Рачунарске мреже

Александар Картељ

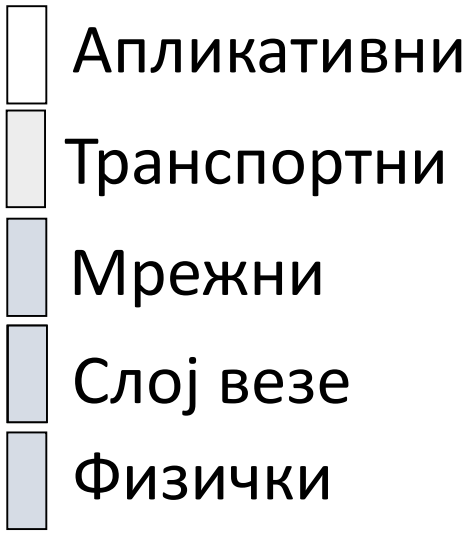
kartelj@matf.bg.ac.rs

Транспортни слој

Преглед

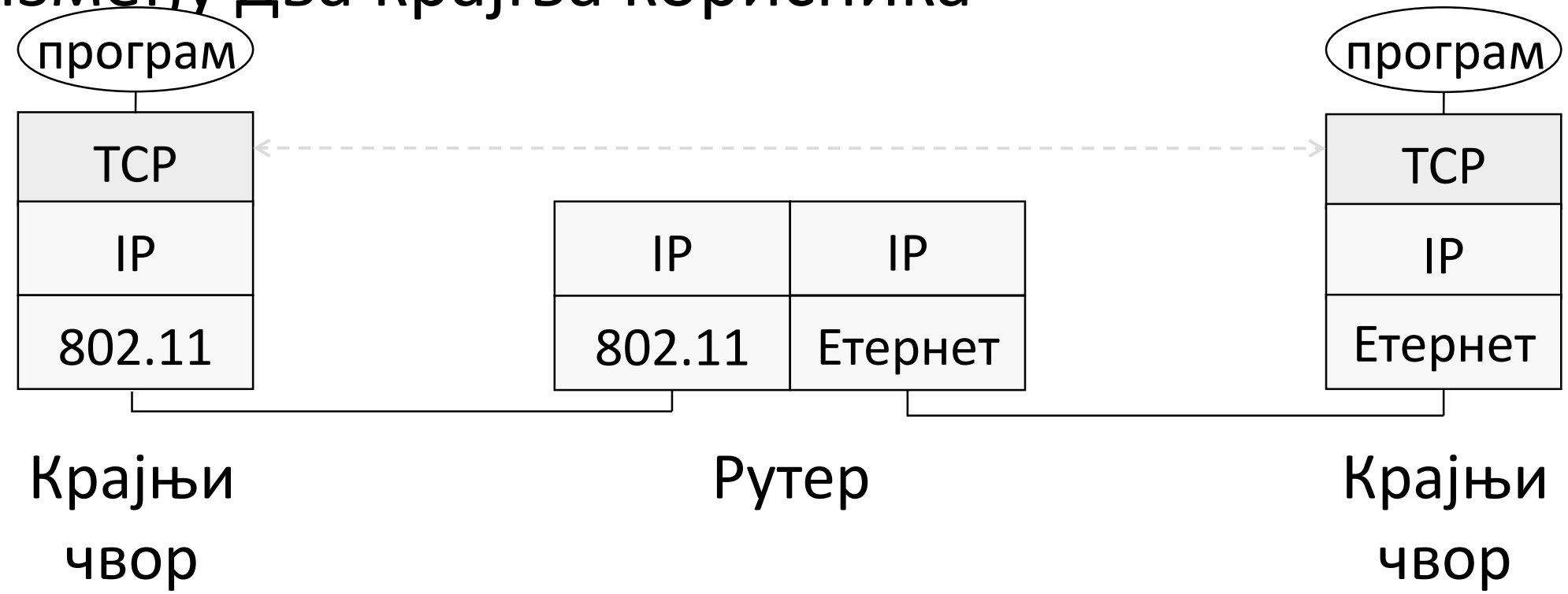
Где смо тренутно?

- Почињемо транспортни слој
  - Надоградња над мрежним слојем
    која омогућава пренос података
    са жељеним степеном поузданости и квалитета



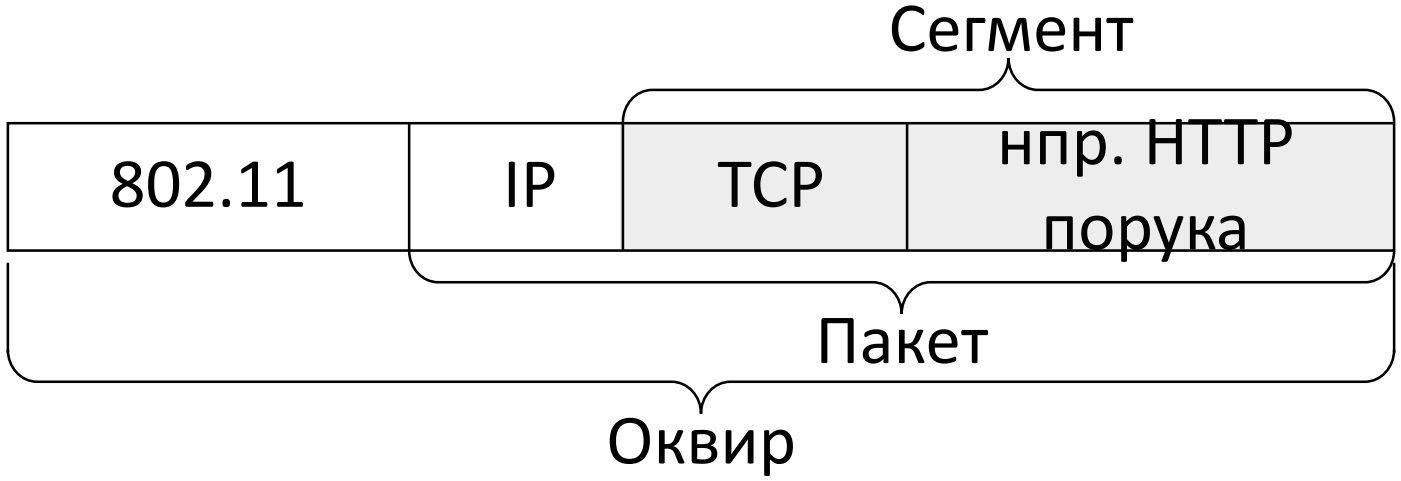
Транспортни слој

- Транспортни слој омогућава комуникацију између два крајња корисника



Транспортни слој (2)

- Јединица информације у транспортном слоју се зове сегмент
- Сегменти се уграђују у пакете, а ови даље у оквире



Типови сервиса на транспортном слоју

- Два основна типа су подржана,
  али се могу реализовати и другачија решења

	Непоуздано	**Поуздано**
Поруке	Датаграми (UDP)	?
Ток података	?	Токови (TCP)

Поређење типова сервиса

- TCP је озбиљно разрађен механизам
  - Не представља надоградњу виртуелног кола!!!
- UDP практично користи датаграм из мрежног слоја

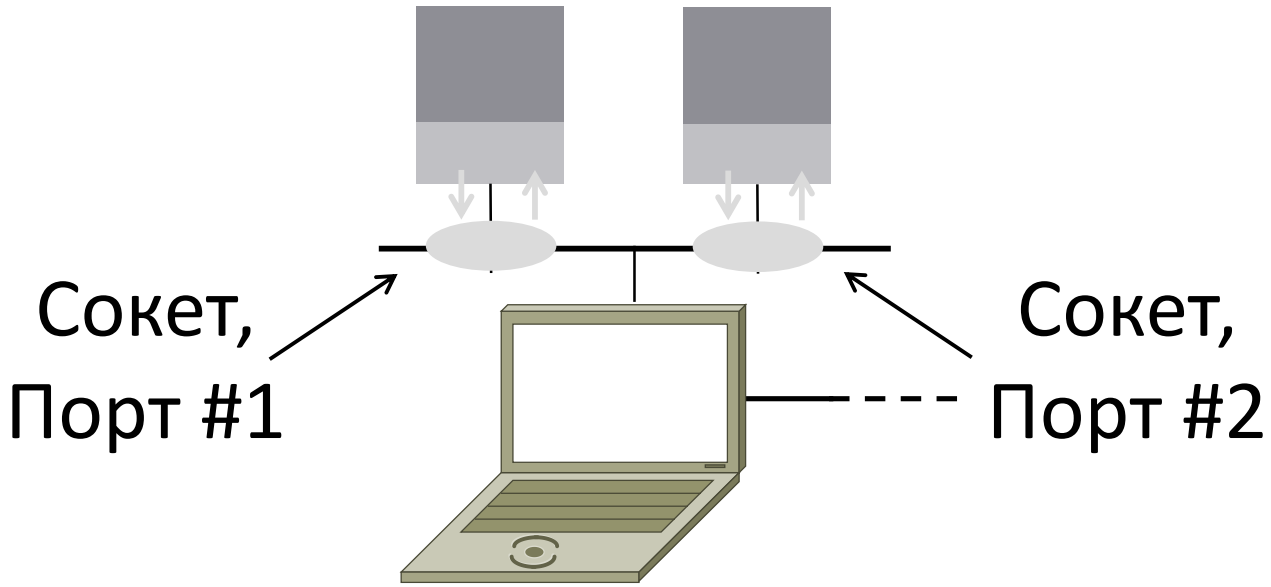
TCP (Токови)	**UDP (Датаграми)**
Остваривање везе	Датаграми
Бајтови се испоручују једном, поуздано и по реду	Поруке се могу изгубити, помешати, дуплирати
Произвољна дужина тока	Ограничена дужина поруке
Контрола тока се прилагођава пошиљаоцу и примаоцу	Шаље се без обзира на стање примаоца
Контрола загушења се прилагођава стању мреже	Шаље се без обзира на стање мреже

Socket API

- Абстракција за употребу мрежних услуга
  - Често се каже и „Мрежни API“, али је заправо у питању употреба транспортних сервиса (а не мрежних)
  - Део свих битнијих оперативних система и програмских језика
- Подржава и токове, као и датаграме

Socket API (2)

- Сокети омогућавају процесима да се повезују на локалну мреже путем различитих портова



Socket API (3)

- Исти API се користи и за токове и за датаграме

Користе се само код Токова

Варијанте са „to/from“ само код датаграма

Операција	Значење
SOCKET	Креира крајњу комуникациону тачку
BIND	Придружује сокет локалној адреси и порту
LISTEN	Најављује спремност за прихватање долазних захтева за везу
ACCEPT	Пасивна успостава везе са тачком која шаље захтев
CONNECT	Активно покушавање за успоставом везе
SEND(TO)	Слање података преко сокета
RECEIVE(FROM)	Примање података преко сокета
CLOSE	Гашење сокета

Портови

- Процеси се идентификују уређеном тројком (IP адреса, протокол, порт)
  - Портови су 16-битни позитивни цели бројеви
- Сервери се обично везују за „опште-познате“ портове
  - Портови са вредностима <1024
- Клијенти обично користе насумичне портове
  - Бира их OS, користе се привремено
  - Не морају бити опште-познати, јер их нико не „циља“

Неки опште-познати портови

Порт	Протокол	Намена
20, 21	FTP	Пренос датотека
22	SSH	Удаљени приступ
25	SMTP	Слање и примање електронске поште
80	HTTP	Приступ WWW
110	POP-3	Примање електронске поште (клијенти)
143	IMAP	Примање електронске поште (клијенти)
443	HTTPS	Сигурни HTTP
543	RTSP	Пренос токова у реалном времену
631	IPP	Дељење штампача

Теме

- Типови сервиса
  - Socket API и портови
  - Датаграми и токови
- UDP
- Повезивање и раскидање везе (TCP)
- Клизни прозори (TCP) – <u>није грешка!</u>
- Контрола тока (TCP)
- Паузе за ретрансмисију (TCP)
- Контрола загушења (TCP)

Урађено

Преостало

Следећи пут

Транспортни слој

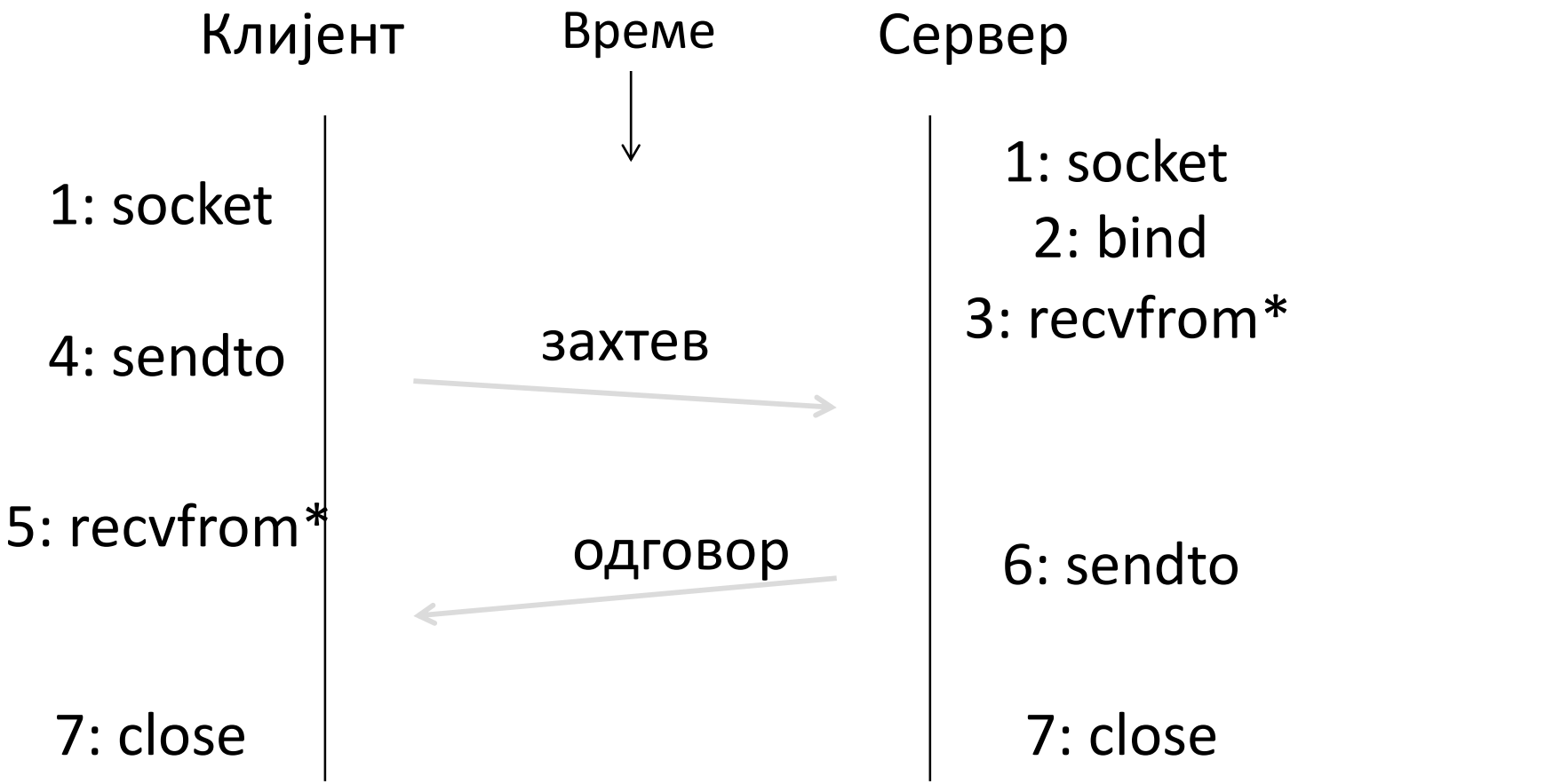
UDP – User (Unreliable) Datagram Protocol

