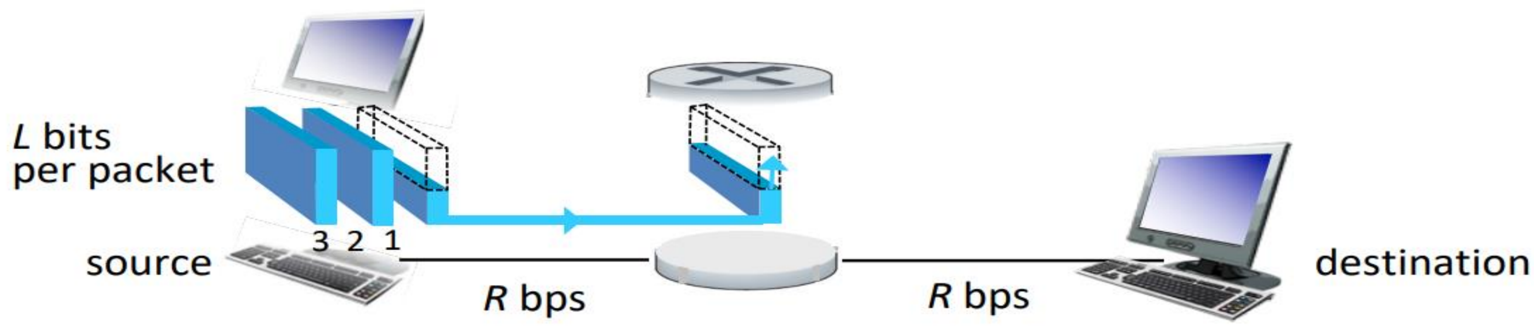
Networks Issues

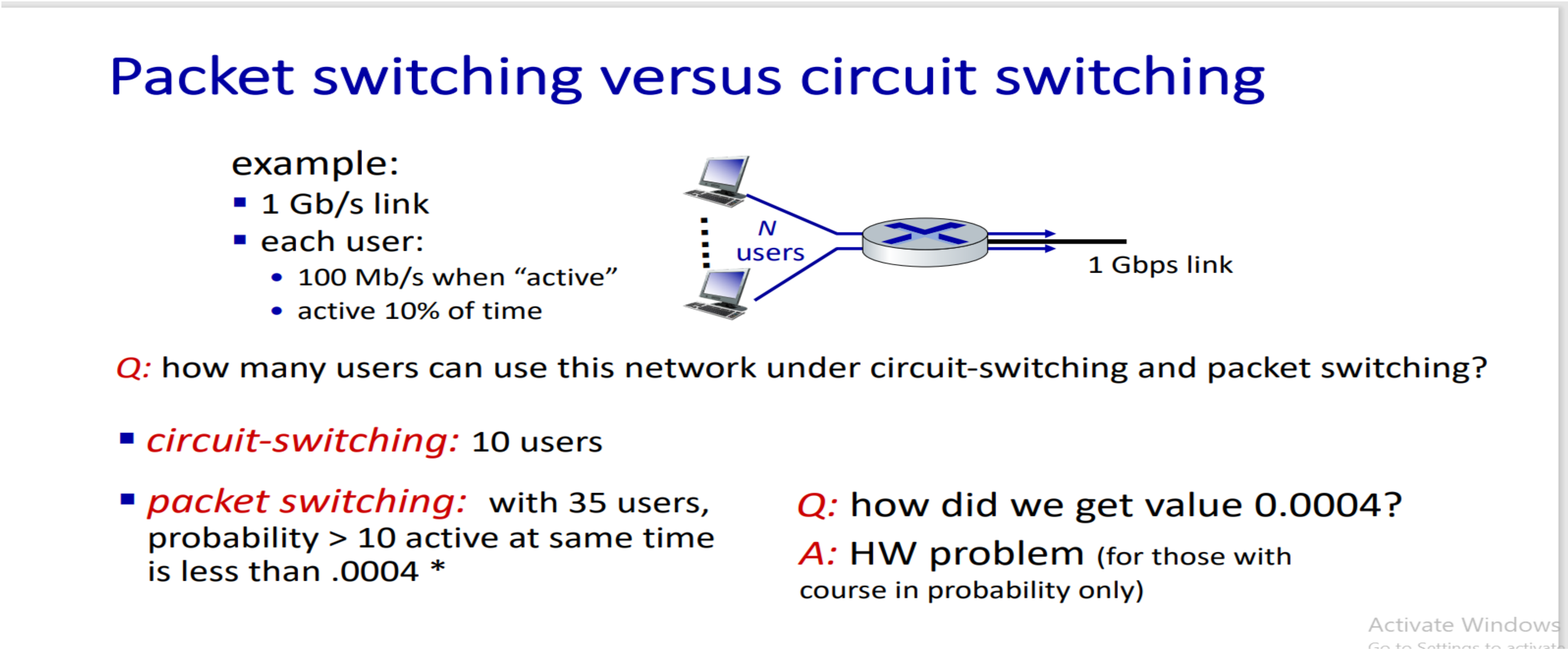
Packet-switching: store-and-forward



- packet transmission delay: takes L/R seconds to transmit (push out) L-bit packet into link at R bps
- *store and forward: entire* packet must arrive at router before it can be transmitted on next link

One-hop numerical example:
- L = 10 Kbits
- R = 100 Mbps
- one-hop transmission delay = 0.1 msec

Transmission delay = L/R = 10*10^3 / 100 * 10^6 = 10^-4 sec = 10^-4 / 10^-3 = 0.1msec



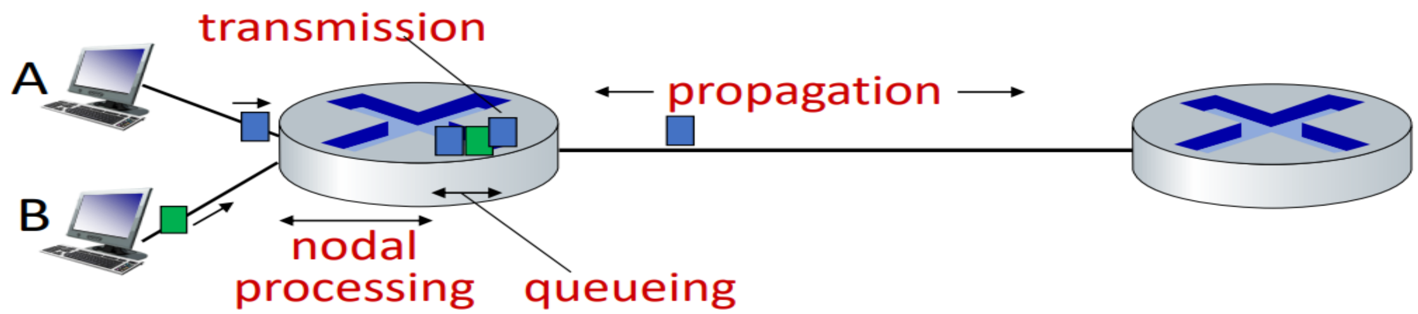
Circuit switched: each user needs 1/10 of link, so can reserve only 10 channels on the link, whether they are using it 10% or 100%.

Packet switched: Each user is using the channel 10% of the time, so probability of a given user being active is $p = 0.1$, and inactive $q = 0.9$.

It's a binomial distribution $X \sim B(35, 0.1)$, so probability $Pr(X=k) = C(35, k)\, p^k q^{(35-k)}$. You need $Pr(X>10)$ which is $1 - Pr(X<=10)$ which is $1-(Pr(X=0) + Pr(X=1) + \dots Pr(X=10))$.

I actually get 0.000424, not "less than 0.0004"

Packet delay: four sources



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

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

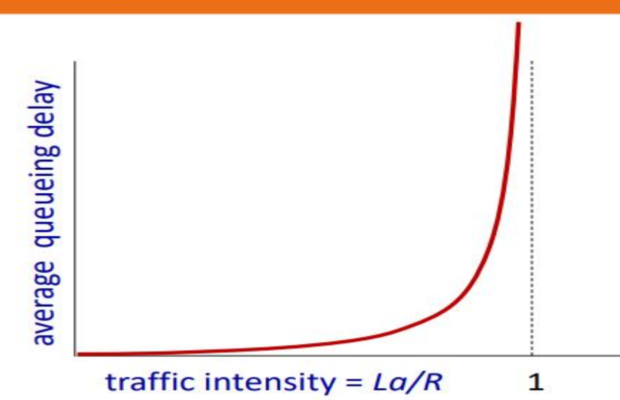
d_{prop}: propagation delay:
- d: length of physical link
- s: propagation speed (~2x10^8 m/sec)
- $d_{prop} = d/s$

d_{trans} and d_{prop}
very different

Packet queueing delay (revisited)

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

$$\frac{L \cdot a}{R} : \quad \frac{\text{arrival rate of bits}}{\text{service rate of bits}} \qquad \text{"traffic intensity"}$$



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

Q) A user can directly connect to a server through either long-range wireless or a twisted-pair cable for transmitting a 1500-bytes file. The transmission rates of the wireless and wired media are 2 and 100 Mbps, respectively. Assume that the propagation speed in air is 3 x 10^8 m/s, while the speed in the twisted pair is 2 x 10^8 m/s. If the user is located 1 km away from the server, what is the nodal delay when using each of the two technologies.

File size = 1500 * 8 bits

Transmission rate of wireless = 2Mbps

Propagation speed of wireless = 3*10^8 m/s

Transmission rate of wired = 100Mbps

Propagation speed of wired = 2*10^8 m/s

1. Wireless

- **Transmission Delay = L(bits)/ R(bps)**

=1500*8(bits)/2*10^6(bps)=0.006(s)=6(ms)

- **Propagation Delay = d(m)/ s(m/s)**

=1*10^3(m)/3*10^8(m/s)=0.00000333333(s)=0.003(ms)

- **Nodal Delay = Transmission Delay + Propagation Delay = 6ms + 0.003ms = 6.003ms**

2. Wired

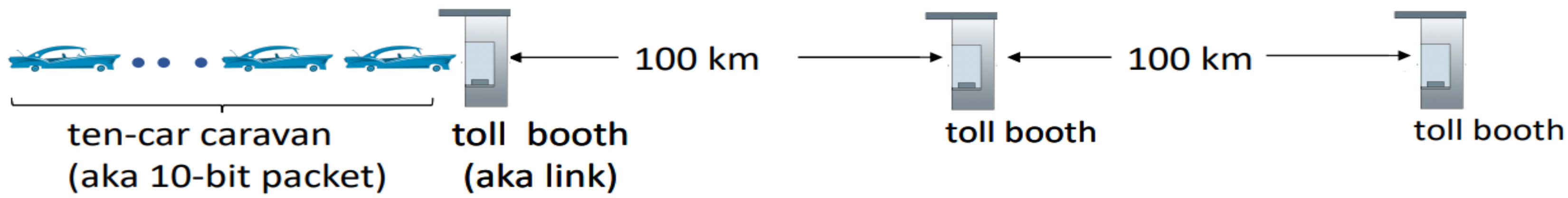
- **Transmission Delay** = L(bits)/ R(bps)

=1500*8(bits)/100*10^6(bps)=0.00012(s)=0.12(ms)

- **Propagation Delay** = d(m)s(m/s)=1000(m)2*108(m/s)=0.000005(s)=0.005(ms)

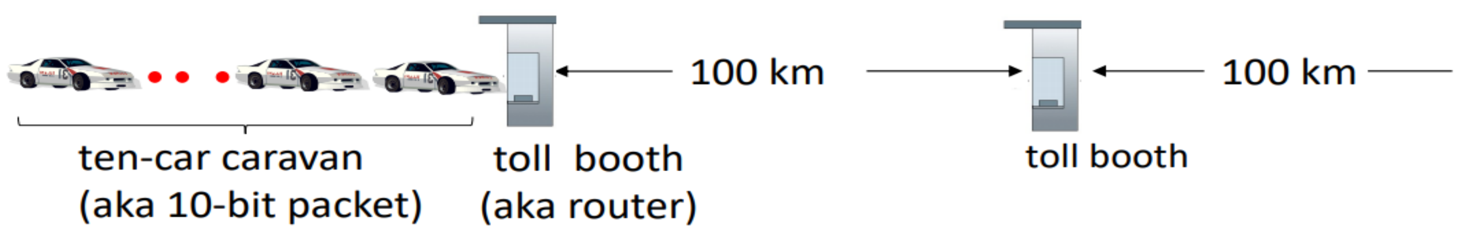
- **Nodal Delay** = Transmission Delay + Propagation Delay = 0.12ms + 0.005ms = 0.125ms

Caravan analogy



- car ~ bit; caravan ~ packet; toll service ~ link transmission
- toll booth takes 12 sec to service car (bit transmission time)
- "propagate" at 100 km/hr
- *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



- 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, first car arrives at second booth; three cars still at first booth

Q)Review the car-caravan analogy in Section 1.4. Assume a propagation speed of 100 km/hour. Tollbooths are 75 km apart, and the cars propagate at 100km/hr. A tollbooth services a car at a rate of one car every 12 seconds. There are 10 cars.

(a.)Suppose the caravan travels 150 km, beginning in front of one tollbooth, passing through a second tollbooth, and finishing just after a third tollbooth. What is the end-to-end delay?

Propagation speed = 100 (km/hr), Distance between tollbooth = 75 (km)

Propogation Delay = Distance / Propagation Speed = 75(km) / 100(km/hr) = 0.75 hr = 45 (min)

Time for taken by each tollbooth to reach 10 cars = 12(s) x 10 = 2 (min)

End-to-end Delay (3 Toolbooths and 2 Hops in between) = 45 x 2 + 2 x 3 = 96 (min)

(b.)Repeat (a.), now assuming that there are eight cars in the caravan instead of ten.

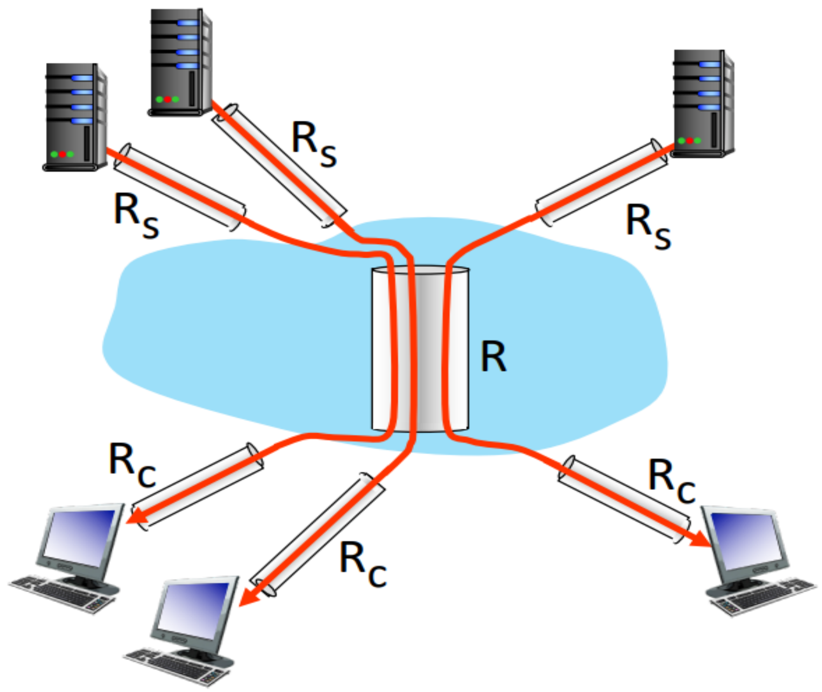
Propagation speed = 100 (km/hr), Distance between tollbooth = 75 (km)

Propogation Delay = Distance / Propagation Speed = 75(km) / 100(km/hr) = 45 (min)

Time for taken by each tollbooth to reach 8 cars = 12(s) x 8 = 96 (s) = 1.6 (min)

End-to-end Delay (3 Toolbooths and 2 Hops in between) = 45 x 2 + 1.6 x 3 = 94.8 (min)

Throughput: network scenario



10 connections (fairly) share backbone bottleneck link *R* bits/sec

- per-connection end-end throughput: *min(R_c,R_s,R/10)*
- in practice: R_c or R_s is often bottleneck

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/

Suppose Host A wants to send a large file to Host B. The path from Host A to Host B has three links, of rate R1 = 500 kbps, R2 = 2 Mbps, and R3 = 1 Mbps.

(a.)Assuming no other traffic in the network, what is the throughput for the file transfer?

Given: R1 = 500 kbps, R2 = 2 Mbps, R3 = 1 Mbps

The throughput for the file transfer = min(R1,R2,R3) = 500 kbps

(b.)Suppose the file is 4 million bytes. Dividing the file size by the throughput, roughly how long will it take to transfer the file to Host B?

L(bits) / R(bps)=4*10^6 (bytes) / 500 (kbps)

= 32*10^6 / 500 * 10 ^3 = 64 sec

(c.)Repeat (a.) and (b.), but now with R2 reduced to 100 kbps.
Throughput = 100 kbps

L(bits) / R(bps)=32*10^6 / 100 * 10 ^3 = 320 sec

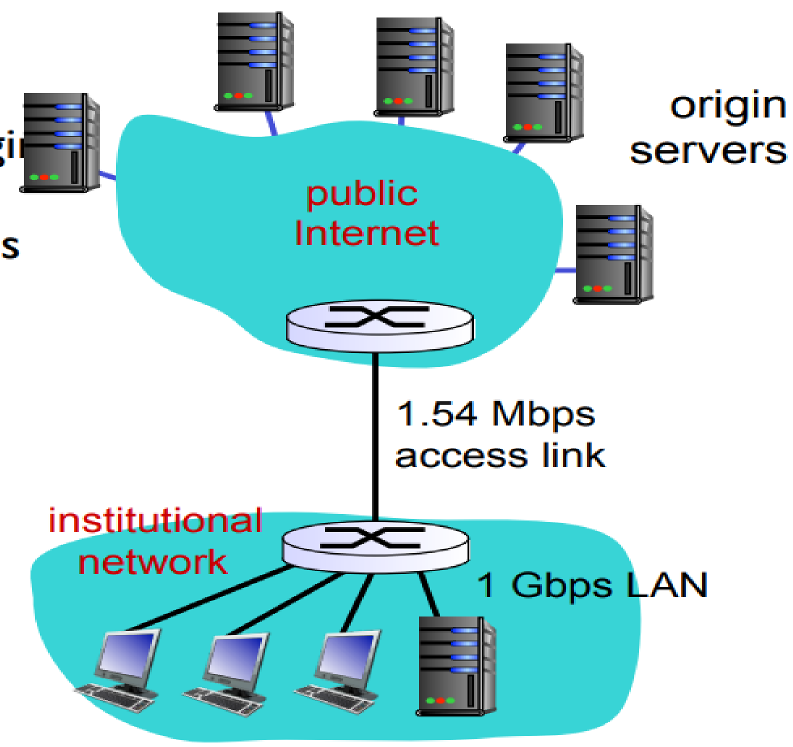
Caching example:

assumptions:

- avg object size: S=100K bits
- avg request rate from browsers to origin servers: A=15/sec
- avg data rate to browsers: R=1.50 Mbps
- access link rate: C=1.54 Mbps
- RTT from institutional router to any origin server: T=200 ms

consequences:

- LAN utilization: 0.15%
- access link utilization ≈ 99% *problem!*
- total delay = Internet delay + access delay + LAN delay
  = 200 ms + ≈minutes + µsecs



Average data rate to browsers = 100k bits * 15 = 1.5Mbps

Access link rate = 1.54Mbps

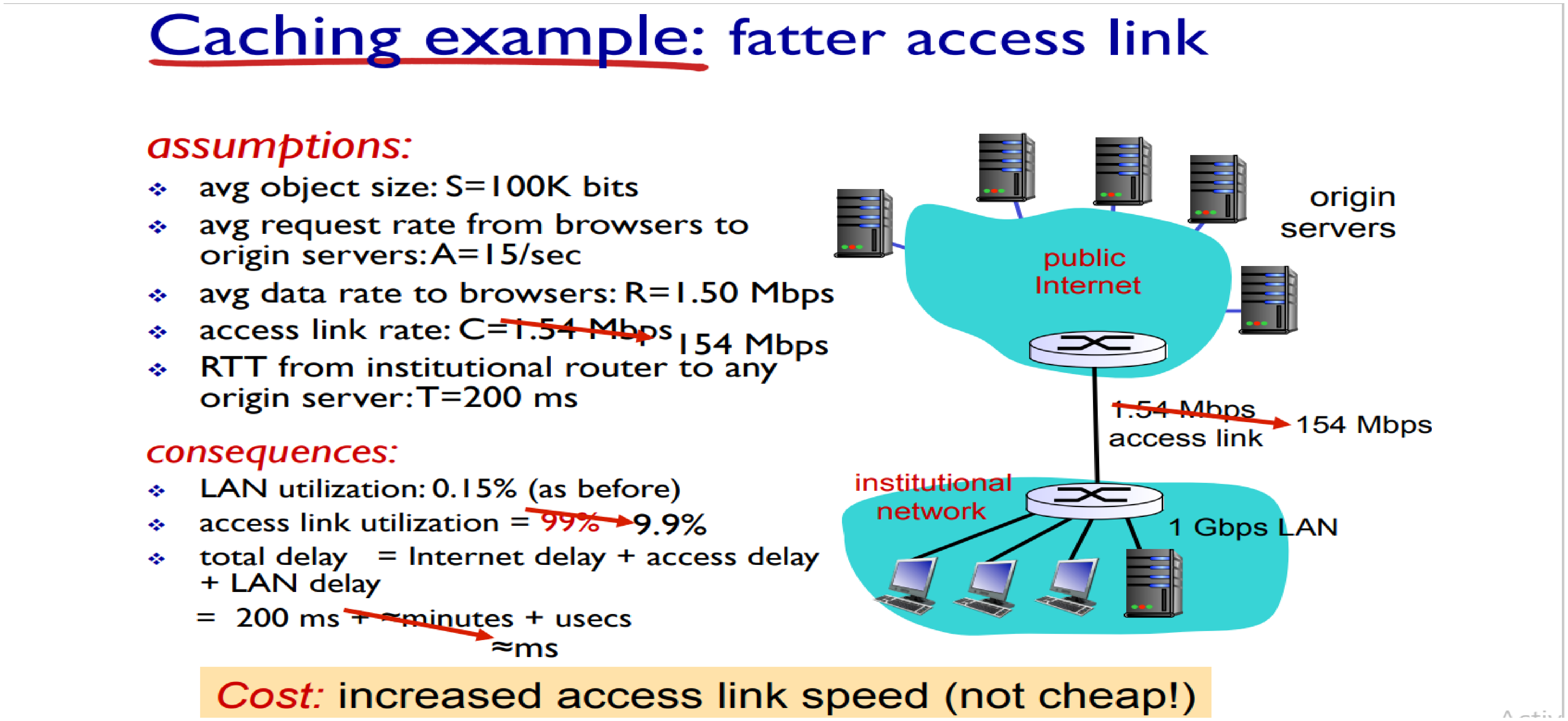
RTT from client to server = 200msec

LAN Link rate = 1Gbps

LAN utilization = Avg data rate to browsers / LAN link rate = 1.5Mbps / 1Gbps = 0.0015 = 0.15%

Access Link utilization = Avg data rate to browsers / access link rate = 1.5Mbps / 1.54 Mbps = 99% (problem in queuing delay)

Total delay = **internet delay (RTT) + Access delay + LAN delay**



Caching example: install local cache

assumptions:

- avg object size: S=100K bits
- avg request rate from browsers to origin servers: A=15/sec
- avg data rate to browsers: R=1.50 Mbps
- access link rate: C=1.54 Mbps
- RTT from institutional router to any origin server: T=200 ms

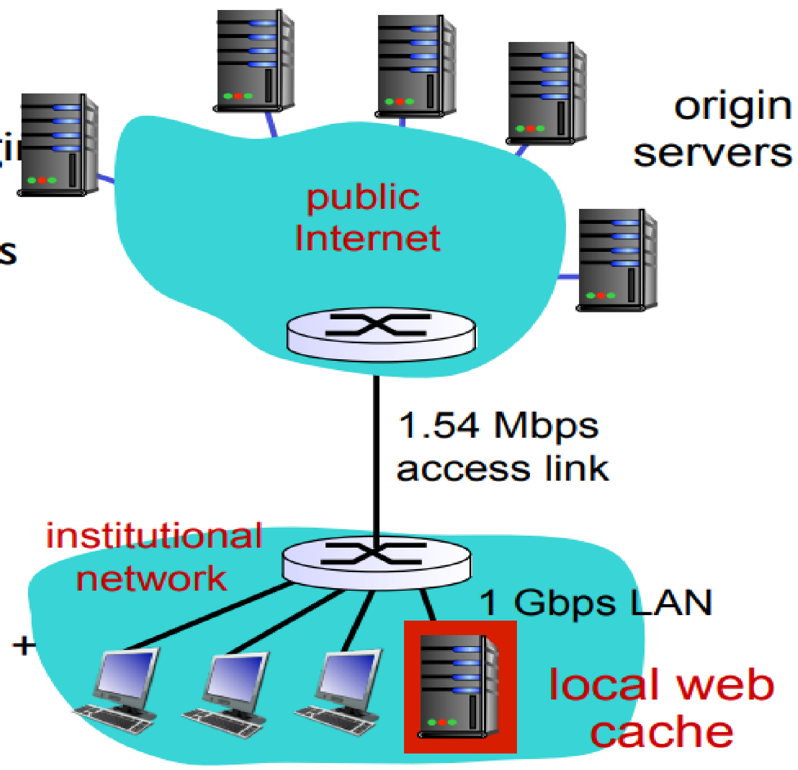
consequences:

- LAN utilization: 0.15% (as before)
- access link utilization = ?
- total delay = ?

lay +

How to compute link utilization, delay?

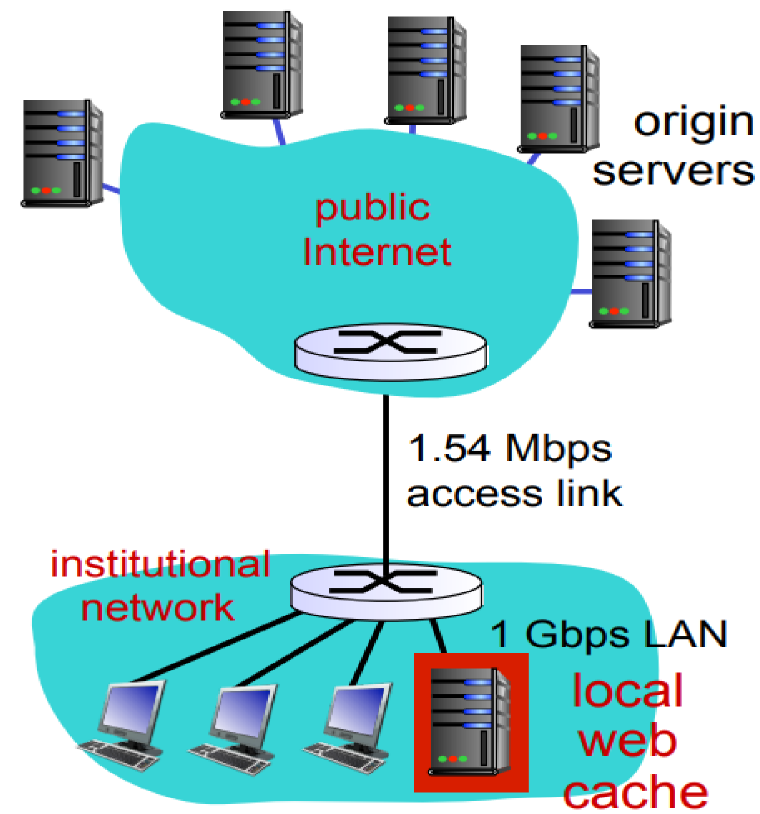
Cost: web cache (cheap!)



