Preddiplomski studij

Računarstvo

Komunikacijske mreže

8. Transportni protokoli u Internetu: TCP i UDP (2. dio)

Ak.g. 2014./2015.

slobodno smijete:

- **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
- **remiksirati** — prerađivati djelo

pod sljedećim uvjetima:

- **imenovanje**. Morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
- **nekomercijalno**. Ovo djelo ne smijete koristiti u komercijalne svrhe.
- **dijeli pod istim uvjetima**. Ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencijske uvjete ovog djela. Najbolji način da to učinite je poveznicom na ovu internetsku stranicu.
Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.
Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.

Tekst licencije preuzet je s http://creativecommons.org/.

Protokoli transportnog sloja u Internetu

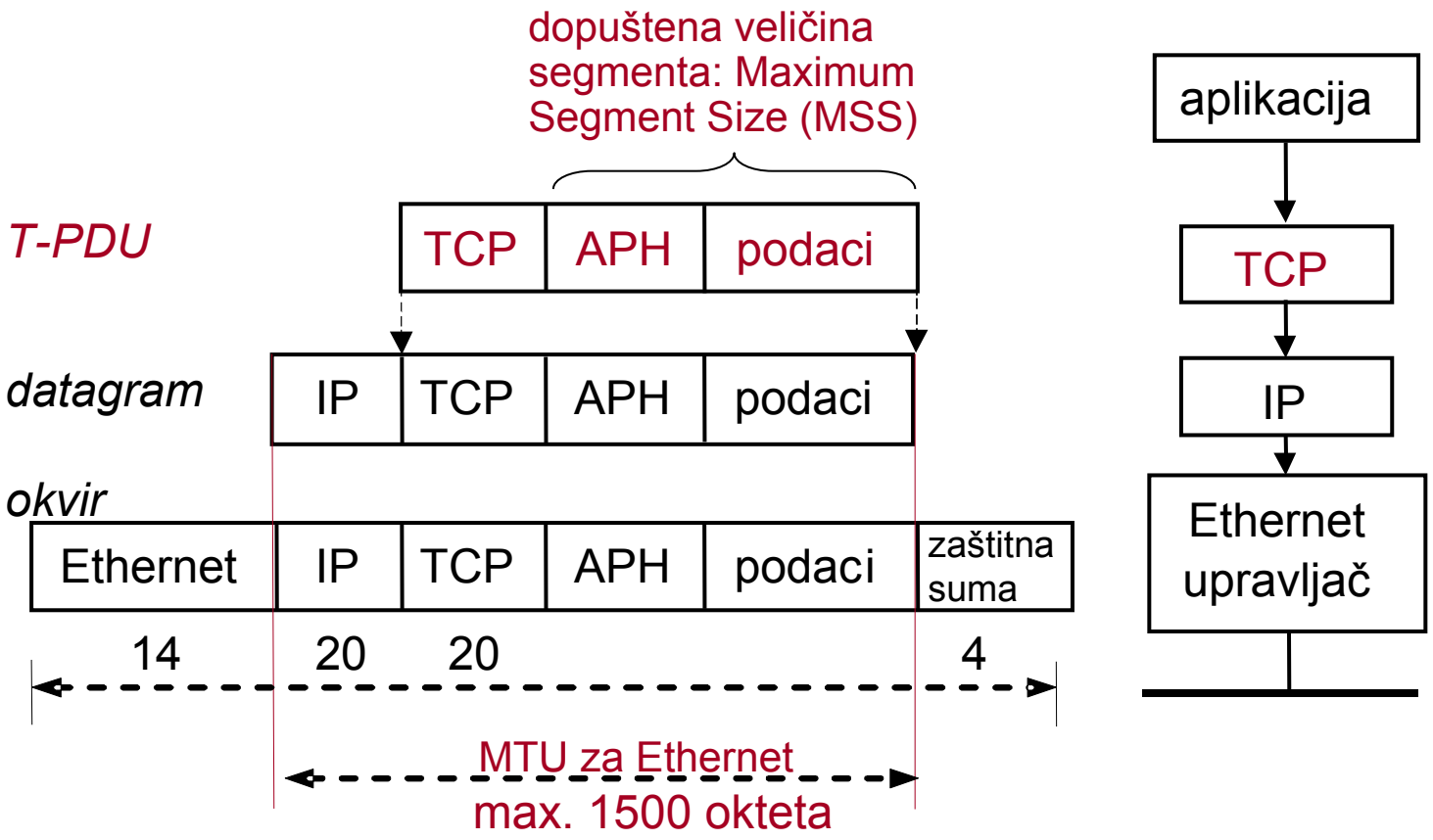
- Transmission Control Protocol
  (nastavak prethodnog predavanja)
  - upravljanje zagušenjem
- User Datagram Protocol
- Primjeri:
  - Kako aplikacija (aplikacijski protokol) koristi protokol TCP?
  - Kako aplikacija (aplikacijski protokol) koristi protokol UDP?

Podsjetimo se...

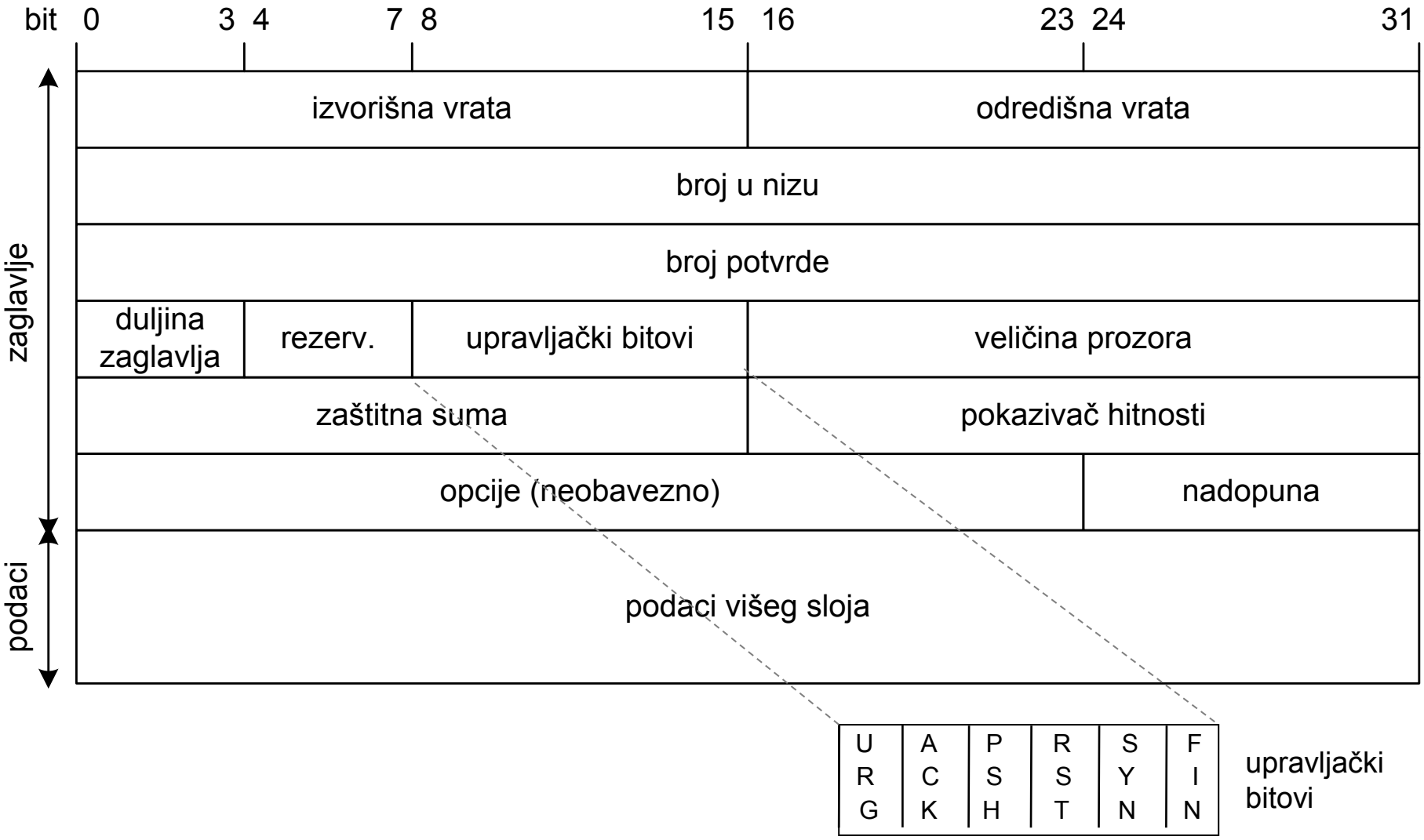
- Transmission Control Protocol (TCP) je spojno-orijentirani, pouzdani internetski protokol transportnog sloja
  - TCP pruža spojnu uslugu transporta struje okteta povrh nespojnog IP-a
  - uspostavlja logičku vezu između procesa na krajnjim računalima
  - osigurava pouzdan transport s kraja na kraj pomoću mehanizama potvrde i retransmisije, uz očuvani redoslijed struje okteta i upravljanje transportnom vezom.

Transportni protokol TCP (podsjetnik)

- dvosmjerni transport kontinuiranog niza podataka, pakiranjem okteta podataka u *segmente,* koje potom predaje protokolu mrežnog sloja

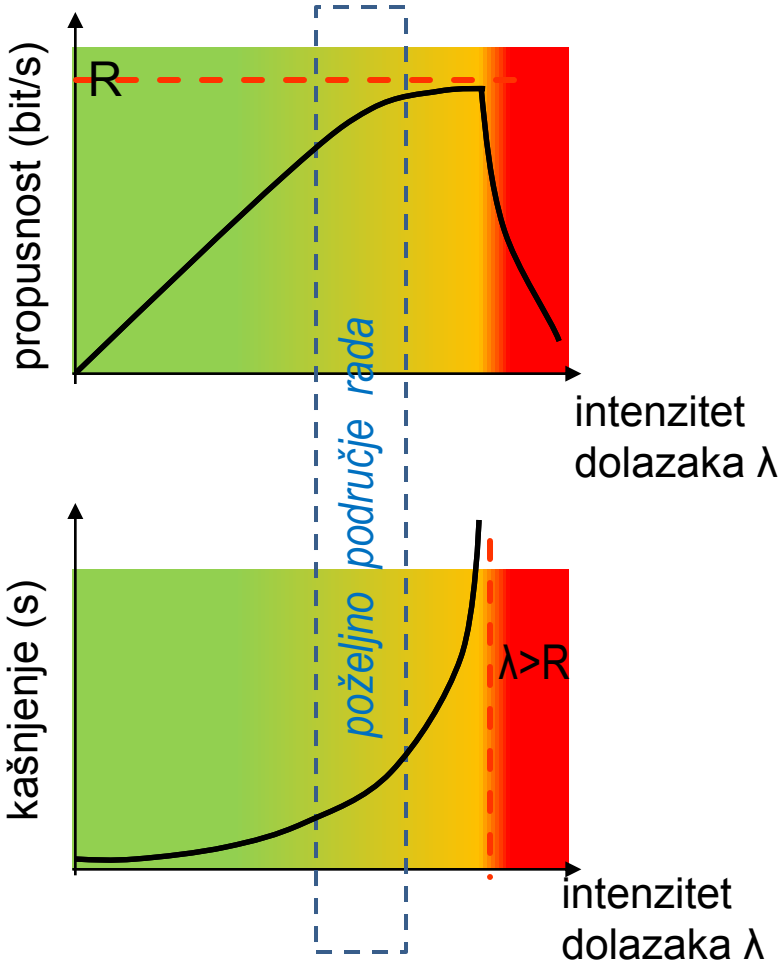


Struktura TCP-segmenta (podsjetnik)



TCP – upravljanje zagušenjem

- uvode se složeni mehanizmi kojima se TCP veza prilagođava (pretpostavljenom) zagušenju u mreži



- faze nastanka zagušenja:
  - normalan rad: λ dovoljno manje od R
  - početak zagušenja: λ se približava R
  - nastupa zagušenje: λ>R
    - drastičan pad performansi!
- pitanja za TCP:
  - kako detektirati zagušenje?

