Redes de Computadores

Camada de Transporte

Professor: Reinaldo Gomes
reinaldo@computacao.ufcg.edu.br

1

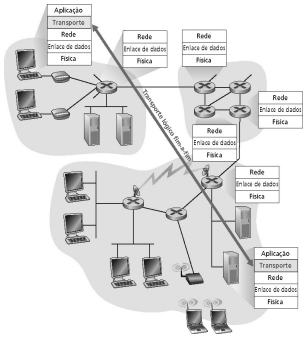
Camada de transporte

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

2

Protocolos e serviços de transporte

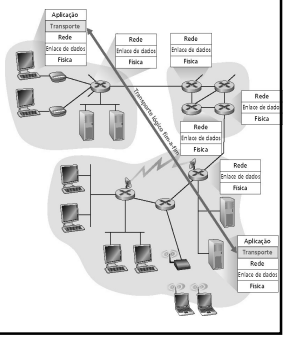
- Fornecem comunicação lógica entre processos de aplicação em diferentes hospedeiros
- Os protocolos de transporte são executados nos sistemas finais
  - Lado emissor: quebra as mensagens da aplicação em segmentos e envia para a camada de rede
  - Lado receptor: remonta os segmentos em mensagens e passa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



3

Protocolos da camada de transporte da Internet

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



4

Principais funções

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

5

Camada de transporte

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

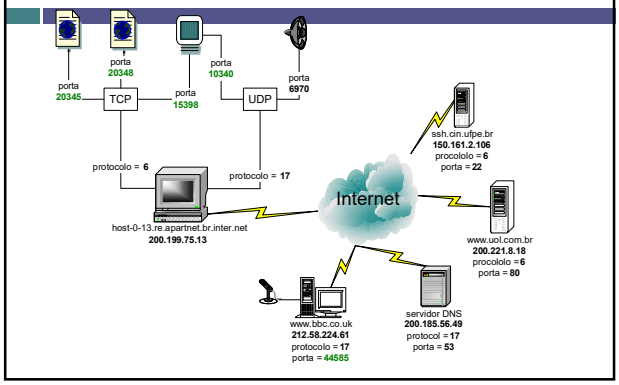
6

Identificação da aplicação no host

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

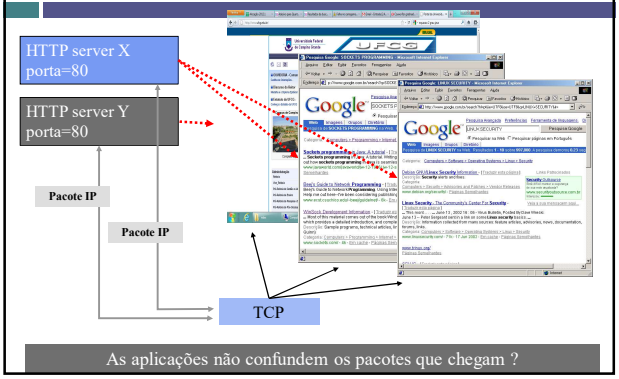
7

Identificação da aplicação no host



8

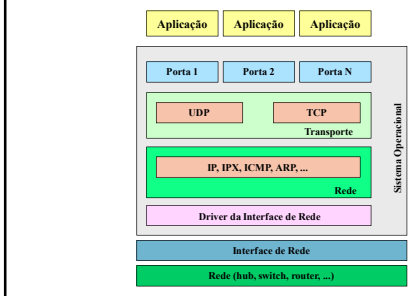
Identificação da aplicação no host



As aplicações não confundem os pacotes que chegam ?

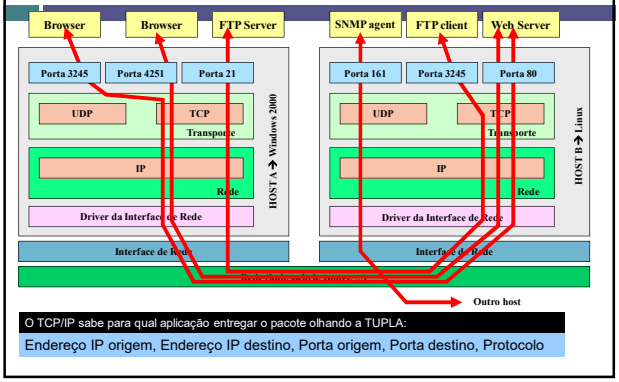
9

Pilha de Protocolos, na prática



10

Pilha de Protocolos, na prática

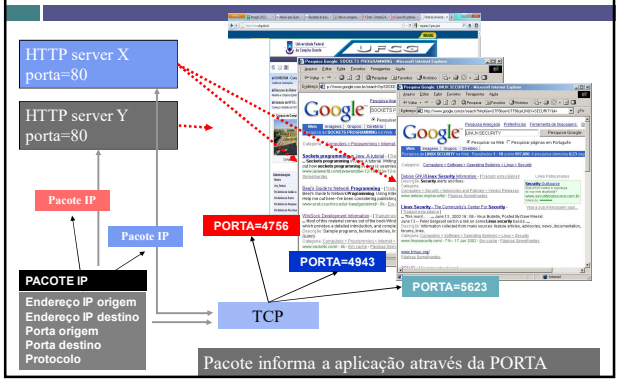


O TCP/IP sabe para qual aplicação entregar o pacote olhando a TUPLA:

Endereço IP origem, Endereço IP destino, Porta origem, Porta destino, Protocolo

11

Identificação da aplicação no host



Pacote informa a aplicação através da PORTA

12

Identificação de aplicações

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

13

Números de portas

- 1-255 — reservadas para serviços padrão portas "bem conhecidas"
- 256-1023 — reservado para serviços Unix
- 1-1023 — Somente podem ser usadas por usuários privilegiados (super-usuário)
- 1024-4999 — Usadas por processos de sistema e de usuário
- 5000- — Usadas somente por processos de usuário

14

Algumas Portas Bem Conhecidas

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

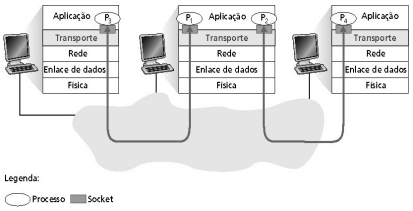
Detalhes em www.iana.org/assignments/port-numbers

15

Demultiplexação/Multiplexação

Demultiplexação no hospedeiro receptor: entrega os segmentos recebidos ao socket correto

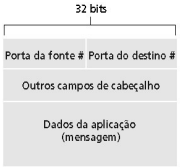
Multiplexação no hospedeiro emissor: coleta dados de múltiplos sockets, envelopa os dados com cabeçalho (usado depois para demultiplexação)



16

Demultiplexação/Multiplexação

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



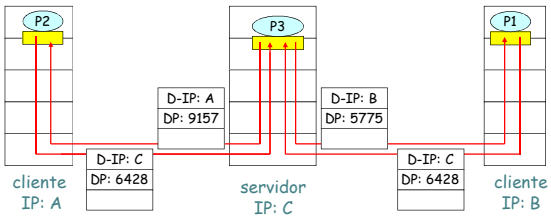
17

Demultiplexação/Multiplexação

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

18

Demultiplexação/Multiplexação



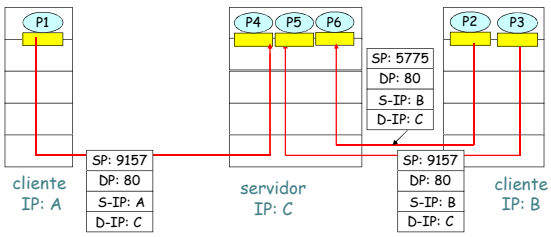
19

Demultiplexação/Multiplexação

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

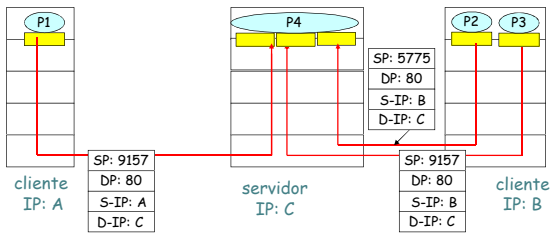
20

Demultiplexação/Multiplexação



21

Demultiplexação/Multiplexação



22

Camada de transporte

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

23

UDP: User Datagram Protocol

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

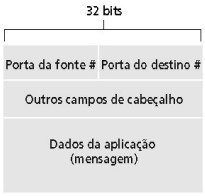
Por que existe um UDP?

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

24

UDP: User Datagram Protocol

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



25

UDP Checksum

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

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

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

26

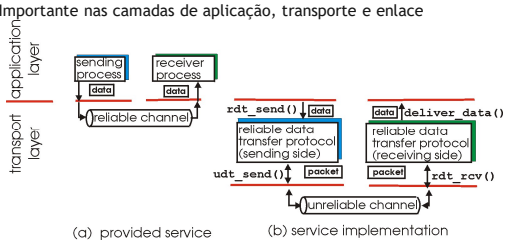
Camada de transporte

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

27

Princícios de transferência confiável de dados

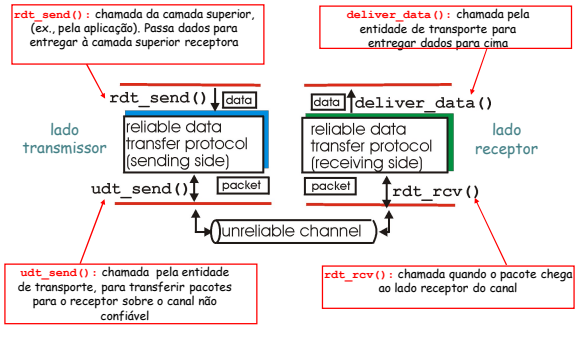
- Importante nas camadas de aplicação, transporte e enlace



