

Transportni sloj I: UDP i pouzdani prijenos

Predavanje 4 • kolegij Računalne mreže

doc. dr. sc. Sandi Baressi Šegota

Fakultet informatike Sveučilišta Jurja Dobrile u Puli

Sadržaj predavanja

- 1 Transportne usluge
- 2 UDP
- 3 Pouzdani prijenos podataka (rdt)
- 4 Protokoli s cjevovodom
- 5 Sažetak

Transportne usluge

Ishodi učenja

Nakon ovog predavanja moći ćete:

- objasniti ulogu transportnog sloja i **(de)multipleksiranje**;
- opisati **UDP** i izračun kontrolnog zbroja;
- izgraditi **pouzdan prijenos** nad nepouzdanim kanalom (rdt);
- objasniti **Go-Back-N** i **Selective Repeat**;
- usporediti ta dva protokola s cjevovodom.

Uloga transportnog sloja

- pruža **logičku komunikaciju između procesa** na različitim hostovima
- radi **samo na krajnjim uređajima** (ne u ruterima)
- pošiljalatelj: poruke → **segmenti**; primatelj: segmenti → poruke

Transport vs mrežni sloj

Analogija: dopisivanje dvaju kućanstava

Mrežni sloj = poštanska služba (host-do-hosta).

Transportni sloj = osoba koja u kući skuplja/dijeli pisma članovima (proces-do-procesa).

Transportni sloj koristi i **proširuje** uslugu mrežnog sloja.

Multipleksiranje i demultipleksiranje

- **mux** (pošiljatelj): skupi podatke iz socketa, dodaj zaglavlje
- **demux** (primatelj): predaj segment pravom socketu
- ključ: brojevi **portova**



Demultipleksiranje: UDP vs TCP

- **UDP (bez veze)** — socket određuje **2-torka**: (odredišna IP, odredišni port)
- **TCP (s vezom)** — socket određuje **4-torka**: (izvor IP, izvor port, odredište IP, odredište port)
- zato TCP može razlikovati više veza na istom portu poslužitelja

Što transport može pružiti?

- **pouzdan prijenos** bez gubitaka i pogrešaka (TCP)
- **redoslijed** isporuke (TCP)
- **kontrolu toka** (ne preplavi primatelja) i **zagušenja** (ne preplavi mrežu)

Čega nema

Ni TCP ni UDP ne jamče **propusnost** ni najveće **kašnjenje**.

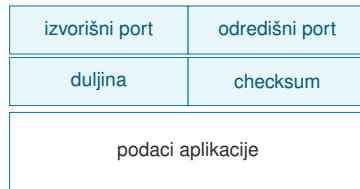
UDP

UDP: obilježja

- **bez veze**, bez rukovanja — mali troškovi
- **best-effort**: moguć gubitak i preslagivanje
- bez kontrole toka i zagušenja $\Rightarrow$ brz start
- koristi ga DNS, DHCP, streaming, igre, QUIC (HTTP/3)

UDP segment

- zaglavlje: samo **8 bajtova**
- portovi, duljina, **checksum**
- ostalo je korisni teret (podaci)



svako polje 16 bita

zaglavlje = 8 bajtova

Kontrolni zbroj (checksum)

- zbroji 16-bitne riječi (uz prijenos)
- **komplement** zbroja = checksum
- primatelj zbroji sve $\rightarrow$ „sve jedinice” ako je OK

riječ 1: 1110011001100110

riječ 2: 1101010101010101

zbroj: 1011101110111011

checksum = komplement zbroja

primatelj zbroji sve + checksum $\Rightarrow$ sve jedinice ako je ispravno

Napomena

Otkriva **neke** pogreške, ali ih ne ispravlja — to je posao viših mehanizama.

Kada koristiti UDP?

