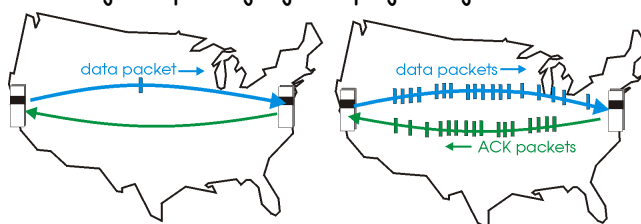


"Pipelined" protokoli

"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

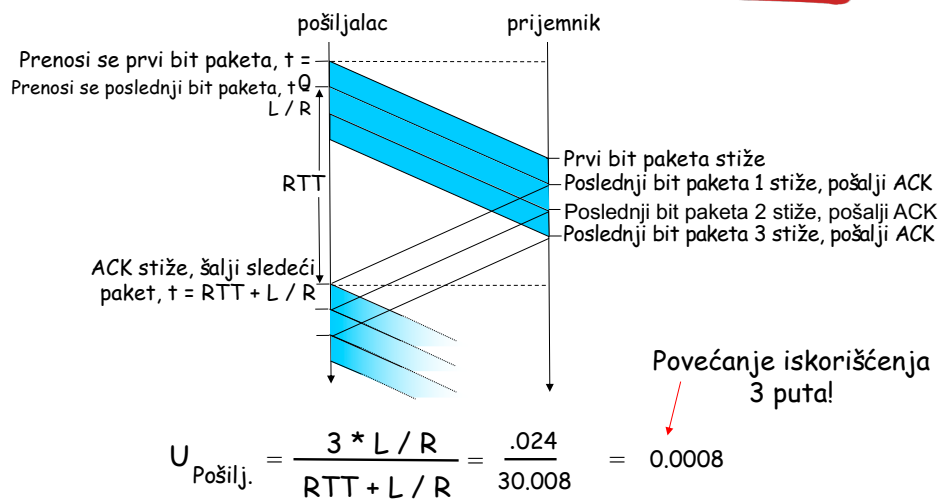
b) Pipeline protokol

- Postoje dvije forme ovog protokola: "go-Back-N", "selective repeat"

Nivo transporta 3-39

39

"Pipelining" povećava iskorišćenje



Nivo transporta 3-40

40

Pipelined protokoli

Go-back-N:

- Pošiljalac može imati do N nepotvrđenih poslatih segmenata
- Prijemnik šalje samo *kumulativne potvrde*
 - Ne potvrđuje segmente ako se jave "praznine"
- Pošiljalac ima timer za najstariji nepotvrđeni paket
 - Kada timer istekne ponovo se šalju svi nepotvrđeni segmenti

Selective Repeat:

- Pošiljalac može imati do N nepotvrđenih poslatih segmenata
- Prijemnik šalje *individualne potvrde* za svaki paket
- Predajnik ima tajmer za svaki nepotvrđeni segment
 - Kada timer istekne ponovo se šalje samo taj segment

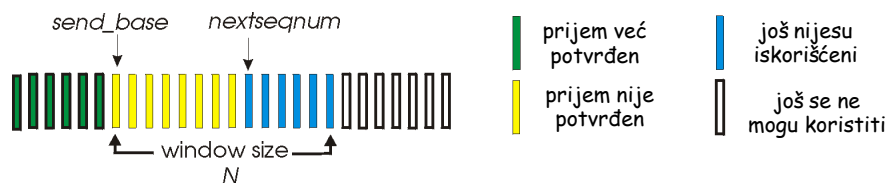
Nivo transporta 4-41

41

Go-Back-N (sliding window)

Pošiljalac:

- k-bitni dugačak broj u sekvenci u zaglavlju paketa znači da se može poslati $N=2^k$ nepotvrđenih paketa
- "prozor" veličine N susjednih nepotvrđenih paketa je dozvoljen
- Zašto ograničavati N?



- Broj u sekvenci se upisuje u polje zaglavlja veličine k bita ($0,2^k-1$). Kod TCP ($k=32$) se ne broje segmenti, već bajti u streamu.
- ACK (n): ACK sve pakete, uključujući n-ti u sekvenci - "kumulativni ACK"
 - Mogu se pojaviti dupli ACK-ovi (vidi prijemnik)

Nivo transporta 4-42

42

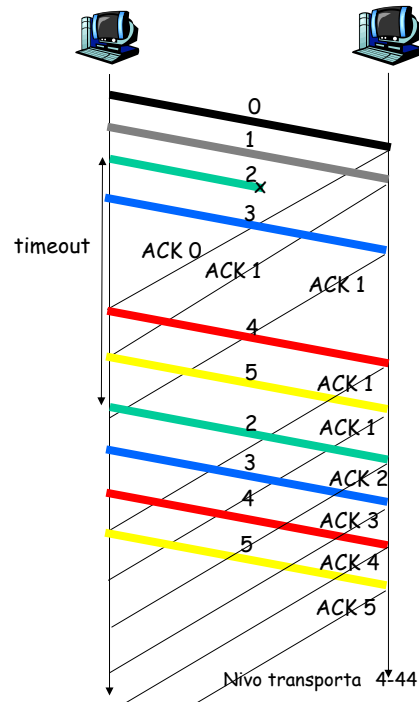
GBN

- timer se inicijalizuje za "najstariji" segment i vezuje za svaki paket čiji prijem još nije potvrđen
- *timeout(n)*: retransmisija paketa n i svih paketa čiji je broj u sekvenci veći od n, u skladu sa veličinom prozora
- uvijek se šalje ACK za korektno primljen paket sa najvećim brojem u sekvenci uz poštovanje *redosleda*
 - Može generisati duple ACK-ove
 - Treba da zapamti samo broj očekivanog paketa
- "out-of-order" paket:
 - odbacuje -> nema baferovanja na prijemu! Zašto?
 - Re-ACK paket sa najvećim brojem u sekvenci

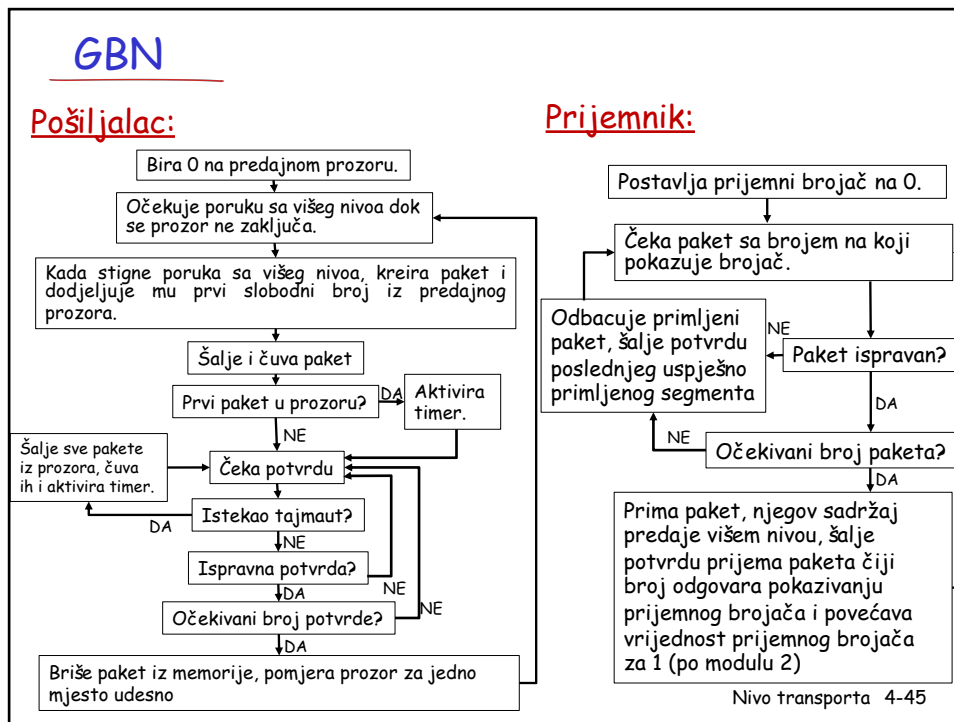
Nivo transporta 4-43

43

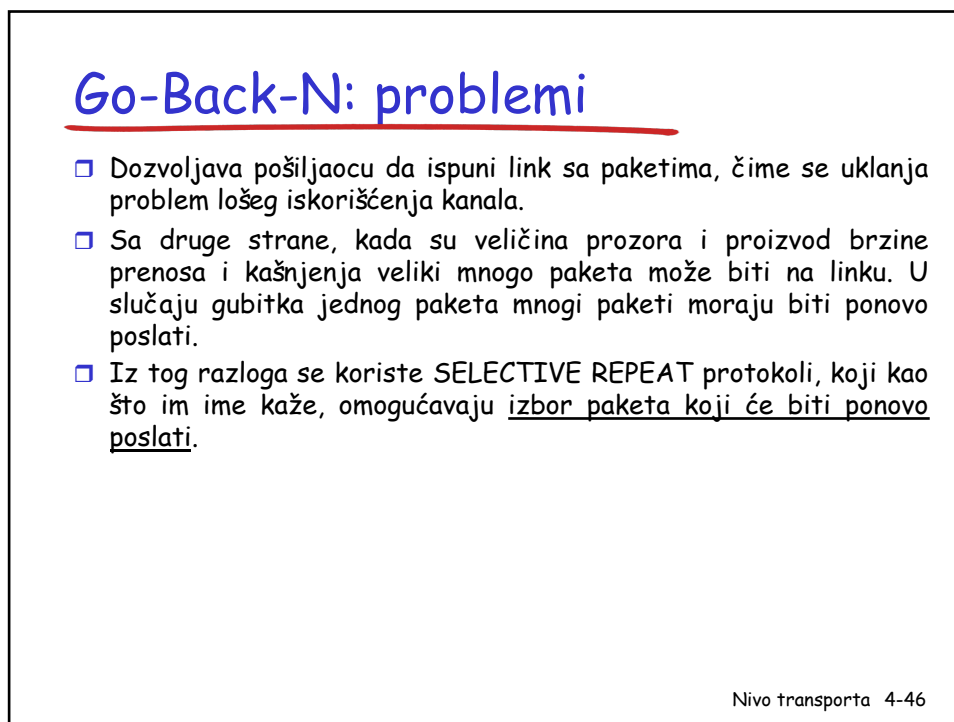
GBN u akciji



44



45



46

"Selective Repeat"

- Prijemnik *pojedinačno* potvrđuje sve ispravno primljene pakete
 - baferuje pakete, ako je to potrebno, za eventualnu redoslednu predaju nivou iznad sebe
- Pošiljalac ponovo šalje samo pakete za koje ACK nije primljen
 - Pošiljalac ima tajmer za svaki paket čiji prijem nije potvrđen
- Prozor pošiljaoca
 - N uzastopnih brojeva u sekvenci
 - Ponovo ograničava broj poslatih paketa, čiji prijem nije potvrđen