- Características dos canais não confiáveis determinarão a complexidade dos protocolos confiáveis de transferência de dados (rdt)

28

Transferência confiável: o ponto de partida



29

Transferência confiável: o ponto de partida

Etapas:
- desenvolver incrementalmente o transmissor e o receptor de um protocolo confiável de transferência de dados (rdt)
- considerar apenas transferências de dados unidirecionais
  - mas informação de controle deve fluir em ambas as direções!
- usar máquinas de estados finitos (FSM) para especificar o protocolo transmissor e o receptor

30

Rdt1.0: transferência confiável sobre canais confiáveis

- canal de transmissão perfeitamente confiável
  - não há erros de bits
  - não há perdas de pacotes
- FSMs separadas para transmissor e receptor:
  - transmissor envia dados para o canal subjacente
  - receptor lê os dados do canal subjacente

(a) rdt1.0: sending side (b) rdt1.0: receiving side

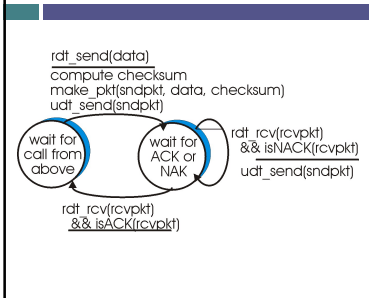
31

Transferência confiável usando um canal com erro de bits

- Canal subjacente pode trocar valores dos bits num pacote
  - Checksum para detectar erros de bits
- *A* questão: como recuperar esses erros:
  - **Reconhecimentos (ACKs):** receptor avisa explicitamente ao transmissor que o pacote foi recebido corretamente
  - **Reconhecimentos negativos (NAKs):** receptor avisa explicitamente ao transmissor que o pacote tem erros
  - Transmissor reenvia o pacote quando da recepção de um NAK
- Mecanismos necessários (rdt2.0):
  - Detecção de erros
  - Retorno do receptor: mensagens de controle (ACK, NAK) rcvr->sender

32

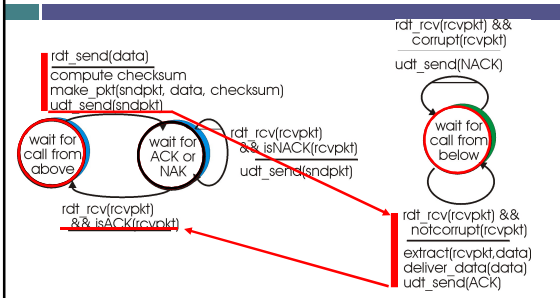
rdt2.0: especificação da FSM



FSM do transmissor FSM do receptor

33

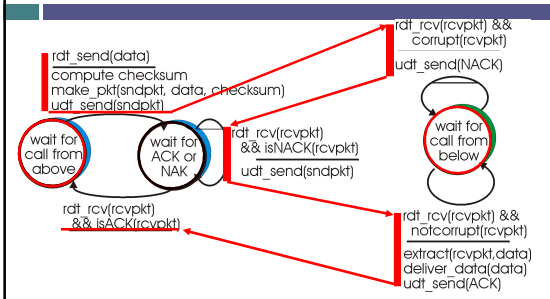
rdt2.0: em ação (ausência de erros)



FSM do transmissor FSM do receptor

34

rdt2.0: em ação (cenário com erros)



FSM do transmissor FSM do receptor

35

Transferência confiável usando um canal com erro de bits e perdas

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

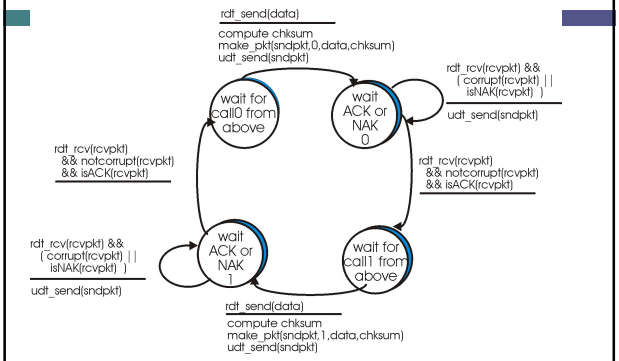
- Transmissor não sabe o que aconteceu no receptor!
- Transmissor deve esperar durante um tempo razoável pelo ACK e se não recebe-lo deve retransmitir a informação
  - Não pode apenas retransmitir: possível duplicata

Tratando duplicatas:

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

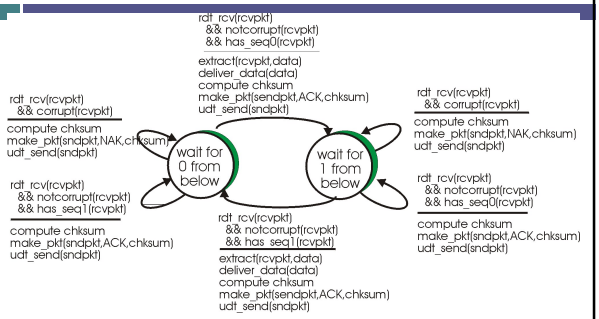
36

rdt2.1: transmissor, trata ACK/NAKs perdidos



37

rdt2.1: receptor, trata ACK/NAKs perdidos



38

rdt2.1: discusssão

Transmissor:

- adiciona número de seqüência ao pacote
- Dois números (0 e 1) bastam. Porque?
- deve verificar se os ACK/NAK recebidos estão corrompidos
- duas vezes o número de estados
  - o estado deve "lembrar" se o pacote "corrente" tem número de seqüência 0 ou 1

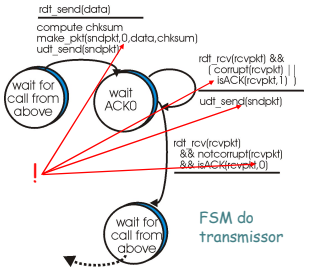
Receptor:

- deve verificar se o pacote recebido é duplicado
  - estado indica se o pacote 0 ou 1 é esperado
- nota: receptor pode não saber se seu último ACK/NAK foi recebido pelo transmissor

39

rdt2.2: um protocolo sem NAK

- mesma funcionalidade do rdt2.1, usando somente ACKs
- ao invés de enviar NAK, o receptor envia ACK para o último pacote recebido sem erro
  - receptor deve incluir explicitamente o número de seqüência do pacote sendo reconhecido
- ACKs duplicados no transmissor resultam na mesma ação do NAK: *retransmição do pacote corrente*



40

rdt3.0: canais com erros e perdas

Nova Hipótese: canal de transmissão pode também perder pacotes (dados ou ACKs)

- checksum, números de seqüência, ACKs, retransmissões serão de ajuda, mas não o bastante

Q: como tratar com perdas?

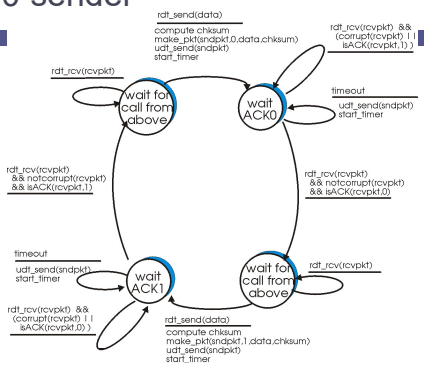
- transmissor espera até que certos dados ou ACKs sejam perdidos, então retransmite
- problemas?

Abordagem: transmissor espera um tempo "razoável" pelo ACK

- retransmite se nenhum ACK for recebido neste tempo
- se o pacote (ou ACK) estiver apenas atrasado (não perdido):
  - retransmissão será duplicata, mas os números de seqüência já tratam com isso
  - receptor deve especificar o número de seqüência do pacote sendo reconhecido
- exige um temporizador decrescente

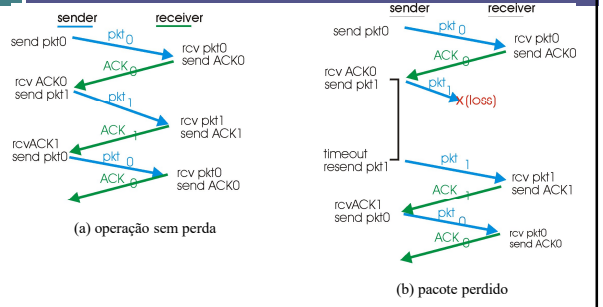
41

rdt3.0 sender



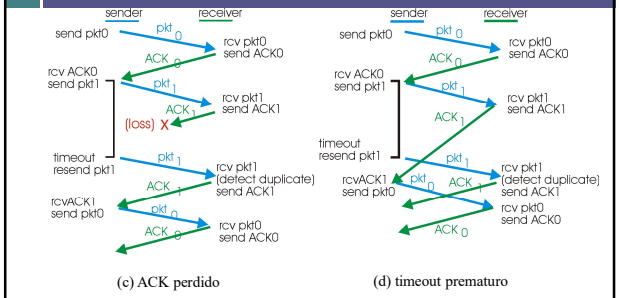
42

rdt3.0 em ação



43

rdt3.0 em ação



44

Desempenho do rdt3.0

- rdt3.0 funciona, mas o desempenho é sofrível
- exemplo: enlace de 1 Gbps, 15 ms de atraso de propagação, pacotes de 1KB:

$$\text{transmissão} = \frac{\text{8kb/pct}}{\text{10**9 b/seg}} = 8\ \mu s$$

$$\text{Utilização} = U = \frac{\text{fração do tempo transmissor ocupado}}{} = \frac{8\ \mu s}{30{,}016\ ms} = 0.00015$$

