

TCP i UDP - transportni protokoli koje koristi internet

Industrijski sistemi i protokoli -

Drugi deo – predavanje 6

05/2024

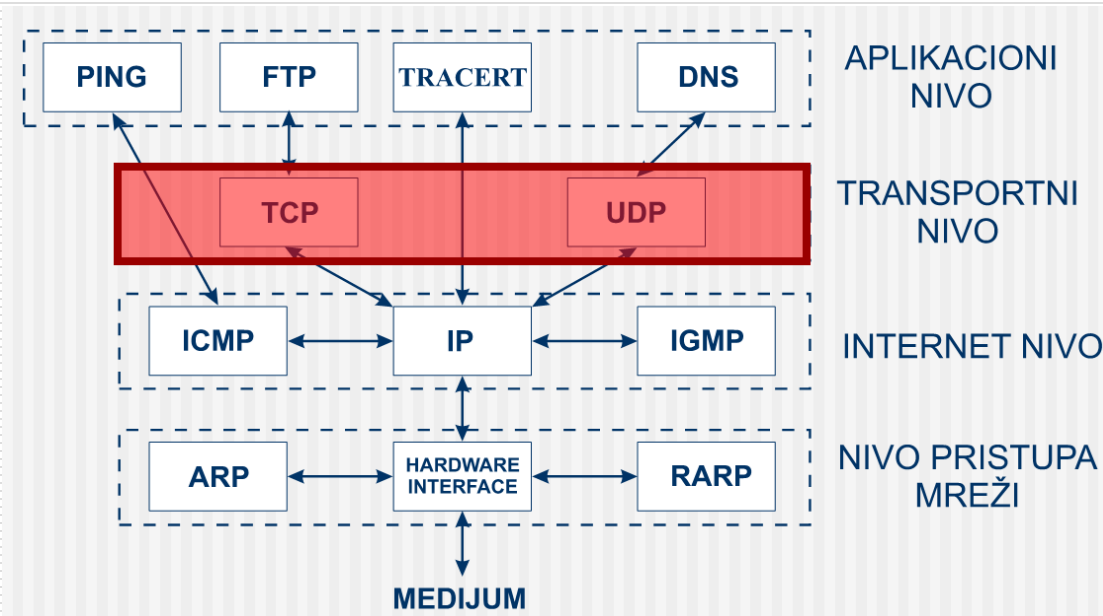
prof. Darko Marčetić

TCP/UDP transportni protokol

- **Transportni protokol je deo TCP/IP modela**
 - **Ethernet** je povezao dva čvora tokom putanje paketa
 - **IP protokol** je doveo paket do odredišne mreže i računara
 - **Transportni nivo (TCP ili UDP) protokoli** sada treba da **poveže taj paket** pristigao od IP protokola **sa aplikacijom** kojoj je namenjen.
 - Naravno, na polazištu su **TCP ili UDP** isto **povezali aplikaciju koja šalje podatke sa IP protokolom.**
 - **Transportni nivo** može da koristi ili **TCP** ili **UDP** protokol. Prvi je pouzdaniji ali sporiji, drugi je brži. Izbor zavisi od toga šta je kojoj aplikaciji potrebnije.

TCP/UDP transportni protokol

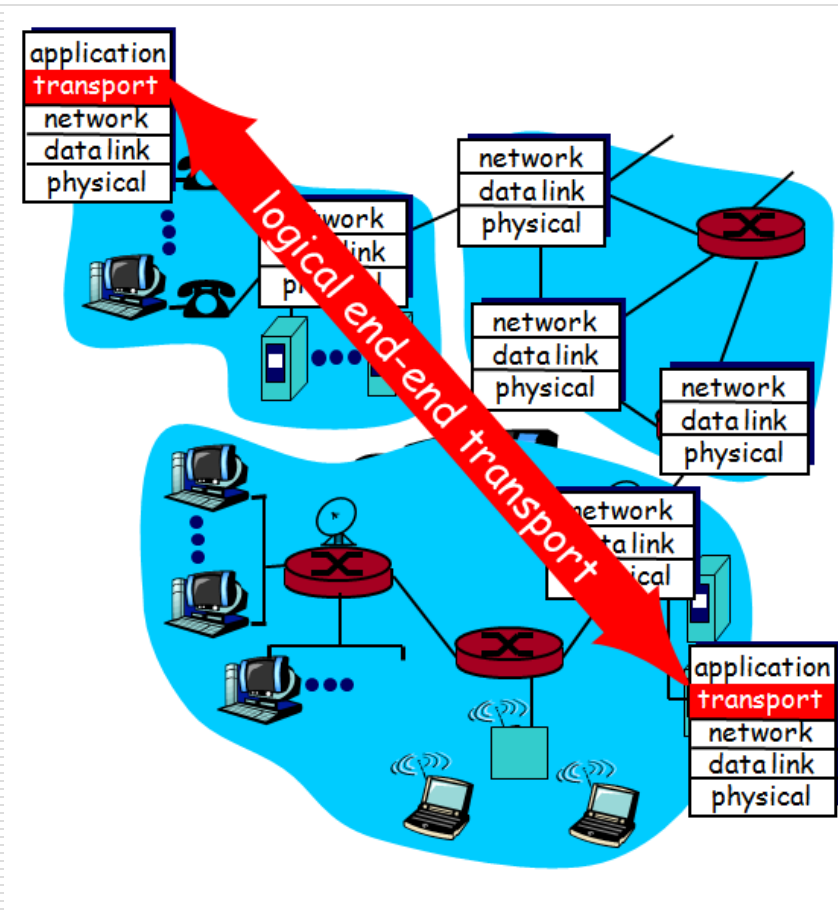
TCP/UDP protokol definišu pravila primenljiva na transportnom sloju koji se nalazi između aplikativnog i IP sloja



- **Aplikacija**
 - komunikacija specifična za aplikaciju
 - Primer: *HyperText Transfer Protocol (HTTP)*, *File Transfer Protocol (FTP)*, *Network News Transfer Protocol (NNTP)*
- **Transportni sloj**
 - komunikacija između procesa
 - oslanja se na mrežni sloj a opslužuje aplikacijski sloj
 - Da pošaljete nešto na internet imate dve mogućnosti : TCP i UDP
- **Mrežni sloj , za internet imate samo jednu mogućnost, IP protokol**
 - logičko povezivanje čvorova
 - sakriva tehničke detalje nižih slojeva

