Redes de Computadores
Camada de Transporte

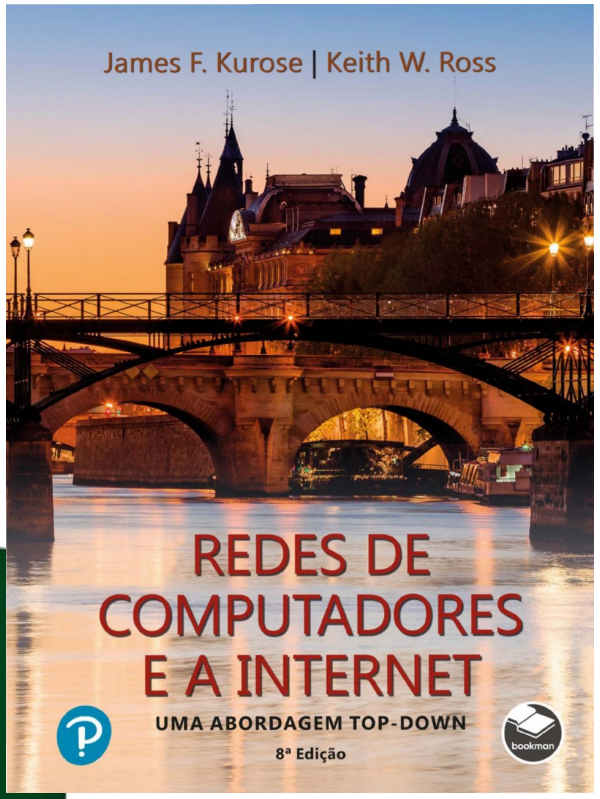
Professor: Fábio Renato de Almeida

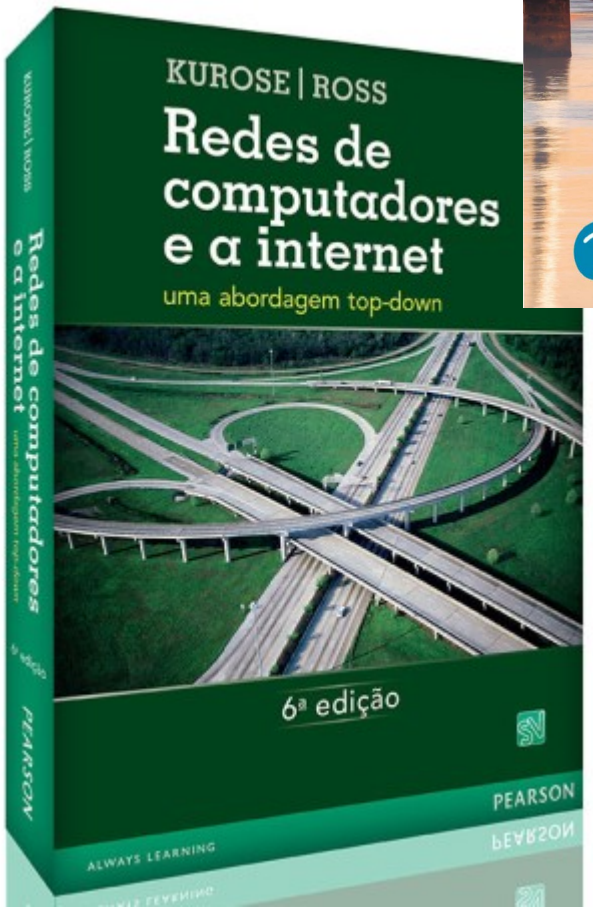
https://github.com/fabiorenatodealmeida

e-mail: fabiorenatodealmeida@hotmail.com

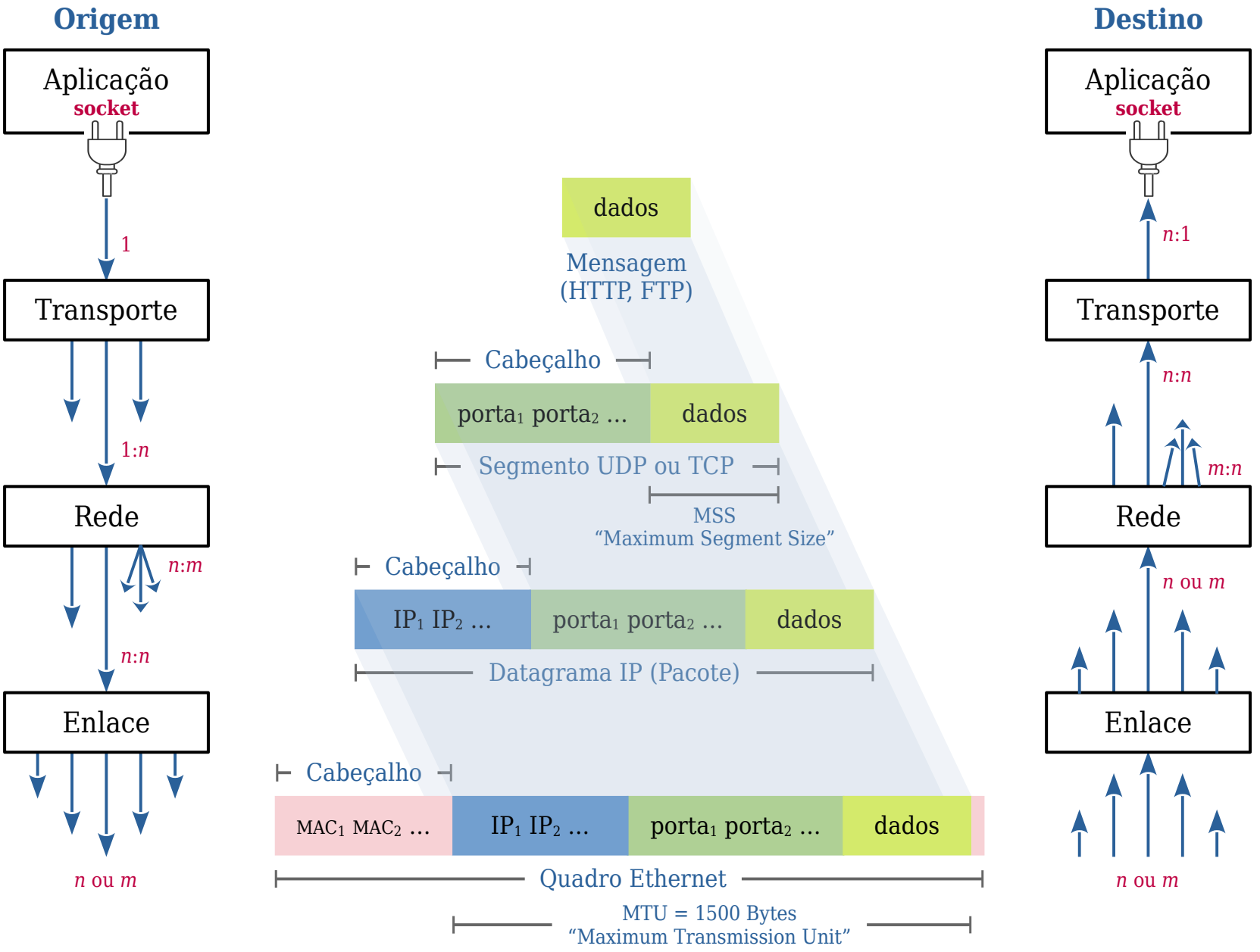
Bibliografia

https://media.pearsoncmg.com/ph/esm/ecs_kurose_compnetwork_8/cw/
https://www.youtube.com/@JimKurose





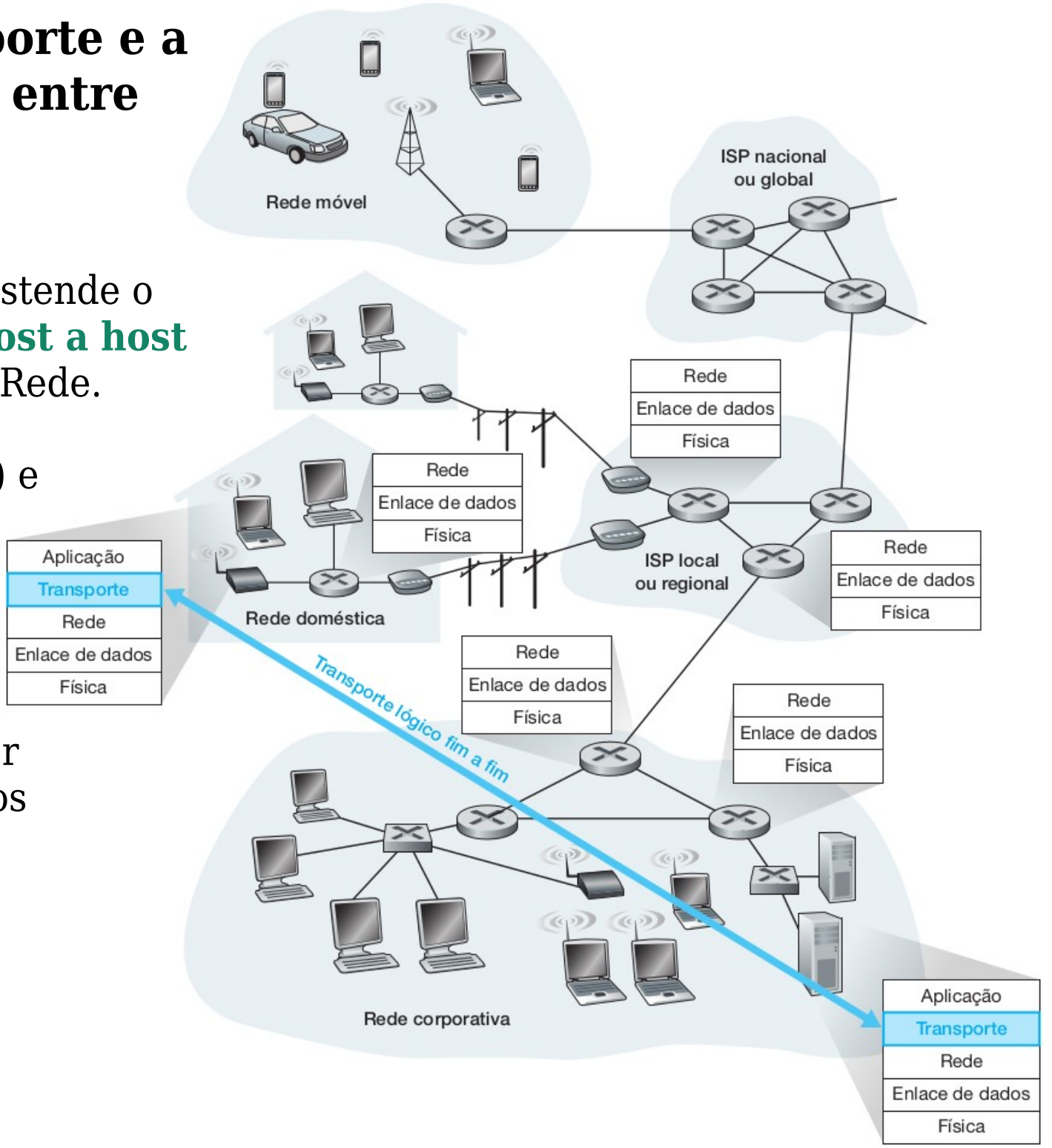
Pilha de Protocolos



A camada de Transporte e a comunicação lógica entre processos

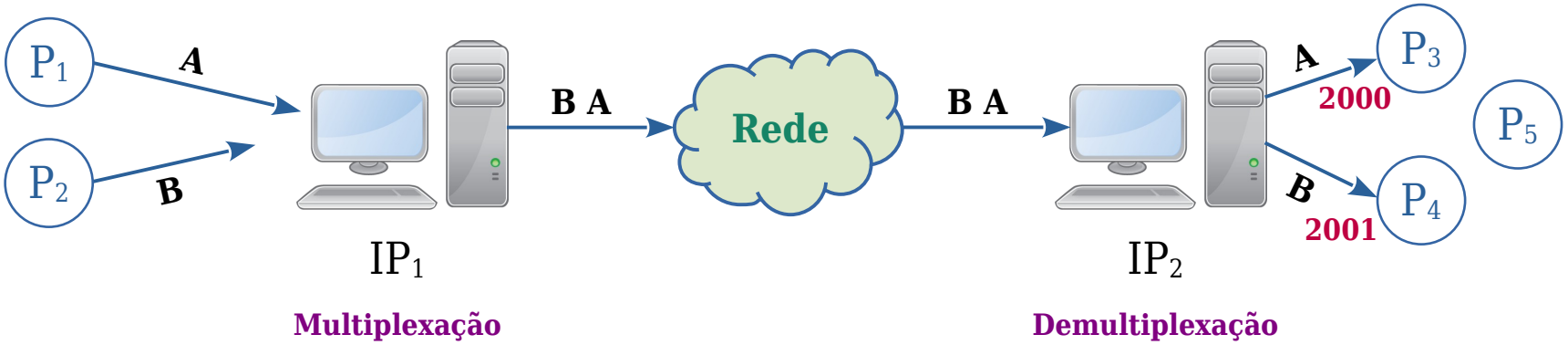
A camada de Transporte estende o serviço de comunicação **host a host** oferecido pela camada de Rede.

Ela multiplexa (na origem) e demultiplexa (no destino) segmentos de dados com base em portas que são associadas aos **sockets**, permitindo que mensagens possam ser endereçadas a seus devidos processos.

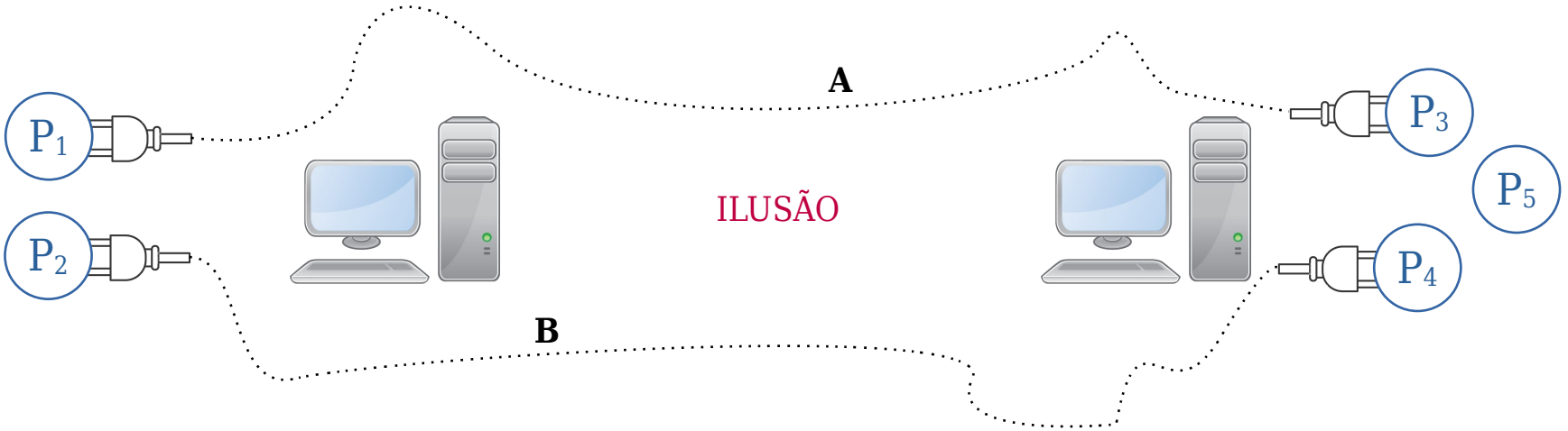


Multiplexação / Demultiplexação na camada de Transporte

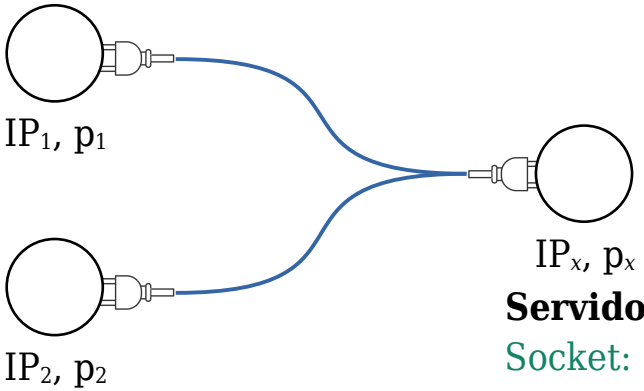
Permite que a entrega **host a host** oferecida pela camada de Rede seja estendida para uma entrega **processo a processo**.



