Redes de Computadores

Protocolo Stop & Wait

Objectivos da lição

- A Internet não garante a entrega dos pacotes pois pode perdê-los ou entregá-los de forma desordenada
- Torna-se então necessário encontrar uma forma de transferir dados de forma fiável entre dois computadores
- Nesta lição veremos o protocolo mais simples que existe para o realizar – o protocolo *stop & wait*

Todas as coisas são difíceis antes de se tornarem fáceis

Thomas Fuller

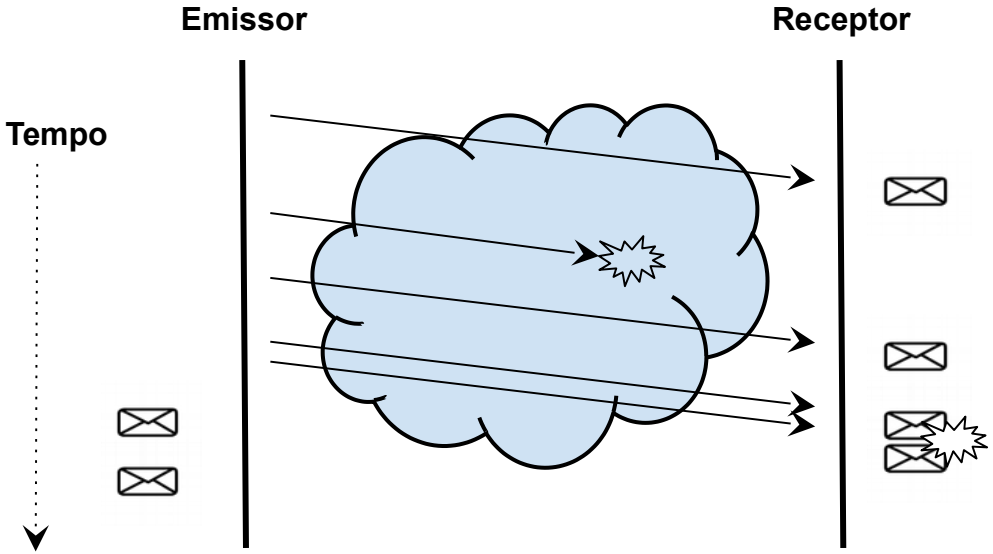
Hipóteses e Objectivo

- Nos canais, os *frames* com erros são recusados pelos mecanismos de controlo de erros e assim os erros são transformados em perdas (omissões)
- A rede também pode introduzir perda de pacotes ou alteração da ordem de entrega dos mesmos
- A rede pode perder pacotes, mas não os perde a todos, mais tarde ou mais cedo alguns pacotes vão chegar ao receptor
- Objectivo: pretende-se transferir dados de forma fiável e maximizar a taxa de transferência, ou débito, extremo a extremo

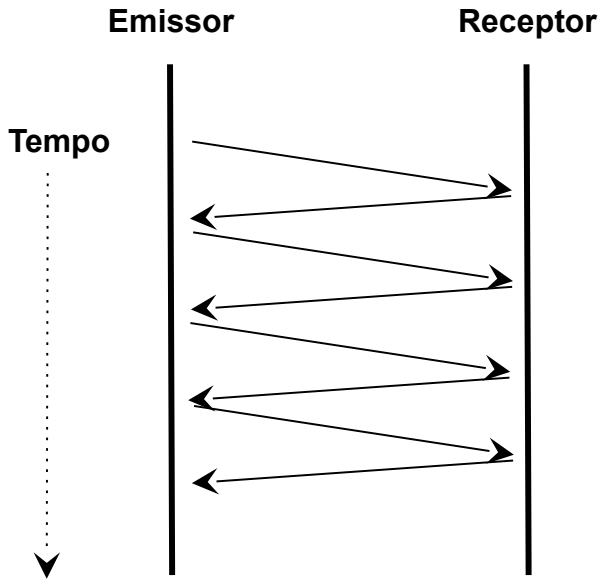
Problemas

É necessário controlo de erros e controlo de fluxo extremo a extremo.

O controlo de erros deverá mascarar a perda de pacotes e a sua chegada fora de ordem. O controlo de fluxos deverá impedir um emissor demasiado rápido de "afogar" um receptor lento.