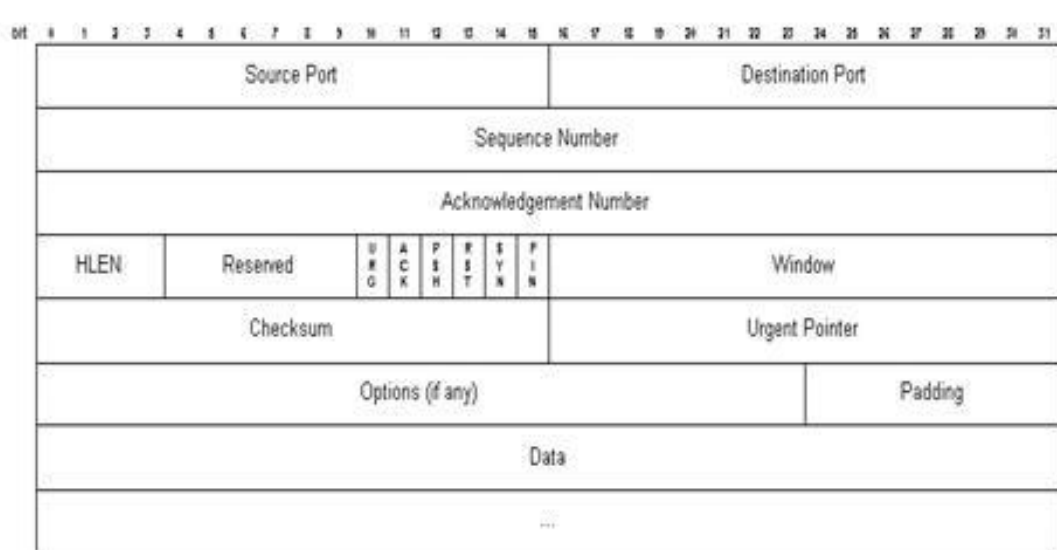
TCP/UDP transportni protokol

- **Transportni protokol ima ulogu samo na krajnjim tačkama putanje**
 - Pošiljaoc razbija aplikacijske podatke u segmente i predaje ih mrežnom sloju
 - Primalac rekonstruiše poruku na osnovu pristiglih segmenata
- **Aplikacija može da bira**
 - Između TCP i UDP
- **TCP protokol obezbeđuje:**
 - prilagođenje podataka za prenos IP protokolom
 - identifikaciju ciljne aplikacije
 - mutipleksiranje i demultipleksiranje
 - pouzdan prenos podataka,
 - isporuku podataka do destinacione aplikacije po tačno utvrđenom redosledu
 - Kontrola toka prenosa
- **UDP protokol obezbeđuje :**
 - prilagođenje podataka za prenos IP protokolom
 - identifikaciju ciljne aplikacije
 - Multipleksiranje i demultipleksiranje

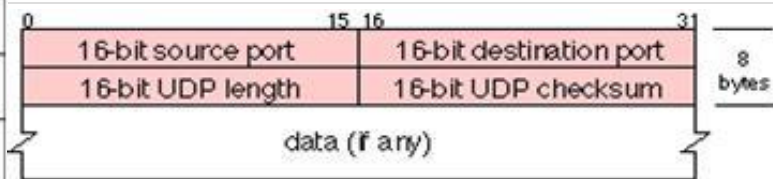


TCP protiv UDP protokola

TCP je složeniji i ima veće zaglavlje ali i zato nudi više od UDP protokola



TCP zaglavlje (header)



UDP zaglavlje (header)

- **TCP i UDP zaglavlje imaju zajedničko samo: source i destinacioni port i checksum**
- **TCP zaglavlje je mnogo veće i nosi više podataka**
- **To znači da je TCP sporiji ali donosi više provera**
 - **three-way handshake**
 - **Numeracija sekvenci**
 - **Detektovanu grešku TCP popravlja (error detection and recovery)**
 - **....**

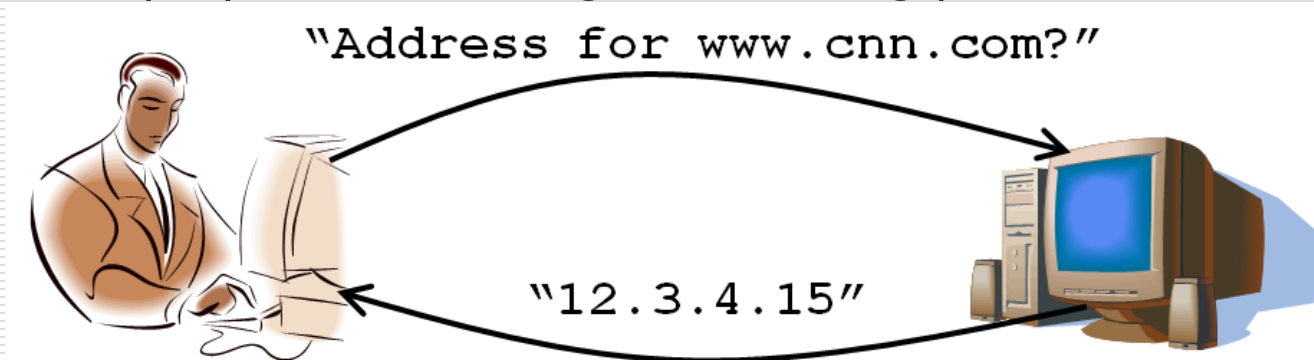
UDP protokol – prost, brz i nepouzdan

- ❑ **Da je TCP u svemu bolji ne bi ni bilo UDP !!!!**
- ❑ **Uprošćena komunikacija između procesa**
 - izbegava se stvaranje dodatnog protoka zbog ponavljanja i kašnjenje zbog zahteva za pouzdanom isporukom koja mora da ispoštuje tačno utvrđeni redosled.
 - poruka se šalje i prima od strane procesa
- ❑ **User Datagram Protocol (UDP)**
 - IP plus broj porta obezbeđuju (de)multiplesiranje
 - opciono može se ugraditi i provera ispravnosti paketa, u slučaju da provera pokaže da je paket oštećen on se odbacuje ALI bez zahteva za retransmisijom !

Popularne aplikacije koje koriste UDP

- **1. Multimedijalni, merni ili kontrolni strimovi podataka**
 - jednostavno , retransmisija izgubljenog ili oštećenog paketa je beskorisna kod ove vrste prenosa
 - svaki paket koji ne stigne u momentu kada se slika trebala reprodukovati postaje nepotreban.
 - npr telefonski poziv, video konferencija, igrice ... I konačno nama važno praćenje i upravljanje procesima u realnom vremenu !!!!!

- **2. Jednostavni upitni protokoli kao što je DNS (domain name servis, server koji na osnovu imena traži IP adrese)**
 - Ovde su korisne informacije su toliko male da bi dodatni protok za uspostavljanje pouzdane komunikacije bio prevelik teret
 - mnogo je racionalnije da aplikacija, ukoliko ne dobije odgovor, ponovi upit po isteku nekog vremenskog perioda.



TCP protokol –pouzdan

- ❑ ***Transmission Control Protocol (TCP)***
- ❑ **Konekciono orjentisani protokol**
 - jasno definisano uspostavljanje i prekidanje konekcije
- ❑ **Servis koji obezbeđuje tok bajtova**
 - TCP servisi šalju i primaju tok bajtova a ne poruka
- ❑ **Isporuka, pouzdana i odvija se po tačno utvrđenom redosledu**
 - oštećeni paketi se detektuju pomoću čeksuma,
 - potvrđivanje (*Acknowledgments*) i retransmisija omogućavaju pouzdanu isporuku
 - broj sekvence omogućava utvrđivanje tačnog rasporeda pristigih paketa i utvrđivanje činjenice da neki paket nedostaje
- ❑ **Kontrola toka**
 - omogućava da se izbegne prepunjavanje prijemnikovih bafera
- ❑ **Kontrola zagušenja**
 - omogućava da se raspoloživi propusni opseg koristi racionalno u cilju opšte dobrobiti.

Izazovi transporta podataka

- ❑ **Idealan slučaj je savršeni kanal**
 - Svi podaci stižu neoštećeni i po redosledu po kom su poslani
- ❑ **Transmisija preko kanala različitih kašnjenja**
 - Svi podaci stižu neoštećeni ali redosled se promeni
- ❑ **Transmisija preko kanala u kojem dolazi do pogrešnog prenosa pojedinih bitova**
 - svi podaci stižu po redosledu po kom su i poslani ali pojedini bitovi su neispravni
 - prijemnik otkriva grešku, i zahteva ponovno slanje
 - pošiljalac ponovo šalje zahtevane podatke
- ❑ **Transmisija preko jako lošeg kanala u kojem dolazi do pogrešnog prenosa pojedinih bitova ali i gubitka celih paketa**
 - neki podaci pristižu oštećeni a neki ni ne pristižu
 - primalac može detektovati oštećene podatke ali ne i izgubljene
 - zbog toga pošiljalac čeka na potvrdu prijema i ako je nema šalje ponovo iste podatke nakon isteka nekog perioda.