Destino do pacote A = (IP_2, 2000)
Destino do pacote B = (IP_2, 2001)



Sockets UDP

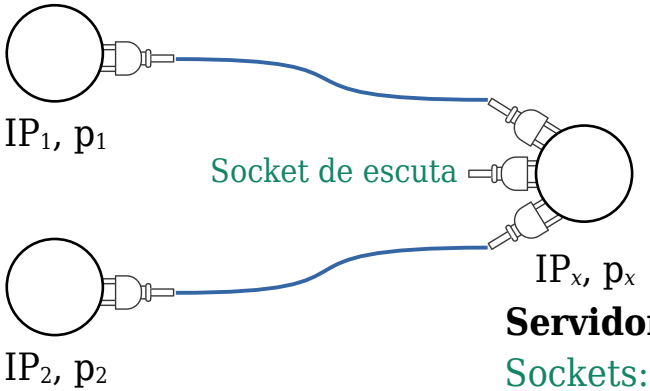


Servidor

Socket: (IP_x, p_x)

Identificação: ($IP_{destino}$, porta $_{destino}$)

Sockets TCP



Servidor

Sockets: (*, *, IP_x, p_x) Socket de escuta (ou entrada)

Recebe segmentos de estabelecimento de conexão: SYN=1

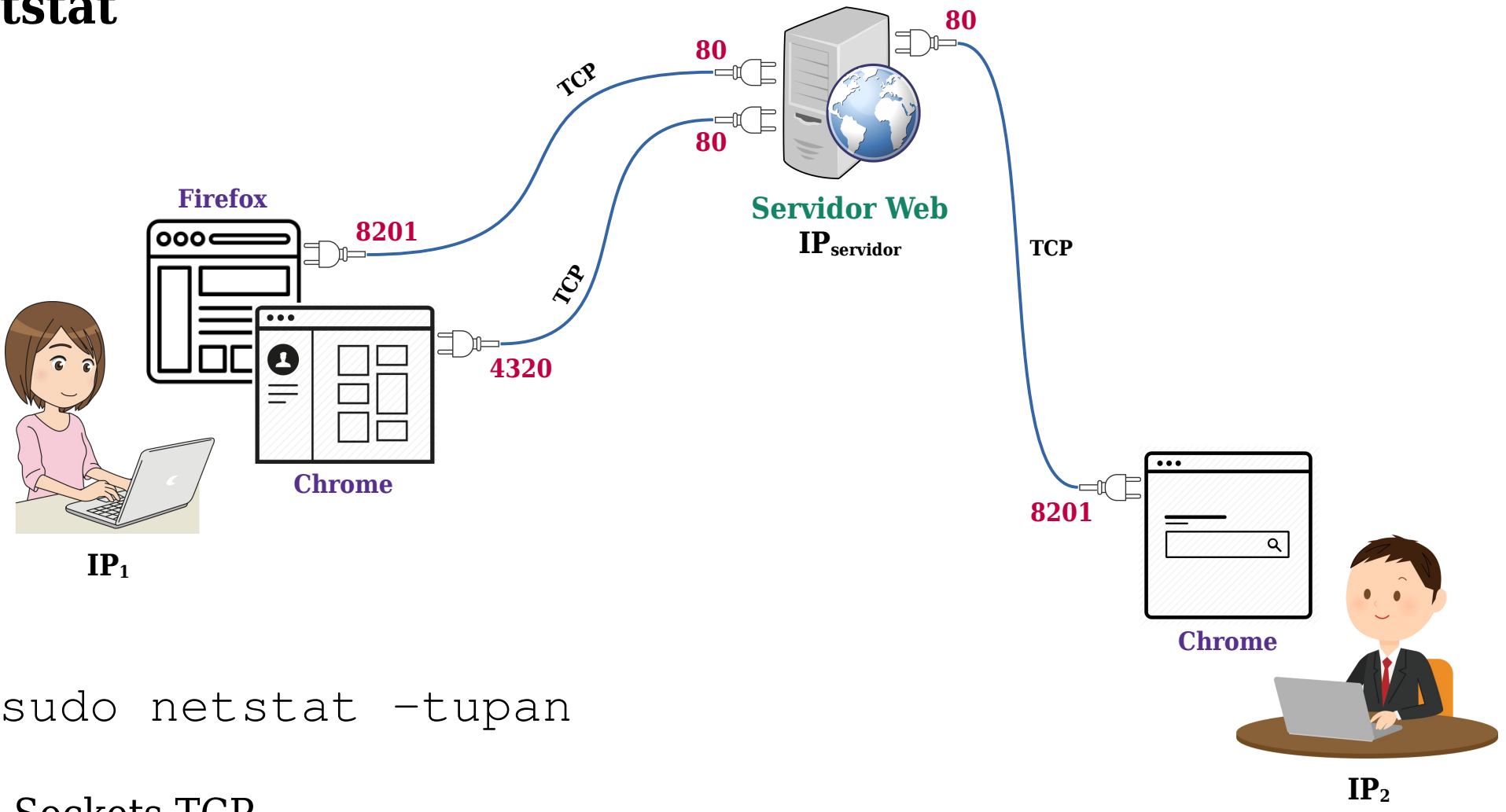
Os outros segmentos são demultiplexados para seus respectivos sockets

(IP_1, p_1, IP_x, p_x)

(IP_2, p_2, IP_x, p_x)

Identificação: (IP_{origem}, porta $_{origem}$, $IP_{destino}$, porta $_{destino}$)

netstat

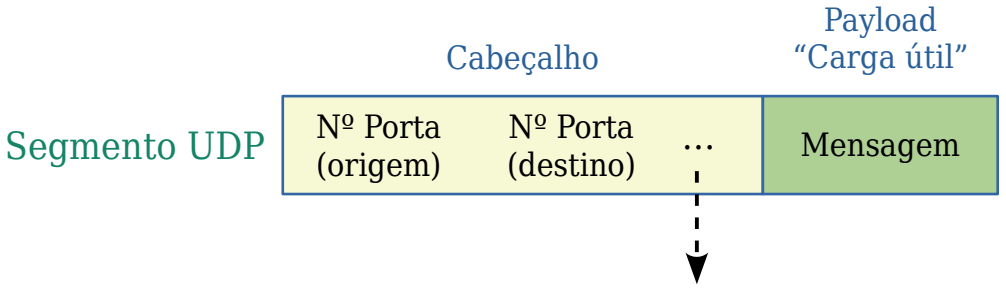



```
$ sudo netstat -tupan
```

`-t` Sockets TCP
`-u` Sockets UDP
`-p` Mostra PID e programa ao qual o socket está associado
`-a` Sockets que representam conexões ativas e portas de escuta
`-n` Exibe endereços numéricos

UDP – User Datagram Protocol

Serviço não orientado à conexão e sem garantias.



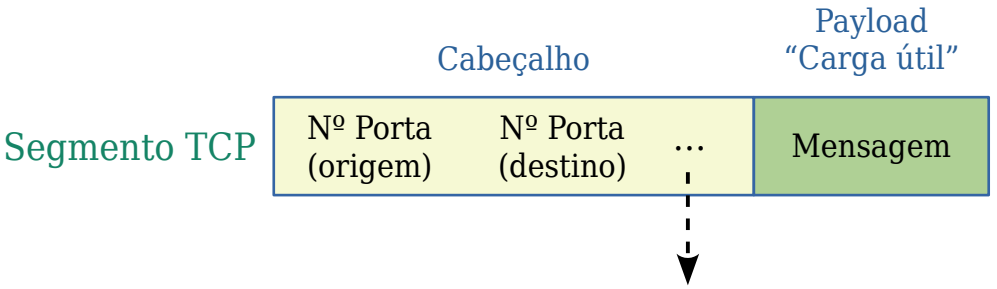
- Comprimento (16 bits): Nº de bytes no segmento UDP.
- Soma de verificação (16 bits): Utilizada na detecção de erros de transmissão (checksum). O protocolo não oferece meios para correção do erro.

Nº Porta: 16 bits → 0..65535

Portas de 0..1023 são portas reservadas e normalmente associadas a serviços conhecidos.

TCP – Transmission Control Protocol

Serviço orientado à conexão e com garantias. Requer apresentação de três vias (handshake).

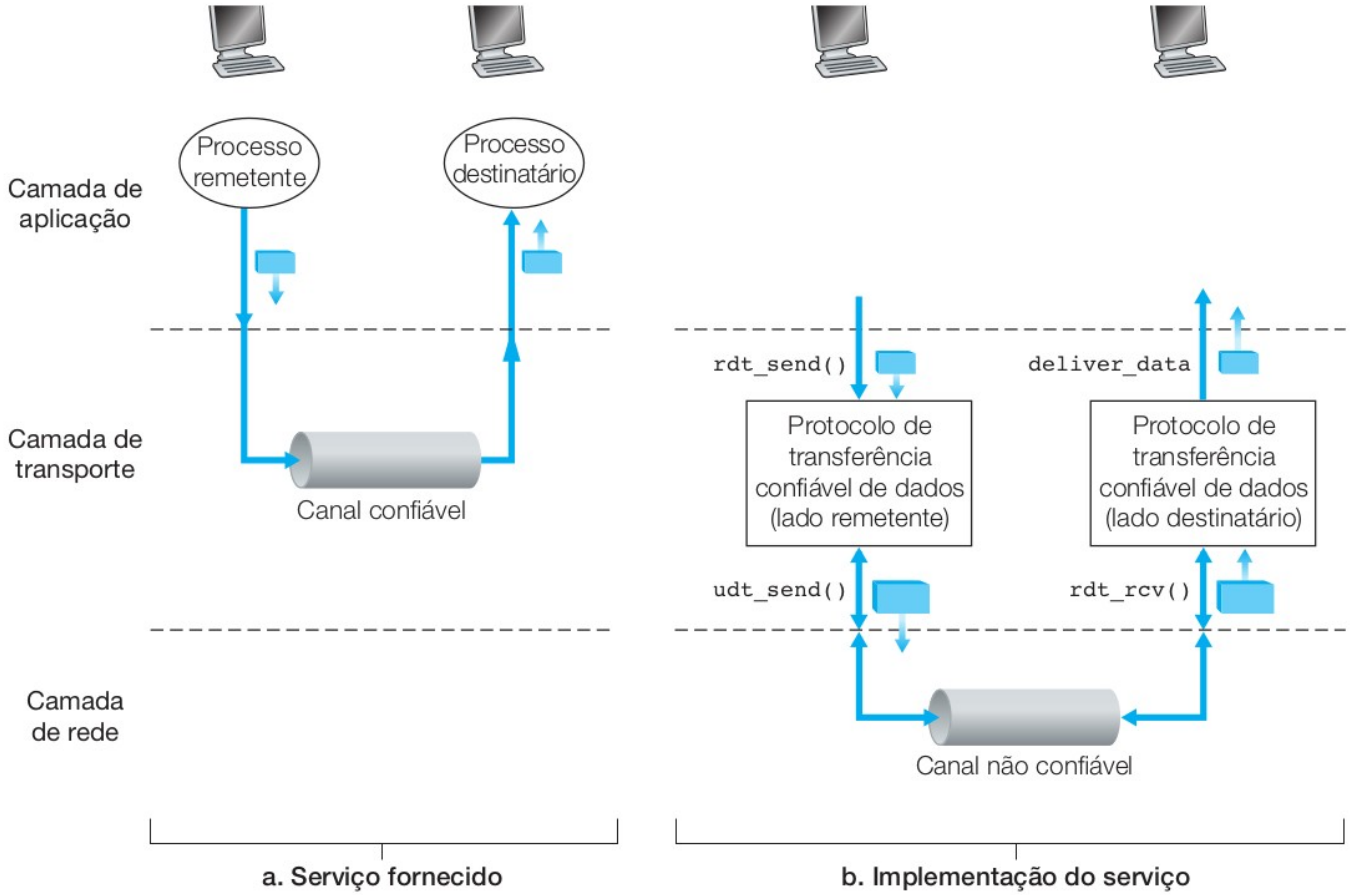


- Controle de fluxo e sequencialização: Sincroniza as taxas de emissão e recepção dos pacotes e garante a ordem de entrega dos mesmos.
- Comprimento (4 bits): Tamanho do cabeçalho TCP em palavras de 32 bits.
- Confirmação de recebimento: Envia pacotes de confirmação do tipo ACK.
- Controle de congestionamento: Ajuste de taxa de transferência com base na utilização da rede.
- Soma de verificação (16 bits): Utilizada na detecção de erros de transmissão (checksum). O protocolo não oferece meios para correção do erro.

Nº Porta: 16 bits → 0..65535

Portas de 0..1023 são portas reservadas e normalmente associadas a serviços conhecidos.

Transferência confiável de dados

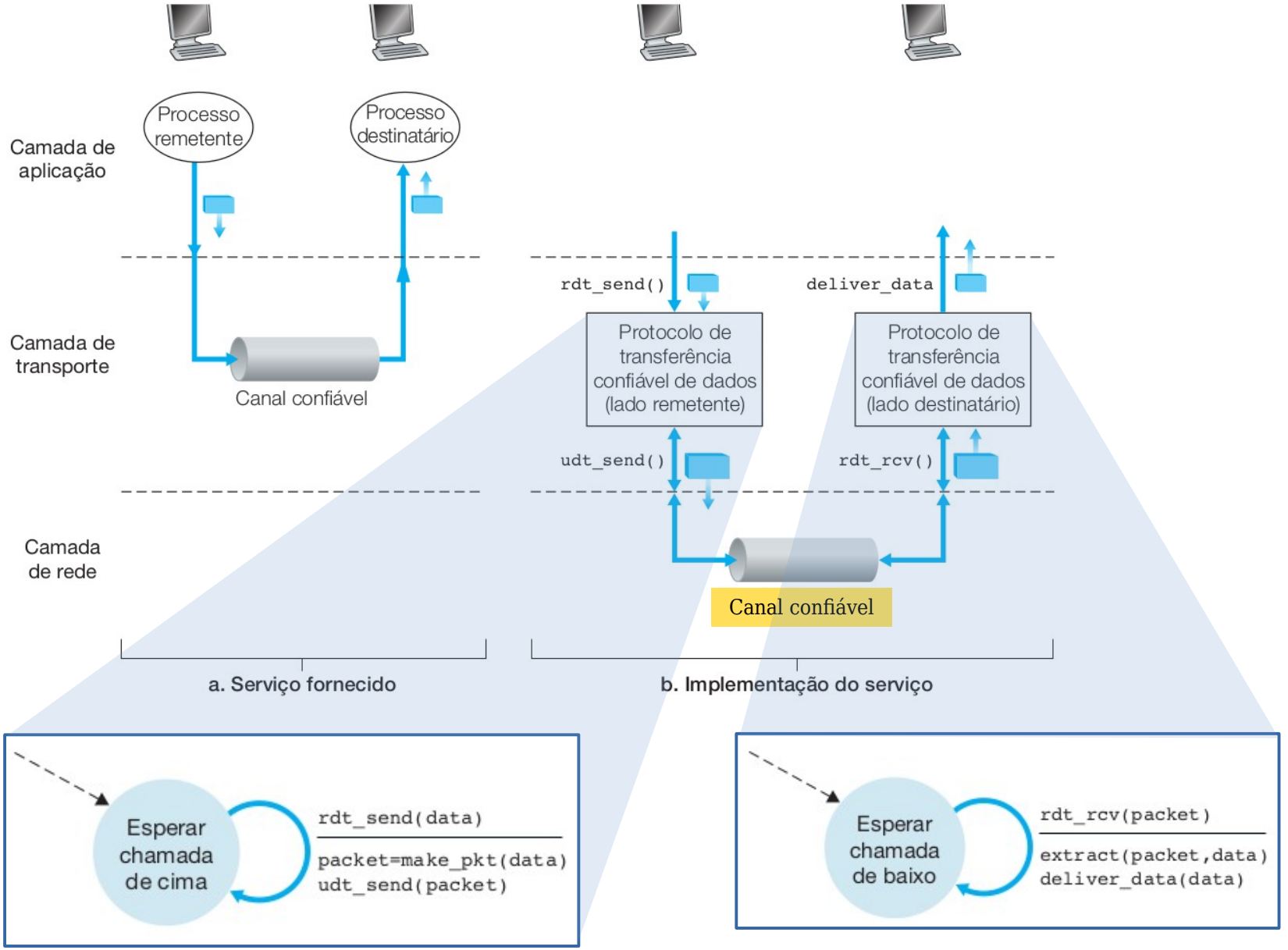


Canal confiável: Não há perda de pacotes ou corrompimento dos dados. Os pacotes são entregues sempre em ordem. É exatamente o que o **TCP** faz.

Tais garantias são implementadas sobre um canal não confiável. A questão é... Como isso é possível?

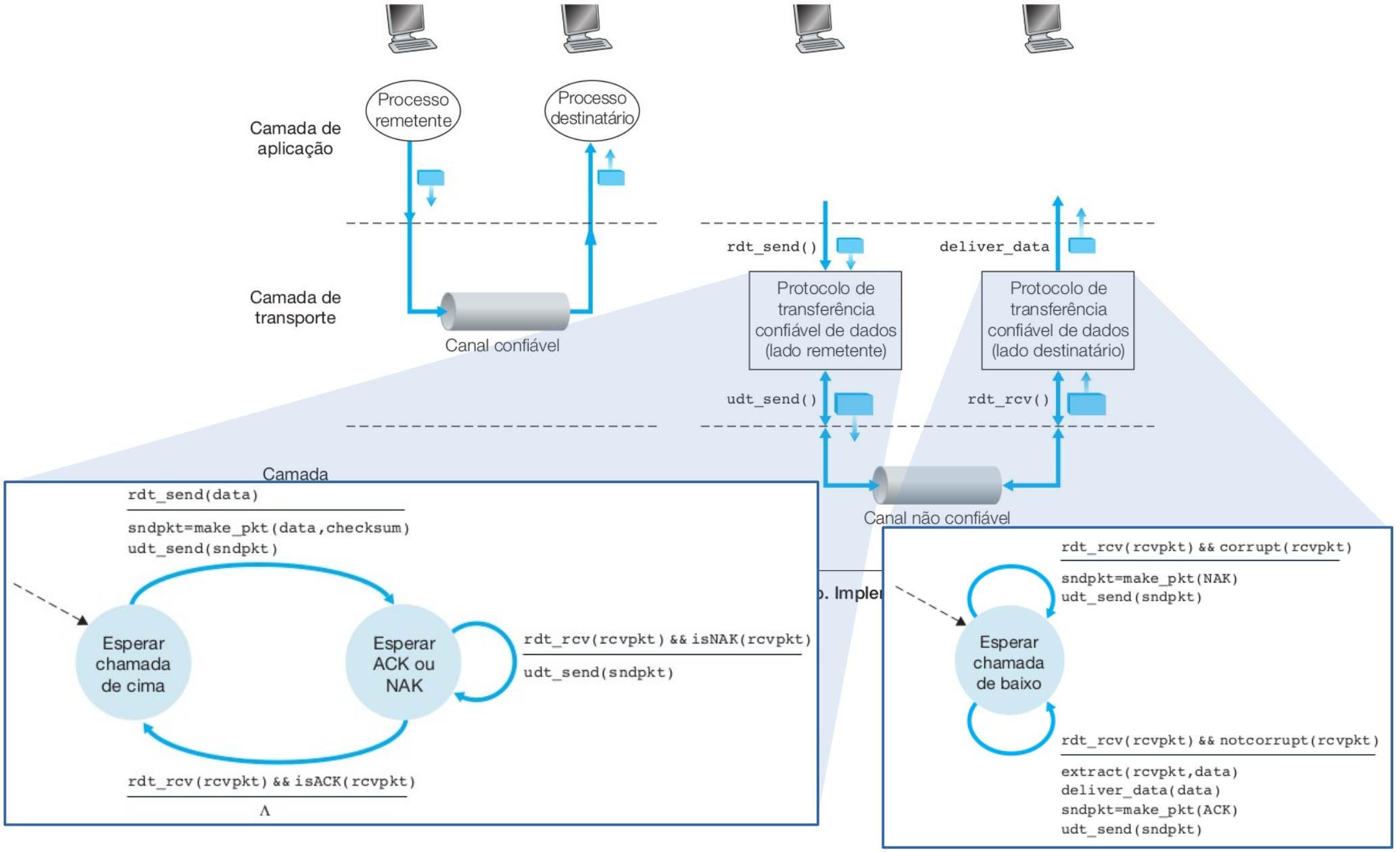
rdt: Reliable Data Transfer – **udt**: Unreliable Data Transfer

Transferência confiável de dados: sobre um canal confiável



Já que nada pode dar errado, não há necessidade do protocolo no lado destinatário fornecer qualquer informação ao remetente.