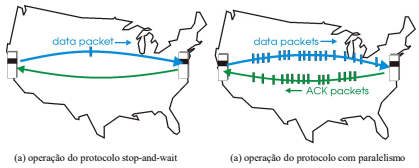
  - Um pacote de 1KB cada 30 ms -> 33kB/seg de vazão sobre um canal de 1 Gbps
  - o protocolo de rede limita o uso dos recursos físicos!

45

Protocolos com Paralelismo (pipelining)

Paralelismo: transmissor envia vários pacotes ao mesmo tempo, todos esperando para serem reconhecidos

- faixa de números de seqüência deve ser aumentada
- armazenamento no transmissor e/ou no receptor



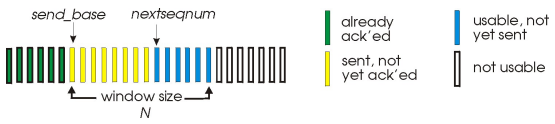
- Duas formas genéricas de protocolos com paralelismo: *go-Back-N, retransmissão seletiva*

46

Go-Back-N

Transmissor:

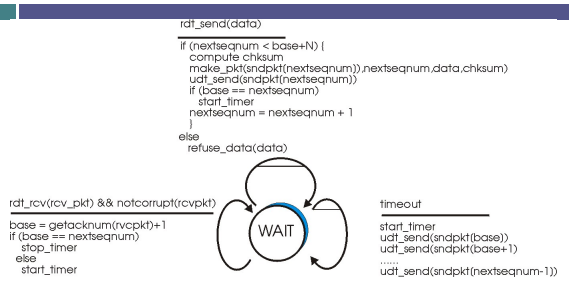
- Número de seqüência com k bits no cabeçalho do pacote
- "janela" de até N, pacotes não reconhecidos, consecutivos, são permitidos



- ACK(n): reconhece todos os pacotes até o número de seqüência N (incluindo este limite). "ACK cumulativo"
  - pode receber ACKS duplicados (veja receptor)
- temporizador para cada pacote enviado e não confirmado
- *timeout(n):* retransmite pacote n e todos os pacotes com número de seqüência maior que estejam dentro da janela

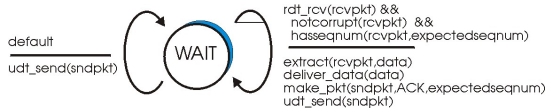
47

GBN: FSM estendida para o transmissor



48

GBN: FSM estendida para o receptor

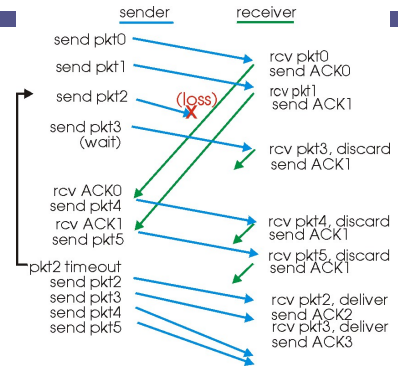


receptor simples:

- somente ACK: sempre envia ACK para pacotes corretamente recebidos com o mais alto número de seqüência *em ordem*
  - pode gerar ACKs duplicados
  - precisa lembrar apenas do número de seqüência esperado (`expectedseqnum`)
- pacotes fora de ordem:
  - descarte (não armazena) -> não há buffer de recepção!
  - reconhece pacote com o mais alto número de seqüência em ordem

49

GBN em ação



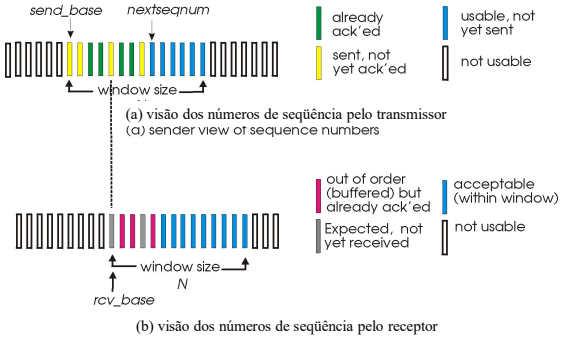
50

Retransmissão Seletiva

- receptor reconhece *individualmente* todos os pacotes recebidos corretamente
  - armazena pacotes, quando necessário, para eventual entrega em ordem para a camada superior
- transmissor somente reenvia os pacotes para os quais um ACK não foi recebido
  - transmissor temporiza cada pacote não reconhecido
- janela de transmissão
  - N números de seqüência consecutivos
  - novamente limita a quantidade de pacotes enviados, mas não reconhecidos

