

7. Transportni protokoli

Prof.dr Igor Radusinović

igorr@ucg.ac.me

Doc.dr Slavica Tomović

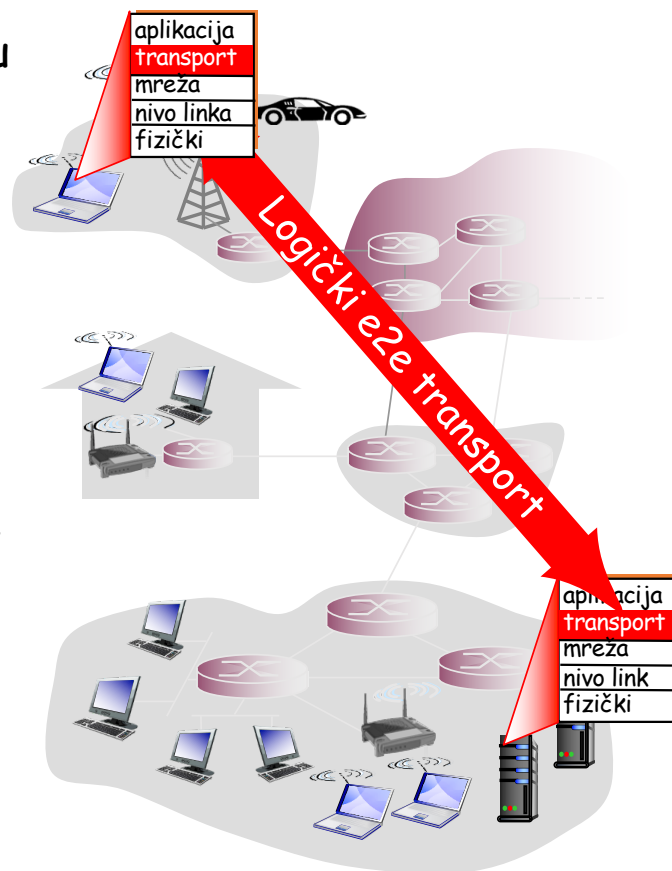
slavicat@ucg.ac.me

Transportni protokoli

- ❑ Transportni servisi
- ❑ UDP
- ❑ Pouzdani prenos podataka
 - ❑ *Stop & Wait*
 - ❑ *Go Back N*
 - ❑ *Selective Repeat*
- ❑ TCP (ukratko)

Transportni servisi i protokoli

- Obezbjeđuju **logičku komunikaciju** između aplikacija koje se odvijaju na različitim hostovima.
- Transportni protokoli se implementiraju na krajnjim sistemima .
 - Predajna strana transportnog protokola dijeli poruke u **segmente** i prosleđuje ih mrežnom nivou.
 - Prijemna strana transportnog protokola desegmentira segmente u poruke i prosleđuje ih nivou aplikacije.
- Više od jednog transportnog protokola je na raspolaganju aplikacijama.
 - Na Internetu dominiraju TCP i UDP



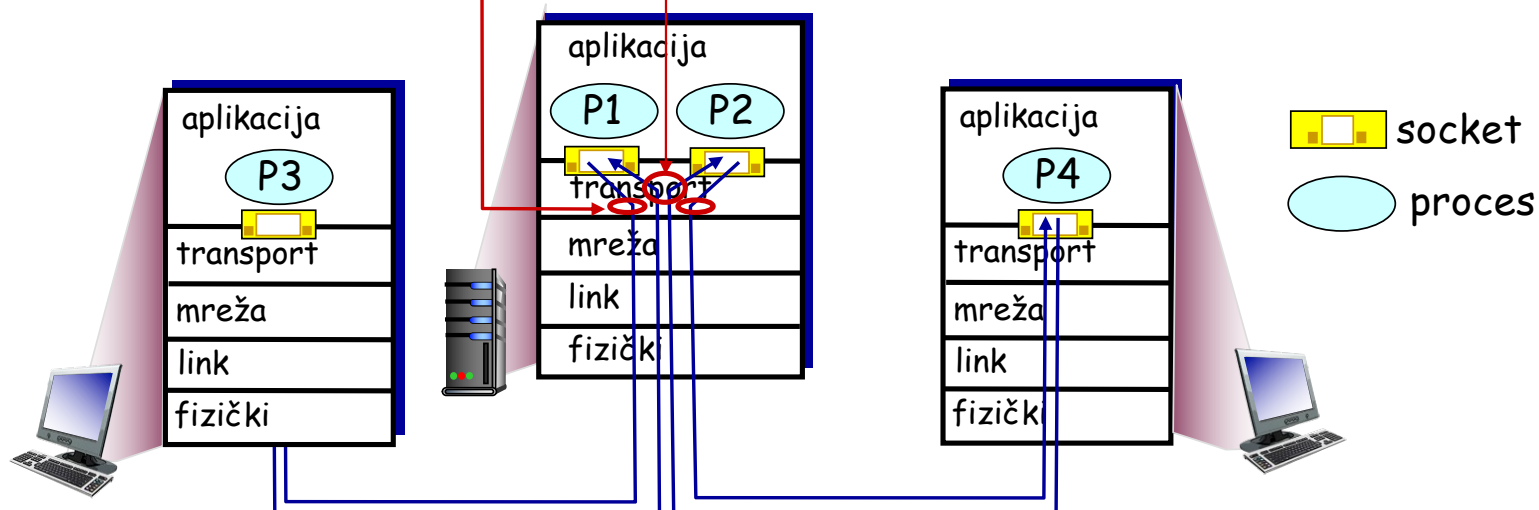
Transportni servisi i protokoli

Multipleksiranje na predaji:

Manipulisanje podacima iz više socket-a i dodavanje transportnog zaglavlja (koristi se za demultipleksiranje).

Demultipleksiranje na prijemu:

Koristi zaglavlje za predaju primljenih segmenata pravom socket-u.



UDP: User Datagram Protocol [RFC 768]

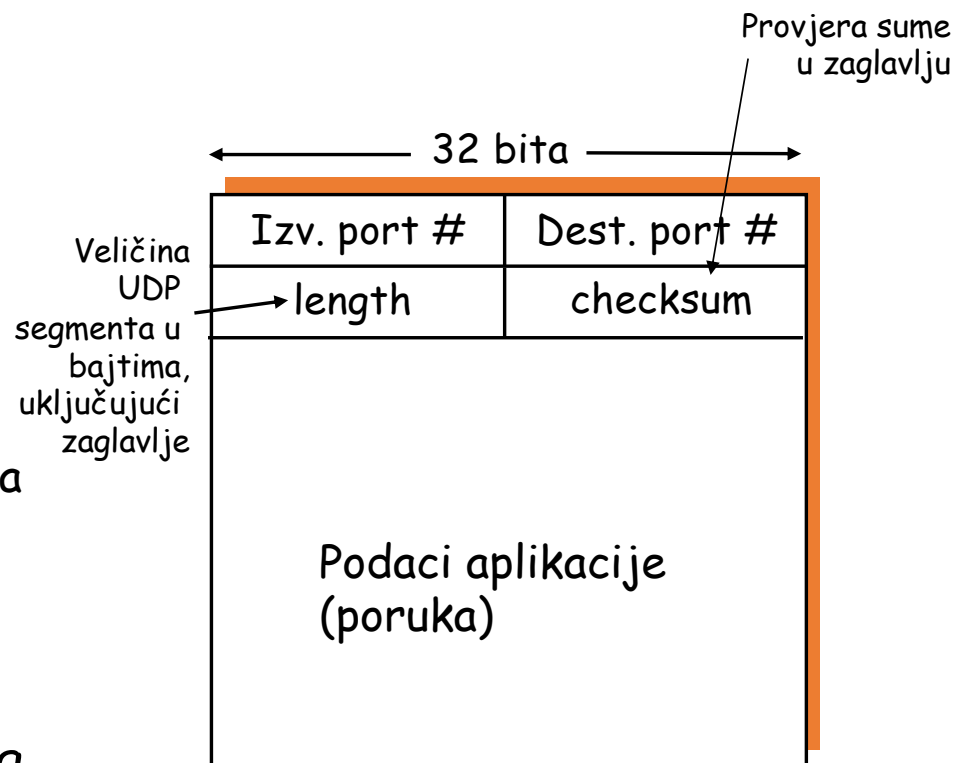
- ❑ Nema poboljšanja koja se nude Internet protokolu.
- ❑ *Best effort* servis tako da UDP segmenti mogu biti:
 - izgubljeni
 - neredosledno predati
- ❑ Nekonektivan
 - Nema uspostavljanja veze (*handshaking*) između pošiljaoca i prijemnika.
 - Svaki UDP segment se tretira odvojeno od drugih segmenata.

Zašto onda UDP?

- ❑ Nema uspostavljanja veze (koja povećava kašnjenje) što je jednostavnije nego kod TCP protokola jer se ne vodi računa o stanju veze.
- ❑ Malo zaglavlje segmenta od svega 8B (TCP zaglavlje ima 20B).
- ❑ Nema kontrole protoka i zagušenja tako da UDP može slati podatke brzinom koju aplikacija diktira.

UDP: User Datagram Protocol [RFC 768]

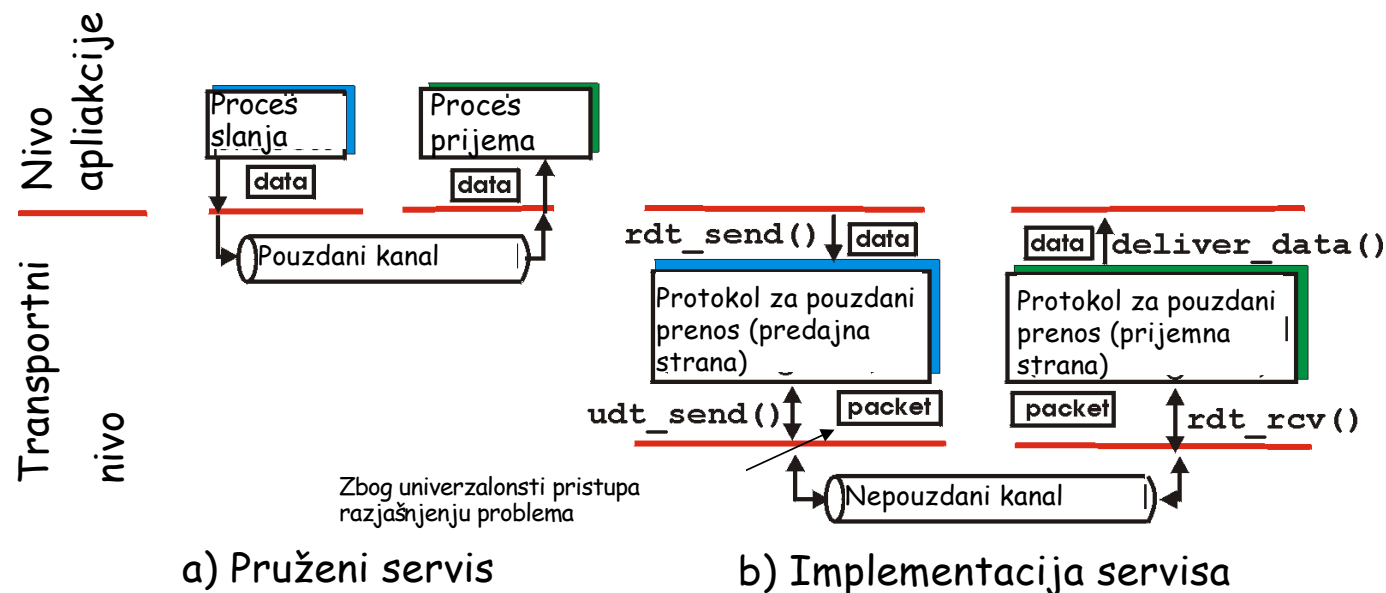
- ❑ Često se koristi za *streaming* multimedijalne aplikacije:
 - tolerantne u odnosu na gubitke i
 - osjetljive na brzinu prenosa
- ❑ Drugi protokoli koji koriste UDP:
 - DNS
 - SNMP (zbog toga što mrežne menadžment aplikacije funkcionišu kada je mreža u kritičnom stanju)
 - RIP (zbog periodičnog slanja RIP update-a)
- ❑ Za pouzdani prenos preko UDP se moraju dodati mehanizmi pouzdanog prenosa podataka na nivou aplikacije.
 - **Oporavak od greške na nivou aplikacije!**
- ❑ Nedostatak kontrole zagušenja je problematičan!



Format UDP segmenta
RFC 768

Pouzdati prenos podataka

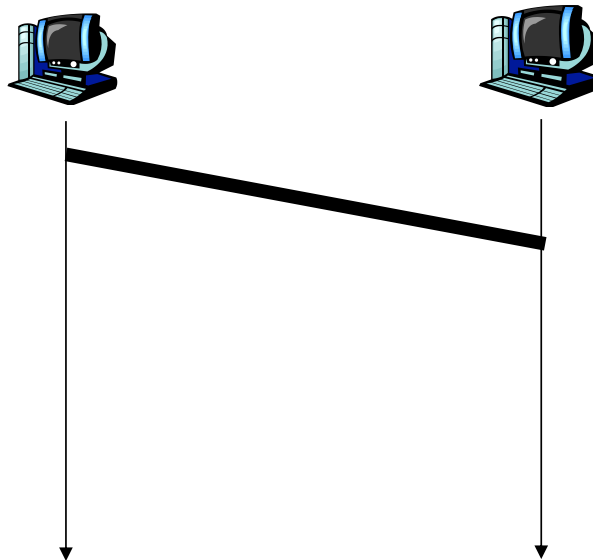
- Važan na nivoima **aplikacije, transporta i linka**.
- Jedna od top-10 karakteristika mreže!



- Karakteristike nepouzdanog kanala će odrediti kompleksnost pouzdanog protokola za prenos podataka.

Pouzdaní prenos podataka

- Kanal je pouzdan:
 - nema greška na bitima
 - nema gubitka segmenata

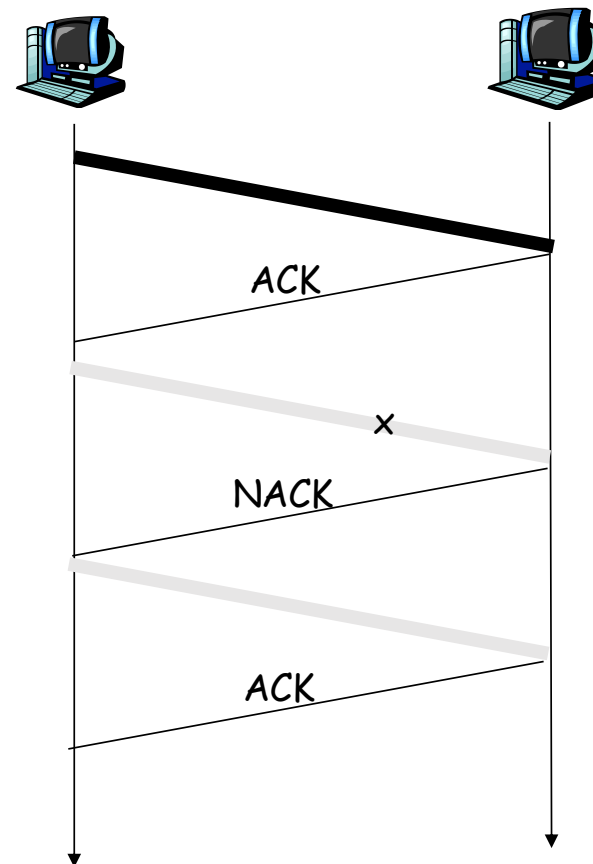


Nema potrebe za mehanizmom pouzdanog prenosa podataka!

Pouzdati prenos podataka

Kanal koji unosi grešku ali ne i gubitke

- ❑ Kanal može zamijeniti vrijednosti bita u segmentu.
- ❑ Potrebno je detektovati grešku na prijemnoj strani. Kako?
- ❑ Prijemna strana o tome mora obavijestiti predajnu stranu potvrdom uspjehnog (**ACK**) ili neuspješnog prijema (**NACK**).
- ❑ Kada prijemna strana primi ACK šalje novi segment, ako primi NACK ponovo šalje prethodni segment (retransmisija).
- ❑ ARQ (*Automatic Repeat reQuest*)



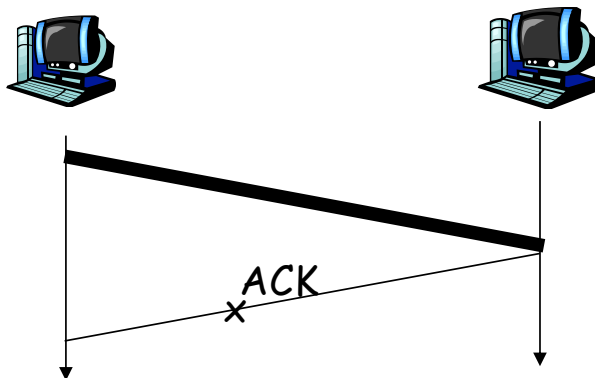
Pouzdati prenos podataka

Šta se dešava kada su ACK/NAK oštećene?