Transferência confiável de dados: sobre um canal com erros de bits

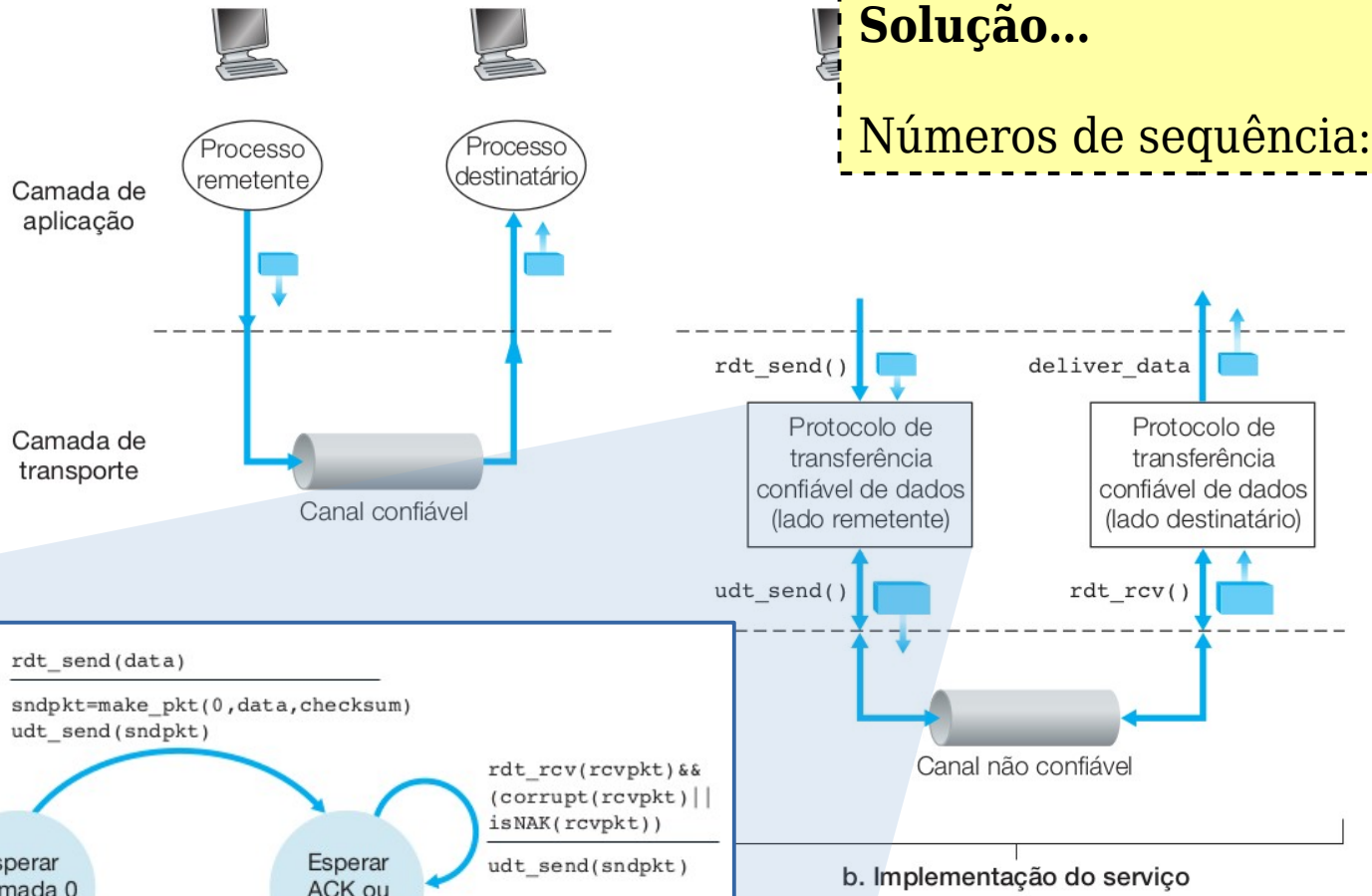


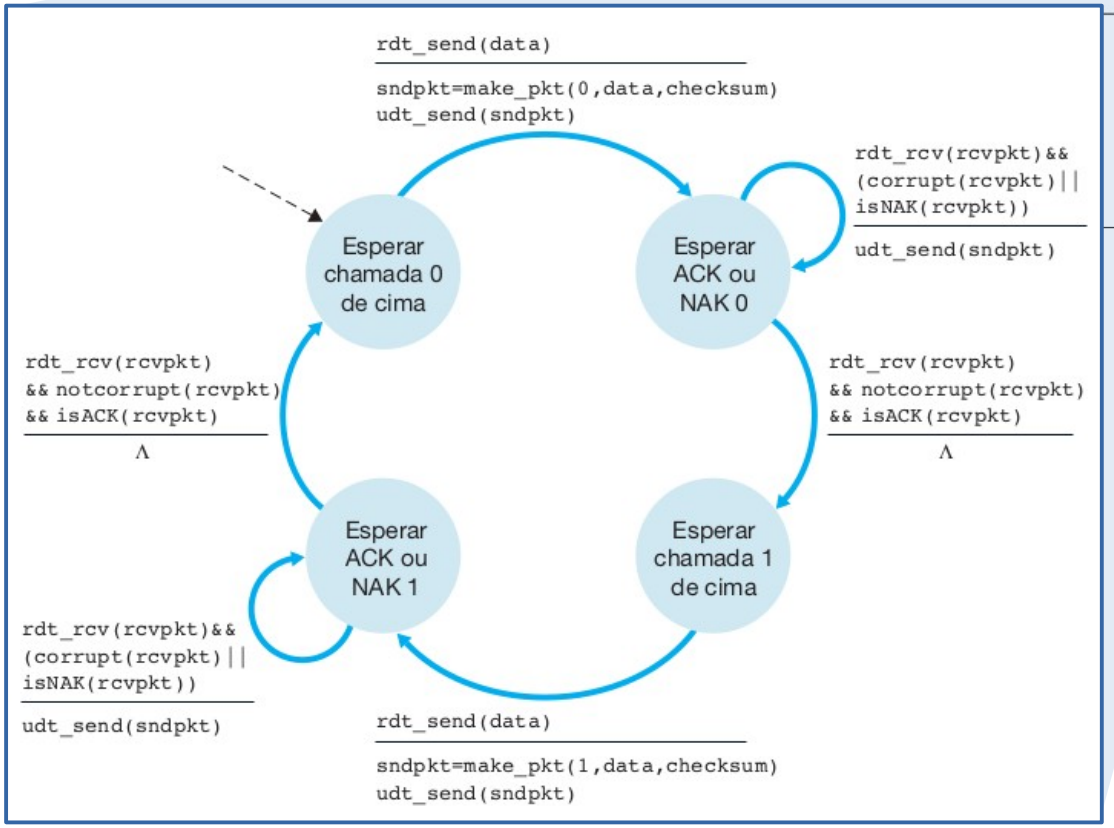
Mas e se o próprio pacote **ACK** (Acknowledgment) ou **NAK** (Negative Acknowledgment) estiver corrompido?

Transferência confiável de dados: sobre um canal com erros de bits

Solução...

Números de sequência: 0, 1, ...

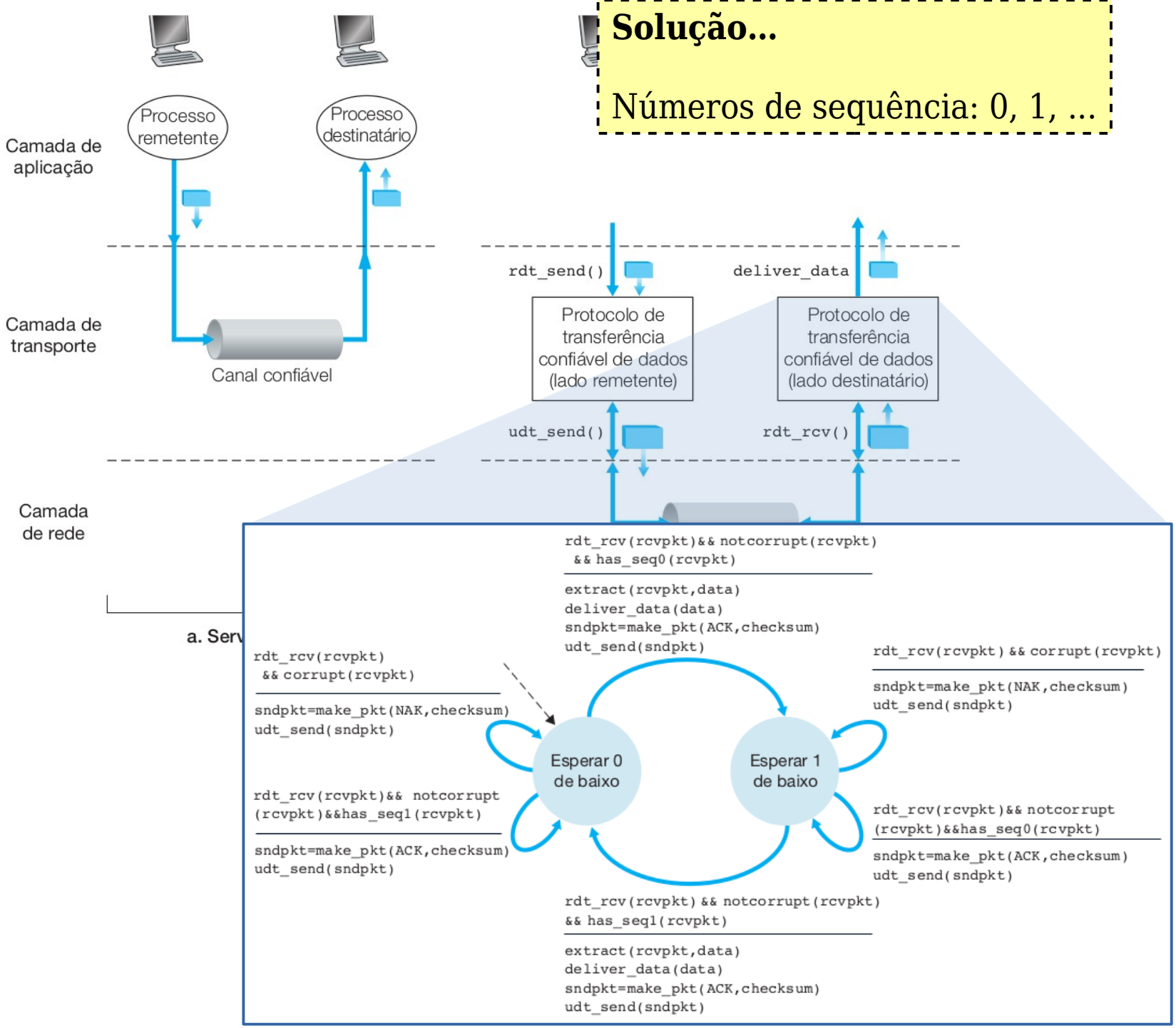




Transferência confiável de dados: sobre um canal com erros de bits

Solução...

Números de sequência: 0, 1, ...

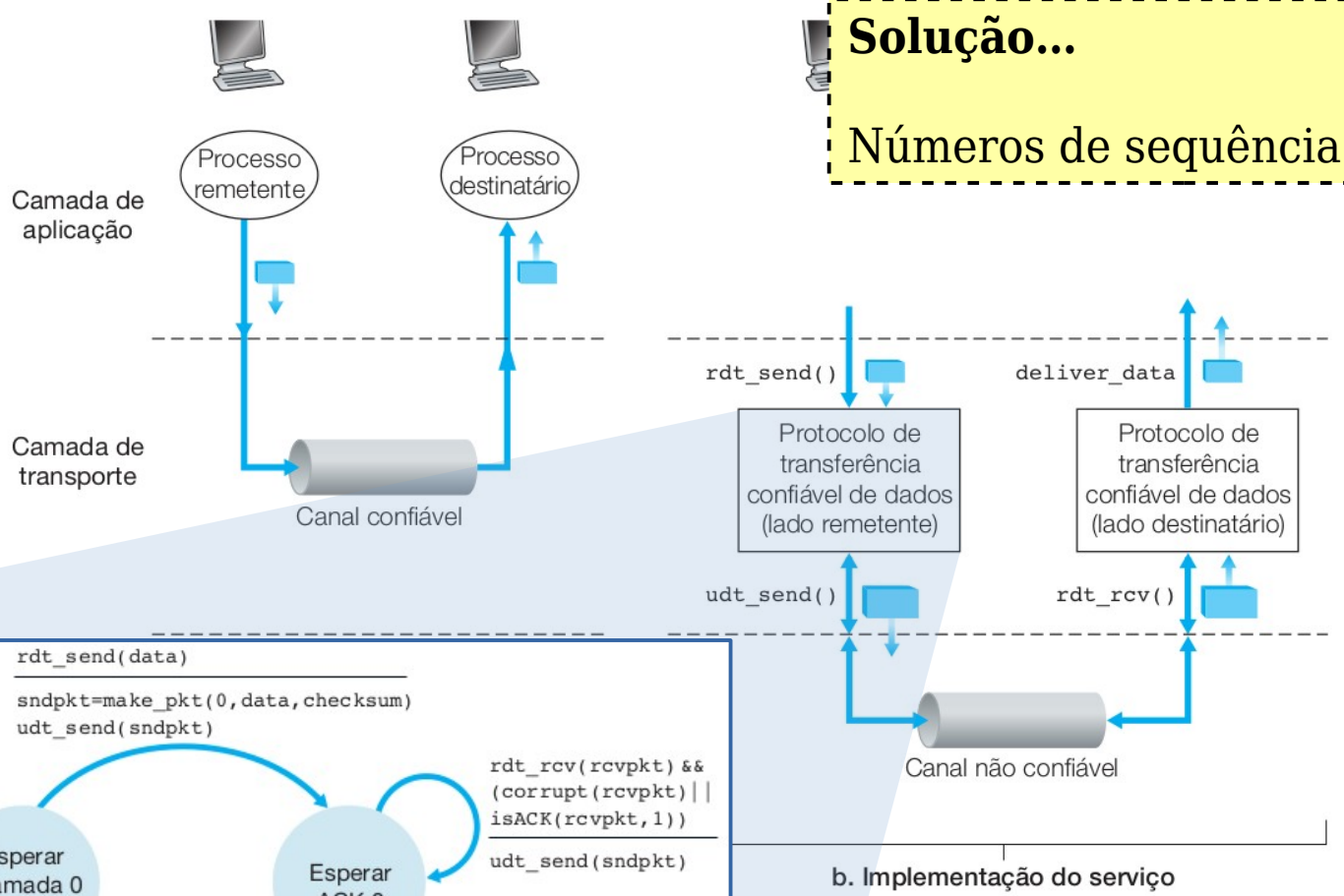


Transferência confiável de dados: sobre um canal com erros de bits

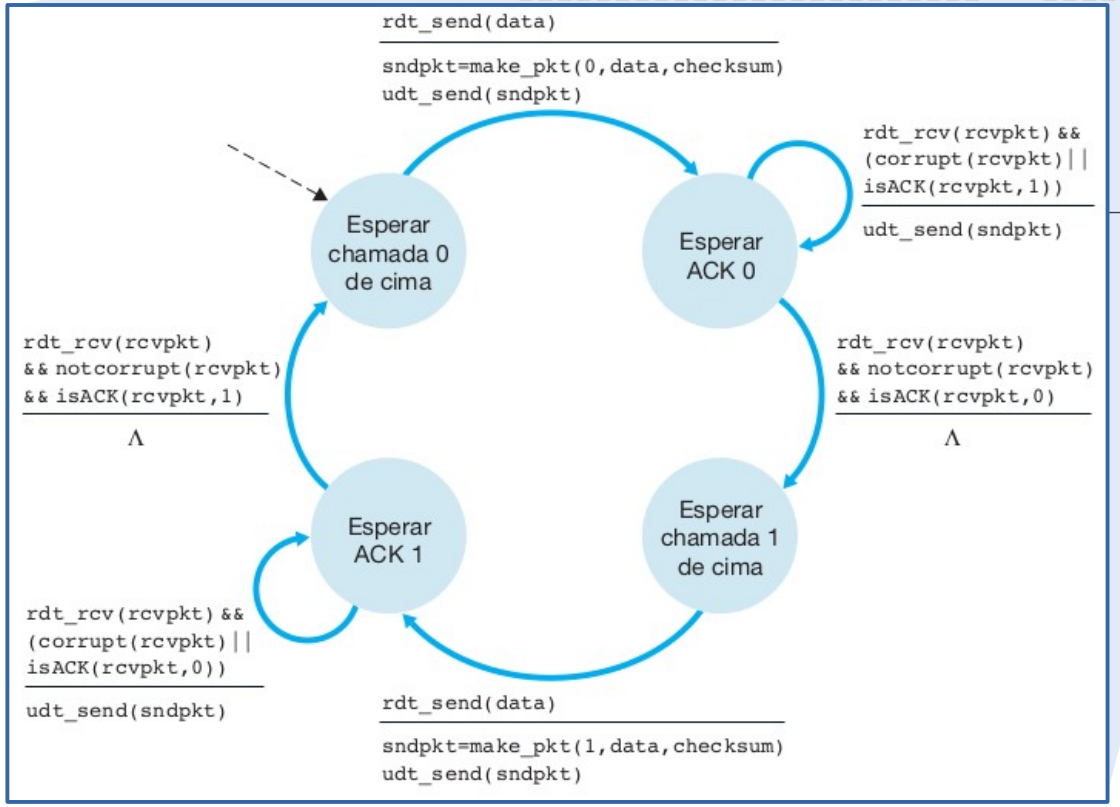
Solução...

Números de sequência: 0, 1, ...