Svojstva TCP koja omogućavaju pouzdan prenos

☐ Čeksum (*Checksum*)

- koristi se za detekciju oštećenih paketa
- ... što dovodi do odbacivanja paketa

☐ Broj sekvence

- omogućava otkrivanje da neki paketi nedostaju
- omogućava slaganje pristiglih paketa u skladu sa redosledom po kojem su poslati

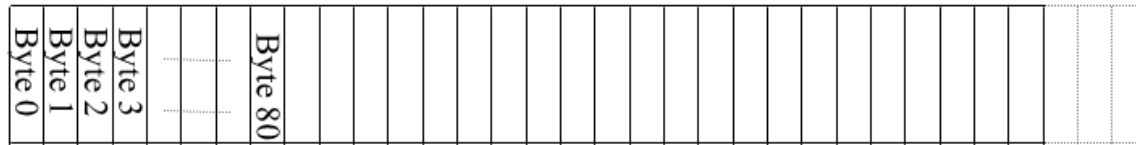
☐ Retransmisija

- pošiljalac ponovo šalje podatke u slučaju potrebe
- tajmaout (*Timeout*) baziran na procenjenom trajanju kružnog puta (*round-trip time*)
- algoritam za brzu retransmisiju

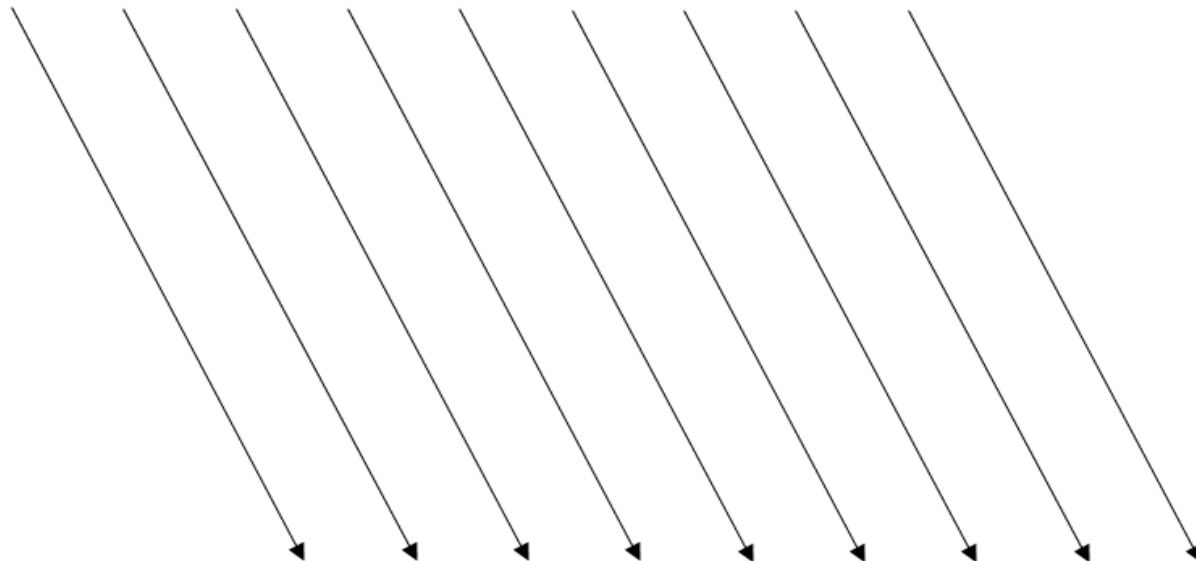
TCP protokol – Segmenti

TCP struja bajtova (*Stream of Bytes*)

Host A



Host B



Simulacija upotrebe TCP Segmenta

Host A



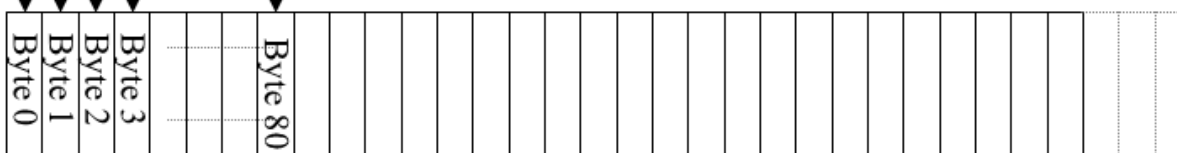
TCP Data

Segment se šalje ako važi:

1. Segment je pun (dostignuta maksimalna veličina segmenta)
2. Nije pun ali isteklo je vreme za punjenje
3. Segment poguran od strane aplikacije