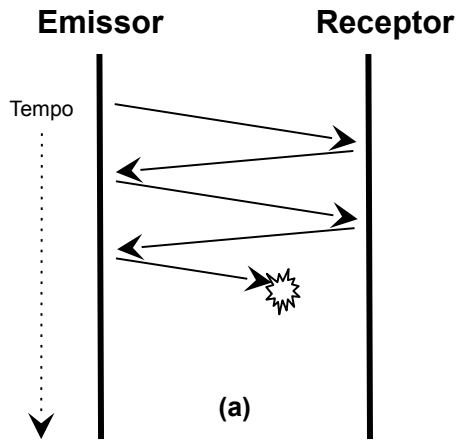


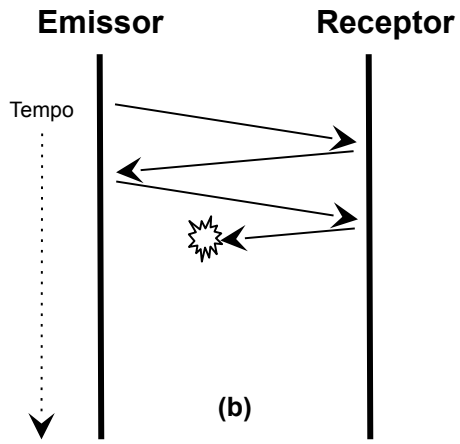
Confirmações de Recepção (ACKs)



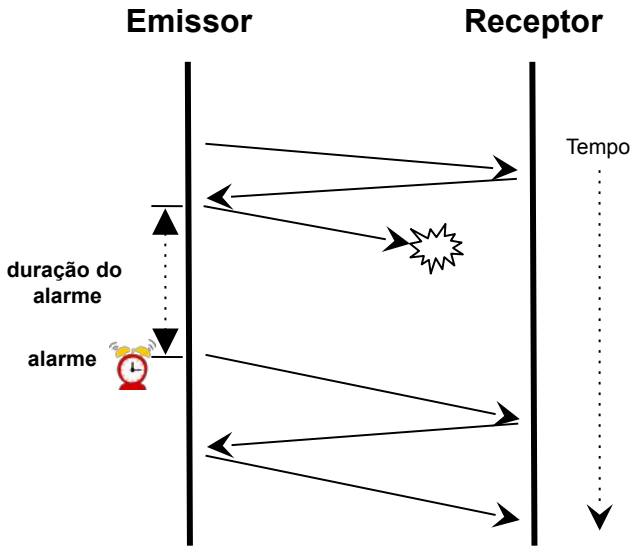
ACK = *Acknowledgement* (confirmação ou aviso de recepção = podes continuar)