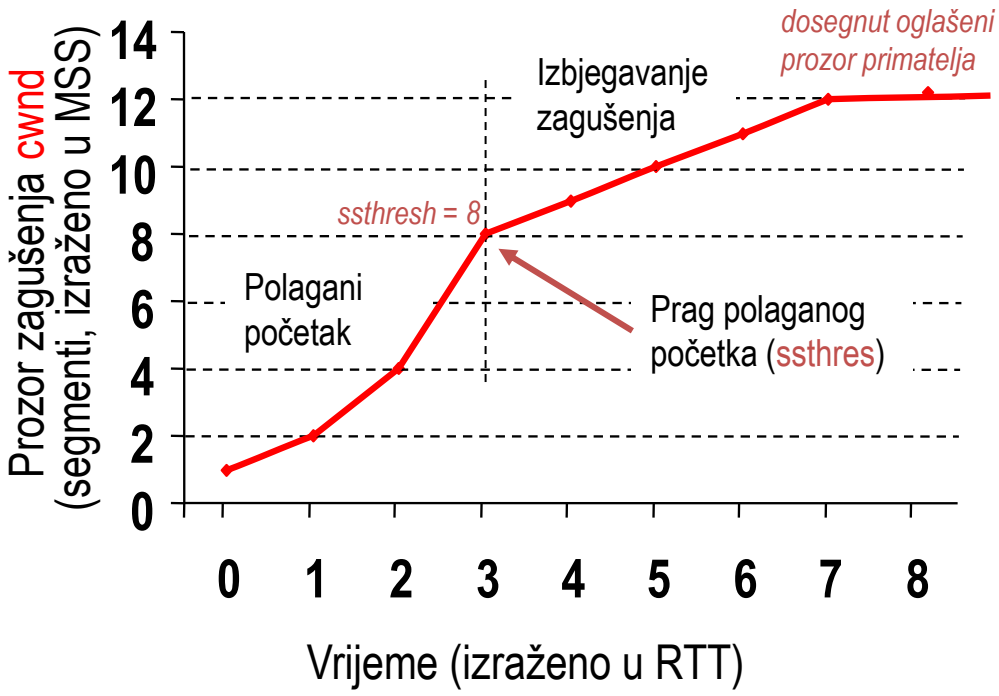
  ideja: učestali gubici, učestali istek RTO
  - što napraviti?

  ideja: naglo smanjiti brzinu slanja, a zatim je postupno povećavati

TCP – upravljanje zagušenjem

- TCP veza se nastoji dinamički prilagoditi raspoloživoj širini pojasa, uz "poštenu" podjelu s drugim vezama
  - problem je što nijedan pošiljatelj ne može unaprijed odrediti "svoj udio" kapaciteta
  - ideja – svatko za sebe postupno povećava brzinu slanja i prati stanje – cilj je da se opće stanje "stabilizira" i zagušenje popusti

  - mehanizam: pošiljatelj uvodi još jedan "prozor": prozor zagušenja
  - efektivni prozor pošiljatelja = min{oglašeni prozor primatelja, prozor zagušenja}
  - rukovanje prozorom zagušenja:
    - eksponencijalni rast u početku (faza "polagani početak")
    - linearni rast kasnije (faza "izbjegavanje zagušenja")
  - postoje i drugi mehanizmi upravljanja zagušenjem (i dalje se razvijaju!)



TCP – upravljanje zagušenjem

- Dva upravljačka mehanizma
  - polagani početak (*Slow Start*)
  - izbjegavanje zagušenja (*Congestion avoidance*)
- Upravljanje zagušenjem odvija se primjenom prozora:
  - prozor zagušenja *cwnd* (*congestion window*)
  - prozor primatelja *rwnd* (*receiver advertised window*)
  - prag polaganog početka *ssthresh* (*slow start threshold size*) služi za osvježavanje vrijednosti *cwnd*
- Prozor pošiljatelja *swnd* (*sender window*)
  - swnd = min{rwnd, cwnd}

Kako TCP ustanovljuje gubitak paketa? (1/2)

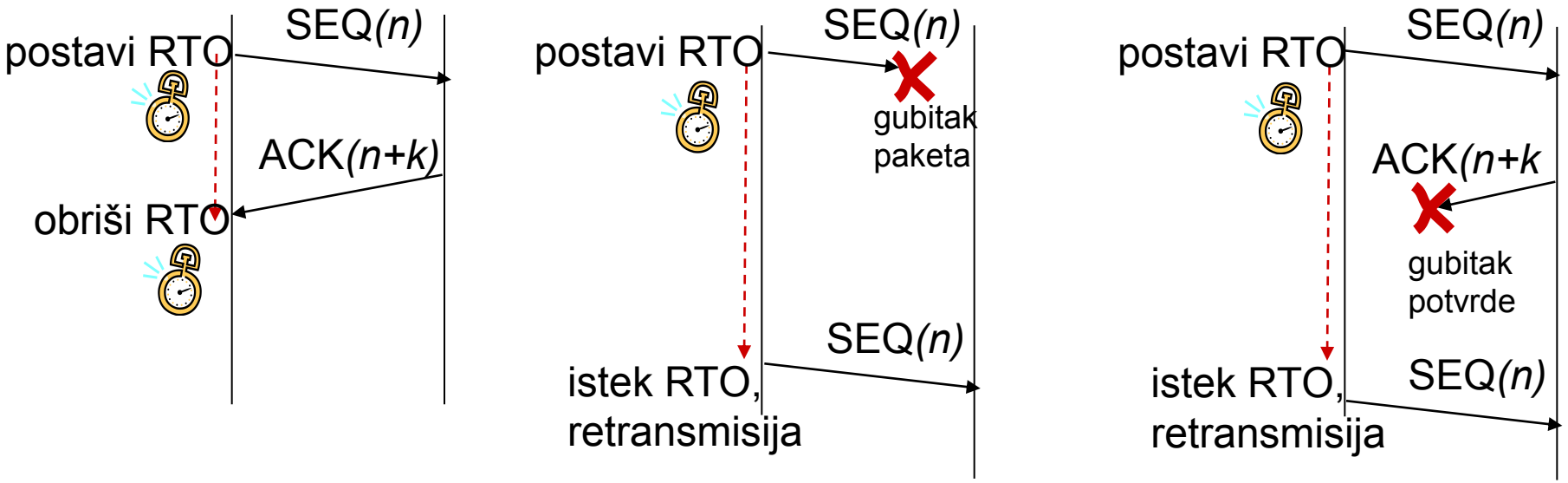
- Gubitak paketa jedan je od indikatora zagušenja – postoji nekoliko načina kako TCP otkriva gubitak paketa, pri čemu nam je već poznat istek RTO
- Vremenska kontrola retransmisije (*Retransmission Time-Out*, RTO)
  - RTO se postavlja u trenutku slanja segmenta (podaci, potvrda) i briše u trenutku primitka potvrde za taj segment
  - istek RTO prije primitka potvrde indikacija je gubitka paketa
  - TCP pošiljatelj smatra da je segment "izgubljen" ako do isteka RTO ne primi potvrdu
    - osim retransmisije, istek RTO utječe i na postavljanje drugih parametara upravljanja zagušenjem (kao što će biti pokazano kasnije)

Kako TCP ustanovljuje gubitak paketa? (2/2)

- Primitak dvostruke potvrde (*Duplicate Acknowledgement*, DUPACK)
  - primitak udvostručene, odnosno "već viđene" potvrde je indikacija da je primatelj primio (neki) segment, ali da to nije očekivani segment koji slijedi u nizu primljenih podataka.
    - potvrda sadrži broj prvog okteta u segmentu koji primatelj očekuje.
    - DUPACK ima isti broj potvrde kao i prethodna potvrda – znači, nastao je prekid niza segmenata
  - podaci iz segmenata koji u međuvremenu stižu privremeno se pohranjuju u memorijski spremnik na primatelju, ali se primljeni segmeti ne mogu potvrditi dok se ne popune/potvrde svi prethodni segmenti!
  - dvostruke potvrde također mogu značiti i promjenu redosljeda segmenata (npr. primitak niza segmenata 0-1-2-4-3-5 na strani primatelja će generirati potvrde 1-2-3-3-5-6).

Vremenska kontrola retransmisije (podsjetnik)

- TCP pošiljatelj postavlja RTO prilikom slanja segmenta
- ako potvrda za segment ne stigne do trenutka isteka RTO, pošiljatelj smatra segment izgubljenim i šalje ga ponovno



(promatramo segment koji počinje na oktetu n i ima duljinu k okteta)

Kako se određuje RTO?