51

Retransmissão seletiva: janelas do transmissor e do receptor



52

Retransmissão seletiva

transmissor

dados da camada superior :

- se o próximo número de seqüência disponível está na janela, envia o pacote

timeout(n):

- reenvia pacote n, restart timer

ACK(n) em [sendbase,sendbase+N]:

- marca pacote n como recebido
- se n é o menor pacote não reconhecido, avança a base da janela para o próximo número de seqüência não reconhecido

receptor

pacote n em [rcvbase, rcvbase+N-1]

- envia ACK(n)
- fora de ordem: armazena
- em ordem: entrega (também entrega pacotes armazenados em ordem), avança janela para o próximo pacote ainda não recebido

pkt n em [rcvbase-N,rcvbase-1]

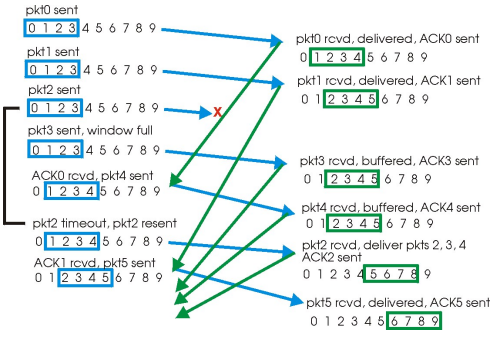
- ACK(n)

caso contrário:

- ignora

53

Retransmissão seletiva em ação



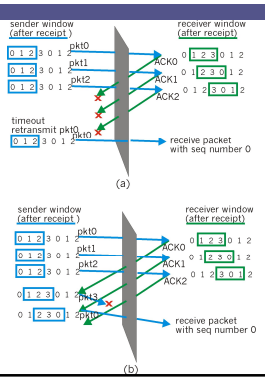
54

Retransmissão seletiva: dilema

Exemplo:

- seqüências: 0, 1, 2, 3
- tamanho da janela=3
- receptor não vê diferença nos dois cenários!
- incorretamente passa dados duplicados como novos (figura a)

Q: qual a relação entre o espaço de numeração seqüencial e o tamanho da janela?



55

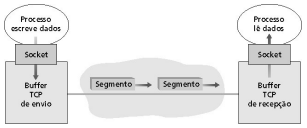
Camada de transporte

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

56

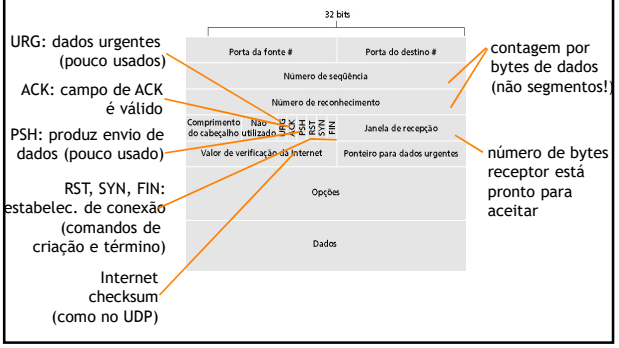
O Protocolo TCP

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



57

Estrutura do Segmento TCP



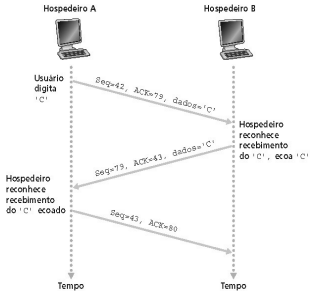
58

Número de Sequência e ACKs no TCP

Números de seqüência:
- Número do primeiro byte nos segmentos de dados

ACKs:
- Número do próximo byte esperado do outro lado
- ACK cumulativo

P.: Como o receptor trata segmentos fora de ordem?
- A especificação do TCP não define, fica a critério do implementador



59

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

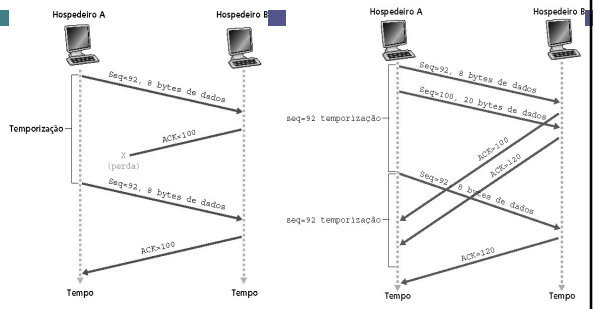
60

TCP: transferência de dados confiável

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

61

TCP: cenários de retransmissão

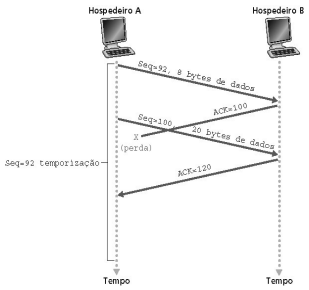


Cenário com perda do ACK

Temporização prematura, ACKs cumulativos

62

TCP: cenários de retransmissão



Cenário de ACK cumulativo

63

Geração de ACK [RFC 1122, RFC 2581]