Perda de Pacotes



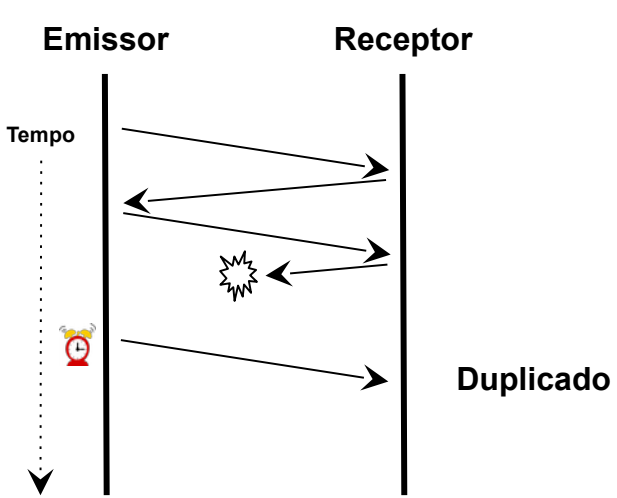


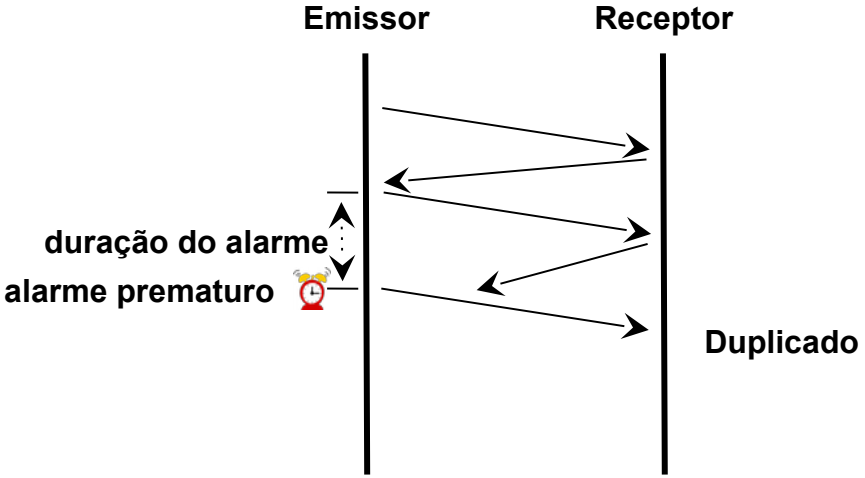
Alarmes Temporizados



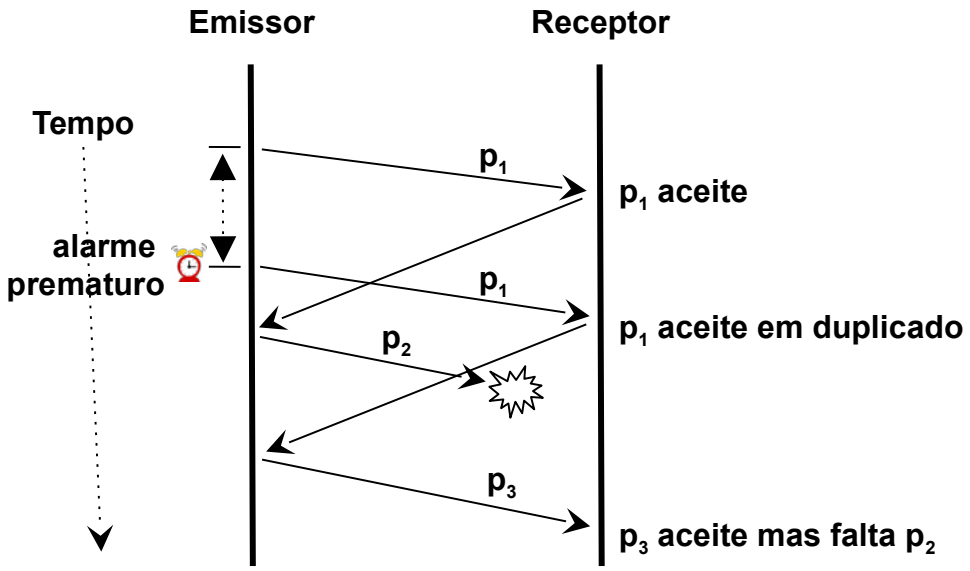
A terminologia usada em inglês nas redes de computadores e nos sistemas distribuídos corresponde ao momento de fim do temporizador do alarme: *timeout*

