

Redes de Computadores

Protocolos de Janela Deslizante

Departamento de Informática da
FCT/UNL

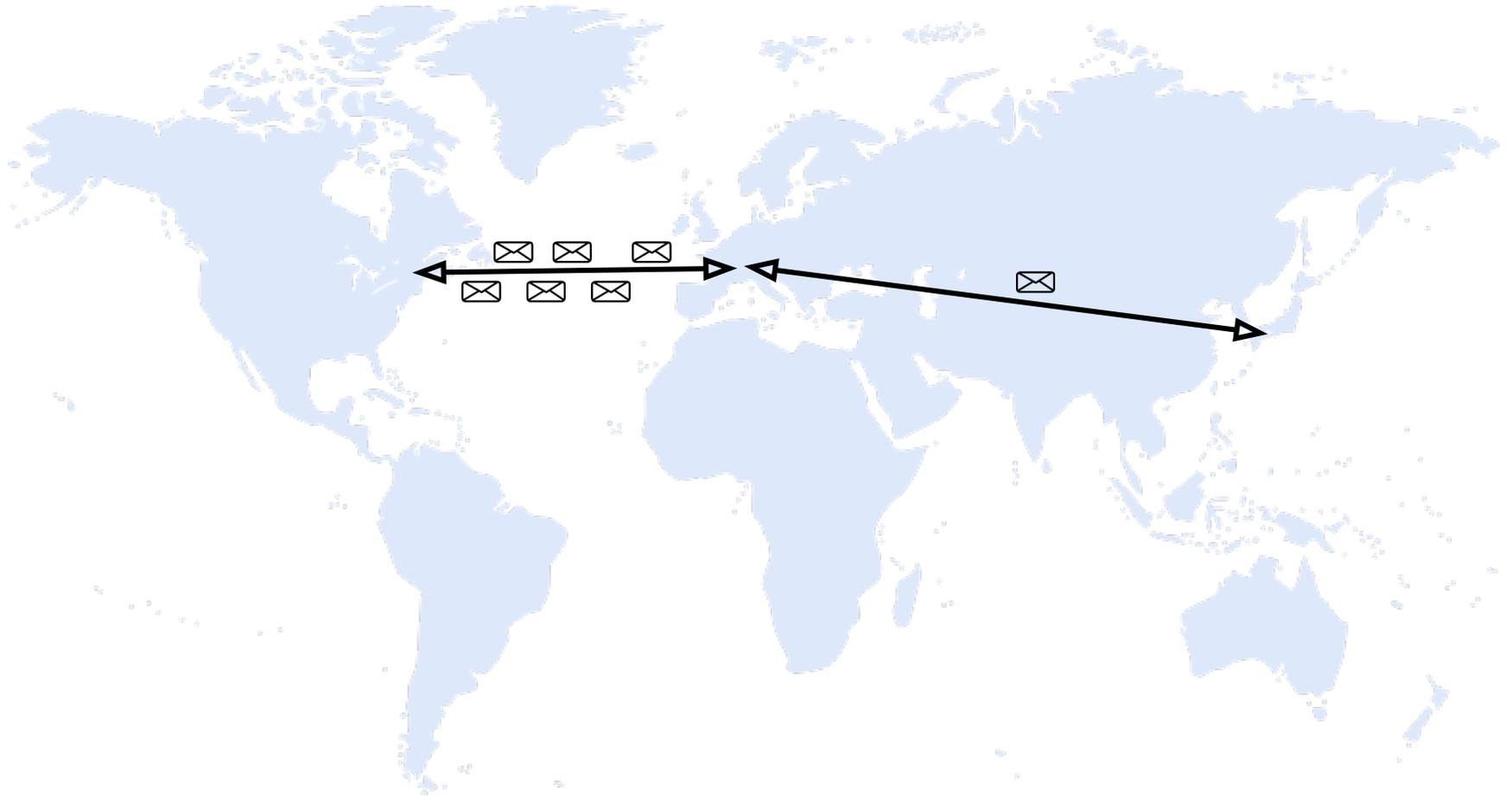
Objectivos da lição

- Para a transmissão fiável de dados uma solução simples consiste em usar um protocolo que só transmite um pacote de cada vez e só passa ao seguinte depois de receber um ACK
- Esses protocolos têm pouco rendimento
- É possível melhorá-los usando uma técnica chamada janela deslizante (*sliding window*) ou *pipelining*

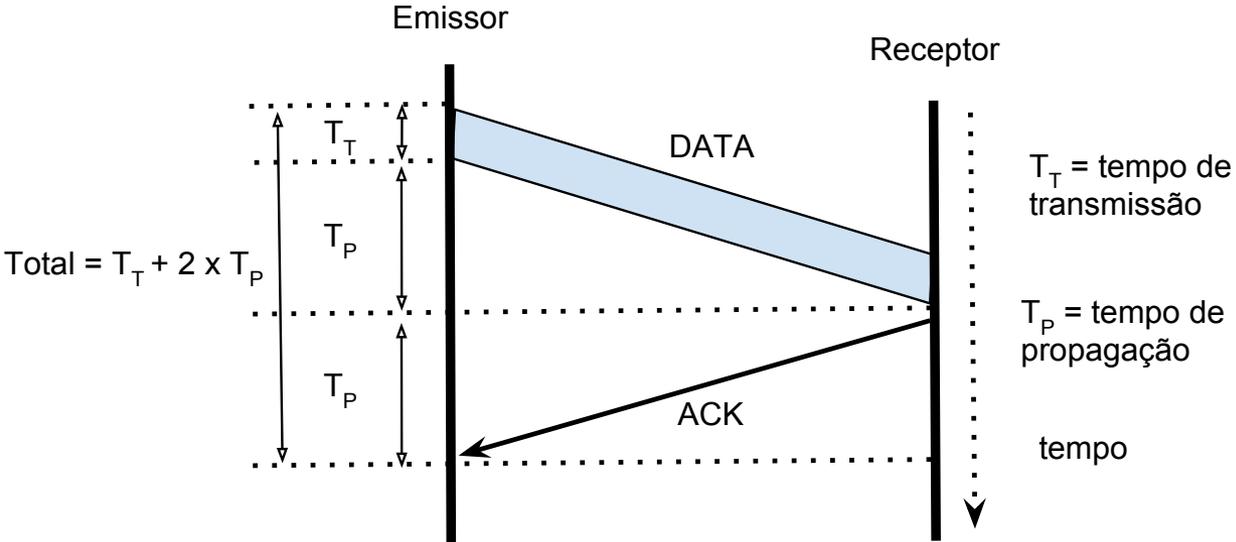
Todas as coisas são difíceis antes de se tornarem fáceis