- ❑ Pošiljalac ne zna šta se dešava na prijemu!
- ❑ Retransmisija je besmislena jer je moguće dupliranje segmenata.

Rješavanje duplikata:

- ❑ Pošiljalac dodaje svakom segmentu broj u sekvenci.
- ❑ Pošiljalac ponovo šalje posmatrani segment ako je ACK/NAK oštećen.
- ❑ Prijemnik odbacuje duple segmente.
- ❑ U ACK/NAK nema broja u sekvenci segmenta koji se potvrđuje jer nema gubitka segmenata, pa se potvrda odnosi na poslednji poslati segment.

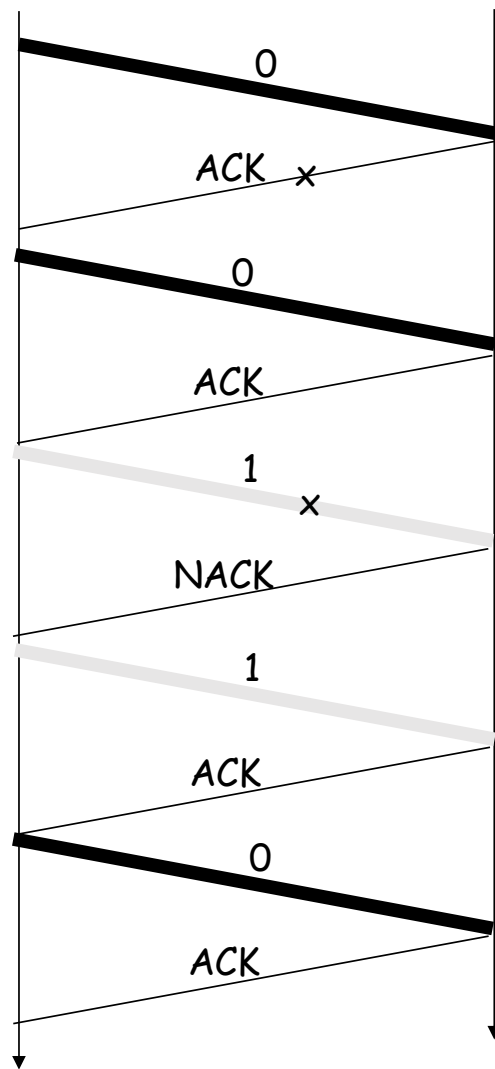


STOP & WAIT

Pošiljac šalje jedan segment, a zatim čeka na odgovor.



Stop & Wait (u kanalu bez gubitaka)



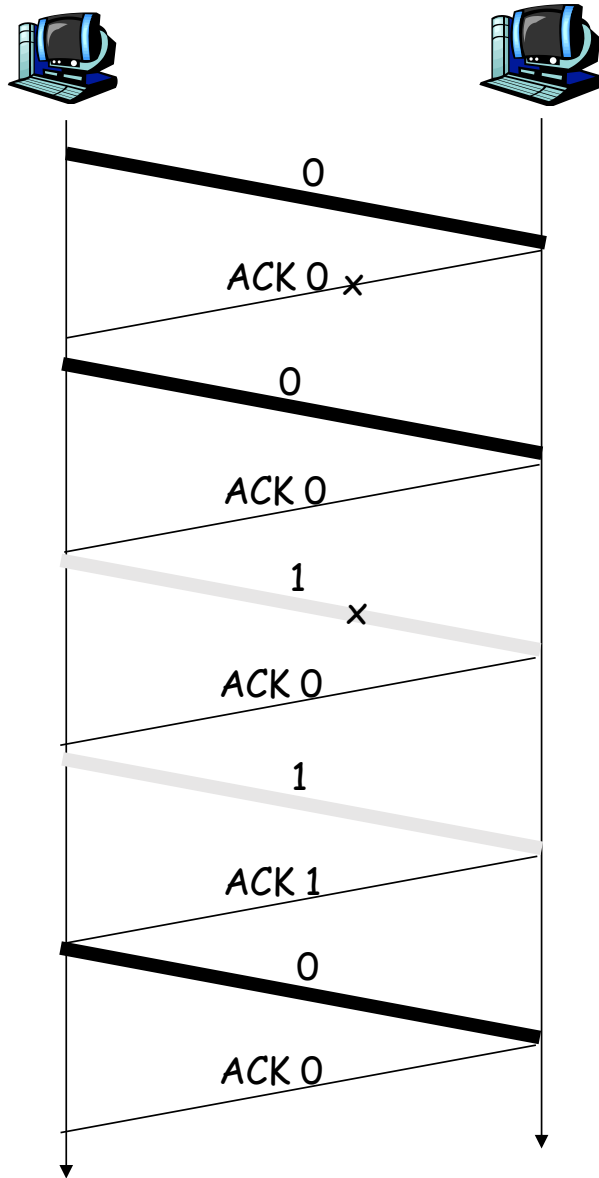
Stop & Wait (u kanalu bez gubitaka)

Pošiljalac:

- ❑ Dodaje **broj u sekvenci** segmentu.
- ❑ Dva broja (0,1) su dovoljna.
Zašto?
- ❑ Pošiljalac posjeduje **brojač** koji pokazuje na broj poslatog segmenta.
- ❑ Mora provjeriti da li je primljeni ACK/NAK oštećen.
- ❑ Nakon potvrde prijema segmenta
 - brojač se povećava za 1 (mod 2),
 - segment se briše
 - pošiljalac čeka podatke sa višeg nivoa.

Prijemnik:

- ❑ Posjeduje **brojač** koji pokazuje na broj očekivanog segmenta.
- ❑ Mora provjeriti da li je primljeni segment duplikat.
 - Stanje indicira da li je 0 ili 1 očekivani broj u sekvenci paketa.
- ❑ Prijemnik ne može znati da li je poslednji ACK/NAK primljen ispravan od strane pošiljaoca.
- ❑ Ako je segment ispravan i očekivan:
 - njegov sadržaj se predaje višem nivou
 - brojač se povećava za 1 (mod 2)



Stop & Wait (u kanalu bez gubitaka) bez NAK

- Iste funkcionalnosti kao u prethodnom slučaju, korišćenjem samo ACK.
- Umjesto NAK, prijemnik šalje ACK za poslednji segment koji je primljen ispravno.
 - Prijemnik mora eksplicitno unijeti broj u sekvenci segmenta čiji se uspješan prijem potvrđuje.
- Dvostruki ACK za isti segment na strani pošiljaoca rezultira istom akcijom kao "ponovo šalji posmatrani segment".

Stop & Wait (kanal sa greškom i gubicima)

Nova pretpostavka: kanal izaziva i gubitak segmenta (podataka ili potvrda).

- Checksum, broj u sekvenci, ACK, retransmisije su od pomoći, ali ne dovoljno.

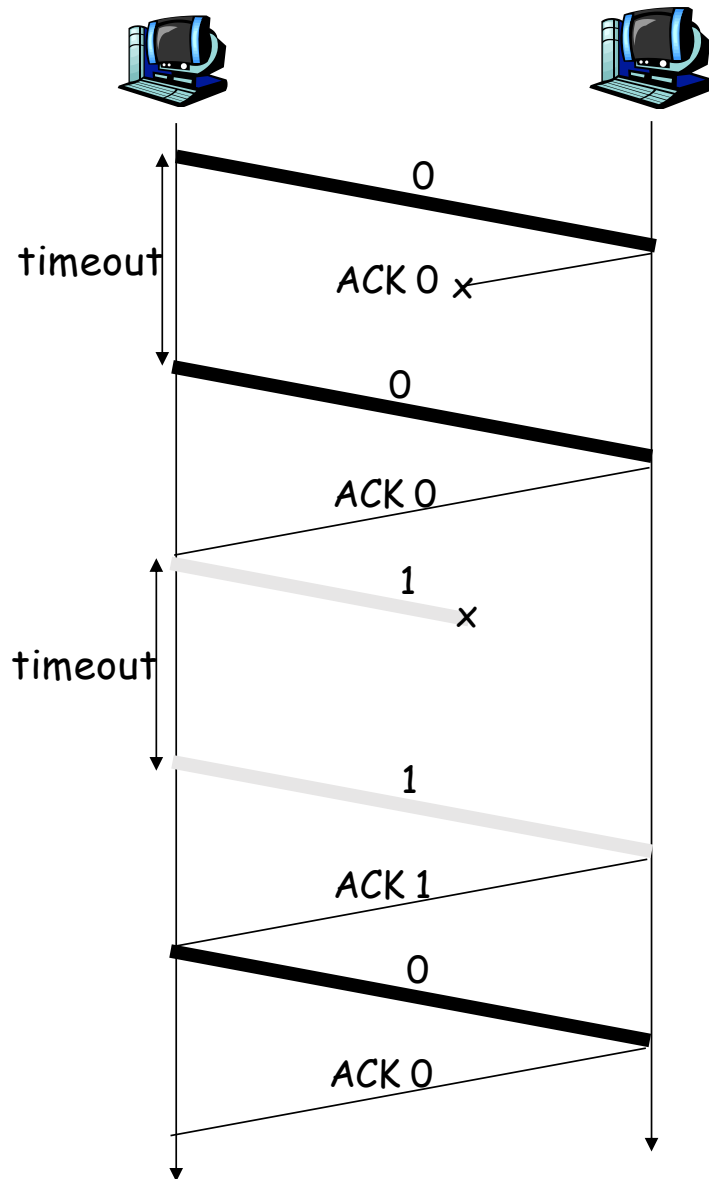
Kako se izboriti sa gubicima?

- Pošiljalac čeka dok se određeni podaci ili ACK izgube, zatim obavlja retransmisiju.
- Koliko je minimalno vrijeme čekanja?
- Koliko je maksimalno vrijeme čekanja?
- Nedostaci?

Pristup: pošiljalac čeka "razumno" vrijeme za ACK

- Uvodi se **timer** na predajnoj strani.
- Retransmisija se obavlja ako se ACK ne primi u timeout intervalu.
- Ako segment (ili ACK) zakasni:
 - Retransmisija će biti duplirana, ali korišćenje broja u sekvenci će to odraditi.
 - Prijemnik mora definisati broj u sekvenci segmenta čiji je prijem već potvrđen.

Stop & Wait (u kanalu sa gubicima)



- Iste funkcionalnosti kao u prethodnom slučaju, korišćenjem samo ACK.
- Umjesto NAK, prijemnik šalje ACK za poslednji segment primljen ispravno.
 - Prijemnik mora eksplicitno unijeti broj u sekvenci segmenta čiji se uspješan prijem potvrđuje.
- Dvostruki ACK za isti segment na strani pošiljaoca rezultira istom akcijom kao "ponovo šalji posmatrani segment".

STOP & WAIT performanse

- S&W funkcioniše, ali ima loše performance.
- Na primjer, link kapaciteta je 1Gb/s, RTT iznosi 15ms, a veličina paketa je 1000B:

$$T_{\text{prenosa}} = \frac{L \text{ (veličina paketa u bitima)}}{R \text{ (propusnost linka, b/s)}} = \frac{8\text{kb/pkt}}{10^9 \text{ b/s}} = 8 \text{ ms}$$

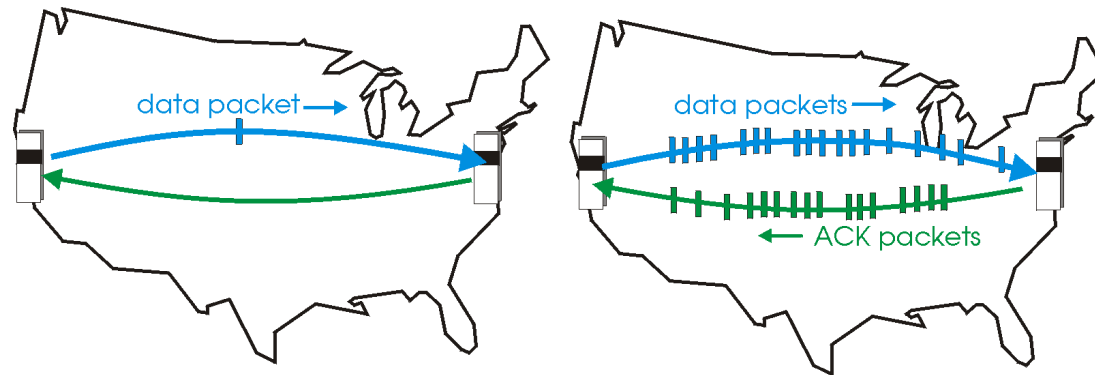
$$U_{\text{pošilj.}} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.00027$$

- $U_{\text{pošiljalac}}$: **iskorišćenje** - dio vremena u kome je pošiljalac zauzet.
- Pošiljalac šalje 1000B paket svakih 30.008ms -> 267kb/s bez obzira što je propusnost linka 1 Gb/s.
- Mrežni protokol ograničava fizičke resurse!
- U praksi je još gore jer je napravljeno nekoliko zanemarivanja!

Pipelined protokoli

Pošiljalac dozvoljava istovremeni prenos više segmenata čiji prijem nije potvrđen.

- Opseg brojeva u sekvenci mora biti proširen.
- Baferovanje više od jednog segmenta na predajnoj i/ili prijemnoj strani.

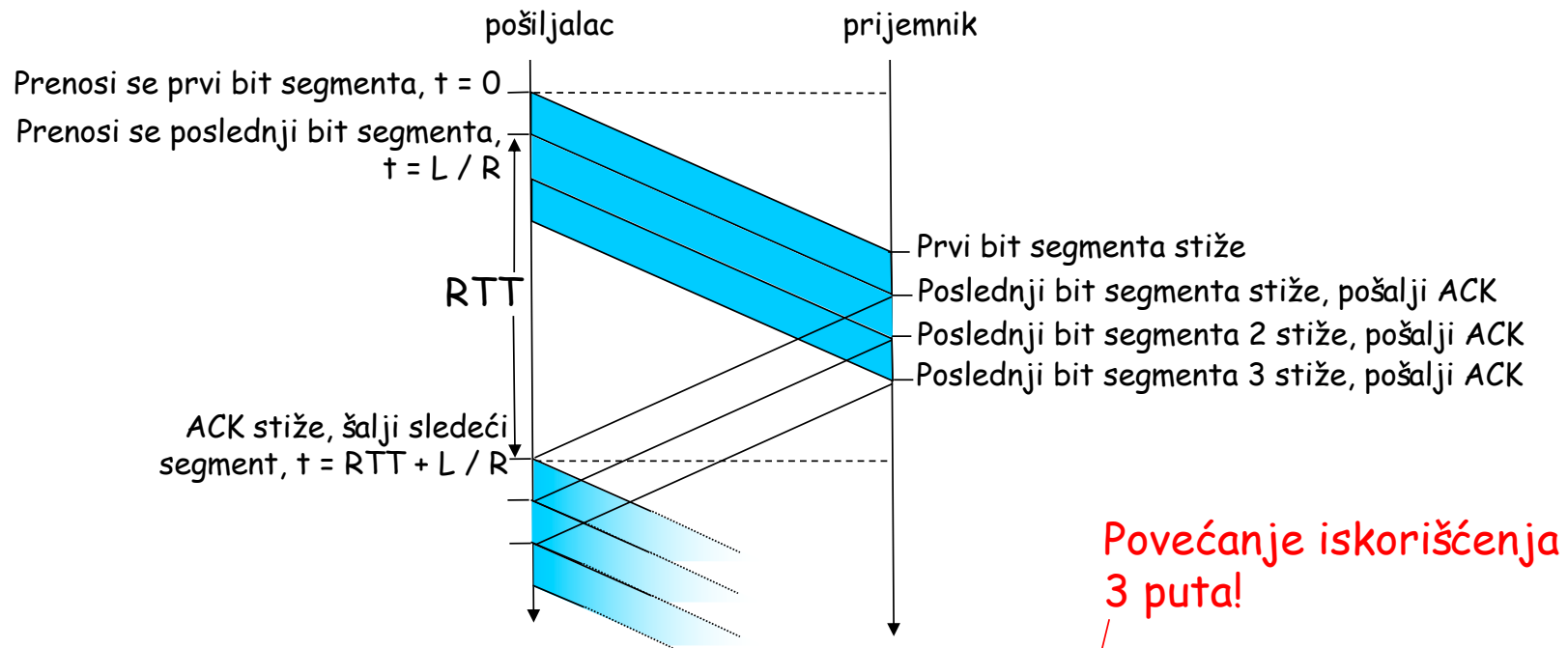


a) *Stop and Wait* protokol b) *Pipeline* protokol

□ Forme ovog protokola:

- Go-Back-N
- Selective Repeat

Pipelining: povećanje iskorišćenja



Povećanje iskorišćenja
3 puta!

$$U_{\text{Pošilj.}} = \frac{3 * L / R}{RTT + L / R} = \frac{.024}{30.008} = 0.0008$$

Pipelined protokoli: pregled

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.
- ❑ Pošiljalac posjeduje **predajni prozor** a prijemnik brojač.