UDP - User Datagram Protocol

- Користе програми којима није (претерано) битна поузданост или се базирају на порукама
  - Voice-over-IP (непоуздано)
  - DNS, RPC (засновано на порукама)
  - DHCP (засновано на порукама)

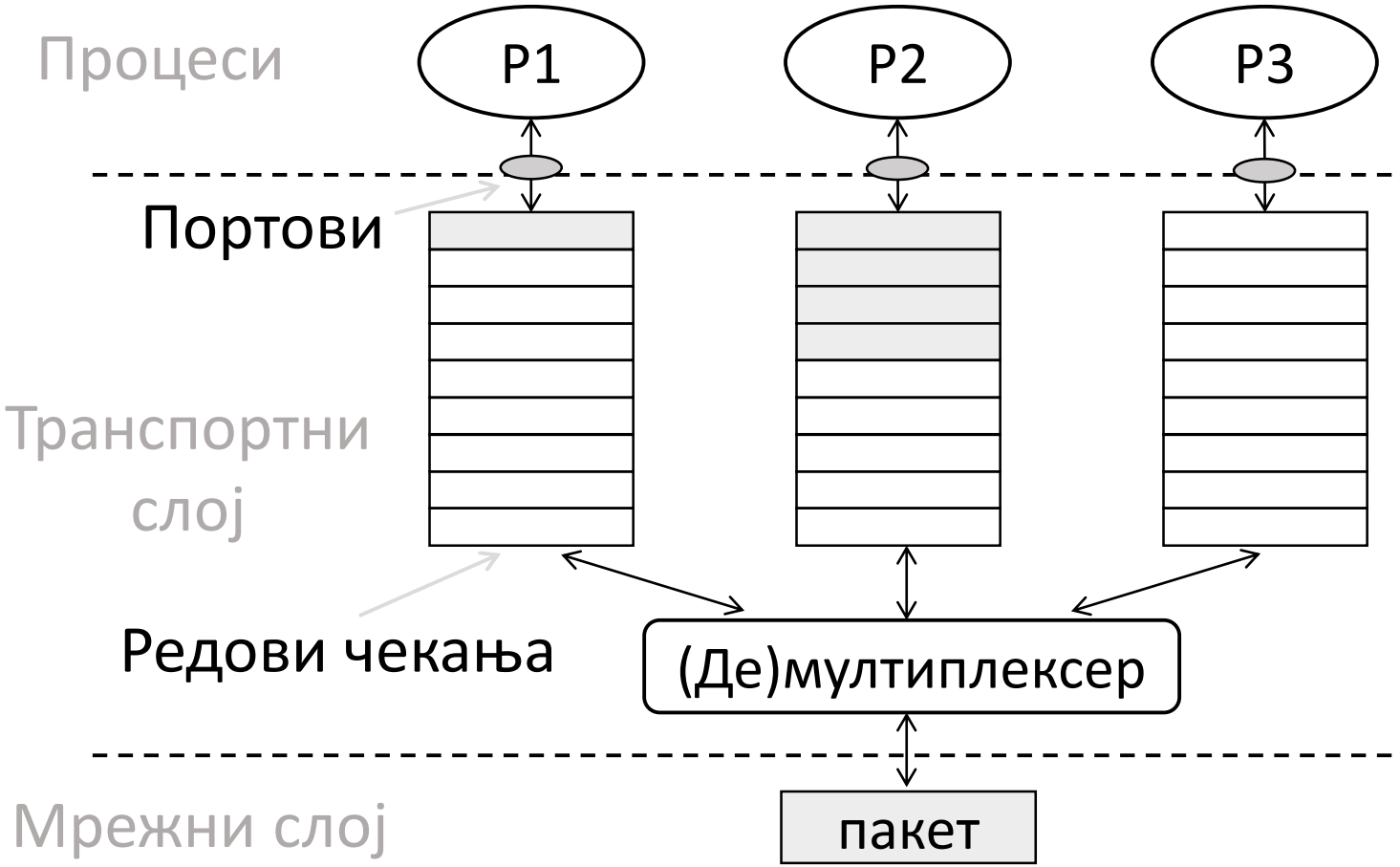
Постоје и други, мање познати протоколи у транспортном слоју за нпр. поуздани пренос порука

Сокети у случају датаграма (2)



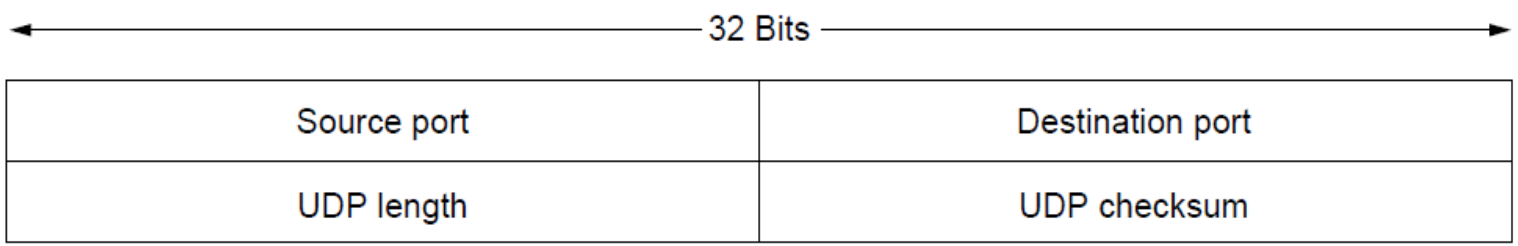
*= блокирајући позив

UDP бафери


Процеси
P1
P2
P3
Портови
Транспортни слој
Редови чекања
(Де)мултиплексер
Мрежни слој
пакет

UDP заглавље

- Користи порт како би препознао процес
- Величина датаграма до 64К
- Контролни збир (16 бита)



UDP

- UDP је завршен!
- Сви даљи механизми ће бити коришћењу у оквиру TCP ...

Транспортни слој

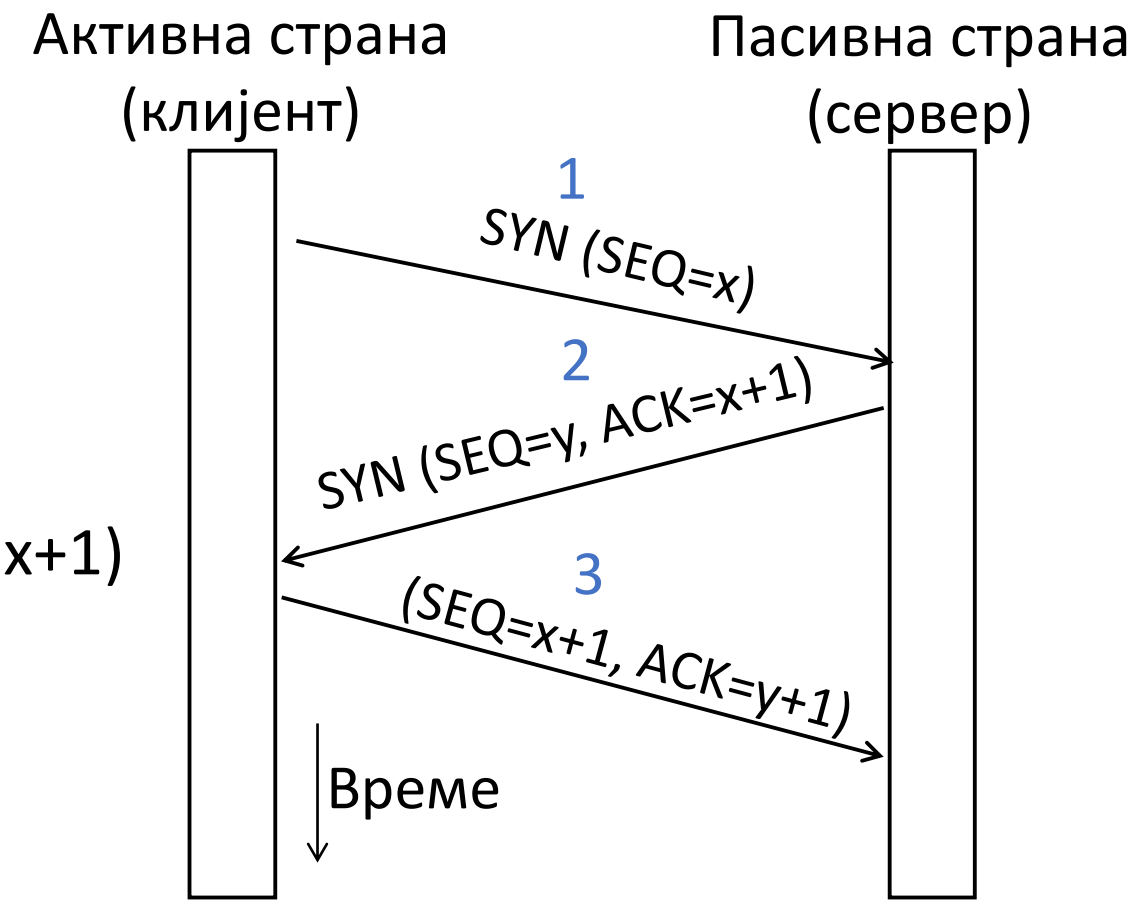
Успостава и прекид везе

Успостава везе

- Крајњи чворови морају бити свесни успоставе везе
  пре било каквог слања или примања података
  - Морају се договорити о скупу параметера, нпр. максимална величина сегмента
- Успостава везе подразумева:
  - Подешавање стања крајњих чворова
  - Попут „позивања“ приликом телефонског разговора
  - Стране такође треба да усагласе почетне бројеве сегмената
    како би се даље могао имплементирати протокол клизних прозора
  - Клизни прозори се овде користе између крајњих тачака
    - Код слоја везе су се користили између суседних тачака

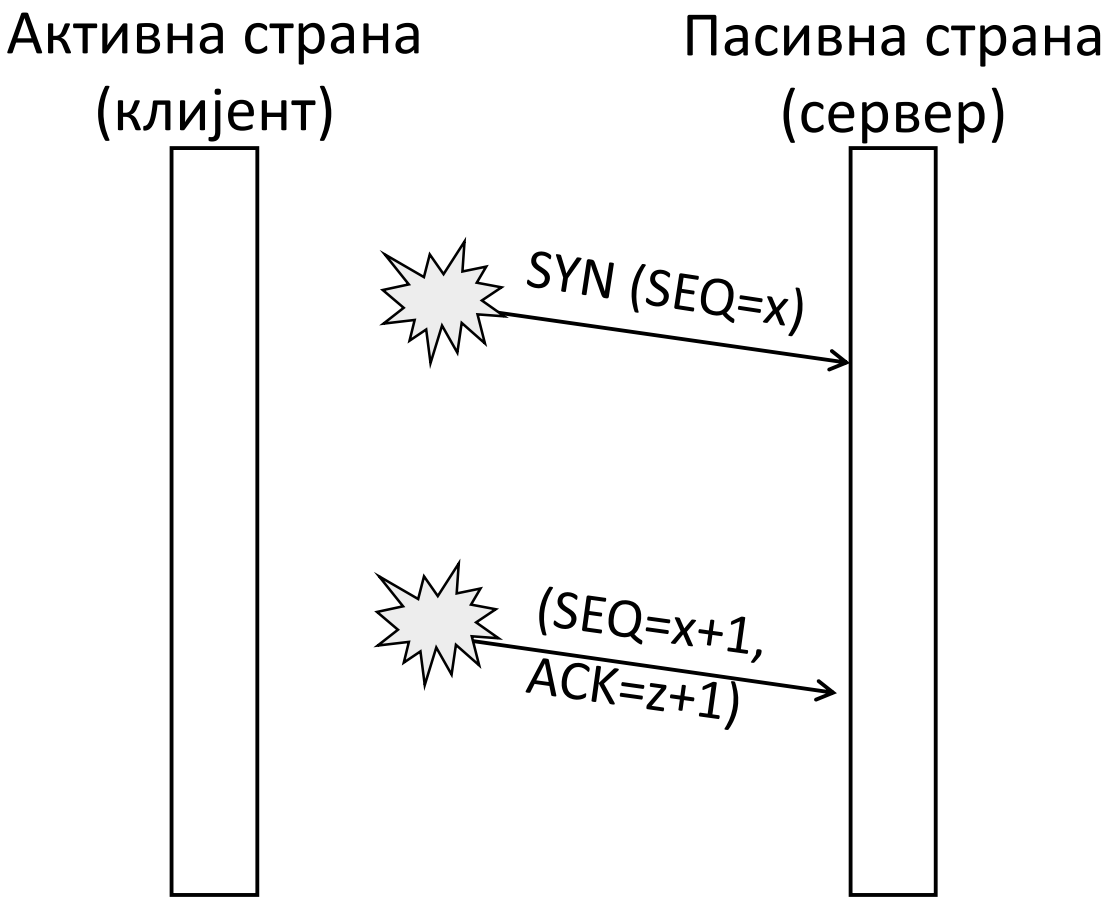
Трофазно руковање

- Три фазе:
  - Клијент пошаље сегмент SYN(x)
    - X – почетни редни број сегмента, обично случајан број из неког опсега
  - Сервер одговара:
    - Шаље број који следећи пут очекује и шаље свој почетни број SYN(y)ACK(x+1)
  - Клијент потврђује број ACK(y+1)
  - SYN сегменти се поново шаљу ако се изгубе



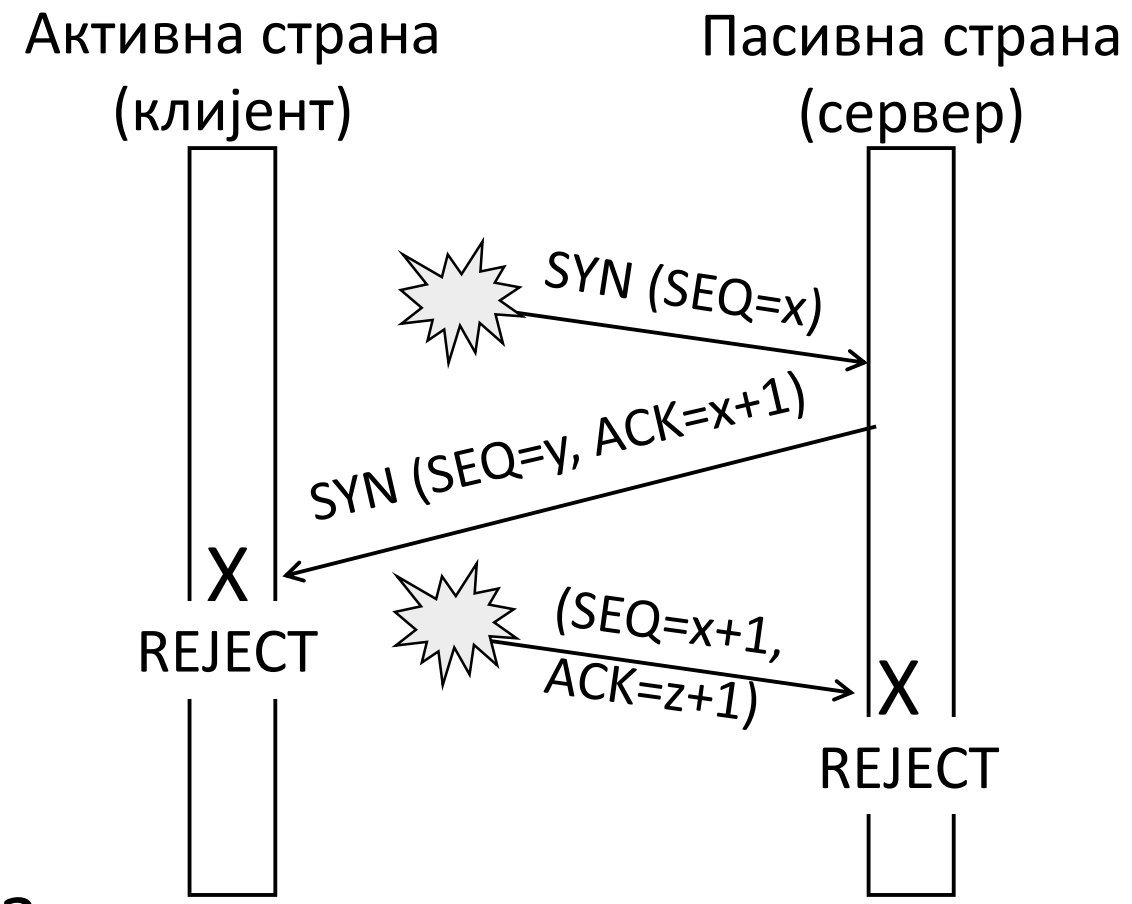
Трофазно руковање (2)

