Cap. 03 – Camada de Transporte

3.1 – Serviços da Camada de Transporte

3.1.1 – Camada de Transporte e de Redes

3.1.2 – Visão Geral da Camada de Transporte

3.2 – Multiplexação / Demultiplexação

3.3 – Transporte não Orientado a Conexão

3.3.1 – Estrutura do Segmento UDP

3.3.3 – Soma de Verificação no UDP

… Cap. 03 – Camada de Transporte

3.4 – Princípios da Transferência Confiável de Dados
3.4.1 – Protocolo de Transf. Confiável de Dados #1
3.4.2 – Protocolo c/ Transf. Confiável de Dados #2
3.4.3 – Protocolo Go-Back-N
3.4.4 – Protocolo de Repetição Seletiva

3.5 – Transporte Orientado por Conexão (TCP)
3.5.1 – Conexão TCP / 3.5.2 – Estrutura do Segmento TCP
3.5.3 – Controle de Temporização / Retransmissão
3.5.4 – Transferência Confiável de Dados
3.5.5 – Controle de Fluxo / 3.5.6 - Gerenciamento de Conexão TCP

… Cap. 03 – Camada de Transporte

3.6 – Princípios do Controle de Congestionamento
3.6.1 – Causas e Custos do Congestionamento
3.6.2 – Mecanismos de Controle de Congestionamento
3.6.3 – Exemplos de Controle de Congestionamento

3.7 – Controle de Congestionamento no TCP
3.7.1 - Partida Lenta
3.7.2 – Prevenção de Congestionamento
3.7.3 – Recuperação Rápida
3.7.4 – Controle de Congestionamento no TCP

Referências Bibliográficas

- James F. Kurose; Keith W. Ross – Redes de Computadores e a Internet: Uma Abordagem Top-Down – Pearson São Paulo; 6th; 2014; ISBN: 978-85-430-1443-2
- Lectures dos autores James F. Kurose; Keith W. Ross “https://gaia.cs.umass.edu/kurose_ross/eighth.htm”
- Notas de Aula do Prof. Maurício Magalhães e Eleri Cardozo da Faculdade de Engenharia Elétrica e de Computação (FEEC) da UNICAMP “www.dca.feec.unicamp.br/[~mauricio/~eleri]”.

3.1 – Serviços da Camada de Transporte

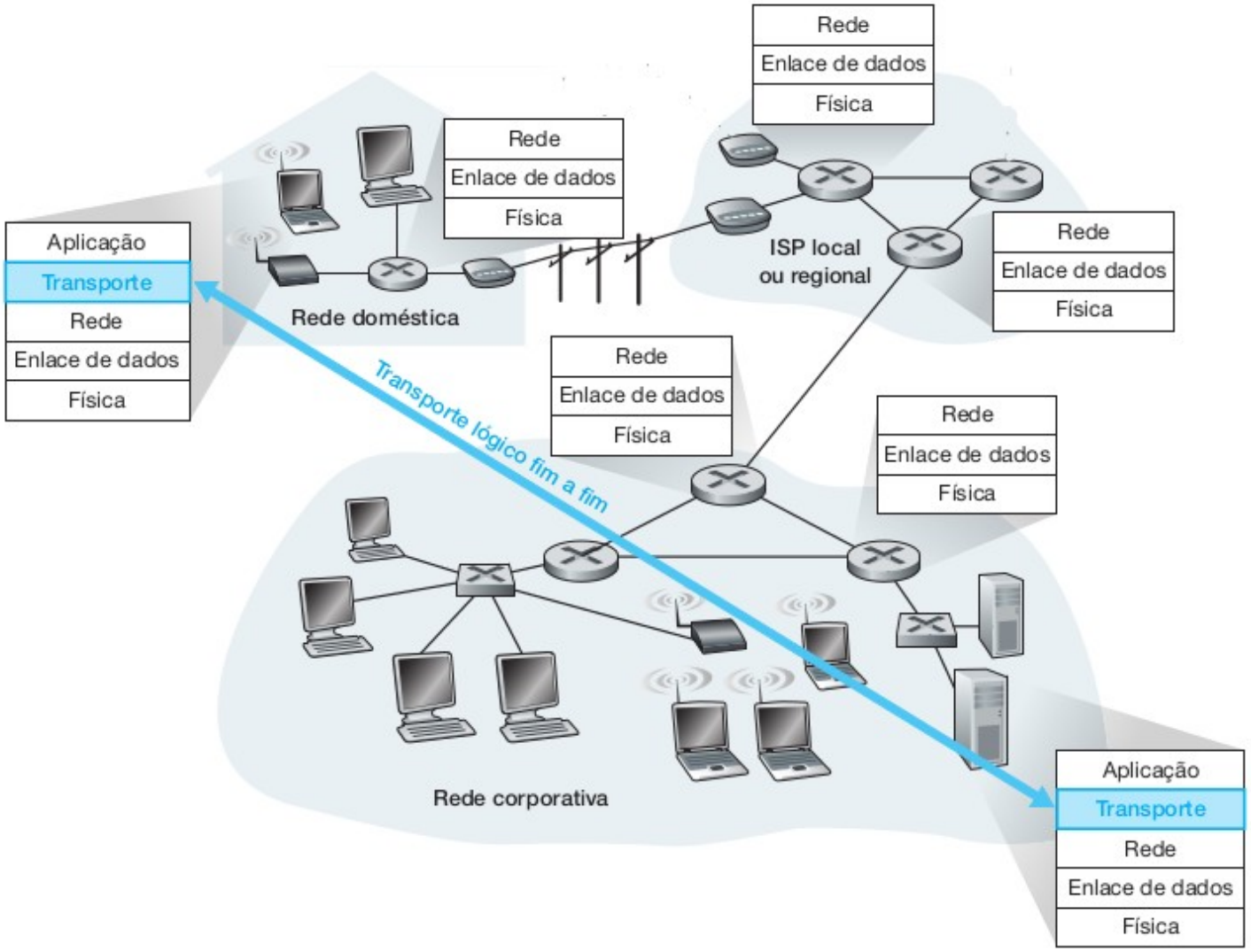
- "**camada de transporte**" .. fornece "comunicação lógica" entre processos de aplicação em execução em diferentes "hosts".
- "**comunicação lógica**" .. do ponto de vista da camada de aplicação, tudo se passa como se os "hosts" que executam os processos estivessem conectados diretamente ou sobre o mesmo sist. operacional.

- "**protocolos de transporte**" .. são implementados nos sistemas finais ou "hosts", mas não em roteadores ou núcleo da rede.

… 3.1 – Serviços da Camada de Transporte

- “**protocolos de transporte**” .. são implementados somente nos “hosts” ou “end systems” !!

- “**protocolos de transporte**” .. não são implementados nos elementos de comutação de mensagens (switches e roteadores)



… 3.1 – Serviços da Camada de Transporte

- “**camada de transporte**” .. converte as mensagens que recebe da aplicação remetente em segmentos na camada de transporte.
- .. mensagens da camada de aplicação são fragmentadas em pedaços menores para atender restrições das camadas subjacentes.
- .. adiciona-se um cabeçalho de camada de transporte a cada pedaço para criar o segmento de camada de transporte.

- .. vários protocolos de camada de transporte podem ser disponibiliza-dos para as aplicações na camada de aplicação.
- p.ex., Arquitetura TCP/IP disponibiliza o TCP e UDP.

3.1.1 Camada de Transporte e de Redes

- "**transporte**" .. protocolo de camada de transporte fornece comunicação lógica entre processos que rodam em "hosts" diferentes.
- "**rede**" .. protocolo de camada de rede fornece comunicação lógica entre "hosts" diferentes, ou seja, "hosts" propriamente ditos.

- "**serviços da camada de transporte**" .. são muitas vezes limitados pelo modelo de serviço do protocolo subjacente (camada rede).
- e.g., considere um protocolo de camada de rede que não dê garantias de largura de banda para segmentos de camada de transporte.
- "**resultado**" .. protocolo de camada de transporte não poderá dar essas mesmas garantias para protocolos da camada de aplicação.

… 3.1.1 Camada de Transporte e de Redes

- "**serviços específicos**" .. podem ser oferecidos por um protocolo de transporte mesmo quando o protocolo de rede subjacente não oferece o serviço correspondente na camada de rede.
- e.g., considere um protocolo de transporte que oferece serviço confiável de transferência de dados a uma aplicação mesmo quando o protocolo subjacente da rede não é confiável.
- … quando o protocolo de rede perde, embaralha ou duplica pacotes, cabe ao protocolo da camada de transporte perceber o embaralhamento ou duplicidade de segmentos e efetuar as correções.

3.1.2 Visão Geral da Camada de Transporte

- "**camada de transporte**" .. disponibiliza 02 protocolos de transporte distintos para a camada de aplicação.
- UDP (User Datagram Protocol) .. oferece à aplicação solicitante um serviço não confiável e não orientado por conexão.
- TCP (Transmission Control Protocol) .. oferece à aplicação solicitante um serviço confiável de entrega de dados e orientado por conexão.

- "**terminologia**" .. mensagem da camada de transporte é identificada como um "**segmento**" de camada de transporte.

… 3.1.2 Visão Geral da Camada de Transporte

- "**terminologia**" .. mensagem da camada de transporte é identificada como um "**segmento**" de camada de transporte.
- Obs.: .. importante mencionar que nas RFCs da Arquitetura Internet há referência para "segmento" se for TCP e "datagrama" se for UDP.

- "**camada de rede**" - contempla o IP (Internet Protocol) que fornece comunicação lógica entre "hosts" ou "end systems".
- "**modelo de serviço**" .. serviço de melhor esforço, o que significa que faz o "melhor possível" para levar pacotes entre "hosts" comunicantes, mas sem qualquer garantia de entrega de dados.

… 3.1.2 Visão Geral da Camada de Transporte

- UDP / TCP .. tem por responsabilidade ampliar o serviço de entrega de pacotes entre 02 sistemas finais para um serviço de entrega entre 02 processos que rodam nos sistemas finais.
- … ampliação da entrega “host” a “host” para a entrega “processo” a “processo” é denominada “multiplexação” / “demultiplexação”.

- UDP .. serviço não confiável, ou seja, não garante que os dados enviados por um remetente cheguem ao processo destinatário.
- p.ex., .. quando chegam e se chegam (dados sem erros !?) ?!

… 3.1.2 Visão Geral da Camada de Transporte

- TCP .. oferece serviço de transferência confiável de dados, todos os dados enviados são entregues na ordem correta.
- … controle de fluxo, nros. de sequência, reconhecimentos e temporizadores são assegurados pelo protocolo e, assim, há garantia de entrega dos dados ao processo destinatário.

- "**controle de congestionamento**" .. não se trata de serviço para a aplicação solicitante, mas sim de um serviço dirigido à Internet.
- "**termos genéricos**" .. controle de congestionamento do TCP evita que qualquer outra conexão TCP "inunde" ou "abarrote" os enlaces e roteadores entre os "hosts" comunicantes.

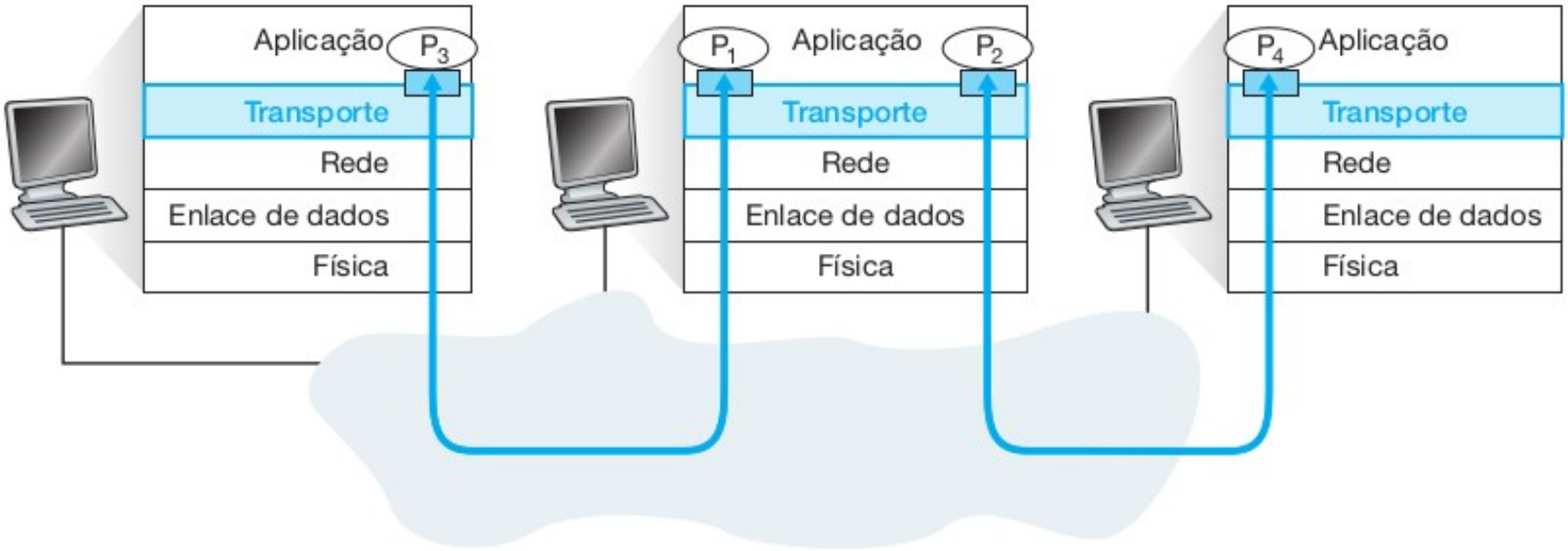
3.2 Multiplexação e Demultiplexação

- "**multiplexação** / **demultiplexação**" .. ampliação do serviço de entrega "host" a "host" provido pela camada de rede para um serviço de entrega "processo" a "processo" para a camada de aplicação.

- e.g., considere o "download" de páginas Web enquanto se executa 01 sessão FTP e 02 sessões Telnet, logo, são necessários 04 processos de aplicação em execução — 02 Telnet, 01 FTP e 01 HTTP.
- … quando a camada de transporte no computador receber dados da camada de rede logo abaixo, precisará direcionar os dados recebidos a um desses 04 processos .. Telnet, FTP ou HTTP.

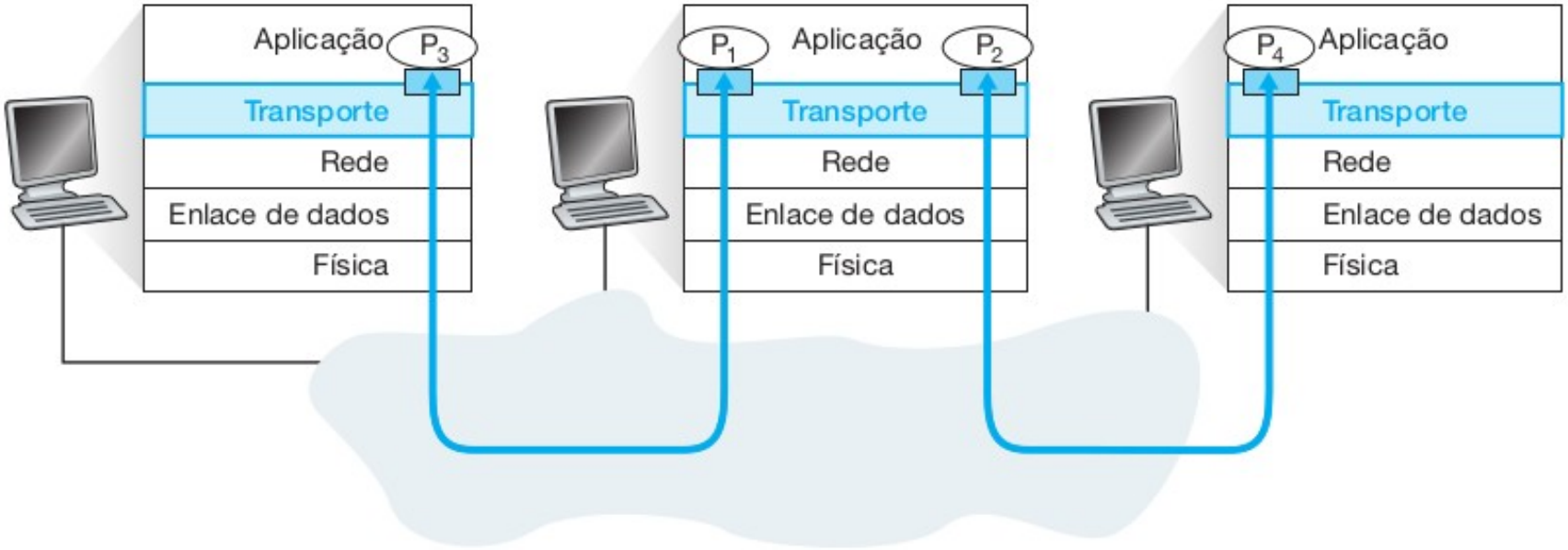
3.2 Multiplexação e Demultiplexação

- e.g., considere o “download” de páginas Web enquanto se executa 01 sessão FTP e 02 sessões Telnet, logo, são necessários 04 processos de aplicação em execução — 02 Telnet, 01 FTP e 01 HTTP.
- … quando a camada de transporte receber dados da camada de rede, precisará direcionar os dados recebidos a um desses 04 processos.



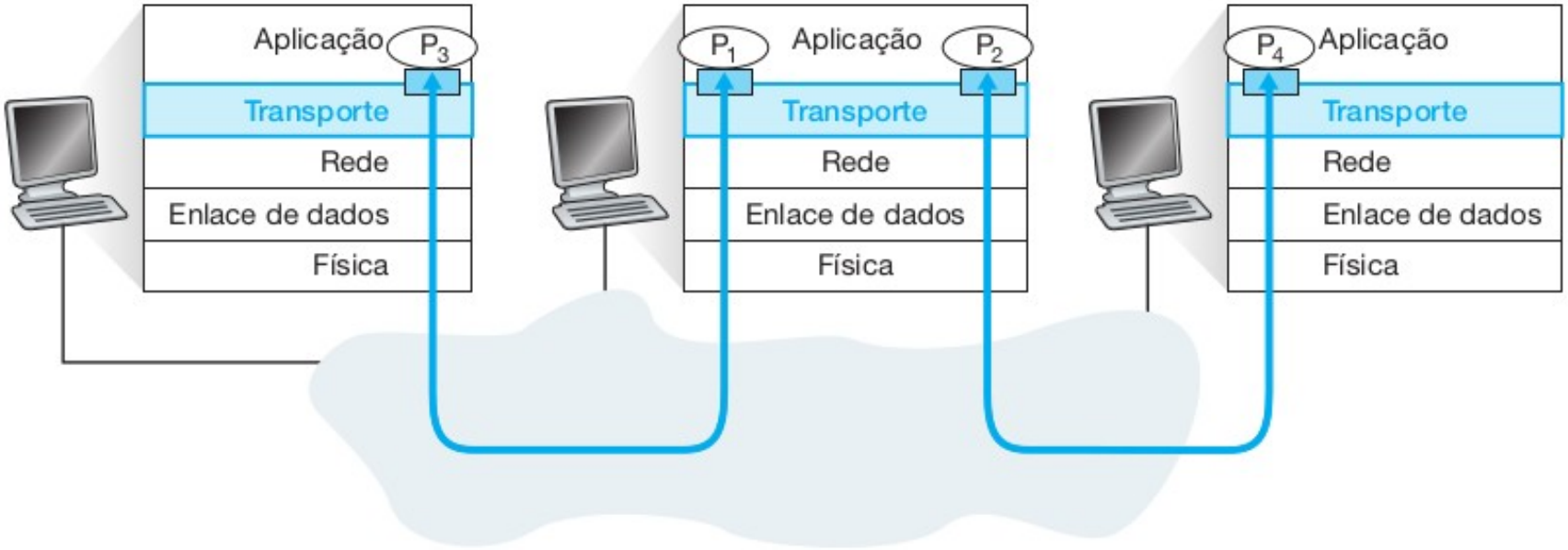
… 3.2 Multiplexação e Demultiplexação

- … aplicação pode conter um ou mais “sockets”, portas pelas quais dados passam da rede para o processo e do processo para a rede.
- … como mostra a figura, a camada de transporte do “host” destinatário na verdade não entrega dados diretamente a um processo, mas a um “socket” intermediário (identificador exclusivo).



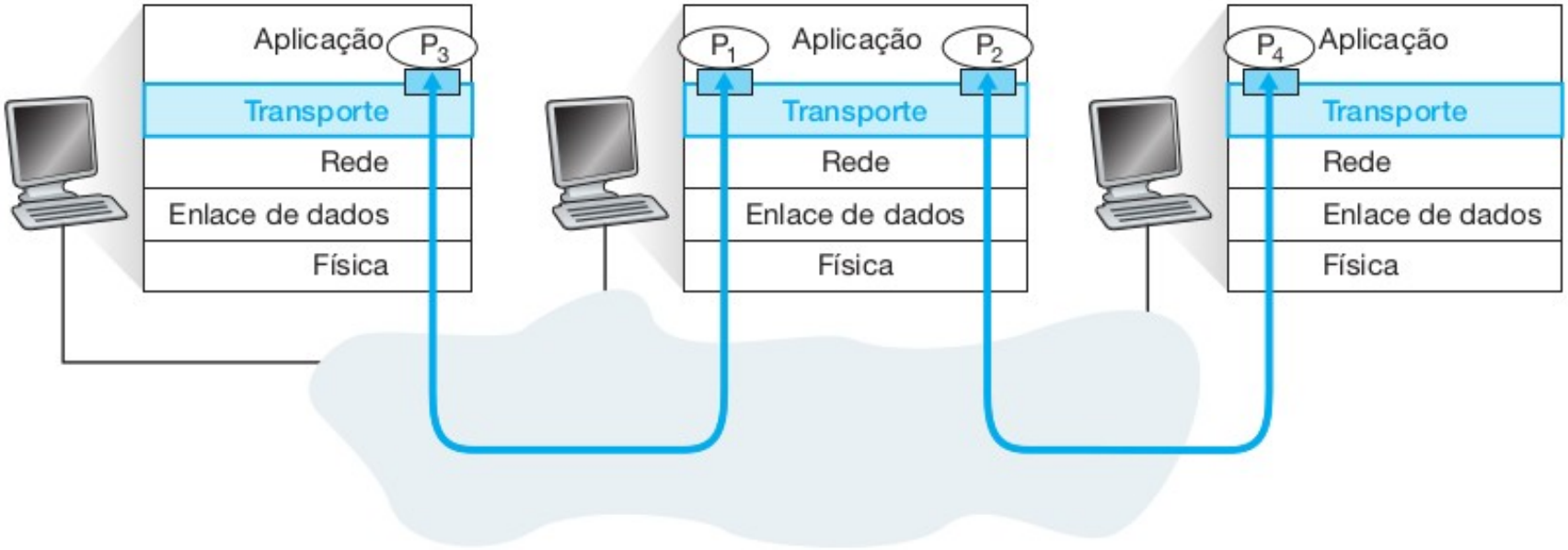
… 3.2 Multiplexação e Demultiplexação

- “**como direcionar ao socket apropriado ?**” .. cada segmento da camada de transporte tem um conjunto de campos para tal finalidade.
- .. no receptor, a camada de transporte examina esses campos para identificar a porta receptora e direcionar o segmento a esse socket.



… 3.2 Multiplexação e Demultiplexação

- "**multiplexação**" .. trabalho de reunir no "host" origem partes de dados de diferentes "sockets" e encapsulá-los com informações de cabeçalho para criar segmentos e repassá-los para a camada de rede.
- "**demultiplexação**" .. tarefa de entregar os dados contidos em um segmento da camada de transporte ao socket correto (destinatário).



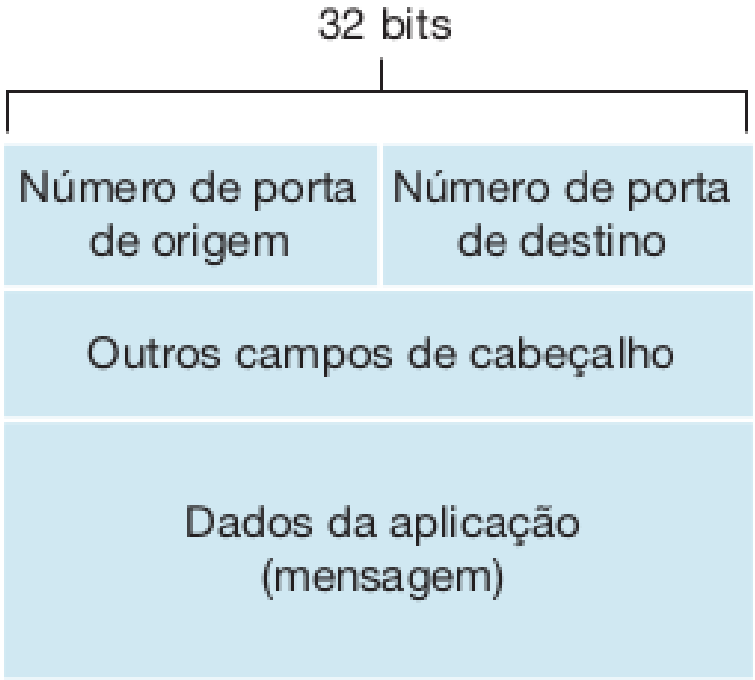
… 3.2 Multiplexação e Demultiplexação

- Como realizar a Multiplexação ?

  (1) necessário que as portas tenham identificadores exclusivos;

  (2) que cada segmento tenha campos especiais que indiquem a porta para a qual o segmento deve ser entregue.

- "**campos especiais**" .. campo de nro. de porta de origem e o campo de nro. de porta de destino.



… 3.2 Multiplexação e Demultiplexação

- "**nro. de porta**" .. número de 16 bits na faixa de 0 a 65535.
- … nros. de porta entre 0 e 1023 são denominados números de porta bem conhecidos e são restritos, o que significa que estão reservados para utilização por protocolos de aplicação bem conhecidos.
- p.ex., HTTP (nro. de porta = 80) e FTP (nro de porta = 21).

- [RFC 1700] .. contempla a lista dos nros. de portas bem conhecidos e atualizada pelo IANA .. http://www.iana.org [RFC 3232].
- IANA – Internet Assigned Number Authority

… 3.2 Multiplexação e Demultiplexação

- Como realizar a Demultiplexação ?

  (1) cada socket do “host” pode receber um nro. designado, naturalmente que por intermédio do sistema operacional de rede.

  (2) quando um segmento chega ao “host” destino, a camada de transporte examina seu nro. de porta de destino e direciona o segmento ao socket correspondente.

  (3) na sequência, os dados do segmento passam pela porta e entram no processo associado a esta porta.

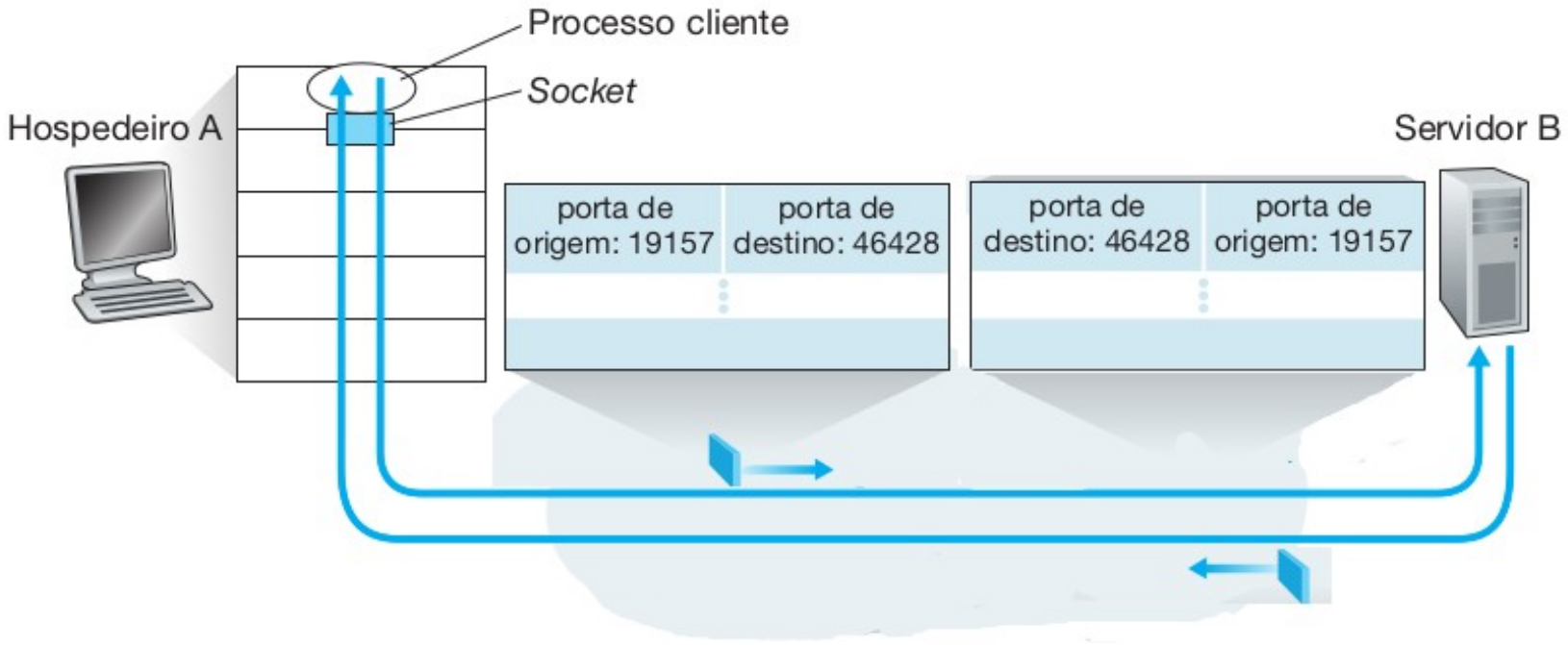
- “**observação**” .. “sistema operacional de rede” é o mesmo que “sistema operacional” + “pilha de comunicação”.

… 3.2 Multiplexação e Demultiplexação

- "**multiplexação / demultiplexação UDP**" .. programa em "python" que roda em um "host" pode criar uma porta UDP com o comando ..

  clientSocket = socket(socket.AF_INET, socket.SOCK_DGRAM)

- … quando um socket UDP é criado dessa maneira, a camada de transporte automaticamente designa um nro. de porta ao socket.

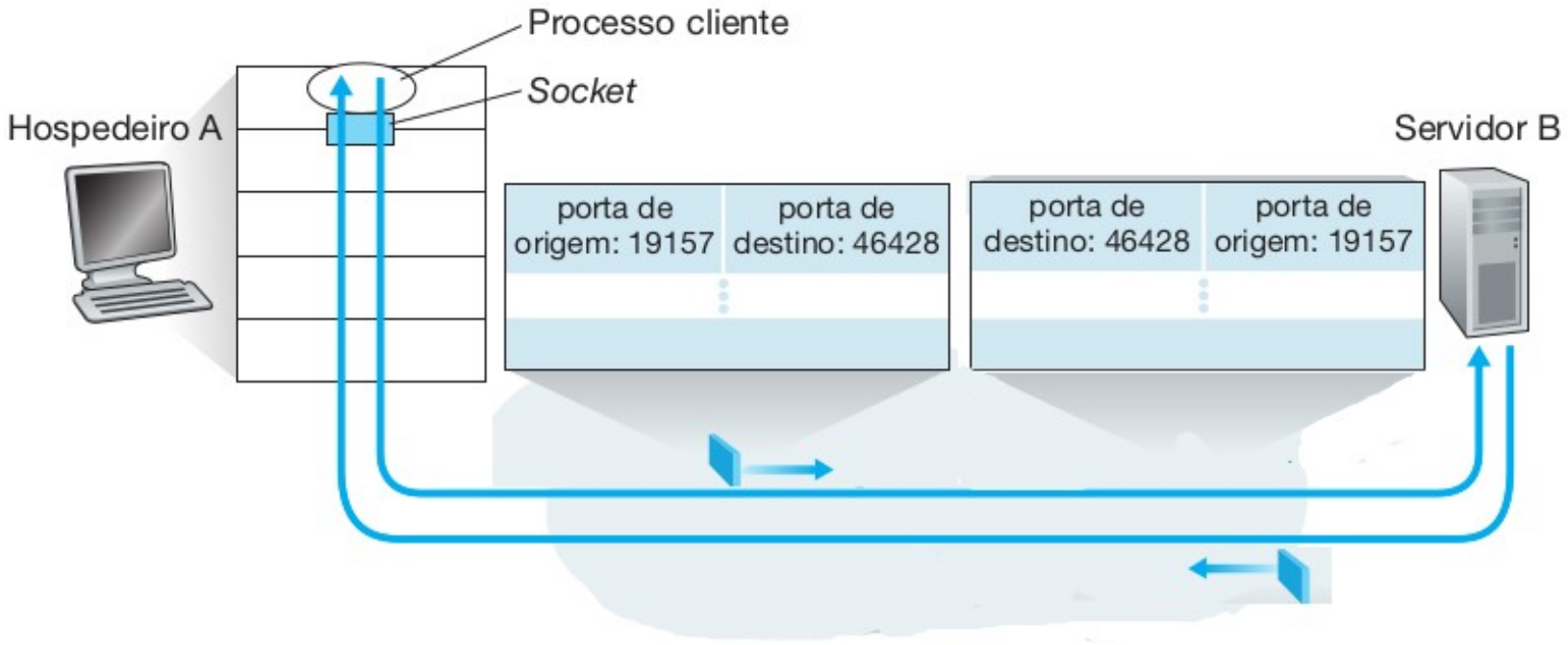


… 3.2 Multiplexação e Demultiplexação

- ... camada de transporte designa um nro. de porta na faixa de 1024 a 65535 que não esteja sendo usado por qualquer outro socket no “host”.

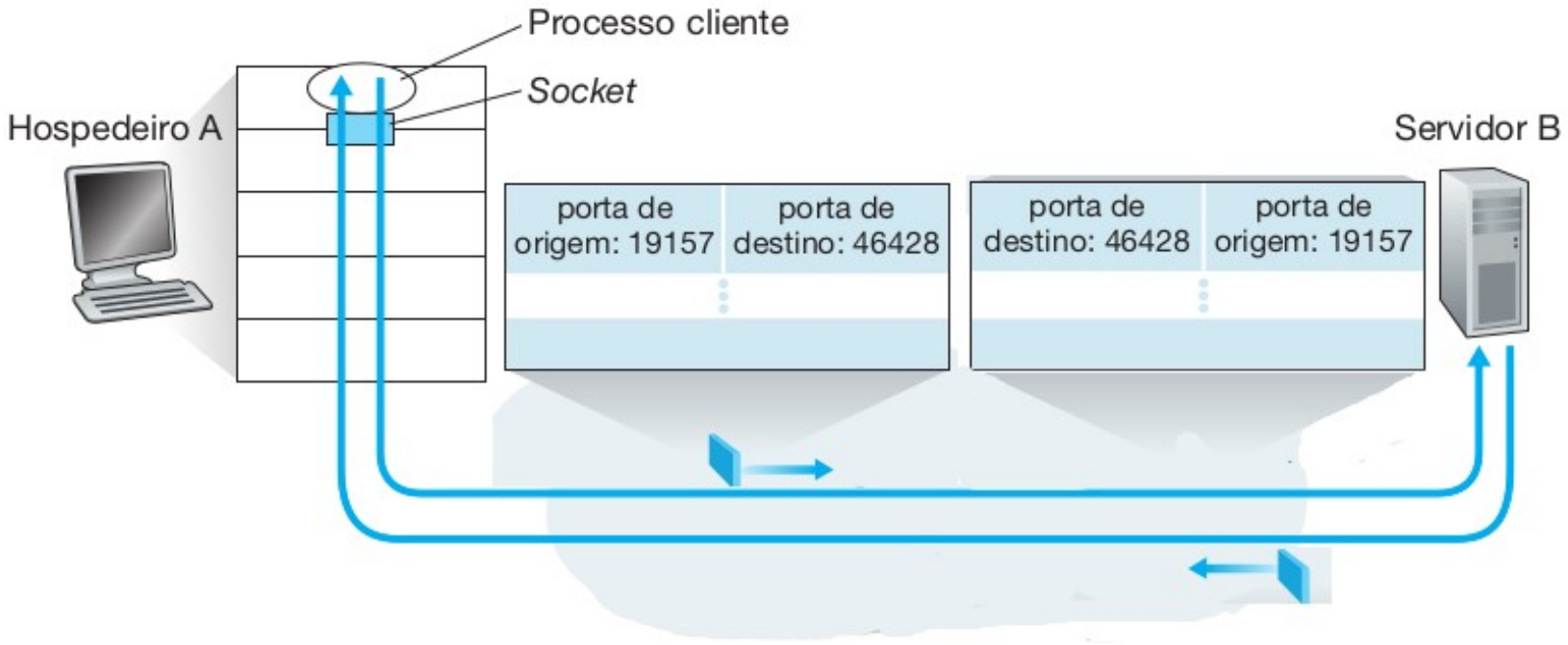
  clientSocket.bind((”, 19157))

- … cliente em geral permite que a camada de transporte designe o nro. de porta de modo automático (de forma transparente).



… 3.2 Multiplexação e Demultiplexação

- … cliente em geral permite que a camada de transporte designe o nro. de porta de modo automático (de forma transparente).
- … já no lado servidor da aplicação, a designação de nro. de porta se dá por nro. de porta específico em razão da necessidade de se identificar o servidor na camada de transporte.



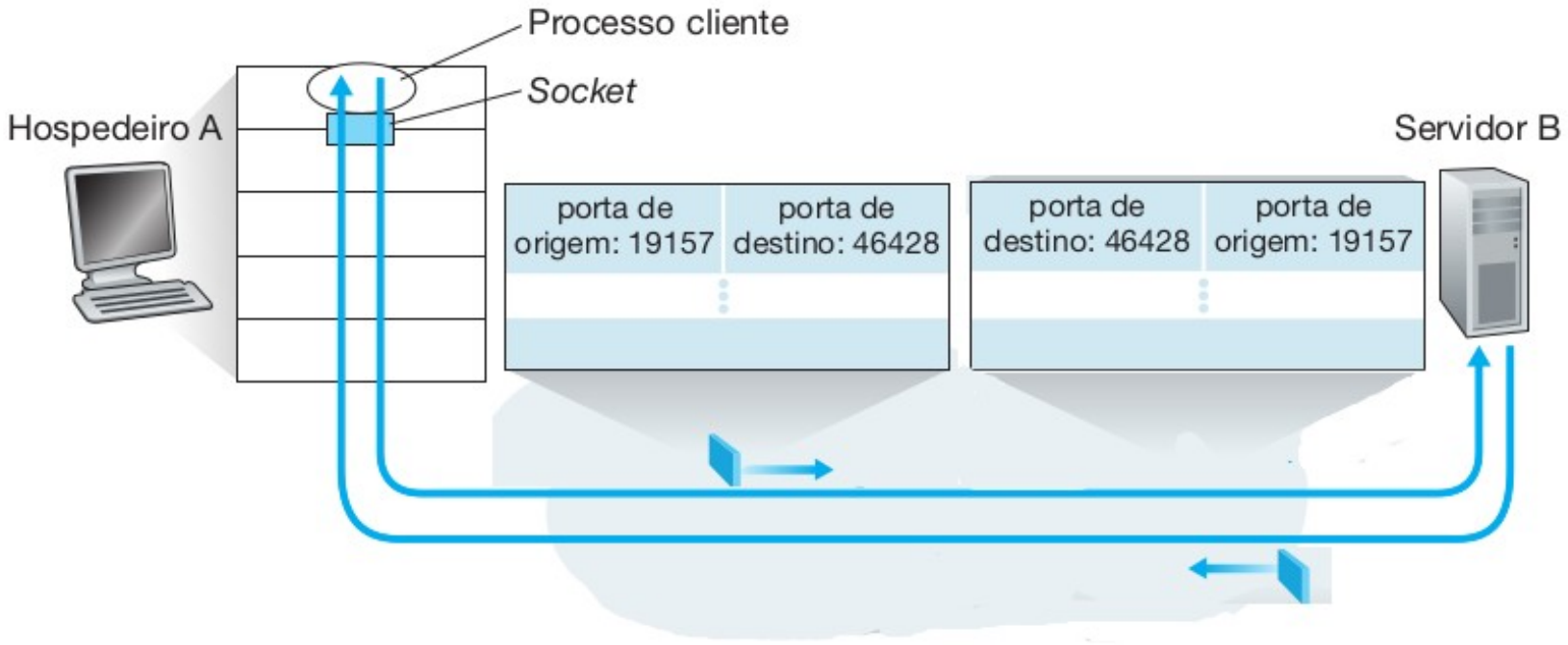
… 3.2 Multiplexação e Demultiplexação

- “**observação**” .. “socket” UDP é identificado por uma tupla de 02 elementos, ou seja, um endereço IP e um nro. de porta de destino.
- e.g., .. o que acontece se 02 segmentos UDP tiverem endereços IP de origem e/ou nros. de porta de origem diferentes, porém o mesmo endereço IP de destino e o mesmo nro. de porta de destino ?
- “**resposta**” ... serão direcionados ao mesmo processo de destino por meio do mesmo “socket” de destino.

- “**dúvida**” .. qual é a finalidade do nro. da porta de origem ?

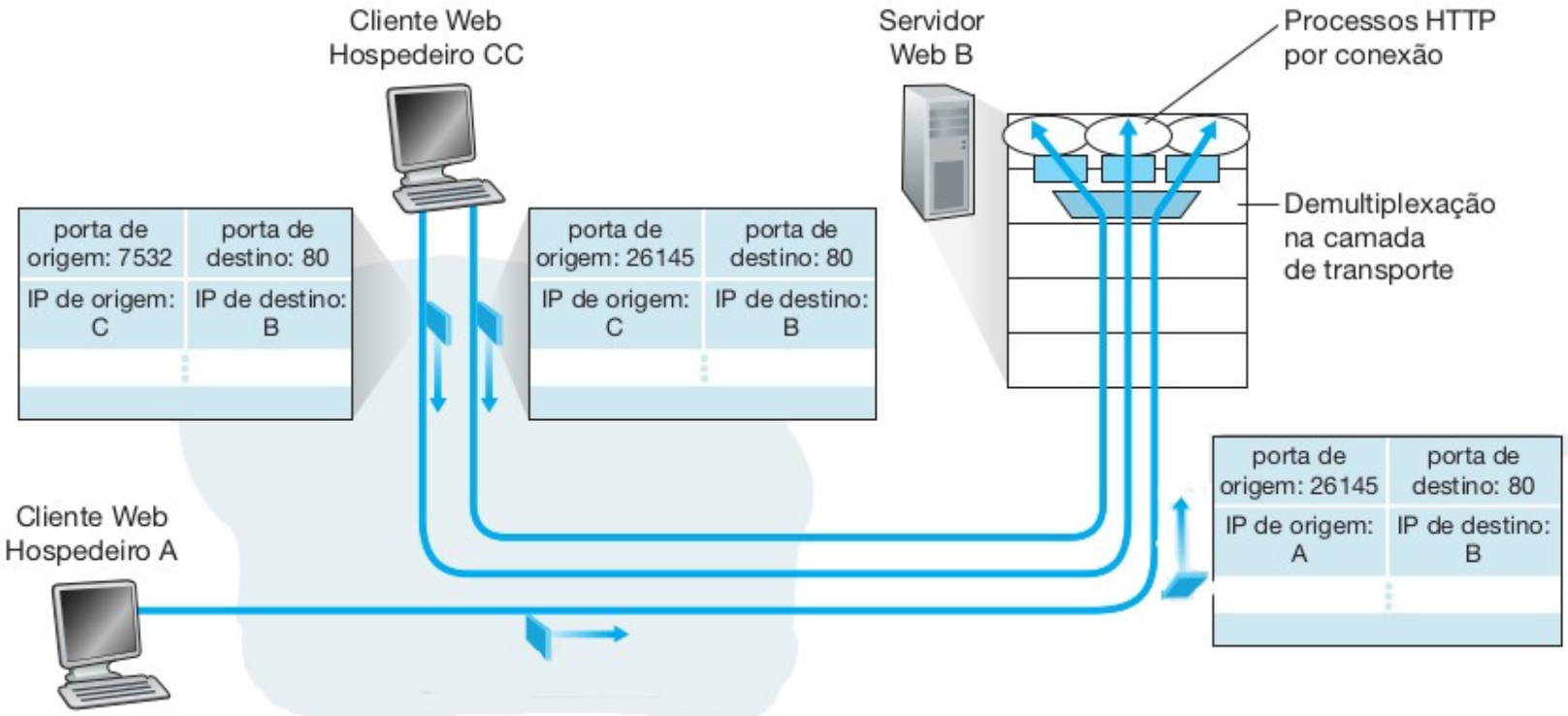
… 3.2 Multiplexação e Demultiplexação

- .. como mostra a figura, no segmento de A >> B, o nro. da porta de origem serve como parte de um “endereço de retorno”, ou seja, quando B quer enviar um segmento de volta para A.
- … endereço de retorno completo é o endereço IP e o nro. de porta de origem de A que pode ser extraído do segmento A > B.



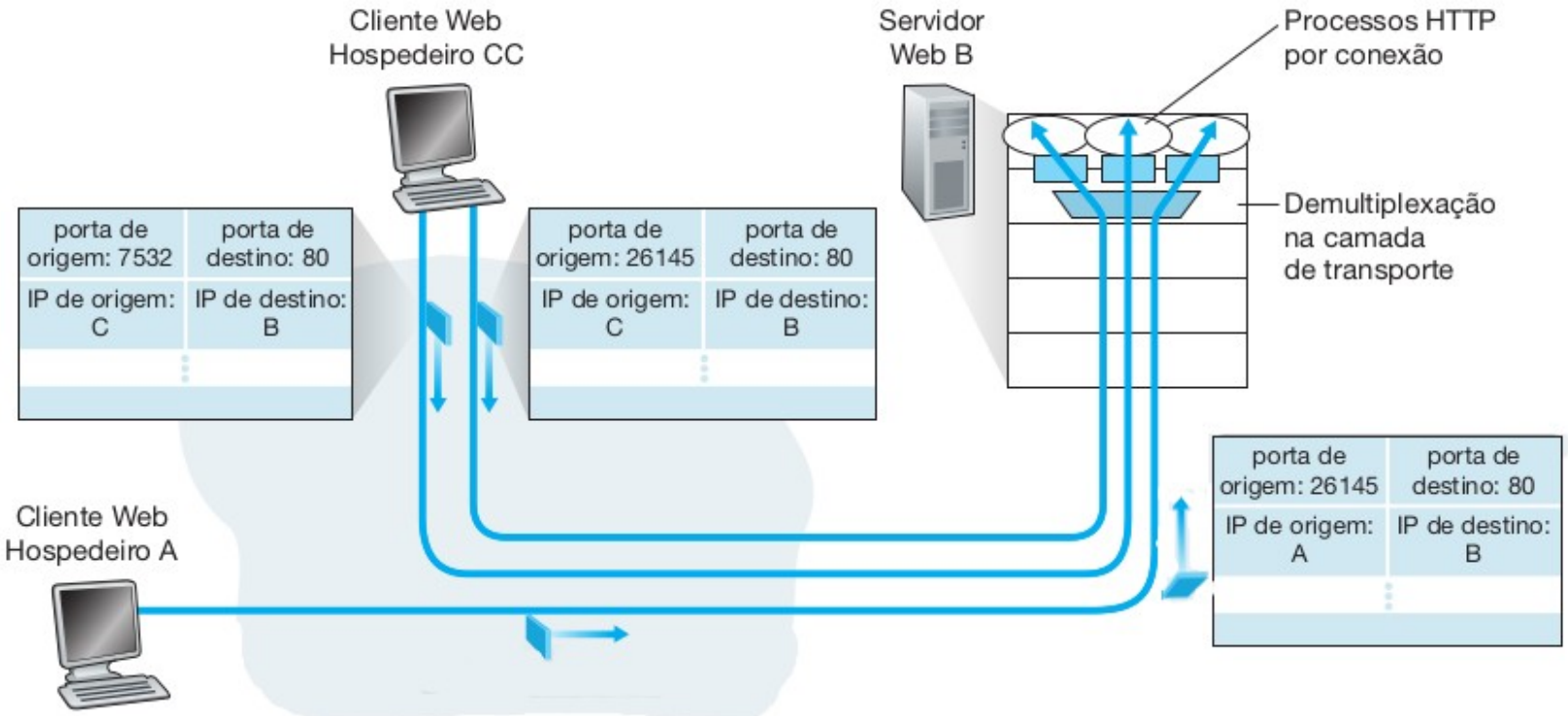
… 3.2 Multiplexação e Demultiplexação

- "**multiplexação / demultiplexação TCP**" .. socket TCP é identificado por uma tupla de 04 elementos, ou seja, endereço IP de origem, nro. da porta de origem, endereço IP de destino e nro. da porta de destino.



… 3.2 Multiplexação e Demultiplexação

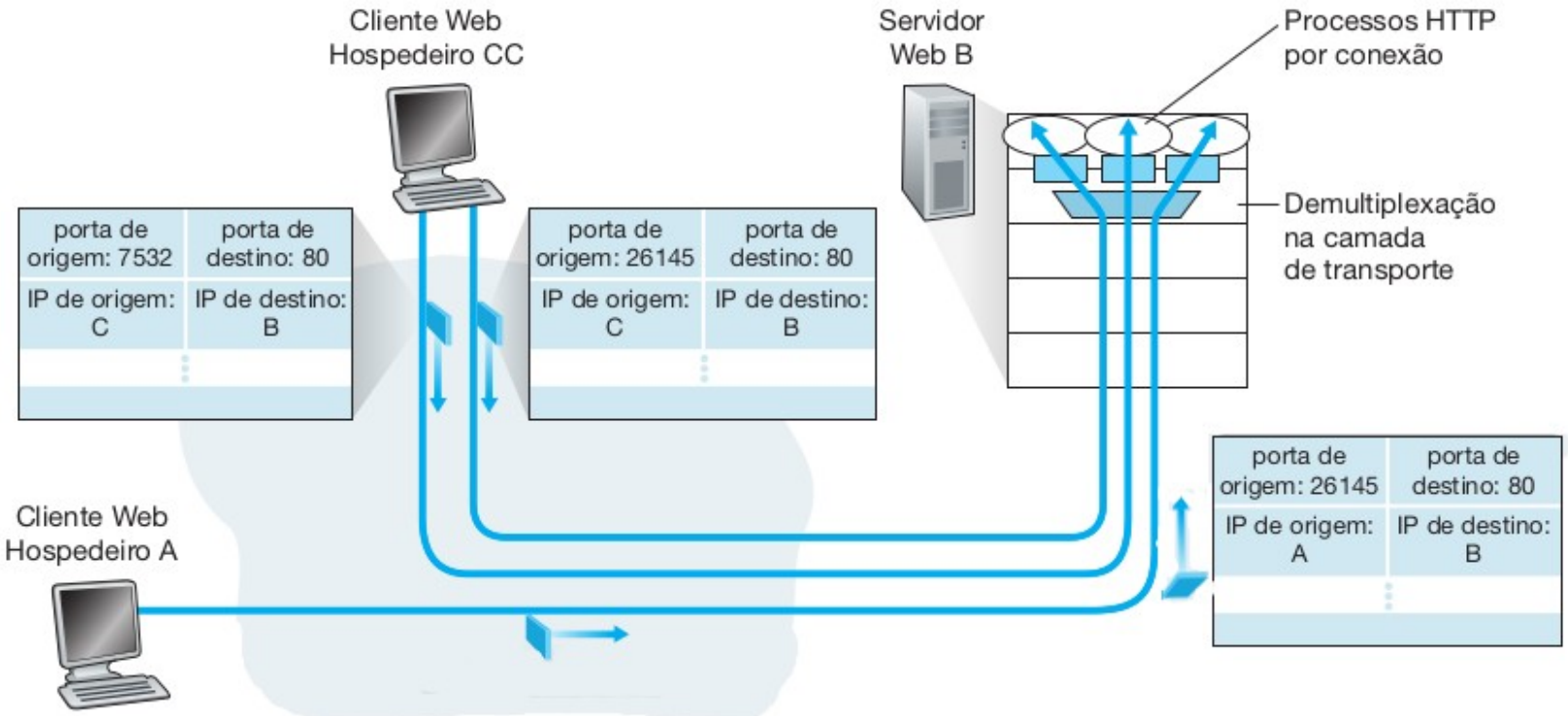
- e.g., considere novamente o exemplo de programação Cliente / Servidor TCP em “python” apresentado na Seção 2.7.2.
- … aplicação servidor TCP tem um “socket de entrada” que espera requisições de estabelecimento de conexão vindas de clientes TCP.



… 3.2 Multiplexação e Demultiplexação

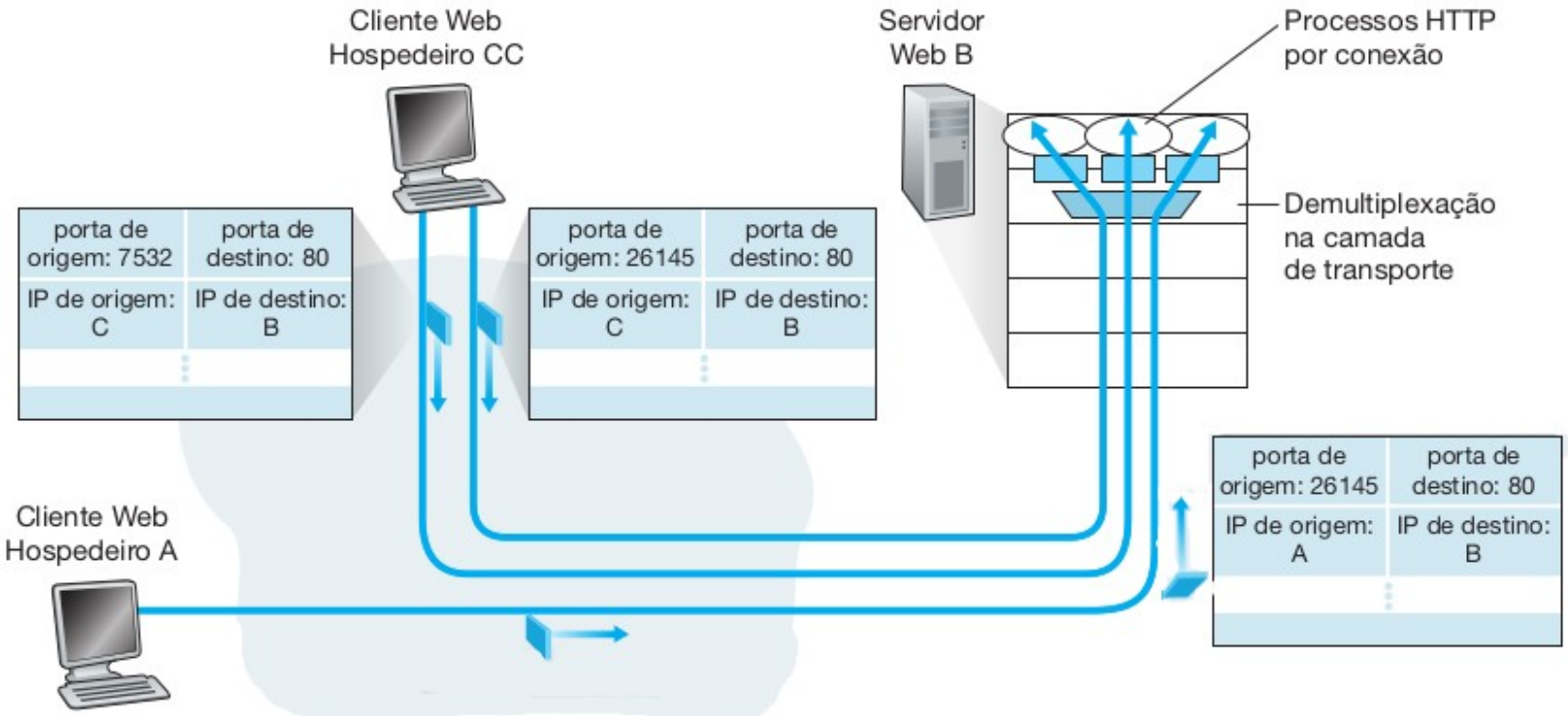
- … cliente TCP cria um socket e envia um segmento de requisição de estabelecimento de conexão com os comandos:

clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,12000))



… 3.2 Multiplexação e Demultiplexação

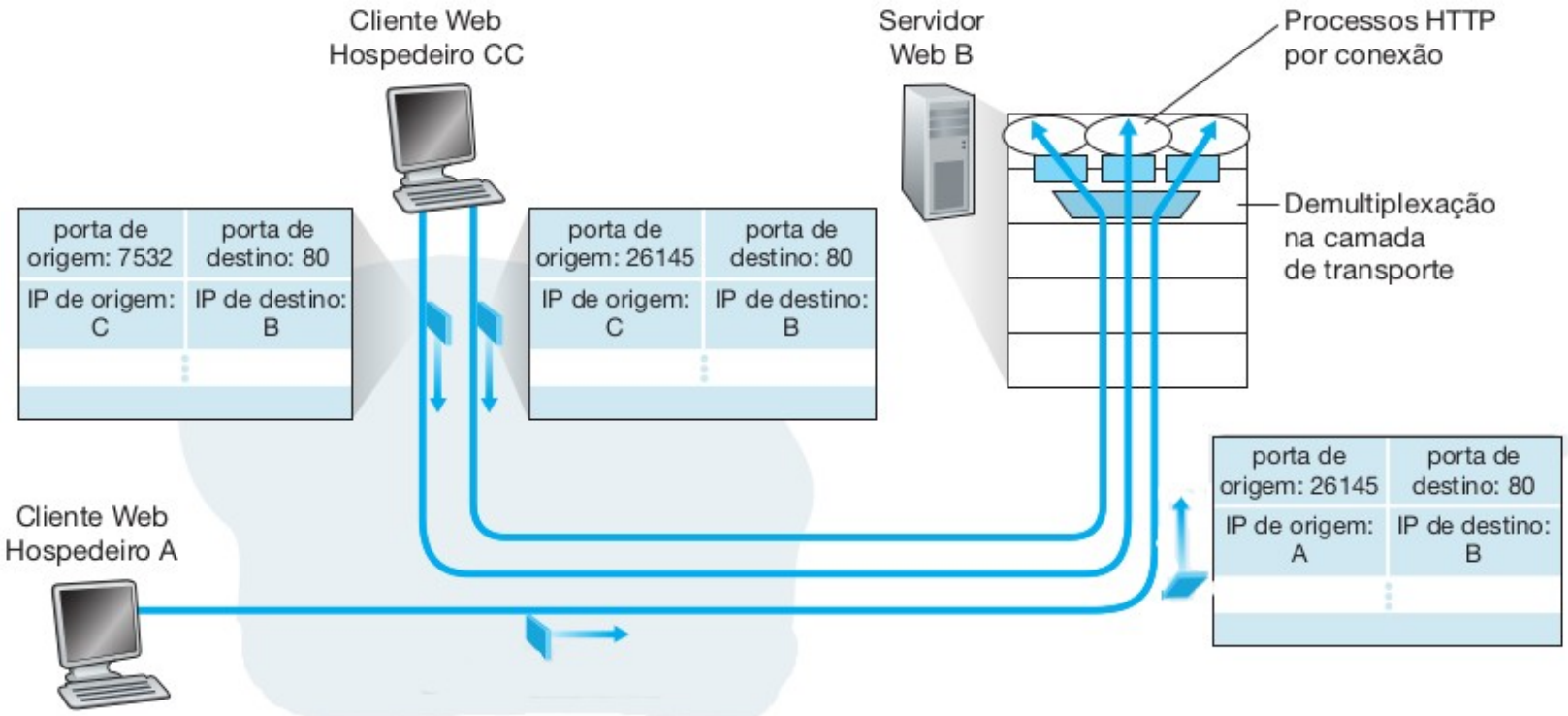
- … requisição de estabelecimento de conexão nada mais é do que um segmento TCP com nro. de porta de destino 12000 e um bit especial de estabelecimento de conexão marcado no cabeçalho TCP.