- RTO se izračunava dinamički
  - treba voditi računa o stanju u mreži (koje se stalno mijenja – nema smisla postaviti RTO statički!
    - prekratki RTO – nepotrebna retransmisija u slučaju uspješne isporuke
    - predugački RTP – nepotrebno čekanje na strani primatelja u slučaju gubitka
  - "poželjna" vrijednost RTO: "nešto veća" od prosječnog vremena koje protekne od slanja segmenta do primitka potvrde o uspješnom primitku za taj segment (to vrijeme naziva se RTT, Round Trip Time.)
- za izračun RTO koristi se izraz: **RTO = RTT + 4 * D**

  gdje je D – srednja devijacija RTT-a, a računa se kao:

  D = srednja vrijednost $| x_i - x |$

Praćenje vrijednosti RTT

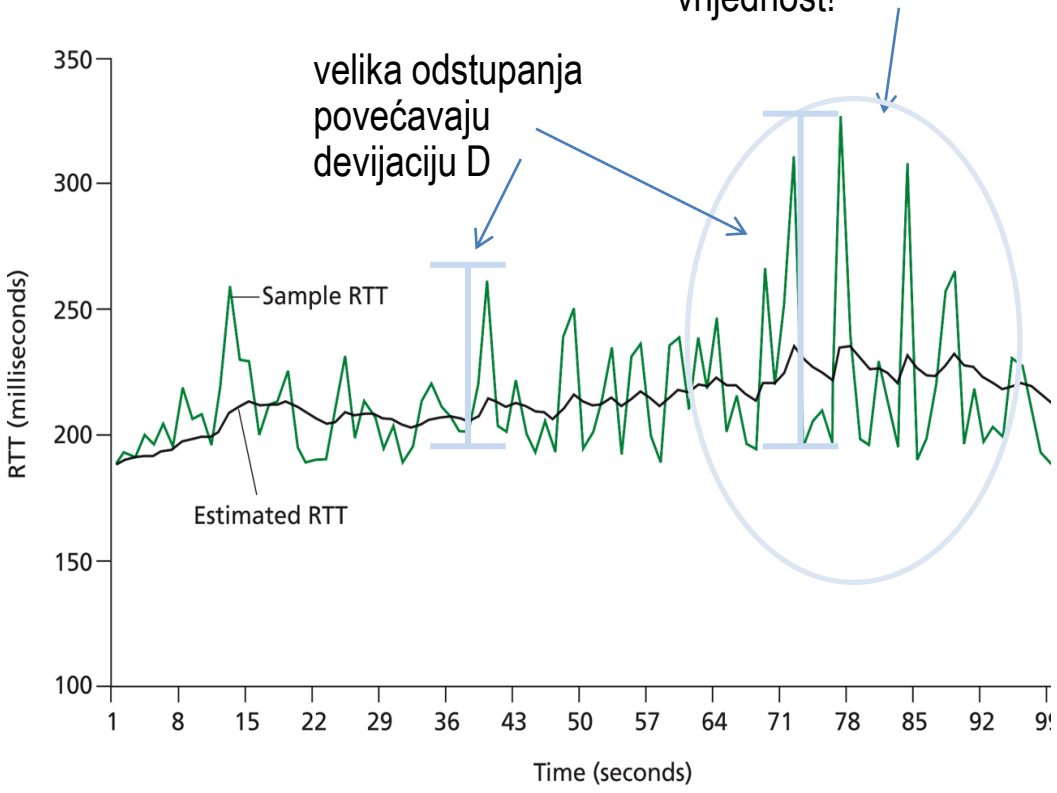
Slika ilustrira odnos vrijednosti RTT izmjerene u nekom trenutku (RTT_izmjereno) i srednje procjenjene vrijedosti RTT(t). RTT(t) se računa kao:

$$RTT(t) = \alpha*RTT(t-1) + (\alpha-1)*RTT\_izmjereno$$

gdje su:

RTT(t) - nova srednja vrijednost
RTT(t-1) - prethodno izračunata srednja vrijednost
RTT_izmjereno - izmjereni iznos RTT zadnjeg potvrđenog segmenta
α - težinski koeficijent (0 <= α < 1).

Uočimo: velike vrijednosti pojedinačnog RTT povećavaju srednju vrijednost!

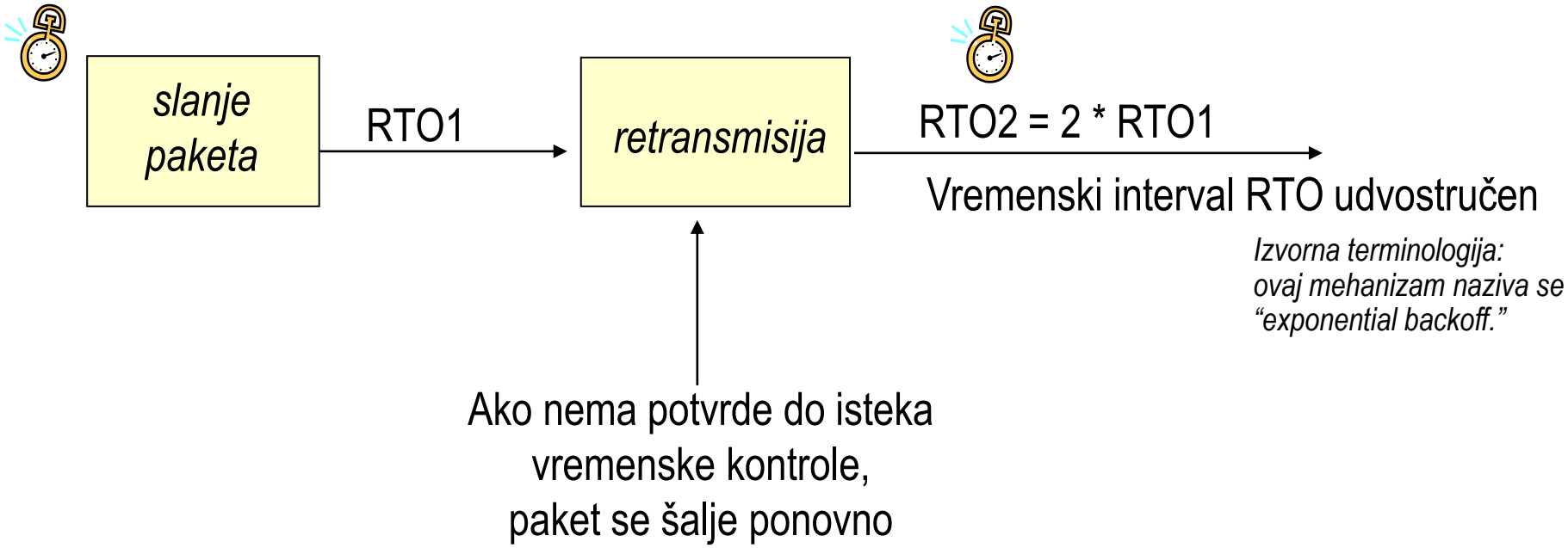


Kako se određuje RTO? (nast.)

- Kako gubitak segmenta utječe na RTT_izmjereno”?
  - ako segment “prođe” tek nakon retransmisije, javlja se problem ispravnog određivanja RTT-a
    - Diskusija:
      - zašto? izvorni i ponovno poslani segment su identični, kao i potvrde!
      - što ako se potvrda odnosi na prvotno poslani segment?
      - što ako se potvrda odnosi na ponovno poslani segment?
  - pogrešno povezivanje potvrde s odgovarajućim segmentom u oba slučaja ima negativne posljedice
- Izračun RTO - ima smisla samo za segmente koji prođu “od prve”!
- Zato se RTO za ponovno poslane segmente računa drugačije, po Karnovom algoritmu.

RTO za ponovljene segmente – Karnov algoritam

- Karnov algoritam:
  - **ne** osvježavaj RTT za segmente koji se šalju ponovno (~~RTO = RTT + 4 * D~~)
  - umjesto toga, udvostručuj RTO prilikom svakog isteka vremenske kontrole (RTO2 = 2 * RTO1), sve dok neki od sljedećih segmenata uspješno ne prođe "od prve" (bude potvrđen nakon što je prvi put poslan) – nakon toga se RTO računa kao inače



Varijable stanja TCP-a

- najveća veličina segmenta (*Maximum Segment Size, MSS*)
  - max. broj podatkovnih okteta koje pošiljatelj smije poslati u jednom TCP segmentu
  - vrijednost MSS ovisi o mrežnoj tehnologiji; odgovara duljini podatkovnog polja okvira na sloju linka, umanjenoj za duljinu IP zaglavlja i TCP zaglavlja
- prozor primatelja (receiver window, *rwnd*)
  - broj okteta koje primatelj objavljuje da može primiti
- prozor zagušenja (congestion window, *cwnd*)
  - inicijalno 1 MSS, povećava se sa svakom potvrdom; najprije eksponencijalno (do vrijednosti praga polaganog početka (*ssthres*) u fazi polaganog početka), a zatim linearno (u fazi izbjegavanja zagušenja)
- prozor pošiljatelja (sender window, *swnd*)
  - najveći broj segmenata koje pošiljatelj smije poslati; definira se kao manja od vrijednosti *rwnd* i *cwnd*
- prag polaganog početka (slow start threshold, *ssthres*)

Upravljanje zagušenjem

- Kad ustanovi gubitak paketa, TCP pošiljatelj pretpostavlja da je to posljedica zagušenja i drastično smanjuje prozor zagušenja, čime usporava brzinu slanja

- ako dođe do isteka vremenske kontrole retransmisije:
  - prag polaganog početka ssthres postavlja se na polovicu veličine prozora prije gubitka paketa, odnosno:

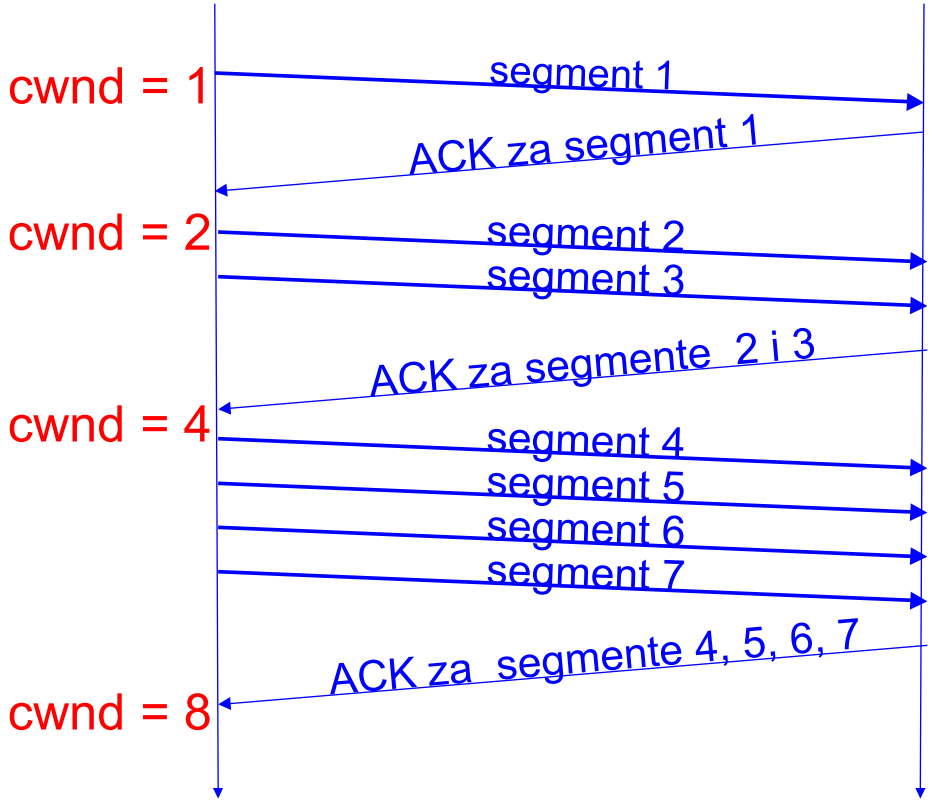
    ssthresh = max { swnd/2 ; 2*MSS }
  - prozor zagušenja cwnd se postavlja na **1 MSS**

- pošiljatelj se vraća u fazu **polaganog početka**

Polagani početak

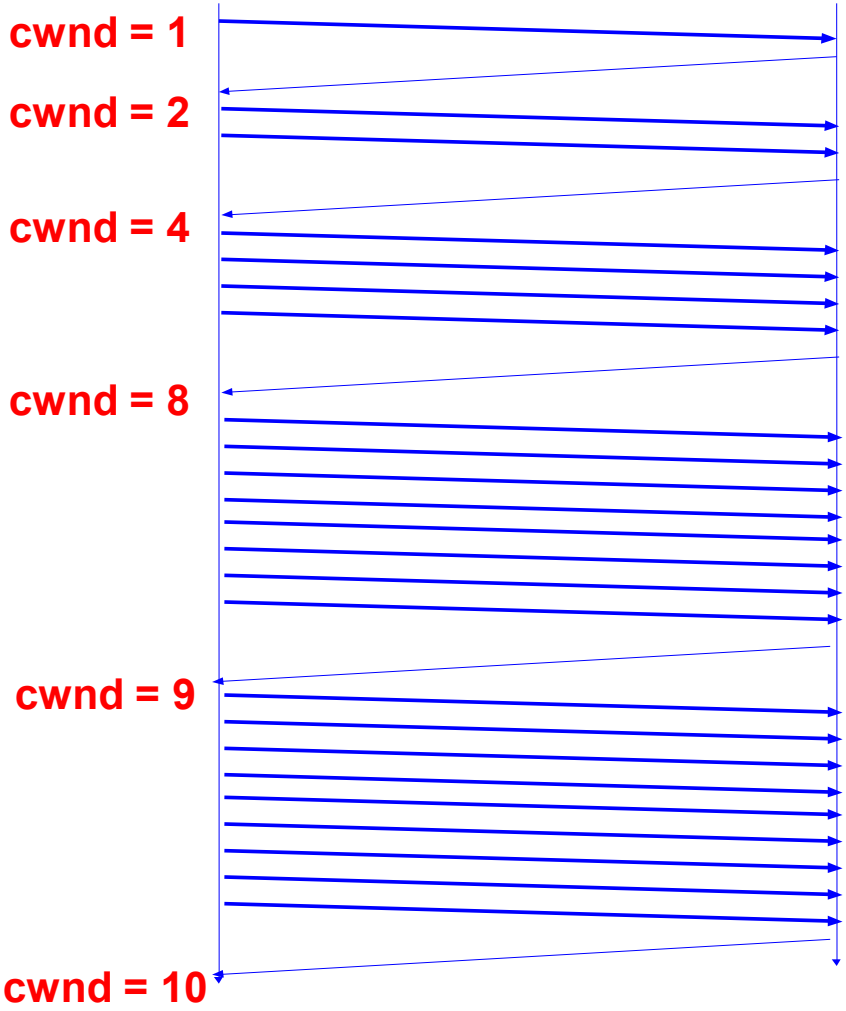
- veličina cwnd se postavlja na cwnd = 1 MSS
- veličina cwnd se povećava za 1 MSS po primitku svake nove potvrde
- cwnd raste **eksponencijalno**
- faza polaganog početka završava kada veličina prozora dosegne prag polaganog početka (ssthresh)

(U ovom primjeru, ssthresh = 8)

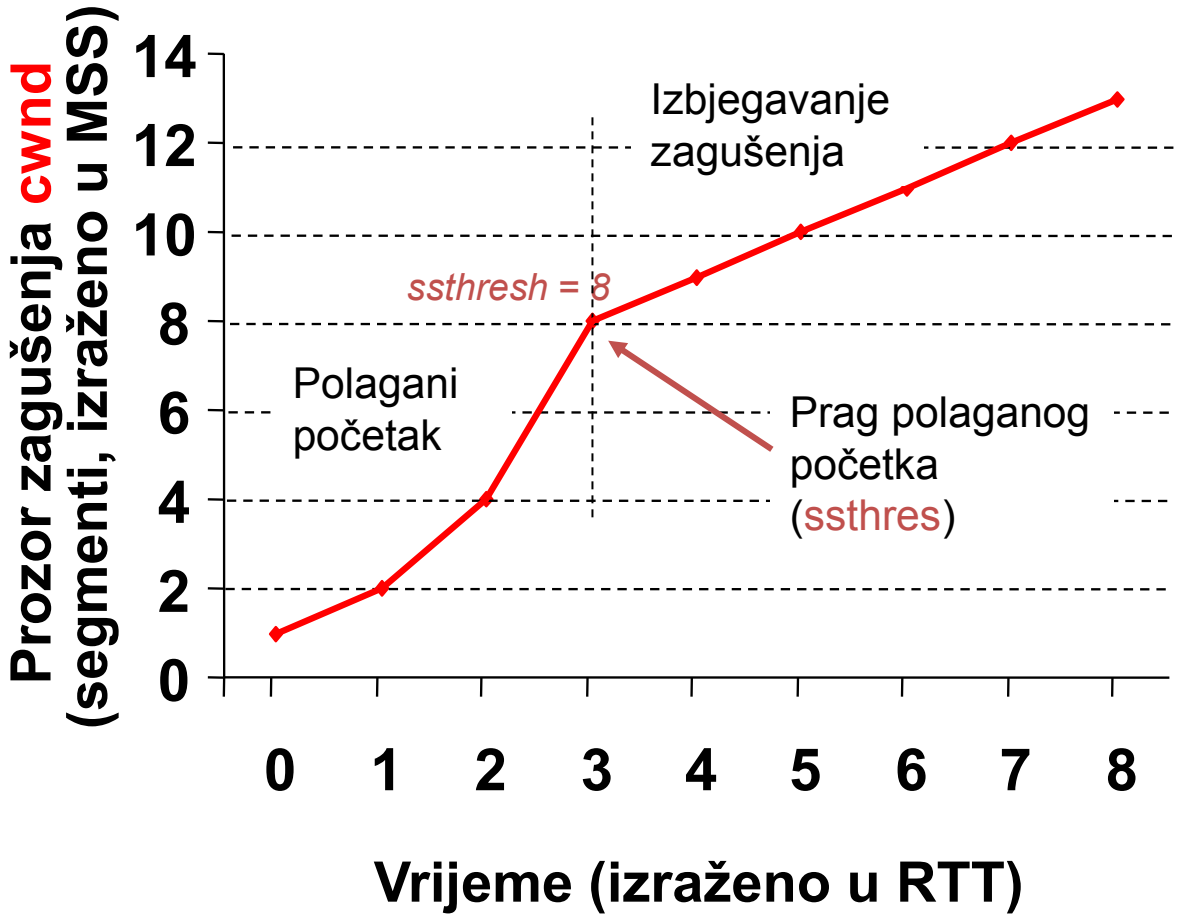


Izbjegavanje zagušenja

- nakon što cwnd dosegne prag polaganog početka (ssthresh) TCP prelazi u fazu **izbjegavanja zagušenja**
- s primitkom svake nove potvrde, pošiljatelj povećava cwnd za 1/cwnd paketa
- za vrijeme izbjegavanja zagušenja cwnd raste **linearno**
  - 1/2 MSS za RTT ako se potvrdi svaki drugi paket
  - 1 MSS za RTT ako se potvrdi svaki paket



Polagani početak i izbjegavanje zagušenja



Primjer

KM-2014_08_dodatak_TCP.jar

- Vizualizacija mehanizama izbjegavanja zagušenja u TCP-u
  - polagani početak
  - izbjegavanje zagušenja
  - izbjegavanje zagušenja i istek RTO
  - brza retransmisija
  - brzi oporavak

Brza retransmisija

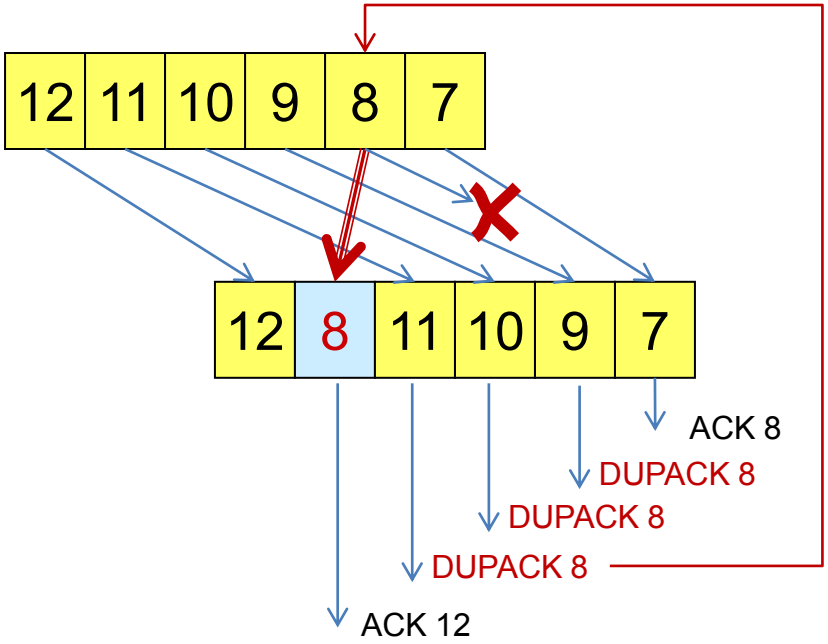
- Nakon što pošiljatelj pošalje onoliko okteta koliko je dopušteno s *cwnd*, blokiran do sljedeće potvrde uspješnog primitka, ili, isteka RTO
  - kako se RTO udvostručava za izgubljeni paket, u slučaju uzastopnog gubitka više paketa, pošiljatelj može dugo čekati na istek RTO, što loše utječe na performanse
  - ponekad su gubici sporadični – u tim slučajevima, bilo bi dobro spriječiti istek RTO, odn. reagirati na indikaciju gubitka segmenta, tj. ako primatelj počne primati ponovljene potvrde od pošiljatelja
- Može li se retransmisija pokrenuti prije isteka RTO?
- Da – to je mehanizam *brze retransmisije*

Upravljanje zagušenjem - brza retransmisija

- brza retransmisija pokreće se kada pošiljatelj primi više dvostrukih potvrda (obično, 3)
- daljnje poboljšanje nakon brze retransmisije je brzi oporavak (*Fast recovery*)
- uočimo razliku u odnosu na istek vremenske kontrole (za takav slučaj slijedi polagani početak)
  - istek vremenske kontrole događa se kada paketi više uopće ne prolaze
  - brza retransmisija događa se kad je neki od paketa izgubljen, ali kasniji paketi prolaze
  - brza transmisija se radi dok vremenska kontrola potvrde još nije istekla

Detekcija gubitka paketa na temelju dvostrukih potvrda i brza retransmisija

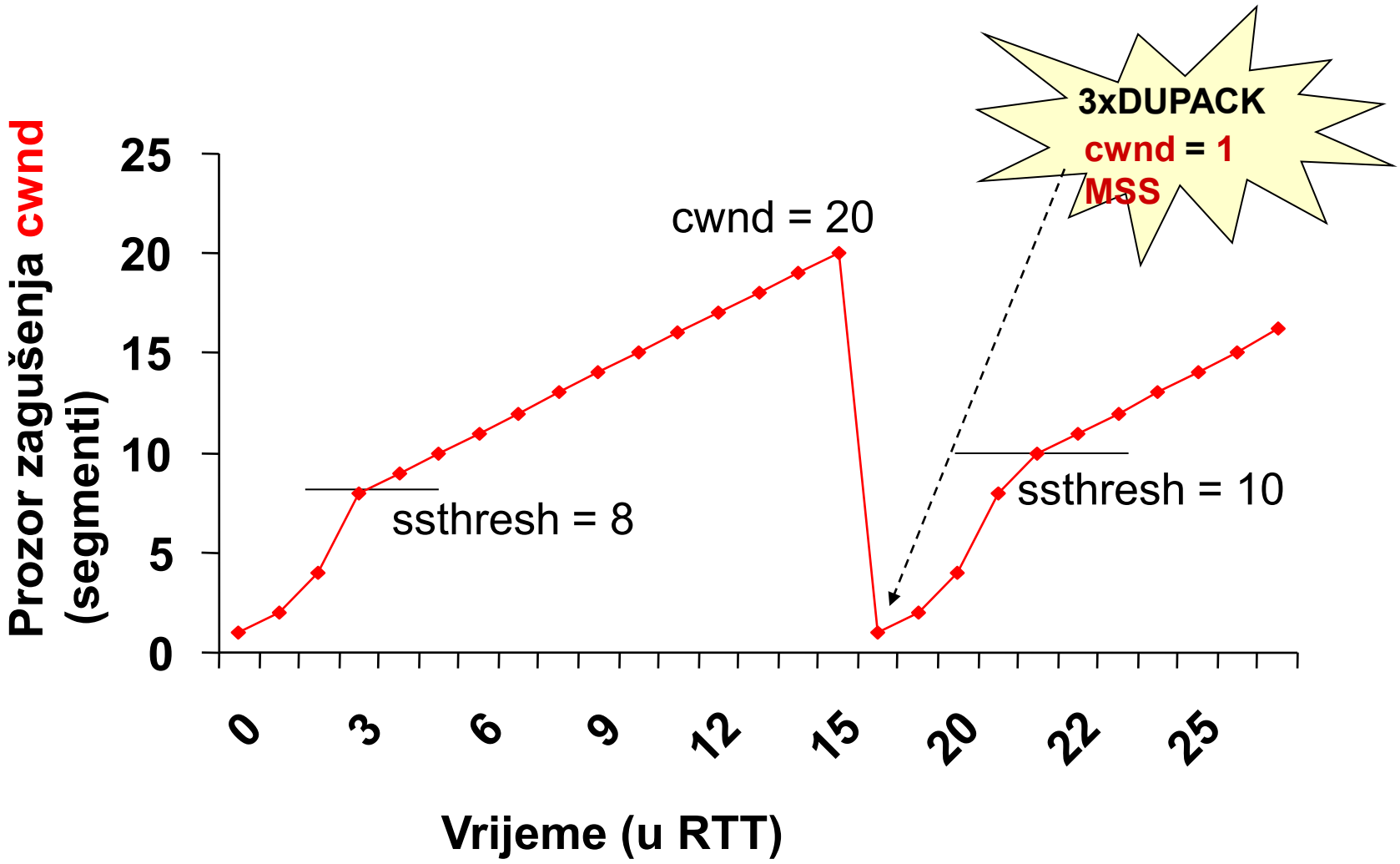
- Dvostruke potvrde (***dup****licate* ***ack****nowledgement*, "DUPACK") se šalju u slučajevima:
  - stvarnog gubitka jednog od segmenata u nizu, ili
  - dostave segmenta izvan redosljeda (iz perspektive primatelja isto što i gubitak)
- TCP pošiljatelj pretpostavlja da je došlo do gubitka segmenta ako primi **tri dvostruke ("već viđene") potvrde zaredom**



3 dvostruke potvrde pojavljuju se kada je paket isporučen najmanje tri paketa "dalje" od svog predviđenog položaja

Napomena: brza retransmisija ima smisla samo ako paketi stižu uz uglavnom očuvani redosljed.

Upravljanje zagušenjem (bez brzog oporavka)



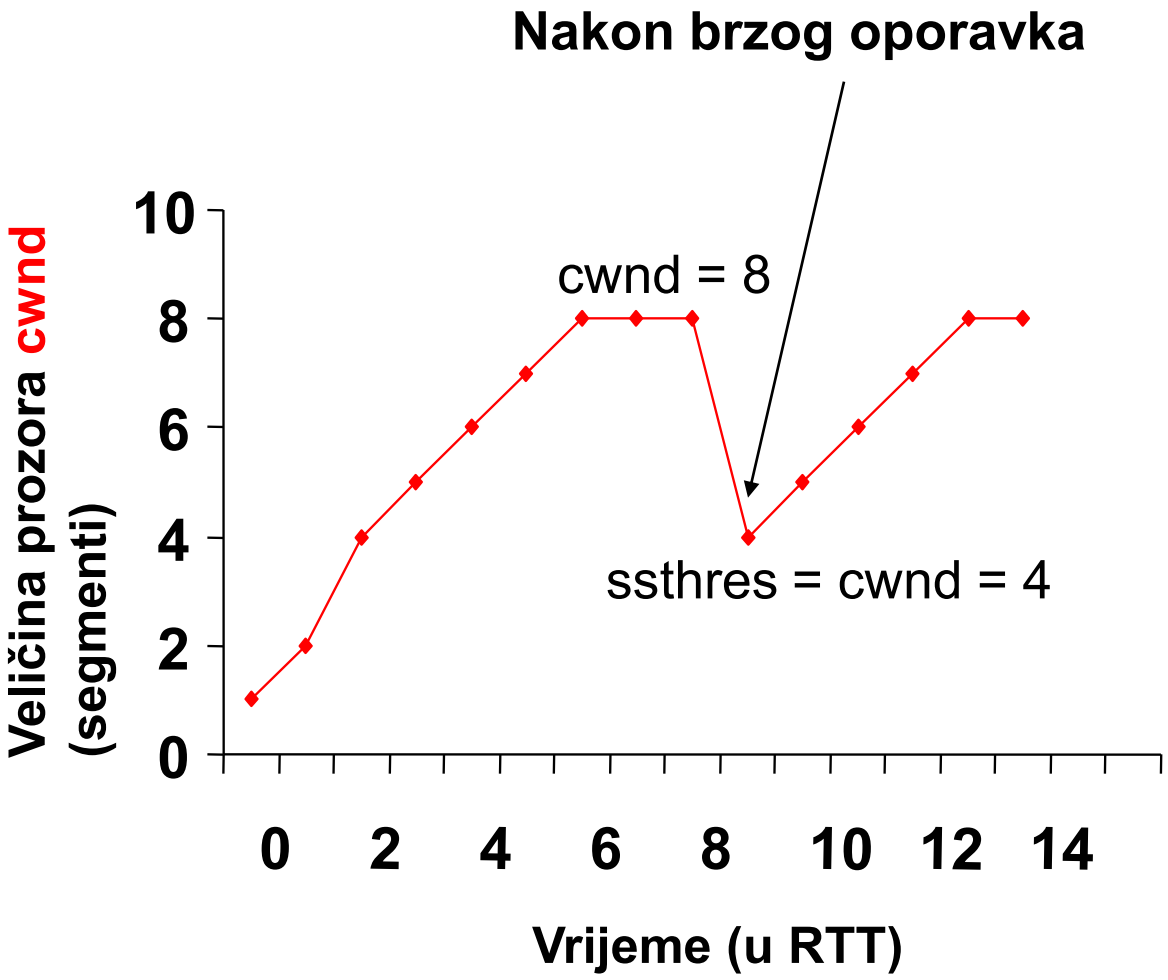
Brzi oporavak

- ssthresh = cwnd/2, a minimalno 2*MSS
- ponovno slanje segmenta koji nedostaje (to je brza retransmisija)
- postavljanje cwnd

  cwnd = ssthresh + broj dvostrukih potvrda (3) *MSS
- kad stigne nova potvrda:

  cwnd = ssthresh
  - ulazi se u fazu izbjegavanja zagušenja (linearni rast)
  - prozor zagušenja je prepolovljen (a ne vraćen na 1 MSS kao prije!)

Upravljanje zagušenjem – brzi oporavak



Nakon brze retransmisije i brzog opravka, veličina prozora se prepolavlja.