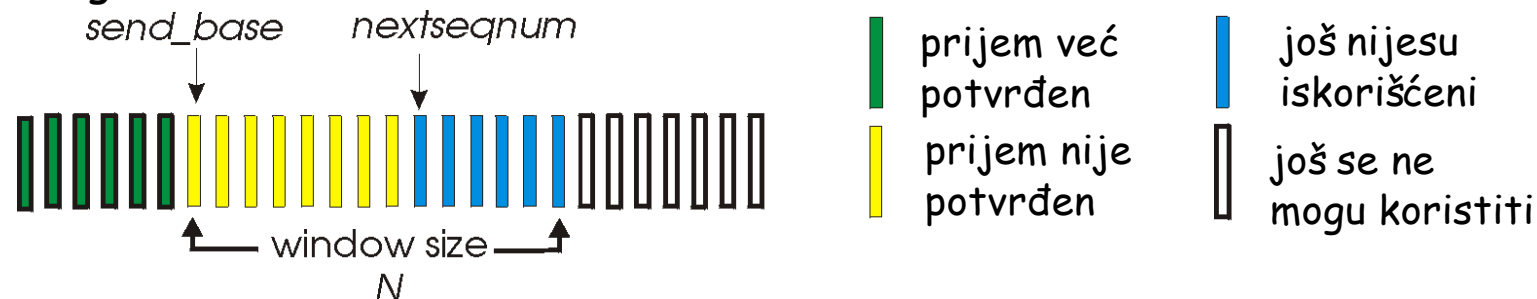
Selective Repeat:

- ❑ 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.
- ❑ Pošiljalac posjeduje **predajni**, a prijemnik **prijemni prozor**.

Go-Back-N (sliding window)

Pošiljalac:

- Broj u sekvenci u zaglavlju segmenta je dugačak k-bita što znači da se može poslati $N=2^k$ nepotvrđenih segmenata
- Prozor veličine N susjednih nepotvrđenih segmenata je dozvoljen
- Zašto ograničavati N?

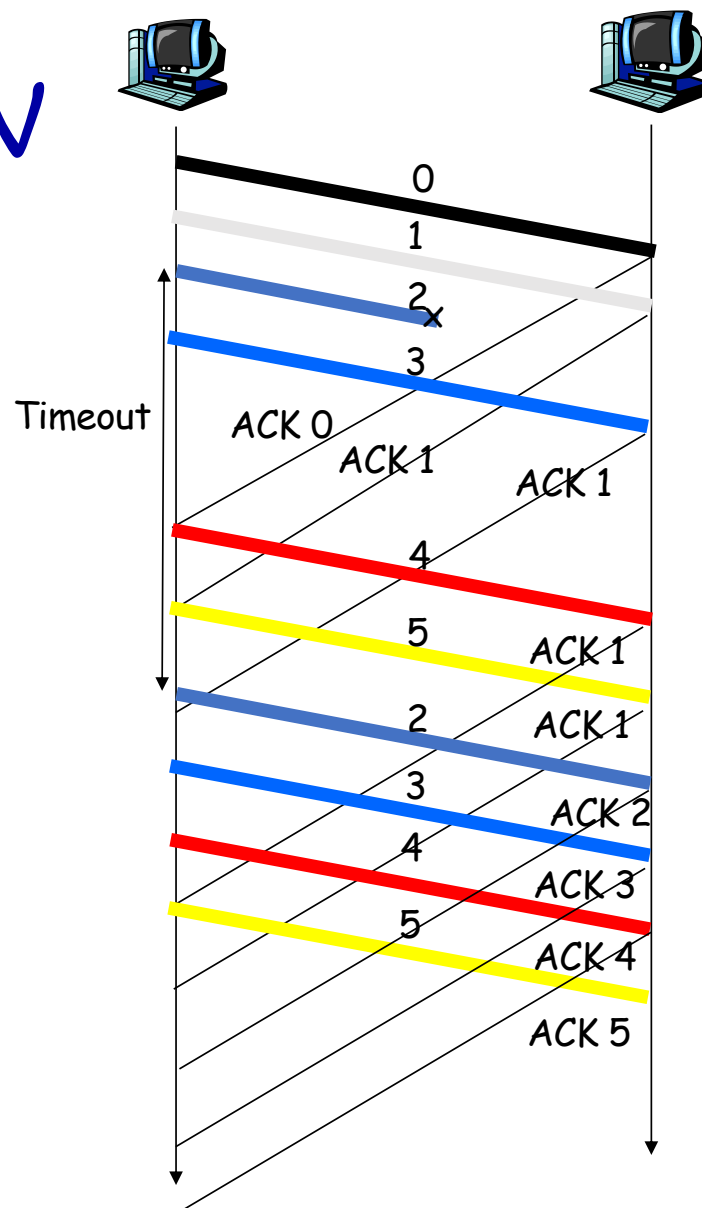


- Broj u sekvenci se upisuje u polje zaglavlja veličine k bita ($0, 2^k-1$). Kod TCP $k=32$, pri čemu se ne broje segmenti, već bajti u bajt streamu.
- ACK(n): ACK sve pakete, uključujući n-ti u sekvenci - **kumulativni ACK**
 - Mogu se pojaviti duple ACK potvrde (obratiti pažnju na prijemnik).

Go-Back-N (sliding window)

- ❑ Timer se inicijalizuje za "najstariji" segment i vezuje za svaki segment čiji prijem još nije potvrđen.
- ❑ Timeout(n): retransmisija segmenta n i svih segmenata čiji je broj u sekvenci veći od n, u skladu sa veličinom prozora.
- ❑ Uvijek se šalje ACK za korektno primljen segment sa najvećim brojem u sekvenci uz poštovanje **redosleda**
 - Može generisati duple ACK potvrde.
 - Treba da zapamti samo broj očekivanog segmenta.
- ❑ *Out-of-order* segment:
 - Odbacuje se -> **Zašto nema baferovanja na prijemu?**
 - Ponovo šalje ACK za segment sa najvećim brojem u sekvenci.

