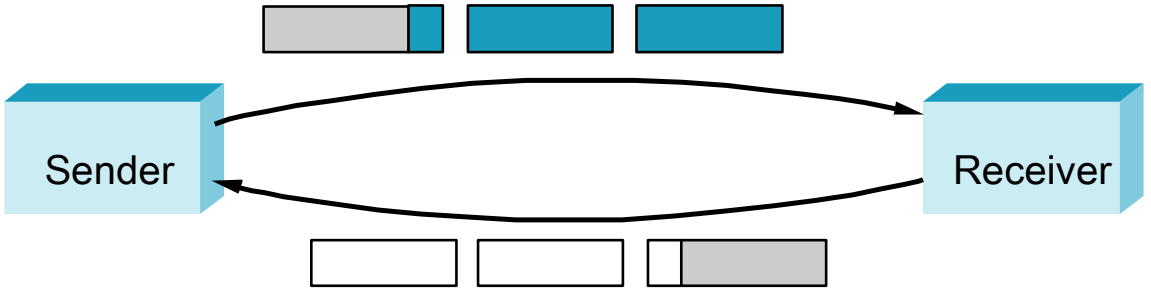
- Com que agressividade o transmissor deve explorar a abertura da janela?



- Quanto tempo esperar antes de enviar mais dados?
  - espera demais: prejudica o desempenho de aplicações interativas
  - espera pouco: utilização ruim da rede (muitos pacotes pequenos)
  - estratégias: temporizações ou “auto-controle” (*self-clocking*)

Silly Window Syndrome

- Com que urgência o transmissor deve enviar pacotes pequenos?



- Controle pelo receptor:
  - só abre uma janela fechada se tiver espaço para um segmento (MSS)
  - atrasa envio de ACKs (*delayed acknowledgements*)
- Controle pelo transmissor (algoritmo de Nagle):
  - um segmento incompleto só é enviado se não há nada em trânsito
    - se há dados para serem confirmados antes do segmento incompleto, espera
    - envia apenas ao receber o ACK do último segmento em trânsito ou ao completar um MSS

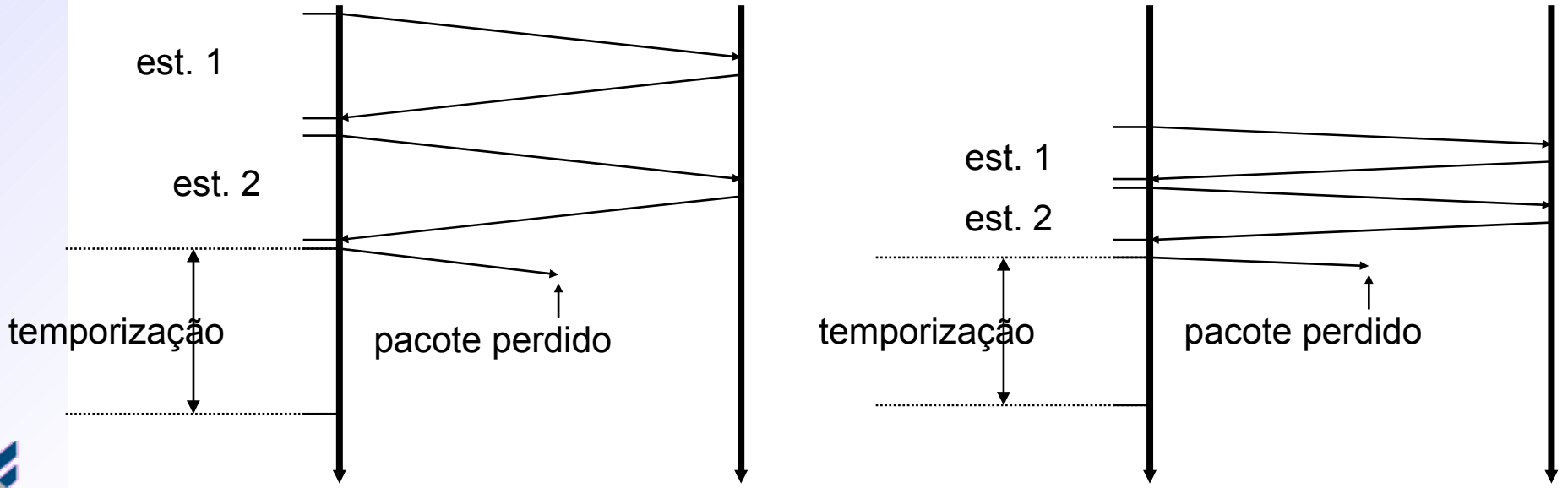
TCP: retransmissão adaptativa (ou temporização adaptável)

- Problema a ser resolvido:
  - Como configurar temporizadores para comunicações em LANs e WANs?
    - LANs: deve esperar pouco
    - WANs: deve esperar mais

UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

TCP: retransmissão adaptativa

- TCP adota temporizações variáveis
  - TCP monitora o atraso de alguns pacotes em cada conexão e modifica o temporizador de retransmissão para acomodar mudanças
  - Mudança é feita em função de análise estatística das mensagens transmitidas **com sucesso**



UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

TCP: retransmissão adaptativa

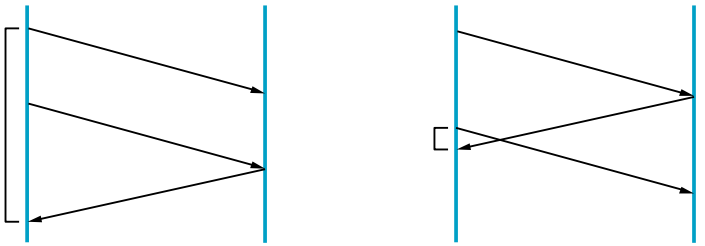
Algoritmos só são tão bons quanto o período do clock (500 ms)

- Algoritmo original:
  - medições com decaimento exponencial

    EstRTT = α x **EstRTT** + β x **SampleRTT**, α + β = 1

    TimeOut = 2 x **EstRTT**

- Algoritmo de Karn/Partridge:
  - descarta dados c/ retransmissões



- Algoritmo de Jacobson/Karels:
  - noção de variância embutida no valor de temporização