Primjer

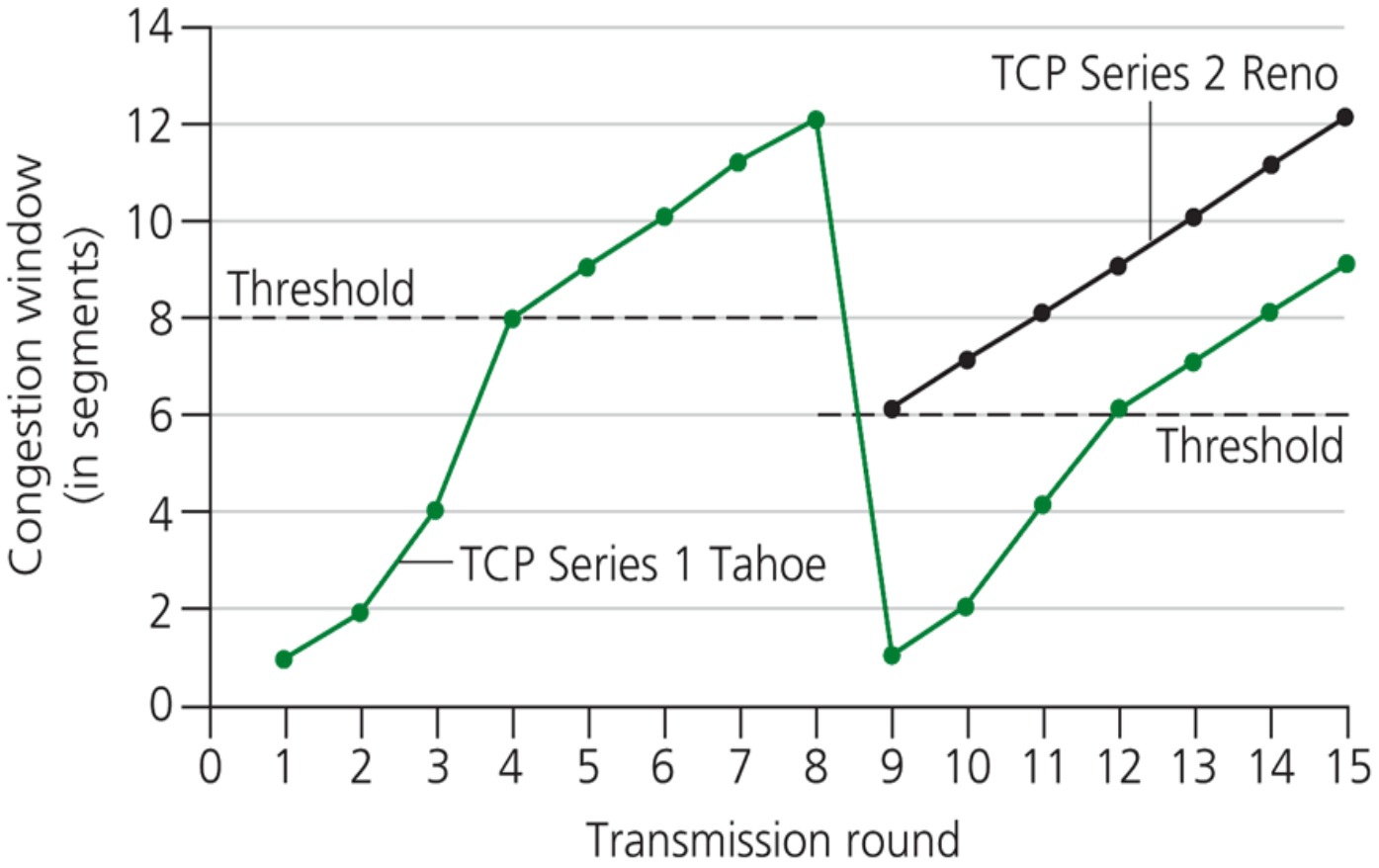
KM-2014_08_dodatak_TCP.jar

- Vizualizacija mehanizama izbjegavanja zagušenja u TCP-u
  - polagani početak
  - izbjegavanje zagušenja
  - izbjegavanje zagušenja i istek RTO
  - brza retransmisija
  - brzi oporavak

Razne izvedbe TCP-a

- TCP Tahoe
  - polagani početak (*slow start*)
  - izbjegavanje zagušenja (*congestion avoidance*)
  - brza retransmisija (*fast retransmit*)
- TCP Reno (temelj za većinu današnjih izvedbi TCP-a)
  - TCP Tahoe + brzi oporavak (*fast recovery*)
- TCP New-Reno
  - TCP Reno + modificirani brzi oporavak (RFC 6582); ostaje u brzom oporavku dok ne povrati sve izgubljene pakete u prozoru
- TCP-Selective Acknowledgements (TCP-SACK)
  - TCP Reno + selektivne potvrde (RFC 2883)
  - TCP-SACK omogućuje primatelju da javi pošiljatelju detaljniju informaciju o svim segmentima koji su uspješno primljeni, na temelju čega pošiljatelj može ponovno poslati samo one segmente koji su bili izgubljeni

Vizualna usporedba raznih izvedbi TCP-a



IZVOR: J. Kurose, K. Ross, "Computer Networking: A Top Down Approach Featuring the Internet", Pearson Addison-Wesley 2005.

Ograničenja protokola TCP

- Što TCP ne radi?
  - nema mehanizme za sigurnost i privatnost podataka
    - postoje razna rješenja na raznim slojevima
  - ne vodi računa o granicama poruke
    - isporučuje niz okteta, neovisno o tome kako aplikacija pošiljatelja grupira podatke (ne vidi granice poruke)
  - ne garantira isporuku višem sloju
    - ali se potrudi prije nego konačno odustane

Primjena protokola TCP

- tamo gdje je aplikaciji najvažnija pouzdanost
  - transfer datoteka
  - elektronička pošta
  - Web
  - transakcijske primjene
  - rad na udaljenom računalu

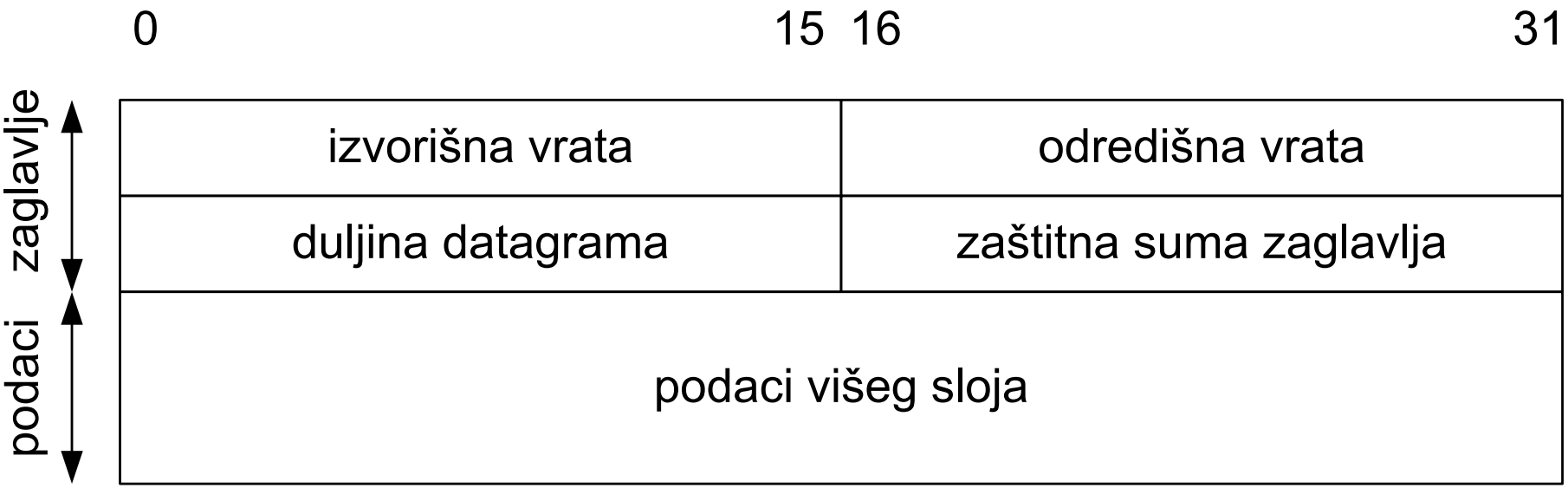
Protokoli transportnog sloja u Internetu

- Transmission Control Protocol
  (nastavak prethodnog predavanja)
  - upravljanje zagušenjem
- **User Datagram Protocol**
- Primjeri:
  - Kako aplikacija (aplikacijski protokol) koristi protokol TCP?
  - Kako aplikacija (aplikacijski protokol) koristi protokol UDP?

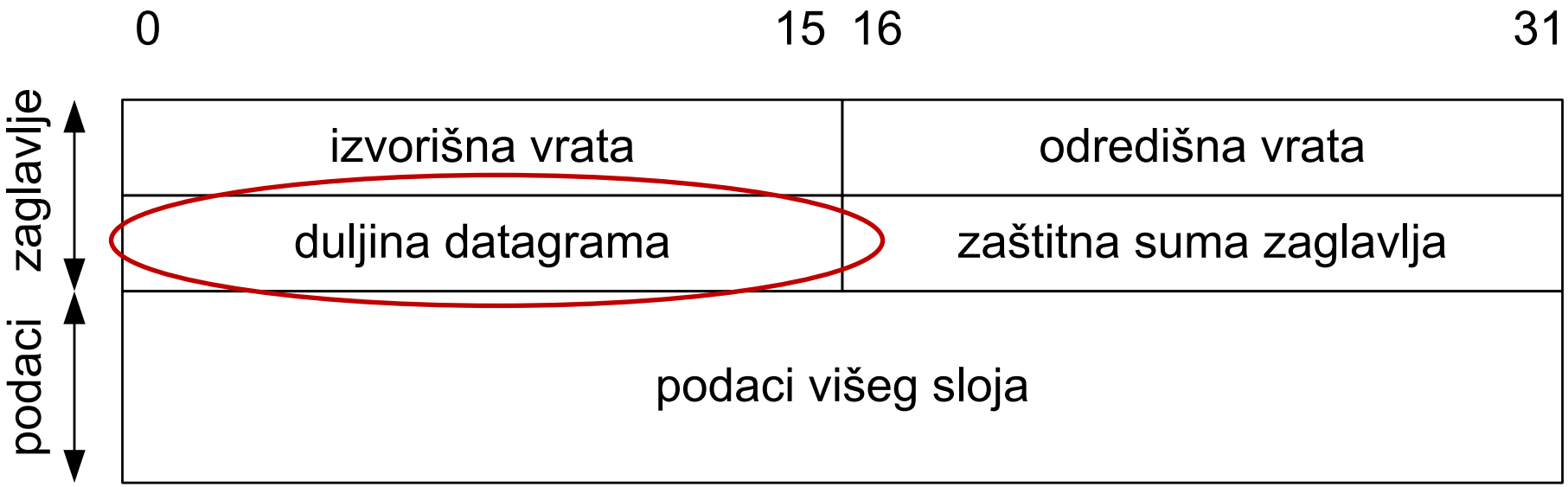
Protokol *User Datagram Protocol* (UDP)

- Jednostavan transportni protokol
- Funkcije:
  - prima podatke od višeg sloja, omata ih u UDP datagram i prosljeđuje mrežnom sloju
  - minimalna funkcionalnost iznad IP-a: multipleksiranje
  - (opcionalno) radi zaštitnu sumu cijelog datagrama
- Ostale značajke:
  - nepouzdan prijenos
  - transfer blokova okteta (datagrami)
  - nema očuvanja redoslijeda
    - datagrami se isporučuju aplikaciji onim redoslijedom kojim su primljeni
  - ne pruža kontrolu toka - ako pošiljatelj prebrzo šalje, datagrami se gube

