TCP

Faculdade de Engenharia da Universiadade do Porto

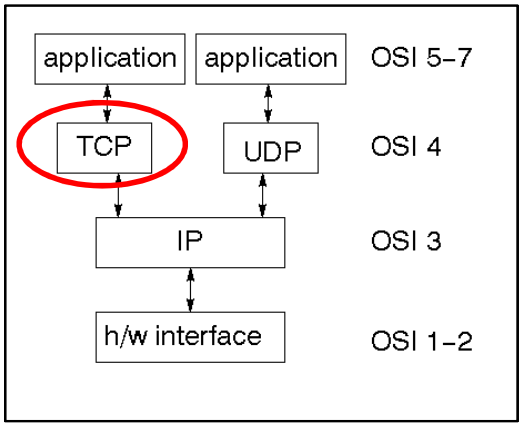
Manuel P. Ricardo

Bibliografia

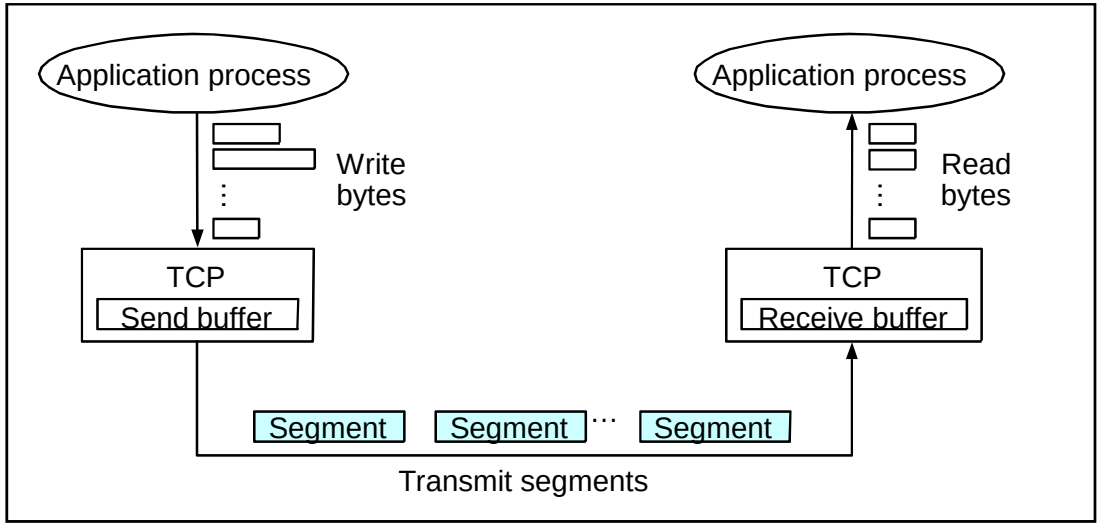
L. Peterson, B. Davie, “Computer Networks – A Systems Approach”, Morgan Kaufamnn, 2000 (Sec. 5.1, 5.2, 6.1, 6.2, 6.3 e 6.4)