    TimeOut = **EstRTT** + 4 **Dev**

UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Retransmissão adaptativa: alg. original

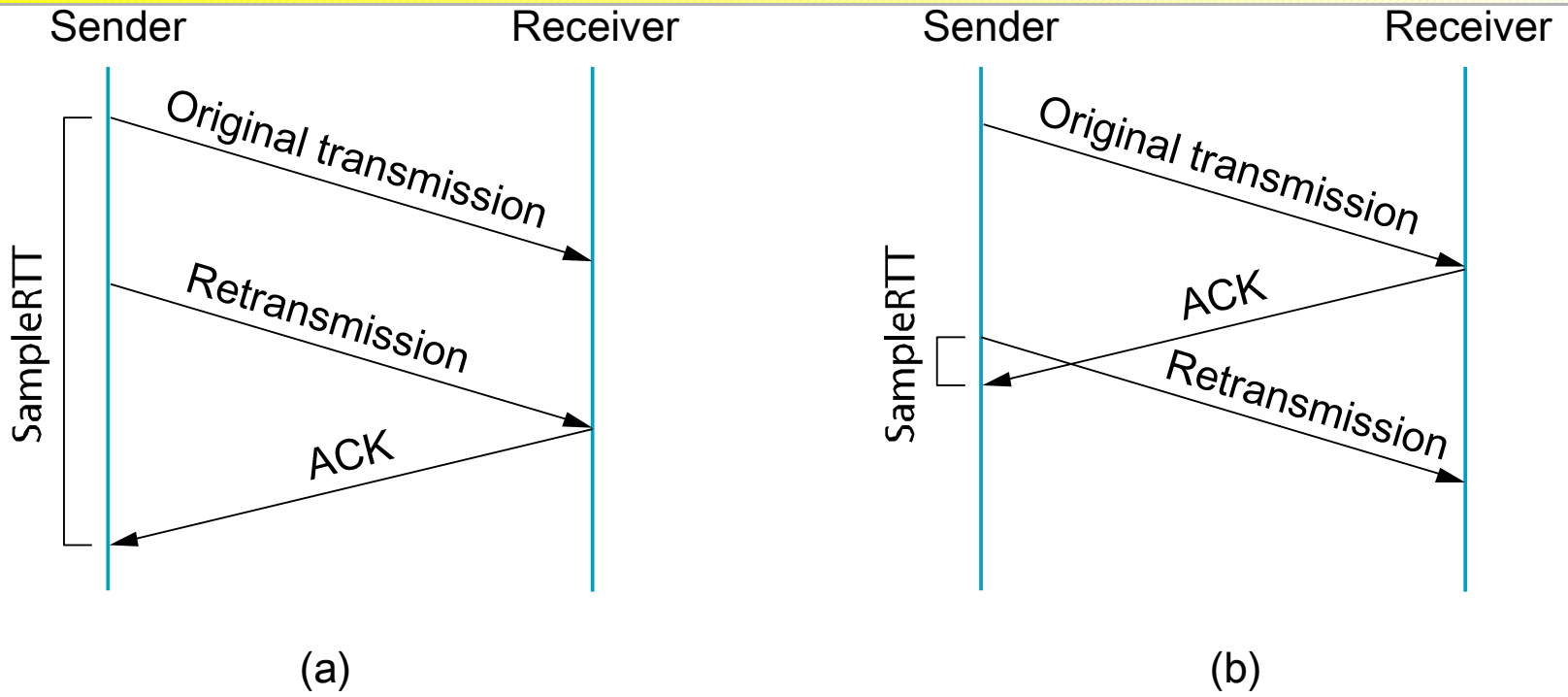
- Mede **`SampleRTT`** para cada par segmento / ACK
- Computa média poderada (exponencial) do RTT

  `EstRTT` = α x **`EstRTT`** + β x **`SampleRTT`**

  α + β = 1; α ~ 0.8 ; β ~ 0.2
- Temporização baseada no **`EstRTT`**

  `TimeOut` = 2 x **`EstRTT`**

Algoritmo de Karn/Partridge



- Não considera medição do RTT quando há retransmissão
- Dobra o valor da temporização a cada retransmissão

Algoritmo de Jacobson/Karels

- Nova forma de calcular o RTT

  Diff = **SampleRTT** - **EstRTT**

  EstRTT = EstRTT + (δ x **Diff)** | $0 < \delta < 1$

- Considera a variância ao definir a nova temporização

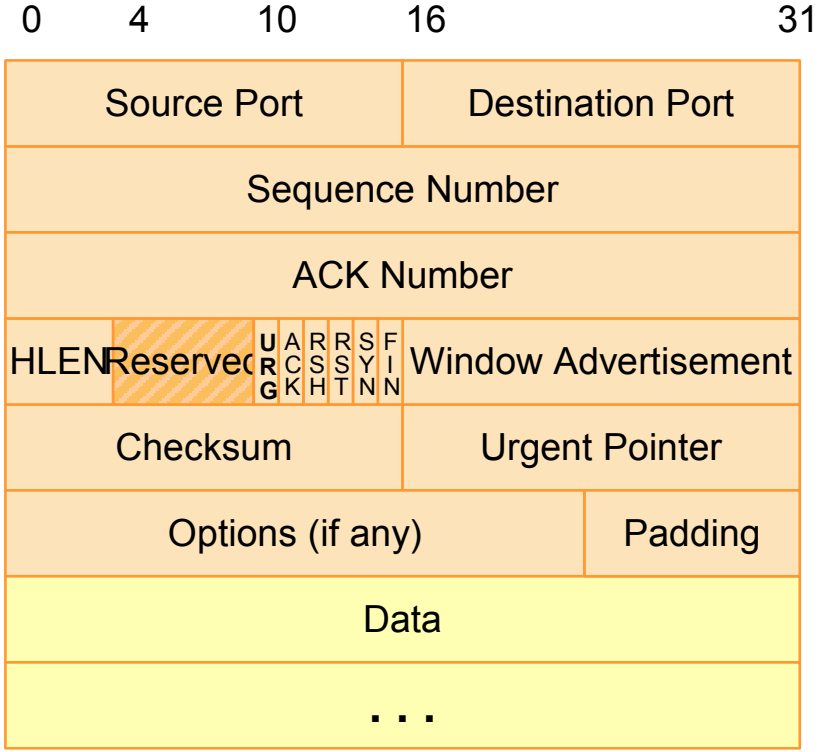
  Dev = Dev + δ(|Diff| - Dev)