TCP Data

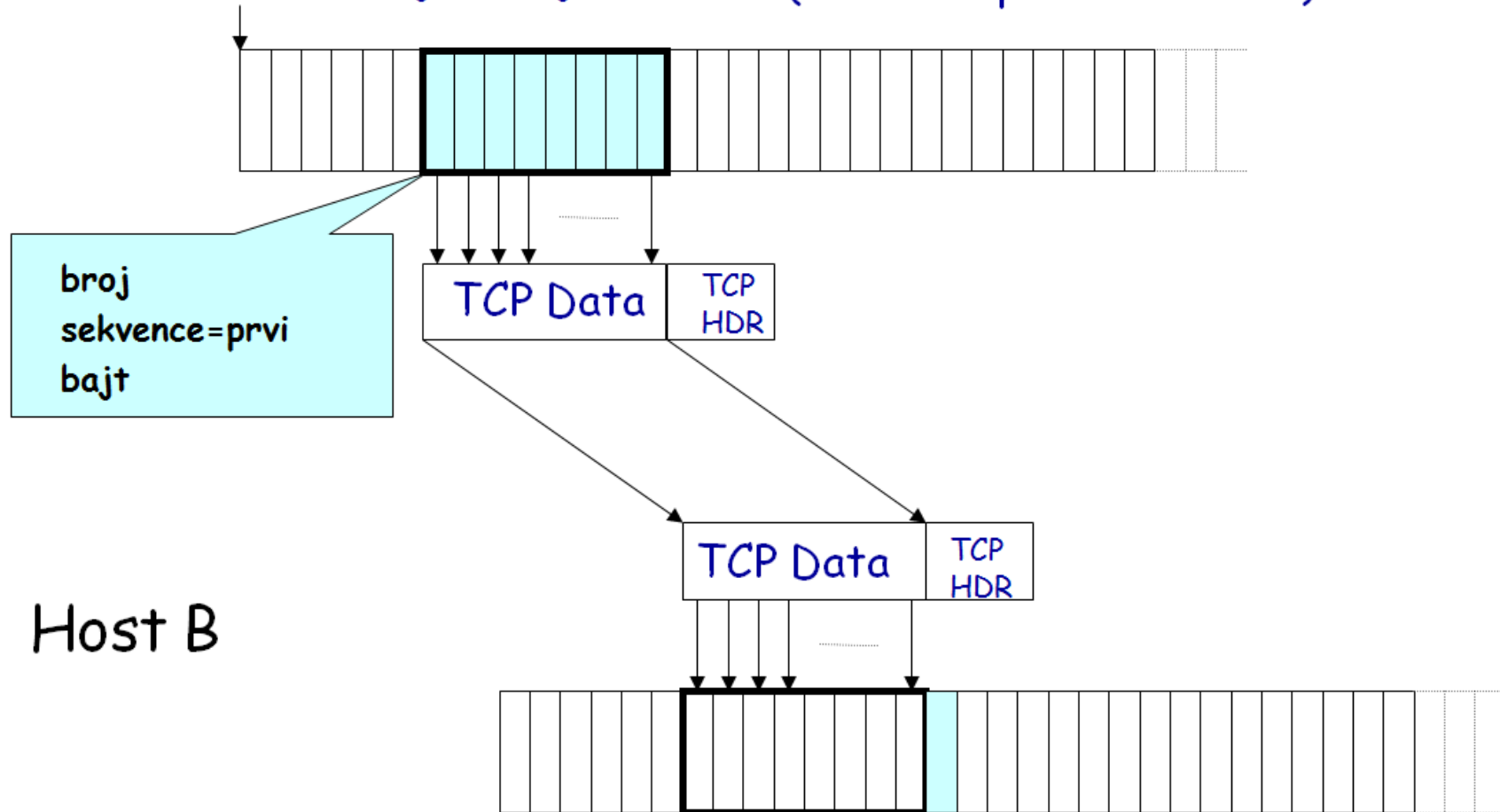
Host B



Broj sekvence

Host A

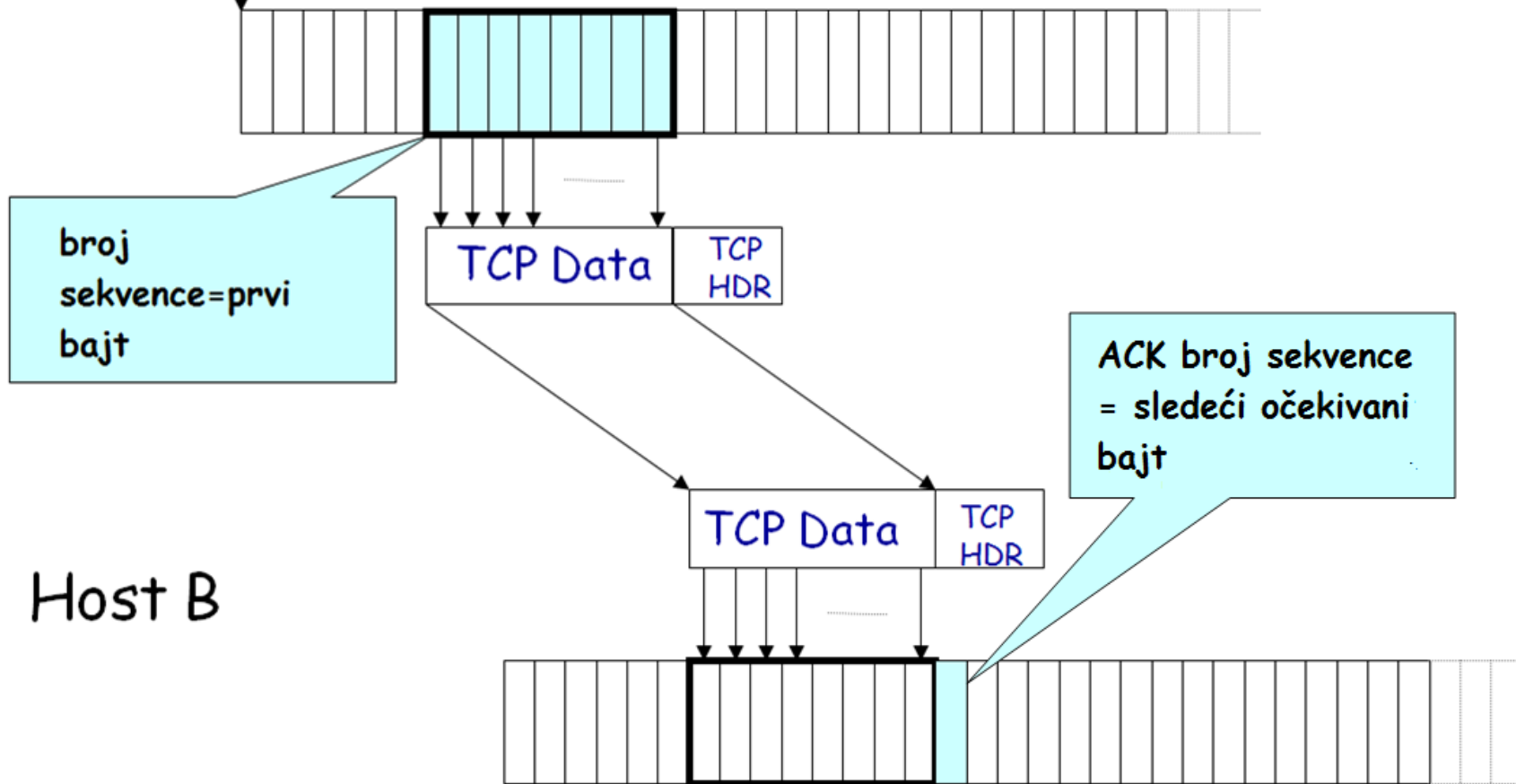
ISN Inicijalni broj sekvence (initial sequence number)



Broj sekvence

Host A

ISN Inicijalni broj sekvence (initial sequence number)



Host B

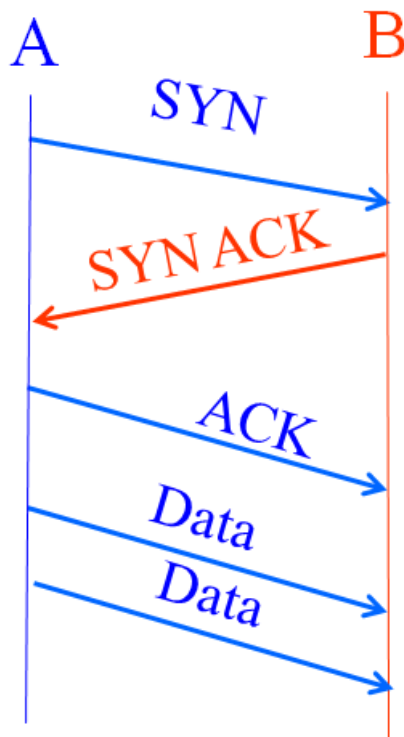
Inicijalni broj sekvence

- **Broj sekvence u kojoj se nalazi prvi bajt**
 - zašto ne bi taj broj bio 0?
- **Problem**
 - IP adresa i broj porta jedinstveno identifikuju konekciju
 - ipak, moguća su ponavljanja upotrebe iste adrese i broja porta
 - tada postoji mogućnost da se neka sekvenca iz stare konekcije ubaci u novu i napravi pometnju
- **Zbog toga neophodno je sa vremena na vreme korigovati inicijalni broj sekvence ISN**
 - ISN se očitava sa 32-bitnog časovnika koji otkucava svake 4 mikrosekunde,
 - znači ciklus traje oko 5 sati
- **Da bi se to moglo da funkcioniše pošiljac i primalac se moraju razmeniti inicijalne brojeve sekvenci**

TCP protokol – Trostruko rukovanje

Three-Way Handshake

TCP uspostavljanje konekcije



Svaki host šalje svoj
ISN drugom hostu

ISN – initial sequence number

Izabran random na predaji i šalje se prijemnom hostu tokom početne sinhronizacije.

□ Trostruko rukovanje

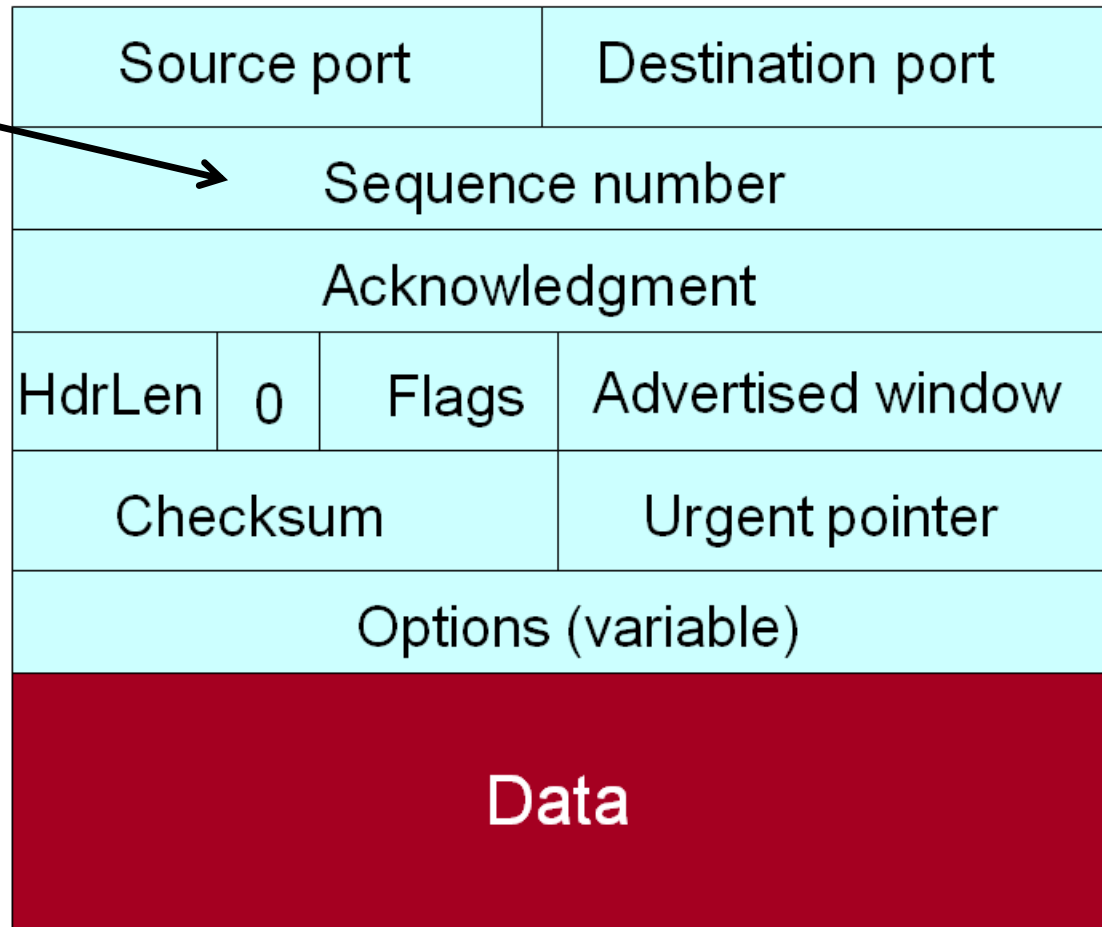
- Host A šalje SYN(open) Hostu B
- Host B šalje SYN ACK (potvrdu prijema) SYN poruke Hostu A
- Host A potvrđuje SYN ACK primljenu od Host B

ISN – initial sequence number

TCP zaglavlje

**Prvi ISN je izabran
randum na predaji**

Flags: SYN
FIN
RST
PSH
URG
ACK



Korak 1: A šalje inicijalni SYN paket

Flags: SYN
FIN
RST
PSH
URG
ACK

A's port		B's port	
A's Initial Sequence Number			
Acknowledgment			
20	0	Flags	Advertised window
Checksum			Urgent pointer
Options (variable)			

A poručuje B da želi da uspostavi konekciju

Korak 2: B odgovara A sa SYN-ACK paket

Flags: SYN
FIN
RST
PSH
URG
ACK

B's port		A's port	
B's Initial Sequence Number			
A's ISN plus 1			
20	0	Flags	Advertised window
Checksum			Urgent pointer
Options (variable)			

B poručuje A da prihvata poziv, i da očekuje sledeći bajt

...po prijemu ovog paketa, A može da počne slanje podataka

Korak 3: A šalje B ACK na SYN-ACK paket

Flags: SYN
FIN
RST
PSH
URG
ACK

A's port			B's port		
Sequence number					
B's ISN plus 1					
20	0	Flags	Advertised window		
Checksum			Urgent pointer		
Options (variable)					

A poručuje B da može da počne sa slanjem podataka

...po prijemu ovog paketa B može da otpočne sa slanjem podataka

Šta ako se SYN paket izgubi?

- ❑ **Pretpostavimo da se SYN paket izgubio**
 - paket izgubljen unutar mreže
 - paket odbačen od strane servera jer je pristupni bafer prepunjen.

- ❑ **Detekcija, nije pristigao SYN ACK**
 - po slanju SYN paketa pošiljalac je aktivirao tajmer
 - ukoliko po isteku vremena ne stigne SYN-ACK paket se šalje ponovo

- ❑ **Problem podešavanje tajmera**
 - pošiljalac nema pojma koliko je prijemnik daleko,
 - obično se kao standardna vrednost koristi podešavanje od 3 do 6 sekundi
 - ta vrednost se kod neki aplikacija može povećavati
 - mada, ukoliko je odziv veći od 6 sekundi aplikacija će imati veoma loše preformanse

Gubitak SYN i WEB *download*

□ **Korisnik klikne na link**

- pretraživač kreira soket(*socket*) i pokreće povezivanje (*connect*)
- pokretanje povezivanja izaziva slanje SYN paketa

□ **Ukoliko se SYN izgubi**

- vreme od 3-6 sekundi može izgledati veoma dugo
- korisnik postaje nestrpljiv
- ponovo klikne na link ili na "osveži" (*reload*)

□ **Korisnik pokreće "prekid" povezivanja (*abort of the connect*)**

- pretraživač kreira novi soket i pokreće ponovo "povezivanje",
- u osnovi, nametnuli smo brže slanje novog SYN paketa
- ponekad to može biti efikasno