TCP

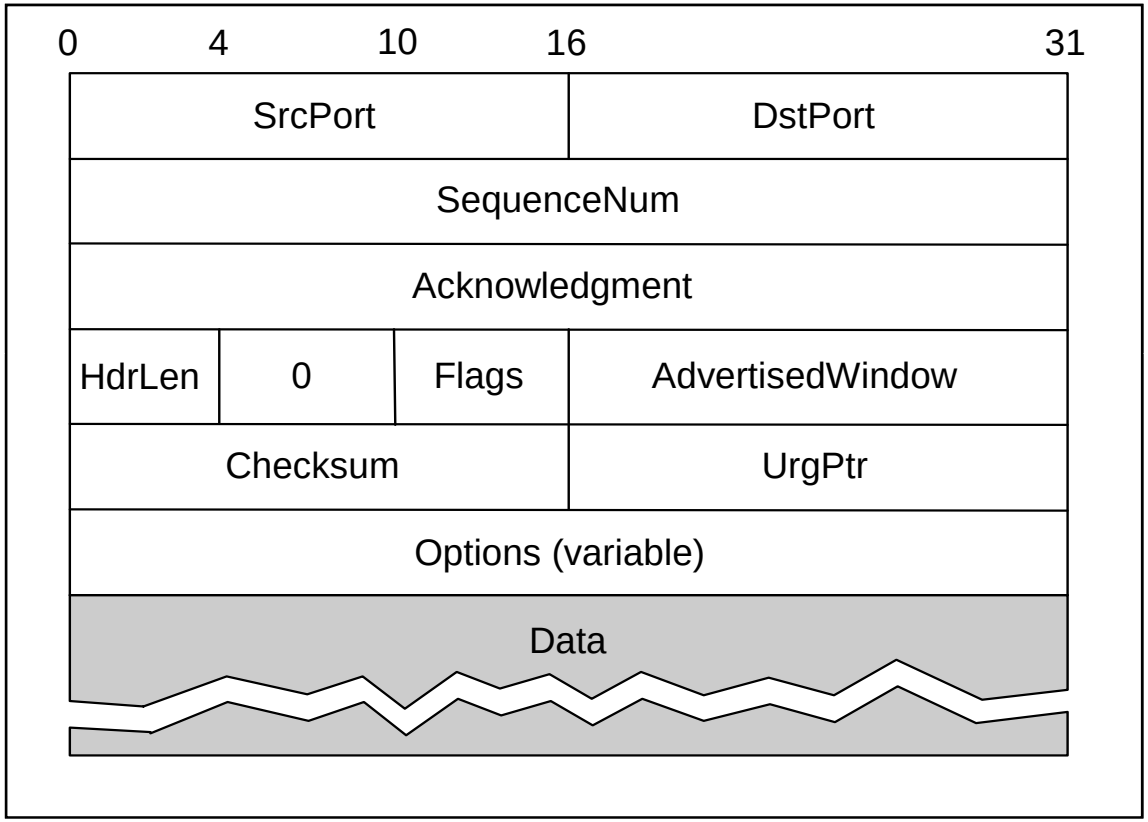
- Características
  - » Orientado às ligações
  - » Full-duplex
  - » Fluxo de bytes
    - – aplicação escreve bytes
    - – TCP envia segmentos
    - – aplicação lê bytes



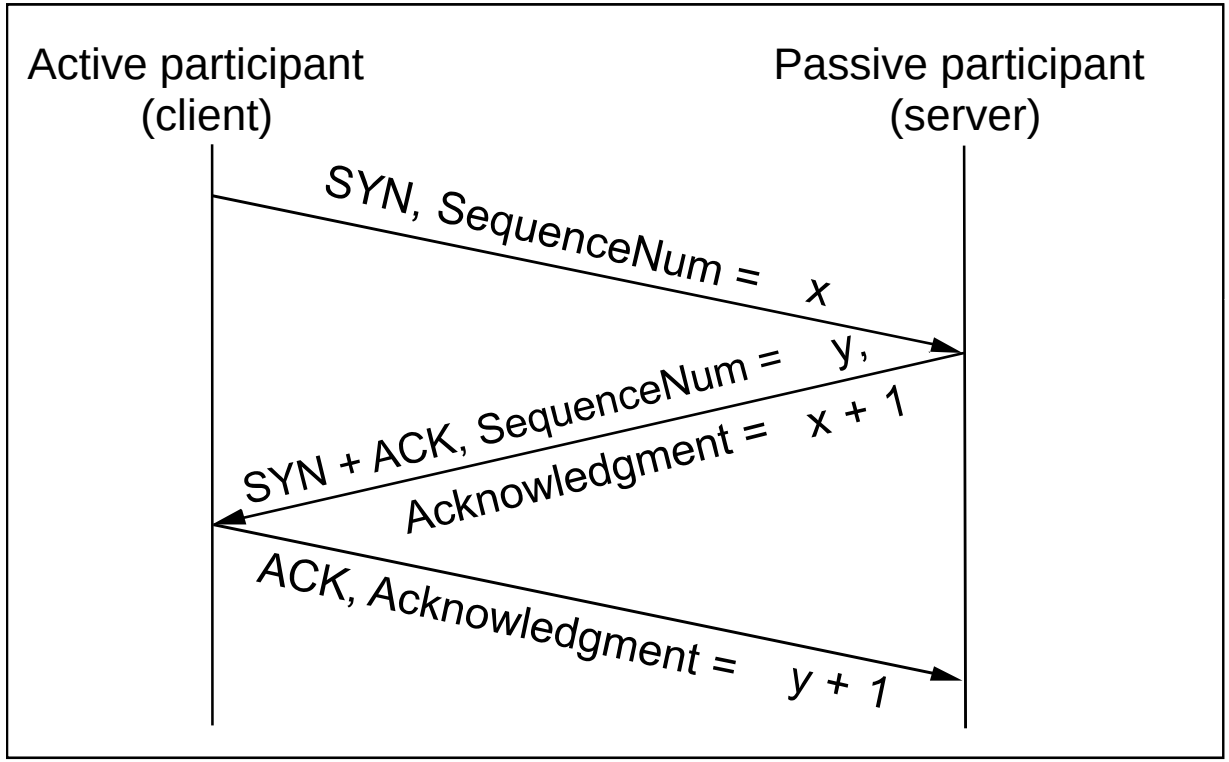
- Mecanismos de controlo
  - » Controlo de fluxo
    - Evita que emissor congestione receptor
  - » Controlo de congestionamento
    - Evita que emissor congestione a rede



