Redes de Computadores II
Aula 15
O Protocolo TCP

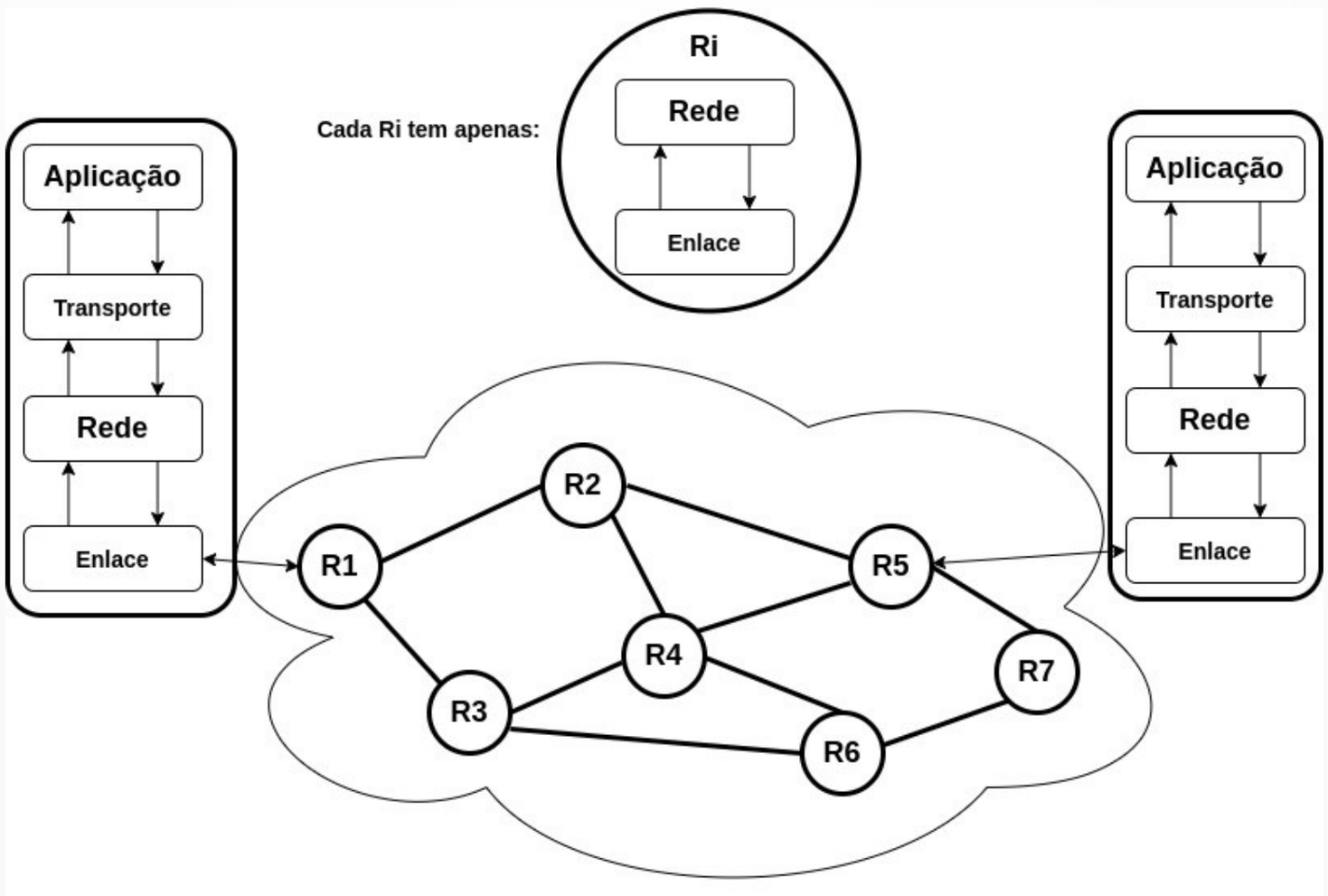
Prof. Elias P. Duarte Jr.
Universidade Federal do Paraná (UFPR)
Departamento de Informática
www.inf.ufpr.br/elias/redes

Sumário da Aula de Hoje

- O Protocolo TCP
- Abertura de conexão
- O header do segmento TCP
- Controle de Fluxo
- Iniciando o Controle de Congestionamento

