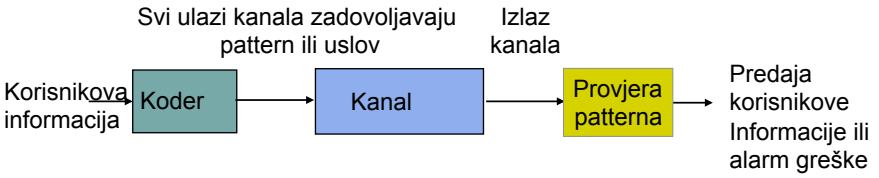
Mehanizmi pouzdanog prenosa

Kontrola greške

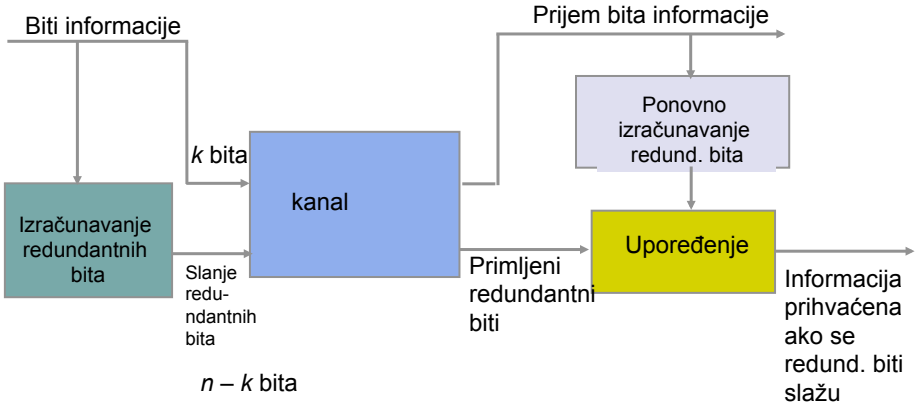
- Telekomunikacioni sistemi unose grešku
- Aplikacije zahtijevaju određeni nivo pouzdanosti
  - Aplikacije prenosa podataka zahtijevaju prenos bez greške
  - Govor & video aplikacije toleriše određeni nivo greške
- Kontrola greške se koristi kada prenosni sistem ne zadovoljava zahtjeve aplikacije
- Kontrola greške obezbjeđuje da se podaci do određenog nivoa prenose bez greške
- Dva osnovna principa:
  - ***Detekcija* greške & retransmisija** (ARQ)
  - "**Forward error *correction*"** (FEC)
- Najčešće se realizuje na nivoima linka i transporta

Ključna ideja

- Svi prenošeni blokovi podataka (“kodne riječi”) treba da zadovoljavaju šablon (pattern)
- Ako primljeni blok ne zadovoljava šablon, znači da se pojavila greška
- Redundansa: Samo podskup svih mogućih blokova može biti kodna riječ
- Problem nastaje kada kanal transformiše kodnu riječ u drugu kodnu riječ!!!!!!



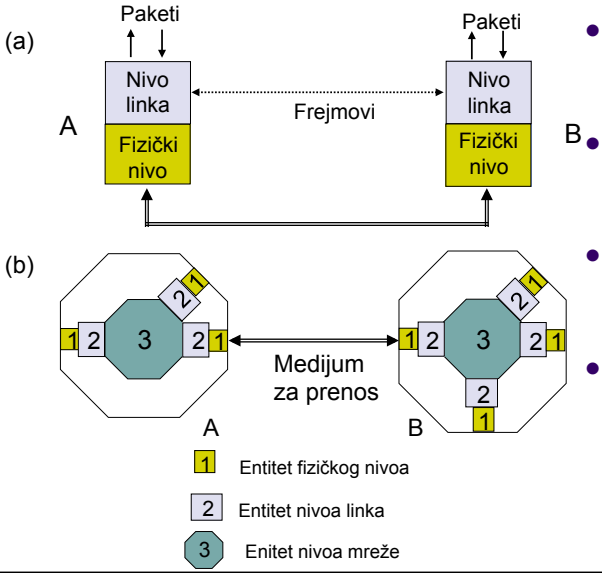
Redundantni biti & detekcija greške



Poređenje pristup End-to-End i Hop-by-Hop

- Kontrola greške (kao i kontrole protoka i zagušenja) može biti implementirana u okviru protokola tako da se obavlja
  - End-to-end (od kraja do kraja mreže)
  - Hop-by-hop (na svakom linku u mreži)

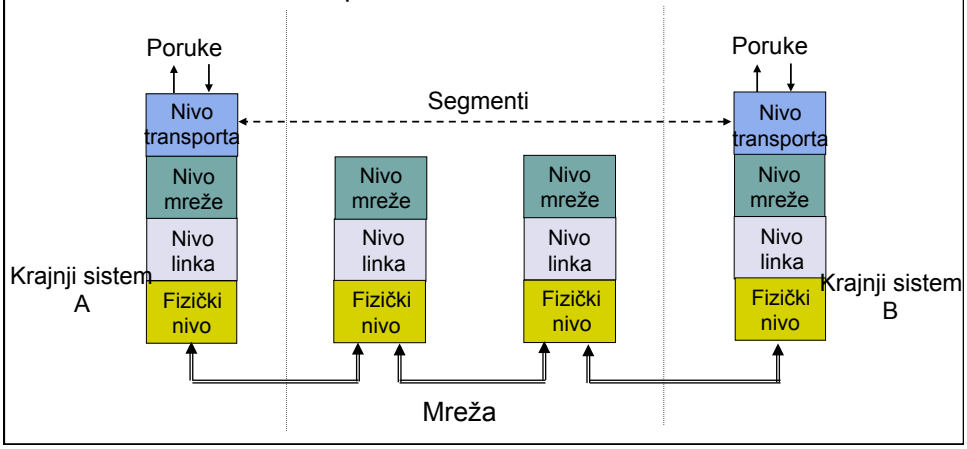
Kontrola greške na nivou linka



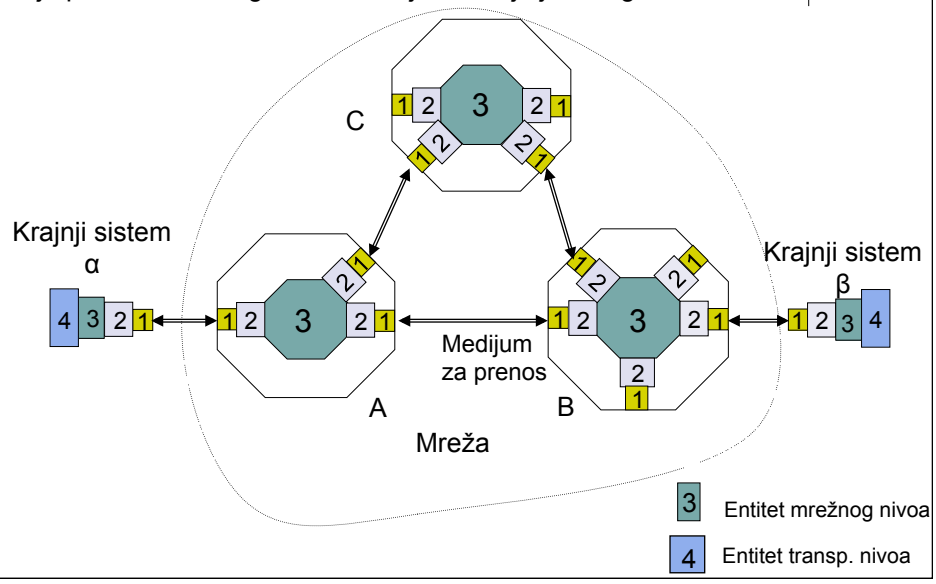
- Nivo linka funkcioniše između direktno povezanih uređaja
- Frejmovi mogu biti oštećeni, izgubljeni ali pristižu po redosledu
- Nivo linka obavlja provjeru greške i retransmisiju
- Obezbjeđuje prenos paketa bez greške između dva sistema

Kontrola greške na transportnom nivou

- Protokol transportnog nivoa (npr. TCP) šalje segmente preko mreže i obavlja kontrolu greške od kraja do kraja i retransmisiju ako se pojavi potreba
- Mreža se smatra nepouzdanom

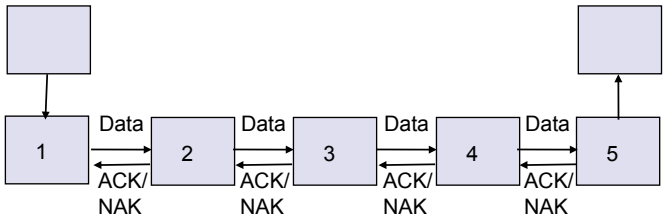


- Segmenti mogu previše kasniti, mogu biti izgubljeni, mogu doći van sekvence jer paketi putuju različitim putanjama zbog toga je posao kontrole greške od kraja do kraja je mnogo teži



Poređenje

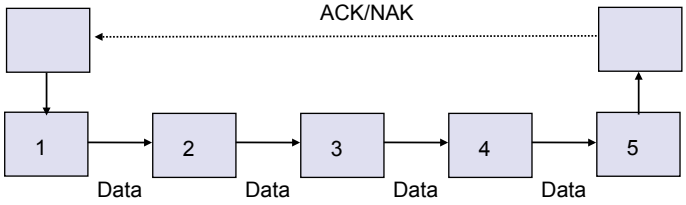
Hop-by-hop



Hop-by-hop ne garantuje E2E ispravnost

Brži oporavak

End-to-end



Mrežna čvorišta su jednostavna

Skalabilnije rješenje ako je inteligencija na obodu mreže

Tehnike detekcije greške

- Šeme ponavljanja
- Šeme provjere parnosti
- Suma provjere (checksum)
- Ciklična provjera redundanse (Cyclic redundancy check)
- Provjere bazirane na Hammingovom kodu
- Hash funkcija
- Horizontalne i vertikalne provjere redundanse
- Šeme polarnosti

Tehnike detekcije greške

Šeme ponavljanja

- Kodna šema koja ponavlja istovjetne bite nekoliko puta.
- Veoma jednostavna metoda koja ima veoma loše performanse u slučaju kanala sa bijelim Gausovim šumom.
- Oznaka (r,1) znači da se svaki bit ponavlja r puta.
- Npr., u slučaju (3,1) koda ponavljanja, povorka bita m = 101001 se kodira u

c = 111000111000000111.

- Detekcija određenog bita se ogleda u brojanju primljenih vrijednosti i utvrđivanju koja se češće pojavljuje
- Npr., neka je u slučaju koda (3,1) primljena sekvenca c = 110001111. Dekodirana poruka je m = 101, zato što se u prva tri bita češće pojavljue 1, druga dva 0, a u treća tri ponovo 1.
- Sa povećanjem r u kanalima sa bijelim Gausovim šumom dobijamo lošije performanse, dok se u kanalu sa fedingom ova šema ponaša znatno bolje.

Tehnike detekcije greške

Provjera parnosti

- Primjenjuje se za provjeru parnosti k informacionih bita
- Sve kodne riječi imaju paran (neparan) broj jedinica
- Prijemnik provjerava da li je broj jedinica paran (neparan)
  - Svi oblici greške koji mijenjaju kodnu riječ tako da je broj jedinica neparan (paran) se mogu detektovati
  - Svi oblici greške koji mijenjaju kodnu riječ tako da je broj jedinica paran (neparan) se ne mogu detektovati
- Bit parnosti se koristi u ASCII kodu

Informacioni Biti: $b_1, b_2, b_3, \ldots, b_k$

Bit provjere: $b_{k+1} = b_1 + b_2 + b_3 + \ldots + b_k$ modulo 2

Kodna riječ: $(b_1, b_2, b_3, \ldots, b_k, b_{k+1})$

Tehnike detekcije greške

Primjer koda jednostruke parnosti

- Informacija (7 bita): (0, 1, 0, 1, 1, 0, 0)
- Bit parnosti: $b_8 = 0 + 1 + 0 + 1 + 1 + 0 + 0 = 1$
- Kodna riječ (8 bita): (0, 1, 0, 1, 1, 0, 0, 1)
- Ako se pojavi jednostruka greška na bitu 3 : (0, 1, 1, 1, 1, 0, 0, 1)
  - Broj jedinica je 5, neparan broj
  - Greška detektovana
- Ako se greška javlja na 3 i 5 bitu:
- (0, 1, 1, 1, 0, 0, 0, 1)
  - Broj jedinica je 4, paran
  - Greška se ne može detektovati!!!!!!!

Tehnike detekcije greške

Koliko je dobra tehnika jednostruke parnosti?

- *Redundansa*: Jednostruki kod provjere parnosti dodaje 1 redundantni bit na *k* informacionih bita:

  zaglavlje = $1/(k + 1)$
- *Pokrivanje*: sve greške sa neparnim brojem grešaka se mogu detektovati
  - Pattern greške je binarni $(k + 1)$-struki blok sa jedinicama gdje se javljaju greške i nulama tamo gdje se one ne javljaju
  - Od 2^{k+1} binarnih $(k + 1)$-strukih blokova, ½ je neparno, tako da se 50% patterna greške detektuje
- Da li je moguće detektovati veći broj grešaka ako se doda više bita provjere?
- Da, sa pravim kodovima

Tehnike detekcije greške

Šta ako su greške slučajne?

- Mnogi prenosni kanali unose greške slučajno, nezavisne jedne od drugih, sa vjerovatnoćom p
- Neke greške su vjerovatnije od drugih:

$$P[10000000] = p(1-p)^7 = (1-p)^8\left(\frac{p}{1-p}\right) \quad \text{i}$$

$$P[11000000] = p^2(1-p)^6 = (1-p)^8\left(\frac{p}{1-p}\right)^2$$

- U bilo kojem valjanom kanalu $p < 0.5$, tako da je $(p/(1-p)) < 1$
- Izgleda da su patterni sa jednom greškom vjerovatniji od patterna sa dvije greške itd.
- Koliko iznosi vjerovatnoća da se pojavi greška koja se ne može detektovati?

Tehnike detekcije greške

Jednostruki kod provjere parnosti sa slučajnim greškama

- Ne mogu se detektovati greške sa parnim brojem pogrešnih bita:

P[nedetkovanja greške] = P[pojava greške koja se ne može detektovati]
= P[greške sa parnim brojem jedinica]

$$= \binom{n}{2} p^2(1-p)^{n-2} + \binom{n}{4} p^4(1-p)^{n-4} + \ldots$$

- Primjer: Izračunati za $n = 32$, $p = 10^{-3}$

$$P[\text{nedetkovanja greške}] = \binom{32}{2}(10^{-3})^2(1-10^{-3})^{30} + \binom{32}{4}(10^{-3})^4(1-10^{-3})^{28} + \ldots$$

$$\approx 496\,(10^{-6}) + 35960\,(10^{-12}) \approx 4.96\,(10^{-4})$$

- Za ovaj primjer, približno jedna od 2000 greški se ne detektuje

Tehnike detekcije greške

Dvodimenzionalna provjera parnosti

- Više bita parnosti poboljšava detekciju
- Organizuje informaciju u kolone
- Dodaje jedan bit parnosti svakoj koloni
- Dodaje finalnu kolonu "parnosti"
- Korišćena je u ranim sistemima za kontrolu greške

1	0	0	1	0	0
0	1	0	0	0	1
1	0	0	1	0	0
1	1	0	1	1	0
1	0	0	1	1	1

Poslednja kolona se sastoji od bita provjere za svaku vrstu

Donja vrsta sadrži bite provjere za svaku kolonu

Mogućnost detekcije greške

1	0	0	1	0	0
0	**0**	0	0	0	1 ←
1	0	0	1	0	0
1	1	0	1	1	0
1	0	0	1	1	1

Jedna greška

1	0	0	1	0	0
0	**0**	0	0	0	1 ←
1	0	0	1	0	0
1	**0**	0	1	1	0 ←
1	0	0	1	1	1

Dvije greške

1	0	0	1	0	0
0	**0**	0	**1**	0	1
1	0	0	1	0	0
1	**0**	0	1	1	0 ←
1	0	0	1	1	1

Tri greške

1	0	0	1	0	0
0	**0**	0	**1**	0	1
1	0	0	1	0	0
1	**0**	**0**	1	0	0
1	0	0	1	1	1

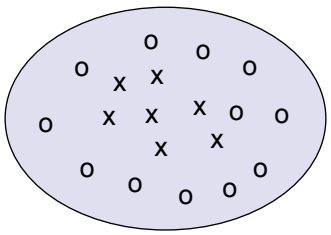
Četiri greške se ne detektuju!!!!!

1, 2, ili 3 greške se uvijek mogu detektovati;
Sve greške sa više od 4 pogrešna bita se ne detektuju

Strelice ukazuju na pogrešne bite provjere

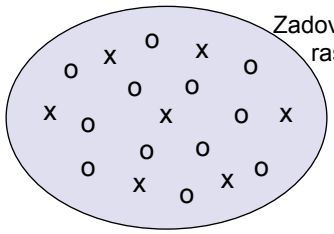
Šta je dobar kod?

- Mnogi kanali izazivaju greške koji imaju manji broj pogrešnih bita
- Ove greške mapiraju prenešenu kodnu riječ u obližnji *n*-struki blok
- Ako su kodne riječi međusobno bliske tada se javlja greška u detekciji
- Dobri kodovi bi trebali maksimizirati rastojanje između kodnih riječi



Mala rastojanja

x = kodne riječi
o = ne kodne riječi



Zadovoljavajuća rastojanja

Tehnike detekcije greške

Suma provjere "Checksum"

- Više Internet protokola (npr. IP, TCP, UDP) koriste bite provjere za detektovanje grešaka u *IP zaglavlju* (ili u zaglavlju i podacima TCP/UDP)
- Suma provjere se izračunava za određeni dio paketa (najčešće samo zaglavlje) i upisuje u posebno polje zaglavlja.
- Suma provjere se ponovo izračunava na mrežnim čvorištima i krajnjoj destinaciji, tako da je izabran algoritam koji je lak za implementaciju u softveru
- Neka se zaglavlje sastoji od L, 16-bitnih riječi, $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{L-1}$
- Algoritam izračunava 16-bitnu sumu provjere $\mathbf{b}_L$

Tehnike detekcije greške

Izračunavanje sume provjere

Suma provjere $\mathbf{b}_L$ se izračunava na sledeći način:

- Svaka 16-bitna riječ se tretira kao prirodan broj. Zatim se ti brojevi saberu i dobija se

  $\mathbf{x} = \mathbf{b}_0 + \mathbf{b}_1 + \mathbf{b}_2 + \ldots + \mathbf{b}_{L-1}$ modulo $2^{16}-1$

- Suma provjere je drugi komplement od sume, tj.

  $\mathbf{b}_L = -\mathbf{x}$ modulo $2^{16}-1$

Na taj način, sadržaj zaglavlja i polje sume provjere moraju zadovoljavati sledeći ***pattern***:

$\mathbf{0} = \mathbf{b}_0 + \mathbf{b}_1 + \mathbf{b}_2 + \ldots + \mathbf{b}_{L-1} + \mathbf{b}_L$ modulo $2^{16}-1$

- Izračunavanje sume provjere se obavlja softverski

Tehnike detekcije greške

- Napomena
  - Kada se sabiraju brojevi, prenos sa najznačajnijeg bita se dodaje rezultatu

	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
prenos	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
suma	1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
komplement	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1
Checksum	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Tehnike detekcije greške

Ciklična provjera redundanse (Cyclic Redundancy Check)

- Koriste se polinomi umjesto vektora za predstavljanje kodnih riječi
- Aritmetika polinoma umjesto kontrolnih suma
- Implementiraju se korišćenjem kola sa pomjeračkim registrima
- Većina standarda u komunikacijama podataka koristi ovu vrstu kodova za detekciju greške
- Ovi kodovi predstavljaju odličnu bazu za realizaciju moćnih metoda za korekciju greške
- Originalnoj poruci odgovara polinom koji se dijeli sa generišućim polinomom. Ostatak pri dijeljenju se upisuje u CRC polje.
- Na prijemu se polinom koji odgovara primljenoj poruci dijeli generišućim polinomom. Ostatak pri dijeljenju se upoređuje sa sadržajem CRC polja. Ako nema razlika prenos je bio uspješan.
- Ovi kodovi se zovu još i polinomijalni kodovi.

Binarna aritmetika polinoma

- Binarni vektor se mapira u polinome

$$(i_{k-1}, i_{k-2}, \ldots, i_2, i_1, i_0) \rightarrow i_{k-1}x^{k-1} + i_{k-2}x^{k-2} + \ldots + i_2x^2 + i_1x + i_0$$

Sabiranje:

$$(x^7 + x^6 + 1) + (x^6 + x^5) = x^7 + x^6 + x^6 + x^5 + 1$$
$$= x^7 + (1+1)x^6 + x^5 + 1$$
$$= x^7 + x^5 + 1 \quad \text{jer je } 1+1=0 \bmod 2$$

Množenje:

$$(x + 1)(x^2 + x + 1) = x(x^2 + x + 1) + 1(x^2 + x + 1)$$
$$= x^3 + x^2 + x + x^2 + x + 1$$
$$= x^3 + 1$$

Binarno dijeljenje polinoma

- Dijeljenje polinoma

djelilac

$x^6 + x^5 : x^3 + x + 1 = x^3+x^2+x$

$x^6 + x^4 + x^3$

$x^5 + x^4 + x^3$

$x^5 + x^3 + x^2$

$x^4 + x^2$

$x^4 + x^2 + x$

x = $r(x)$ ostatak

Napomena: Stepen r(x) je manji nego stepen djelioca

Tehnike detekcije greške

CRC

- Kod ima binarni *generišući polinom* stepena *n–k*

$$g(x) = x^{n-k} + g_{n-k-1}x^{n-k-1} + \ldots + g_2x^2 + g_1x + 1$$

- *k informacionih bita* definiše polinom stepena $k - 1$

$$i(x) = i_{k-1}x^{k-1} + i_{k-2}x^{k-2} + \ldots + i_2x^2 + i_1x + i_0$$

- Cilj je pronaći *polinom ostatka* sa stepenom reda $n - k - 1$ (maksimalno) tako da je

$$x^{n-k}i(x) = q(x)g(x) + r(x)$$

- Definiše se *polinom kodne riječi* stepena $n - 1$

$$\underbrace{b(x)}_{n \text{ bita}} = \underbrace{x^{n-k}i(x)}_{n \text{ bita}} + \underbrace{r(x)}_{n-k \text{ bita}}$$

Primjer: $k = 4$, $n = 7$

Generišući polinom: $g(x) = x^3 + x + 1$

Informacija: (1,1,0,0) $i(x) = x^3 + x^2$

Kodiranje: $x^3 i(x) = x^6 + x^5$

$$
\begin{array}{l}
x^3 + x^2 + x \\
\hline
x^3 + x + 1 \;)\; x^6 + x^5 \\
\quad x^6 + \quad x^4 + x^3 \\
\hline
\qquad x^5 + x^4 + x^3 \\
\qquad x^5 + \quad x^3 + x^2 \\
\hline
\qquad\quad x^4 + \quad x^2 \\
\qquad\quad x^4 + \quad x^2 + x \\
\hline
\qquad\qquad\quad x
\end{array}
\qquad
\begin{array}{l}
1110 \\
\hline
1011 \;)\; 1100000 \\
\quad 1011 \\
\hline
\qquad 1110 \\
\qquad 1011 \\
\hline
\qquad\quad 1010 \\
\qquad\quad 1011 \\
\hline
\qquad\qquad 010
\end{array}
$$

Prenošena kodna riječ:

$b(x) = x^6 + x^5 + x$

$\Longrightarrow$ $\underline{b} = (1,1,0,0,0,1,0)$

Pattern u polinomijalnom kodiranju

- Sve kodne riječi zadovoljavaju sledeći **pattern**:

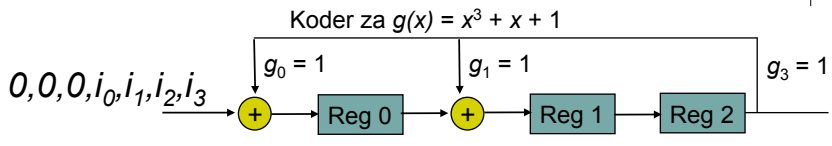
$$b(x) = x^{n-k} i(x) + r(x) = q(x)g(x) + r(x) + r(x) = q(x)g(x)$$

- Sve kodne riječi su multipli od *g(x)!!!!!!*
- Prijemnik dijeli primljene n-torke sa g(x) i provjerava da li je ostatak nula
- Ako ostatak nije nula, tada primljena n-torka nije kodna riječ

Implementacija pomjeračkog registra

1. Prijem informacionih bita $i_{k-1}, i_{k-2}, \ldots, i_2, i_1, i_0$
2. Dodavanje $n - k$ nula informacionim bitima
3. Uvesti sekvencu u kolo pomjeračkog registra koje obavlja dijeljenje polinoma
4. Poslije n shift-ova, pomjerački registar sadrži ostatak

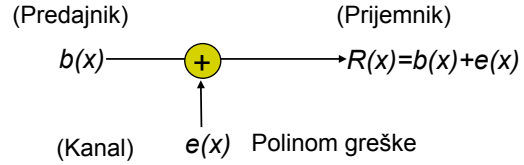
Kolo za dijeljenje



Clock	Ulaz	Reg 0	Reg 1	Reg 2
0	-	0	0	0
1	$1 = i_3$	1	0	0
2	$1 = i_2$	1	1	0
3	$0 = i_1$	0	1	1
4	$0 = i_0$	1	1	1
5	0	1	0	1
6	0	1	0	0
7	0	**0**	**1**	**0**
	Biti provjere:	$r_0 = 0$	$r_1 = 1$	$r_2 = 0$

$\Longrightarrow r(x) = x$

Greške koje se ne mogu detektovati



- *e(x)* ima jedinice na mjestima greške, a nule na drugim mjestima
- Prijemnik dijeli primljeni polinom *R(x)* sa *g(x)*
- Problem: ako je *e(x)* multipl od *g(x)*, i kodna riječ različita od nule tada

  $R(x) = b(x) + e(x) = q(x)g(x) + q'(x)g(x)$
- *Skup grešaka koje se ne mogu detektovati je skup nenultih polinoma.*

Dizajniranje dobrih polinomijalnih kodova

- Treba izabrati generišući polinom tako da najvjerovatniji oblici greške ne budu multipli od *g(x)*
- *Detektovanje pojedinačnih grešaka*
  - $e(x) = x^i$ *za grešku na* i + 1 bitu
  - Ako *g(x)* ima više od jednog člana ne može dijeliti x^i *bez ostatka*
- *Detektovanje dvostrukih grešaka*
  - $e(x) = x^i + x^j = x^i(x^{j-i}+1)$ *gdje je* $j>i$
  - Ako *g(x)* ima više od jednog člana, ne može dijeliti x^i *bez ostatka*
  - Ako je g(x) *prost* polinom, ne može dijeliti x^m+1 bez ostatka za svako $m<2^{n-k}-1$ (Potrebno je obezbijediti da kodna riječ bude manja od $2^{n-k}-1$)
  - Prosti polinomi se mogu naći uz konsultaciju knjiga iz teorije kodova

Dizajniranje dobrih polinomijalnih kodova (nastavak)

- *Detekcija neparnog broja grešaka*
  - Pretpostavimo da polinomi kodnih riječi imaju paran broj jedinica, tada sve greške sa neparnim brojem pogrešnih bita se lako detektuju
  - Također, *b(x)* za *x* = 1 je 0 jer *b(x)* ima paran broj jedinica
  - To znači da *x* + 1 mora biti faktor za svaki *b(x)*
  - Treba odabrati *g(x)* = (*x* + 1) *p(x)* gdje je *p(x)* prost polinom

Standardni generišući polinomi

CRC = cyclic redundancy check

- CRC-8:
  - $= x^8 + x^2 + x + 1$ ATM
- CRC-16:
  - $= x^{16} + x^{15} + x^2 + 1$ Bisync
  - $= (x + 1)(x^{15} + x + 1)$
- CCITT-16:
  - $= x^{16} + x^{12} + x^5 + 1$ HDLC, XMODEM, V.41
- CCITT-32: IEEE 802, DoD, V.42
  - $= x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

Tehnike korekcije greške

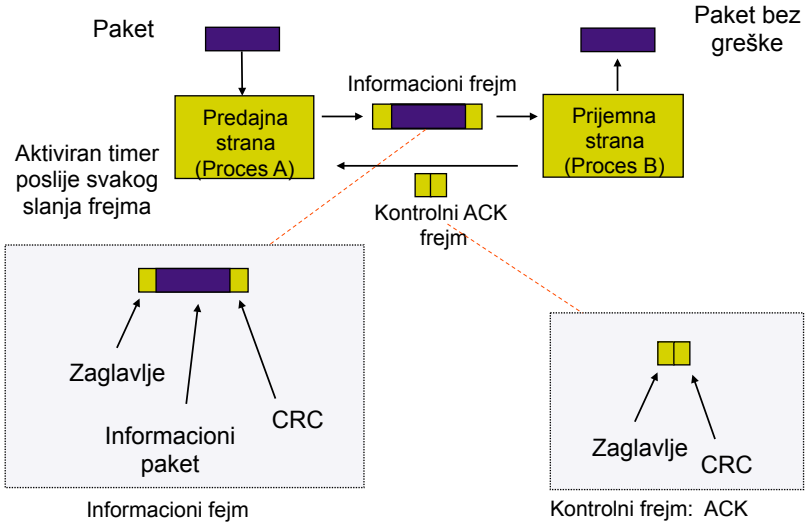
- Automatic Repeat reQuest (ARQ)
  - Stop-and-wait ARQ,
  - Go-Back-N ARQ
  - Selective Repeat ARQ.
  - Hybrid ARQ
- Kodovi za korekciju greške
  - Forward Error Correction (FEC) koriste prethodno opisane tehnike (provjera parnosti, checksum, CRC, Hammingov kod...) za detekciju greške
  - Hammingovi kodovi mogu ispraviti greške na jednom bitu i detektovati dvobitne greške
  - Za greške višeg reda se mogu koristiti konvolucioni i blok kodovi.

ARQ

- Automatic Repeat reQuest (ARQ)
  - Stop-and-wait ARQ,
  - Go-Back-N ARQ
  - Selective Repeat ARQ.
  - Hybrid ARQ (kombinacija ARQ i FEC)
- Osnovni elementi ARQ**:**
  - *Pouzdan kod za detekciju greške*
  - *ACK-ovi* (positive acknowledgments- pozitivne potvrde)
  - *NAK-ovi* **(**negative acknowlegments – negativne potvrde)
  - *Timeout mehanizam*

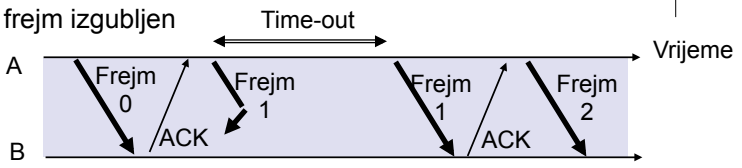
Stop-and-Wait ARQ

Poslati frejm, čekati ACK

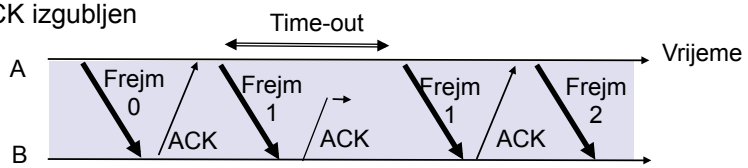


Potreba za numeracijom frejmova (brojevi u sekvenci)

(a) 1. frejm izgubljen



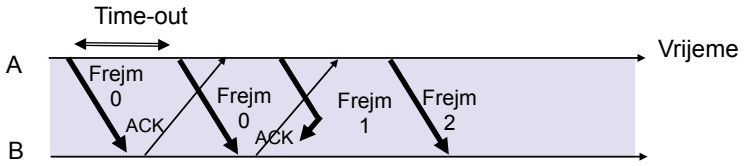
(b) ACK izgubljen



- U oba slučaja predajna strana A funkcioniše na isti način
- U slučaju (b) prijemna strana B prihvata frejm 1 dva puta
- Kako prijemna strana zna da je treći primljeni frejm, frejm 1?
- ***Treba dodati broj u sekvenci!!!!***
- S_{posl} je broj u sekvenci poslednjeg poslatog frejma

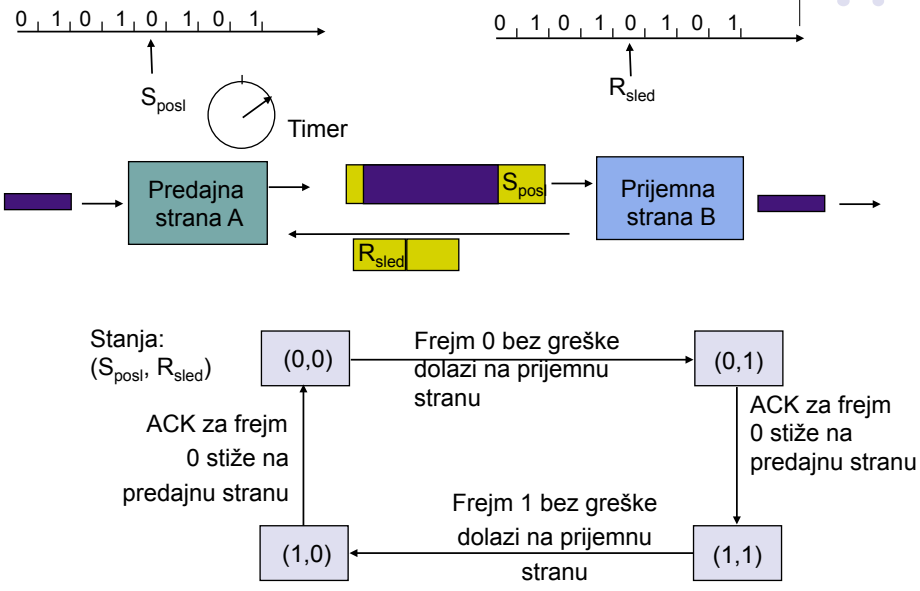
Potreba za numeracijom frejmova (brojevi u sekvenci)

(c) Prijevremeni Time-out



- Prijemna strana nije dobro razumjela dvostruki ACK,
- Drugi ACK kao da potvđuje frejm 1
- Kako da predajna strana zna da su poslata dva ACK za frejm 0?
- ***Treba dodati broj u sekvenci i u zaglavlje ACK***
- R_{sled} je broj u sekvenci frejma kojeg očekuje prijemna strana
- Implicitno potvrđuje prijem svih prethodnih frejmova

Jednobitni broj u sekvenci je dovoljan



Stop-and-Wait ARQ

Predajna strana

Aktivno stanje

- Čeka zahtjeve sa viših nivoa za slanje paketa
- Kada zahtjev stigne, šalje frejm sa ažuriranom vijednošću S_{posl} i CRC
- Prelazi u stanje čekanja

Stanje čekanja

- Čeka na ACK ili isticanje tajmera; blokira zahtjeve sa viših nivoa
- Ako istekne timeout
  - Ponovo šalje frejm i resetuje timer
- Ako primi ACK:
  - Ako je broj u sekvenci netačan ili je detektovao grešku: ignoriše ACK
  - Ako je broj u sekvenci ispravan ($R_{sled} = S_{posl} + 1$) i nema greške prima frejm i prelazi u aktivno stanje

Prijemna strana

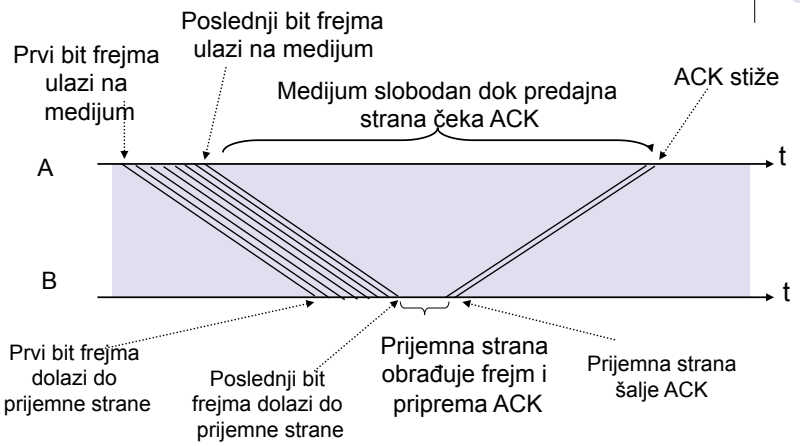
Uvijek se nalazi u aktivnom stanju

- Čeka na dolazak novog frejma
- Kada frejm stigne, provjerava greške
- Ako nema grešaka i broj u sekvenci je korektan ($S_{posl}=R_{sled}$), tada
  - Prima frejm,
  - ažurira R_{sled},
  - šalje ACK frejm za R_{sled},
  - predaje paket višem nivou
- Ako nema grešaka a pogrešan je broj u sekvenci
  - odbacuje frejm
  - šalje ACK frejm sa R_{sled}
- Ako je greška detektovana
  - Odbacuje frejm

Primjena Stop-and-Wait ARQ

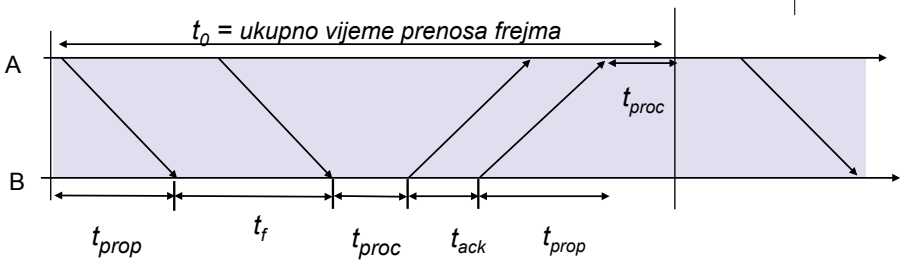
- IBM *Binary Synchronous Communications protocol* (Bisync): protokol nivoa linka
- *Xmodem*: modem file transfer protocol
- *Trivial File Transfer Protocol* (RFC 1350): jednostavan protokol za prenos fajlova preko UDP protokola

Efikasnost Stop-and-Wait



- Frejmu veličine 10000 bita na linku kapaciteta 1 Mb/s treba 10 ms za prenos
- Ako čeka na ACK = 1 ms, efikasnost je = 10/11= 91%
- Ako čeka na ACK = 20 ms, efikasnost =10/30 = 33%

Stop-and-Wait Model



$$t_0 = 2t_{prop} + 2t_{proc} + t_f + t_{ack}$$

$$= 2t_{prop} + 2t_{proc} + \frac{n_f}{R} + \frac{n_a}{R}$$

Broj bita u frejmu koji nosi informaciju

Broj bita u ACK frejmu

Kapacitet kanala (Brzina prenosa)

Efikasnost S&W u kanalu bez greške

Efektivna brzina prenosa:

Biti zaglavlja i CRC polja

$$R_{eff}^{0} = \frac{\text{broj informacionih bita predatih destinaciji}}{\text{vrijeme potrebno za prenos}} = \frac{n_f - n_o}{t_0},$$

Efikasnost prenosa:

$$\eta_0 = \frac{R_{eff}}{R} = \frac{\frac{n_f - n_o}{t_0}}{R} = \frac{1 - \frac{n_o}{n_f}}{1 + \frac{n_a}{n_f} + \frac{2(t_{prop} + t_{proc})R}{n_f}}$$

Uticaj veličine zaglavlja

Uticaj veličine ACK frejma

Uticaj proizvoda kašnjenja i brzine prenosa

Primjer: Uticaj proizvoda kašnjenja i brzine prenosa

n_f=1250B =10000bita, n_a=n_o=25B=200bita

2xkašnjenjexbrzina prenosa Efikasnost	1 ms 200 km	10 ms 2000 km	100 ms 20000 km	1 s 200000 km
R=1Mb/s	$2.24*10^4$ 88%	$4.04*10^4$ 49%	$2.204*10^5$ 9%	$2.0204*10^6$ 1%
R= 1Gb/s	$2*10^6$ 1%	$2*10^7$ 0.1%	$2*10^8$ 0.01%	$2*10^9$ 0.001%

Stop-and-Wait ne funkcioniše dobro za vrlo velike brzine prenosa ili velika kašnjenja uslijed propagacije

Efikasnost S&W u kanalu sa greškom

- Neka je $1 - P_f$ vjerovatnoća da frejm stiže bez greške
- Srednji broj pokušaja prenosa do prvog uspješnog prenosa je $1/(1-P_f)$
- Srednje ukupno vrijeme prenosa je $t_0/(1 - P_f)$

$$\eta_{SW} = \frac{R_{eff}}{R} = \frac{\frac{n_f - n_o}{\left(\frac{t_0}{1-P_f}\right)}}{R} = \frac{1 - \frac{n_o}{n_f}}{1 + \frac{n_a}{n_f} + \frac{2(t_{prop} + t_{proc})R}{n_f}}(1 - P_f)$$

Efekat gubitka frejma

Primjer: Uticaj BER-a (Bit Error Rate)

n_f=1250B=10000bita, n_a=n_o=25B=200bita

Pronaći efikasnost za slučajne greške vjerovatnoće po bitu p=0, 10^{-6}, 10^{-5}, 10^{-4}

$$1 - P_f = (1-p)^{n_f} \approx e^{-n_f p} \quad \text{za veliko } n_f \text{ i malo } p$$

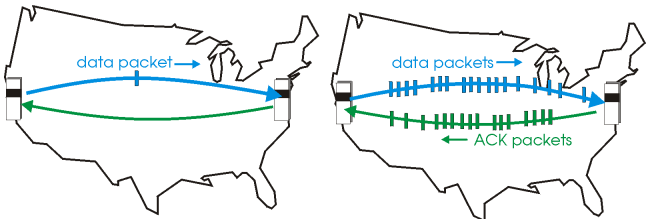
$1 - P_f$ **Efikasnost**	p=0	p=10^{-6}	p=10^{-5}	p=10^{-4}
1Mb/s	1	0.99	0.905	0.368
i 1ms	88%	86.6%	79.2%	32.2%

Greška obara efikasnost !!!!!!

“Pipelined” protokoli

“Pipelining”: pošiljalac dozvoljava istovremeni prenos više paketa čiji prijem nije potvrđen

- Opseg brojeva u sekvenci mora biti proširen
- Baferovanje na predajnoj i/ili prijemnoj strani



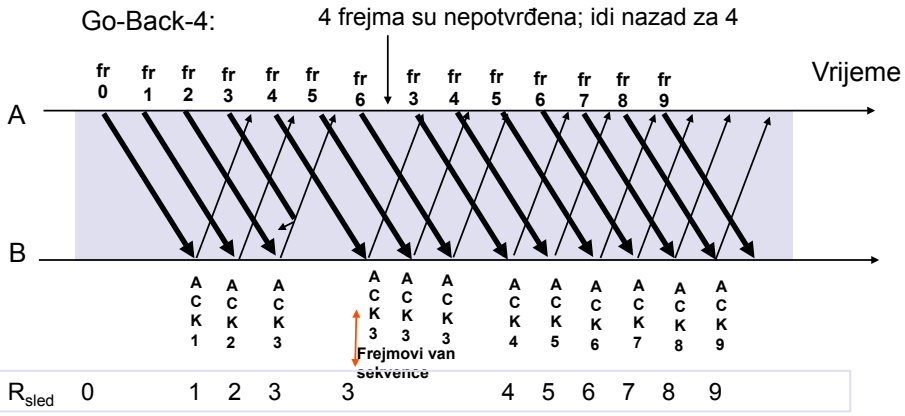
a) Stop and wait protokol b) Pipeline protokol

- Najpoznatiji predstavnici ovih protokola su: *“go-Back-N”, “selective repeat”*

Go-Back-N

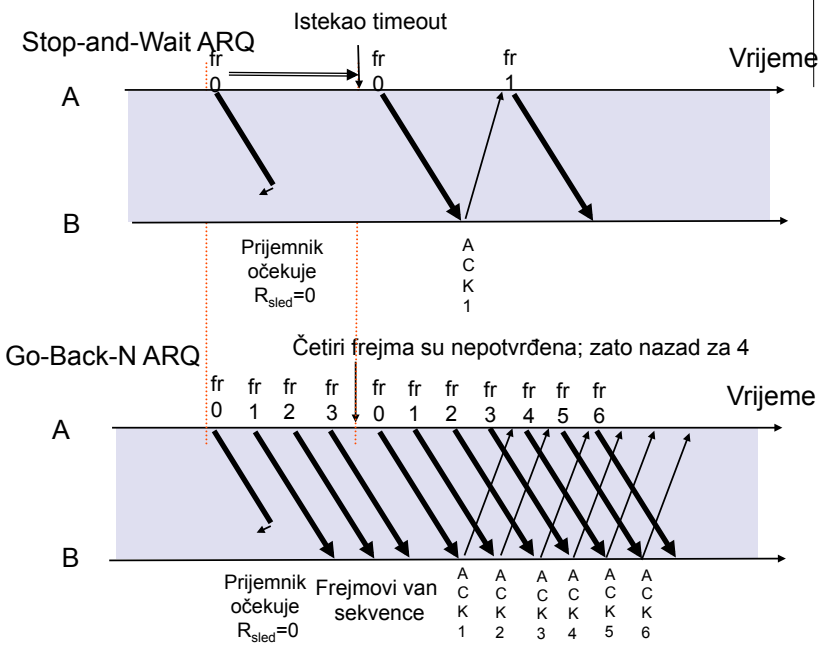
- Poboljšani Stop-and-Wait bez čekanja!
- Zauzima se kanal neprekidnim slanjem frejmova
- Dozvoljava prozor od W_s nepotvrđenih frejmova
- Koristi *m*-bitnu sekvencu numeracije
- Ako ACK za “najstariji” frejm stigne prije nego što je prozor postao prazan, može da se nastavi slanje podataka
- Ako je prozor prazan, zaustavlja slanje i šalje sve nepotvrđene frejmove
- Alternativa: Korišćenje timeout mehanizma

Go-Back-N ARQ



- Pipelined prenos frejmova čini kanal uvijek zauzetim
- Frejm sa greškom i frejmovi van sekvence se ignorišu i odbacuju
- Predajna strana mora da se vrati unazad za veličinu prozora kada je prozor ispražnjen

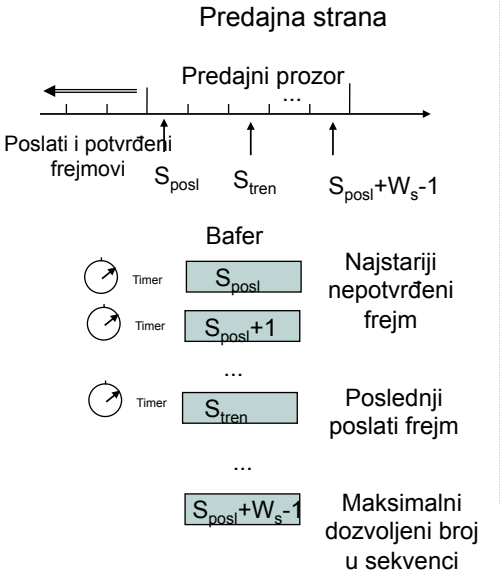
Veličina prozora mora biti dovoljna da pokrije RTT (round trip time)

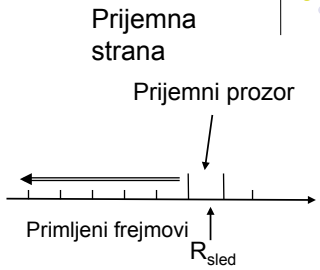


Go-Back-N sa Timeout mehanizmom

- Problem sa opisanim Go-Back-N:
  - Ako se frejm izgubi, a izvorišna strana nema frejmova za slanje, tada prozor neće biti ispražnjen i oporavka neće biti
- Najpoznatija rješenja su aktivirati timeout za najstariji ili svaki frejm
  - Kada timeout istekne, obaviti retransmisiju svih nepotvrđenih frejmova

Go-Back-N predajna & prijemna strana



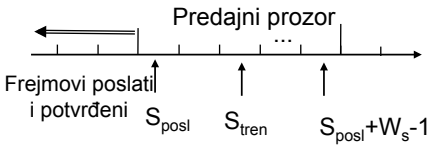


Prijemna strana prihvata samo frejmove bez greške i sa brojem u sekvenci R_{sled}

Kada takav frejm stigne R_{sled} se poveća za jedan, tako da se prijemni prozor pomjera udesno za jedan

Pomjeranje prozora

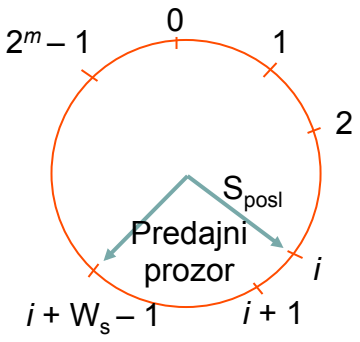
Predajna strana



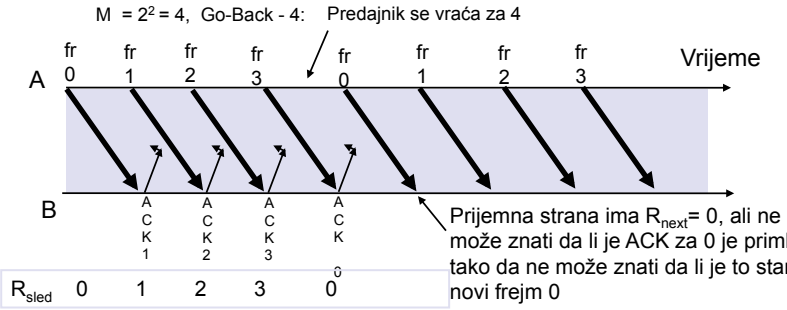
Predajna strana čeka potvrdu bez greške sa brojem S_{posl}

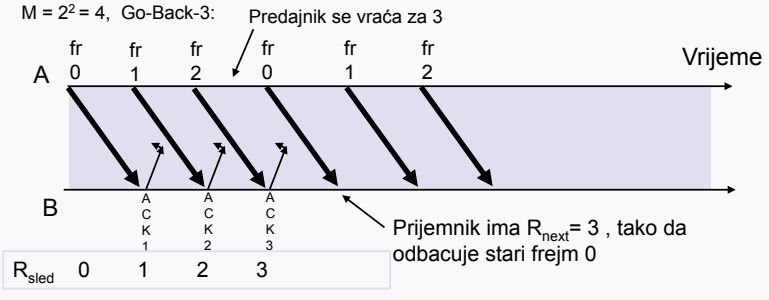
Kada takva potvrda stigne, S_{posl} se povećava za jedan, a predajni prozor se pomjera udesno za jedan

m-bitna sekvenca numeracije

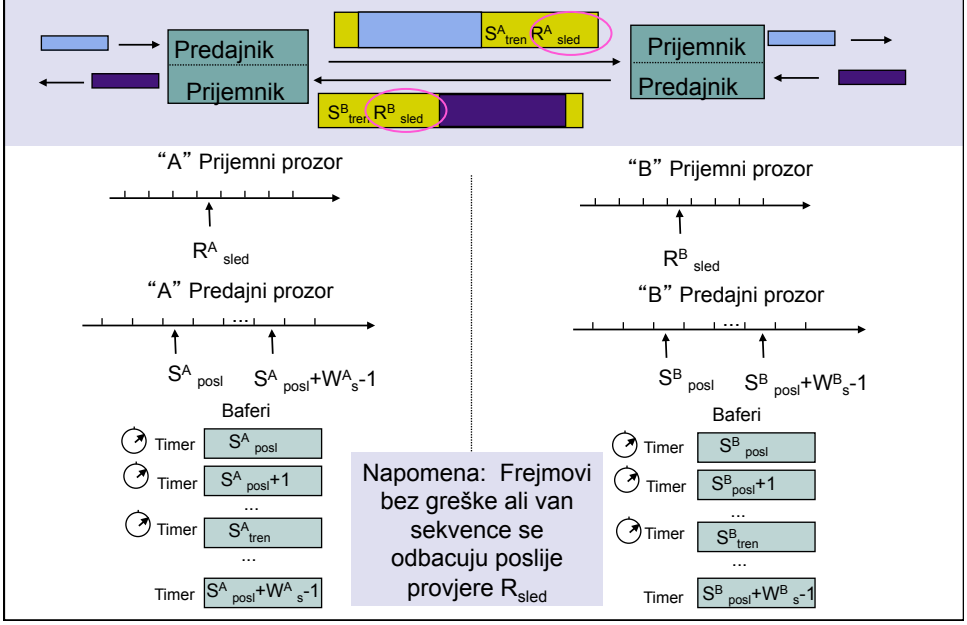


Maksimalna dozvoljena veličina prozora je $W_s = 2^m-1$





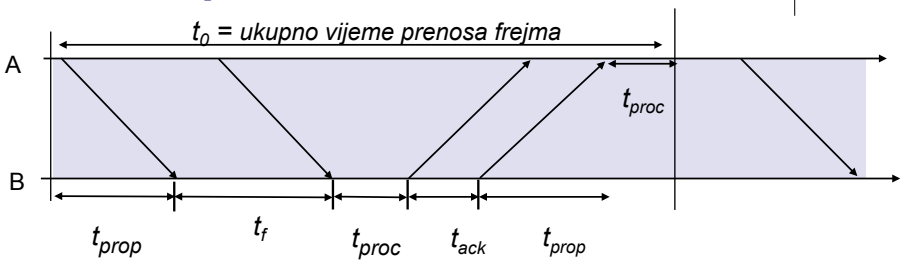
ACK Piggybacking u dvosmjernim GBN



Primjena Go-Back-N ARQ

- *HDLC* (High-Level Data Link Control): protokol nivoa linka
- *V.42 modem*: kontrola greške kod telefonskih modemskih linkova

Potrebno Timeout vrijeme i veličina prozora



- Timeout vrijeme treba da bude dovoljno za:
  - Dva vremena propagacije + 1 vrijeme obrade: 2 T_{prop} + T_{proc}
  - Slanje frejma koji počne da se šalje neposredno po prijemu posmatranog frejma
  - Sledeći frejm nosi ACK, T_f
- W_s bi trabalo da bude veliko tako da je kanal zauzet tokom cijelog T_{out}

Potrebna veličina prozora za proizvod kašnjenja i brzine prenosa

Frejm = 1250B=10,000bita, R = 1Mb/s		
$2(t_{prop} + t_{proc})$	**2 x Kašnj. x Brz. pren.**	**Prozor**
10 ms	10000 bita	2
100 ms	100,000 bita	11
1s	1,000,000 bita	101
10s	10,000,000 bita	1001

Efikasnost Go-Back-N

- GBN je efikasan, ako je W_s dovoljno veliko tako da je medijum uvijek zauzet, pri čemu u kanalu nema gubitka
- Neka je P_f vjerovatnoća pogrešnog prenosa frejma, tada je vrijeme predaje frejma:
  - t_f ako prvi pokušaj prenosa uspije $(1 - P_f)$
  - $t_f + W_s t_f/(1-P_f)$ ako prvi pokušaj prenosa ne uspije P_f

$$t_{GBN} = t_f(1-P_f) + P_f\{t_f + \frac{W_s t_f}{1-P_f}\} = t_f + P_f \frac{W_s t_f}{1-P_f} \quad \text{i}$$

$$\eta_{GBN} = \frac{\frac{n_f - n_o}{t_{GBN}}}{R} = \frac{1 - \frac{n_o}{n_f}}{1 + (W_s - 1)P_f}(1-P_f)$$

Proizvod kašnjenja i brzine prenosa određuje W_s

Primjer: Uticaj BER na GBN

n_f=1250B= 10000 bita, n_a=n_o=25B= 200 bita

Uporediti efikasnosti S&W i GBN za nivoe BER p = 0, 10^{-6}, 10^{-5}, 10^{-4} i R = 1 Mb/s RTT=100 ms

1Mb/sx100ms = 100000bita = 10frejmova → Koristiti W_s = 11

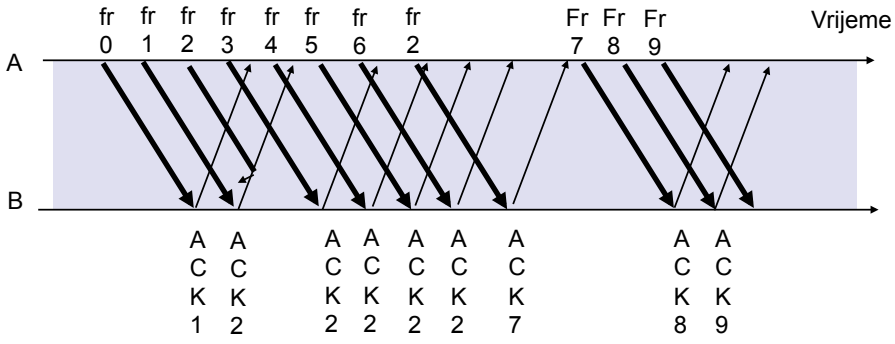
Efikasnost	p=0	p=10^{-6}	p=10^{-5}	p=10^{-4}
S&W	8.9%	8.8%	8.0%	3.3%
GBN	98%	88.2%	45.4%	4.9%

- *Go-Back-N predstavlja značajno poboljšanje u odnosu na Stop-and-Wait za veliki proizvod kašnjenja i brzine prenosa*
- *Go-Back-N postaje neefikasan ako BER raste*

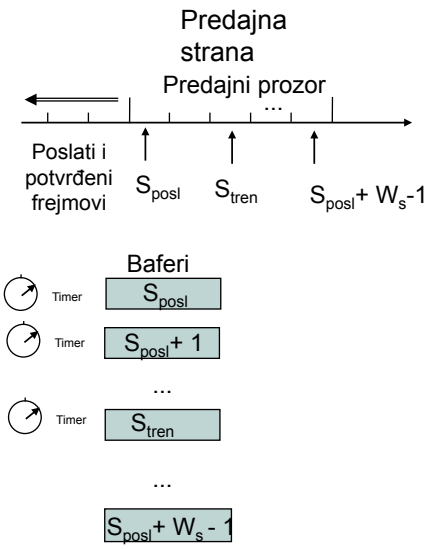
Selective Repeat ARQ

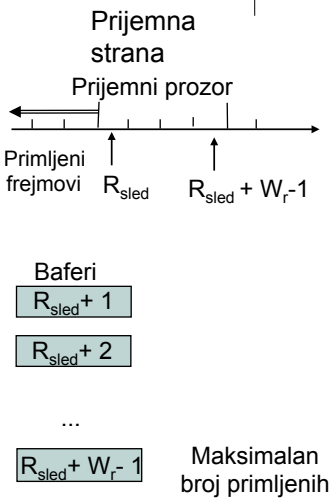
- Go-Back-N ARQ je neefikasan zato što se u slučaju gubitka ponovo šalje N frejmova
- Selective Repeat ponovo šalje samo jedan frejm
  - Timeout izaziva retransmisiju tačno određenog frejma
  - Ponavljanje potvrde već potvrđenog frejma izaziva retransmisiju najstarijeg frejma
- Prijemna strana nadzire prijemni prozor sa brojevima u sekvenci frejmova koji mogu biti primljeni
  - Frejmovi bez greške, ali van sekvence i unutar prijemnog prozora se baferuju
  - Dolazak frejma sa R_{sled} izaziva pomjeranje prozora za najmanje 1 udesno

Selective Repeat ARQ (Ws=Wr=5)

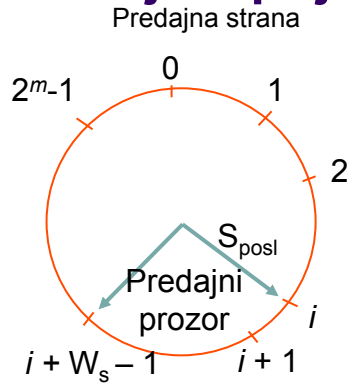


Selective Repeat ARQ

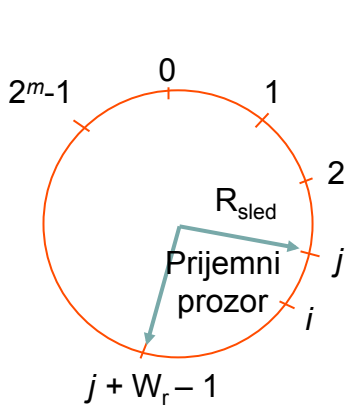




Predajni i prijemni prozori



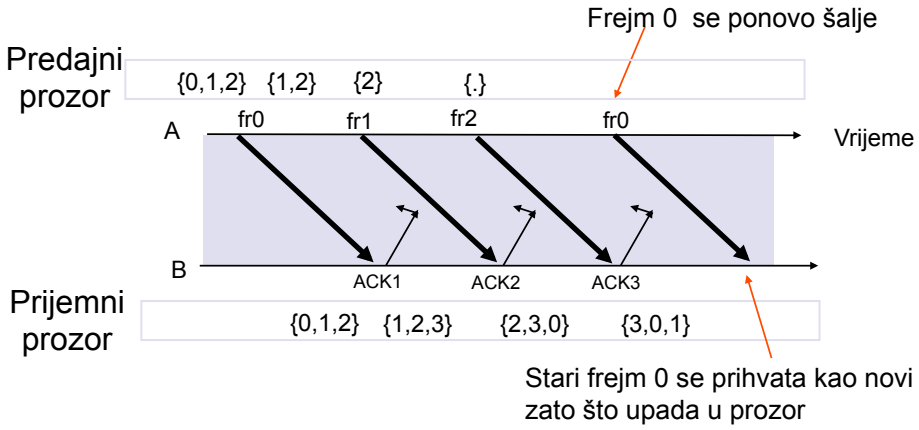
Kada ACK stigne pomjera se za k unaprijed sa
$R_{sled} = S_{posl} + k$
$k = 1, \ldots, W_s-1$



Pomjera se za 1 ili više kada stigne frejm sa brojem u sekvenci jednakom R_{sled}

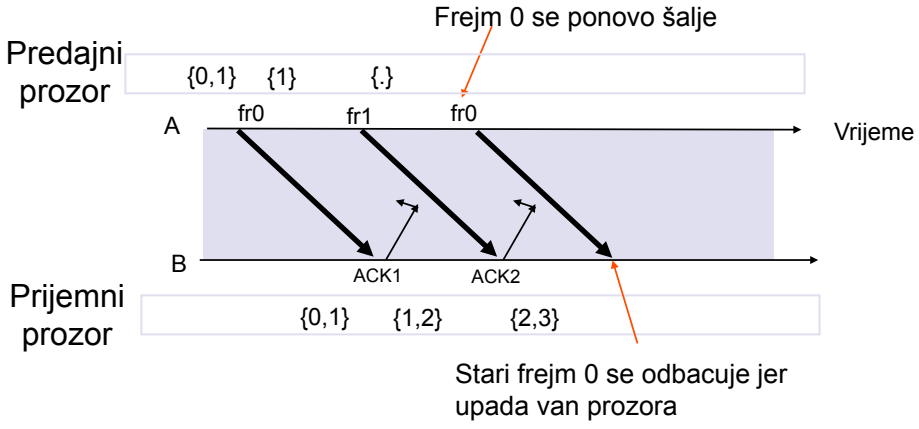
Koje su vrijednosti W_s i W_r dozvoljene?

- Primjer: $M=2^2=4$, $W_s=3$, $W_r=3$



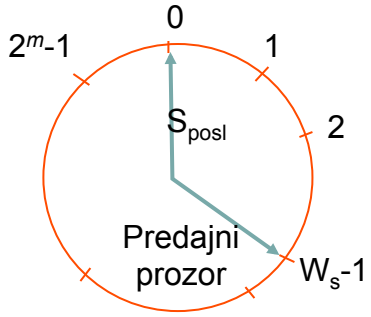
$W_s + W_r = 2^m$ je maksimalno dozvoljeno!!!!!

- Primjer: $M=2^2=4$, $W_s=2$, $W_r=2$

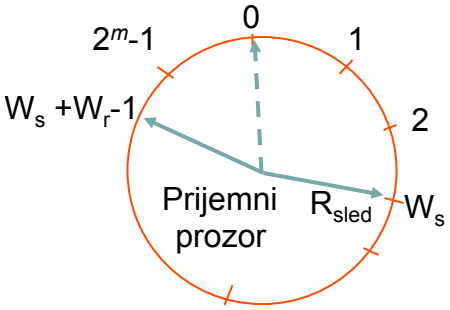


Zašto $W_s + W_r = 2^m$ funkcioniše?

- Predajna strana šalje frejmove od 0 do Ws-1; predajni prozor je prazan
- Svi stižu do prijemne strane
- Sve potvrde su izgubljene
- Predajna strana ponovo šalje frejm 0



- Prijemni prozor počinje sa {0, …, W_r}
- Prozor se pomjera na {W_s,…,W_s+W_r-1}
- Prijemna strana odbacuje frejm 0 zato što je van prozora



Primjena Selective Repeat ARQ

- *TCP* (Transmission Control Protocol)
- *Service Specific Connection Oriented Protocol*: kontrola greške za signalne poruke u ATM mreži

Efikasnost Selective Repeat

- Neka je vjerovatnoća greške u prenosu frejma P_f, tada srednje vijeme potrebno za prenos frejma:
  - $t_f/(1-P_f)$

$$\eta_{SR} = \frac{\frac{n_f - n_o}{t_f/(1-P_f)}}{R} = (1 - \frac{n_o}{n_f})(1 - P_f)$$

Primjer: Uticaj BER-a na Selective Repeat

n_f=1250B= 10000bita, n_a=n_o=25B= 200bita

Uporediti efikasnost S&W, GBN & SR za nivoe BER-a p=0, 10^{-6}, 10^{-5}, 10^{-4} i R= 1 Mb/s & 100ms

Efikasnost	p=0	$p=10^{-6}$	$p=10^{-5}$	$p=10^{-4}$
S&W	8.9%	8.8%	8.0%	3.3%
GBN	98%	88.2%	45.4%	4.9%
SR	98%	97%	89%	36%

- *Selective Repeat je značajno bolji od GBN i S&W, ali ni on nije imun na uticaj povećanja BER-a*

Poređenje efikasnosti ARQ mehanizama

Neka su n_a i n_o zanemarljivo mali u odnosu na n_f, i $L = 2(t_{prop}+t_{proc})R/n_f=(W_s-1)$, tada je

Selective-Repeat:

$$\eta_{SR} = (1-P_f)(1-\frac{n_o}{n_f}) \approx (1-P_f)$$

Go-Back-N: *Za $P_f \approx 0$, SR & GBN su identični*

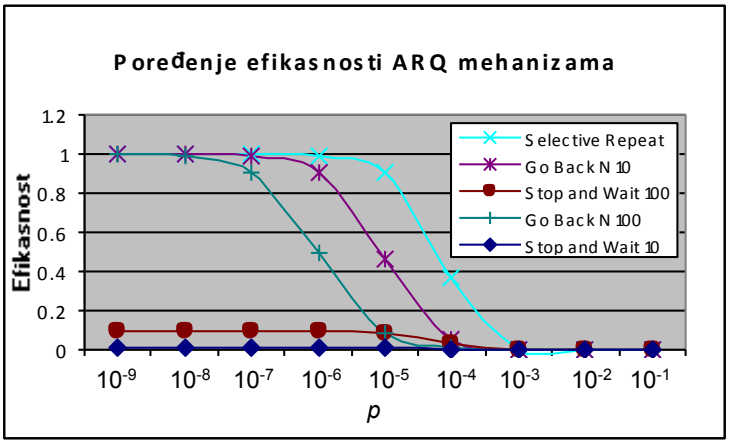
$$\eta_{GBN} = \frac{1-P_f}{1+(W_S-1)P_f} = \frac{1-P_f}{1+LP_f}$$

Za $P_f \rightarrow 1$, GBN & SW su identični

Stop-and-Wait:

$$\eta_{SW} = \frac{(1-P_f)}{1+\frac{n_a}{n_f}+\frac{2(t_{prop}+t_{proc})R}{n_f}} \approx \frac{1-P_f}{1+L}$$

Efikasnost ARQ mehanizama



Proizvod kašnjenja i brzine prenosa = 10, 100