Formato do Segmento



Estabelecimento da Ligação



Controlo de Fluxo

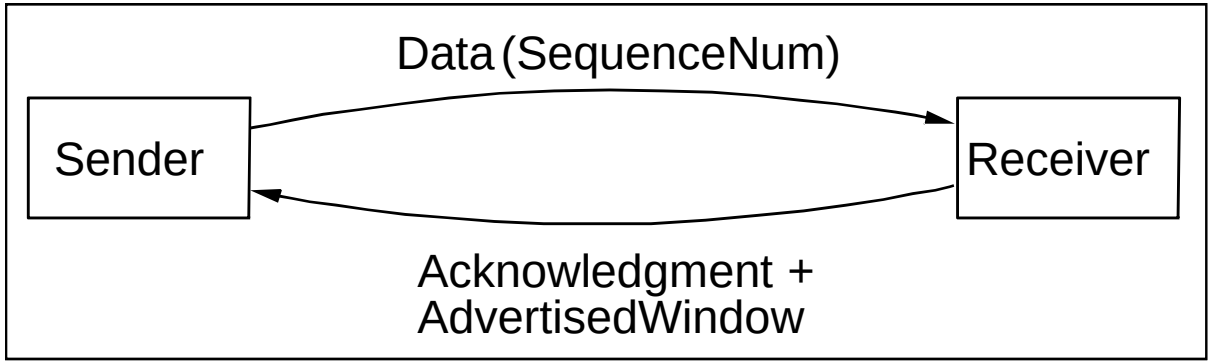
Janela Deslizante

- Emissor
  - » envia dados com indicação do número de ordem do 1º byte

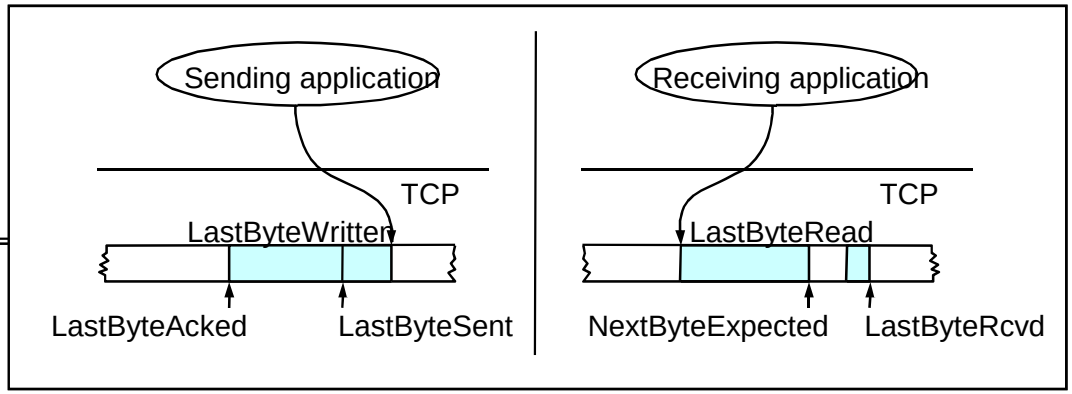
    Data (SequenceNum)
- Receptor
  - » confirma recepção indicando número ordem próximo byte a receber

    Acknowledgment
  - » indica espaço livre em memória

    AdvertisedWindow



Controlo de Fluxo



- Comprimento do buffer
  - no emissor → **`MaxSendBuffer`**
  - no receptor → **`MaxRcvBuffer`**
- No receptor

```
LastByteRcvd - LastByteRead < = MaxRcvBuffer
AdvertisedWindow = MaxRcvBuffer -(LastByteRcvd - LastByteRead)
```