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

64

Retransmissão rápida

- Com freqüência, o tempo de expiração é relativamente longo:
  - Longo atraso antes de reenviar um pacote perdido
- Detecta segmentos perdidos por meio de ACKs duplicados
  - Transmissor freqüentemente envia muitos segmentos
  - Se o segmento é perdido, haverá muitos ACKs duplicados
- Se o transmissor recebe 3 ACKs para o mesmo dado, ele supõe que o segmento após o dado confirmado foi perdido:
  - Retransmissão rápida: reenvia o segmento antes de o temporizador expirar

65

TCP Round Trip Time e temporização

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

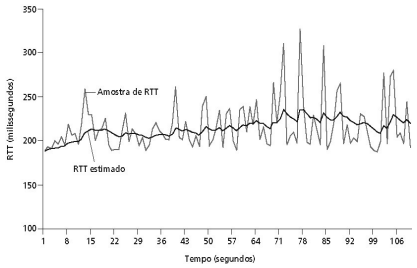
- Maior que o RTT
  - Nota: RTT varia
- Muito curto: temporização prematura
  - Retransmissões desnecessárias
- Muito longo: a reação à perda de segmento fica lenta

P.: Como estimar o RTT?

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

66

Exemplos de estimativa do RTT



67

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

68

TCP: controle de fluxo

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

- Processos de aplicação podem ser lentos para ler o buffer

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

- Serviço de **speed-matching**: encontra a taxa de envio adequada à taxa de vazão da aplicação receptora