… 3.2 Multiplexação e Demultiplexação

- … quando o sist. oper. de rede do servidor que executa o processo servidor recebe o segmento de requisição de conexão e cuja porta de destino é 12000, o mesmo localiza o processo que está a espera.

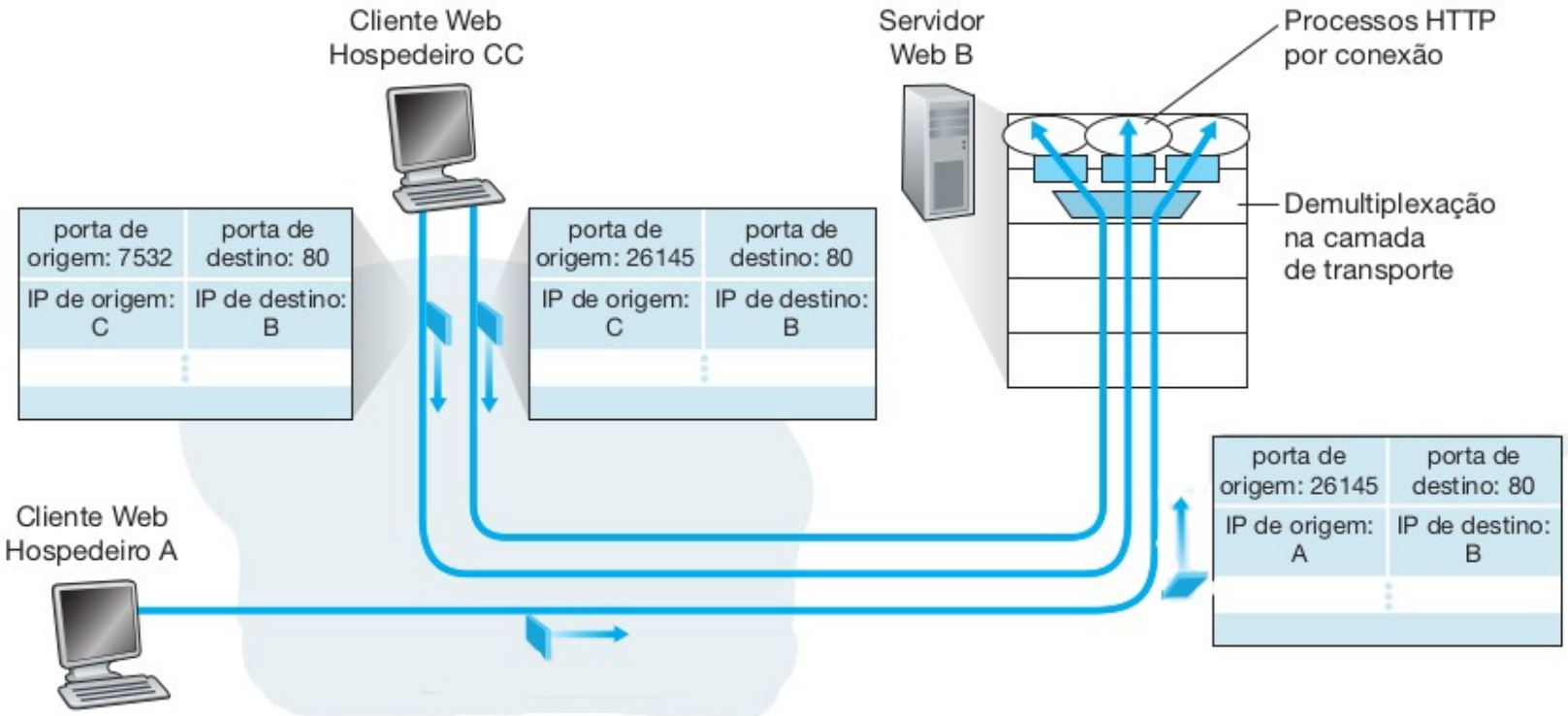
  connectionSocket, addr = serverSocket.accept()



… 3.2 Multiplexação e Demultiplexação

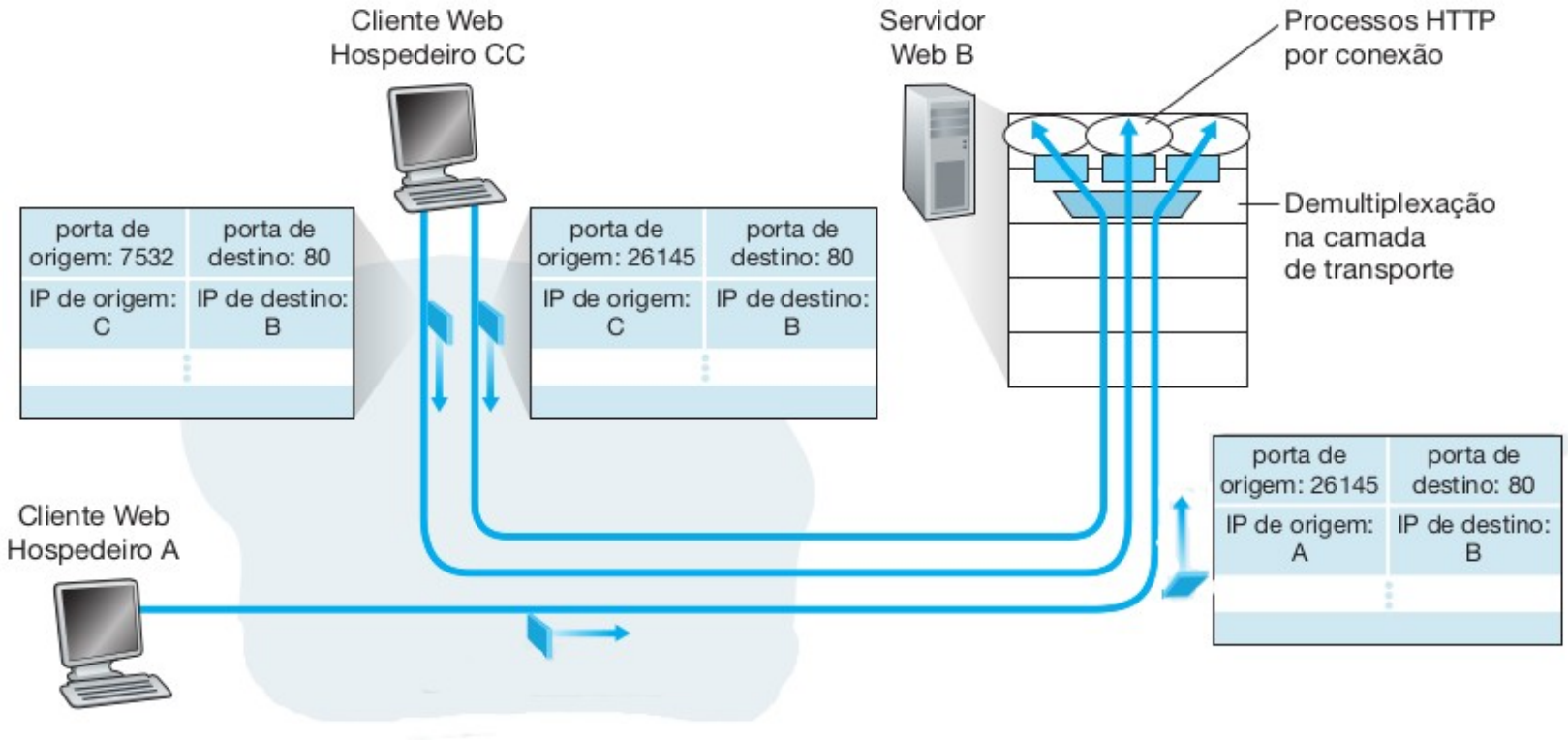
- … camada de transporte no servidor também percebe os 04 campos no segmento de requisição de conexão:

  (1) nro. da porta de origem, (2) endereço IP do “host” de origem, (3) nro. da porta de destino e (4) seu próprio endereço IP.



… 3.2 Multiplexação e Demultiplexação

- .. “socket” de conexão recém-criado é identificado pelos 04 valores e, assim, todos os segmentos subsequentes que chegarem com a mesma tupla serão demultiplexados para esse “socket”.
- .. com a conexão TCP ativa, Cliente e Servidor podem enviar dados um para o outro (conexão full-duplex).



3.3 - Transporte Não Orientado a Conexão

- "**RFC 768 (UDP)**" – … faz tão pouco quanto um protocolo de camada de transporte pode fazer, ou seja, contempla multiplexação / demultiplexação e verificação de erros simples.

- UDP não é orientado a conexão, ou seja, não há apresentação entre as partes antes do envio de segmentos entre as partes.
- "sem handshaking" entre remetente e destinatário UDP.
- cada segmento UDP é tratado de forma independente dos outros.

… 3.3 - Transporte Não Orientado a Conexão

- "**serviço**" .. por se tratar de um serviço de "melhor esforço", segmentos UDP podem ser perdidos ou entregues fora da ordem, logo, os segmentos são entregues na ordem em que chegam.
- .. as mensagens da aplicação recebem os nros. da porta origem e de destino para o serviço de multiplexação / demultiplexação.
- .. na sequência 02 outros pequenos campos são adicionados para serem repassados à camada de rede, que encapsula em um pacote IP.

- "**dúvida**" .. por que construir uma aplicação sobre UDP em vez de sobre o TCP, ou seja, o TCP não é preferível ante ao UDP ?

… 3.3 - Transporte Não Orientado a Conexão

- Razões para que aplicações se adaptem ao UDP, ou seja, preferência pelo UDP ante ao “serviço de entrega confiável” do TCP.

- “**controle no nível de aplicação**” .. não há restrição no remetente quanto ao envio de segmentos ou datagramas, assim tão logo o UDP monte os segmentos, ele os repassa à camada de rede.
- “**sem conexão**” - envia msgs. sem nenhuma formalidade dos pares, ou seja, não introduz atraso face ao estabelecimento de uma conexão.

… 3.3 - Transporte Não Orientado a Conexão

- Razões para que aplicações se adaptem ao UDP, ou seja, preferência pelo UDP ante ao “serviço de entrega confiável” do TCP.

- “**não há estado de conexão**” - não há reserva de “buffers” de envio e recebimento; de parâmetros de controle de congestionamento; ou de parâmetros de sequênciamento e reconhecimento.
- “**sobrecarga**” - cabeçalho UDP é muito pequeno (08 bytes), senão o menor cabeçalho, é um dos menores cabeçalhos em protocolos.

… 3.3 - Transporte Não Orientado a Conexão

- Aplicações que “utilizam” o Protocolo UDP:

- RIP (Route Information Protocol) .. atualizações são enviadas periodicamente aos nós vizinhos, assim, atualizações perdidas ou antigas são naturalmente substituídas por atualizações mais recentes.
- SNMP (Simple Network Management Protocol) .. aplicações de gerenciamento de rede funcionam quando a rede está em estado sobrecarregado, situação em que a transf. confiável é difícil.
- DNS (Domain Name System) .. serviço de nomes global evita atrasos ao se estabelecer uma conexão e, assim, lança-se mão de serviço sem conexão entre o cliente e o servidor.

… 3.3 - Transporte Não Orientado a Conexão

- Aplicações que “utilizam” o Protocolo UDP:

- “VoIP – Videoconferência – Aplicações Multimídia” .. por se tratar de aplicações que toleram uma pequena perda de pacotes, a transferência confiável de dados não é absolutamente crítica para o sucesso.
- … adicionalmente algumas destas aplicações reagem mal ao controle de congestionamento do TCP, o que permite que sejam executadas sobre UDP e apresentem um comportamento melhor.

… 3.3 - Transporte Não Orientado a Conexão

- "**tabela**" .. mostra algumas aplicações populares da Internet, bem como os protocolos da camada de transporte que são utilizados.
- TCP predomina nas aplicações de multimídia, voz sobre IP, videoconferência em tempo real e aplicações de áudio e vídeo sob-demanda.

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

… 3.3 - Transporte Não Orientado a Conexão

- .. quando as taxas de perdas de pacote são baixas, em conjunto com o bloqueio de tráfego UDP por razões de segurança, o TCP tem se tornado um protocolo mais atrativo para transporte de mídia.

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

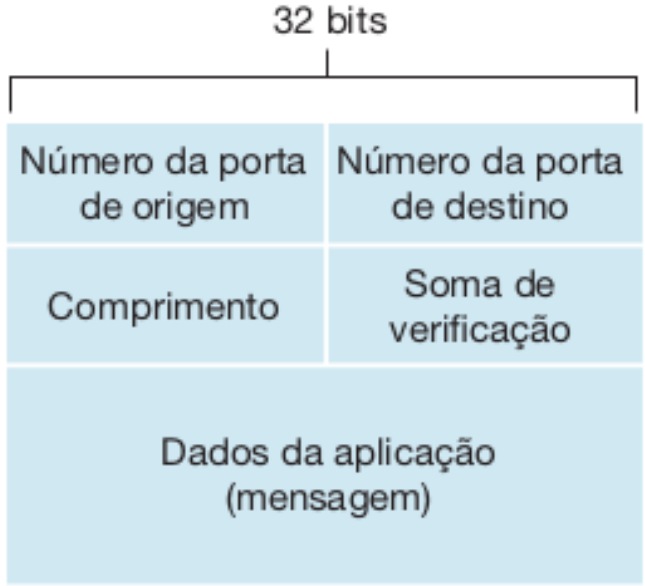
3.3.1 – Segmento ou Datagrama UDP

- "**RFC 768 (UDP)**" .. define a estrutura do segmento UDP que acomoda os dados da aplicação no campo "payload" do segmento UDP.

- … contempla apenas 04 campos, cada um consistindo de 02 bytes, sendo que os nros. de portas permitem que o "host" repasse para o processo correto da aplicaçaão os dados recebidos.

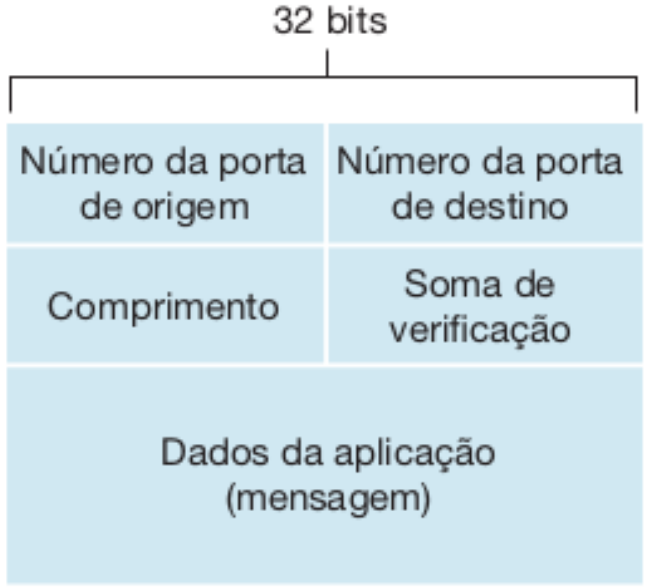
… 3.3.1 – Segmento ou Datagrama UDP

- .. nros. de portas permitem que o “host” destinatário passe os dados da aplicação ao processo correto que está funcionando no “host” destinatário, isto é, realize a função de demultiplexação.



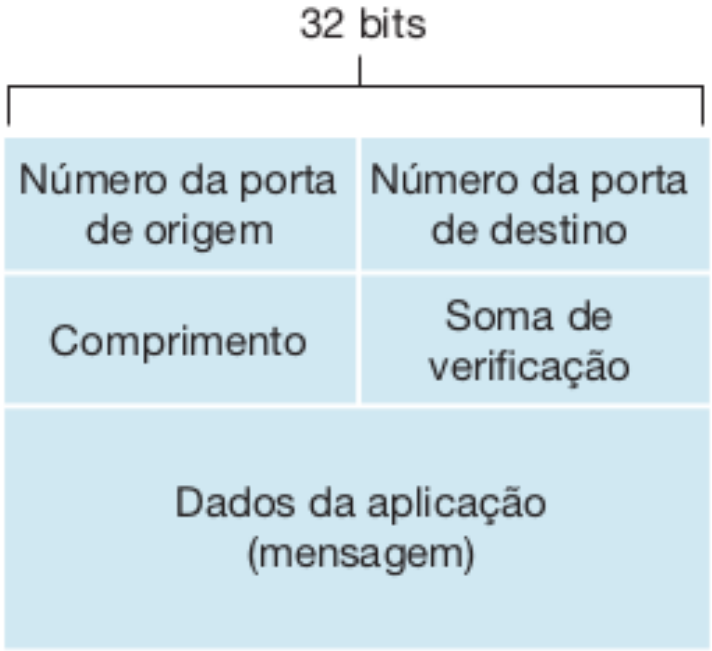
… 3.3.1 – Segmento ou Datagrama UDP

- "**comprimento**" .. comprimento do segmento em bytes, ou seja, cabeçalho e dados são incluídos no comprimento do segmento.
- … comprimento explícito se faz necessário porque o tamanho do campo de dados pode ser diferente de um segmento UDP para o outro.
- … campo de comprimento especifica o comprimento do segmento UDP, incluindo o cabeçalho, em bytes.



3.3.1 – Segmento ou Datagrama UDP

- "**soma de verificação**" .. usada pelo "host" receptor para verificar se há erros passíveis de serem identificados no segmento (destinatário).
- … soma de verificação também é calculada para alguns dos campos no cabeçalho IP, além do segmento UDP.



3.3.2 – Soma de Verificação UDP

- "**soma de verificação**" - detecta erros de bits no segmento UDP, p.ex., ruídos nos enlaces ou alterações quando no roteador.
- "complemento 1" da soma de todas as palavras de 16 bits do segmento levando-se em conta o "vai um" em toda a soma.
- … remetente calcula o complemento de 1 da soma de todas as palavras de 16 bits do segmento levando em conta o "vai um" nas somas.
- "**resultado**" = campo de soma de verificação do segmento.

- e.g., como exemplo simples do cálculo da soma de verificação, considere 03 palavras de 16 bits cada, quais sejam: 0110.0110.0110.0000; 0101.0101.0101.0101 e 1000.1111.0000.1100

… 3.3.2 – Soma de Verificação UDP

- e.g., como exemplo simples do cálculo da soma de verificação, considere 03 palavras de 16 bits cada, quais sejam: 0110.0110.0110.0000; 0101.0101.0101.0101 e 1000.1111.0000.1100.

  0110.0110.0110.0000 >> 1a Palavra de 16 bits

  0101.0101.0101.0101 >> 2a Palavra de 16 bits

  1011.1011.1011.0101 >> Soma das 02 primeiras Palavras de 16 bits.

… 3.3.2 – Soma de Verificação UDP

- e.g., como exemplo simples do cálculo da soma de verificação, considere 03 palavras de 16 bits cada, quais sejam: 0110.0110.0110.0000; 0101.0101.0101.0101 e 1000.1111.0000.1100.

  1011.1011.1011.0101 >> 1a Palavra + 2a Palavra

  1000.1111.0000.1100 >> 3a Palavra

  0100.1010.1100.0010 >> resultado

- "**observação**" .. na última adição tem o "vai um" no bit mais significativo que foi somado ao bit menos significativo.

… 3.3.2 – Soma de Verificação UDP

- e.g., como exemplo simples do cálculo da soma de verificação, considere 03 palavras de 16 bits cada, quais sejam: 0110.0110.0110.0000; 0101.0101.0101.0101 e 1000.1111.0000.1100.

  0100.1010.1100.0010 >> Soma das 03 Palavras de 16 bits.

- “complemento de 1” .. obtido pela conversão de todos os 0 em 1 e de todos os 1 em 0, ou seja, complemento de 1 da soma de 0100.1010.1100.0010 >> 1011.0101.0011.1101.
- “**soma verificação**” .. 1011010100111101 (16 bits)

… 3.3.2 – Soma de Verificação UDP

- "**dúvida**" .. por que o UDP fornece a soma verificação (detecção de erro), uma vez que outros protocolos da camada de enlace como o Ethernet já oferecem a verificação de erros ?

- … não há garantia de que todos os enlaces entre fonte e destino contemplem verificação de erros nas camadas inferiores.

… 3.3.2 – Soma de Verificação UDP

- "**dúvida**" .. por que o UDP fornece a soma verificação (detecção de erro), uma vez que outros protocolos da camada de enlace como o Ethernet já oferecem a verificação de erros ?

- "**princípio fim-a-fim**" .. para funcionalidades definidas como de nível mais alto ou fim a fim, "funções" colocadas em níveis mais baixos podem ser consideradas redundantes ou terem pouco valor.
- … por isso, o custo de fornecê-las no nível mais alto, ainda que sejam contempladas em níveis mais baixos (mas nem sempre se sabe).
- e.g., neste contexto, destaca-se a "detecção de erros".

… 3.3.2 – Soma de Verificação UDP

- “**medida de segurança**” … considerando que o IP execute sobre qualquer protocolo de camada de enlace, é útil que a camada de transporte forneça verificação de erros como medida de segurança.
- … embora o UDP forneça verificação de erros, ele nada faz para recuperar-se de um erro.
- … algumas implementações do UDP apenas descartam o segmento danificado, enquanto outras passam o segmento com erros e alguma mensagem de aviso para a aplicação.

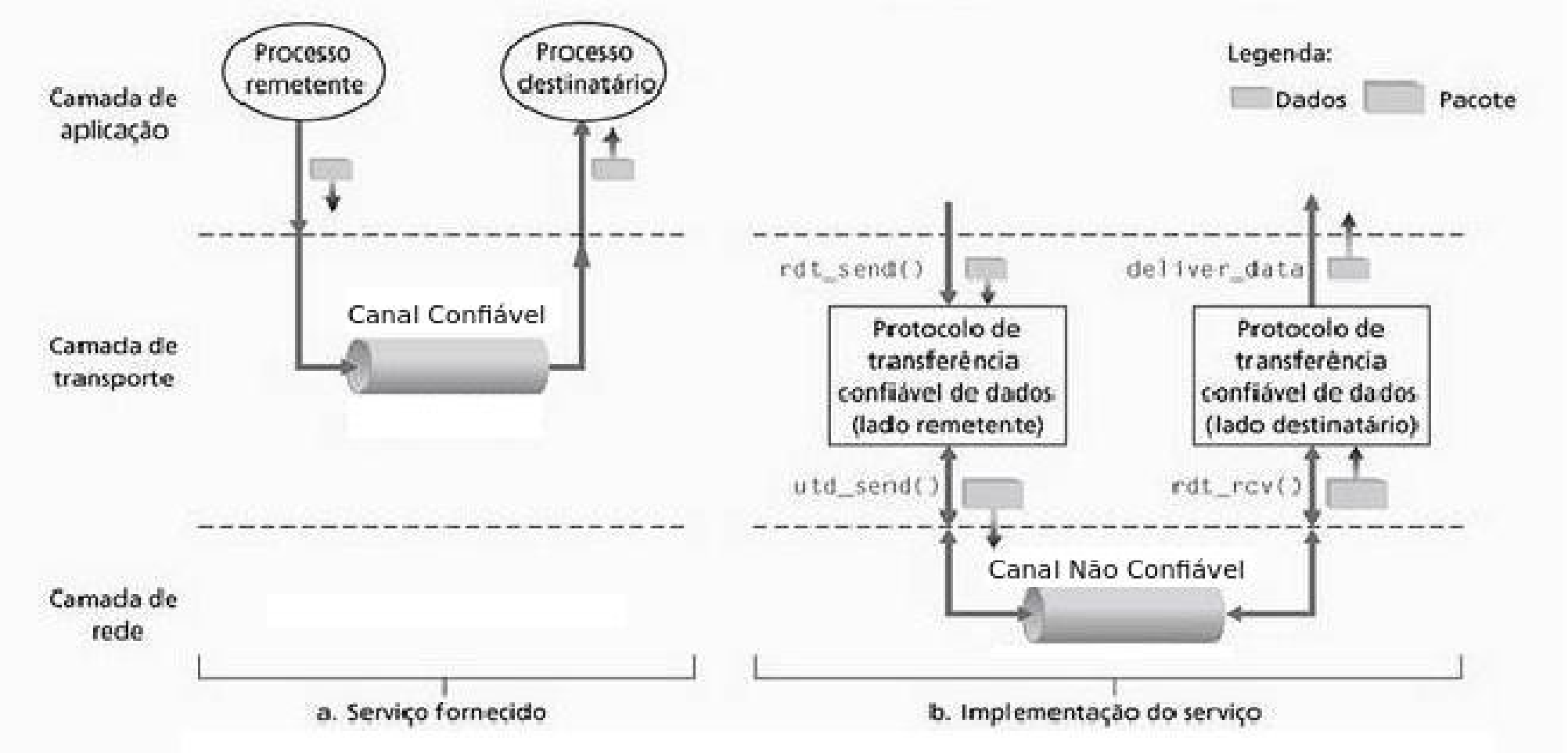
3.4 – Princípios da Transf. Confiável de Dados

- "**transferência confiável de dados**" - ocorre não somente na camada de transporte, mas também na camada de enlace e de aplicação, ou seja, é um problema de central importância.

- "**abstração de serviço**" .. visão da camada superior de que a camada logo abaixo e provedora de serviço disponibiliza um canal confiável através do qual dados podem ser transferidos.
- "**canal confiável**" - dados são transferidos sem corrupção, sem perdas e com garantia de entrega na ordem que foram enviados.

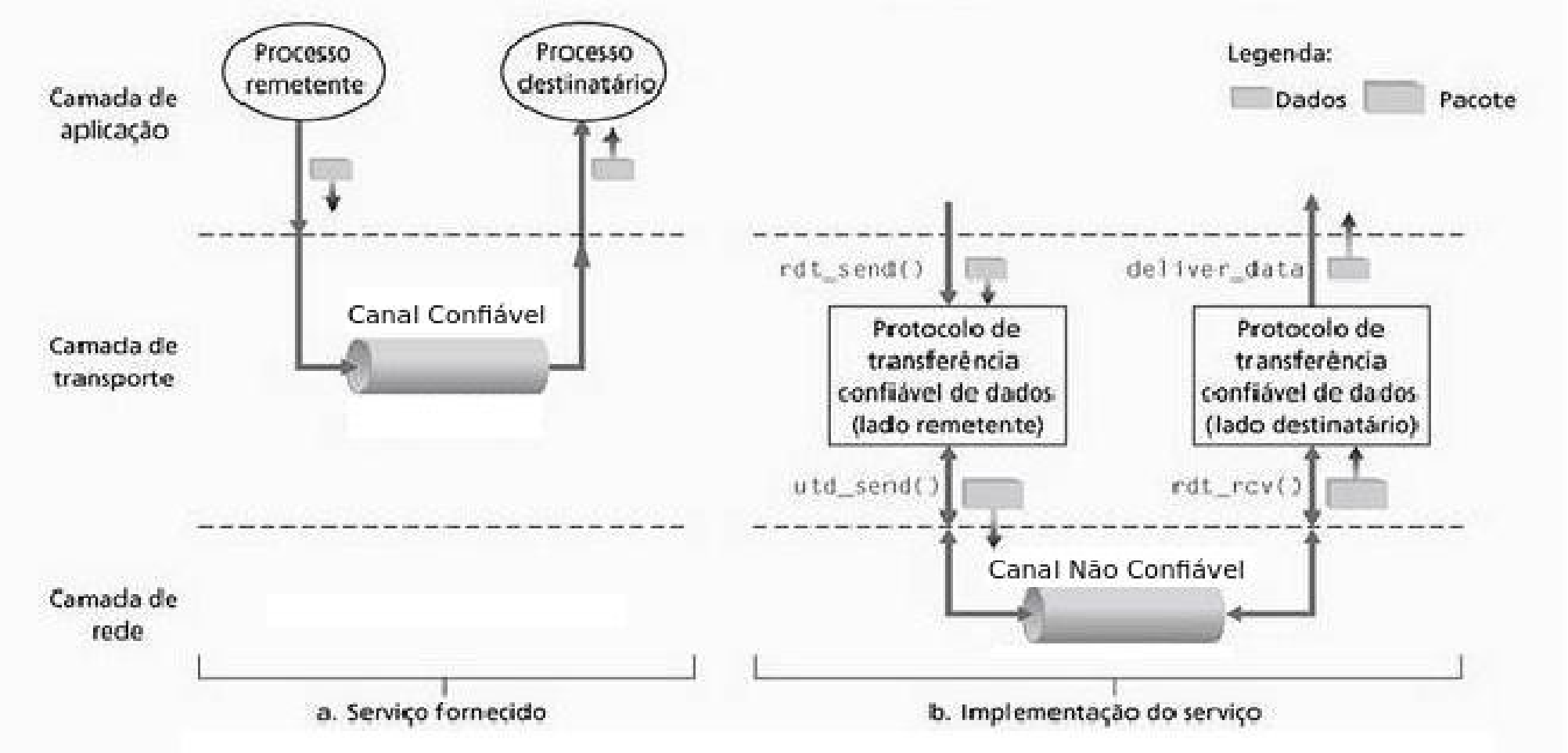
… 3.4 – Princípios da Transf. Confiável de Dados

- "**abstração de serviço**" .. visão da camada superior de que a camada provedora disponibiliza um canal confiável p/ transmissão de dados.



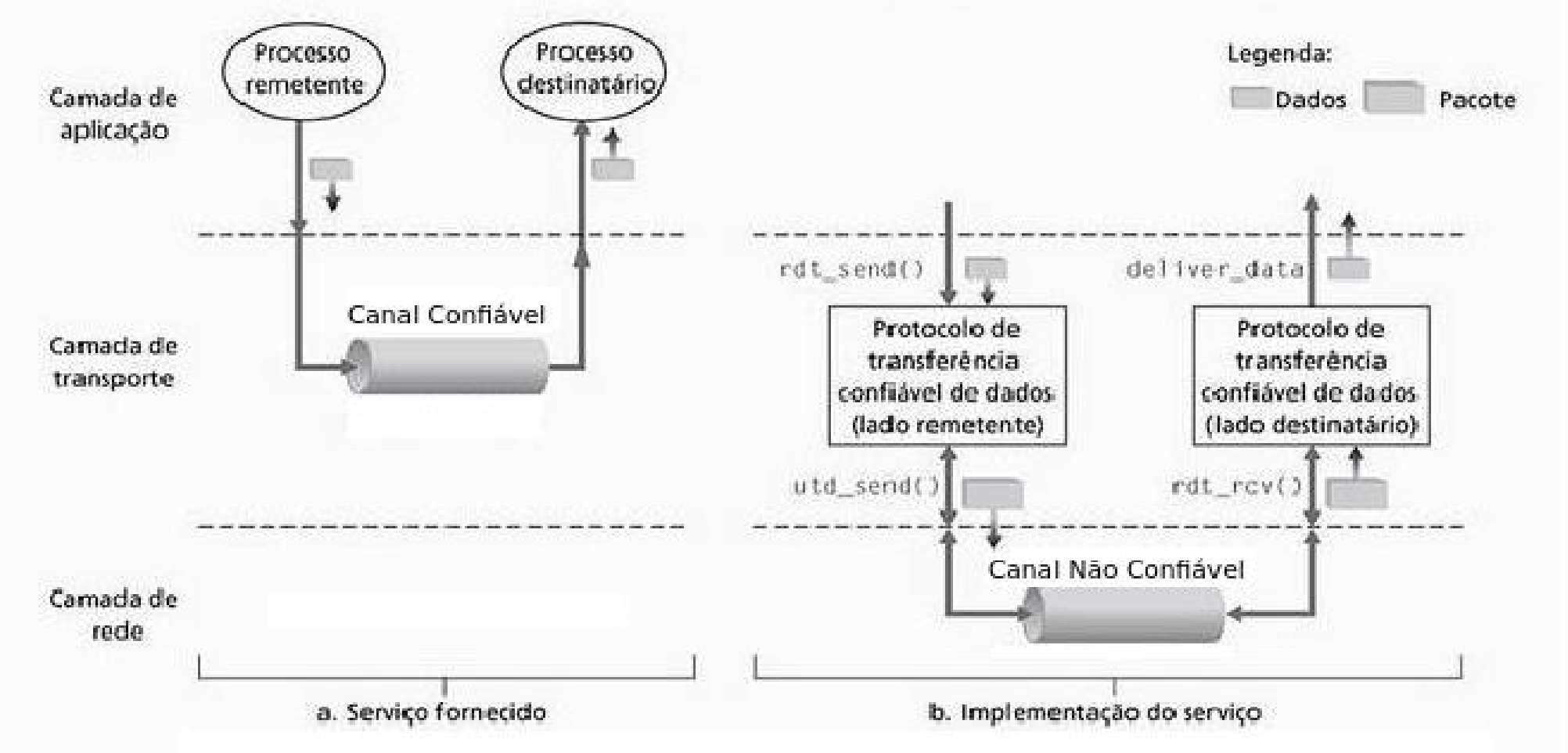
… 3.4 – Princípios da Transf. Confiável de Dados

- "**canal confiável**" .. dados são transferidos sem corrupção, sem perdas e com garantia de entrega na ordem que foram enviados.



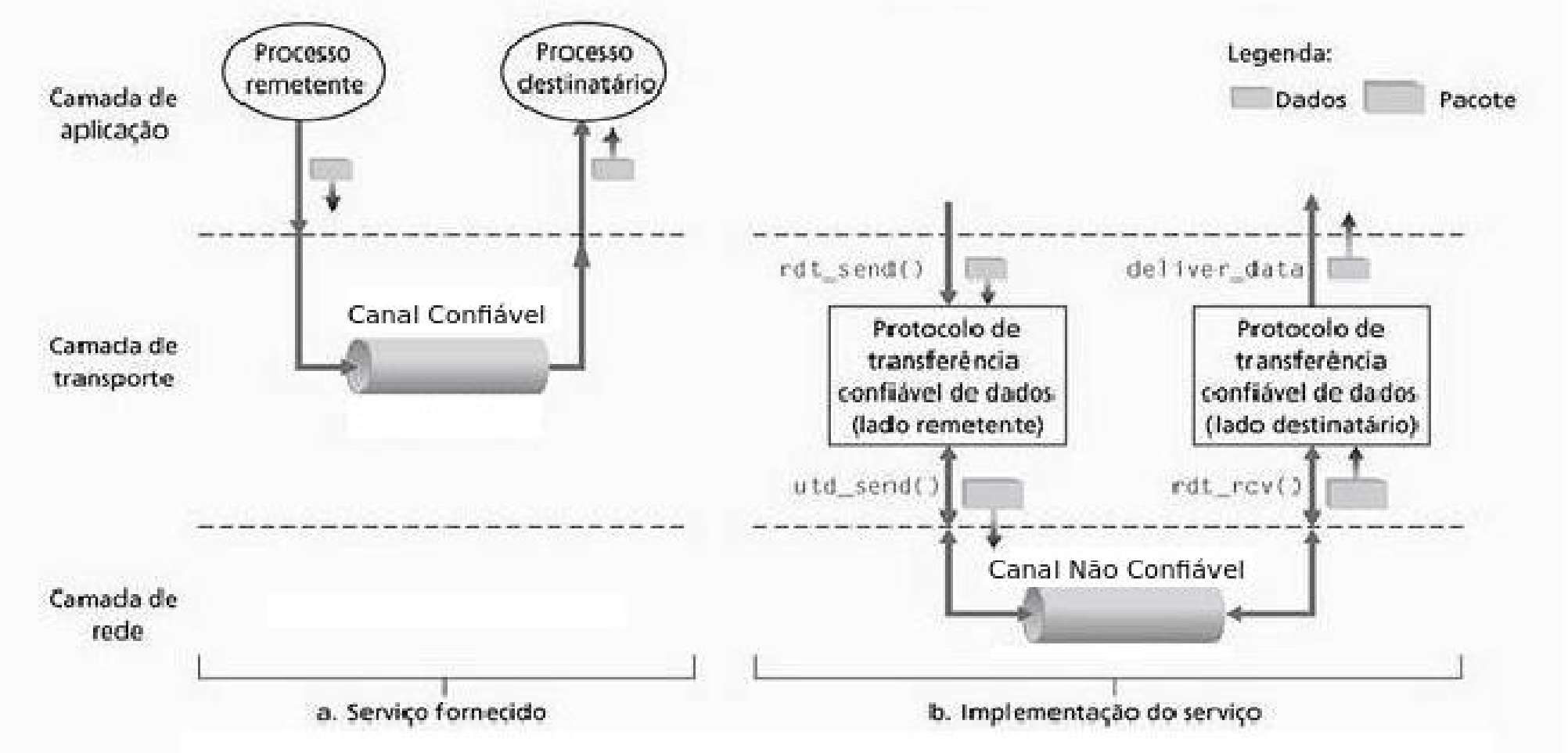
… 3.4 – Princípios da Transf. Confiável de Dados

- "**rdt_send(...)**" .. chamada de cima, p.ex., camada de aplicação pela qual dados são passados para enviar ao destinatário.



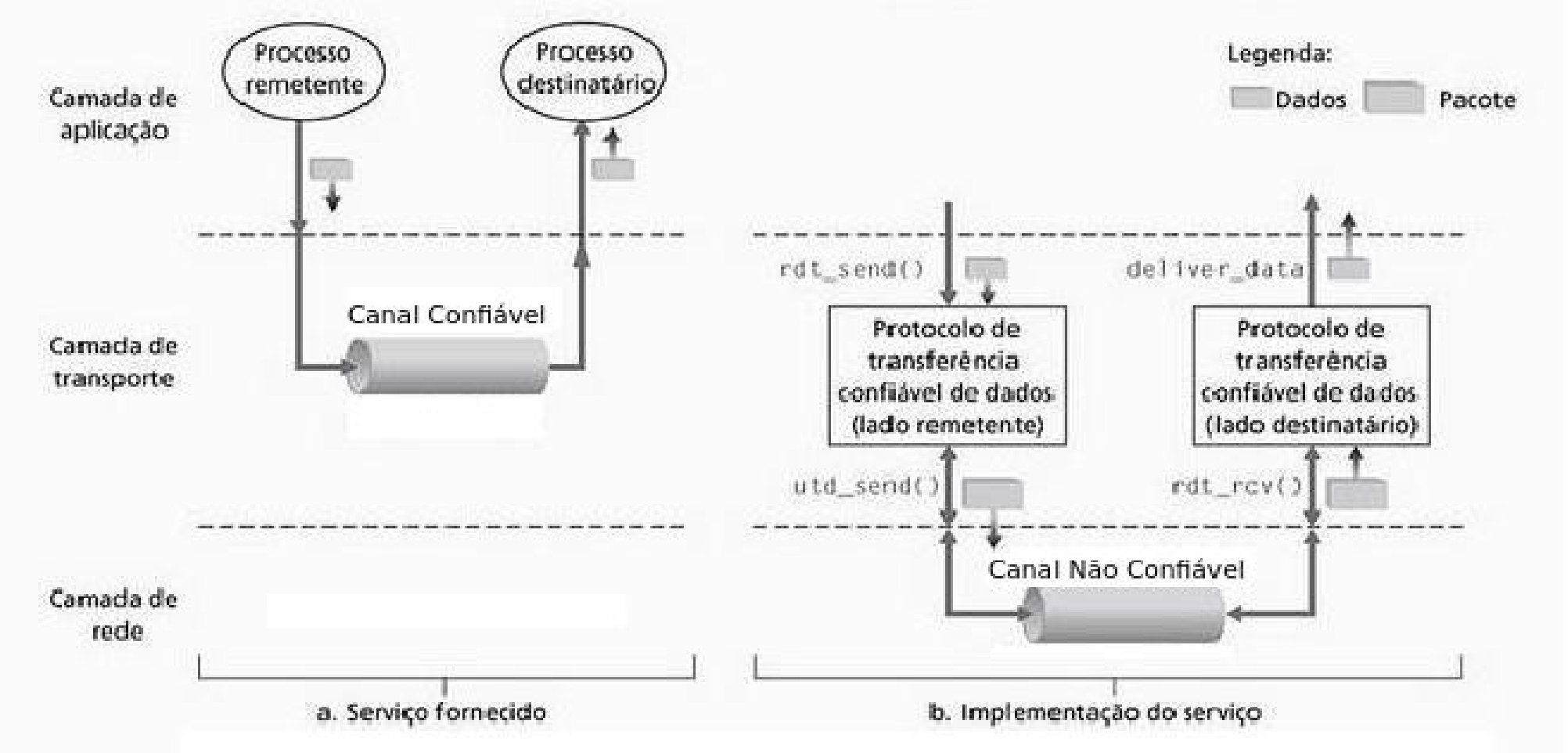
… 3.4 – Princípios da Transf. Confiável de Dados

- "**udt_send(...)**" .. chamada pelo serviço de transferência confiável para transferir o segmento por um canal não confiável ao destinatário.



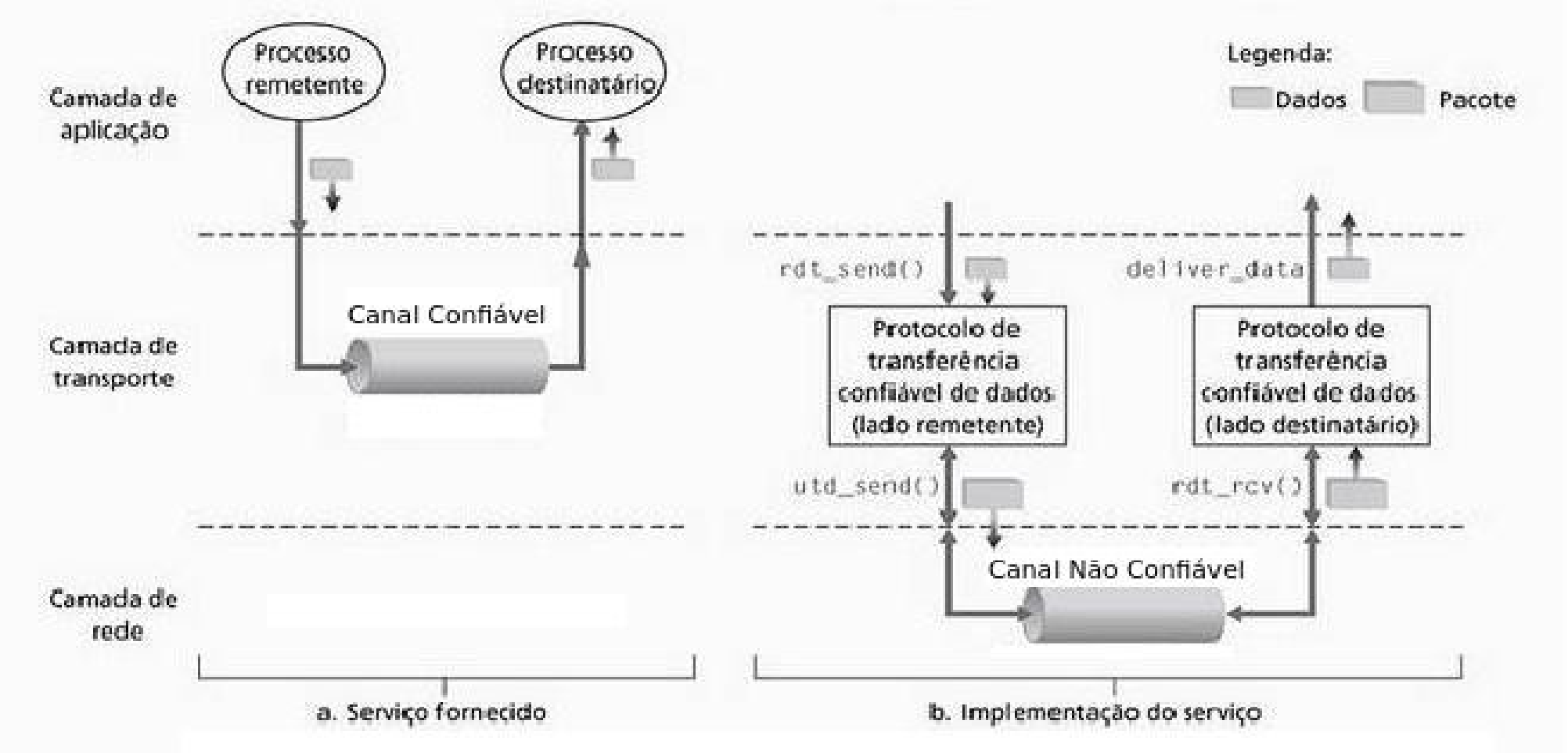
… 3.4 – Princípios da Transf. Confiável de Dados

- "**rdt_rcv(...)**" .. chamado ou invocado quando o segmento chega no lado destinatário do enlace de comunicação.



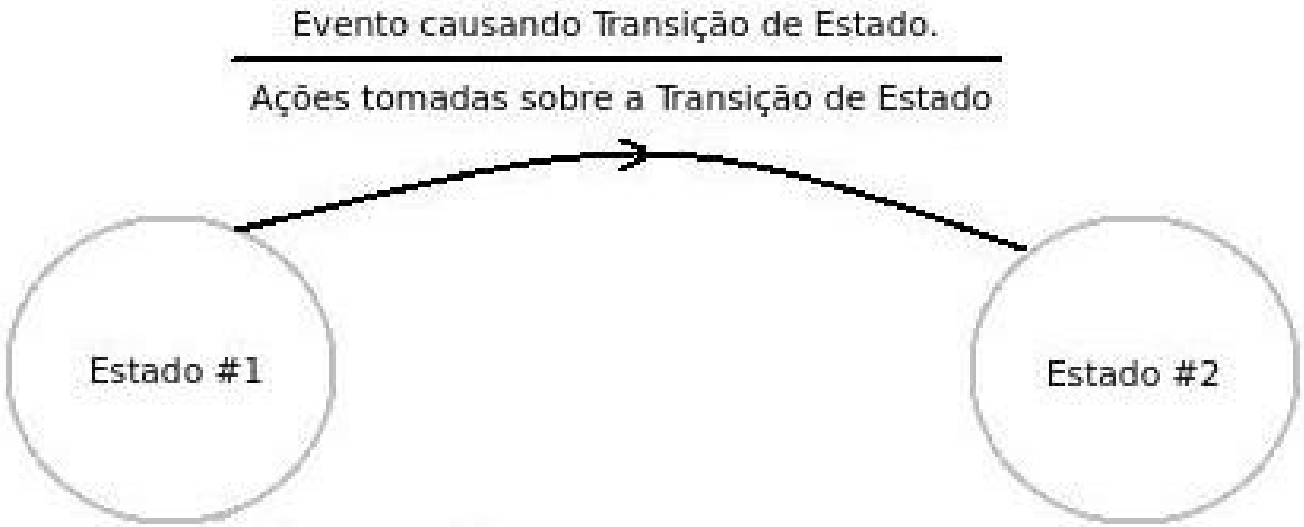
… 3.4 – Princípios da Transf. Confiável de Dados

- "**deliver_data(...)**" .. chamado pelo serviço de transferência confiável para enviar/entregar dados na camada acima, p.ex., aplicaçao.



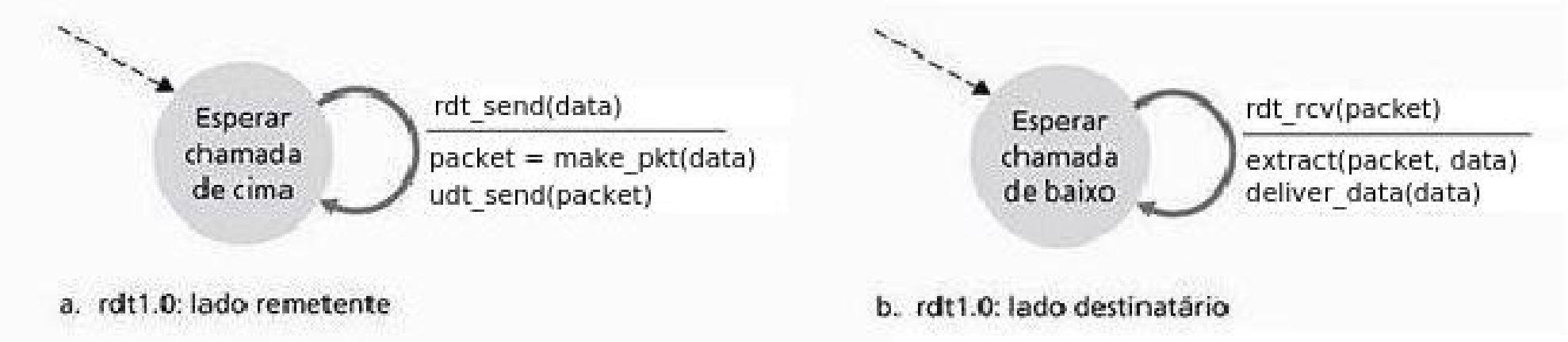
… 3.4 – Princípios da Transf. Confiável de Dados

- "**proposta do protocolo**" .. desenvolvimento incremental do remetente e destinatário com transferência unidirecional de dados (msgs. de dados), já as mensagens de controle trafegam nos 02 sentidos.
- **Finite State Machine** (FSM) .. utilizada para especificar remetente e destinatário, logo, é necessário identificar todos os estados.



… 3.4.1 – Protocolo de Transf. Confiável de Dados

- **RDT 1.0** .. canal subjacente é **completamente confiável**, ou seja, **sem erros** de bits e **sem perdas** de pacotes.
- … fluxo de pacotes corre do remetente para o destinatário através de um canal totalmente confiável, assim, não há a necessidade do lado destinatário fornecer qualquer informação ao remetente.
- … adicionalmente, o remetente e destinatário operam a qualquer velocidade de transferência de dados.



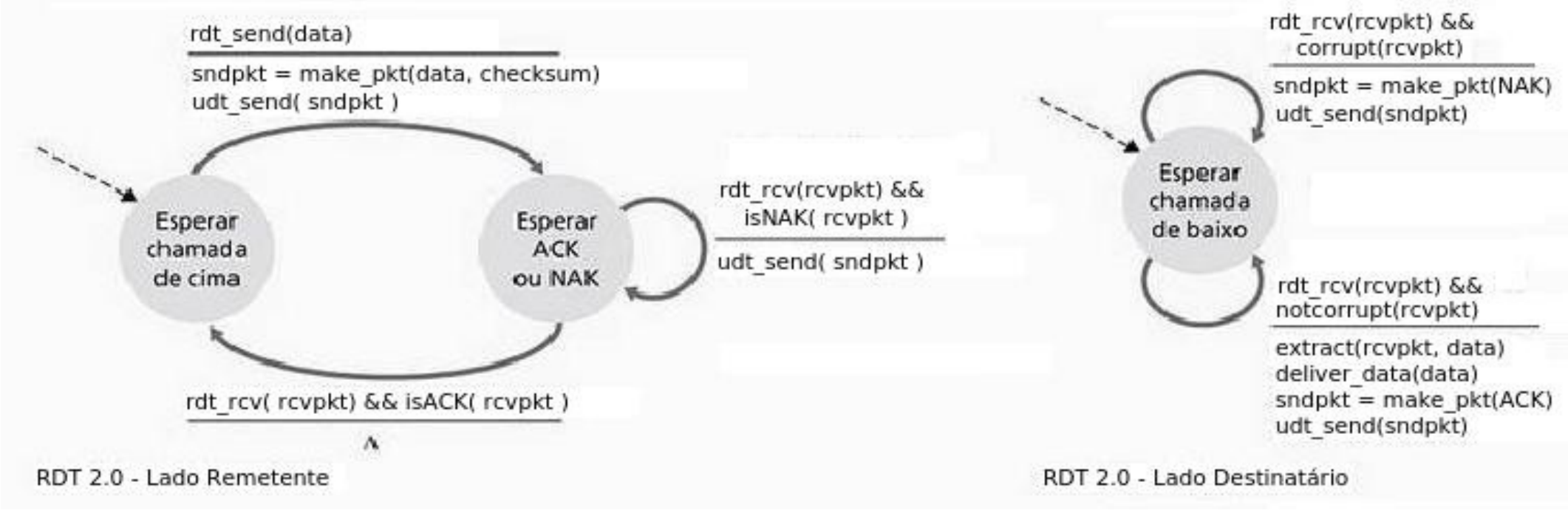
… 3.4.1 – Protocolo de Transf. Confiável de Dados

- **RDT 2.0** .. canal subjacente **não** é **completamente confiável**, ou seja, canal pode corromper os bits, mas não há perda de mensagem.
- … neste caso, receptor informa o lado remetente se o que foi recebido contém ou não erros, assim, o remetente pode retransmitir a mensagem ou enviar a próxima msg. caso haja mais dados a transmitir.

- "**Automatic Repeat reQuest**"… protocolos com msgs. de reconhecimento positivo e negativo são conhecidos por "ARQ Protocols".
- .. contemplam 03 características .. a) detecção de erros; b) realimentação do remetente e c) retransmissão da mensagem.

… 3.4.1 – Protocolo de Transf. Confiável de Dados

- … protocolo de transferência confiável de dados que emprega detecção de erros, reconhecimentos positivos e negativos.



… 3.4.1 – Protocolo de Transf. Confiável de Dados

- "**problema**" - canal pode corromper as msgs. ACK e NACK e, assim, até estas msgs podem não ser identificadas corretamente.

- "**soluções**" - algumas alternativas para contornar o problema:
- (1) ... adicionar bits ao campo no cabeçalho relacionado a erros de transmissão (p.ex., soma verificação) para permitir que o remetente não somente detecte, mas também se recupere de erros de bits.
- .. solução parece promissora, embora exija redundância de dados, no entanto não resolve o problema de perda de pacote.

… 3.4.1 – Protocolo de Transf. Confiável de Dados

- "**soluções**" - algumas alternativas para contornar o problema:
- (2) … reenviar a msg. corrente quando uma msg. ACK ou NACK truncada for recebida, no entanto, este método introduz pacotes duplicados no canal (necessidade de tratar pacotes duplicados).

- "**dúvida**" … com pacotes duplicados no canal, é necessário identificar o que foi enviado corretamente e o que precisa ser reenviado.
- … para resolver o este problema, adiciona-se um novo campo para identificar as mensagens de dados, ou seja, "**nro. de sequência**".

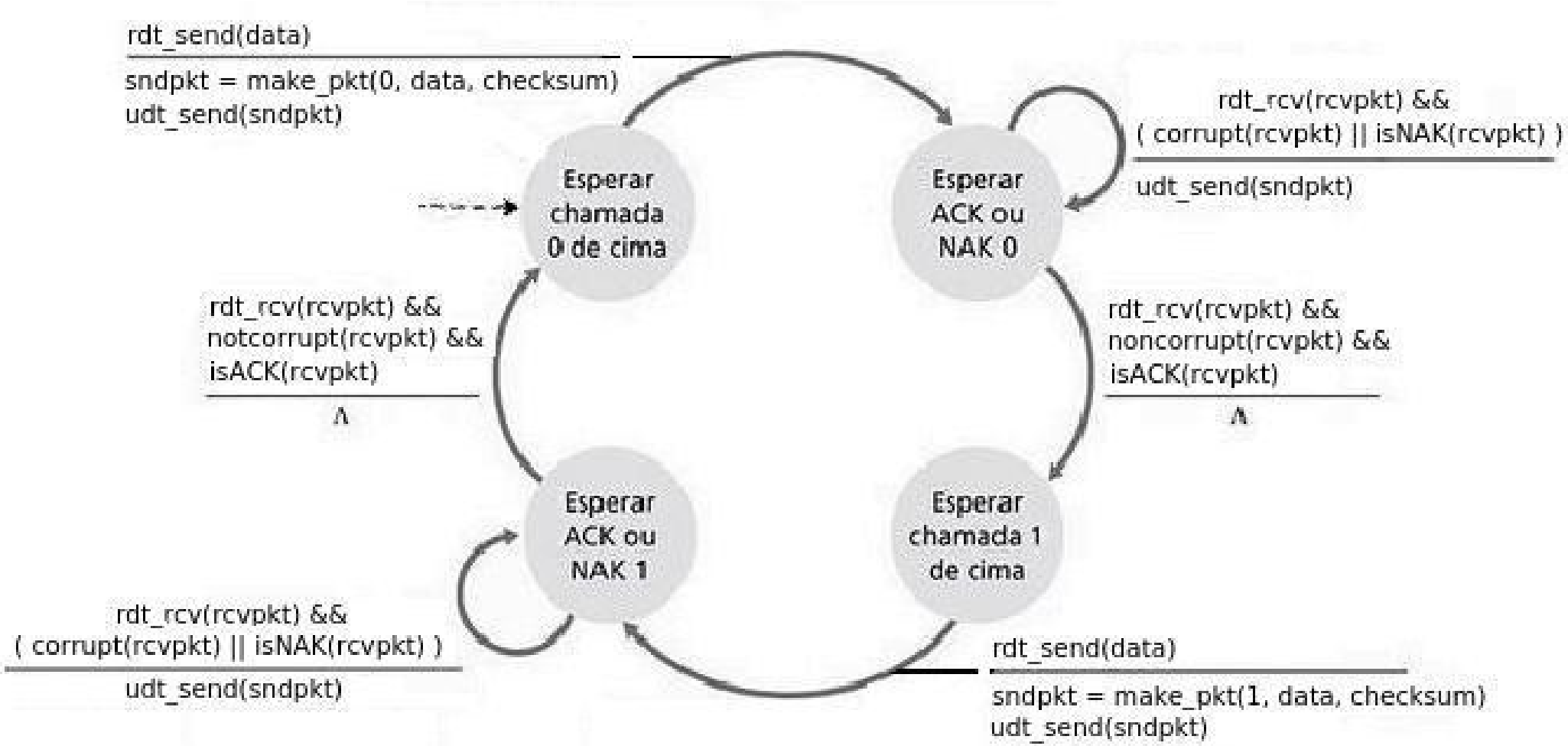
… 3.4.1 – Protocolo de Transf. Confiável de Dados

- **RDT 2.1** – versão **corrigida** do **RDT 2.0**
- "**premissa**" .. canal não perde msgs., logo, ACKs e NACKs não precisam indicar o nro. de sequência da msg. que estão reconhecendo.