A Camada de Transporte

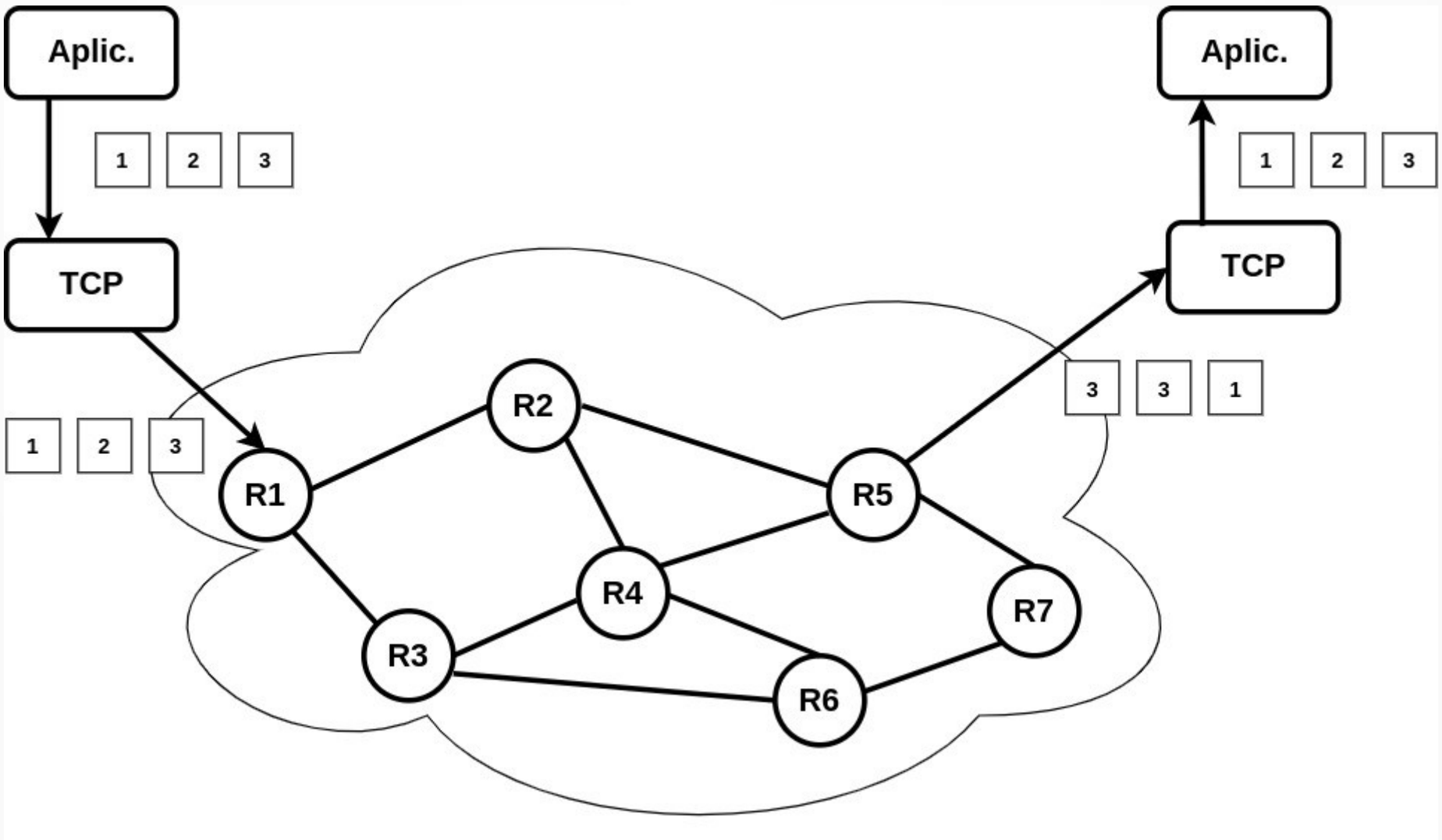
- Comunicação de processos



Transporte na Internet

- Dois protocolos: UDP & TCP
- UDP: não confiável e não orientado à conexão
  - sobra trabalho para o programador de aplicação
- TCP: confiável e orientado à conexão
  - resolve os problemas que o IP “deixa passar”