  TimeOut = μ x **EstRTT** + ϕ x **Dev** | $\mu = 1$ e $\phi = 4$

- Observações
  - algoritmo só é tão bom quanto o período do clock (500 ms)
  - medição precisa da temporização é importante para controle de congestionamento (mais tarde)

Segmento TCP

- ***Options***: extensões do protocolo
  - Tamanho de Segmento Máximo (MSS)
    - Pacote de pesquisa pelo MSS
    - Visa evitar fragmentação IP
  - Fator de escala do tamanho da janela
    - Permite janelas maiores que 32 KB
  - *Timestamp*
    - Permite reconhecer pacotes que tenham se extraviado pela rede durante conexões muito rápidas
- Encontradas em quase todas as implementações modernas
  - Essenciais para se garantir desempenho nas redes de alta capacidade (LFNs)

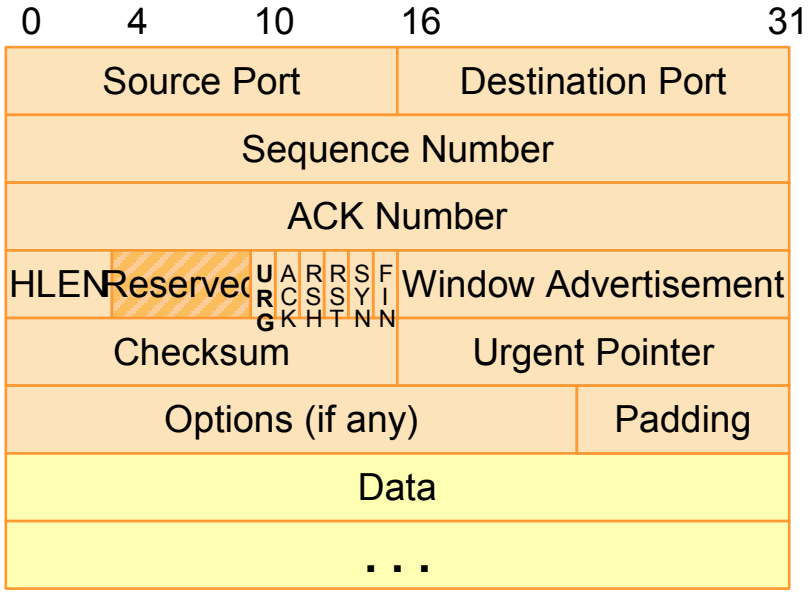


UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Segmento TCP: opções

- Proteção contra *wrap-around* de no. de seqüência
  - Campo seq. number: 32 bits

Banda	Tempo até consumir seq.
T1 (1,5 Mbps)	6,4 horas
Ethernet (10 Mbps)	57 minutos
T3 (45 Mbps)	13 minutos
FDDI (100 Mbps)	6 minutos
STS-3 (155 Mbps)	4 minutos
STS-12 (622 Mbps)	55 segundos
STS-24 (1,2 Gbps)	28 segundos



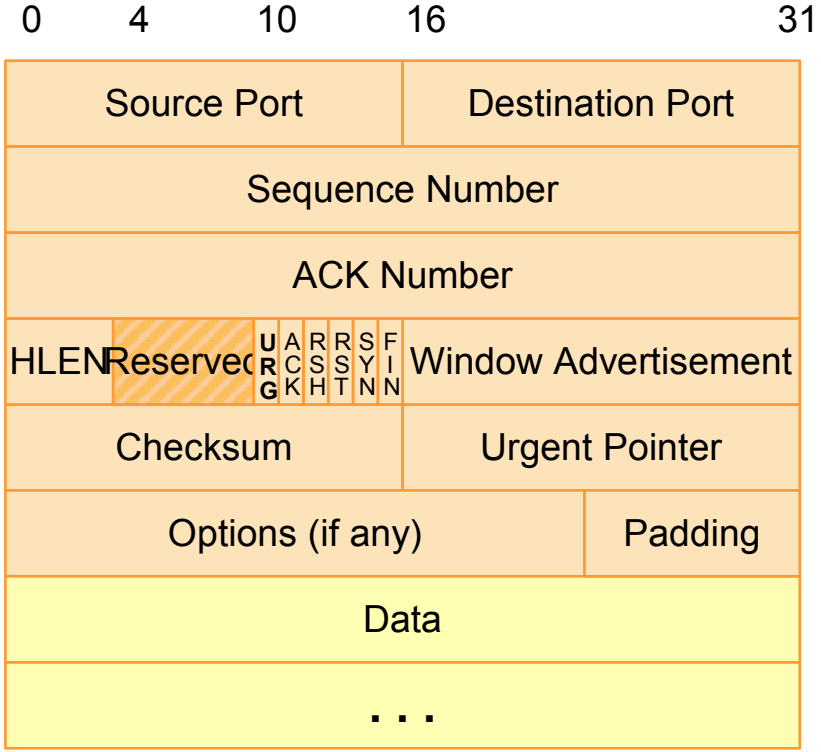
- Opção: Timestamp

UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Segmento TCP: opções

- Garantia de preenchimento do canal de transmissão
  - Campo *window advertisement*: 16 bits, RTT = 0,1s

Banda	Produto Delay x BW
T1 (1,5 Mbps)	18 KB
Ethernet (10 Mbps)	122 KB
T3 (45 Mbps)	549 KB
FDDI (100 Mbps)	1,2 MB
STS-3 (155 Mbps)	1,8 MB
STS-12 (622 Mbps)	7,4 MB
STS-24 (1,2 Gbps)	14,8 MB