- Претпоставимо да се десило кашњење и ретрансмисија:
  - Стижу закаснели дупликати од стране клијента



Трофазно руковање (3)

- Претпоставимо да се десило кашњење и ретрансмисија:
  - Стижу закаснели дупликати од стране клијента
- Видимо да то неће проћи, тј. веза ће бити одбијена на обе стране!
  - Редни бројеви ће се разликовати
  - Бирају се насумично, обично из великог опсега, нпр. 32 бита

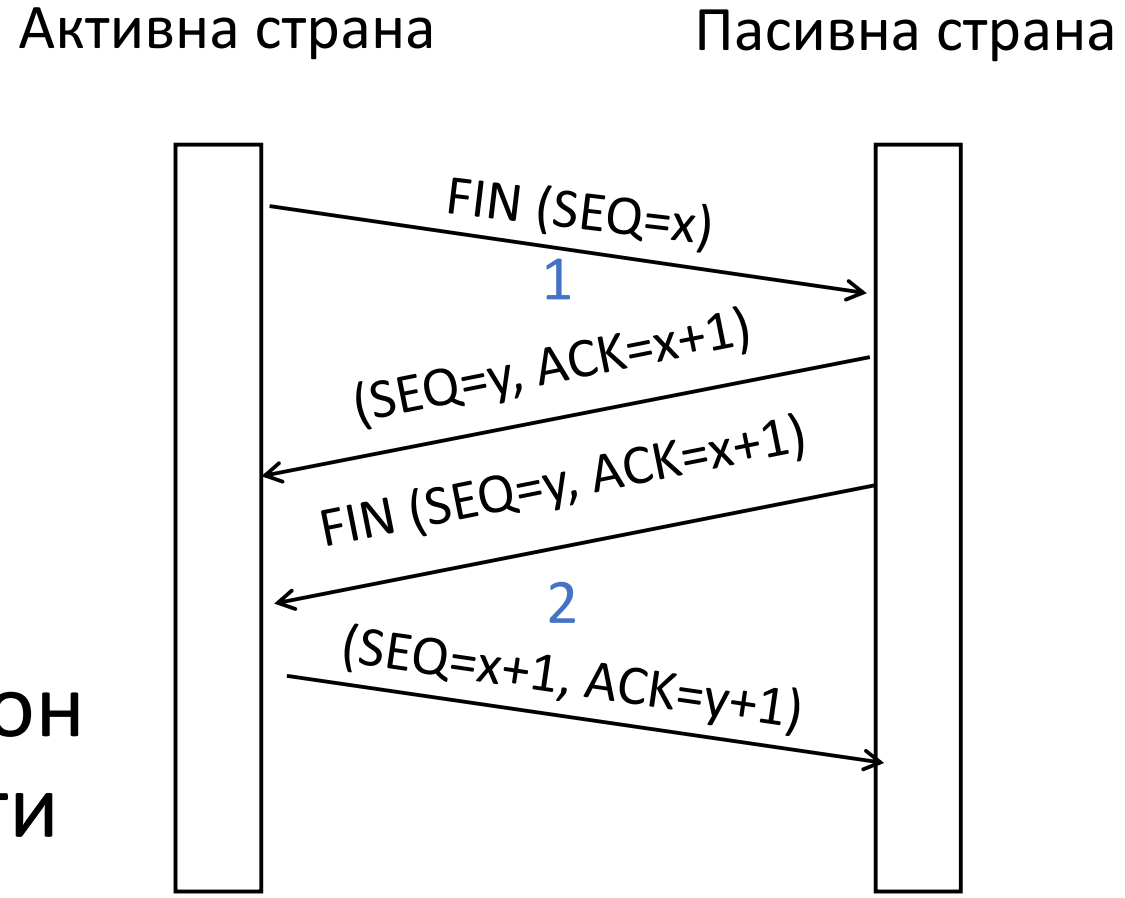


Прекидање везе

- Обе стране треба да прекину везу
  - Стање везе на обе стране мора да буде поништено
- Јако је битно ово урадити поуздано:
  - Не сме да се деси да једна страна затвори,
    а да друга то не уради
  - Једна страна је увек иницијатор прекида (активна),
    а друга је пасивна
  - Не мора клијент бити активна страна,
    овде и сервер може да захтева прекид

Прекидање везе (2)

- Два корака:
  - Активни шаље FIN(x), пасивни потврђује са ACK(x+1)
  - Пасивни шаље FIN(y), активни потврђује са ACK(y+1)
  - FIN се поново шаљу ако се изгубе
- Свако гаси своју страну везе након слања FIN и добијања ACK за исти



Транспортни слој

Протокол клизних прозора (опет)

Протоколи клизних прозора

- Зашто поново радимо ове протоколе?
  - У слоју везе, ови протоколи су се односили на пренос података кроз кабл између два суседна чвора
  - Овде омогућавају пренос између крајњих тачака, било где на Интернету

Протоколи клизних прозора (2)

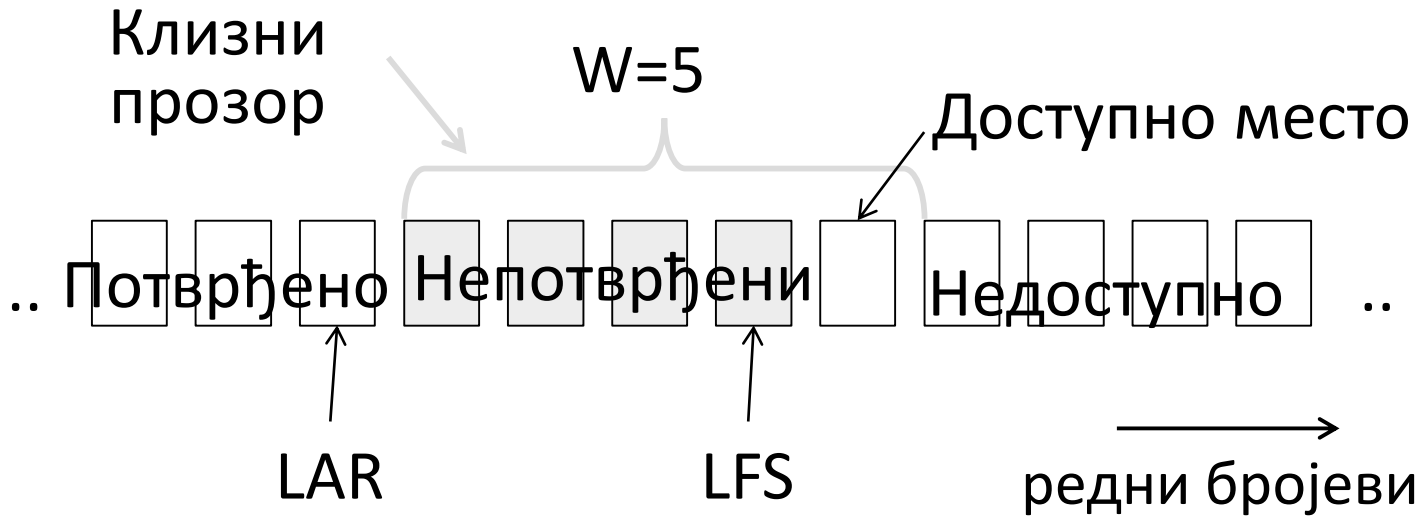
- Повећана поузданост на нижим нивоима доприноси ефикасности на вишим нивоима
  - Али није нужно имати те механизме на нижим нивоима
  - У екстремном случају, могло би све да се ради на транспортном:
    - Контрола тока
    - Провера грешака
    - Ово би било мање ефикасно, зашто?

Протоколи клизних прозора (3)

- Постоји доста варијација ових протокола, у зависности од:
  - Баферисања
  - Потврђивања порука
  - Ретрансмисија
- Врати се N
  - Једноставна верзија, може бити неефикасна
- Селективно понављање
  - Сложеније, али ефикасније
- Заједничко за све је да:
  - Од апликативног слоја добијају сегменте по реду!
  - Апликативном слоју шаљу сегменте по реду!

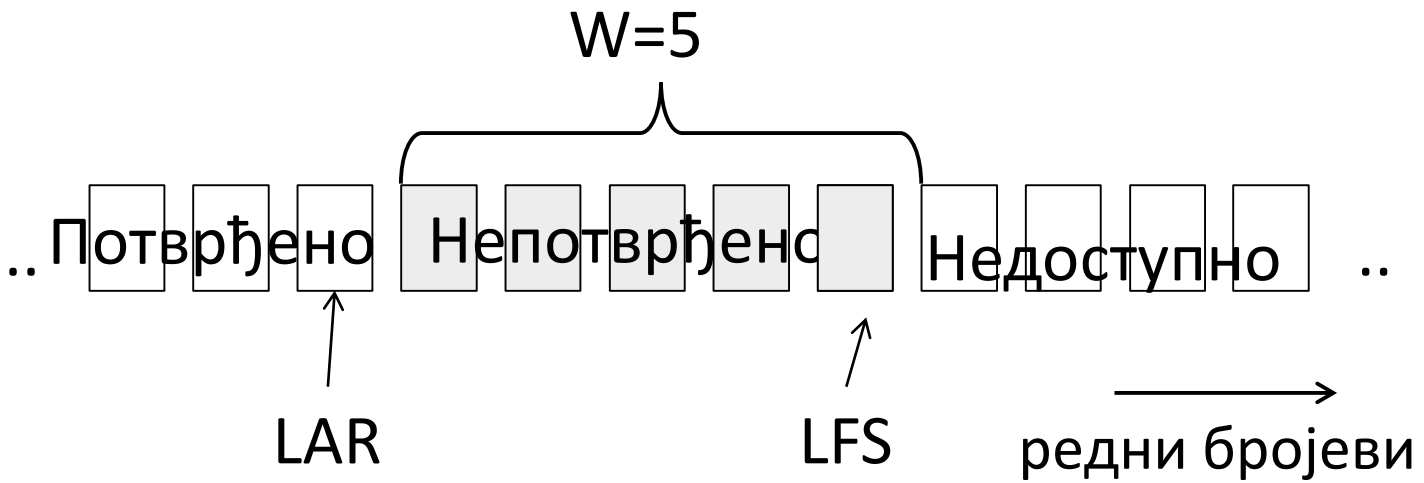
Клизни прозори - пошиљалац

- Пошиљалац баферише највише W сегмената док не стигну потврде за њих
  - LFS=последњи послат сегмент
    LAR=последњи потврђени сегмент пре кога су сви потврђени
  - Шаље док год је LFS – LAR ≤ W



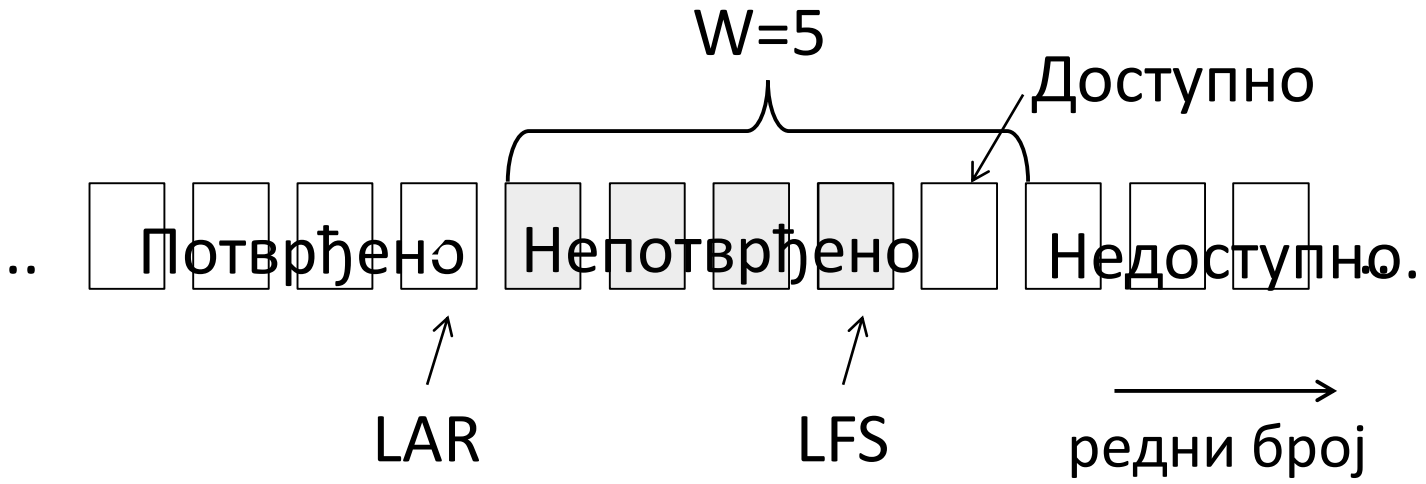
Клизни прозори – пошиљалац (2)

- Транспортни слој прихвата од апликативног још један сегмент ...
  - Транспортни шаље, јер је LFS–LAR → 5



Клизни прозори – пошиљалац (3)

- У међувремену стиже потврда за наредни сегмент...
  - Прозор се помера, и једно место у баферу је ослобођено
  - LFS–LAR → 4 (поново може да се шаље један)



Клизни прозори – прималац варијанта „Врати се N“

- Прималац има бафер величине 1
  - Притом чува и вредност променљиве
    LAS = редни број последњег сегмента прослеђеног апликативном слоју
- Након примања сегмента:
  - Ако је редни број LAS+1, онда га:
    - прихвати,
    - проследи апликативном,
    - ажурирај LAS:=LAS+1
    - и пошаљи потврду
  - Иначе одбаци

Клизни прозори – прималац варијанта „Селективно понављање“

- Прималац прослеђује апликативном по реду,
  а баферише сегменте и ако нису по реду (бафер величине W)

- Путем ACK сегмента, прималац:
  - потврђује највиши уређени сегмент,
  - а додатно и шаље информацију о сегментима који нису по реду
- TCP користи овај приступ

Клизни прозори – прималац варијанта „Селективно понављање“ (2)

- Баферише W сегмената
- Одржава стање променљиве LAS
- Прихвата ако је из опсега [LAS+1, LAS+W] и притом:
  - Баферише сегменте из опсега [LAS+1, LAS+W]
  - Прослеђује апликативном слоју ако стигне сегмент са бројем LAS+1
    - Притом, ажурира LAS:=LAS+1
  - Шаље потврду

