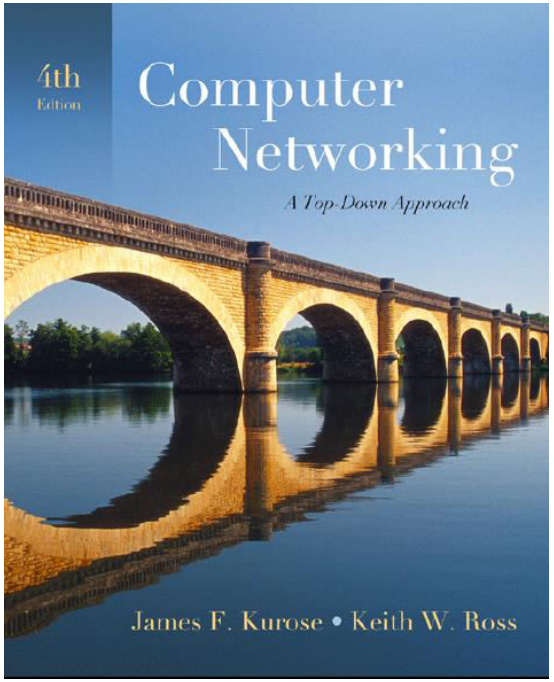
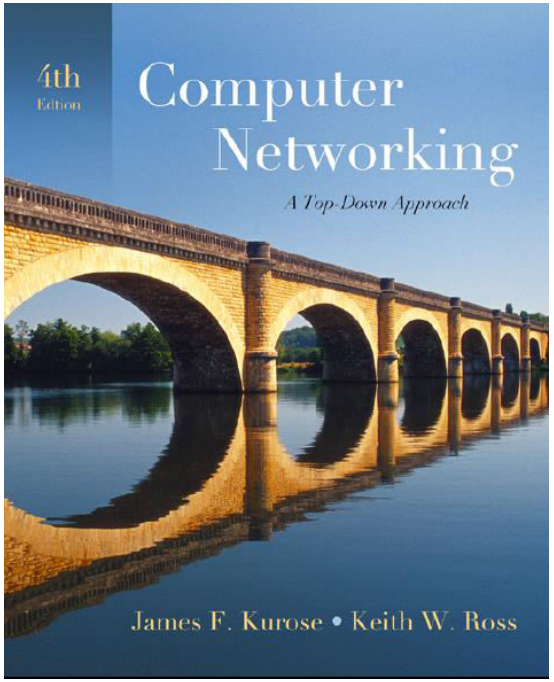
KOLOKVIJUM-1

MREŽNO RAČUNARSTVO
3- GODINA



Computer Networking: A Top Down Approach, 4th edition.
Jim Kurose, Keith Ross
Addison-Wesley, July 2007.

Chapter 1 Introduction

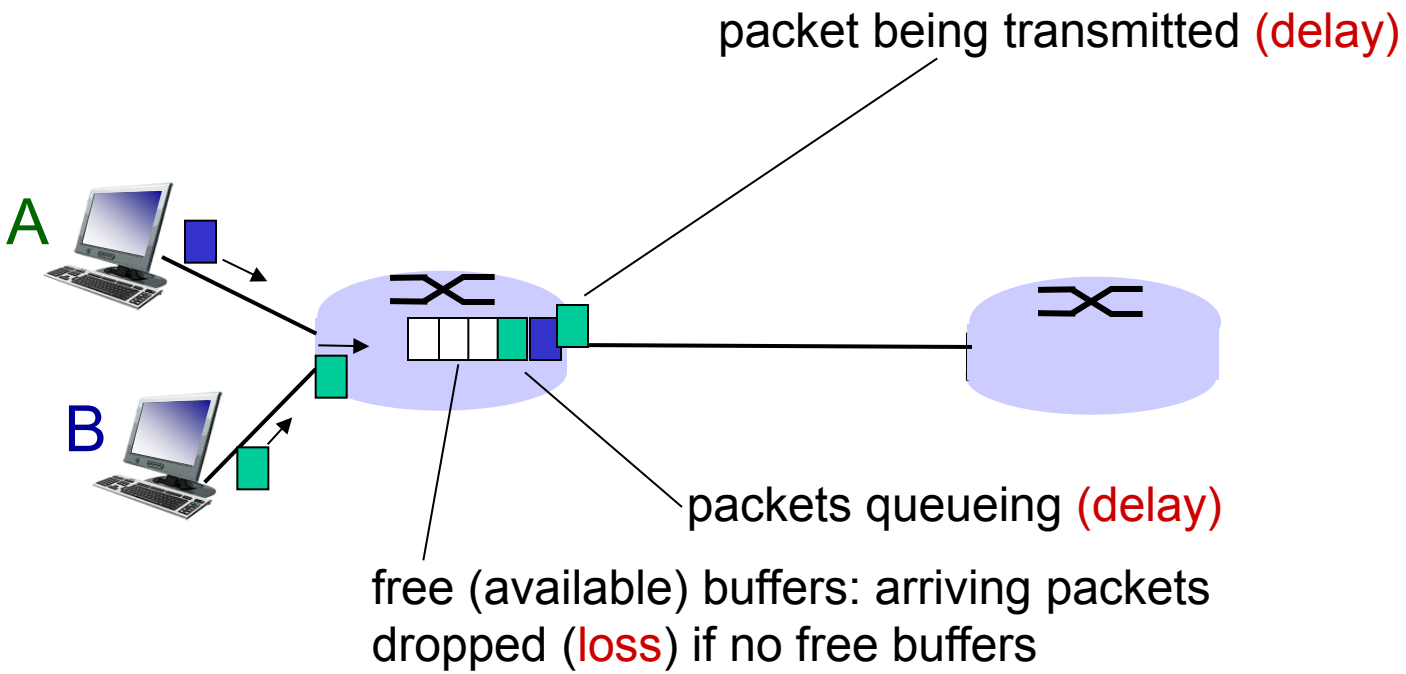


Computer Networking: A Top Down Approach, 4th edition.
Jim Kurose, Keith Ross
Addison-Wesley, July 2007.

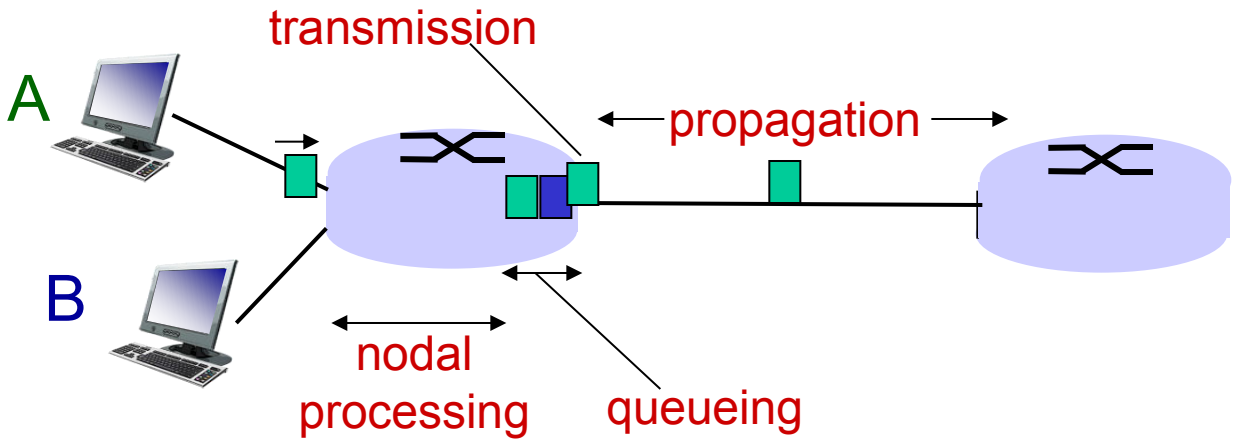
How do loss and delay occur?

packets *queue* in router buffers

- packet arrival rate to link (temporarily) exceeds output link capacity
- packets queue, wait for turn



Four sources of packet delay



$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

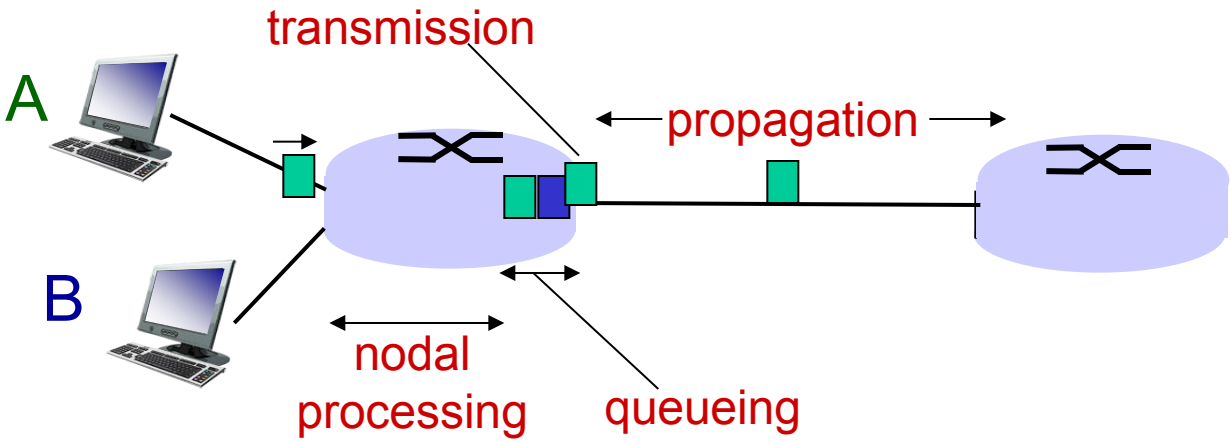
d_{proc}: nodal processing

- check bit errors
- determine output link
- typically < msec

d_{queue}: queueing delay

- time waiting at output link for transmission
- depends on congestion level of router

Four sources of packet delay



$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

d_{trans}: transmission delay:

- L: packet length (bits)
- R: link *bandwidth (bps)*
- $d_{trans} = L/R$

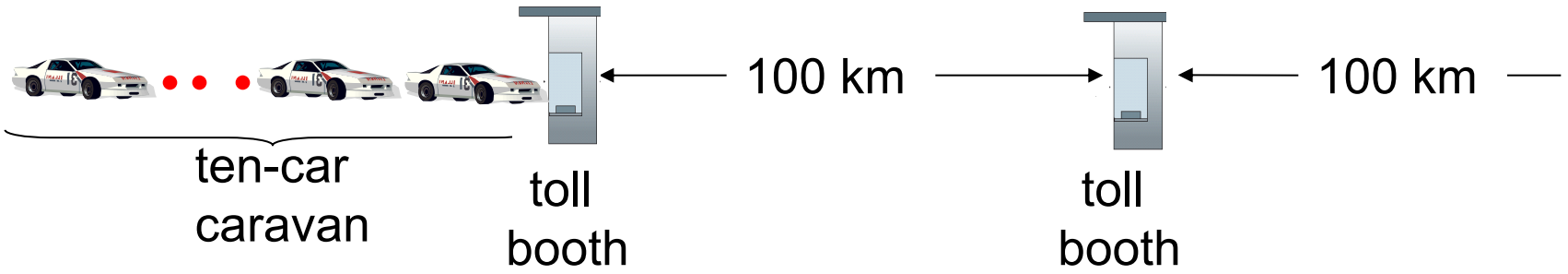
d_{prop}: propagation delay:

- d: length of physical link
- s: propagation speed in medium (~2×10^8 m/sec)
- $d_{prop} = d/s$

d_{trans} and d_{prop} *very* different

* Check out the Java applet for an interactive animation on trans vs. prop delay

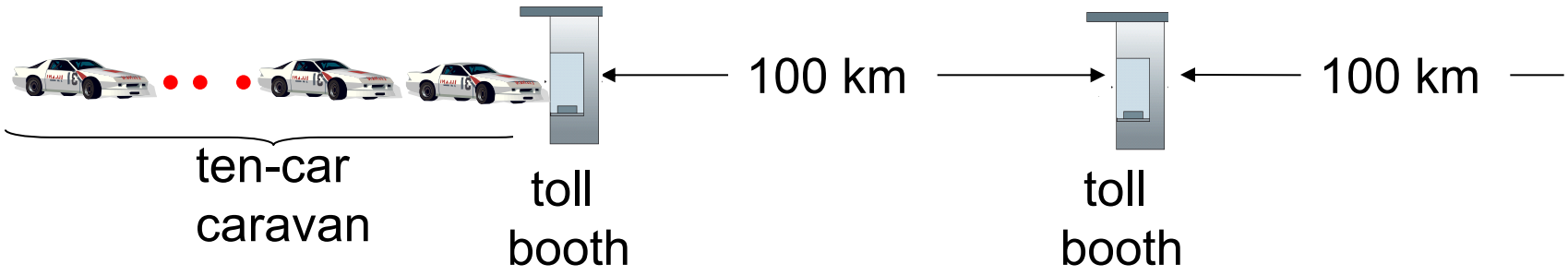
Caravan analogy



- cars "propagate" at 100 km/hr
- toll booth takes 12 sec to service car (bit transmission time)
- car~bit; caravan ~ packet
- *Q:* How long until caravan is lined up before 2nd toll booth?

- time to "push" entire caravan through toll booth onto highway = 12*10 = 120 sec
- time for last car to propagate from 1st to 2nd toll both: 100km/(100km/hr)= 1 hr
- *A:* 62 minutes

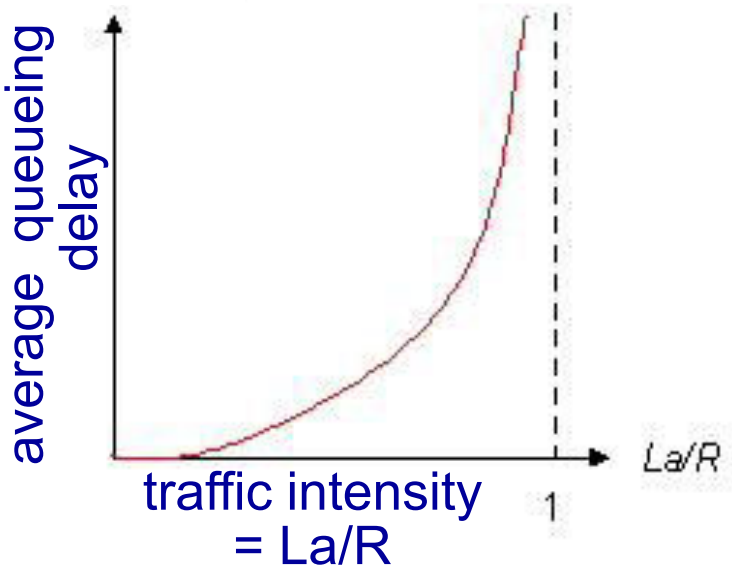
Caravan analogy (more)



- suppose cars now "propagate" at 1000 km/hr
- and suppose toll booth now takes one min to service a car
- *Q:* Will cars arrive to 2nd booth before all cars serviced at first booth?
  - *A: Yes!* after 7 min, 1st car arrives at second booth; three cars still at 1st booth.

Queueing delay (revisited)

- *R:* link bandwidth (bps)
- *L:* packet length (bits)
- a: average packet arrival rate



- *La/R* ~ 0: avg. queueing delay small
- *La/R* -> 1: avg. queueing delay large
- *La/R* > 1: more "work" arriving than can be serviced, average delay infinite!

La/R ~ 0

La/R -> 1

* Check out the Java applet for an interactive animation on queuing and loss

Zadaci

1. Zamislite slanje paketa od F bitova putanjom od Q linkova brzine prenosa R [b/s]. Mreža je samo blago opterećena tako da nema kašnjenja usled stajanja u redu. Kašnjenje usled propagacije je bezbačajno.
   a. Prtpostavimo da je u pitanju mreža sa komitiranjem paketa i virtuelnim kolima. Neka trajanje pripreme virtuelnih kolima. Neka trajanje pripreme virtuelnih kola bude ts sekundi. Pretpostavimo i to da izvorni slojevi dodaju ukupno h bitov zaglavlja svakom paketu. Koliko je vremena potrebno za slanje ove datoteke od izvora do odredišta?
   b. Pretpostavimo da je u pitanju mreža sa datagramima i komutiranjem paketa, da se koristi usluga bez konenkcije, kao i to da svaki paket ima zaglavlje dužie 2h. Koliko je vremena sada potrebno za slanje paketa?
   c. Konačno, pretpostavimo da je reč o mreži sa komutiranjem vodova i da brzina prenosa između izvora i odredišta iznosi R [b/s]. Ukoliko je ts trajanje priprime veze, a h broj bitova zaglavlje koje se dodaje čitavom paketu. Koliko je vremena potrebno za slanje ovog paketa?

Rešenje:

a) vreme neophodno za transfer paketa kroz jedan link je:

$(L+h)/R$.

Vreme neophodno za prenos kroz Q linkova je:

$Q(L+h)/R$.

Možemo napisati izraz koji sumira tj. obuhvata date oblike kašnjenja: $t_s + Q(L+h)/R$.

b) $Q(L+2h)/R$

c) Nema store-and-forward kašnjenja prilikom transfera. Možemo napisati sledeći izraz za ukupno kašnjenje.

$$t_s + (h+L)/R.$$

2. Dva centralna aspekta umrežavanja su kašnjenje usled propagacije i kšnjenje usled prenosa. Zamislite dva računara A i B koji su povezani jednim linkom čija je brzina prenosa R b/s. Pretpostavimo da su ova dva računara međusobno udaljena m metara i da je brzina propagacije linka s metara u sekundi. Računar A treba da pošalje paket dužine L bitova računaru B.
   a. Izrazite kašnjenje ulsed propagacije dprop u funkciji m i s.
   b. Odredite trajnje prenosa paketa dprenos u odnosu na L i R.
   c. Zanemarujući kašnjenje ulsed obrade i stajanja u redu, napišite izraz za kašnjenje od jednog do drugog kraja.
   d. Oderdite udaljenost m tako da dprop bude jednako sa dprenos.

Rešenje:

a) $d_{prop} = m / s$; [s].

b) $d_{trans} = L / R$; [s].

c) $d_{end-to-end} = (m / s + L / R)$;[s].

g) $m = \frac{L}{R} S$;[m].

3. Na primer možemo posmatrati link kapaciteta 1 Mb/s. Kada su korisnici aktivni, oni stvaraju saobraćaj od 100 kb/s, , ali verovatnoća njihove aktivnosti je samo p=0,1. Pretpostavimo sada da se link od 1Mb/s zameni linkom od 1Gb/s.
   a. Koliko je N, tj. maksimalan broj korisnika koje komutiranje vodova može da podrži istovremeno?
   b. Pretpostavimo da je u pitanju mreža sa komutiranjem paketa od M koirisnika. Napišite formulu (u funkciji p, M, N) za izračunavanje verovatnoće da više od N korisnika istovremeno šalje svoje podatke.

Rešenje:

a) pretpostavimo da N potencijalni korisnik ima mogućnost istovremene realizacije transfera pojedinačnih, kapaciteta od 100Kb/s kroz link kapaciteta od 1Gb/s. Približno rešenje je oko 10000 korisnika.

b) $$\sum_{n=N+1}^{M} \binom{M}{n} p^n (1-p)^{M-n}$$

4. Uzmimo u obzir stajanje u redu u privremenoj memoriji rutera (koji prethodi izlaznom linku). Pretpostavimo da je dužina svih paketa L bitova, brzina prenosa je R [b/s] i da N paketa istovremeno stiže u privremenu memoriju svakih LN/R sekundi. Odredite prosečno kašnjenje paketa usled stajanja u redu. (pretpostavimo, kašnjenje usled stajanja u redu prvog paketa je nula, drugog L/R, trećeg 2L/R. N-ti paket je već prenet kada u ruter dospe druga serija paketa.)

Rešenje:

Neophodno je LN/R sekundi za prenos N paketa. Pretpostavka je, da kada grupa od N paketa stigne do rutera privremena memorija tj. bafer je prazan .

Kašnjenje usled stajanja u redu prvog pakate od N paketa je nula. Kašnjenje u redu drugog paketa je $1xL/R$ sekundi, trećeg paketa 2xL/R sekundi itd. Na osnovu prethodnog napisanog, sledi da je kašnenje n -og paketa: $(n-1)L/R$ sekundi.

Na osnovu prethodne analize možemo napisati da je prosečno kašnjenje paketa usled stanja u redu:

$$\frac{1}{N}\sum_{n=1}^{N}(n-1)L/R = \frac{L}{R}\frac{1}{N}\sum_{n=0}^{N-1}n = \frac{L}{R}\frac{1}{N}\frac{(N-1)N}{2} = \frac{L}{R}\frac{(N-1)}{2}\ .$$

5. Pretpostavimo da računar A do računara B treba poslati veliku datoteku od F bitova. Između računara A i B postoje dva linka (i jedan komutator) koji nisu zagušeni (nema stajanja u redu). Računar A segmentira datoteku na segmente od po S bitova i svakom segmentu dodaje zaglavlje od 40 bitova, čime se dobijaju paketi dužine L=40+S bitova. Brzina prenosa svakog linka iznosi R [b/s]. Pronađite vrednost parametara S koja minimizira kašnjenje prilikom premeštanja datoteke sa računara A na računar B. Zanemarite kašnjenje usled propagacije.

Rešenje:

Vreme za koje prvi paket stigne na odredište je: $\frac{S+40}{R}\times 2$ [s]. Nakon prvog paketa ostali paketi bivaju isporučeni na odredište svakih $\frac{S+40}{R}$ sekundi. Vreme koje je neophodno za isporučivanje celog fajla je definisano sledećom funkcijom:

$$\mathrm{d}_{\text{ukupno}} = \frac{S+40}{R}\times 2 + (\frac{F}{S}-1)\times(\frac{S+40}{R}) = \frac{S+40}{R}\times(\frac{F}{S}+1)$$

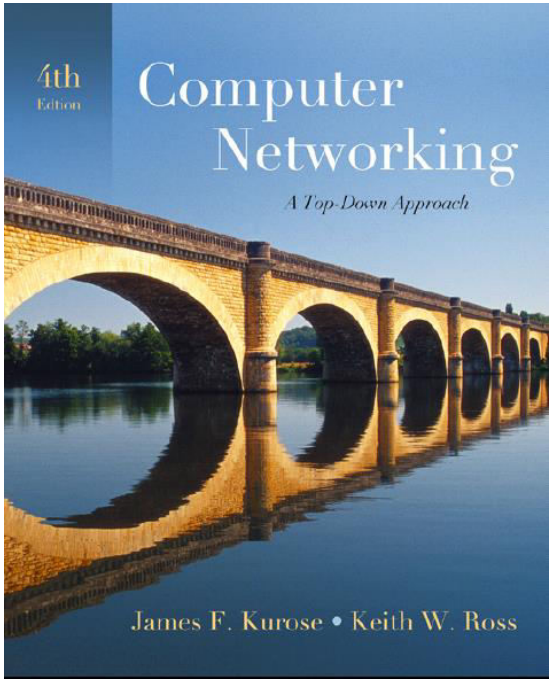
Da bi definisali minimalnu vrednost kašnjenja, nephodno naći parcijalni izvod po promenljivoj S. Dati dobijeni brojevni izraz je neophodno izjednačiti sa 0. Zatim možemo definisati promenljivu S.

$$\frac{\partial(\mathrm{d}_{\text{ukupno}})}{\partial S} = 0 \Rightarrow \frac{F}{R}(\frac{1}{S} - \frac{40+S}{S^2}) + \frac{1}{R} = 0 \Rightarrow S = \sqrt{40F}$$

Chapter 2
Application Layer

RAČUNARSKE MREŽE I WEB PROGRAMIRANJE
IV- GODINA

prof. dr Zlatko Langović

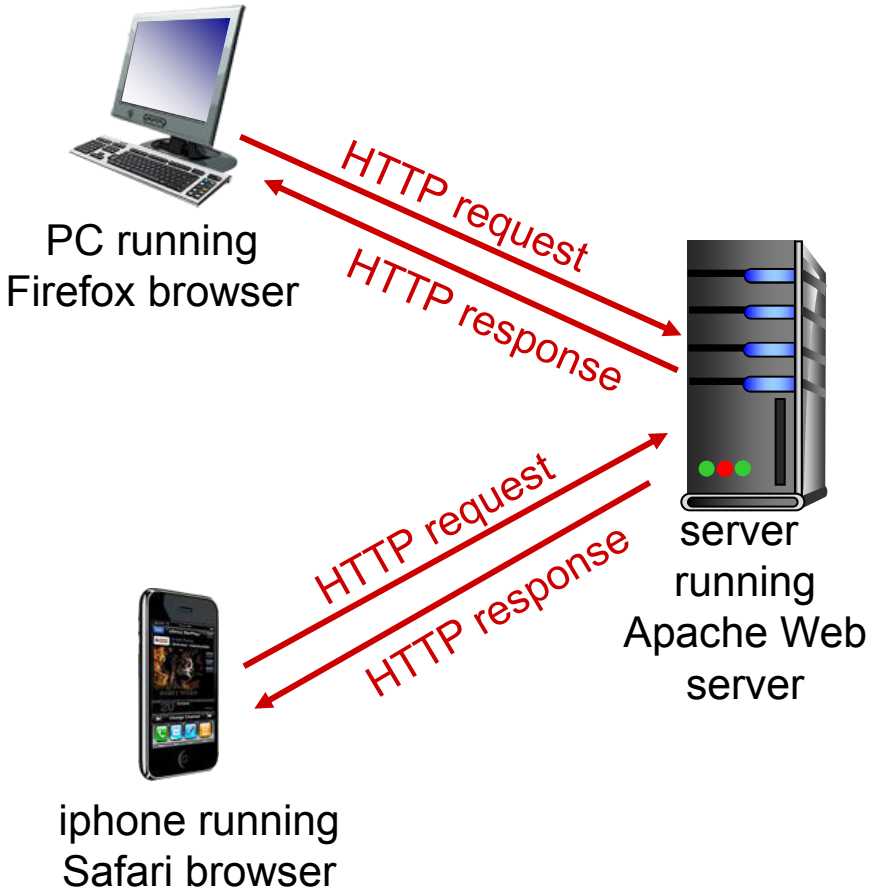


Computer Networking: A Top Down Approach, 4th edition.
Jim Kurose, Keith Ross
Addison-Wesley, July 2007.

HTTP overview

HTTP: hypertext transfer protocol

- Web's application layer protocol
- client/server model
  - *client:* browser that requests, receives, (using HTTP protocol) and "displays" Web objects
  - *server:* Web server sends (using HTTP protocol) objects in response to requests



HTTP overview (continued)

uses TCP:

- client initiates TCP connection (creates socket) to server, port 80
- server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed

HTTP is "stateless"

- server maintains no information about past client requests

aside

protocols that maintain "state" are complex!

- past history (state) must be maintained
- if server/client crashes, their views of "state" may be inconsistent, must be reconciled

HTTP connections

non-persistent HTTP

- at most one object sent over TCP connection
  - connection then closed
- downloading multiple objects required multiple connections

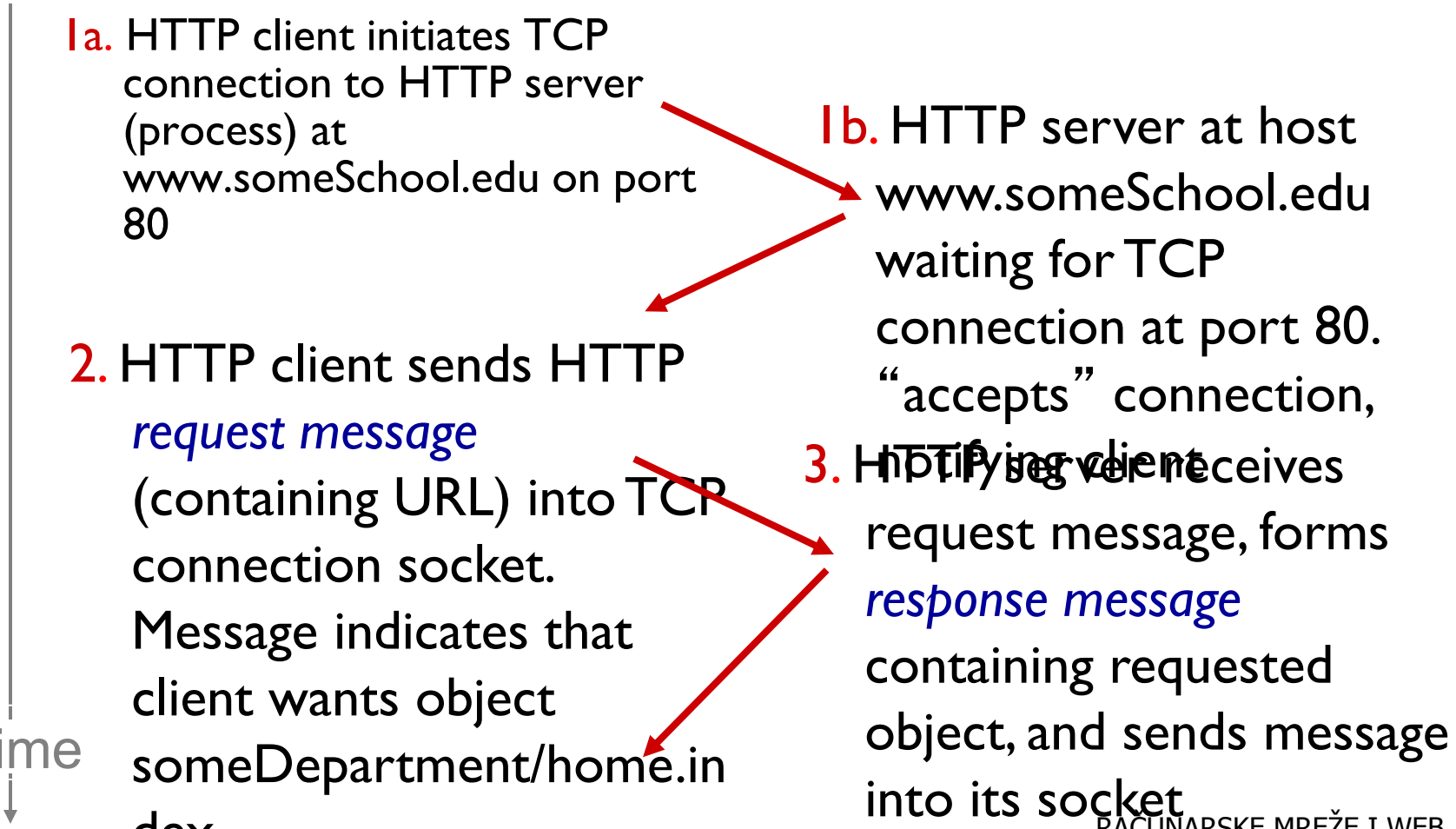
persistent HTTP

- multiple objects can be sent over single TCP connection between client, server

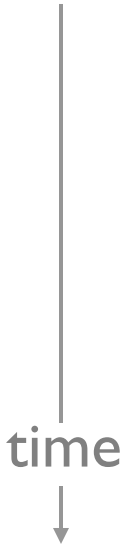
Non-persistent HTTP

suppose user enters URL:
`www.someSchool.edu/someDepartment/home.index`

(contains text, references to 10 jpeg images)



Non-persistent HTTP (cont.)



4. HTTP server closes TCP connection.

5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects

6. Steps 1-5 repeated for each of 10 jpeg objects

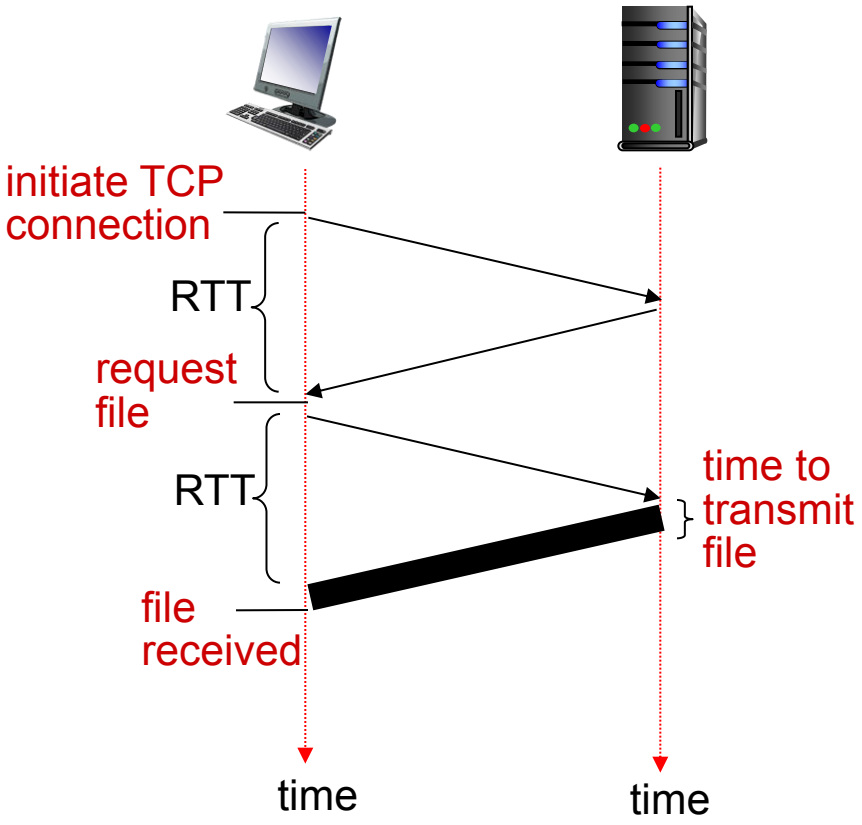
Non-persistent HTTP: response time

RTT (definition): time for a small packet to travel from client to server and back

HTTP response time:

- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return
- file transmission time
- non-persistent HTTP response time =

  2RTT+ file transmission time



Persistent HTTP

non-persistent HTTP issues:

- requires 2 RTTs per object
- OS overhead for *each* TCP connection
- browsers often open parallel TCP connections to fetch referenced objects

persistent HTTP:

- server leaves connection open after sending response
- subsequent HTTP messages between same client/server sent over open connection
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects

Zadaci

6. Pretpostavimo da ste u svom čitaču izborom hiperveze otvorili neku veb stranu čija IP adresa nije keširana u vašem lokalnom računaru, tako da je za njeno pribavljanje neophodno pretraživanje DNS baze podataka. Takođe ćemo pretpostaviti i to da je dobijanju IP adrese prethodila poseta n DNS servera (RTT1, RTT2, ..., RTTn). Zatim ova strana je povezana sa linkom koji se odnosi na samo jedan objekat-malo HTML teksta. Sa RTT0 označićemo RTT vreme između lokalnog računara i servera na kome se ovaj objekat nalazi. Ukoliko zanemarimo trajanje prenosa objekta, koliko vremena protekne od trenutka kada korisnik izabere hipervezu do trenutka kada klijent primi tranženi objekat?

Rešenje:

Ukupno vreme neophodno za dobijanje IP adrese je:

$$RTT_1 + RTT_2 + \cdots + RTT_n = \sum_{i=1}^{n} RTTi \ .$$

Nakon dobijanja IP adrese najbližeg web servera sledi proces transfera objekta koji sadrži malo HTML teksta. Ako je RTT_0 vreme koje je neophodno za transfer objekta od klijetskog do serverskog računara onda sledi da je.ukupno vreme:

$$2RTT_o + RTT_1 + RTT_2 + \cdots + RTT_n = 2RTT_o + \sum_{i=1}^{n} RTTi$$

7. Posmatrajući prethodni problem, pretposavimo da HTML datoteka referencira tri veoma mala objekta na istom serveru. Ako zanemarimo trajanje prenosa, kojliko je vremena potrebno kod:
   a. Nepostojanih HTTP veza bez paralelnih TCP konekcija,
   b. Nepostojanih HTTP veza sa paralelnim TCP konekcijama i
   c. Postojanih HTTP veza sa cevovodnom obradom?

Rešenje:

a)

$$RTT_1 + \cdots + RTT_n + 2RTT_o + 3 \cdot 2RTT_o$$

$$= 8RTT_o + RTT_1 + \cdots + RTT_n = 8RTT_o + \sum_{i=1}^{n} RTTi$$

.

b)

$$RTT_1 + \cdots + RTT_n + 2RTT_o + 2RTT_o$$

$$= 4RTT_o + RTT_1 + \cdots + RTT_n = 4RTT_o + \sum_{i=1}^{n} RTTi$$

.

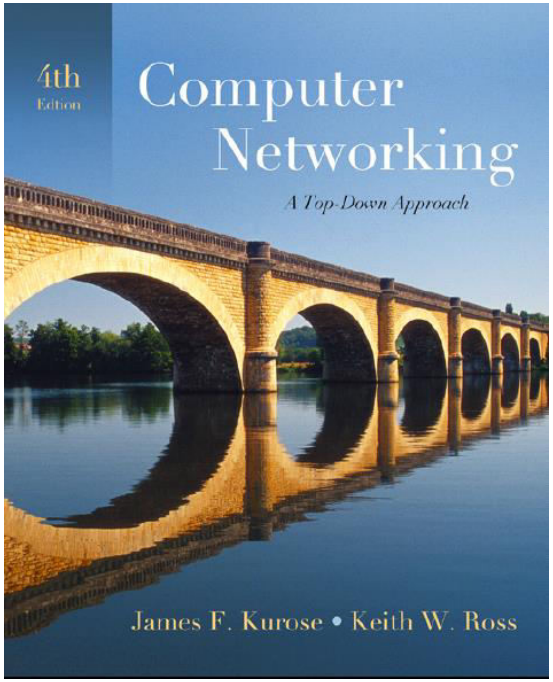
c)

$$RTT_1 + \cdots + RTT_n + 2RTT_o + RTT_o$$

$$= 3RTT_o + RTT_1 + \cdots + RTT_n = 3RTT_o + \sum_{i=1}^{n} RTTi$$

.

Chapter 3
Transport Layer



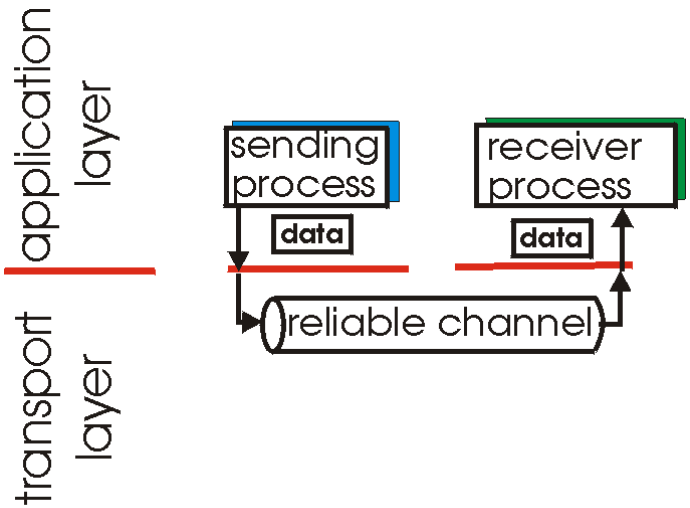
RAČUNARSKE MREŽE I WEB PROGRAMIRANJE
IV- GODINA

prof. dr Zlatko Langović

Computer Networking: A Top Down Approach
4th edition.
Jim Kurose, Keith Ross
Addison-Wesley, July 2007.

Principles of Reliable data transfer

- important in app., transport, link layers
- top-10 list of important networking topics!

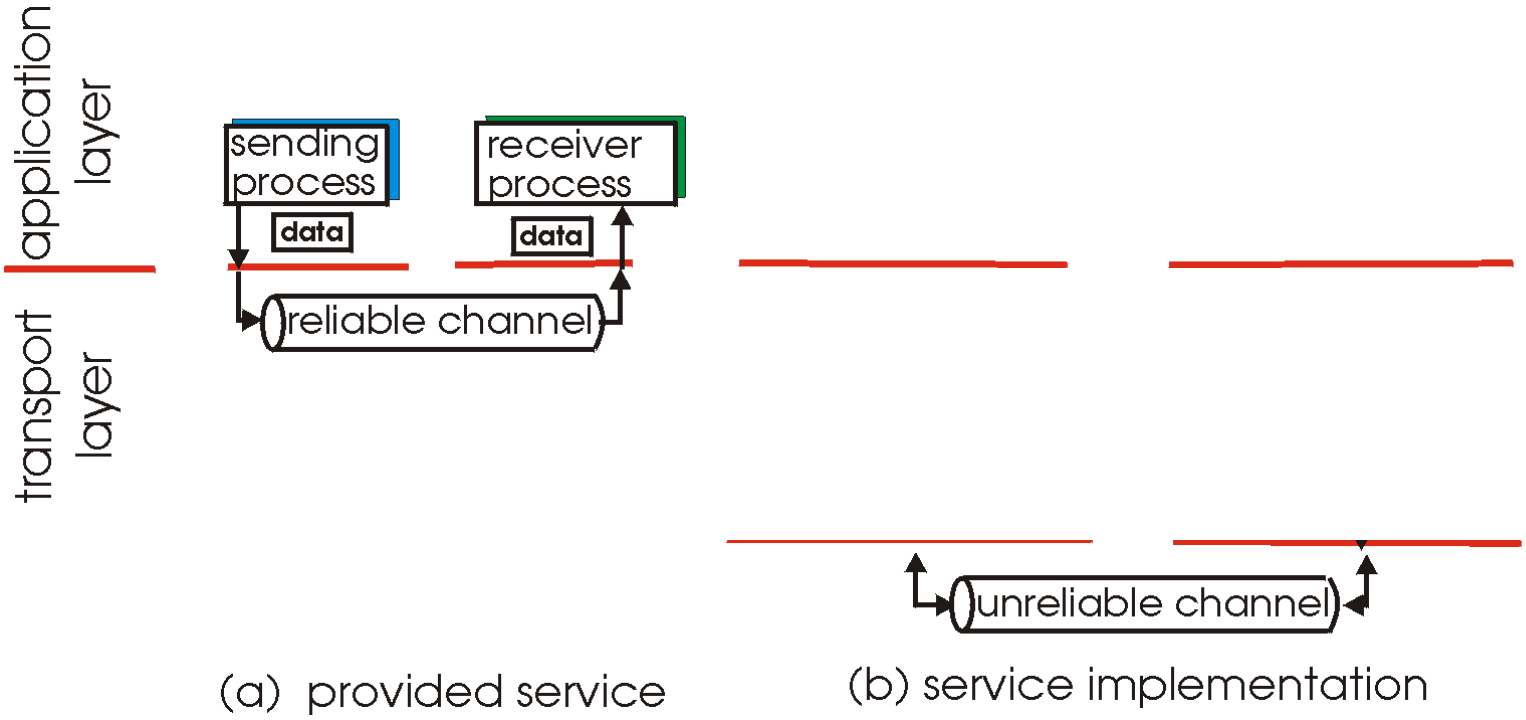


(a) provided service

- characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

Principles of Reliable data transfer

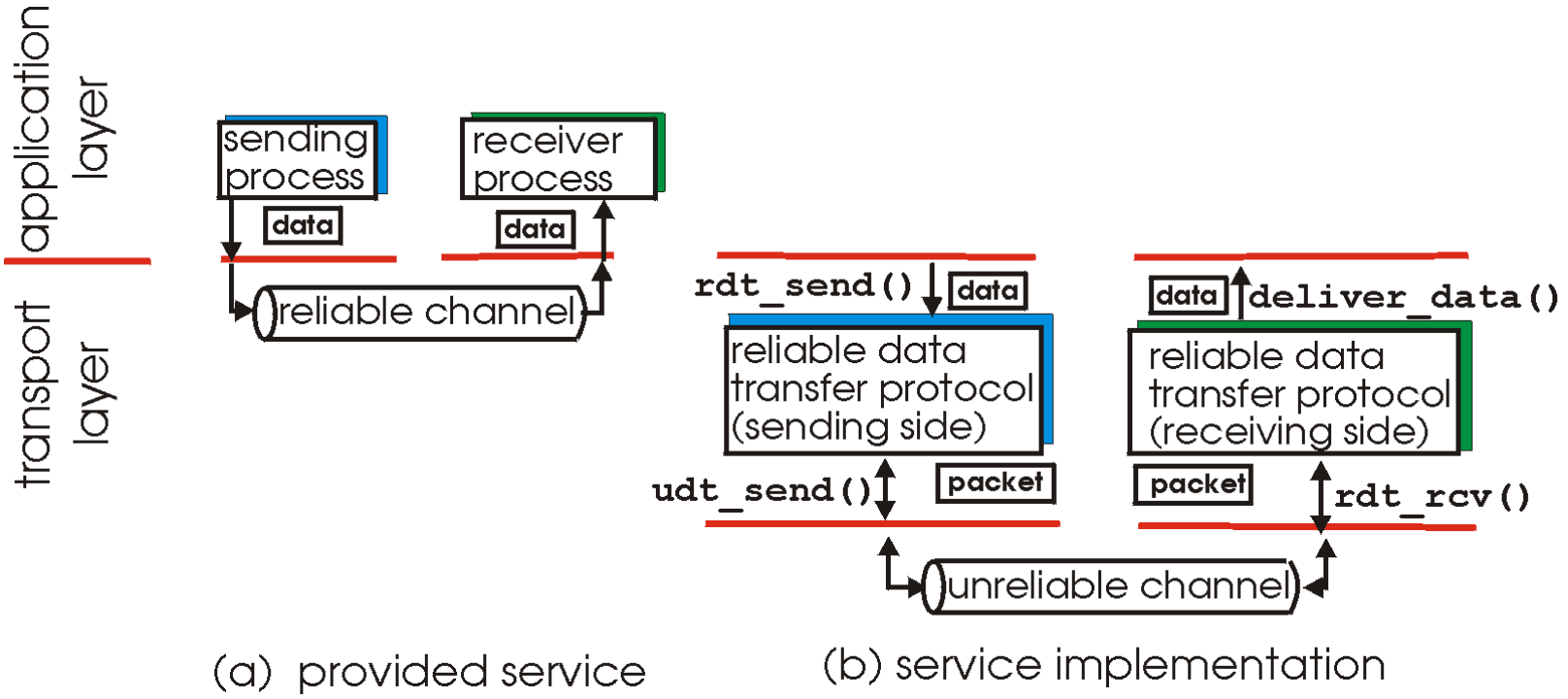
- important in app., transport, link layers
- top-10 list of important networking topics!



- characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

Principles of Reliable data transfer

- important in app., transport, link layers
- top-10 list of important networking topics!

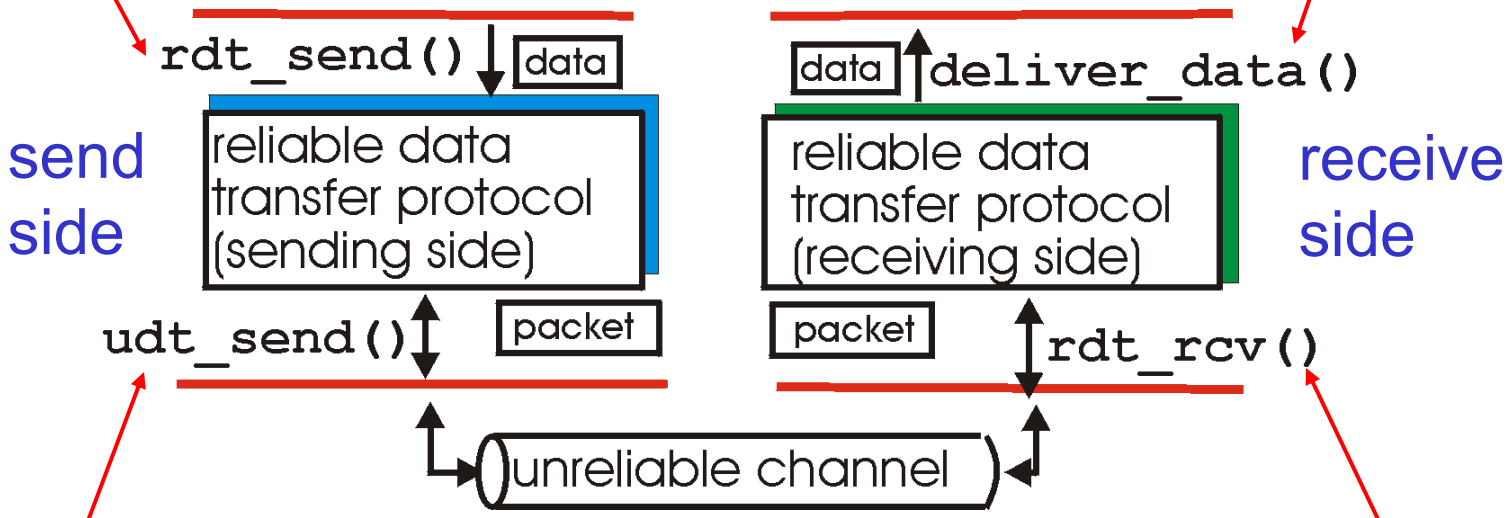


- characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

Reliable data transfer: getting started

`rdt_send()`: called from above, (e.g., by app.). Passed data to deliver to receiver upper layer

`deliver_data()`: called by **`rdt`** to deliver data to upper



`udt_send()`: called by rdt, to transfer packet over unreliable channel to receiver

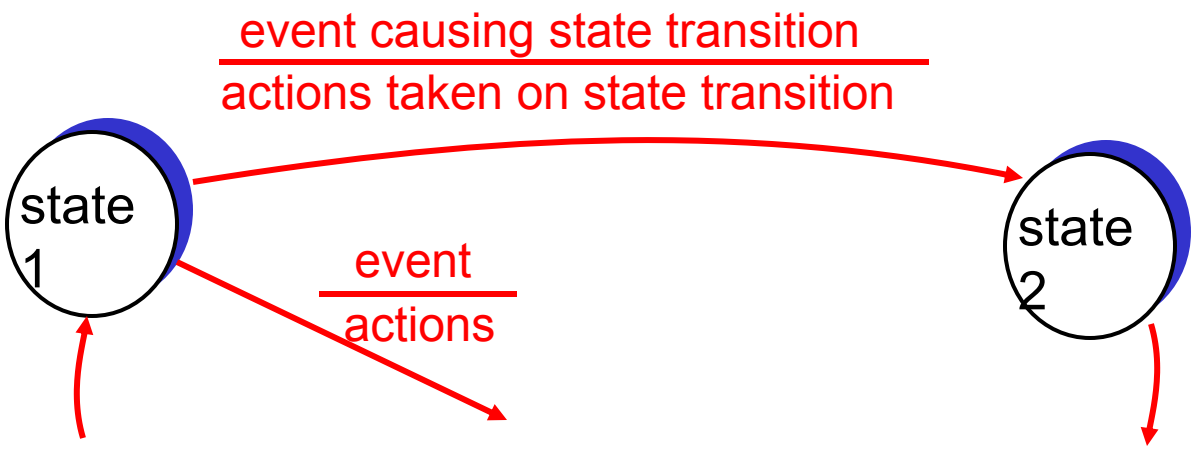
`rdt_rcv()`: called when packet arrives on rcv-side of channel

Reliable data transfer: getting started

We'll:

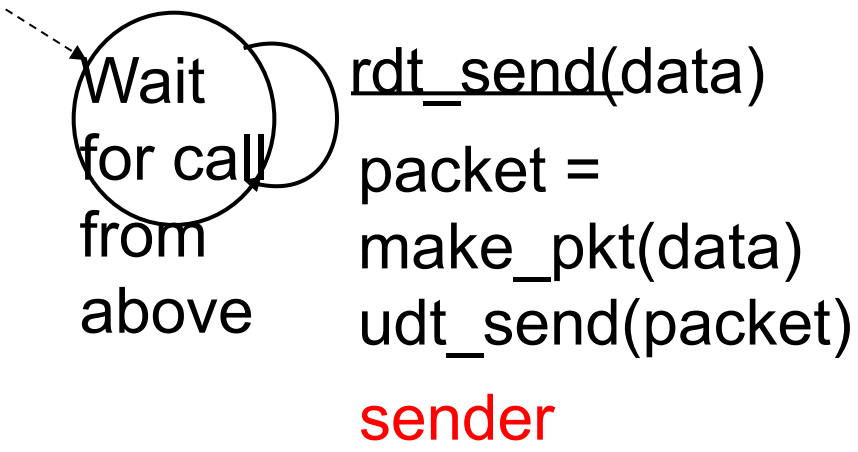
- incrementally develop sender, receiver sides of reliable data transfer protocol (rdt)
- consider only unidirectional data transfer
  - but control info will flow on both directions!
- use finite state machines (FSM) to specify sender, receiver

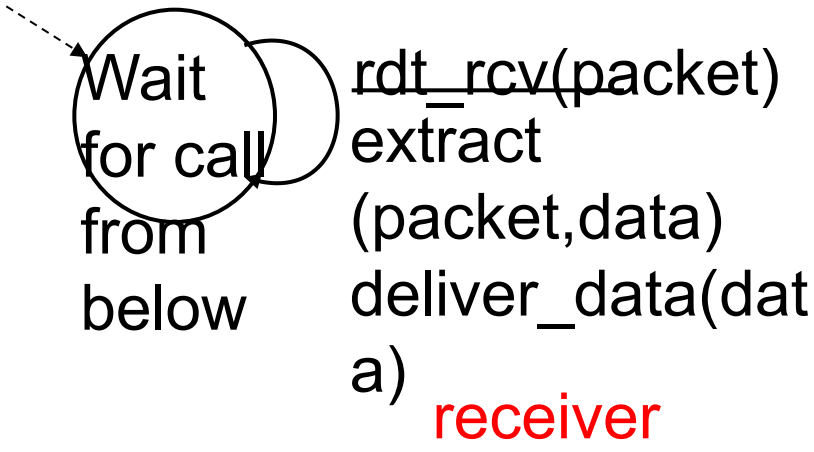
state: when in this "state" next state uniquely determined by next event



Rdt1.0: reliable transfer over a reliable channel

- underlying channel perfectly reliable
  - no bit errors
  - no loss of packets
- separate FSMs for sender, receiver:
  - sender sends data into underlying channel
  - receiver read data from underlying channel

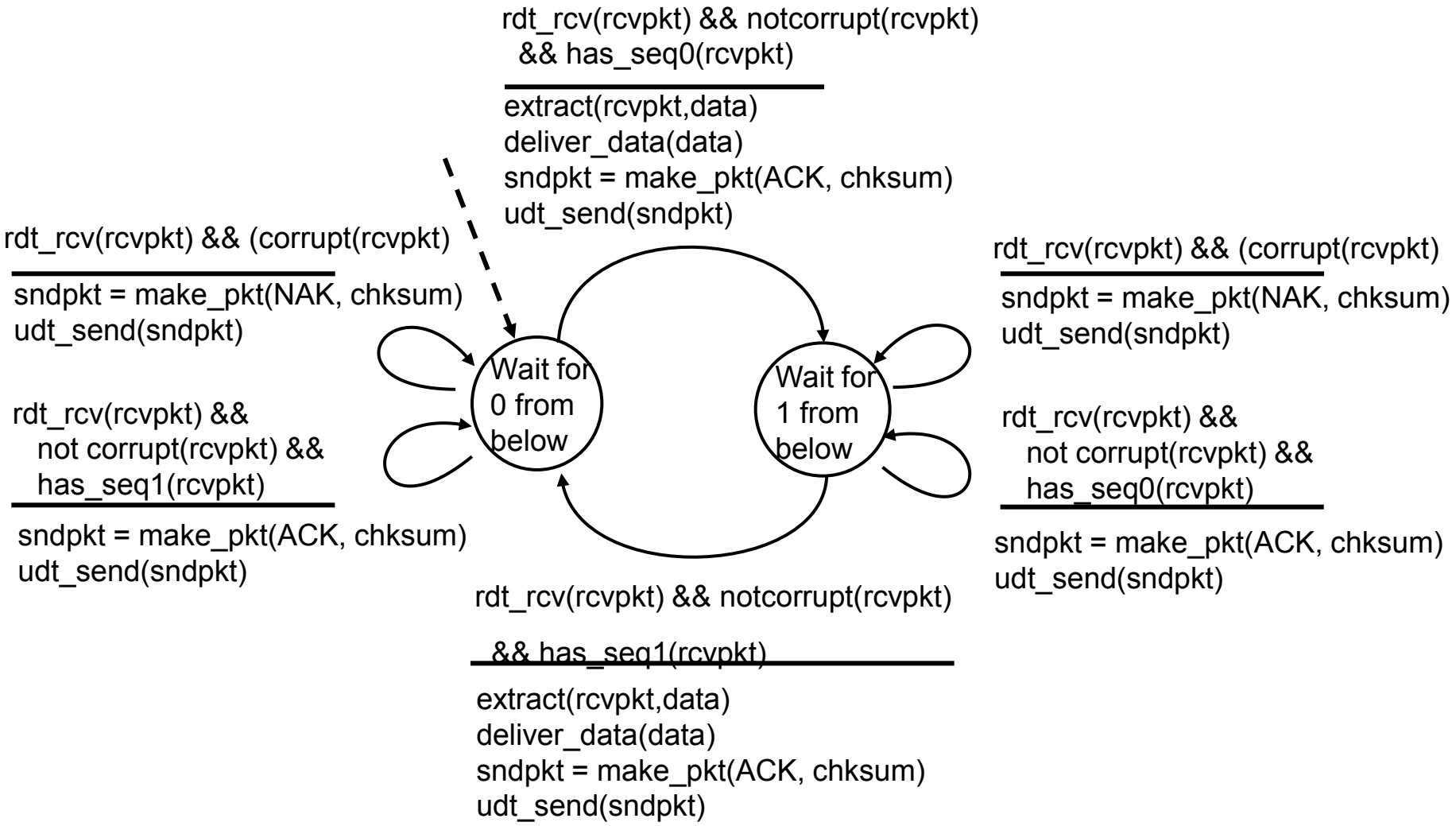




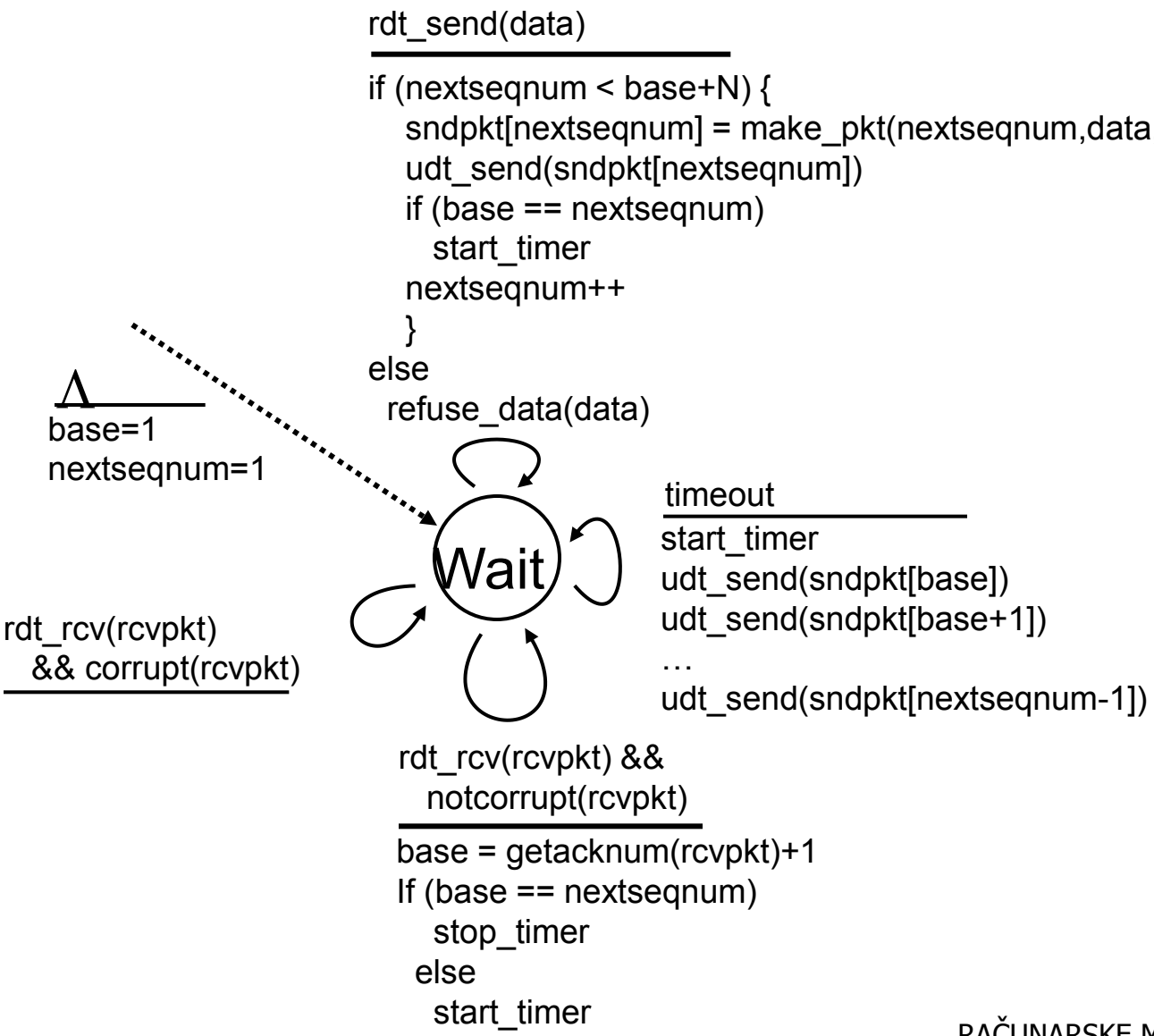
Rdt2.0: channel with bit errors

- underlying channel may flip bits in packet
  - checksum to detect bit errors
- *the* question: how to recover from errors:
  - *acknowledgements (ACKs):* receiver explicitly tells sender that pkt received OK
  - *negative acknowledgements (NAKs):* receiver explicitly tells sender that pkt had errors
  - sender retransmits pkt on receipt of NAK
- new mechanisms in **`rdt2.0`** (beyond **`rdt1.0`**):
  - error detection
  - receiver feedback: control msgs (ACK,NAK) rcvr->sender

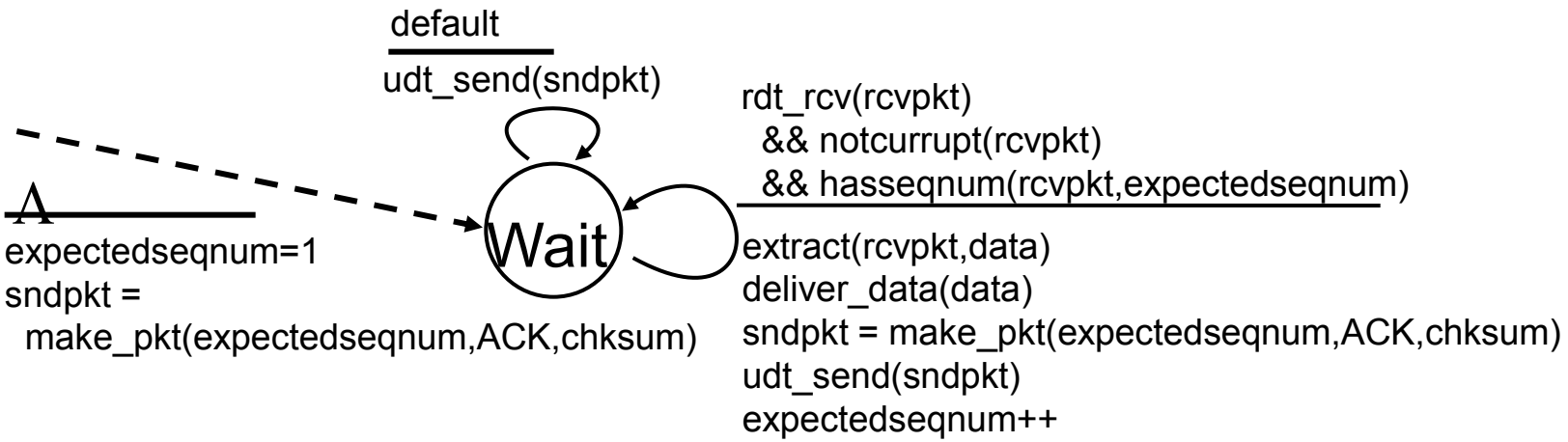
rdt2.1: receiver, handles garbled ACK/NAKs



GBN: sender extended FSM



GBN: receiver extended FSM



ACK-only: always send ACK for correctly-received pkt with highest *in-order* seq #

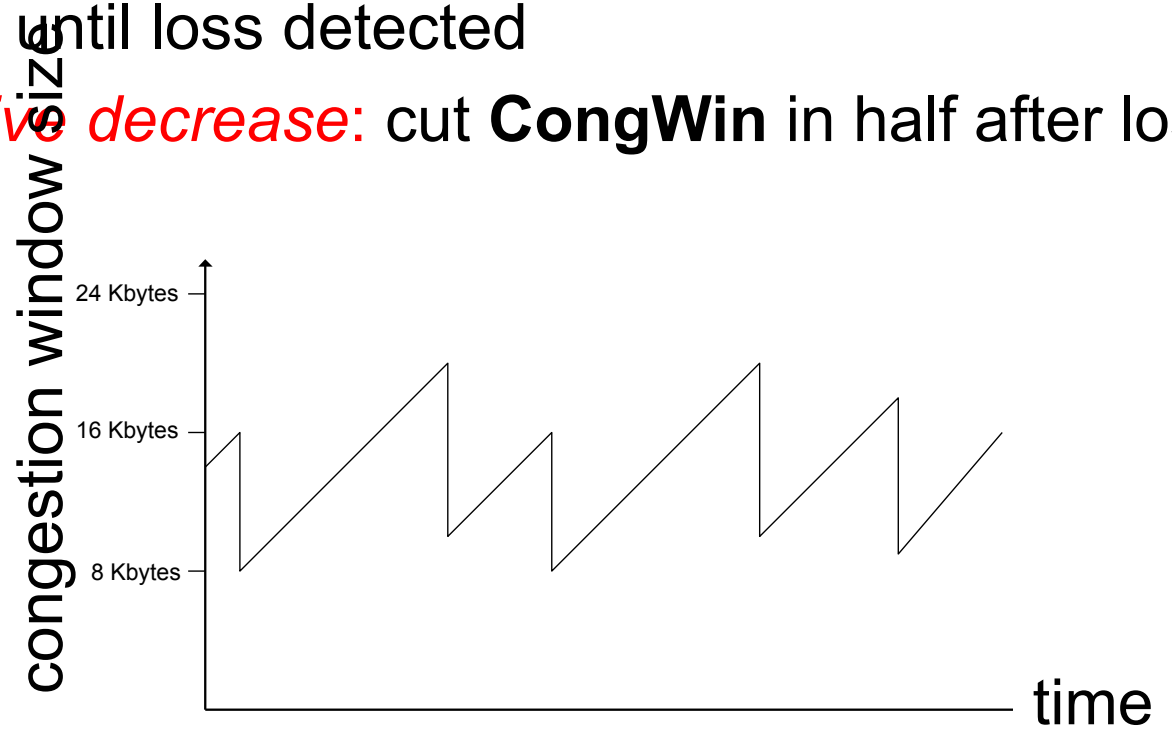
- may generate duplicate ACKs
- need only remember **`expectedseqnum`**

- out-of-order pkt:
  - discard (don't buffer) -> no receiver buffering!
  - Re-ACK pkt with highest in-order seq #

TCP congestion control: additive increase, multiplicative decrease

- *Approach:* increase transmission rate (window size), probing for usable bandwidth, until loss occurs
  - *additive increase:* increase **CongWin** by 1 MSS every RTT until loss detected
  - *multiplicative decrease*: cut **CongWin** in half after loss

Saw tooth behavior: probing for bandwidth



TCP Congestion Control: details

- sender limits transmission:

  LastByteSent-LastByteAcked

  ≤ CongWin

- Roughly,

$$\text{rate} = \frac{\text{CongWin}}{\text{RTT}} \text{ Bytes/sec}$$

- **CongWin** is dynamic, function of perceived network congestion

How does sender perceive congestion?

- loss event = timeout *or* 3 duplicate acks
- TCP sender reduces rate (**CongWin**) after loss event

three mechanisms:

- AIMD
- slow start
- conservative after timeout events

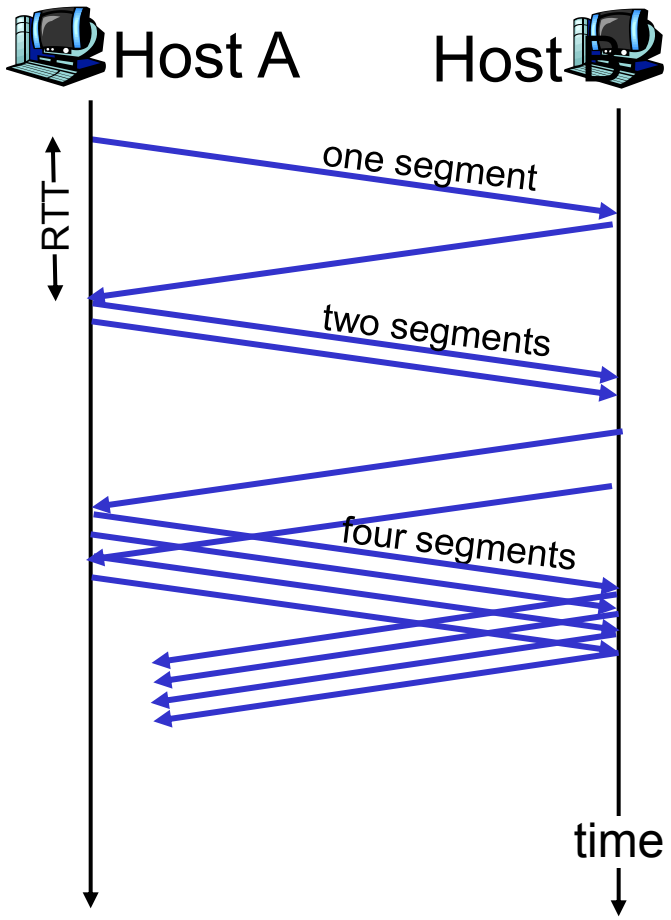
TCP Slow Start

- When connection begins, **`CongWin`** = 1 MSS
  - Example: MSS = 500 bytes & RTT = 200 msec
  - initial rate = 20 kbps
- available bandwidth may be >> MSS/RTT
  - desirable to quickly ramp up to respectable rate

- When connection begins, increase rate exponentially fast until first loss event

TCP Slow Start (more)

- When connection begins, increase rate exponentially until first loss event:
  - double **CongWin** every RTT
  - done by incrementing **CongWin** for every ACK received
- Summary: initial rate is slow but ramps up exponentially fast



Refinement: inferring loss

- After 3 dup ACKs:
  - **CongWin** is cut in half
  - window then grows linearly
- But after timeout event:
  - **CongWin** instead set to 1 MSS;
  - window then grows exponentially
  - to a threshold, then grows linearly

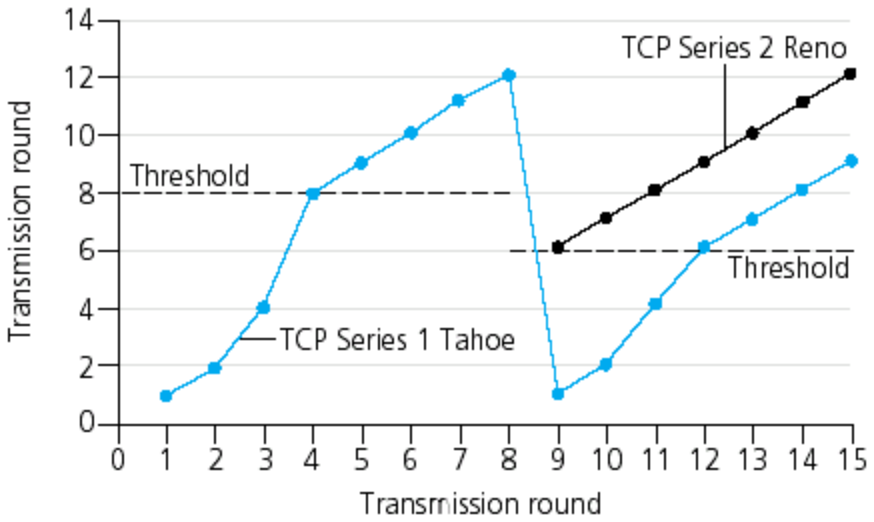
Philosophy:

- 3 dup ACKs indicates network capable of delivering some segments
- timeout indicates a "more alarming" congestion scenario

Refinement

Q: When should the exponential increase switch to linear?

A: When **CongWin** gets to 1/2 of its value before timeout.



Implementation:

- Variable Threshold
- At loss event, Threshold is set to 1/2 of CongWin just before loss event

Summary: TCP Congestion Control

- When **CongWin** is below **Threshold**, sender in slow-start phase, window grows exponentially.
- When **CongWin** is above **Threshold**, sender is in congestion-avoidance phase, window grows linearly.
- When a triple duplicate ACK occurs, **Threshold** set to **CongWin/2** and **CongWin** set to **Threshold**.
- When timeout occurs, **Threshold** set to **CongWin/2** and **CongWin** is set to 1 MSS.

TCP sender congestion control

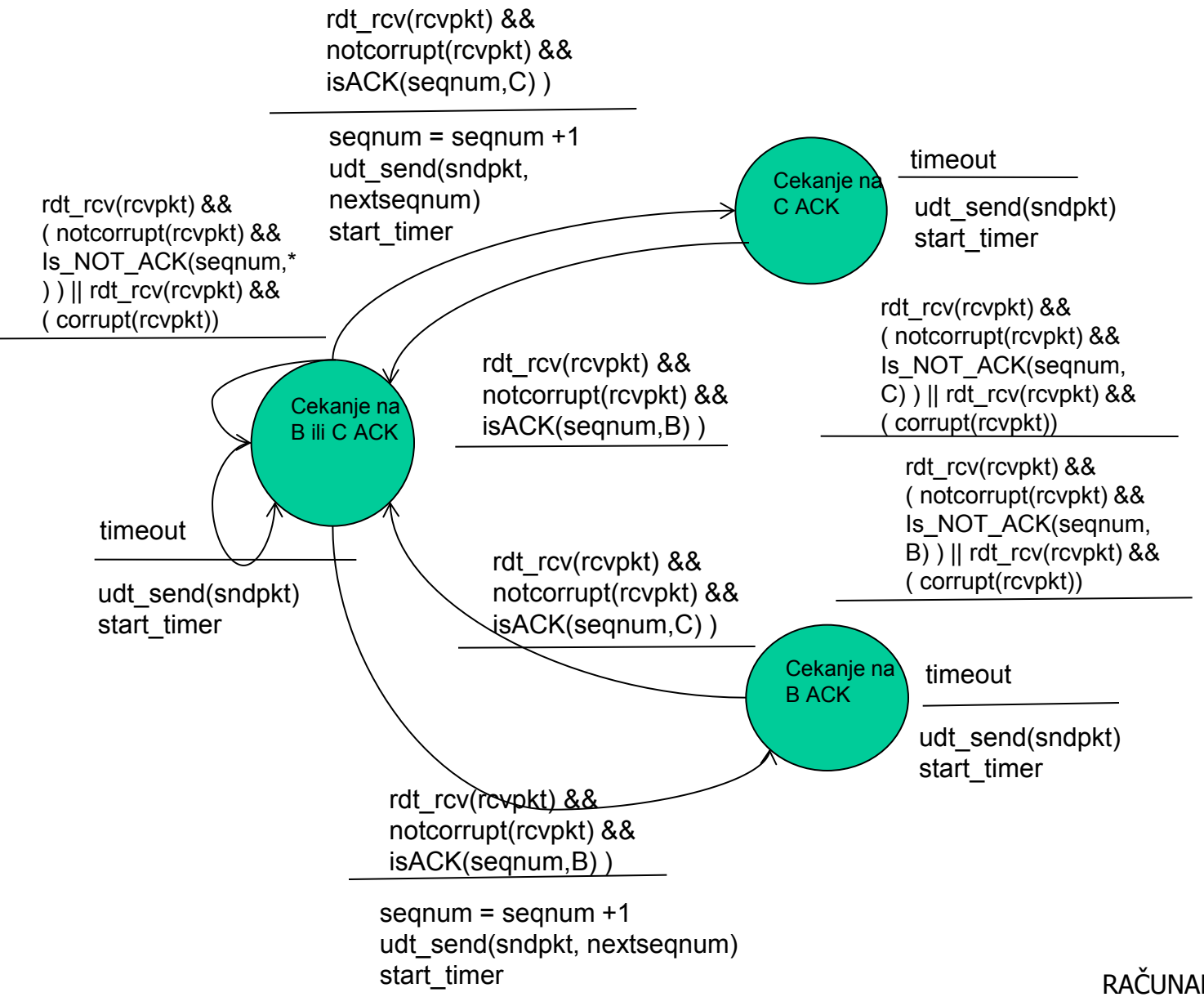
State	Event	TCP Sender Action	Commentary
Slow Start (SS)	ACK receipt for previously unacked data	CongWin = CongWin + MSS, If (CongWin > Threshold) set state to "Congestion Avoidance"	Resulting in a doubling of CongWin every RTT
Congestion Avoidance (CA)	ACK receipt for previously unacked data	CongWin = CongWin+MSS * (MSS/CongWin)	Additive increase, resulting in increase of CongWin by 1 MSS every RTT
SS or CA	Loss event detected by triple duplicate ACK	Threshold = CongWin/2, CongWin = Threshold, Set state to "Congestion Avoidance"	Fast recovery, implementing multiplicative decrease. CongWin will not drop below 1 MSS.
SS or CA	Timeout	Threshold = CongWin/2, CongWin = 1 MSS, Set state to "Slow Start"	Enter slow start
SS or CA	Duplicate ACK	Increment duplicate ACK count for segment being acked	CongWin and Threshold not changed

Zadaci

8. Analizirajte scenario u kojem računar A želi istovremeno da šalje poruke računarima B i C. A povezan sa B i C kanalom za difuzno emitovanje-kanal će preneti računarima B i C paket poslat iz računara A. Pretpostavite da kanal za difuzno emitovanje koji povezuje A, B i C može nezavisno da gubi i oštećuje poruke (na primer, da poruka poslata iz A bude pravilno priljena u B, ali ne i u C). Projektuje protokol za kontrolu grešaka u stilu "stani i čekaj", za pouzdan prenos paketa od A do B i C, takav da A neće prihvatati nove podatke od gornjeg sloja, sve dok ne utvrdi da su i B i C pravilno primili trenutni paket. Napravite konačne automate za A i C. (Napomena: Konačni automati za B trebalo bi u osnovi da bude isti kao za C.)

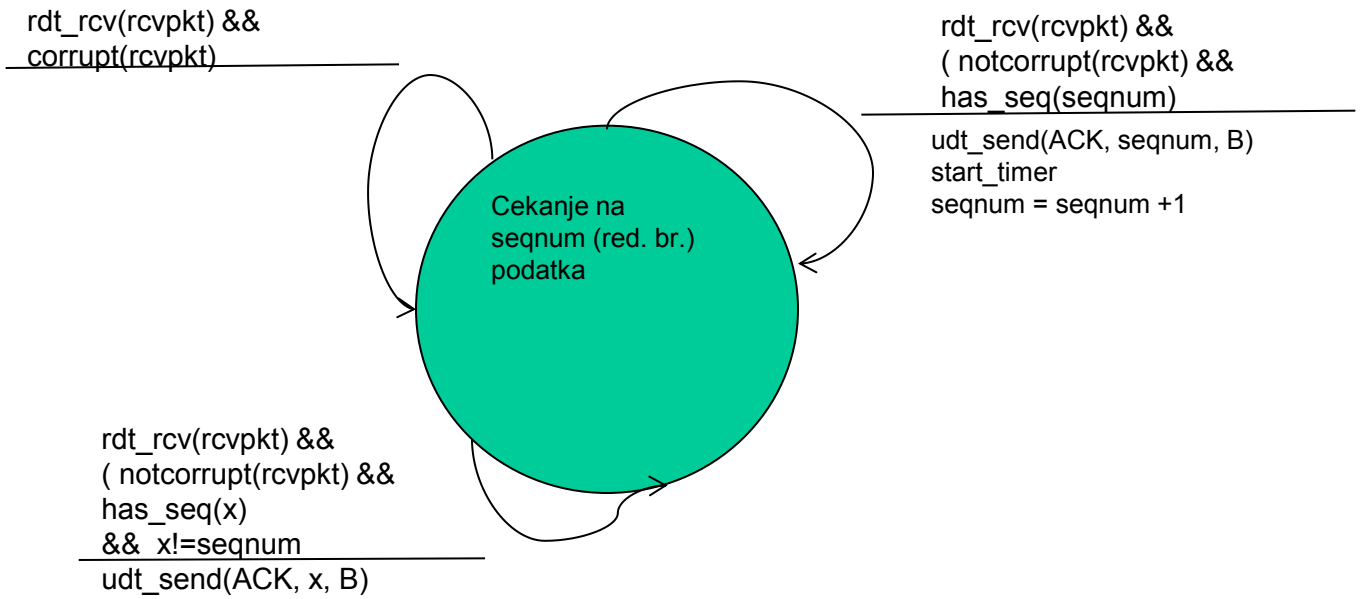
Rešenje:

Pošiljaoc

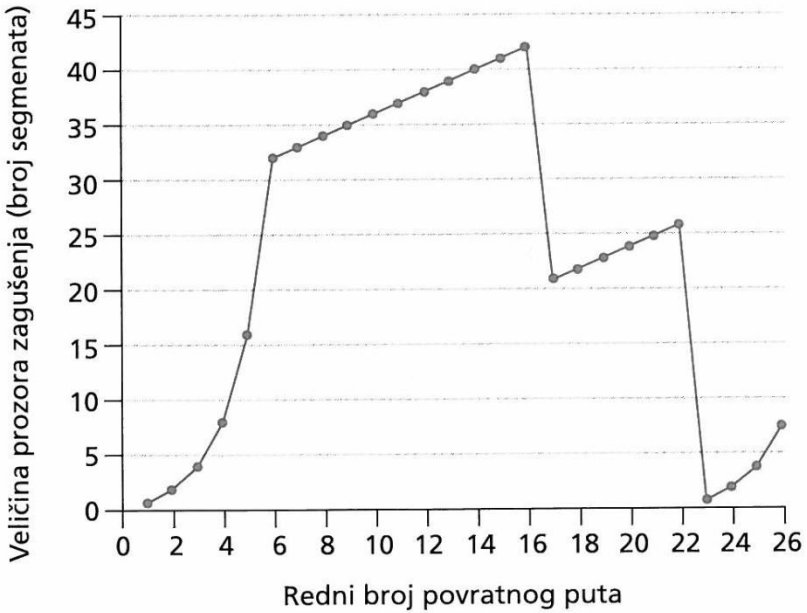


Rešenje:

Primalac B



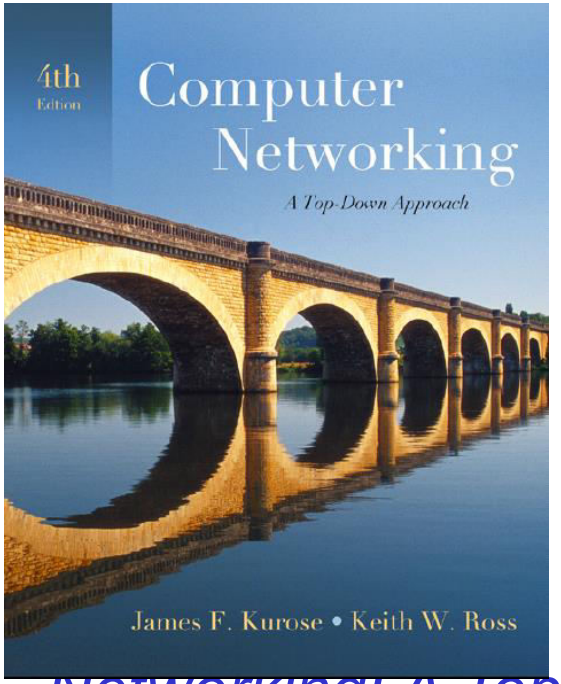
9. Razmotrite sledeći dijagram veličine TCP prozora u funkciji vremena. Pod pretpostavkom da se prikazano ponašanje u protokolu TCP Reno, odgovorite na sledeća pitanja (pogledajte sliku na sledećen slajdu)
   - Ozančite vremenske intervale u kojima TCP vrši spori početak.
   - Ozančite vremenske intervale u kojima TCP izvršava izbegavanje zagušenja.
   - Da li se gubitak segmenta nakon 16. povratnog puta otkriven na osnovu tri identična ACK-a ili na osnovu tajm-auta.
   - Da li se gubitak segmenta nakon 22. povratnog puta otkriven na osnovu tri identična ACK-a ili na osnovu tajm-auta.
   - Koja je vrednost praga Threshold tokom prve povratene putanje.


Veličina prozora zagušenja (broj segmenata)
45
40
35
30
25
20
15
10
5
0
0 2 4 6 8 10 12 14 16 18 20 22 24 26
Redni broj povratnog puta

Rešenje:

- TCP spori strart se izvršava u intervalima [1,6] i [23,26].
- TCP izbegavanje zagušenja izvršava se u intervalima [6,16] i [17,22].
- gubitak segmenta nakon 16. povratnog puta otkriven na osnovu tri identična ACK-a . U slučaju gubitka segmenta usled tajm-auta veličina prozora zagušnja bila bi 1, što nije u ovom slučaju.
- gubitak segmenta nakon 22. povratnog puta otkriven na osnovu tajm-auta, te je prozor zagušenje jednak 1 .
- početna vrednost praga Threshold tokom prve povratne putanje je 32. koji se moze definisati sa tačkom A tj. njenim koordinatama. Možeemo napisati A(6, **32**), data tačka, definiše kraj faze sporog početka odnosno eksponencijalnog rasta brzine slanja i početak faze linearnog rasta tj. perioda izbegavanja zagušenja.

Chapter 4
Network Layer



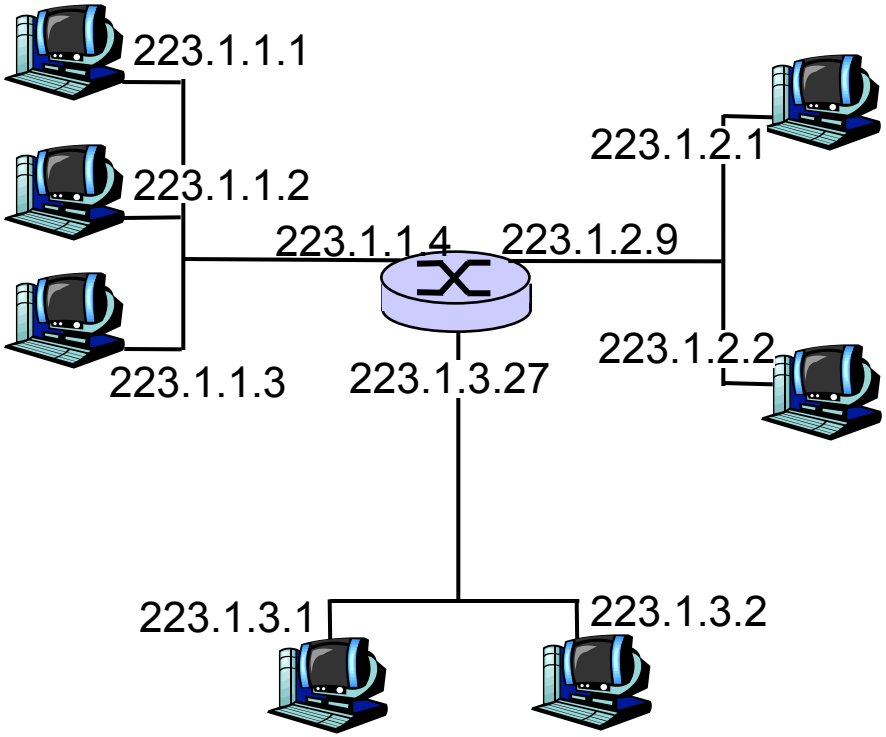
RAČUNARSKE MREŽE I WEB PROGRAMIRANJE
IV- GODINA

prof. dr Zlatko Langović

Networking: A Top Down Approach
4th edition.
Jim Kurose, Keith Ross
Addison-Wesley, July 2007.

IP Addressing: introduction

- IP address: 32-bit identifier for host, router *interface*
- *interface:* connection between host/router and physical link
  - router's typically have multiple interfaces
  - host typically has one interface
  - IP addresses associated with each interface

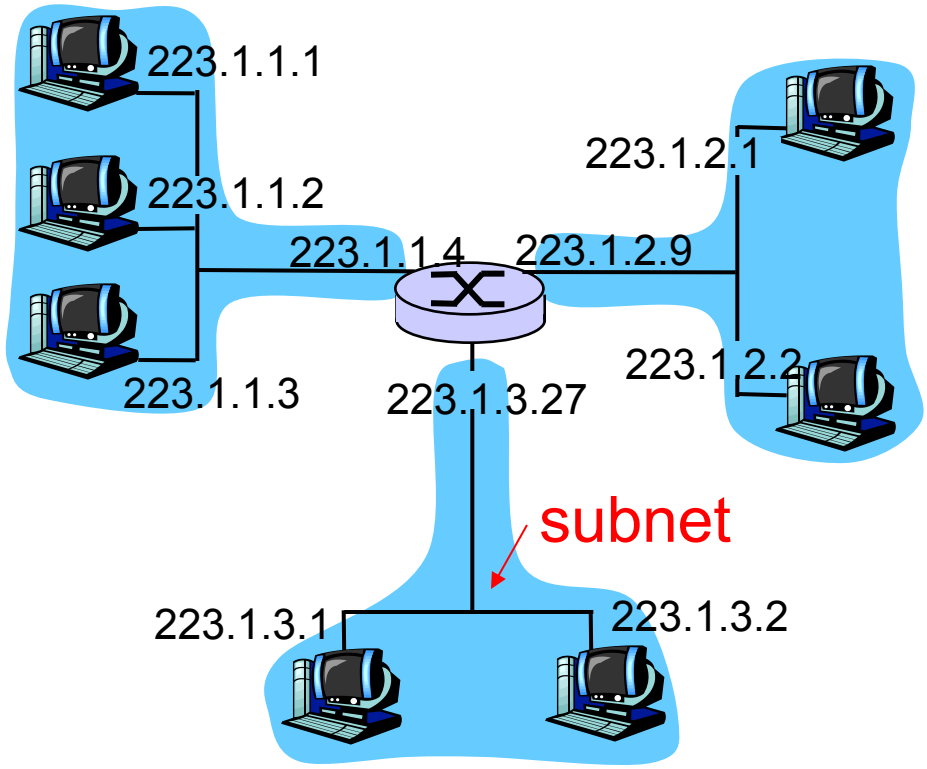


223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

Subnets

- IP address:
  - subnet part (high order bits)
  - host part (low order bits)
- *What's a subnet ?*
  - device interfaces with same subnet part of IP address
  - can physically reach each other without intervening router

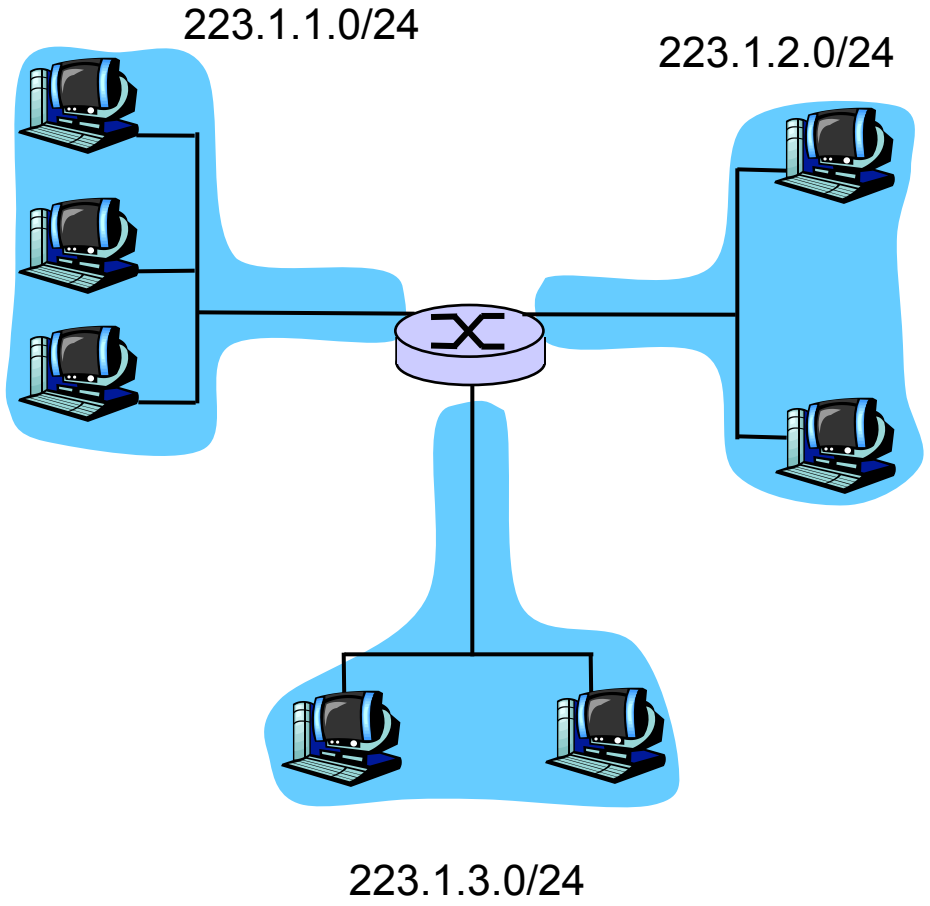


network consisting of 3 subnets

Subnets

Recipe

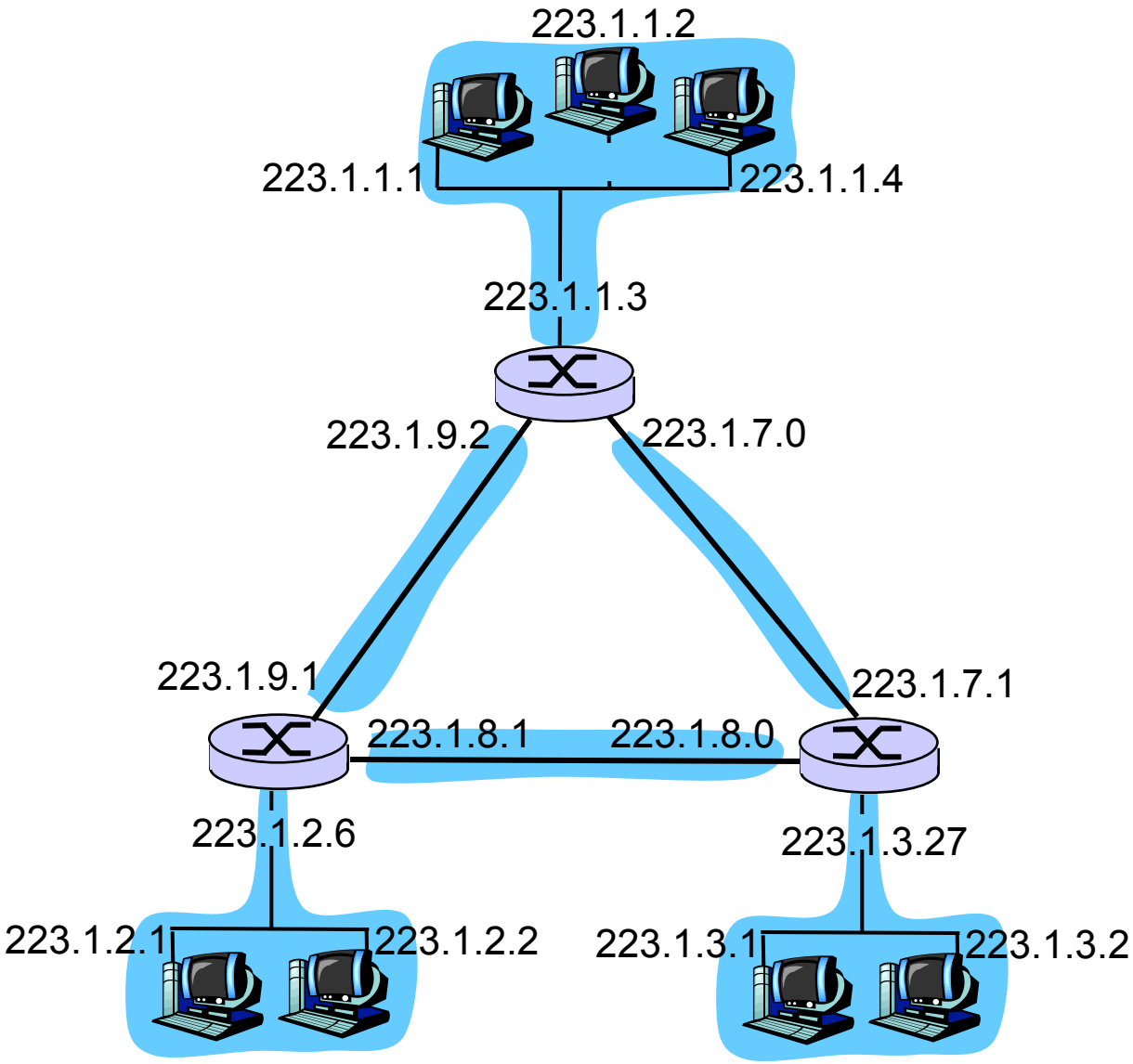
- To determine the subnets, detach each interface from its host or router, creating islands of isolated networks. Each isolated network is called a subnet.



Subnet mask: /24

Subnets

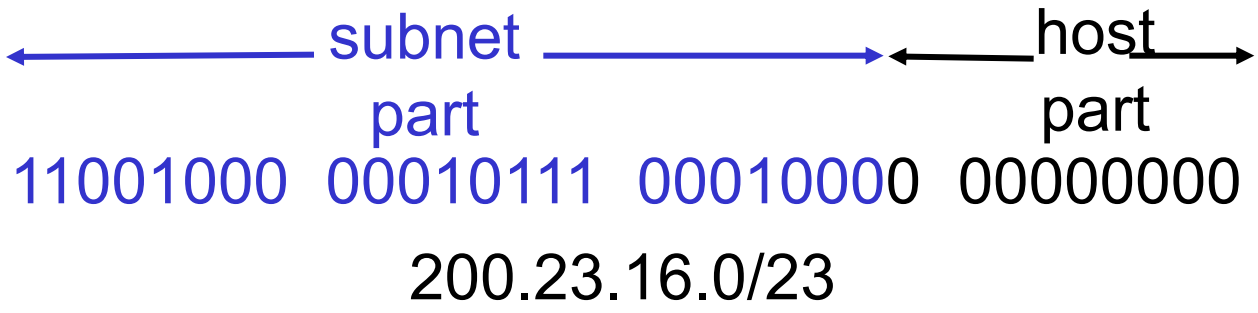
How many?



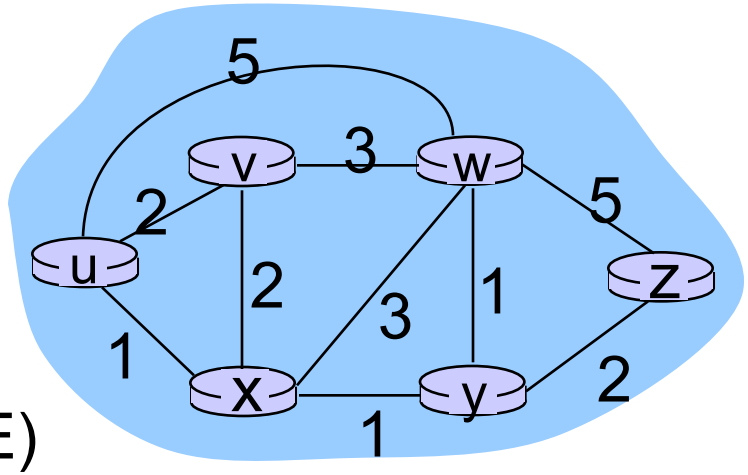
IP addressing: CIDR

CIDR: Classless InterDomain Routing

- subnet portion of address of arbitrary length
- address format: a.b.c.d/x, where x is # bits in subnet portion of address



Graph abstraction



Graph: G = (N,E)

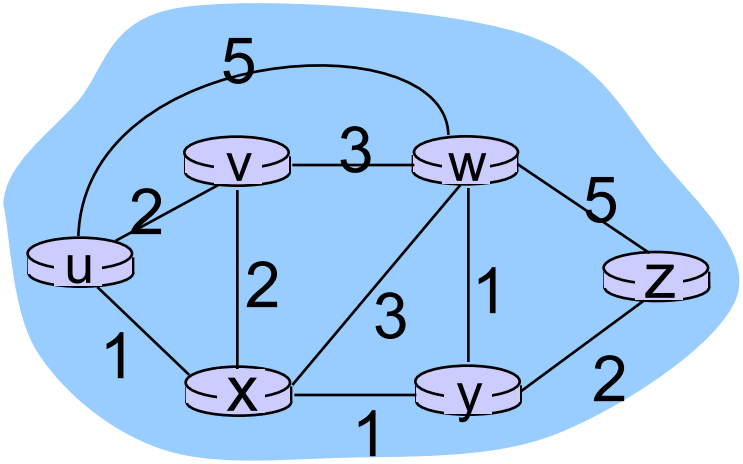
N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (v

Remark: Graph abstraction is useful in other network contexts

Example: P2P, where N is set of peers and E is set of TCP connections

Graph abstraction: costs



- c(x,x') = cost of link (x,x')

  - e.g., c(w,z) = 5

- cost could always be 1, or inversely related to bandwid
or inversely related to congestion

Cost of path $(x_1, x_2, x_3, \ldots, x_p)$ = $c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1}$

Question: What's the least-cost path between u and z ?

Routing algorithm: algorithm that finds least-cost path

Routing Algorithm classification

Global or decentralized information?

Global:

- all routers have complete topology, link cost info
- "link state" algorithms

Decentralized:

- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- "distance vector" algorithms

Static or dynamic?

Static:

- routes change slowly over time

Dynamic:

- routes change more quickly
  - periodic update
  - in response to link cost changes

A Link-State Routing Algorithm

Dijkstra's algorithm

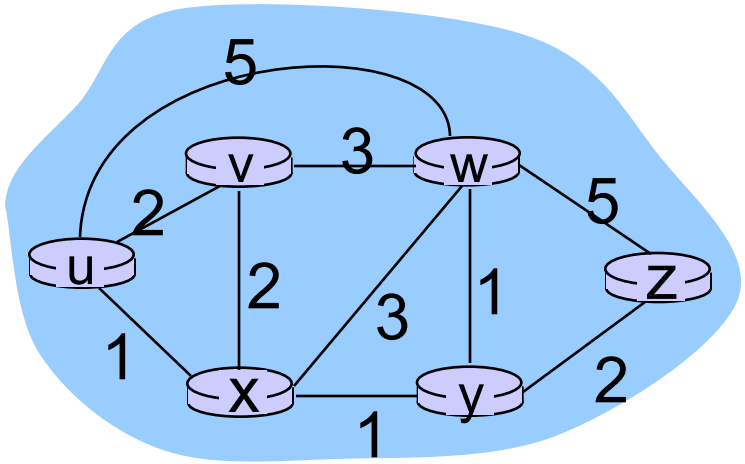
- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ("source") to all other nodes
  - gives forwarding table for that node
- iterative: after k iterations, know least cost path to k dest.'s

Notation:

- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current value of cost of path from source to dest. v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

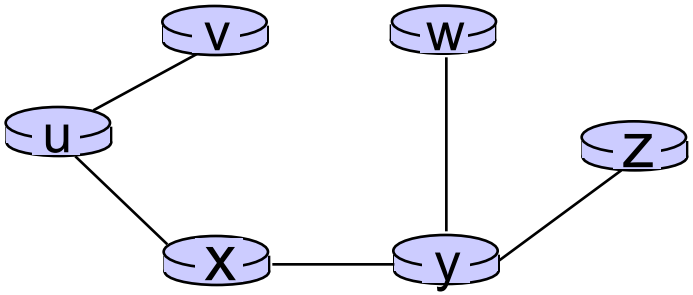
Dijkstra's algorithm: example

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Dijkstra's algorithm: example (2)

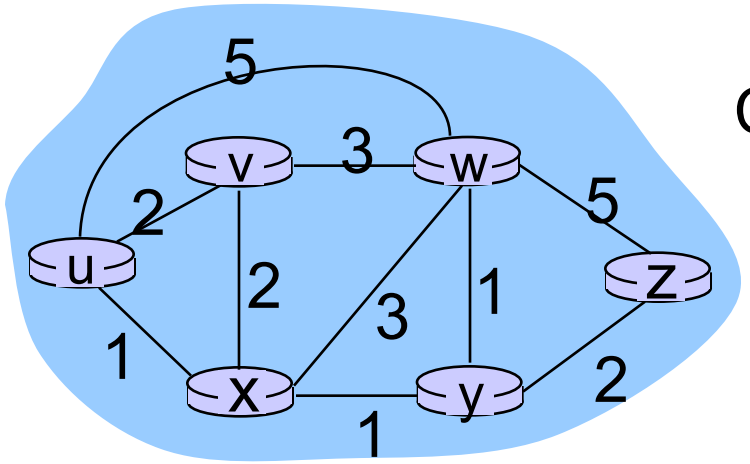
Resulting shortest-path tree from u:



Resulting forwarding table in u:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Bellman-Ford example



Clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$d_u(z) = \min \{ c(u,v) + d_v(z), \; c(u,x) + d_x(z), \; c(u,w) + d_w(z) \}$$
$$= \min \{2 + 5, \; 1 + 3, \; 5 + 3\} = 4$$

Node that achieves minimum is next hop in shortest path ➜ forwarding table

Distance Vector Algorithm

- $D_x(y)$ = estimate of least cost from x to y
- Node x knows cost to each neighbor v: $c(x,v)$
- Node x maintains distance vector $\mathbf{D}_x = [D_x(y): y \in N]$
- Node x also maintains its neighbors' distance vectors
  - For each neighbor v, x maintains $\mathbf{D}_v = [D_v(y): y \in N]$

Distance vector algorithm (4)

Basic idea:

- From time-to-time, each node sends its own distance vector estimate to neighbors
- Asynchronous
- When a node x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow min_v\{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

- Under minor, natural conditions, the estimate $D_x(y)$ *converge to the actual least cost* $d_x(y)$

Distance Vector Algorithm (5)

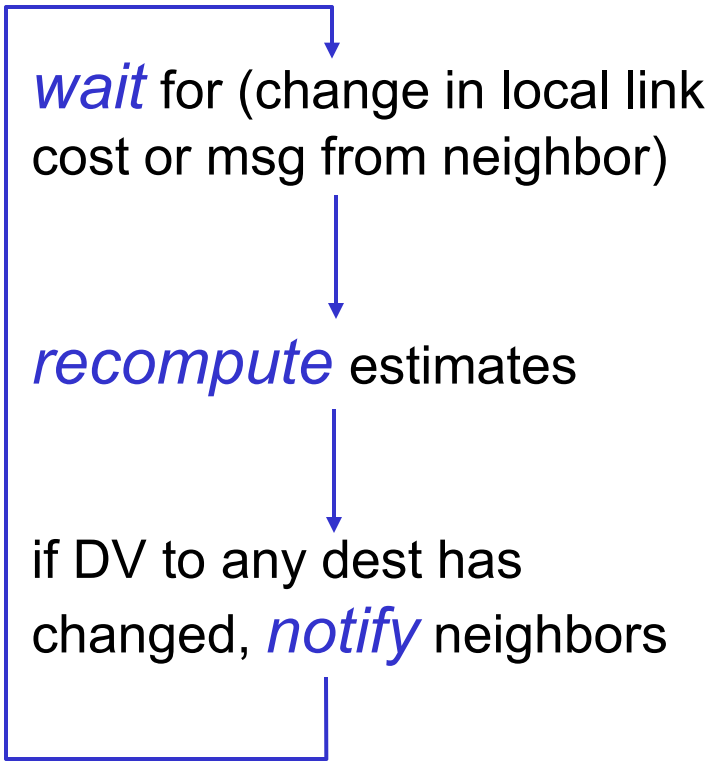
Iterative, asynchronous: each local iteration caused by:

- local link cost change
- DV update message from neighbor

Distributed:

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

Each node:



$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) +$
$D_y(z), c(x,z) + D$
$= \min\{2+1 , 7+0\} = 3$

node x table

from	cost to x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from	cost to x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from	cost to x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from	cost to x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from	cost to x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

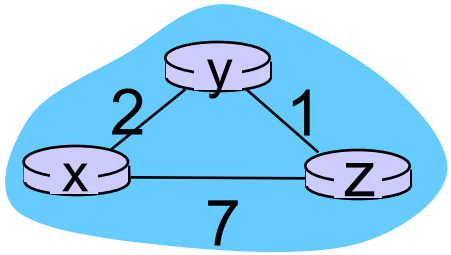
from	cost to x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

from	cost to x	y	z
x	∞	∞	
y	∞	∞	∞
z	7	1	0

from	cost to x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from	cost to x	y	z
x	0	2	3
y	2	0	1
z	3	1	0



time

Zadaci

10. Zamislite ruter koji povezuje tri podmreže: Podmrežu 1, Podmrežu 2 i Podmrežu 3. Pretpostavite da svi interfejsi u tim podmrežama moraju da imaju prefiks 223.1.17/24. Takođe pretpostavite da Podmreža 1 mora da podrži najviše 125 interfejsa a Podmreže 2 i 3 treba da podrže najviše po 60 interfejsa. Odredite tri mrežne adrese (oblika a.b.c/x) koji će zadovoljiti date zahteve.

Rešenje:

- Mrežni prefiks (deo adrese koji definiše mrežu) definisan je sa 24 bita. Dok je, sufiks deo, koji definiše host sisteme, određen sa 8 bita. Međutim, potrebna vrednost broja host interfejsa je 125 odnosno $128=2^7$, te sledi 8-1=7. Drugim rečima od 8 bita host dela, jedan bit je "neiskorišećen" te može "preći" u mrežni deo: 24+1=25. Možemo napisati sledeći izraz za Podmeržu - 1: 223.1.17.0/25;
- Mrežni prefiks (deo adrese koji definiše mrežu) definisan je sa 24 bita. Dok je, sufiks deo, koji definiše host sisteme, određen sa 8 bita. Međutim, potrebna vrednost broja host interfejsa je 60 odnosno $64=2^6$, te sledi 8-2=6. Drugim rečima od 8 bita host dela, dva bita su "neiskorišećena" te mogu "preći" u mrežni deo: 24+2=26. Možemo napisati sledeći izraz za Podmeržu - 2: 223.1.128/26 (0+128=128);
- Mrežni prefiks (deo adrese koji definiše mrežu) definisan je sa 24 bita. Dok je, sufiks deo, koji definiše host sisteme, određen sa 8 bita. Međutim, potrebna vrednost broja host interfejsa je 60 odnosno $64=2^6$, te sledi 8-2=6. Drugim rečima od 8 bita host dela, dva bita su "neiskorišećena" te mogu "preći" u mrežni deo: 24+2=26. Možemo napisati sledeći izraz za Podmeržu - 3: 223.1.192/26 (128+64=192).

11. Razmotrite mrežu prikazanu na datoj slici. Pomoću Dijkstrinog algoritma.
    - Izračunajte najkraću putanju od s do svih ostalih mrežnih čvorova.

Rešenje:

N'	D(s), p(s)	D(t), p(t)	D(u), p(u)	D(v), p(v)	D(w), p(w)	D(y), p(y)	D(z), p(z)
	inf	inf	inf	3,x	1,x	6,x	inf
w	Inf	inf	4,w	2,w		6,x	inf
wv	Inf	11,v	3,v			3,v	inf
wvu	7,u	5,u				3,v	inf
wvuy	7,u	5,u					17,y
wvuyt	6,t						7,t
Wvuyts							7,t

12. Razmotrite sledeću mrežu. Koristite algoritam rastojanja. Popunite stavke u tabeli rastojanja.

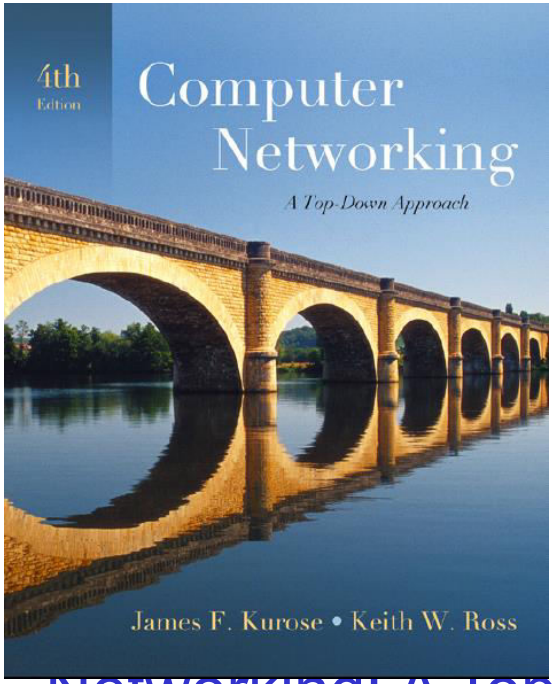
Rešenje:

				Cena	
		u	**v**	**x**	**y**
From	**v**	6	5	8	9
	x	4	5	2	3
	y	13	14	11	10

Chapter 5
Link Layer and LANs

RAČUNARSKE MREŽE I WEB PROGRAMIRANJE
IV- GODINA

prof. dr Zlatko Langović



Networking: A Top Down Approach 4th edition.
Jim Kurose, Keith Ross
Addison-Wesley, July 2007.

Slotted ALOHA

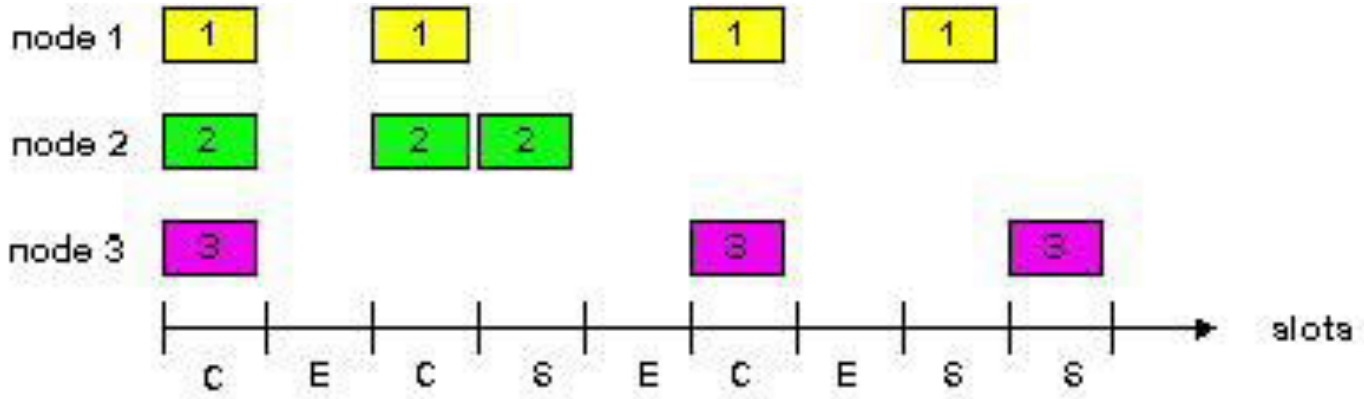
Assumptions:

- all frames same size
- time divided into equal size slots (time to transmit 1 frame)
- nodes start to transmit only slot beginning
- nodes are synchronized
- if 2 or more nodes transmit in slot, all nodes detect collision

Operation:

- when node obtains fresh frame, transmits in next slot
  - *if no collision:* node can send new frame in next slot
  - *if collision:* node retransmits frame in each subsequent slot with prob. p until success

Slotted ALOHA



Pros

- single active node can continuously transmit at full rate of channel
- highly decentralized: only slots in nodes need to be in sync
- simple

Cons

- collisions, wasting slots
- idle slots
- nodes may be able to detect collision in less than time to transmit packet
- clock synchronization

Slotted Aloha efficiency

Efficiency : long-run fraction of successful slots (many nodes, all with many frames to send)

- *suppose:* N nodes with many frames to send, each transmits in slot with probability *p*
- prob that given node has success in a slot = $p(1-p)^{N-1}$
- prob that *any* node has a success = $Np(1-p)^{N-1}$

- max efficiency: find p^* that maximizes $Np(1-p)^{N-1}$
- for many nodes, take limit of $Np^*(1-p^*)^{N-1}$ as N goes to infinity, gives:

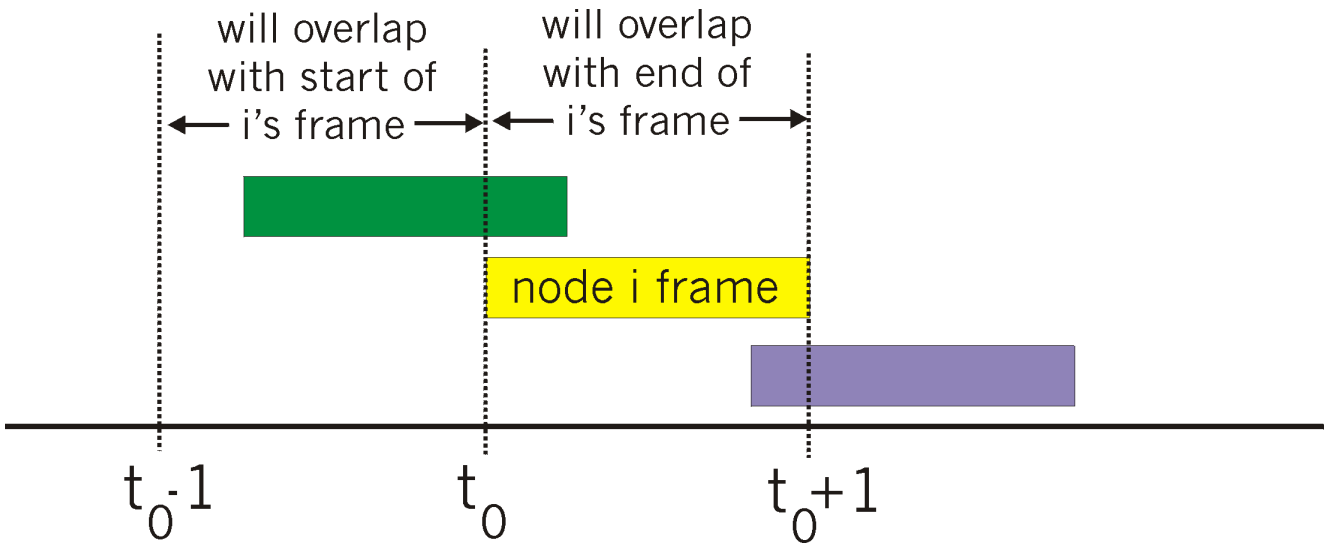
Max efficiency = 1/e = .37

At best: channel used for useful transmissions 37% of time!

!

Pure (unslotted) ALOHA

- unslotted Aloha: simpler, no synchronization
- when frame first arrives
  - transmit immediately
- collision probability increases:
  - frame sent at t_0 collides with other frames sent in $[t_0-1, t_0+1]$



Summary of MAC protocols

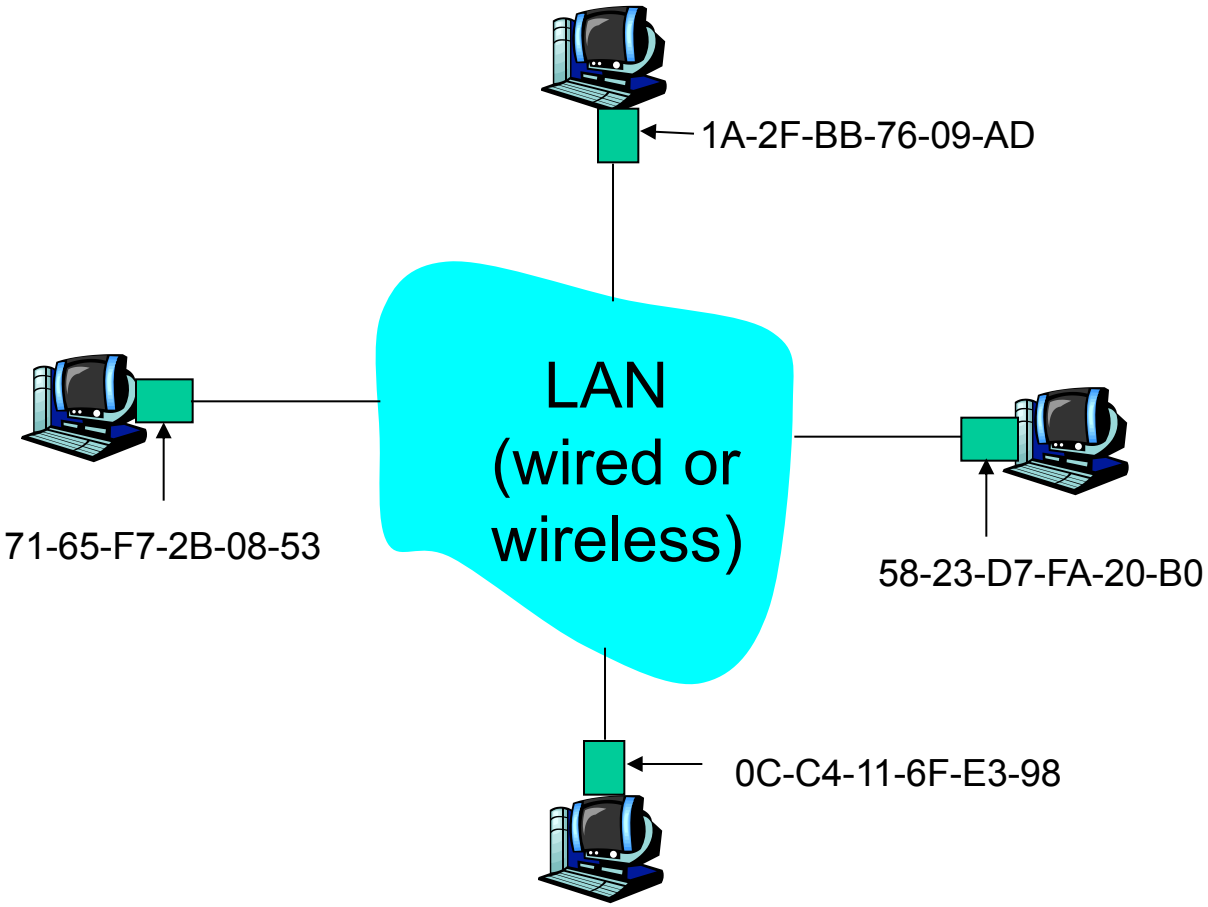
- *channel partitioning,* by time, frequency or code
  - Time Division, Frequency Division
- *random access* (dynamic),
  - ALOHA, S-ALOHA, CSMA, CSMA/CD
  - carrier sensing: easy in some technologies (wire), hard in others (wireless)
  - CSMA/CD used in Ethernet
  - CSMA/CA used in 802.11
- *taking turns*
  - polling from central site, token passing
  - Bluetooth, FDDI, IBM Token Ring

MAC Addresses and ARP

- 32-bit IP address:
  - *network-layer* address
  - used to get datagram to destination IP subnet
- MAC (or LAN or physical or Ethernet) address:
  - function: *get frame from one interface to another physically-connected interface (same network)*
  - 48 bit MAC address (for most LANs)
    - burned in NIC ROM, also sometimes software settable

LAN Addresses and ARP

Each adapter on LAN has unique LAN address



Broadcast address =
FF-FF-FF-FF-FF-FF

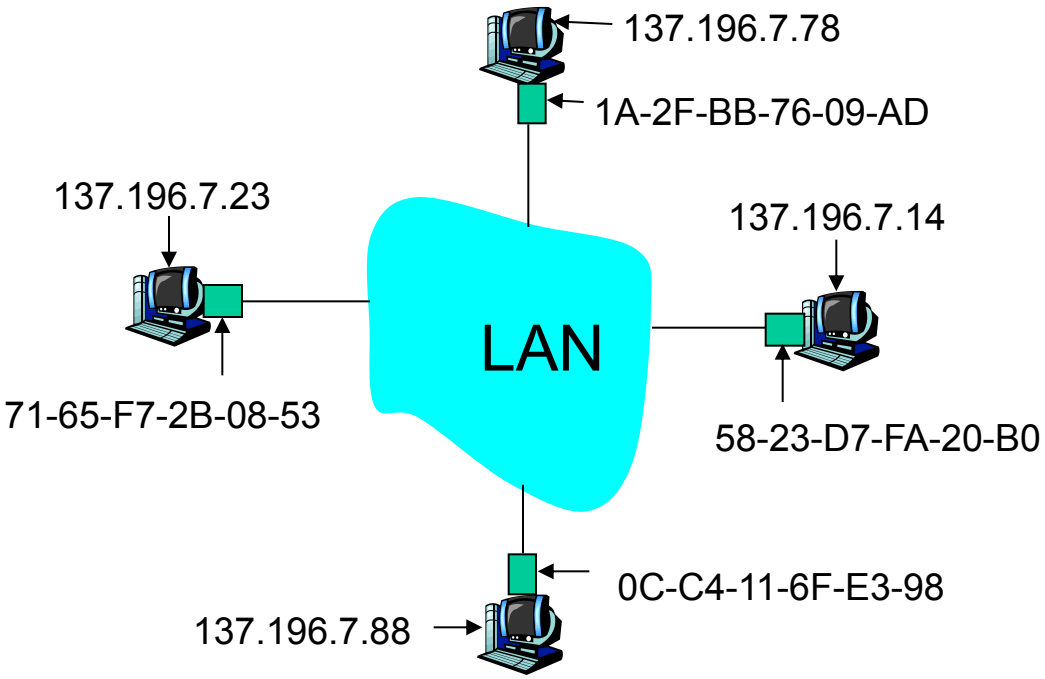
= adapter

LAN Address (more)

- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- analogy:
  - (a) MAC address: like Social Security Number
  - (b) IP address: like postal address
- MAC flat address ➜ portability
  - can move LAN card from one LAN to another
- IP hierarchical address NOT portable
  - address depends on IP subnet to which node is attached

ARP: Address Resolution Protocol

Question: how to determine MAC address of B knowing B's IP address?



- Each IP node (host, router) on LAN has ARP table
- ARP table: IP/MAC address mappings for some LAN nodes

  < IP address; MAC address; TTL>
  - TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

ARP protocol: Same LAN (network)

- A wants to send datagram to B, and B's MAC address not in A's ARP table.
- A broadcasts ARP query packet, containing B's IP address
  - dest MAC address = FF-FF-FF-FF-FF-FF
  - all machines on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A's MAC address (unicast)

- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
  - soft state: information that times out (goes away) unless refreshed
- ARP is "plug-and-play":
  - nodes create their ARP tables *without intervention from net administrator*

Zadaci

13. U odeljku 5.3 definisali smo ALOHE protokol sa odsečcima.
    - Ako je N aktivnih čvorova, efikasnost ALOHE sa odsečcima je dat izrazom Np(1-p)N-1. pronađite vrednost p koja daje maksimalnu vrednost datog izraza.
    - Koristeći vrednost p pronađenu u delu pod a., pronađite efikasnost ALOHE sa odsečcima ako N teži beskonačnosti. ((1-1/N) teži ka 1/e kada N teži beskonačnosti.)

Rešenje:

a)Možemo napisati sledeći brojevni izraz odnosno funkciju. Naći prvi izvod date funkcije.

$$E(p) = Np(1-p)^{N-1} \quad (1)$$

$$E'(p) = N(1-p)^{N-1} - Np(N-1)(1-p)^{N-2}$$

$$= N(1-p)^{N-2}[(1-p) - p(N-1)]$$

$$E'(p) = 0 \Rightarrow (1-p) - p(N-1) = 0 \Rightarrow p_{\max} = \frac{1}{N}$$

b)Zamenjujući vrednost promenljive p sa p_{max} u funkciji(1) možemo napisati:

$$E(p_{\max}) = N\frac{1}{N}(1-\frac{1}{N})^{N-1} = (1-\frac{1}{N})^{N-1} = \frac{(1-\frac{1}{N})^N}{1-\frac{1}{N}} \Rightarrow \lim_{N\to\infty} \frac{(1-\frac{1}{N})^N}{1-\frac{1}{N}}$$ na osnovu datog

izraza slede sledeće jednakosti:

$$\lim_{N\to\infty}(1-\frac{1}{N}) = 1 \qquad \lim_{N\to\infty}(1-\frac{1}{N})^N = \frac{1}{e}$$

Na kraju dobijamo rešenje:

$$\lim_{N\to\infty} E(p_{\max}) = \frac{1}{e}$$

14. Pokažite da je efikasnost čiste ALOHE jednaka 1/(2e).

Rešenje:

$$E(p) = Np(1-p)^{2(N-1)} \qquad (1)$$

Izvod date funkcije po promenljivoj p.

$$E'(p) = N(1-p)^{2(N-2)} - Np2(N-1)(1-p)^{2(N-3)}$$

$$= N(1-p)^{2(N-3)}[(1-p) - p2(N-1)]$$

$$E'(p) = 0 \Rightarrow (1-p) - p2(N-1) = 0 \Rightarrow p_{\max} = \frac{1}{2N-1}$$

Zamenjujući vrednost promenljive p sa $p_{\max}$ u datoj funkciji (1), možemo napisati:

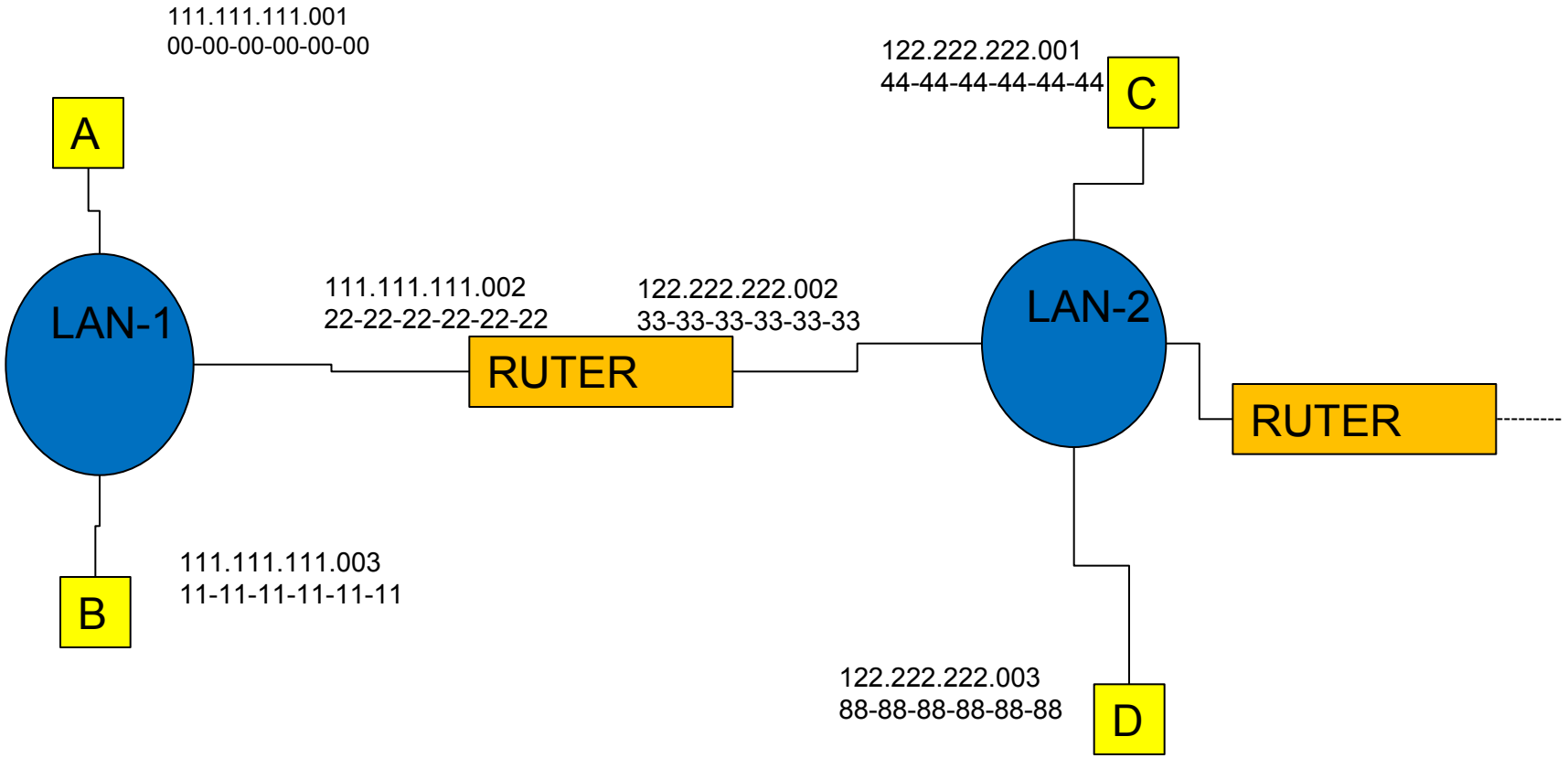
$$E(p_{\max}) = \frac{N}{2N-1}(1 - \frac{1}{2N-1})^{2(N-1)}$$

Na kraju dobijamo brojevni izraz koji definiše rezultat:

$$\lim_{N\to\infty} E(p_{\max}) = \frac{1}{2} \cdot \frac{1}{e} = \frac{1}{2e}$$

15. Razmotrite tri LAN-a međusobno povezana pomoću dva rutera.
    - Dodelite IP adresu svim interfejsima. Za LAN-1 koristite adrese u obliku 111.111.111.xxx; za LAN- 2 koristite adrese u obliku 122.222.222.xxx;
    - Dodelite MAC adrese svim adapterima.

Rešenje:


111.111.111.001
00-00-00-00-00-00
A
122.222.222.001
44-44-44-44-44-44
C
LAN-1
111.111.111.002
22-22-22-22-22-22
122.222.222.002
33-33-33-33-33-33
RUTER
LAN-2
RUTER
111.111.111.003
11-11-11-11-11-11
B
122.222.222.003
88-88-88-88-88-88
D