- … adicionamos nro. de sequência na msg. de dados para determinar se a msg. recebida é uma retransmissão ou uma nova mensagem.
- … remetente sabe que uma msg. ACK ou NACK recebida (truncada ou não) foi gerada em resposta a sua msg. de dados que foi transmitida mais recentemente (última transmissão de dados).

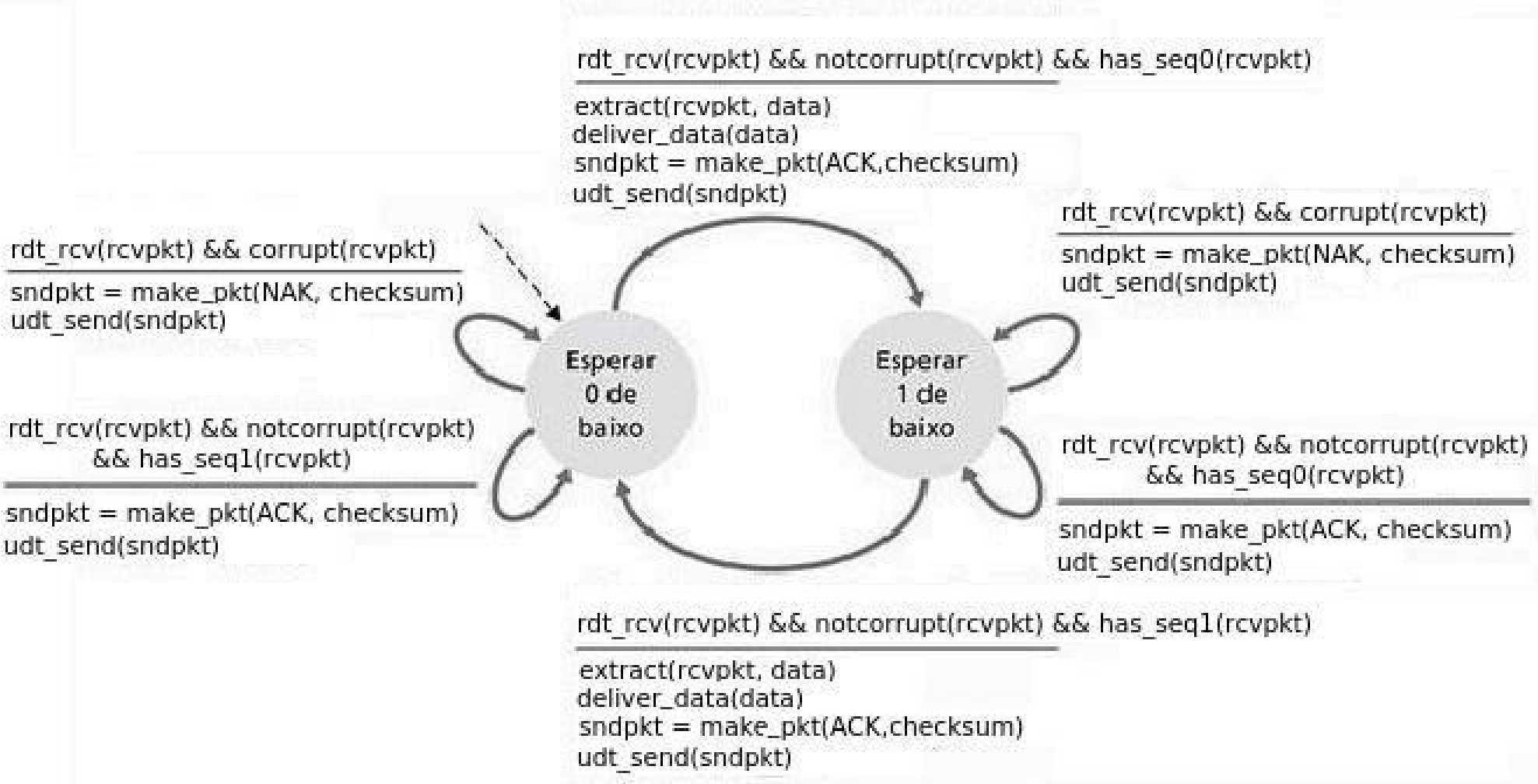
… 3.4.1 – Protocolo de Transf. Confiável de Dados

- **RDT 2.1 – versão corrigida do RDT 2.0** – Remetente.



… 3.4.1 – Protocolo de Transf. Confiável de Dados

- **RDT 2.1 – versão corrigida do RDT 2.0** – Destinatário.



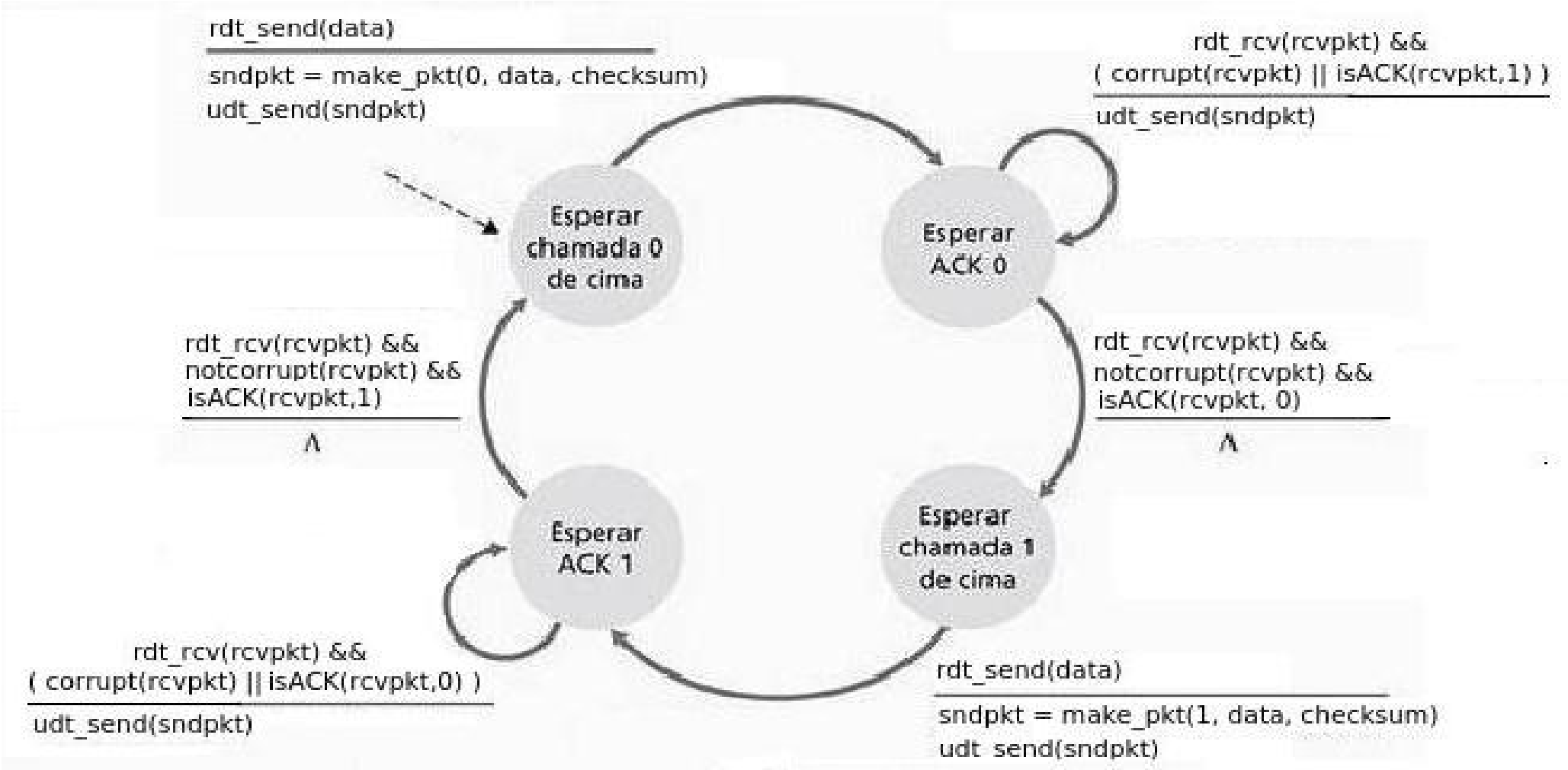
… 3.4.1 – Protocolo de Transf. Confiável de Dados

- **RDT 2.2** – **versão RDT 2.1 usando-se apenas ACKs**.
- "**revisão RDT 2.1**" .. ao receber um pacote fora de ordem, o destinatário envia um ACK para o remetente e, ao receber um pacote corrompido, o destinatário envia um NACK ao remetente.

- RDT 2.2 - … consegue-se o mesmo efeito de um pacote NACK se, em vez de enviarmos um NACK, enviarmos um ACK em seu lugar para o último pacote corretamente recebido.
- "ACKs Duplicados" .. um remetente que recebe 02 ACKs para o mesmo pacote sabe que o destinatário não recebeu corretamente o pacote seguinte aquele para o qual estão sendo dados 02 ACKs.

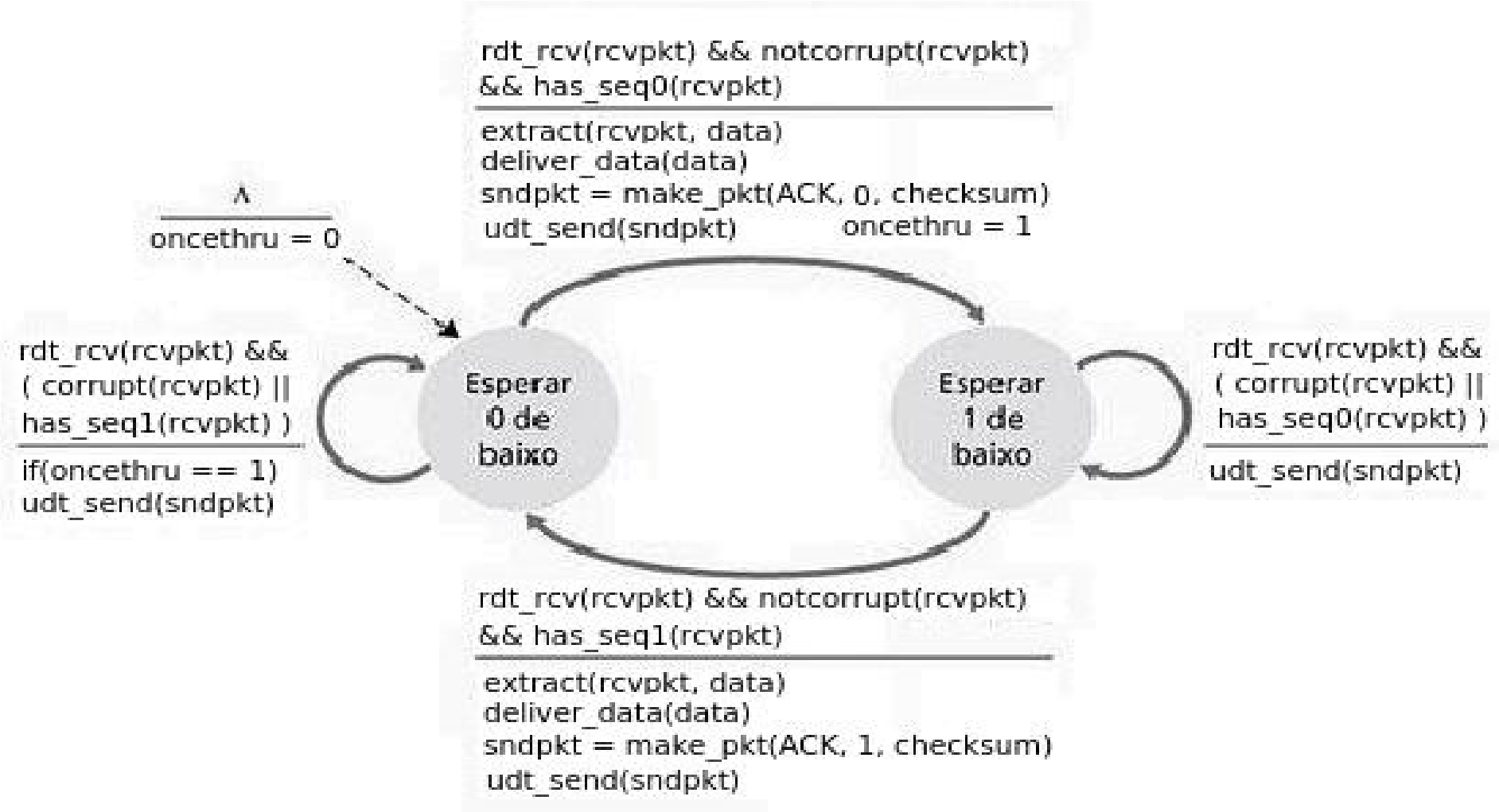
... 3.4.1 – Protocolo de Transf. Confiável de Dados

- FSM do RDT 2.2 - Remetente



... 3.4.1 – Protocolo de Transf. Confiável de Dados

- FSM do RDT 2.2 - Destinatário



… 3.4.1 – Protocolo de Transf. Confiável de Dados

- "**RDT 3.0**" .. **canal** subjacente pode **corrromper os bits** bem como **perder pacotes**, ou seja, **não é confiável**.
- … 02 novas preocupações surgem .. necessidade de detectar a perda de pacote e providências a serem tomadas quando isto ocorrer.
- … em RDT 2.2 as técnicas de "soma de verificação", "nros. de sequência", "ACKs" e "retransmissões" atenderam estas necessidades.

- "**solução #1**" … dentre as diversas maneiras de resolver este problema, pode-se atribuir ao remetente a tarefa de detecção e recuperação.

… 3.4.1 – Protocolo de Transf. Confiável de Dados

- "**solução #2**" .. resolve-se o problema permitindo que o remetente retransmita um pacote após um certo período de tempo.
- … normalmente, este intervalo de tempo ou "timeout" considera o tempo de ida e volta do pacote mais o tempo de processamento.
- … se durante este intervalo de tempo não for recebido um ACK, o remetente retransmite o pacote.

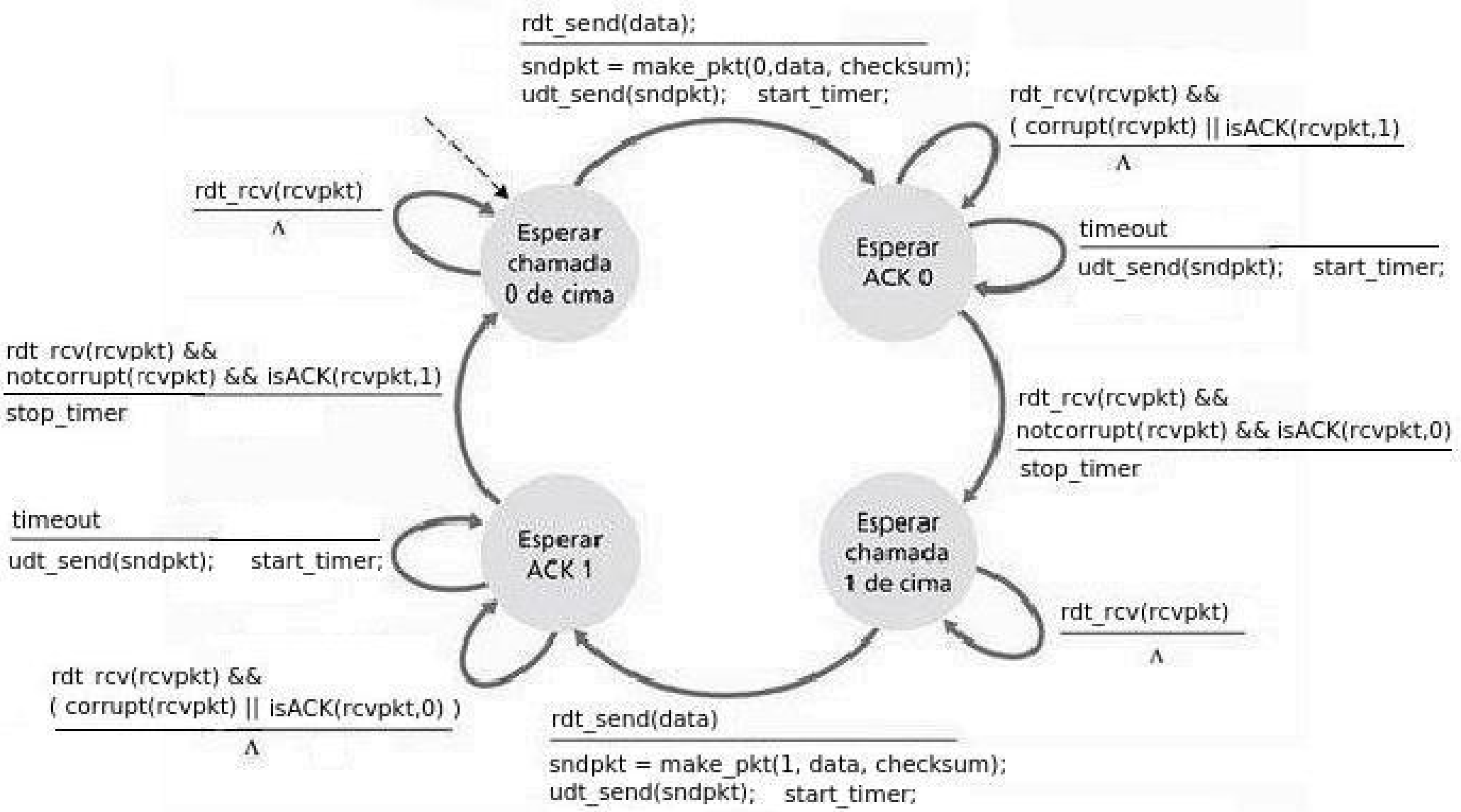
- Obs.: … isto no entanto não significa que o pacote tenha se perdido, mas que o atraso foi maior do que o preestabelecido.
- … esta situação também introduz a possibilidade de pacotes duplicados no canal entre remetente e destinatário.

… 3.4.1 – Protocolo de Transf. Confiável de Dados

- "**ponto de vista do remetente**" - remetente não sabe se o pacote transmitido foi perdido, se um ACK foi perdido ou se o pacote ou ACK simplesmente estão muito atrasados.
- … mas em todos os casos a ação é a "retransmissão do pacote".

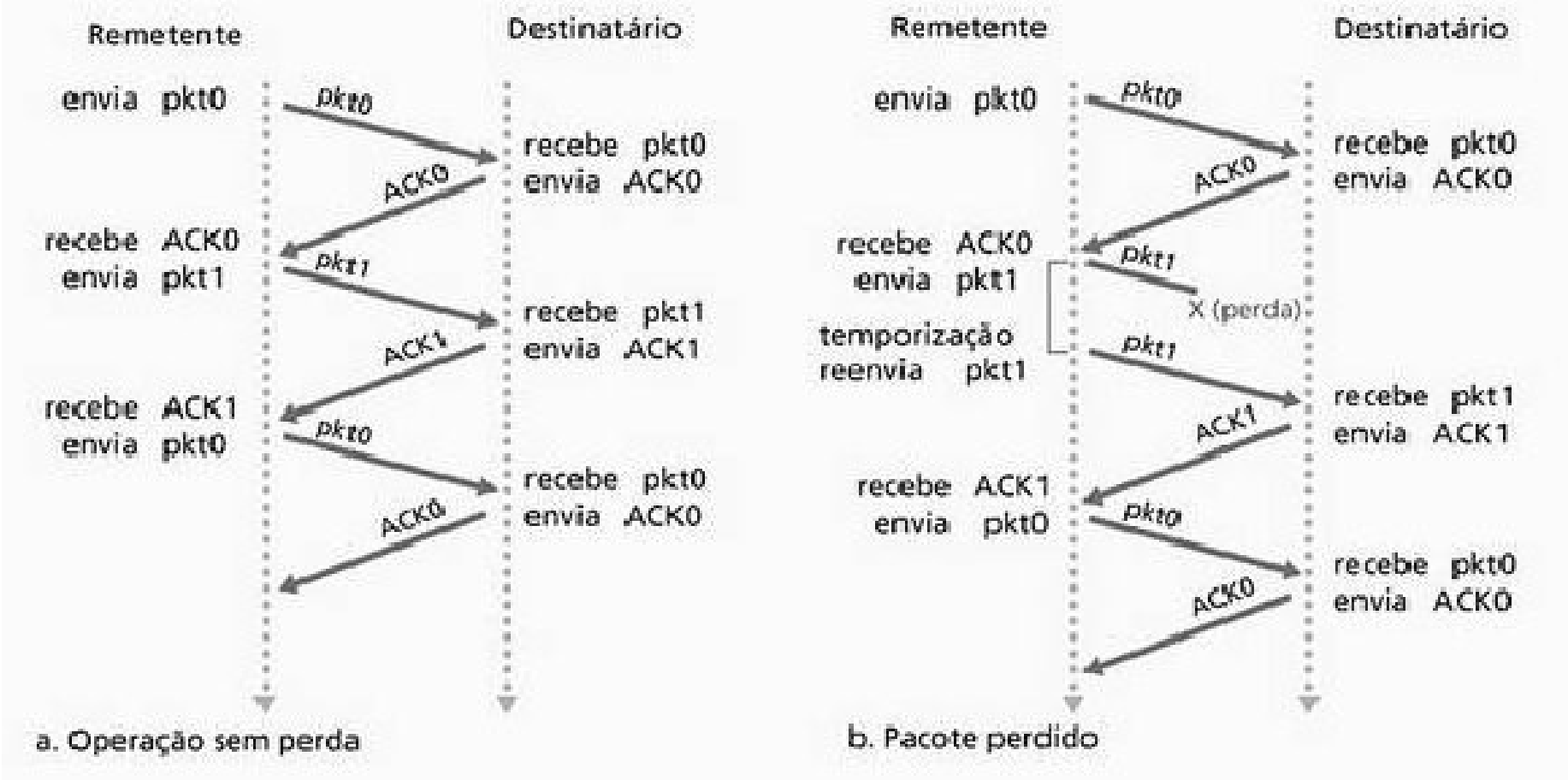
- "**mecanismo de retransmissão**" … faz-se necessário um temporizador de contagem regressiva que interrompe o processo remetente após decorrido um "**tempo preestabelecido**" = "**timeout**".

... 3.4.1 – Protocolo de Transf. Confiável de Dados



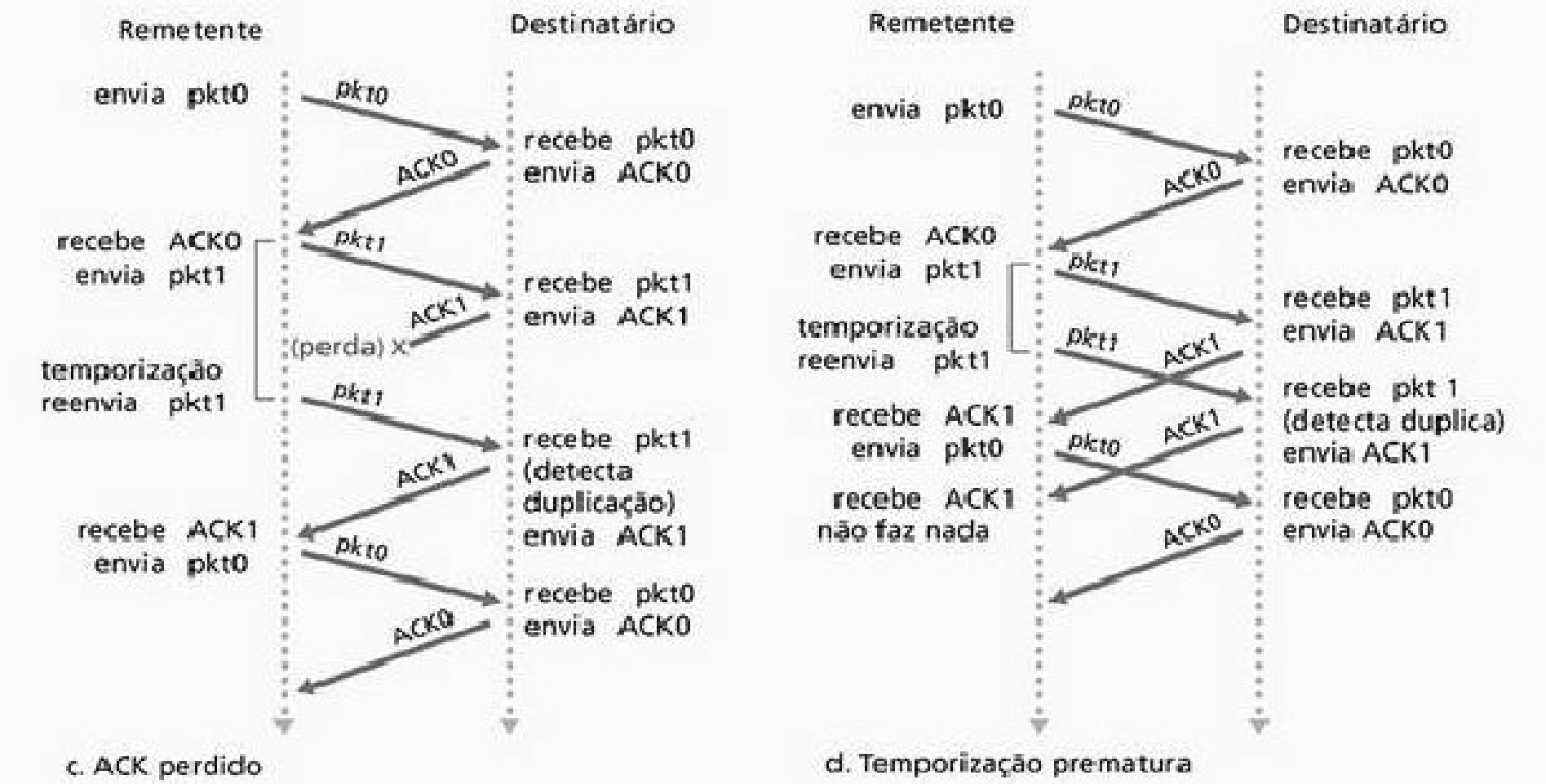
… 3.4.1 – Protocolo de Transf. Confiável de Dados

- “**RDT 3.0**” – **Protocolo Bit Alternante**.



… 3.4.1 – Protocolo de Transf. Confiável de Dados

- "**RDT 3.0**" – **Protocolo Bit Alternante**.



3.4.2 – Transf. Confiável de Dados com Paralelismo

- "**desempenho**" - RDT 3.0 apresenta um desempenho baixo em redes de alta velocidade uma vez que é um protocolo do tipo pare e espere (temporizador de contagem regressiva).
- … atraso de propagação tem grande influência no desempenho de redes de grandes distâncias e de alta velocidade.

a. Um protocolo pare e espere em operação

… 3.4.2 – Transf. Confiável de Dados com Paralelismo

- e.g., considere um canal de capacidade R = 1 Gbps (10^9 bps) e pacotes de comprimento L = 1000 bytes (8.000 bits).

- "**considerações**" e/ou suposições acerca dos atrasos:
- T (transmissão) = L/R = 8.000 bits / 10^9 bits/seg = 8 microsegundos
- T (propagação) ou RTT = 30 milisegundos .. atraso de propagação de ida e volta à velocidade da luz entre 02 sistemas finais.

- $T\,(total)_{ATRASO}$ = RTT/2 + L/R = 15,008 milisegundos

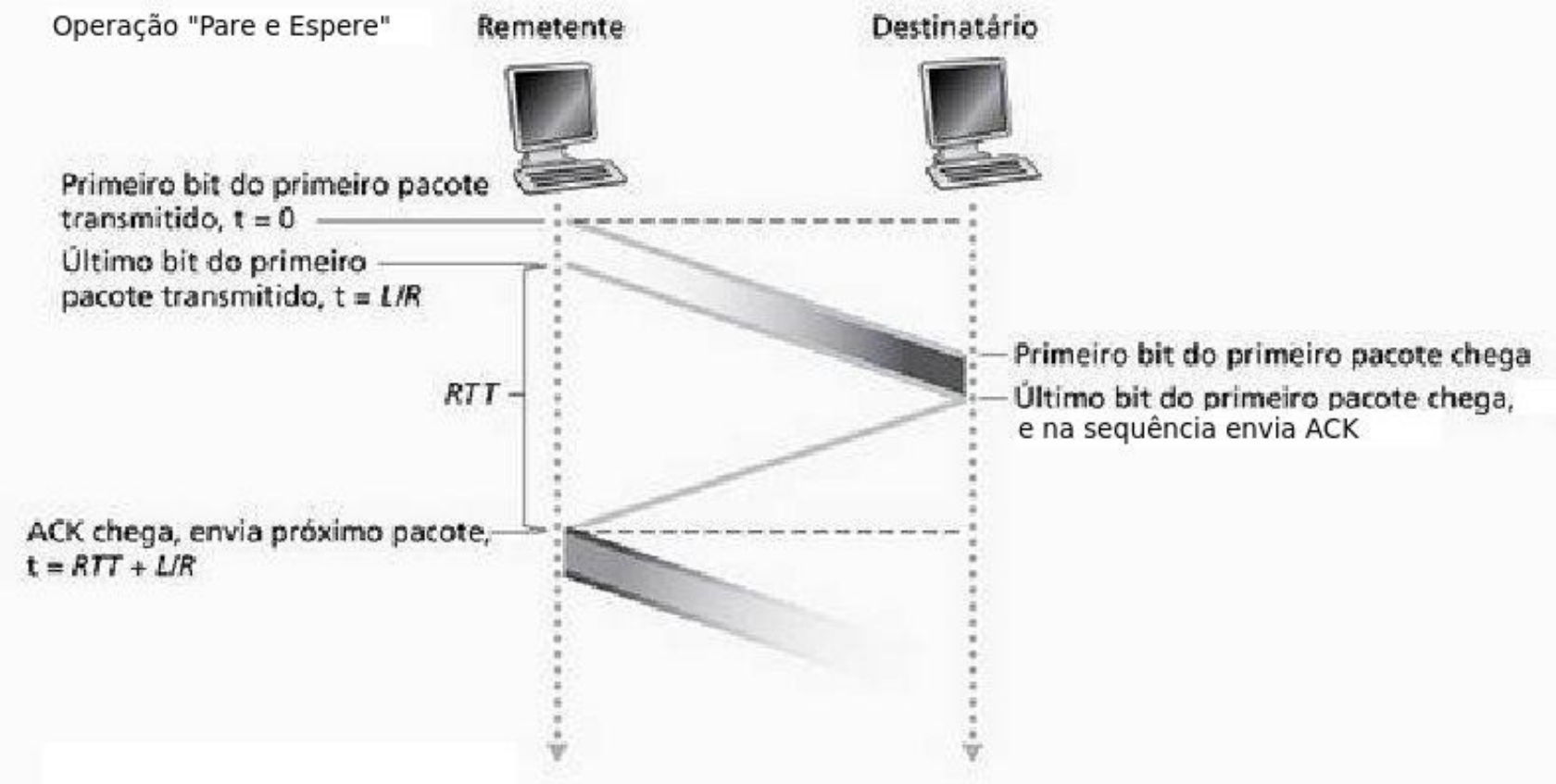
… 3.4.2 – Transf. Confiável de Dados com Paralelismo

- e.g., considere um canal de capacidade R = 1 Gbps (10^9 bps) e pacotes de comprimento L = 1 Kbytes (8.000 bits).
- T_{ATRASO} = RTT/2 + L/R = 15,008 milisegundos

- … sob o princípio de “pare e espere”, o ACK emergirá de volta no remetente em t = RTT + L/R = 30,008 milisegundos;
- … somente agora o remetente pode transmitir a próxima mensagem, ou seja, em 30,008 ms o remente utilizou apenas 0,008 ms para transmitir >> restante do tempo ficou esperando pelo ACK !?

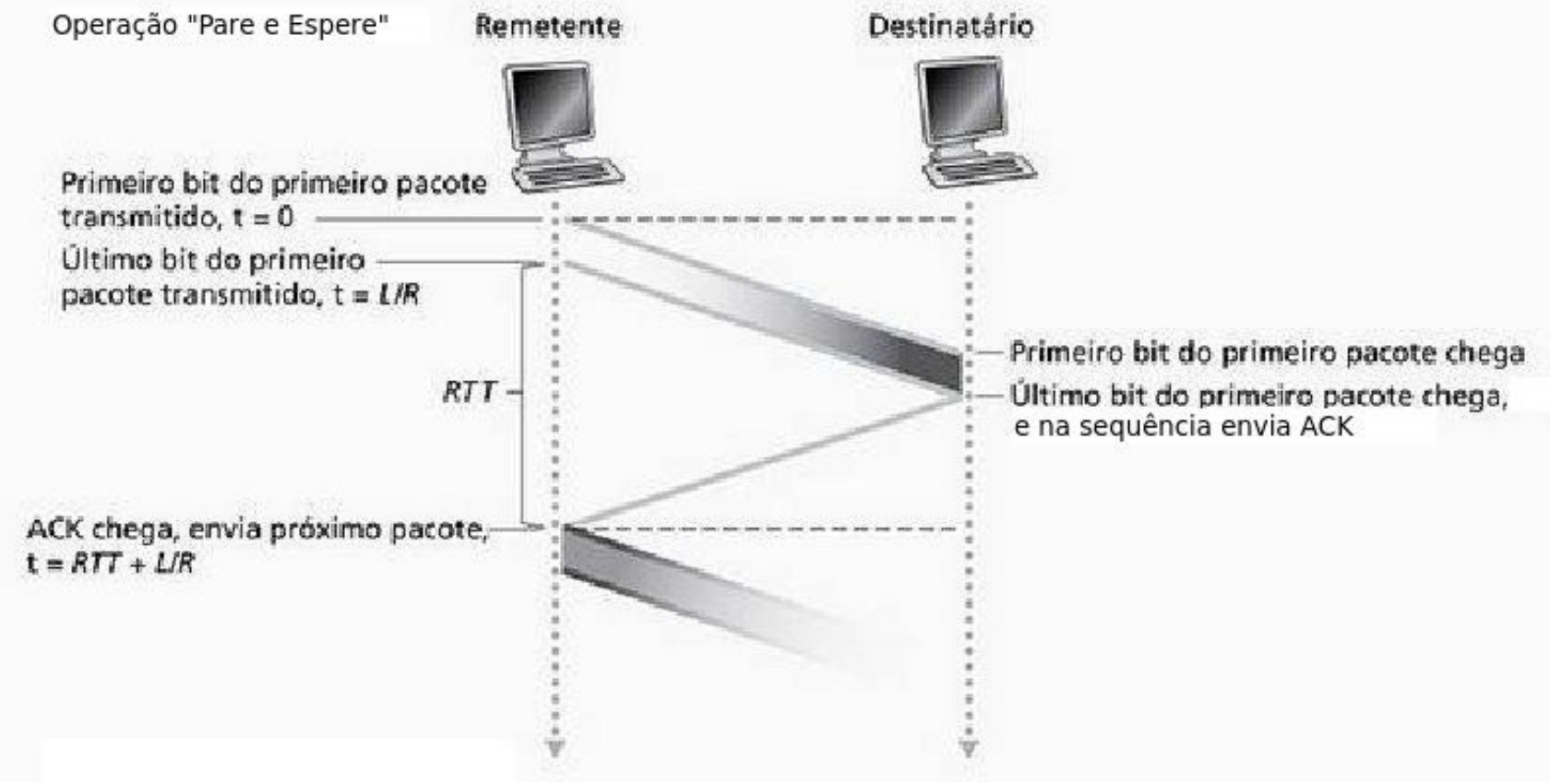
… 3.4.2 – Transf. Confiável de Dados com Paralelismo

- "**utilização**" .. fração de tempo em que o remetente está realmente ocupado enviando bits no canal >> $U = T_{TRANS.} / (T_{TRANS.} + T_{PROG.})$
- $U_{REMETENTE} = 0,008 / 30,008 = 0,00027 = 2,7$ centésimos de 1 %.



… 3.4.2 – Transf. Confiável de Dados com Paralelismo

- … ao enviar 1000 bytes em 30,08 milisegundos, vazão efetiva é de 267 Kbps – mesmo sobre um enlace de 10^9 bps (1000 Mbps).



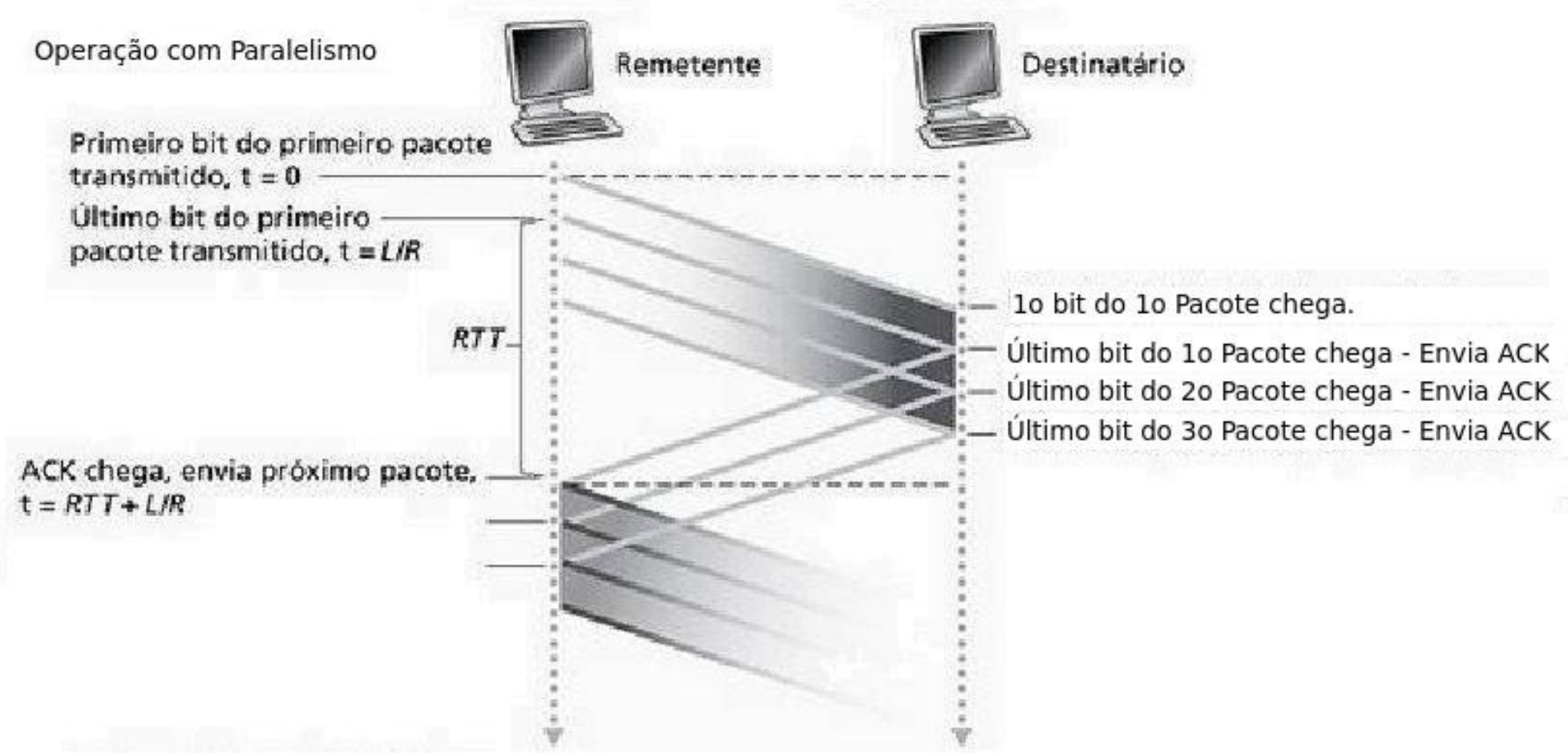
… 3.4.2 – Transf. Confiável de Dados com Paralelismo

- "**solução**" - remetente envia vários pacotes em sequência e avança no envio de mais pacotes a medida que recebe pacotes de reconhe-cimento referente ao que foi enviado.

b. Um protocolo com paralelismo em operação

... 3.4.2 – Transf. Confiável de Dados com Paralelismo

- .. remetente envia 03 pacotes antes e na sequência aguarda pacotes de reconhecimento (ACKs).



… 3.4.2 – Transf. Confiável de Dados com Paralelismo

- "**pipelining**" .. aumenta a taxa de utilização do canal mantendo em trânsito um nro. grande de pacotes sem que o pacote de reconhe-cimento correspondente tenha sido recebido.

- "**solução**" .. quais as "exigências do paralelismo" ??
- (1) faixa de nro. de sequência.
- (2) reserva de recursos (remetente e destinatário).
- (3) nro de sequência e buffers (relação entre as variáveis).

… 3.4.2 – Transf. Confiável de Dados com Paralelismo

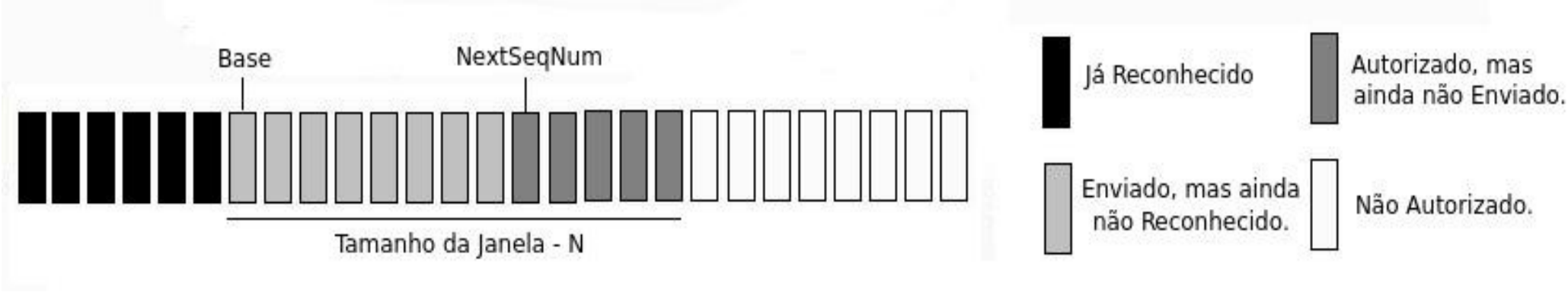
- "**solução**" .. quais as "exigências do paralelismo" ??

- "**faixa de nro. de sequência**" .. deve ser ampliada para garantir que cada pacote em trânsito seja identificado univocamente na sessão.
- "**reserva de recursos**" .. necessário reservar "buffers" para pacotes que foram transmitidos mas ainda não foram reconhecidos.
- .. não somente no remetente, mas o destinatário necessita reservar "buffers" para pacotes recebidos fora de ordem.
- "**nro. de sequência e buffers**" .. depende da maneira como o protocolo de transferência de dados responde a pacotes perdidos, corrompidos e demasiadamente atrasados.

3.4.3 – Protocolo Go-Back-N

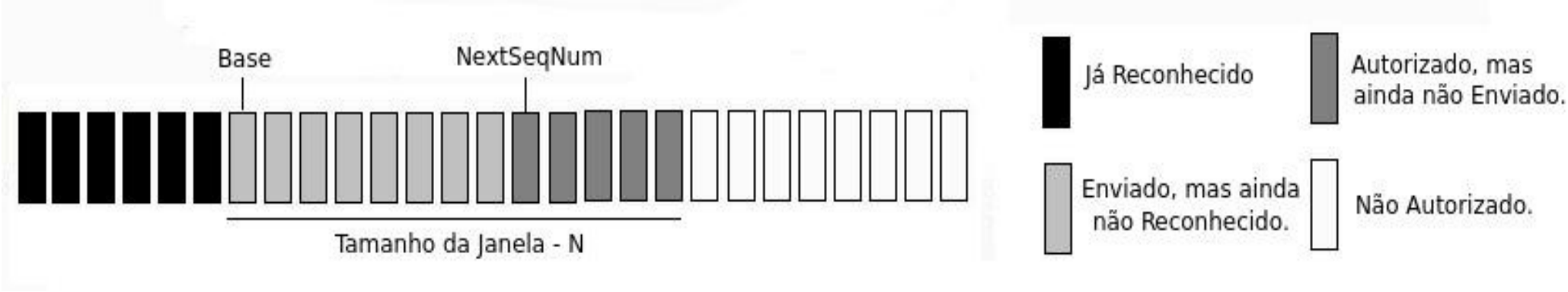
- "**Go-Back-N**" - remetente é autorizado a transmitir múltiplos pacotes sem esperar por um reconhecimento, mas fica limitado a "N" pacotes transmitidos por vez e não reconhecidos.

- "**base**" .. nro. de sequência do pacote mais antigo não reconhecido.
- "**nextseqnum**" .. menor nro. de sequência não utilizado, ou seja, nro. de sequência do próximo pacote a ser enviado.



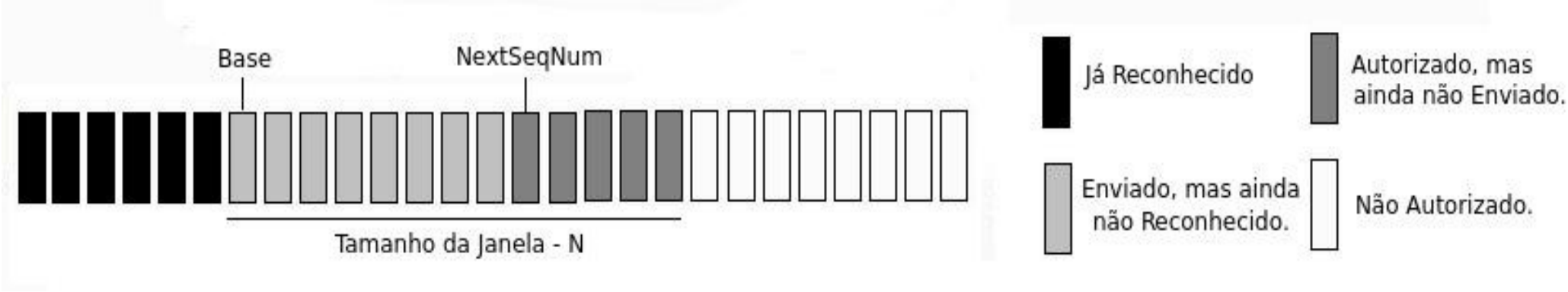
… 3.4.3 – Protocolo Go-Back-N

- "**Go-Back-N**" .. remetente é autorizado a transmitir múltiplos pacotes sem esperar por um reconhecimento, mas fica limitado a "N" pacotes transmitidos por vez e não reconhecidos.
- "**janela de tamanho N**" .. faixa de nro. de sequência permitido para pacotes transmitidos mas ainda não reconhecidos.
- … na prática o nro. de sequência de um pacote é carregado em um campo do comprimento fixo no cabeçalho do pacote.



… 3.4.3 – Protocolo Go-Back-N

- "**Go-Back-N**" .. remetente é autorizado a transmitir múltiplos pacotes sem esperar por um reconhecimento, mas fica limitado a "N" pacotes transmitidos por vez e não reconhecidos.
- … se "k" é o nro. de bits do campo de nro. de sequência do pacote, a faixa de nros. de sequência será [0, 2^k – 1].
- … com faixa finita de nros. de sequência, toda a aritmética que envolve nros. de sequência deve usar módulo "2^k" (lista circular).



… 3.4.3 – Protocolo Go-Back-N

- Remetente GBN responde a 03 eventos:

- (1) “**rdt_send(...)**” .. primeiramente verifica se a janela está cheia, ou seja, se há “N” mensagens pendentes de reconhecimento.
- “rdt_send(...)” .. caso não esteja cheia, pacotes são criados e enviados e, caso esteja cheia, devolve os dados para a aplicação.

… 3.4.3 – Protocolo Go-Back-N

- Remetente GBN responde a 03 eventos:

- (2) “**recebimento de ACK**” .. reconhecimento de pacote com nro. de sequência “n” é tomado como um reconhecimento cumulativo, logo, pacotes até este nro. de sequência foram corretamente recebidos.

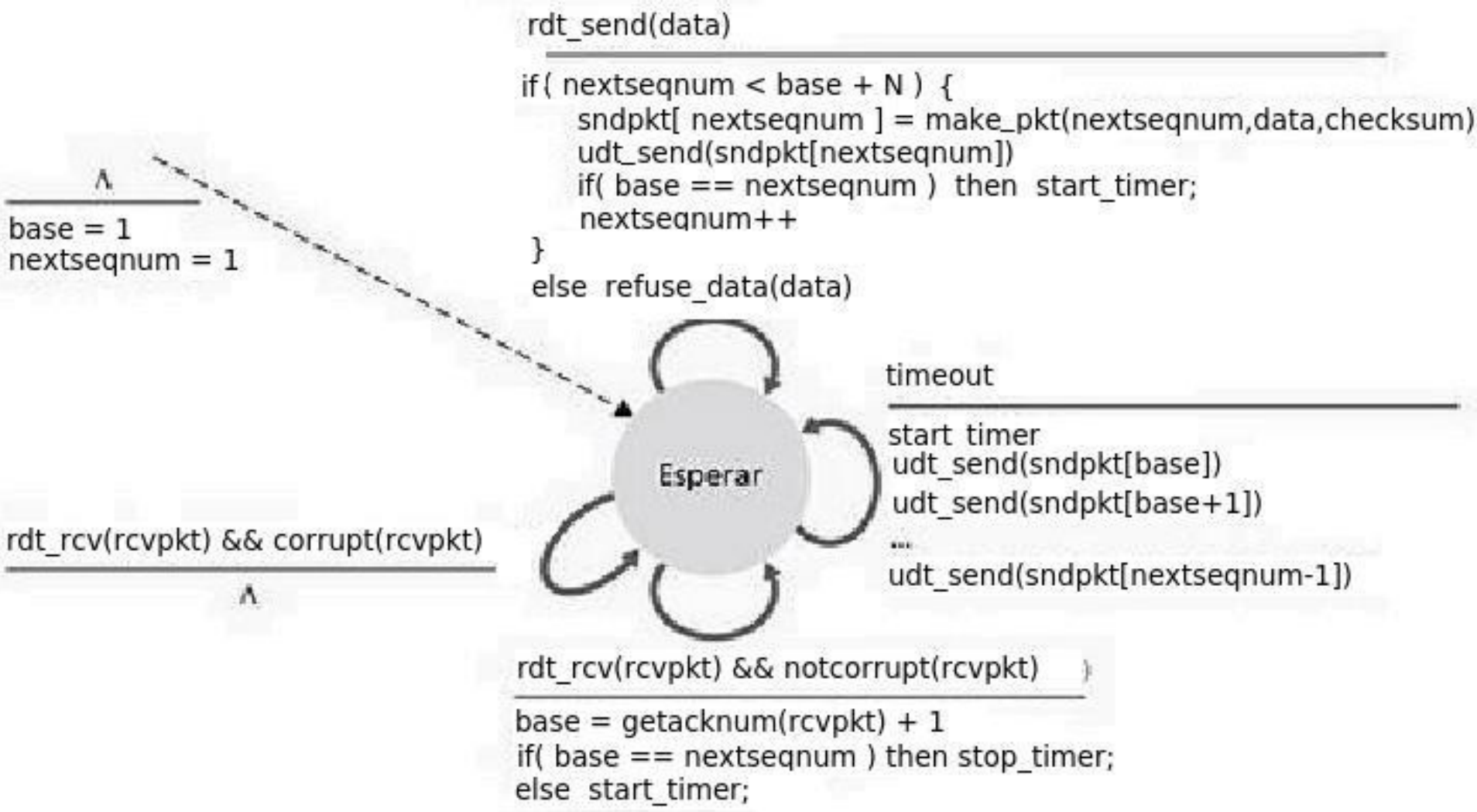
… 3.4.3 – Protocolo Go-Back-N

- Remetente GBN responde a 03 eventos:

- (3) “**esgotamento de temporização**” .. se ocorrer “timeout”, remetente envia todos os pacotes que tinham sido previamente enviados mas que ainda não foram reconhecidos.
- … temporização utiliza máquina de estados do remetente e vale para o pacote mais antigo transmitido, mas ainda não reconhecido.

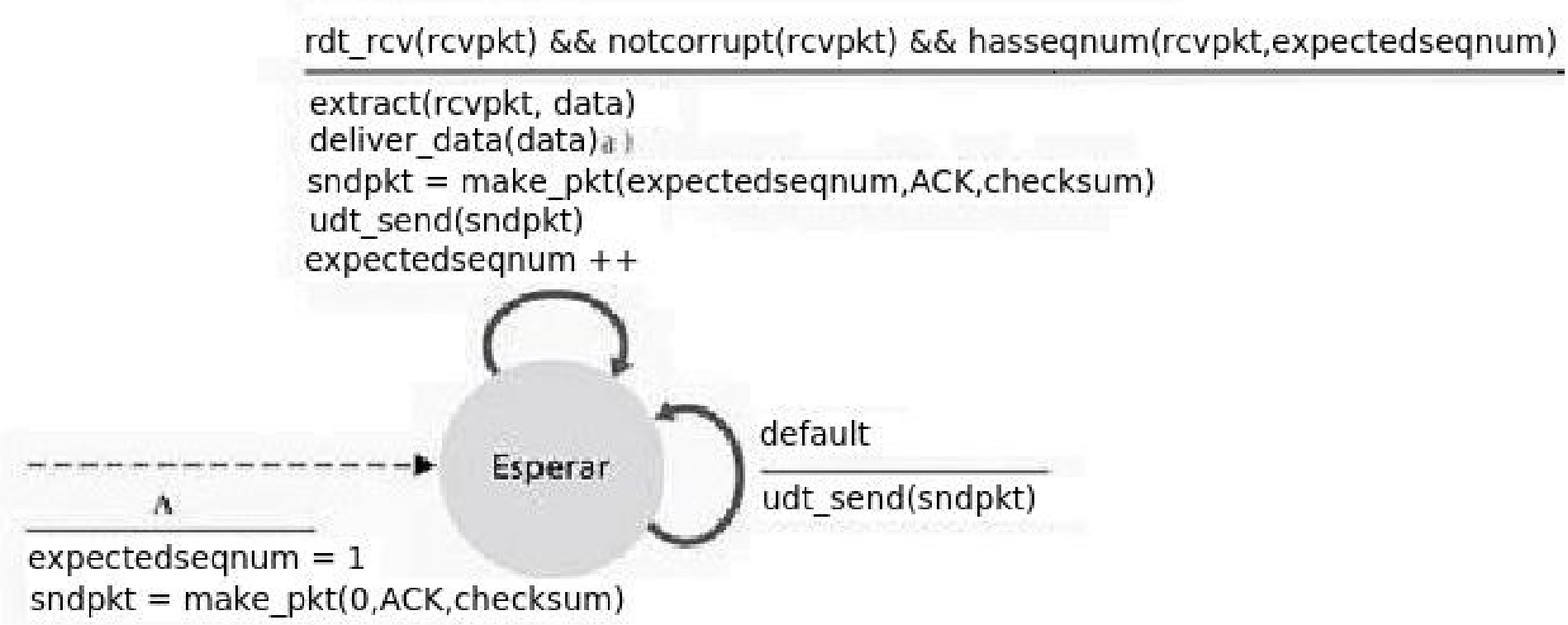
… 3.4.3 – Protocolo Go-Back-N

- FSM do Remetente no G-Back-N



... 3.4.3 – Protocolo Go-Back-N

- FSM do Destinatário no G-Back-N



… 3.4.3 – Protocolo Go-Back-N

- "**descarte de pacotes**" - pacotes que chegam fora de ordem são descartados, pois a entrega à camada superior deve ser na ordem.

- e.g., suponha que o pacote "n" seja aguardado, mas quem chega é o pacote "n+1" e considerando que os dados devem ser entregues na ordem certa, a ação padrão é o descarte do pacote "n+1" !!

- "**dúvida**" .. porque não salvar o pacote "n+1" até receber o "n" ?
- … no entanto, se "n" for perdido, os pacotes "n", "n+1", "n+2" … serão retransmitidos e, portanto, a salvaguarda de "n+1" foi em vão !!

… 3.4.3 – Protocolo Go-Back-N

- "**descarte de pacotes**" - pacotes que chegam fora de ordem são descartados, pois a entrega à camada superior deve sere na ordem.

- e.g., suponha que o pacote "n" seja esperado, mas quem chega é o pacote "n+1" e considerando que os dados devem ser entregues na ordem certa, porque não salver o pacote "n+1" até receber o "n" ?

- "**vantagem do descarte de pacotes**" .. simplicidade de manipulação de buffers no destinatário, pois não é necessário colocar no buffer nenhum pacote que esteja fora de ordem.
- "**desvantagem**" .. jogar fora um pacote recebido corretamente, pois a sua retransmissão pode ser truncada ou mesmo se perder.

... 3.4.3 – Protocolo Go-Back-N

- e.g., exemplo do protocolo GBN para janela de 04 pacotes onde o remetente envia os pacotes 0, 1, 2 e 3 e na sequência aguarda que um ou mais desses pacotes seja reconhecido.