- No Emissor

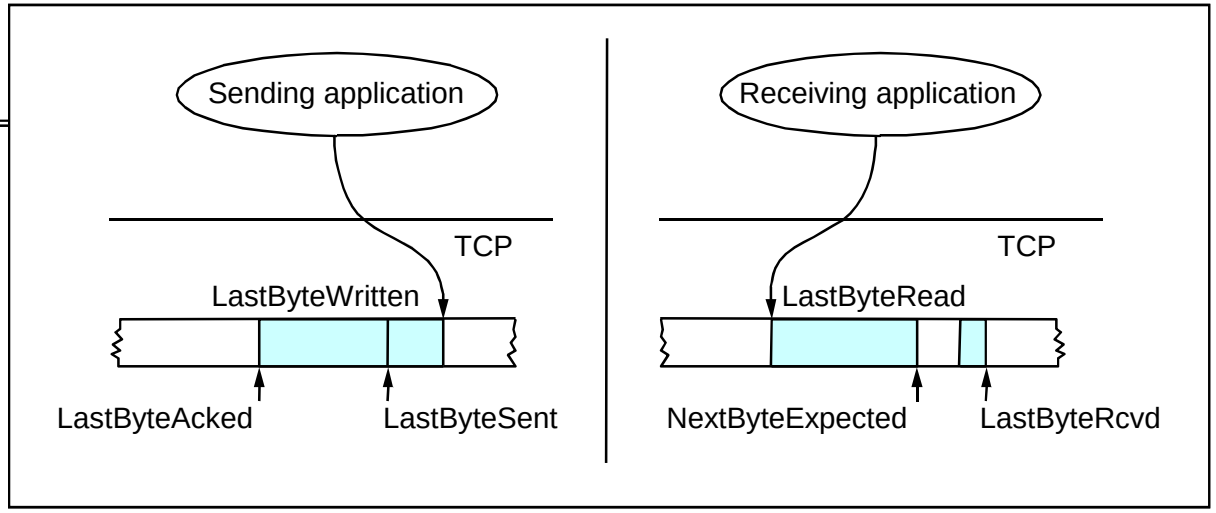
```
LastByteWritten - LastByteAcked < = MaxSendBuffer
LastByteSent - LastByteAcked < = AdvertisedWindow
EffectiveWindow = AdvertisedWindow - (LastByteSent - LastByteAcked)
```

  Processo bloqueia se quer enviar (write) y bytes e

```
(LastByteWritten - LastByteAcked) + y > MaxSenderBuffer
```

- **`ACK`** enviado como resposta à chegada de um segmento

Janela Deslizante



- No emissor
  - **`LastByteAcked < = LastByteSent`**
  - **`LastByteSent < = LastByteWritten`**
  - Buferiza bytes entre **`LastByteAcked`** e **`LastByteWritten`**

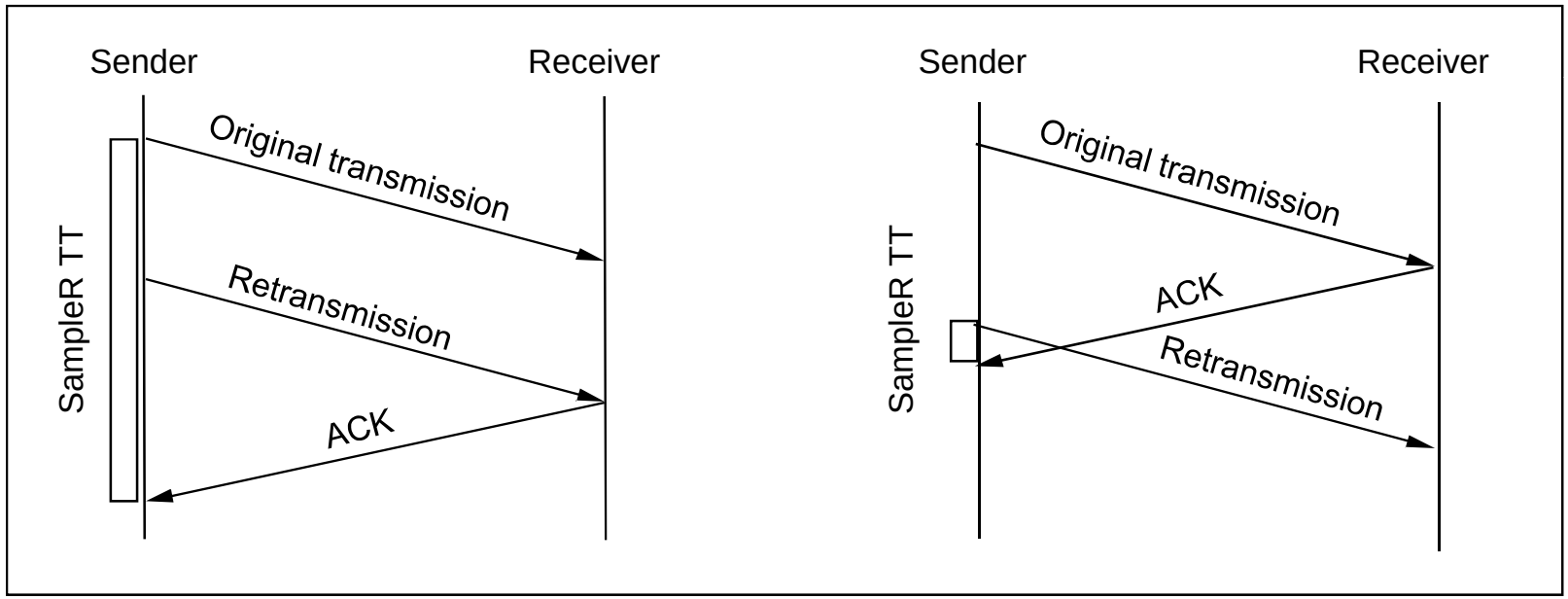
- No receptor
  - **`LastByteRead <  NextByteExpected`**
  - **`NextByteExpected < = LastByteRcvd +1`**
  - Buferiza bytes entre **`NextByteRead`** e **`LastByteRcvd`**