Клизни прозори - ретрансмисије

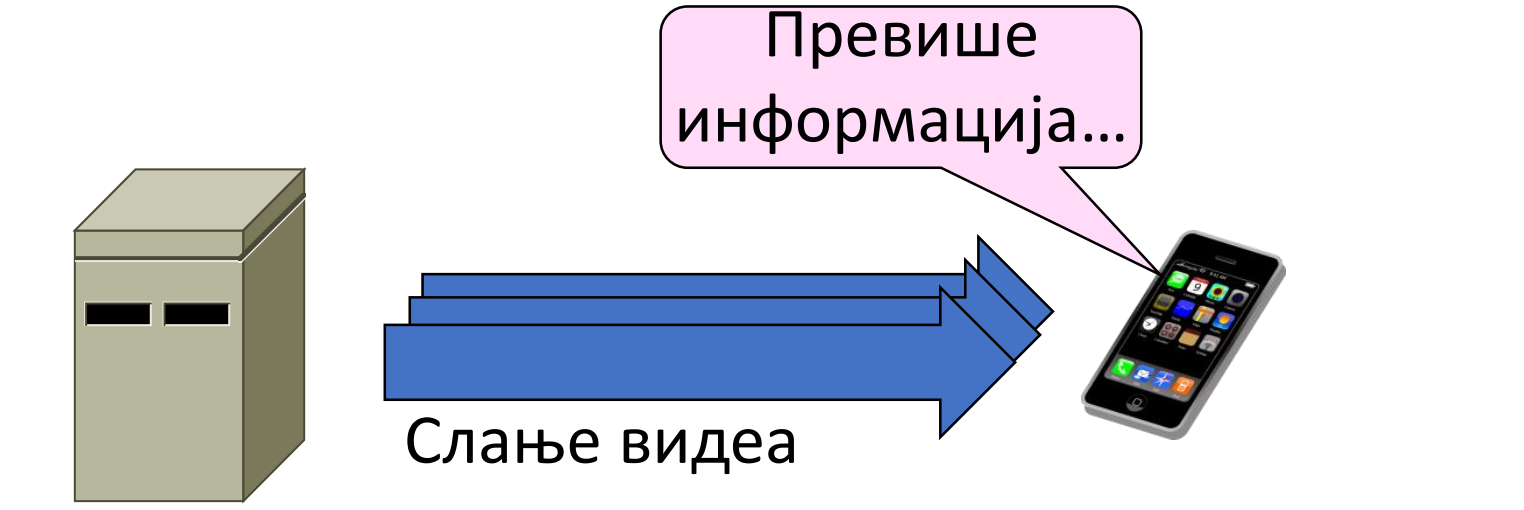
- „Врати се N“ пошиљалац има један тајмер:
  - Када истекне, поново шаље све баферисане сегменте почев од LAR+1
- „Селективно понављање“ пошиљалац има тајмер за сваки непотврђени сегмент:
  - По истеку, шаље поново
  - У просеку ради мање ретрансмисија

Транспортни слој

Контрола тока

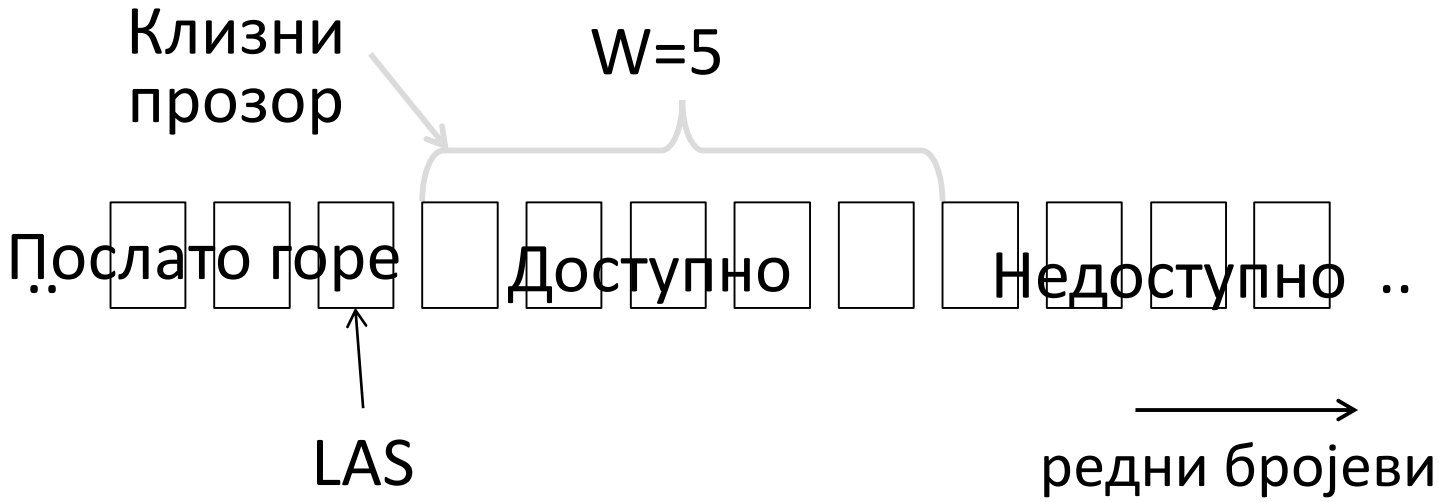
Контрола тока

- Шта ако прималац споро прихвата податке?
  - Мобилни телефон можда спорије шаље информације апликативном слоју
  - Потребно је некако усагласити брзину слања...



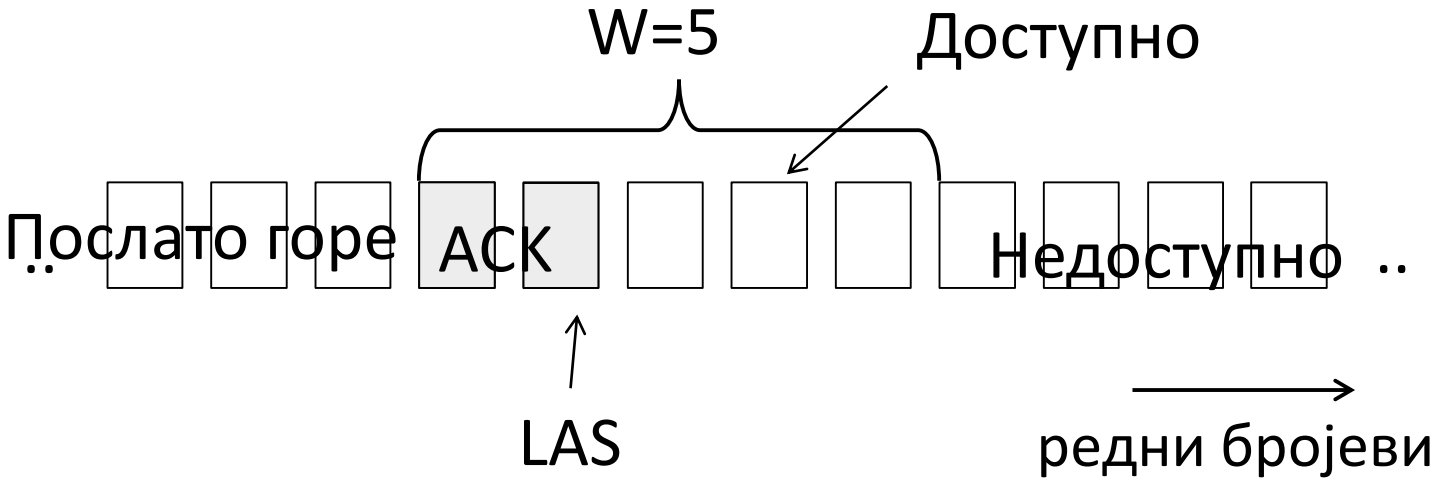
Клизни прозори - прималац

- Нека нпр. прималац има бафер величине W
  - Иницијално, бафер је празан
  - Део пре тога је послат апликативном слоју...



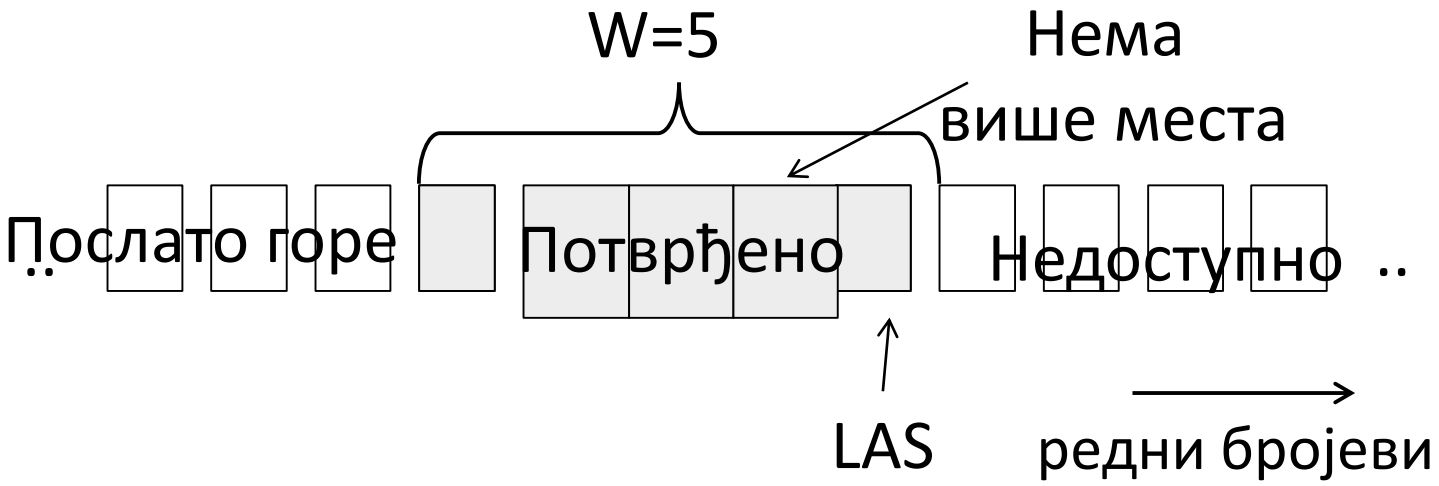
Клизни прозори - прималац (2)

- Нека након тога пристигну два сегмента, али програм и даље не позива recv()
  - LAS расте, али не можемо да померимо прозор, јер апликативни слој још није добио податке!



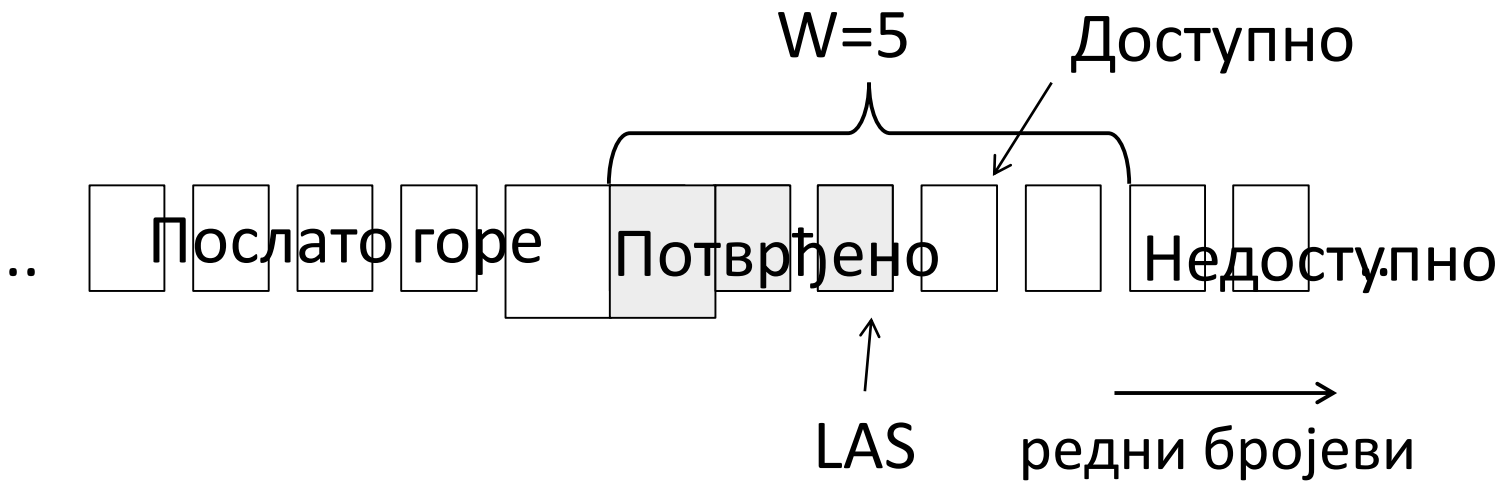
Клизни прозори - прималац (3)

- Када стигну наредни сегменти, попуњава се бафер
  - Након што се попуни, више није могуће примати, све док апликативни слој не прихвати сегменте



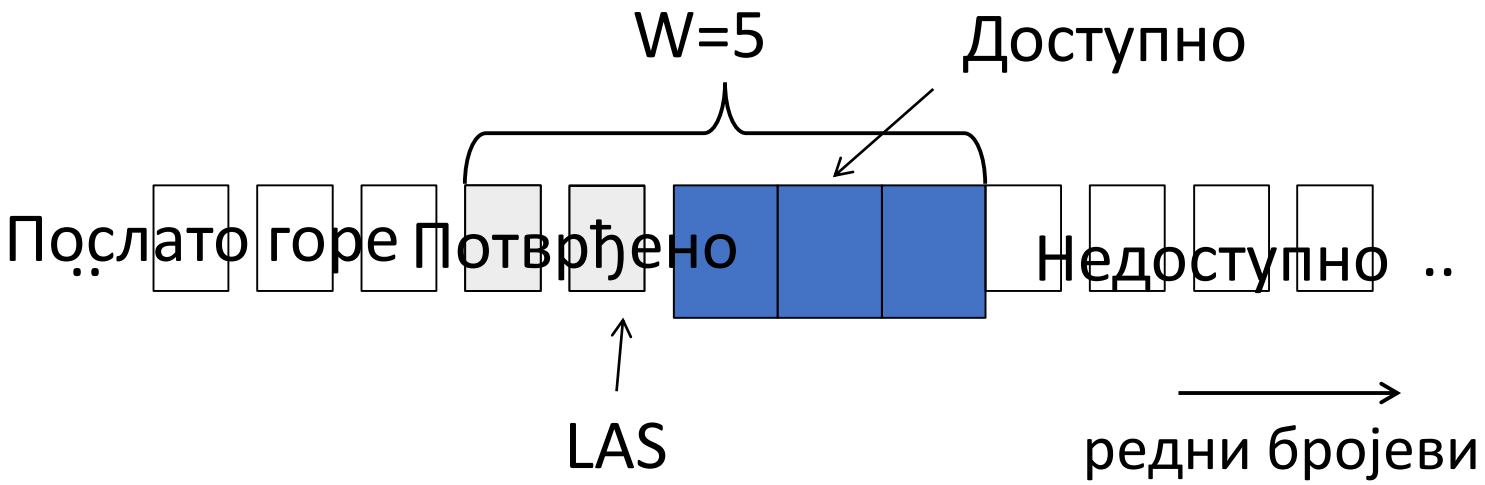
Клизни прозори - прималац (4)

- Апликативни коначно прихвата два сегмента
  - Прозор се помера



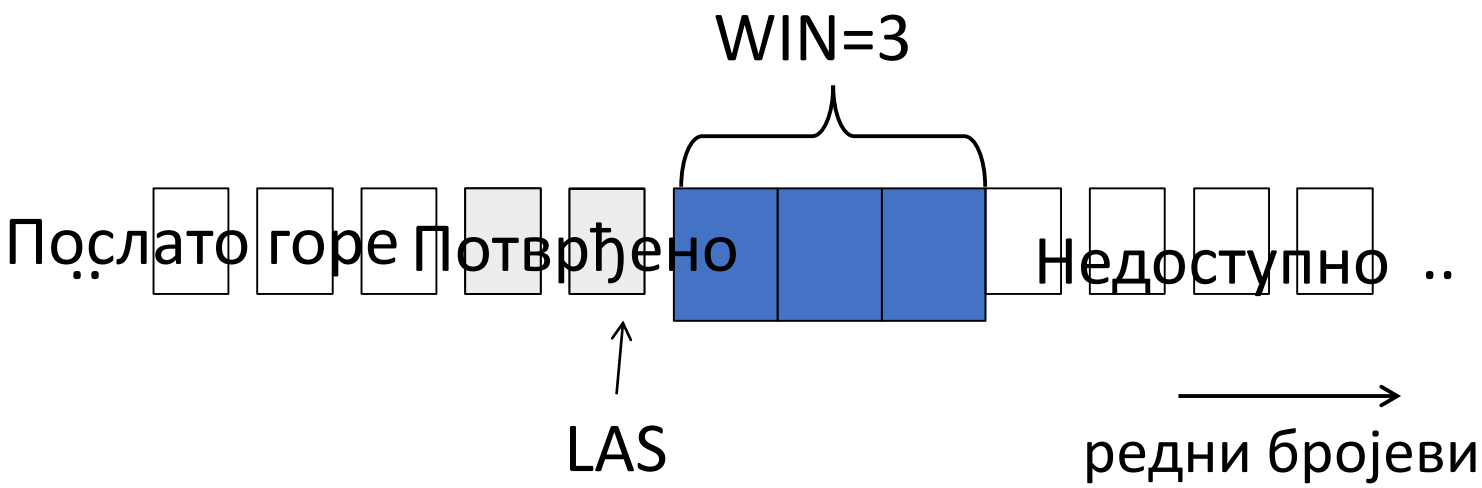
Контрола тока

- Избегавање губитка на страни примаоца:
  - Прималац говори пошиљаоцу доступно стање бафера
  - WIN= број доступних места у баферу