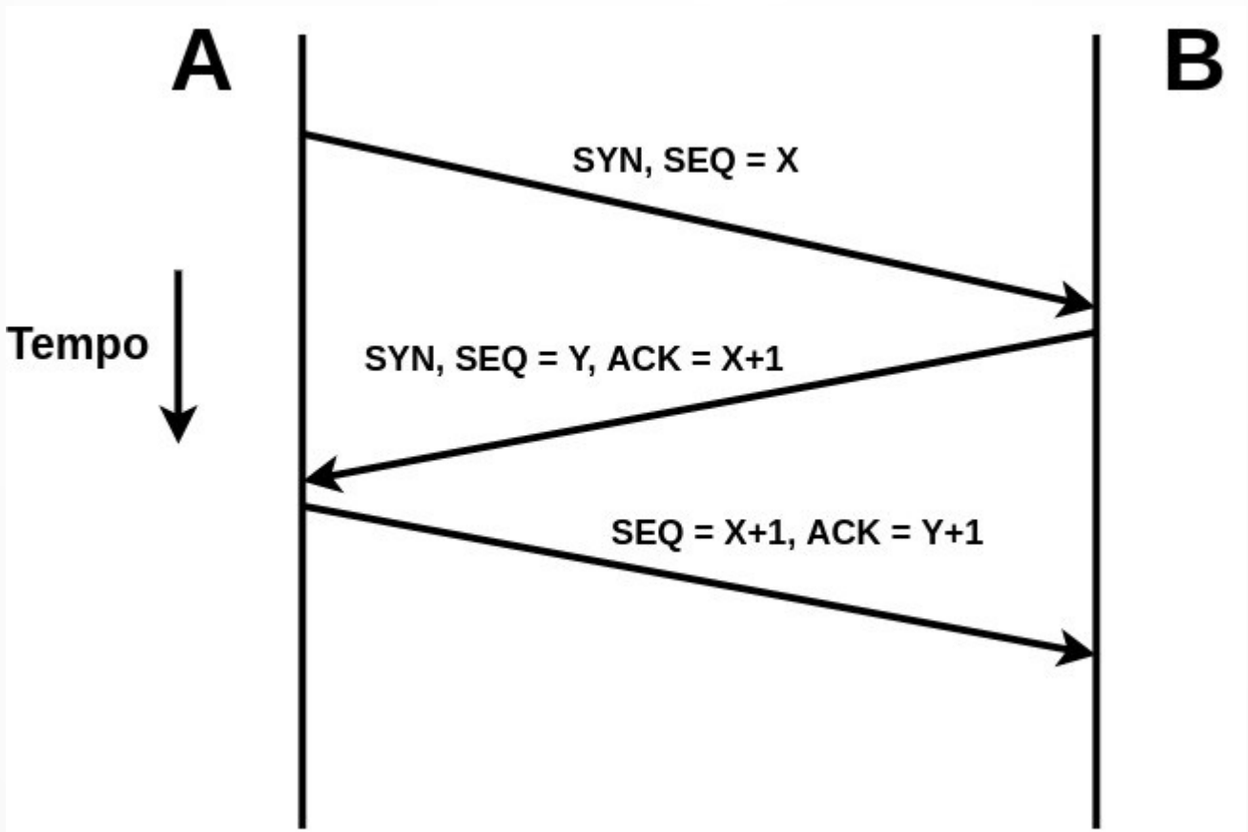
TCP: Confiável e Orientado à Conexão



Começando o TCP

- O TCP é um protocolo orientado à conexão
- Antes de comunicar: precisa estabelecer conexão entre os dois processos
- A abertura de conexão do TCP se chama “Three-Way Handshake”
  - Tem sido traduzida como “aperto de mão em 3 vias”

Abertura de Conexão TCP



Three-Way Handshake

- SYN: um flag do TCP usado na abertura da conexão apenas
  - indica que os dois processos vão “sincronizar”
- SEQ: o número de sequência do primeiro byte do segmento
  - o primeiro byte da conexão tem número de sequência aleatório
  - evita duas conexões contíguas iniciando em 1, 2,
  - pacotes perdidos na rede podem confundir

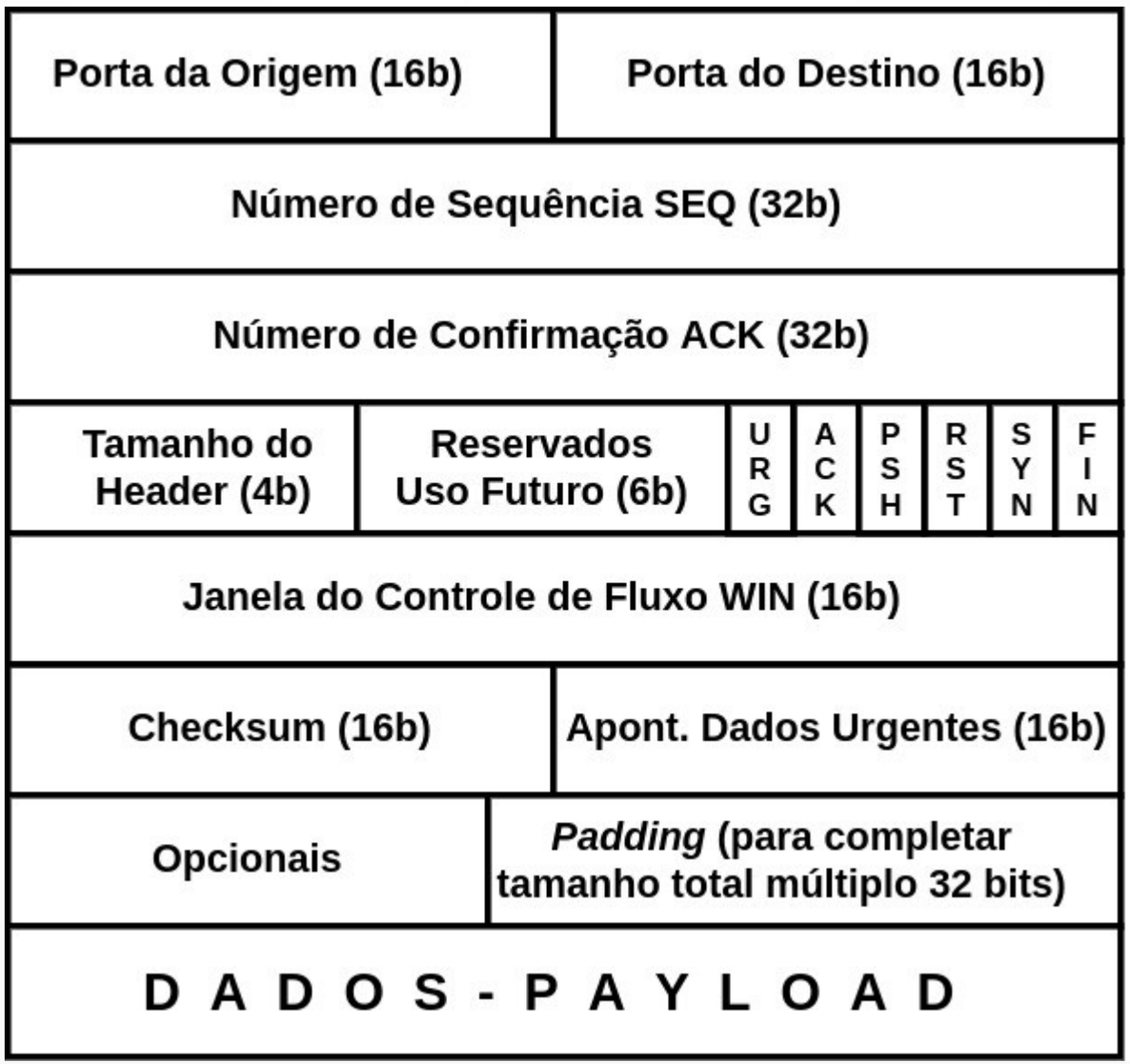
Abertura de Conexão TCP

- ACK: confirmação de recebimento do TCP
- Feita da seguinte maneira
- Para dizer que recebeu até o byte com SEQ = X, responde ACK = X+1
  - Por exemplo: recebi todos os bytes até o 8 → ACK = 9
  - Por exemplo: recebi todos os bytes até o 51 → ACK = 52
- Confirmações sempre contínuas (há não ser que opcionais sejam usados)

TCP SYN Flooding

- Um dos ataques mais famosos da história da Internet
- Baseado justamente na abertura de conexão TCP
- Faz um número massivo de solicitações, nunca envia o passo 3 para nenhuma
- Se a gerência de memória é pobre: overflow!
- Nos permite compreender a natureza de ataques aos protocolos

O Header do Segmento TCP



Portas de Origem & Destino

- Usadas na identificação dos processos que se comunicam
- Lembrar que o conjunto de portas TCP é disjunto do conjunto de portas UDP
- De 0..1023: *Well-Known Ports*, as portas usadas por protocolos padronizados

Número de Sequência: SEQ

- Número de ordem do 1º byte de dados do *payload*
- Identifica os dados sendo transmitidos
  - assim o TCP consegue fazer confirmações
  - e retransmissões em caso de perda
- Lembre-se que o SEQ do 1º byte do 1º segmento da conexão não é igual a 1 → é aleatório
  - por exemplo se o 1º é 51, o 2º é 52, o 3º é 53, ...
- Conexão TCP entre dois processos A e B: dados de A para B tem SEQs distintos dos dados de B para A

Número de Confirmação: ACK

- Confirmação de recebimento: Acknowledgment
- O TCP usa uma técnica chamada *piggybacking*
- Há protocolos que têm um pacote especial só para o ACK
- No TCP (*piggybacking*) o mesmo pacote de leva dados de A para B...
- ... confirma o recebimento de dados de B por A
- ACK do TCP indica o SEQ do próximo byte esperado
  - que é o SEQ do último byte recebido + 1

Tamanho do Header

- Mesma história do IP: se preciso dizer quantos bytes tem o header é porque...

Tamanho do Header

- Mesma história do IP: se preciso dizer quantos bytes tem o header é porque o tamanho do header é variável
- Também devido aos “Opcionais” - neste caso TCP
- Servidores comerciais ou de terceiros: não vão aceitar seus opcionais!
- Só usável em uma organização ou 1 indivíduo
- Quase sempre: 20 bytes, o mesmo do IP
- Também em palavras de 4 bytes (32 bits)

6 Bits Reservados para Uso Futuro

- Ótima decisão de projeto
- Dá margem à evolução do protocolo
- TCP é considerado um dos maiores sucessos da história da computação
- 40 anos e resistiu a diversas mudanças profundas de tecnologia de redes

Os 6 Flags do TCP

- Todos importantes e usados em situações específicas:
  - **URG**
  - **ACK**
  - **PSH**
  - **RST**
  - **SYN**
  - **FIN**

O Flag URG: URGent

- Indica que há dados URGentes no segmento
- O que são dados urgentes?

O Flag URG: URGent

- Indica que há dados URGentes no segmento
- O que são dados urgentes?
- Depende da aplicação: por exemplo <ESCAPE> ou <CONTROL-C> podem representar saída imediata
- O TCP: não define os dados urgentes
- O TCP permite que aplicações comuniquem dados urgentes
- Sintaxe e semântica definidas pelas aplicações

Apontador para Dados Urgentes

- Indica onde no campo *payload* estão os dados urgentes
  - Na verdade indica o último byte dos dos dados urgentes
  - O destinatário então prioriza o payload até este ponto com urgência
- O flag URG, se ligado, indica que o campo dados urgentes efetivamente mostra dados urgentes
  - Caso contrário: este campo deve ser ignorado

O Flag ACK: ACKnowledgment

- Se estiver ligado: o segmento leva uma confirmação de recebimento
- Caso contrário, o campo “Número de Confirmação” deve ser ignorado
- Lembrando: leva o número de sequência do próximo byte esperado
  - Por exemplo: ACK 51, chegou até o byte 50;
  - Por exemplo: ACK 25647, chegou até o byte 25646
- Está confirmando o recebimento de *todos* os bytes

O Flag PSH: PuSH

- PUSH → “empurre”
- Os bytes deste segmento devem ser entregues imediatamente para a aplicação no destino
- O TCP faz diversos controles de uso da rede → vamos estudar a frente
- Antes do pacote ser efetivamente transmitido ou efetivamente entregue (*delivered*) para a aplicação no destino: nem sempre pode esperar – ex. mouse remoto
- Especialmente no caso de uma série de pacotes pequenos
- O PSH determina o *delivery* imediato!

O Flag RST: ReSeT

- Usado pelo TCP em resposta a segmentos “malucos”
- Para os quais o TCP não tem outra ação possível
- Exemplo: chega um pacote em uma conexão inexistente
- Também: problemas de parâmetros
- Significado: “resete esta sua conexão, pois me mandou um segmento que não faz sentido”

O Flag SYN: SYNchronize

- Ligado no estabelecimento da conexão, como vimos

O Flag FIN: FINalize

- Ligado no encerramento da conexão
- Vamos ver mais para a frente

Checksum do TCP

- O TCP usa o mesmo algoritmo do IP, ICMP, UDP
- Código de detecção de erros:
  - soma grupos de 16 bits em complemento de 1
  - tira o complemento do resultado
  - destinatário inclui o checksum na soma: zero OK!
- Da mesma forma que o UDP, o TCP inclui alguns campos do header IP no cálculo do checksum

Controle de Fluxo do TCP

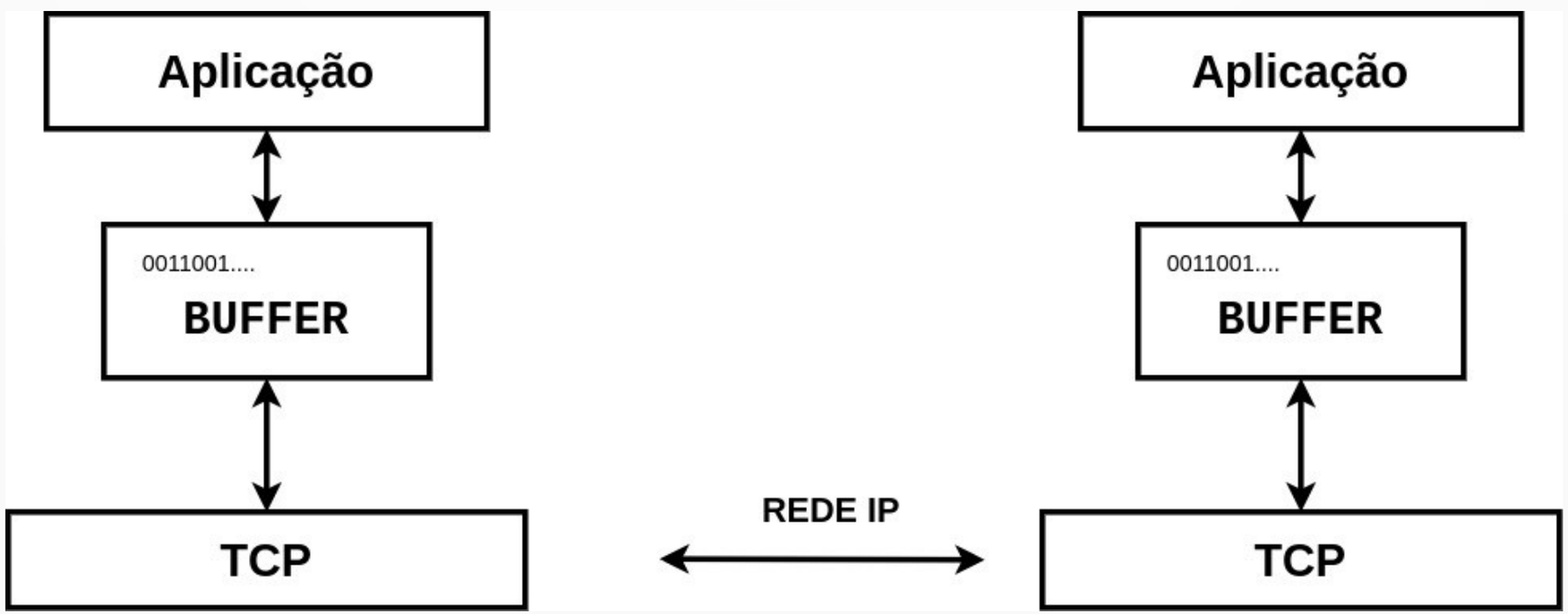
- Na aula passada [re-]vimos o conceito de controle de fluxo (genérico)
- Hoje vamos ver o controle de fluxo do TCP propriamente dito
- O controle de fluxo serve para...

Controle de Fluxo do TCP

- Na aula passada [re-]vimos o conceito de controle de fluxo (genérico)
- Hoje vamos ver o controle de fluxo do TCP propriamente dito
- O controle de fluxo serve para definir a melhor taxa em que origem e destino podem comunicar
  - em particular: a origem não deve mandar mais dados do que o destino consegue receber

Controle de Fluxo do TCP

- Lembrar: cada byte transmitido em cada direção da conexão TCP tem um número de sequência
- No caso do TCP: destinatário tem um **buffer** de recepção de dados



Janela do Controle de Fluxo TCP

- Campo WIN (WINdow – janela) do header do segmento TCP
- O TCP informa sempre ao outro processo quantos bytes livres têm na sua janela
- Ou seja a janela de controle de fluxo do TCP é o número de bytes livres do buffer do receptor

Controle de Fluxo do TCP: Exemplo

- Considere que o buffer do destino tem tamanho total 80 bytes, SEQ do 1º byte = 1
- Recebe 30 bytes: manda ACK = 31, WIN = 50
- Recebe 20 bytes: manda ACK = 51, WIN = 30
- Recebe 30 bytes: manda ACK = 81, WIN = 0

Controle de Fluxo do TCP: Exemplo

- Considere que o buffer do destino tem tamanho total 80 bytes, SEQ do 1º byte = 1
- Recebe 30 bytes: manda ACK = 31, WIN = 50
- Recebe 20 bytes: manda ACK = 51, WIN = 30
- Recebe 30 bytes: manda ACK = 81, WIN = 0
- Agora o emissor não pode mandar mais dados, tem que esperar WIN > 0
- Só quando a aplicação do destino recebe os dados do TCP do destino WIN vai ser novamente > 0

Passa Um Tempo e Delivery!

- Pronto a aplicação recebeu os dados e o buffer tem novamente 80 bytes livres
- Envia: ACK = 81, WIN = 80
- Mas considere que a rede IP perde o pacote com este segmento
- Como você descreve a situação que se formou?

Passa Um Tempo e Delivery!

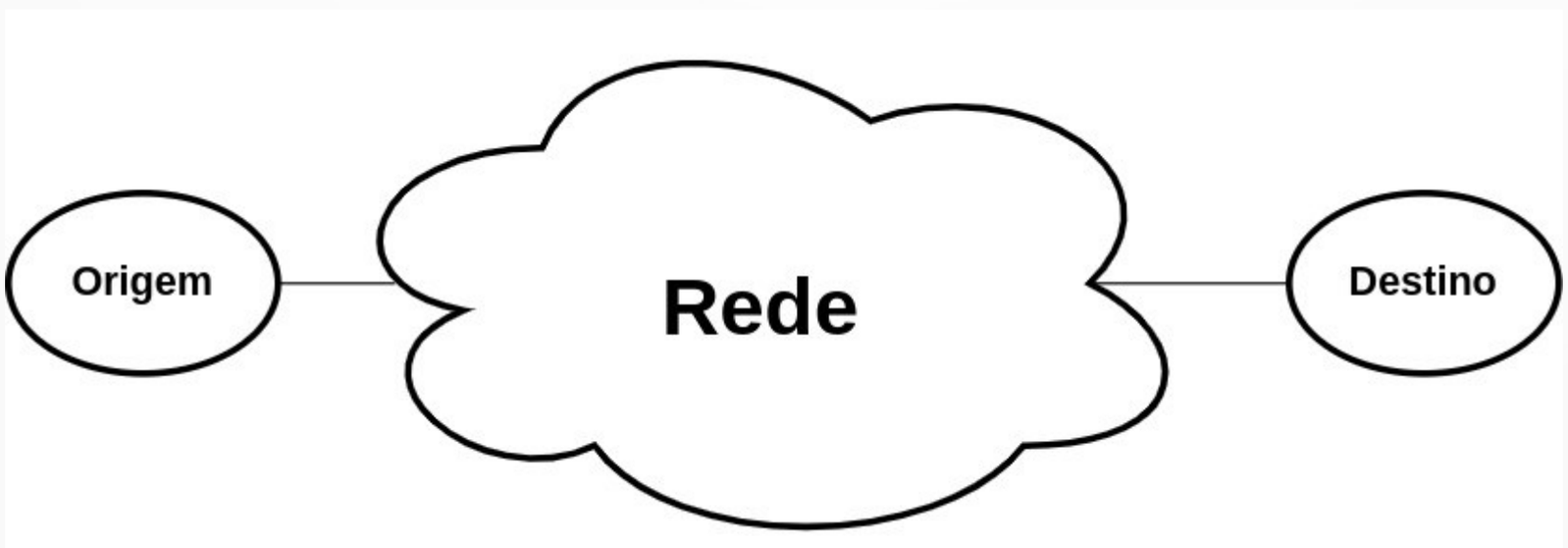
- Pronto a aplicação recebeu os dados e o buffer tem novamente 80 bytes livres
- Envia: ACK = 81, WIN = 80
- Mas considere que a rede IP perde o pacote com este segmento
- Como você descreve a situação que se formou?
- Deadlock!
  - O emissor aguarda segmento com WIN > 0
  - O receptor já enviou segmento com WIN > 0 e aguarda

Solução do Deadlock

- O TCP tem diversos *timers* – temporizadores
- O Persistence Timer é usado neste caso
  - após enviar um segmento com WIN > 0
  - o Persistence Timer é ligado
  - se não chegar mais dados até o tempo limite
  - comunica novamente com o emissor
- Outro timer importante do TCP: Keep Alive Timer
  - Usado quando a conexão fica em silêncio por muito tempo