Apenas **ACK**



b. Implementação do serviço

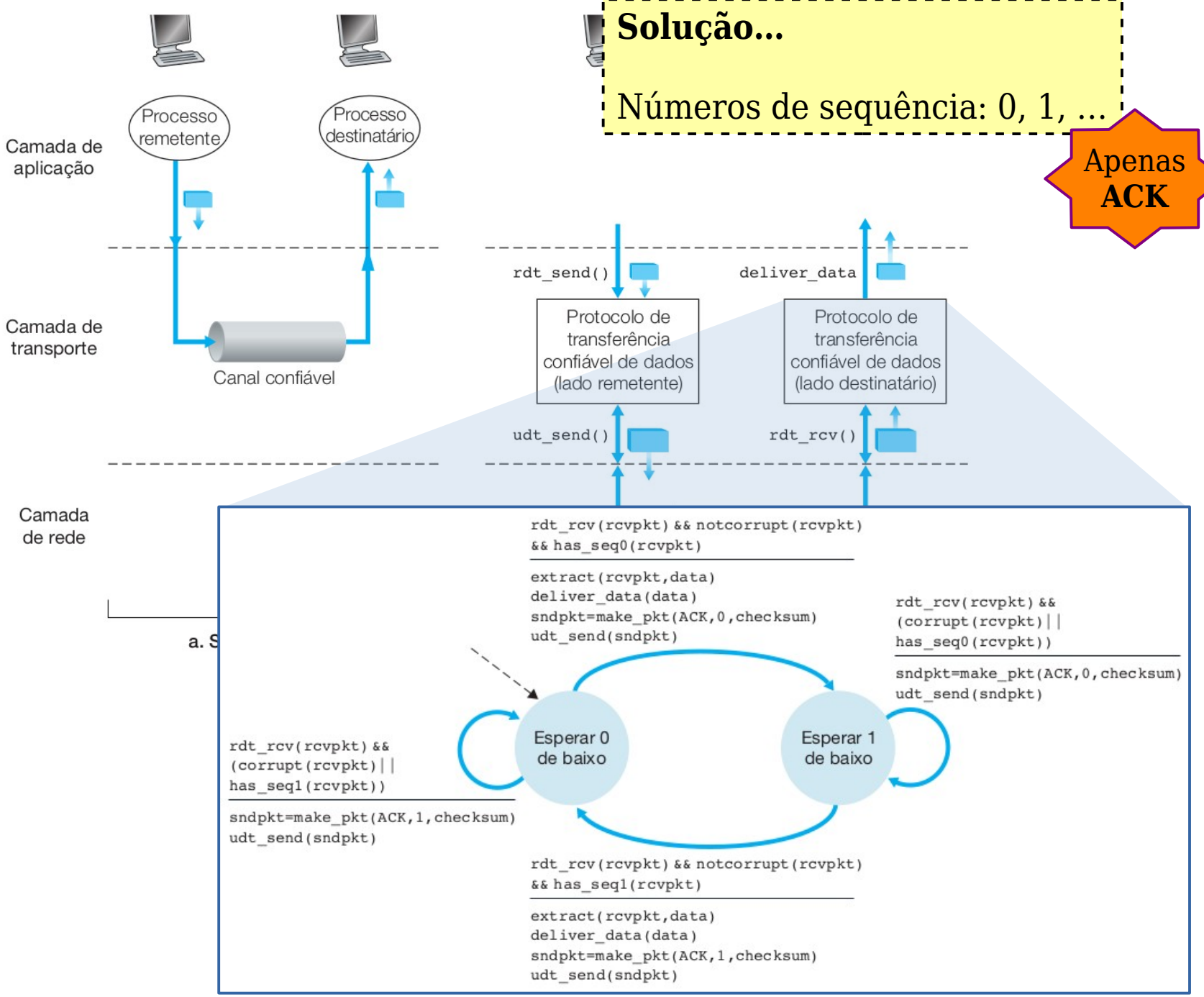


Transferência confiável de dados: sobre um canal com erros de bits

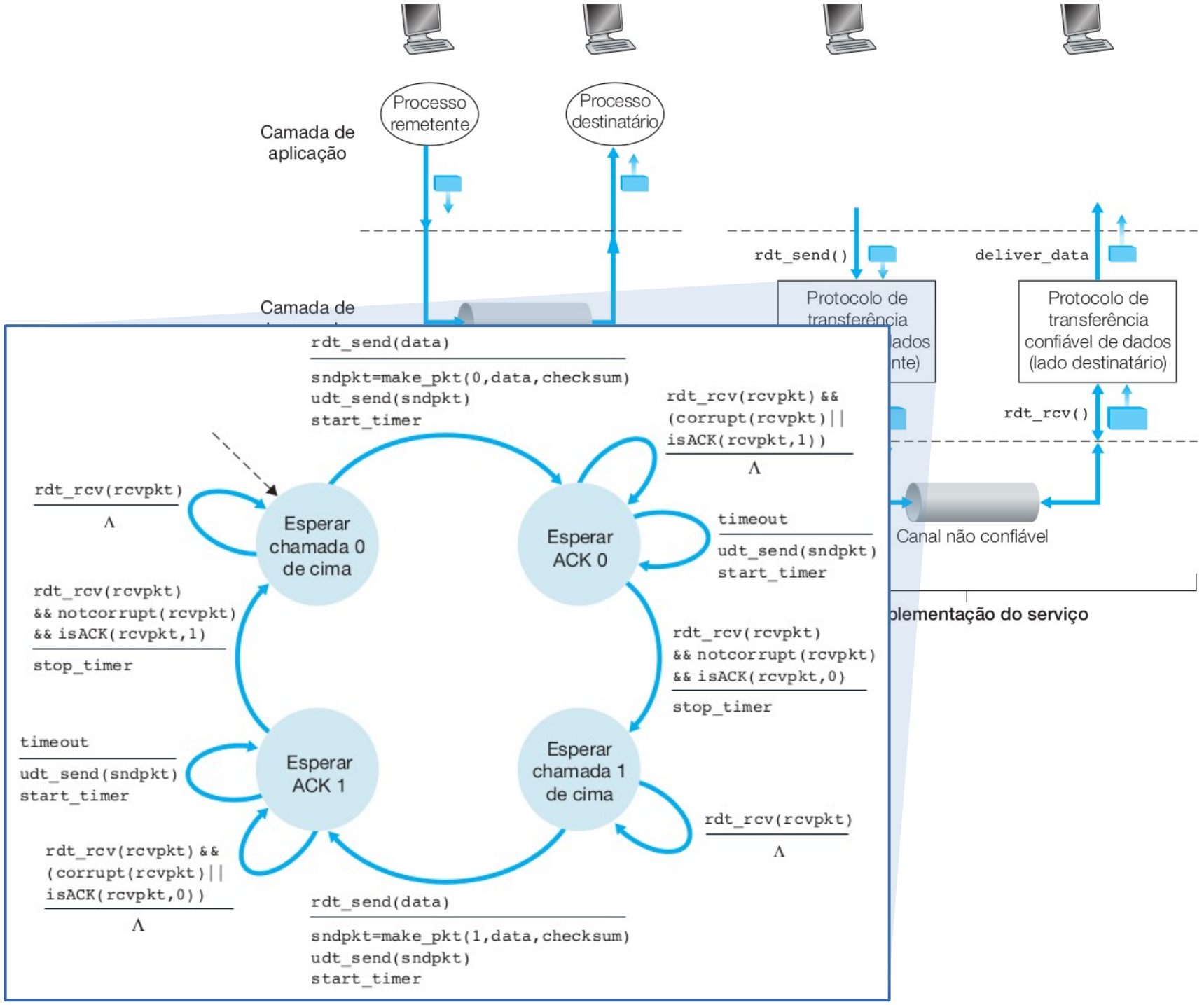
Solução...

Números de sequência: 0, 1, ...

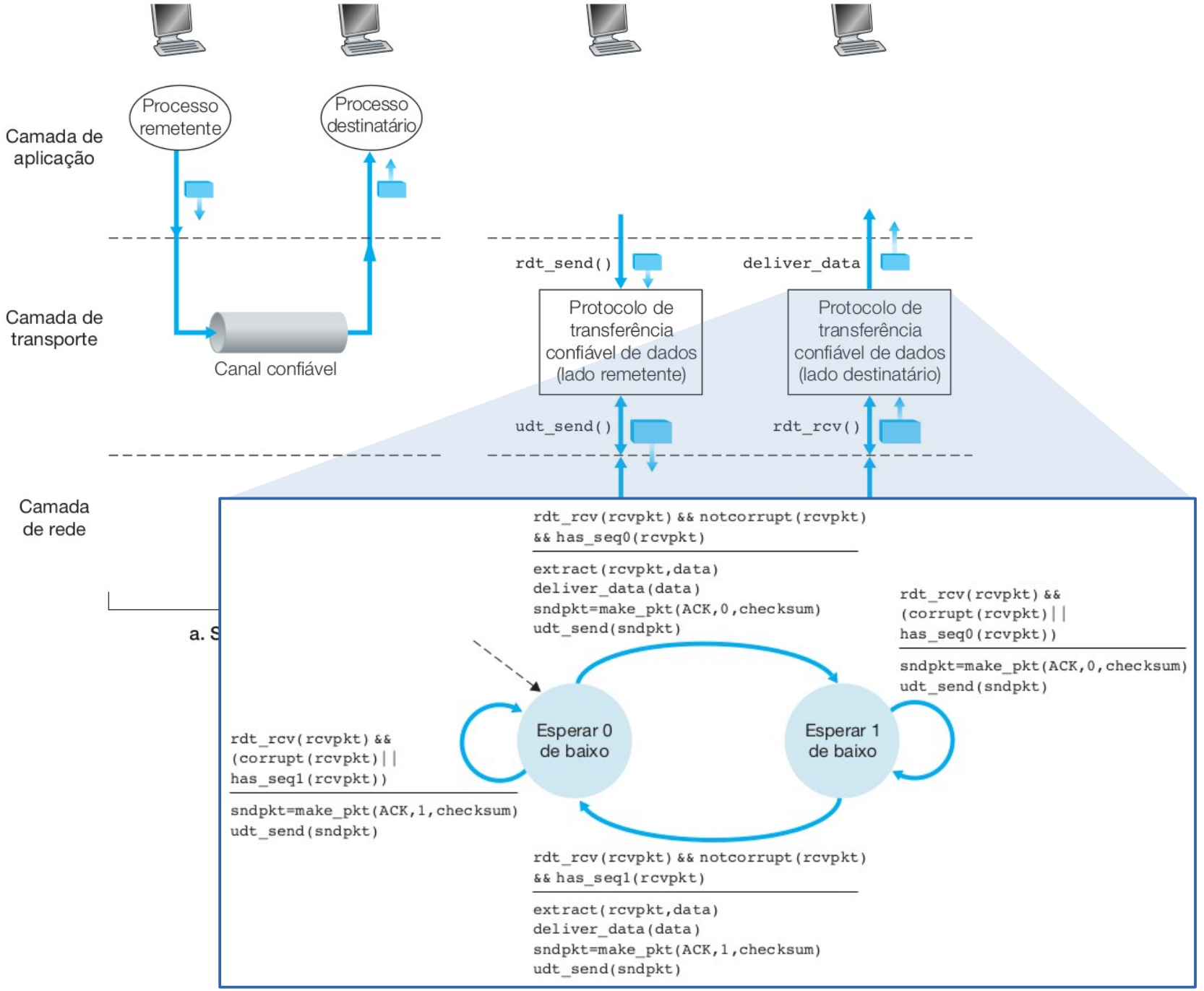
Apenas **ACK**



Transferência confiável de dados: considerando a perda de pacotes



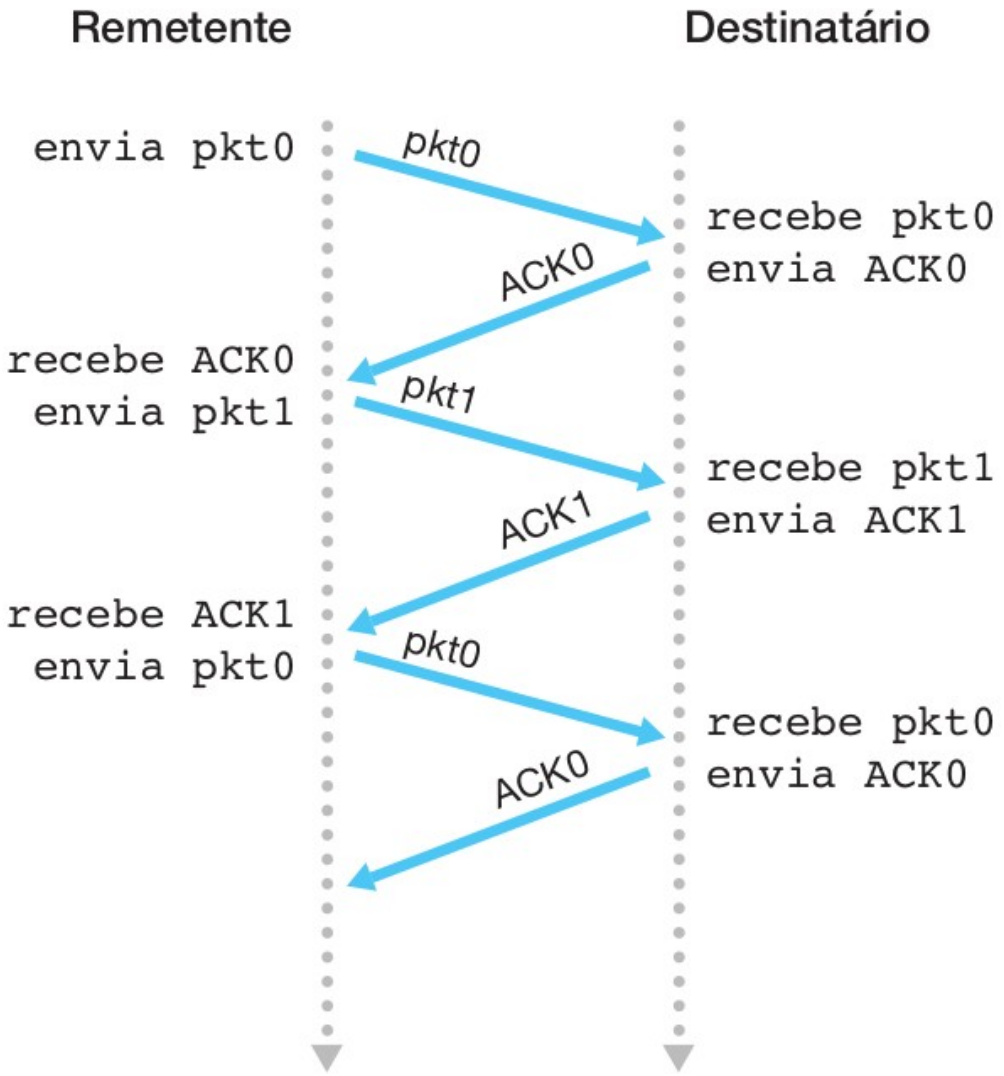
Transferência confiável de dados: considerando a perda de pacotes



O mesmo que na versão anterior

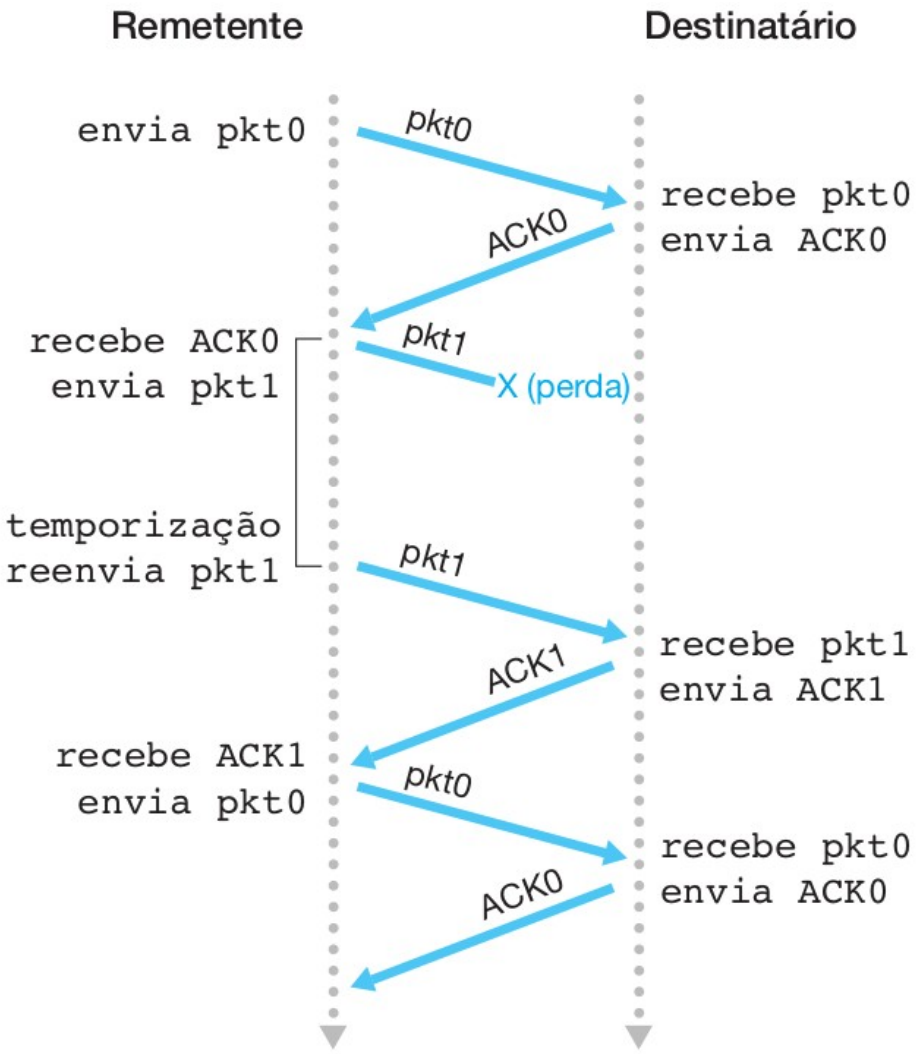
Transferência confiável de dados em ação...

Operação sem perda



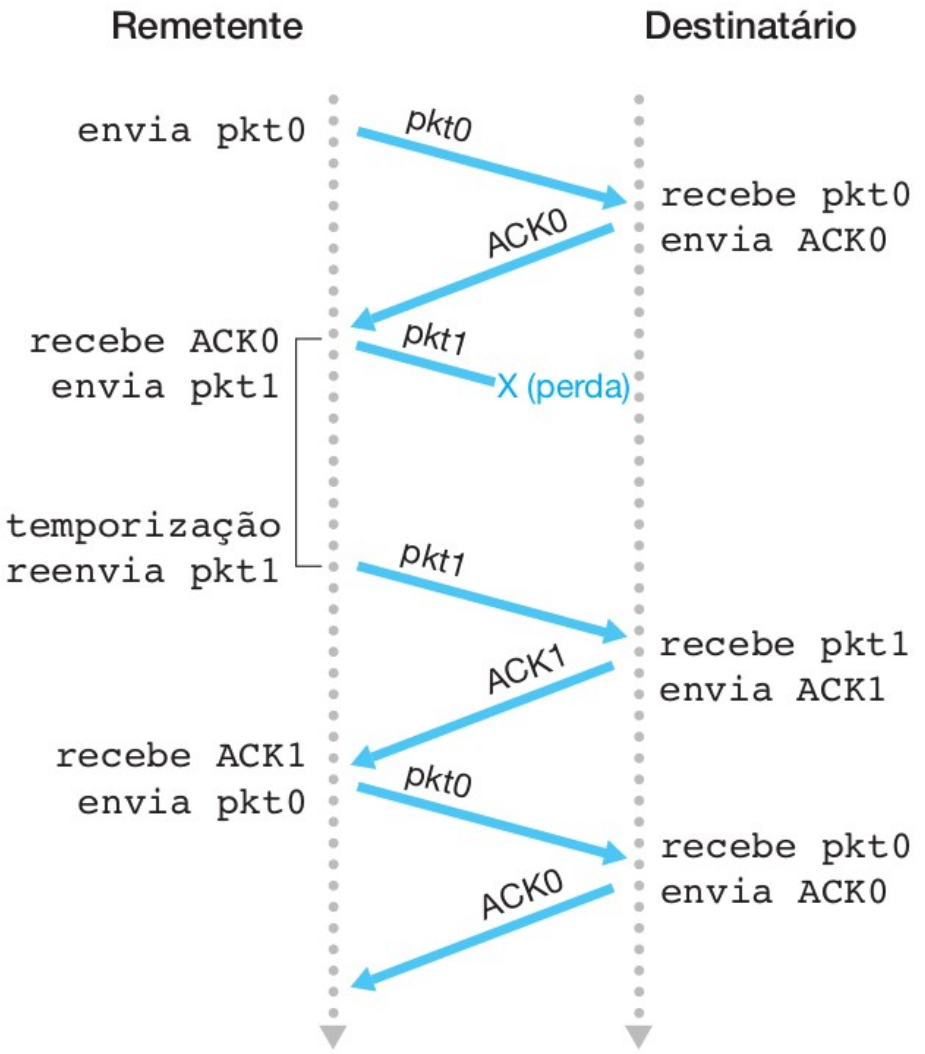
Transferência confiável de dados em ação...