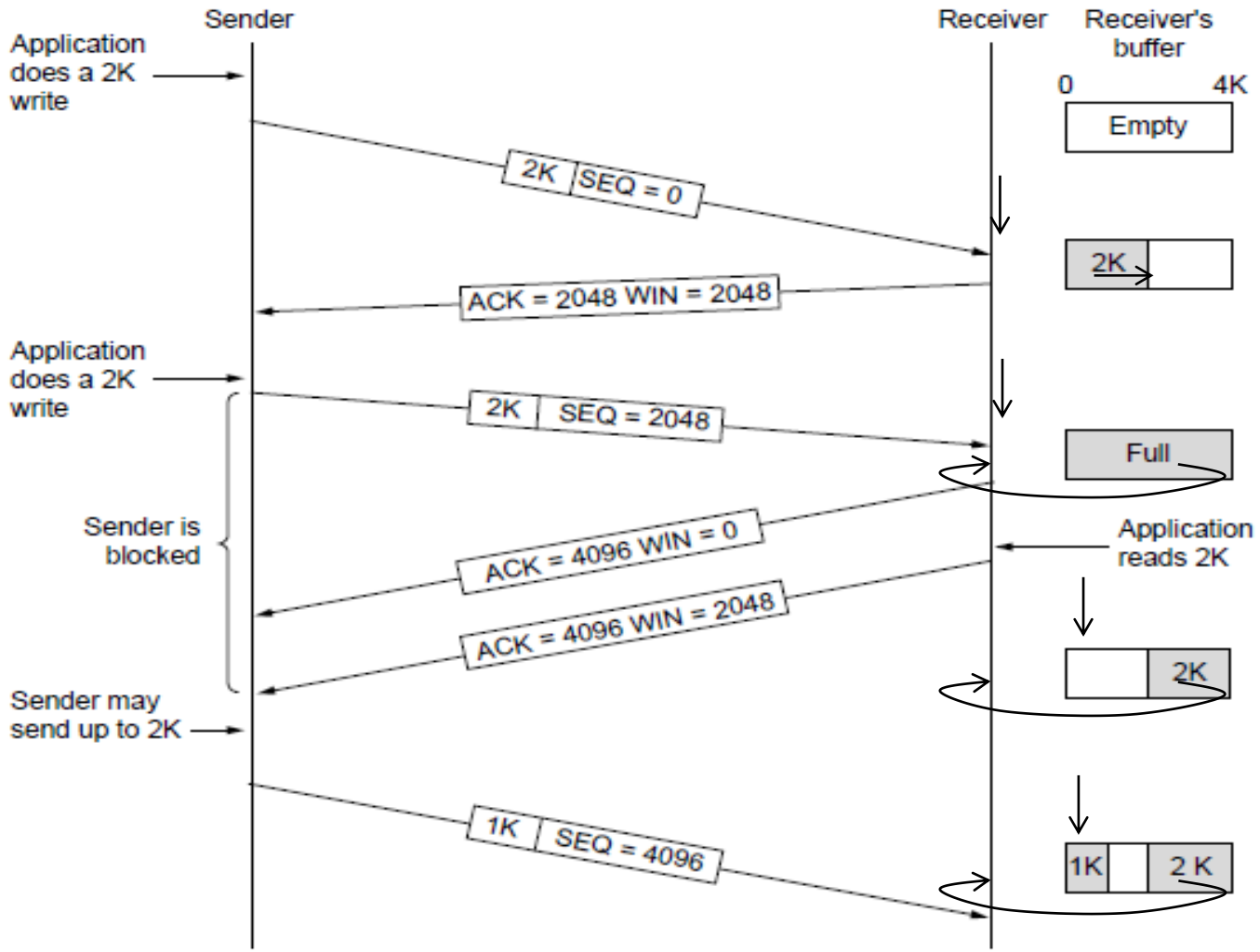
Контрола тока (2)

- Пошиљалац користи WIN
  као ефективну информацију о величини прозора



Контрола тока (3)

- TCP пример
  - 4KB бафер код примаоца
  - Бафер је циркуларан



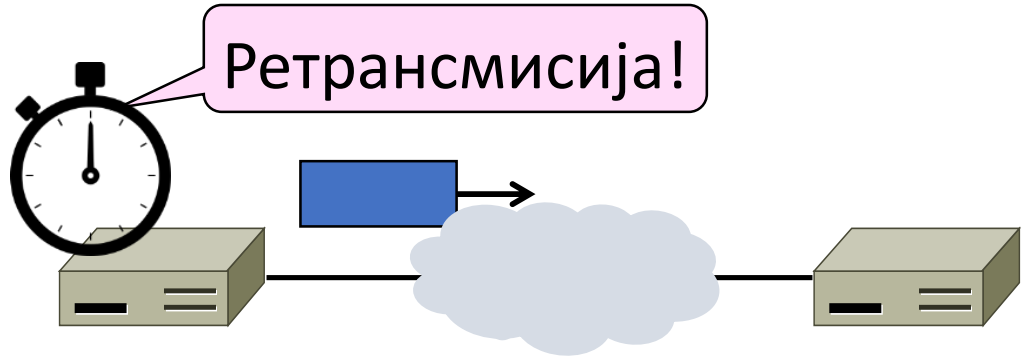
- SEQ + величина сегмента < ACK+WIN

Транспортни слој

Паузе (тајмаути) за ретрансмисију

Ретрансмисије

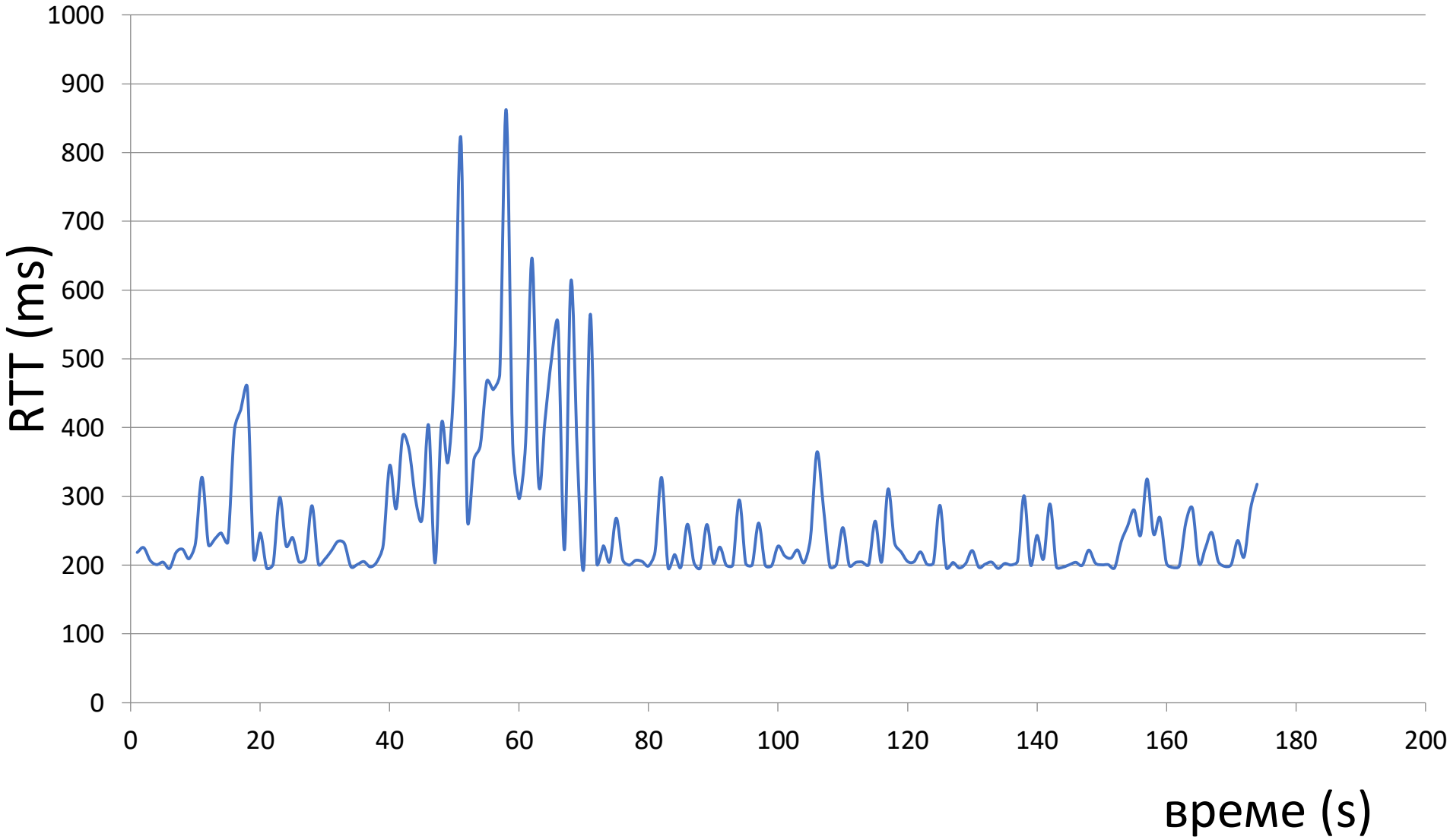
- Стратегија за детекцију губитка је истек паузе?
  - Постави тајмер када је сегмент послат
  - Деактивирај тајмер када се добије потврда
  - Ако се тајмер активира, уради ретрансмисију
- Да ли је ово добра стратегија?



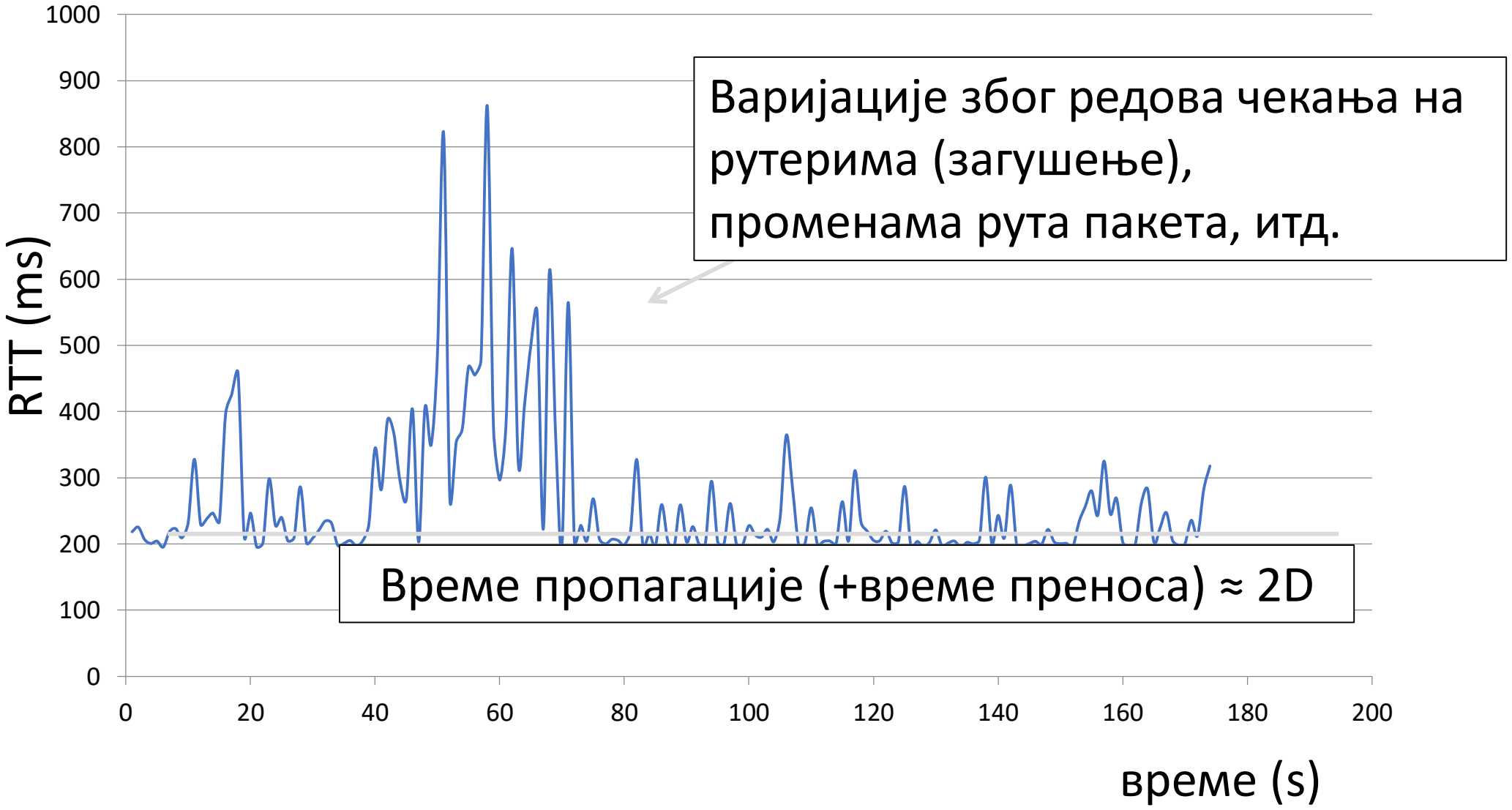
Одређивање трајања паузе

- Пауза мора да буде добро оцењена
  - Превелике паузе успоравају кретање прозора
  - Прекратке изазивају сумњиве ретрансмисије
- Лако се одређује за LAN (слој везе)
  - Кратак кабл, познате руте → предвидљив RTT
- Тешко за Интернет (транспортни слој)
  - Широк опсег, променљив RTT

Промена RTT кроз време (за исти пренос)



Промена RTT кроз време (2)

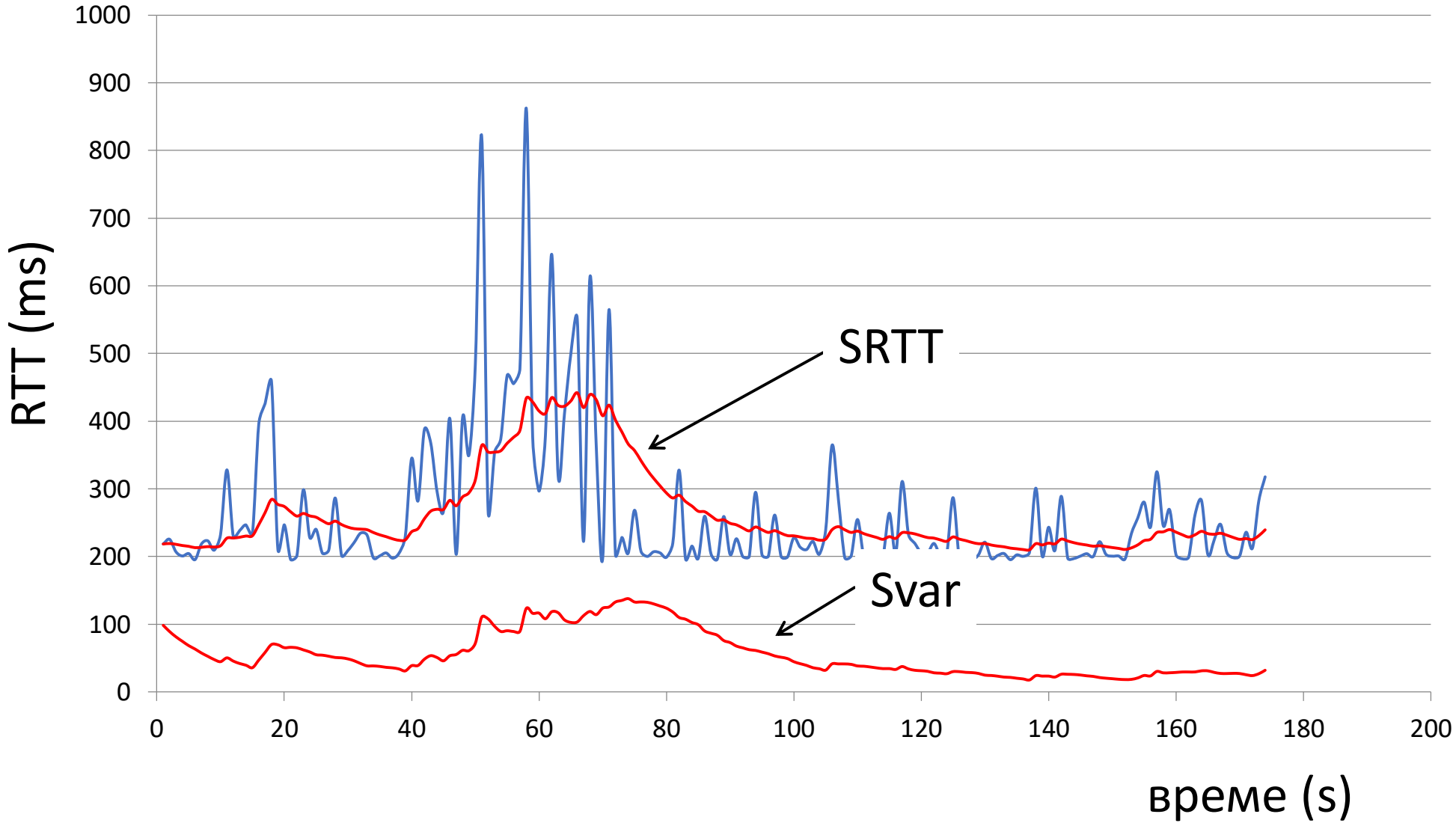


Промена RTT кроз време (3)

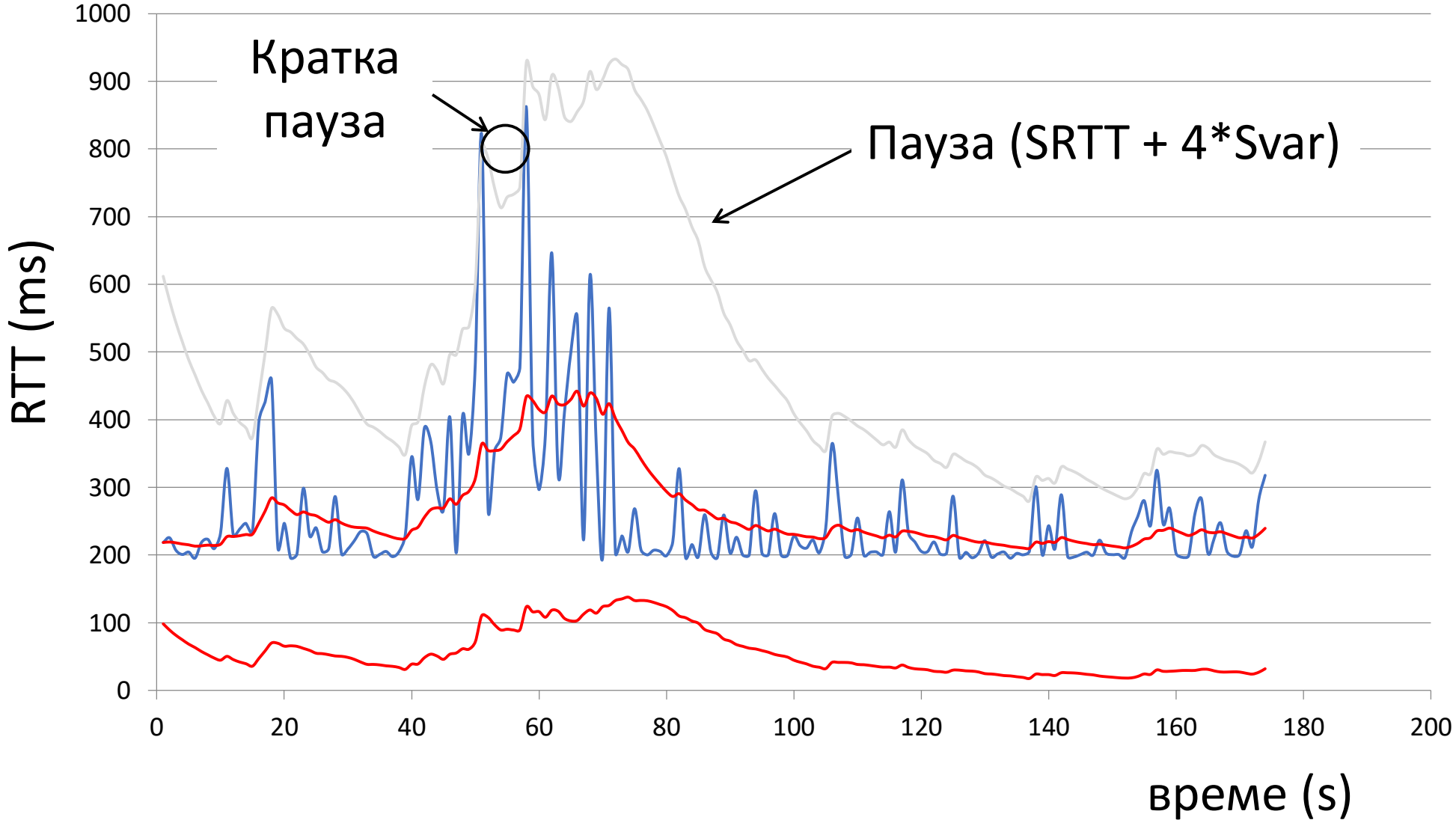
Прилагодљиве паузе (тајмаути)

- Идеја је да се оцени краткорочни RTT и његова варијанса
- И онда то искористи за постављање трајања паузе
- Формула заснована на померајућим просецима:
  1. $SRTT_{N+1} = 0.9*SRTT_N + 0.1*RTT_{N+1}$
  2. $Svar_{N+1} = 0.9*Svar_N + 0.1*|RTT_{N+1} - SRTT_{N+1}|$
- Пауза треба да буде увек изнад оцене RTT:
  - TCP $Timeout_N = SRTT_N + 4*Svar_N$
  - Што је већа варијанса, мање смо сигурни, па је и горња граница више удаљена

Пример прилагодљиве паузе



Пример прилагодљиве паузе (2)



Транспортни слој

TCP – Transmission Control Protocol

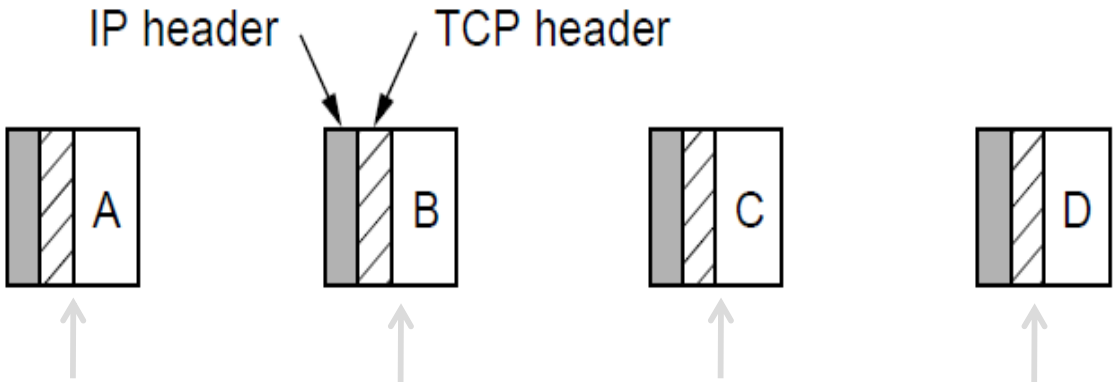
TCP својства

- Поуздан ток бајтова
- Засован на везама
- Клизни прозори зарад поузданости
  - Са прилагодљивим паузама
- Контрола тока за споре примаоца

Поуздан ток бајтова

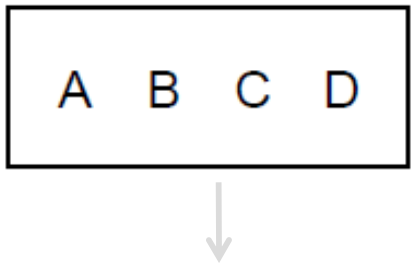
- Сегменти се могу кретати неуређено и непоуздано кроз мрежни и остале ниже слојеве
  - Међутим, транспортни слој их уређује, проверава, и апликативном шаље поуздано и по реду

Пошиљалац



Четири независно послата сегмента, сваки са по 512 бајтова

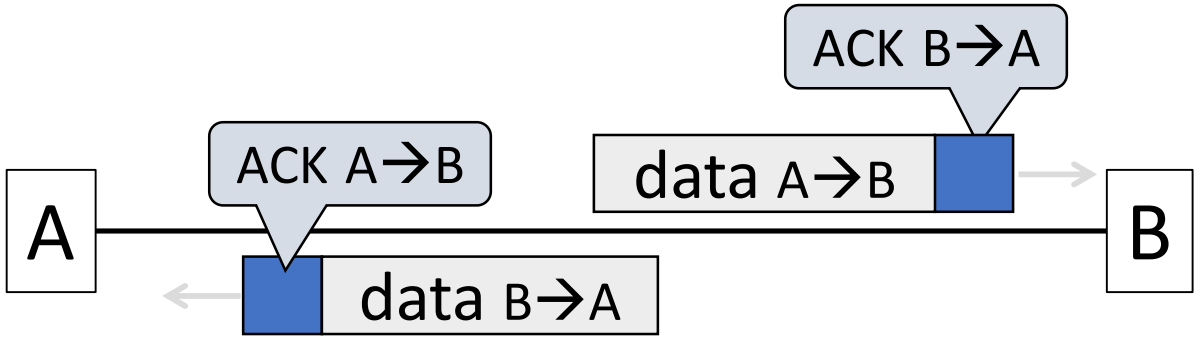
Прималац



2048 уређена бајта послата поуздано апликативном слоју

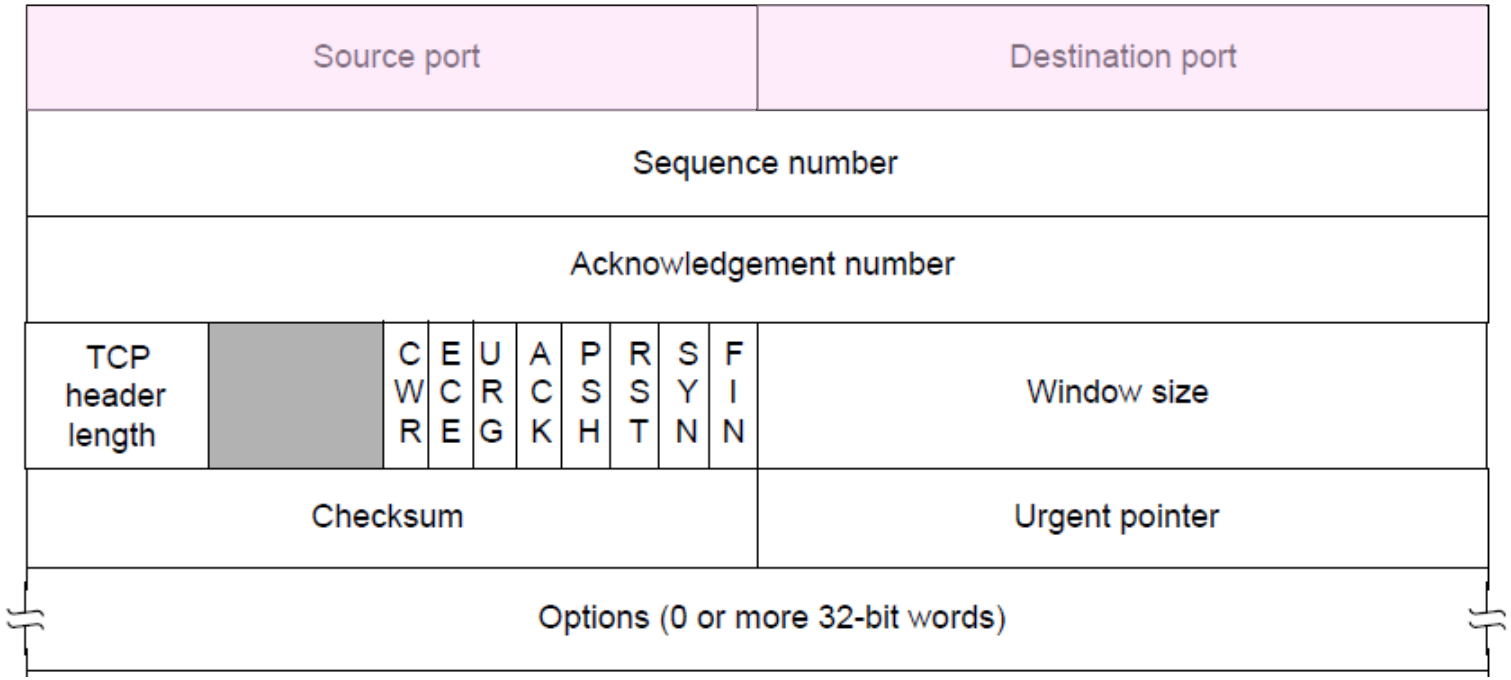
Поуздан ток бајтова (2)

- Слање и примање података у оба смера
  - Контролне информације (нпр. ACK) се често шаљу као делови долазних сегмената за податке из другог смера (шлепање)