Thomas Fuller

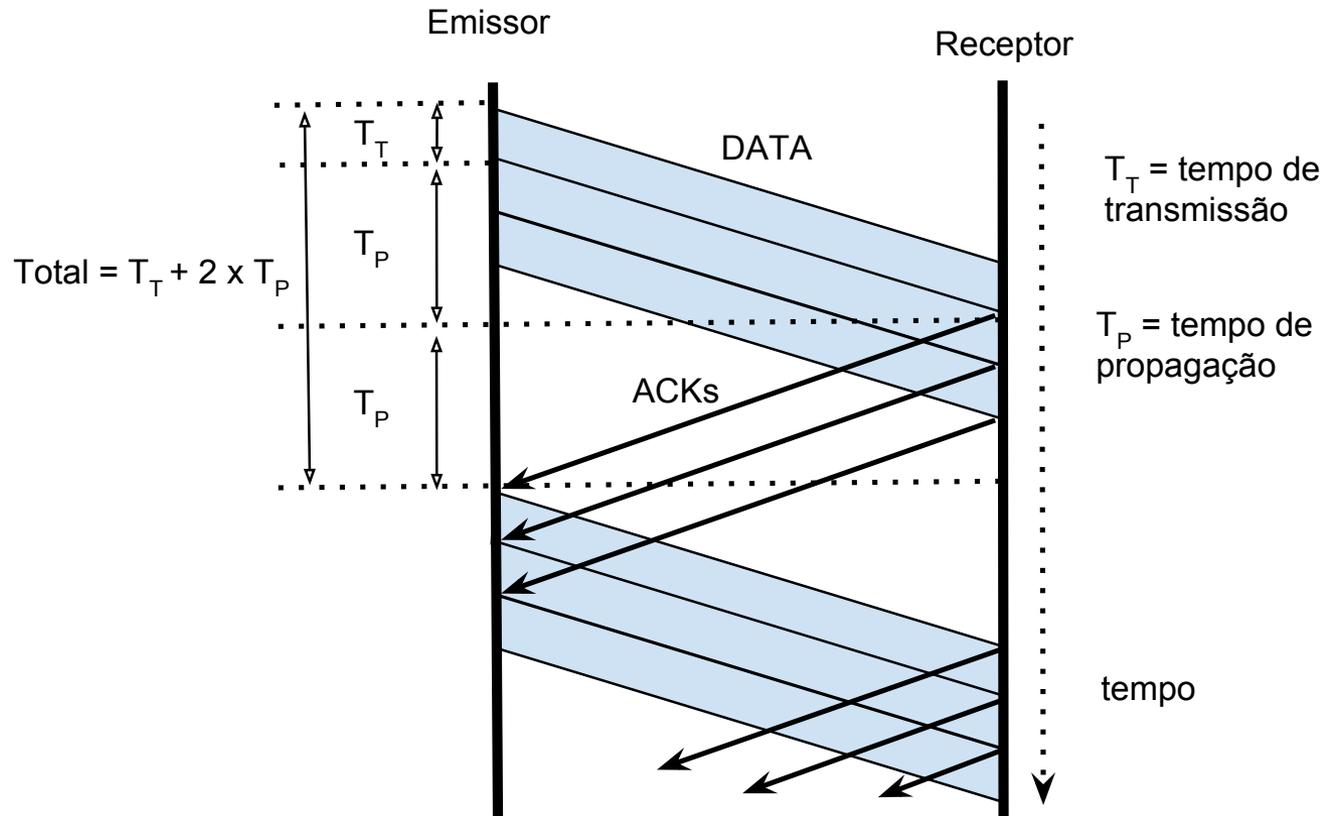
O Problema



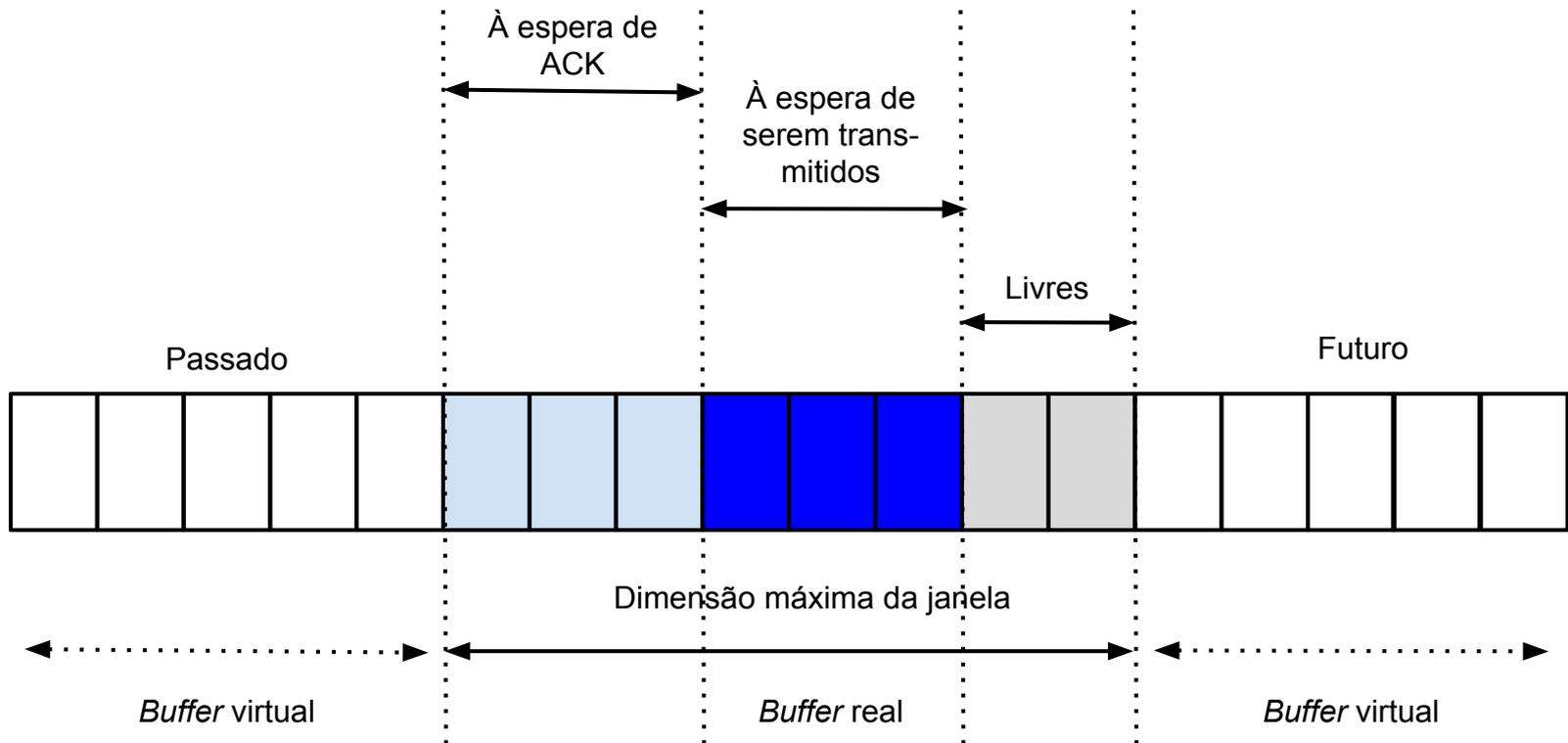
Desempenho do Protocolo S&W



Solução: Transmitir "Adiantado"