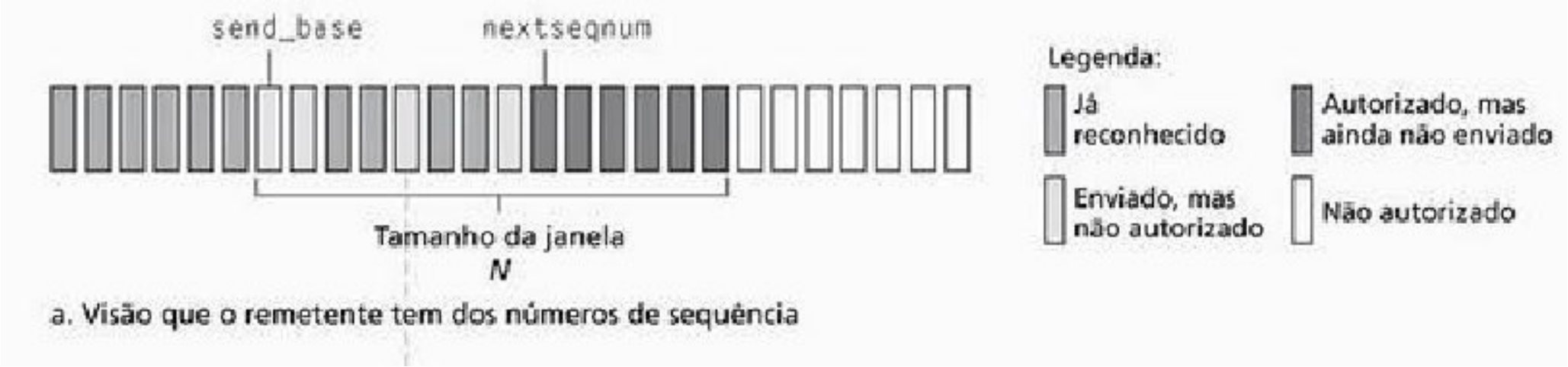
3.4.4 – Protocolo de Repetição Seletiva

- "**Go Back N**" .. remetente potencialmente transmite nro. grande de pacotes para na sequência aguardar o reconhecimento evitando-se assim problemas de subutilização de canal.
- "**problema**" .. pode-se deparar com muitos pacotes pendentes na rede quando o tamanho da janela é grande, assim como na situação em que o produto do atraso pela largura de banda é grande.

- "**Selective Repeat**" – evita retransmissões desnecessárias posto que se propõe a retransmissão somente dos pacotes suspeitos de terem sido recebidos com erro (corrompidos) ou perdidos.
- … janela de tamanho "N" limita o nro. de pacotes pendentes de reconhecimento na rede, mas alguns pacotes da janela podem já ter sido reconhecidos pelo remetente >> retransmite somente o necessário !!

… 3.4.4 – Protocolo de Repetição Seletiva

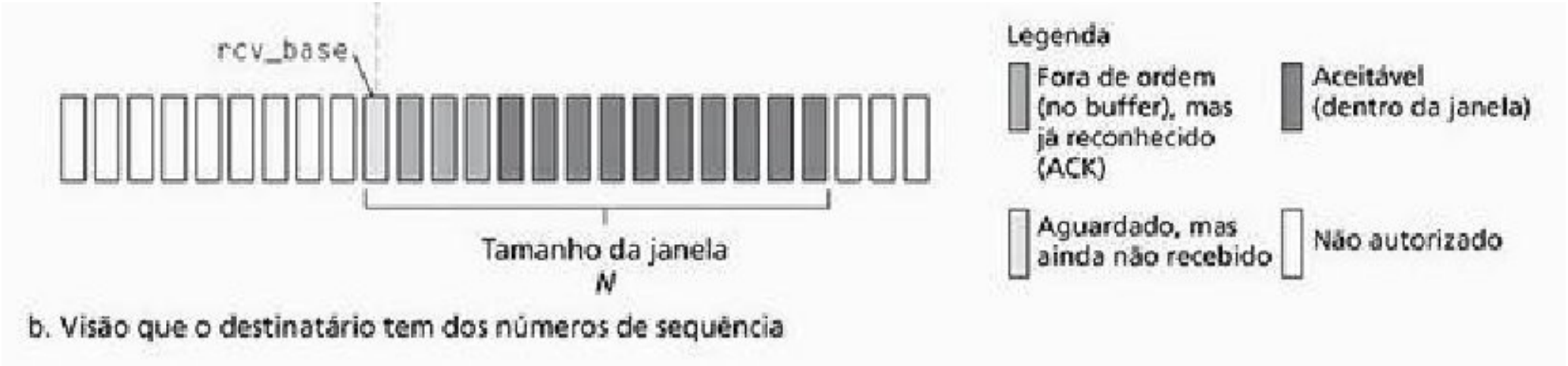
- … janela de tamanho “N” limita o nro. de pacotes pendentes não reconhecidos dentro da rede, mas alguns pacotes da janela podem já ter sido reconhecidos pelo remetente.
- … remetente já recebeu ACKs para alguns pacotes dentro da janela de envio enquanto outros pacotes ainda aguardam o reconhecimento.



a. Visão que o remetente tem dos números de sequência

… 3.4.4 – Protocolo de Repetição Seletiva

- … destinatário reconhece um pacote corretamente recebido ainda que esteja ou não na ordem que se espera receber.
- … pacotes fora de ordem ficam no buffer até que todos os pacotes faltantes sejam recebidos, quando então o conjunto de pacotes poderá ser entregue à entidade da camada superior.



b. Visão que o destinatário tem dos números de sequência

… 3.4.4 – Protocolo de Repetição Seletiva

- Açoes do Protocolo SR – Remetente:

- "**dados recebidos**" .. verifica se o próximo nro. de sequência está disponível e dentro da janela do remetente, permitindo que os dados sejam empacotados e enviados ao destinatário.
- … se nro. de sequência não está dentro da janela, dados são armazenados ou devolvidos à aplicação.
- "**temporização**" .. cada pacote tem o seu próprio "timeout" lógico, já que apenas um pacote será transmitido quando a "timeout" se esgotar.

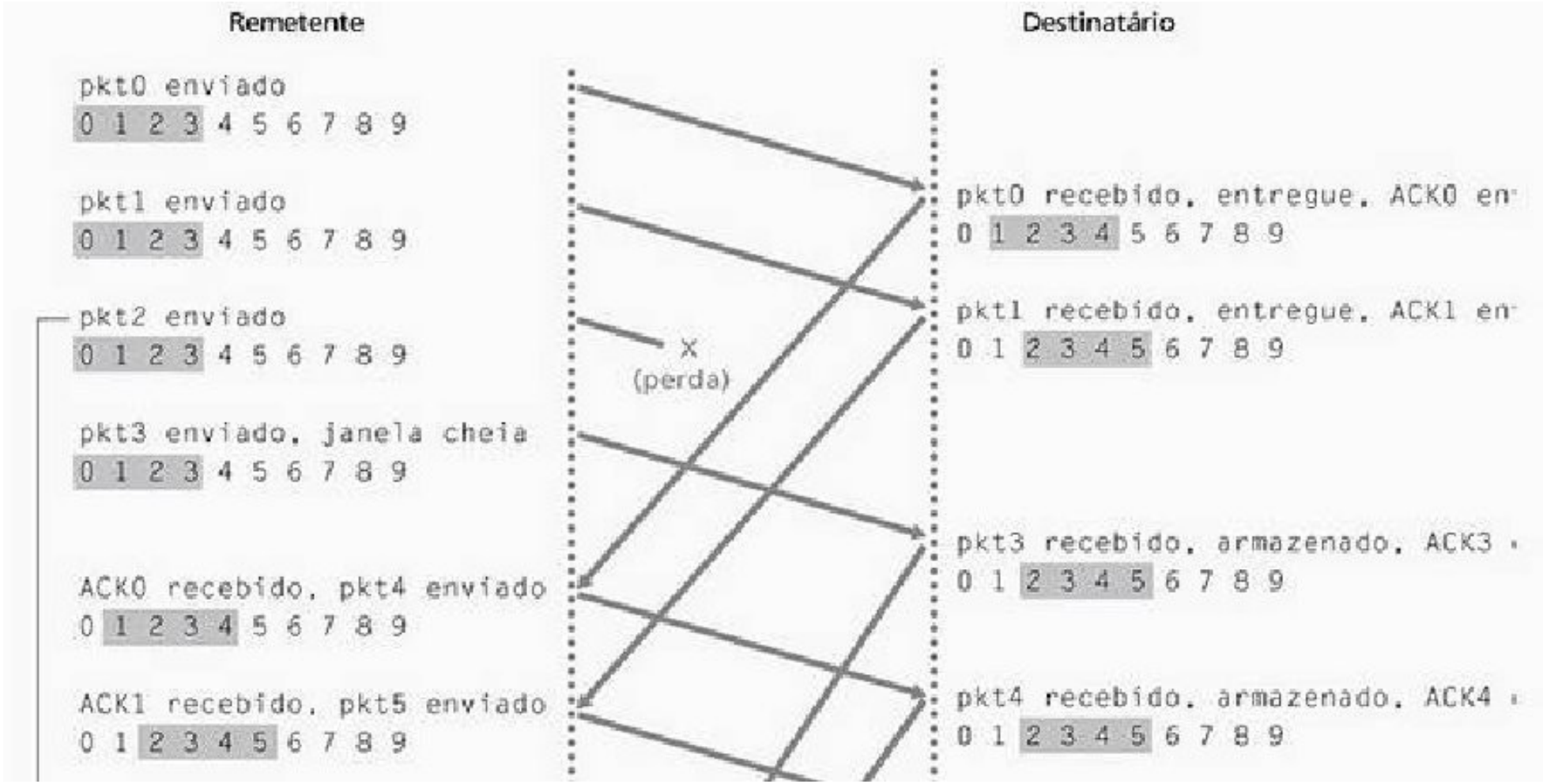
… 3.4.4 – Protocolo de Repetição Seletiva

- Açoes do Protocolo SR – Remetente:

- "**reconhecimento recebido**" - remetente marca pacote da janela quando recebe ACK, permitindo que a janela seja avançada caso o nro. de sequência seja igual ao "send_base".
- … ao deslocar a janela e caso haja pacotes não transmitidos com nro. de sequência que agora estão dentro da janela, estes pacotes podem ser transmitidos (pois "agora" estão dentro da janela).

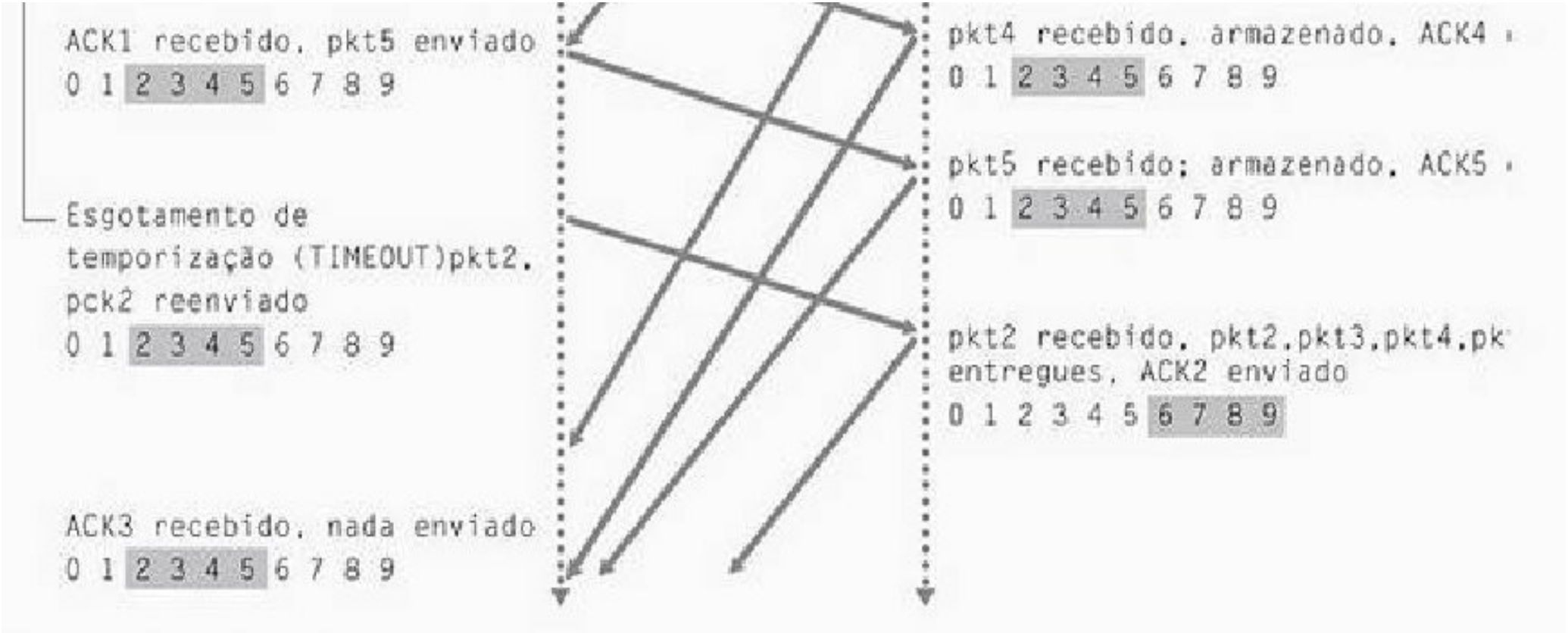
… 3.4.4 – Protocolo de Repetição Seletiva

- Operação do Protocolo SR:

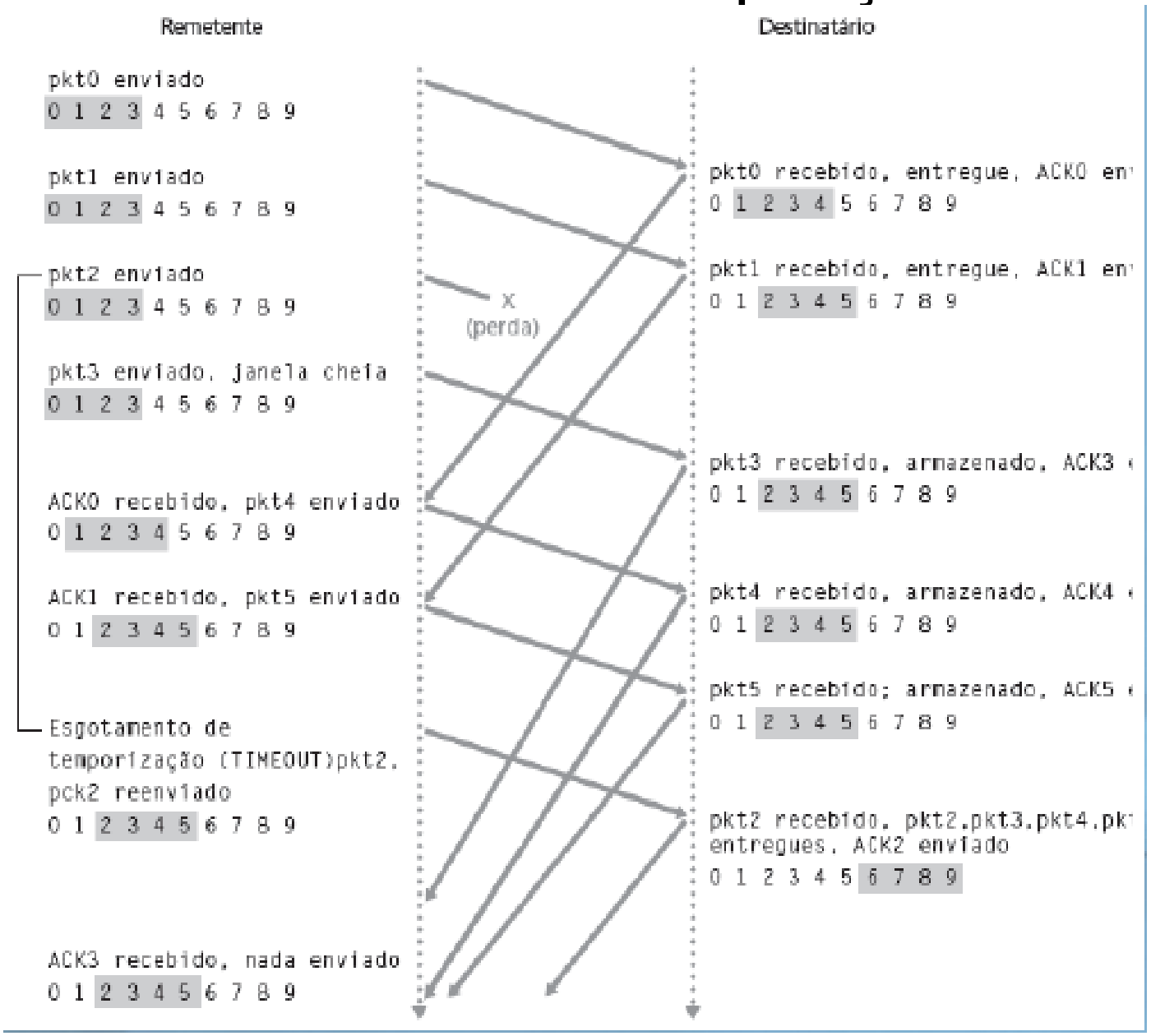


… 3.4.4 – Protocolo de Repetição Seletiva

- Obs.: remetente e destinatário nem sempre tem uma visão idêntica do que foi recebido corretamente e do que não foi, ou seja, janelas do remetente e destinatário nem sempre coincidem.



… 3.4.4 – Protocolo de Repetição Seletiva



… 3.4.4 – Protocolo de Repetição Seletiva

- Açoes do Protocolo SR – Destinatário:

- "**nro. de sequência**" entre [rcv_base, rcv_base + N – 1] recebido corretamente faz com que um ACK seletivo seja enviado ao remetente e caso não tenha sido recebido, o mesmo é armazenado.
- … se o pacote tiver nro. de sequência igual a base da janela, então ele e quaisquer outros pacotes já armazenados no buffer e numerados consecutivamente, serão entregues à aplicação >> desloca a janela.

… 3.4.4 – Protocolo de Repetição Seletiva

- Açoes do Protocolo SR – Destinatário:

- e.g., quando rcv_base = 2 é recebido, ele e os pacotes 3, 4 e 5 podem ser entregues à camada superior (aplicação).
- pacote com nro. de sequência entre [rcv_base - N, rcv_base – 1] que tenha sido recebido gera um ACK ainda que este pacote já tenha sido reconhecido anteriormente pelo destinatário.
- qualquer outro caso, simplesmente ignore !!

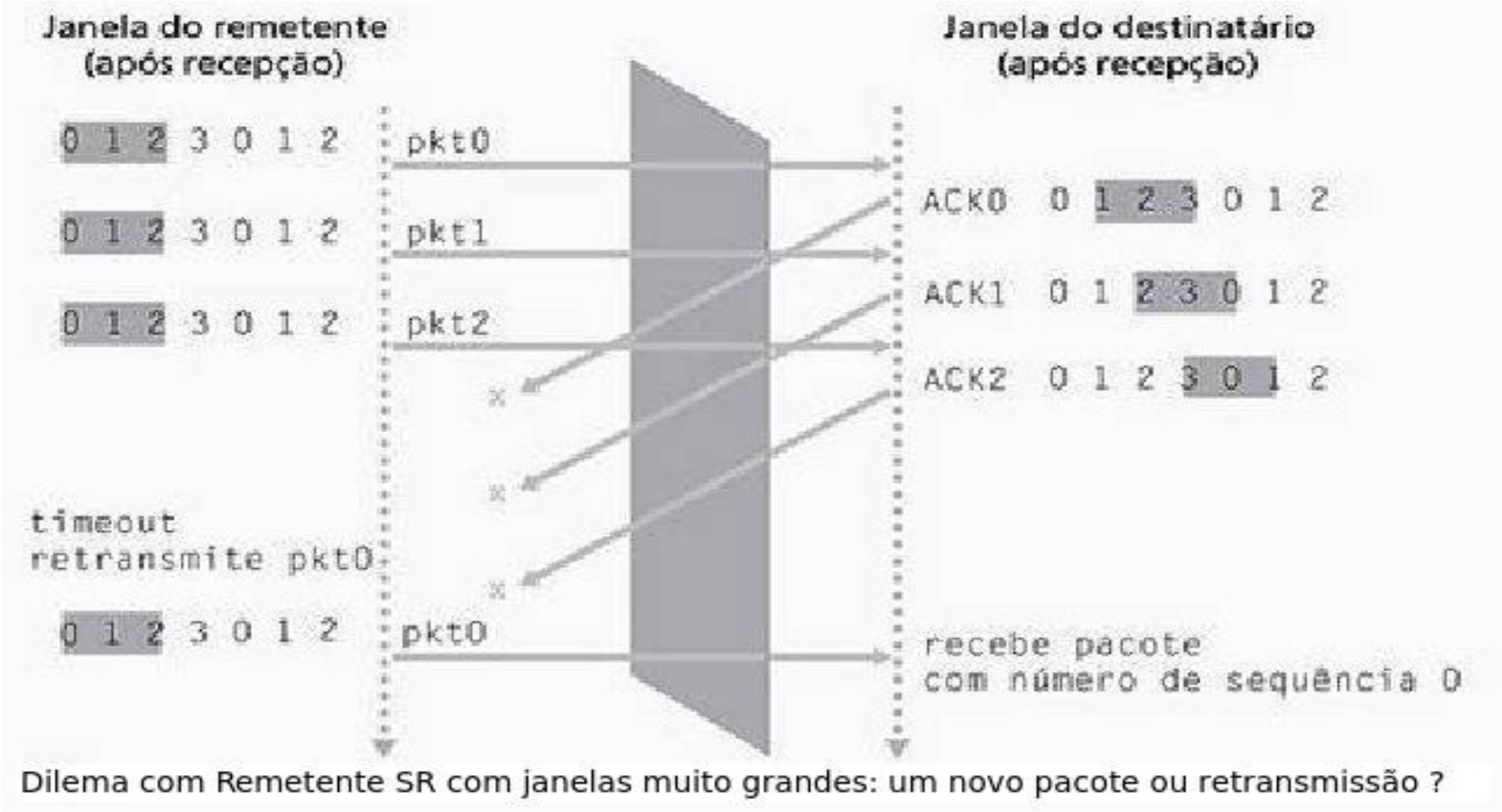
… 3.4.4 – Protocolo de Repetição Seletiva

- e.g., suponha que os pacotes 0, 1 e 2 sejam transmitidos, recebidos e reconhecidos corretamente no destinatário.
- … considere que a janela é 3 e a faixa de nro. de sequências = (0, 1, 2, 3), logo, a janela no destinatário está sobre o 4º, 5º e 6º pacotes, que têm os nro. de sequências 3, 0 e 1.

- "**problema**" .. falta de sincronização entre remetente e destinatário.
- "**falta de sincronização**" - presente quando temos faixa finita de sequência e janela do remetente e destinatário estão distantes, geram importantes consequências.

… 3.4.4 – Protocolo de Repetição Seletiva

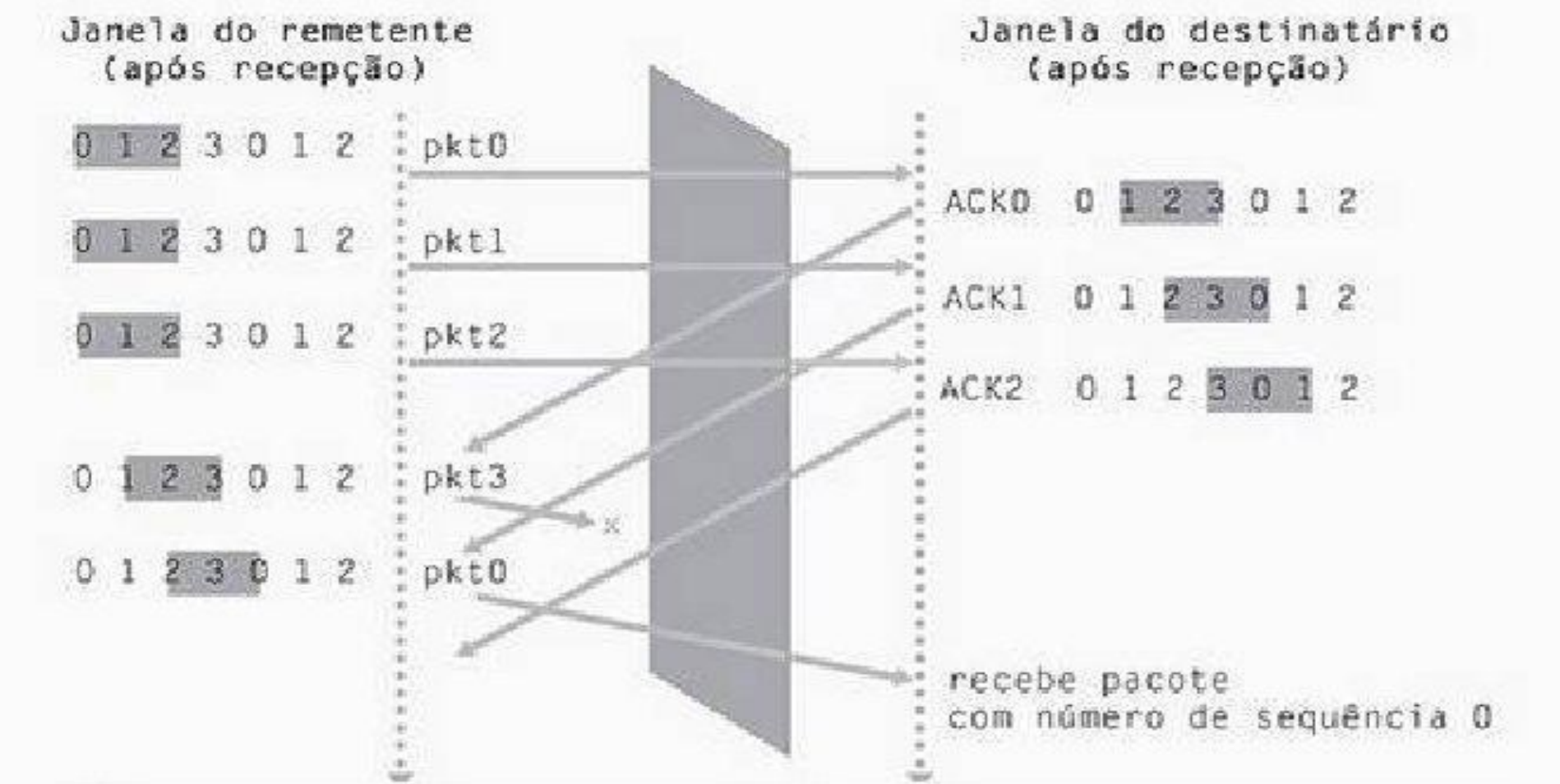
- … falta de sincronização entre as janelas ?!



Dilema com Remetente SR com janelas muito grandes: um novo pacote ou retransmissão ?

… 3.4.4 – Protocolo de Repetição Seletiva

- … falta de sincronização entre as janelas ?!



Dilema com Remetente SR com janelas muito grandes: um novo pacote ou retransmissão ?

… 3.4.4 – Protocolo de Repetição Seletiva

- … para o destinatário, os dois cenários são idênticos, pois não há como distinguir a retransmissão do 1º pacote da transmissão original do 5º pacote – ambos os cenários são idênticos.

- Qual deve ser o tamanho da Janela ?
- … tamanho da janela pode ser menor ou igual à metade do tamanho do espaço de nros. de sequências para os protocolos SR =< ½ * 2^k.

3.5 – Transporte Orientado a Conexão: TCP

- "**resumo**" .. princípios da transferência confiável de dados (timeout, nro. de sequência, reconhecimento positivo e negativo e janela).

- "**soma de verificação**" .. usada para detectar erros de bits em um segmento transmitido, exigindo o recálculo no receptor.
- "**temporizador**" .. usado para controlar a temporização/retransmissão de um segmento, possivelmente porque o segmento ou seu ACK correspondente foi perdido dentro do canal.
- .. destinatário pode receber cópias duplicadas quando ocorrer esgotamento de temporização, ou seja, o segmento está atrasado, mas não perdido (esgotamento de temporização prematuro).
- .. destinatário pode receber cópias duplicadas quando um segmento foi recebido pelo destinatário mas o ACK foi perdido.

… 3.5 – Transporte Orientado a Conexão: TCP

- "**resumo**" .. princípios da transferência confiável de dados (timeout, nro. de sequência, reconhecimento positivo e negativo e janela).

- "**número de sequência**" .. numeração sequencial de segmentos de dados entre remetente e destinatário permitindo que o destinatário detecte um segmento perdido.
- .. segmentos com números de sequência duplicados permitem que o destinatário detecte cópias duplicadas de um pacote.
- "**reconhecimento**" .. usado pelo destinatário para avisar o remetente que um segmento ou conjunto de segmentos foi recebido corretamente.
- .. reconhecimentos normalmente portam o nro. de sequência de um segmento específico (individual) ou de segmentos (cumulativo), que estão sendo reconhecidos.

… 3.5 – Transporte Orientado a Conexão: TCP

- "**resumo**" .. princípios da transferência confiável de dados (timeout, nro. de sequência, reconhecimento positivo e negativo e janela).

- "**reconhecimento negativo**" .. usado pelo destinatário para avisar o remetente que um pacote não foi recebido corretamente.
- .. reconhecimentos negativos normalmente portam o número de sequência do pacote que não foi recebido corretamente.
- "**janela**" .. remetente fica restrito a enviar segmentos com números de sequência que caiam dentro de uma determinada faixa.
- .. ao permitir que vários segmentos sejam transmitidos, ainda que não reconhecidos, a utilização do remetente pode ser aumentada em relação ao modo de operação pare e espere.

… 3.5 – Transporte Orientado a Conexão: TCP

- "**resumo**" .. princípios da transferência confiável de dados (timeout, nro. de sequência, reconhecimento positivo e negativo e janela).

Mecanismo	Uso, comentários
Soma de verificação	Usada para detectar erros de bits em um pacote transmitido.
Temporizador	Usado para controlar a temporização/retransmissão de um pacote, possivelmente porque o pacote (ou seu ACK) foi perdido dentro do canal. Como pode ocorrer esgotamento de temporização quando um pacote está atrasado, mas não perdido (esgotamento de temporização prematuro), ou quando um pacote foi recebido pelo destinatário mas o ACK remetente-destinatário foi perdido, um destinatário pode receber cópias duplicadas de um pacote.
Número de sequência	Usado para numeração sequencial de pacotes de dados que transitam do remetente ao destinatário. Lacunas nos números de sequência de pacotes recebidos permitem que o destinatário detecte um pacote perdido. Pacotes com números de sequência duplicados permitem que o destinatário detecte cópias duplicadas de um pacote.
Reconhecimento	Usado pelo destinatário para avisar o remetente que um pacote ou conjunto de pacotes foi recebido corretamente. Reconhecimentos normalmente portam o número de sequência do pacote, ou pacotes, que estão sendo reconhecidos. Reconhecimentos podem ser individuais ou cumulativos, dependendo do protocolo.
Reconhecimento negativo	Usado pelo destinatário para avisar o remetente que um pacote não foi recebido corretamente. Reconhecimentos negativos normalmente portam o número de sequência do pacote que não foi recebido corretamente.
Janela, paralelismo	O remetente pode ficar restrito a enviar somente pacotes com números de sequência que caiam dentro de uma determinada faixa. Permitindo que vários pacotes sejam transmitidos, ainda que não reconhecidos, a utilização do remetente pode ser aumentada em relação ao modo de operação pare e espere.

… 3.5 – Transporte Orientado a Conexão: TCP

- "**objetivo**" .. discutir o Protocolo TCP .. RFCs 793, 1122, 1323, 2018 e 2581, bem como perceber como os mecanismos da transferência confiável de dados estão contemplados no TCP.

Mecanismo	Uso, comentários
Soma de verificação	Usada para detectar erros de bits em um pacote transmitido.
Temporizador	Usado para controlar a temporização/retransmissão de um pacote, possivelmente porque o pacote (ou seu ACK) foi perdido dentro do canal. Como pode ocorrer esgotamento de temporização quando um pacote está atrasado, mas não perdido (esgotamento de temporização prematuro), ou quando um pacote foi recebido pelo destinatário mas o ACK remetente-destinatário foi perdido, um destinatário pode receber cópias duplicadas de um pacote.
Número de sequência	Usado para numeração sequencial de pacotes de dados que transitam do remetente ao destinatário. Lacunas nos números de sequência de pacotes recebidos permitem que o destinatário detecte um pacote perdido. Pacotes com números de sequência duplicados permitem que o destinatário detecte cópias duplicadas de um pacote.
Reconhecimento	Usado pelo destinatário para avisar o remetente que um pacote ou conjunto de pacotes foi recebido corretamente. Reconhecimentos normalmente portam o número de sequência do pacote, ou pacotes, que estão sendo reconhecidos. Reconhecimentos podem ser individuais ou cumulativos, dependendo do protocolo.
Reconhecimento negativo	Usado pelo destinatário para avisar o remetente que um pacote não foi recebido corretamente. Reconhecimentos negativos normalmente portam o número de sequência do pacote que não foi recebido corretamente.
Janela, paralelismo	O remetente pode ficar restrito a enviar somente pacotes com números de sequência que caiam dentro de uma determinada faixa. Permitindo que vários pacotes sejam transmitidos, ainda que não reconhecidos, a utilização do remetente pode ser aumentada em relação ao modo de operação pare e espere.

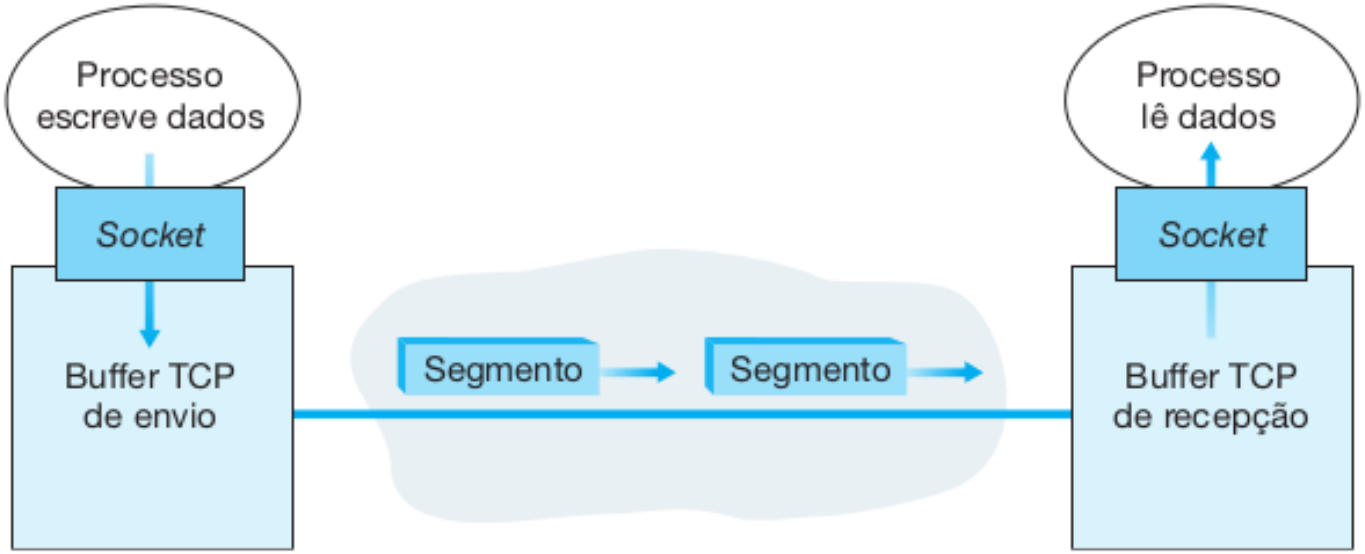
3.5.1 – Conexão TCP

- "**orientado a conexão**" - antes que um processo de aplicação possa enviar dados a outro processo, os 02 processos precisam estabelecer os parâmetros da transferência de dados.
- … não se trata de um circuito TDM ou FDM, tampouco de um circuito virtual, uma vez que o estado da conexão reside nos "hosts".
- … ou seja, roteadores intermediários são completamente alheios às conexões, pois enxergam apenas os datagramas.

- e.g., seja uma conexão entre 02 "hosts" "A" e "B", então trata-se de uma conexão ponto-a-ponto "full-duplex", ou seja, dados podem fluir de "A" para "B" ao mesmo tempo em que de "B" para "A".

… 3.5.1 – Conexão TCP

- e.g., remetente e destinatário trocam 03 segmentos especiais frequentemente denominados 03 Way Handshake (03 vias).
- … uma vez estabelecida a conexão, os “hosts” podem trocar dados um com o outro através do “socket”.
- … dados são direcionados para o “buffer” de envio, que é um dos “buffers” reservados durante a apresentação em 03 vias.



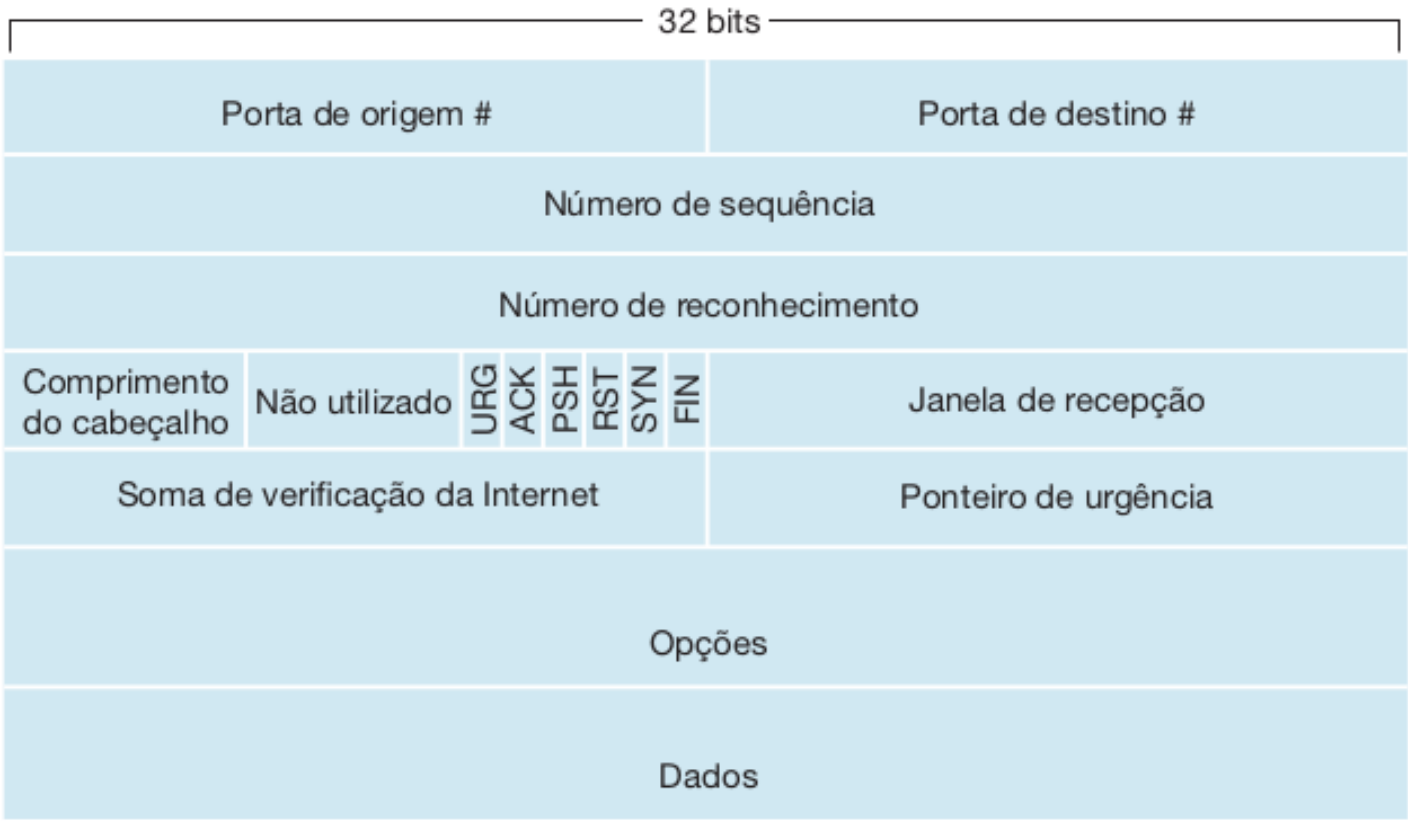
… 3.5.1 – Conexão TCP

- "**Maximum Segment Size**" – MSS … limita a quantidade máxima de dados que pode ser colocada, por vez, em um segmento.
- … normalmente estabelecido ao se determinar a MTU da Camada de Enlace, cujos valores comuns são: 1460; 536; 512 bytes.
- … protocolo combina cada porção de dados do cliente com um cabeçalho para gerar o segmento a ser repassado para a camada de rede.
- … no receptor os dados são colocados no "buffer" de recepção para serem entregues na ordem correta para a aplicação.

- Obs.: Cada lado da conexão tem seus próprios "buffers", variáveis e 01 "socket" de conexão, ou seja, IP e Port associados à "aplicação".

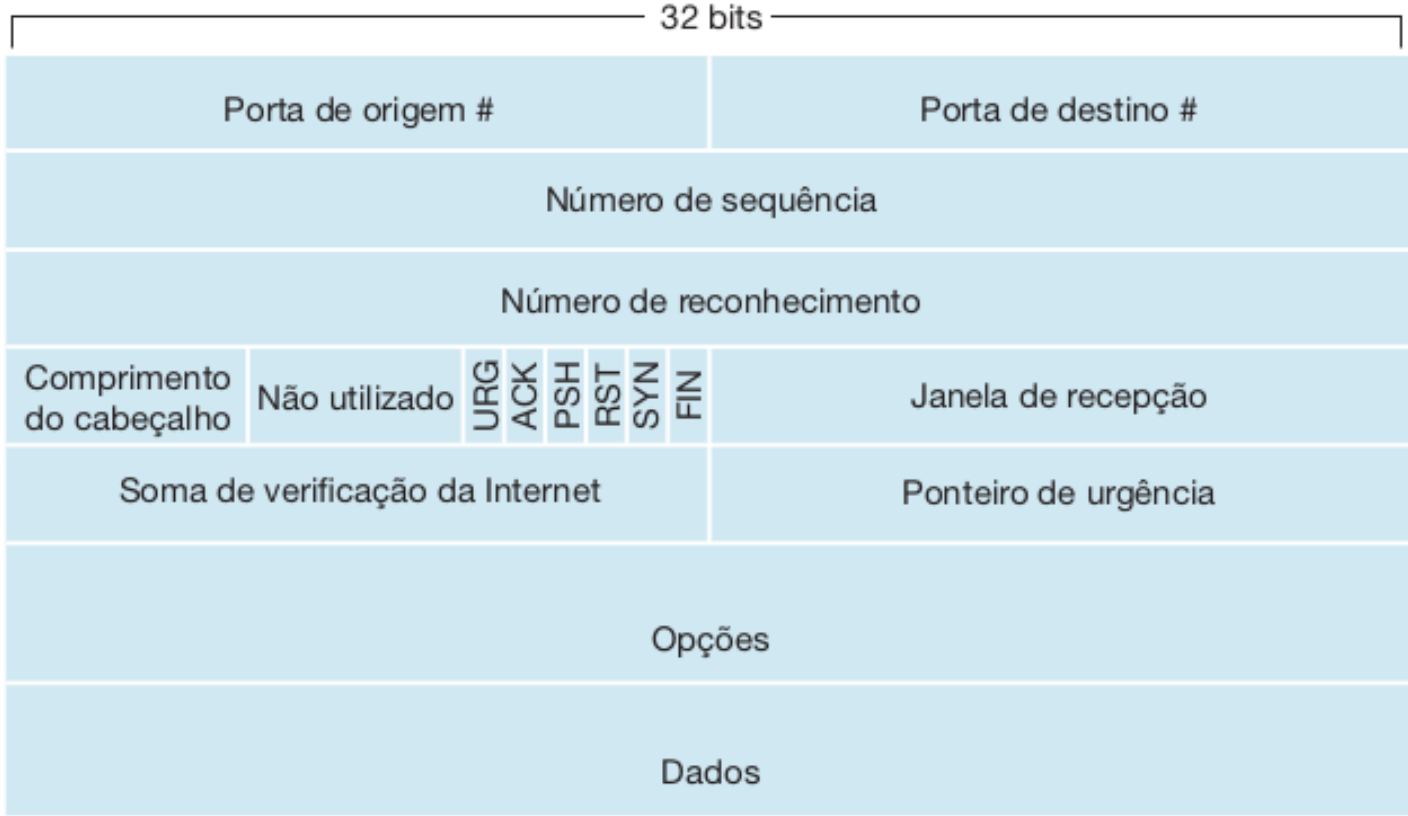
3.5.2 – Estrutura do Segmento TCP

- "**Segmento TCP**" - como acontece no UDP, o cabeçalho inclui nro. de porta de fonte e nro. de porta de destino, que são usados na multiple-xação e demultiplexação de/para aplicação.



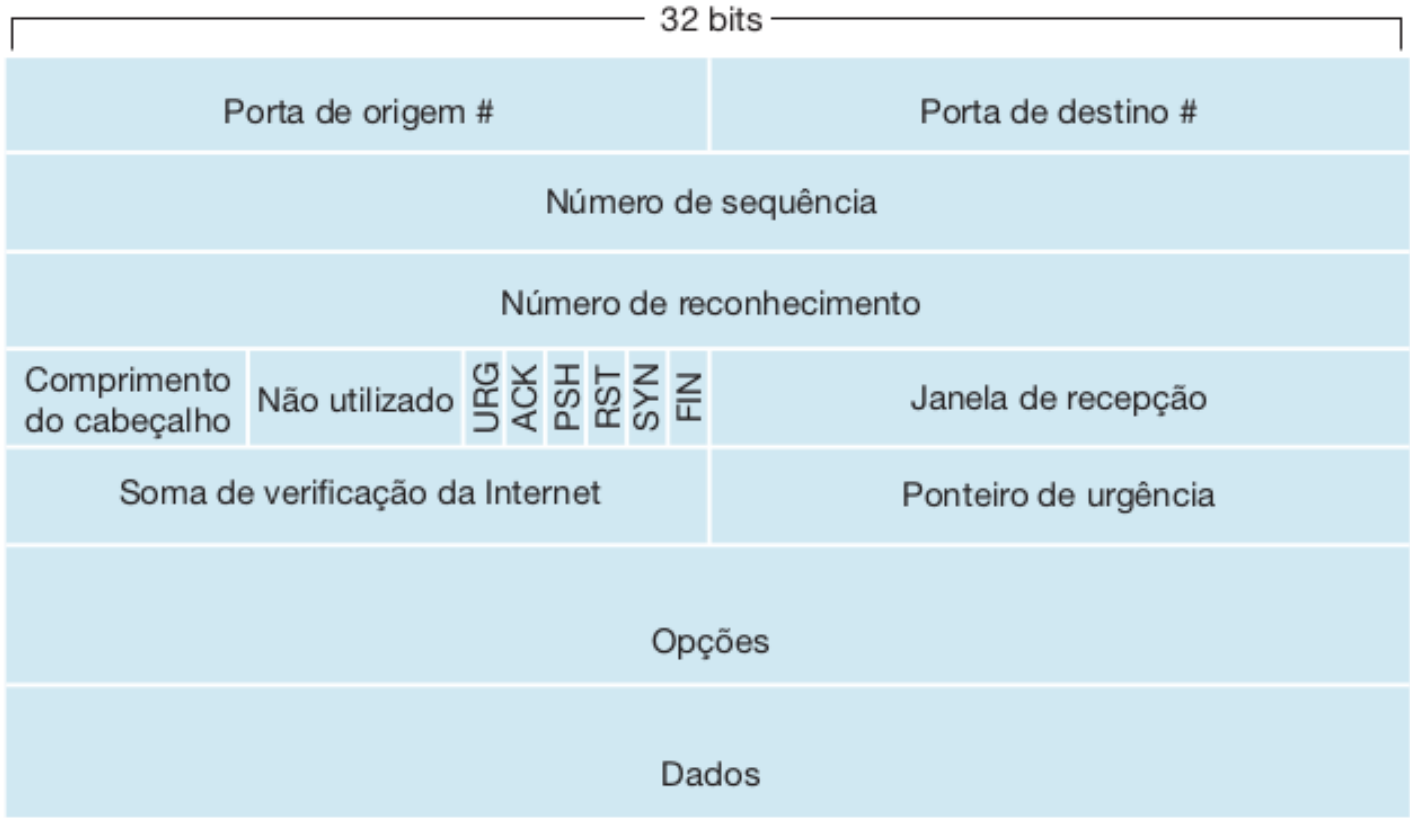
… 3.5.2 – Estrutura do Segmento TCP

- "**nro. de sequência**" (32 bits) .. usado pelo remetente e destinatário na implementação de um serviço confiável de transferência de dados.
- .. de fundamental importância, pois é nro. do 1º byte do segmento, assim, os dados são vistos como uma cadeia de bytes ordenada.



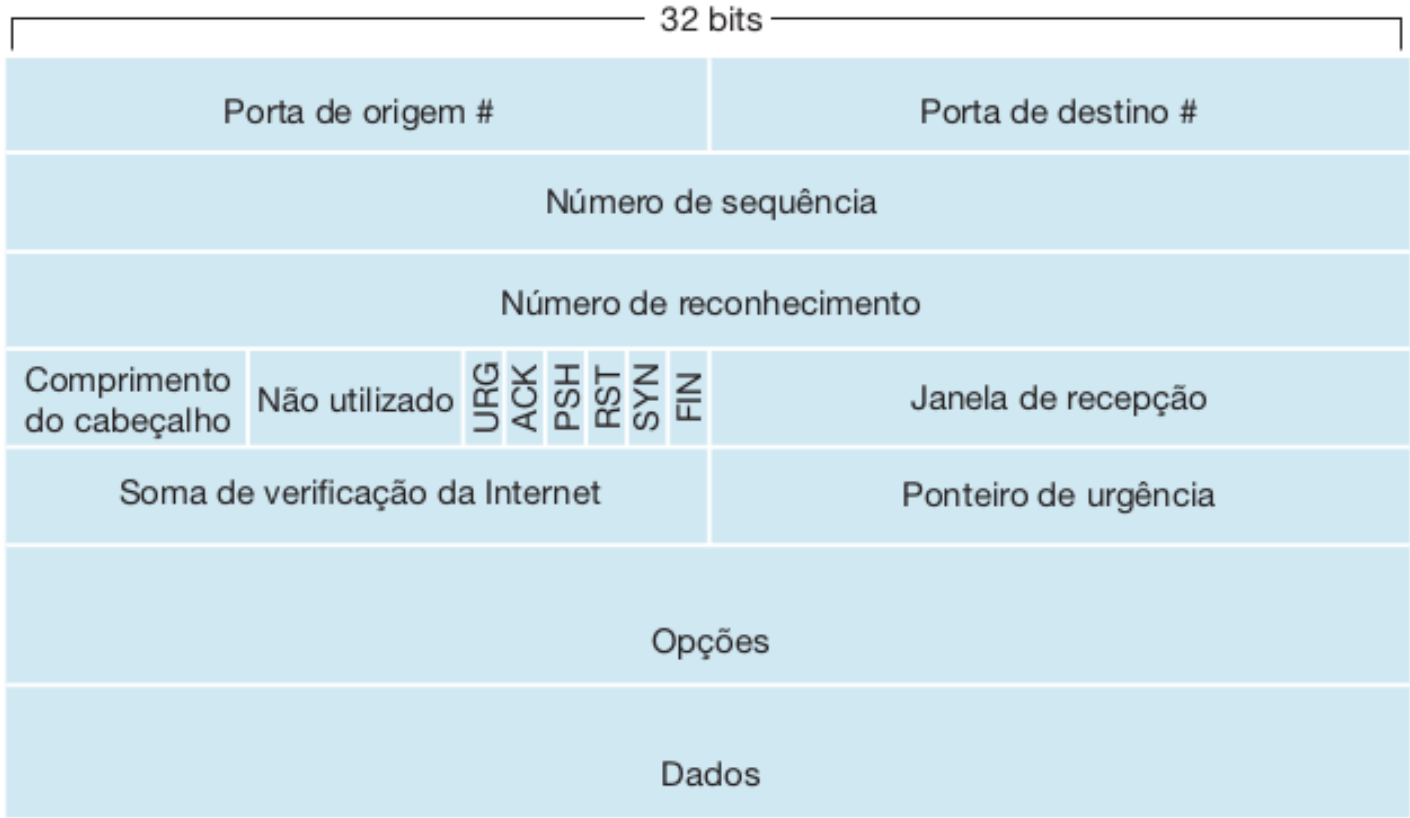
… 3.5.2 – Estrutura do Segmento TCP

- "**nro. de reconhecimento**" (32 bits) .. usado pelo remetente e destinatário na implementação de um serviço confiável de transf. de dados.
- .. de fundamental importância, pois é nro. de sequência do próximo byte que o "host" destino aguarda do "host" remetente.



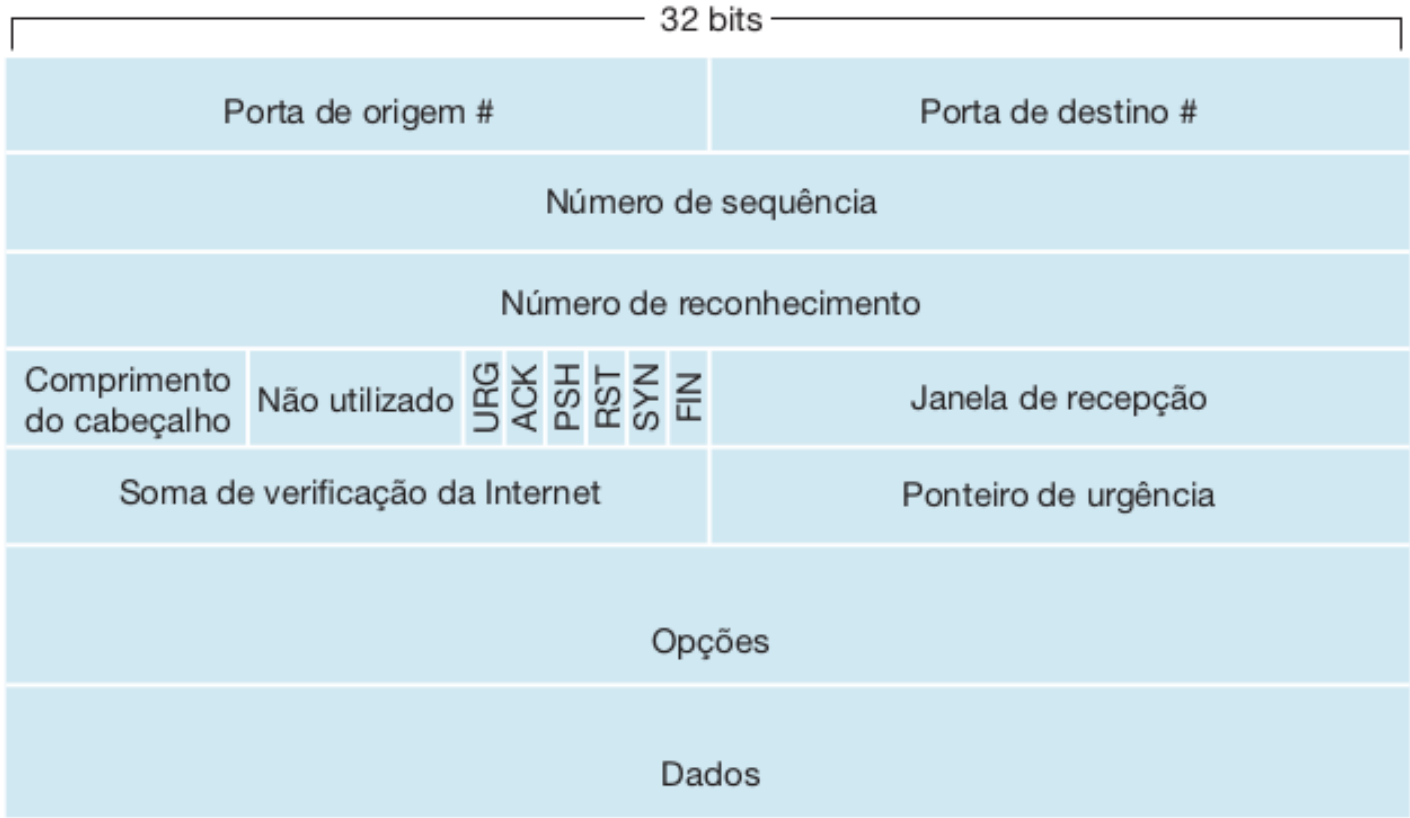
… 3.5.2 – Estrutura do Segmento TCP

- "**comprimento**" (4 bits) .. especifica o comprimento do cabeçalho em palavras de 32 bits, assim pode ter comprimento variável.
- … cabeçalho típico é de 20 bytes, mas pode conter até 60 bytes.