Anomalias e Alarmes





Anomalias Conjugadas



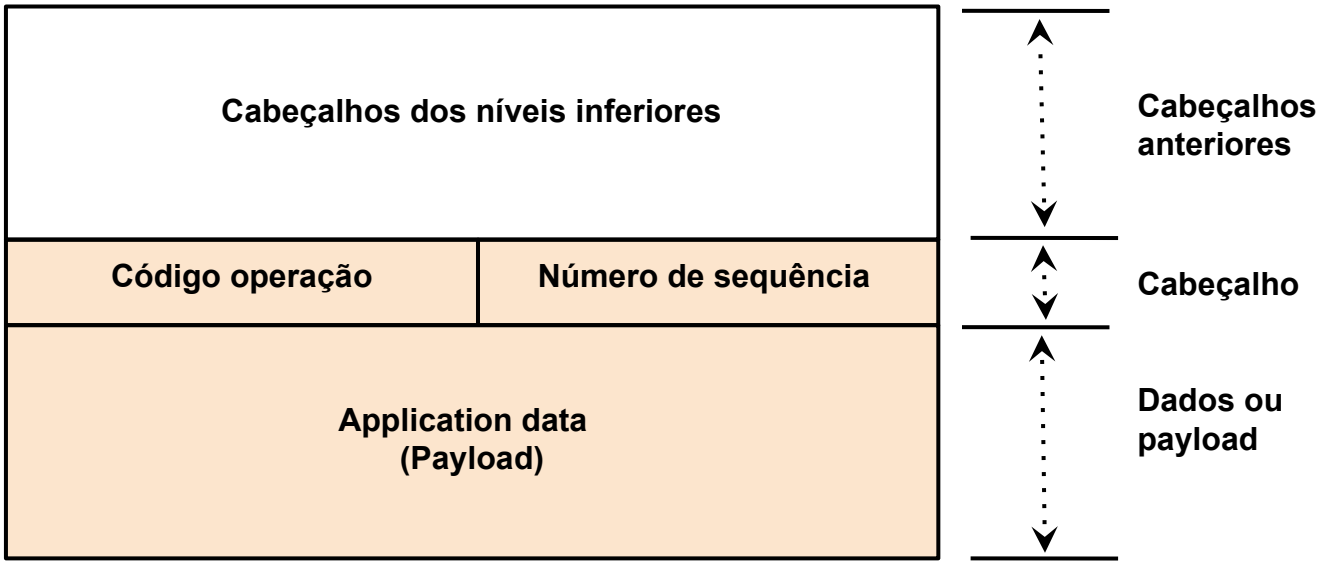
Problemas ainda mal resolvidos

- A solução anterior não resolve ainda o problema da perda de um ACK, a qual pode conduzir à aceitação de um pacote em duplicado
- O mesmo problema poderá ser introduzido por um receptor lento (face ao *timeout*) a enviar o ACK
- Um alarme (*timeout)* mal regulado, muito curto por exemplo, poderá conduzir ao mesmo problema
- Na verdade as velocidades relativas do emissor e receptor podem não ser constantes, nem conhecidas a priori
- Note-se que uma duração de alarme muito elevada, uma solução simplista, leva a uma recuperação demasiado lenta de uma perda

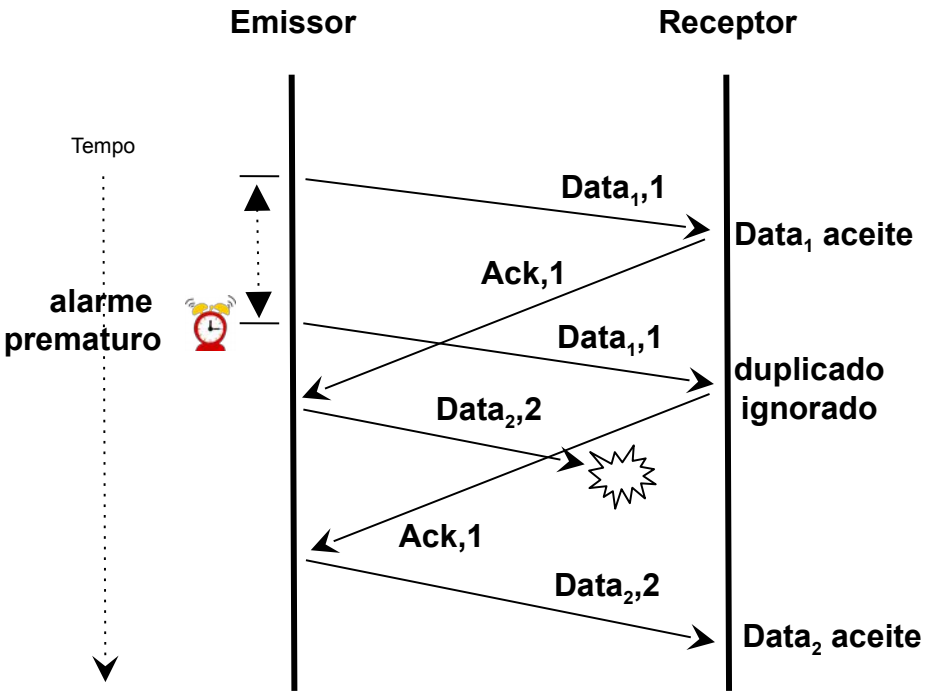
Números de sequência

- A solução consiste em introduzir números de sequência únicos para cada pacote. Estes números permitem ao receptor distinguir dados esperados, dados repetidos, etc.