- Opção: fator de escala da janela

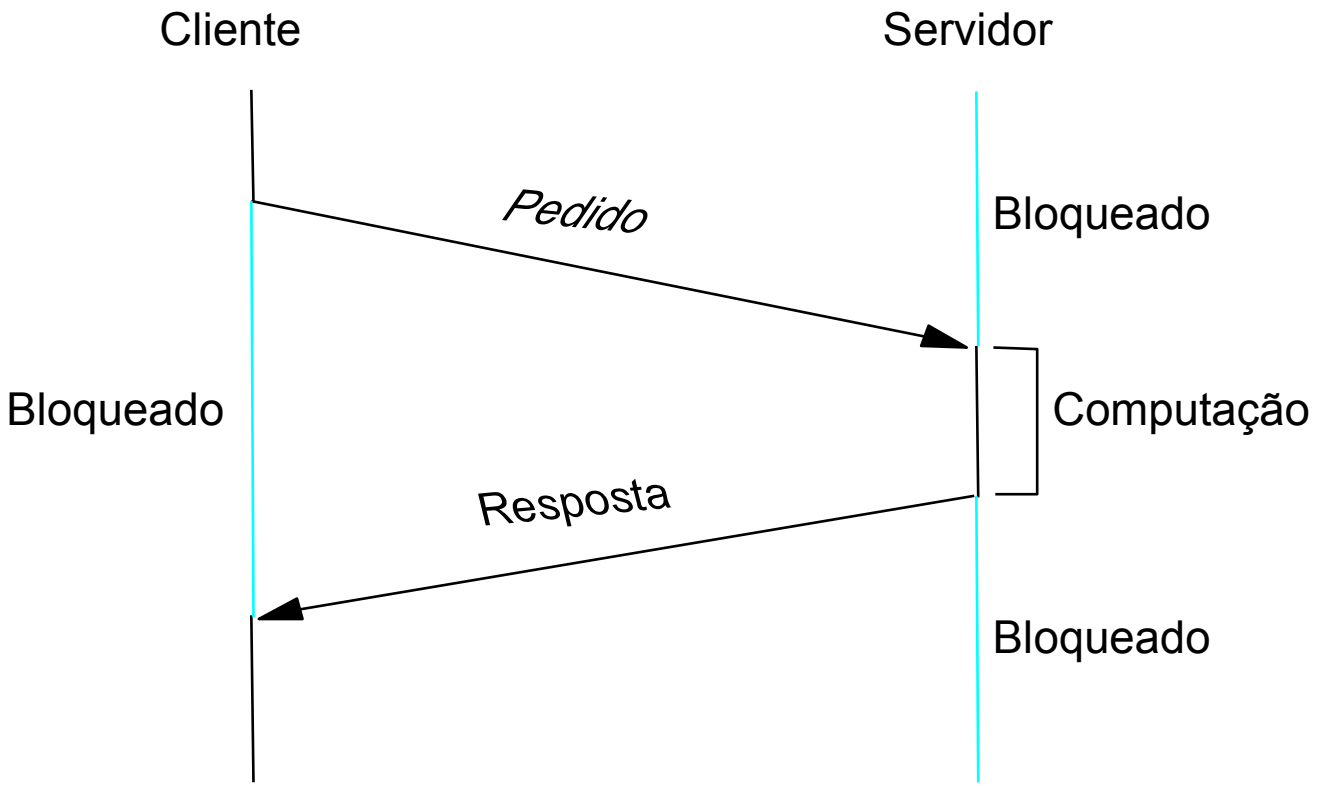
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Serviços fim-a-fim de interesse

- Um “serviço mínimo”: UDP
- Um fluxo de bytes bi-direcional: TCP
- Requisições e respostas: RPC(s)

Chamada de procedimentos remotos (RPC: *Remote Procedure Call*)

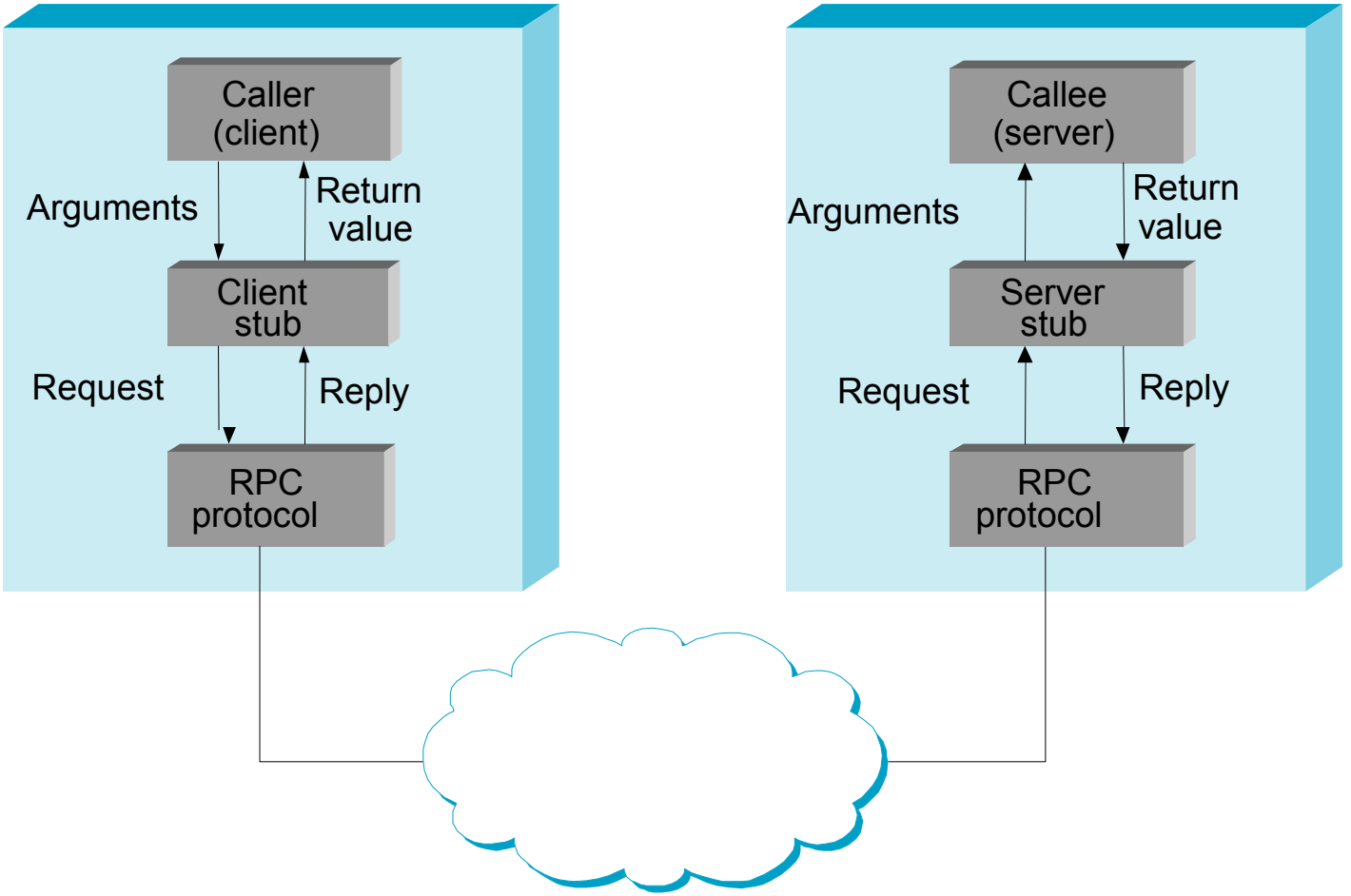
- Evolução:



UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

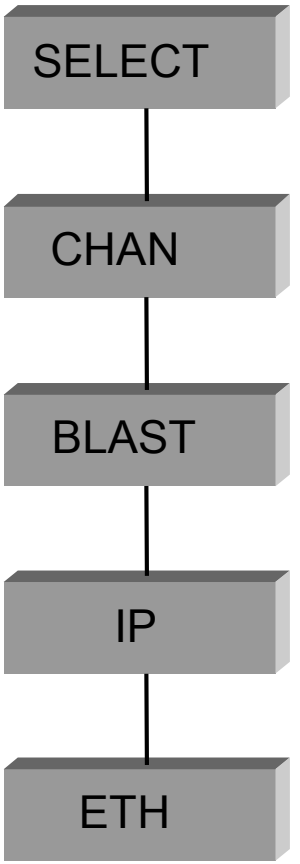
Chamada de procedimentos remotos (RPC: *Remote Procedure Call*)

- *Stubs*: adequação de dados, associação de métodos do programa a RPCs



Pilha de protocolos RPC

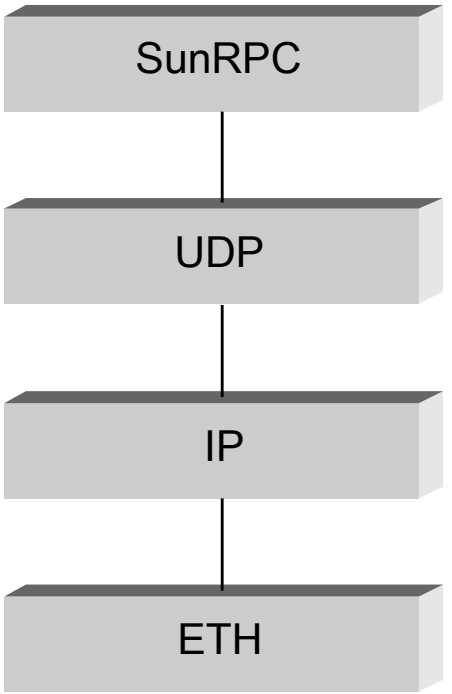
- BLAST: fragmenta e remonta mensagens
  - Considera que operação usual é em LAN
  - Tenta dentro de certos limites recuperar perdas
- CHAN: sincroniza pedidos e respostas
  - Garante entrega das mensagens
  - Sincroniza cliente e servidor
  - Implementa semântica da chamada (*at-most-once*)
- SELECT: associa processos a pedidos
  - Cada pedido recebido é associado ao procedimento correto para sua execução
  - Equivalente síncrono de UDP (demultiplexação)
  - Define um espaço de endereçamento para identificação dos procedimentos



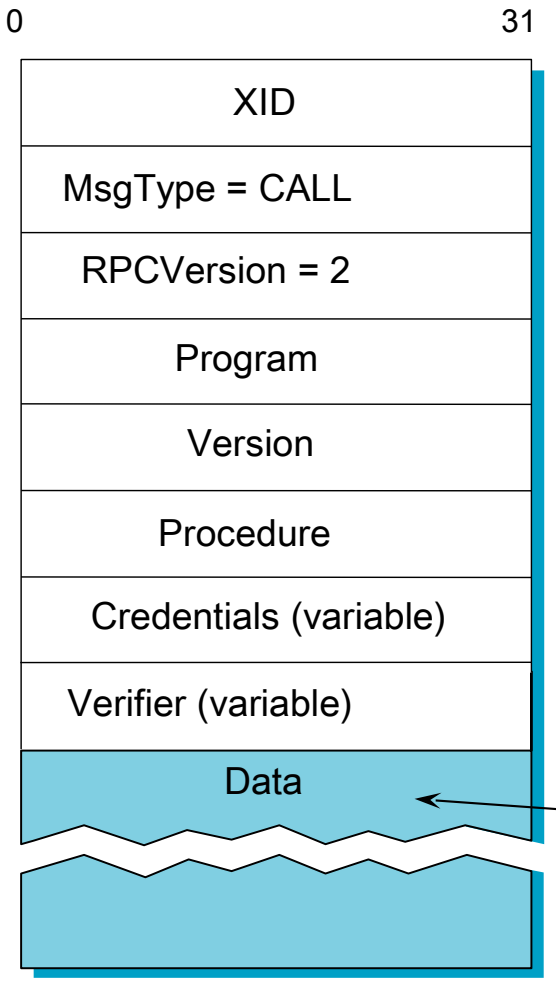
UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