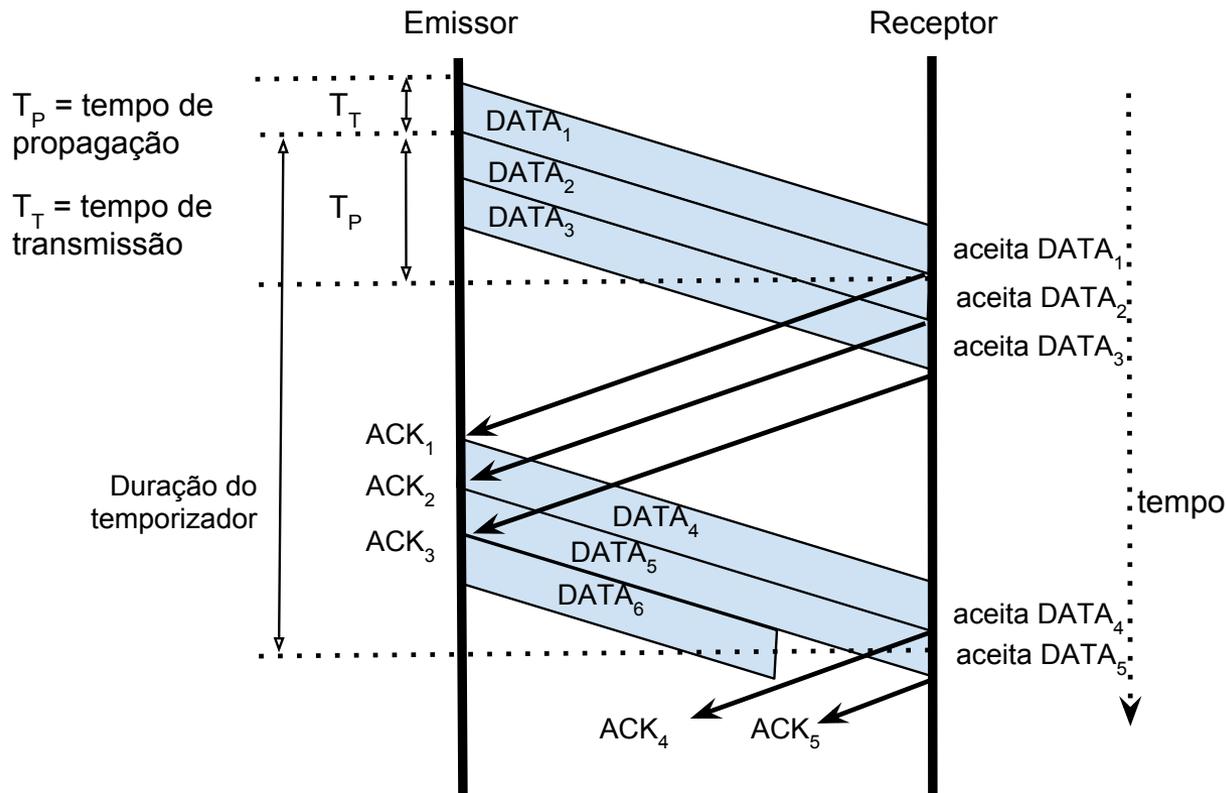
Funcionamento da Janela



Funcionamento da Janela

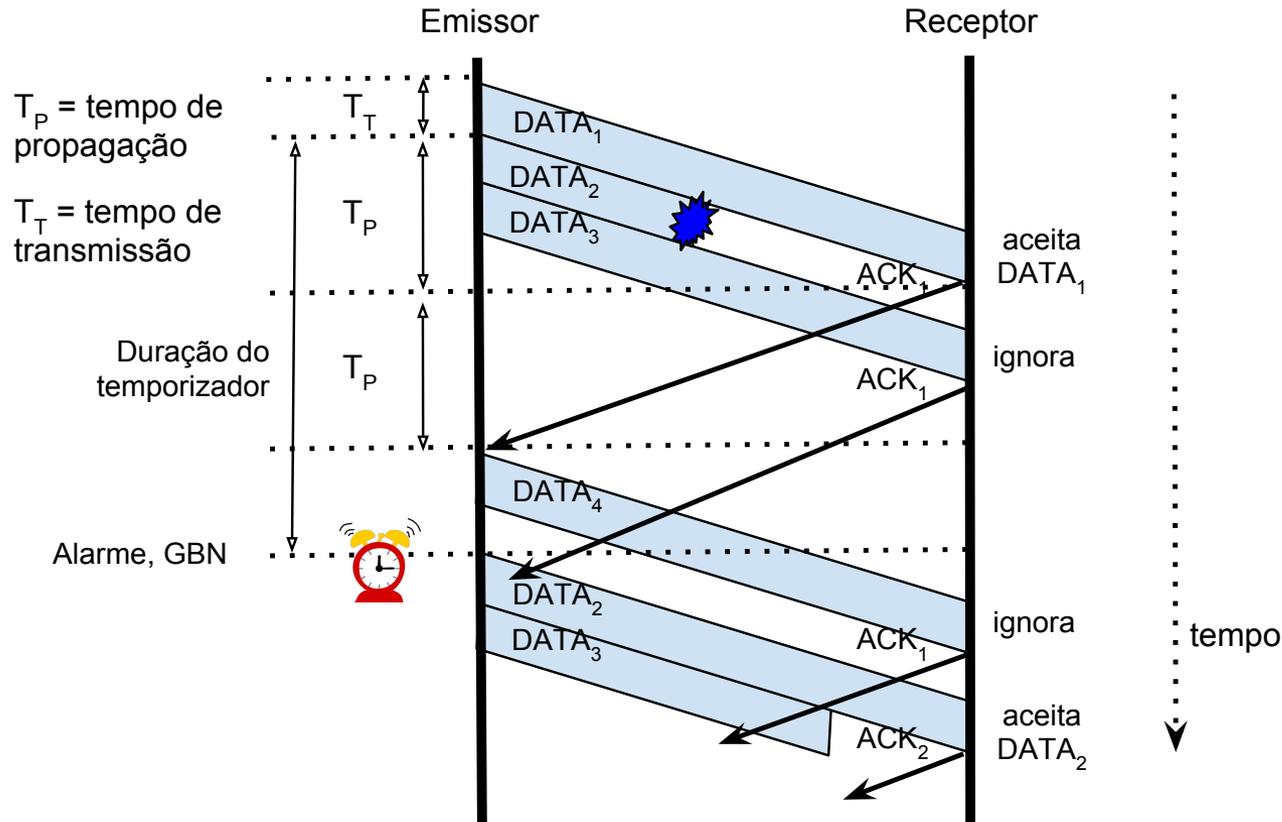
- O emissor tem um *buffer* (a janela) onde estão os pacotes já transmitidas e que ainda não foram *ACK'd*, assim como os pacotes que estão à espera de serem transmitidas mas ainda não houve tempo
- Quando chegam *ACKs*, a janela desliza para a direita e os pacotes já *ACK'd* podem ser esquecidos pois já se tem a certeza que foram bem recebidos
- A dimensão da janela é limitada pelas seguintes razões: controlo de fluxo, eventual desperdício em caso de perda e controlo da saturação da rede (ver a seguir)

Recetor com Janela para um Pacote



Uma janela para um pacote no recetor é suficiente caso a aplicação consuma muito rapidamente os pacotes chegados

Mas se Há Erros



não há espaço no recetor para os pacotes fora de ordem, pelo que o emissor tem de voltar atrás!

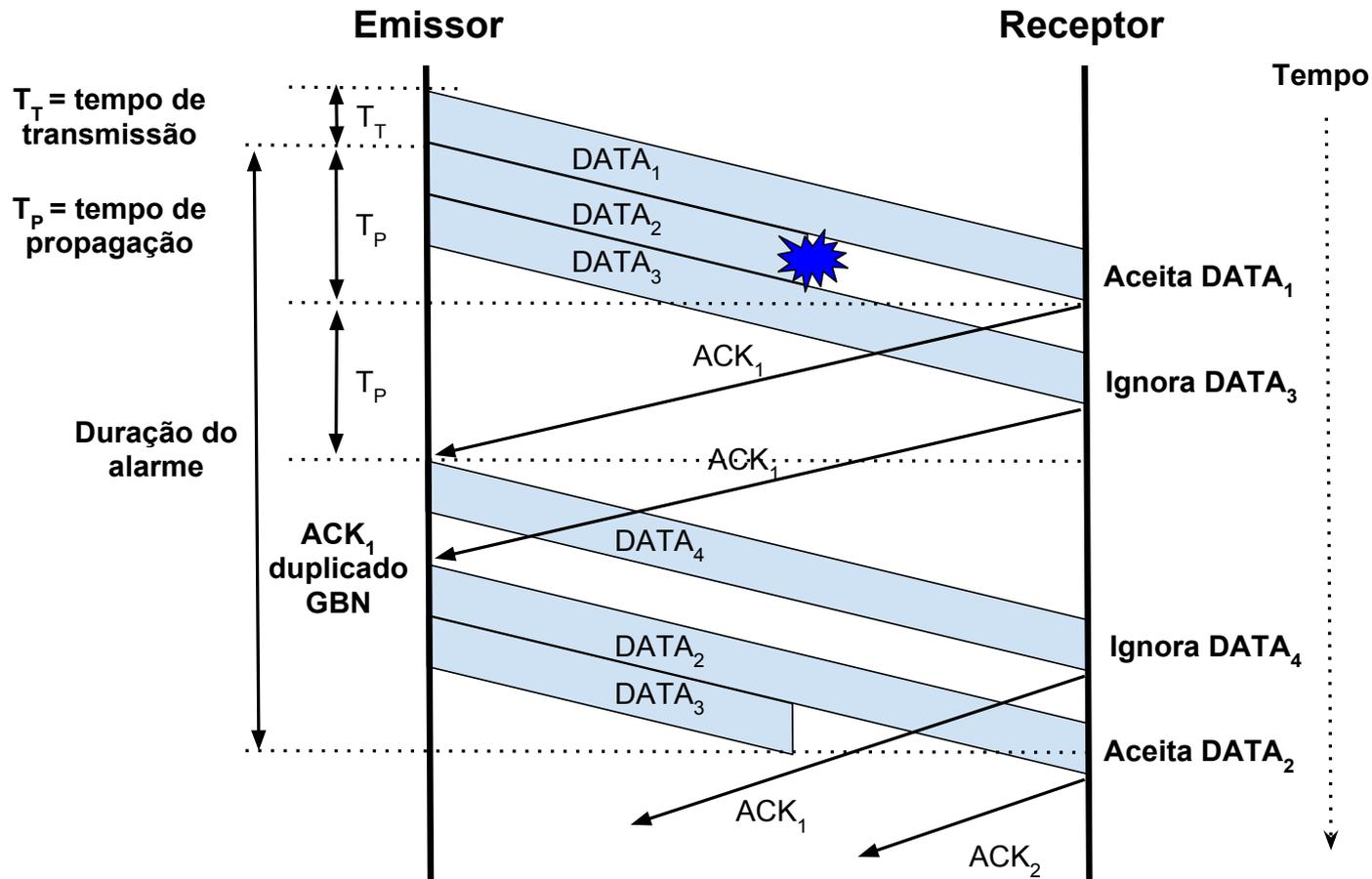
Go-Back-N (GBN)

- Como o recetor não guarda os pacotes fora de ordem, em caso de anomalia o emissor não recebe o ACK adequado, e só lhe resta recomeçar a emissão por ordem de todos os pacotes a partir do pacote cujo ACK faltou, ou seja, o mais à esquerda da janela.
- Existe sempre um alarme associado ao pacote emitido mais antigo, que terá de ser atualizado sempre que este pacote é ACKed
- Existem outras formas de detetar antes do disparo do alarme que um pacote se perdeu (ver a seguir)

GBN e Recuperação

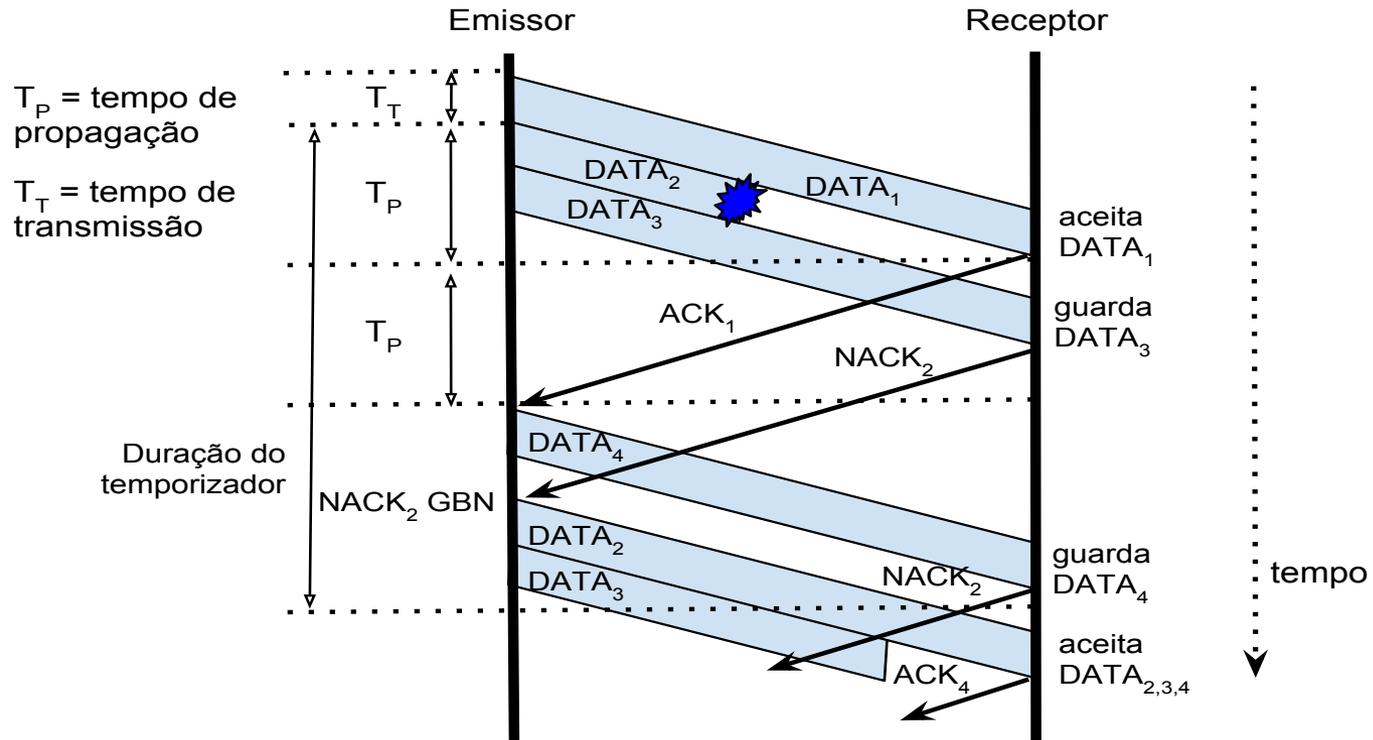
- Quanto mais cedo o emissor detetar que se perdeu um pacote melhor, pois isso evita continuar a emitir pacotes que vão ser deitados fora
- Existem pelo menos duas formas de o emissor detetar que um pacote se perdeu mesmo que o alarme ainda não tenha disparado
 - Interpretar os ACKs duplicados como sinal de que um pacote se perdeu
 - O recetor emitir uma indicação explícita (um NACK) de que um pacote se perdeu

GBN Com Recuperação Mais Rápida



Um **ACK** cumulativo repetido pode ser interpretado como um sinal de que um pacote se perdeu e serve para entrar em **GBN** mais cedo

Janela do Recetor Maior Que Um Pacote

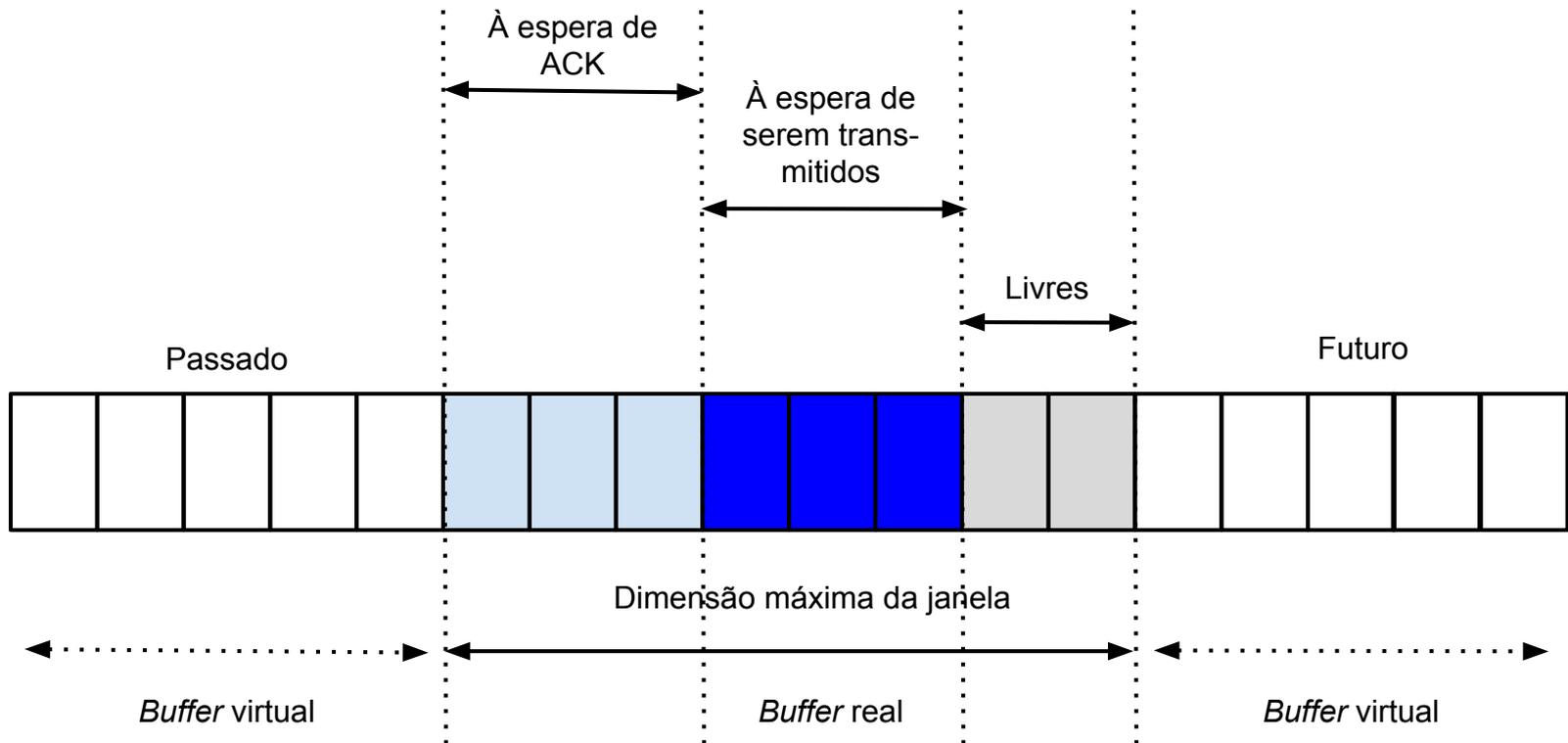


Uma janela do recetor maior que um também ajuda

Ações Executadas pelo GBN

- Por cada mensagem n recebida, o recetor
 - Guarda o pacote se este estiver na ordem
 - Se estiver fora de ordem, mas tiver a janela de receção > 1 tenta guardá-lo também
 - Em qualquer caso envia um ACK cumulativo de tudo o que recebeu até aí na ordem, e opcionalmente pode enviar (também) um NACK
 -
- O emissor arma um alarme (*timeout*) associado ao pacote mais antigo enviado
- Sempre que o *timeout* dispara no emissor, ou este recebe um NACK, volta atrás e recomeça a enviar os pacotes na janela
- Quando o emissor recebe um ACK "útil", isto é, à esquerda da janela, faz avançar a janela e reajusta o valor do alarme

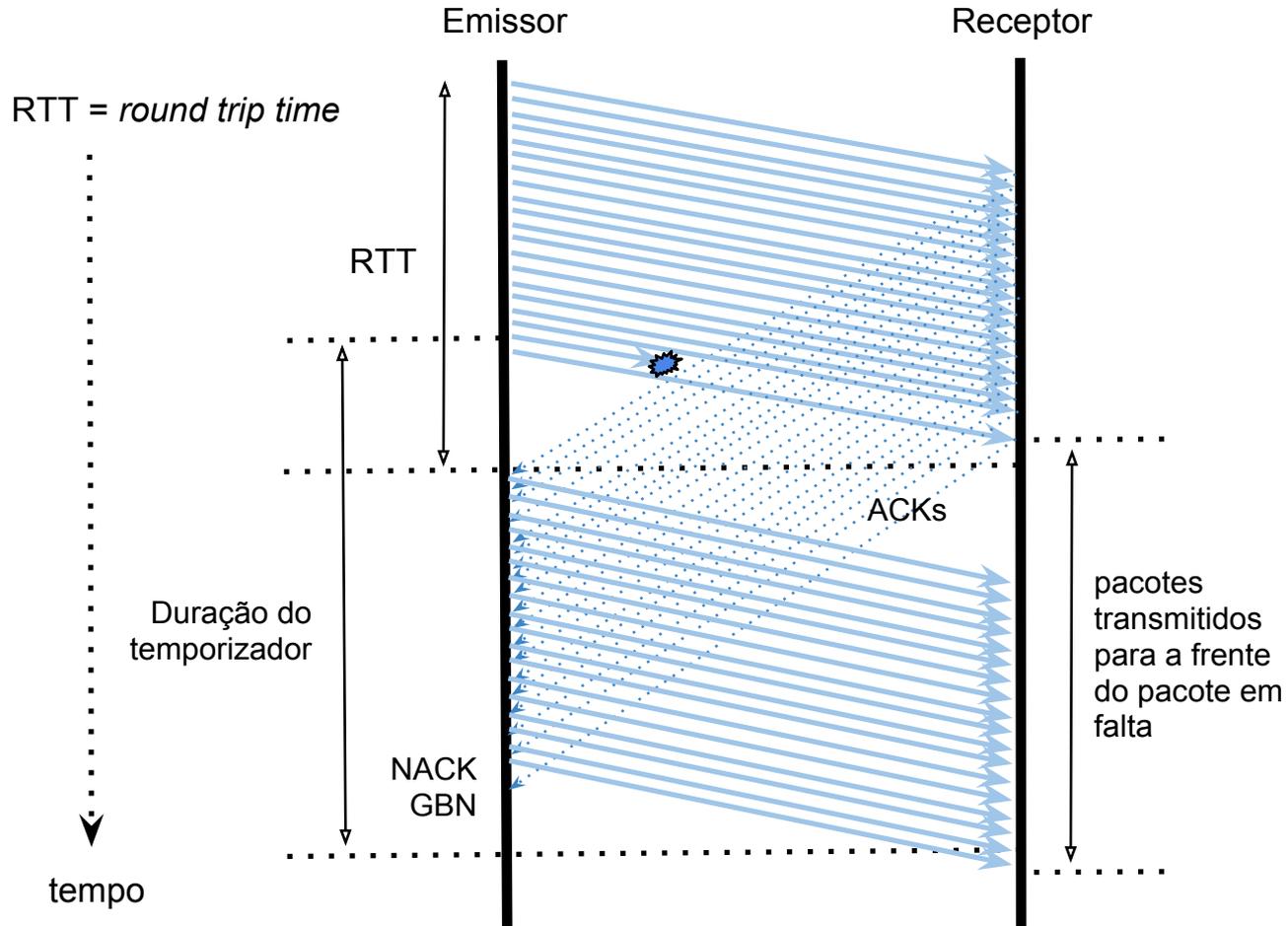
Janela do Emissor



Podemos Melhorar?

- Quando o RTT é muito baixo, ou o valor do Tt / RTT se aproxima de 1, uma janela de emissão relativamente pequena é suficiente para manter o emissor quase sempre a emitir
- Se houverem erros de transmissão (admitamos que não são frequentes) o funcionamento GBN introduz atrasos e eventualmente emissões em duplicado. A penalização, se existir, é proporcional ao tamanho da janela
- Se o valor de Tt é muito pequeno (canal de alta capacidade) e o RTT é muito grande (canal muito extenso), a janela do emissor tem de ser muito grande e o funcionamento GBN é muito penalizante se houverem erros, mesmo que espaçados

Custo da Recuperação com GBN



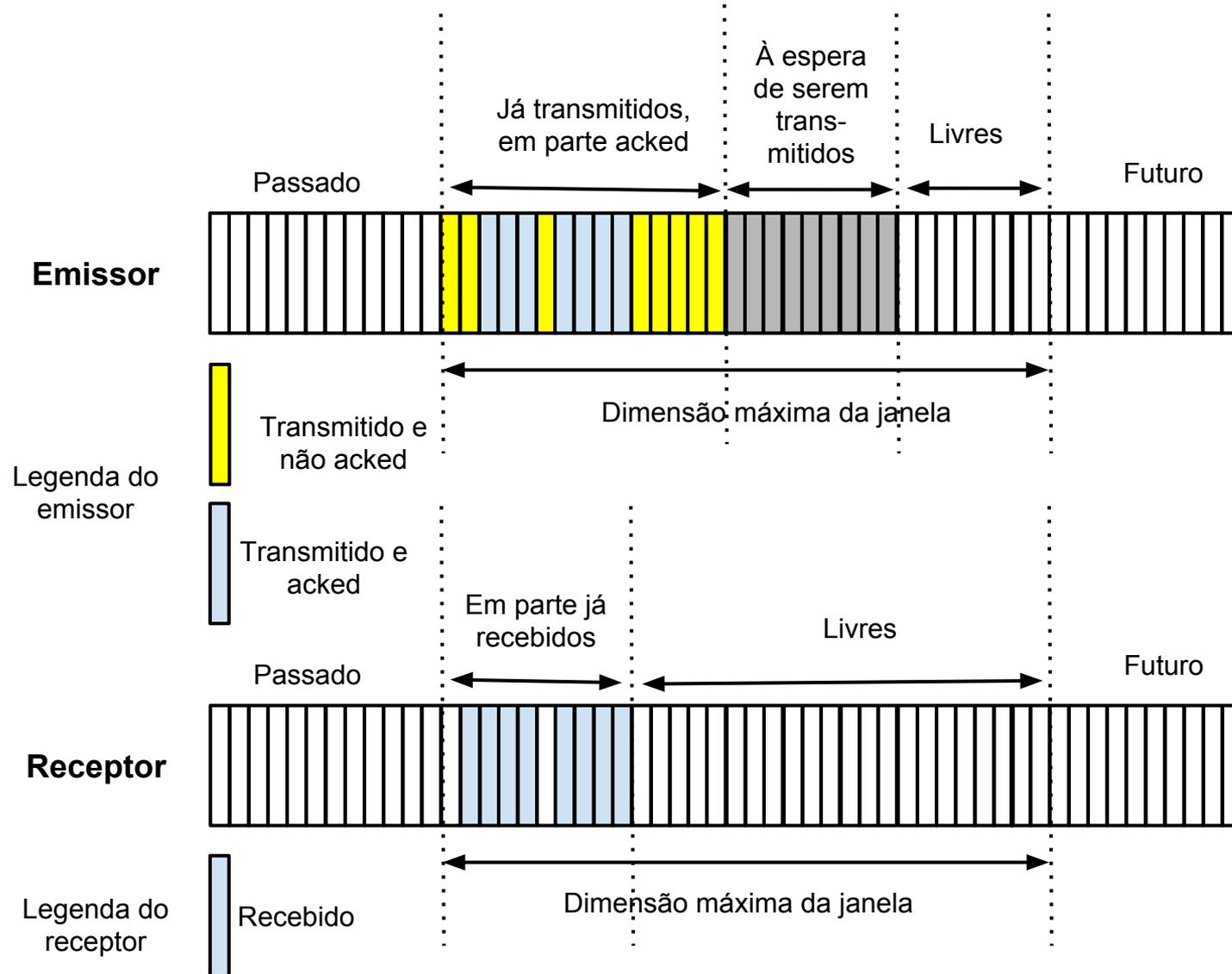
Repetição Seletiva (RS)

- É possível melhorar o algoritmo introduzindo ACKs independentes para cada pacote enviado e recebido
- Significa isso que só são retransmitidos os pacotes para os quais o emissor recebeu um NACK ou um alarme disparou
- Nesses casos, o pacote perdido é retransmitido, mas não se executa o procedimento GBN. O emissor continua sempre a enviar novas mensagens enquanto o tamanho da janela o permitir
- Esta versão do protocolo designa-se por *selective repeat* (SR) ou *repetição seletiva* (RS)