Retransmissão Adaptativa (Algoritmo Original)

- RTT → Round Trip Time (tempo de ida e volta)
- Medida de **`sampleRTT`** para cada par **`segmento/ACK`**
- Cálculo da média pesada do **`RTT`**
  - » **`RTT = a x RTT + (1-a) x SampleRTT`**
    `a em [0.8, 0.9]`
- **`TimeOut = 2 x RTT`**

Melhoria de Karn/Partridge

- Não mede **`sampleRTT`** em caso de retransmissão
- Duplica valor do timeout em cada retransmissão



Melhoria de Jacobson/ Karels

- Novo método de cálculo do **RTT**
  - » **Dif = sampleRTT - RTT**
  - » **Desv = Desv + p (|Dif| - Desv)**
    - – **p em [0,1]**
  - » **RTT = RTT + (p x Dif)**
- Considera variância no cálculo do timeout
  - » **TimeOut = m x RTT + f x Desv**
  - » **m = 1, f = 4**
- Mecanismo de timeout preciso
  - » importante para controlo de congestionamento

Controlo de Congestionamento

TCP - Controlo de Congestionamento

- Princípio de funcionamento
  - » Cada fonte determina a sua capacidade de geração de tráfego
  - » Baseada em critérios de
    - – justiça entre fluxos
    - – utilização máxima de recursos

      (capacidade de comutação, capacidade de transporte dos links)

- **ACKs** recebidos regulam a transmissão de pacotes

  ➔ são o *relógio* da fonte

Subida Aditiva/Descida Multiplicativa

- Mudanças na capacidade de canal ➔ ajuste na transmissão
- Nova variável de estado por ligação → **`CongestionWindow`**
  - » Limita a quantidade de dados em trânsito
    - – **`MaxWin = MIN(CongestionWindow, AdvertisedWindow)`**
    - – **`EffWin = MaxWin - (LastByteSent - LastByteAcked)`**
- Objectivo
  - » Se congestionamento da rede diminui ➔ aumenta **`CongestionWindow`**
  - » Se congestionamento da rede aumenta ➔ diminui **`CongestionWindow`**
- **`Débito (byte/s) ➔ CongestionWindow/RTT`**

Subida Aditiva/Descida Multiplicativa

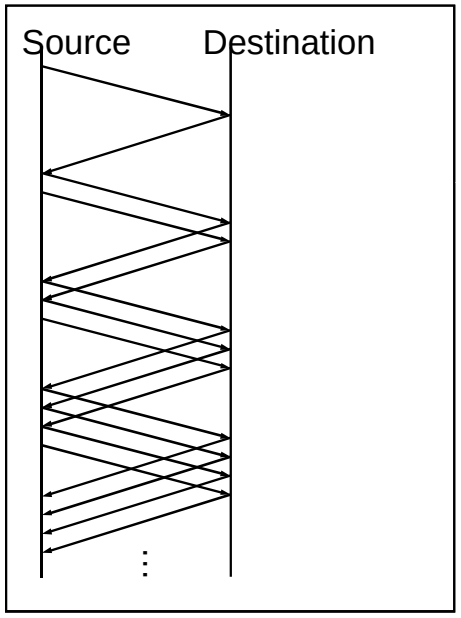
- Como sabe a fonte se/quando a rede está congestionada?