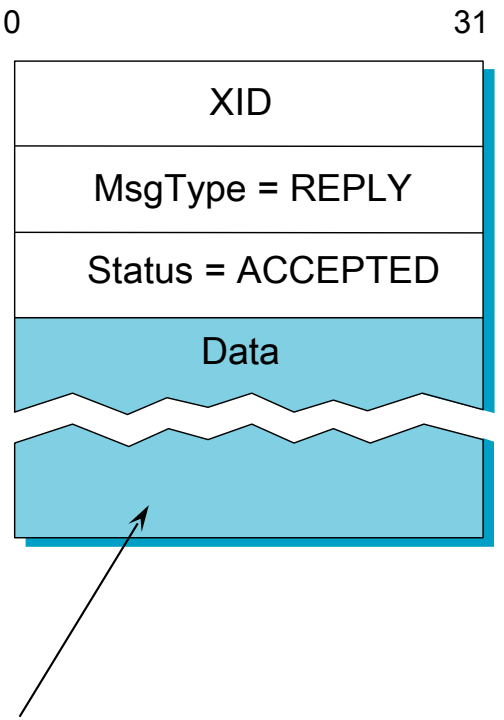
SunRPC

- IP oferece o equivalente a BLAST
  - exceto retransmissão seletiva
- SunRPC implementa CHAN
  - Diferentes semânticas possíveis
- UDP+SunRPC fazem SELECT
  - UDP identifica o programa (porto)
  - SunRPC direciona para o procedimento dentro do programa



SunRPC





Representação independente de máquina garantida por linguagem de representação de tipos: XDR (*eXtended Data Representation*)

UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Remote Method Invocation (RMI)

- Implementação dos conceitos de RPC em Java (OO)
- O que se exporta é a interface de um objeto remoto
- *Stubs* se tornam objetos com a mesma interface, mas que fazem a comunicação de argumentos e valores de retorno

Web Services/SOAP

- *Web Services* é o termo usado para se referir a modelos de serviço (RPC) implementados sobre protocolo da web, HTTP
- SOAP (*Service Oriented Access Protocol*) é um dos protocolos mais conhecidos propostos para esse fim
- Outro protocolo é REST (), que cria uma interface mais próxima de HTTP que de RPC tradicional
- Interfaces para especificação e implementação de clientes em servidores de *Web Services* existem em diferentes ambientes

Serviços fim-a-fim de interesse

- Um “serviço mínimo”: UDP
- Um fluxo de bytes bi-direcional: TCP
- Requisições e respostas: RPC(s)
- Outros?
  - Bytes x mensagens? Garantia x *best-effort*?
  - *Handshake* inicial/final x estabelecimento/término implícito?
  - Janela deslizante x taxa fixa?
  - ...

Estamos saltando um capítulo inteiro, agora!

- Controle de congestionamento e recursos (cap. 6)
  - Como garantir que conexões diferentes compartilhem a capacidade dos links da rede de forma justa?
  - Conexões podem tentar ajustar sua taxa de transmissão
  - Roteadores podem tentar atuar pela escolha dos pacotes a descartar
  - A arquitetura da rede pode ser alterada para incluir garantias
- Material tratado na disciplina de Tópicos Avançados em Redes de Computadores