69

Controle de fluxo TCP: como funciona

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

70

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

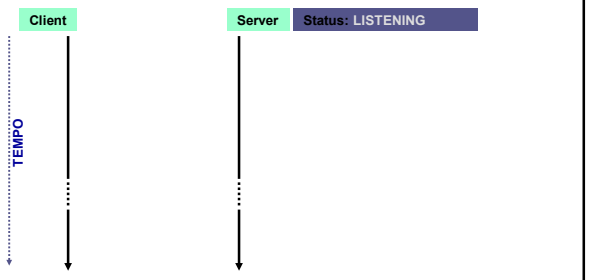
71

O Protocolo TCP

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

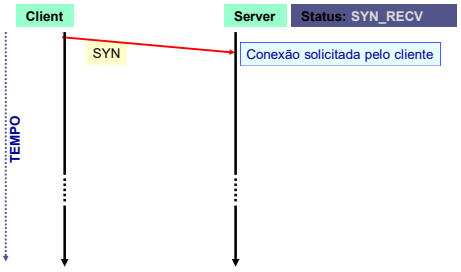
72

TCP – Como Funciona



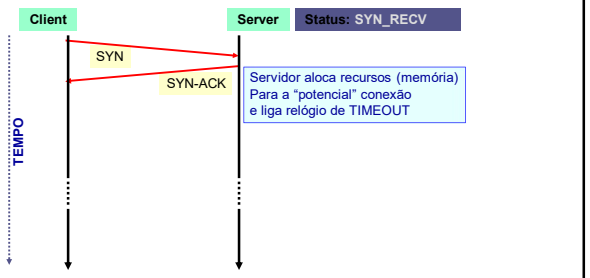
73

TCP – Como Funciona



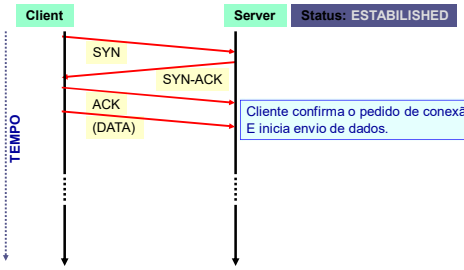
74

TCP – Como Funciona



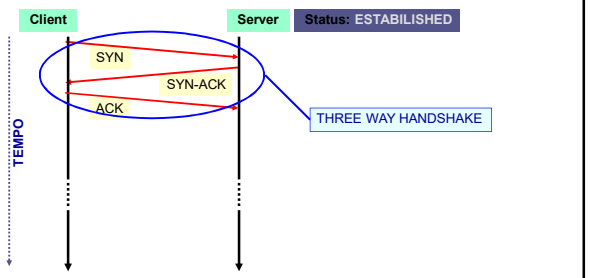
75

TCP – Como Funciona



76

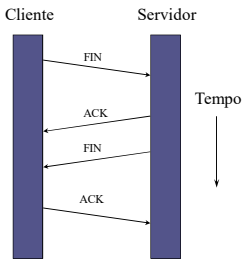
TCP – Como Funciona



77

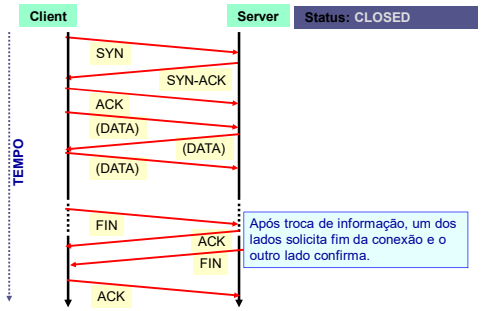
O Protocolo TCP

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



78

TCP – Como Funciona



79

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

80

Janela de Congestionamento

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

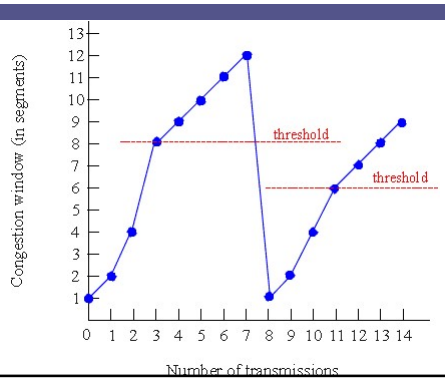
81

Controle de Congestionamento

- O controle é feito através de duas variáveis adicionadas em cada lado da conexão:
  - ***Janela de Congestionamento***
    - Janela do TCP explicada anteriormente
  - ***Limiar***
    - Serve para controlar o crescimento da janela de congestionamento

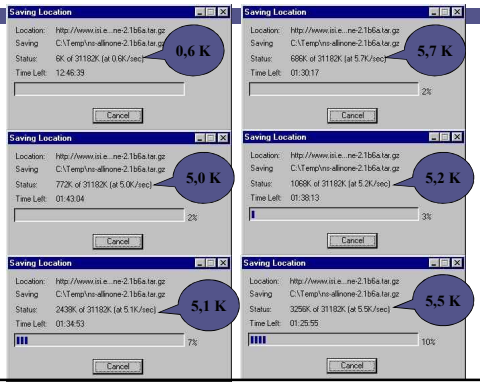
82

Graficamente ...



83

ontrole de Congestionamento TCP



84

Janela do Receptor

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

85

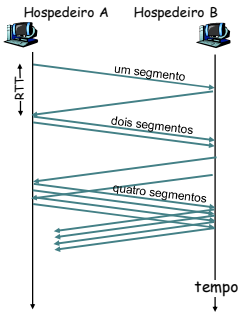
Evolução de uma Conexão TCP

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

86

Evolução de uma Conexão TCP

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



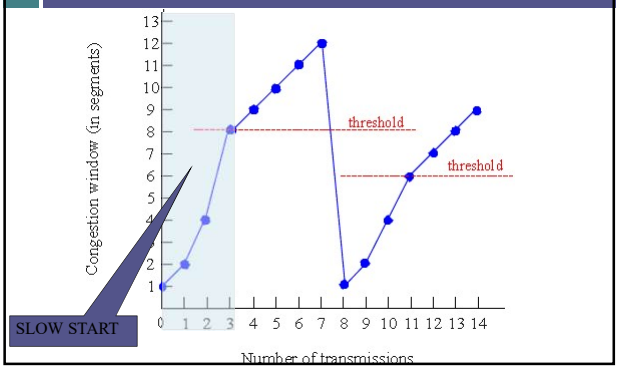
87

Duas Fases dessa Evolução

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

88

Graficamente ...



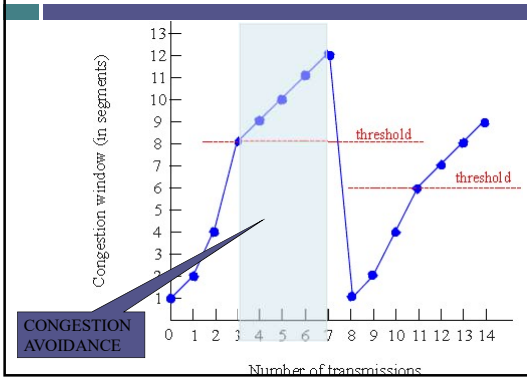
89

Duas Fases dessa Evolução

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

90

Graficamente ...



91

E quando ocorrer um problema?

92

Evolução de uma Conexão TCP

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

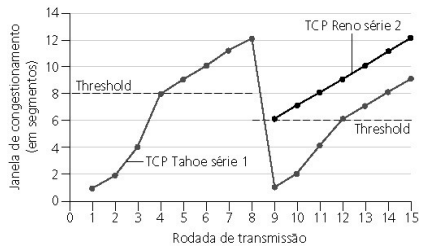
93

Resumo

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

94

Graficamente ...



95