➔ Por ocorrência de timeout!

- » Redes fixas → pacotes raramente sofrem erros
- » Ocorrência de timeout ➔ perda de pacote
- » Perda de pacotes ➔ congestionamento
  - – Filas nos routers cheias

Subida Aditiva/Descida Multiplicativa

- Algoritmo
  - » Incrementa **`CongestionWindow`** de **`1`** pacote
    - – Por cada **RTT** (Round Trip Time) ➔ Subida linear
  - » Divide **`CongestionWindow`** por **2**
    - – Sempre que há perda de pacote ➔ Descida multiplicativa

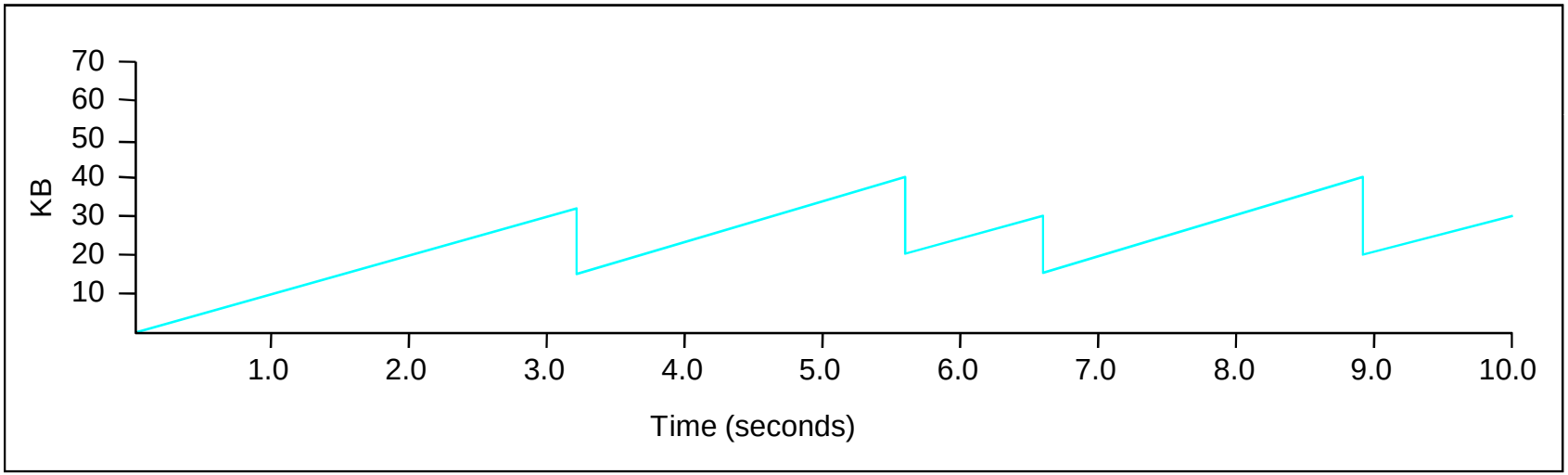


- Na prática,
  - » Incrementa ligeiramente por ACK recebido
  - » **`Increment= MSS * (MSS / CongestionWindow)`**
  - » **`CongestionWindow += Increment`**