Pacote perdido

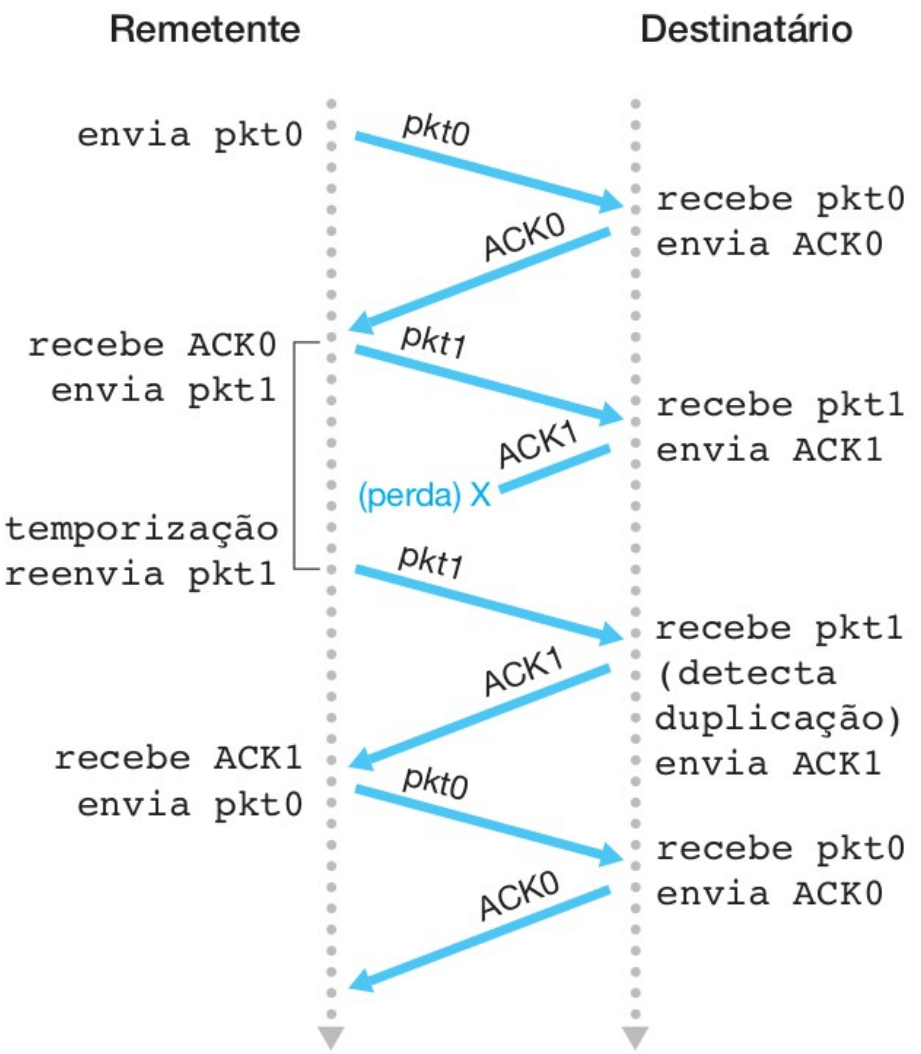


Transferência confiável de dados em ação...

Pacote perdido

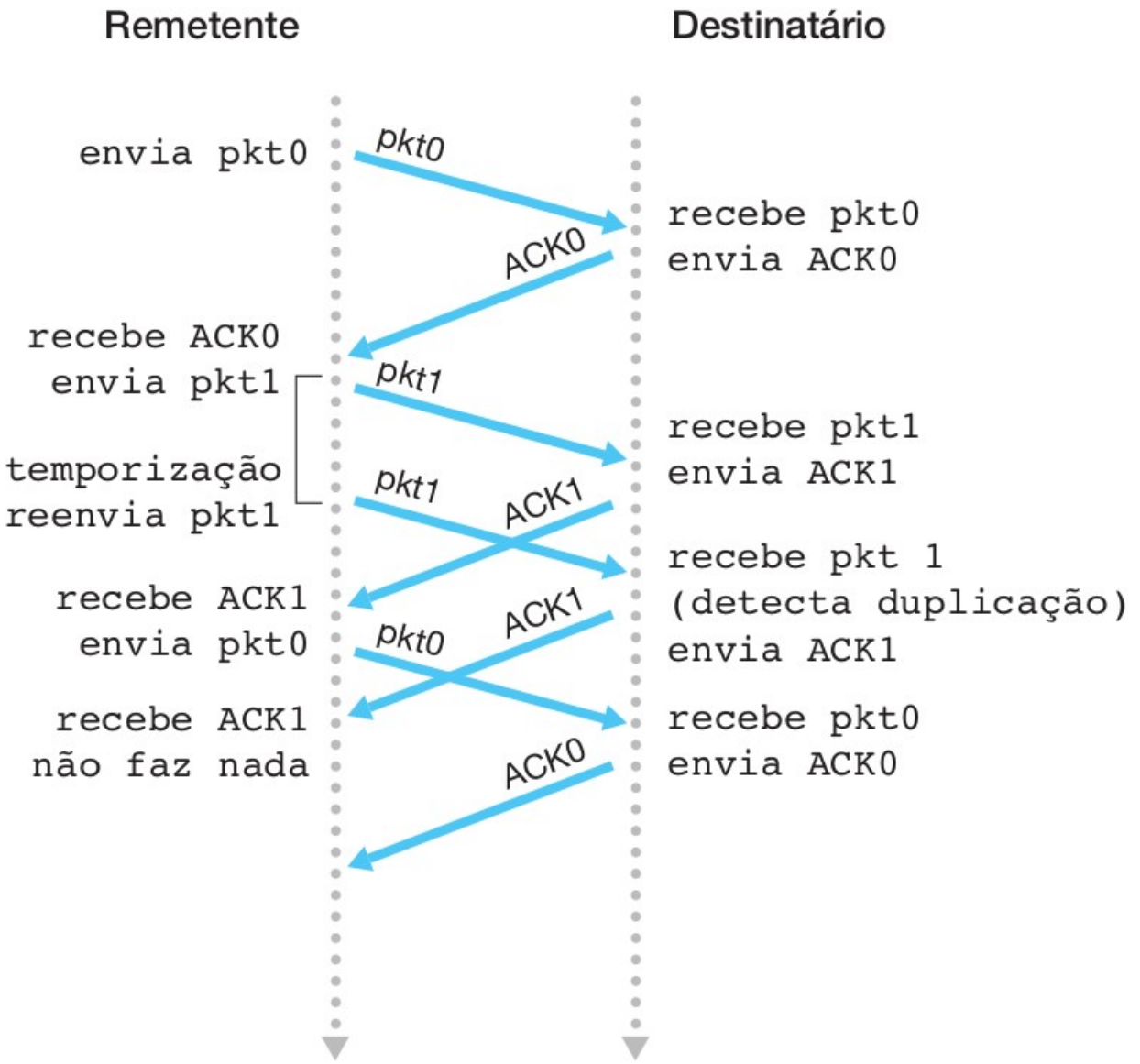


ACK perdido

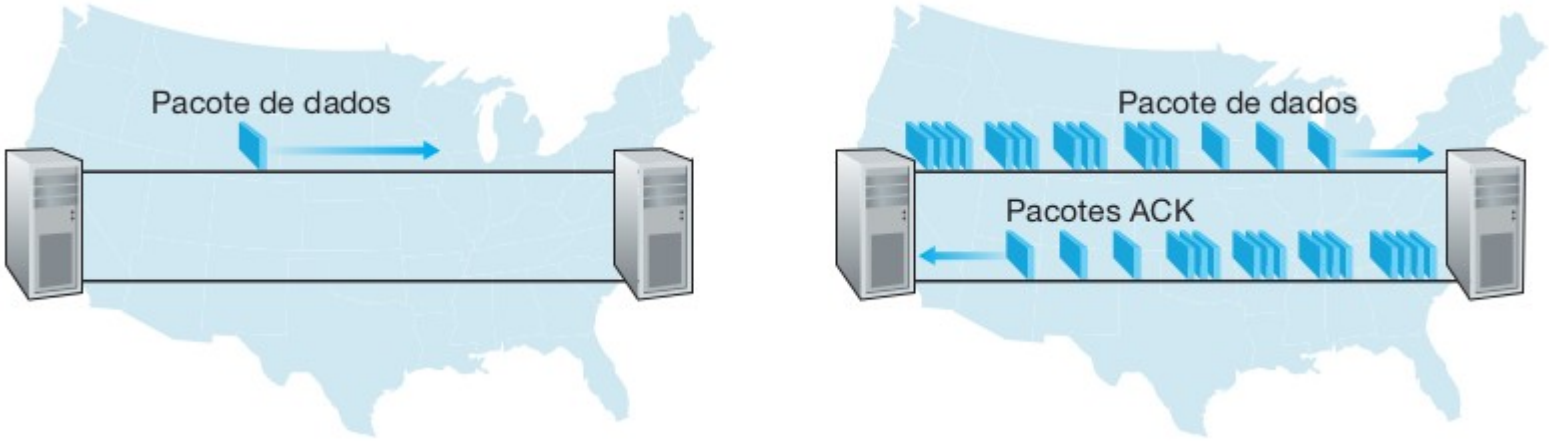


Transferência confiável de dados em ação...

Temporização prematura



Transferência confiável de dados em pipeline

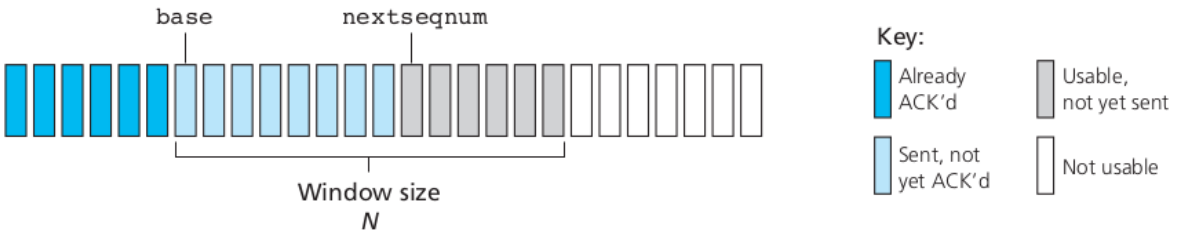


Como obter transmissão em pipeline?

- Protocolo Go-Back-N
- Protocolo Selective Repeat (Repetição Seletiva)

Em ambos os casos, até N pacotes podem ser enviados sem que tenham sido confirmados, mas é o protocolo de repetição seletiva que retransmite apenas pacotes suspeitos de não terem sido entregues.

Go-Back-N



← Remetente

```
rdt_send(data)
if(nextseqnum<base+N){
   sndpkt[nextseqnum]=make_pkt(nextseqnum,data,checksum)
   udt_send(sndpkt[nextseqnum])
   if(base==nextseqnum)
      start_timer
   nextseqnum++
   }
else
   refuse_data(data)
```

Λ

base=1
nextseqnum=1

Esperar

timeout

start_timer
udt_send(sndpkt[base])
udt_send(sndpkt[base+1])
...
udt_send(sndpkt[nextseqnum-1])

rdt_rcv(rcvpkt) && corrupt(rcvpkt)

Λ

rdt_rcv(rcvpkt) && notcorrupt(rcvpkt)

```
base=getacknum(rcvpkt)+1
If(base==nextseqnum)
   stop_timer
else
   start_timer
```

Janela de transmissão: auxilia no controle de fluxo (taxas de transmissão e recepção) e congestionamento da rede.

Destinatário →

```
rdt_rcv(rcvpkt)
  && notcorrupt(rcvpkt)
  && hasseqnum(rcvpkt,expectedseqnum)
extract(rcvpkt,data)
deliver_data(data)
sndpkt=make_pkt(expectedseqnum,ACK,checksum)
udt_send(sndpkt)
expectedseqnum++
```

Esperar

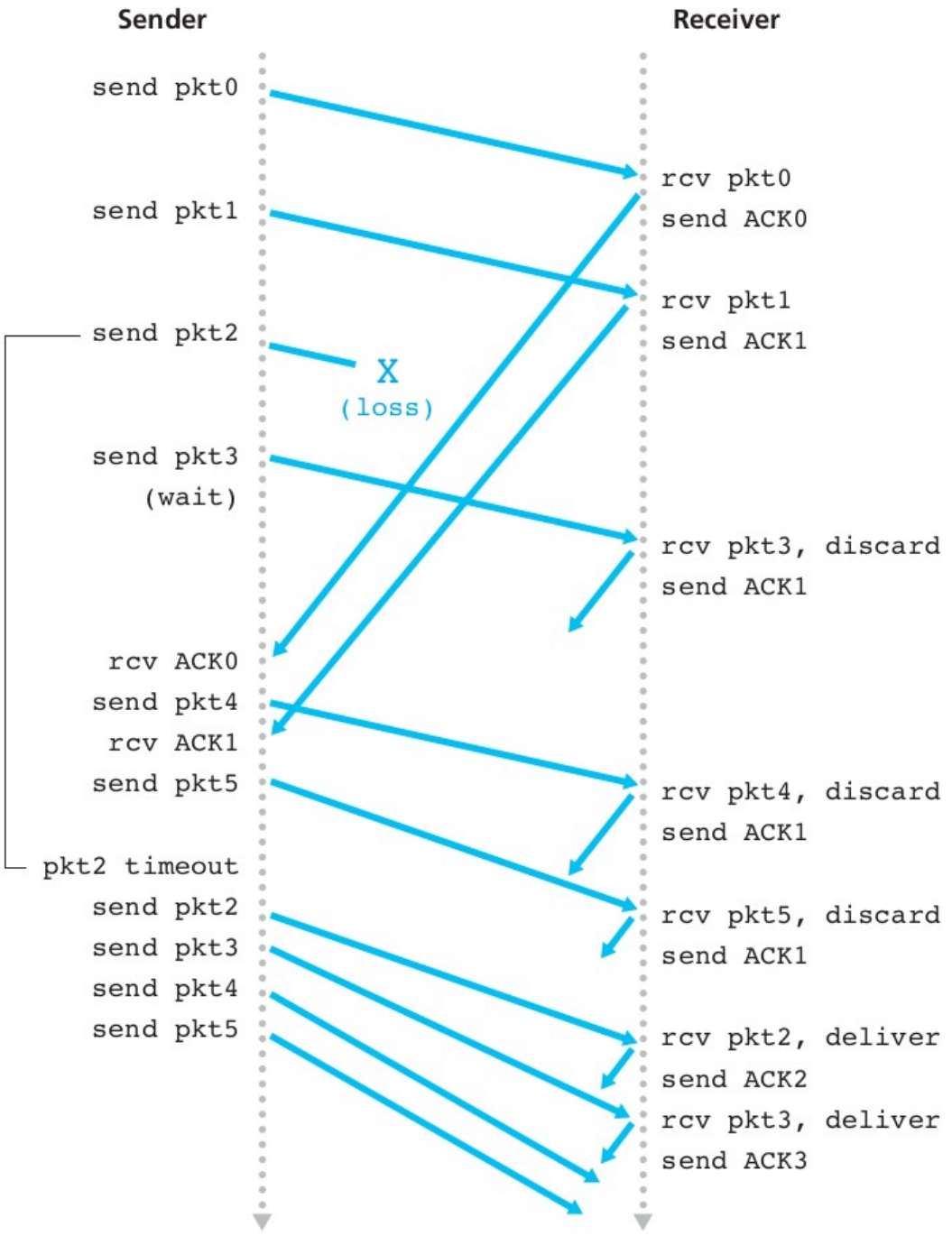
default

udt_send(sndpkt)

Λ

expectedseqnum=1
sndpkt=make_pkt(0,ACK,checksum)

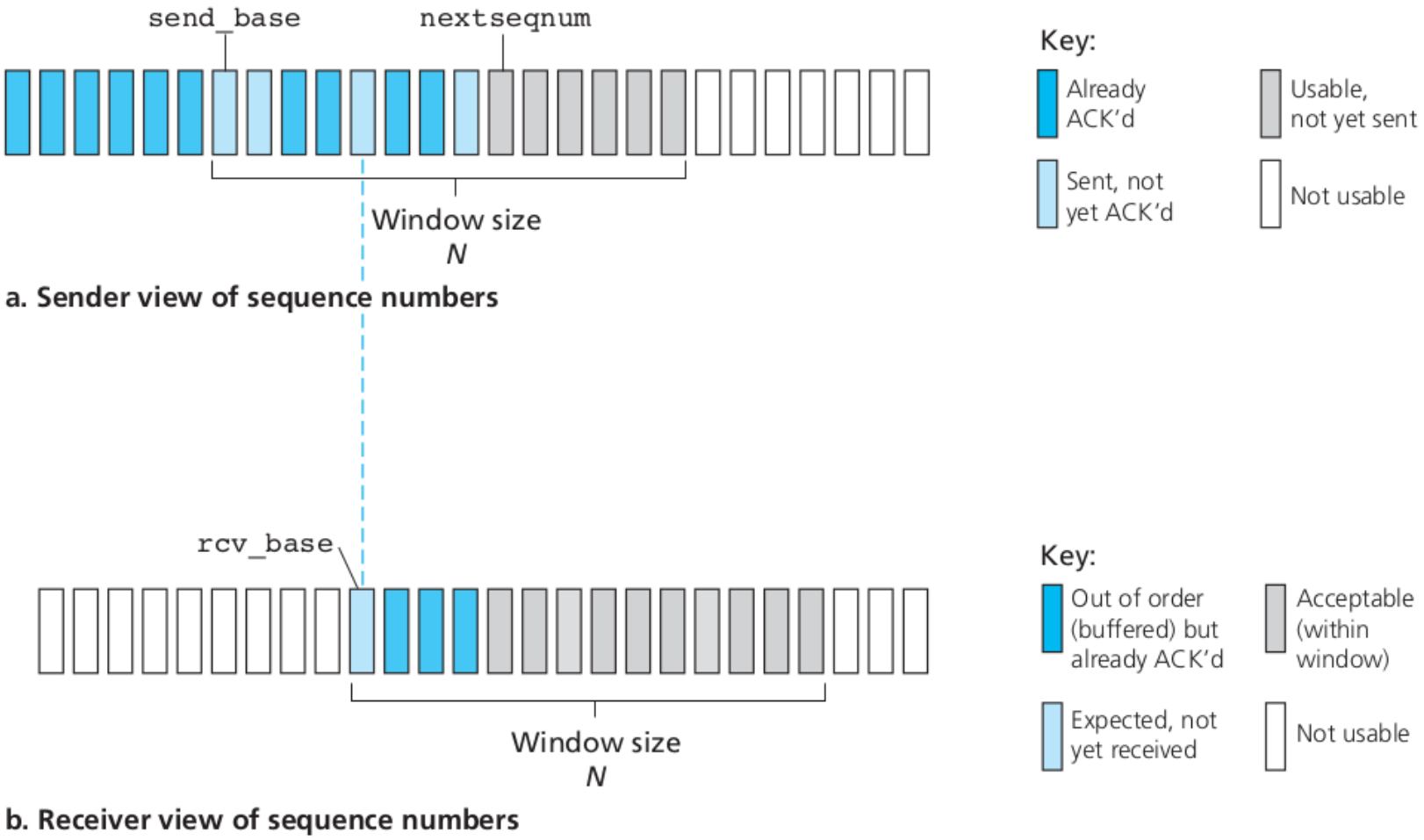
Go-Back-N em ação...



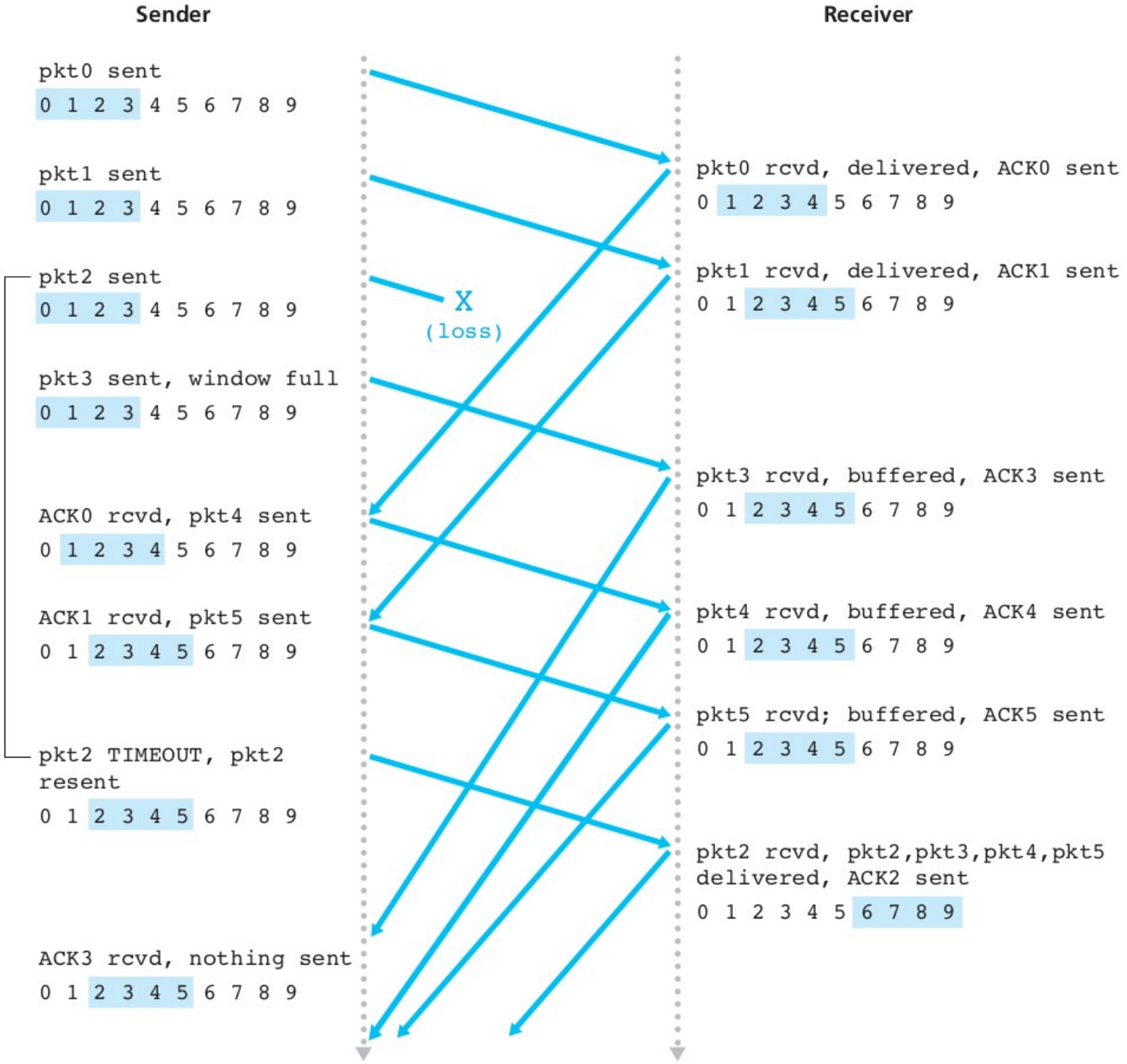
https://media.pearsoncmg.com/ph/esm/ecs_kurose_compnetwork_8/cw/

Repetição Seletiva

Análogo ao protocolo Go-Back-N, mas com uma janela de recepção no lado destinatário.



Repetição Seletiva em ação...



https://media.pearsoncmg.com/ph/esm/ecs_kurose_compnetwork_8/cw/

Mecanismos de transferência confiável de dados – Resumo

Soma de verificação: Detecção de erros de bits.

Temporizador: Utilizado na retransmissão de pacotes perdidos, corrompidos ou não confirmados → pode gerar cópias duplicadas.

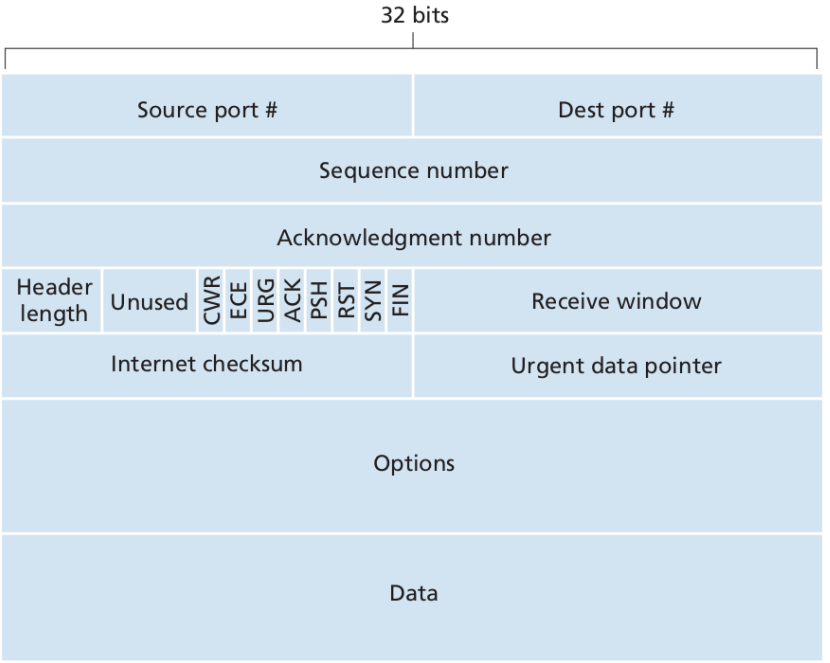
Número de sequência: Uma lacuna indica um pacote não recebido, um número duplicado indica um pacote que já foi recebido.

Reconhecimento (ACK): Informa ao remetente que um pacote foi recebido. O reconhecimento pode ser cumulativo ou individual.

Reconhecimento negativo (NAK): Informa ao remetente que um pacote ainda não foi recebido.

Janela (transmissão em pipeline): Possibilita o envio de vários pacotes em série. Permite o controle de fluxo entre o remetente e o destinatário e também a adaptação ao congestionamento da rede.

De volta ao TCP...



Números de sequência e de reconhecimento: Usados pelo serviço confiável de transferência de dados oferecido pelo TCP.

Comprimento do cabeçalho (4 bits): Tamanho do cabeçalho TCP em palavras de 32 bits.

Janela de recepção: Número de bytes que o host emissor está disposto a aceitar (controle de fluxo).

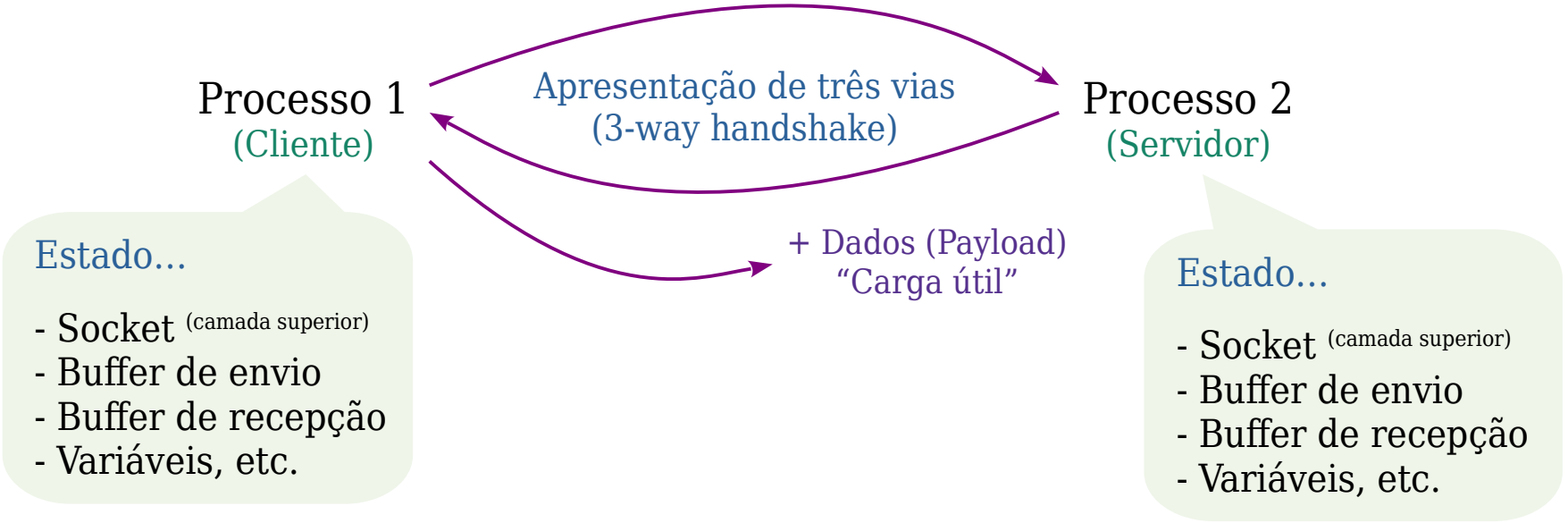
Opções: Opcional e de comprimento variável. Usado para negociar o MSS entre os hosts comunicantes ou para ajustar a escala da janela de recepção em redes de alta velocidade.

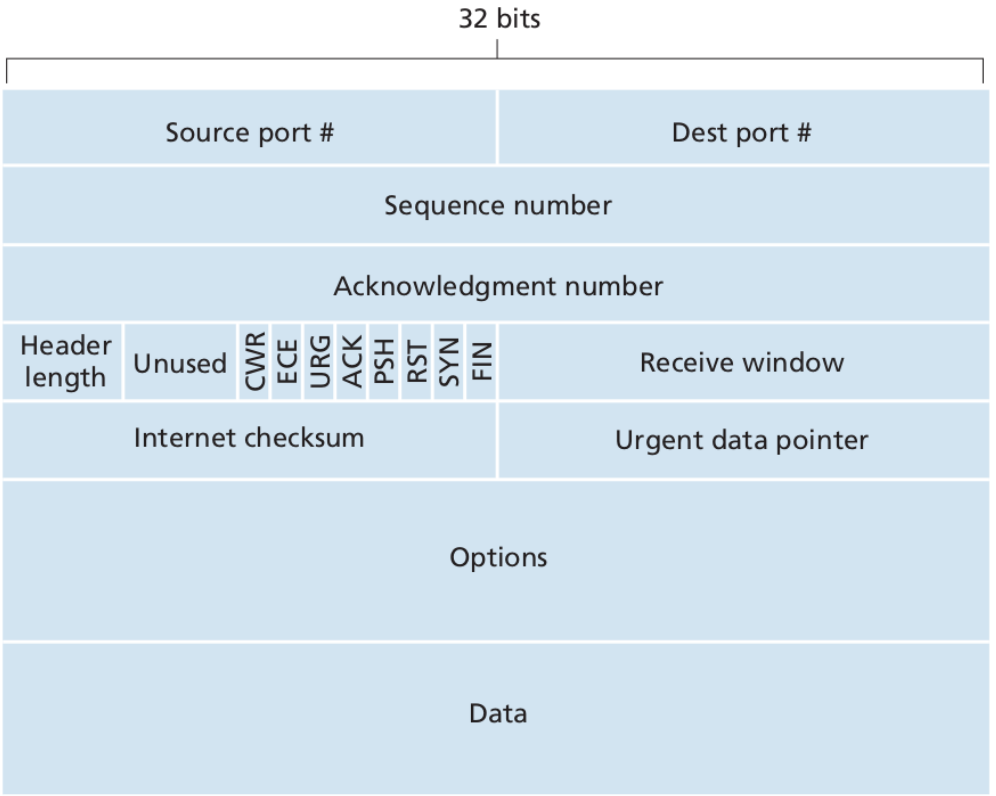
Flags...

- CWR, ECE: Usados na notificação explícita de congestionamento.
- URG: Indica que o segmento contém dados urgentes. “Ponteiro de urgência” aponta para o último byte desses dados urgentes.
- ACK: Indica se o valor carregado em “Número de reconhecimento” é fornecido.
- PSH: Indica que os dados devem ser entregues a camada superior imediatamente.
- RST: Usado em uma resposta a uma tentativa de conexão quando o serviço na porta destino não existe.
- SYN, FIN: Usados para iniciar e encerrar uma conexão TCP.

URG, PSH e “Ponteiro de urgência” não são usados na prática.

Estabelecimento de uma conexão TCP





Apresentação de três vias

1. Cliente: envia segmento SYN

Número de sequência = *a* (aleatório)
SYN = 1

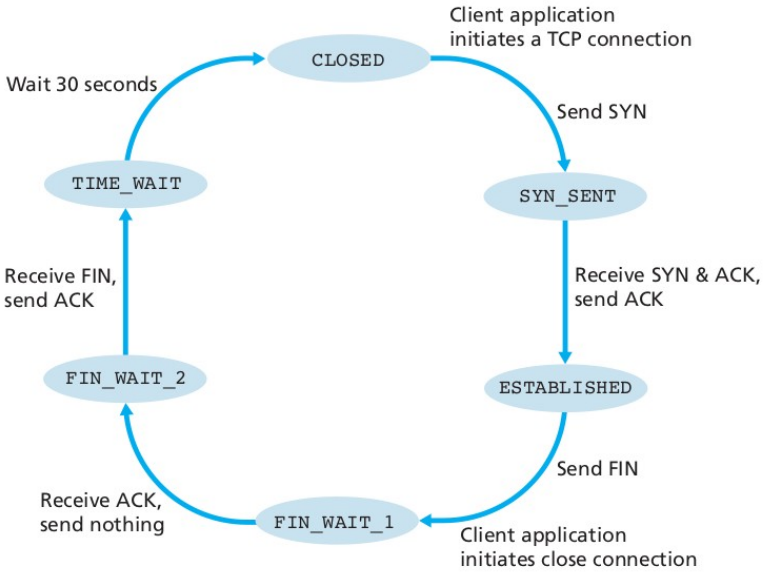
2. Servidor: aloca estado e envia segmento SYN/ACK

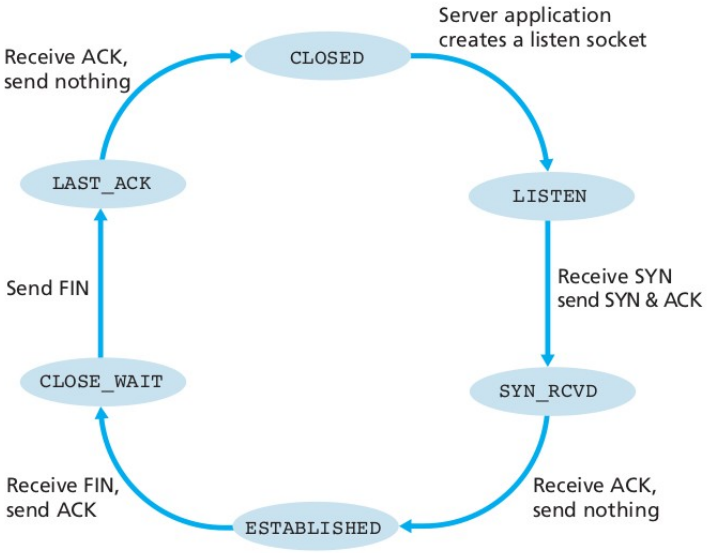
Número de sequência = *b* (aleatório)
Número de reconhecimento = *a* + 1
SYN = 1
ACK = 1

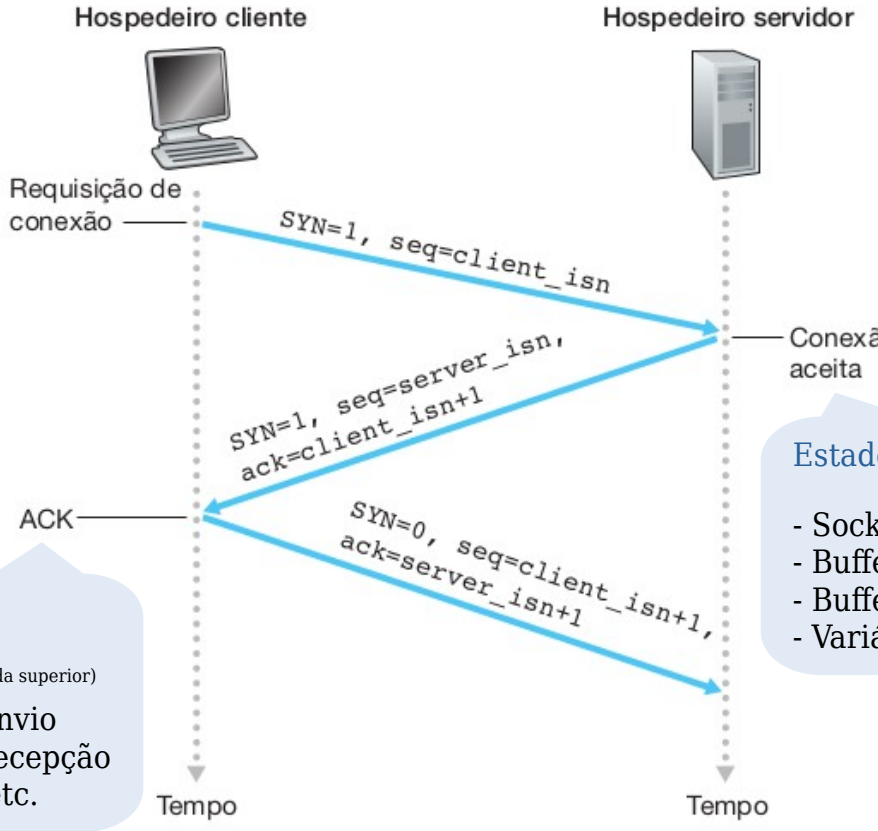
3. Cliente: aloca estado e inicia transmissão de dados

Número de sequência = *a* + 1
Número de reconhecimento = *b* + 1
ACK = 1
Dados

Estabelecimento de uma conexão TCP







Estado...