Redes de Computadores

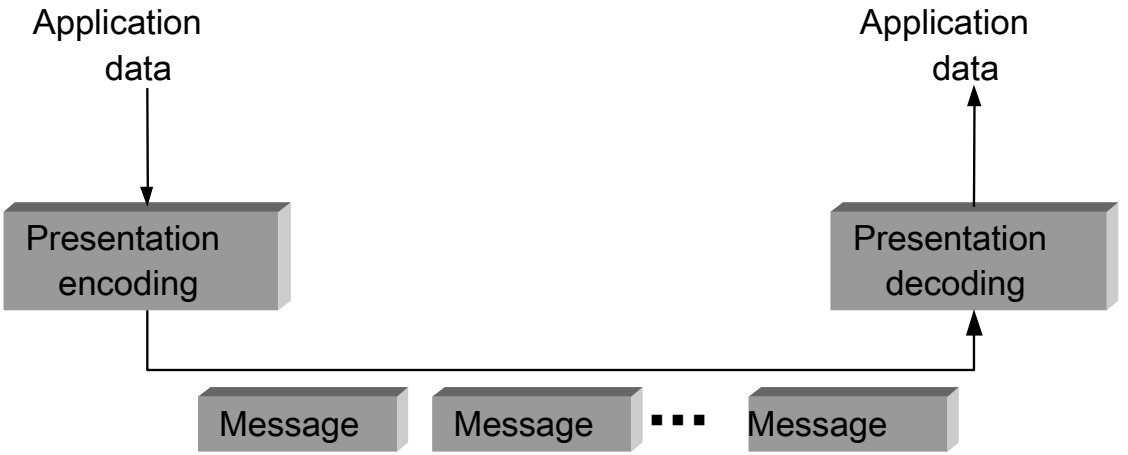
Dados de fim-a-fim

Tipos de formatação de dados

UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Formatação de apresentação

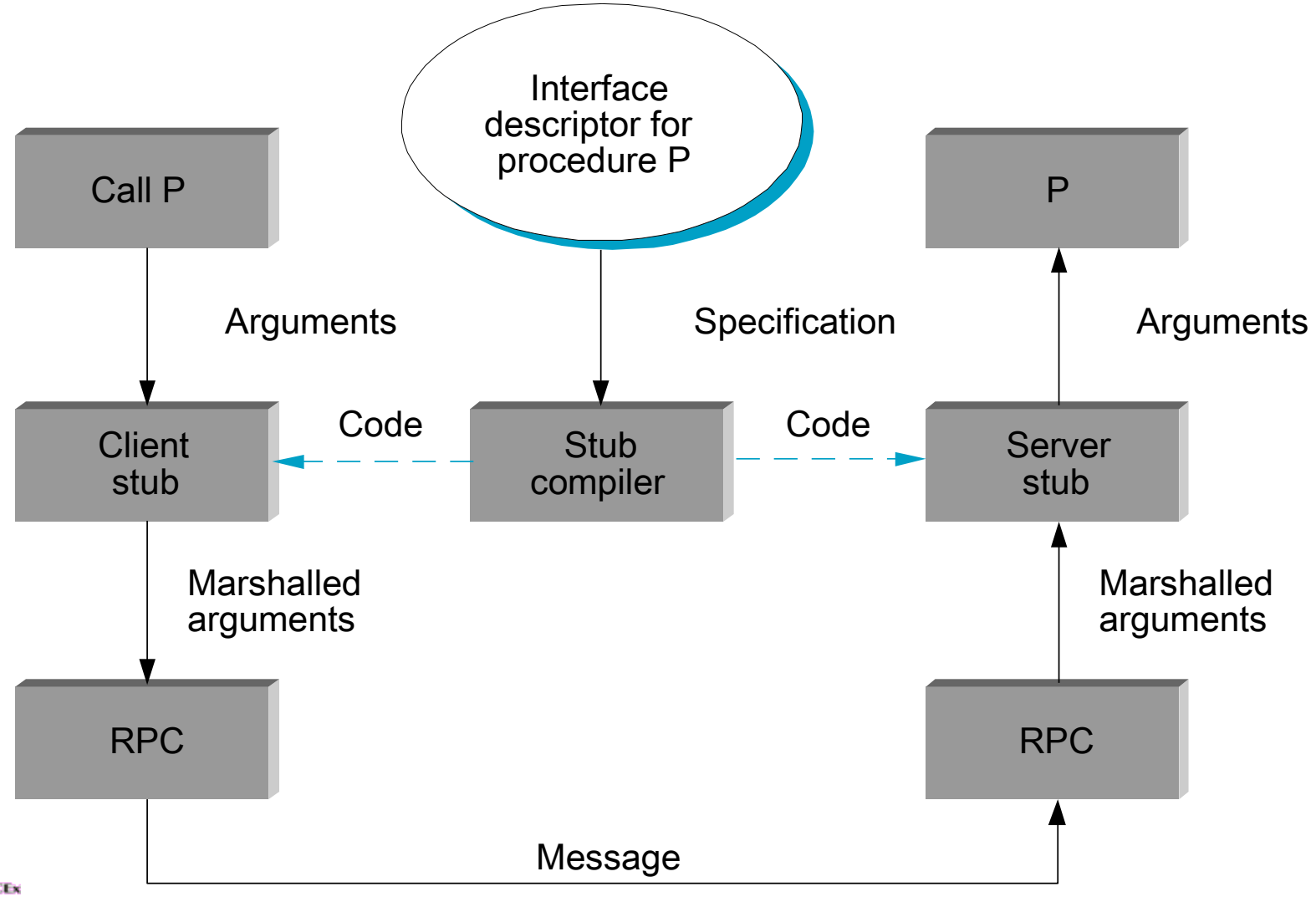
- Codificação (*marshalling*) de dados em mensagens
- Decodificação (*unmarshalling*) das mensagens em dados da aplicação



- Tipos de dados:
  - inteiros
  - ponto flutuante
  - “strings”
  - arranjos
  - registros

- Tipos de dados não considerados
  - imagens
  - vídeo
  - documentos multimídia

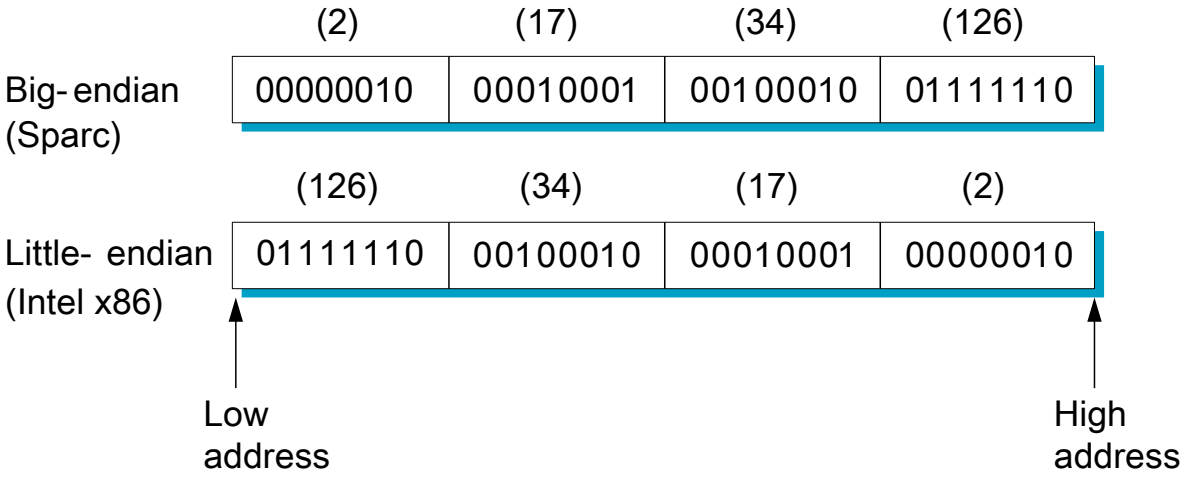
Formatação de apresentação



Dificuldades

- Representação de tipos de dados básicos
  - ponto flutuante: IEEE 754 x representações não padronizadas
  - inteiros: *big-endian* x *little-endian*

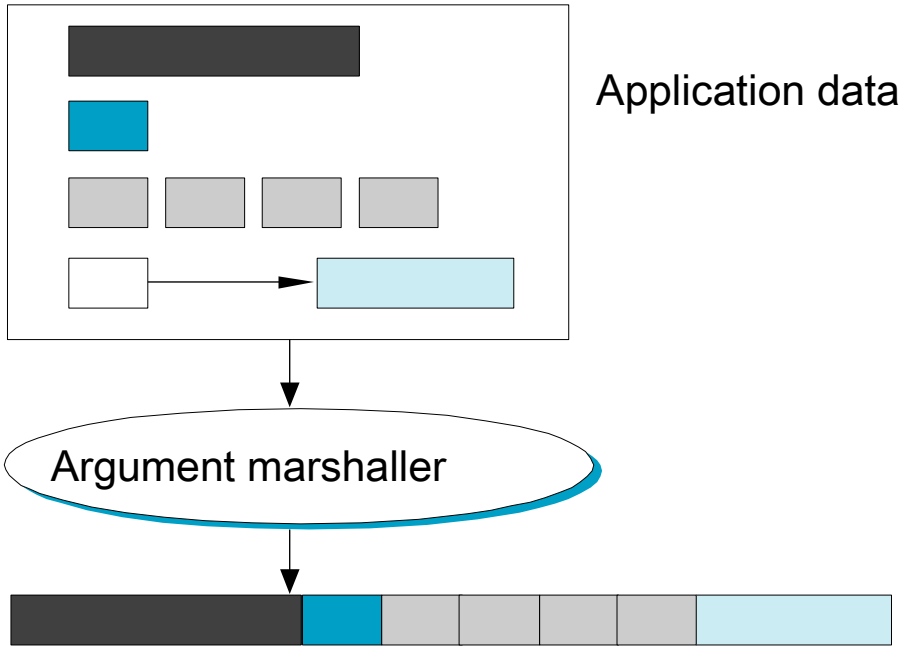
    (p.ex., 34.677.374 = 0000 0010 0001 0001 0010 0010 0111 1110)