  - » **`MSS → Maximum Segment Size`**

Subida Aditiva/Descida Multiplicativa

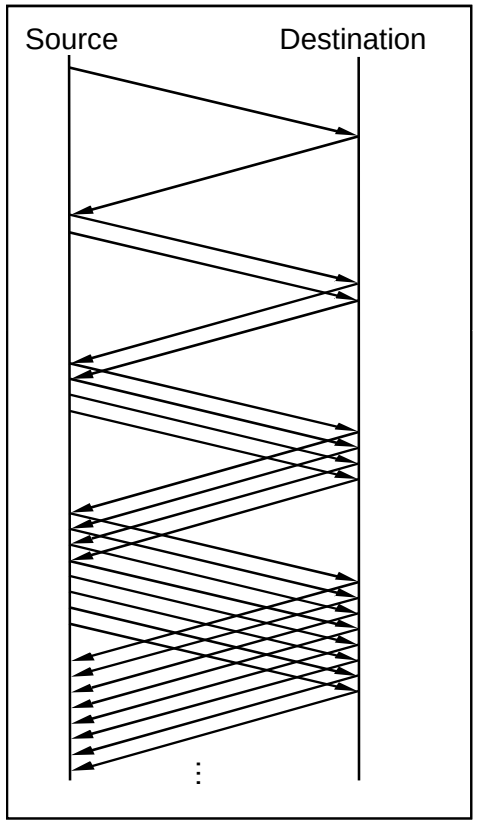
- Funcionamento **Dente de Serra**



Arranque Lento ☺

- Objectivo
  - » Determinar capacidade de transmissão disponível

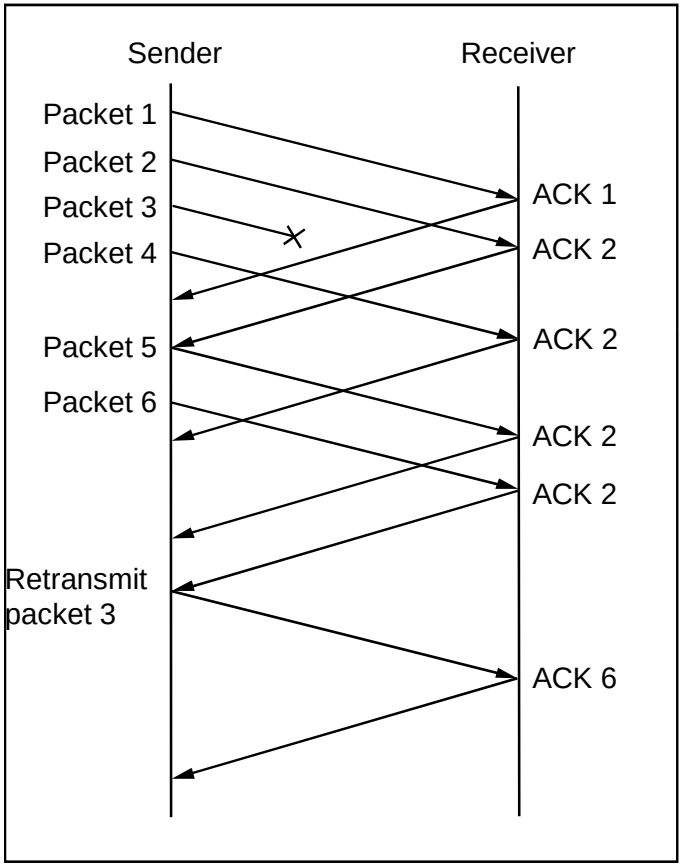
- Aproximação
  - » Começar com **`CongestionWindow = 1 pacote`**
  - » Duplicar **`CongestionWindow em cada RTT`**



Retransmissão e Recuperação Rápidas

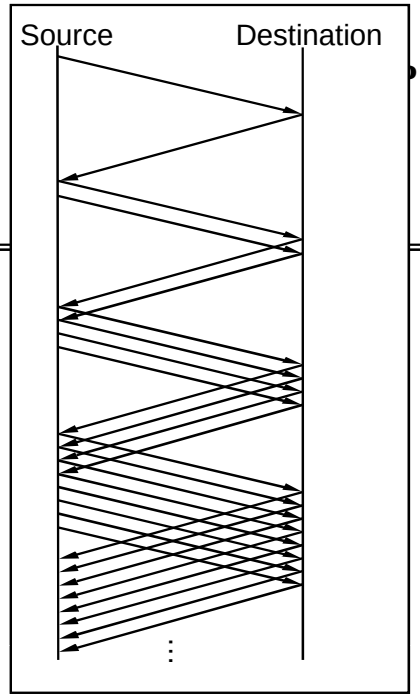
- Problema
  - » Se timeout TCP grande
    - → periodo de inactividade pode ser grande

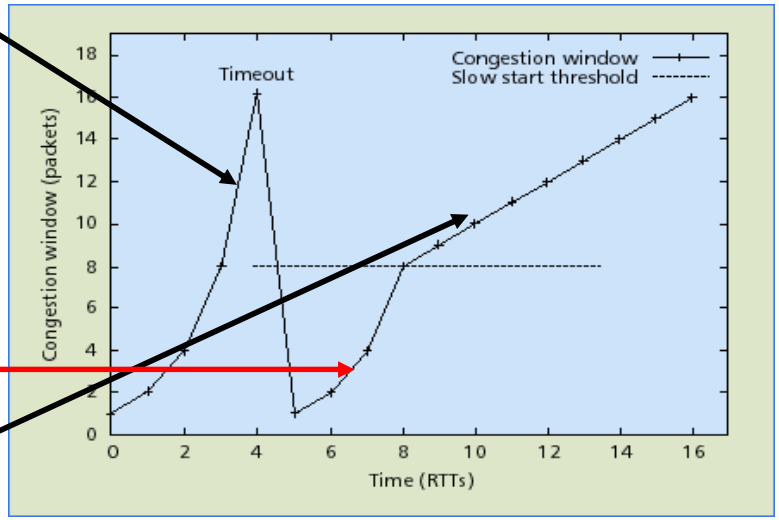
- Solução
  - » Retransmissão rápida
    - → utilização de ACKs repetidos (3)