Go-Back-N

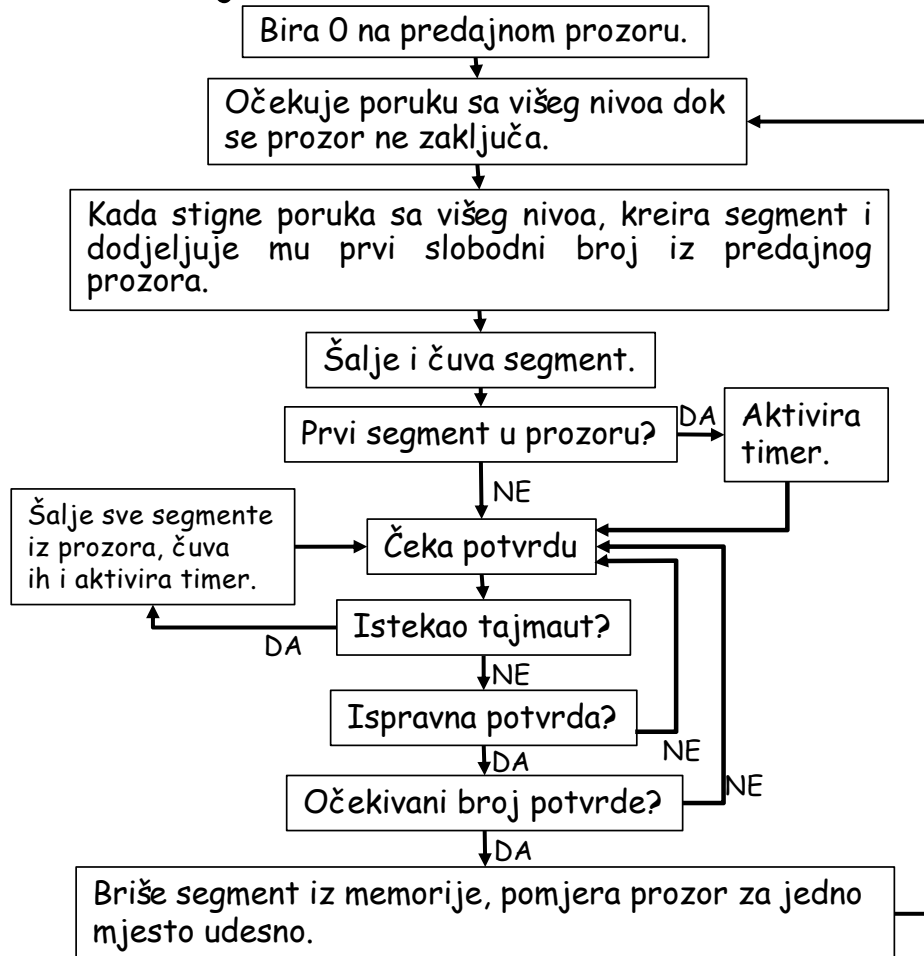


Go-Back-N

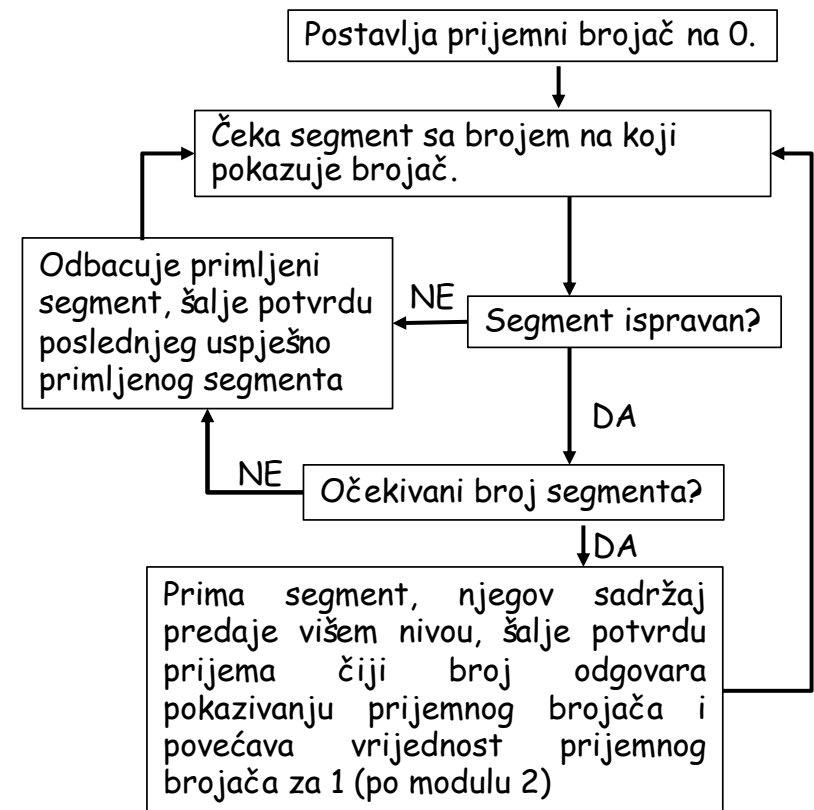
- ❑ Dozvoljava pošiljaocu da ispuni link segmentima, čime se uklanja problem lošeg iskorišćenja kanala.
- ❑ Sa druge strane, kada su veličina prozora i proizvod brzine prenosa i kašnjenja veliki mnogo segmenata može biti na linku. U tom slučaju zbog gubitka jednog segmenta mnogi segmenti se moraju iznova poslati (potpuno nepotrebno).
- ❑ Iz tog razloga se koriste **Selective Repeat** protokoli, koji kao što im ime kaže omogućavaju izbor segmenata koji će biti ponovo poslati.

Go-Back-N

Pošiljalac:



Prijemnik:



Selective Repeat

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

Selective Repeat

Pošiljalac

Podaci dolaze odozgo :

- Ako je sledeći broj u sekvenci u prozoru dostupan, poslati paket.

Timeout(n):

- Ponovo poslati segment n , restartovati tajmer.

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

- Markirati segment n kao da je primljen.
- Ako je n najmanji nepotvrđeni segment, proširiti osnovu prozora na bazi narednog najmanjeg broja nepotvrđenog segmenta.

Prijemnik

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

- Poslati ACK(n)
- *Out-of-order*: baferovati
- *In-order*: predati (takođe baferovati, predati u *in-order*), povećati prozor na sledeći segment koji još nije primljen

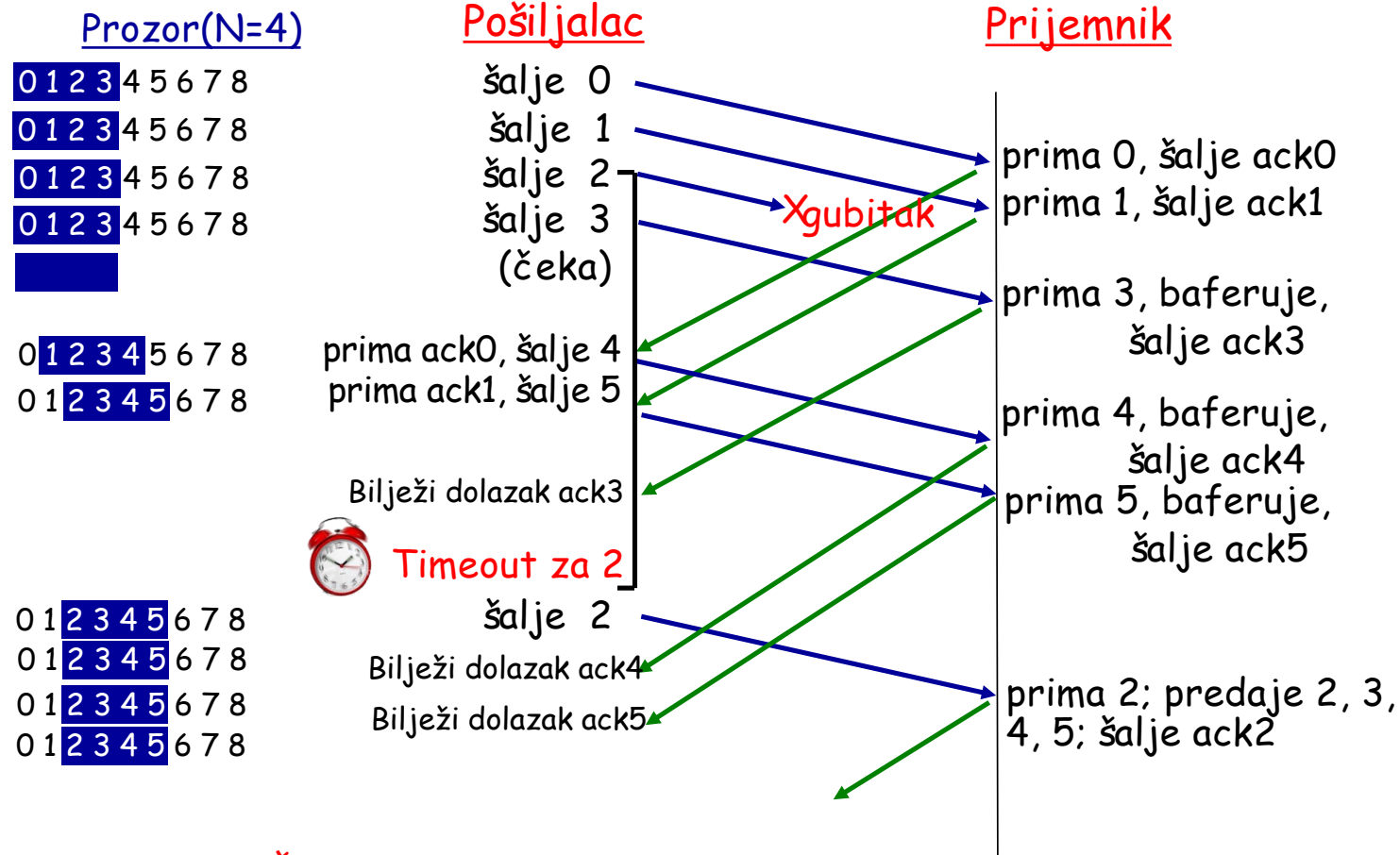
Segment n u [rcvbase-N, rcvbase-1]