- Socket (camada superior)
- Buffer de envio
- Buffer de recepção
- Variáveis, etc.

Estado...

- Socket (camada superior)
- Buffer de envio
- Buffer de recepção
- Variáveis, etc.

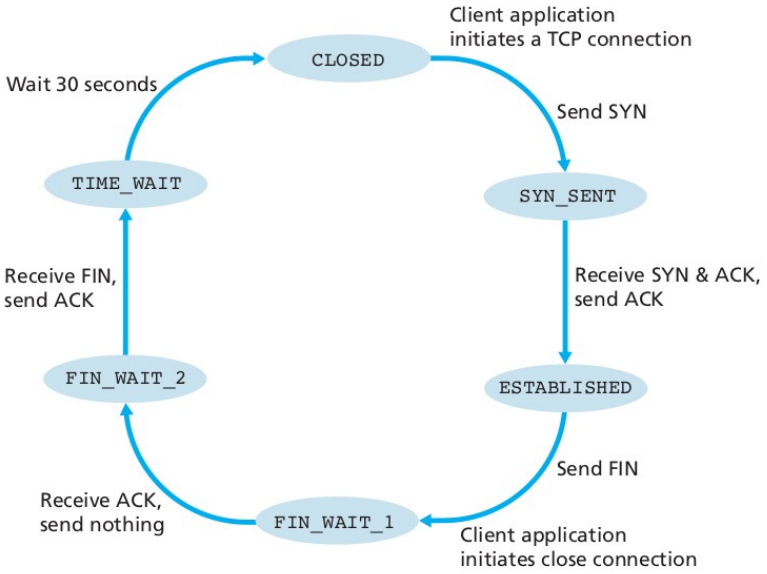
Ataque DoS SYN

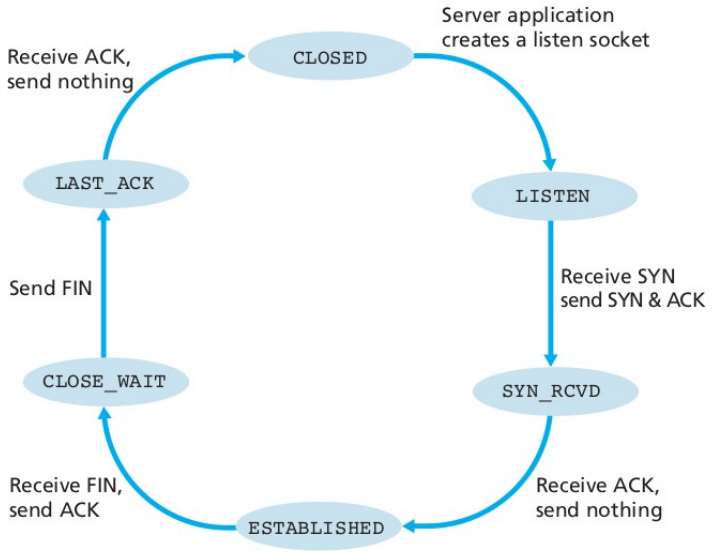
"Ataque de inundação SYN"

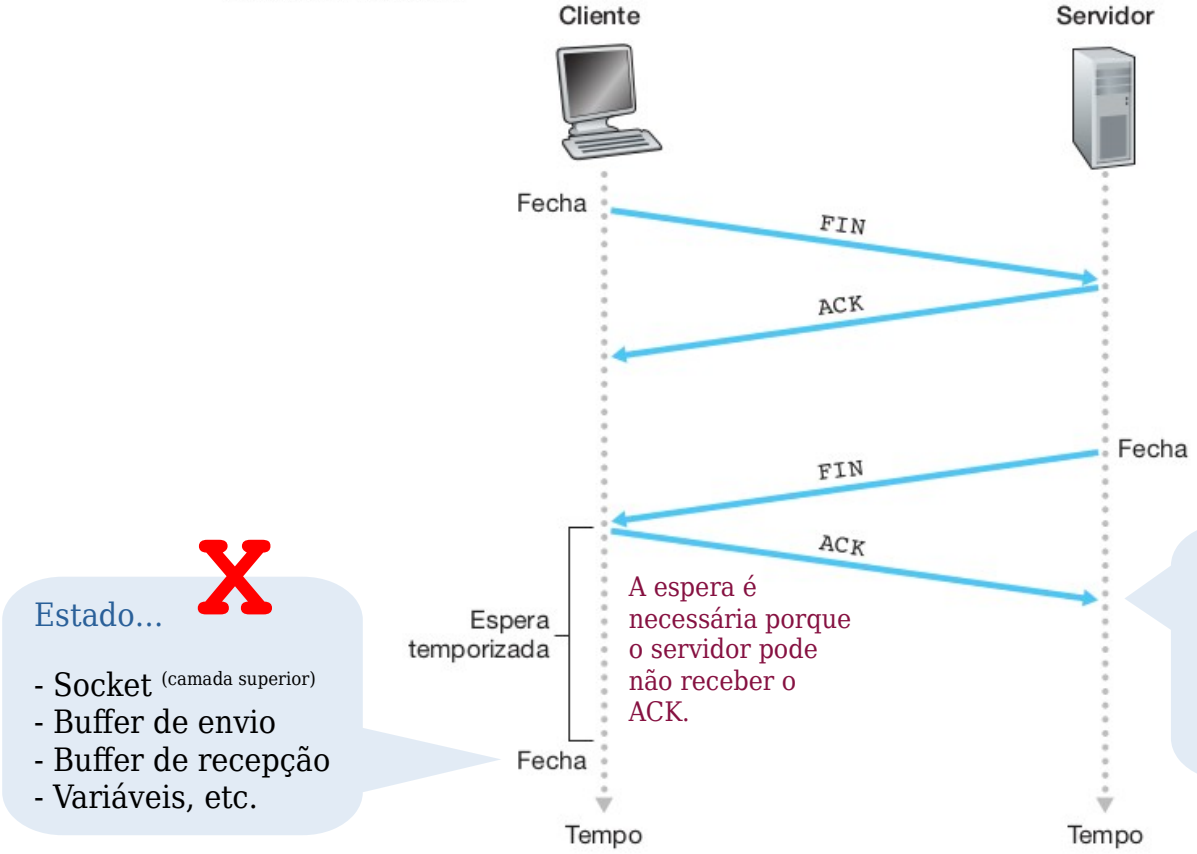
Proteção via SYN cookies

Basicamente o servidor atrasa a instanciação do estado até que o cliente confirme via ACK o número de sequência gerado no servidor por um hash envolvendo IPs, portas e um número secreto (cookie).

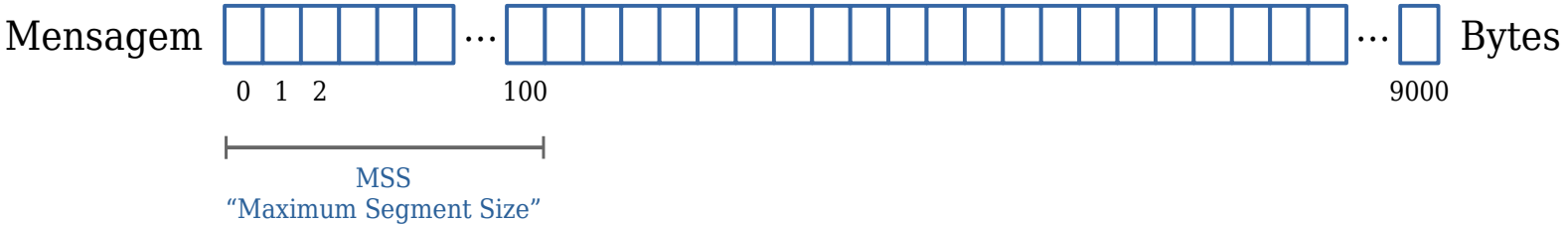
Encerramento de uma conexão TCP



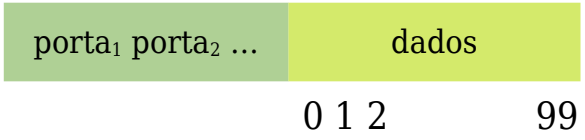




Princípio básico de funcionamento do TCP



Segmento TCP 1



Número de sequência = (a + 1) + 0

a: Número aleatório escolhido durante o estabelecimento da conexão.

Minimiza a possibilidade de um segmento de uma conexão já encerrada ser tomado como válido.

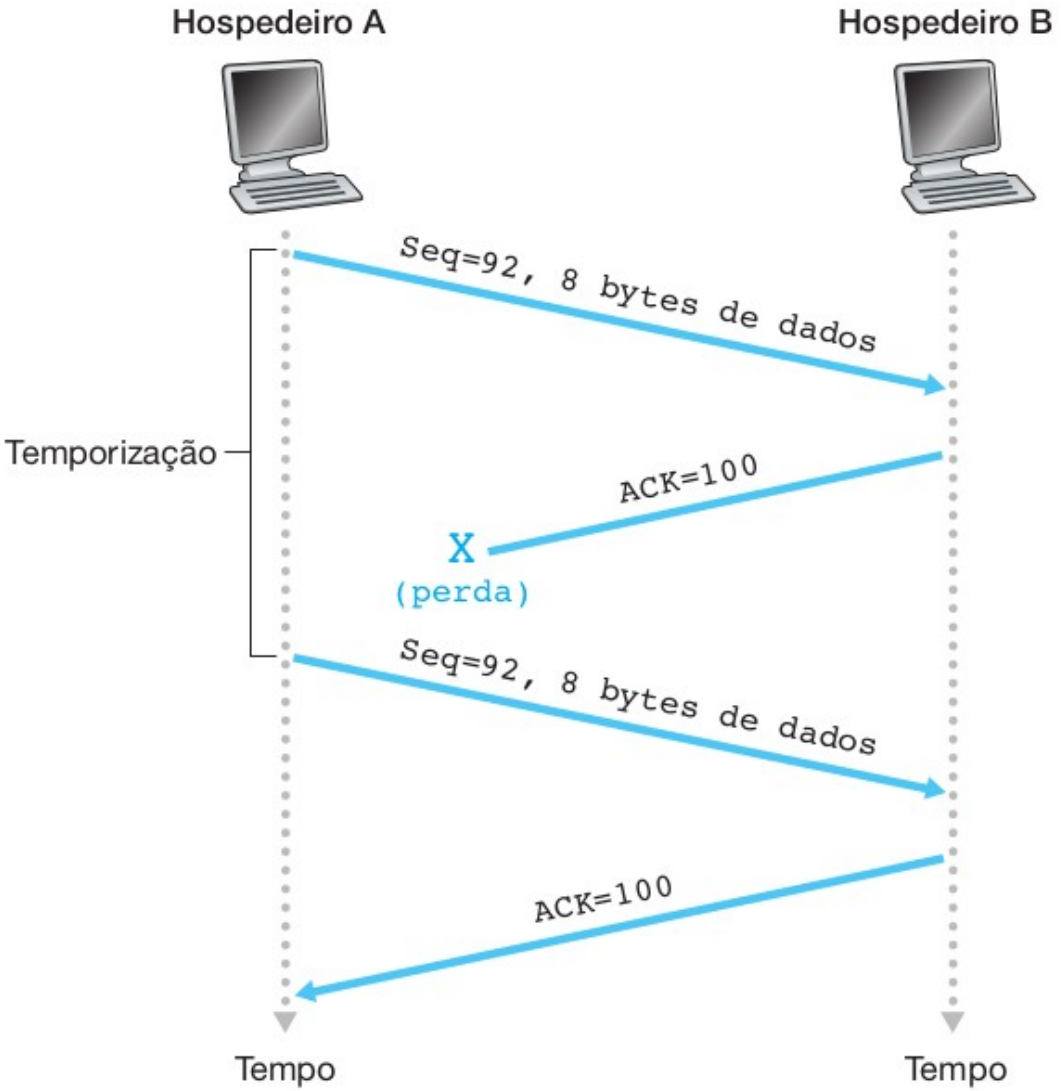
Segmento TCP 2

Número de sequência = (a + 1) + 100

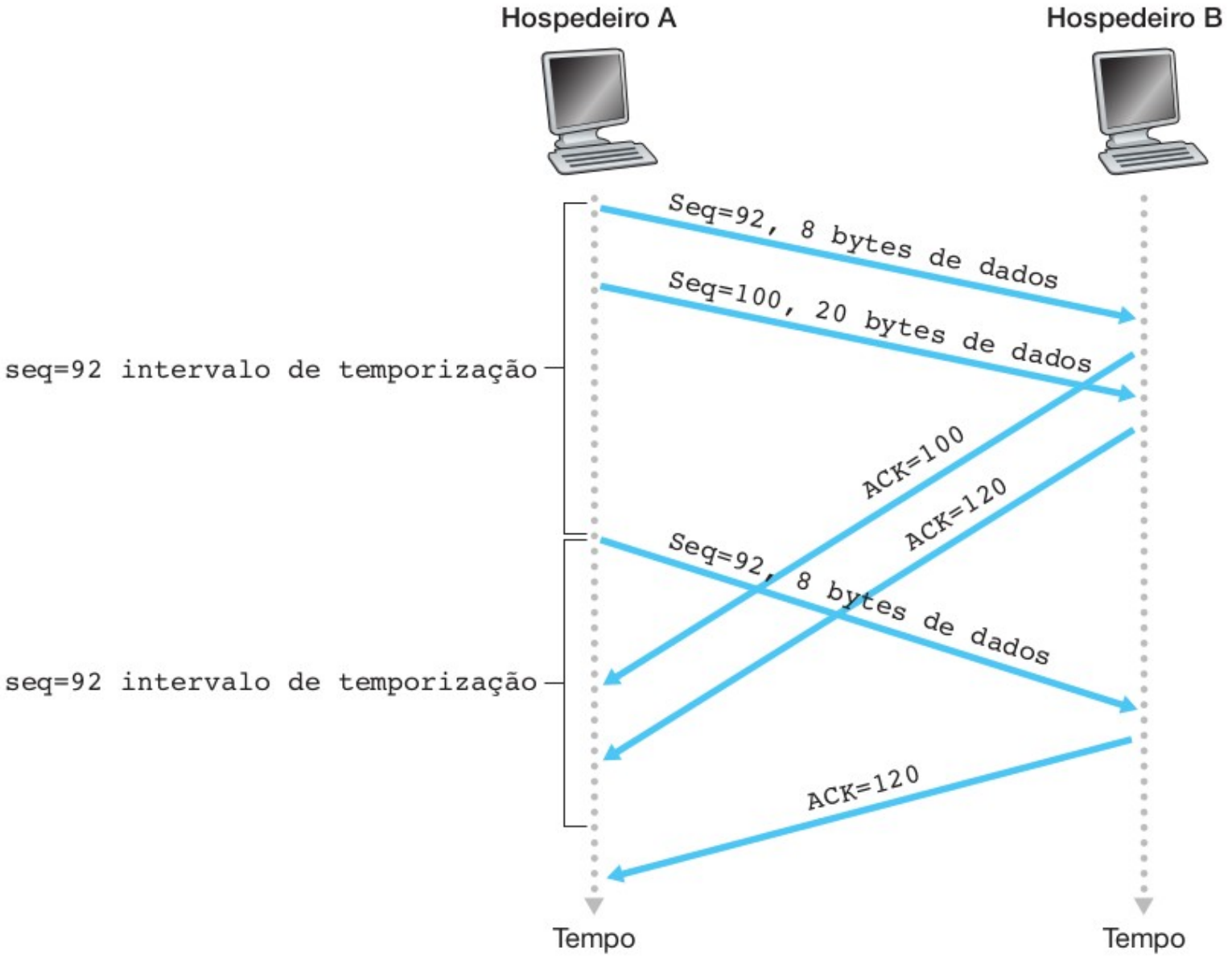
Em um segmento TCP enviado pelo cliente (servidor), o número de reconhecimento que “pega carona” com o envio indica (quando a flag ACK=1) o número de sequência do próximo byte a ser enviado pelo servidor (cliente).

No TCP, mensagens **ACK** “pegam carona” com respostas do servidor.

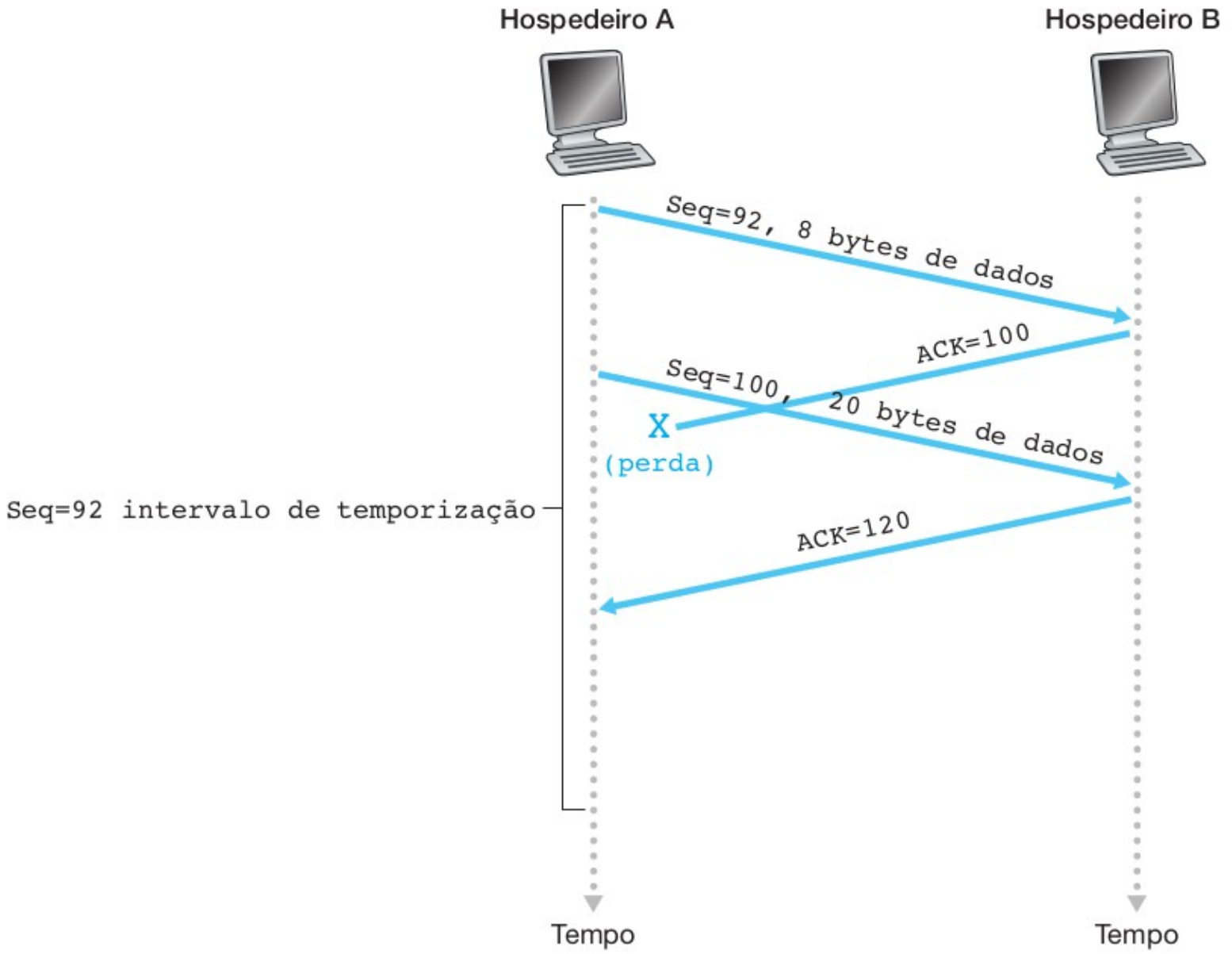
TCP e a transferência confiável de dados



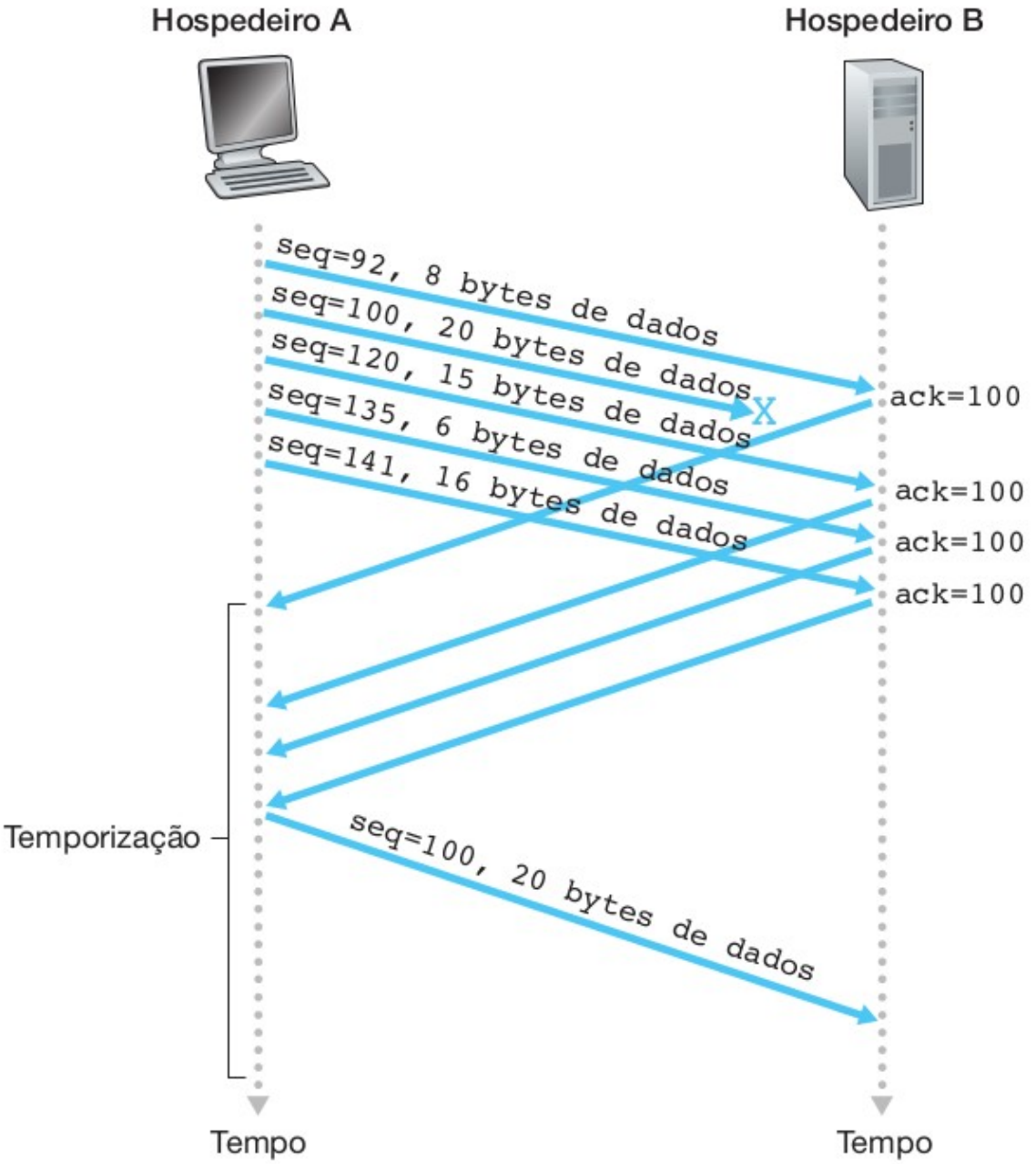
TCP e a transferência confiável de dados



TCP e a transferência confiável de dados



TCP e a transferência confiável de dados

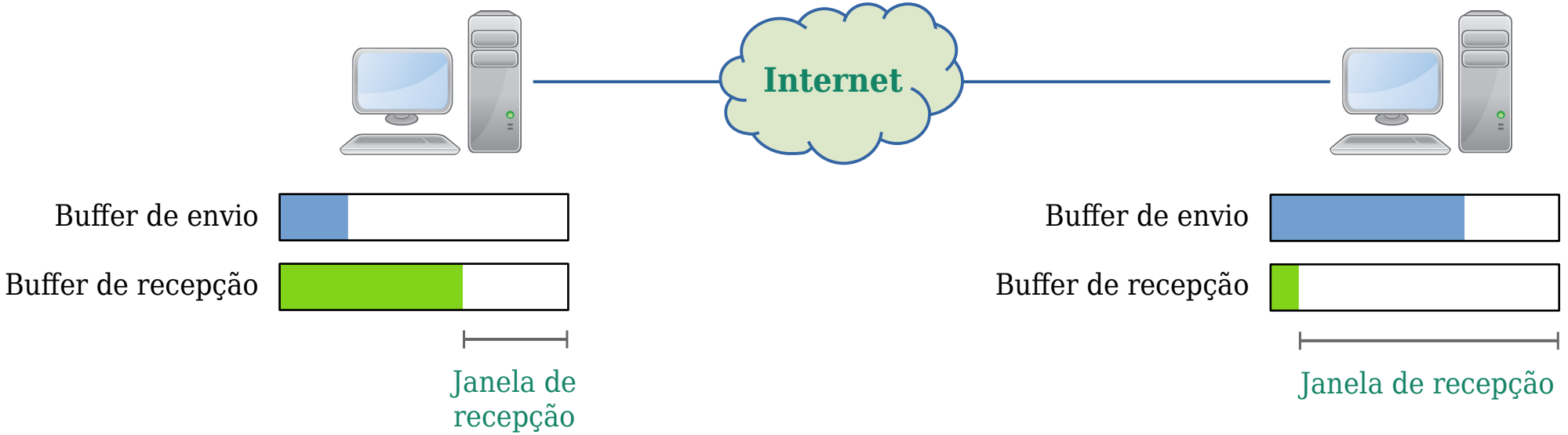


TCP: Go-Back-N ou Repetição Seletiva?

No TCP, ACKs são cumulativos (Go-Back-N), contudo o protocolo também apresenta alguns benefícios do protocolo de repetição seletiva.

Logo: TCP → Go-Back-N + Repetição Seletiva

Controle de fluxo no TCP (serviço de compatibilização de velocidade)



Ao receber um pacote, o nó receptor lê o campo TCP “Janela de recepção” de modo a identificar o valor atual da janela de recepção no processo emissor.

Um processo remetente evita o transbordamento do buffer de recepção de seu destinatário garantindo que o número de bytes enviados e ainda não confirmados (**ACKs**) juntamente com os próximos pacotes a serem enviados não ultrapassem o limite da janela de recepção do destinatário.

Considerações sobre o congestionamento de rede

- A taxa agregada em que os pacotes chegam em um roteador deve ser menor que a capacidade do enlace de saída.

- Um pacote descartado por um roteador implica sempre em uma retransmissão a ser feita, mas principalmente indica um trabalho desperdiçado de um roteador anterior que encaminhou esse pacote: isso pode levar a uma taxa de vazão efetiva igual a zero.

Controle de congestionamento no TCP

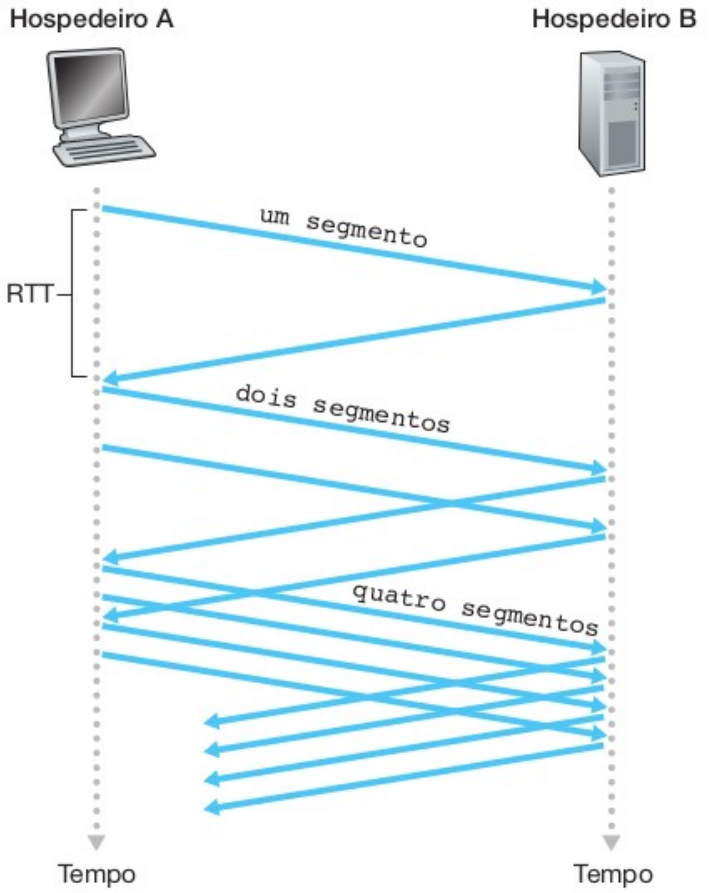
Os sistemas finais limitam suas janelas de recepção observando o comportamento da rede:

- Excesso de pacotes perdidos (retransmitidos) ou ACKs duplicados recebidos.
- Atrasos crescentes de RTT.

Alternativamente, um roteador (congestionado ou quase) pode marcar um datagrama IP em trânsito (bits ECN – Explicit Congestion Control) para que o destinatário, ao recebê-lo, envie uma mensagem ACK ao emissor notificando sobre o congestionamento da rede: bits CWR (Congestion Window Reduced) e ECE (Explicit Congestion notification Echo) no segmento TCP.

Considerações sobre o congestionamento de rede

O TCP utiliza uma variável de janela de congestionamento em cada socket. Dados enviados e não confirmados juntamente com dados a enviar não devem ultrapassar a janela de recepção do destinatário nem a janela de congestionamento local.



"A estratégia do TCP consiste em aumentar sua taxa de transmissão em resposta aos ACKs que chegam até que ocorra um evento de perda, momento em que a taxa de transmissão diminui. Desse modo, o remetente TCP aumenta sua taxa de transmissão para buscar a taxa pela qual o congestionamento se inicia, recua dela e de novo faz a busca para ver se a taxa de início do congestionamento foi alterada."

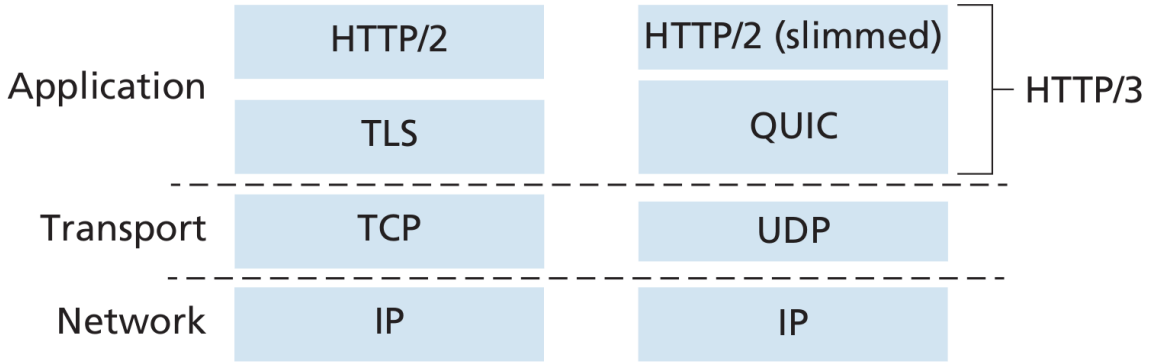
Nas primeiras versões do TCP não havia nenhuma sinalização explícita de congestionamento pela rede — os ACKs e eventos de perda serviam como sinais implícitos — e cada remetente TCP atuava apenas sobre informações locais e em momentos diferentes.

Mas e se os serviços de transporte oferecidos pelo UDP e TCP não forem adequados à uma aplicação?

Neste caso, o desenvolvedor pode escolher implementar seu próprio mecanismo de transporte na camada de aplicação utilizando o serviço de entrega simples oferecido pelo UDP.

Exemplo: QUIC – Quick UDP Internet Connections

QUIC é um protocolo da camada de aplicação que utiliza o serviço UDP para transportar mensagens HTTP com segurança, confiabilidade e controle de congestionamento.



Características

- Orientado à conexão e com pacotes criptografados. O QUIC combina as apresentações necessárias para estabelecer a conexão com aquelas que são necessárias para autenticação e criptografia.
- Assim como no HTTP/2 sobre o TCP, o QUIC possibilita mais de um fluxo com uma única conexão.
- Apesar de atuar sobre o UDP, oferece confiabilidade e controle de congestionamento, o que o torna mais amigável ao tráfego TCP.

nmap: Ferramenta de varredura de portas

E se o servidor não estiver aceitando conexões na porta de destino?

O servidor responde com um segmento TCP especial de reinicialização: bit de flag **RST** ativado. O protocolo UDP responde com um datagrama ICMP.

Como o nmap funciona? O nmap envia um segmento TCP SYN com um número de porta arbitrário para um destinatário.

1. O destinatário responde com um segmento TCP SYN/ACK. Neste caso o nmap informa que há um serviço ativo.

2. O destinatário responde com um segmento TCP RST. Neste caso o nmap informa que não existe um serviço ativo na porta requisitada.

3. O destinatário não responde. Neste caso não há como o nmap saber se o alvo está ativo ou se o pacote não passou por um firewall.

```
$ sudo nmap -sTU -p 1-65535 [-6] IP
```

-sTU: varre portas TCP e UDP
-6: IPv6

Questões de revisão

1. Considere uma conexão TCP entre o hospedeiro A e o hospedeiro B. Suponha que os segmentos TCP que trafegam do hospedeiro A para o hospedeiro B tenham número de porta de origem *x* e número de porta de destino *y*. Quais são os números de porta de origem e de destino para os segmentos que trafegam do hospedeiro B para o hospedeiro A?

2. Descreva por que um desenvolvedor de aplicação pode escolher rodar uma aplicação sobre UDP em vez de sobre TCP.

3. É possível que uma aplicação desfrute de transferência confiável de dados mesmo quando roda sobre UDP? Caso a resposta seja afirmativa, como isso acontece?

4. Suponha que um processo no hospedeiro C possua um socket UDP com número de porta 6789 e que o hospedeiro A e o hospedeiro B, individualmente, enviem um segmento UDP ao hospedeiro C com número de porta de destino 6789. Os dois segmentos serão encaminhados para o mesmo socket no hospedeiro C? Se sim, como o processo no hospedeiro C saberá que os dois segmentos vieram de dois hospedeiros diferentes?

5. Suponha que um servidor Web seja executado no computador C na porta 80. Esse servidor utiliza conexões persistentes e, no momento, está recebendo solicitações de dois computadores diferentes, A e B. Todas as solicitações estão sendo enviadas por meio do mesmo socket no computador C? Se estão passando por diferentes sockets, dois deles possuem porta 80? Discuta e explique.

Questões de revisão

6. Em um protocolo de transferência confiável de dados, como o TCP, por que precisamos de números de sequência?

7. Em um protocolo de transferência confiável de dados, como o TCP, por que precisamos de temporizadores?

8. Falso ou verdadeiro: O hospedeiro A está enviando ao hospedeiro B um arquivo grande por uma conexão TCP. Suponha que o hospedeiro B não tenha dados para enviar para o hospedeiro A. O hospedeiro B não enviará reconhecimentos para A porque ele não pode dar carona aos reconhecimentos dos dados.

9. Falso ou verdadeiro: O tamanho do campo "Janela de recepção" em uma conexão TCP nunca muda enquanto dura a conexão.

10. Falso ou verdadeiro: Imagine que o hospedeiro A envie ao hospedeiro B, por uma conexão TCP, um segmento com o número de sequência 38 e 4 bytes de dados. Nesse mesmo segmento, o número de reconhecimento será necessariamente 42.

Questões de revisão

11. Suponha que o hospedeiro A envie dois segmentos TCP um atrás do outro ao hospedeiro B sobre uma conexão TCP. O primeiro segmento tem número de sequência 90 e o segundo, número de sequência 110.

    a) Quantos dados tem o primeiro segmento?

    b) Suponha que o primeiro segmento seja perdido, mas o segundo chegue a B. No reconhecimento que B envia a A, qual será o número de reconhecimento?