Nivo transporta 4-47

47

"Selective repeat"

Pošiljalac

Podaci odozgo:

- Ako je sledeći broj u sekvenci u prozoru dostupan, šalji paket

timeout(n):

- Ponovo šalji paket n, restartuj tajmer

ACK(n) u [sendbase, sendbase+N]:

- Markiraj paket n kao da je primljen
- Ako je n najmanji nepotvrđeni paket, proširi osnovu prozora na bazi narednog najmanjeg broja nepotvrđenog paketa

Prijemnik

paket n u [rcvbase, rcvbase+N-1]

- Pošalji ACK(n)
- out-of-order: baferuj
- in-order: predaj (takođe baferuj, predaj u in-order), povećavaj prozor na sledeći paket koji još nije primljen

paket n u [rcvbase-N, rcvbase-1]

- ACK(n)

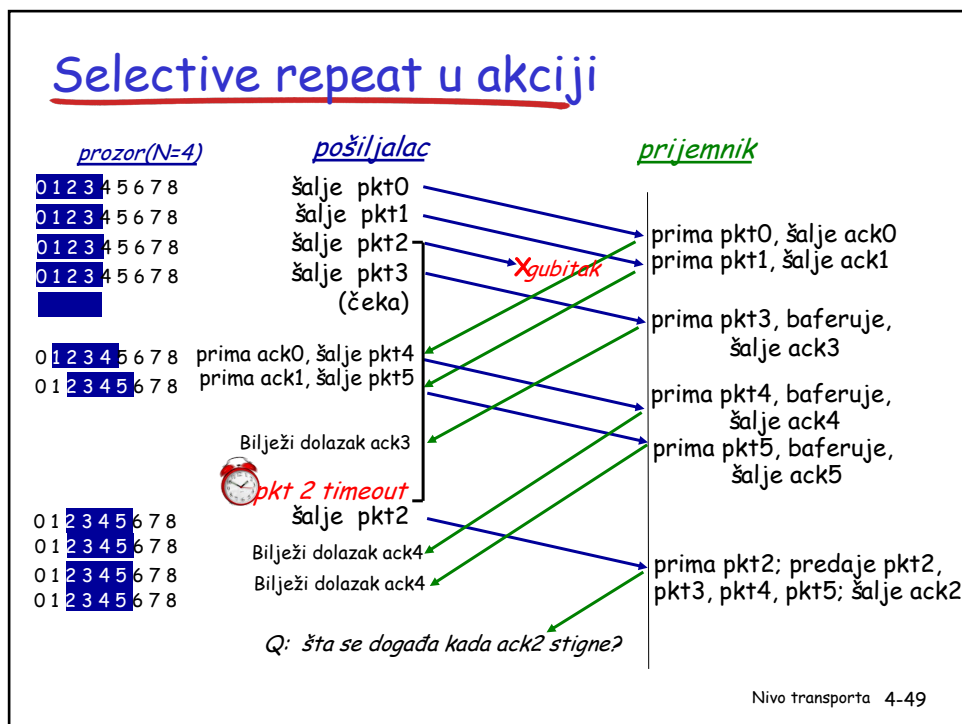
drugačije:

- ignoriši

Nivo transporta 4-48

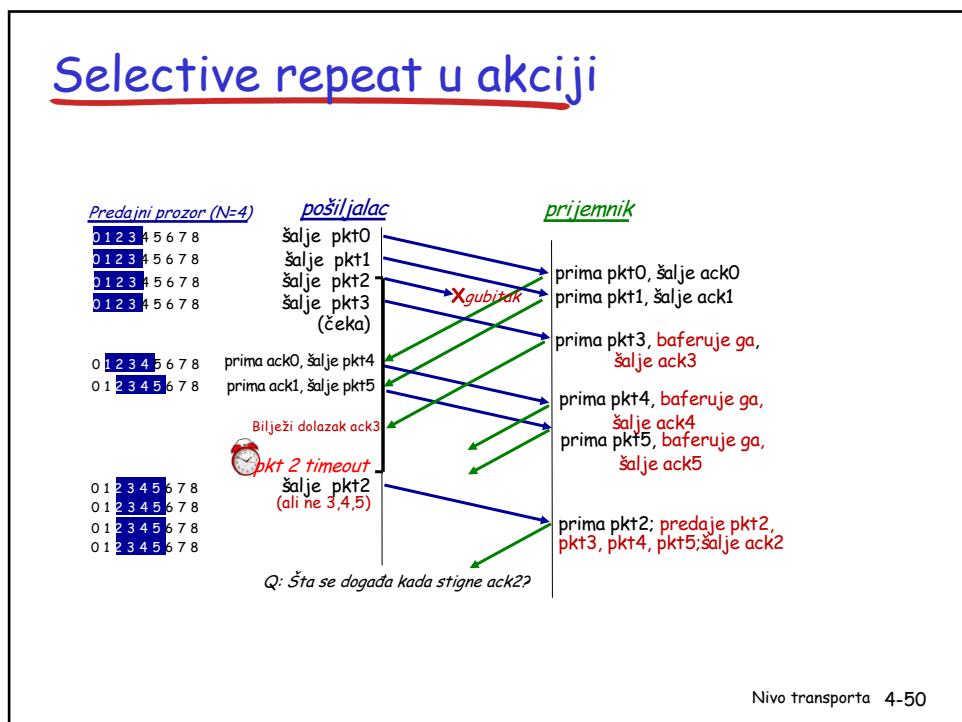
48

Selective repeat u akciji



49

Selective repeat u akciji



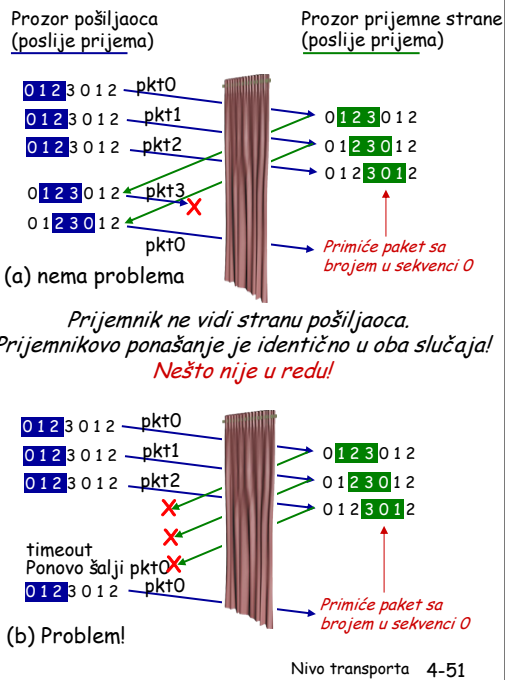
50

Selective repeat: dilema

primjer:

- Brojevi u sekvenci: 0, 1, 2, 3
- Veličina prozora=3
- Prijemnik ne vidi razliku u dva scenarija!

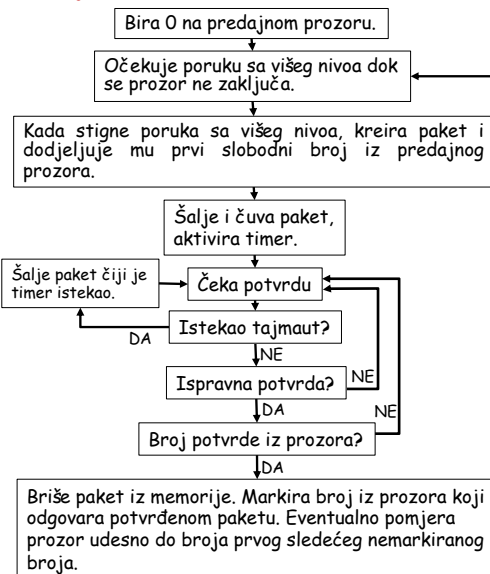
- Duplikat se prima kao novi
- (b)
P: kakva je relacija između veličine broja u sekvenci i veličine prozora kako bi se izbjegao problem pod (b)?



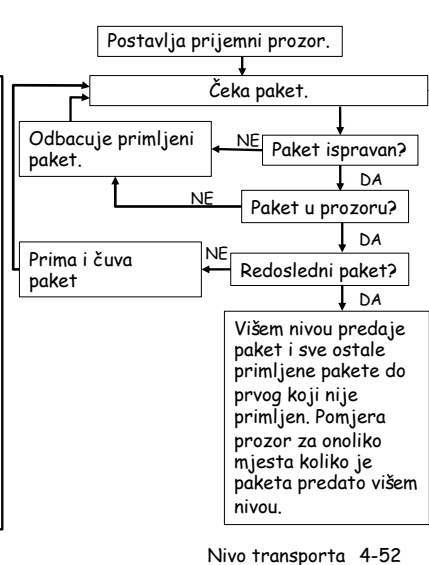
51

SR

Pošiljalac:



Prijemnik:



Nivo transporta 4-52

52

4. Nivo transporta

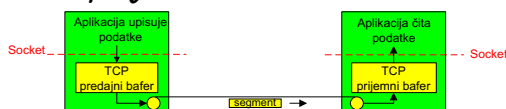
- 4.1 Servisi nivoa transporta
- 4.2 Multipleksiranje i demultipleksiranje
- 4.3 Nekonektivni transport: UDP
- 4.4 Principi pouzdanog prenosa podataka
- 4.5 Konektivni transport: TCP
 - Struktura segmenta
 - Pouzdani prenos podataka
 - Kontrola protoka
 - Upravljanje konekcijom