TCP protokol – retransmisija

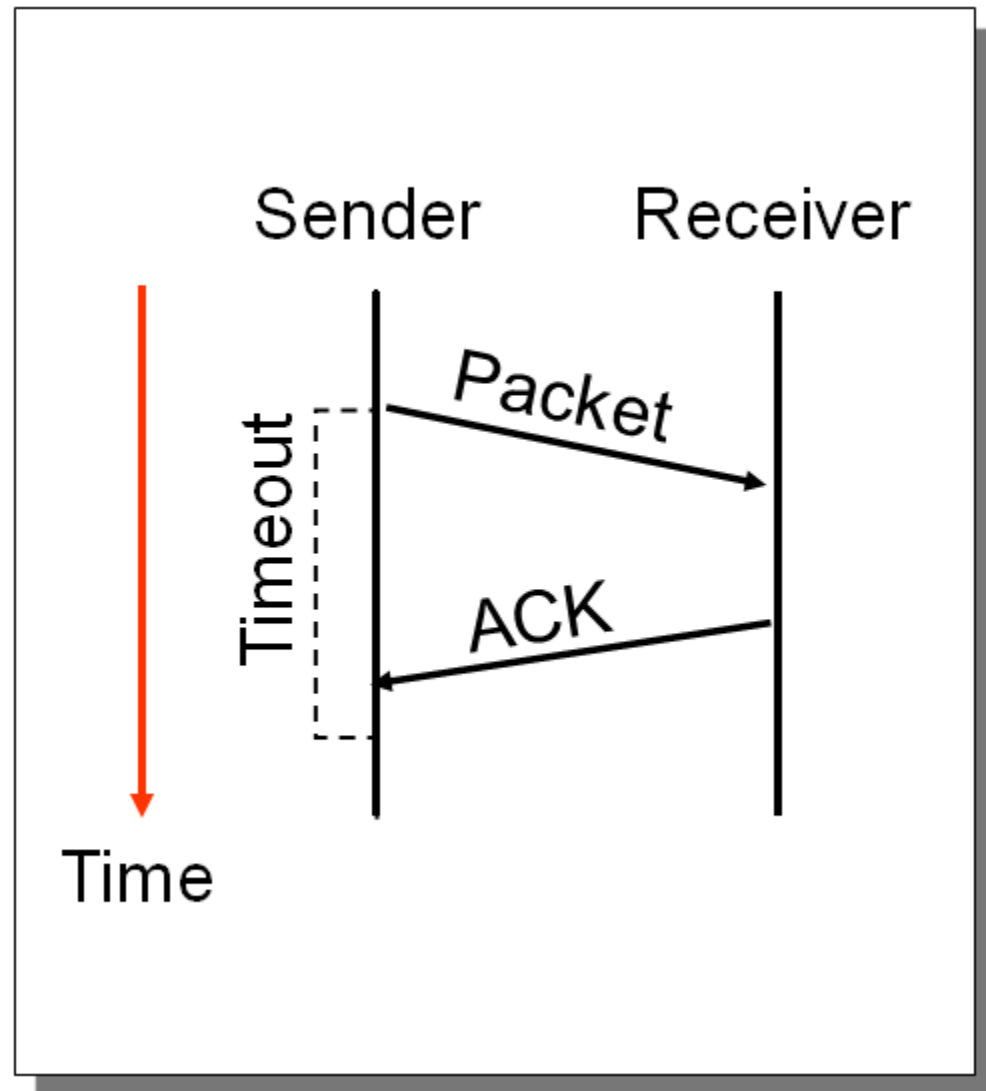
Automatski zahtev za retransmisiju

□ Automatski zahtev za retransmisiju (*Automatic Repeat reQuest* (ARQ))

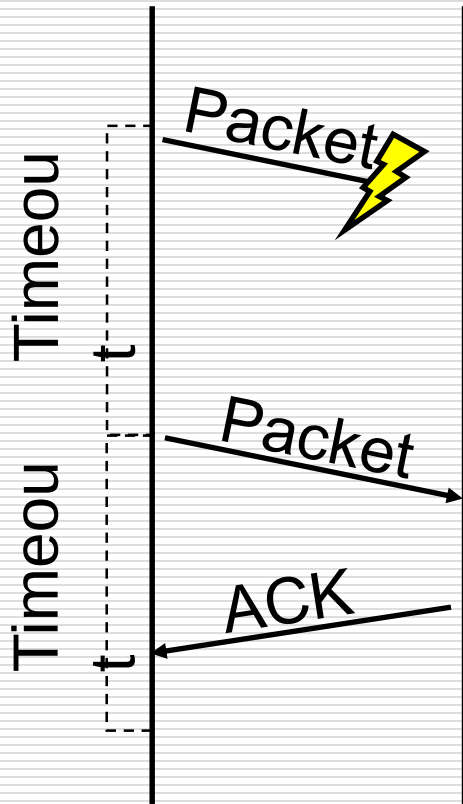
- prijemnik po prijemu paketa šalje ACK
- pošiljalac čeka na ACK sve dok ne istekne vreme određeno tajmerom

□ Najjednostavniji ARQ protokol

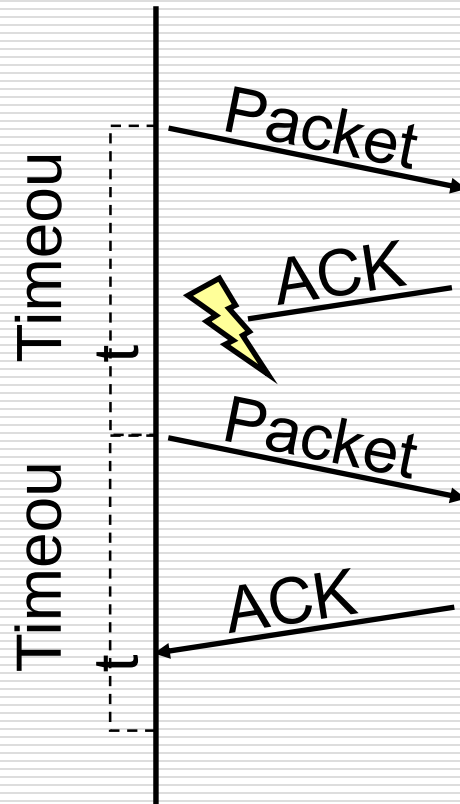
- Stani i čekaj
- Pošalji paket , stani i čekaj na ACK