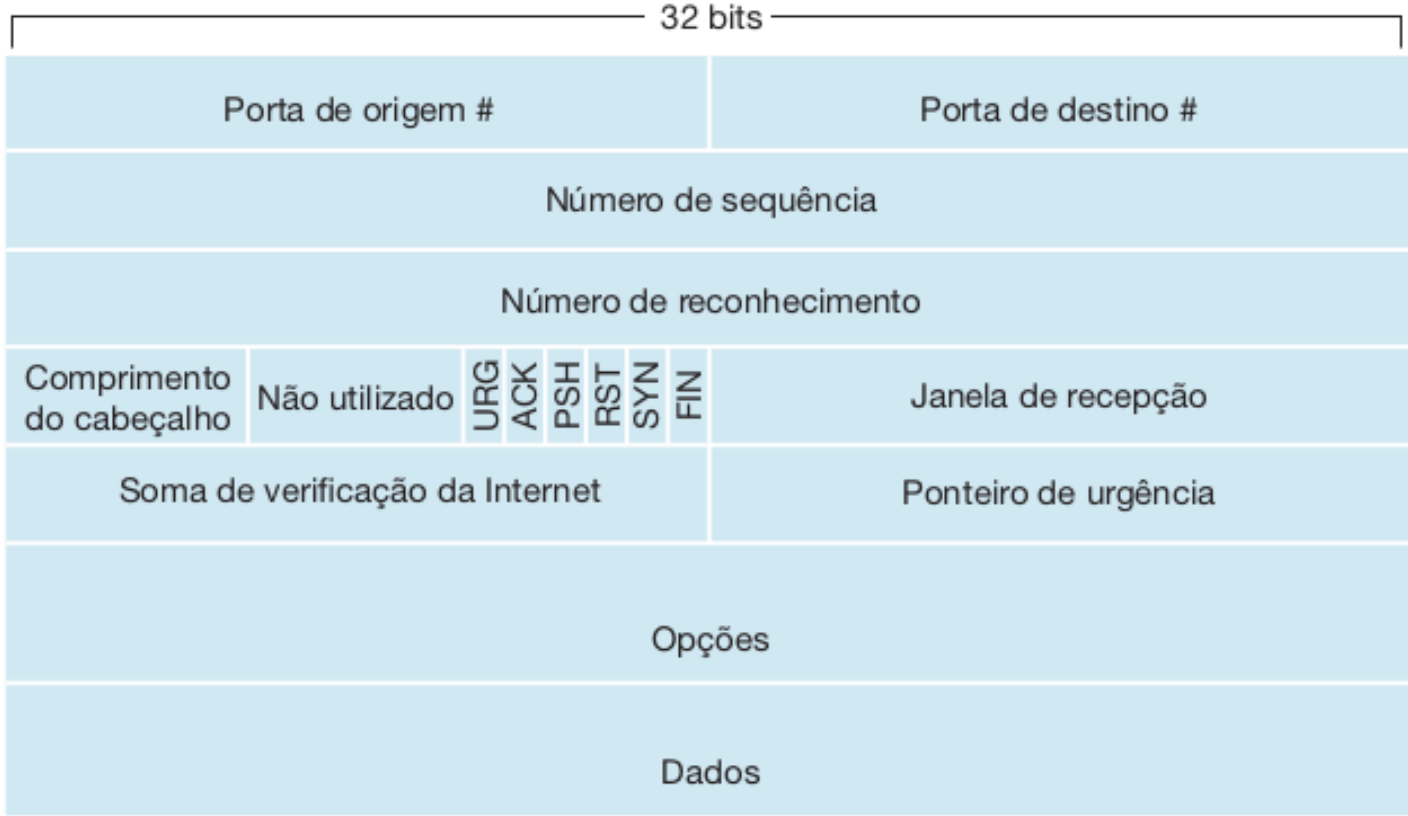
… 3.5.2 – Estrutura do Segmento TCP

- "**opções**" - opcional e de comprimento variável, é utilizado quando um remetente e um destinatário negociam MSS ou como fator de aumento de escala de janela (redes de alta velocidade).



… 3.5.2 – Estrutura do Segmento TCP

- "**flags**" .. ACK valida o campo de reconhecimento, enquanto que o RST, SYN e FIN indicam estabelecimento e encerramento de conexão.
- PSH .. repasse imediato dos dados à camada superior, enquanto URG .. indica que há dados no segmento marcados como urgentes.



… 3.5.2 – Estrutura do Segmento TCP

- e.g., suponha que uma cadeia de bytes consista de um arquivo de 500.000 bytes e que o MSS seja de 1000 bytes.
- … são construídos 500 segmentos a partir desta cadeia de bytes, numerados com 0, 1000, 2000, 3000, … 499.000.
- … cada um destes nros. de sequência é inserido no campo nro. de sequência no cabeçalho do segmento correspondente.

… 3.5.2 – Estrutura do Segmento TCP

- e.g., suponha que um “host” “A” tenha recebido de um “host” “B” um segmento contendo os bytes de 0 a 535 e outro segmento contendo os bytes de 900 a 1000;
- “**fato**” .. por alguma razão o “host” “A” ainda não recebeu os bytes de 536 a 899, ou seja, “A” ainda está esperando pelo segmento 536 para recriar a cadeia de dados de “B”.

- TCP provê “Reconhecimento Cumulativo”, ou seja, reconhecimento de bytes até o primeiro byte que estiver faltando na cadeia.

… 3.5.2 – Estrutura do Segmento TCP

- e.g., suponha que um “host” “A” inicie uma sessão com o “host” “B”, assim “A” é rotulado de cliente e “B” de servidor usando o “telnet”.
- “**idéia**” .. esta aplicação ilustra muito bem nros. de sequência e de reconhecimento, ao se operar um dispositivo remotamente.

- “**telnet**” - aplicação interativa que define um protocolo da camada de aplicação utilizado para “login” remoto que opera sobre o TCP.

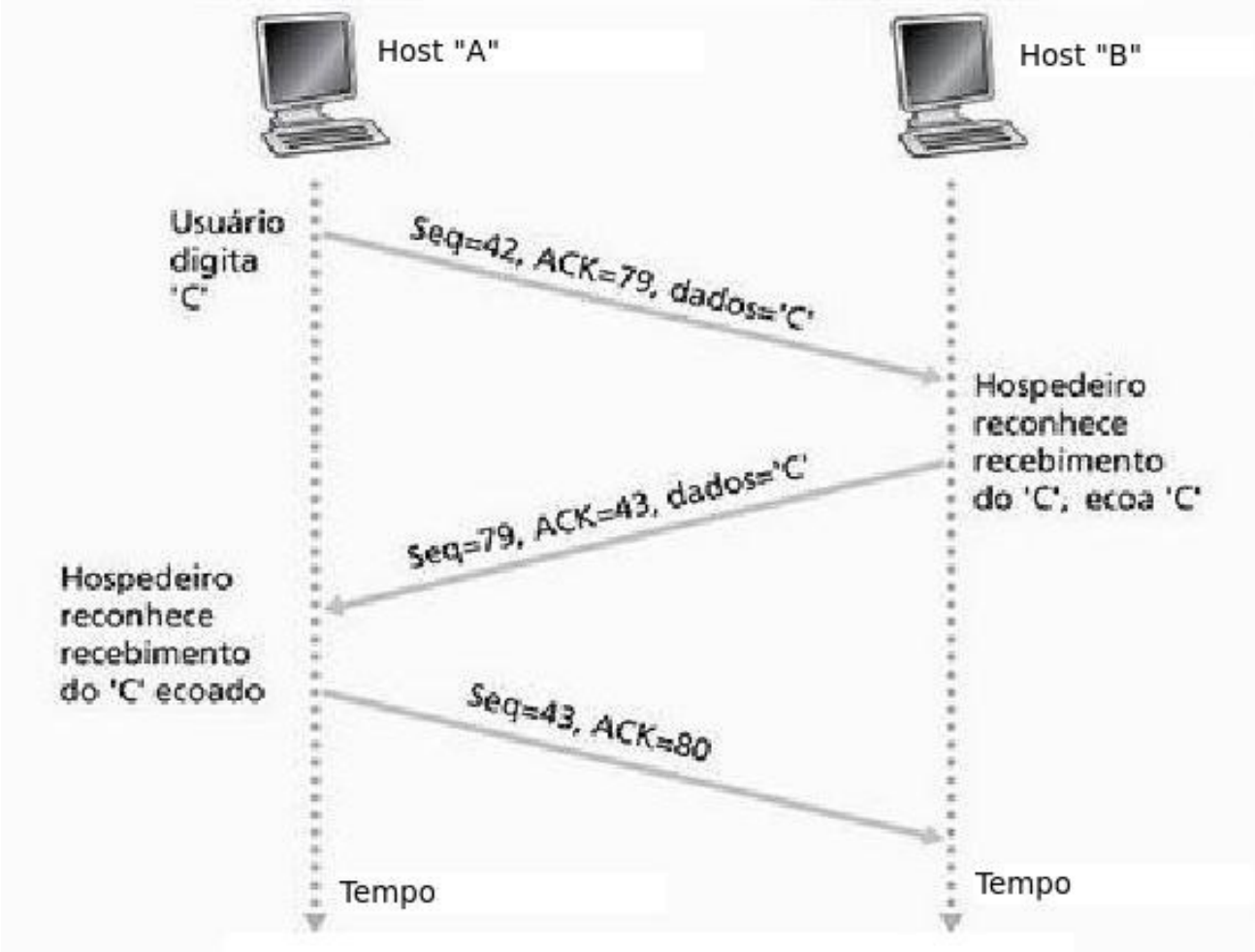
… 3.5.2 – Estrutura do Segmento TCP

- e.g., suponha que um “host” “A” inicie uma sessão com o “host” “B”, assim “A” é rotulado de cliente e “B” de servidor.
- “**idéia**” .. .. esta aplicação ilustra muito bem nros. de sequência e de reconhecimento, ao se operar um dispositivo remotamente.

- … cada caractere digitado pelo usuário no cliente será enviado ao servidor, que por sua vez devolverá um cópia do mesmo ao cliente para que seja apresentado na tela “telnet” do usuário.
- … assim, cada caractere atravessa a rede 02 vezes entre o momento que o usuário aperta o teclado e o momento em que o caractere é apresentado em seu monitor (usuário).

… 3.5.2 – Estrutura do Segmento TCP

- … seja a sequência de segmentos entre cliente e servidor e nros. de sequência iniciais para cliente e servidor iguais a 42 e 79.



3.5.3 – Estimativa RTT e Temporização

- TCP assim como o RDT 3.0 utiliza um mecanismo de controle de temporização / retransmissão para recuperar segmentos perdidos.
- .. neste contexto, como estes mecanismos de controle de temporização e congestionamento são implementados em um protocolo real ?!
- … evidentemente, o intervalo de controle deve ser maior que o tempo para o envio do segmento e seu reconhecimento na conexão (RTT).

- “**conclusão**” … surge então a necessidade de se estimar o RTT !!

… 3.5.3 – Estimativa RTT e Temporização

- “**SampleRTT**” **-** tempo transcorrido entre o momento do envio de um segmento e o momento em que o segmento de reconhecimento do segmento transmitido é recebido.
- “**Round Trip Time**” .. variável de segmento para segmento devido ao congestionamento nos roteadores e variações de carga nos “hosts”.

… 3.5.3 – Estimativa RTT e Temporização

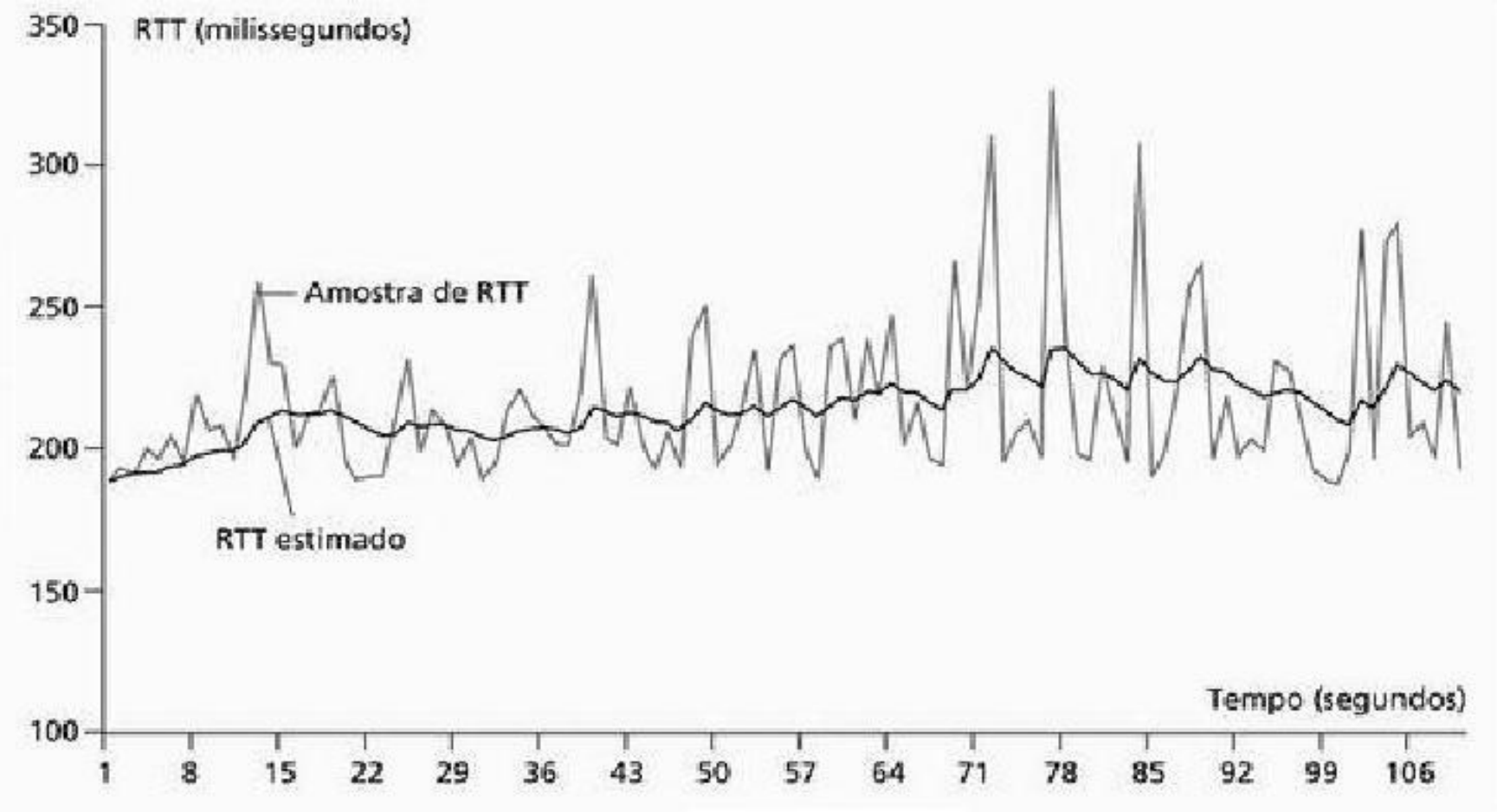
- SampleRTT - … como sofrem variação de seg. para seg. devido ao congestionamento nos roteadores e carga nos “hosts”, o TCP estima o “Round Trip Time” tendo por base a amostra RTT ...
- EstimatedRTT = (1 – alfa) * EstimatedRTT + alfa * SampleRTT

  … onde a RFC2988 recomenda “alfa” = 0.125
- EstimatedRTT = 0.875 * EstimatedRTT + 0.125 * SampleRTT

- Obs.: … em estatística este tipo de média é denominada Média Móvel Exponencial Ponderada (MMEP)

… 3.5.3 – Estimativa RTT e Temporização

- “SampleRTT” e “EstimatedRTT” para um conexão entre os “hosts” “gaia.cs.umass.edu” e “fantasia.eurocom.fr”



… 3.5.3 – Estimativa RTT e Temporização

- Além da medida “EstimatedRTT” é valioso a medida de variabilidade ou variação do RTT (Desvio RTT);
- DevRTT = (1 – beta) * DevRTT + beta * | SampleRTT – EstimatedRTT |

  … onde a RFC2988 recomenda “beta” = 0.25
- DevRTT = 0.75 * DevRTT + 0.25 * | SampleRTT – EstimatedRTT |

- DevRTT é igual a Média Móvel Exponencial Ponderada (MMEP) da diferença entre SampleRTT e EstimatedRTT.
- pouca variação em SampleRTT acarreta pouca variação em DevRTT.
- muita variação em SampleRTT acarreta grande variação em DevRTT.

3.5.4 – Transf. Confiável de Dados

- "**serviço da camada de rede**" .. assim como datagramas IP podem transbordar dos buffers dos roteadores e jamais alcançar seu destino, os mesmos podem ser corrompidos bem como chegar fora de ordem.
- "**impacto na camada de transporte**" … como segmentos da camada de transporte são carregados pela rede em datagramas, eles também podem sofrer os mesmos problemas !!

- "**serviço de transf. confiável de dados**" **-** garante que a cadeia de dados que um processo lê no seu "buffer" de reconhecimento ...

  ** não seja corrompido;

  ** não tenha duplicações;

  ** não tenha lacunas;

  ** esteja na sequência.

… 3.5.4 – Transf. Confiável de Dados

- "**timeout**" … discutimos anteriormente a proposta de temporizador adicional para cada segmento transmitido mas ainda não reconhecido.
- "**problema**" .. implementação do gerenciamento de temporizadores pode exigir considerável sobrecarga.

- RFC2988 – … procedimentos para gerenciamento de temporizadores utilizam apenas 01 temporizador de transmissão, mesmo que haja vários segmentos transmitidos ainda não reconhecidos.
- .. são 03 eventos para tratar: i) dados recebidos da aplicação acima; ii) esgotamento do temporizador; e iii) recebimento de ACK.

… 3.5.4 – Transf. Confiável de Dados

- "**premissa**" .. remetente não é compelido pelo controle de fluxo ou de congestionamento, além disso o tamanho dos dados vindos de cima é menor que o MSS e transf. de dados ocorre apenas em uma direção.
- NextSeqNum = InitialSeqNumber
- SendBase = InitialSeqNumber
- forever(loop)
- switch(event)
- event: "**data received from the application above**"
- event: "**timer timeout**"
- event: "**ACK received with ACK field value of "y"**"
- end of switch(event)
- enf of forever(loop)

… 3.5.4 – Transf. Confiável de Dados

- "**premissa**" .. remetente não é compelido pelo controle de fluxo ou de congestionamento, além disso o tamanho dos dados vindos de cima é menor que o MSS e transf. de dados ocorra apenas em uma direção.
- …
- event: "**data received from the application above**"
- create TCP segment with sequence number NextSeqNum
- if(timer currently not running) start timer;
- pass segment to IP
- NextSeqNum = NextSeqNum + length(data)
- break;

… 3.5.4 – Transf. Confiável de Dados

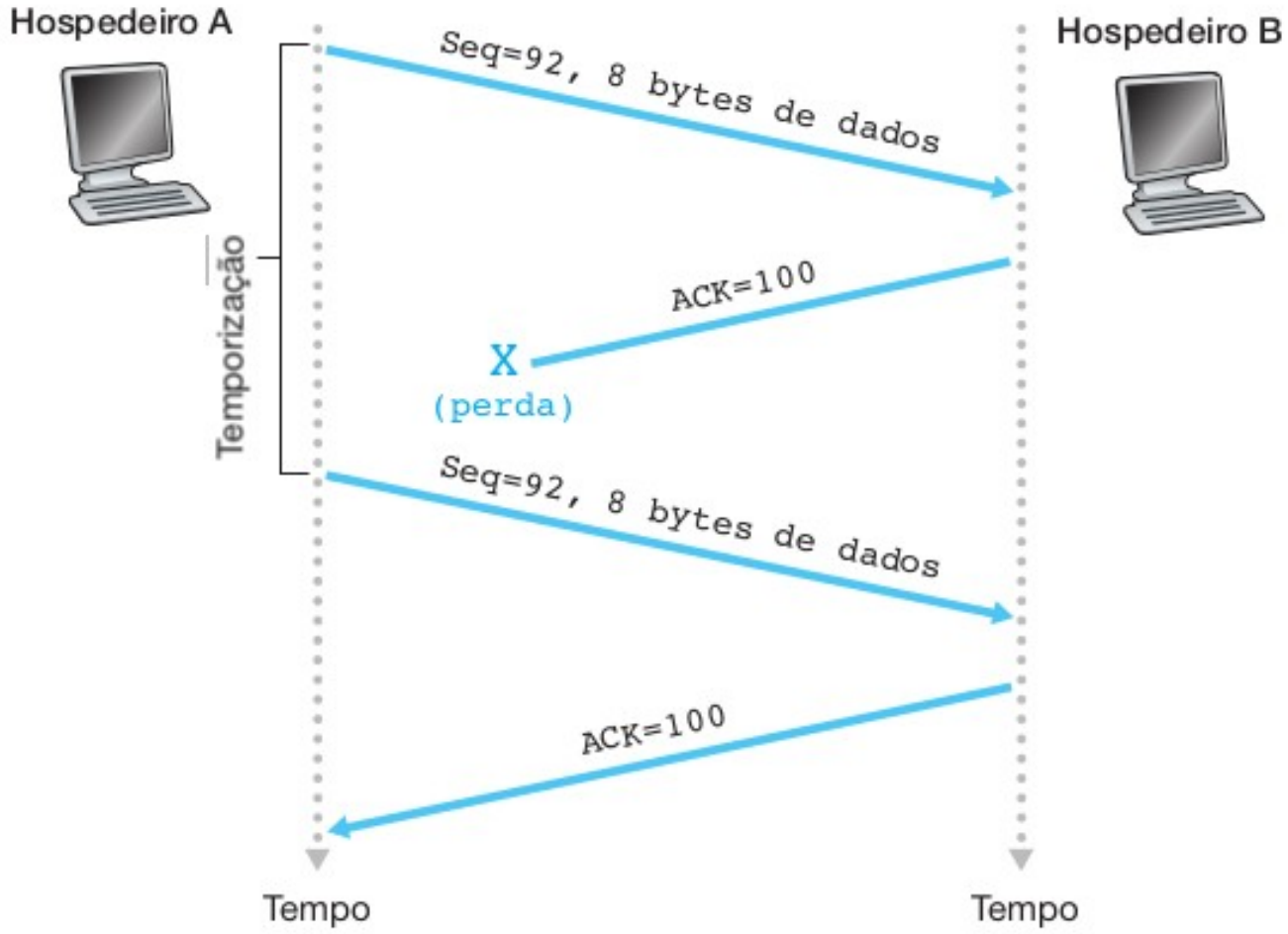
- "**premissa**" .. remetente não é compelido pelo controle de fluxo ou de congestionamento, além disso o tamanho dos dados vindos de cima é menor que o MSS e transf. de dados ocorra apenas em uma direção.
- …
- /* TimeoutInterval = calculado a partir do EstimatedRTT e DevRTT */
- event: "**timer timeout**"
- retransmit not-yet-acknowledged segment with smallest sequence number;
- start timer
- break;
- ...

… 3.5.4 – Transf. Confiável de Dados

- "**premissa**" .. remetente não é compelido pelo controle de fluxo ou de congestionamento, além disso o tamanho dos dados vindos de cima é menor que o MSS e transf. de dados ocorra apenas em uma direção.
- …
- /* SendBase – nro. de sequência do mais antigo byte não reconhecido. */
- event: "**ACK received with ACK field value of "y"**"
- if("y" > SendBase) {
- SendBase = y
- if(there are currently any not-yet-acknowledged segments) start timer;
- }
- end of switch(event)
- end of forever(loop)

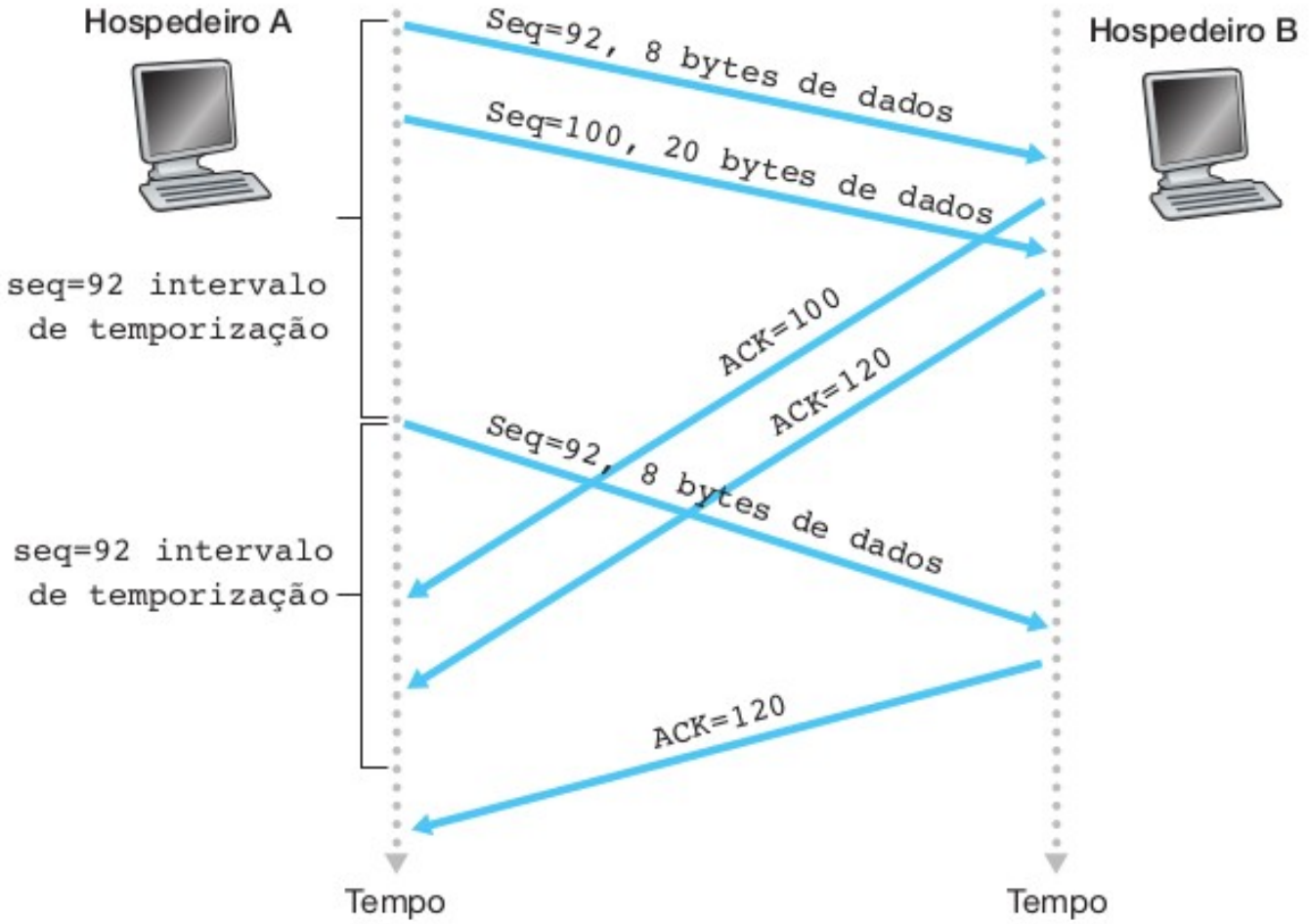
… 3.5.4 – Transf. Confiável de Dados

- e.g. #01 .. “A” envia segmento a “B”, mas ACK correspondente é perdido e com o “timeout”, tem-se uma retransmissão.



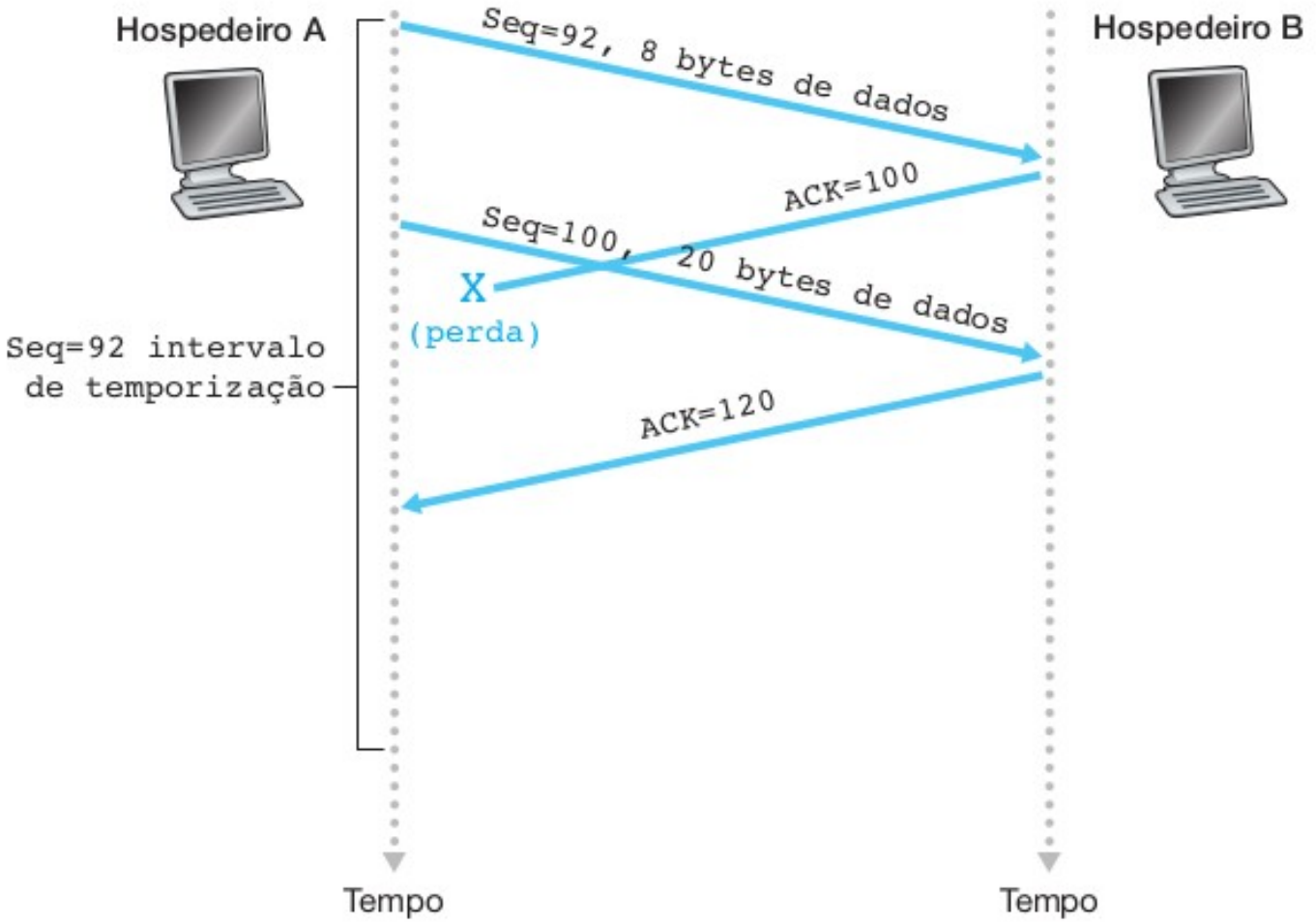
… 3.5.4 – Transf. Confiável de Dados

- e.g. #02 .. “A” envia 02 segmentos a “B”, mas seg. 92 é retransmitido em razão de “timeout” fruto de ACK atrasado.



… 3.5.4 – Transf. Confiável de Dados

- e.g. #03 .. “A” envia 02 segmentos a “B”, mas ACK do seg. 100 se perde, enquanto que o ACK do seg. 120 é recebido antes do “timeout”.



… 3.5.4 – Transf. Confiável de Dados

- "**duplicação do tempo de expiração**" … sempre que um evento de "timeout" se faz presente, o TCP retransmite o seg. ainda não reconhecido que tenha o menor nro. de sequência.
- … mas a cada retransmissão, há o ajuste do próximo tempo de expiração para o dobro do valor anterior em vez de derivá-lo.

- "**estratégia**" .. minimizar o nro. de retranmissões, estimando e recalculando a todo momento o RTT, pois é a base para o "timeout".

… 3.5.4 – Transf. Confiável de Dados

- e.g., suponha que “TimeOutInterval” associado ao mais antigo seg. ainda não reconhecido seja 0,75 segundos no momento em que o temporizador expirar pela primeira vez.
- … retransmite esse segmento e ajusta o novo “timeout” para 1,5 segundos, ou seja, 2 * 0,75 (tempo que expirou anteriormente).
- … se expirar novamente, o protocolo retransmite novamente o segmento e ajusta o tempo de “timeout” para 3,0 segundos.

… 3.5.4 – Transf. Confiável de Dados

- … mas sempre que o temporizador é iniciado após qualquer um dos outros 02 eventos, ou seja, “dados recebidos da aplicação” ou “reconhecimento recebido”, o “TimeoutInterval” seja derivado.
- “**TimeoutInterval**” .. derivado dos valores mais recentes de “EstimatedRTT” e “DevRTT” (variação do EstimatedRTT).

… 3.5.4 – Transf. Confiável de Dados

- "**expiração do temporizador**" .. causa mais provável é o congestionamento da rede, ou seja, nro. grande de pacotes chegando em roteadores no caminho entre a fonte e o destino.
- … causa descarte de pacotes e/ou longos atrasos em fila, logo se a fonte continuar a retransmitir pacotes persistentemente durante o congestionamento, o congestionamento da rede pode piorar.
- "**estratégia**" … agir educadamente, ou seja, cada requerente retransmite após intervalos de tempo cada vez mais longos.

… 3.5.4 – Transf. Confiável de Dados

- "**análise**"… quando um segmento é perdido, esse longo período força o remetente a atrasar o reenvio do seg. perdido, mas por outro lado a retransmissão aumenta o atraso fim-a-fim.
- … felizmente, o remetente pode detectar perda de segmentos bem antes de ocorrer o evento de expiração, observando os ACKs duplicados (ACK que reconhece um seg. já reconhecido).
- … quando destinatário recebe um seg. com um nro. de sequência que é maior do que o nro. de seq. subsequente, esperado e na ordem, ele detecta uma lacuna na corrente de dados.
- … esta lacuna pode ser o resultado de seg. perdidos ou reordenados dentro da rede … "**mas ainda assim o que fazer**" ?!

… 3.5.4 – Transf. Confiável de Dados

- … esta lacuna pode ser o resultado de seg. perdidos ou reordenados dentro da rede … "**mas ainda assim o que fazer**" ?!
- … como não é possível utilizar o reconhecimento negativo, TCP reconhece novamente o último byte de dados que recebeu na ordem.

- … se remetente receber 03 ACKs duplicados para os mesmos dados, ele tomará isso como indicação de que o seg. que se seguiu ao segmento reconhecido 03 vezes foi perdido.
- "**retransmissão rápida**" **-** retransmissão do segmento antes que o temporizador do mesmo segmento expire e para o qual o remetente recebeu 03 ACKs ou ACKs Duplicados (sinônimo).

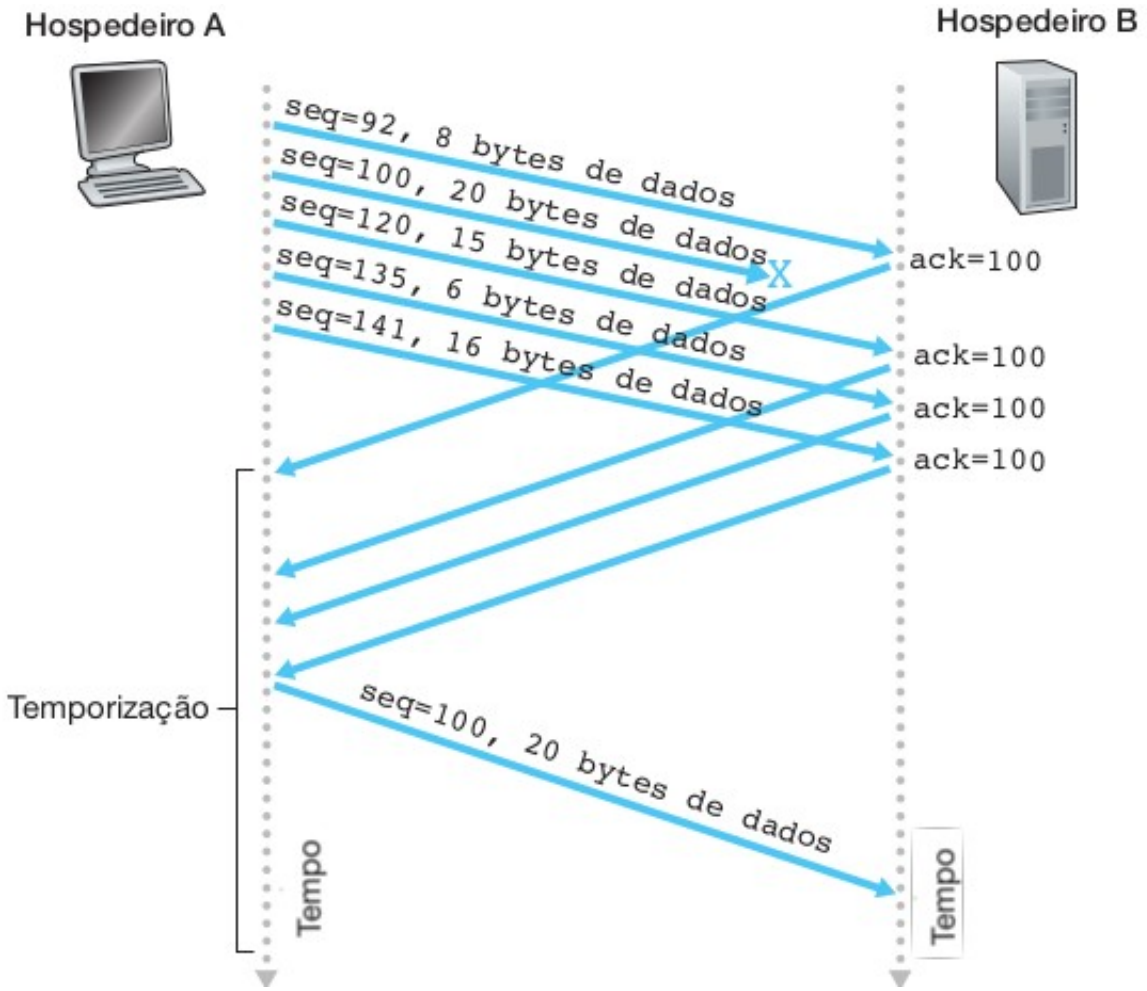
… 3.5.4 – Transf. Confiável de Dados

- "**retransmissão rápida**" - retransmissão do segmento antes que o temporizador do mesmo segmento expire e para o qual o remetente recebeu 03 ACKs ou ACKs Duplicados (sinônimo).

```
evento: ACK recebido, com valor do campo ACK y
      if (y > SendBase) {
            SendBase=y
            if (atualmente ainda não há segmentos reconhecidos)
                  iniciar temporizador
            }
      else { /* um ACK duplicado para segmento já reconhecido */
            incrementar número de ACKs duplicados recebidos para y
            if (número de ACKS duplicados recebidos para y==3)
                  /* retransmissão rápida do TCP */
                  reenviar segmento com número de sequência y
            }
      break;
```

… 3.5.4 – Transf. Confiável de Dados

- "**retransmissão rápida**" - retransmissão do segmento antes que o temporizador expire e para o qual o remetente recebeu 03 ACKs.



… 3.5.4 – Transf. Confiável de Dados

- … esta lacuna pode ser o resultado de seg. perdidos ou reordenados dentro da rede … "**mas ainda assim o que fazer**" ?!
- … como não é possível utilizar o reconhecimento negativo, TCP reconhece novamente o último byte de dados que recebeu na ordem.

Evento	Ação do TCP Destinatário
Chegada de seg. na ordem com nro. de sequência esperado Todos os dados até o nro. de seq. esperado já reconhecidos.	ACK retardado. Espera até 500 mili segundos pela chegada de um outro seg. na ordem. Se o seg. seguinte na ordem não chegar nesse intervalo, envia um ACK.
Chegada de seg. na ordem com nro. de sequência esperado. Um outro seg. na ordem esperando por transmissão de ACK	Envio imediato de um único ACK cumulativo, reconhecendo ambos os segmentos na ordem.
Chegada de um segmento fora de ordem com nro. de seq. mais alto do que o esperado. Lacuna detectada.	Envio imediato de um ACK duplicado, indicando nro. de seq. do byte seguinte esperado (extremidade mais baixa da lacuna).
Chegada de um segmento que preenche, parcial ou completamente, a lacuna nos dados recebidos.	Envia imediato de um ACK, contanto que o segmento comece na extremidade baixa da lacuna.

… 3.5.4 – Transf. Confiável de Dados

- “Go-Back-N” ou “Repetição Seletiva” !?
- TCP contempla ACKs cumulativos, no entanto, para segmento corretamente recebido e fora de ordem não há ACK individual do destinatário.

- Há semelhanças com o Go-Back-N ?!
- “**semelhança**” - … remetente precisa lembrar-se do nro. de sequência do byte seguinte a ser enviado (NextSeqNum) e do menor nro. de sequência de um byte transmitido mas não reconhecido (SendBase);
- “**diferença**” - … por outro lado muitas implementações armazenam segmentos recebidos corretamente, mas fora de ordem (Go-Back-N não! mas se parece com o “Seletive Repeat”)

… 3.5.4 – Transf. Confiável de Dados

- Há semelhanças com o Repetição Seletiva ?!
- e.g., considere o envio de uma sequência de segmentos “1, 2, …, N” pelo remetente em que todos os segs. chegam na sequência e sem erro (dentro das temporizações) exceto um dos segmento “n” < “N”;

  (1) TCP retransmite no máximo 01 segmento, ou seja, segmento “n”.

  (2) TCP pode não retransmitir caso ACK de “n+1” chegue ao remetente antes do final da temporização para o segmento “n”.

- “**modificação**” - permite que o destinatário reconheça segmentos fora de ordem, em vez de apenas reconhecer cumulativamente o último segmento recebido corretamente e na ordem (SR genérico ?!).

3.5.5 – Controle de Fluxo

- "**buffer de dados**" - … armazena dados que foram recebidos corretos e em sequência no destinatário.
- … processo de aplicação irá ler os dados a partir deste "buffer", mas não necessariamente no momento em que são recebidos.
- … dependendo da velocidade com que o remetente encaminha os dados, o "buffer" de recepção pode ser saturado !!

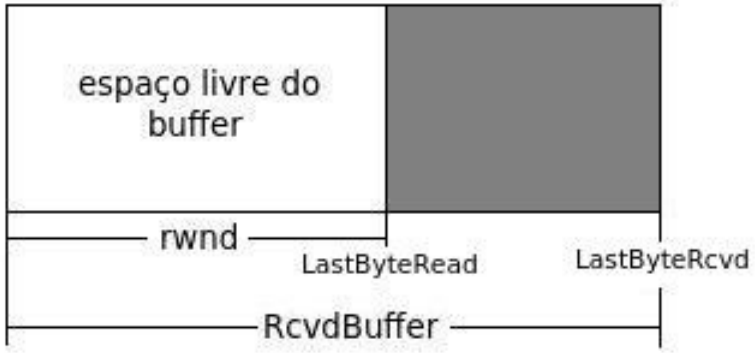
… 3.5.5 – Controle de Fluxo

- "**controle de fluxo**" - elimina a possibilidade do remetente estourar ou saturar o "buffer" de recepção do destinatário;
- … para isto, mantém-se no remetente a "janela de recepção", ou seja, variável que dá ao remetente a idéia de espaço livre no "buffer" disponível no destinatário e acordado no estabelecimento da conexão.

… 3.5.5 – Controle de Fluxo

- Como prover Controle de Fluxo através de “rwnd” ?!
- … remetente informa ao destinatário quanto espaço ele tem no “buffer” de conexão colocando o valor corrente de “rwnd” no campo de janela de recepção de cada segmento que envia ao destinatário.

Host B - aloca Buffer de Recepção (RcvBuffer)



TCP não tem permissão para saturar o buffer alocado, então:
LastByteRcvd - LastByteRead <= RcvdBuffer

Janela de Recepção (RcvWindow) contempla o espaço livre:
rwnd = RcvBuffer - [LastByteRcvd - LastByteRead]

LastByteRead = nro do último byte da cadeia de dados lidos do buffer pelo processo de aplicação.

LastByteRcvd = nro do último byte da cadeia de dados que chegou da rede e foi colocado no buffer de recepção

… 3.5.5 – Controle de Fluxo

- Como prover Controle de Fluxo através de “rwnd” ?!
- … remetente informa ao destinatário quanto espaço ele tem no “buffer” de conexão colocando o valor corrente de “rwnd” no campo de janela de recepção de cada segmento que envia ao destinatário.

- … remetente monitora as variáveis “LastByteSent” e “LastByteAcked”, cuja diferença é a quantidade de dados não reconhecidos;
- … mantendo-se a quantidade de dados não reconhecidos < “rwnd”, remetente tem certeza que não transbordará o buffer de recepção no destinatário, ou seja, LastByteSent - LastByteAcked <= rwnd

3.5.6 – Gerenciamento de Conexão TCP

- "**lembrete**" .. remetente e destinatário só podem trocar segmentos de dados após o estabelecimento de conexão.
- "**estabelecimento de conexão**" .. pressupõe a inicialização de variáveis como nros. de sequência do remetente e destinatário, "buffers", informações de controle de fluxo (RcvWindow), etc.

- "**cliente**" .. inicia conexão:
  Socket clientSocket = new Socket("hostname", "port #");

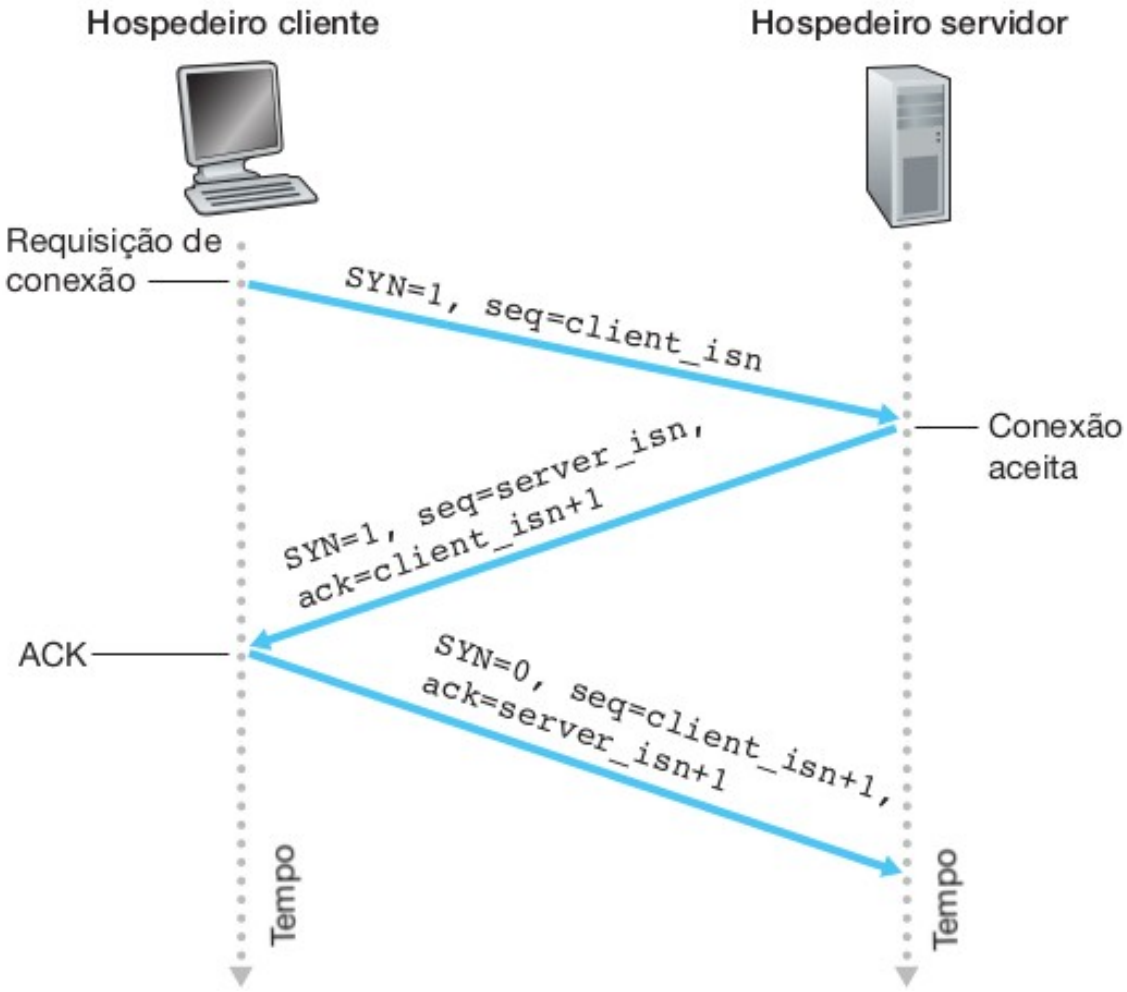
- "**servidor**" .. contactado pelo cliente:
  Socket connectionSocket = welcomeSocket.accept( );

… 3.5.6 – Gerenciamento de Conexão TCP

- "**estabelecer conexão**" .. remetente e destinatário trocam 03 segmentos frequentemente denominados "3 Way Handshake".
- "**etapa #1**" .. cliente envia segmento SYN ao servidor no qual especifica nro. de sequência inicial a ser utilizado (não envia dados).
- "**etapa #2**" .. servidor recebe segmento SYN e responde com um SYNACK no qual especifica nro. de sequência inicial a ser utilizado.
- … servidor aloca "buffers" de recepção e de envio, mas não trafega dados neste segmento SYNACK.
- "**etapa #3**" .. cliente recebe SYNACK e responde com segmento ACK que pode ou não conter dados propriamente ditos.

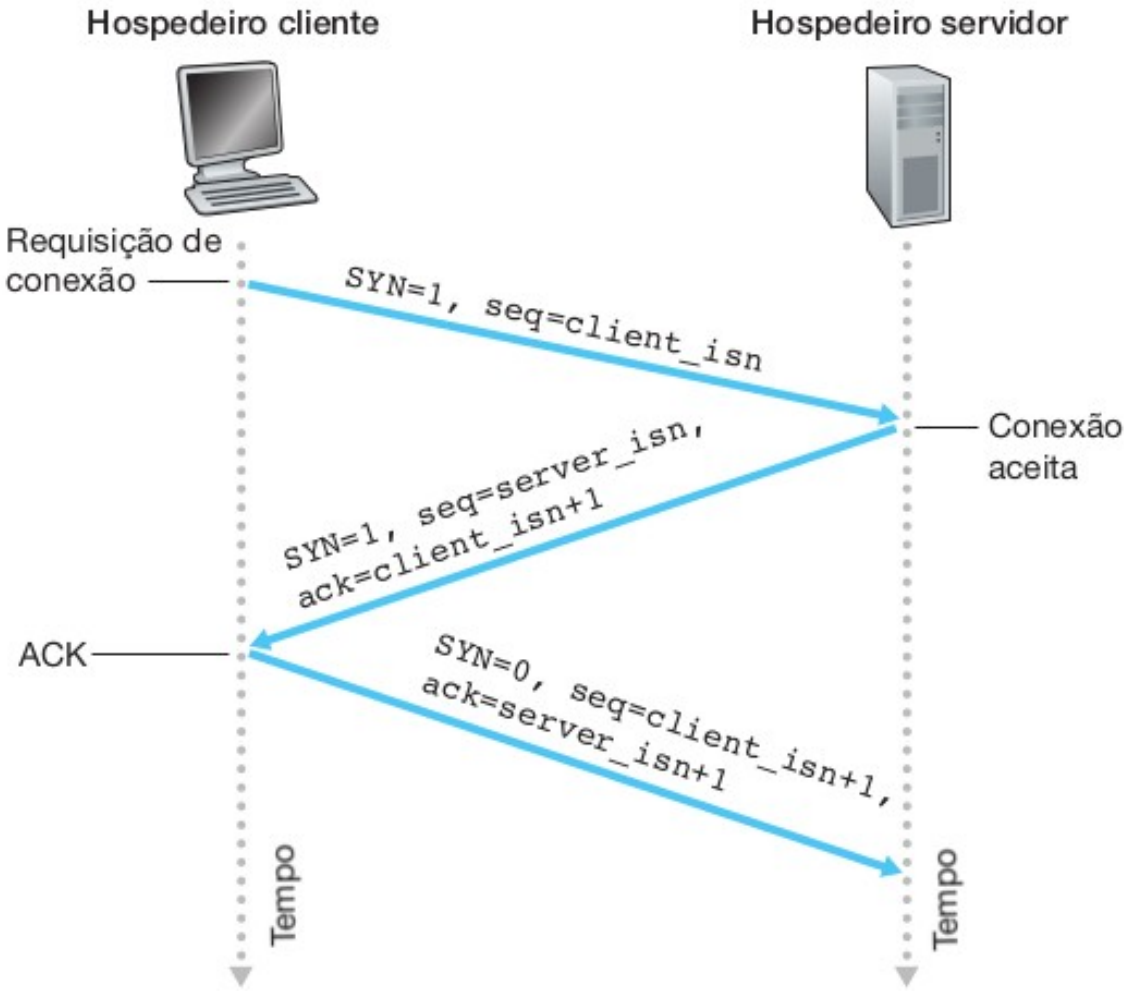
… 3.5.6 – Gerenciamento de Conexão TCP

- "**etapa #1**" .. cliente envia segmento SYN ao servidor no qual especifica nro. de sequência inicial a ser utilizado (não envia dados);



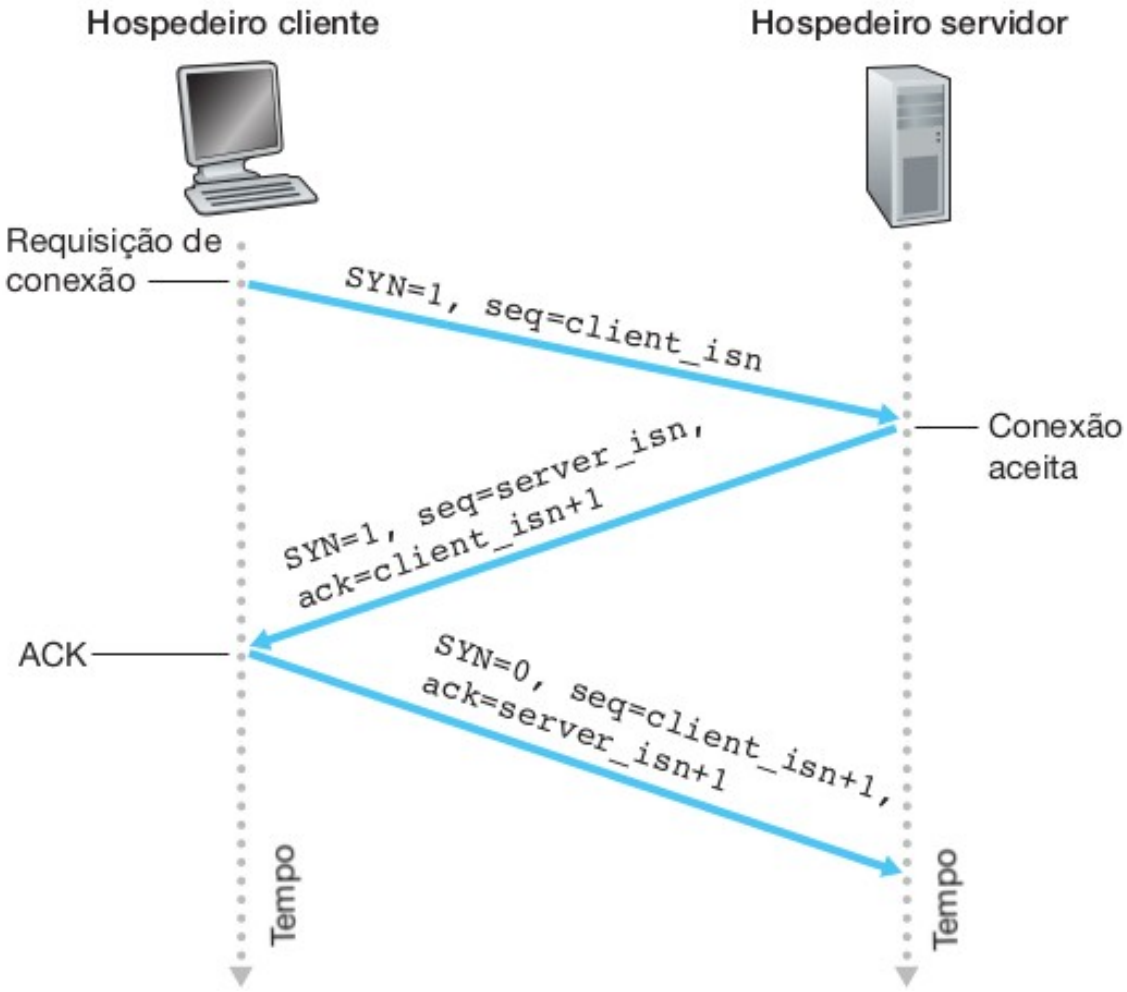
… 3.5.6 – Gerenciamento de Conexão TCP

- "**etapa #2**" .. servidor recebe segmento SYN e responde com um SYNACK no qual especifica nro. de sequência inicial a ser utilizado.



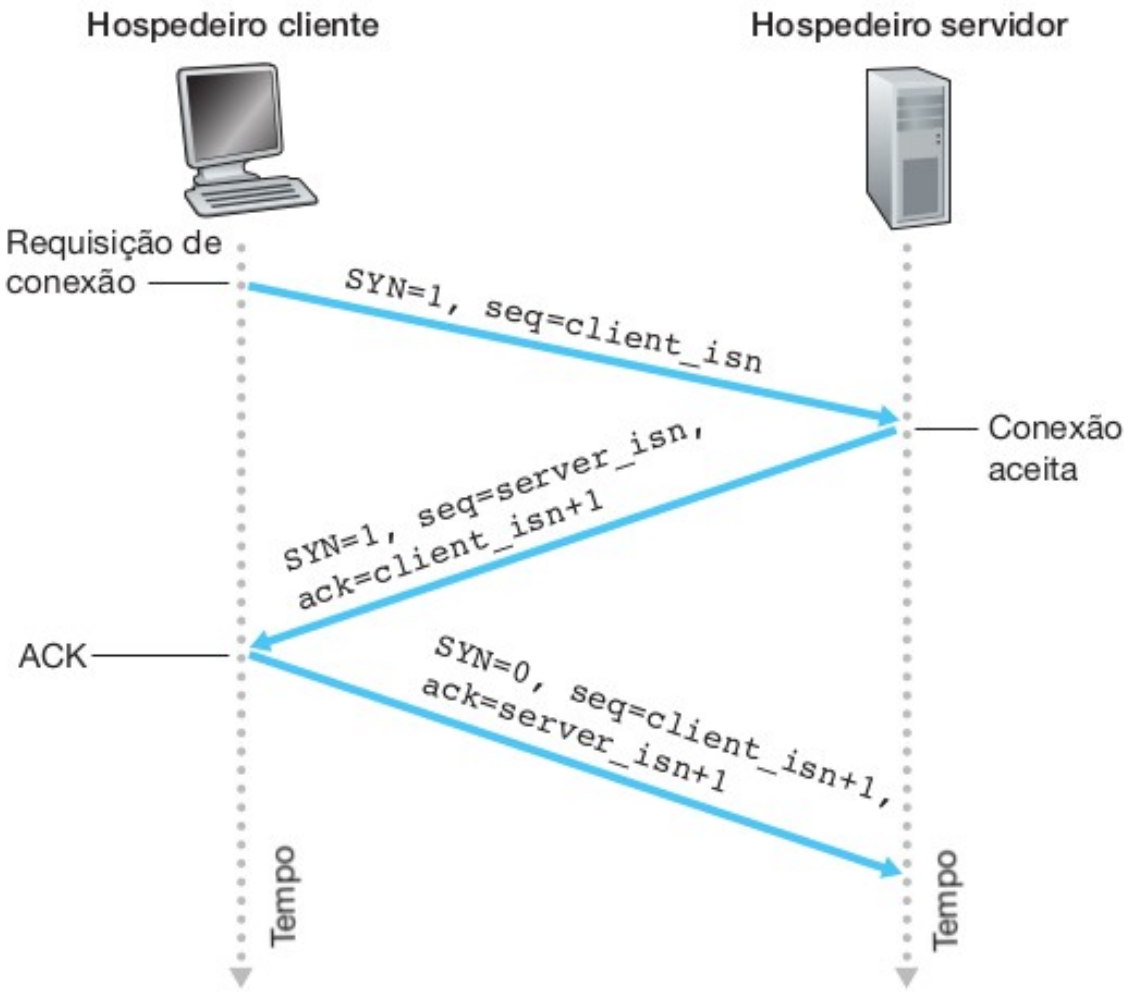
… 3.5.6 – Gerenciamento de Conexão TCP

- "**etapa #3**" .. cliente recebe SYNACK e responde com segmento ACK que pode ou não conter dados propriamente ditos.