Nivo transporta 4-53

53

TCP: Pregled RFC-ovi: 793, 1122, 1323, 2018, ..., 7323

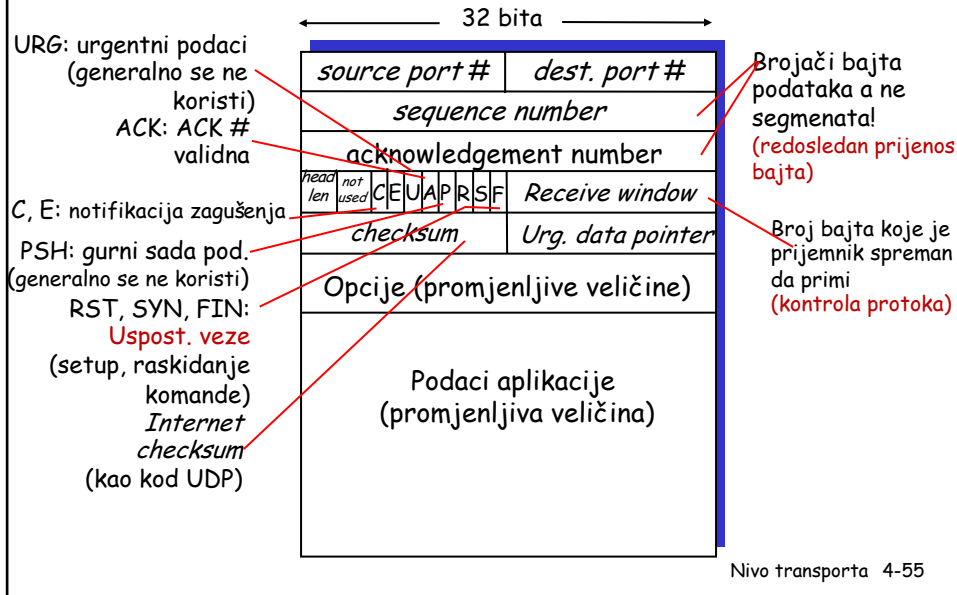
- tačka-tačka:
 - Jedan pošilj, jedan prij.
- pouzdan, redosledan prenos bajta:
 - nema "granica poruka"
- "pipelined":
 - TCP kontrola zagušenja i protoka podešava veličinu prozora
- *Baferi za slanje & prijem*
- Kumulativne potvrde
- "full duplex" prenos:
 - U istoj vezi prenos se obavlja u oba smjera
 - MSS: maksimalna veličina polja podataka u segmentu
- konektivan:
 - "handshaking" (razmjena kontrolnih poruka) inicira je pošiljalac, razmjenjuje stanja prije slanja
- kontrola protoka:
 - Pošiljalac ne može "zagušiti" prijemnika



Nivo transporta 4-54

54

TCP struktura segmenta (21B-1480B)



55

TCP brojevi u sekvenci, ACK-ovi

Brojevi u sekvenci:

- Dodjeljuje se broj prvom bajtu iz sadržaja segmenta
- Inicijalne vrijednosti se utvrđuju na slučajan način.

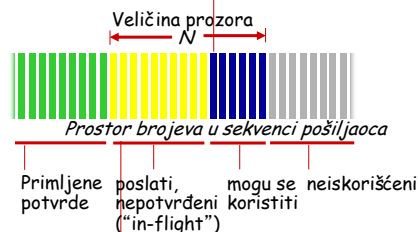
Potvrde (ACK):

- Broj sekvence sledećeg bajta koji se očekuje sa druge strane
- kumulativni ACK

P: Kako se prijemnik ponaša prema *out-of-order* segmentima?

Odlazni segment pošiljaoca

Source port #	dest port #
sequence number	
acknowledgement number	
rwnd	
checksum	urg pointer



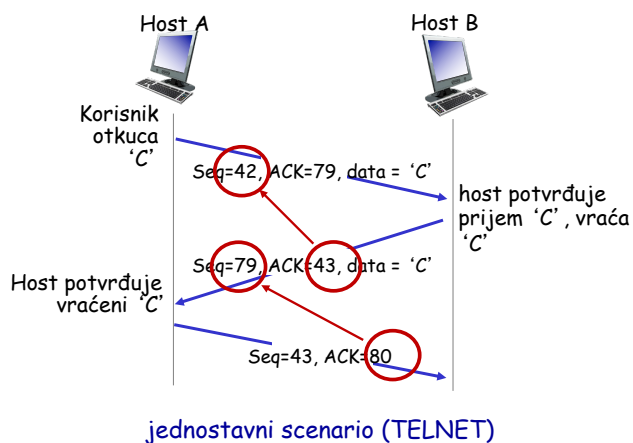
Dolazni segment pošiljaoca

source port #	dest port #
sequence number	
acknowledgement number	
A rwnd	
checksum	urg pointer

Nivo transporta 4-56

56

TCP brojevi u sekvenci, potvrde



Nivo transporta 4-57

57

4. Nivo transporta

- 4.1 Servisi nivoa transporta
- 4.2 Multipleksiranje i demultipleksiranje
- 4.3 Nekonektivni transport: UDP
- 4.4 Principi pouzdanog prenosa podataka
- 4.5 Konektivni transport: TCP
 - Struktura segmenta
 - Pouzdaní prenos podataka
 - Kontrola protoka
 - Upravljanje konekcijom

Nivo transporta 4-58

58

TCP pouzdani prenos podataka

- TCP kreira pouzdani servis korišćenjem IP nepouzdanog servisa
- *Pipelined* segmenti
- Kumulativne potvrde
- TCP koristi jedan retransmisioni tajmer
- Retransmisije su triggerovane sa:
 - *timeout* događajima
 - duplim ack-ovima
- Na početku treba razmotriti pojednostavljenog TCP pošiljaoca:
 - Ignorišu se duplirani ack-ovi
 - Ignorišu se kontrole protoka i zagušenja

Nivo transporta 4-59

59

Događaji vezani za TCP pošiljaoca

1. Podaci primljeni od aplikacije:
 - Kreiranje segmenta sa sekvencom brojeva
 - Broj u sekvenci je redni broj prvog bajta podataka u segmentu
 - Startuje se tajmer ako to već nije urađeno
 - Interval *timeout*-a se izračunava po odgovarajućoj formuli
2. timeout:
 - Ponovo se šalje segment koji je izazvao *timeout*
 - restartovati tajmer
3. Ack primljen:
 - Ako se potvrdi prijem ranije nepotvrđenog segmenta
 - Napraviti odgovarajući *update*
 - startovati tajmer ako postoje segmenti koji čekaju

Nivo transporta 4-60

60

TCP Round Trip Time i Timeout

Kako postaviti TCP vrijeme *timeout*-a?