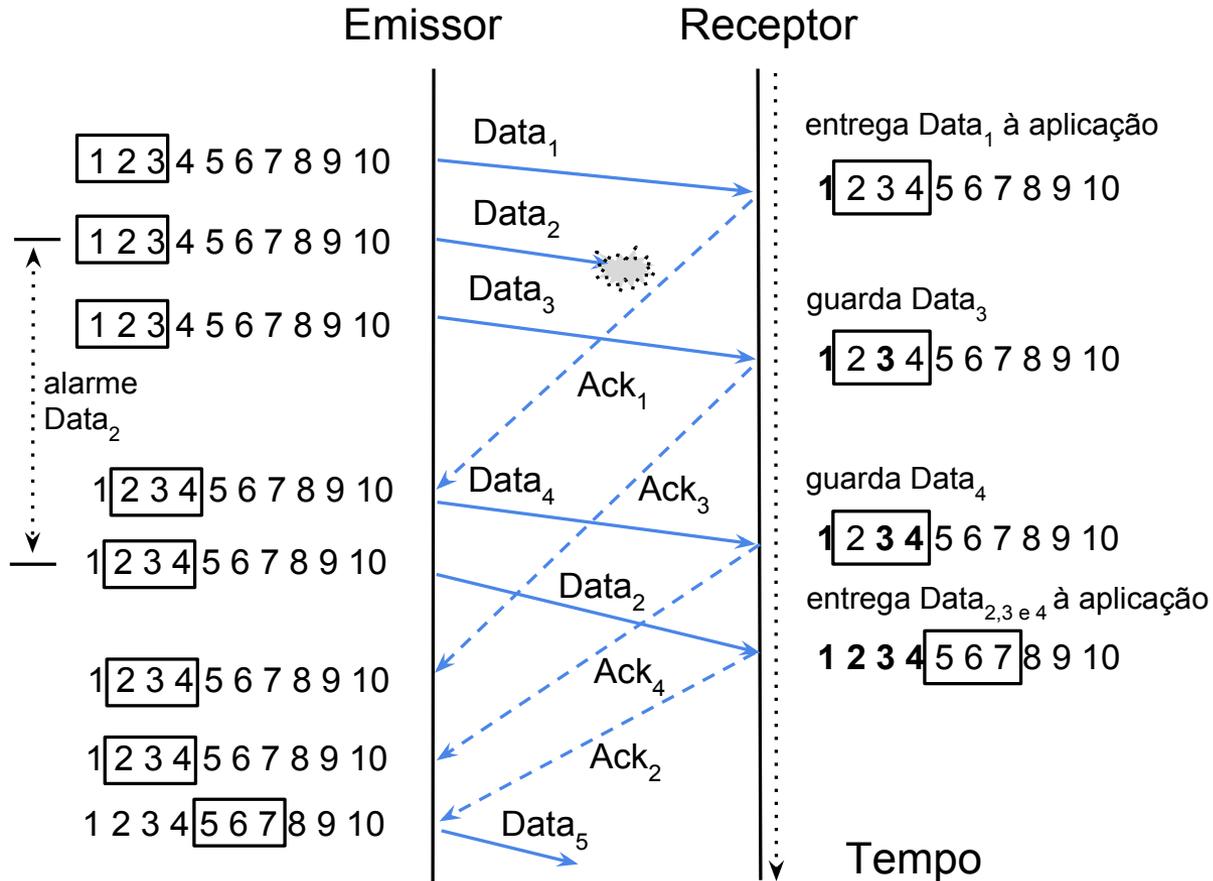
Funcionamento do Protocolo RS

- Por cada pacote n recebido, o receptor envia o respectivo $ACK(n)$ (pode também enviar um $NACK$ do mais antigo pacote que lhe falta se tem um buraco)
- Por cada pacote n enviado, o emissor arma um *timeout* $T(n)$
- Sempre que um *timeout* $T(n)$ dispara no emissor (ou este recebe um $NACK(n)$) o pacote n é reenviado
- O emissor continua limitado pela dimensão máxima da sua janela

Janelas no SR



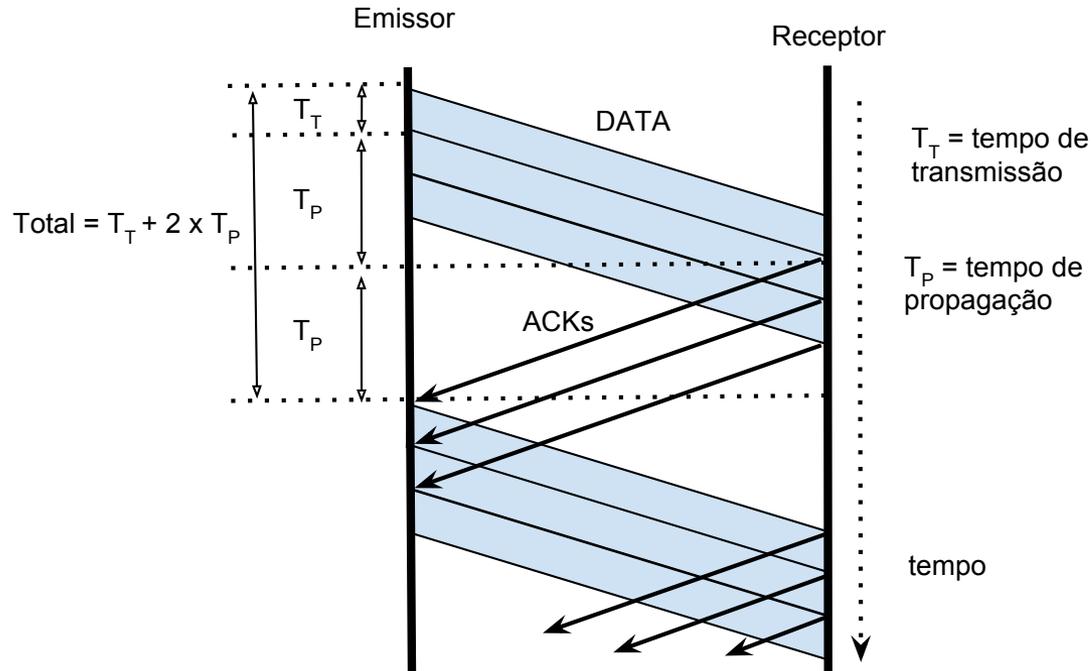
Exemplo



Nomenclatura

- Janelas (emissor, receptor)
- (1,1) — *stop & wait*
- (N,*) — janela deslizante ou *pipelining*
- (N,1) — janela deslizante com *Go-Back-N (GBN)*
- (N,M) — com ACKs apenas do que foi bem recebido de forma contígua (ACKs acumulativos) continua a ser GBN mas com recuperação mais rápida
- (N,M) — com ACKs do que foi bem recebido mesmo que de forma não contígua (ACKs selectivos) — *selective repeat*

Desempenho Sem Erros



Débito útil médio extremo a extremo do protocolo

\approx Dimensão "útil" da janela em bits / ($T_T + RTT$)

$\approx N \times$ Débito do protocolo Stop & Wait

Canais com Débito Muito 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 micros segundos
- Se o RTT for de alguns milissegundos e o T_T desprezável, o débito útil médio extremo a extremo permitido pelo protocolo tende para:

Débito útil médio extremo a extremo do protocolo
= Dimensão da janela "útil" em bits / RTT

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

Dimensão da Janela do Emissor

- Não vale a pena ser maior do que o que se consegue emitir continuamente durante um RTT
- Depende também da versão do protocolo pois janelas enormes com GBN incrementam o desperdício potencial a quando da recuperação dos erros (são estes frequentes?)
- Uma janela muito grande também pode potencialmente afogar um receptor lento (controlo de fluxo) ou saturar a rede (controlo da saturação)
- Por isso TCP usa uma janela de dimensão variável

Dimensão da Janela do Recetor

- Teoricamente não vale a pena ser maior do que a do emissor
- No caso geral, implica gerir bocados em falta antes de os entregar à aplicação o que é mais complicado
- Por isso a solução GBN e janela do recetor = 1 não é assim tão má desde que a relação T_T / RTT não seja demasiado pequena
- Por isso o TCP usa uma janela de receção de valor constante e pode ou não usar a versão SR

Valor dos Alarmes

- Necessariamente superiores ao do RTT
- Se não existirem *buffers* na rede entre o emissor e o receptor (e.g. ambos estão ligados por um canal ponto a ponto direto), o RTT é constante e o valor do *timeout* é mais fácil de estimar
- Se houverem *buffers* pelo meio (e.g. comutadores de pacotes a funcionarem em modo *store & forward* e com filas de espera significativas), o RTT é variável e o valor do *timeout* é mais difícil de estimar
- O protocolo TCP usa um valor de *timeout* ajustado dinamicamente

Conclusões

- A relação entre o tempo de transmissão e o RTT é determinante para o rendimento de um protocolo stop & wait
- Quando esse rendimento é baixo (e.g. $T_t \ll RTT$) é fundamental usar protocolos de janela deslizante para melhorar o rendimento
 - É o caso dominante na Internet quando os parceiros em comunicação estão "longe"
- Trata-se de um protocolo complexo com imensos parâmetros ajustáveis a cada cenário
 - O protocolo TCP é um protocolo de janela deslizante que adapta dinamicamente esses parâmetros à cada situação concreta em que é usado