TCP – ArranqueLento

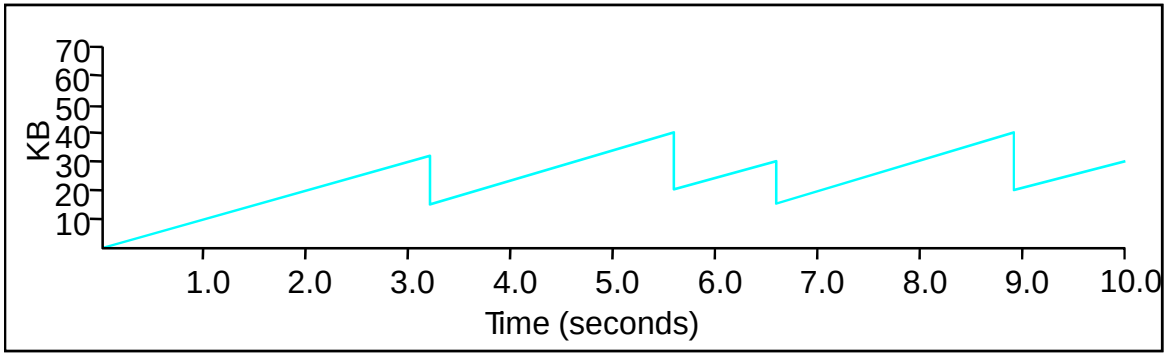
- TCP tem `congestionWindow`
  - » número de pacotes em trânsito, sem causar congestionamento
  - » Novos pacotes enviados se permitidos por
    - – `congestionWindow`, e
    - – `Advertisedwindow`, do receptor
- *ArranqueLento* ☺
  - » Emissor começa com **`congestionWindow=1sgm`**
  - » Duplica **`congestionWindow`** em cada **`RTT`**
- Quando detecta perda de pacote, por timeout
  - » **`threshold = ½ congestionWindow`**
  - » **`congestionWindow=1sgm`**
    (router esvazia filas)
  - » Pacote perdido retransmitido
  - » *ArranqueLento* enquanto
    `congWindow<threshold`
  - » Depois → fase de *PrevençãoDeCongestionamento*

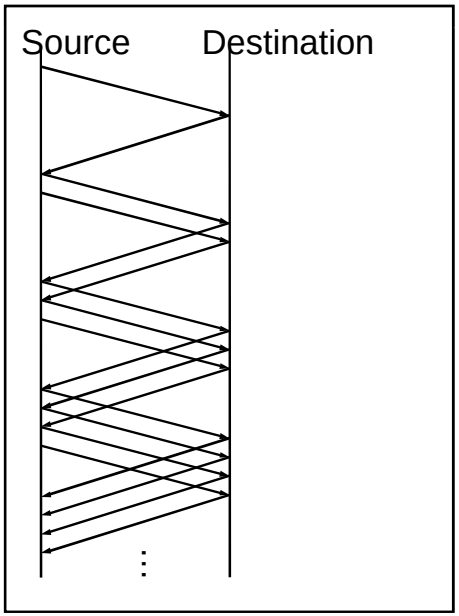




PrevençãoDeCongestionamento (Congestion Avoidance)

- ***PrevençãoDeCongestionamento*** (subida aditiva)
  - » Incrementa **`congestionWindow`** de 1 sgm, por cada RTT
- Detecção de perda de pacote, por recepção de 3 ACKs duplicados
  - » Deduz que pacote se perdeu,
    - – não por congestionamento severo, porque segm seguintes chegaram destino
  - » Retransmite pacote perdido
  - » **`congestionWindow=congestionWindow / 2`**
  - » Fase de *PrevençãoDeCongestionamento*





TCP – Controlo de Congestionamento

- Na realidade, um pouco mais complexo
- RFC 2001, “TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms”