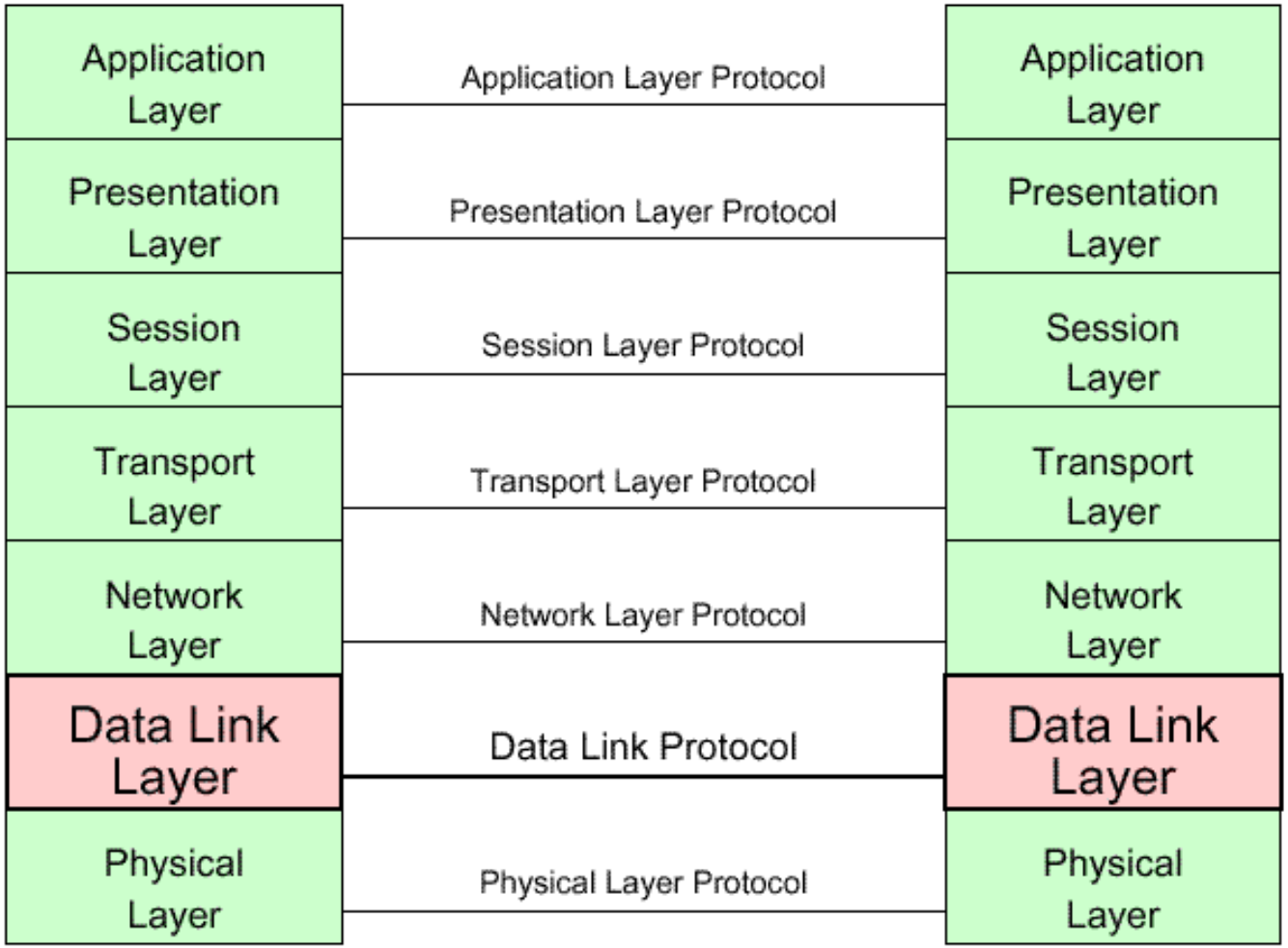
Controlo da Ligação de Dados

FEUP/DEEC
Redes de Computadores
MIEEC – 2010/11
José Ruela

Camada de Ligação de Dados (modelo OSI)



Ligação de Dados – conceito e objectivos

- Uma ligação física constitui o suporte para a transmissão de sinais que representam sequências de bits e que está sujeita a erros e outras anomalias
- O processo de transmissão não requer nem confere qualquer organização aos dados trocados na ligação, nem permite gerir e controlar a troca de dados
- A comunicação de dados entre sistemas exige funções adicionais, por exemplo
  - Estruturar e delimitar sequências de bits em unidades de comunicação (por exemplo, caracteres, tramas)
  - Gerir e controlar a troca de dados (quando transmitir, identificar quem transmite / recebe)
  - Detectar e eventualmente corrigir (recuperar) erros de transmissão
  - Regular o fluxo de dados entre emissor e receptor
- A execução destas funções (e possivelmente outras) requer um Protocolo de Ligação de Dados (ou de Ligação Lógica)
- Uma Ligação de Dados consiste numa associação lógica estabelecida entre entidades que utilizam uma ligação física não fiável para trocar dados de forma estruturada, controlada e, se necessário, fiável

Ligação de Dados – funções protocolares

Um protocolo de Ligação de Dados deve suportar as seguintes funções

- Sincronismo de trama
  - Delineação ou delimitação de início e fim de uma trama (*framing*)
- Gestão da ligação
  - Estabelecimento, manutenção e terminação da ligação de dados, e controlo da comunicação
- Endereçamento
  - Identificação de entidades físicas e lógicas que partilham uma ligação física, permitindo em particular a multiplexagem de dados e de informação de controlo na mesma ligação de dados ou de múltiplas ligações de dados numa ligação física
- Controlo de erros
  - Detecção e, eventualmente, recuperação de erros (tipicamente por retransmissão)
- Controlo de fluxo
  - Regulação do débito de dados do emissor
- Transparência
  - Transferência de dados independente de códigos
- Recuperação de erros de protocolo
  - Reposição do contexto da comunicação, que pode ser perdido devido a erros de protocolo

Organização dos dados – tramas

- Nos primeiros sistemas de comunicação de dados a informação a transmitir era do tipo alfa-numérico (*texto*), isto é, sequências de *caracteres* representados por palavras de um código (e.g., ASCII), situação que ainda se mantém em muitas aplicações actuais
- Embora neste contexto um caracter possa ser visto como a estrutura mais elementar, a unidade de comunicação processada por protocolos de ligação de dados é uma *trama*, existindo duas grandes alternativas no que se refere à organização de uma trama e suporte das funções protocolares
- Em protocolos *orientados ao caracter* uma trama é constituída por caracteres
  - » Neste caso, a organização da trama e os mecanismos protocolares são suportados por palavras de um código (e.g., ASCII) às quais é atribuído um significado específico e, no modo nativo de operação do protocolo, a informação transportada numa trama é constituída por caracteres alfa-numéricos (texto) representados por palavras do mesmo código
  - » Para garantir independência dos dados (transparência) em relação ao código usado pelo protocolo são necessários mecanismos adicionais
- Em protocolos *orientados ao bit* (e ao *byte*) uma trama é constituída por vários campos com funções específicas, de modo a garantir independência em relação a códigos
  - » Os dados são uma representação codificada da informação (texto, áudio, vídeo, imagens, etc.), mas o processo de codificação (de mais alto nível) é transparente a este nível protocolar
  - » Os restantes campos da trama são interpretados de acordo com as regras definidas pelo protocolo, sem relação com qualquer código

Transmissão de dados – alternativas

- A transmissão de dados organizados em tramas pode ser feita de dois modos diferentes
- A forma mais elementar, designada *transmissão assíncrona*, consiste no envio de grupos de bits (*caracteres*) de forma independente, isto é, sem impor qualquer relação temporal entre caracteres sucessivos
  - » Neste contexto, a palavra *caracter* é usada para designar uma sequência de bits com um mecanismo próprio de delineação (ou sincronização) distinto do mecanismo de delineação de tramas (de mais alto nível)
  - » Um caracter transporta um determinado número de bits de informação (tipicamente entre 5 e 8) e é delimitado por bits de início (*Start*) e fim (*Stop*)
    - – Exemplo: uma palavra do código ASCII com 7 bits de informção e um bit de paridade opcional, para além dos bits de *Start* e *Stop*
- Uma vez que em protocolos de ligação de dados a unidade de comunicação é a trama, é possível uma solução alternativa designada *transmissão síncrona*
  - » Em transmissão síncrona, os elementos que formam uma trama (sequências de bits, *bytes* ou caracteres, conforme o tipo de protocolo), são transmitidos contiguamente, sincronizados com uma referência temporal comum (sem *Start* e *Stop* bits)
  - » Neste caso o mecanismo de delineação da trama está associado ao processo de transmissão contígua de bits

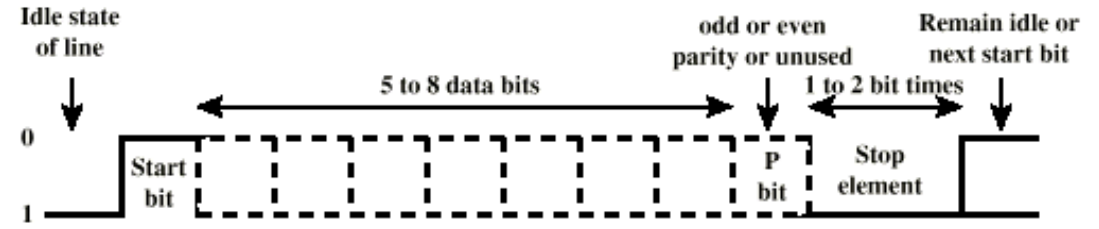
Transmissão síncrona e assíncrona

- Embora seja natural associar a transmissão assíncrona a protocolos orientados ao caracter, nestes protocolos é também possível (e mais eficiente) usar transmissão síncrona
- Os protocolos orientados ao bit (e ao *byte*), pela sua natureza e âmbito de aplicação, suportam-se normalmente em transmissão síncrona, mas é também possível, em certos casos, usar transmissão assíncrona
  - » Neste caso, os bits de informação transportados em cada caracter (termo usado na acepção da transmissão assíncrona) não são interpretados pelo protocolo como formando uma palavra de um determinado código (transparência)
- Em qualquer dos modos de transmissão (assíncrona ou síncrona) é necessário garantir sincronismo ao nível do bit, entre emissor e receptor, permitindo assim ao receptor recuperar a sequência de bits transmitidos
  - » Idealmente, o receptor deve utilizar um sinal de relógio "sincronizado" em frequência e fase com os dados recebidos
  - » O grau de precisão do relógio depende da técnica de transmissão utilizada, sendo crítico em transmissão síncrona
- É igualmente requerida sincronização a nível de caracter (nos casos em que os dados estejam organizados em caracteres) e a nível de trama

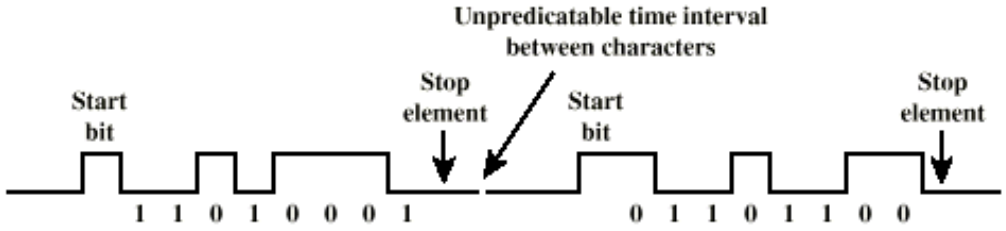
Transmissão assíncrona – caracterização

- A designação *transmissão assíncrona* traduz dois factos relacionados
  - » A ocorrência irregular (assíncrona) dos caracteres (grupos de bits) enviados de forma independente (com mecanismo próprio de delineação)
  - » A inexistência de uma referência temporal comum ao emissor e ao receptor, isto é, o relógio do receptor é independente do do emissor (embora deva ter a mesma frequência nominal)
- O sincronismo de bit realiza-se com base no sincronismo de caracter
  - » Os caracteres são enviados um a um com intervalos variáveis entre si (embora nada impeça que sejam contíguos)
  - » Na ausência de transmissão (*idle*) o nível de sinal mantém-se constante (1 lógico)
  - » Cada caracter é iniciado com um *Start* bit com o valor lógico 0
  - » A transição 1→0 produzida pelo *Start* bit indica o início do caracter (sincronismo de caracter) e fornece a referência de fase para o relógio do receptor
  - » Os bits subsequentes são recuperados por amostragem a partir do *Start* bit
  - » O caracter é terminado com um a dois *Stop* bits (valor lógico 1) para permitir distinguir o próximo *Start* bit, mesmo que não haja intervalo *idle* entre caracteres (novo caracter imediatamente disponível após transmissão do anterior)

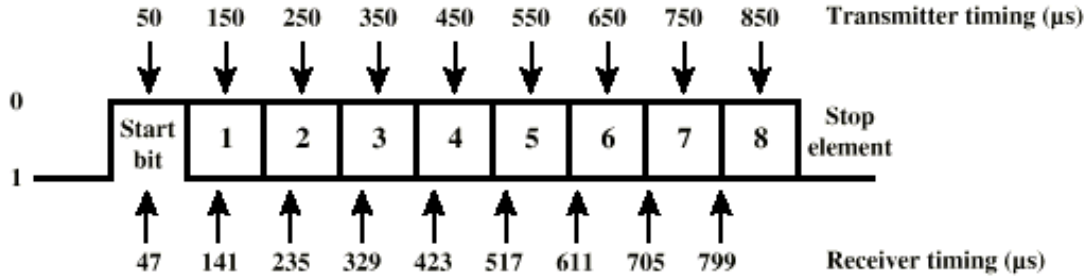
Transmissão assíncrona – exemplo



(a) Character format



(b) 8-bit asynchronous character stream



(c) Effect of timing error

Transmissão assíncrona – análise

- O método não requer elevada precisão na frequência de relógio do receptor
  - » Basta garantir o sincronismo de bit durante um caracter, pois o processo de sincronização recomeça com o início do caracter seguinte (*Start* bit)
  - » Para garantir o sincronismo de bit durante um caracter é necessário que o relógio do receptor esteja sincronizado em fase com o *Start* bit, o que não é possível garantir, dado o facto de o relógio ser gerado independentemente dos dados
    - – O problema da sincronização de fase pode ser minimizado usando um relógio com uma frequência múltipla da frequência nominal (e.g., 8 ou 16 vezes) e um processo de contagem de ciclos deste relógio para determinar os instantes de amostragem

- Propriedades
  - » Método simples e económico, mas ineficiente
    - – *Start* e *Stop* bits introduzem um *overhead* de 2 a 3 bits por caracter (que tem tipicamente entre 5 e 8 bits de informação)
    - – Transmissão *idle* entre caracteres impede a partilha da linha por vários terminais
  - » A precisão do método não é adequada para transmissão de blocos de dados (tramas) com número elevado de bits
    - – O método é apenas aceitável para transmissão a baixa velocidade e a distâncias curtas (devido à reduzida imunidade ao ruído e à elevada susceptibilidade a erros de amostragem)

Transmissão síncrona

» A *transmissão síncrona* exige uma elevada precisão (em frequência e fase) do relógio do receptor relativamente aos dados, pois é utilizada em transmissão de unidades de dados de comprimento variável (tramas) que podem ser constituídas por um número muito elevado de bits

– O relógio usado pelo receptor pode ser enviado numa linha separada, o que para além de consumir recursos está sujeito aos mesmos problemas que afectam o sinal de dados (ruído, distorção de amplitude e de fase), pelo que apenas se justifica em transmissões a pequenas distâncias

– Mais usualmente, o relógio do receptor é extraído do próprio sinal codificado ou a partir de uma portadora enviada com o sinal

» A transmissão síncrona é mais eficiente e flexível que a transmissão assíncrona podendo ser usada a muito alta velocidade e a grande distância

Sincronismo de trama

- A comunicação de dados é baseda na troca de unidades de dados (tramas), com uma estrutura (formato) conhecida
  - » Em geral as tramas têm comprimento variável, até um valor máximo predefinido (e que depende de vários factores)
  - » Em ATM as unidades básicas de comunicação têm comprimento fixo (53 octetos) e designam-se células
- Esta estrutura deve ser reconhecida pelo receptor na sequência de bits recebidos, o que requer a identificação dos limites das tramas
- O sincronismo ou delineação de tramas (*framing*) é o processo de definir e localizar os limites (início e fim) de uma trama numa sequência de bits
- Este processo não deve ser confundido com mecanismos de sincronização de tramas físicas, usados em sistemas de multiplexagem temporal para constituir múltiplos canais no mesmo suporte físico (embora possam ser usados métodos baseados em princípios semelhantes)
  - » Exemplos são os sistemas digitais de multiplexagem temporal plesiócrona (PDH) e síncrona (SONET/SDH)

Métodos de sincronismo de trama

- O sincronismo de trama é realizado de formas diferentes conforme as tramas tenham comprimento fixo (e conhecido) ou variável
- Para tramas de comprimento fixo (e.g., uma trama física SONET/SDH ou uma célula ATM), é apenas necessário identificar o início da trama e adicionar o respectivo comprimento para localizar o fim da trama
  - » Para o efeito podem ser usados padrões periódicos de bits ou pode ser introduzida correlação em sequências de bits que ocorrem periodicamente (um exemplo é o de células ATM em que o quinto octeto do cabeçalho – usado para detecção e correcção de erros – está correlacionado com os primeiros quatro octetos)
- Para tramas de comprimento variável, podem ser usados caracteres de sincronismo ou padrões especiais de bits para identificar o início de uma trama, enquanto a identificação do fim de uma trama se pode basear em métodos explícitos ou implícitos (e.g., caracteres ou padrões especiais de bits, inclusão de um campo de comprimento no cabeçalho, ou um acontecimento que possa ser associado ao fim da trama)

Técnicas de stuffing

- Nos primeiros protocolos de ligação de dados, o sincronismo de trama (assim como outras funções protocolares) realizava-se com recurso a caracteres (palavras) especiais do mesmo código usado para representar sequências de caracteres alfa-numéricos (*texto*) – um exemplo popular é o código ASCII
  - » Este processo, por si, não garante transparência, uma vez que os dados transportados numa trama não podem conter sequências arbitrárias de bits
  - » Para se conseguir transparência é necessário recorrer a sequências especiais de caracteres que usam um caracter de escape (*character stuffing*)
- Os protocolos orientados ao bit foram desenvolvidos para ultrapassar as limitações dos protocolos orientados ao caracter e, em particular, permitir independência relativamente a códigos (transparência)
  - » Em ligações ponto-a-ponto as tramas são tipicamente delimitadas por uma sequência especial de bits (*flag*) e a transparência consegue-se com uma técnica de *bit stuffing*
  - » Uma variante designada *byte stuffing* pode ser usada quando as tramas são organizadas como uma sequência de octetos (*bytes*) – esta técnica, usada em protocolos orientados ao *byte*, não deve ser confundida com a técnica de *character stuffing* usada nos protocolos orientados ao caracter
- Em redes locais (LANs) e em ATM usam-se mecanismos de delimitação não baseados em *stuffing*

Protocolos orientados ao caracter

- Em protocolos orientados ao caracter, uma trama começa com um ou mais caracteres de sincronismo – o caracter ASCII usado para este fim é SYN (*Synchronous Idle*) e tem o valor 0x16
- Outros caracteres ASCII são usados na organização da trama
  - » SOH (*Start of Heading*) precede o cabeçalho (se existir)
  - » STX (*Start of Text*) indica que se segue um campo de dados (*texto*)
  - » ETX (*End of Text*) ou ETB (*End of Transmission Block*) indicam o fim do campo de dados iniciado com STX – quando uma mensagem é fragmentada para transmissão em múltiplas tramas, ETB é usado em tramas intermédias e ETX na última trama
- Com este método básico só é possível transmitir dados codificados com recurso ao mesmo código usado pelo protocolo (este era o objectivo original em aplicações baseadas em caracteres alfa-numéricos)
- A transparência consegue-se com recurso a um caracter de escape – DLE (*Data Link Escape*) em ASCII – e delimitando o campo de dados (transparente) com DLE STX e DLE ETX (ou DLE ETB)
  - » Se o caracter DLE existir no campo de dados deve ser substituído pela sequência DLE DLE (*character stuffing*), realizando-se no receptor a remoção do DLE inserido
- Um exemplo popular dum protocolo orientado ao caracter é o Bisync (*Binary Synchronous*)

Character stuffing

- Formato normal duma trama
  - » Usado para transportar caracteres codificados com o mesmo código usado pelo protocolo (e.g., ASCII)

SYN	SYN	SOH	Cabeçalho	STX	Dados (texto)	ETX	BCC

- Formato da trama modificado
  - » Usado para transmissão transparente de dados (independente do código)
  - » Se o caracter DLE ocorrer no campo de dados deve ser substituído por DLE DLE
  - » Outras sequências de escape (iniciadas com DLE) podem ser usadas com outros objectivos

SYN	SYN	SOH	Cabeçalho	DLE	STX	Dados transparentes	DLE	ETX	BCC

BCC – *Binary Check Character* (para detecção de erros)

Protocolos orientados ao bit

- Em protocolos orientados ao bit, uma trama é interpretada como uma sequência de bits organizados em campos – tipicamente um campo de endereço, um campo de controlo, um campo de dados e um campo *Frame Check Sequence* (FCS) para detecção de erros
- Uma trama é delimitada com uma sequência especial de bits (01111110) designada *flag* – uma trama pode começar e terminar com várias *flags*, mas uma única *flag* pode ser usada para terminar uma trama e começar a seguinte (se a tramas forem contíguas)
- O campo de dados pode ser constituído por uma sequência arbitrária de bits – não está condicionado pelo uso de qualquer código (transparência) e não é necessariamente um múltiplo de oito bits (*byte*), uma vez que as palavras do código usado para representar a informação podem ter qualquer comprimento
- Os restantes campos podem também ter qualquer padrão, uma vez que são codificados para suportar uma multiplicidade de funções (são tipicamente constituídos por um número inteiro de octetos, antes de *stuffing*)
- Para se conseguir transparência total é necessário impedir que a sequência 01111110 ocorra no interior da trama (em todos os campos com excepção das *flags* reais), para que não seja interpretada pelo receptor como uma *flag* verdadeira
- A técnica usada com este objectivo designa-se *bit stuffing* e consiste na inserção pelo emissor de um zero após cinco uns consecutivos (qualquer que seja o valor do bit seguinte) e na sua remoção pelo receptor
- *Bit stuffing* é usado no protocolo HDLC (*High-level Data Link Control*) e suas variantes, com excepção do PPP (*Point-to-Point Protocol*), que usa *byte stuffing*

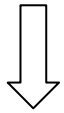
Bit stuffing

- Formato da trama

Flag	Endereço	Controlo	Dados	FCS	Flag

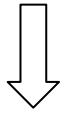
- Exemplo de *bit stuffing* / *destuffing*

Sequência original 0011111101111111111000111110100



Stuffing

Sequência transmitida (após *stuffing*) 0011111010111110111110100011111100100



Destuffing

Sequência recuperada (após *destuffing)* 0011111101111111111000111110100

Protocolos orientados ao byte

- Alguns protocolos adoptam uma estrutura de trama idêntica ao de protocolos orientados ao bit, cuja referência é o HDLC, com a restrição que os campos da trama (em particular o campo de dados) são constituídos por um número inteiro de octetos (*bytes*) – o protocolo PPP é um exemplo
- Estes protocolos orientados ao *byte* são também inerentemente transparentes, uma vez que o sincronismo de trama e outras funções protocolares são independentes do código usado para representar informação
  - » São menos flexíveis que os protocolos orientados ao bit, devido à restrição de usar palavras com um comprimento múltiplo de oito bits
- Para se conseguir transparência também é necessário evitar que qualquer octeto no interior da trama seja idêntico a uma *flag*
- Com este objectivo, pode usar-se a técnica de *bit stuffing* ou, em alternativa, usar um octeto especial de escape (ESC) que é inserido quando ocorrer um octeto idêntico a uma *flag* – esta técnica designa-se *byte stuffing*
  - » O uso deste octeto para se conseguir transparência significa que a ocorrência na trama de um octeto idêntico a ESC tem de ser também objecto de *stuffing*

Byte stuffing (PPP)

- O PPP usa a mesma estrutura de trama que o HDLC
- A *flag* é a sequência 01111110 (0x7E), como em HDLC
- No PPP o octeto de escape (ESC) é 01111101 (0x7D)
- É necessário realizar *byte stuffing* quando ocorrer no interior da trama um octeto com um padrão idêntico ao de uma *flag* ou ESC
  - » A *flag* é substituída por dois octetos – ESC seguido de 0x5E (OU exclusivo da *flag* com 0x20)
  - » ESC é também substituído por dois octetos – ESC seguido de 0x5D (OU exclusivo de ESC com 0x20)
- O receptor realiza a função inversa – quando ESC é encontrado na sequência de octetos, é removido, e o octeto seguinte é substituído após se realizar o OU exclusivo com 0x20 (obtendo-se 0x7E ou 0x7D, recuperando-se assim a *flag* ou ESC, respectivamente)

Indicação explícita do comprimento (byte count)

- Neste método (*byte count*), o comprimento da trama (de facto, do campo de dados), é incluído explicitamente no cabeçalho da trama – o campo correspondente ocupa uma posição conhecida no cabeçalho, sendo acedido a partir do início da trama, que é identificado por um dos métodos conhecidos
- Este método tem uma limitação séria – se o campo de comprimento for corrompido (devido a erros de transmissão), o fim da trama não é correctamente identificado, e a trama será recuperada com um número incorrecto de octetos
  - » O mecanismo de detecção de erros poderá detectar, em muitos casos, este tipo de erro (*framing error*), uma vez que operando sobre um número de octetos diferente do original faz a verificação com base numa sequência de bits diferente daquela que o mecanismo estaria a proteger
  - » O receptor sincroniza-se novamente na trama seguinte
  - » O método pode ser melhorado com protecção adicional do cabeçalho
- Este método foi adoptado no DDCMP (*Digital Data Communications Message Protocol*), usado na DECNET
- O campo de comprimento pode ser combinado com um delimitador de fim de trama, o que providencia um mecanismo de verificação adicional
  - » Uma trama só será aceite se o delimitador de fim aparecer na posição determinada pelo campo de comprimento e não for sinalizado qualquer erro pelo mecanismo de detecção

Outros métodos de delineação de tramas

- LANs IEEE 802.3 / Ethernet
  - » É usado o código bifásico (Manchester)
  - » A trama é iniciada com um Preâmbulo (7 *bytes*) constituído pela repetição do padrão 10101010 seguido dum *Start of Frame Delimiter* (10101011)
  - » O fim da trama é sinalizado pela ausência de transições do sinal transmitido
- LANs IEEE 802.5 (*Token Ring*)
  - » É usado um código bifásico (Manchester) diferencial
  - » A trama é delimitada por *Start of Frame Delimiter* e *End of Frame Delimiter*, codificados com símbolos que violam a regra de codificação
- LANs FDDI
  - » É usado o código 4B5B
  - » São usados símbolos especiais do código para representar o Preâmbulo e um *Start Delimiter* no início da trama e um *End Delimiter* no fim
- ATM
  - » A delineação (início da célula) baseia-se na correlação entre o quinto octeto (de protecção do cabeçalho) e os restantes quatro octetos do cabeçalho

Mecanismos de controlo

- Os mecanismos de Controlo de Erros e de Fluxo suportados em protocolos de Ligação de Dados são normalmente usados em conjunto com um mecanismo de confirmação de tramas
  - » O mecanismo de Controlo de Erros requer a capacidade de Detecção de Erros

- Estes três mecanismos baseiam-se na utilização de tramas de controlo (ou de Supervisão) distintas das tramas de dados (ou de Informação)

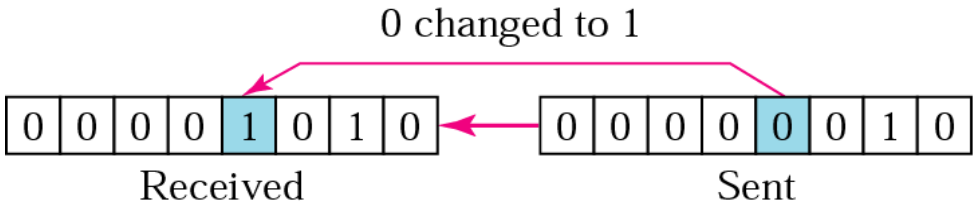
- Dada a relação entre estes mecanismos, as tramas de controlo podem realizar simultaneamente várias funções

Detecção de erros

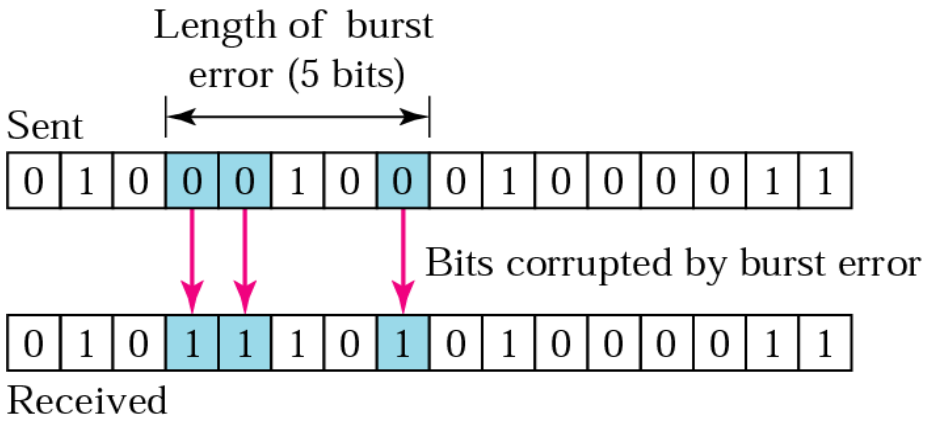
- O processo de transmissão pode corromper uma trama de diferentes maneiras
- Uma trama recebida pode diferir da trama transmitida devido a causas diversas
  - » Devido às limitações do meio (ruído, distorção, etc.), um ou mais bits podem ser alterados, podendo estes erros ocorrer aleatoriamente (de forma independente) ou em *bursts*
  - » Devido a problemas de sincronização, alguns bits podem ser adicionados ou removidos da sequência original
  - » Uma trama pode ser de tal modo corrompida que a sua estrutura seja afectada, isto é, uma trama pode ser partida em duas ou mais (ou inversamente) – isto pode ocorrer, por exemplo, quando um erro cria uma falsa *flag* ou destrói uma *flag* verdadeira
- Os mecanismos de detecção de erros têm por objectivo detectar erros resultantes das causas referidas – as tramas afectadas por erros detectados devem ser descartadas, sendo invocados mecanismos de recuperação adequados (Controlo de Erros)
- Alguns erros podem não ser detectados pelo mecanismo de detecção, pelo que tramas erradas podem ser aceites para processamento na camada de Ligação de Dados
  - » Os efeitos destes erros podem ser eventualmente detectados na camada de Ligação de Dados (e.g., um valor inválido no cabeçalho), numa camada superior ou não ser detectados
  - » Se estes erros permanecerem não detectados na camada de Ligação de Dados, as tramas correspondentes são tratadas pelo protocolo como se fossem válidas
- Os mecanismos de Controlo de Erros integram os mecanismos de detecção de erros e as acções de recuperação subsequentes

Tipos de erros de transmissão

- Erros simples
  - » Ocorrem de forma aleatória e são independentes



- Erros em *burst*
  - » Não são independentes e afectam múltiplos bits próximos
  - » O comprimento do *burst* é definido pelo primeiro e último bit errados



Princípios de detecção de erros

- As técnicas de detecção de erros usam o conceito de redundância, que consiste em adicionar *k* bits (redundantes) a *n* bits de informação, de acordo com um algoritmo conhecido do emissor e do receptor
  - » Algoritmos diferentes têm capacidade de detecção diferente
- O emissor aplica o algoritmo aos *n* bits de informação para gerar *k* bits redundantes e transmite no total $n + k$ bits
- O receptor aplica o mesmo algoritmo aos *n* bits de informação que recebe – os *k* bits redundantes gerados pelo receptor são comparados com os *k* bits correspondentes realmente recebidos
  - » Se forem diferentes, é certo que ocorreu erro na transmissão (erro simples ou múltiplo, que pode ocorrer em qualquer posição)
  - » Se forem iguais, ou não ocorreu qualquer erro ou ocorreu um erro não detectável pelo algoritmo – é desejável que a probabilidade de erros não detectáveis seja baixa

Verificação de paridade simples

- Em algoritmos de verificação de paridade simples, um bit de paridade é adicionado a *n* bits de informação, de modo que o número total de 1s seja par (paridade par) ou ímpar (paridade ímpar)
  - » Este algoritmo permite detectar erros simples ou, mais geralmente, qualquer número ímpar de erros num bloco de *n +1* bits
  - » O algoritmo não detecta um número par de erros que ocorra no bloco
- Este método é usado em protocolos orientados ao caracter – um exemplo é o uso de caracteres do código ASCII com 7 bits de informação e um bit extra de paridade, num total de 8 bits por caracter
- Considerando a sequência seguinte de caracteres ASCII com 7 bits

  1110111 1101110 1010110 1101100 1100100

  os bits de facto enviados (assumindo paridade par por caracter) são

  11101110 11011101 10101100 11011000 11001001

Verificação de paridade bi-dimensional

- Em algoritmos de verificação de paridade bi-dimensional, um bloco de bits é suposto dividido em linhas (cada linha tipicamente representa um caracter com um bit de paridade) e uma linha redundante de bits é adicionada a todo o bloco (LRC – *Longitudinal Redundancy Check*)
- Exemplo (para o mesmo conjunto de cinco caracteres)

	1110111	0
	1101110	1
	1010110	0
	1101100	0
	1100100	1
Octeto de paridade	1000111	0

Bits de paridade

- A verificação de paridade bi-dimensional permite detectar todos os erros simples, duplos e triplos e a maior parte de erros quádruplos

Soma de paridade (checksum)

- A informação a proteger é organizada em palavras (por exemplo palavras de 8 ou 16 bits)
- Todas as palavras a transmitir são “somadas” e a soma (*checksum*) é transmitida a seguir às palavras de dados
- O receptor calcula a soma das palavras de dados recebidas e compara-a com a soma recebida
- Este método é usado, por exemplo, para proteger o cabeçalho de pacotes IP
  - » A soma é realizada em aritmética “complemento para 1” sobre palavras de 16 bits (o cabeçalho normal de um pacote IP é constituído por 10 palavras de 16 bits)
  - » A soma (*checksum*) é o “complemento para 1” do resultado da adição

Complemento para 2 e para 1 – exemplos

- Aritmética complemento para 2

Decimal	Binary
0	0000 0000
1	0000 0001
2	0000 0010
3	0000 0011
-1	1111 1111
-2	1111 1110
-3	1111 1101

- Exemplo: - 3 + 5 = 2

```
  1111 1101
  0000 0101
-----------
(1) 0000 0010
```

(o transporte na soma é descartado)

- Aritmética complemento para 1

Decimal	Binary
0	0000 0000
1	0000 0001
2	0000 0010
3	0000 0011
-0	1111 1111
-1	1111 1110
-2	1111 1101
-3	1111 1100

```
  1111 1100
  0000 0101
-----------
(1) 0000 0001
(transporte)  1
-----------
  0000 0010
```

(o transporte na soma é adicionado a LSB)

Verificação de redundância cíclica (CRC)

- Os métodos de verificação de redundância cíclica (CRC – *Cyclic Redundancy Check*) baseiam-se em divisão polinomial
- Uma unidade de dados com *n* bits de informação é representada por um polinómio M(x) de grau *n - 1* – o valor de cada bit é o coeficiente de um termo do polinómio
- O método requer um polinómio divisor C(x) de grau *k* (o número de bits redundantes a acrescentar)
  - » As propriedades de detecção do método dependem da escolha de C(x)
- Os *k* bits redundantes são gerados impondo que o polinómio P(x) que representa os *n* + *k* bits a transmitir seja múltiplo de C(x), isto é, o resultado da divisão de P(x) por C(x) deve ser zero
- O receptor divide o polinómio que representa os *n* + *k* bits recebidos por C(x) e, no caso de o resto não ser nulo, pode concluir-se que ocorreu um erro na transmissão – caso contrário, ou não ocorreu qualquer erro, ou o polinómio E(x) que representa o padrão de erro é divisível por C(x)

Propriedades de detecção de códigos CRC

- Códigos CRC detectam
  - » Todos os erros simples, desde que os termos x^k e x^0 de C (x) tenham coeficientes não nulos
  - » Todos os erros duplos, se C(x) contiver um factor com pelo menos três termos
  - » Qualquer número ímpar de erros, se C(x) contiver o factor (x + 1)
  - » Qualquer *burst* de erros com comprimento inferior ou igual a *k* bits e a maior parte de *bursts* de erros com comprimento superior a *k* bits, se C(x) tiver grau *k*
- Exemplos de polinómios CRC

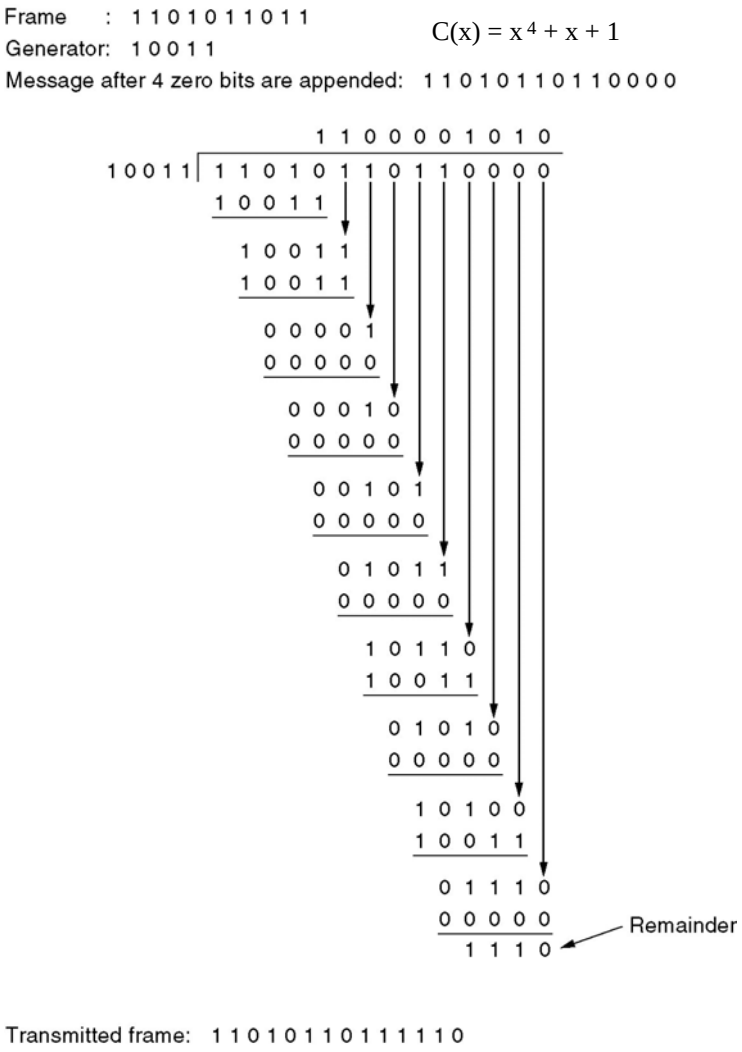
Name	Polynomial	Application
CRC-8	$x^8 + x^2 + x + 1$	**ATM header**
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$	**ATM AAL**
ITU-16	$x^{16} + x^{12} + x^5 + 1$	**HDLC**
ITU-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	**LANs**

Divisão polinomial

» A divisão polinomial é realizada em aritmética módulo 2
  – A adição e a subtração são realizadas bit a bit, sem transporte – são idênticas e equivalentes à operação XOR bit a bit

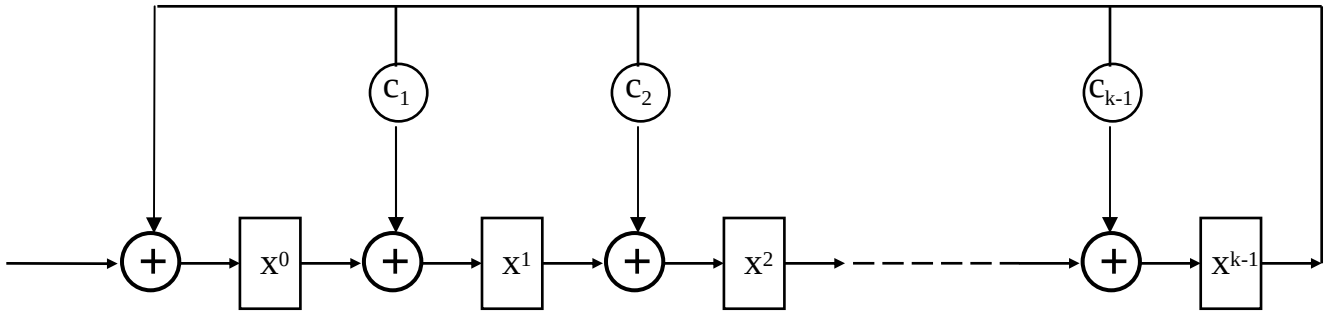
» No exemplo de geração de CRC, começa-se por acrescentar aos coeficientes do polinómio M(x), que representa os dados, um número de zeros igual ao grau *k* do polinómio gerador C(x)
  – Tal corresponde a multiplicar M(x) por x^k
» O resto da divisão terá no máximo grau *k-1*, a que correspondem *k* coeficientes
  – O resto da divisão polinomial representa o CRC gerado, que ocupará, para efeito de transmissão, as *k* posições inicialmente preenchidas com zeros
» As subtracções efectuadas nos passos sucessivos da divisão polinomial são realizadas por meio da operação XOR, bit a bit

Exemplo de geração de CRC

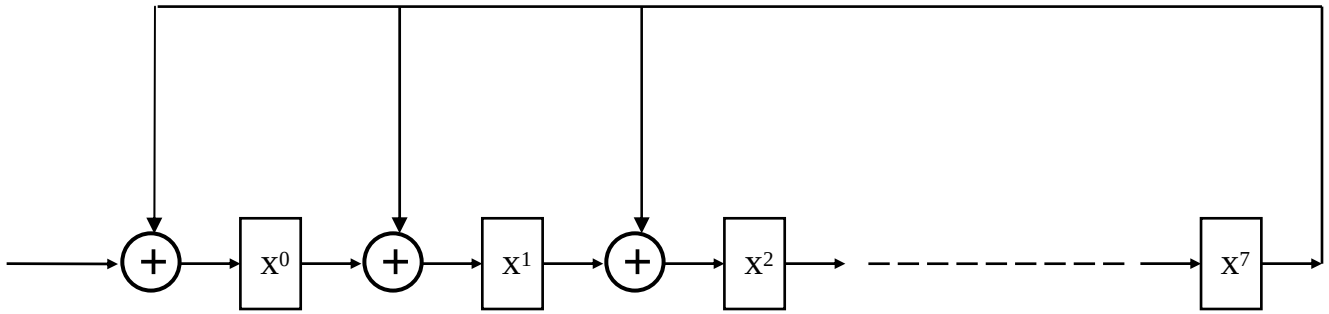


Gerador de CRC – exemplos

» Circuito genérico (simplificado), admitindo que os coeficientes c_k e c_0 do polinómio C(x) não são nulos



» Exemplo para o polinómio gerador $C(x) = x^8 + x^2 + x + 1$ (CRC-8 / ATM *header*)



Controlo de Erros – objectivos

- Em geral, os mecanismos de Controlo de Erros na camada de Ligação de Dados lidam com tramas perdidas ou descartadas, o que pode ocorrer devido a outras causas para além de erros de transmissão
  - » A interrupção do meio físico pode originar a perda de uma ou mais tramas
  - » Uma trama válida pode ser descartada pelo receptor devido a *overflow* de *buffers*
  - » Uma trama válida pode ser descartada pelo receptor, se isso for determinado pelas regras do protocolo (por exemplo, de acordo com os mecanismos de controlo de erros ou de fluxo)
- Os mecanismos de Controlo de Erros na camada de Ligação de Dados têm como objectivo providenciar um serviço fiável, que permita recuperar tramas perdidas ou descartadas – as acções de recuperação dependem da(s) trama(s) em causa, do estado do protocolo e do mecanismo específico de recuperação adoptado
- Mesmo que a camada de Ligação de Dados providencie um serviço fiável, a camada de Rede pode perder pacotes (especialmente no caso da comutação de datagramas) ou encaminhar pacotes para um destino errado (o que se traduz na perda de pacotes nos destinos reais e em “falsos” pacotes nos destinos onde são de facto entregues)
  - » Neste caso, se for necessário providenciar um serviço fiável às Aplicações, a recuperação deverá ser realizada na camada de Transporte, isto é, extremo a extremo (e portanto num ambiente diferente do da Ligação de Dados)

Controlo de Erros – princípios básicos

- As tramas são protegidas por um código detector de erros
- A estratégia básica consiste em realizar confirmações positivas de tramas de dados correctamente recebidas, por meio de tramas de controlo, de momento genericamente designadas por ACK (*Acknowledgement*)
- A ausência (não recepção) de uma trama ACK após um intervalo de tempo predefinido (*time-out*) é interpretada pelo emissor como indício de que uma trama de dados ainda não confirmada pode não ter sido correctamente recebida e que portanto pode ter de ser retransmitida
  - As retransmissões podem originar duplicados, que devem ser detectados e eliminados
- É possível ainda usar tramas que realizam confirmação negativa (NAK), sinalizando explicitamente a necessidade de retransmissão
- O emissor deve mater temporariamente em memória uma cópia de cada trama de dados transmitida, para eventual retransmissão
  - A recepção de confirmações permite ao emissor apagar da memória as tramas sinalizadas pelo receptor como correctamente recebidas

Controlo de Fluxo – princípios básicos

- Em protocolos de Ligação de Dados, o Controlo de Fluxo é realizado por meio de tramas de controlo
  - » O receptor pode conceder ao emissor um crédito explícito (valor especificado na trama de controlo) ou implícito (valor fixo conhecido), o que permite ao emissor transmitir um determinado número de tramas de dados (relativamente a uma trama de referência) sem necessidade de novas autorizações
    - – O mais usual é o crédito ser implícito (em protocolos de Transporte, como o TCP, o crédito é explícito)
  - » O receptor pode solicitar que o emissor suspenda temporariamente o envio de tramas de dados
    - – É normalmente usado como complemento do crédito implícito, visto que no caso de mecanismos de crédito explícito é possível indicar um crédito nulo, se necessário

Necessidade de numeração de tramas

- No caso ideal de não ocorrerem erros nem ser necessário controlar o fluxo do emissor, este poderia enviar tramas a qualquer momento sem quaisquer restrições e sem necessidade de as identificar
- O mecanismo de Controlo de Erros exige que sejam identificadas as tramas de dados correctamente recebidas (e que assim são confirmadas) e as não recebidas ou recebidas com erros (e que portanto requerem retransmissão)
- O mecanismo de Controlo de Fluxo exige também um processo de referenciar tramas de dados para efeito da concessão de crédito
- Os protocolos de Ligação de Dados devem por isso providenciar um mecanismo de identificação (numeração) de tramas de dados
  - » O número de sequência **N(s)** de uma trama de dados deve ser transportado no respectivo cabeçalho
  - » Tramas que transportam confirmações devem referenciar tramas de dados enviadas em sentido oposto, por meio de um número de sequência **N(r)** no seu cabeçalho

Método de numeração de tramas

- Com o objectivo de reduzir o *overhead*, procura-se limitar o número de bits usados para numerar as tramas de dados
  - » Usando **k** bits para o efeito é possível dispor de **M** = 2^{k} identificadores diferentes (0, 1, …., M - 1) – numeração módulo M
- Os identificadores têm de ser reutilizados, visto que, sendo em número finito, se repetem ciclicamente
  - » Não é possível que tramas de dados diferentes, transmitidas e ainda não confirmadas (pendentes), tenham o mesmo número de sequência
  - » O número máximo de tramas de dados pendentes depende do módulo de numeração e da estratégia de retransmissão
- A existência de um limite superior ao número de tramas de dados pendentes constitui, em certos casos, uma restrição ao envio de novas tramas e, portanto, uma forma indirecta de Controlo de Fluxo

Modelo e parâmetros de comunicação

- As tramas são recebidas na mesma ordem com que são enviadas
  - » A ocorrência de perda(s) manifesta-se pela recepção de tramas com números de sequência não consecutivos
- Tempo de transmissão
  - » Tempo de envio de todos os bits da trama para o meio

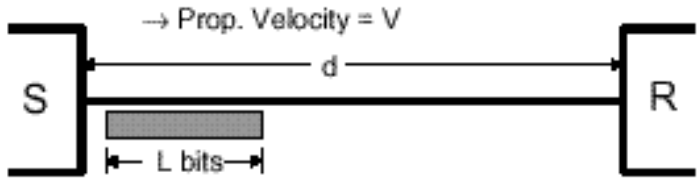
$$T_t = L/R$$

- Tempo de propagação
  - » Tempo que um bit demora a propagar-se no meio entre o emissor e o receptor

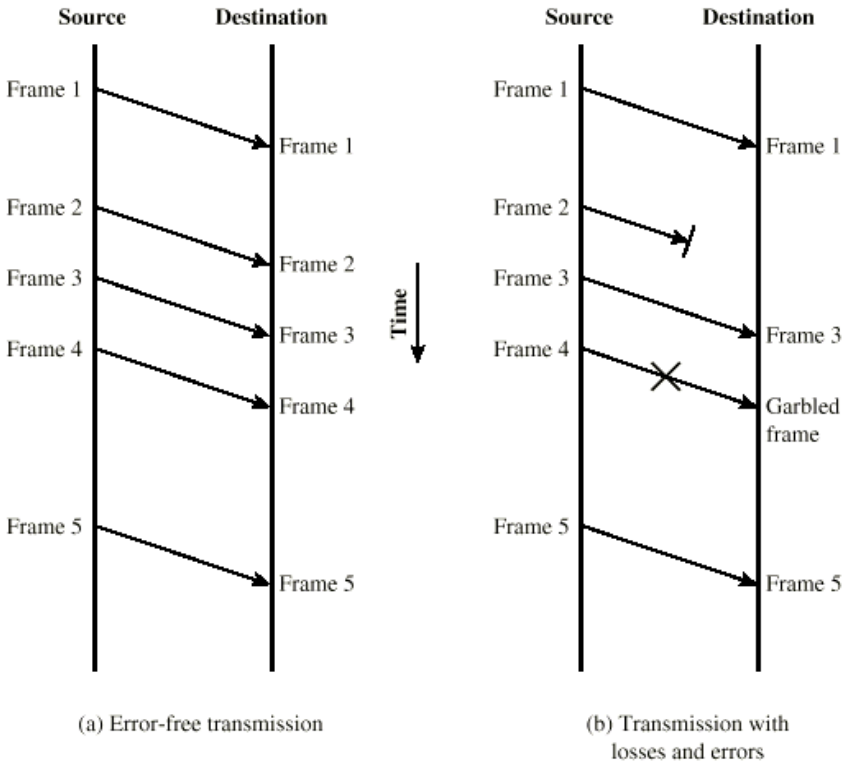
$$\tau = d/V$$

$$\boldsymbol{a = \tau / T_t}$$

- Técnicas usadas
  - » *Stop and Wait*
  - » Janela deslizante (*Sliding Window*)



➔ *R – data rate* (bit/s)



Stop and Wait – princípio de funcionamento

- Funcionamento básico sem erros
  - » O Emissor transmite uma trama de dados ou de Informação (I) e fica à espera de receber uma confirmação positiva (ACK) enviada pelo Receptor
  - » O Receptor recebe a trama de dados e responde com confirmação (ACK)
  - » Após receber a confirmação, o Emissor pode enviar uma nova trama
- Funcionamento com erros
  - » Se o Emissor não receber a confirmação até expirar um temporizador (*time-out*), envia de novo a trama
  - » Se o Receptor receber uma trama de dados com erro deve descartá-la, mas se reconhecer e aceitar o cabeçalho (como presumivelmente válido), pode enviar uma confirmação negativa (NAK), solicitando o reenvio da trama pelo Emissor (antes que expire o temporizador)
    - – Esta possibilidade não é normalmente considerada em protocolos orientados ao bit, mas pode ser explorada em protocolos orientados ao caracter (baseando-se no eventual reconhecimento de caracteres de controlo no cabeçalho)
- O *Stop and Wait* é, em primeiro lugar, um mecanismo muito elementar de confirmação de tramas de dados que pode, no entanto, impor sérias restrições à transmissão continuada de tramas (realizando assim controlo de fluxo)

Stop and Wait e Controlo de Fluxo

- O *Stop and Wait* constitui uma forma de Controlo de Fluxo, pois ACK apenas autoriza o emissor a transmitir uma nova trama de dados (concede um crédito implícito de uma unidade)
  - » O emissor suspende a transmissão até receber ACK
  - » ACK concede crédito para transmitir uma nova trama de dados
- O processo de comunicação introduz um atraso inevitável entre o envio de uma trama de dados e a recepção do ACK respectivo
  - » Neste caso o Controlo de Fluxo é indirecto e certamente indesejável
- O receptor pode introduzir deliberadamente um atraso adicional, retardando (ou até cancelando) o envio de ACK
  - » Neste caso o Controlo de Fluxo é intencional, mas um atraso excessivo no envio de ACK pode induzir um *time-out* (interpretado como ausência de confirmação) e despoletar uma retransmissão desnecessária, pelo que este mecanismo deve ser usado com prudência

Stop and Wait – tamanho das tramas

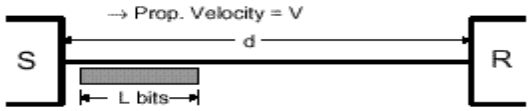
- O sistema de comunicação pode requerer fragmentação em tramas pequenas
- Vantagens de tramas de tamanho pequeno
  - » Aconselhável se existirem limitações de memória no receptor
  - » Os erros são detectados mais rapidamente
  - » Se ocorrerem erros a quantidade de dados a retransmitir é menor
  - » Evita-se que uma estação ocupe o meio por períodos longos (pode ser importante se o meio for partilhado por várias ligações)
- Desvantagens de tramas de tamanho pequeno
  - » Maior *overhead* (maior peso relativo do cabeçalho)
  - » A eficiência do *Stop and Wait* pode diminuir drasticamente se o tamanho das tramas for muito pequeno, o que é agravado com transmissão a alta velocidade e a grande distância (parâmetro $a = \tau / T_t$ muito elevado)

Desempenho de Stop and Wait – análise

» WAN ATM
- d = 1000 km
- L = 424 bit, R = 155.52 Mbit/s
- T_t = 2.7 μs
- Fibra óptica → 5 μs/km → τ = 5 ms
- a = 1852
- **S = 1 / 3705 = 0.0003**

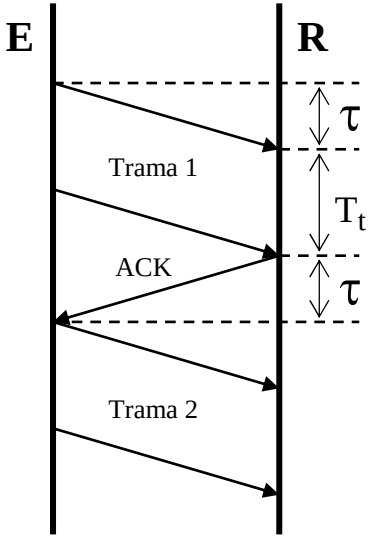
» LAN
- d = 0.1 ~ 10 km
- L = 1000 bit, R = 10 Mbit/s
- T_t = 100 μs
- Cabo coaxial → 4 μs/km → τ = 0.4 ~ 40 μs
- a = 0.004 ~ 0.4
- **S = 0.55 ~ 0.99** (e se R = 100 Mbit/s?)

» Modem sobre linha telefónica
- d = 1000 m
- L = 1000 bit, R = 28.8 kbit/s
- T_t = 34.7 ms
- UTP → 5 μs/km → τ = 5 μs
- a = $1.44*10^{-4}$
- **S ≃ 1.0**



➔ R – *data rate* (bit/s)

$$T_t = \frac{d}{V} \qquad \tau = \frac{L}{R}$$



$$a = \frac{\tau}{T_t} = \frac{d}{V} \times \frac{R}{L}$$

$$S = \frac{T_t}{\tau + T_t + \tau} = \frac{1}{1 + 2a}$$

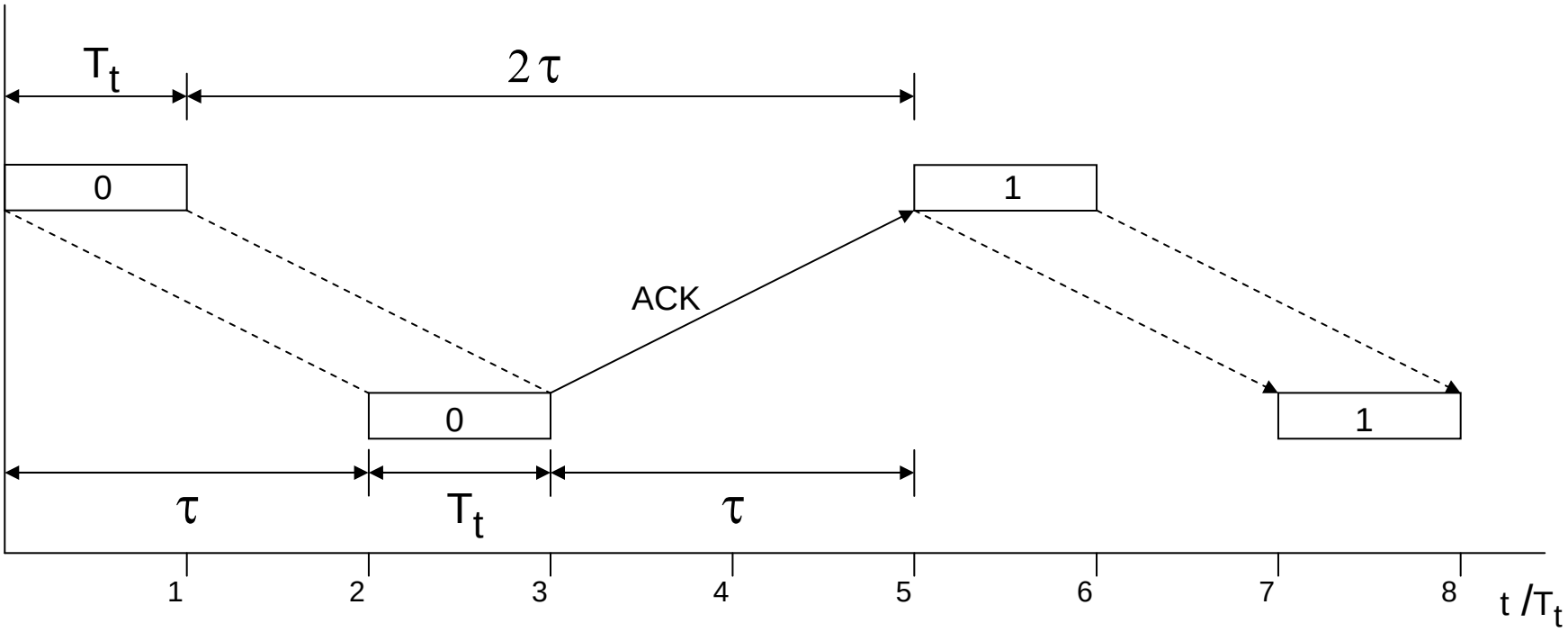
Desempenho de Stop and Wait – exemplo

$$S = \frac{T_t}{T_t + 2\,\tau} = \frac{1}{1 + 2\,a}$$

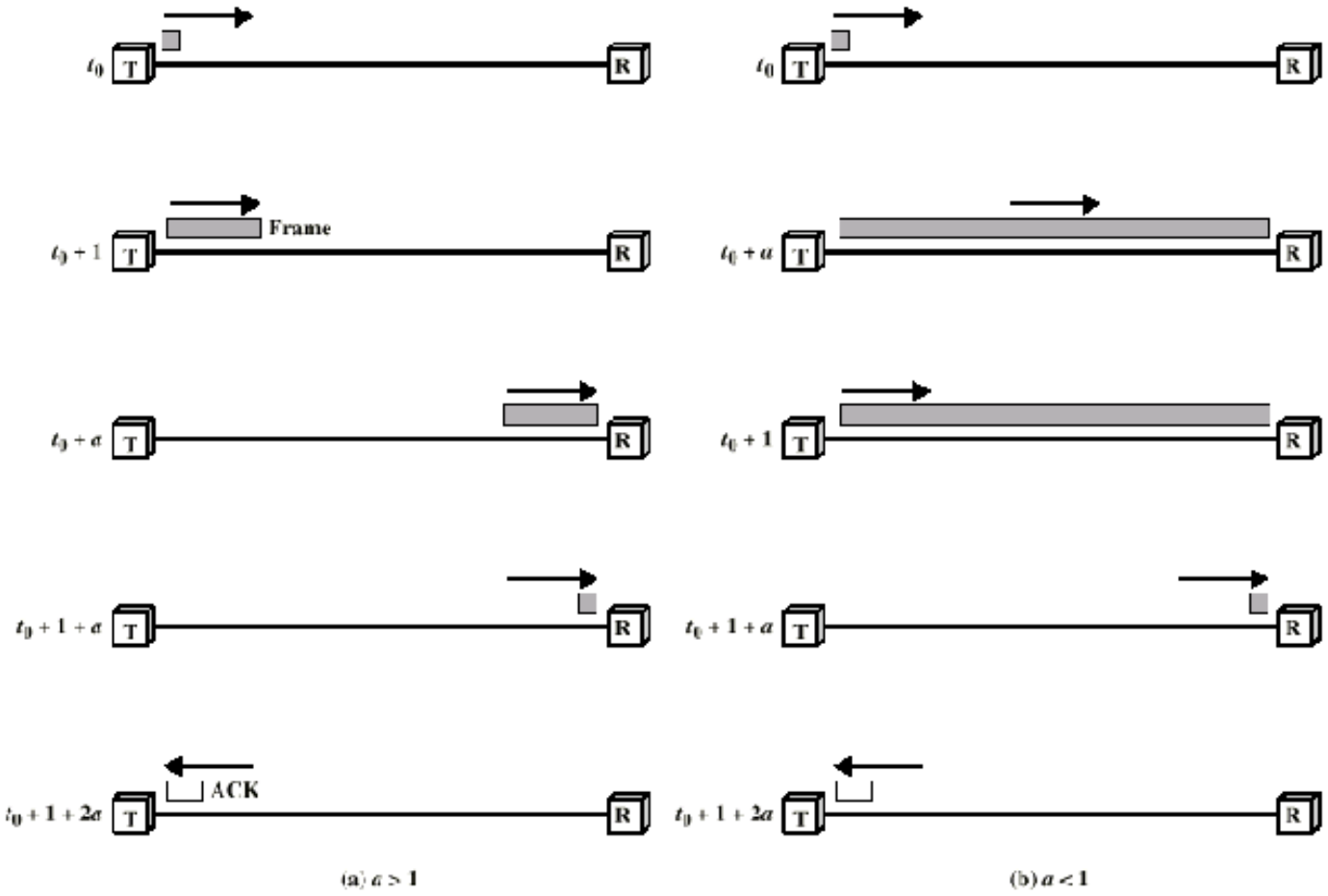
a = 2

1 + 2 a = 5

S = 0.2



Stop and Wait – utilização (eficiência)



Stop-and-Wait Link Utilization (transmission time = 1; propagation time = a)

Janela Deslizante (Sliding Window)

- Podemos generalizar o mecanismo de confirmação e permitir que o emissor transmita várias tramas de dados antes de receber qualquer confirmação (ou autorização)

- Estes protocolos são do tipo Janela Deslizante (*Sliding Window*)
  - » As tramas de dados têm de ser numeradas para ser possível ao mecanismo de confirmação referenciar tramas correcta e incorrectamente recebidas
  - » São possíveis diferentes estratégias de retransmissão no caso de erros
  - » O número máximo de tramas de dados que o emissor pode ter pendentes designa-se Janela de Transmissão (*W – Window*)
  - » O tamanho máximo possível da Janela de Transmissão (W_{max}) está relacionado com o módulo de numeração de tramas e com a estratégia de retransmissão (a analisar)
  - » O mecanismo *Stop and Wait* pode ser visto como um caso particular de *Sliding Window* com janela unitária

Sliding Window e Controlo de Fluxo

- O *Sliding Window* constitui também uma forma de Controlo de Fluxo por limitar o número de tramas que o emissor pode ter pendentes
- Permite ultrapassar as limitações de desempenho do *Stop and Wait* (devidas ao Controlo de Fluxo implícito)
  - » Na ausência de erros, e no caso de tráfego persistente, o objectivo típico será atingir uma utilização plena da ligação (100%)
    - – Se tal acontecer, o emissor não chega a ter de interromper a transmissão por falta de créditos (janela esgotada), pelo que neste caso não ocorre, de facto, Controlo de Fluxo
  - » Nem sempre é possível atingir uma utilização de 100% (com tráfego persistente) e pode ser necessário, em certos casos, realizar Controlo de Fluxo – uma forma de o fazer consiste em atrasar as confirmações (ou seja, atrasar a rotação da janela)
- Normalmente os protocolos de Ligação de Dados não dispõem de mecanismos de indicação explícita de crédito variável
  - » Por este motivo, a realização de Controlo de Fluxo pode requerer o envio de uma trama de controlo para suspender a transmissão e mais tarde de outra para reactivar a transmissão e o mecanismo normal de janela – este método designa-se *Stop and Go*

Sliding Window – mecanismo básico

- Funcionamento básico sem erros

  » O Emissor pode ter múltiplas tramas de dados pendentes (transmitidas e não confirmadas)

  – O Emissor pode enviar até *W* tramas de dados (tamanho da janela de transmissão negociada) sem receber ACK

  » O Receptor deve dispor de *buffers* para receber *W* tramas de dados

  » As tramas de dados são numeradas sequencialmente (módulo $M = 2^k$), sendo **k** o número de bits usados para codificar os **M** números de sequência

  » Por convenção ACK referencia por meio de N(r) o número de sequência N(s) da próxima trama de dados esperada na sequência correcta

  » Designa-se habitualmente *Receiver Ready* (RR) a trama de controlo usada, em condições normais, para realizar confirmação

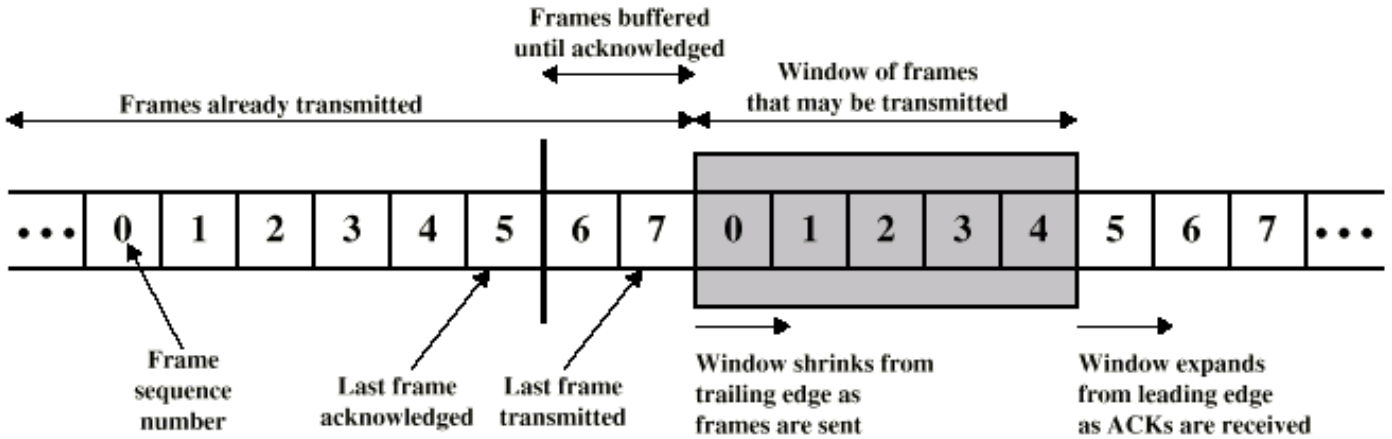
Sliding Window – extensão (piggyback)

- As tramas de dados (I) transportam igualmente um número de sequência N(r), o que permite o envio de confirmações em tramas de dados (*piggyback***)** na presença de tráfego bidireccional (extensão ao funcionamento básico)

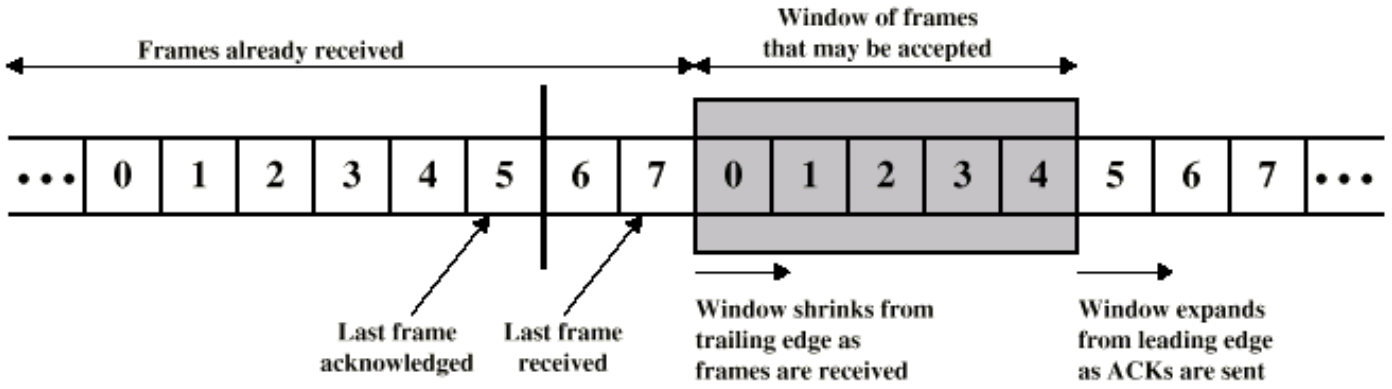
- Quando uma estação tiver de realizar a confirmação de uma ou mais tramas I recebidas procede do seguinte modo
  - » Se tiver dados para transmitir, pode evitar o envio de RR, uma vez que as tramas de dados (I) em sentido oposto realizam confirmação, por meio de N(r)
  - » Se não tiver dados para transmitir, usa RR para realizar a confirmação

- Se uma estação tiver uma trama de dados (I) para transmitir mas não tiver nova(s) trama(s) de dados para confirmar, o valor de N(r) transportado na trama I indica, de acordo com a convenção, o número da próxima trama de dados (I) que espera receber (na prática repete a confirmação de tramas anteriormente confirmadas)

Sliding Window – visão do emissor e do receptor

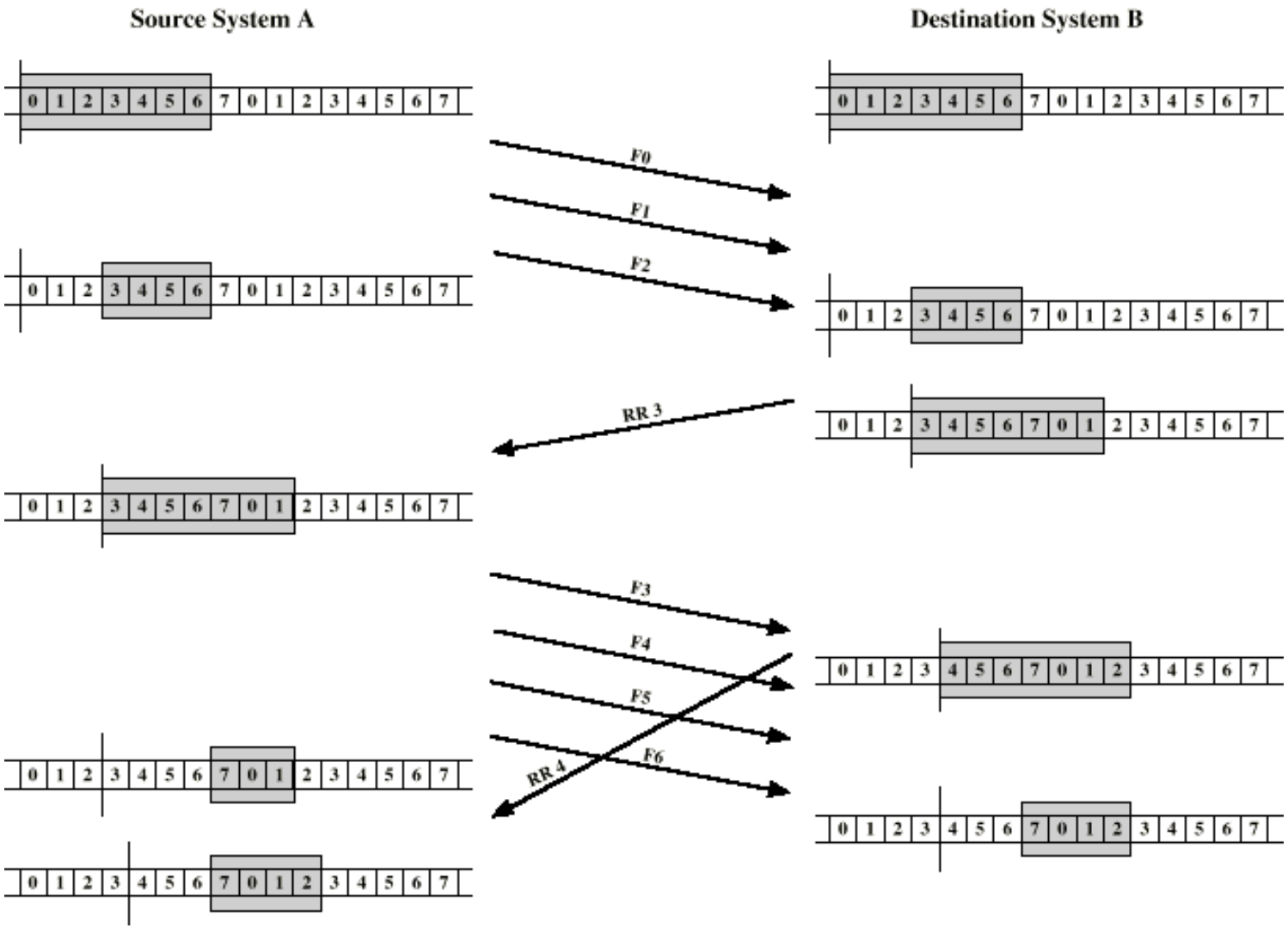


(a) Sender's perspective



(b) Receiver's perspective

Sliding Window – exemplo de funcionamento



Desempenho de Sliding Window – exemplo

$$S = \frac{W * T_t}{T_t + 2\tau} = \frac{W}{1 + 2a} \qquad (W < 1 + 2a)$$

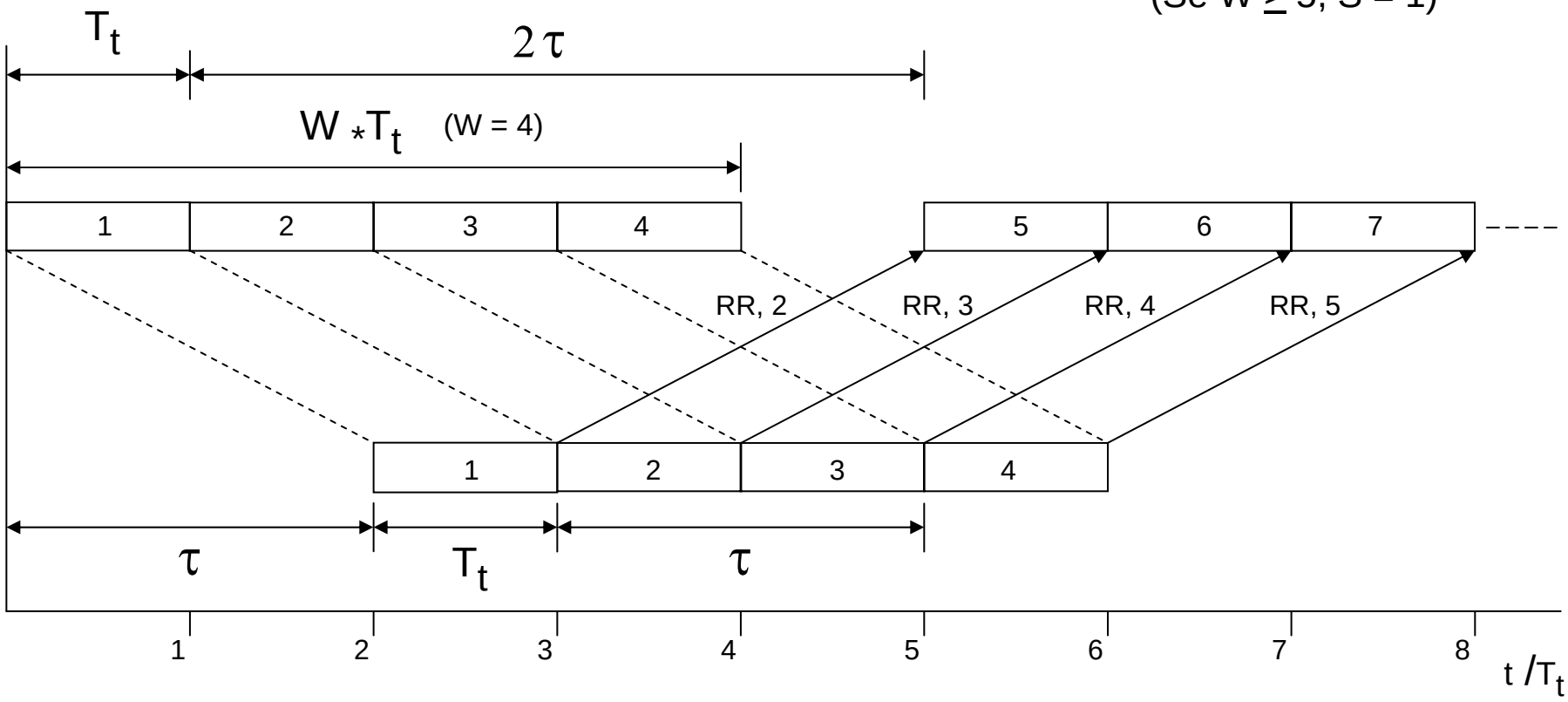
$$S = 1 \qquad (W \geq 1 + 2a)$$

a = 2

1 + 2 a = 5

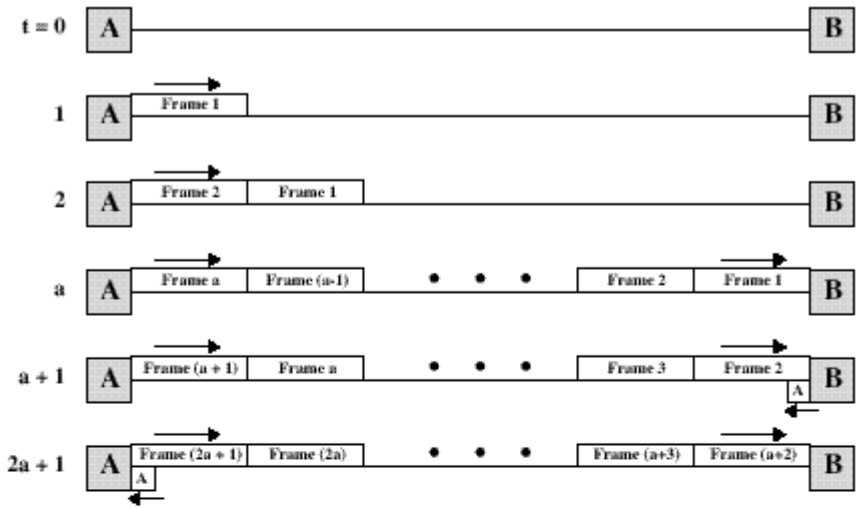
W = 4, S = 0.8

(Se W ≥ 5, S = 1)



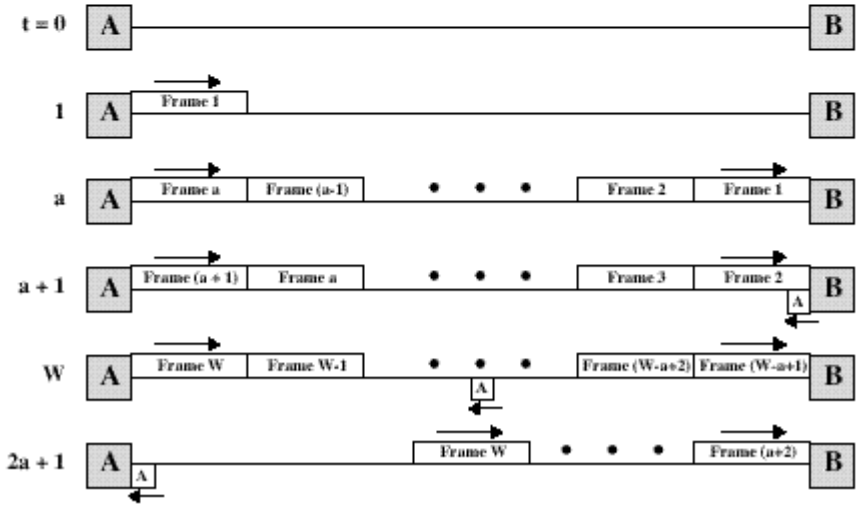
Sliding Window – utilização (eficiência)

- Se W ≥ 1+2a ➔ S = 1



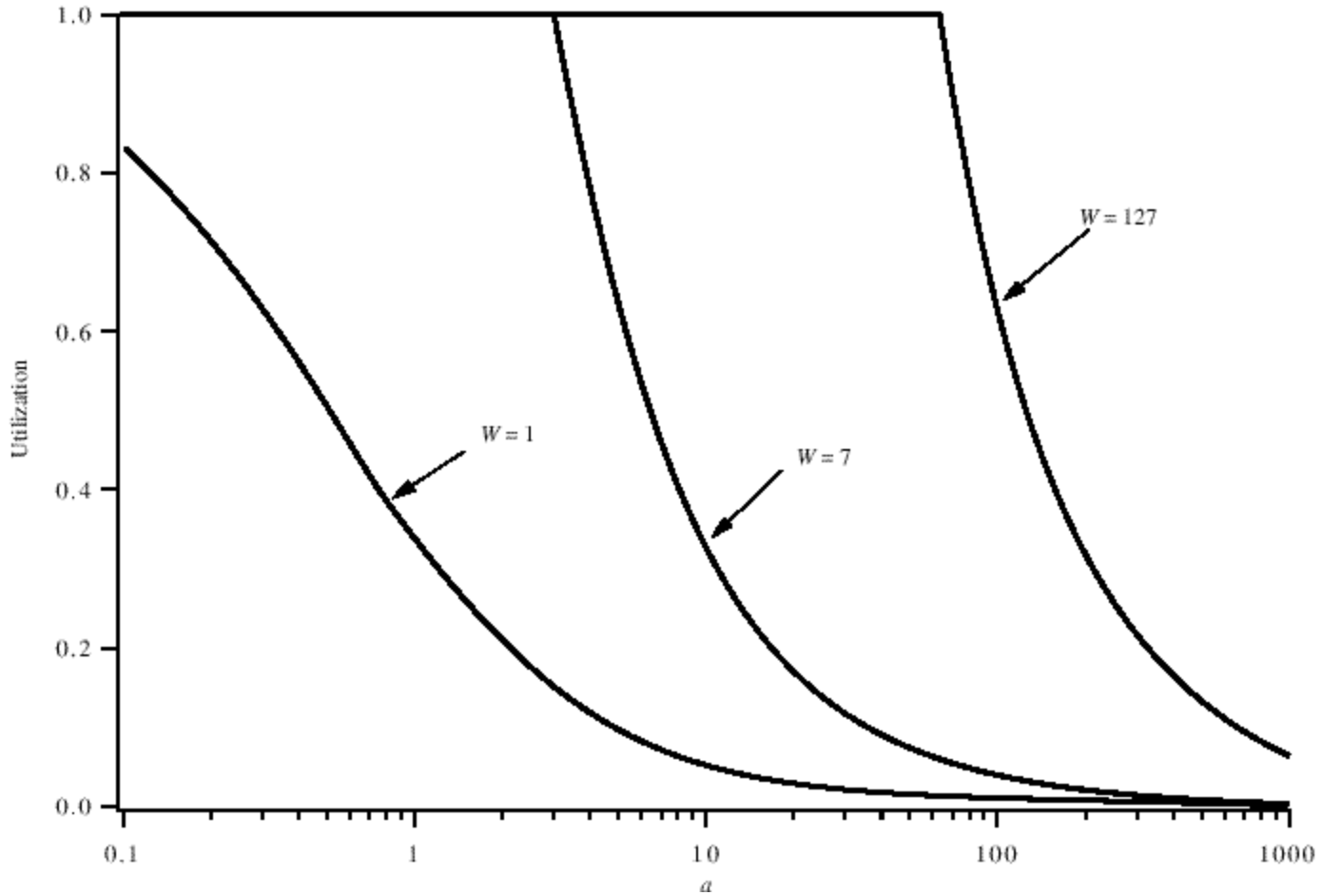
(a) W > 2a + 1

- Se W < 1+2a ➔ S = W/(1+2a)



(b) W < 2a + 1

Sliding Window – S = f (a, W)



Controlo de Fluxo Stop and Go

- O receptor pode suspender temporariamente novas transmissões (mecanismo *Stop and Go*), sem prejuízo de poder confirmar uma ou mais tramas pendentes
- É necessário uma trama de controlo, designada habitualmente *Receiver Not Ready* (RNR), para suspender temporariamente a transmissão (*Stop*)
  - » Antes de reactivar a transmissão (*Go*) pode eventualmente descartar alguma(s) trama(s) recebida(s)
- A reactivação da transmissão será feita recorrendo a uma trama *Receiver Ready* (RR), caso o receptor não tenha descartado qualquer trama de dados após o envio de RNR
  - » No caso de descarte, procede como se tivesse ocorrido perda de trama(s) de dados (situação a discutir no contexto do Controlo de Erros)
- RNR realiza igualmente a função de confirmação (tal como RR, em condições normais), uma vez que transporta um número de sequência N(r)

Controlo de Erros – técnicas ARQ

- Numa Ligação de Dados as tramas são recebidas pela mesma ordem com que são enviadas, mas estão sujeitas a erros, que podem originar a perda ou o descarte de tramas (quer pelo mecanismo de detecção de erros quer pelos próprios mecanismos protocolares)
- A técnica mais usual de recuperação baseia-se na retransmissão de tramas, designando-se habitualmente *Automatic Repeat Request* (ARQ)
  - É baseada em confirmações positivas e negativas
  - Tramas de dados não aceites pelo receptor são descartadas e terão de ser retransmitidas
- A recuperação pode ser iniciada quer pelo emissor quer pelo receptor
  - Pelo emissor, após *time-out* (ausência de confirmação positiva)
  - Pelo receptor, mediante envio de confirmação negativa, normalmente designada *Reject* (REJ) – não é estritamente necessário, mas é vantajoso, pois pode permitir iniciar a recuperação antes da ocorrência de *time-out* no emissor
- Versões ARQ mais usadas
  - *Stop and Wait ARQ*
  - *Go-back-N / Continuous Reject ARQ* (retransmissão contínua)
  - *Selective Repeat / Selective Reject ARQ* (retransmissão selectiva)

Variáveis de estado e numeração de tramas

» Para facilitar a operação dos protocolos ARQ, cada estação deve manter duas variáveis de estado V(s) e V(r), com o objectivo de reter o valor corrente dos números de sequência N(s) e N(r), respectivamente

- **V(s)** – número de sequência da próxima trama de dados a ser transmitida pelo emissor
- **V(r)** – número de sequência da próxima trama de dados esperada pelo receptor

» O valor das variáveis de estado é colocado a 0 após o estabelecimento da ligação de dados

» A actualização das variáveis de estado deve ser feita de modo que

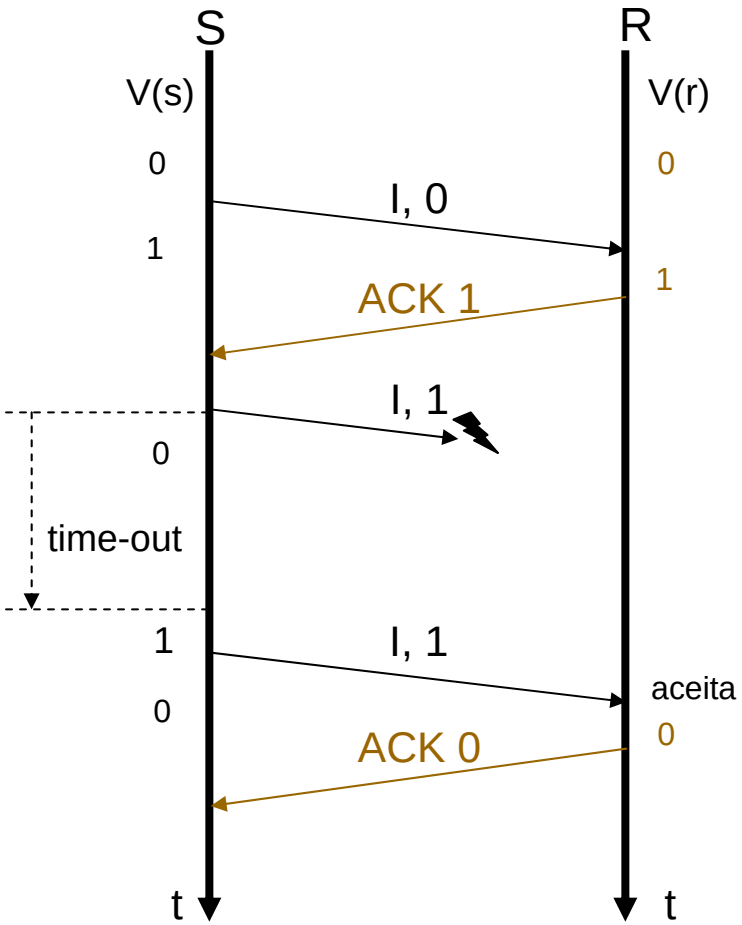
- O campo N(s) possa ser preenchido com o valor corrente de V(s)
- O campo N(r) possa ser preenchido com o valor corrente de V(r)

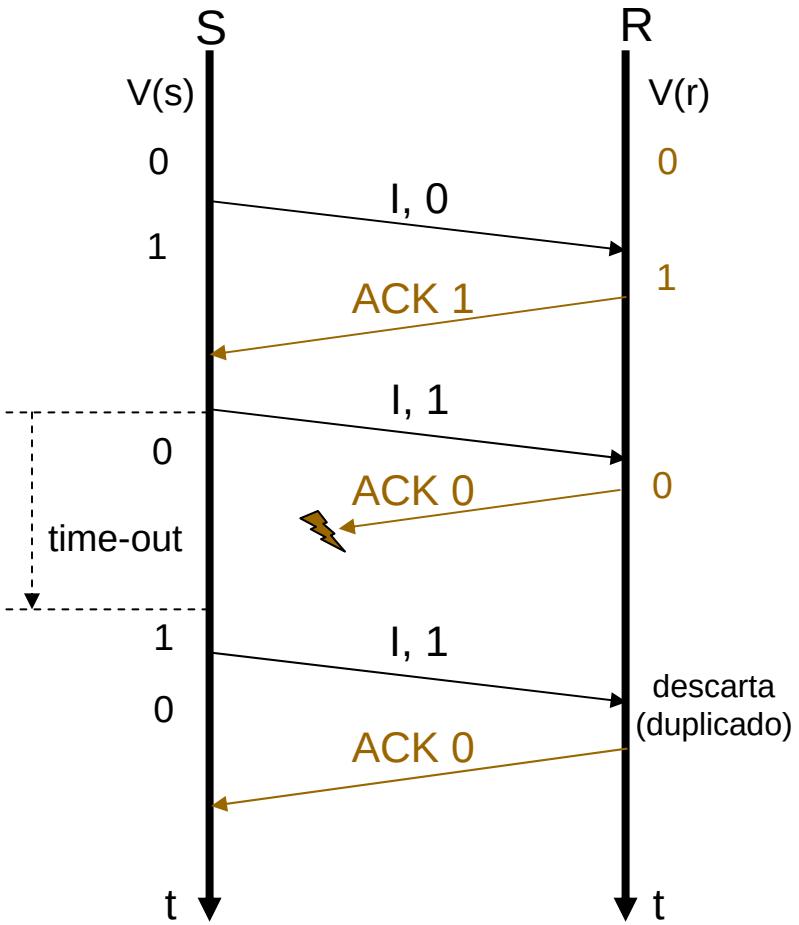
Stop and Wait – numeração de tramas

- Aparentemente não seria necessário numerar as tramas de dados (pois no máximo apenas uma poderá estar pendente)

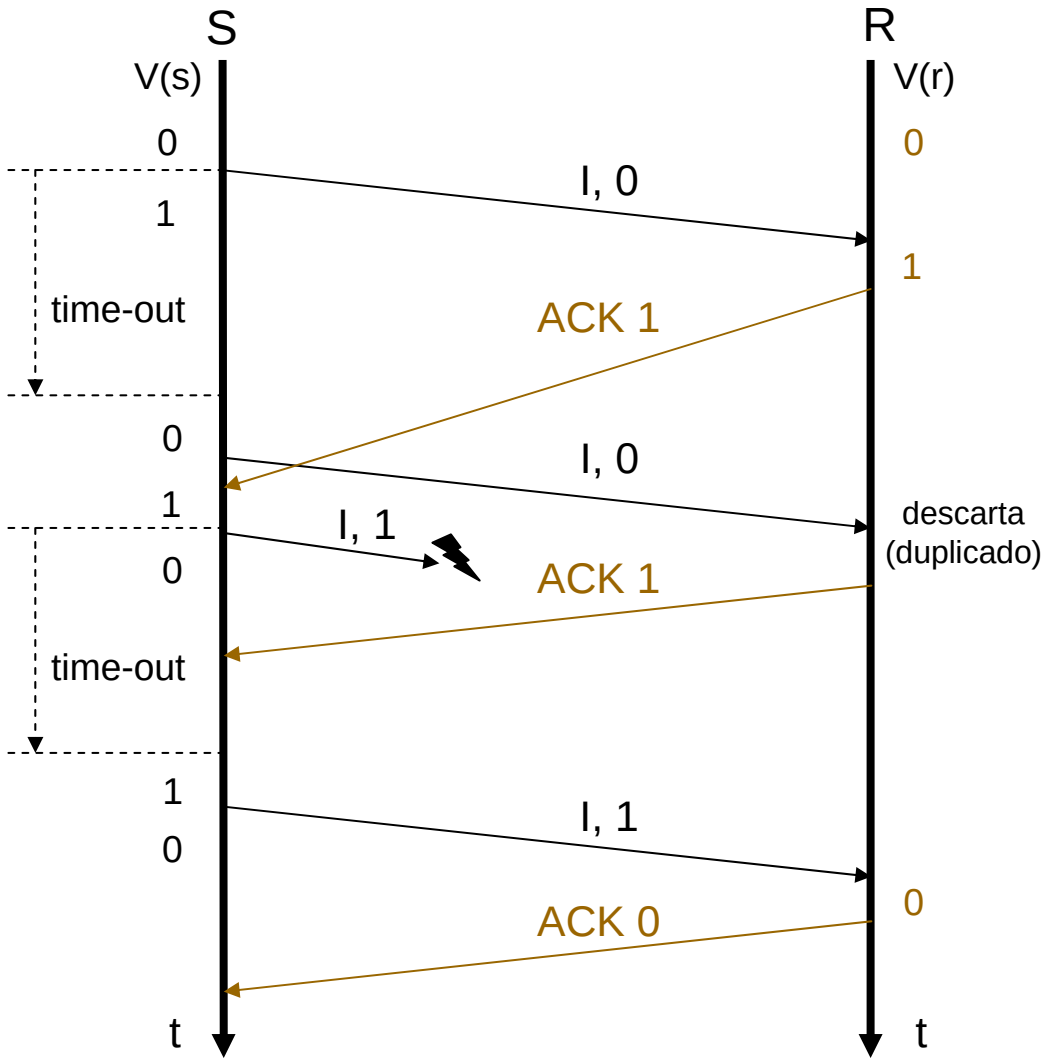
- A numeração de tramas de dados (e também das confirmações) é efectivamente necessária devido à possibilidade de perda de tramas (de dados e confirmações) ou atrasos excessivos de confirmações
  - » Se a confirmação se perder, o Emissor repete a trama de dados que, se não fosse numerada, seria interpretada pelo Receptor como uma nova trama
  - » Se a confirmação for recebida após ter expirado o temporizador, o Emissor perde a relação entre tramas enviadas e confirmações recebidas, caso as confirmações não sejam numeradas
  - » Estas ambiguidades podem ser removidas se as tramas forem numeradas
  - » Nestes casos, como há no máximo uma trama pendente de confirmação, basta uma numeração módulo 2 – isto é, os números de sequência usados são 0 e 1

Stop and Wait – perda de trama I ou ACK



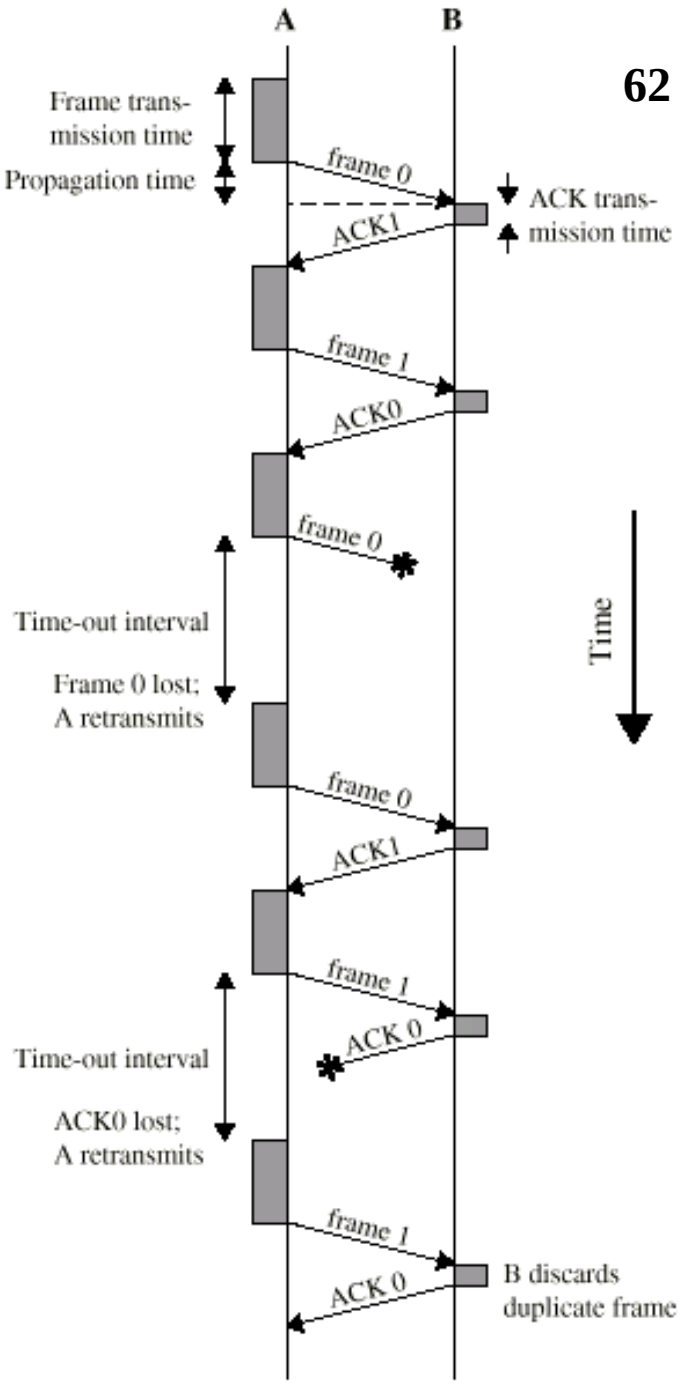


Stop and Wait – ACK atrasado e perda de trama I



Stop and Wait ARQ

- Baseado no *Stop and Wait*
  - » O Emissor transmite uma única trama e espera até receber ACK (ou *time-out*)
- Erros possíveis
  - » Trama de dados com erros
    - – O Receptor elimina a trama (e não envia ACK)
    - – O Emissor retransmite a trama depois de *time-out*
    - – Em protocolos orientados ao caracter a recepção duma trama de dados com erros (mas com cabeçalho reconhecido) pode ser sinalizada com NAK, o que desencadeia a retransmissão
  - » ACK com erros (ou com atraso excessivo)
    - – O Emissor não recebe ACK antes de *time-out*
    - – O Emissor retransmite a trama depois de *time-out*
    - – O Receptor recebe dados em duplicado
    - – O Receptor só pode detectar e eliminar duplicados se as tramas de dados forem numeradas
      - Receptor elimina duplicado e envia ACK
      - ACK0 e ACK1 usados em alternância
- Simples mas ineficiente



Eficiência de Stop and Wait ARQ

» P_e – probabilidade de uma trama ser afectada por erro (depende do *Bit Error Ratio* no canal)

» Prob [k] – Probabilidade de serem necessárias *k* tentativas para transmitir uma trama

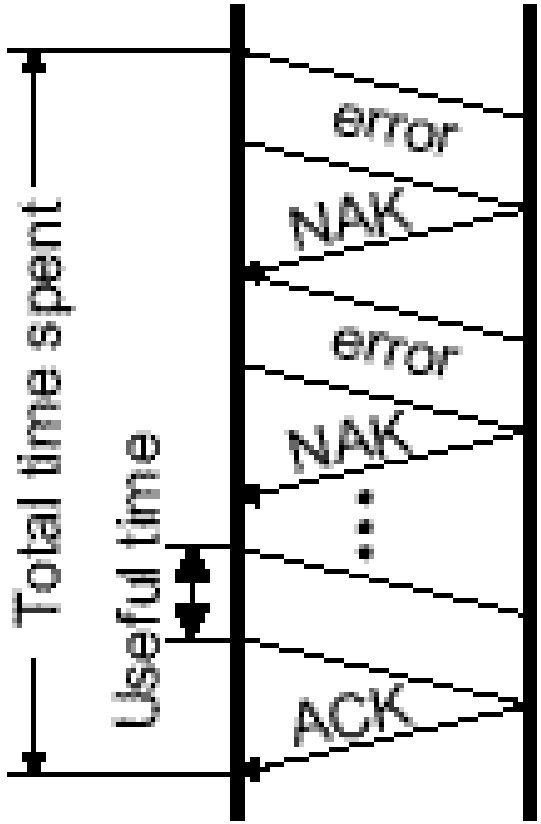
$$\text{Prob}\left[\text{k}\right] = P_e^{\,k-1}(1-P_e)$$

» N_r – número médio de tentativas para transmitir uma trama com sucesso

$$N_r = \sum_{k=1}^{+\infty} k * \text{Prob}\,[k] = \frac{1}{1-P_e}$$

» Eficiência do *Stop and Wait* ARQ

$$S = \frac{tempo\_útil}{tempo\_total} = \frac{T_t}{N_r(T_t + 2*\tau)} = \frac{1}{N_r(1+2a)} = \frac{1-P_e}{1+2a}$$



Go-Back-N ARQ – princípio de operação

- Baseado em *Sliding Window*
- O emissor transmite tramas de dados (I) numeradas sequencialmente, dentro da janela
- O receptor apenas aceita tramas de dados na sequência esperada (isto é, a janela do receptor tem tamanho unitário) – tramas fora de sequência podem ser duplicados (devido à retransmissão de tramas anteriores) ou novas tramas
- Tramas de dados aceites pelo receptor são confirmadas com *Receiver Ready* ou por *piggyback* (ou, excepcionalmente, com *Receiver Not Ready* no caso do receptor estar a realizar controlo de fluxo *Stop and Go*)
- Várias tramas de dados podem ser confirmadas em bloco
- Tamanho máximo da Janela de transmissão $W_{max} = 2^k - 1$

Go-Back-N ARQ – recuperação de erros

- Em protocolos orientados ao bit, tramas com erro são descartadas pelo receptor sem qualquer acção (assumindo, como é usual, um único campo de protecção de toda a trama); se o erro ocorrer numa trama de dados, só as tramas de dados seguintes recebidas fora da sequência esperada dão origem a procedimentos de recuperação (são rejeitadas, mesmo que recebidas sem erro)
  - » O Receptor
    - Envia confirmação negativa (*Reject* – REJ) após recepção da primeira trama recebida fora de sequência; REJ (re)confirma tramas anteriores correctamente recebidas
    - Ignora (descarta) todas as tramas de dados fora de sequência até receber a trama de dados em falta, mas não repete o envio de REJ (apenas pode existir um REJ pendente)
  - » O Emissor volta atrás na sequência de transmissão (*Go-Back-N*)
    - (Re)transmite a trama de dados indicada em REJ e as tramas de dados subsequentes
- Em protocolos orientados ao caracter o receptor pode enviar REJ (NAK) se uma trama de dados com erros for recebida na sequência correcta e tiver estrutura reconhecida (não consideramos este caso na análise)
- Os procedimentos de recuperação dependem do tipo de trama afectada pelo(s) erro(s) (I, RR, REJ)

Go-Back-N ARQ – transmissão de tramas de dados

- Quando o emissor tiver uma trama de dados pronta a transmitir e não existirem condições restritivas, adopta os seguintes procedimentos
  - » Coloca o valor corrente de V(s) no campo N(s) da trama
  - » Incrementa (módulo M) o valor de V(s), apontando assim para a próxima trama de dados a transmitir
  - » Activa um temporizador (se inactivo)

- O emissor suspende a transmissão de novas tramas de dados se ocorrer uma das seguintes condições
  - » Número de tramas de dados pendentes igual ao tamanho da janela
  - » Recepção de RNR (pode eventualmente admitir-se o envio apenas da trama referenciada por N(r) para manter o processo activo – na prática equivale a reduzir momentaneamente o tamanho da janela a um)
  - » Ocorrência de *time-out* (inicia processo especial de recuperação)

Go-Back-N ARQ – time-out

- A ocorrência de *time-out* pode ter várias causas
  - » Perda da última trama de dados de uma sequência (impede o envio de REJ)
  - » Perda da confirmação (RR) da última trama de dados de uma sequência
    - – A perda de uma confirmação RR não terá consequências se se referir a uma trama de dados intermédia e entretanto forem geradas novas confirmações para tramas de dados subsequentes, antes que ocorra *time-out* (confirmação em bloco)
  - » Perda da trama REJ enviada após o receptor detectar a perda de uma trama de dados (não é possível o envio de outro REJ)
  - » Perda da trama de dados retransmitida após recepção de REJ (não é possível o envio de outro REJ)

- Recuperação após *time-out*
  - » O emissor envia uma trama de controlo (RR) como comando e invocando *polling*, forçando o receptor a responder (com indicação do estado respectivo)
    - – O emissor poderia em alternativa retransmitir a trama de dados não confirmada mais antiga, o que poderia originar um duplicado e tornar o processo de recuperação mais complexo
  - » A trama de resposta ao *polling* indica em N(r) qual a trama de dados a partir da qual o emissor deve (re)transmitir (retomar o processo de transmissão normal)

Go-Back-N ARQ – recepção de tramas de controlo

» O emissor de tramas de dados analisa o valor N(r) recebido em tramas de controlo e em tramas de dados (*piggyback*); se se tratar duma confirmação efectiva de alguma(s) trama(s) de dados pendente(s), cancela o temporizador e reactiva-o se subsistir(em) trama(s) de dados não confirmada(s)

» Se a trama recebida for

- RR – a transmissão de tramas de dados pode continuar ou recomeçar, caso estivesse suspensa (e.g., RNR anteriormente recebido ou janela esgotada)
- RNR – deve suspender-se o envio de tramas de dados (podendo eventualmente transmitir-se a trama de dados referenciada por N(r), para evitar *deadlocks* resultantes duma eventual perda do RR ou REJ subsequente; a alternativa seria activar um temporizador após a recepção de RNR)
- REJ – deve(m) retransmitir-se a(s) trama(s) de dados a partir da referenciada por N(r), começando por fazer-se V(s) = N(r) (*Go-Back-N*)

» A recepção de uma trama de controlo em resposta ao comando accionado na sequência de *time-out* termina a situação de excepção e permite retomar os procedimentos normais

Go-Back-N ARQ – recepção de tramas de dados

» Em *Go-Back-N* o receptor apenas aceita tramas de dados em sequência, pelo que deve comparar o valor do campo N(s) com V(r)

» No caso de a estação não estar a realizar controlo de fluxo

– Se N(s) = V(r)

• Aceita a trama (eliminando uma eventual condição de rejeição estabelecida por REJ anterior)

• Incrementa (módulo M) o valor de V(r), apontando assim para a próxima trama de dados que espera receber

• Na primeira oportunidade envia uma confirmação, por meio de RR ou *piggyback*, colocando o valor corrente de V(r) no campo N(r) da trama respectiva

– Se N(s) ≠ V(r)

• Descarta a trama

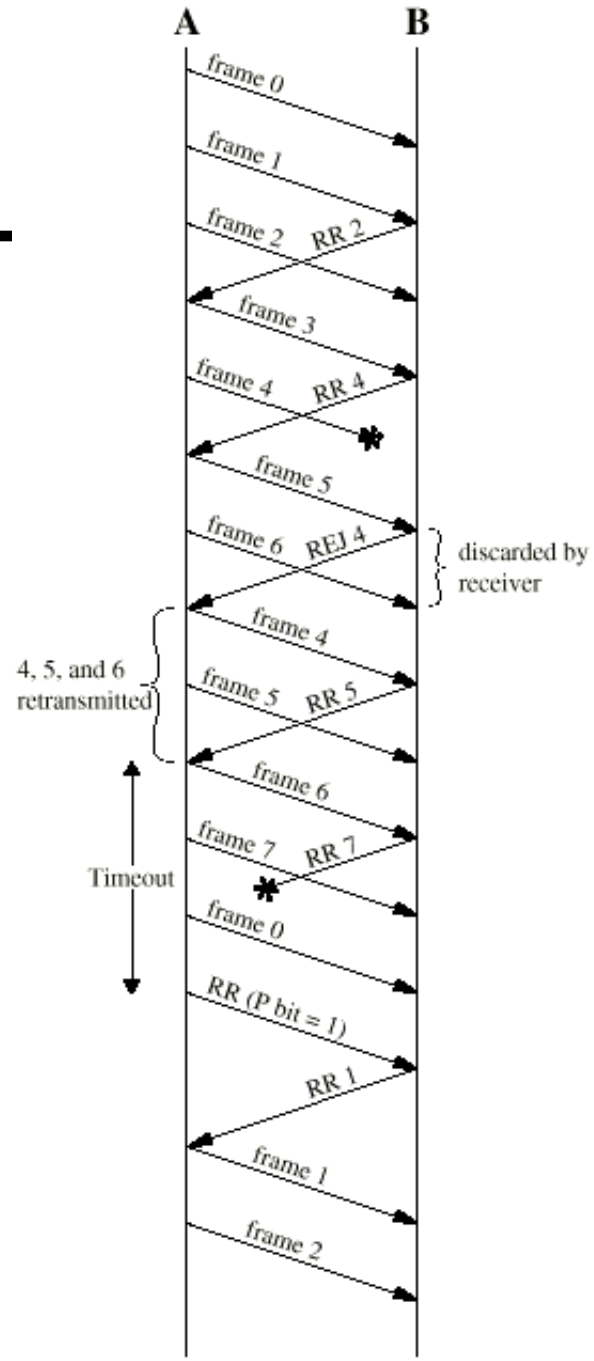
• Envia REJ, com o valor corrente de V(r) no campo N(r), se não houver já REJ pendente

» No caso de a estação se encontrar a realizar controlo de fluxo

– Envia RNR, se ainda o não fez, confirmando a(s) trama(s) de dados realmente aceite(s)

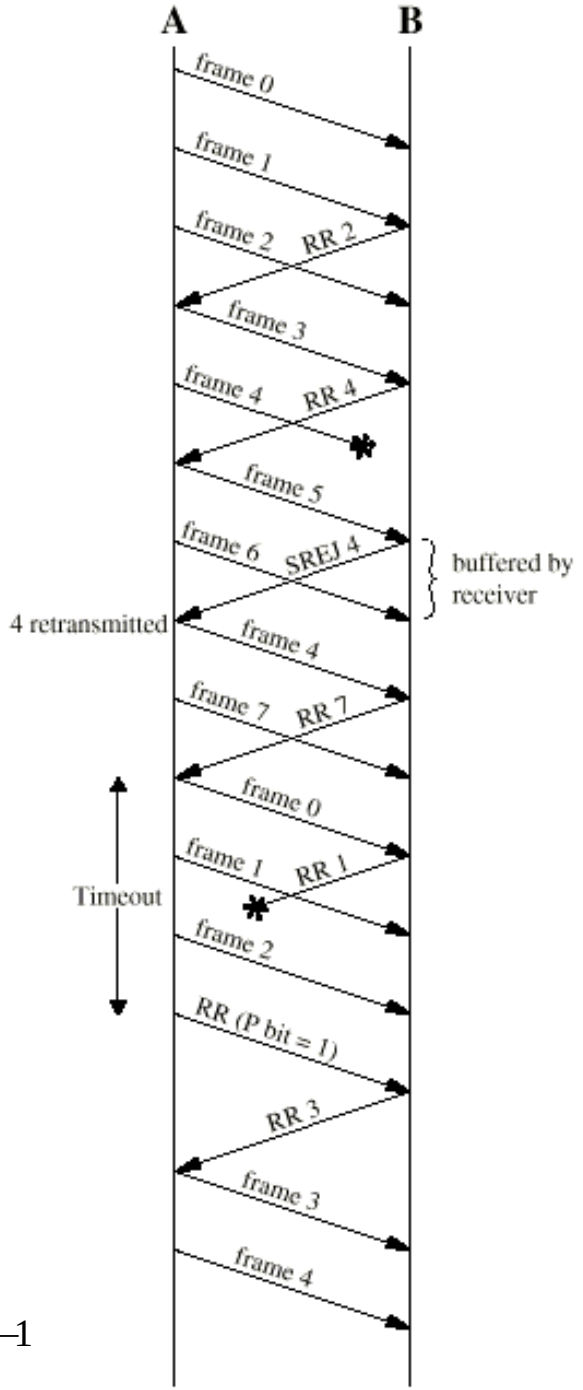
– Pode aceitar ou não a trama de dados recebida, mantendo a informação correspondente para posteriormente accionar RR ou REJ, conforme o caso

Go-Back-N ARQ – exemplo



Selective Repeat ARQ

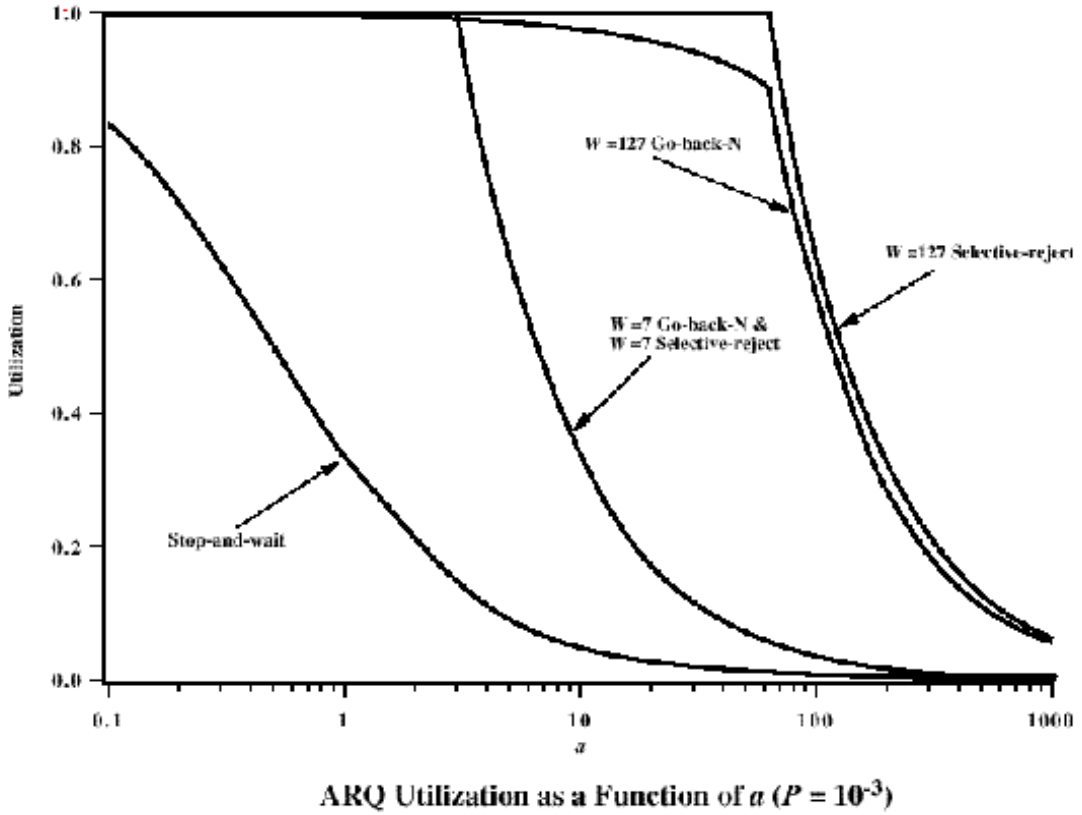
- Baseado em *Sliding Window*
- O receptor aceita tramas fora de sequência
  - » Confirma negativamente com SREJ tramas de dados em falta (um SREJ por trama)
  - » Confirma positivamente com RR apenas blocos de tramas de dados consecutivas
- O emissor apenas retransmite as tramas sinalizadas por SREJ, o que minimiza o número de retransmissões
  - » Vantajoso se W muito grande (e.g., satélite)
- Requer
  - » No receptor
    - – *Buffers* maiores
    - – Lógica para reinserção de tramas
  - » No emissor
    - – Lógica mais complexa para enviar tramas fora da sequência natural
- Tamanho máximo da Janela $W_{max} = 2^{k-1}$



Eficiência de Go-Back-N e Selective Repeat ARQ

» *Go-Back-N ARQ*

$$S = \begin{cases} \dfrac{1-P_e}{1+2aP_e} & W \geq 1+2a \\ \dfrac{W(1-P_e)}{(1+2a)(1-P_e+WP_e)} & W < 1+2a \end{cases}$$



ARQ Utilization as a Function of a ($P = 10^{-3}$)

» *Selective Repeat ARQ*

$$S = \begin{cases} 1-P_e & W \geq 1+2a \\ \dfrac{W(1-P_e)}{1+2a} & W < 1+2a \end{cases}$$

P_e – probabilidade de uma trama ser afectada por erro

High Level Data Link Control (HDLC)

- HDLC (ISO 33009, ISO 4335)
- Protocolo orientado ao bit (independente de códigos)
- Tipos de estações
  - Estação Primária
    - Controla a operação da ligação (uma ligação por cada estação Secundária)
    - As tramas enviadas designam-se **comandos**
  - Estação Secundária
    - Controlada pela estação Primária
    - As tramas enviadas designam-se **respostas**
  - Estação Combinada
    - Combina funções Primárias e Secundárias; pode enviar **comandos** e **respostas**
- Configurações da ligação
  - Não balanceada (não equilibrada)
    - Uma estação Primária e uma ou mais Secundárias
    - *Full-duplex* ou *half-duplex*
  - Balanceada (equilibrada)
    - Duas estações Combinadas
    - *Full-duplex* ou *half-duplex*

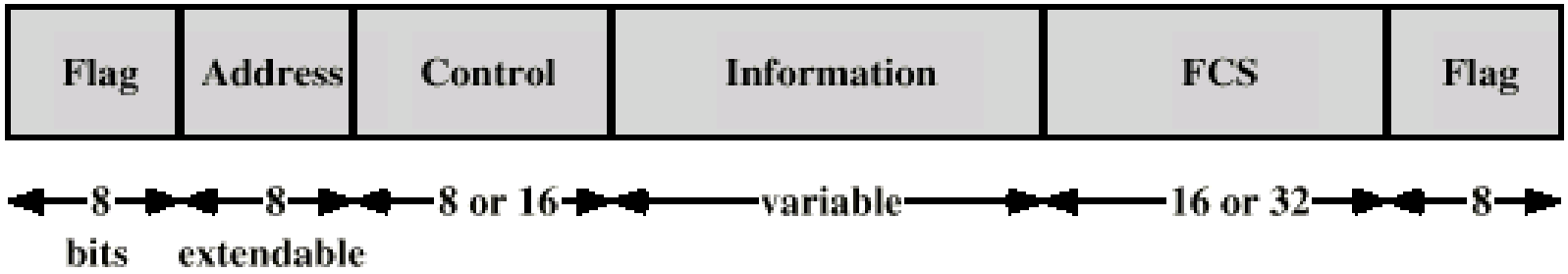
Modos de transferência de dados em HDLC

- *Normal Response Mode* (NRM)
  - Configuração não balanceada
  - A estação Primária é responsável pelo estabelecimento e gestão da ligação
  - Uma estação Secundária só responde a comandos da estação Primária (*Poll* / *Select*)
  - Usado em configurações *multidrop* (e.g., Primário: Computador, Secundárias: terminais)
- *Asynchronous Response Mode* (ARM)
  - Configuração não balanceada
  - A estação Primária é responsável pelo estabelecimento e gestão da ligação
  - A estação Secundária pode transmitir sem permissão da estação Primária, após estabelecida a ligação
  - Raramente usado
    - Foi proposta na primeira versão de X.25 (LAP) para garantir operação simétrica do terminal e do nó de comutação (actualmente sem interesse); requeria uma Estação Primária e uma Estação Secundária em cada sistema e portanto o estabelecimento de duas ligações ARM
- *Asynchronous Balanced Mode* (ABM)
  - Configuração balanceada
  - Qualquer das estações pode estabelecer a ligação e transmitir sem permissão da outra
  - Usado em X.25 (LAPB), RDIS (LAPD) e *Frame Relay* (LAPF)

Estrutura de uma trama HDLC

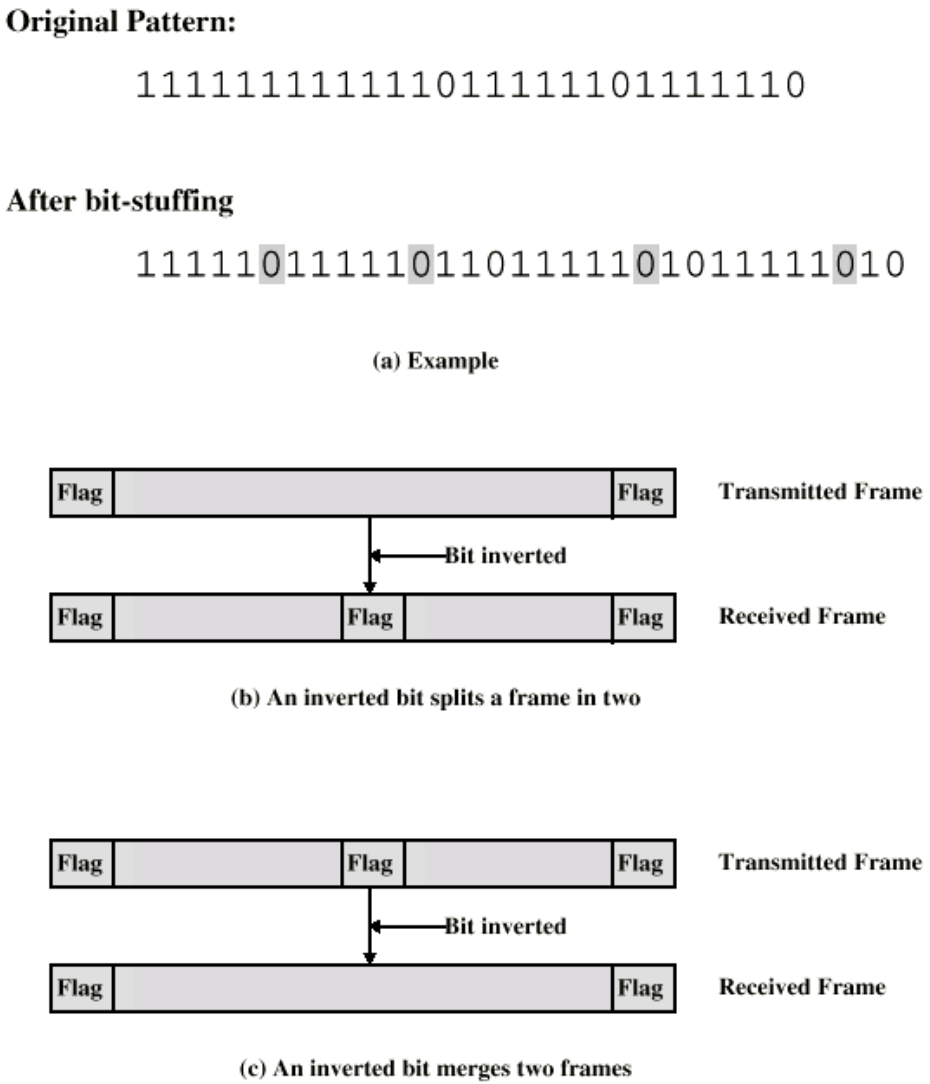
» Uma trama HDLC é constituída por vários campos
  – Os campos têm tamanho fixo ou variável (existindo mecanismos que permitem estabelecer e reconhecer as fronteiras entre campos)
  – As funções associadas a cada campo dependem da respectiva estrutura e de valores dos bits que o constituem, de acordo com a posição que ocupam no campo

» Formato único para todas as tramas de controlo e dados



Trama HDLC – delineação (flags)

- Tramas HDLC são delimitadas por *flags* – sequências 01111110
  - » Uma *flag* pode terminar uma trama e começar outra
- Entre tramas, o emissor pode enviar *flags* ou sequência de 1s (*idle*)
- *Bit stuffing* é usado para evitar o falso reconhecimento de *flags* dentro da trama
  - » O Emissor insere um 0 após uma sequência de cinco 1s
  - » Quando o Receptor detecta cinco 1s consecutivos verifica o 6º e o 7º bits
    - Se 6º bit = 0 → elimina bit
    - Se 6º bit = 1 e 7º bit = 0 → aceita *flag*
    - Se 6º bit = 1 e 7º bit = 1 → indica que o emissor abortou a transmissão (por meio de uma sequência de 1s)



Trama HDLC – campo de endereço

» Por convenção o endereço identifica a estação secundária que enviou ou vai receber a trama (resposta e comando, respectivamente); um bit (C/R) neste campo permite estabelecer a diferença entre comandos e respostas

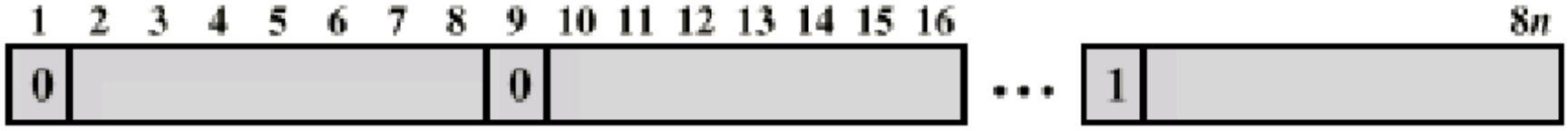
» Formato básico – 8 bits

» Formato expandido

– Múltiplo de 8 bits

– O bit menos significativo (LSB) de cada octeto indica se este é o último octeto (LSB = 1) ou não (LSB = 0)