- Duže od RTT-a
 - ali RTT varira
- Suviše kratko: prerani *timeout*
 - nepotrebne retransmisije
- Previše dugo: spora reakcija na gubitak segmenta
- Potrebna je aproksimacija RTT-a

Kako aproksimirati RTT?

- **SampleRTT**: mjeriti vrijeme od slanja segmenta do prijema ACK
 - Ignorirati retransmisije
 - Radi se za samo jedan nepotvrđeni segment
- **SampleRTT** će varirati, želja je za što boljom estimacijom RTT
 - Više mjerenja, a ne samo trenutno **SampleRTT**

Da li **SampleRTT** vezivati za svaki nepotvrđeni segment?
Zašto ignorirati retransmisije?

Nivo transporta 4-61

61

TCP Round Trip Time and Timeout

$$\text{EstimatedRTT} = (1 - \alpha) * \text{EstimatedRTT} + \alpha * \text{SampleRTT}$$

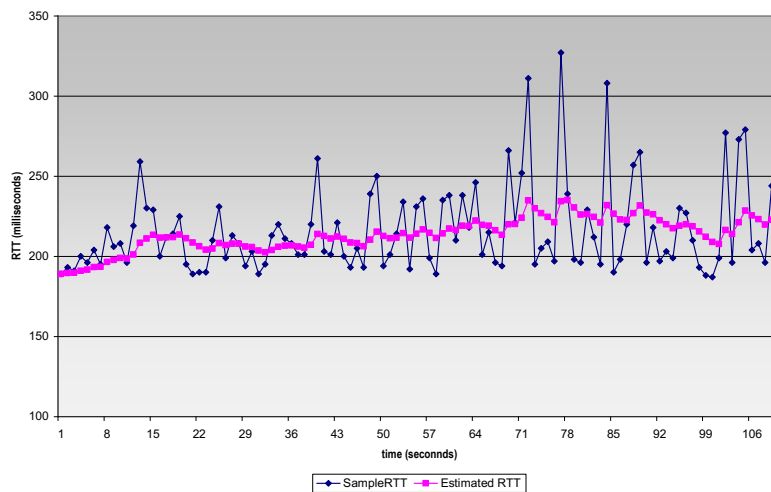
- Uticaj prošlosti opada po eksponencijalnoj raspodjeli
- *Exponential weighted moving average* (EWMA) ili eksponencijalno ponderisani klizni prosjek
- Tipična vrijednost: $\alpha = 0.125$

Nivo transporta 4-62

62

Primjer RTT estimacije:

RTT: gaia.cs.umass.edu to fantasia.eurecom.fr



Nivo transporta 4-63

63

TCP Round Trip Time i Timeout

Setovanje timeout-a

- EstimatedRTT + "sigurnosna margina"
 - Velika varijacija u EstimatedRTT → velika sigurnosna margina
- Prvo se estimira koliko SampleRTT odstupa od EstimatedRTT:

$$\text{DevRTT} = (1-\beta) * \text{DevRTT} + \beta * |\text{SampleRTT} - \text{EstimatedRTT}|$$

(tipično, $\beta = 0.25$)

← EWMA od ove razlike

Tada se setuje timeout interval:

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4 * \text{DevRTT}$$

Nivo transporta 4-64

64

TCP generisanje ACK [RFC 1122, RFC 2581]

Događaj na prijemu

TCP akcije prijemnika

Dolazak *in-order* segmenta sa očekivanim brojem u sekvenci. Svi podaci do očekiv. broja su potvrđ.

ACK sa kašnjenjem. Čeka do 500ms za sledeći segment. Ako nema sledećeg, šalje ACK.

Dolazak *in-order* segmenta sa očekiv. brojem u sekvenci. Potvrđ. prijema drugog segmenta u toku.

Odmah šalje jednu kumulativnu ACK, potvrđujući oba in-order segmenta

Dolazak *out-of-order* segmenta sa većom vrijednosti broja u sekv. od očekivane. Detektovan prekid.

Odmah šalje duplikat ACK, indicirajući broj u sekvenci očekivanog bajta.

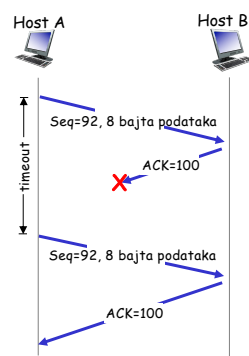
Dolazak segmenta koji djelimično ili potpuno popunjava prekid.

Odmah šalje ACK, omogućavajući da segment popuni prekid

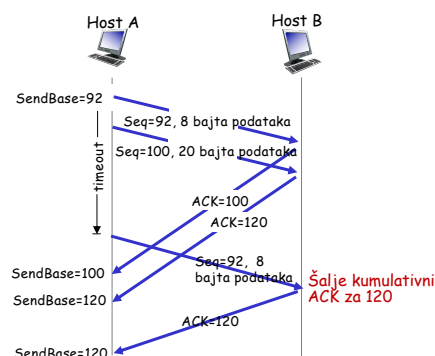
Nivo transporta 4-65

65

Scenariji retransmisije kod TCP-a



Scenario gubitka ACK

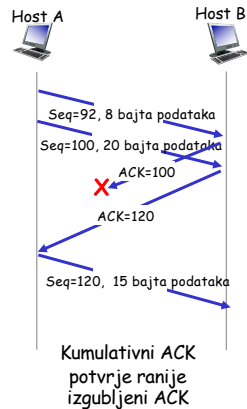


Nedovršeni timeout (nema retransmisije drugog dok ne dodje potvrda za prvi)