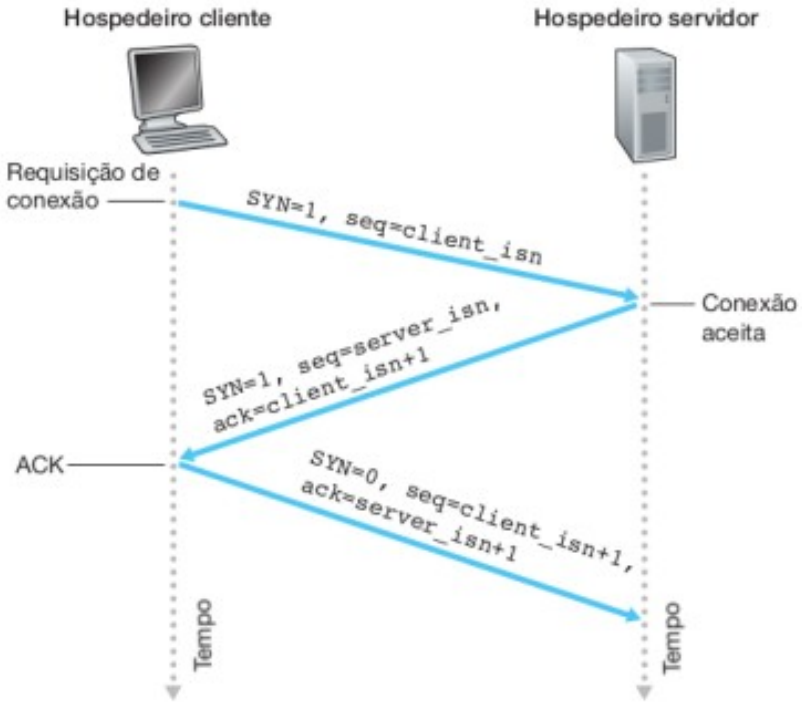
… 3.5.6 – Gerenciamento de Conexão TCP

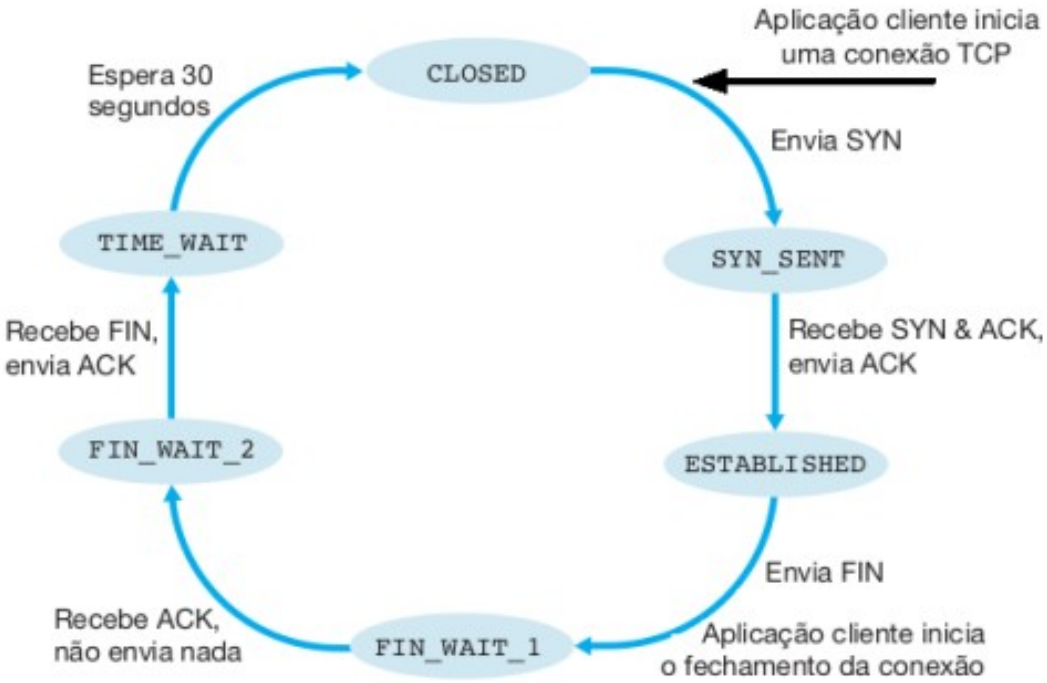
- "**troca de segmento**" .. atenção para a troca dos nros. de sequência, bem como os bits de "flags" utilizados !!



… 3.5.6 – Gerenciamento de Conexão TCP

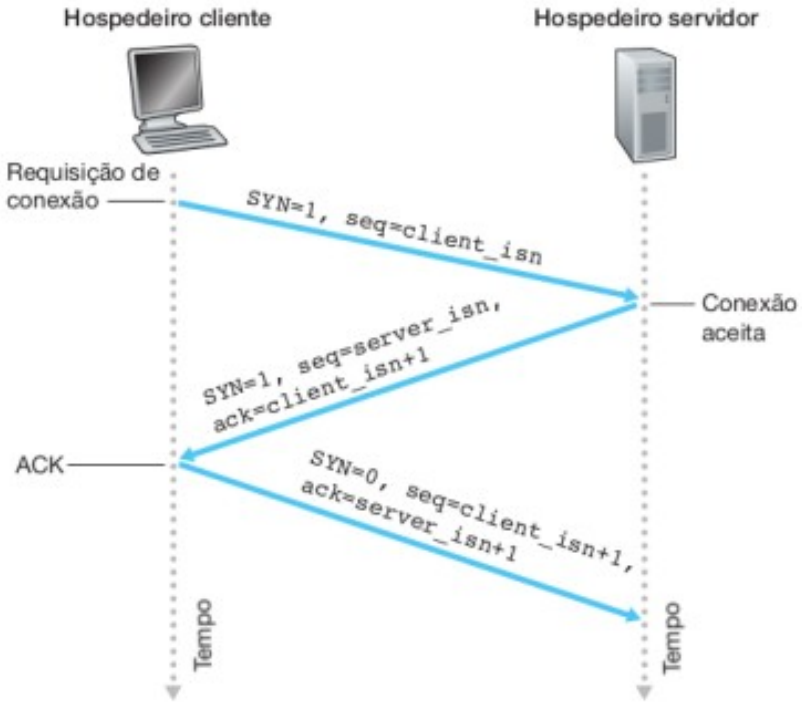
- "**FSM para Cliente TCP – estabelecimento**" .. estados pelos quais um cliente passa no estabelecimento e encerramento de conexão.
- … no estado CLOSED o cliente pode abrir uma nova conexão com um "socket" e na sequência envia um segmento SYN ao servidor.

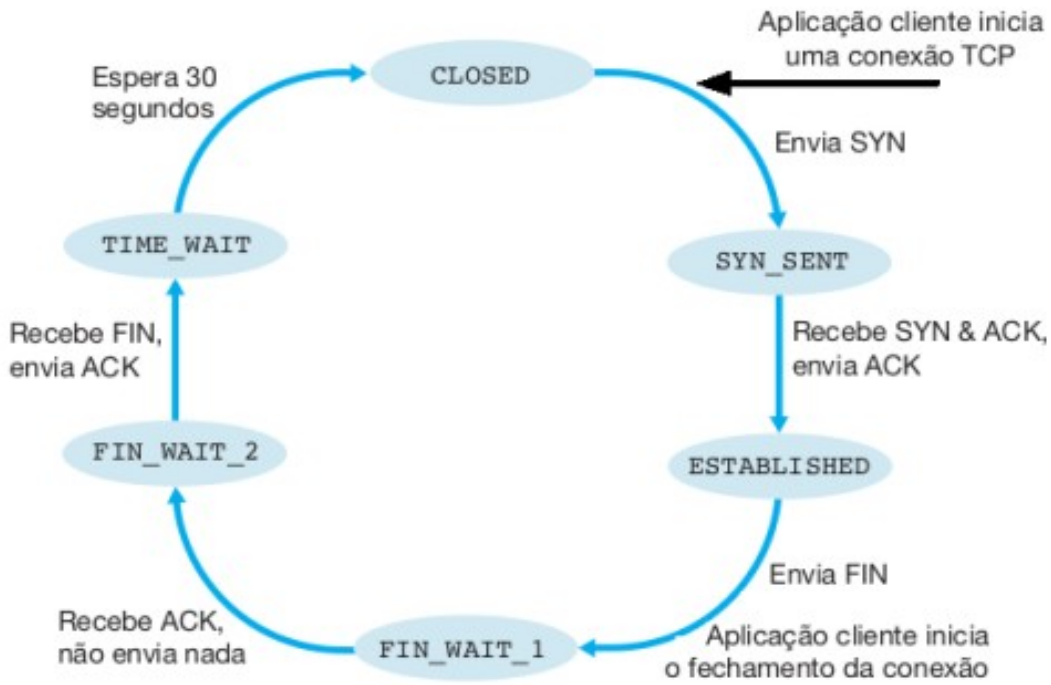




… 3.5.6 – Gerenciamento de Conexão TCP

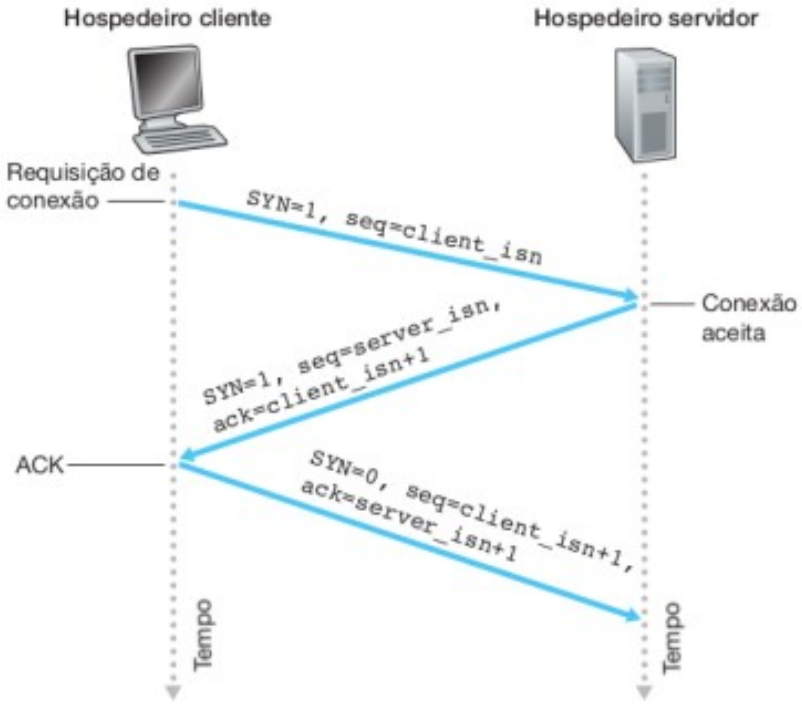
- … após o envio, o TCP cliente entra no estado SYN_SENT e, enquanto isso, o TCP cliente espera por um segmento do TCP servidor.
- … servidor envia um reconhecimento para o segmento anterior do cliente e tem o bit SYN ajustado para o valor 1.

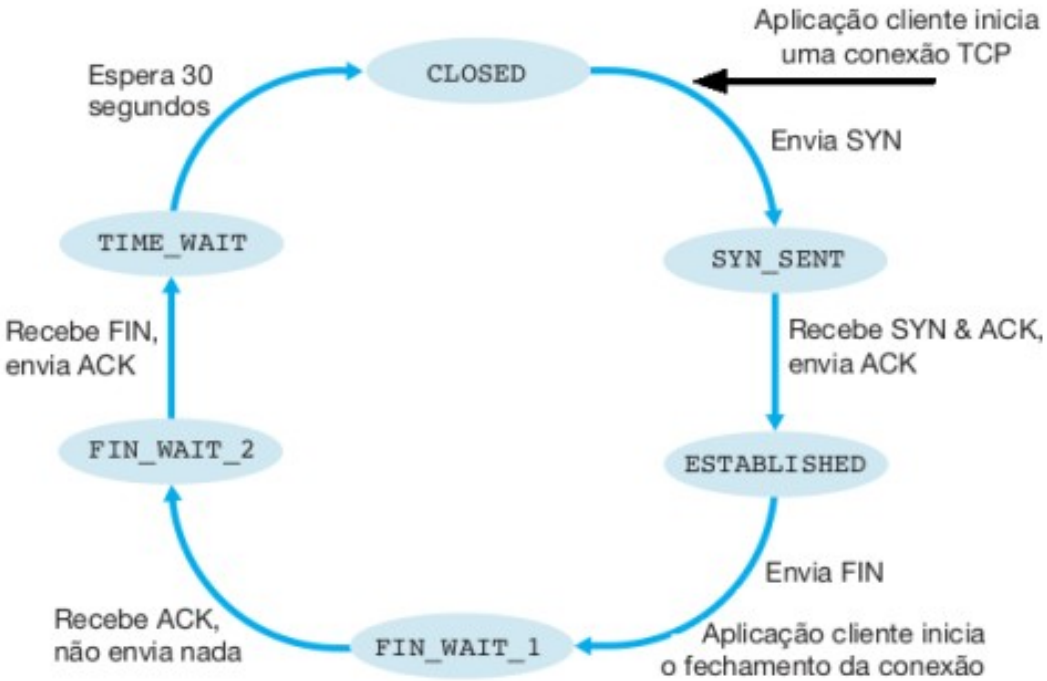




… 3.5.6 – Gerenciamento de Conexão TCP

- … assim que recebe esse segmento, o TCP cliente entra no estado ESTABLISHED, que permite o envio e recepção de segmentos TCP que contêm carga útil de dados (isto é, gerados pela aplicação).





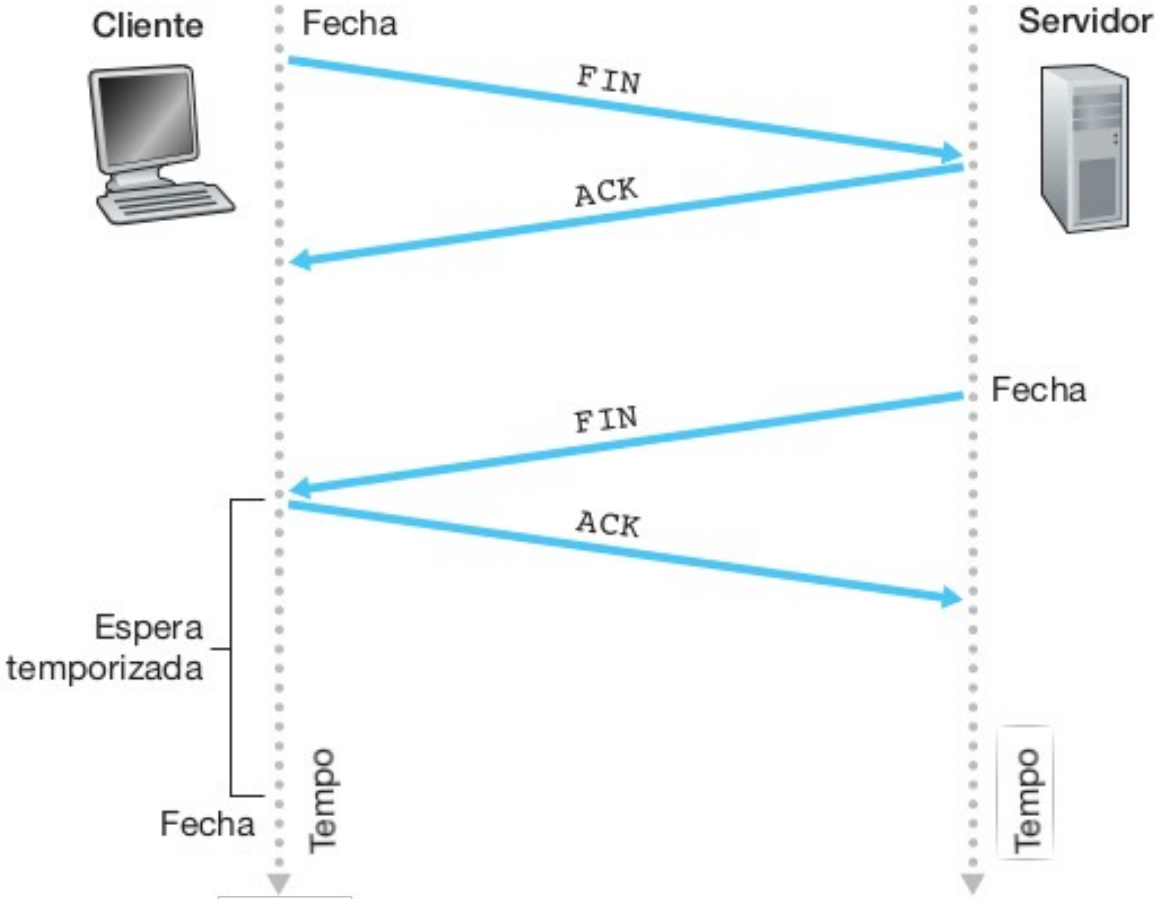
… 3.5.6 – Gerenciamento de Conexão TCP

- “**encerramento de conexão**” .. “**cliente**” .. fecha conexão: clientSocket.close( );

- e.g., remetente e destinatário trocam segmentos ACK e FIN em 04 etapas para encerrar um conexão:
- “**etapa #1**” .. cliente envia segmento de controle FIN ao servidor.
- “**etapa #2**” .. servidor recebe FIN e envia ACK ao cliente, fecha conexão e na sequência envia mais um segmento FIN.
- “**etapa #3**” .. cliente recebe FIN e responde com ACK e na sequência aguarda em espera temporizada.
- “**etapa #4**” **-** servidor recebe ACK e encerra conexão, libera variáveis, buffers e socket previamente reservados no início da conexão.

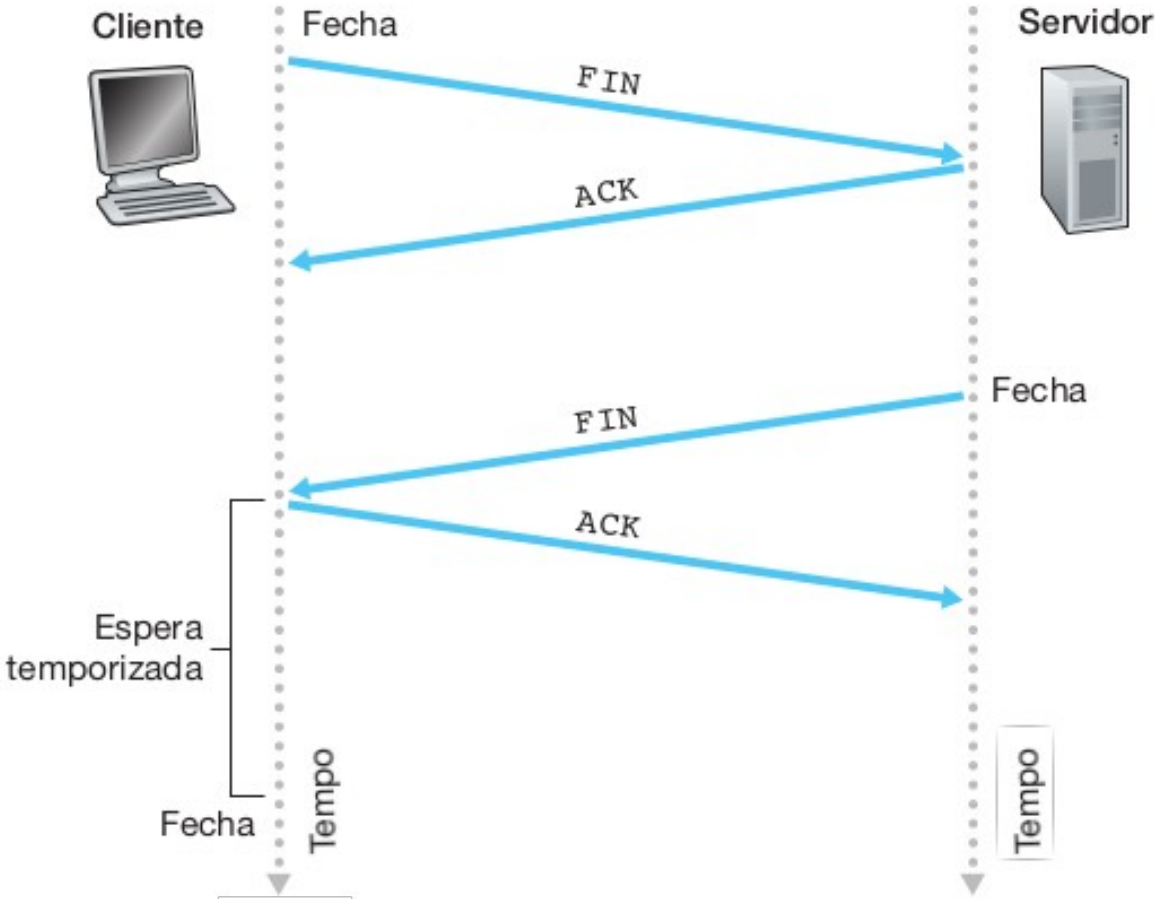
… 3.5.6 – Gerenciamento de Conexão TCP

- e.g., suponha que a aplicação cliente decida que quer fechar a conexão, então, envia-se um segmento TCP com o bit FIN ajustado em 1 e entra-se no estado FIN_WAIT_1.



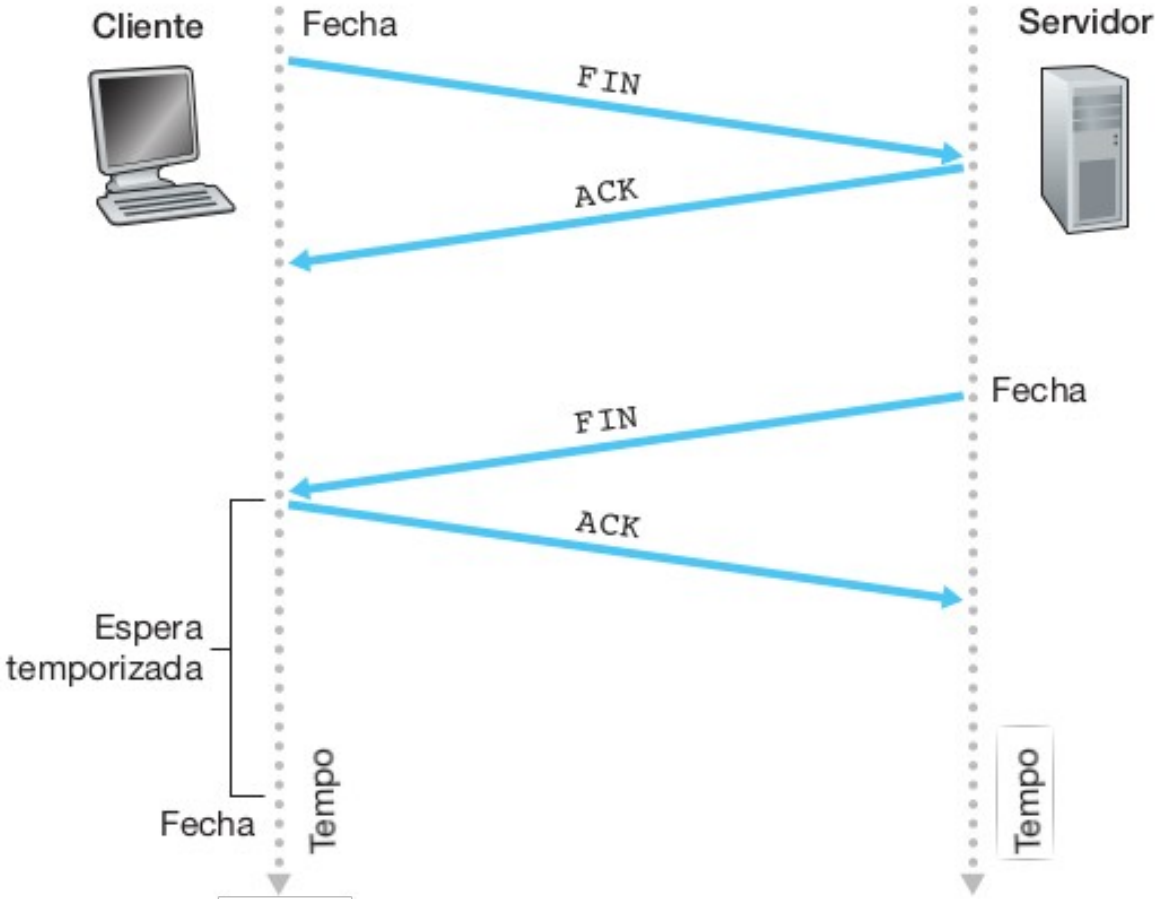
… 3.5.6 – Gerenciamento de Conexão TCP

- .. no estado FIN_WAIT_1, o TCP cliente espera por um segmento TCP do servidor com um reconhecimento - ACK.



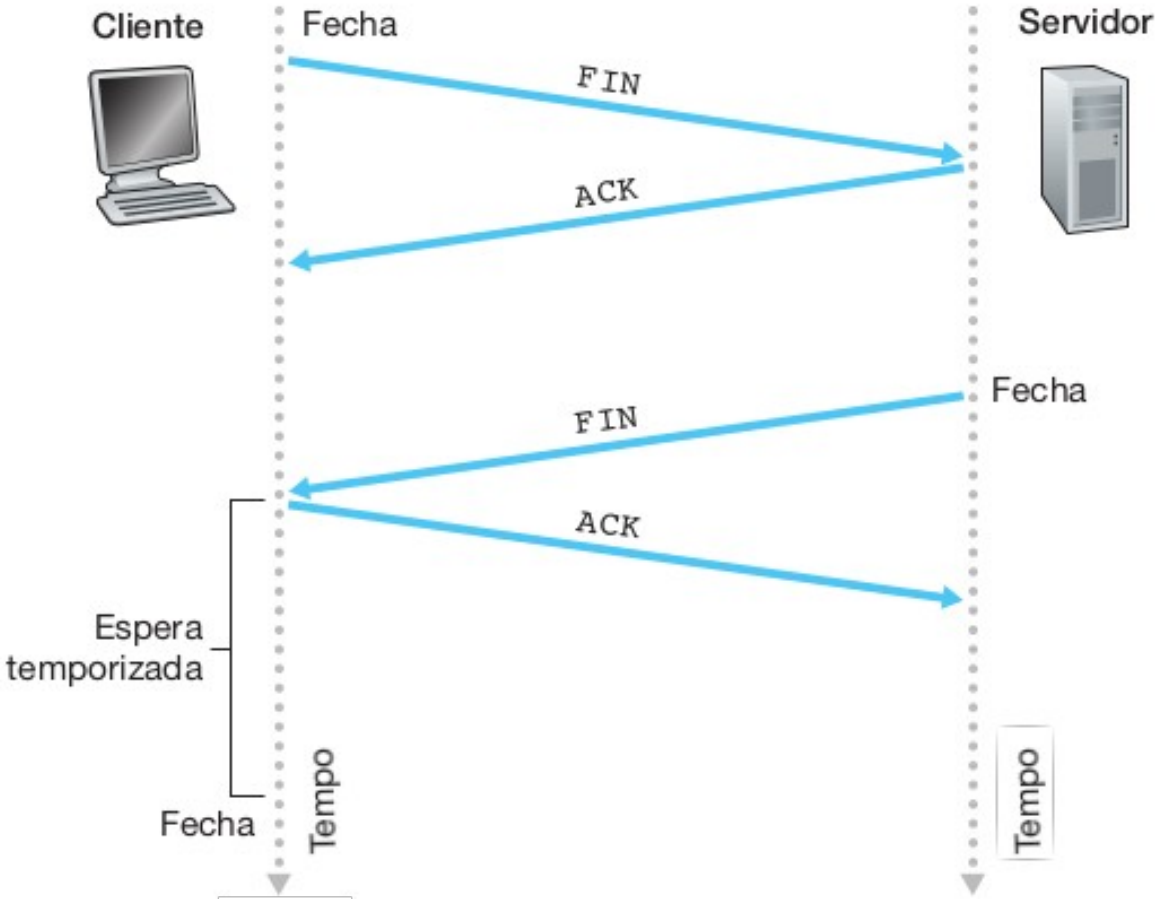
… 3.5.6 – Gerenciamento de Conexão TCP

- .. quando o remetente recebe esse segmento, o TCP cliente entra no estado FIN_WAIT_2, no qual espera por outro segmento do servidor com o bit FIN ajustado para 1.



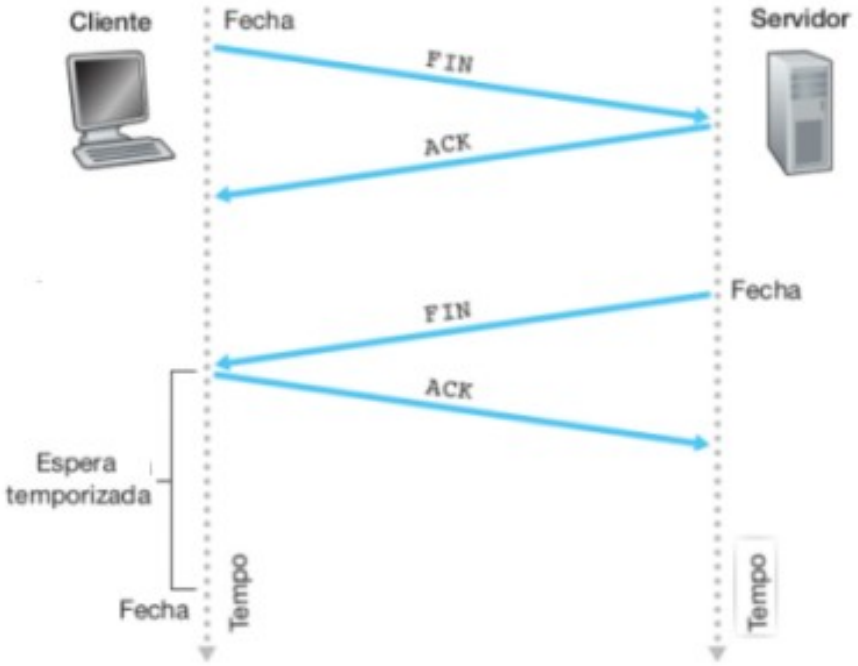
… 3.5.6 – Gerenciamento de Conexão TCP

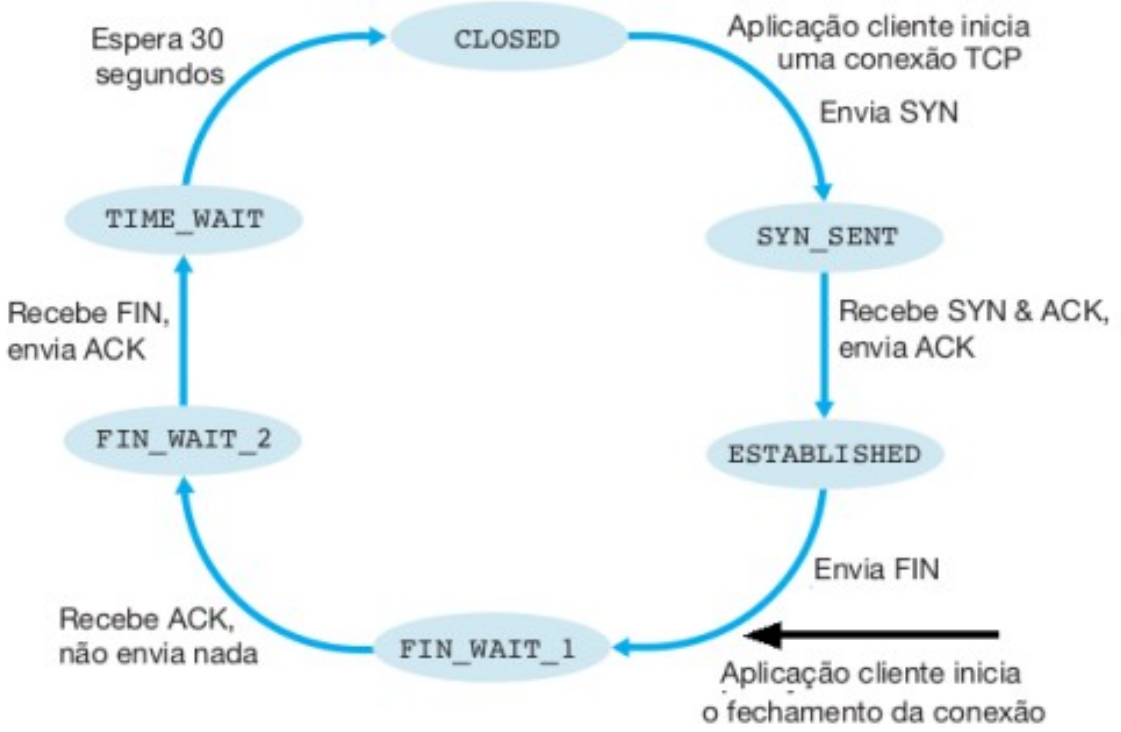
- … o cliente TCP reconhece o segmento do servidor e entra no estado TIME_WAIT, permitindo que o cliente reenvie o reconhecimento final, caso o ACK já enviado seja perdido.



… 3.5.6 – Gerenciamento de Conexão TCP

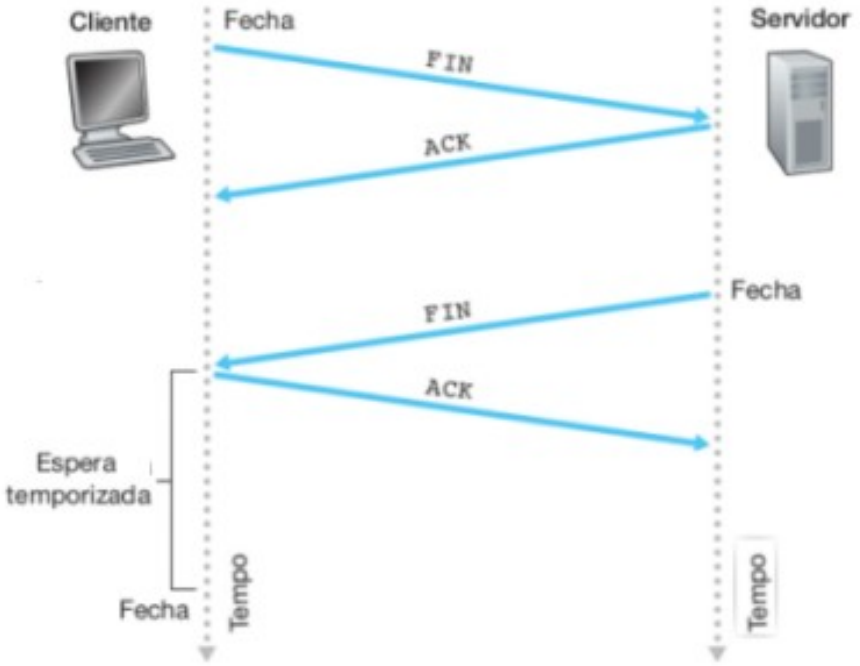
- "**FSM para Cliente TCP – encerramento**" .. suponha que um cliente decida fechar a conexão, então, envia-se um segmento TCP com o bit FIN ajustado em 1 e entra-se no estado FIN_WAIT_1.

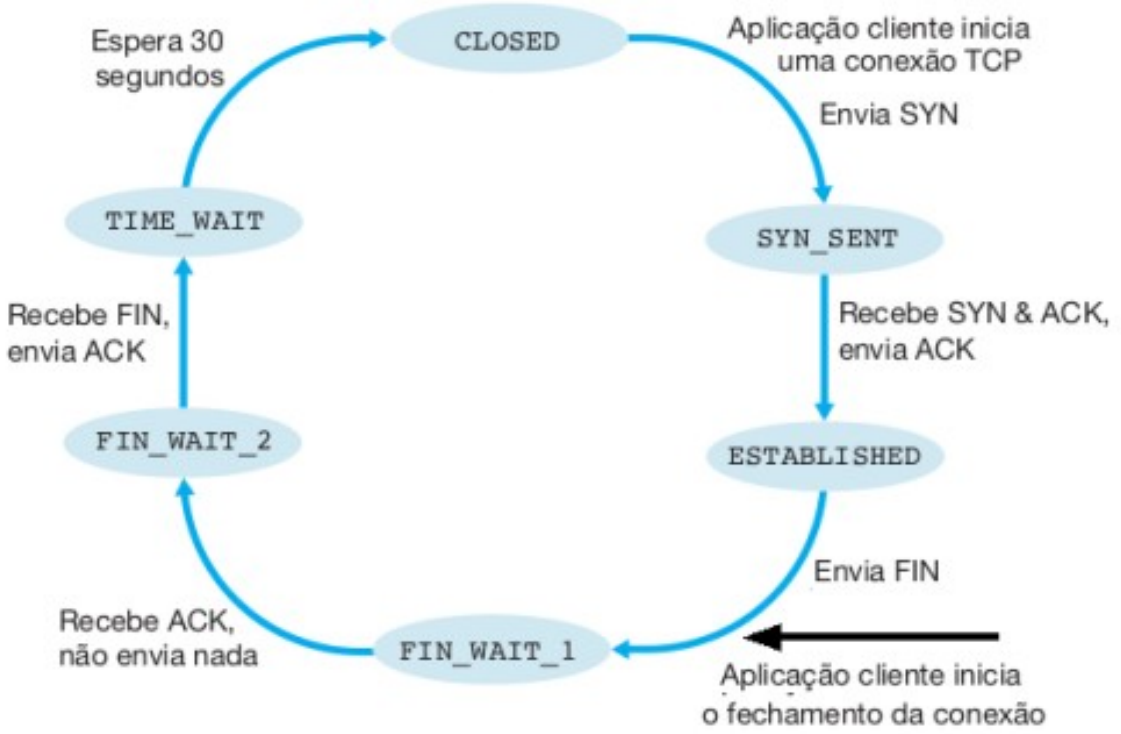




… 3.5.6 – Gerenciamento de Conexão TCP

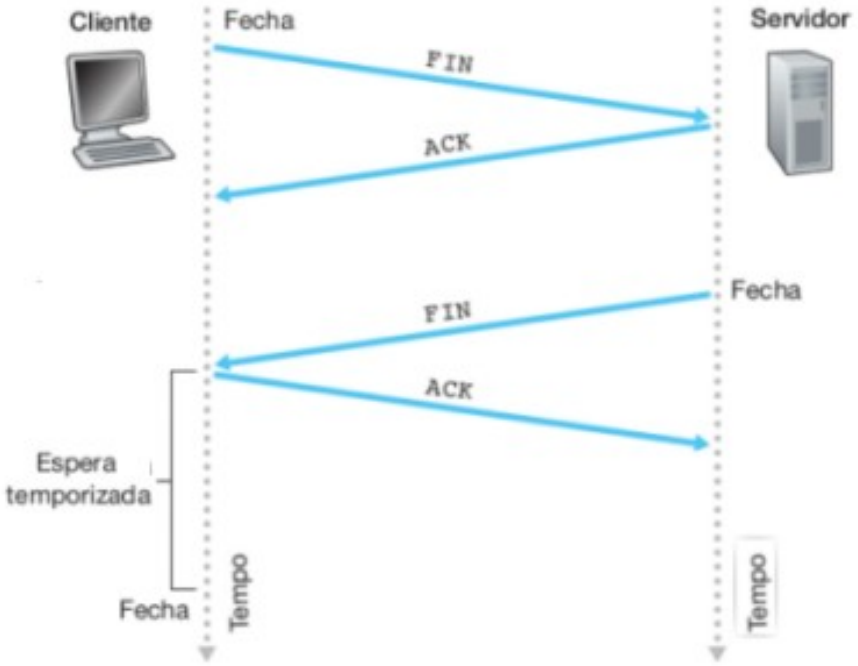
- .. no estado FIN_WAIT_1, o cliente espera por um segmento de reconhecimento do servidor, ou seja, ACK.

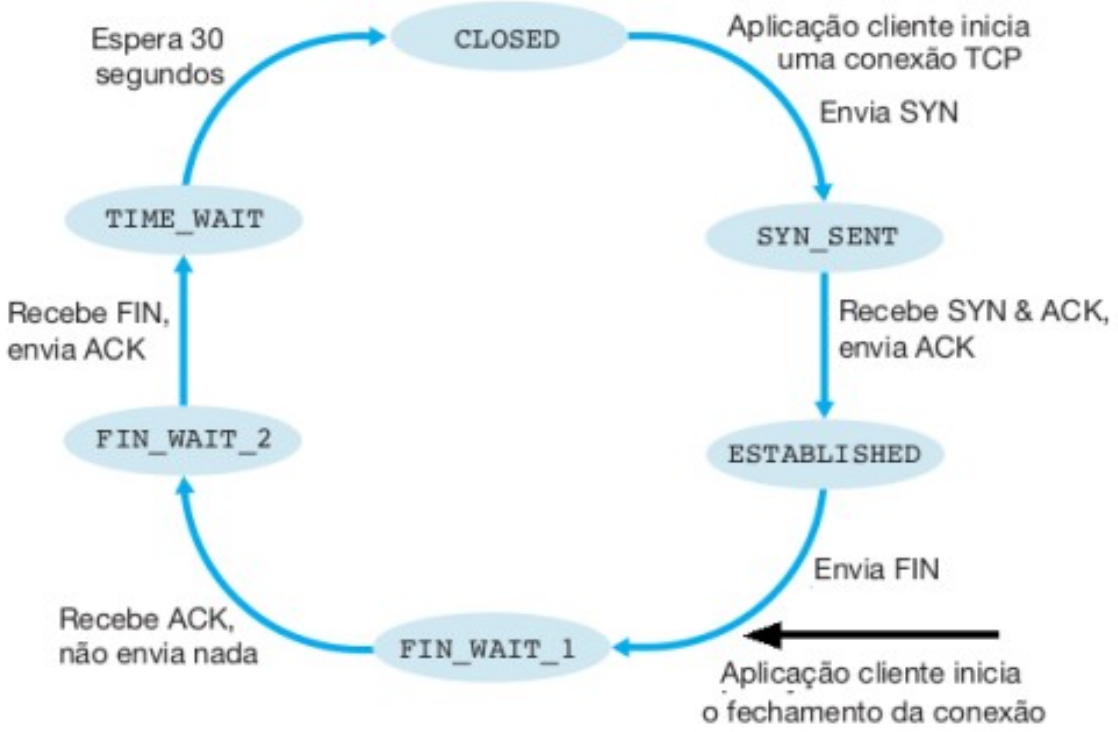




… 3.5.6 – Gerenciamento de Conexão TCP

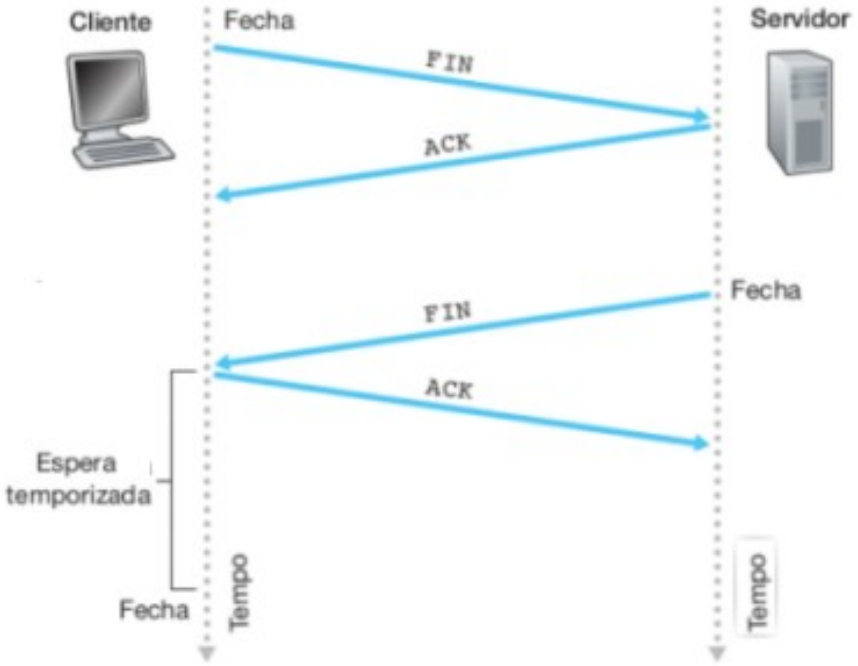
- .. quando o cliente recebe esse reconhecimento (ACK = 1), o estado muda para FIN_WAIT_2, no qual o cliente espera por outro segmento do servidor com o bit FIN ajustado para 1.

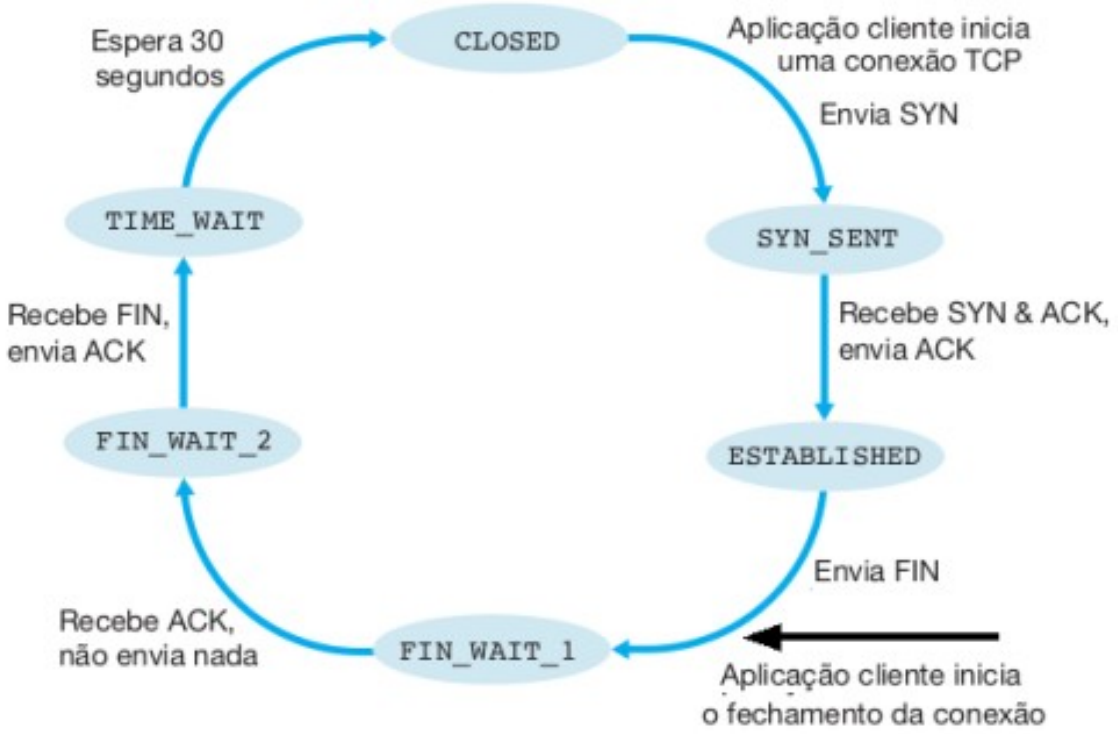




… 3.5.6 – Gerenciamento de Conexão TCP

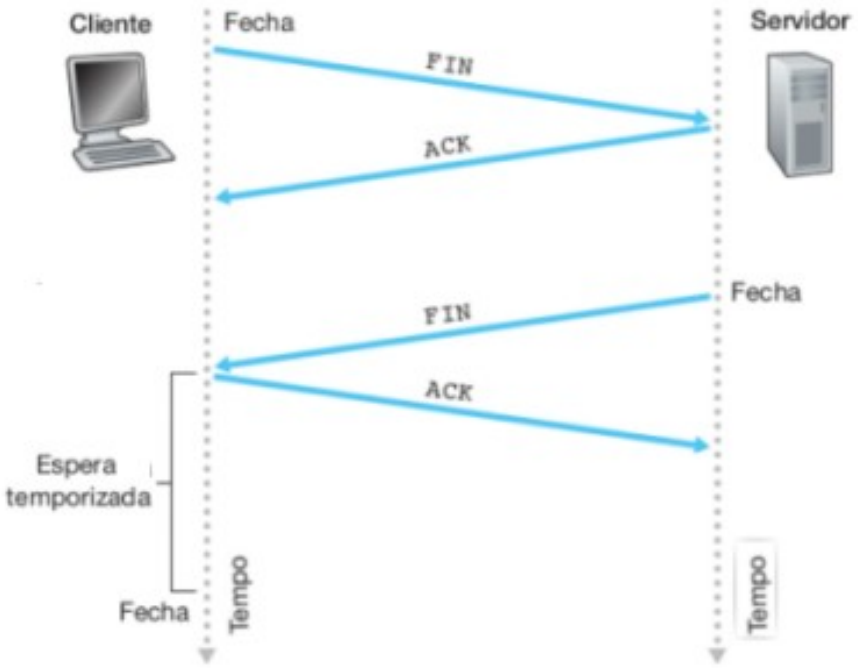
- .. no estado FIN_WAIT_2, o cliente aguarda outro segmento do servidor com o bit FIN ajustado para 1.

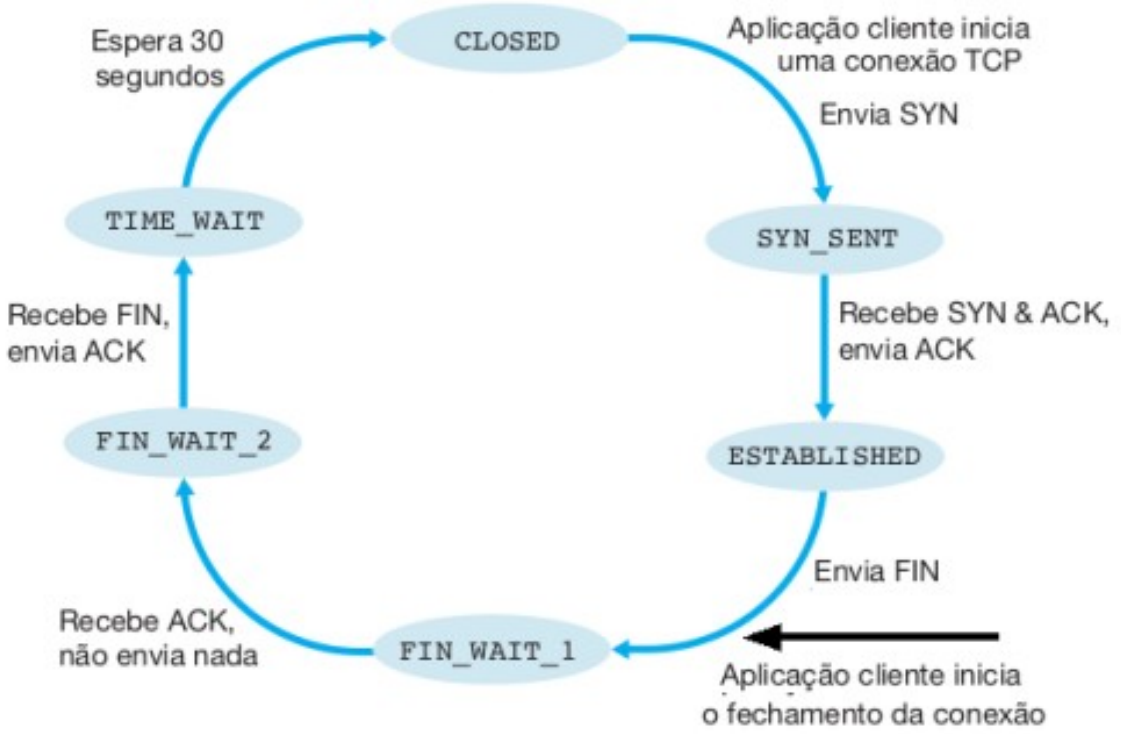




… 3.5.6 – Gerenciamento de Conexão TCP

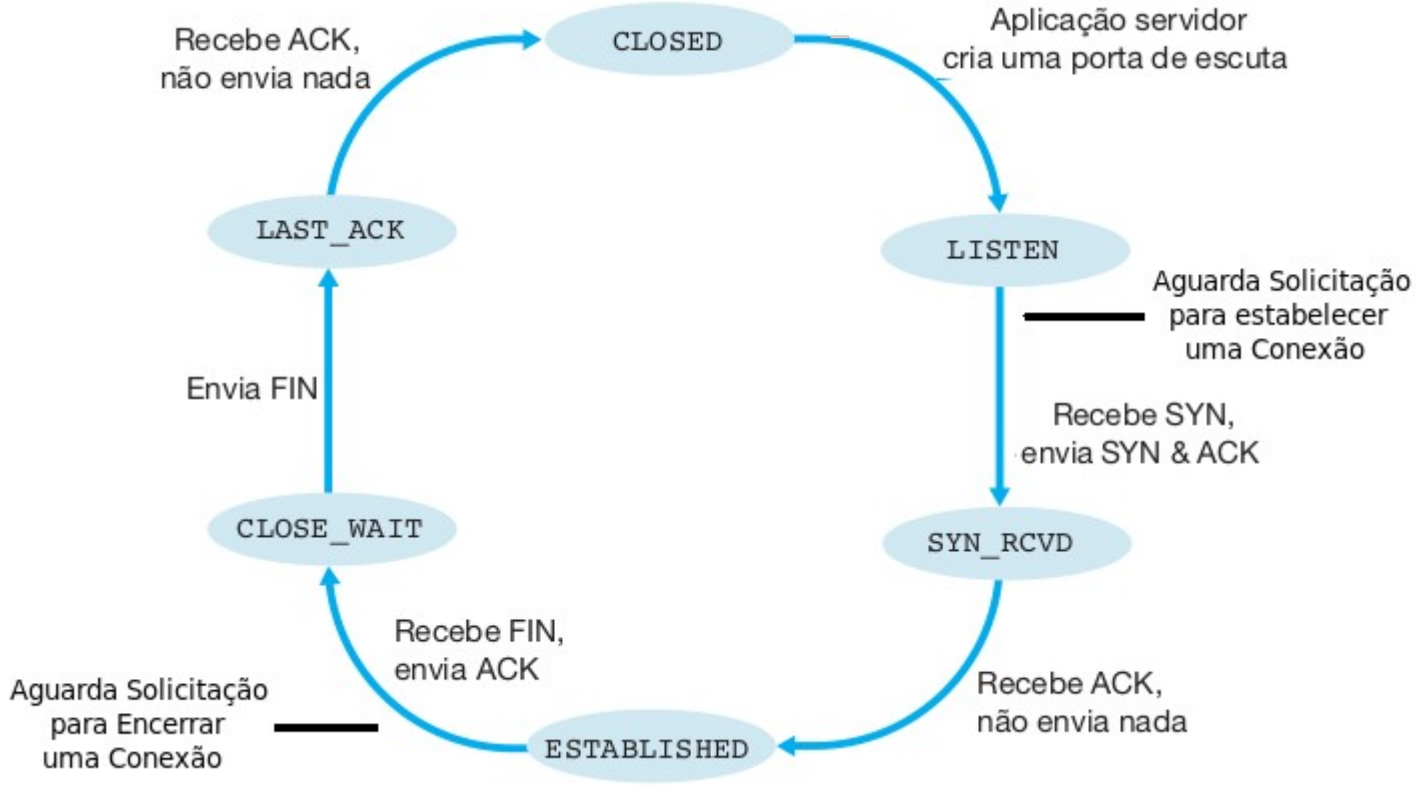
- … ao receber e reconhecer o segmento, o cliente entra no estado TIME_ WAIT, permitindo que o mesmo reenvie o reconhecimento final, caso o reconhecimento “já enviado” seja perdido.





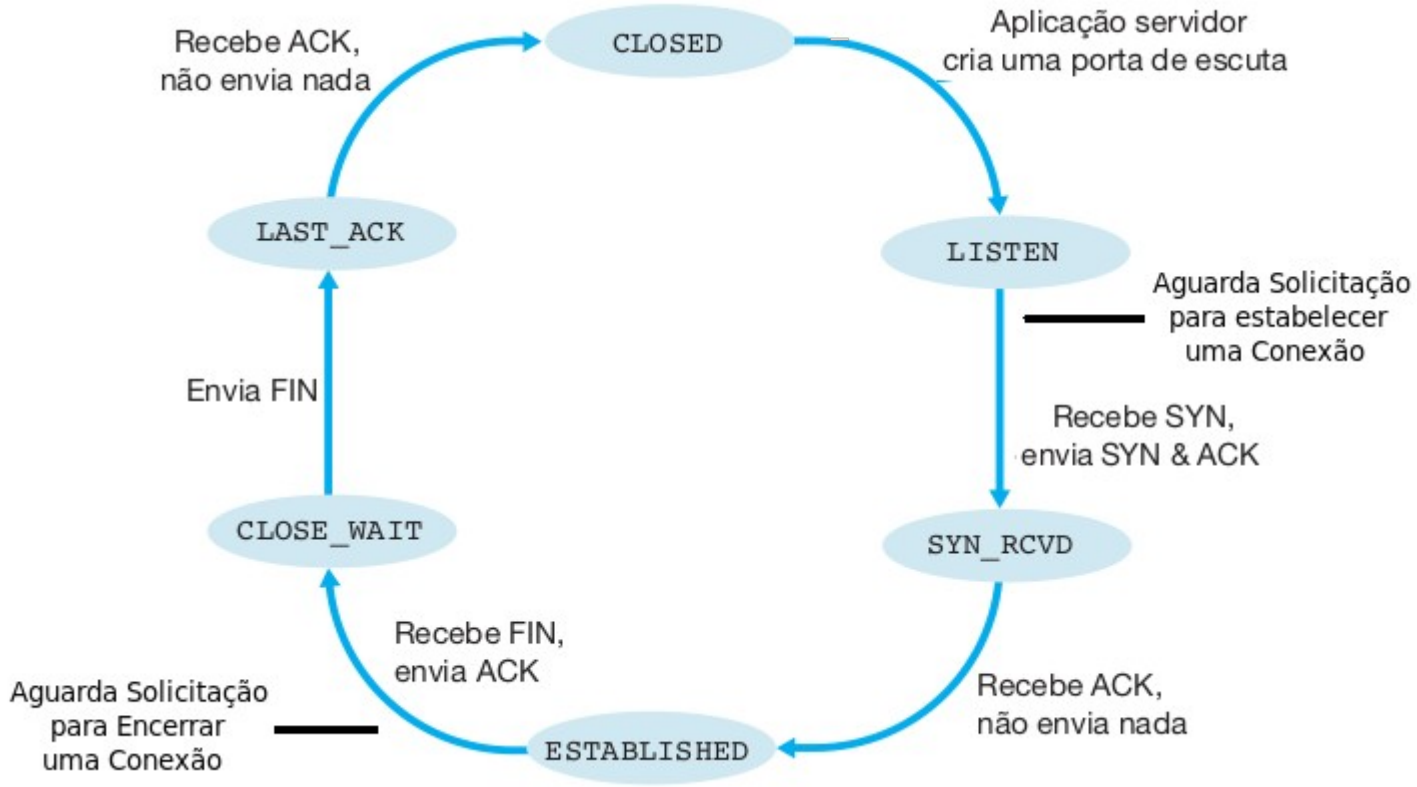
… 3.5.6 – Gerenciamento de Conexão TCP

- "**FSM para Servidor TCP**" .. estados visitados pelo TCP do lado servidor, tanto no estabelecimento / encerramento de uma conexão.
- .. não são contemplados cenários patológicos, p.ex., ambos os lados de uma conexão querem iniciar ou fechar ao mesmo tempo.