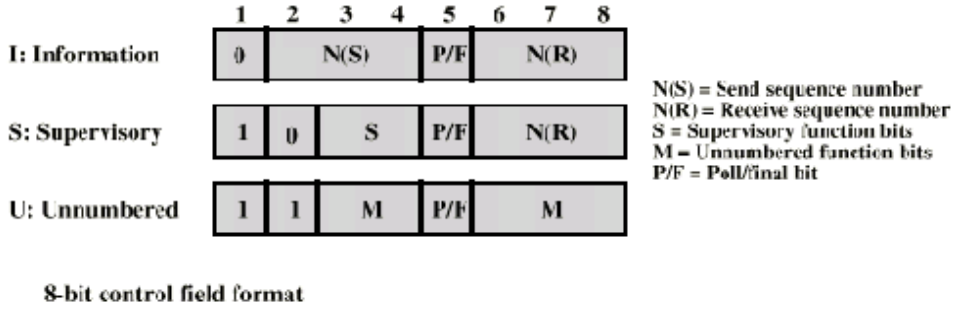
» Endereço de *broadcast* (difusão) – 11111111



Extended Address Field

Trama HDLC – campo de controlo

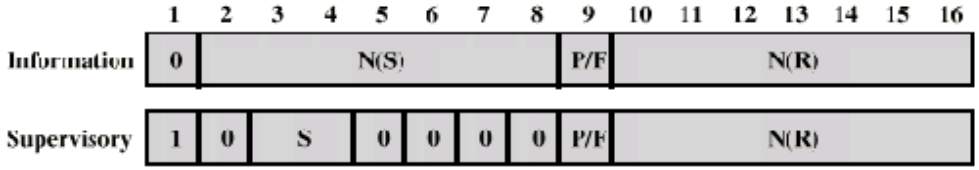
» Três tipos de tramas (funções)
  – Informação (I)
    • Contém dados de camadas superiores
    • Confirmação de tramas (*piggyback*)
  – Supervisão (S)
    • Confirmação de tramas, Controlo de Erros e de Fluxo (RR, RNR, REJ, SREJ)
  – Não numeradas (U)
    • Controlo da ligação (estabelecimento, reinicialização, terminação)



8-bit control field format

» Números de sequência
  – N(S) em tramas I
  – N(R) em tramas I e S

» Bit P/F (*Poll* / *Final*)
  – Permite sincronizar um comando com uma resposta (o funcionamento normal do protocolo não garante uma relação um para um entre comandos e respostas)
  – Em comandos, P = 1 solicita resposta
  – Em respostas F = 1 indica resposta solicitada por um comando com P = 1
  – Usado na recuperação após *time-out*



16-bit control field format

Trama HDLC – campos de Informação e FCS

- Campo de Informação
  - » Apenas presente em tramas I e em algumas tramas U (e.g., UI, FRMR)
  - » Contém tipicamente (mas não obrigatoriamente) um número inteiro de octetos
  - » Comprimento variável (valor máximo fixado previamente)

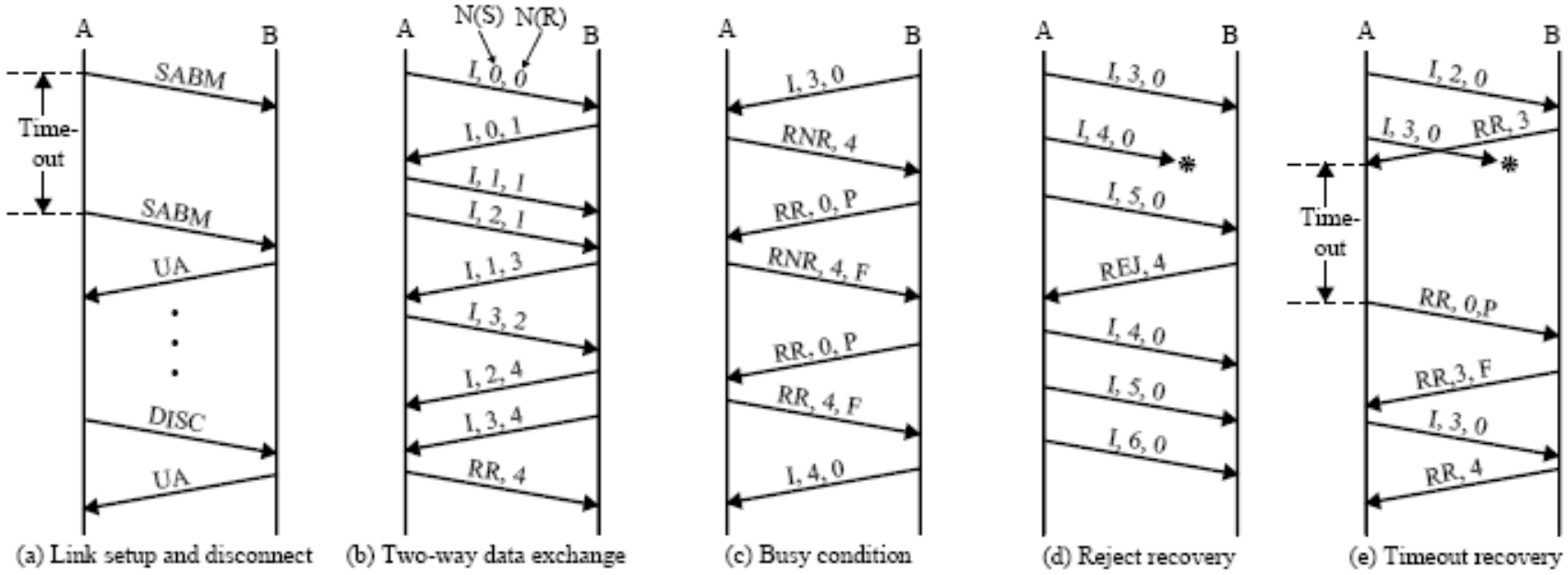
- FCS (*Frame Check Sequence*)
  - » Baseado num código polinomial
    - – Normalmente ITU-16 (16 bits)
    - – Por vezes ITU-32 (32 bits)

HDLC – lista de tramas

Name	Command/ Response	Description
Information (I)	C/R	Exchange user data
Supervisory (S)		
Receive Ready (RR)	C/R	Positive acknowledgement; ready to receive I-frame
Receive Not Ready (RNR)	C/R	Positive acknowledgement; not ready to receive
Reject (REJ)	C/R	Negative acknowledgement; go back N
Selective Reject (SREJ)	C/R	Negative acknowledgement; selective reject
Unnumbered (U)		
Set normal response mode (SNRM)	C	Set mode
Set normal response extended mode (SNRME)	C	Set mode; extended = 7 bit sequence numbers
Set asynchronous response mode (SARM)	C	Set mode
Set asynchronous response extended mode (SARME)	C	Set mode; extended = 7 bit sequence numbers
Set asynchronous balanced mode (SABM)	C	Set mode
Set asynchronous balanced extended mode (SABME)	C	Set mode; extended = 7 bit sequence numbers
Set initialization mode (SIM)	C	Initialize link control functions in addressed station
Disconnect (DISC)	C	Terminate logical link connection
Unnumbered Acknowledgment (UA)	R	Acknowledge acceptance of one of the set mode commands
Disconnected Mode (DM)	R	Responder in disconnected mode
Request Disconnect (RD)	R	Request for DISC Command
Request Initialization Mode (RIM)	R	Initialization needed; request for SIM command
Unnumbered Information (UI)	C/R	Unacknowledged data
Unnumbered Poll (UP)	C	Used to solicit response
Reset (RSET)	C	Used for recovery; resets N(r) in addressed station
Exchange Identification (XID)	C/R	Used to request/report capabilities
Test (TEST)	C/R	Exchange identical information fields for testing
Frame Reject (FRMR)	R	Reports receipt of unacceptable frame

HDLC – exemplo de funcionamento

» Três fases: estabelecimento, transferência de dados, terminação



LAPB (X.25) e LAPD (RDIS)

- *LAPB – Link Access Procedures, Balanced*
  - » Nível 2 do X.25 (ITU-T)
  - » Subconjunto do HDLC (ABM)
  - » Ligação ponto-a-ponto entre equipamento terminal (DTE) e nó da rede de comutação de pacotes
- *LAPD – Link Access Procedures, D-Channel*
  - » RDIS / ISDN (ITU-T)
  - » ABM
  - » Números de sequência com 7 bits (numeração estendida)
  - » Campo de endereço com 16 bits
    - – Permite multiplexagem de ligações de dados na interface utilizador-rede, o que requer dois níveis de identificação (SAPI, TEI) para suportar:
      - Múltiplas entidades que usam o serviço LAPD em cada sistema (sinalização, comunicação em modo pacote ou trama, gestão, etc.) – *Service Access Point Identifier* (SAPI)
      - Múltiplos terminais na mesma interface com a rede – *Terminal Endpoint Identifier* (TEI); cada terminal necessita de ter pelo menos um TEI atribuído

LAPF (Frame Relay)

- *LAPF – Link Access Procedures for Frame-Mode Bearer Services*
  - » Serviços de suporte baseados na comutação de tramas transportadas em circuitos virtuais – são usados canais lógicos distintos para transferência de dados e procedimentos de sinalização (*out of band*)
  - » Dois subníveis
    - – *DL-core* – funções básicas (apenas campo de endereço)
    - – *DL-control* – confirmação, controlo de erros e controlo de fluxo
  - » ABM
  - » Números de sequência com 7 bits (*DL-control*)
  - » CRC de 16 bits
- Campo de endereços com 2, 3 ou 4 octetos
  - » Identificação da Ligação de Dados
    - – *Data Link Connection Identifier* (DLCI)
  - » Funções diversas
    - – Notificação de congestionamento e indicação para descarte prioritário de tramas pela rede

PPP (Point-to-Point Protocol)

- Protocolo de Ligação de Dados usado em ligações físicas ponto a ponto na Internet
  - » Definido nos RFCs (*Request for Comments*) 1661, 1662 e 1663
- Características
  - » Suporta detecção de erros
  - » Suporta múltiplos protocolos
  - » Permite negociação de endereços IP
  - » Permite autenticação
- Mecanismos
  - » Delineação de tramas e detecção de erros
  - » Protocolo de controlo de ligação (LCP – *Link Control Protocol*) para estabelecimento, teste e terminação de ligações e negociação de opções
  - » Protocolo de controlo de rede (NCP – *Network Control Protocol*) específico de cada protocolo de rede para negociação de opções da camada de Rede

Tramas PPP

- Formato baseado em HDLC
  - » Número inteiro de octetos (*byte stuffing* em vez de *bit stuffing*)
  - » Delineação por *flags* (com *byte stuffing* se ocorrer uma *flag* / ESC na trama)
  - » Campo de Endereço (1 octeto)
    - – 11111111 (indica que trama deve ser aceite por qualquer estação)
  - » Campo de Controlo (1 octeto)
    - – Valor *default* 00000011 (trama não numerada) – transmissão não fiável (erros são detectados mas não são recuperados por retransmissão)
    - – RFC 1663 define um modo de transmissão fiável para usar em meios adversos (*wireless*)
  - » Campo de Protocolo (1 ou 2 octetos)
    - – Identifica o protocolo transportado no campo de dados (LCP, NCP, IP, etc.)
  - » Campo de dados (tamanho variável – número inteiro de octetos)
    - – O valor máximo pode ser negociado, sendo o *default* igual a 1500 octetos
  - » Campo de verificação de paridade (2 ou 4 octetos)
    - – Tipicamente 2 octetos, podendo ser negociados 4 octetos
  - » LCP permite negociar a omissão dos campos de Endereço e de Controlo

LCP – Link Control Protocol

- Após estabelecimento da ligação física (por exemplo, por *dial-up*) inicia-se a fase de negociação de opções
- Se esta fase se completar com sucesso passa-se à fase de autenticação, durante a qual as duas partes verificam as respectivas identidades
- Na fase seguinte é invocado o protocolo NCP apropriado para configurar a camada de Rede (de acordo com o protocolo de Rede acordado na fase de negociação)
- Completada a fase de configuração, pode dar-se início à troca de dados, de acordo com o protocolo seleccionado (por exemplo, IP)
- Concluída a troca de dados a ligação PPP é terminada e a ligação física abandonada

Logical Link Control (LANs IEEE 802)

» Parte da arquitectura das LANs IEEE 802
  – Controlo da ligação dividido em duas sub-camadas
    • *Medium Access Control* (MAC)
    • *Logical Link Control* (LLC)

» Sub-camada MAC
  – Protocolo de acesso a um meio partilhado / comutação de tramas (*bridges* / *LAN switches*)
  – Endereçamento físico das estações (endereços MAC)
  – Detecção de erros (eliminação de tramas com erros, sem recuperação a nível MAC)

» Sub-camada LLC
  – Endereçamento lógico permite identificação (multiplexagem) de protocolos de camadas superiores
    • DSAP – *Destination Service Access Point*
    • SSAP – *Source Service Access Point*
  – Três tipos de serviços
    • Confirmado, com conexão (fiável) – procedimentos semelhantes a HDLC
    • Confirmado, sem conexão
    • Não confirmado, sem conexão (não fiável) – o mais comum (usado, por exemplo, com TCP/IP)

Protocolos orientados ao caracter

» Os protocolos orientados ao caracter utilizam caracteres de controlo de um código (por exemplo, o código ASCII) para delimitar os blocos de dados (tramas) e para supervisionar a troca de dados

» Os protocolos orientados ao caracter apresentam limitações importantes que os tornam pouco eficientes e mal adaptados para aplicações interactivas

- Caracterizam-se por ligação rígida a um código, o que requer versões diferentes para códigos diferentes, tornando-os pouco flexíveis para futuras expansões (difícil modificação) e inerentemente não transparentes (a transparência consegue-se recorrendo a caracteres de escape, o que aumenta a complexidade)
- Utilizam procedimentos do tipo bidireccional alternado, o que exige um número elevado de *turnarounds* da ligação lógica (em particular para a confirmação individual de tramas de dados)
- Em geral apenas uma função é realizada por cada trama enviada
- Normalmente apenas as tramas de dados estão protegidas contra erros
- A necessidade de distinguir dados e caracteres de controlo cria *overheads* adicionais

» Um exemplo de um protocolo deste tipo é o Bisync (*Binary Synchronous*)

Caracteres de controlo (ASCII)

- **SYN** *Synchronous Idle*; caracter de sincronismo de trama
- **SOH** *Start of Heading*; delimitador de início de cabeçalho
- **STX** *Start of Text*; delimitador de início de texto (dados)
- **ETB** *End of Transmission Block*; delimitador de fim de dados numa trama intermédia de uma mensagem
- **ETX** *End of Text*; delimitador de fim de dados na última trama de uma mensagem
- **ENQ** *Enquiry*; solicita uma resposta (usado em *Polling / Selecting*)
- **ACK** *Acknowledgement*; confirmação (positiva) de recepção de trama
- **NAK** *Negative Acknowledgement*; solicita retransmissão em virtude de ter sido recebida uma trama com erro (confirmação negativa)
- **EOT** *End of Transmission*; indica fim de transmissão (terminação da ligação)
- **DLE** *Data Link Escape*; modifica o significado do caracter seguinte
  - **DLE STX** inicia um bloco de dados transparentes numa trama
  - **DLE ETB** termina um bloco de dados transparentes numa trama intermédia duma mensagem
  - **DLE ETX** termina um bloco de dados transparentes na última trama duma mensagem
  - **DLE DLE** transmissão de DLE num bloco de dados transparentes
  - **DLE 0 / DLE 1** realiza **ACK 0 / ACK 1**

Formatos de tramas

Exemplos de formatos de tramas de dados e controlo

- SYN SYN SOH Cabeçalho ETB BCC
- SYN SYN SYN STX Dados ETB/ETX BCC
- SYN SYN SOH Cabeçalho STX Dados ETB/ETX BCC
- SYN SYN DLE STX Dados transparentes DLE ETB/ETX BCC
- SYN SYN Endereço ENQ
- SYN SYN ACK (ou SYN SYN DLE 0 / SYN SYN DLE 1)
- SYN SYN NAK
- SYN SYN EOT

As tramas que contêm cabeçalho e/ou dados são protegidas por um código detector de erros baseado em caracteres de controlo de paridade no fim da trama (BCC – *Binary Check Character*), do tipo LRC (*Longitudinal Redundancy Check)* ou CRC *(Cyclic Redundancy Check)*

Polling

» **P**: O Primário inicia *Polling* ao Secundário designado (início da ligação)

SYN SYN Endereço (Poll) ENQ

» **Caso 1** – o Secundário não tem dados para transmitir

S: O Secundário devolve controlo ao Primário (fim da ligação), enviando

SYN SYN EOT

» **Caso 2** – o Secundário tem dados para transmitir

S: O Secundário envia dados de acordo com um formato apropriado; antes de transmitir a trama de dados seguinte espera confirmação do Primário

P: O Primário confirma individualmente cada trama recebida

SYN SYN DLE 0 / DLE 1

S: No final da troca de dados, para devolver controlo ao Primário (fim da ligação), o Secundário envia

SYN SYN EOT

» Após conclusão da ligação, o Primário pode iniciar novo *Polling* ou *Selecting*

Selecting

» **P**: O Primário inicia *Selecting* ao Secundário designado (início da ligação)

SYN SYN Endereço (Select) ENQ

» **Caso 1** – o Secundário está pronto a receber dados

S: O Secundário envia

SYN SYN DLE 0

P: O Primário inicia a transferência de dados, esperando uma confirmação do Secundário antes de enviar uma nova trama

S: O Secundário confirma uma a uma cada trama recebida

SYN SYN DLE 0 / DLE 1

P: No final da troca de dados, para terminar a ligação, o Primário envia

SYN SYN EOT

» **Caso 2** – o Secundário não está em condições de receber dados

S: O Secundário envia

SYN SYN NAK

P: O Primário pode tentar novamente (repetindo o processo um número de vezes predefinido) ou terminar a ligação com

SYN SYN EOT

» Após conclusão da ligação, o Primário pode iniciar novo *Polling* ou *Selecting*