- Tradicionalmente, para se poupar espaço nos cabeçalhos, interessa minimizar o número de bits a usar. O número de sequência pode então ser reutilizado ciclicamente desde que não se introduza confusão caso a ordem das mensagens possa ser trocada



Protocolo *stop & wait*



Funcionamento

Emissor:

- Por cada pacote a transmitir, dá-lhe um número de sequência novo, transmite-o e activa um alarme. Depois:
  - Se chega um ACK com o número de sequência esperado - passa adiante
  - Se o temporizador dispara – reenvia o pacote e activa de novo o alarme
  - Se chega um ACK com número de sequência errado - ignora-o

Receptor:

- Sempre que recebe um pacote:
  - Envia um ACK do último pacote correctamente recebido. Se o pacote é novo guarda-o, senão ignora-o

Correção (*Safety*)

- Admitindo que a rede mais tarde ou mais cedo consegue que uma mensagem seja entregue ao receptor, isto é, que a rede permite sempre algum progresso *mesmo que pequeno*:
- O protocolo garante (tente demonstar):
  - Que cada pacote é corretamente entregue ao receptor
  - Que esse pacote não é confundido com outro
  - Que todos os pacotes chegarão ao receptor

Regulação dos Alarmes

- O valor do temporizador do alarme (*timeout*) é importante para a recuperação de erros
  - Um valor demasiado grande leva a que o emissor seja lento a recuperar dos erros
  - Um valor demasiado curto leva a duplicados inúteis
  - Em qualquer caso a correção não está em causa desde que se use sempre um alarme e o receptor envie sistematicamente um ACK perante todos os pacotes recebidos, maus ou bons
- Para evitar repetições inúteis o valor pode ser folgado
  - É possível tentar estimar um valor mais adequado
  - Pense como

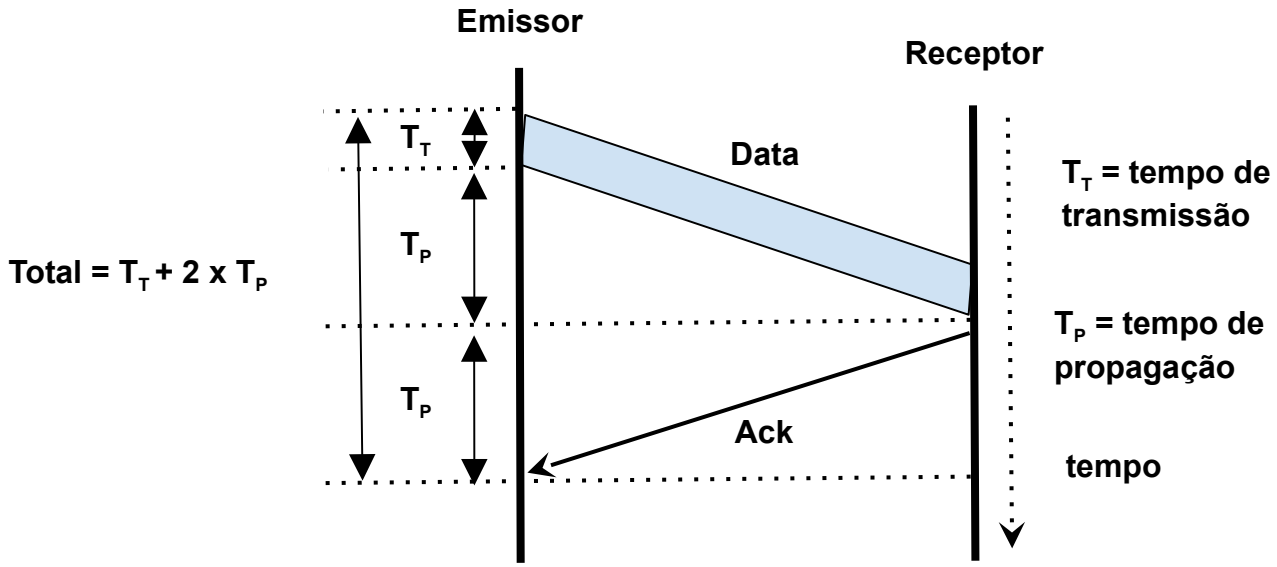
Desempenho sem Erros

- Dois computadores A e B ligados por um canal directo com 10.000 Km de comprimento e o débito de 1 Mbps
- Pacotes com 1.000 bytes
- Ignoram-se os bits dos cabeçalhos
- Ignora-se o tempo de transmissão de um ACK
- Ignora-se o tempo de processamento

Tempo de propagação = 10.000 / 200.000 = 50 ms, RTT = 100 ms
1000 bytes = 8.000 bits = $8 . 10^3$
Tempo de transmissão = $8 . 10^3 / 10^6$ = 8 ms

- Quanto tempo leva o emissor a colocar cada pacote no receptor ?
- Qual a diferença para um protocolo em que o emissor estivesse sempre a emitir?