Nivo transporta 4-66

66

Scenariji retransmisije kod TCP-a



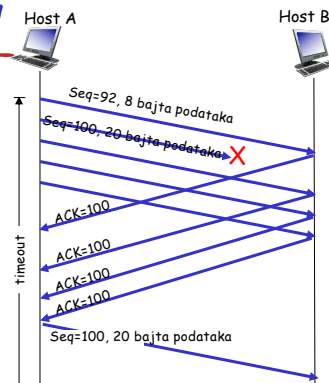
Nivo transporta 4-67

67

"Fast Retransmit"

- *Timeout period* je često predugačak:
 - Dugo kašnjenje prije slanja izgubljenog paketa
- Detekcija izgubljenog segmenta preko dupliranih ACK-ova.
 - Pošiljalac često šalje mnogo segmenata
 - Ako je segment izgubljen, najvjerovatnije će biti dosta duplih ACK-ova.

Da li TCP ima GBN ili "selective repeat" kontrolu greške?
Zašto 3 a ne dva ACK?



TCP fast retransmit

Ako pošiljalac primi 3 ACK za isti segment, pretpostavlja se da je segment poslije potvrđenog izgubljen:

fast retransmit: ponovno slanje segmenta prije nego što je tajmer istekao

Nivo transporta 4-68

68

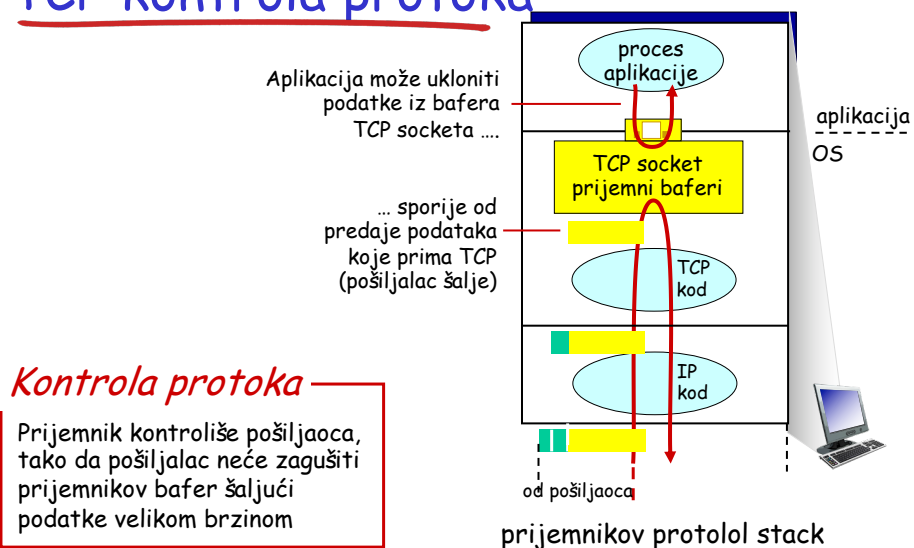
4. Nivo transporta

- 4.1 Servisi nivoa transporta
- 4.2 Multipleksiranje i demultipleksiranje
- 4.3 Nekonektivni transport: UDP
- 4.4 Principi pouzdanog prenosa podataka
- 4.5 Konektivni transport: TCP
 - Struktura segmenta
 - Pouzdani prenos podataka
 - Kontrola protoka
 - Upravljanje konekcijom

Nivo transporta 4-69

69

TCP kontrola protoka

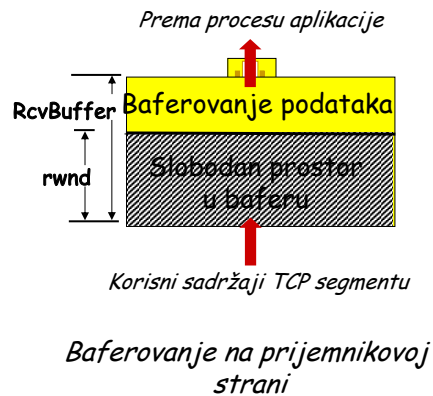


Nivo transporta 4-70

70

TCP kontrola protoka

- Prijemnik oglašava slobodan prostor u baferu podešavanjem vrijednosti polja `rwnd` u zaglavlju TCP segmenta
 - Veličina `RcvBuffer` se podešava u opcijama socketa (tipična vrijednost 4096B)
 - Mnogi OS podešavaju `RcvBuffer`
- Pošiljalac ograničava broj nepotvrđenih (*in-flight*) podataka na vrijednost prijemnikovog `rwnd`
- Garantuje da se ne prepuni bafer