- ACK(n)

Drugačije:

- ignorisati

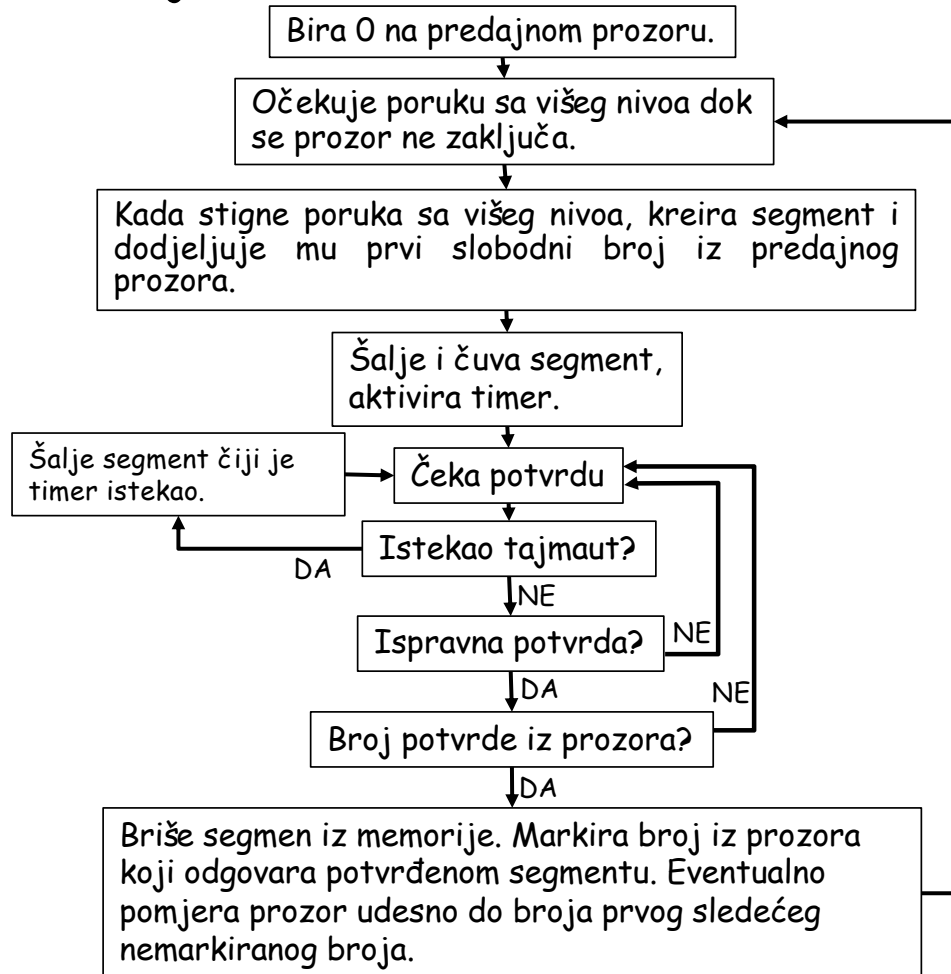
Selective Repeat



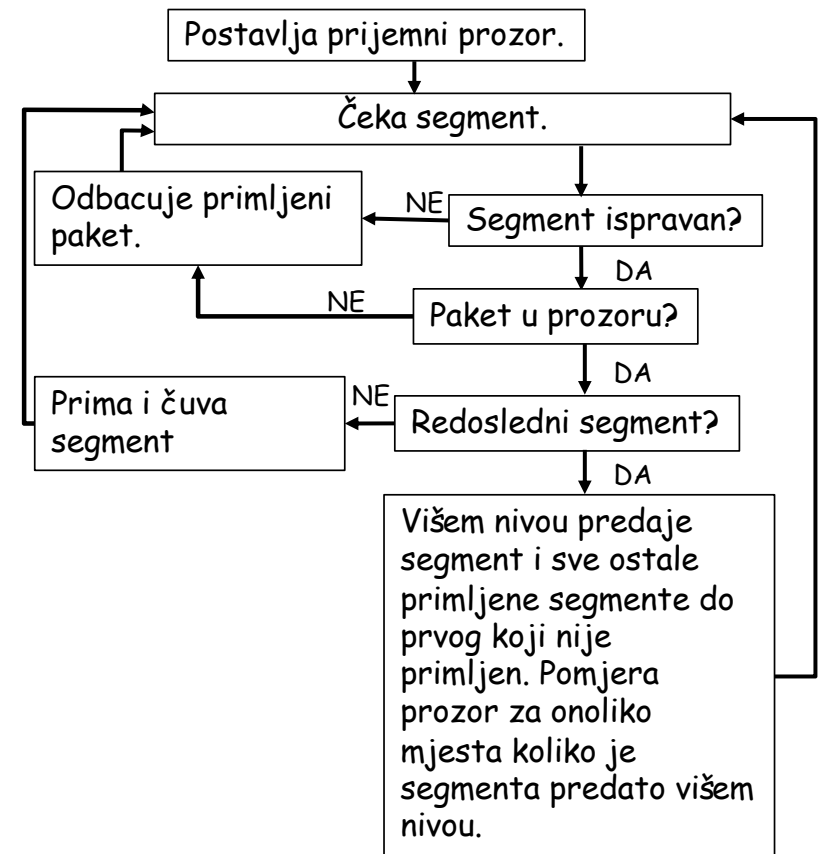
Šta se događa kada ack2 stigne?

Selective Repeat

Pošiljalac:

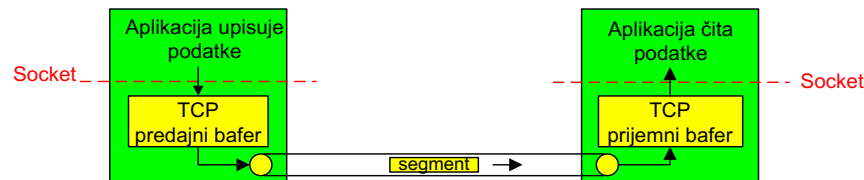


Prijemnik:



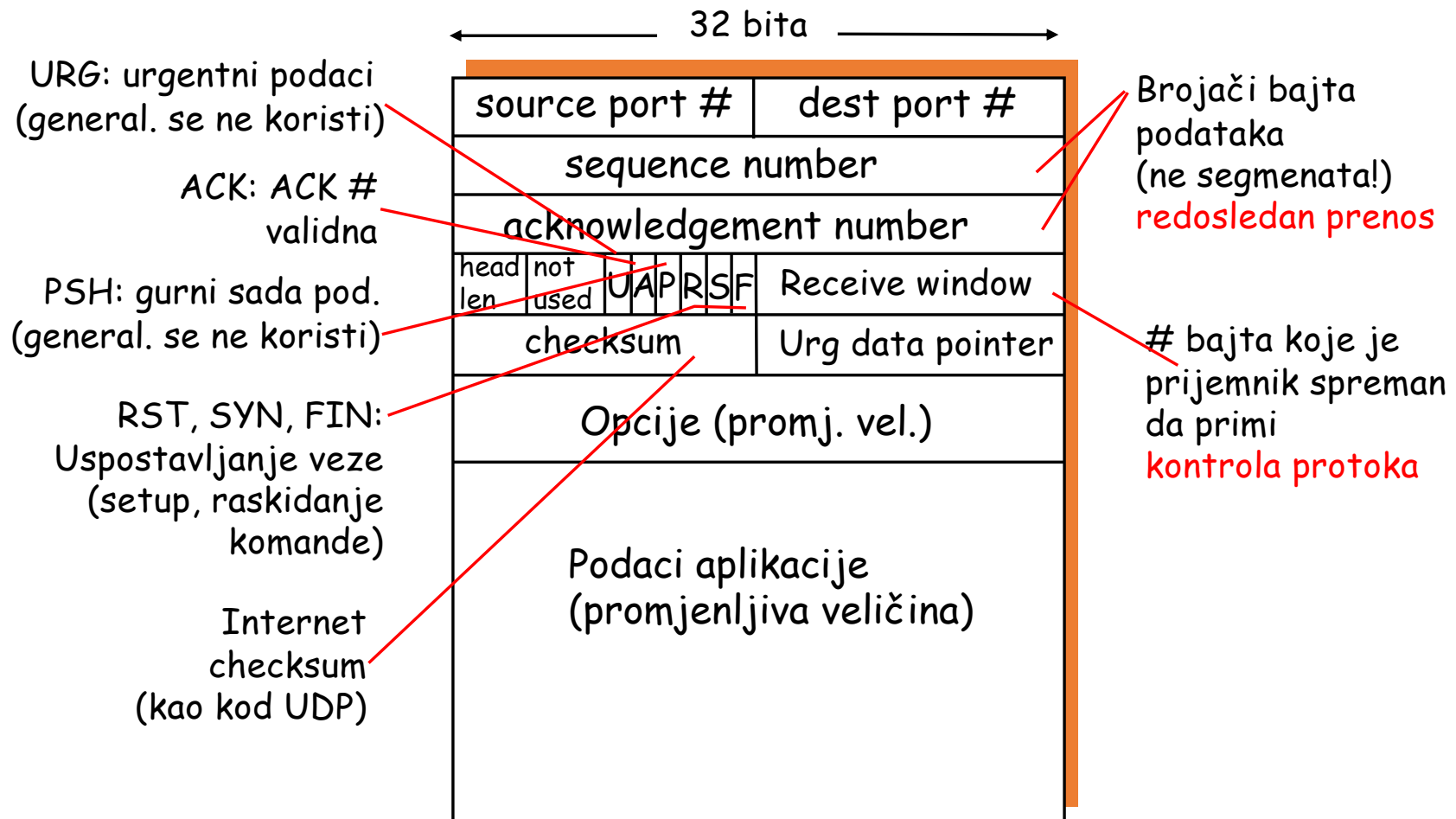
TCP - RFC [793, 1122, 1323, 2018, 2581]

- ❑ Tačka-tačka:
 - Jedan pošiljalac, jedan prijemnik
- ❑ Pouzdan i redosledan *pipelined* prenos bajta:
 - Nema "granica poruka".
 - TCP kontrola zagušenja i protoka podešavaju veličinu prozora.
- ❑ Baferi za slanje & prijem



- ❑ *Full duplex* prenos:
 - U istoj vezi prenos u dva smjera.
 - MSS: maksimalna veličina podataka sloja aplikacije u segmentu (1460B, 536B, 512B)
- ❑ Konektivan: **kasnije u RM!**
 - *Handshaking* (razmjena kontrolnih poruka) inicira je pošiljalac, razmjenjuje stanja prije slanja.
- ❑ Kontrola protoka: **kasnije u RM!**
 - Pošiljalac ne može "zagušiti" prijemnika.
- ❑ Kontrola zagušenja **kasnije u RM!**

Struktura TCP segmenta



TCP pouzdani prenos

- ❑ TCP kreira pouzdani prenos po IP nepouzdanom servisu.
- ❑ *Pipelined* segmenti.
- ❑ Kumulativne potvrde
- ❑ TCP koristi jedan retransmisioni tajmer.
- ❑ Retransmisije su trigerovane sa:
 - Timeout događajima.
 - Duplim ack-ovima.
- ❑ Razmatra se pojednostavljeni TCP pošiljalac:
 - Ignorišu se duplirani ACK segmenti.
 - Ignorišu se kontrole protoka i zagušenja.

TCP pouzdani prenos (pošiljalac)

1. Podaci primljeni od aplikacije:

- ❑ Kreiranje segmenta sa odgovarajućim brojem u sekvenci.
- ❑ Broj u sekvenci je byte-stream broj prvog bajta podataka u segmentu.
- ❑ Startuje se tajmer (ako to već nije urađeno).
- ❑ *Timeout* interval se izračunava po odgovarajućoj formuli.

2. Timeout:

- ❑ Ponovo se šalje segment koji je izazvao timeout.
- ❑ Restartuje se tajmer.

3. Ack primljen:

- ❑ Ako se potvrdi prijem ranije nepotvrđenog segmenta treba :
 - napraviti odgovarajući update.
 - startovati tajmer ako postoje segmenti koji čekaju.

TCP pouzdani prenos (prijemnik)

Događaj na prijemu	TCP akcije prijemnika
Dolazak <i>in-order</i> segmenta sa očekivanim brojem u sekvenci. Svi podaci do očekivanog broja su potvrđeni.	ACK sa kašnjenjem. Čeka do 500ms za sledeći segment. Ako nema sledećeg, šalje ACK.
Dolazak <i>in-order</i> segmenta sa očekiv. brojem u sekvenci. Potvrđ. prijema drugog segmenta u toku.	Odmah šalje jednu kumulativnu ACK, potvrđujući oba <i>in-order</i> segmenta.
Dolazak <i>out-of-order</i> 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.

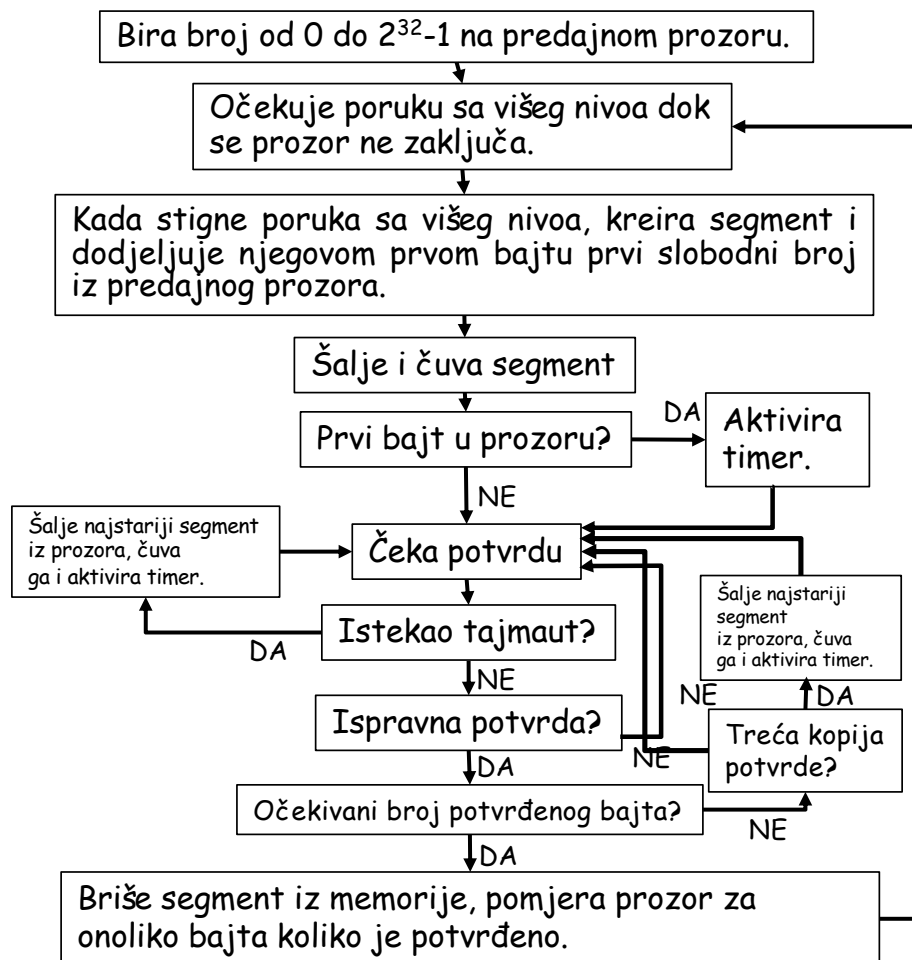
TCP pouzdani prenos (*Fast Retransmit*)

- *Time out 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, najvjerojatnije će biti dosta dupliranih potvrda ACK.
- Ako pošiljalac primi 3 ACK za iste podatke, pretpostavlja se da je segment poslije potvrđenog izgubljen.
 - *Fast retransmit* novo slanje segmenta prije nego što je tajmer istekao.

Da li TCP ima GBN ili SR kontrolu greške?
Zašto 3 a ne dva ACK?

TCP

Pošiljalac:



Prijemnik:

