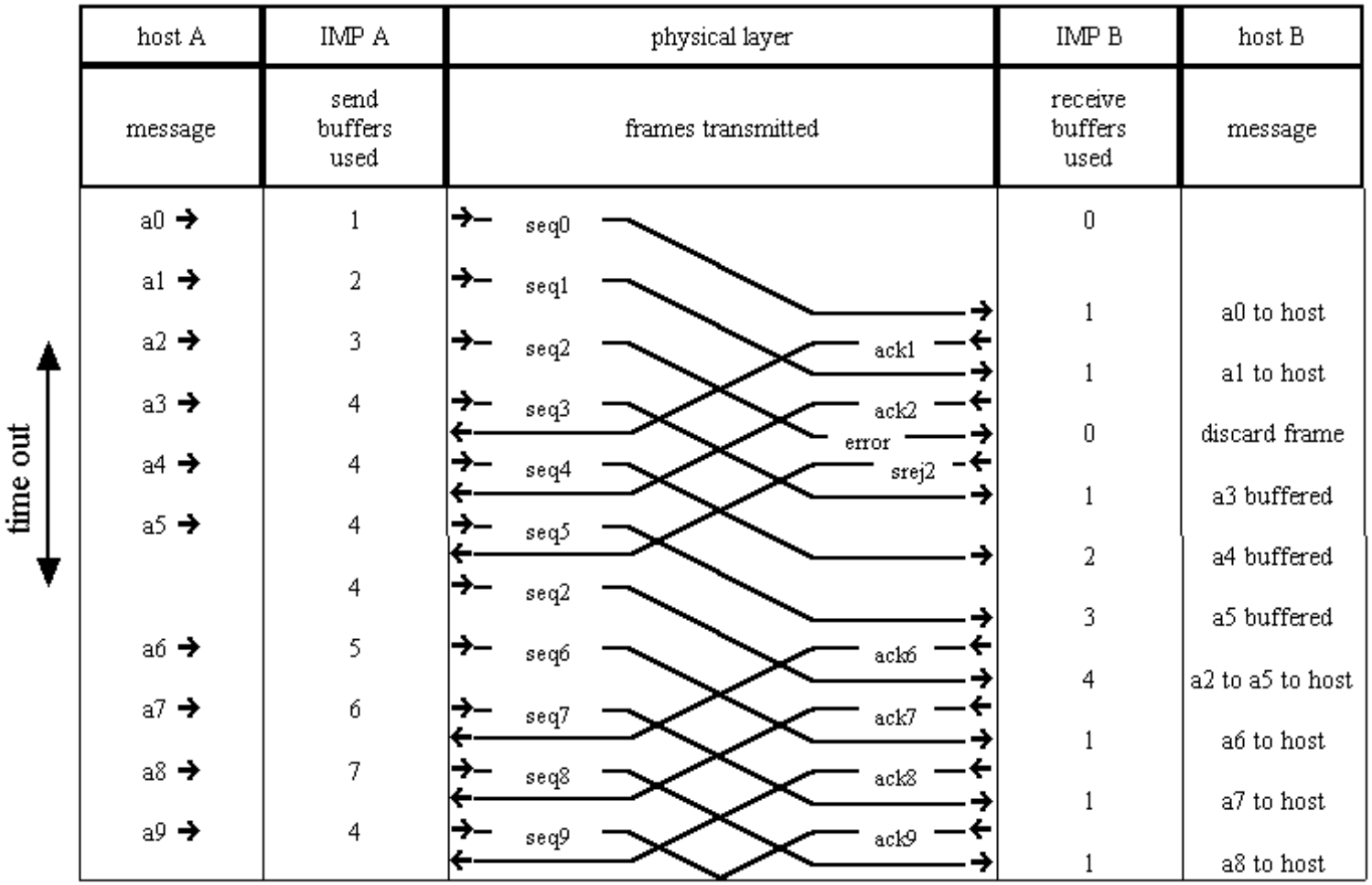


Řízení datového spoje

O čem přednáška je?

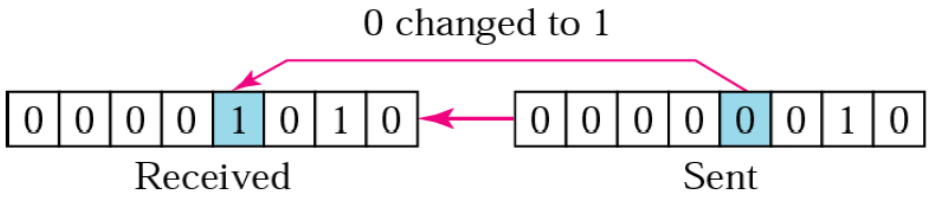
- spojová vrstva
- klasifikace chyb
- zabezpečovací a opravné kódy
- řízení toku dat
- chybové řízení

Proč nestačí služby fyzické vrstvy

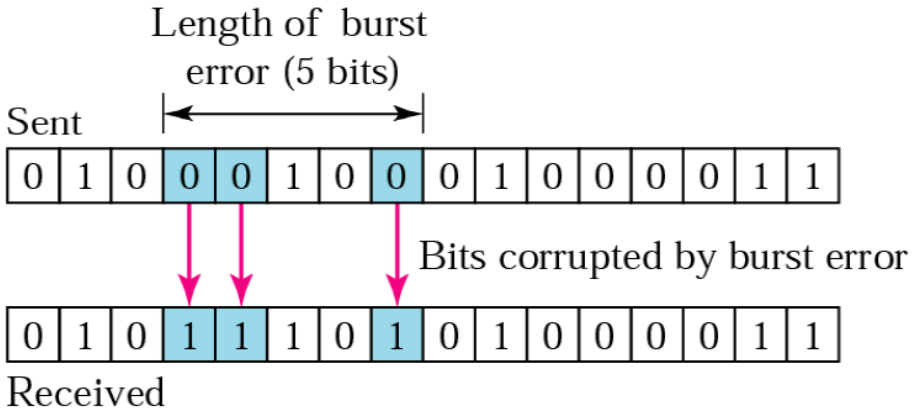
- neumí rozlišovat zda přenášejí data nebo řídící informace
- nezajišťují opakování chybně přenesené informace
- nepodporují ovládání toku informace ze zdroje do média
- nepodporují určení entity mající právo vysílat do média
- nepodporují komunikaci mezi nesousedními partnery
- spojová vrstva zajišťuje
  - synchronizaci rámců
  - chybové řízení
  - řízení toku dat
  - adresování stanic na médiu
  - rozlišování řídících informací a dat
  - správu datového spoje

Klasifikace chyb

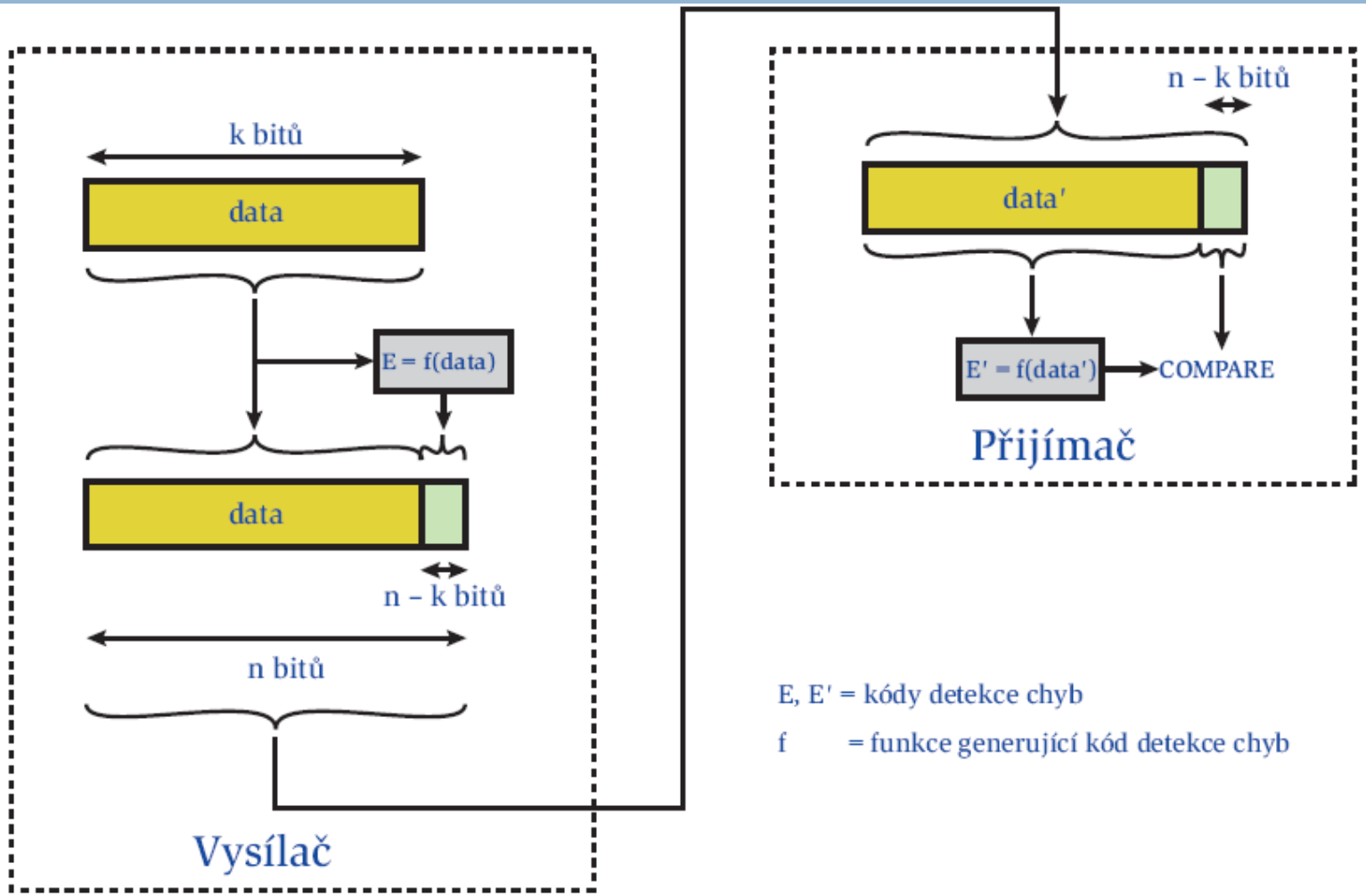
- 1bitové chyby
  - neovlivňují sousední bity
  - zdroj typicky bílý šum



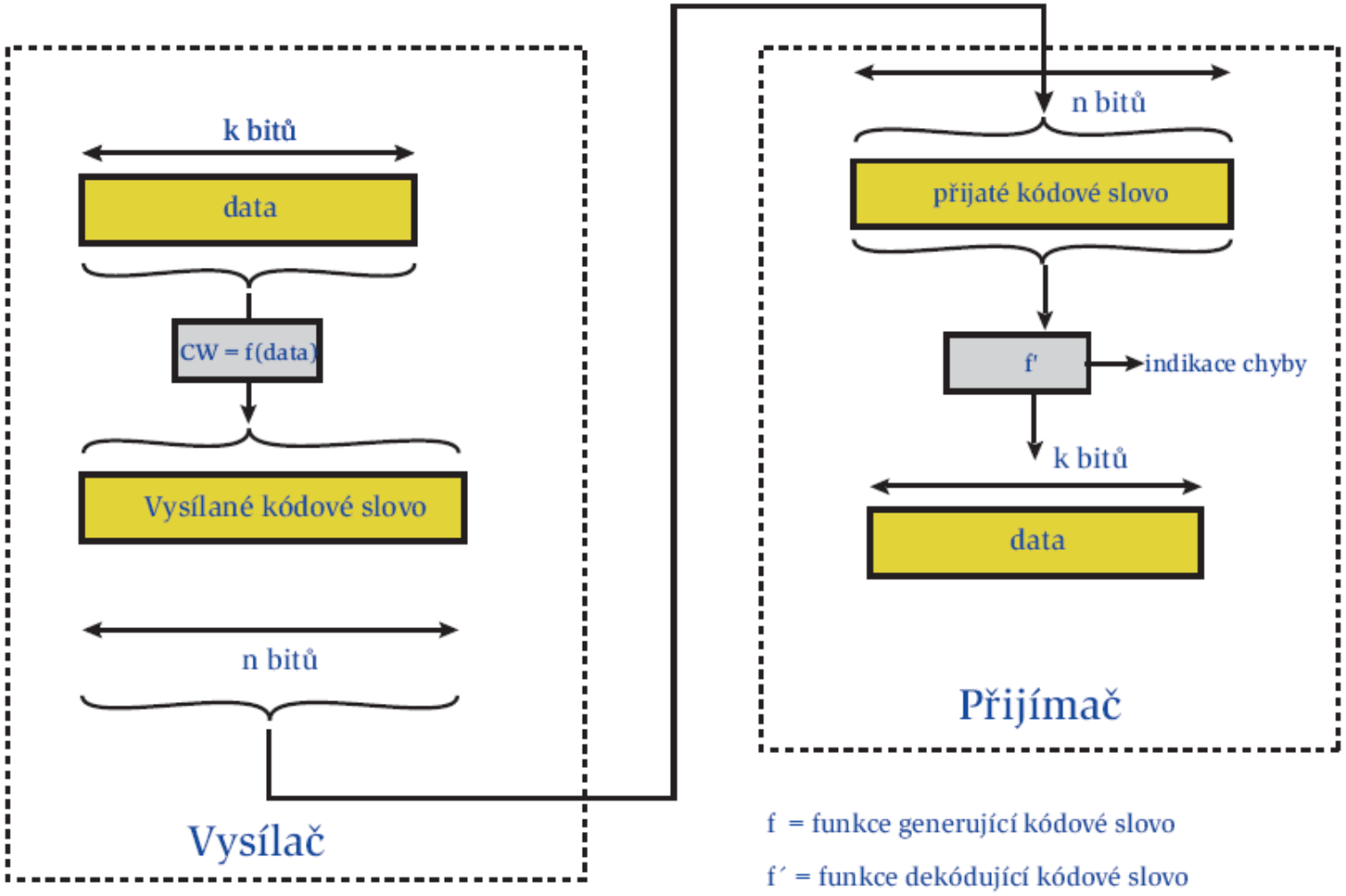
- dávkové chyby
  - chybné posloupnosti bitů délky N
  - mezi posledním chybným a prvním chybným bitem další dávky je alespoň N bitů správných
  - důležité u vyšších přenosových rychlostí



Proces detekce chyb, generické schéma

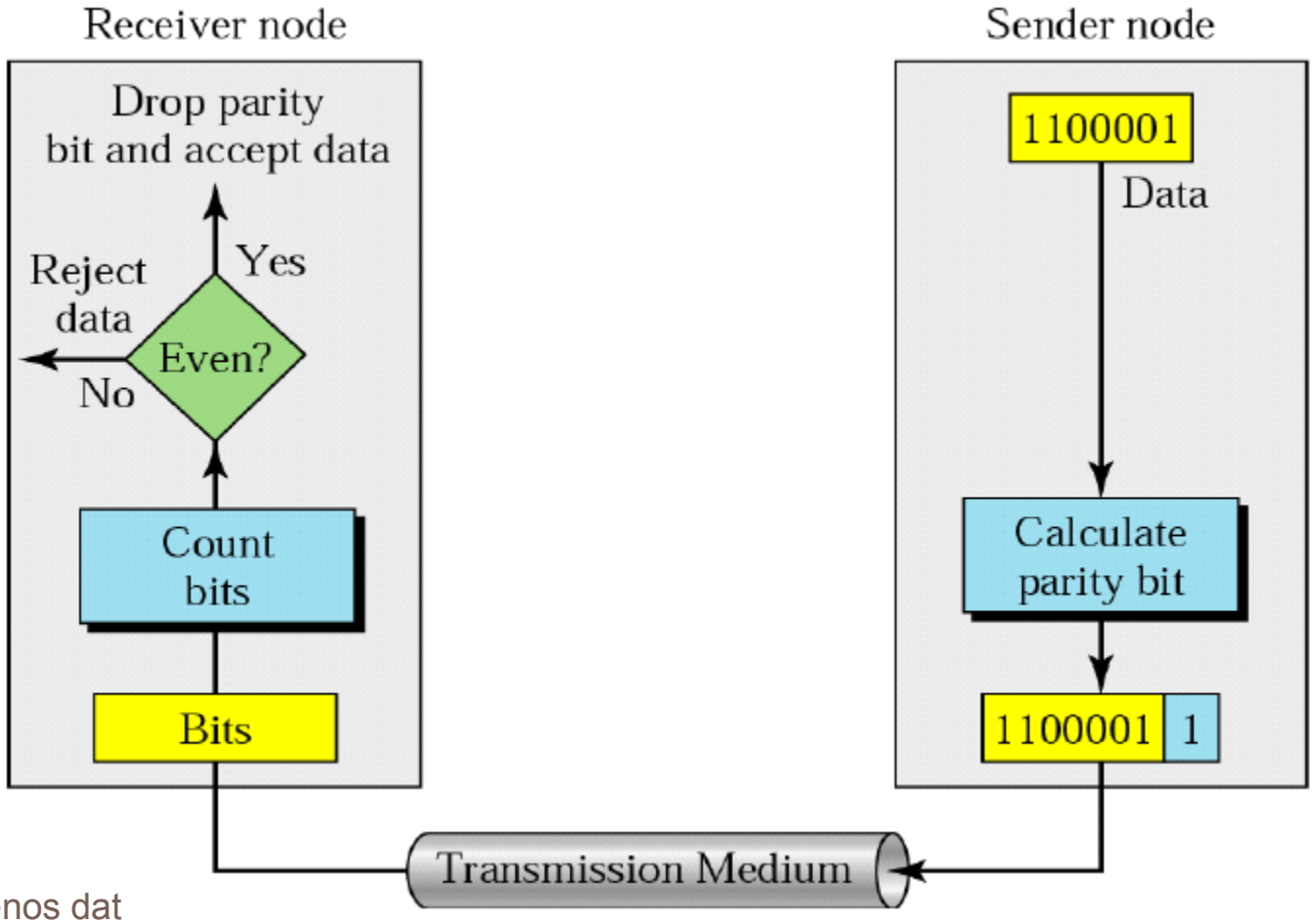


Proces detekce chyb, další možné schéma



Parita

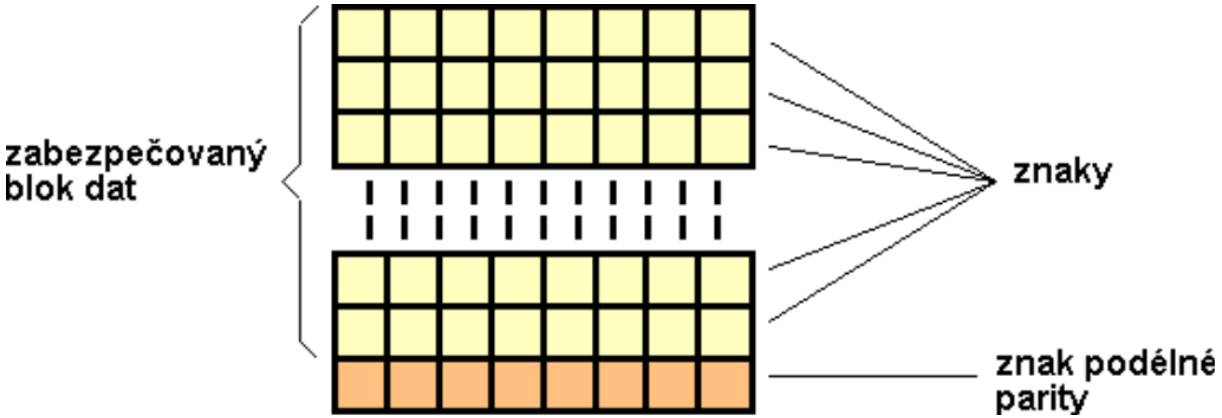
- výsledný kód znaku musí mít sudý nebo lichý počet 1



Typy paritních zabezpečení

- příčná parita – sudá nebo lichá parita oktetu

- podélná parita – po řádech oktetů přes celý blok



- křížová parita – kombinace příčné a podélné parity

Blokové kódy (n,k)

- přidávání paritního bitu k symbolu je příklad techniky
  blokového kódování
- blokový kód (n, k), $n > k$
  - ✓ algoritmus přiřazuje k-bitovému datovému slovu
    n-bitové (vysílané) kódové slovo
  - ✓ v množině 2^n kódových slov existuje 2^k validních kódových slov odpovídajících jednotlivým datovým slovům, těch je 2^k
  - ✓ kódové slovo některých algoritmů zobrazuje originální datové slovo (má pak formát „původní data + redundantní bity"), jedná se pak o tzv. systematické kódy
  - ✓ kódová slova kódů, které nejsou systematické, nezobrazují originální datové slovo (mají formát jedinečných kódových slov)
  - ✓ přidávání paritního bitu k ASCII znaku je příklad systematického blokového kódu typu (8,7)

Lineární blokové kódy

- Blokový kód (n, k) je lineární, když platí $c = dG$, kde
  - ✓ c je n-bitové kódové slovo, d je k-bitové slovo dat
  - ✓ $G_{k,n}$ je definiční matice s koeficienty 1 a 0
  - ✓ násobení matice se řeší v aritmetice $mod\ 2$
- Příklad – blokový Hammingův kód (7,4)
  - ✓ příslušná definiční matice G = $\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$
  - ✓ kódové slovo odpovídající datovému slovu $[1010]$ je $[1010001]$, poněvadž $[1010]G = [1010001]$
  - ✓ Hammingův kód (7,4) je systematický kód

Lineární blokové kódy

- Vlastnosti definiční matice lineárních blokových kódů
  - ✓ Neobsahuje tzv. nulový řádek, tj. posloupnost tvořenou pouze nulami
  - ✓ Řádky matice jsou lineárně nezávislé.
    - žádný řádek matice G nevznikne součtem několika jiných řádků z téže matice G
  - ✓ minimální Hammingova vzdálenost d mezi libovolným vytvořeným kódovým slovem a jinými kódovými slovy je konstantní
  - ✓ Hammingova vzdálenost – míra „odlišnosti" slov (viz další text)

Hammingova vzdálenost

- počet pozic, na kterých se řetězce hodnot stejné délky liší, d
- neboli počet záměn, které je potřeba provést pro změnu jednoho z řetězců na druhý
- pro binární slova je Hammingova vzdálenost dána počtem bitů, ve kterých se daná slova liší
- Příklad: počet získáme z non-ekvivalence dvou kódových slov

  1010101010
  1100110010
  – – – – –
  0110011000

  Hammingova vzdálenost uvedených slov = 4

Systematické lineární kódy

- Lineární blokový kód je systematický, když má generující matici ve tvaru

$$G_{k,n} = (I_{k,k}|P_{k,n-k})$$

kde $I_{k,k}$ je čtvercová jednotková matice o k řádcích a $P_{k,n-k}$ je matice o k řádcích a $n-k$ sloupcích.

- blokový Hammingův (7,4) kód je systematický kód , datové slovo / kódové slovo: [1010] / [1010001]

✓ příslušná definiční matice G =

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Cyklické kódy

- speciální případ lineárních blokových kódů
- Cyklický kód je lineární blokový kód (n,k) s vlastností:
  každý cyklický posuv kódového slova takového kódu je opět kódové slovo tohoto kódu
  - ✓ cyklické kódy jsou velmi oblíbené:
    – efektivně implementují detekci výskytu chyb
    – jsou snadno implementovatelné v hardware
- Příklad: lineární blokový kód (3,2) s definiční maticí
  $G = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$ je cyklický kód
  - ✓ pro datová slova 00, 01, 10 a 11
    generuje (validní) kódová slova 000, 101, 110, a 011
  - ✓ každé generované kódové slovo po cyklickém posuvu vlevo či vpravo dává opět validní kódové slovo
  - ✓ ilustrovaný kód je cyklický kód, nikoli však systematický kód

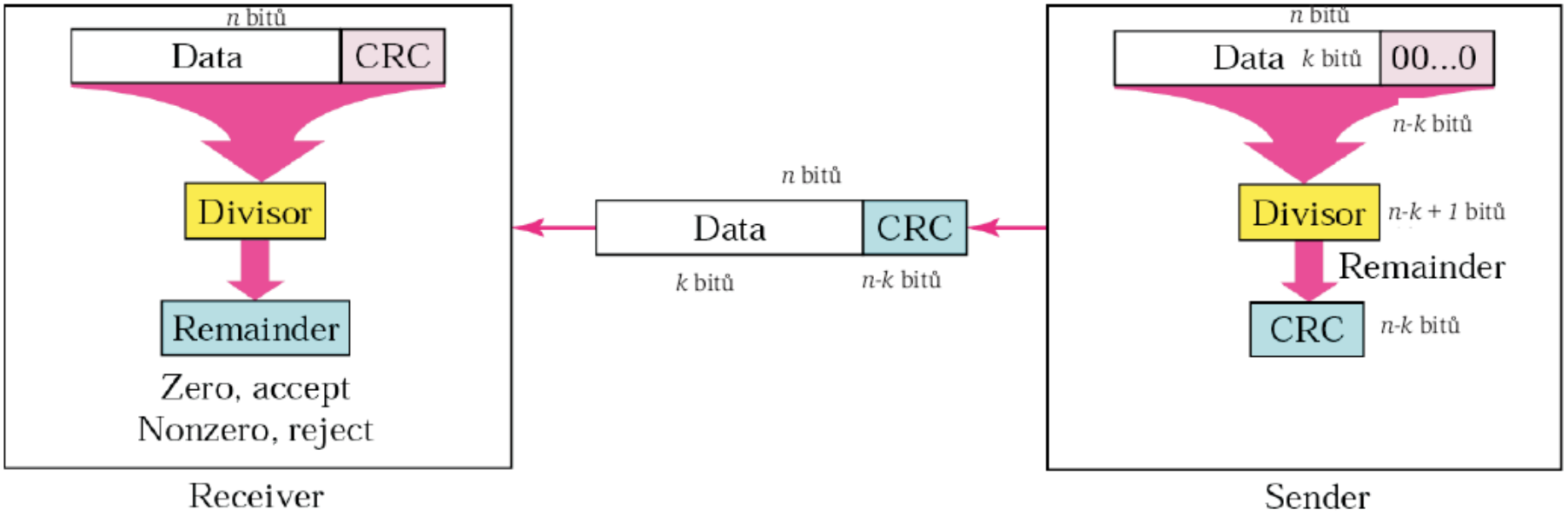
Cyklické kódy s kontrolní redundancí, CRC

- CRC, *Cyclic Redundancy Check*
- CRC jsou cyklické (tedy i lineární blokové) kódy
- CRC jsou určeny pro detekci chyb v přijaté informaci
- pro blok k bitů dat (zprávu) se vygeneruje $(n\text{-}k)$ bitová posloupnost (FCS, *Frame Check Sequence*) přidávaná ke k bitům zprávy,
  kde n je délka generovaného kódového slova
- data + FCS tvoří kódové slovo o délce n bitů,
  CRC je systematický kód

Cyklické kódy s kontrolní redundancí, CRC

- FCS se generuje se tak, aby vytvořené kódové slovo, tj. zřetězení dat a FCS, bylo číslo bezezbytku dělitelné jistým předem daným číslem – dělitelem, resp. tzv. klíčem
- FCS se generuje jako zbytek po dělení zprávy (posunuté vlevo o délku FCS) klíčem
- Klíč hraje roli definiční matice lineárních blokových kódů
- Předmětem přenosu je kódové slovo, rámec, o délce n bitů, obsahující na počátku originální blok dat – zprávu
- Přijímač přijaté kódové slovo (rámec) vydělí klíčem (dělitelem)
- Je-li zbytek dělení nulový, přenos proběhl bez chyb

Generátor a kontrolér CRC



Polynomový model

- zpráva (D) se chápe jako polynom s binárními koeficienty rovnými hodnotám bitů zprávy
  - ✓ pokud $D = 11001_2$,
    pak její reprezenatce v polynomovém modelu je
    $D(X) = X^4 + X^3 + 1$
  - ✓ polynom odpovídající k-bitovému slovu je řádu $k - 1$
- dělitelé (klíče) jsou definované několika standardy, např.

CRC-12: $G(X) = X^{12} + X^{11} + X^3 + X^2 + X + 1$

CRC-16: $G(X) = X^{16} + X^{15} + X^2 + 1$

CRC-CCITT: $G(X) = X^{16} + X^{12} + X^5 + 1$

CRC-32: $G(X) = X^{32} + X^{26} + X^{23} + X^{22} + X^{+6} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$

Polynomový model, 2

- vysílaný n-bitový rámec obsahující k-bitovou zprávu D s připojeným $n-k$-bitovým FCS reprezentuje polynom $T(X) = X^{n-k}D(X) + FCS$,
  - ✓ násobení X^{n-k} posouvá $D(X)$ o $n-k$ bitů vlevo
- Příklad
  - ✓ 10-bitová posloupnost dat [1010100101] má polynomiální reprezentaci $D(X) = X^9 + X^7 + X^5 + X^2 + 1$
  - ✓ nechť FCS je 3-bitová posloupnost [111]
  - ✓ odpovídající kódové slovo [1010100101111] má polynomiální reprezentaci $X^3(X^9 + X^7 + X^5 + X^2 + 1) + X^2 + X + 1 = X^{12} + X^{10} + X^8 + X^5 + X^3 + X^2 + X + 1.$

Polynomový model, 3

CRC proces tvorby n-bitového kódového slova:

- k-bitová zpráva $D(X)$,
  $(n-k)$-bitový klíč (dělitel, zabezpečovací polynom) $G(X)$
- $X^{n-k}D(X)/G(X) = Q(X) + R(X)/G(X)$, podíl + zbytek
- úprava násobením $G(X)$: $X^{n-k}D(X) = Q(X) \times G(X) + R(X)$
- v aritmetice $mod\ 2$ je součet a rozdíl ekvivalentní a tudíž platí
  $X^{n-k}D(X) + R(X) = Q(X) \times G(X)$
- tj. pokud se použije $FCS = R(X)$,
  pak kódové slovo $T(X) = X^{n-k}D(X) + FCS$
  je dělitelné klíčem $G(X)$ beze zbytku

Polynomový model, příklad

- Nechť zpráva $D = 10011_2$, tedy $D(X) = X^4 + X + 1$
- Nechť 3-bitový klíč má reprezentaci
  $G(X) = X^2 + X + 1$ (tj. binárně 111_2),
  pak FCS bude 2-bitový:

✓ $x^2(x^4+x+1)/x^2+x+1=$, tj.

$$x^6 + 0 + 0 + x^3 + x^2 + 0 + 0 \;/\; x^2 + x + 1 = x^4 + x^3 + 1$$
$$x^6 + x^5 + x^4$$
$$\text{———————}$$
$$x^5 + x^4 + x^3 + x^2 + 0 + 0$$
$$x^5 + x^4 + x^3$$
$$\text{———————}$$
$$x^2 + 0 + 0$$
$$x^2 + x + 1$$
$$\text{———————}$$
$$x + 1$$

✓ $x + 1$ je reprezentace 11_2 a kódové slovo je tedy $D = 1001111_2$

Polynomový model, příklad

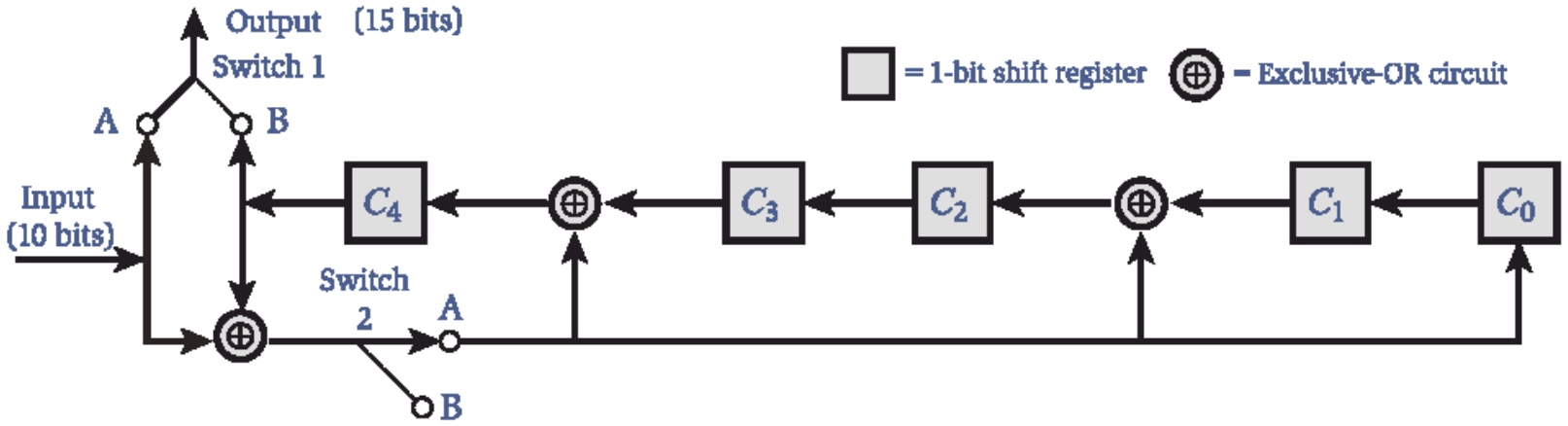
- takže rámec $T = D + FCS = 1001111_2$ pak reprezentuje polynom $T(X) = X^6 + X^3 + X^2 + X + 1$
- a polynom $T(X) = X^6 + X^3 + X^2 + X + 1$ je dělitelný klíčem $G(X) = X^2 + X + 1$ bezezbytku

Schopnosti detekce chyb

- Platí, že při použití CRC jsou detekovatelné (bez důkazu)
  - ✓ všechny 1bitové chyby pokud $G(X)$ má více než 1 nenulový term
  - ✓ všechny liché počty chyby pokud $G(X)$ obsahuje činitel $X + 1$
  - ✓ všechny dávky chyb pokud délka dávky nepřevýší $n - k$
  - ✓ s velkou pravděpodobností úspěchu všechny dávkové chyby s délkou větší než stupeň polynomu, . . .
- Polynom CRC-12: $G(X) = X^{12} + X^{11} + X^3 + X^2 + X + 1$ bude detekovat všechny dávkové chyby s lichým počtem chybných bitů, všechny dávkové chyby kratší nebo rovné 12 bitů a 99,97 % chyb delších než 12 bitů
- CRC kódy jsou velmi užitečné při detekci chyb při přenosu dlouhých rámců

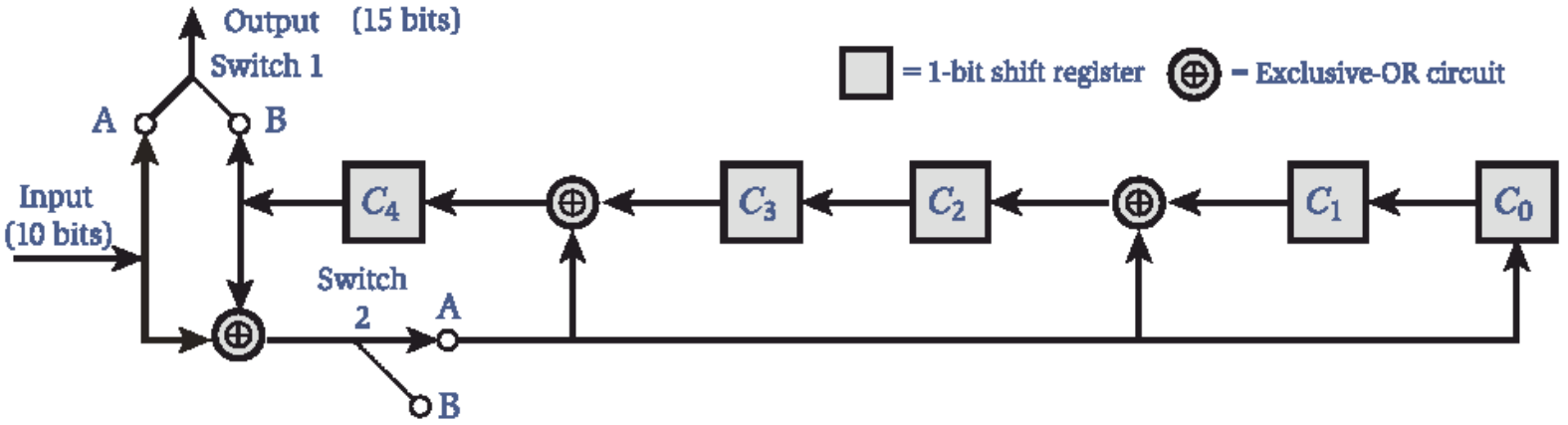
CRC, digitální logika

- ilustrace zapojení generátoru FCS (pro klíč $X^5 + X^4 + X^2 + 1$)
- počet pamětí v generátoru udává řád nejvyšší klíče (5)
- umístění hradel XOR udávají ostatní koeficienty klíče
  (X^4 a X^2 říkají, že jsou na výstupech hradel C_3 a C_1,
  1 říká, že hradlo se uplatňuje na každý vstup)



CRC, digitální logika, vysílač

- počátečně jsou paměti posuvného registru vynulované
- při vstupu k-bitové zprávy jsou přepínače v poloze A
- k-bitová zpráva, dělenec, vstupuje do registru krok po kroku
- po ukončení vstupu zprávy registr obsahuje FCS –
  zbytek po dělení zprávy dělícím polynomem (klíčem)
- přepínače se pro výstup FCS nastaví do polohy B



CRC, digitální logika, přijímač

- použije se stejná logika
- každý bit přicházející zprávy se vsouvá do registru
- pokud je přenos bezchybný,
  po přijetí dat obsahuje registr FCS
- přijetím bitů FCS se registr vynuluje

Řízení toku dat

- vysílač by neměl vysílat větší objemy dat,
  než je přijímač schopen zpracovat,
  nemá-li docházet ke ztrátám dat
  - ✓ jedná se o problém kapacit vyrovnávacích pamětí přijímače
  - ✓ rámce jsou vysílané sekvenčně
  - ✓ rámec se vysílá po dobu TT, *Transmission Time*
  - ✓ každý bit rámce se šíří médiem od vysílače k přijímači po dobu PT, *Propagation Time*
- Rámec:

- Protokoly pro řízení toku dat
  - ✓ Stop and Wait
  - ✓ Sliding Windows

Chybové řízení, typy chyb

- Porušené rámce
  - ✓ Rámec přijímač rozpoznal, ale detekoval v něm chybné bity
  - ✓ Rámec obsahuje pole pro uvedení hodnoty kódu použitého pro detekci výskyty chyby během přenosu
  - ✓ lze použít detekci chyb kontrolou parity
  - ✓ lze použít detekci chyb kódy typu *Cyclic Redundancy Check*
- Ztracené rámce
  - ✓ přijímač nerozpoznal rámec
  - ✓ a chybu odhalí až z narušené posloupnosti pořadových čísel rámců

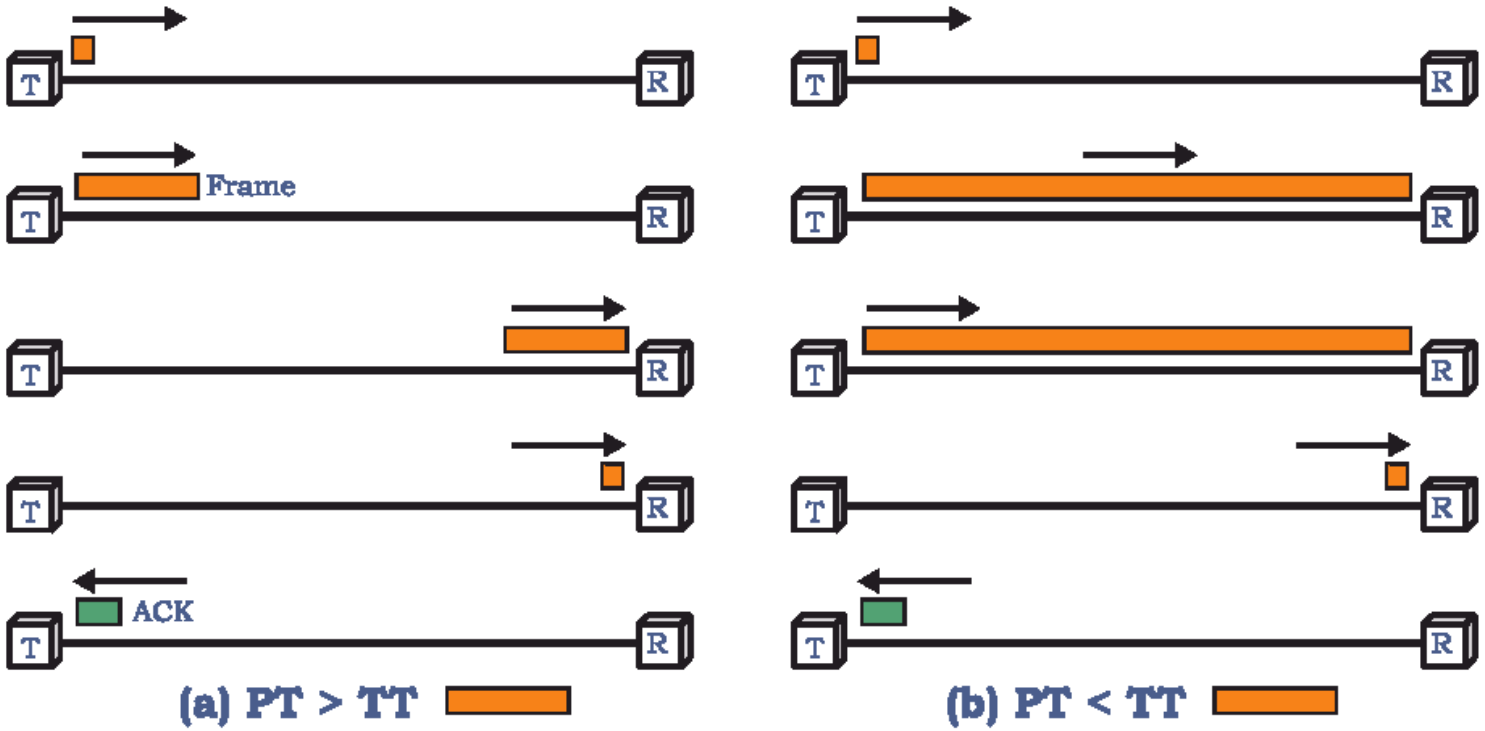
Protokol řízení toku dat, Stop-and-Wait

- Vysílač smí vyslat další rámec pouze když přijme od přijímače potvrzení práva vyslat rámec (ACK, *Acknowledgment*)
- Aplikace řízení toku dat typu Stop-and-Wait na vysílání „dlouhých" zpráv nemusí být efektivní
  - ✓ Pokud $PT > TT$

    (případ vysokých rychlostí přenosu dat a velkých vzdáleností), pak valná část spoje po dlouhou dobu neobsahuje žádné signálové prvky
  - ✓ I v případě $PT < TT$ pro rámce,

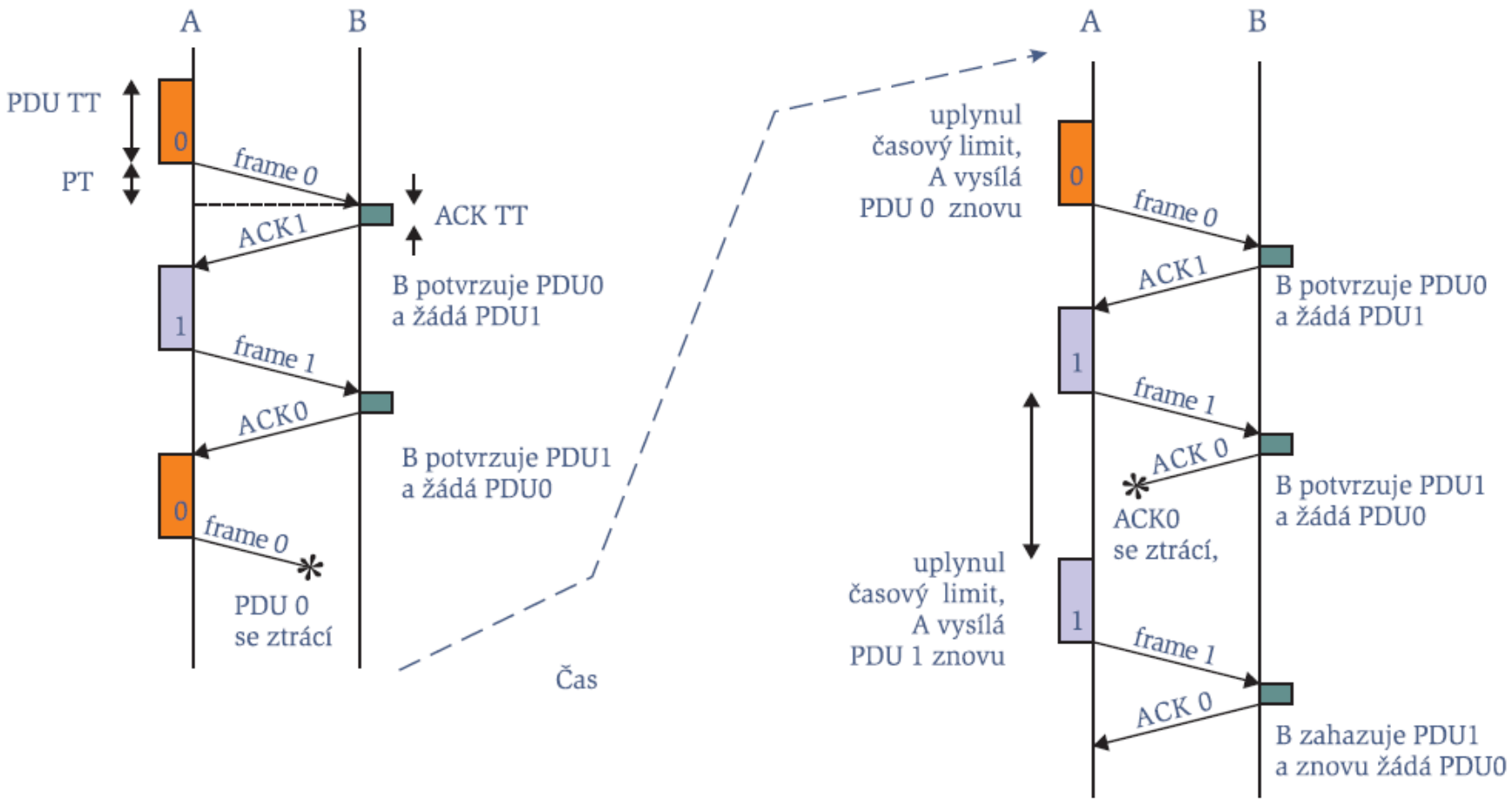
    pro zpětné vysílání ACK opět vesměs platí $PT > TT$

Využití spoje, Stop-and-Wait



Protokoly typu *Stop-and-Wait* nejsou efektivní při současných rychlostech vysílání (i Gb/s až Tb/s) a délkách spojů (např. satelity, 36 000 km)

Příklad protokolu Stop-and-Wait

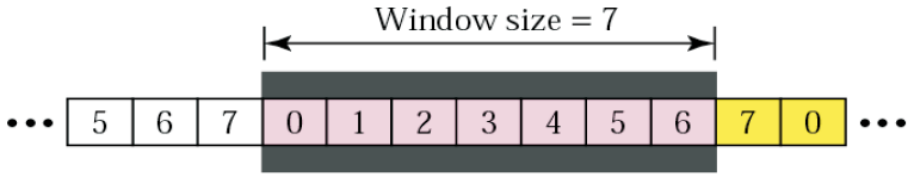


Protokol řízení toku dat, Sliding Window

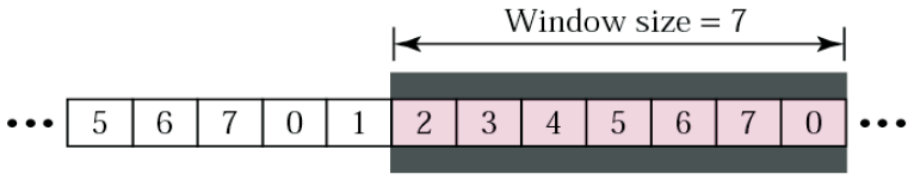
- Vysílač může vyslat až W rámců bez čekání na jejich indiviuální potvrzování
- Po vyslání W rámců bez čekání na jejich indiviuální potvrzování smí vysílač vysílat dalších až W rámců po té, co získá od přijímače potvrzení správného přenosu až W rámců
- Každý rámec je číslovaný, potvrzení (ACK) obsahuje číslo rámce. Interval pořadových čísel je vymezen bitovou šířkou pole čísla v rámci k
  - ✓ Rámce jsou číslovány $\boldsymbol{modulo}\ \mathbf{2}^{k}$, pro $k = 3 : 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, \ldots$
- Koncept sliding windows používají dva protokoly pro DLC: Go-Back-N ARQ a Selective Repeat ARQ

Go-Back-N, vysílač

- Vyslané rámce uchovává vysílač ve vyrovnávacích pamětech dokud nezíská potvrzení jejich bezchybného přenosu
- Počet vyrovnávacích pamětí vymezuje okno, *window*
- Při intervalu čísel rámců $< 0, 2^k >$ je šířka okna $2^k - 1$
- Typicky je šířka okna konstatní
- Okno ,,klouže/plave'' po posloupnosti přenášených rámců



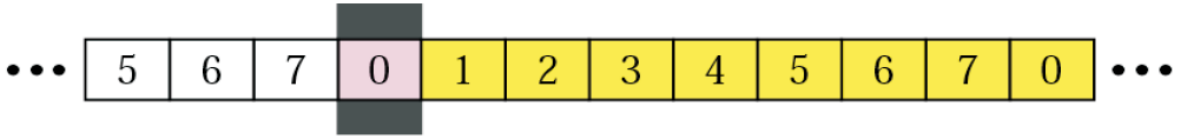
a. Before sliding



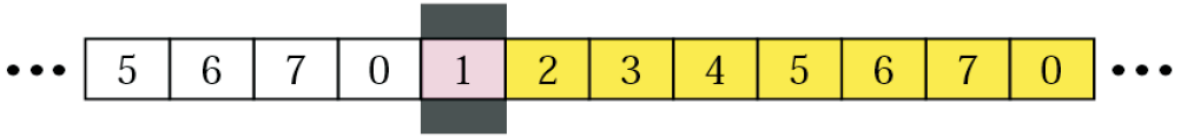
b. After sliding two frames

Go-Back-N, přijímač

- Rozměr okna přijímače je vždy 1
- Přijímač může přijmout pouze další rámec v pořadí vysílání rámců
- Rámec, který přijde mimo toto pořadí, se ignoruje a bude vyslán znovu
- Klouzání okna po posloupnosti rámců



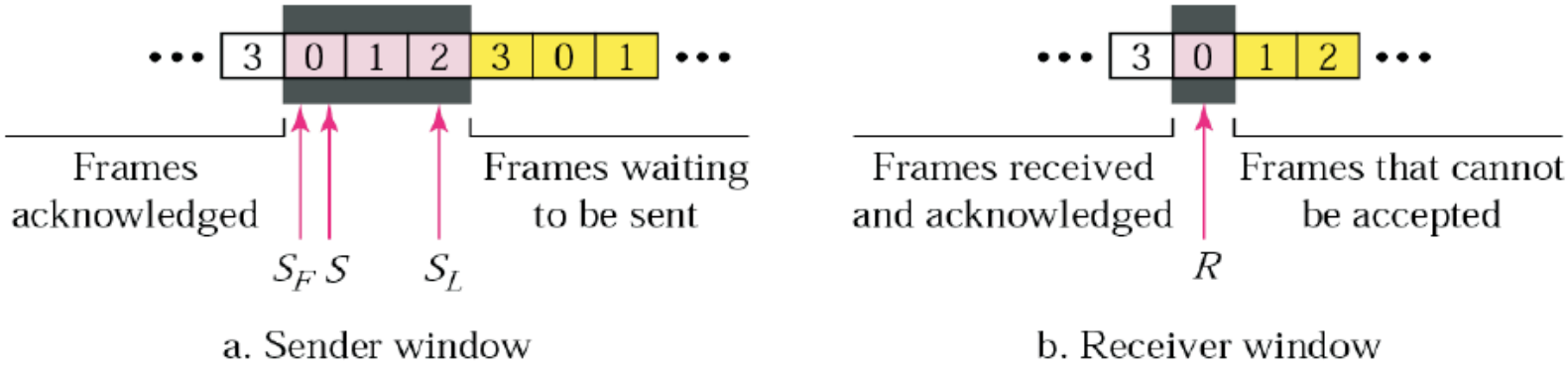
a. Before sliding



b. After sliding

Go-Back-N, řídící proměnné

- vysílač
  - ✓ S, pořadové číslo posledně vyslaného rámce
  - ✓ S_F, pořadové číslo prvního rámce v okně
  - ✓ S_L, pořadové číslo posledního rámce v okně
  - ✓ Platí $S_L - S_F + 1 = W$, kde W je šířka okna
- přijímač
  - ✓ R, pořadové číslo očekávaného rámce (ostatní přijaté se ignorují)



Go-Back-N, potvrzování

- časové hlídání
  - ✓ časovače se udržují na straně vysílače,
  - ✓ spouští se časovač pro každý vyslaný rámec
  - ✓ pokud vysílač neobdrží do uplynutí limitu potvrzení rámce, vysílá znovu všechny rámce počínaje rámcem s překročeným časovým limitem čekání na potvrzení
- potvrzování
  - ✓ poškozené rámce nebo rámce přijaté mimo pořadí přijímač ignoruje
  - ✓ jako indikaci úspěšného přijetí rámce přijímač posílá vysílači pozitivní potvrzení, ACK
  - ✓ přijímač nemusí potvrzovat přijaté rámce individuálně, může posílat kumulativní potvrzení několika rámců (až do šíře okna)
  - ✓ potvrzení přijetí rámce n udává číslo očekávaného rámce, tj. $n + 1$

Vylepšení Sliding Window

- Přijímač může rámce potvrzovat bez povolení dalšího přenosu (rámcem s příkazem Receive Not Ready, RNR)
- Pro pokračování přenosu musí přijímač vyslat normální potvrzení (rámec s příkazem Receive Ready, RR)
- V duplexním režimu přijímač používá pro potvrzování piggybacking:
  - ✓ Pokud má připravena k vyslání data a chce přijatá data potvrzovat, pošle datový rámec ve kterém uvede v potvrzovacím poli potvrzovací číslo
  - ✓ Pokud nemá připravena data pro vyslání, použije pro zaslání potvrzovacího čísla potvrzovací rámec, tj. RR nebo RNR
  - ✓ Pokud má připravena k vyslání data a nechce přijatá data potvrzovat, pošle datový rámec ve kterém zopakuje v potvrzovacím poli poslední potvrzovací číslo

Go-Back-N, poškozený rámec

- Založeno na principu *sliding window*
- Pokud při příjmu nedojde k chybě dat,
  rámec ACK uvádí číslo očekávaného rámce
- Pro pozitivní potvrzování lze použít
  - ✓ samostatný rámec RR
  - ✓ pigybacking – potvrzování v datových rámcích protistrany
- Pokud dojde k chybě dat – posílá se negativní potvrzení rámce
  - ✓ pomocí rámce REJ, *Reject*
  - ✓ poškozený rámec (i) přijímač potvrzuje negativně (REJ i) a
  - ✓ poškozený rámec i a všechny následně přijaté a
    dosud nepotvrzené rámce likviduje, dokud správně nepřijme rámec i
  - ✓ vysílač se musí vrátit k rámci i
    a tento a všechny následující vyslané rámce vyslat znovu

Go-Back-N, ztracený rámec

- vysílač vyšle rámec i a rámec $i + 1$ a rámec i se ztratí
  - ✓ přijímač při přijetí rámec $i + 1$ rozpozná chybu v pořadovém čísle
  - ✓ přijímač si vyžádá zopakování přenosu rámců počínaje rámcem i a rámců po vyslaných po rámci i – pomocí REJ i
- vysílač vyšle rámec i a ten se ztratí
  - ✓ přijímač nic nedostane, nic proto žádnou formou nepotvrzuje
  - ✓ vysílači uplyne časový limit a vyšle RR s indikací žádosti o partnerovo potvrzení (tzv. P bit = 1)
  - ✓ přijímač reaguje vysláním RR s číslem očekávaného rámce (i)
  - ✓ vysílač zopakuje vyslání rámce i
  - ✓ nebo vysílač automaticky vysílá rámec i po uplynutí přijímacího časového limitu

Go-Back-N, ztracené potvrzení

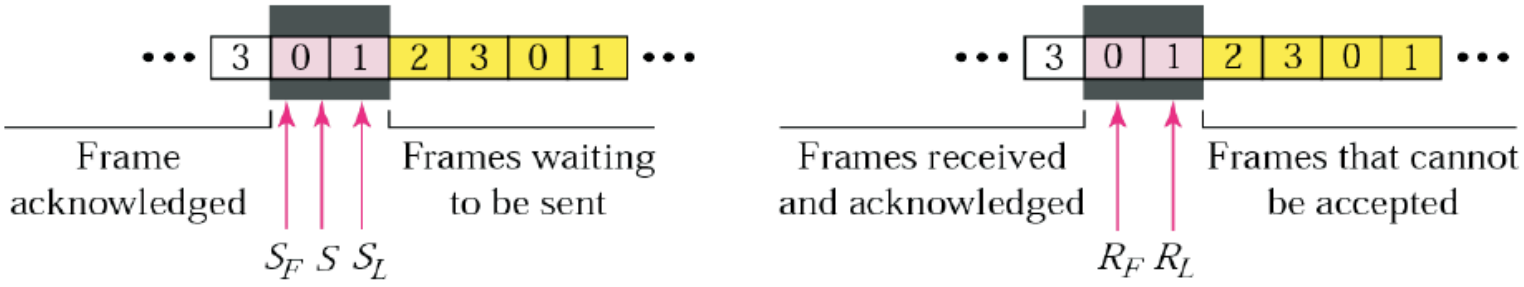
- přijímač po přijetí rámec i posílá potvrzení RR $i+1$
- potvrzení RR $i+1$ se ztratí
- potvrzení jsou kumulativní, může přijít RR $i+n$ dříve než uplyne časový limit pro rámec i
- pokud uplyne časový limit, vysílač i vysílá RR s P-bitem
  - ✓ toto se opakuje po uplynutí časového limitu reakce na P-bit až do vyčerpání limitu počtu opakování – pak se dělá *reset*
- ztratí se potvrzení REJ
  - ✓ stejné situace jako při ztrátě rámce – *vysílač vyšle rámec* i *a ten se ztratí*

Go-Back-N, důvody pro Selective repeat

- Velmi jednoduchý proces na straně přijímače
  - ✓ přijímač sleduje stav v jediné proměnné
  - ✓ rámce přijaté mimo pořadí není třeba držet ve vyrovnávacích pamětech (likvidují se)
- Protokol neefektivní na spojích s velkou úrovní šumu
  - ✓ časté opakované vysílání více rámců
  - ✓ snižuje se dostupná efektivní šířka pásma, přenos je pomalejší
- řešení pro spoje s velkou úrovní šumu – Selective repeat ARQ
  - ✓ opakují se pouze chybně přenesené rámce
  - ✓ zpracování na straně přijímače je složitější

Selective Repeat, okna vysílače a přijímače

- Vysílač
  - ✓ shodná struktura s vysílačem s protokolem *Go back N*
  - ✓ jediná odlišnost – rozměr okna = polovina 2^m
- Přijímač
  - ✓ rozměr okna = polovina 2^m
  - ✓ hraniční meze okna vymezují proměnné R_F a R_L
  - ✓ Přijímač může zasílat i negativní potvrzení, NAK
    - sdělení čísla poškozeného rámce před uplynutím časového limitu



a. Sender window

b. Receiver window

Číslování rámců modulo N

- jaká musí být velikost okének, aby nedocházelo k záměně?
- Stop-and-wait
  - stačí dvě hodnoty (0,1) pro rozpoznání tj. modulo 2
- Go-Back-N
  - při velikosti vysílacího okénka K stačí N=K+1
  - při ztrátě všech potvrzení v okénku musí přijímač rozpoznat, že se jedná o rámec v novém okénku
- Selective Repeat
  - při vysílání okénko nepřekročíme, ale můžeme ho překročit při potvrzování $\Rightarrow$ N=2K+1

Efektivnost metod řízení toku dat

- Stop-And-Wait ARQ, rychlost přenosu dat 1 Mb/s, doba obrátky bitu 20 ms, rámec délky 1 000 b
  - ✓ systém by mohl vyslat do získání potvrzení $10^6 \times 20 \times 10^{-3} = 20\,000$ bitů
  - ✓ efektivnost $1\,000/20\,000 = 0{,}05$, tj. 5 %
- Go-Back-N ARQ, rychlost přenosu dat 1 Mb/s, doba obrátky bitu 20 ms, rámec délky 1 000 b, šířka okna 15 rámců
  - ✓ efektivnost $15\,000/20\,000 = 0{,}75$, tj. 75 %
  - ✓ v případě poruchového spoje ovšem efektivnost může silně klesat v důsledkou opakovaného vysílání rámců