Nivo transporta 4-71

71

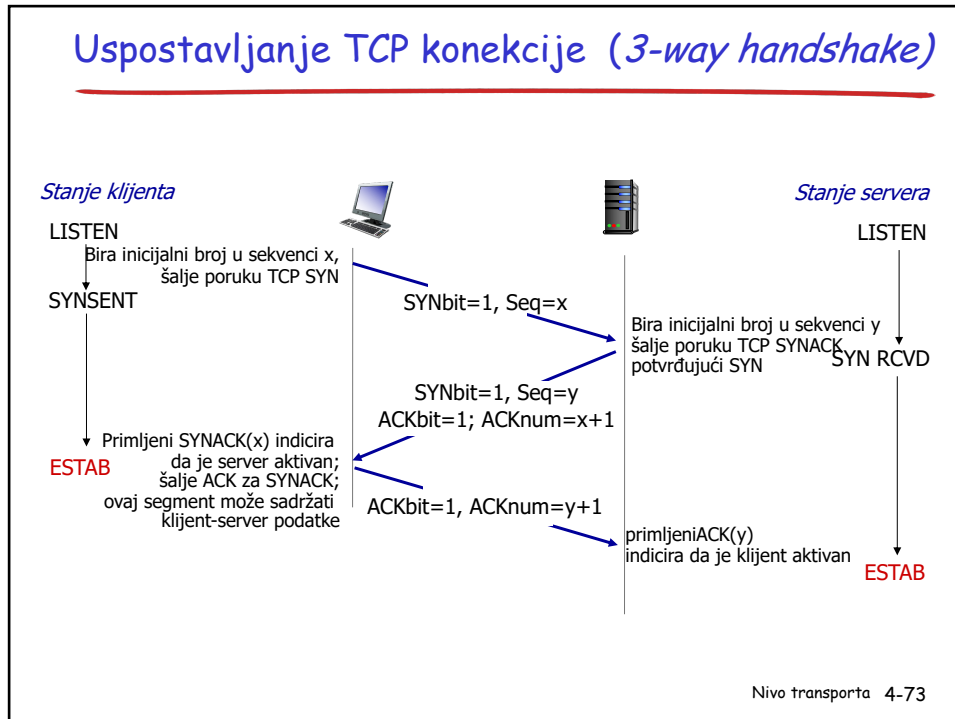
4. Nivo transporta

- 4.1 Servisi nivoa transporta
- 4.2 Multipleksiranje i demultipleksiranje
- 4.3 Nekonektivni transport: UDP
- 4.4 Principi pouzdanog prenosa podataka
- 4.5 Konektivni transport: TCP
 - Struktura segmenta
 - Pouzdani prenos podataka
 - Kontrola protoka
 - Upravljanje konekcijom

Nivo transporta 4-72

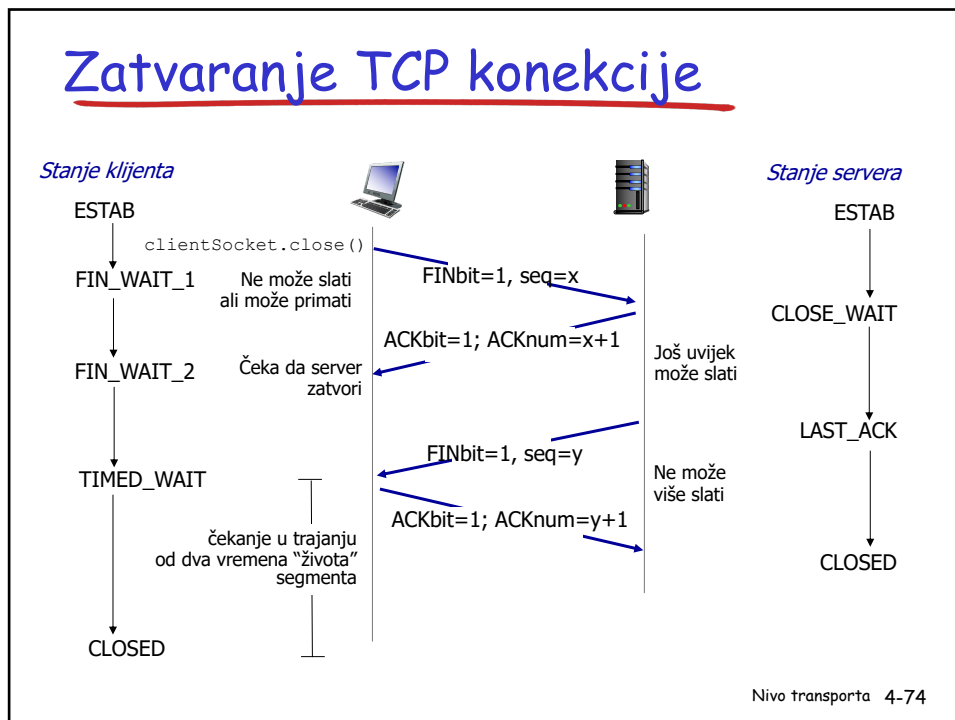
72

Uspostavljanje TCP konekcije (3-way handshake)



73

Zatvaranje TCP konekcije



74

Nivo transporta

Ispitna pitanja

1. Objasniti multipleksiranje i demultipleksiranje na transportnom nivou
2. UDP protokol (Servisi. Slika i objašnjenje funkcija polja segmenta.)
3. S&W (Objasniti funkcionisanje predajne i prijemne strane. Grafički iustrovati konkretan primjer. Efikasnost)
4. GBN (Objasniti funkcionisanje predajne i prijemne strane. Grafički iustrovati konkretan primjer.)
5. SR (Objasniti funkcionisanje predajne i prijemne strane. Grafički iustrovati konkretan primjer.)
6. TCP (Servisi. Slika i objašnjenje funkcija polja segmenta)
7. TCP pouzdani prenos (Objasniti funkcionisanje predajne i prijemne strane. Fast retransmit)
8. Objasniti kako se za potrebe TCP protokola izračunava RTT
9. TCP kontrola protoka
10. Objasniti i grafički ilustrovati upravljanje TCP konekcijom