Desempenho do Protocolo Stop & Wait



Cada pacote leva 8 ms a transmitir, mas em cada ciclo de 108 ms apenas se transmite um pacote

1.000 pacotes levam 108 segundos a transferir ao invés de 8 segundos

O desempenho real é 8/108 ≈ 7,4% do máximo teórico

Taxa de Utilização do Canal

$$T_u = T_t / (T_t + 2 \times Tp)$$

ou

$$T_u = T_t / (T_t + RTT)$$

T_u - taxa de utilização

T_t - tempo de transmissão de uma mensagem,

T_p - tempo de propagação de extremo a extremo

RTT - tempo de ida e volta (2 x T_p)

Débito, RTT e Taxa de Utilização

Taxa de utilização de um canal pelo protocolo stop & wait numa transferência entre dois computadores ligados directamente por um canal sem erros, usando pacotes de 10.000 bits, variando o débito e o RTT.

Taxa Utilização	Débito (Mbps)	T_T (ms)	RTT (ms)	Comentários
99%	1	10	0,1	Débito baixo, RTT desprezável
50%	100	0,1	0,1	Débito razoável, RTT desprezável
33%	1	10	20	Débito baixo, RTT baixo
0,5%	100	0,1	20	Débito razoável, RTT baixo
4,8%	1	10	200	Débito baixo, RTT alto
0,05%	100	0,1	200	Débito razoável, RTT alto
0,005%	1000	0,01	200	Débito alto, RTT alto

Canais de Débito Elevado

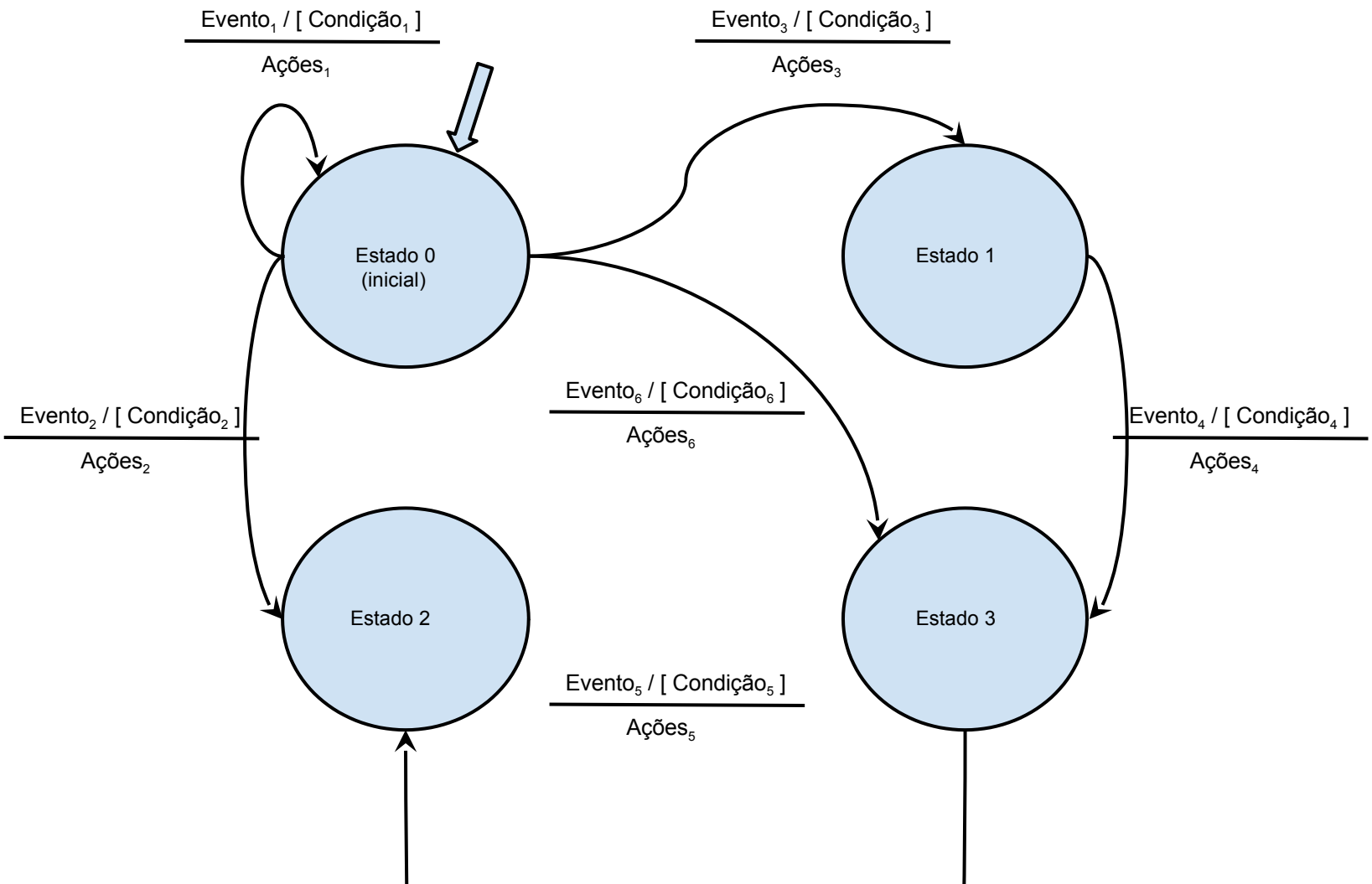
- Quando os canais têm débito muito elevado, o tempo de transmissão diminui drasticamente. Por exemplo, se um canal tem a capacidade de 1 Gbps, transmitir 10.000 bits leva 10^{-5} segundos, isto é, 10 micro segundos
- Se o RTT for de alguns milissegundos, a taxa de utilização do canal tende sempre para valores muito baixos. Nestes casos, o débito útil médio extremo a extremo permitido pelo protocolo tende para:

Débito útil médio extremo a extremo do protocolo S&W

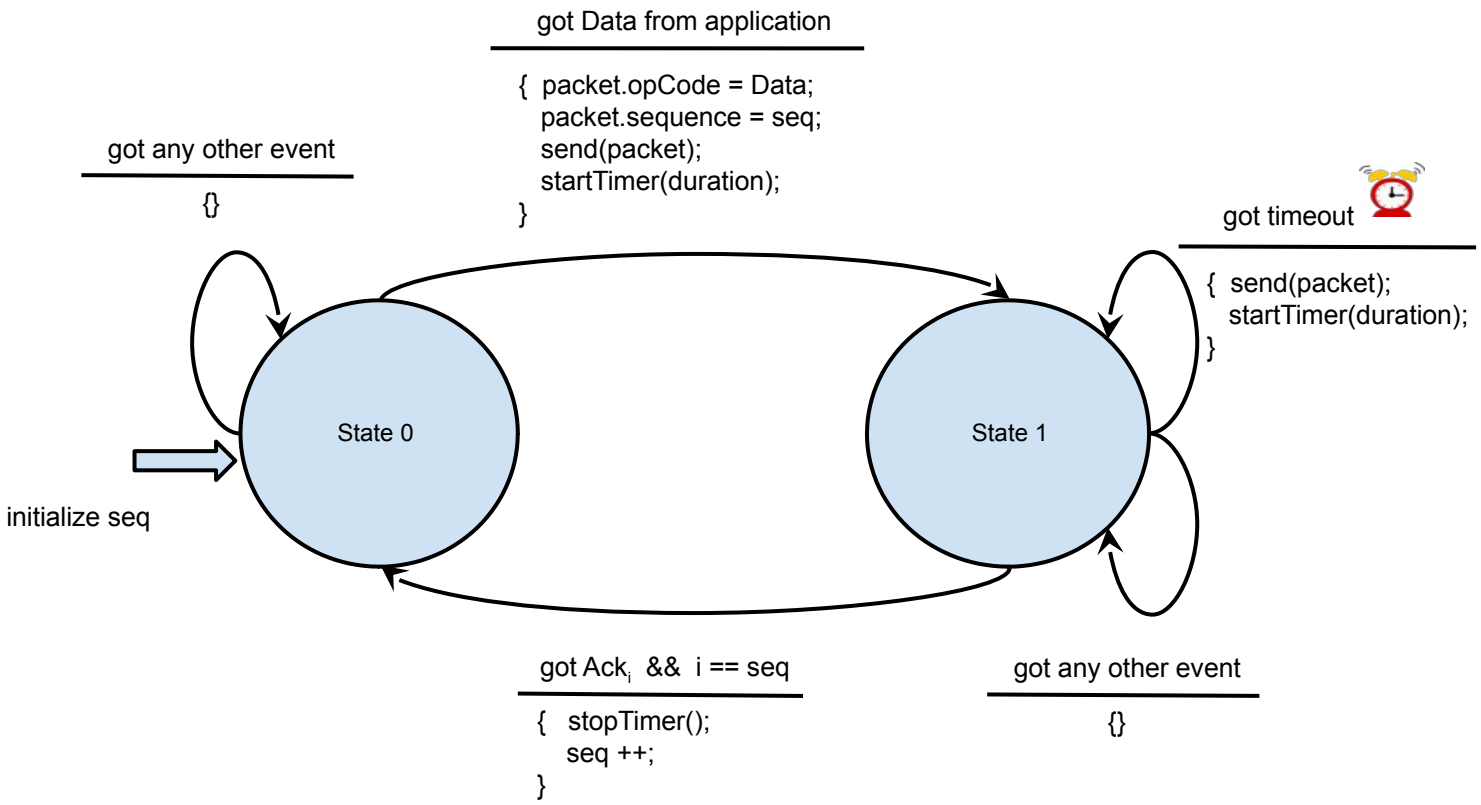
= Dimensão do pacote / RTT

- O débito útil extremo a extremo diz-se *goodput* em inglês

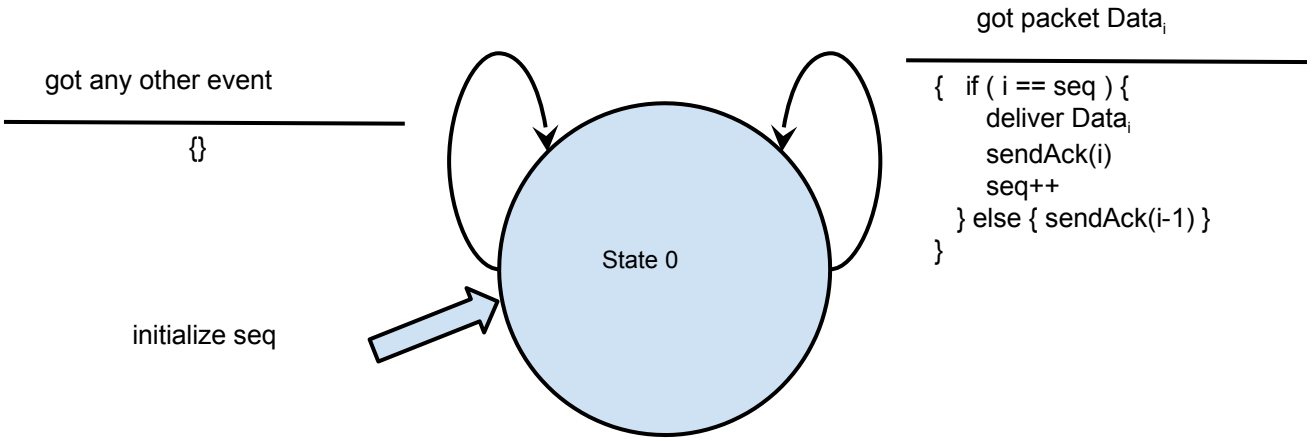
Máquinas de Estados com Acções



Máquinas de Estados com Acções
Emissor Stop & Wait



Máquinas de Estados com Acções
Receptor Stop & Wait



Conclusões

- Mesmo que um canal ou a Internet tenham erros e perdam pacotes
  - É possível realizar um protocolo de transferência fiável de dados nos extremos
  - Acabámos de ver um, também conhecido por *stop & wait*
- Infelizmente tem um desempenho fraco quando o tempo de trânsito extremo a extremo é significativo face ao tempo de transmissão
  - O emissor avança ao ritmo permitido pelo RTT, pois
  - O receptor não pode emitir um novo pacote enquanto não chegar o ACK do último pacote emitido