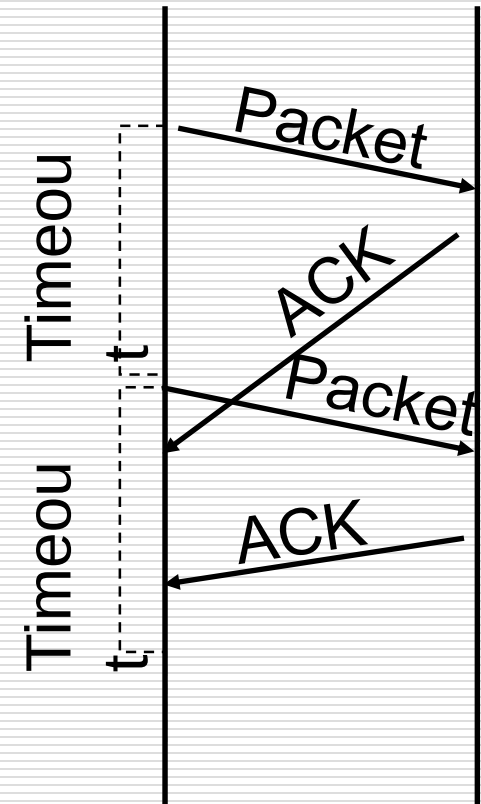
Razlozi za retransmisiju



Paket izgubljen



ACK izgubljen
DUPLI
PAKET



kratak tajmaut
DUPLIRANI
PAKETI

Koliko dugo pošiljalac treba da čeka

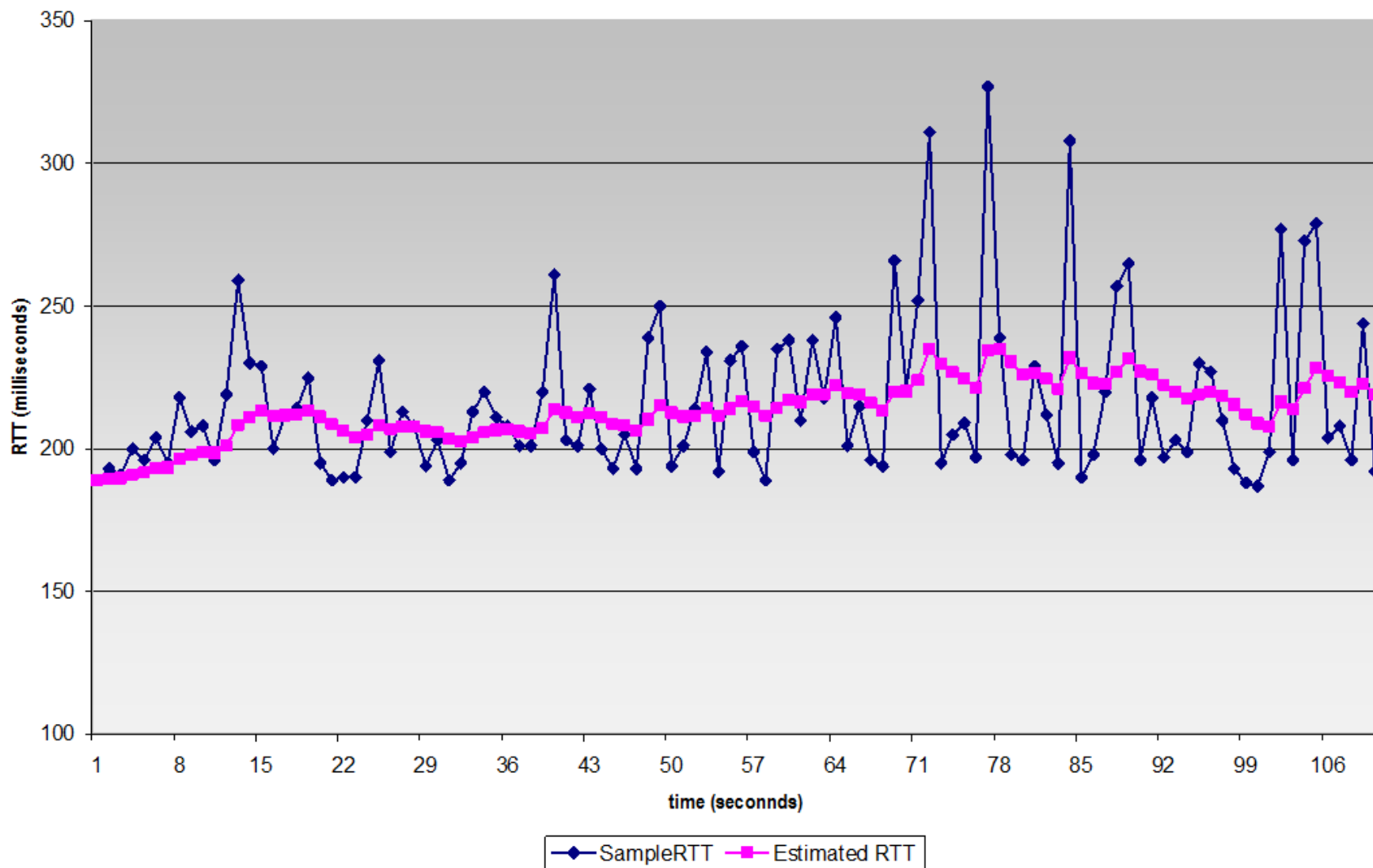
- ❑ **Pošiljalac podešava tajmout za čekanje na ACK**
 - ako je prekratak, rasipanje resursa nepotrebnim retransmisijama
 - predugačak, može se desiti da aplikacija funkcioniše ali zbog loših preformasi može postati neupotrebljiva

- ❑ **TCP podešava tajmout u fukciji trajanja kružnog puta (*round-trip time* RTT)**
 - ACK se očekuje nakon isteka RTT
 - plus dodajemo proizvoljni interval da bi zaračunali procesiranje podataka

- ❑ **Pitanje, kako da procenimo RTT**
 - procenimo na osnovu RTT za ACK
 - zatim se prati srednja vrednost RTT

Primer procenjivanje RTT

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



Mane prethodnog pristupa

- ❑ **Posmatrajmo retransmisiju izgubljenog paketa**
 - po prijemu ACK ne znamo da li je ACK potvrda prijema prvog ili drugog paketa
 - ako pretpostavimo da je ACK potvrda prijema prvog paketa onda će uzorak RTT biti prevelik

- ❑ **Slučaj dupliranih paketa**
 - ako pretpostavimo da je ACK potvrda prijema drugog poslatog onda će uzorak RTT biti premali

- ❑ **Jednostavno rešenje Karn/Partridge algoritam**
 - Za procenu RTT uzimamo samo ACK na pakete koji su poslani samo jednom

Dodatno ograničenje

□ Zanemarujemo brze promene RTT

- Ukoliko su varijacije izmerenog RTT male tada će procenjeni RTT (dobijen usrednjavanjem svih uzoraka) dobro pratiti realni RTT
- međutim ukoliko imamo velike i brze varijacije izmerenog RTT tada će procenjeni RTT značajno odstupati od realnog RTT

□ Rešenje je da direktno uračunavamo varijacije

- $\text{Procenjeni RTT}(i+1) = \text{Procenjeni RTT}(i) + k * (\text{izmereni RTT}(i) - \text{procenjeni RTT}(i))$
- za svako slanje paketa

TCP protokół – klizni prozor

Sliding Window

Razlozi za klizni prozor

□ Stani i čekaj je neefikasno

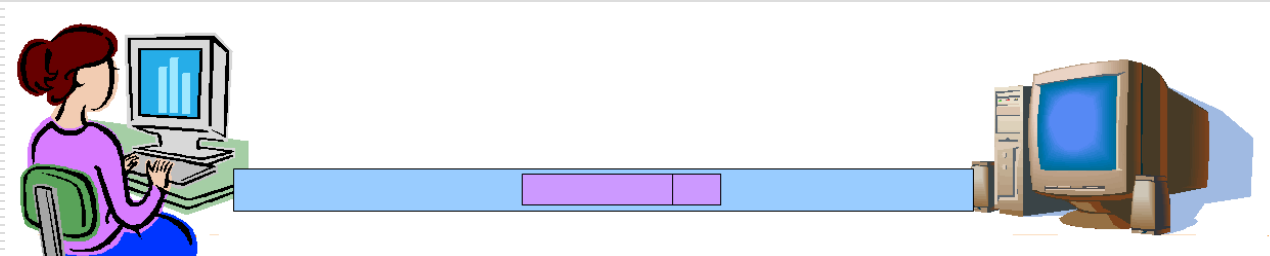
- U ovom slučaju samo jedan segment je u pokretu
- Ovo je naročito loše ukoliko je proizvod kašnjenja i propusnog opsega (*delay-bandwidth product*) velik

□ Primer Imamo link brzine 1,5Mb/s sa vremenom kružnog puta (RTT) od 45ms

- Znači imamo *delay-bandwidth product*=67,5 Kbits ili 8KBytes

□ Pošto možemo da šaljemo samo jedna paket za jedan RTT

- Tada ako je segment veličine 1KB (1024Bytes maks je 1500) to je 8Kbit.
- Svakih 45ms prenosimo jedna segment i to daje
- brzina=8Kb * (1/45ms)=182Kb/sekudi
- To je 1/8 od ukupnog propusnog opsega od 1,5 Mb/s



Klizni prozor

- **Pri rukovanju prijemnik i predajnik se dogovore o veličini prozora**
 - broj semenata koji će se poslati bez potvrde

- **Ograničenje propusni opseg**
 - U prethodnom primeru nema smisla slati više od 8 segmenata istovremeno jer tada važi
 - brzina = $8 \times 8\text{Kb} \times (1/45\text{ms}) = 1.5\text{Mb/sekudi}$

- **Bafer prijemnika**
 - ako je link brži od onog u primeru ograničava nas bafer prijemnika
 - prijemnik mora imati bafer u kojem drži pakete za koje još nije poslao potvrdu i veličina bafera ograničava broj segmenata
 - mora se proveriti da li su svi paketi stigli i da li su svi neoštećeni

TCP zaglavlje prijemni bafer

- pri rukovanju prijemnik oglašava prozor koji on može da prihvati
 - Predajnik usaglasi sa svojim mogućnostima