install local cache

assumptions:

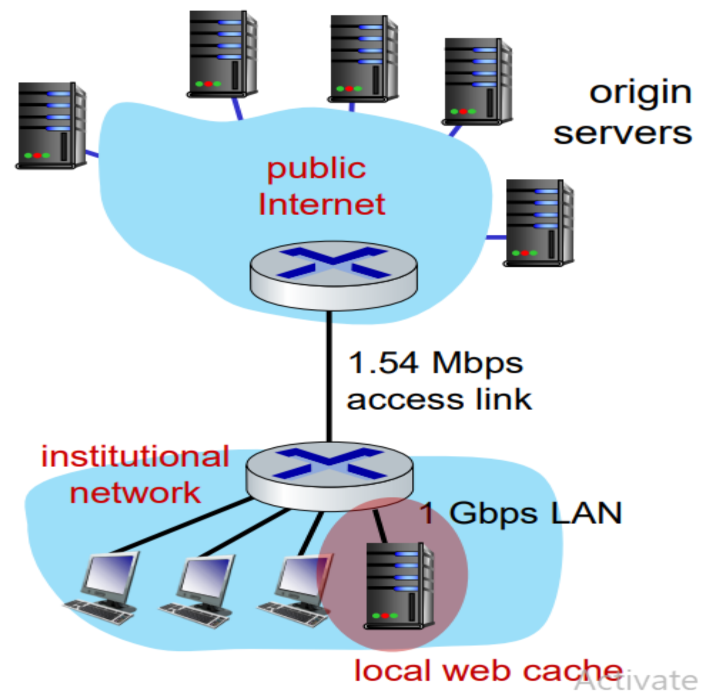
- avg object size: S=100K bits
- avg request rate from browsers to origin servers: A=15/sec
- avg data rate to browsers: R=1.50 Mbps
- access link rate: C=1.54 Mbps
- RTT from institutional router to any origin server: T=200 ms



Calculating access link utilization, end-end delay with cache:

suppose cache hit rate is 0.4:

- 40% requests served by cache, with low (msec) delay
- 60% requests satisfied at origin
  - rate to browsers over access link
    = 0.6 * 1.50 Mbps = .9 Mbps
  - access link utilization = 0.9/1.54 = .58 means low (msec) queueing delay at access link
- average end-end delay:
  = 0.6 * (delay from origin servers)
  + 0.4 * (delay when satisfied at cache)
  = 0.6 (2.01) + 0.4 (~msecs) = ~ 1.2 secs



lower average end-end delay than with 154 Mbps link (and cheaper too!)

Internet checksum: example

example: add two 16-bit integers

		1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
		1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
wraparound	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
sum		1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum		0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

Note: when adding numbers, a carryout from the most significant bit needs to be added to the result

Internet checksum: weak protection!

example: add two 16-bit integers

		1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0 1
		1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1 0
wraparound	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	
sum		1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0	
checksum		0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1	

Even though numbers have changed (bit flips), *no* change in checksum!

TCP sequence numbers, ACKs

Sequence numbers:

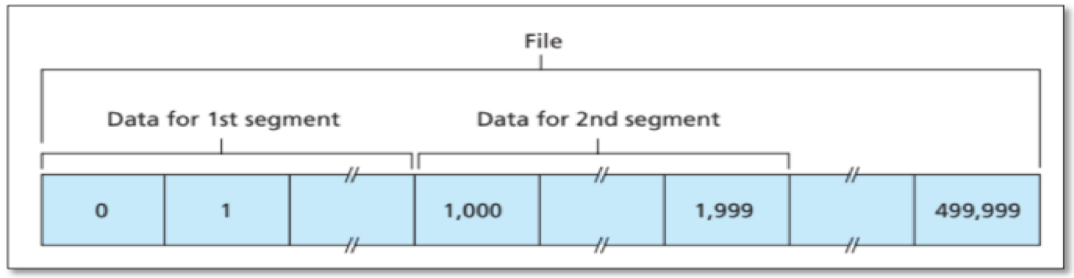
- byte stream "number" of first byte in segment's data

Acknowledgements:

- seq # of next byte expected from other side
- cumulative ACK

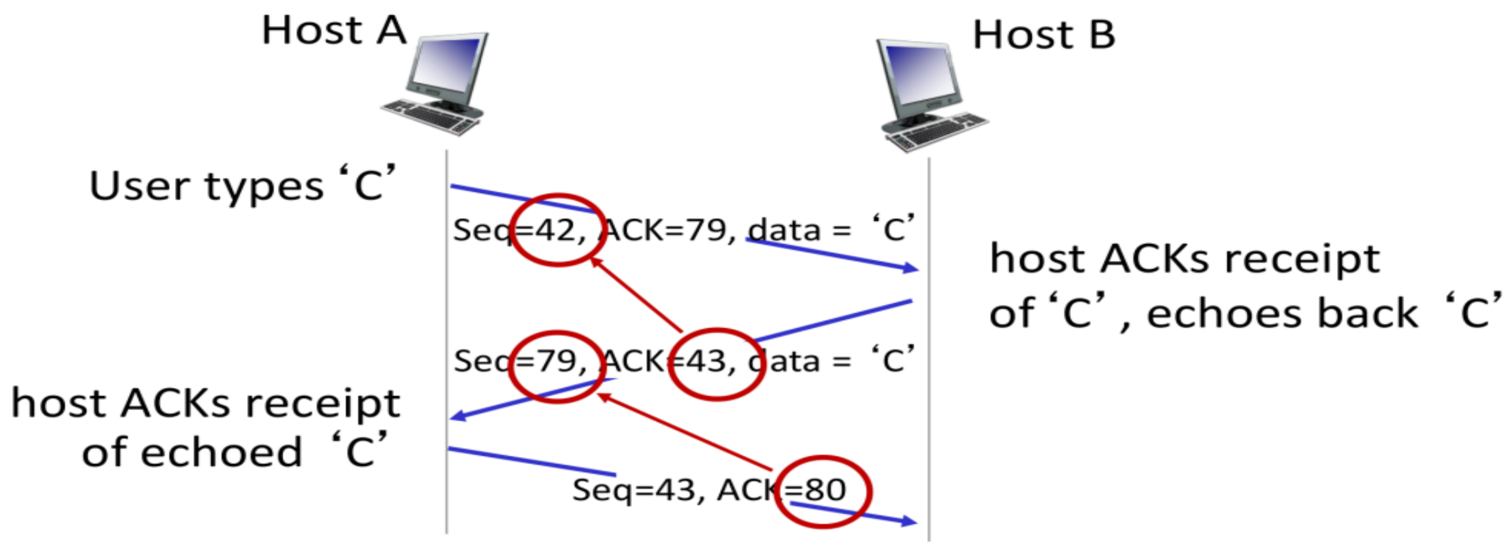
Q: how receiver handles out-of-order segments

- *A:* TCP spec doesn't say, - up to implementor



- File size: 500,000 bytes.
- MSS: 1,000 bytes
- TCP constructs 500 segments
- The first segment sequence number 0
- The second segment sequence number 1,000
- The third segment sequence number 2,000, and so on.

TCP sequence numbers, ACKs

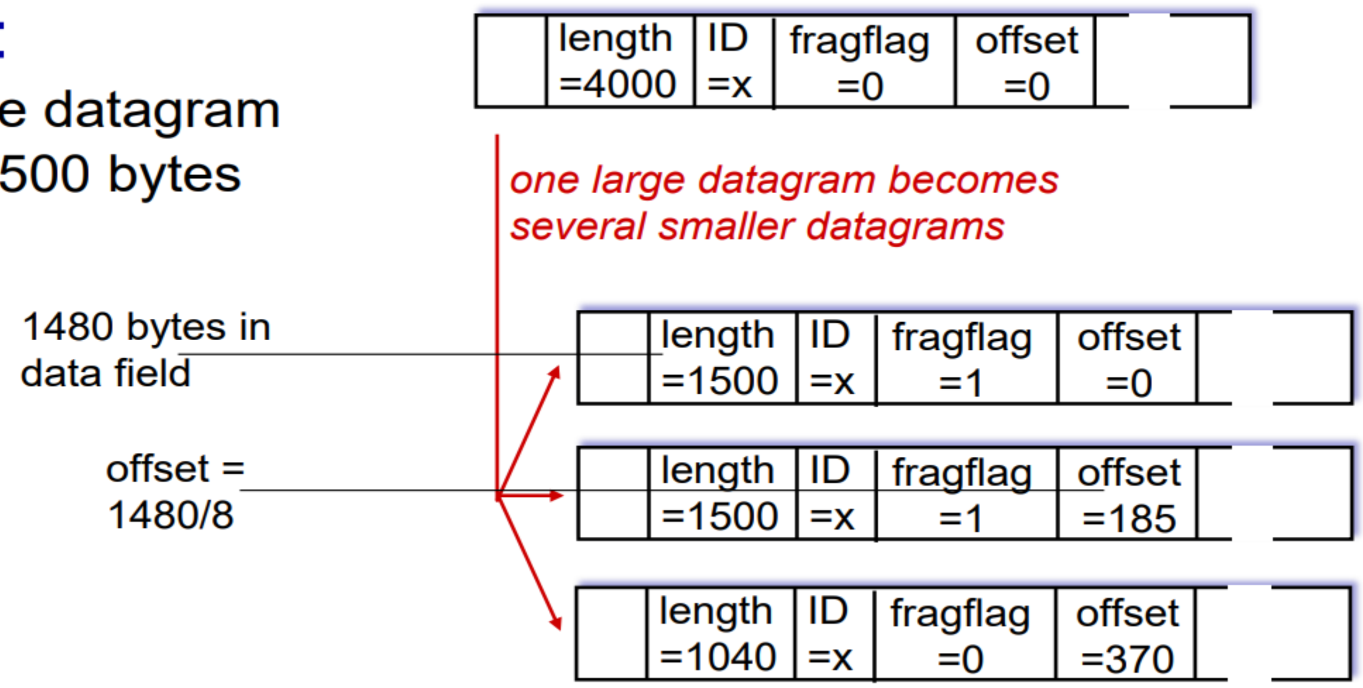


simple telnet scenario

IP Datagram: fragmentation/reassembly

example:

- 4000 byte datagram
- MTU = 1500 bytes

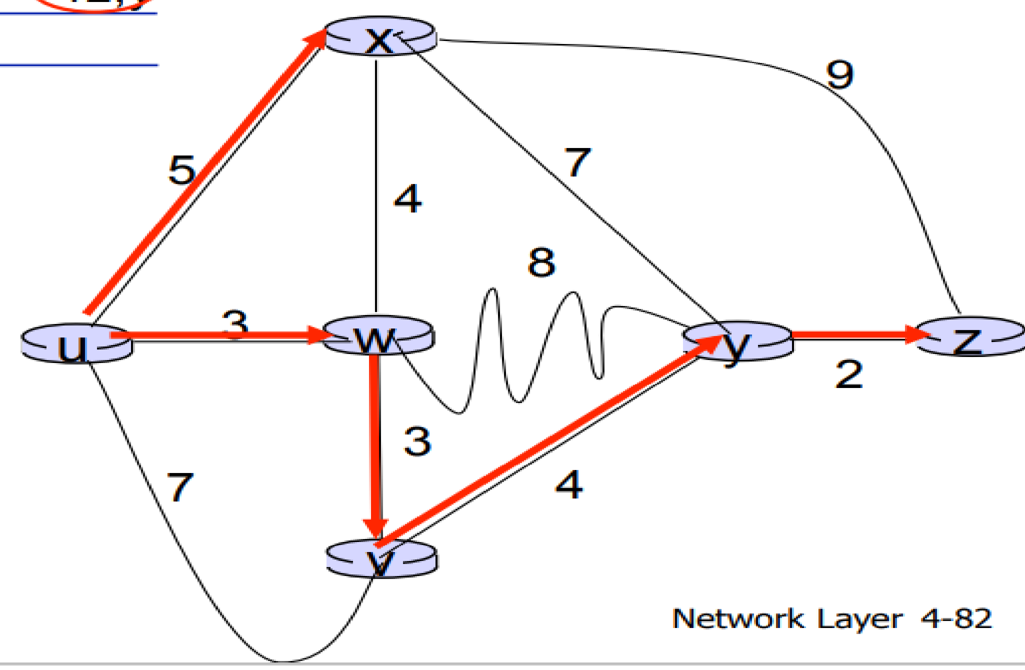


Dijkstra's algorithm: example

Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

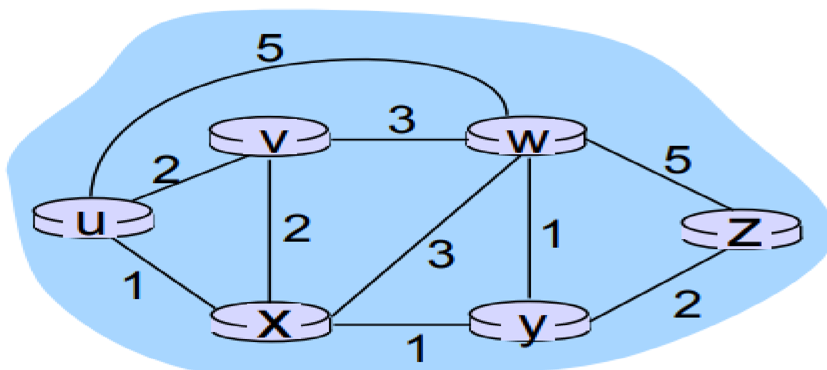
notes:

- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)



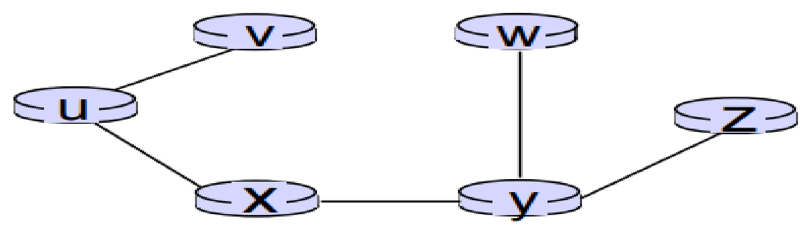
Dijkstra's algorithm: another example

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)

Distance vector algorithm

Bellman-Ford equation (dynamic programming)

let

$d_x(y)$:= cost of least-cost path from x to y

then

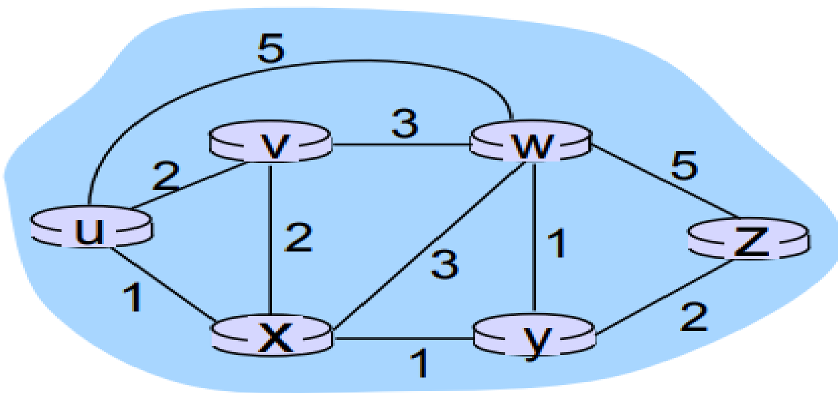
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

cost from neighbor v to destination y

cost to neighbor v

min taken over all neighbors v of x

Bellman-Ford example



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

B-F equation says:

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

node achieving minimum is next
hop in shortest path, used in forwarding table