- kad je **brzina** važnija od potpune pouzdanosti (streaming, igre)
- kad su poruke **male** i česte (DNS)
- kad aplikacija **sama** dodaje potrebnu pouzdanost (QUIC)

Pouzdani prijenos podataka (rdt)

Problem pouzdanog prijenosa

- cilj: **pouzdan** kanal nad **nepouzdanim** (gubitak, pogreške)
- alati: **potvrde** (ACK), **sekvencijski brojevi**, **timeout**, retransmisija
- gradimo postupno: rdt1.0 → rdt2.0 → rdt2.1/2.2 → rdt3.0

rdt1.0 — savršen kanal

- pretpostavka: kanal je **potpuno pouzdan**
- pošiljalac samo šalje, primatelj samo prima
- nema potreba za potvrđama ni retransmisijom

Poanta

Polazna točka — stvarnost je složenija, pa dodajemo mehanizme.

rdt2.0 — bit-pogreške

- kanal može **oštetiti** bitove (otkriva checksum)
- **ARQ** (Automatic Repeat reQuest): primatelj šalje **ACK** ili **NAK**
- na NAK pošiljalatelj **ponovno** šalje paket

Tri nova elementa

detekcija pogreške (checksum), povratna informacija (ACK/NAK), retransmisija.

rdt2.0 — fatalna mana

- što ako se **ACK ili NAK** ošteti?
- pošiljalatelj ne zna što se dogodilo
- ponavljanje bez oznake $\Rightarrow$ **duplikati**

Rješenje

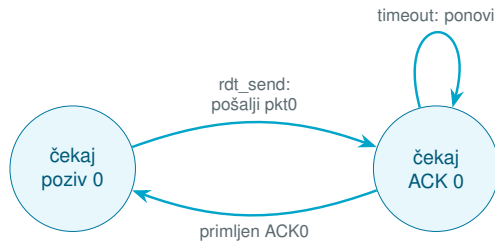
Dodaj **sekvencijski broj** paketu $\Rightarrow$ rdt2.1.

rdt2.1 i rdt2.2

- **rdt2.1** — **sekvencijski brojevi** (0 i 1); primatelj odbacuje duplikate
- **rdt2.2** — **bez NAK-a**: primatelj šalje ACK **s brojem** zadnjeg ispravnog paketa
- dva ACK-a za isti paket = implicitni NAK

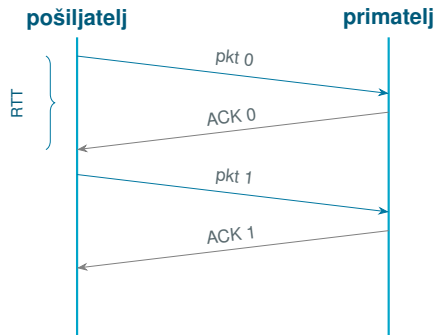
rdt3.0 — gubitak paketa

- kanal može i **izgubiti** pakete
- pošiljalatelj čeka „razumno” i koristi **timeout**
- po isteku: **ponovno** pošalji
- „protokol izmjeničnog bita”



pojednostavljen pošiljalatelj (stani-i-čekaj)

rdt3.0 — rad (stani-i-čekaj)



Pregled inačica rdt

Inačica	Novi problem koji rješava
rdt1.0	savršen kanal (ništa)
rdt2.0	bit-pogreške (ACK/NAK, checksum)
rdt2.1	oštećen ACK/NAK (sekvencijski brojevi)
rdt2.2	ista funkcija bez NAK-a
rdt3.0	gubitak paketa (timeout)

Performanse: iskoristivost

Iskoristivost linka pri „stani-i-čekaј”:

$$U = \frac{L/R}{RTT + L/R}$$

Primjer: $L = 8000$ b, $R = 1$ Gb/s, $RTT = 30$ ms:

$$U = \frac{8 \mu s}{30,008 \text{ ms}} \approx 0,00027$$

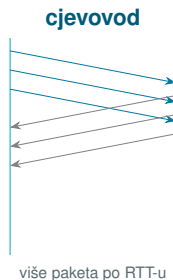
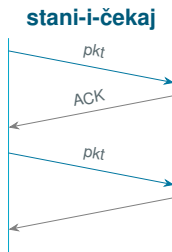
Zaključak

Čak 0,03% iskoristivosti! Rješenje je **cjevovod** — slati više paketa odjednom.

Protokoli s cjevovodom

Ideja cjevovoda (pipelining)

- pošalji **više** paketa bez čekanja svakog ACK-a
- bolja iskoristivost linka
- potreban raspon **sekvencijskih brojeva** i međuspremnici



Dobitak cjevovoda: primjer

Uz prozor od **3** paketa iskoristivost se **utrostruči**:

$$U = \frac{3(L/R)}{RTT + L/R}$$

Općenito, veći prozor $\Rightarrow$ veća iskoristivost, do zasićenja linka.

Poanta

Cjevovod „puni” link dok čekamo potvrde — osnova stvarnog TCP-a.

Dva pristupa

Go-Back-N

Primatelj prihvaća samo **u nizu**; kumulativni ACK; pošiljalatelj vraća se na N.

Selective Repeat

Primatelj **spremi** i pakete izvan reda; **pojedinačni** ACK; ponavlja se samo izgubljeni.

Go-Back-N: prozor

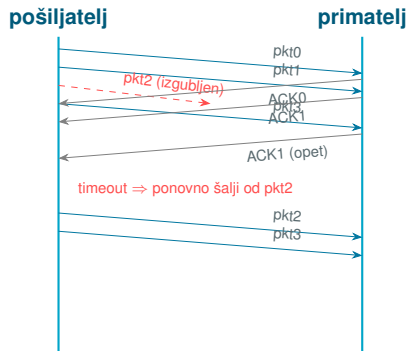


- prozor do **N** nepotvrđenih paketa
- **kumulativni** ACK potvrđuje sve do broja

Go-Back-N: pravila

- **Pošiljatelj**: jedan timer (za najstariji nepotvrđeni); timeout $\Rightarrow$ ponovno šalje **sve** od njega
- **Primatelj**: prihvaća samo **sljedeći očekivani**; sve izvan reda **odbacuje** i ponavlja zadnji ACK

Go-Back-N: rad



Selective Repeat: prozor

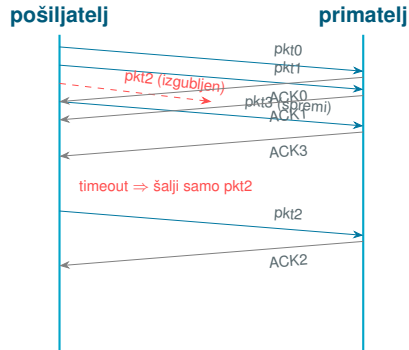


- svaki paket ima **vlastiti** timer i **vlastiti** ACK
- primatelj **međuspremljuje** pakete izvan reda

Selective Repeat: pravila

- **Pošiljatelj**: ponavlja **samo** paket kojemu je istekao timer
- **Primatelj**: potvrđuje **svaki** ispravan paket, sprema izvan reda, predaje redom kad se popuni praznina

Selective Repeat: rad



Go-Back-N vs Selective Repeat

Go-Back-N	Selective Repeat
kumulativni ACK	pojedinačni ACK
primatelj bez međuspremnik	primatelj sprema izvan reda
ponavlja cijeli niz	ponavlja samo izgubljeni
jednostavniji	učinkovitiji pri gubicima

Koji protokol odabrati?

- **GBN** — jednostavan, dobar kad su gubici rijetki
- **SR** — učinkovitiji kad su gubici češći ili je prozor velik
- stvarni **TCP** kombinira ideje obaju (kumulativni ACK + selektivne dopune)

Dilema sekvencijskih brojeva

- raspon brojeva je **konačan** (i ponavlja se)
- kod SR-a prozor mora biti $\leq$ **pola** raspona brojeva
- inače primatelj ne može razlikovati novi paket od ponovljenog

Sažetak

Ključne poruke

- transportni sloj daje **komunikaciju procesa** uz (de)multipleksiranje
- **UDP** je brz, bez veze, s kontrolnim zbrojem
- pouzdan prijenos gradimo **ACK-ovima, brojevima i timeoutima** (rdt)
- „stani-i-čekaj” loše iskorištava link $\Rightarrow$ **cjevovod**
- **GBN** i **SR** dva su načina cjevovodnog pouzdanog prijenosa

Sljedeće predavanje

Transportni sloj II: TCP i kontrola zagušenja.

Pitanja za provjeru

- Čime se razlikuje UDP od TCP demultipleksiranja?
- Zašto rdt2.0 treba sekvencijske brojeve?
- Zašto je iskoristivost „stani-i-čekaj” tako niska?
- Kako se GBN i SR ponašaju kad se izgubi jedan paket?
- Zašto SR prozor mora biti najviše pola raspona brojeva?

Hvala na pažnji!

Pitanja i rasprava

doc. dr. sc. Sandi Baressi Šegota • FIPU