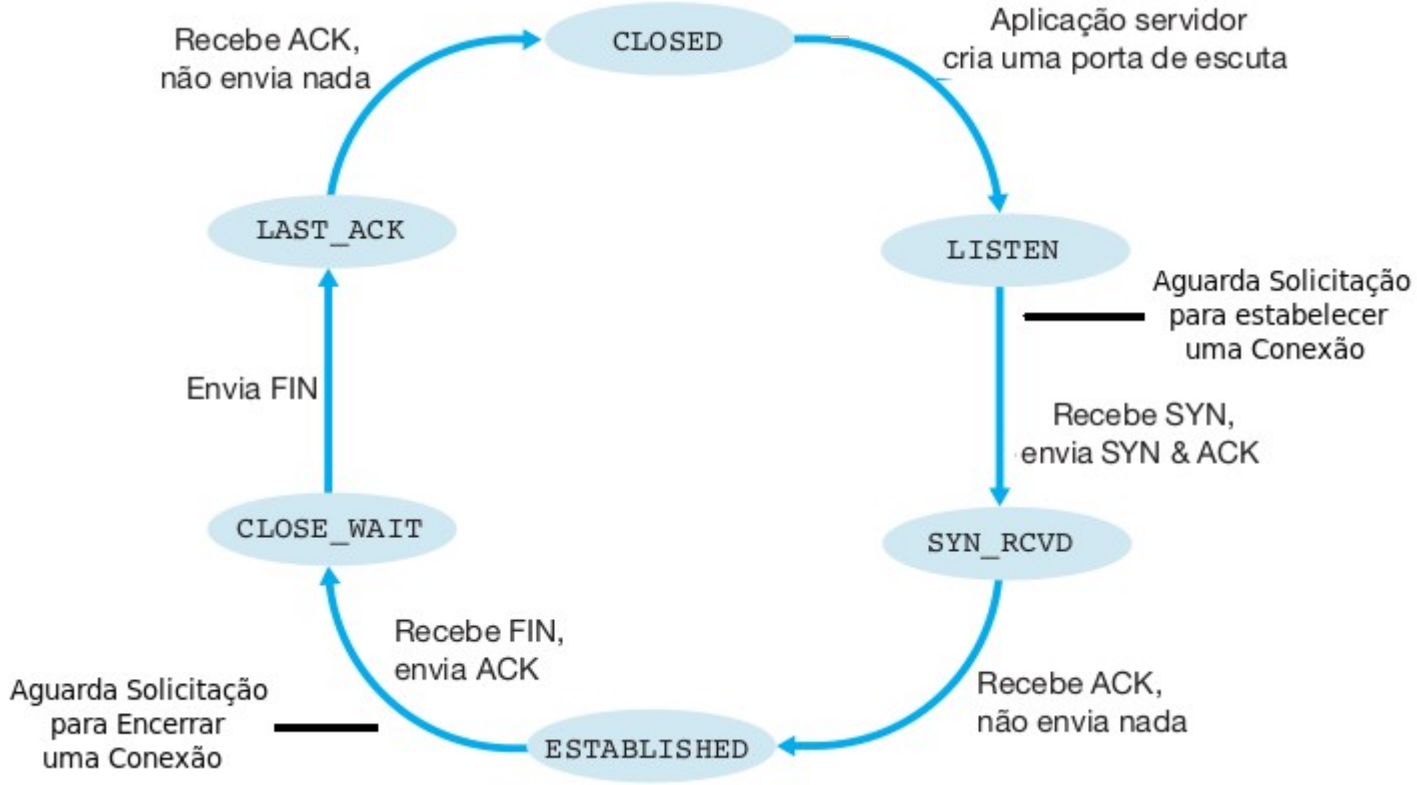
… 3.5.6 – Gerenciamento de Conexão TCP

- … nesses dois diagramas de transição de estados, mostramos apenas como uma conexão TCP é em geral estabelecida e fechada.



… 3.5.6 – Gerenciamento de Conexão TCP

- … nesses dois diagramas de transição de estados, mostramos apenas como uma conexão TCP é em geral fechada.
- "**observação**" .. estados visitados pelo TCP do lado servidor, admitindo-se que é o cliente iniciou o encerramento da conexão.



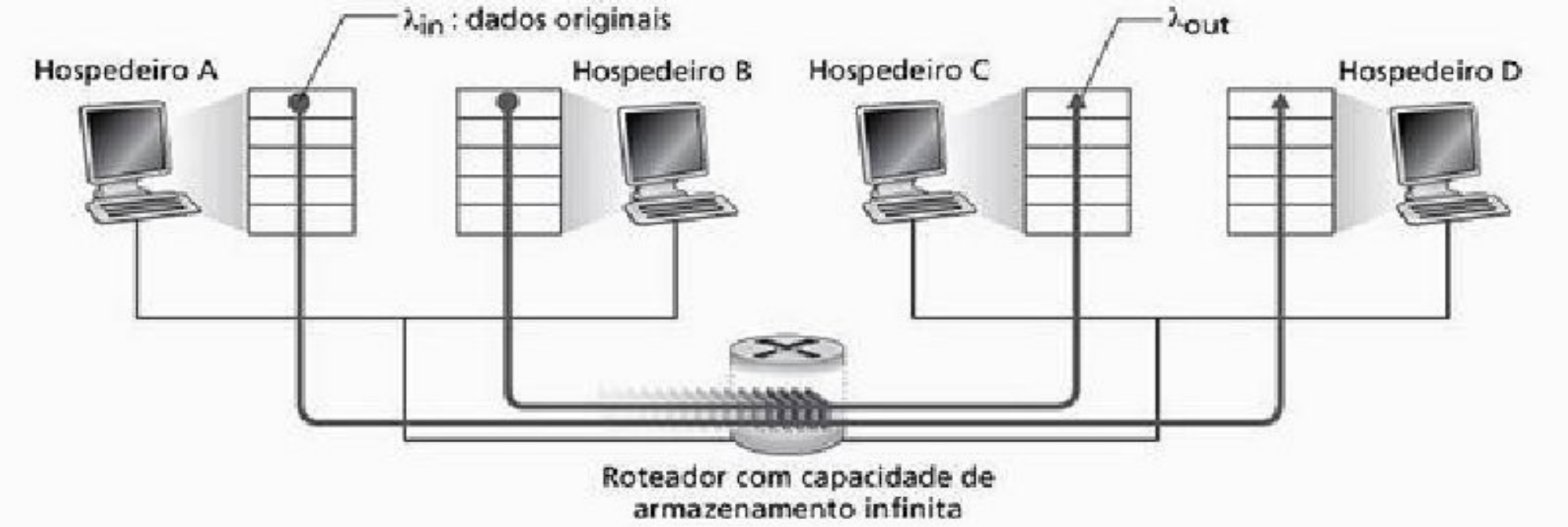
3.6 – Princípios do Controle de Congestionamento

- "**congestionamento**" .. inúmeras fontes transmitindo dados contribuem conjuntamente para esgotar os recursos ao longo dos enlaces entre as diversas fontes e diversos destinos.
- … para tratar as causas do congestionamento, mecanismos são necessários para regular o envio de dados pelo remetente.

- "**como se manifesta**" .. congestionamento é considerado um dos maiores problemas fundamentalmente importantes em redes.
- "perda de pacotes" .. saturação de "buffers" de roteadores.
- "longos atrasos" .. enfileiramento nos "buffers" dos roteadores.

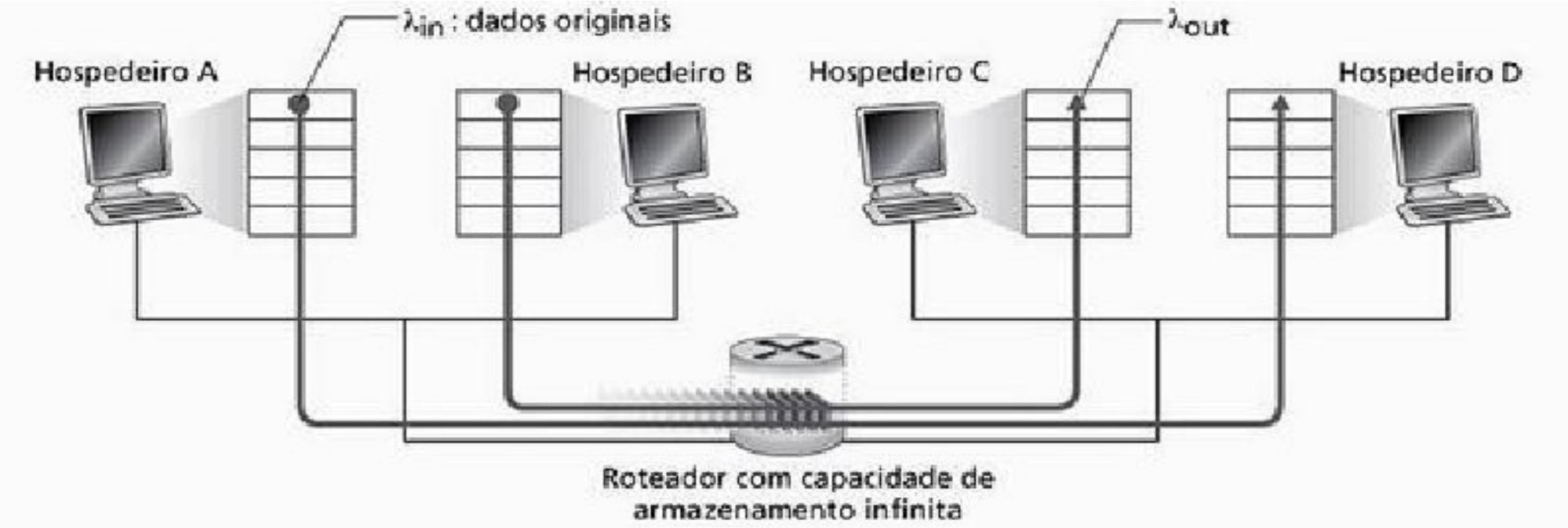
3.6.1 – Causas e Custos do Congestionamento

- **Cenário #1** .. 02 remetentes e 01 roteador com “buffer” infinito que não gera perdas de pacotes em razão da falta de armazenamento.
- .. “A” e “B” enviam dados para a conexão que gerenciam a uma taxa média de “lambda $_{in}$” bytes/segundo.



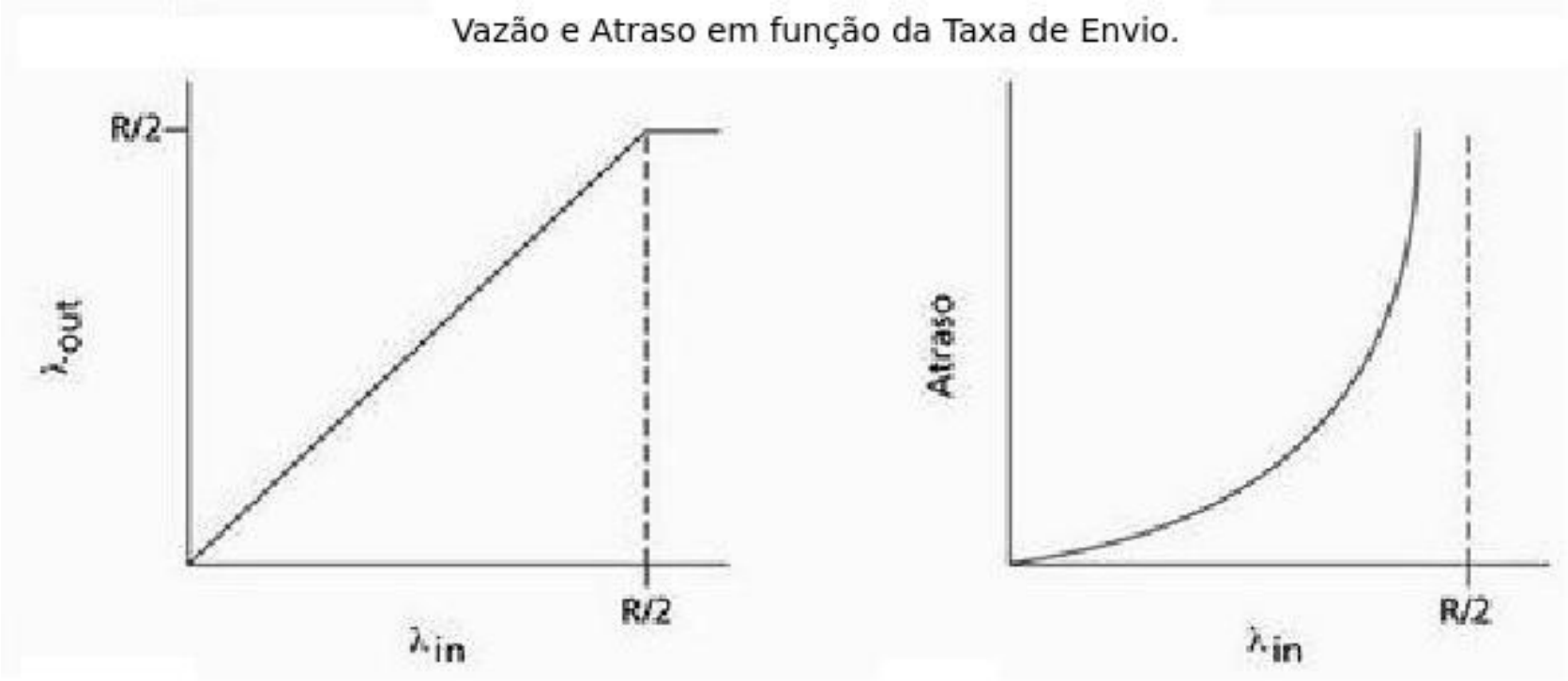
… 3.6.1 – Causas e Custos do Congestionamento

- … roteador contempla buffers que armazenam datagramas que chegam mesmo quando excedem a capacidade do enlace de saída e, neste cenário, esta capacidade de armazenamento é infinita !!



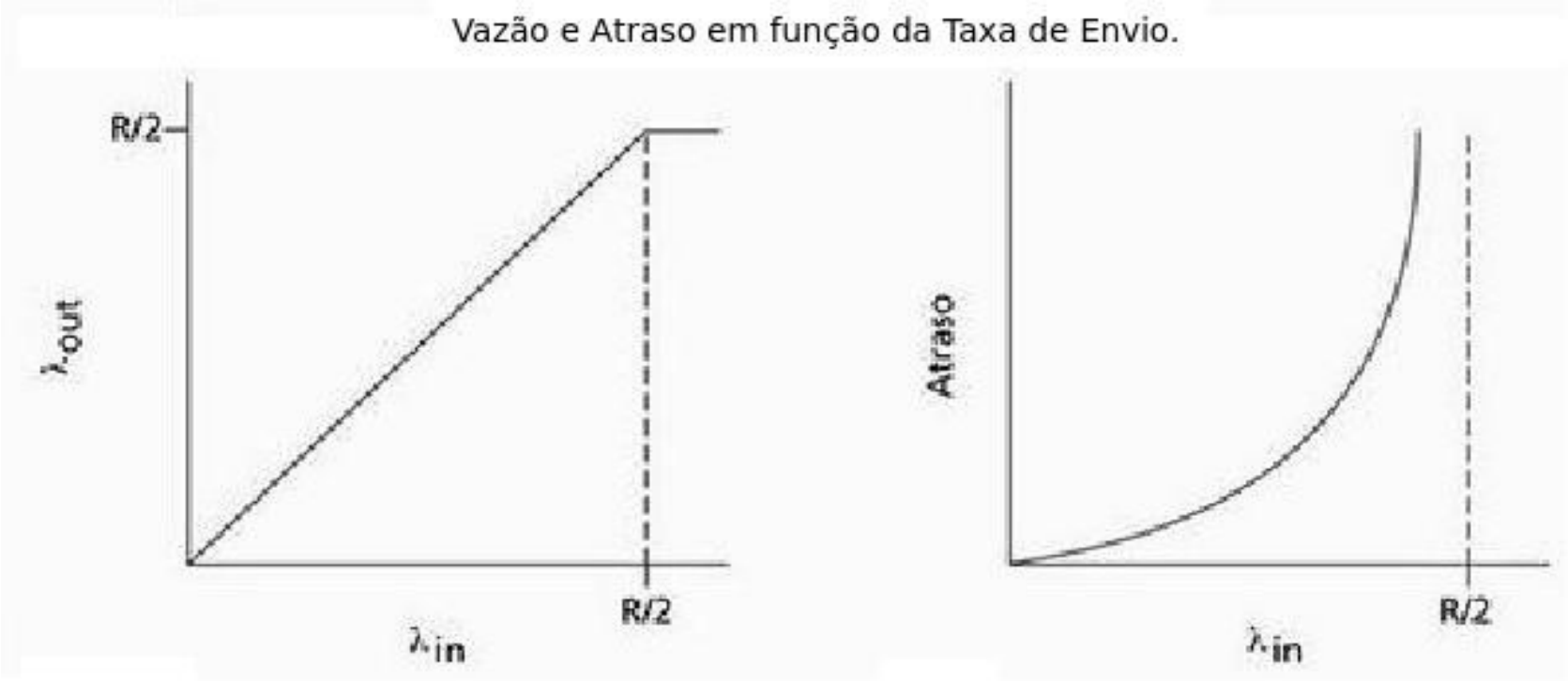
… 3.6.1 – Causas e Custos do Congestionamento

- "**vazão de conexão**" .. para taxa de transmissão de 0 a R/2, tem-se no destinatário uma vazão = velocidade de envio do remetente.
- … limite superior da vazão é consequência do compartilhamento da capacidade do enlace em 02 conexões.



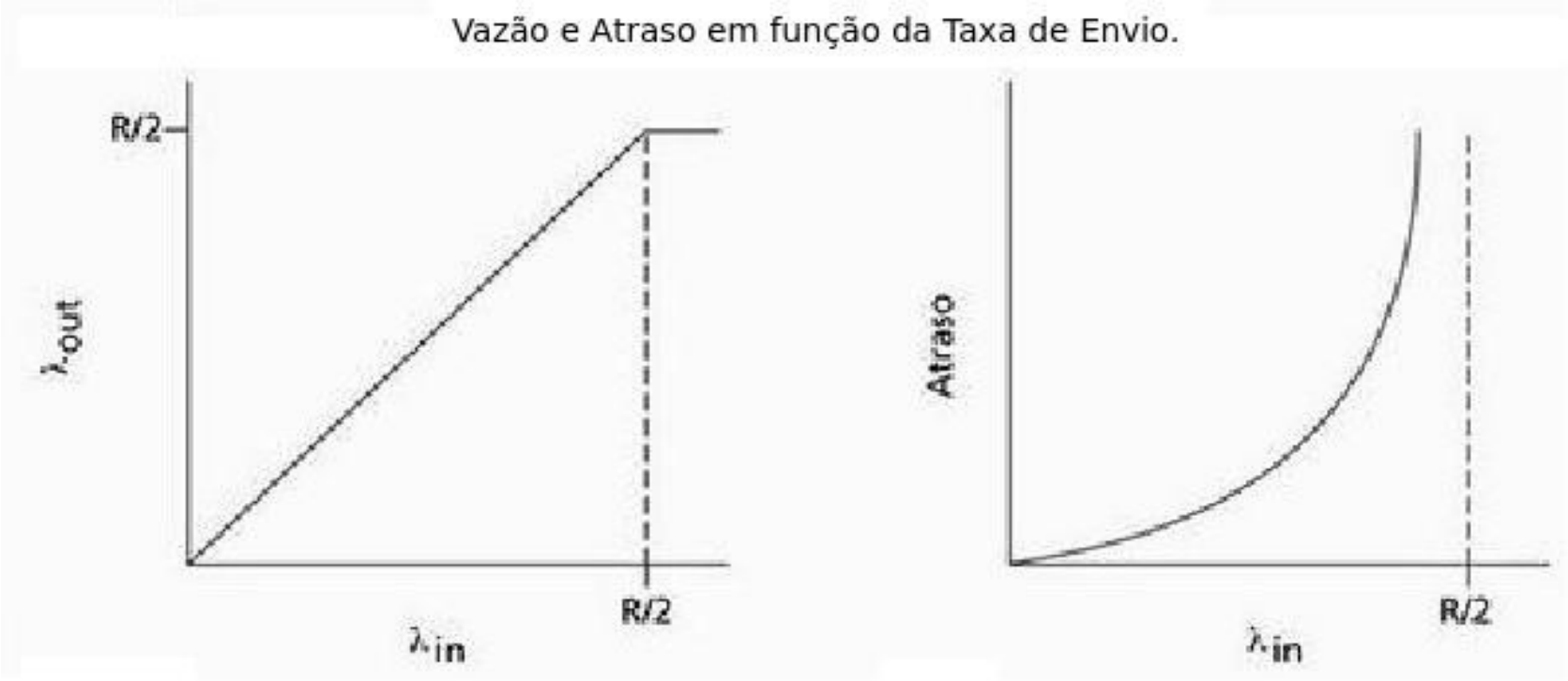
… 3.6.1 – Causas e Custos do Congestionamento

- … enlace não consegue entregar datagramas no destinatário com taxas em regime que excedam R/2.
- … não importa quão alta sejam as taxas, “hosts” jamais alcançarão uma vazão maior do que “R/2” da capacidade do enlace.



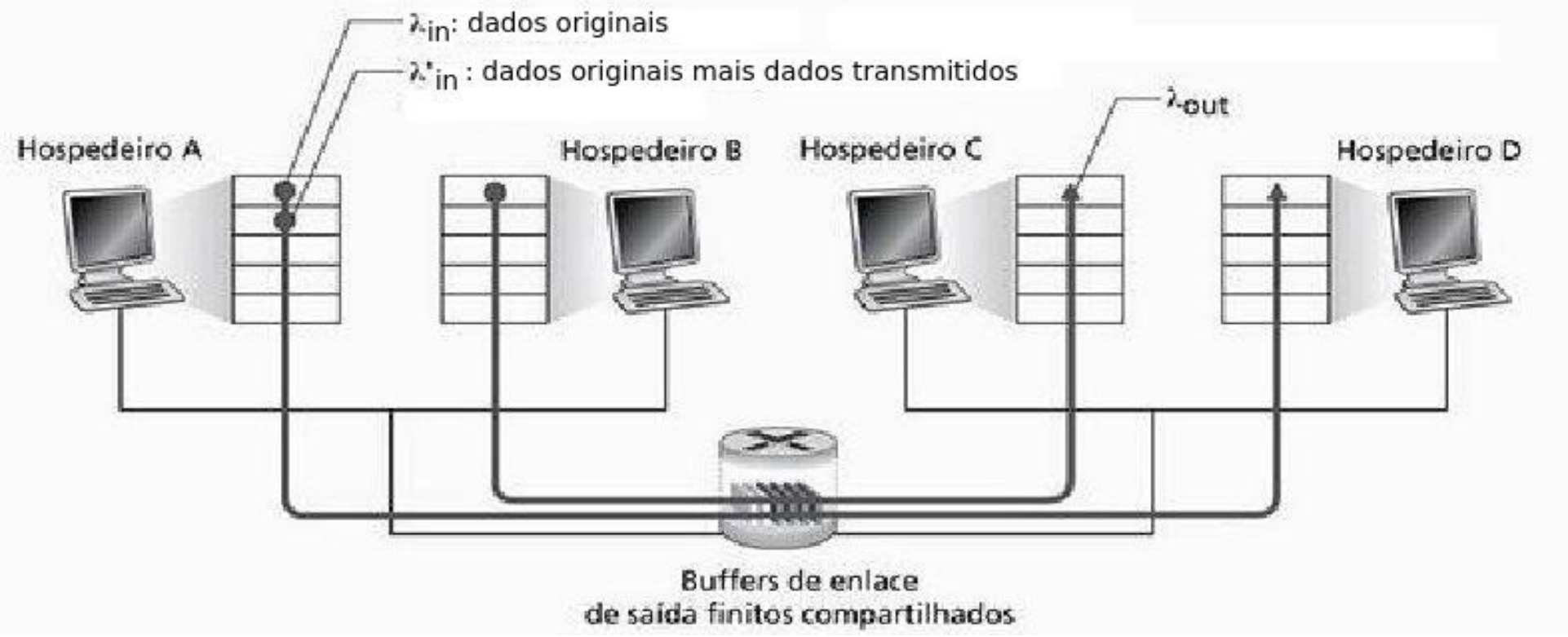
… 3.6.1 – Causas e Custos do Congestionamento

- … no entanto, o efeito de se operar na capacidade máxima do enlace eleva o atraso médio para níveis muito altos.
- … quando a taxa ultrapassa R/2, o nro. médio de pacotes na fila do roteador é ilimitado e o atraso se torna infinito.



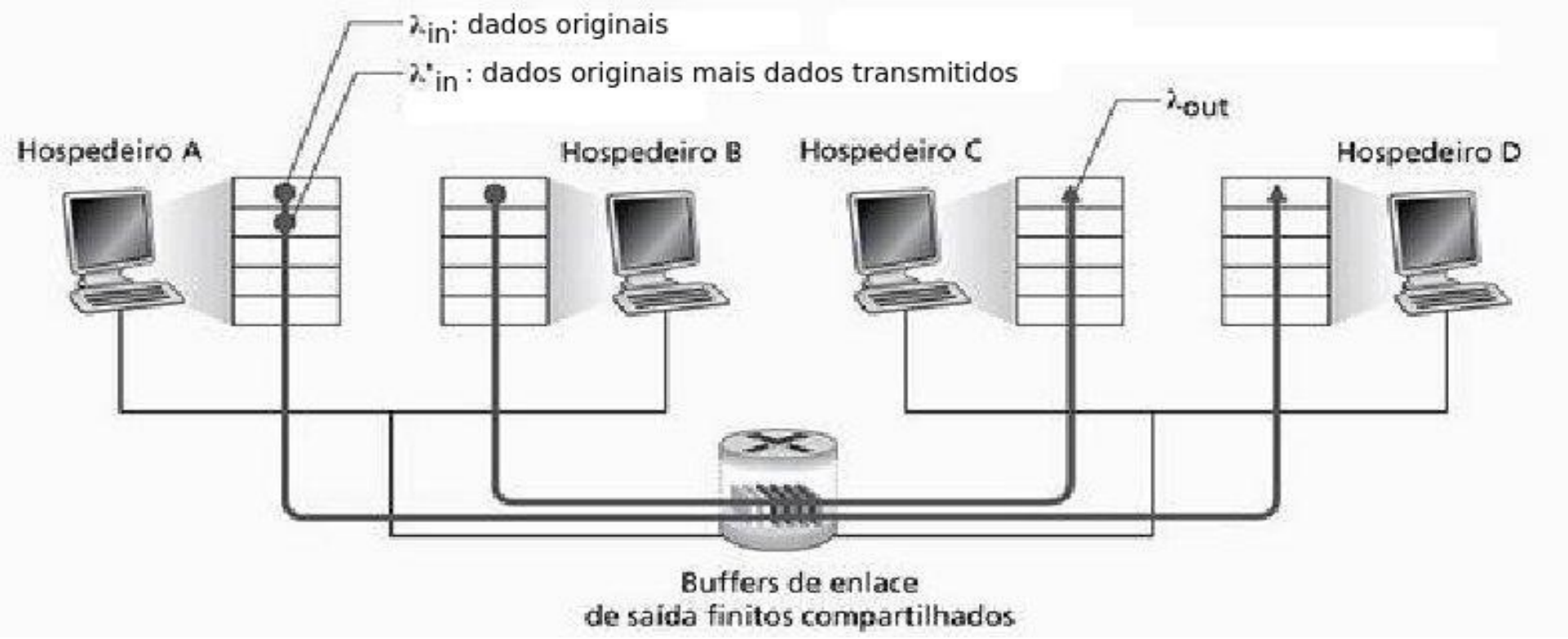
… 3.6.1 – Causas e Custos do Congestionamento

- **Cenário #2** .. 02 remetentes e 01 roteador com “buffers” finitos o que abre a possibilidade de descarte de pacotes que chegam a um buffer de roteador que eventualmente esteja cheio.



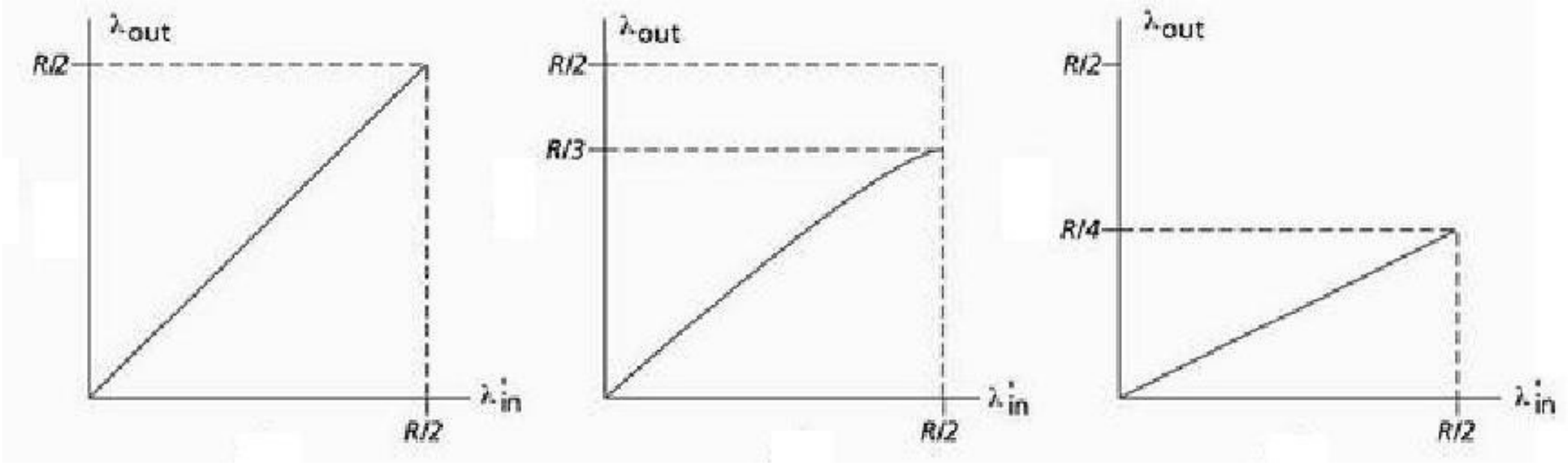
… 3.6.1 – Causas e Custos do Congestionamento

- … seja λ_{in} bytes/segundo a taxa de envio com que a aplicação envia os dados originais para dentro do “socket”.
- … seja λ'_{in} bytes/segundo a taxa com que a camada de transporte envia segmentos para dentro da camada de rede.



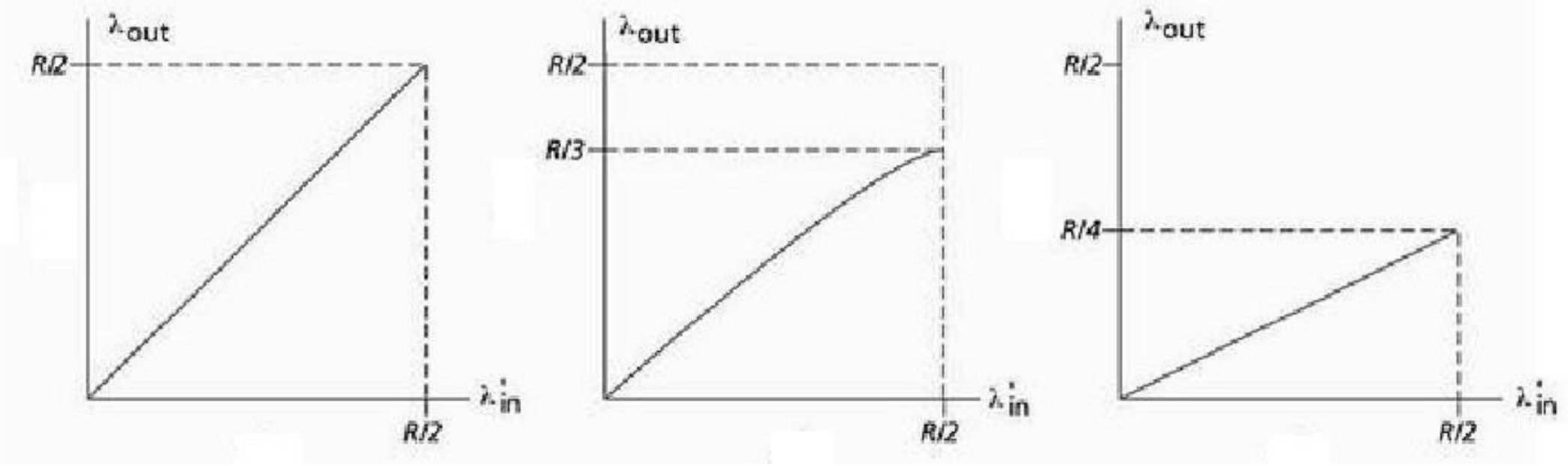
… 3.6.1 – Causas e Custos do Congestionamento

- … se “A” envia um pacote somente quando o “buffer” estiver livre, não teremos nenhuma perda, assim λ_{in} será igual a λ'_{in}.
- … neste caso, taxa média de envio do “host” não pode ultrapassar “R/2” já que admitimos que nunca ocorre perda de pacote (1º Gráfico)



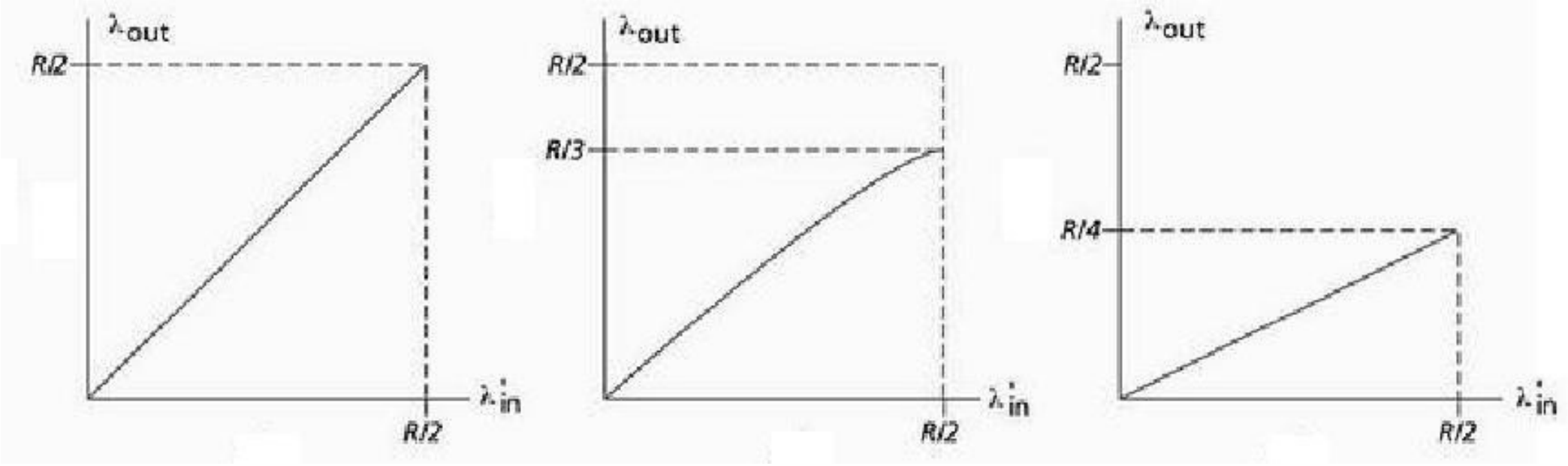
… 3.6.1 – Causas e Custos do Congestionamento

- … remetente retransmite somente quando sabe, com certeza, que o pacote não reconhecido foi de fato perdido.
- … para um carga λ'_{in} (λ_{in} + retransmissões) igual a “R/2” teremos a taxa de chegada de dados = “R/3”, ou seja, 0.333 * R são dados originais e 0.166*R são retransmissões (2º Gráfico)



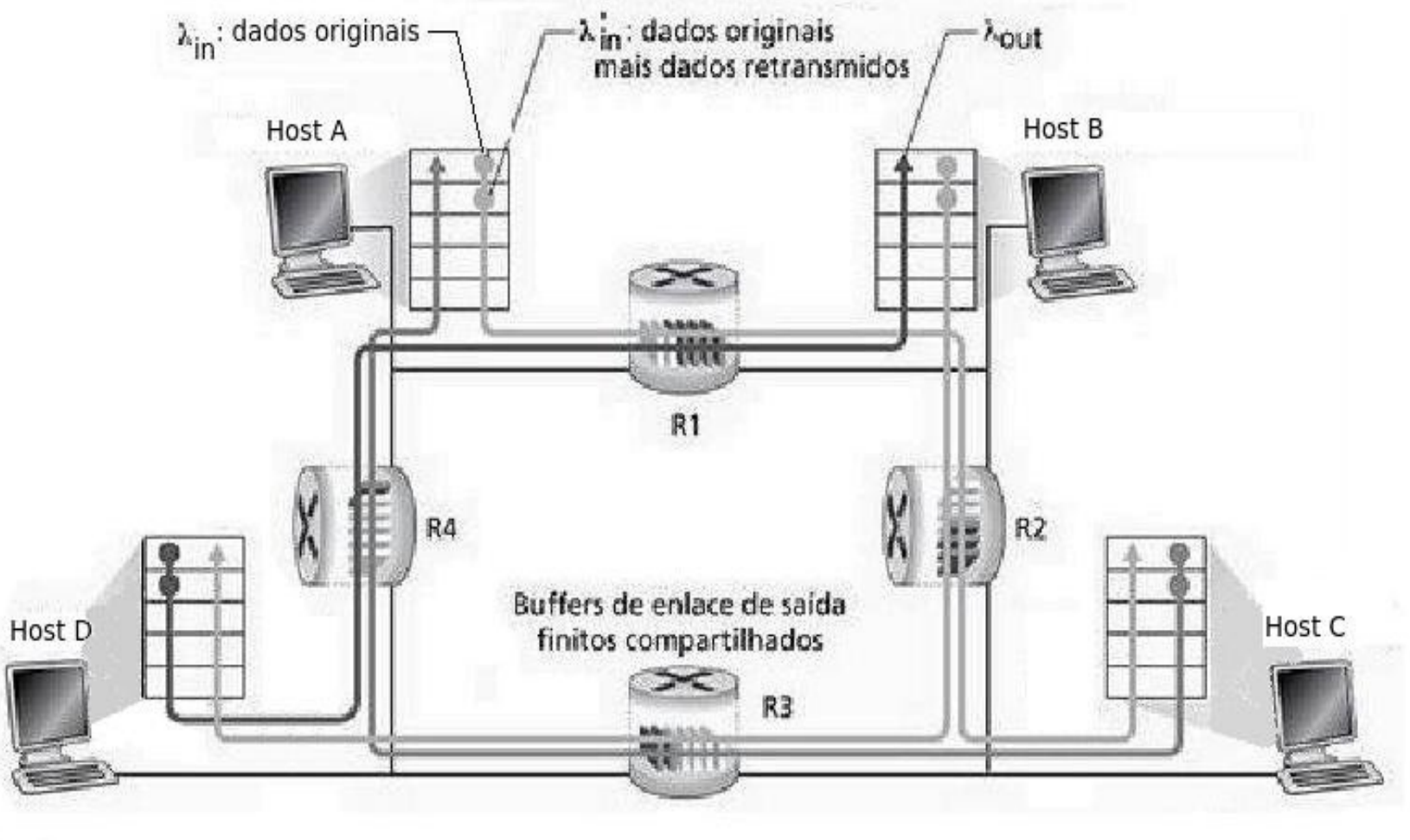
… 3.6.1 – Causas e Custos do Congestionamento

- … há um custo adicional referente as retransmissões feitas pelo remetente face aos grandes atrasos, o que exige do roteador o uso de sua largura de banda para repassar cópias desnecessárias.
- … teremos uma vazão com valor assintótico de “R/4” à medida que a carga oferecida se aproxima de “R/2” (3º Gráfico).



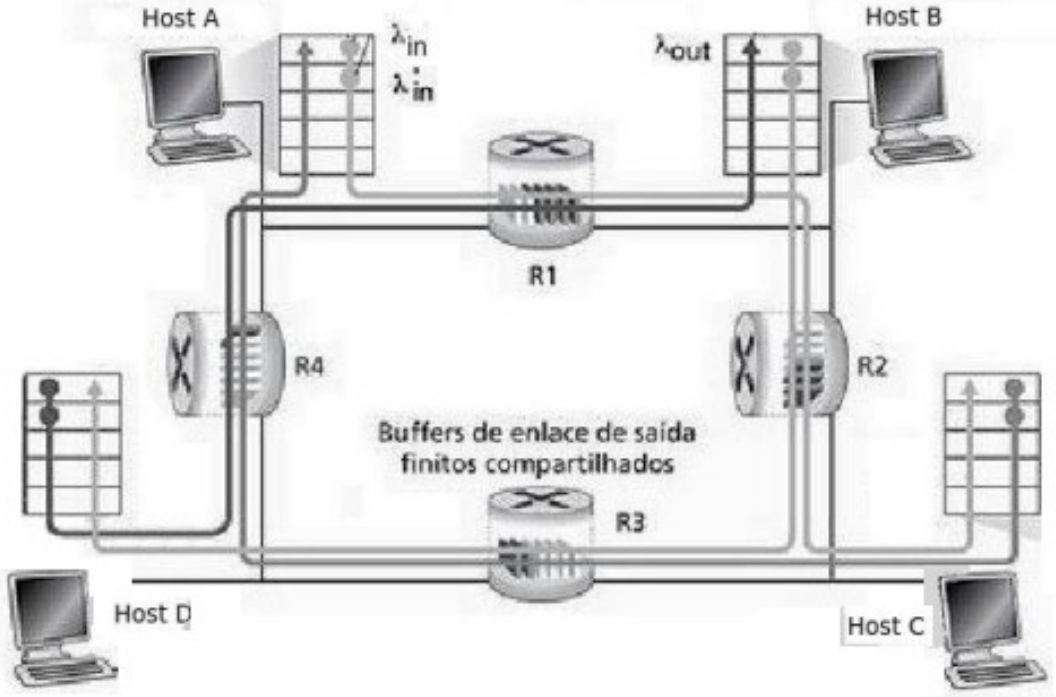
… 3.6.1 – Causas e Custos do Congestionamento

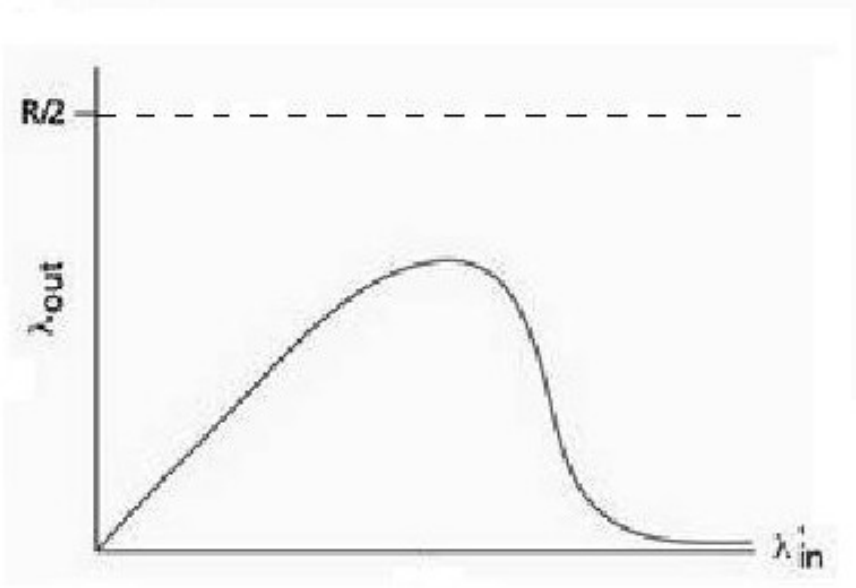
- **Cenário #3** .. 04 remetentes e 04 roteadores c/ “buffers” finitos estão organizados de modo que cada trajeto contemple 02 saltos.



… 3.6.1 – Causas e Custos do Congestionamento

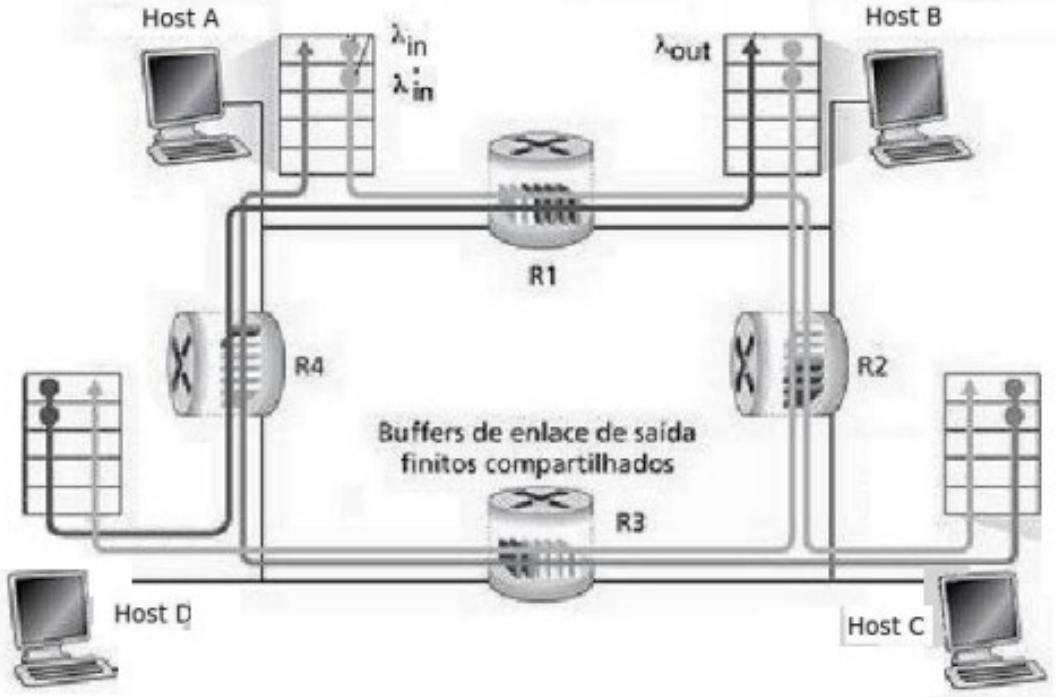
- e.g. .. considere a conexão entre “A” e “C” que passa por “R_1” e “R_2”, que por sua vez compartilha “R_1” com a conexão entre “D” e “B.
- .. para valores pequenos de λ_{in} são raros os casos de esgotamento de buffers, assim a vazão é igual a carga oferecida.

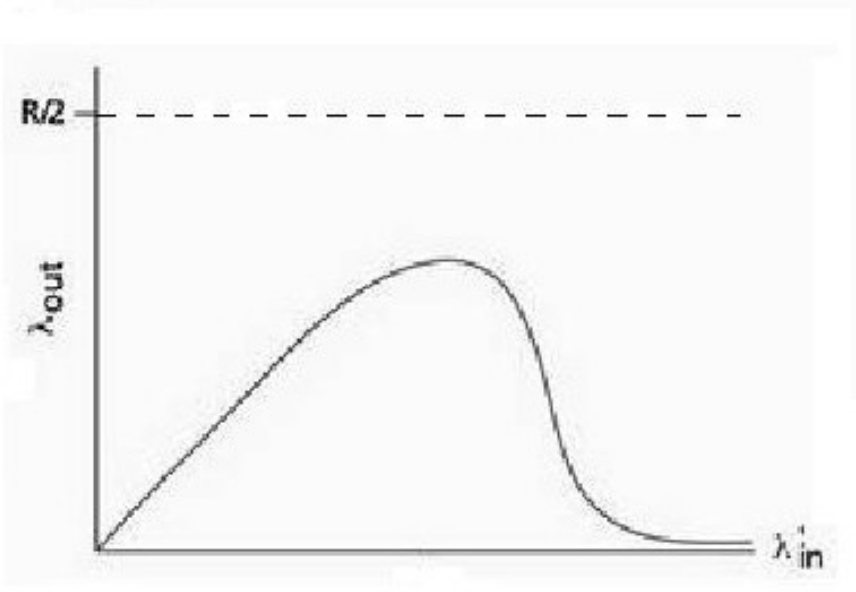




… 3.6.1 – Causas e Custos do Congestionamento

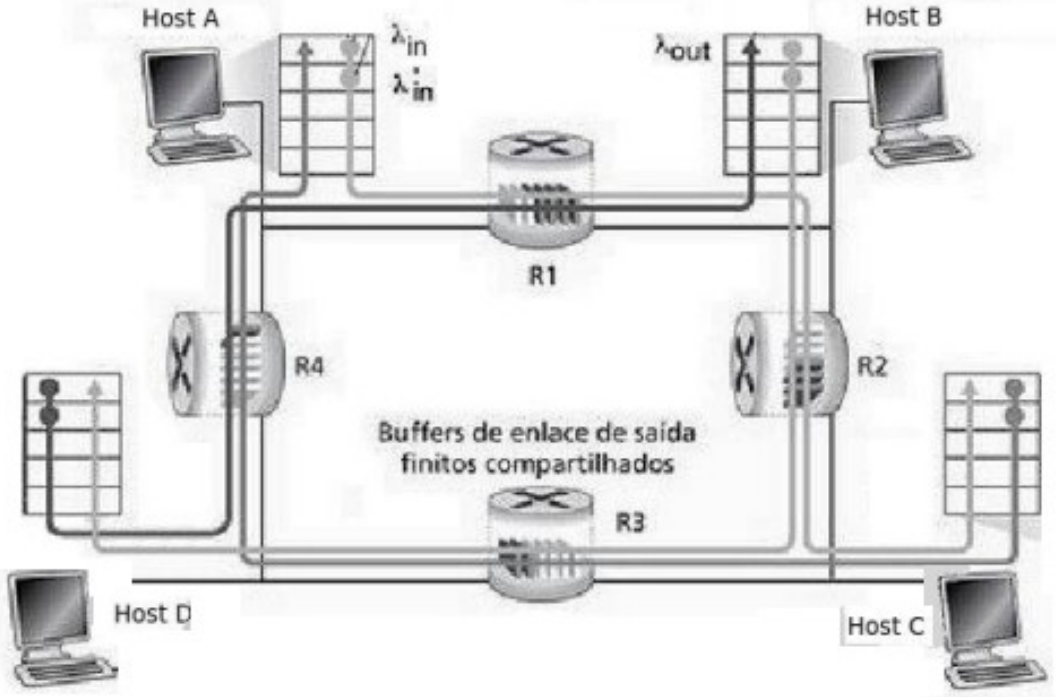
- .. para valores λ_{in} ligeiramente maiores tem-se uma vazão correspondente maior uma vez que mais dados originais são transmitidos.
- .. com λ'_{in} extremamente alto para todas as conexões, então a taxa de chegada do tráfego “B-D” em “R_2” poderá ser >> do que em “A-C”

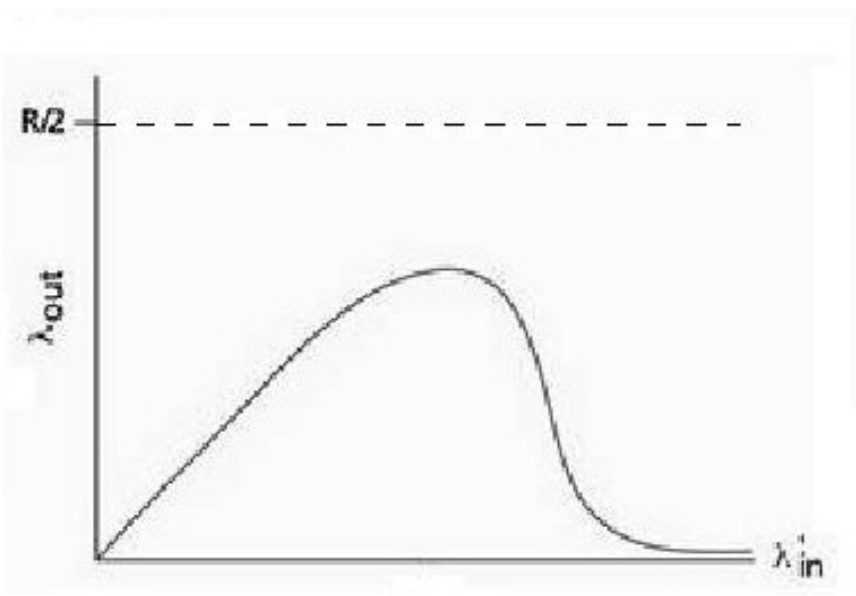




… 3.6.1 – Causas e Custos do Congestionamento

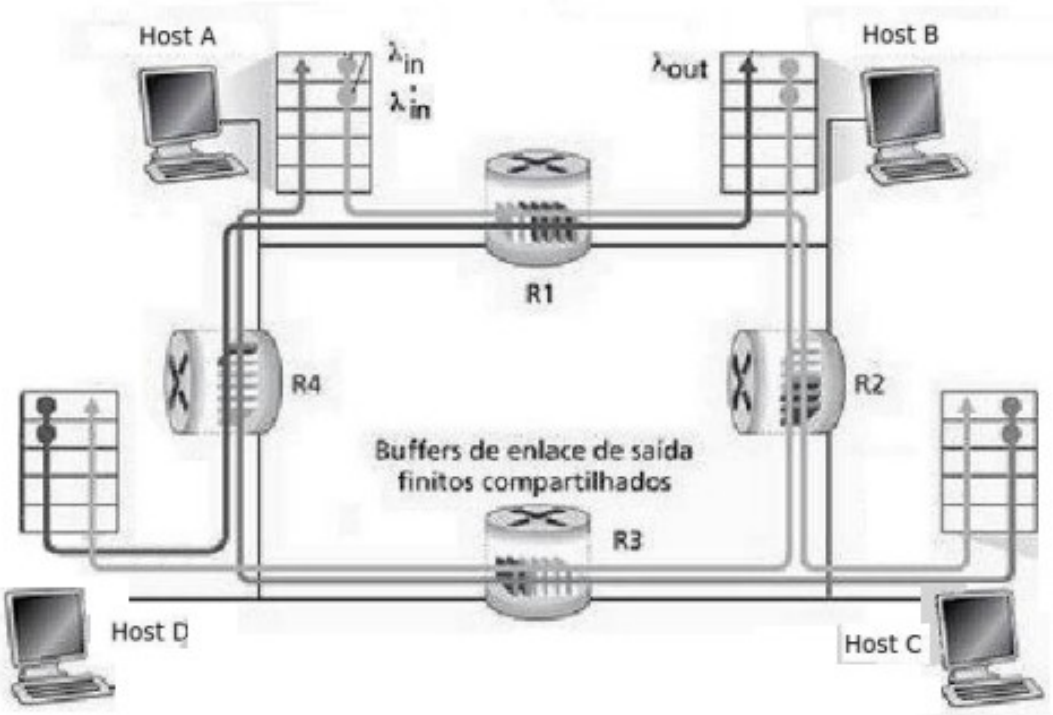
- .. como tráfegos “A-C” e “B-D” competem em “R_2” pelo espaço limitado no buffer, a quantidade de “A-C” que consegue passar por “R_2” diminui cada vez mais com o aumento da carga de “B-D”.

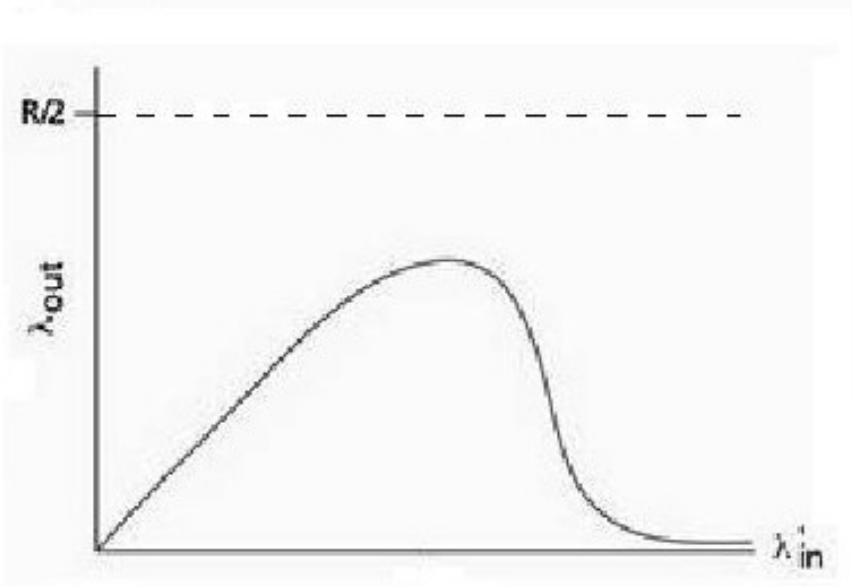




… 3.6.1 – Causas e Custos do Congestionamento

- .. a medida que a carga se aproxima do infinito, um buffer vazio em "R_2" é imediatamente preenchido em"B-D" e a vazão de "A-C" cai a 0.