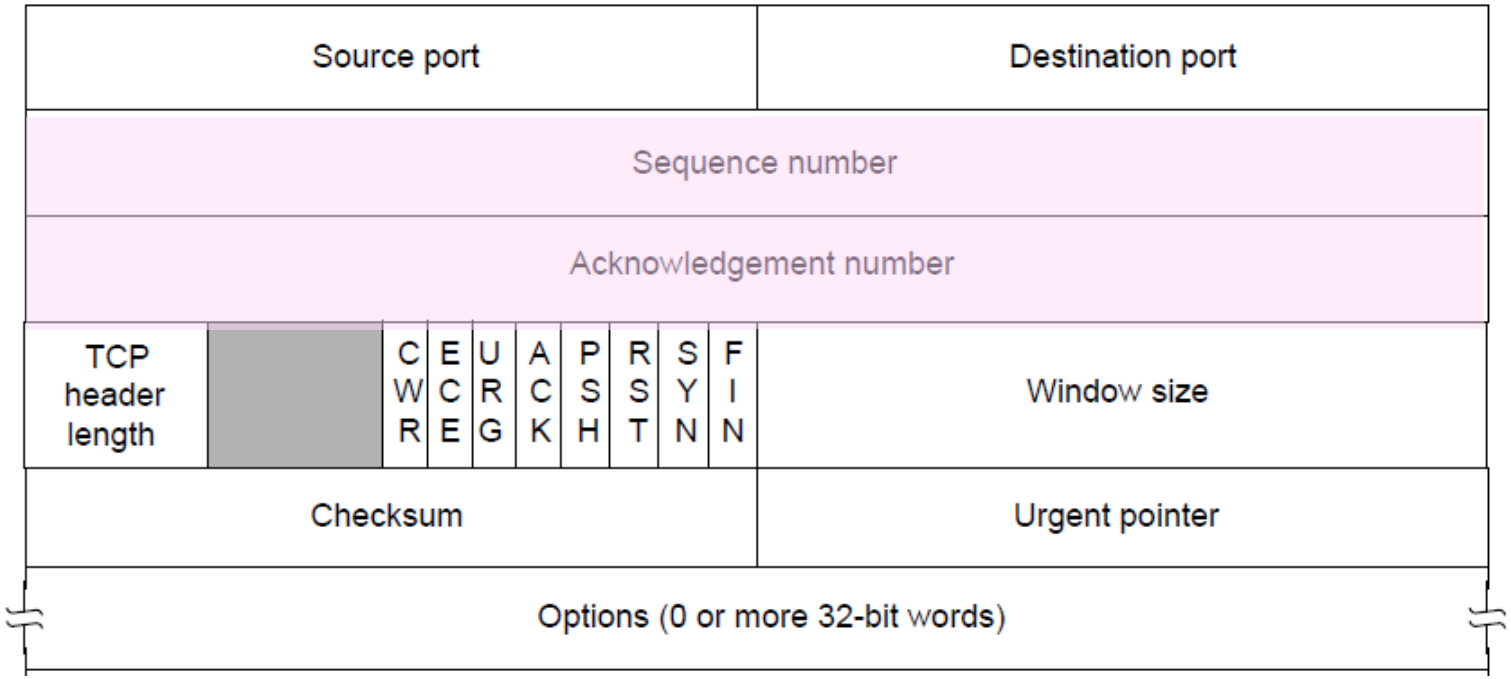
TCP заглавље

- Портови идентификују програме (socket API)
  - 16-битни идентификатори



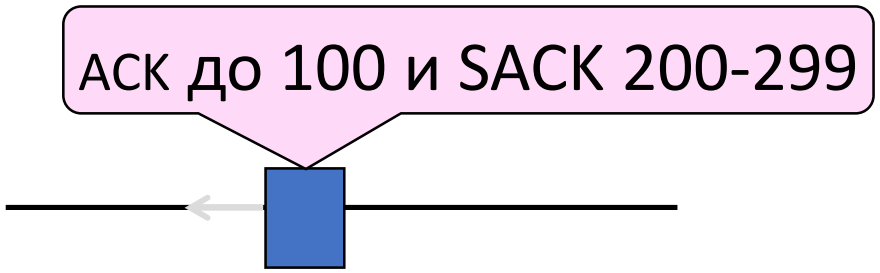
TCP заглавље (2)

- SEQ/ACK бројеви се користе у оквиру протокола клизних прозора



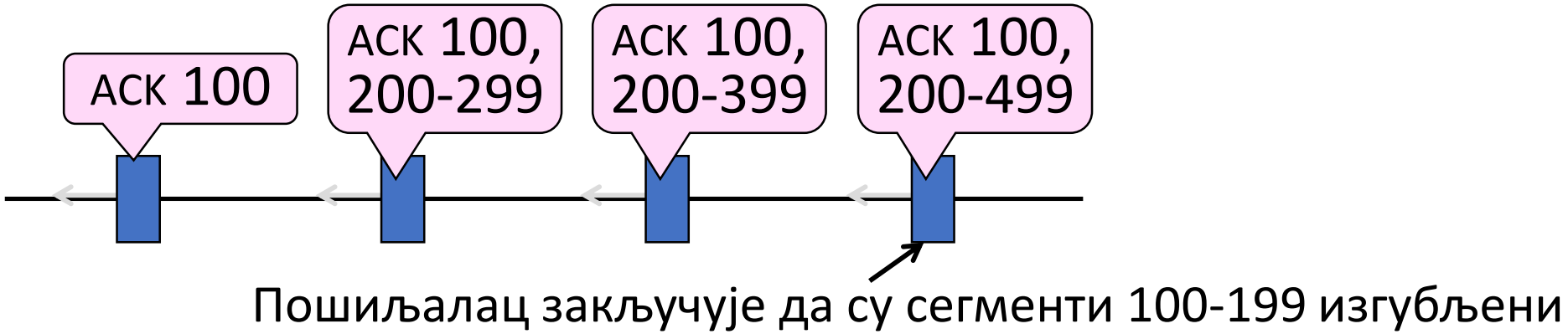
TCP клизни прозори - прималац

- Кумулативни ACK говори који је следећи очекивани бајт (“LAS+1”)
- Опционо, користе се и селективни ACK-ови (SACK) зарад оптимизације
  - Листање до три опсега примљених бајтова



TCP клизни прозори – пошиљалац

- Користи прилагодљиву паузу за ретрансмисију сегмената који почињу од LAS+1
- Користи хеуристику како би брже закључио који сегменти су изгубљени и тиме избегао истек паузе
  - Хеуристика: “три дуплирана ACK-а” имплицирају губитак



Остали детаљи о TCP

- Постоји још много детаља у вези са TCP протоколом...
- Такође постоје и други протоколи на транспортном слоју ...

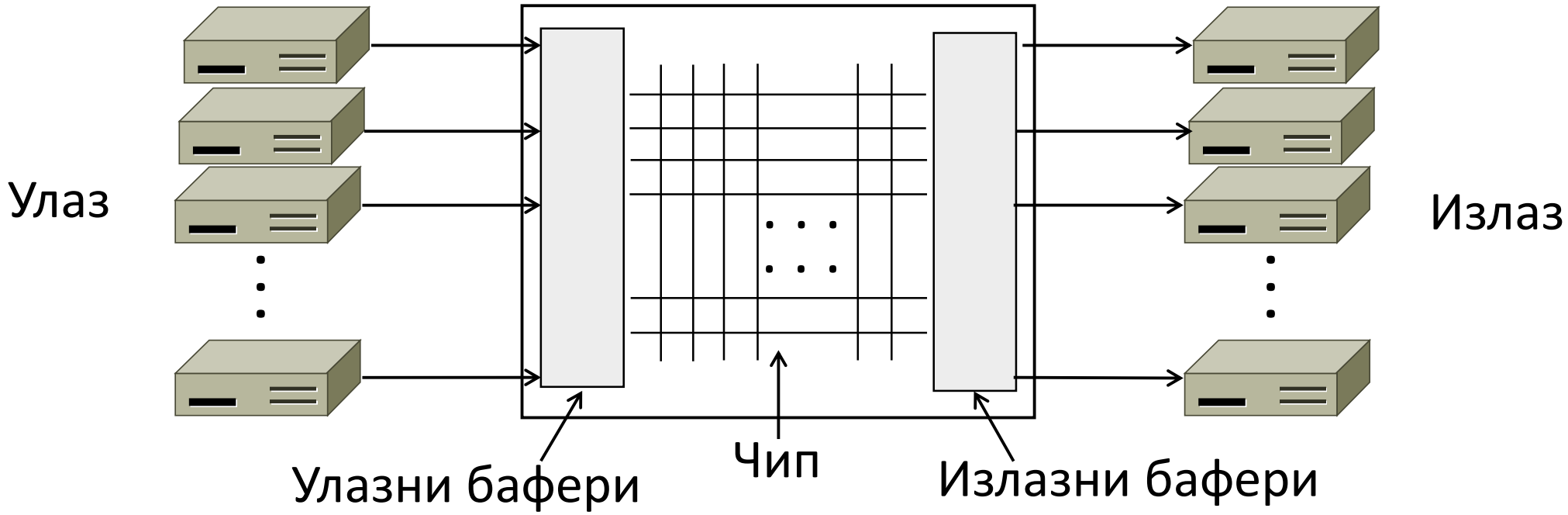
- Проблем загушења на транспортном слоју ћемо само формулисати
  - Поред тога, биће дата идеја за решавање овог проблема

Транспортни слој

Контрола загушења, формулација проблема и скица решења

Природа загушења

- Као што знамо, рутери и скретнице користе бафере, зарад побољшања перформанси
- Организација бафера је обично FIFO (редови чекања)

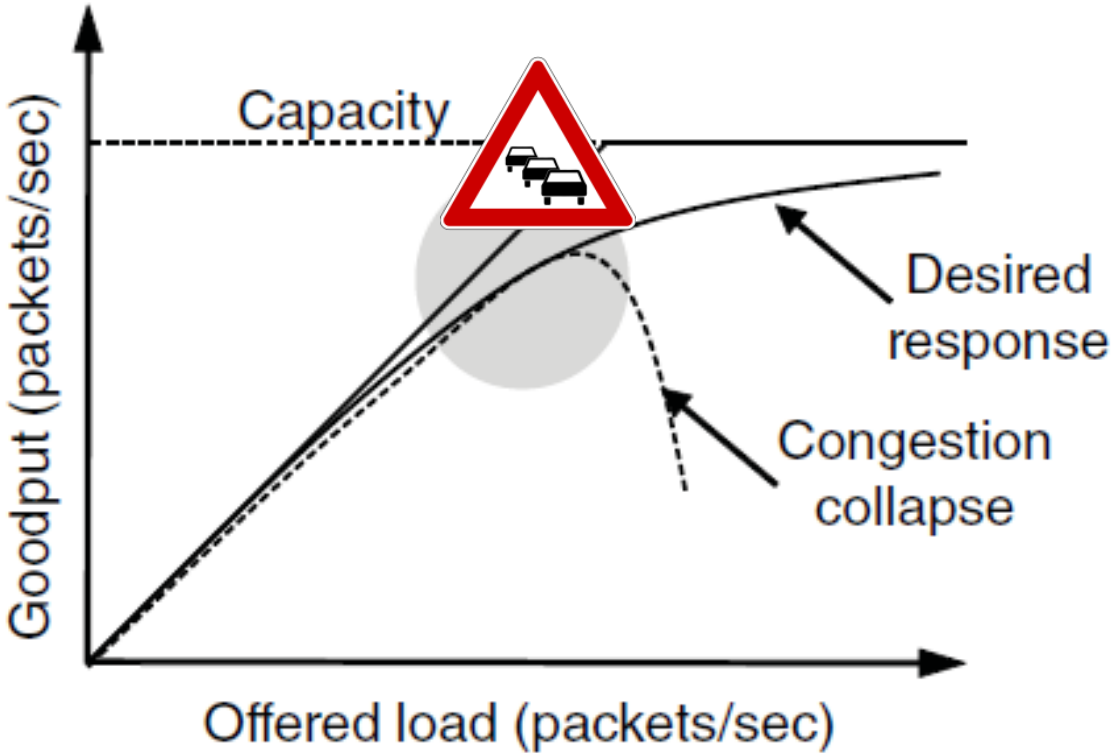


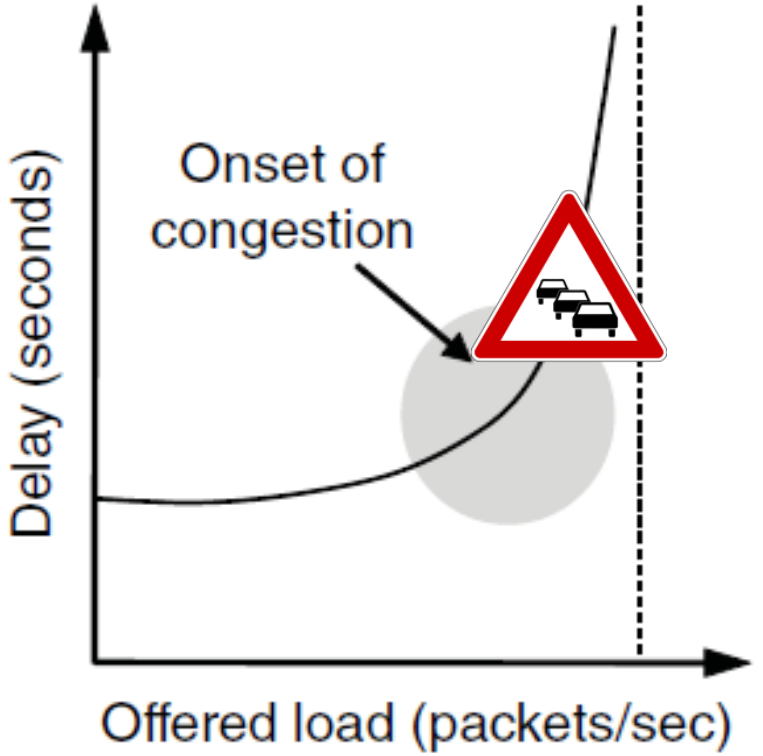
Природа загушења (2)

- Редови чекања помажу при апсорбовању краткорочних скокова у саобраћају (енг. Traffic bursts)
- Међутим, они нису дизајнирани
  за дугорочна или средњерочна стања
  у којима је улазни саобраћај већи од излазног саобраћаја
- Оваква стања се називају загушењем
  - Нпр. гужва у друмском саобраћају је последица шаблонске употребе путева (излазак, одлазак на посао и сл.)

Ефекти загушења

- Перформансе се драстично смањују како се повећава оптерећење – изазива се колапс





Алокација протока

- Битан задатак у решавању проблема загушења је додељивање капацитета пошиљаоцима
  - Добра додела треба да буде ефикасна и равноправна
- Ефикасност подразумева да је скоро цео капацитет употребљен, али нема загушења
- Равноправност подразумева да сваки пошиљалац добија рационални удео протока

Алокација протока (2)

- Изгледа да транспортни и мрежни слој морају да раде заједно на решавању овог проблема
- Мрежни слој детектује загушење
  - Само је он свестан овога
    (транпспортни је на вишем логичком нивоу, а слој везе на нижем)
- Транспортни слој изазива загушење
  - Али он може и да га разреши, тако што редукује оптерећење

Алокација протока (3)

- Оквирна идеја:
  - Пошиљаоци прилагођавају свој одлазни саобраћај на основу онога што детектују из мреже
  - Ово прилагођавање треба да има у виду ефикасност и равноправност
  - Прилагођавање мора да буде стално, јер се стање мреже стално мења

Транспортни слој

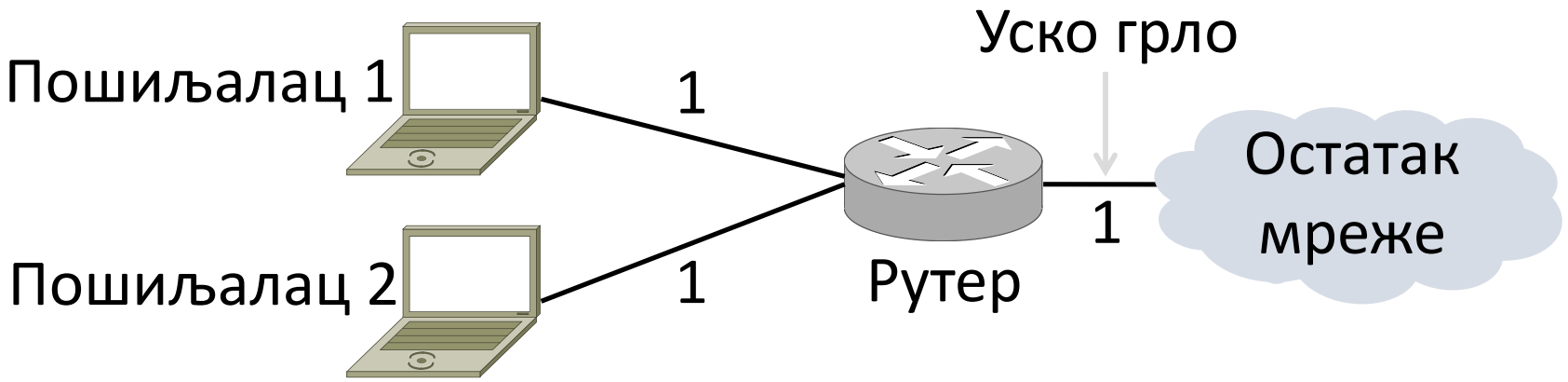
Адитивно повећање и умножено смањење
(Additive Increase Multiplicative Decrease – AIMD)

AIMD

- AIMD је контролни механизам који омогућава достизање добре алокације:
  - Пошиљаоци адитивно повећавају брзину слања података док мрежа не постане загушена
  - Након тога је умножено смањују када уоче загушење
  - TCP користи ово у некој форми