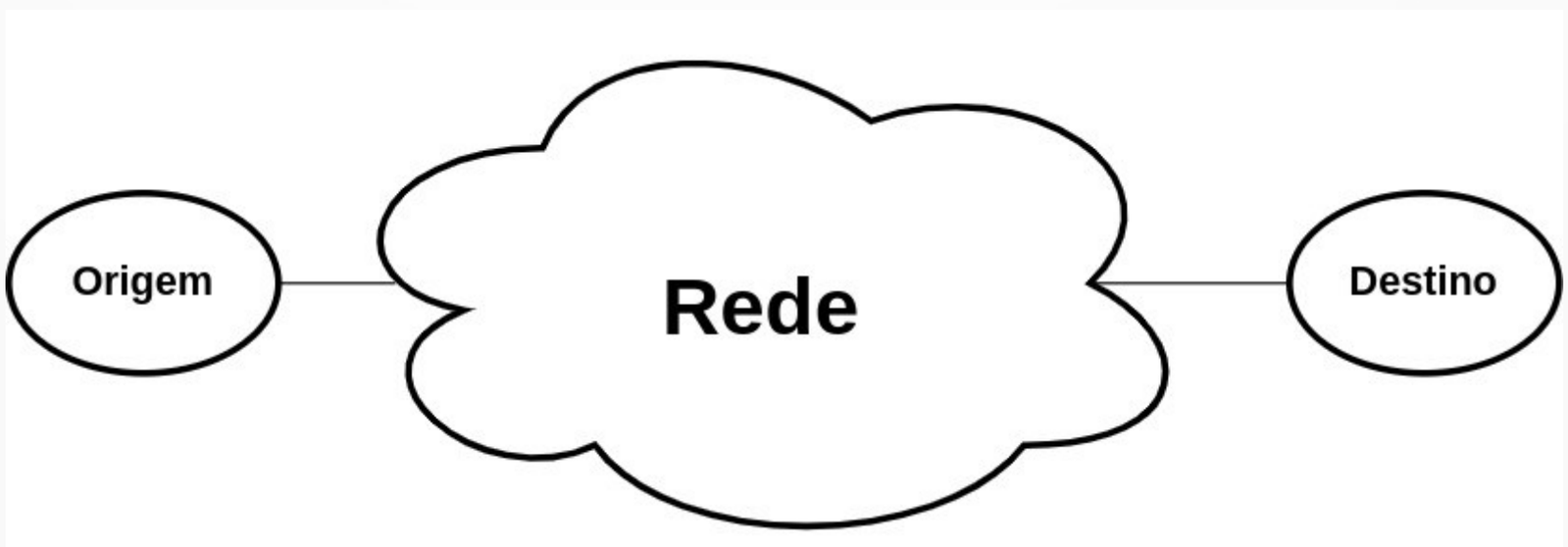
Controle de Congestionamento

- O controle de fluxo permite à origem estimar a capacidade do destino:



Controle de Congestionamento

- O controle de fluxo permite à origem estimar a capacidade do destino:
- Entre a origem e destino: a rede!
- O controle de congestionamento permite à origem estimar a capacidade da rede



Controle de Congestionamento

- Uma janela de congestionamento é constantemente atualizada para refletir esta estimativa
- Janela = número de bytes
- Vamos chamar a janela do controle de congestionamento de JCONG

Controles de Fluxo & Congestionamento

- Antes de **toda** transmissão as duas janelas são comparadas
  - Janela do Controle de Congestionamento (JCONG)
  - Janela do Controle de Fluxo (WIN)
- Só pode transmitir o menor valor para o número de bytes

Controle de Congestionamento do TCP

- Não definido no padrão original
- Cada implementação do TCP calcula localmente
- Diversas estratégias foram propostas ao longo das décadas
- Uma versão do TCP que implementa uma estratégia leva o nome da cidade onde ocorreu a reunião do IETF que a padronizou
  - TCP Reno, TCP Tahoe, TCP Vegas, TCP Westwood, TCP Illinois, etc. etc. etc.

Conclusão

- Nesta aula continuamos o estudo do protocolo TCP
  - que já tinha sido começado na aula anterior
- Serviço confiável e orientado à conexão
  - Par perfeito com o IP
  - UDP só em casos específicos
- Estudamos o header do TCP: vários campos
- Controle de Fluxo do TCP
- Começamos o Controle de Congestionamento

Obrigado!

Lembrando: a página da disciplina é:
https://www.inf.ufpr.br/elias/redes