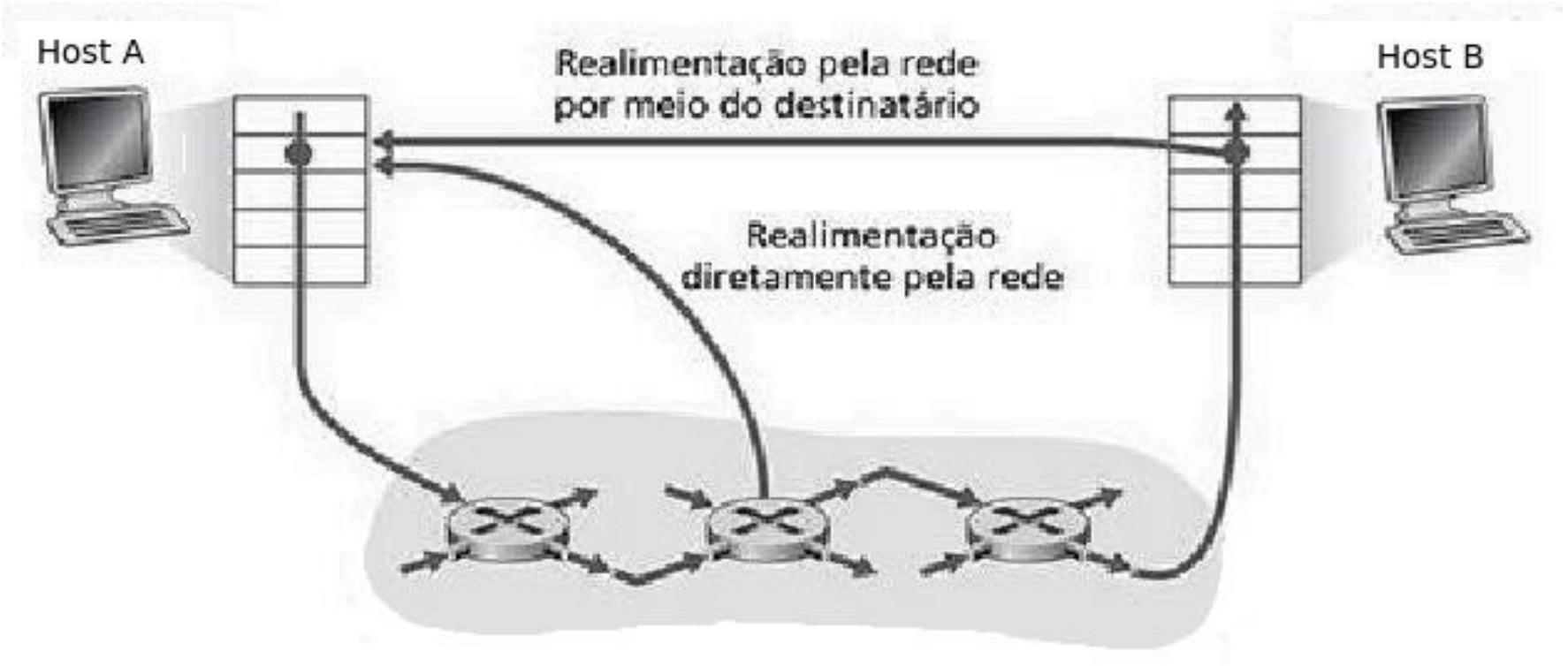
3.6.2 – Mecanismo de Controle de Cong.

- "**mecanismos de controle de congestionamento**" .. podem ser categorizados conforme suporte / apoio explícito ou não da camada de rede para a camada de transporte.

- "**controle de congestionamento fim-a-fim**" .. camada de rede não oferece nenhum suporte explícito à camada de transporte.
- "**controle de congestionamento assistido pela rede**" .. camada de rede fornece realimentação específica de informações acerca do estado de congestionamento da rede ao remetente.

… 3.6.2 – Mecanismo de Controle de Cong.

- “**mecanismos de controle de congestionamento**” .. abordagens de realimentação quanto aos dados sobre congestionamento indicados pela rede, mas não necessariamente informados pela rede.



… 3.6.2 – Mecanismo de Controle de Cong.

- “**controle de congestionamento fim-a-fim**” - camada de rede não oferece nenhum suporte explícito à camada de transporte.
- .. camada de transporte assume o mecanismo de controle fim-a-fim e assume a perda de segmentos como indicação de congestionamento.
- .. camada de transporte pode reduzir o tamanho da janela.

- “**controle de congestionamento assistido pela rede**” - camada de rede fornece realimentação específica de informações acerca do estado de congestionamento da rede ao remetente.
- .. ATM ABR permite que o roteador informe explicitamente ao remetente a taxa de transmissão que o roteador pode suportar no enlace.

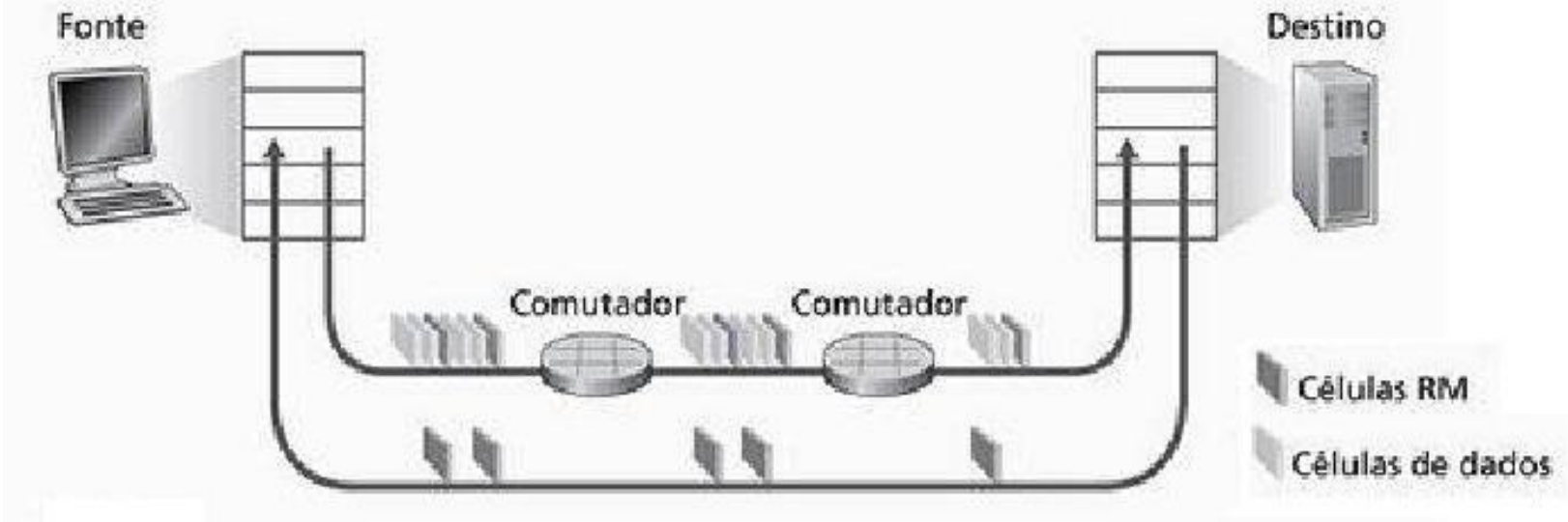
3.6.3 – Controle de Cong. ATM ABR

- ABR (Available Bit Rate) .. abordagem de controle de congestionamento notavelmente diferente da adotada pelo TCP.
- … ATM adota uma abordagem de comutação de pacotes orientada para circuito virtual, ou seja, cada comutador entre a fonte e o destino mantém estado do circuito virtual.
- … estado do circuito virtual em comutadores de rede torna-os ideais para realizar controle de congestionamento assistido pela rede.

- ABR .. serviço de transferência de dados elástico tira proveito da largura de banda disponível quando a rede está vazia.
- … quando está congestionada, limita a taxa de transmissão para algum valor mínimo predeterminado de taxa de transmissão.

… 3.6.3 – Controle de Cong. ATM ABR

- ATM ABR .. células de dados são transmitidas de uma fonte a um destino por meio de comutadores intermediários.
- … no meio das células de dados encontramos as células de gerencia-mento (RM – Resource Management) que contém informações acerca do congestionamento da rede.
- … células de gerenciamento fornecem realimentação direta da rede e também realimentação da rede por intermédio do destinatário.



… 3.6.3 – Controle de Cong. ATM ABR

- Serviço ABR provê 03 mecanismos para sinalizar informações relacionadas ao congestionamento dos comutadores:
- (1) Bit EFCI (Explicit Forward Congestion Indication) .. comutador pode modificar este bit dentro de célula de dados para “1” a fim de sinalizar congestionamento ao destinatário.
- … quando o destinatário recebe células de dados com EFCI = 1, o mesmo modifica células RM através do Bit CI e as envia ao remetente.
- … usando bits EFCI em células de dados e CI em células RM, remetente pode ser notificado sobre congestionamento por um comutador.

- Obs.: CI (Congestion Indicator) e NI (No Increase) .. como já mencionado, células RM (Resource Management) remetente / destinatário estão intercaladas com células dados.

… 3.6.3 – Controle de Cong. ATM ABR

- Serviço ABR provê 03 mecanismos para sinalizar informações relacionadas ao congestionamento dos comutadores:
- Obs.: .. normalmente tem-se uma célula RM a cada 32 células de dados, não obstante a taxa de intercalação é um parâmetro ajustável.

- (2) Células RM podem ter os bits NI e CI alterados para “1” nos comutadores do circuito para indicar congestionamento leve ou severo.
- … destinatário repassa ao remetente as células RM que recebeu com os seus bits CI e NI alterados ou não pelos comutadores.
- … remetente ao receber células RM, pode atuar na taxa de envio de células de dados tendo por base as inform/es de congest/o.

… 3.6.3 – Controle de Cong. ATM ABR

- Serviço ABR provê 03 mecanismos para sinalizar informações relacionadas ao congestionamento dos comutadores:

- (3) Bit ER (Explicit Rate) .. células RM contém campo de 02 bytes;
- .. comutador congestionado pode reduzir o valor contido no campo ER de uma célula RM que está passando pelo comutador.
- .. campo ER será ajustado para a taxa mínima suportável por todos os comutadores no trajeto fonte-destino.
- .. por fim, o remetente recebe a célula RM e se adequa a nova taxa.

3.7 – Controle de Congestionamento no TCP

- TCP usa controle de congestionamento fim-a-fim em vez de controle de congestionamento assistido pela rede.
- .. cada remetente limita a taxa na qual envia tráfego para sua conexão como uma função do nível percebido de congestionamento de rede.

- “**questionamentos a serem respondidos**” !!??
- como limitar a taxa na qual envia tráfego pela conexão ?
- como perceber que há congestionamento na rede ?
- que algoritmo utilizar para modificar a taxa de envio como uma função do congestionamento fim-a-fim ?

… 3.7 – Controle de Congestionamento no TCP

- "**limitação da taxa de envio**" .. ao monitorar uma variável da janela de congestionamento "cwnd", é possível limitar a taxa na qual um remetente envia tráfego para dentro da rede.
- "**idéia**" .. quantidade de dados não reconhecidos em um remetente não pode exceder o mínimo entre "cwnd" e "rwnd", ou seja:
- LastByteSent – LastByteAcked ≤ min(cwnd, rwnd)

- Obs.: … para focalizar a discussão no controle de congestionamento, admite-se que o buffer de recepção é grande o suficiente, logo a limitação da janela de recepção é desprezada.
- … ou seja, quantidade de dados não reconhecidos no remetente estará limitada exclusivamente por "cwnd".

… 3.7 – Controle de Congestionamento no TCP

- … se considerarmos que perdas e atrasos são desprezíveis em uma conexão, podemos enviar “cwnd” bytes no início de cada RTT e na sequência aguardar o seu reconhecimento.
- “**taxa de envio do remetente**” … é aproximadamente “cwnd/RTT” bytes/segundo, assim, ajustando o valor de “cwnd” é possível ajustar a taxa na qual envia dados pela conexão.

- “**perscepção do congestionamento**” .. como um remetente percebe que há congestionamento no caminho entre fonte e destino ?!
- “perda” .. ocorrência de esgotamento de temporização.
- “perda” .. recebimento de 03 ACKs duplicados do destinatário.

… 3.7 – Controle de Congestionamento no TCP

- "**perscepção do congestionamento**" .. como um remetente percebe que há congestionamento no caminho entre fonte e destino ?!

- "**ocorrências**" .. quando há congestionamento excessivo, um ou mais buffers de roteadores ao longo do caminho transborda, fazendo com que um datagrama contendo segmento TCP seja descartado.
- .. datagrama descartado causa evento de perda no remetente, o que é interpretado como uma indicação de congestionamento.

- ?! como o TCP usa reconhecimento para acionar ou regular aumento do tamanho da janela de congestionamento ?!

… 3.7 – Controle de Congestionamento no TCP

- Princípios do Controle de Congestionamento TCP:
- "**segmento perdido**" .. implica em congestionamento, assim a taxa do remetente deve diminuir quando um segmento é perdido.
- "**pergunta**" .. como o remetente deve diminuir sua janela de congestionamento e, portanto, a taxa de envio em resposta ao suposto (pode ser que ACK esteja atrasado) evento de perda ??

- "**segmento reconhecido**" .. indica que a rede está encaminhando os datagramas, assim, taxa do remetente pode aumentar quando um ACK chegar para um segmento anteriormente não reconhecido.
- … chegada de reconhecimento é uma indicação absoluta de que os segmentos estão sendo enviados com sucesso.

… 3.7 – Controle de Congestionamento no TCP

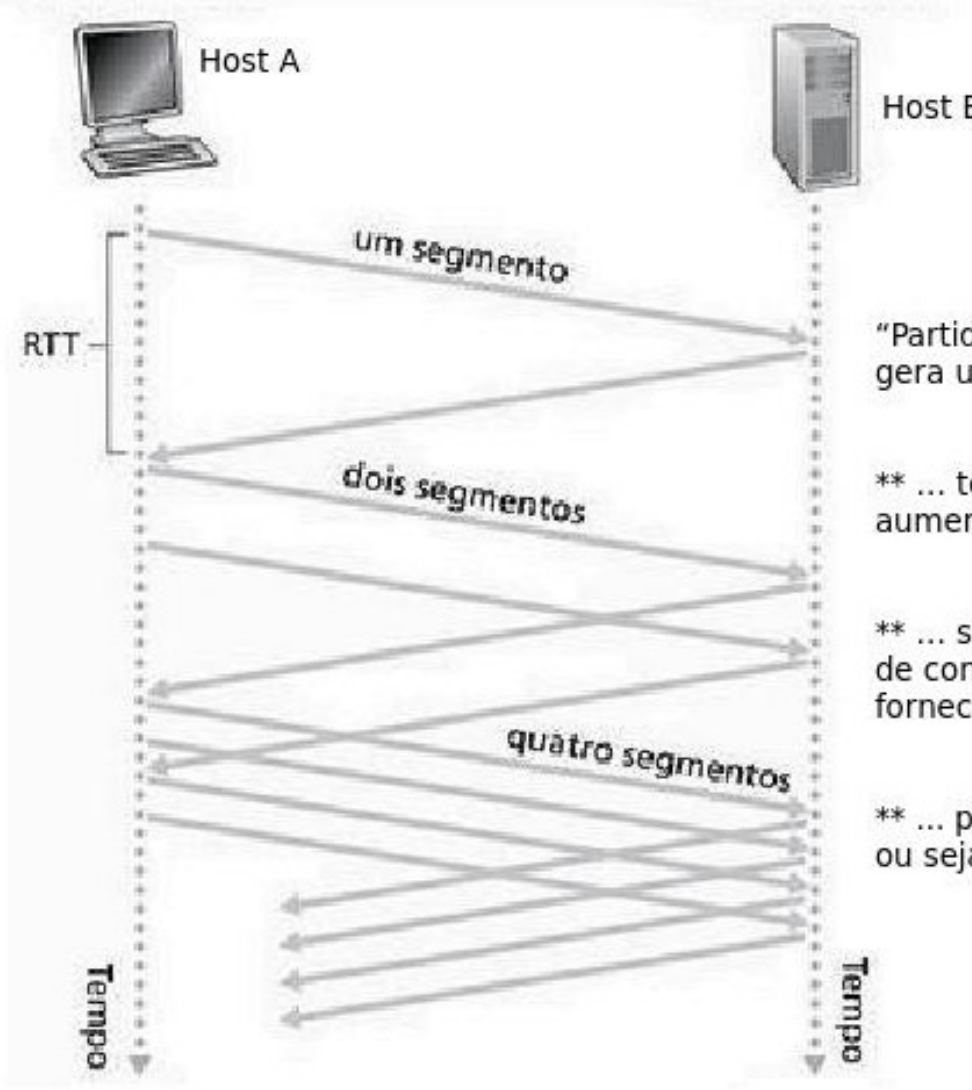
- Princípios do Controle de Congestionamento TCP:
- "**largura de banda**" - … aumenta-se a taxa de envio a medida que ACKs são recebidos até que um evento de perda se faça presente, momento em que a taxa de transmissão deve diminuir.
- .. aumenta-se a taxa de envio para encontrar a taxa na qual inicia-se o congestionamento, e na sequência recua-se dessa taxa.
- .. não há nenhuma sinalização explícita de congestionamento, ou seja, ACKs e eventos de perdas servem como sinais implícitos.

- Algoritmo p/ Controle de Congestiamento: (1) "partida lenta"; (2) "contenção do congestionamento"; (3) "recuperação rápida".

… 3.7 – Controle de Congestionamento no TCP

- "**Partida Lenta**" - ... "cwnd" normalmente é iniciado em 01 MSS, o que gera um taxa de envio de "MSS/RTT".
- … toda vez que 01 segmento transmitido é reconhecido, TCP aumenta tamanho da janela de 01 MSS e envia 02 segmentos.
- … se esses segmentos são reconhecidos, remetente aumenta a janela de congestionamento de 01 MSS para cada reconhecimento de seg., fornecendo uma janela de congestionamento de 04 MSSs.
- … este processo se repete, o que causa a multiplicação da taxa de envio a cada RTT, ou seja, embora a taxa seja lenta no início, o crescimento é exponencial e, portanto, muito rápido.

... 3.7 – Controle de Congestionamento no TCP



"Partida Lenta" - ... "cwnd" normalmente é iniciado em 01 MSS, o que gera um taxa de envio de "MSS/RTT";

** ... toda vez que 01 segmento transmitido é reconhecido, TCP aumenta tamanho da janela em mais 01 MSS e envia 02 segmentos;

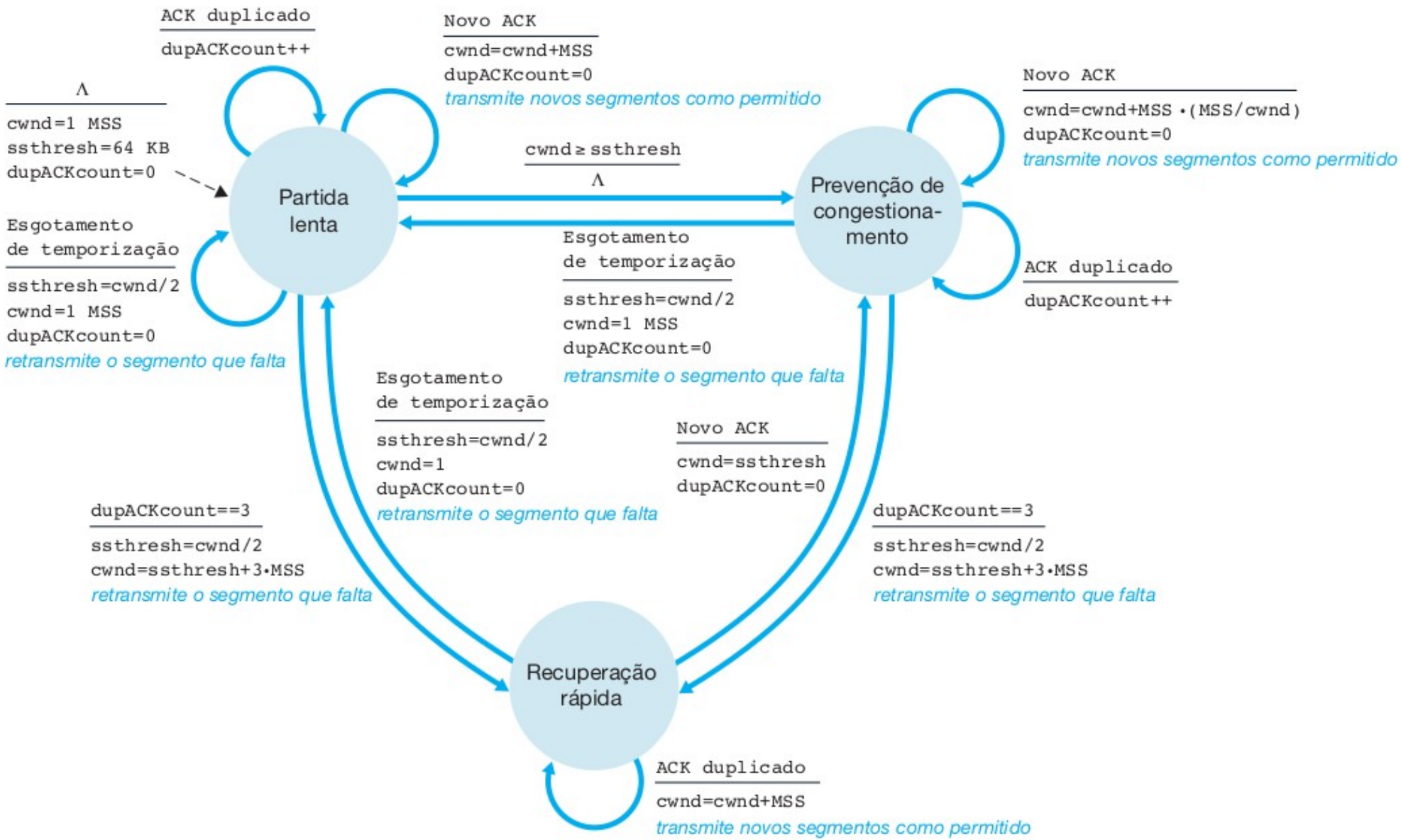
** ... se esses segmentos são reconhecidos, remetente aumenta a janela de congestionamento de 01 MSS para cada reconhecimento de segmento, fornecendo uma janela de congestionamento de 04 MSSs;

** ... processo resulta em uma multiplicação da taxa de envio a cada RTT, ou seja, embora a taxa seja lenta no início, o crescimento é exponencial.

… 3.7 – Controle de Congestionamento no TCP

- Descrição do FSM do Controle de Congestionamento:
- "**evento de perda**" **-** … se houver perda, remetente estabelece o valor de "cwnd" para 01 MSS e inicia o processo de partida lenta.
- .. altera variável "ssthresh" (slow start threshold) para "cwnd/2", ou metade do valor da janela de congestionamento quando da ocorrência do evento de perda (congestionamento detectado).
- .. "ssthresh" é alterado para "cwnd/2", pois seria uma atitude precipitada continuar a duplicar "cwnd" ao atingir ou ultrapassar o "ssthresh".
- .. como será visto, o aumento de "cwnd" se dá com mais cautela quando está no modo de "**prevenção do congestionamento**".
- .. se 03 ACKs duplicados forem detectados, tem-se transição do estado de "**partida lenta**" para o estado de "**recuperação rápida**".

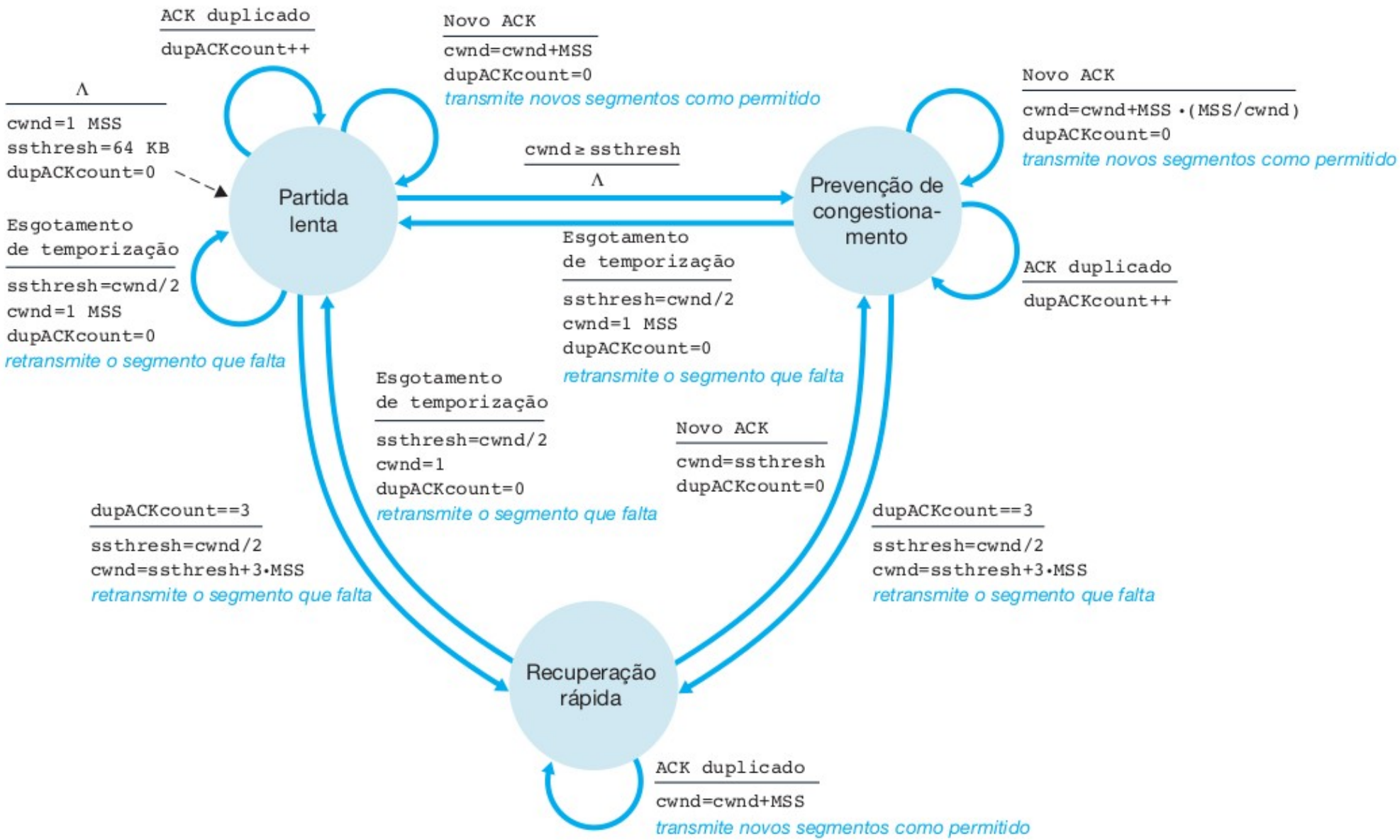
… 3.7 – Controle de Congestionamento no TCP



… 3.7 – Controle de Congestionamento no TCP

- "**Prevenção de Congestionamento**" ... atenção para "ssthresh" !!
- … ao atingir este estado, "cwnd" é ½ do seu valor quando o congestionamento foi encontrado pela última vez.
- … remetente aumenta "cwnd" por MSS no momento em que um novo reconhecimento chega, e.g., seja MSS = 1460 bytes e cwnd = 14600 bytes, então 10 segmentos são enviados dentro de um 01 RTT;
- … cada ACK que chega aumenta o tamanho da janela de cong. em 1/10 MSS, assim, valor da janela de cong. terá aumentado em 01 MSS após os ACKs quando todos os segmentos tiverem sido recebidos.
- … reduz valor "cwnd" para ½ e adiciona 03 MSSs para contabilizar ACKs DUP triplos recebidos, na sequência, registra o valor "ssthresh" como ½ de "cwnd" quando os 03 ACKs DUP triplos foram recebidos.

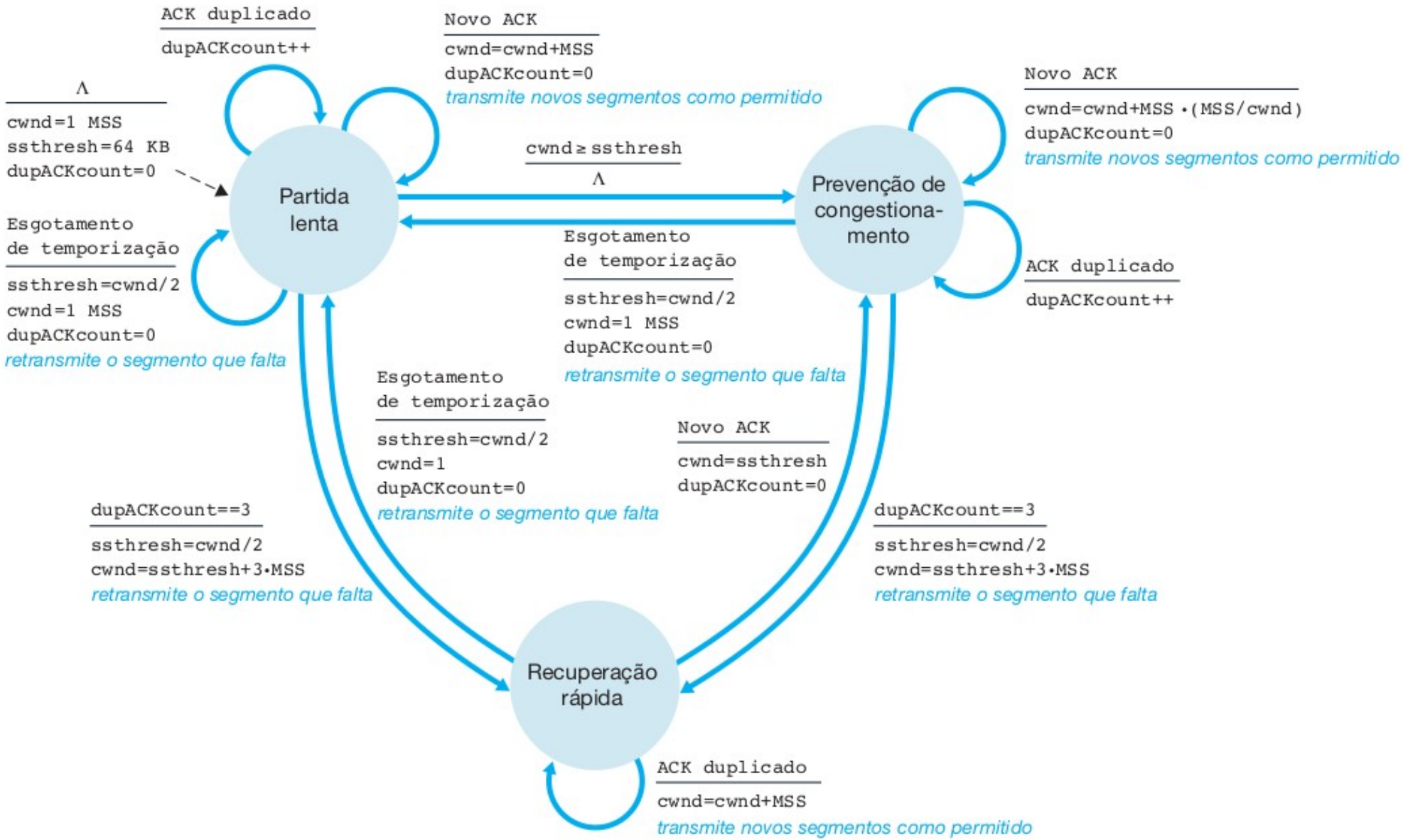
… 3.7 – Controle de Congestionamento no TCP



… 3.7 – Controle de Congestionamento no TCP

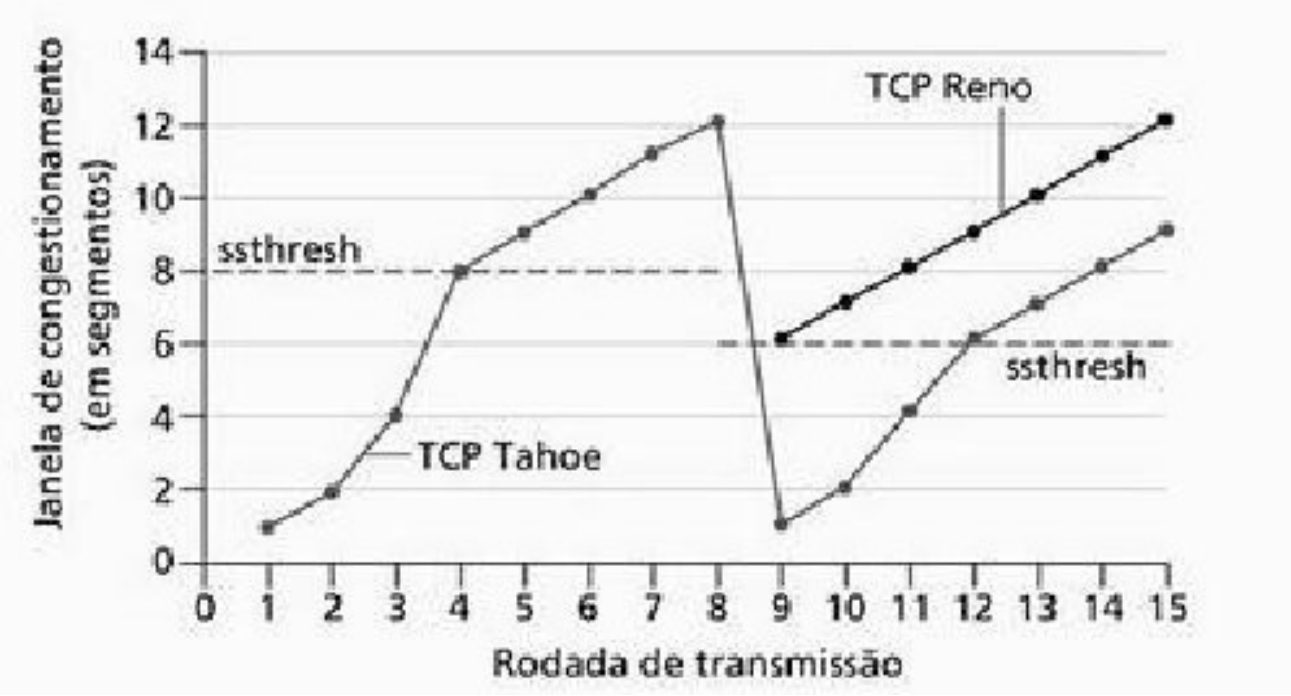
- “**Recuperação Rápida**” ...
- … valor de “cwnd” é incrementado de 01 MSS para cada ACK duplicado recebido no segmento perdido que fez com que o TCP entrasse no modo de recuperação rápida (ACK DUP triplo).
- … quando ACK chega no segmento perdido, TCP entra no modo de prevenção de congestionamento após reduzir “cwnd”.
- … se esgotamento de temporização ocorrer, a recuperação rápida é alterada para o modo partida lenta.

… 3.7 – Controle de Congestionamento no TCP



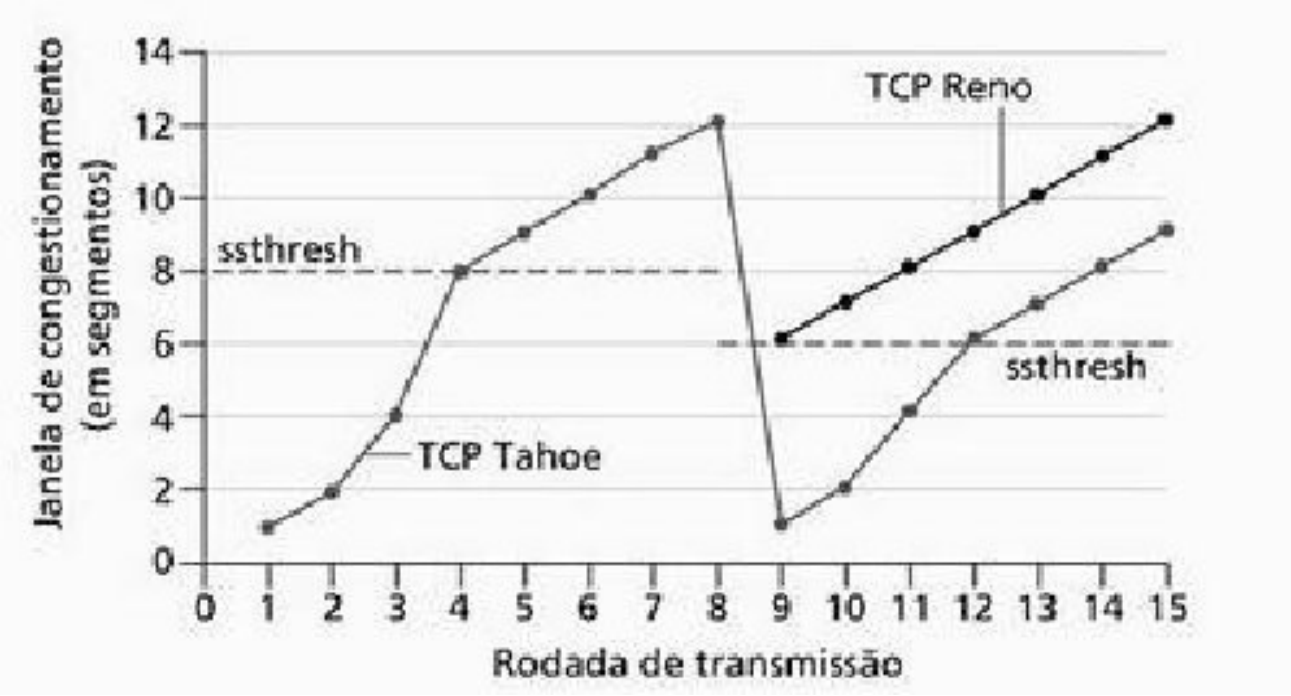
… 3.7 – Controle de Congestionamento no TCP

- TCP para versões Reno e Tahoe .. “ssthresh” inicia com 08 * MSSs e ambos apresentam o mesmo comportamento nas primeiras 08 sessões com elevação exponencial na partida lenta.
- … elevação é linear até que ocorra ACK DUP triplo (08 sessão de transmissão) quando então a janela de cong. é 12 MSSs



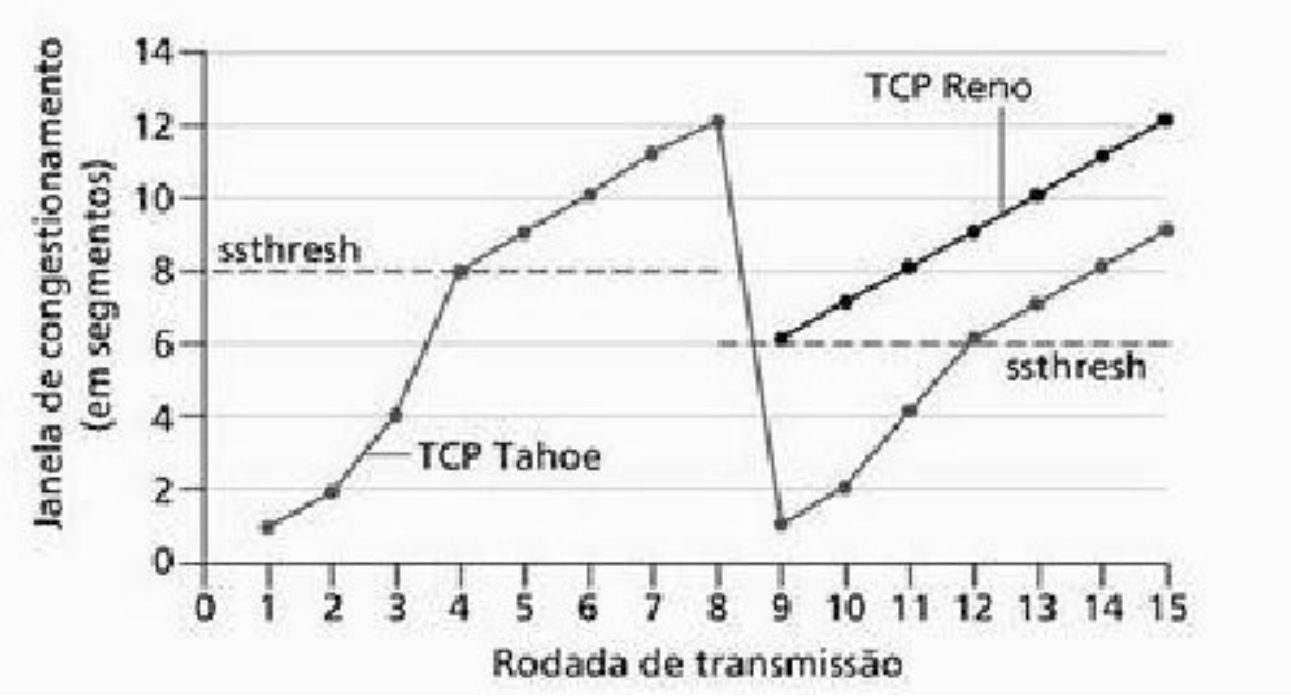
… 3.7 – Controle de Congestionamento no TCP

- "**retrospectiva do controle de congestionamento**" - ...
- .. quando uma conexão se inicia e supondo que as perdas são indicadas por ACKs DUP triplos e não por esgotamento de temporização ..
- .. controle de congestionamento consiste em aumento linear da variável "cwnd" de 01 MSS por RTT.



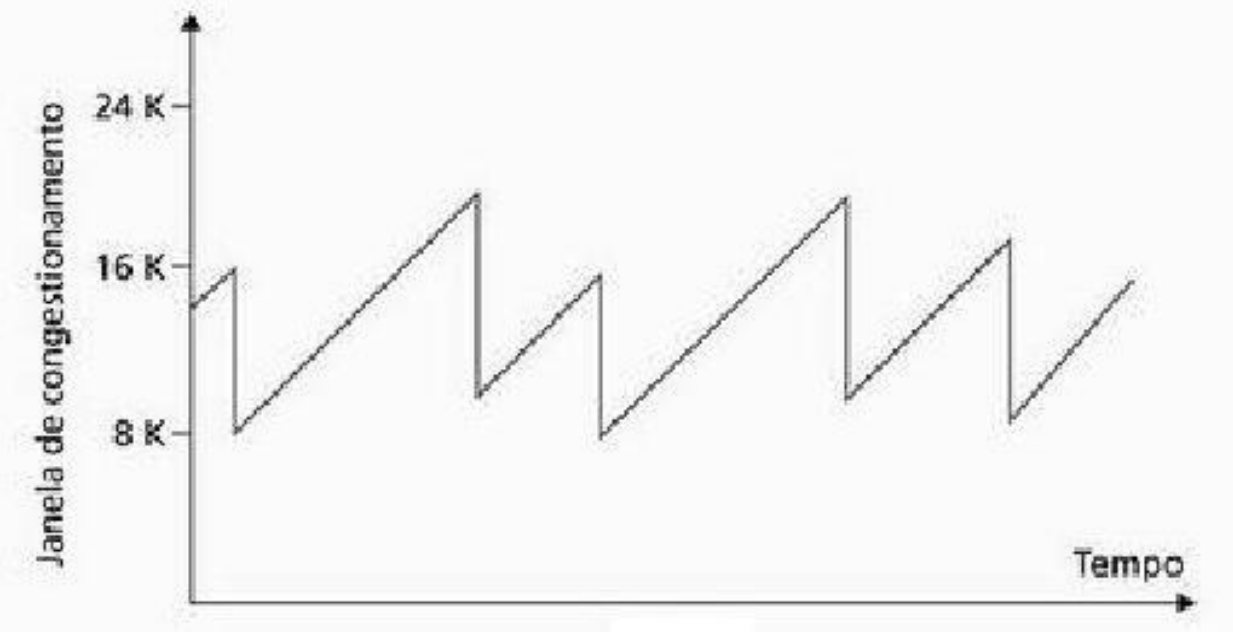
… 3.7 – Controle de Congestionamento no TCP

- "**retrospectiva do controle de congestionamento**" - ...
- … "cwnd" cai pela metade na presença de 01 ACK duplicado triplo, por esta razão, o Cont. de Contestionamento no TCP é denominado "Aumento Aditivo, Diminuição Multiplicativa" (AIMD).



… 3.7 – Controle de Congestionamento no TCP

- "**conclusão**" - aumento linear do tamanho da janela de cong. até que um ACK duplicado triplo ocorra, então, há a redução da janela pela metade e o processo inicia-se novamente.



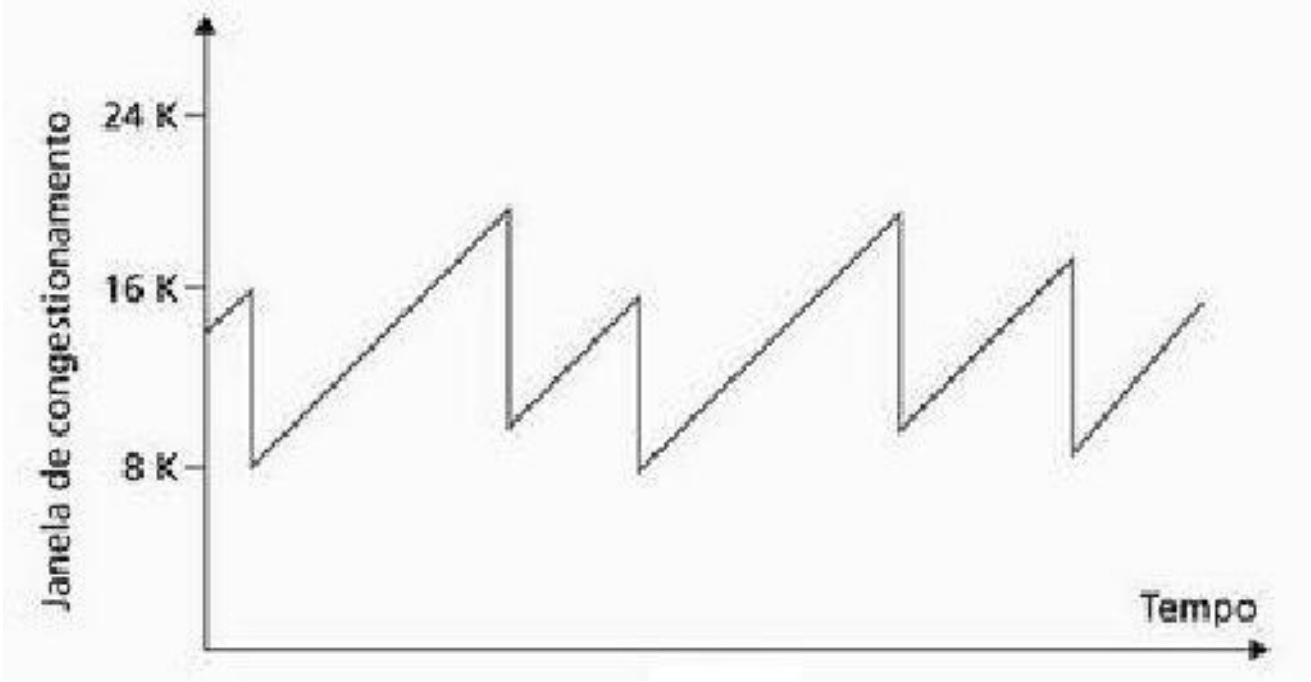
… 3.7 – Controle de Congestionamento no TCP

- Princípios do Algoritmo de Vegas (procura manter a vazão):
- detectar congestionamento nos roteadores entre a fonte e o destino antes que ocorra a perda do datagrama.
- diminuir linearmente a taxa ao ser detectada a perda de datagrama, que pode ser prevista observando-se o RTT.

- Obs.: … quanto maior o RTT dos datagramas, maior será o congestionamento nos roteadores.

… 3.7 – Controle de Congestionamento no TCP

- "**vazão média**" .. para uma conexão ativa em um longo tempo a taxa de envio de dados é um função da janela de congestionamento (e.g., "w" bytes) bem como do RTT corrente.
- .. aproximadamente "w/RTT" bytes / segundo.



… 3.7 – Controle de Congestionamento no TCP

- … como há uma busca constante por largura de banda adicional, através do aumento de 01 MSS a cada RTT, seja “W” o valor “w” quando ocorre o evento de perda.
- .. taxa de transmissão oscila entre “W / (2*RTT)” e “W/RTT”

- … tais suposições conduzem a um modelo macroscópico e simplificado do comportamento do TCP em regime permanente:
- vazão média de uma conexão = 0.75 * W / RTT.

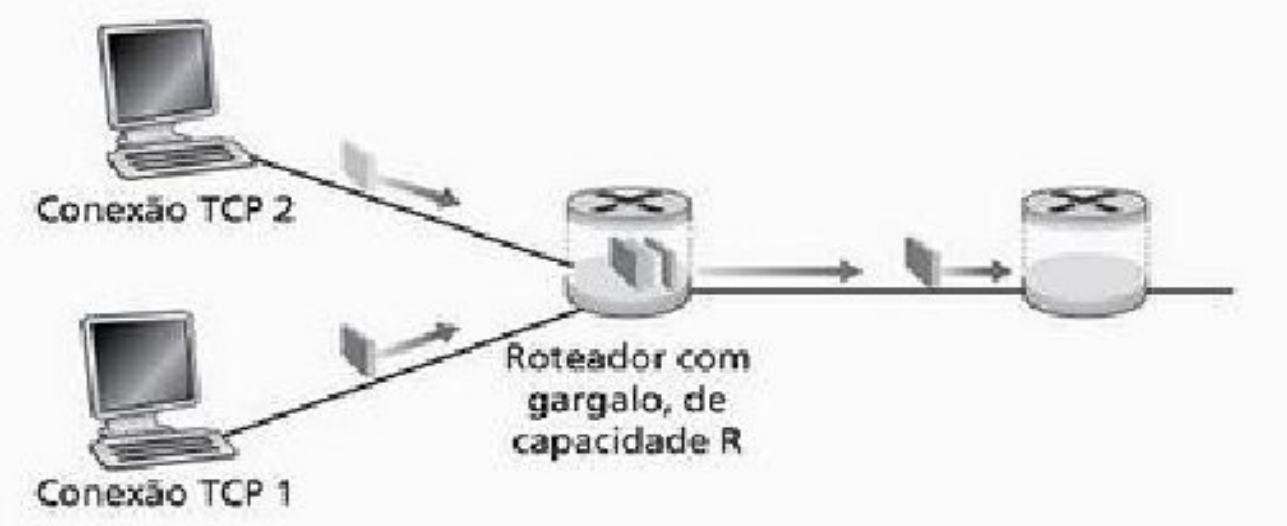
3.7.1 – Equidade no TCP

- e.g., considere “K” conexões cada qual com um caminho fim a fim diferente, mas todas passando por um enlace de “R” bps;
- … nenhum dos outros enlaces no caminho entre fonte e destino está congestionado e, portanto, possuem abundante taxa de transmissão.

- “**provocação**” .. tem-se controle justo de congestionameto ?!
- “**controle de congestionamento justo**” ... taxa média de transmissão de cada conexão é aproximadamente “R/K”, isto é, cada uma obtém uma parcela igual da largura de banda do enlace.

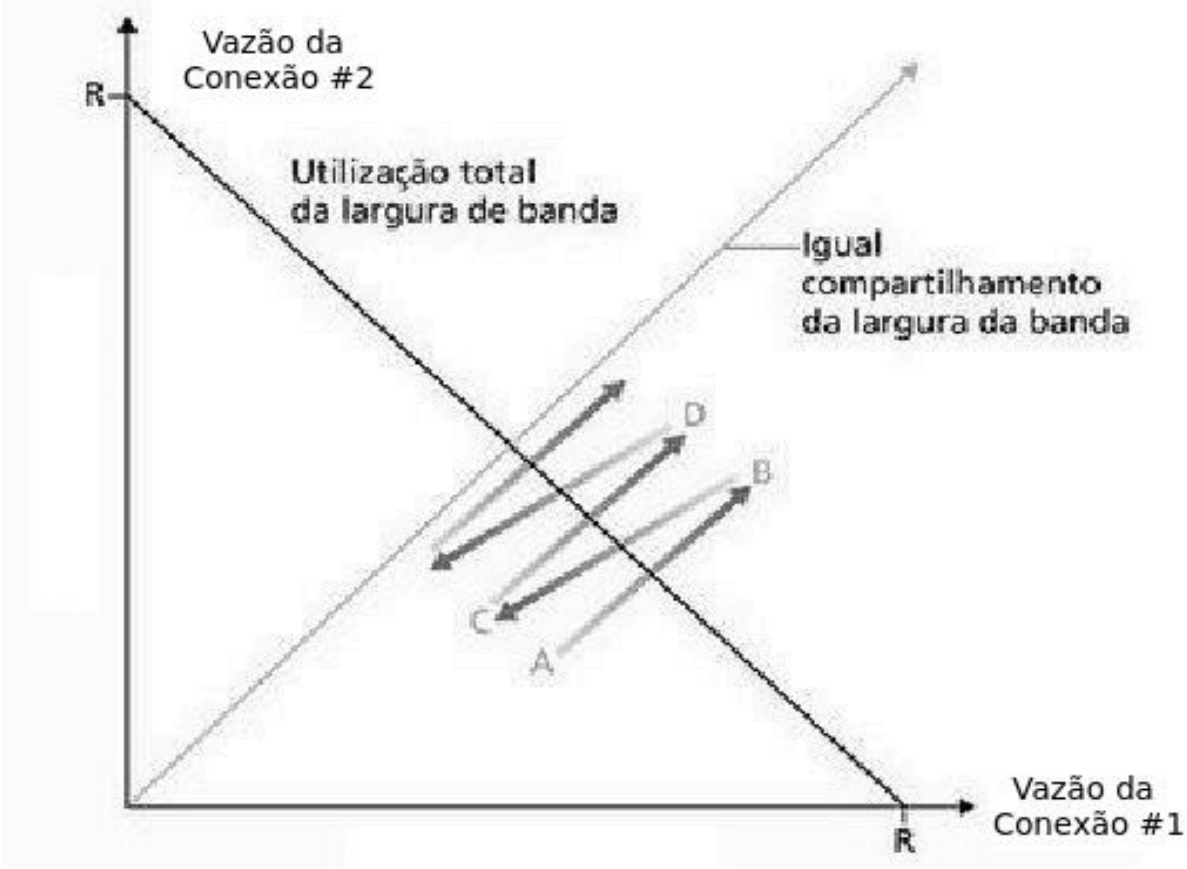
… 3.7.1 – Equidade no TCP

- e.g., considere 02 conexões TCP compartilhando um único enlace com taxa de transmissão “R” como mostra a figura abaixo;
- “**suposição**” .. ambas as conexões tem os mesmos RTT e MSS e nenhuma outra conexão ou datagramas UDP atravessam este enlace.
- … se o tamanho da janela de congestionamento for o mesmo na situação de RTT e MSS e iguais em ambas as conexões, espera-se a mesma vazão para as 02 conexões.



… 3.7.1 – Equidade no TCP

- … se o compartilhamento da largura de banda ocorrer equitativa-mente, então a vazão alcançada deverá cair ao longo da linha de 45 graus que parte da origem (soma das vazões = R bps).



… 3.7.1 – Equidade no TCP

- e.g., suponha que os tamanhos das janelas sejam tais que as conexões #1 e #2 alcancem as vazões indicadas pelo ponto “A”.
- .. como a soma das larguras de banda consumida é menor do que “R”, não teremos perdas e ambas as conexões aumentarão suas janelas como resultado do algoritmo de “prev. do congestionamento”.
- .. como este aumento é igual para as duas (reta de 45 graus) o pto. “B” é atingido após algum tempo, ou seja, vazão maior do que “R”.
- .. se as conexões experimentarem perda de pacotes (Pto. B), ambas as conexões irão reduzir suas janelas por um fator de “2” (Pto. C).
- … este processo se repete (Ptos. D, E, e F) até estabilizar, não importando aonde elas comecem no espaço bidimensional.

… 3.7.1 – Equidade no TCP

- Obs.: … vimos como o TCP regula a taxa de transmissão de uma aplicação por meio do mecanismo de cont. de congestionamento.
- … justamente por esta razão, muitas aplicações não utilizam o TCP, pois não querem que sua taxa de transmissão seja limitada, ainda que a rede esteja congestionada.
- … tais aplicações normalmente utilizam o UDP, pois não será imposto controle de congestionamento ainda que ele exista na rede.

- Aplicações UDP não cooperam e não são justas da Perspectiva do TCP, pois não ajustam suas taxas de transmissão de forma adequada em prol do bom funcionamento de toda a rede.

… 3.7.1 – Equidade no TCP

- … se por um lado não há equidade entre UDP e TCP nesta perspectiva, como impedir que uma aplicação não utilize múltiplas conexões paralelas na tentativa de amenizar o controle de congestionamento ?
- "**fato**" .. aplicação que utiliza múltiplas conexões paralelas consegue uma fração maior da largura de banda de um enlace congestionado.

- e.g. considere 01 enlace "R" bps que suporta 09 conexões em curso tipo cliente-servidor e cada aplicação utiliza 01 conexão;
- … se surgir uma nova aplicação que utiliza conexão, então cada aplicação conseguirá a mesma taxa de transmissão = R/10;
- "**pergunta**" .. em que condição não teremos o balanceamento entre as vazões médias para cada uma das conexões ?!

… 3.7.1 – Equidade no TCP

- e.g. considere 01 enlace R bps que suporta 09 conexões em curso entre cliente e servidor e cada aplicação utiliza 01 conexão.
- … se surgir uma nova aplicação que utiliza conexão, então cada aplicação conseguirá a mesma taxa de transmissão = R/10.
- "**pergunta**" .. em que condição não teremos o balanceamento entre as vazões médias para cada uma das conexões ?!

- … se em vez disso, a nova aplicação utilizar 11 conexões paralelas, então ela conseguirá uma alocação injusta de mais do que "R/2" da capacidade total do enlace.

- Obs.: … múltiplas conexões paralelas não são incomuns.