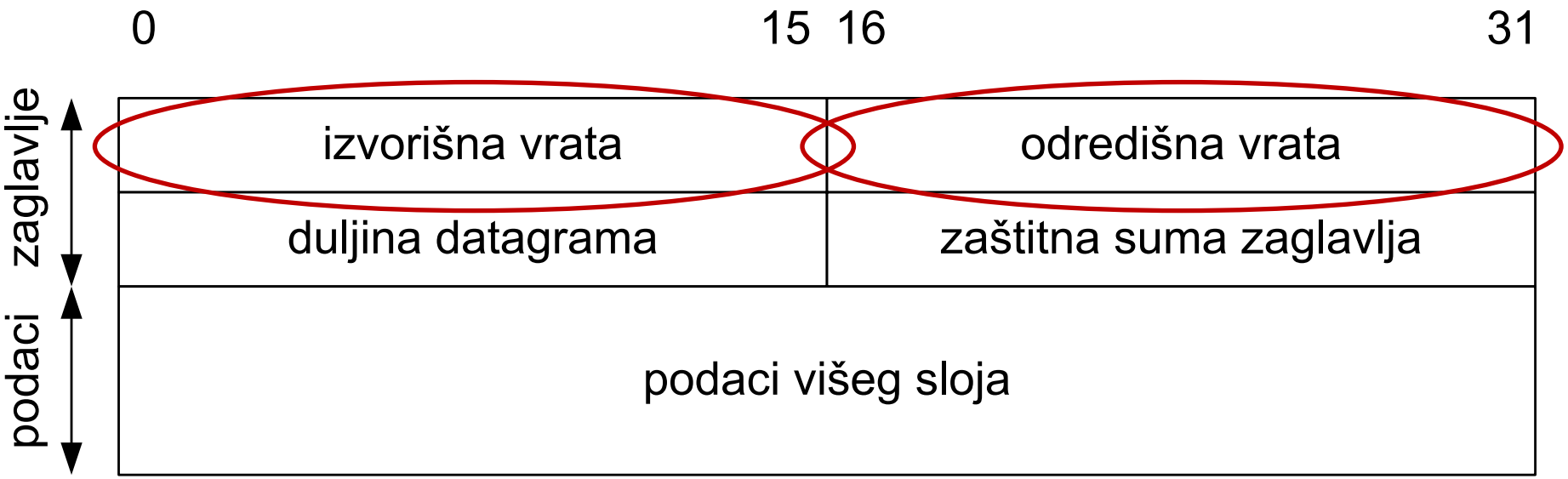
Format UDP datagrama



UDP zaglavlje – polja vezana uz ulogu omatanja

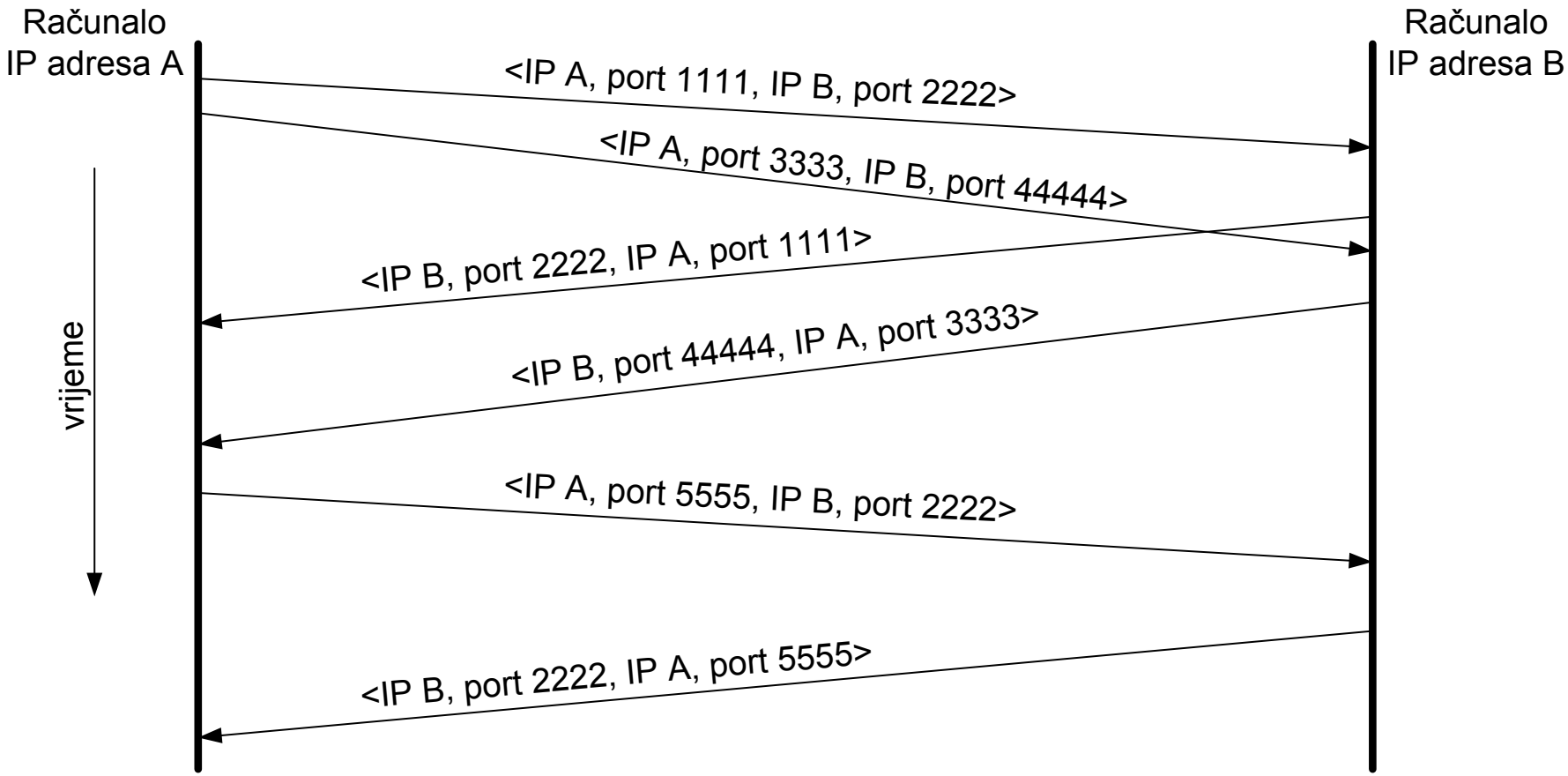


UDP zaglavlje – polja vezana uz ulogu multipleksiranja



Napomena: brojevi UDP-vrata neovisni od brojeva TCP-vrata!

UDP – primjer multipleksiranja tokova



Oznake: <IP adresa izvora, vrata na izvoru, IP adresa odredišta, vrata na odredištu>

Ograničenja protokola UDP

- Što UDP ne radi?
  - ne uspostavlja vezu prije slanja podataka
  - ne potvrđuje primitak podataka
  - ne garantira isporuku podataka
  - ne otkriva gubitak paketa, niti radi retransmisiju izgubljenih paketa
  - ne garantira očuvanje redoslijeda
  - ne pruža kontrolu toka niti kontrolu zagušenja

OK, gdje se UDP koristi?

Primjene protokola UDP

- tamo gdje je aplikaciji dostava podataka *na vrijeme* važnija od dostave *svih* poslanih podataka (prije ili kasnije)
  - višemedijske aplikacije u stvarnom vremenu
  - na primjer: internetska telefonija, višekorisničke igre
- pogodan za kratku komunikaciju (tamo gdje je *overhead* uspostave veze neprihvatljiv)
  - brzi zahtjev/odgovor
  - na primjer: upiti za razlučivanje adrese (DNS), dinamička dodjela adrese (DHCP)
- višeodredišne primjene i difuzija
  - način komunikacije 1:n ili n:m

Osvrt na izbor transportnog sloja u aplikacijama

- Kako transportni sloj u Internet mreži utječe na maksimalnu brzinu kojom aplikacije mogu međusobno razmijenjivati podatke?

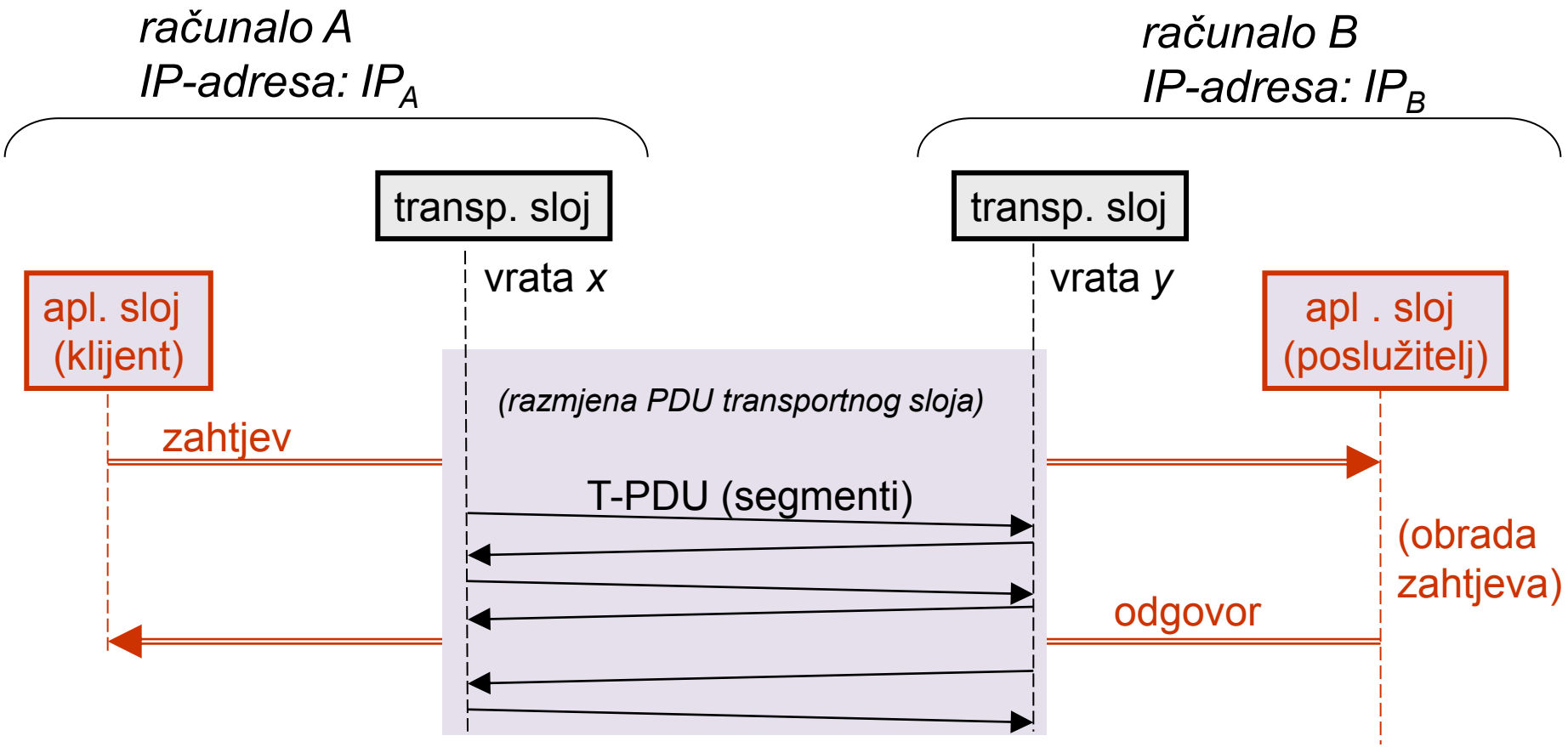
- Ako TCP pošiljatelj želi poslati veliku količinu podataka primatelju, o čemu sve ovisi brzina kojom će podaci biti preneseni?
  - ponavljanje slanja
  - kontrola toka
  - kontrola zagušenja