- *Layouts* de estruturas: definidos pelo compilador

Processo de conversão

- Tipos de dados
  - básicos (int, float); devem ser convertidos
  - planos (registros, arranjos); devem ser empacotados
  - complexos (apontadores); devem ser linearizados



Processo de conversão

- Estratégias de conversão
  - forma intermediária canônica (2 x N)
  - receptor faz a transformação (N x N)

- Dados rotulados ou não

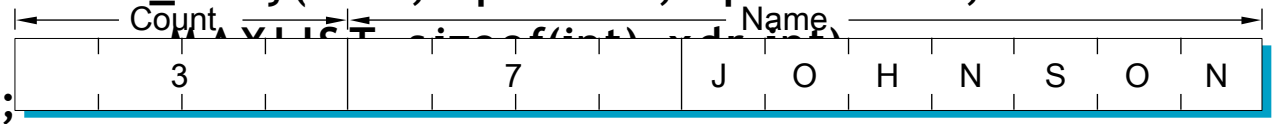
- *Stubs*
  - compilados
  - interpretados

eXternal Data Representation (XDR)

- Definido pela Sun para ser usado no SunRPC
- Sistema de tipos de C (sem apontadores para funções)
- Forma canônica intermediária
- Sem rótulos (*untagged*) – exceto comprimento de arranjos
- *Stubs* compilados

eXternal Data Representation (XDR)

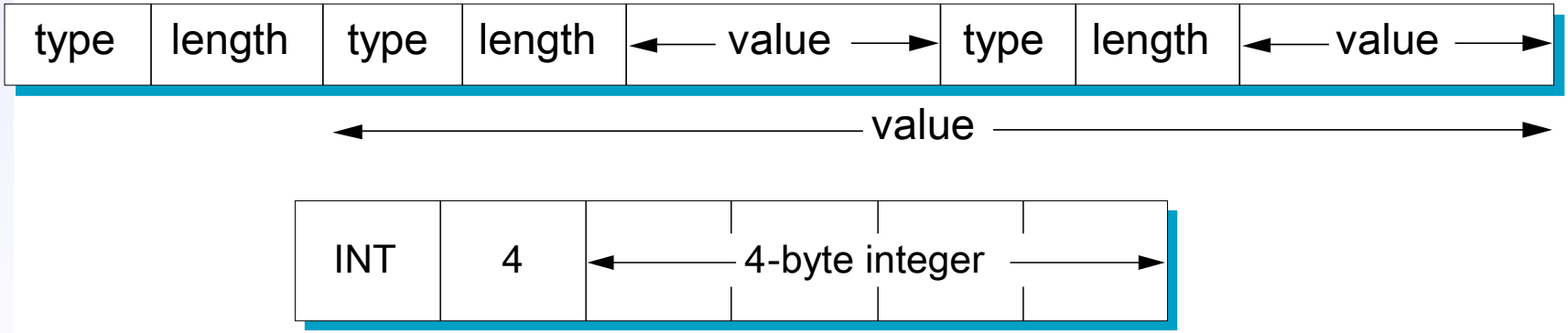
```
#define MAXNAME 256;
#define MAXLIST 100;

struct item {
  int    count;
  char   name[MAXNAME];
  int    list[MAXLIST];
};

bool_t xdr_item(XDR *xdrs, struct item *ptr)
{
  return( xdr_int(xdrs, &ptr->count)
       && xdr_string(xdrs, &ptr->name, MAXNAME)
       && xdr_array(xdrs, &ptr->list, &ptr->count,
                    MAXLIST, sizeof(int), xdr_int)
  );
}
```



Abstract Syntax Notation One (ASN-1)

- Padrão ISO
- Essencialmente o sistema de tipos de C
- Forma canônica intermediária
- Rotulada
- Stubs compilados ou interpretados
- BER: *Basic Encoding Rules*

  `(tag, length, value)`



Network Data Representation (NDR)

- Definido pelo consórcio DCE
- Essencialmente o sistema de tipos de C
- Receptor faz a conversão
- Itens de dados individuais sem rótulos
- *Stubs* compilados da linguagem IDL
- Rótulo de arquitetura de quatro bytes

- IntegerRep
  - 0 = big-endian
  - 1 = little-endian
- CharRep
  - 0 = ASCII
  - 1 = EBCDIC
- FloatRep
  - 0 = IEEE 754
  - 1 = VAX
  - 2 = Cray
  - 3 = IBM

eXtended Markup Language (XML)

- Representação textual de registros usando rótulos (*tags*)
- Esquemas (*schemas*) definem a interpretação dos campos
- Utilizado por exemplo em SOAP (RPC p/ *Web Services*)
- Processamento normalmente mais complexo que soluções anteriores
- Representação textual ocupa mais espaço

Estamos saltando:

- Compactação de dados (seção 7.2)
  - Compactação sem perdas
  - Compactação de imagens (jpeg)
  - Compactação de vídeo (mpeg)
  - Compactação de áudio (mp3)

Material tratado em diversos contextos

  - Algoritmos
  - Tratamento/distribuição de multimídia
  - Aplicações avançadas

Serviços fim-a-fim de interesse

- Um “serviço mínimo”: UDP
- Um fluxo de bytes bi-direcional: TCP
- Requisições e respostas: RPC(s)

Estamos saltando mais um capítulo, também:

- Segurança de rede (cap. 8)
  - Algoritmos de criptografia
  - Mecanismos de segurança
  - Exemplos de sistemas (PGP, SSH, HTTPS, IPSEC)
  - *Firewalls*

Material suficiente para um curso específico (Tópicos)