Flags: SYN
FIN
RST
PSH
URG
ACK

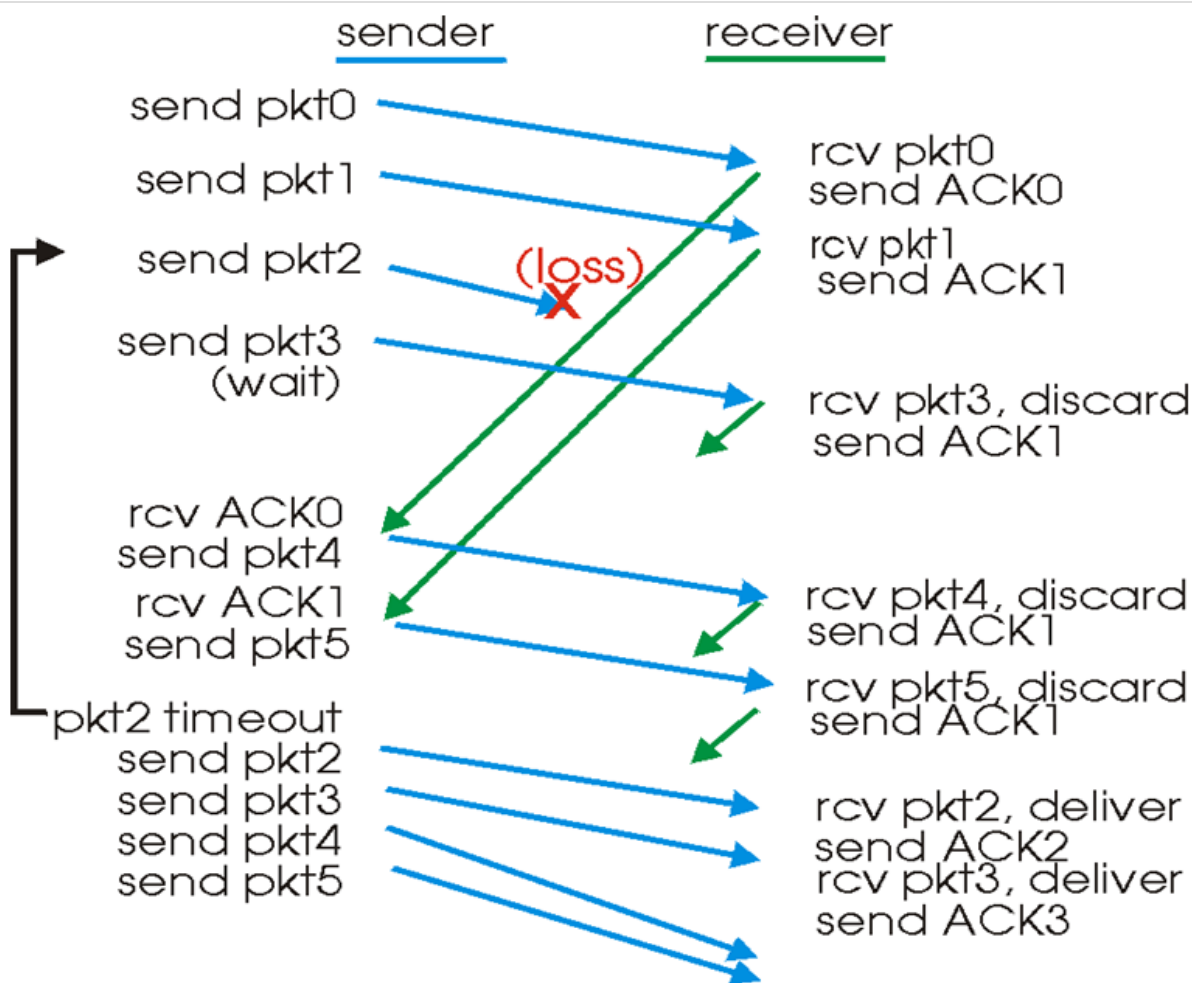
Source port		Destination port	
Sequence number			
Acknowledgment			
HdrLen	0	Flags	Advertised window
Checksum		Urgent pointer	
Options (variable)			
Data			

TCP protokol – brza retransmisija

Tajmaut je neefikasan

□ Retransmisija bazirana na tajmautu

- paket se pošalje i predajnik čeka da istekne tajmer
- kad tajmer istekne paket se ponovo šalje



Brza retransmisija

□ Problem

- Izgubili smo n -ti paket ali $n+1$ i $n+2$ itd... su stigli

□ Ideja: pijemnik šalje ACK za svaki pristigli paket zahtevajući n -ti

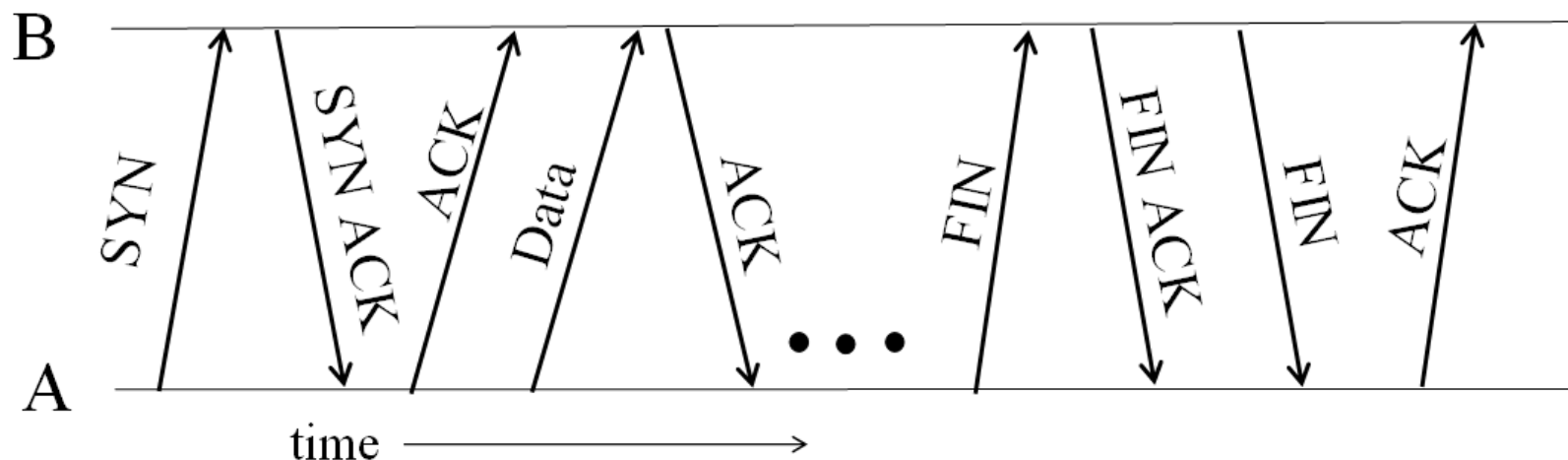
- ACK javlja da se čeka na n -ti paket ponovljeni ACK za n -ti paket znači da je sledeći paket stigao
 - ponovljeni ACK za n -ti paket znači da je sledeći paket stigao
- Pošiljalac uočava duplirani ACK sa zahtevom za n -ti paket
 - pretpostavlja da je n -ti paket izgubljen
 - i posle tri primljena ACK ponovo šalje paket

TCP protokol – prekidanje konekcije

Prekidanje konekcije

□ Prekidanje konekcije

- predajnik šalje (FIN) prijemniku koji prima poslednje bajtove i prekida konekciju
- prijemnik odgovara sa (FIN ACK)
- prijemnik šalje reset RTS za trenutni prekid konekcije bez prijema poslednjih bajtova



Prekidanje konekcije

□ Poslato FIN: close()

- Proces je završio slanje podataka preko soketa
- Proces poziva "close()" za zatvaranje soketa
- TCP iz bafera šalje preostale bajtove...
- ... zatim TCP šalje FIN

□ Prijem FIN: EOF

- Proces očitava podatke iz soketa
- Eventualno, pri očitavanju prijemnik očitava EOF i vraća potvrdu

Transportni protokoli – zaključak:

□ **UDP nepouzdan ali brz**

- Ubacuje pakete ka IP protokolu bez proveriti da li tamo ima nekog da ih primi i da li su stigli

□ **TCP pouzdan**

- Trostruka sinhronizacija pre starta slanja
- Prati broj sekvence, zna koji paket kome pripada
- Kontrolise tok primenom kliznog prozora
- Omogucava retransmisija baziranu na tajmoutu
- Detekcija greške bazirana na ceksumu