- A kod UDP-a?
  - aplikacije moraju same voditi računa o ispuštenim podacima i same moraju prilagođavati brzinu uvjetima u mreži

Protokoli transportnog sloja u Internetu

- Transmission Control Protocol
  (nastavak prethodnog predavanja)
  - upravljanje zagušenjem
- User Datagram Protocol
- Primjeri:
  - Kako aplikacija (aplikacijski protokol) koristi protokol TCP?
  - Kako aplikacija (aplikacijski protokol) koristi protokol UDP?

Odnos procesa aplikacijskog i transportnog sloja



Razlike u svojstvima transportnih protokola TCP i UDP

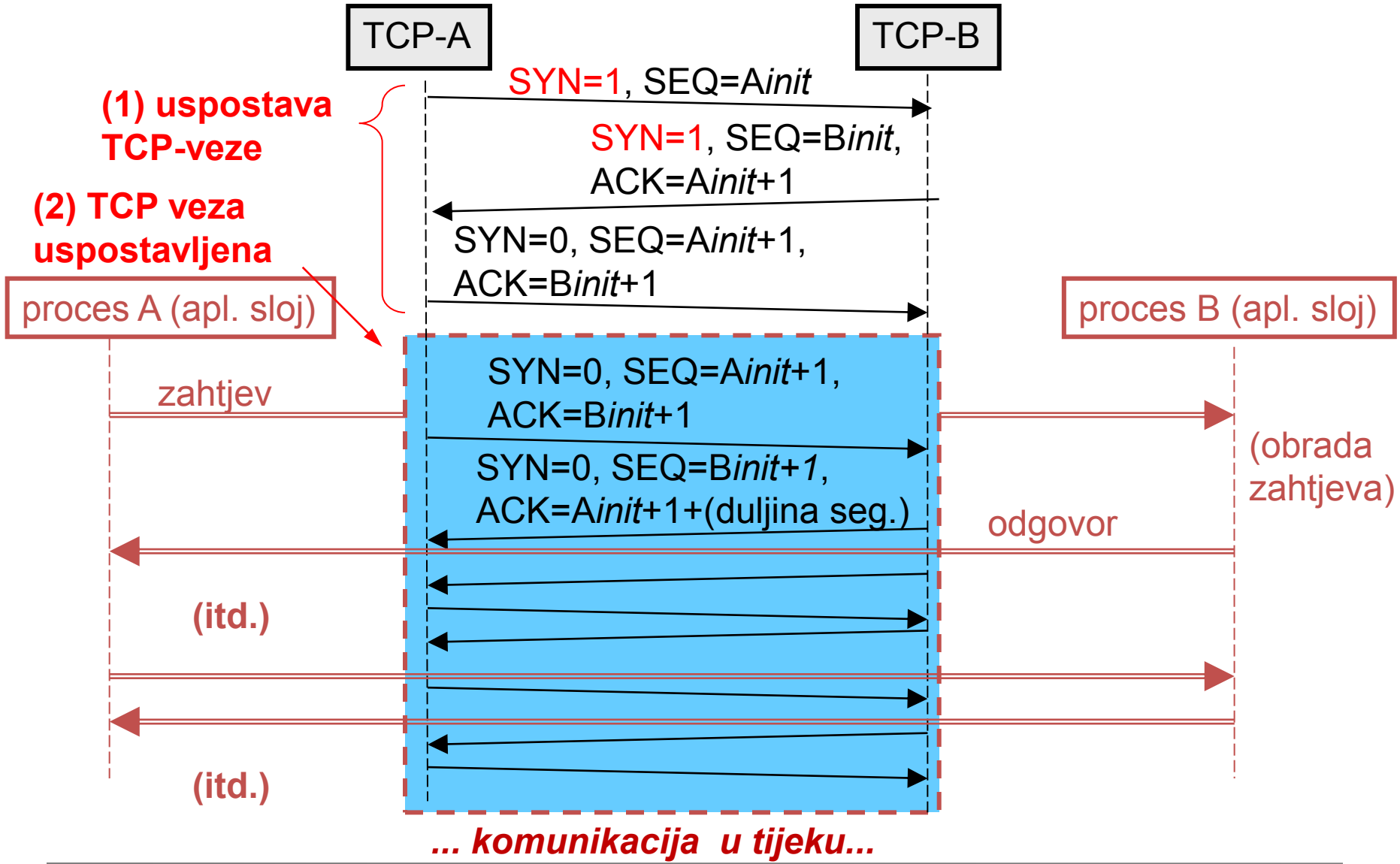
TCP

- spojno-orijentiran
  - pruža spojnu uslugu transporta struje okteta povrh nespojnog mrežnog protokola IP
  - uspostavlja logičku vezu između procesa na krajnjim računalima (definirana parom vrata)
- pouzdan
  - osigurava pouzdan transport s kraja na kraj pomoću mehanizama potvrde i retransmisije, uz očuvani redoslijed struje okteta, uz upravljanje transportnom vezom
  - kontrola toka i kontrola zagušenja

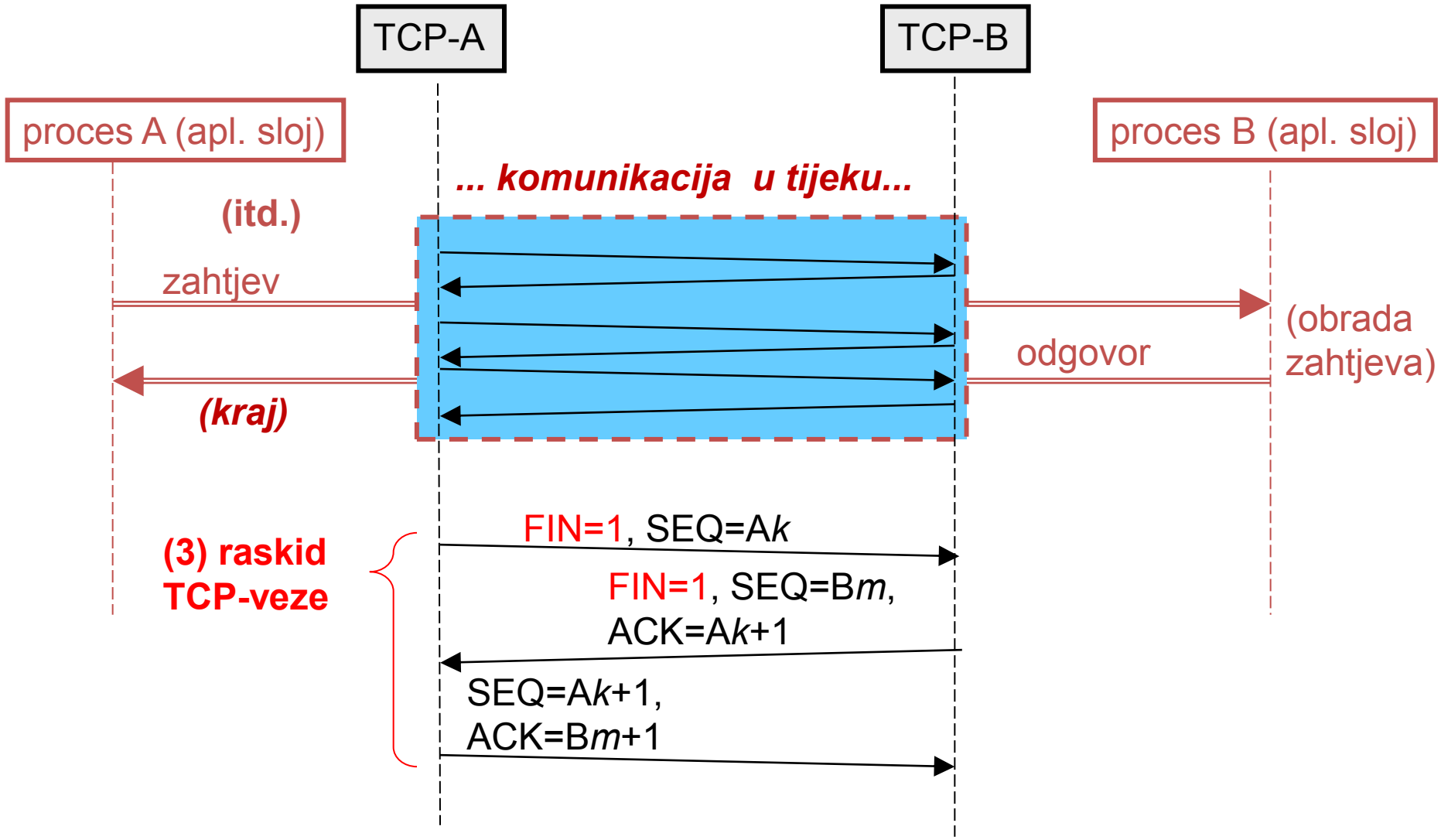
UDP

- nespojno orijentiran
  - pruža nespojnu uslugu transporta blokova okteta povrh IP-a
  - ne uspostavlja vezu prije slanja podataka procesi se povezuju preko broja vrata
- nepouzdan
  - ne potvrđuje primitak podataka
  - ne garantira isporuku podataka
  - ne otkriva gubitak paketa, niti radi retransmisiju izgubljenih paketa
  - ne garantira očuvanje redoslijeda
  - ne pruža kontrolu toka niti kontrolu zagušenja

Primjer: Usluga koja koristi TCP (1/2)



Primjer: Usluga koja koristi TCP (2/2)



Primjer: Usluga koja koristi UDP