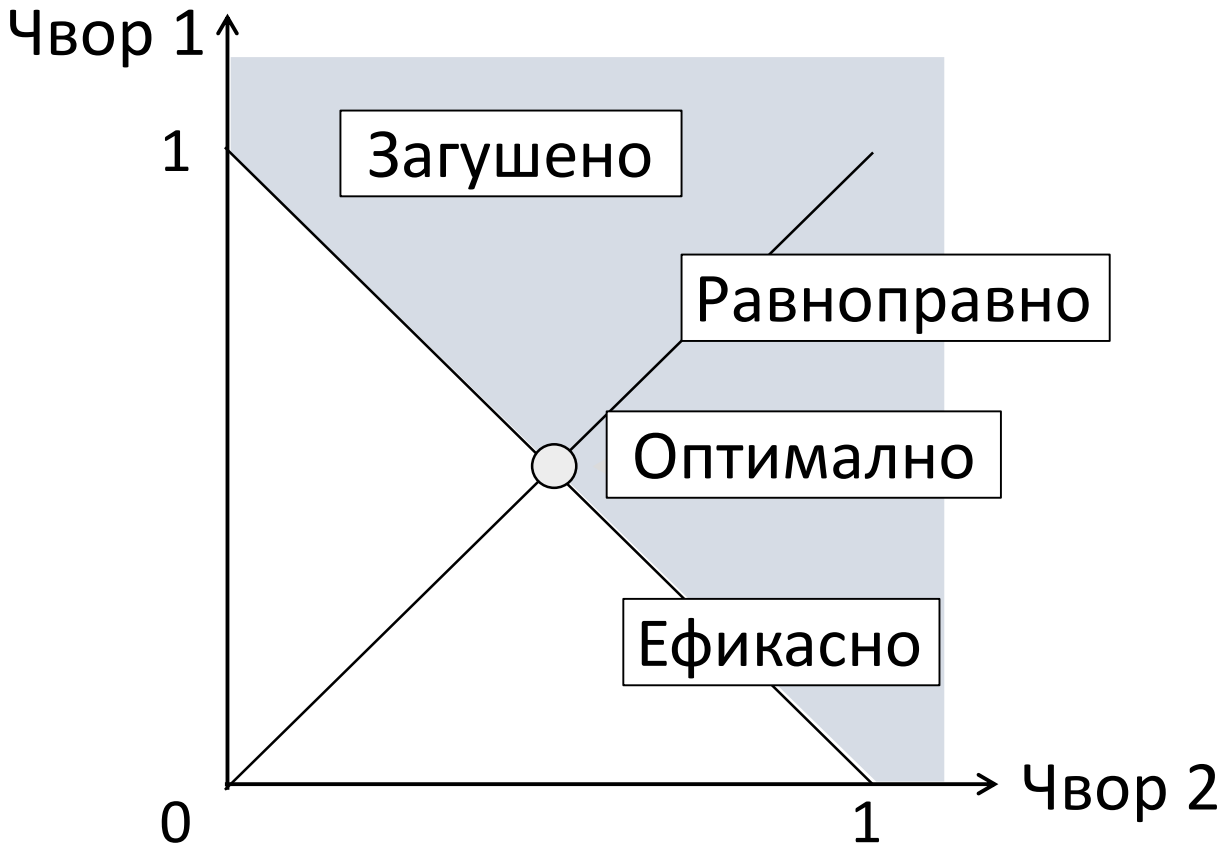
AIMD (2)

- Подаци пошиљаоца 1 и 2 пролазе кроз исту тачку загушења „уско грло“
  - Међутим, пошиљаоци не могу да комуницирају директно
- Рутер је тај који нпр. може да сигнализира
  - Шаље бинарно 0/1 ако (не/)постоји загушење



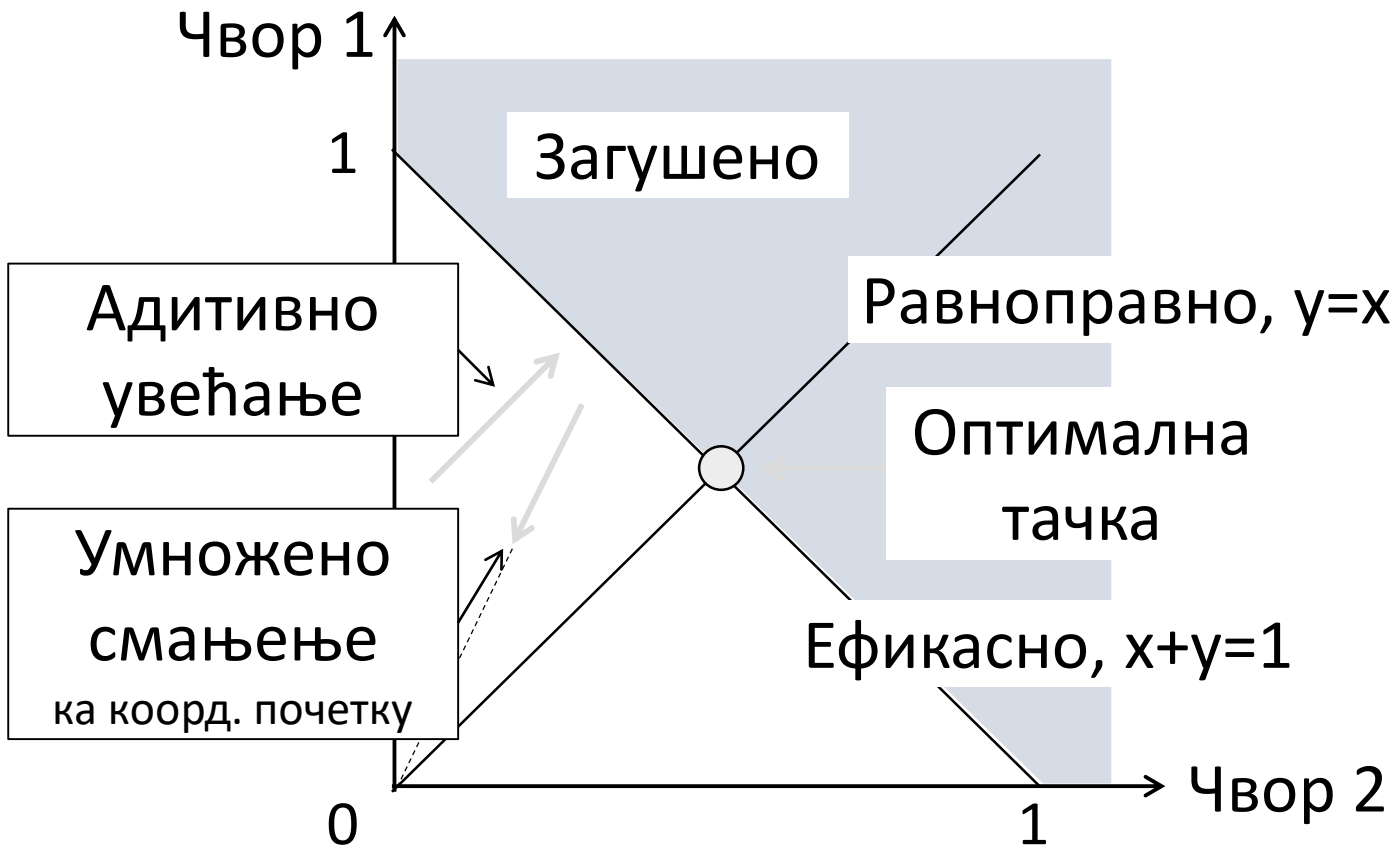
AIMD (3)

- Свака алокација је допустива, али нису све добре...



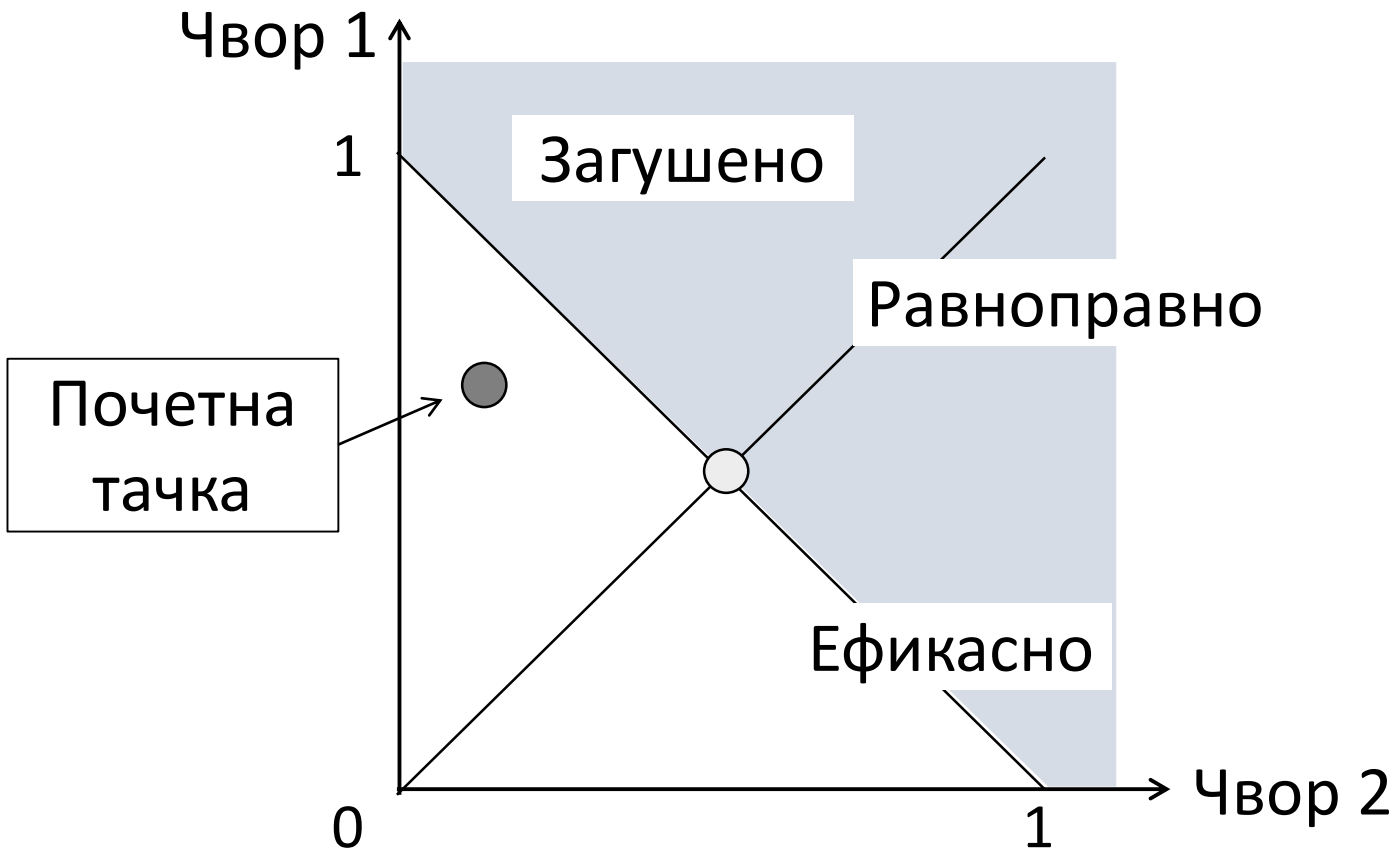
AIMD (4)

- AI и MD померају тачку алокације на следећи начин:

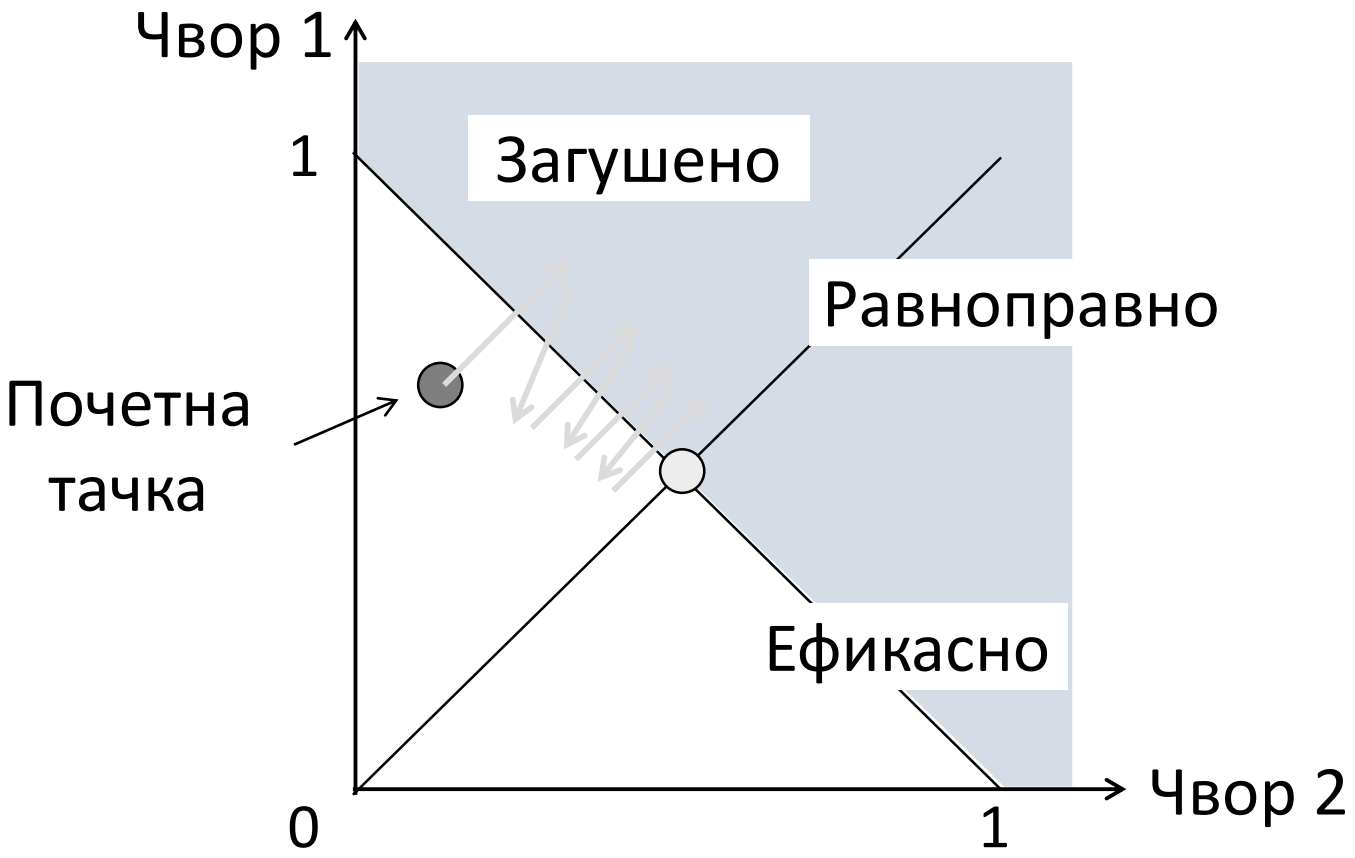


AIMD (5)

- Адитивно увећање помера под углом од 45 степени



AIMD (6)



- Умножено смањење враћа релативно према тренутном протоку
- Већи проток
  →
  већа дужина пројекције

AIMD карактеристике

- Конвергира ка оптималној тачки алокације
  - Пресеку правих ефикасности и равноправности
  - Ради и у вишедимензионом сценарију
- Остали приступи не раде посао, пробајте: MIAD, MIMD, AIAD
- Захтева само бинарни одговор/сигнал од мреже да би радило

Бинарни одговори мреже

- Неколико могућих типова сигнала је у употреби
  - TCP користи први

Сигнал	**Пример протокола**	**+/-**
Губитак пакета	TCP NewReno Cubic TCP (Linux)	Поуздано детектује Касно чује
Кашњење пакета	Compound TCP (Windows)	Рано чује Прави претпоставку
Сигнал рутера	TCP са експлицитним сигналом загушења	Рано чује Захтева подршку рутера