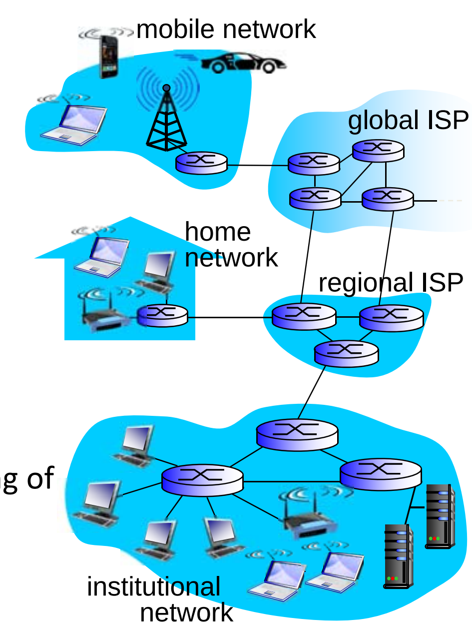
Network Architectures

- Reference
  - Chapter 01, *Computer Networking: A Top Down Approach,* 6/E, Jim Kurose, Keith Ross, Addison-Wesley
  - Adapted from part of the slides provided by the authors

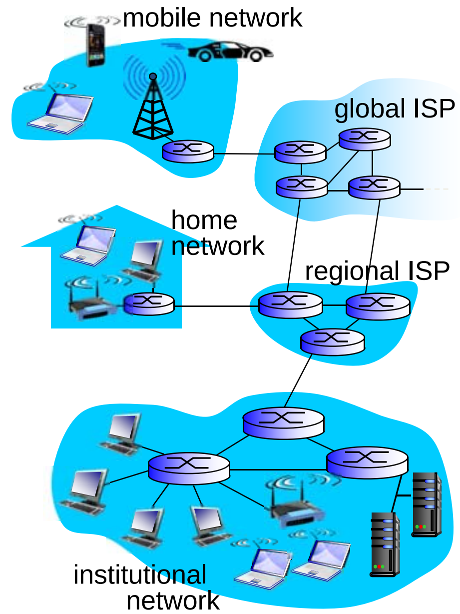
What's the Internet: "nuts and bolts" view

- millions of connected computing devices:
  - *hosts* = *end systems*
  - running *network apps*
- communication links
  - fiber, copper, radio, satellite
  - transmission rate: bandwidth
- Packet switches: forward packets (chunks of data)
  - routers and switches
- *Internet:* "network of networks"
  - Interconnected ISPs
- *protocols* control sending, receiving of msgs
  - e.g., TCP, IP, HTTP, Skype, 802.11
- *Internet standards*
  - RFC: Request for comments
  - IETF: Internet Engineering Task Force



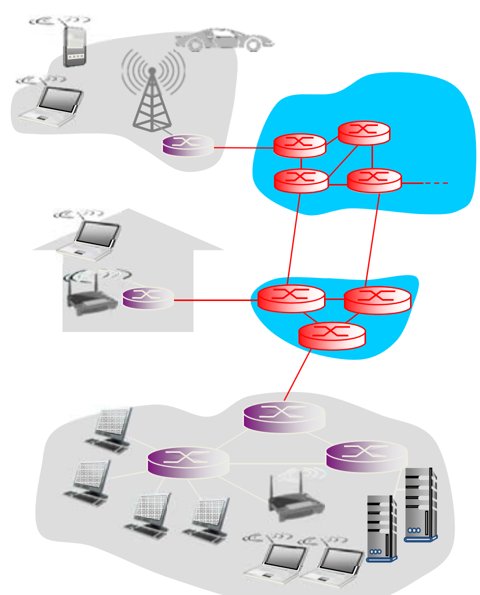
A closer look at network structure:

- *network edge:*
  - hosts: clients and servers
  - servers often in data centers
- *access networks, physical media:* wired, wireless communication links
- *network core:*
  - interconnected routers
  - network of networks

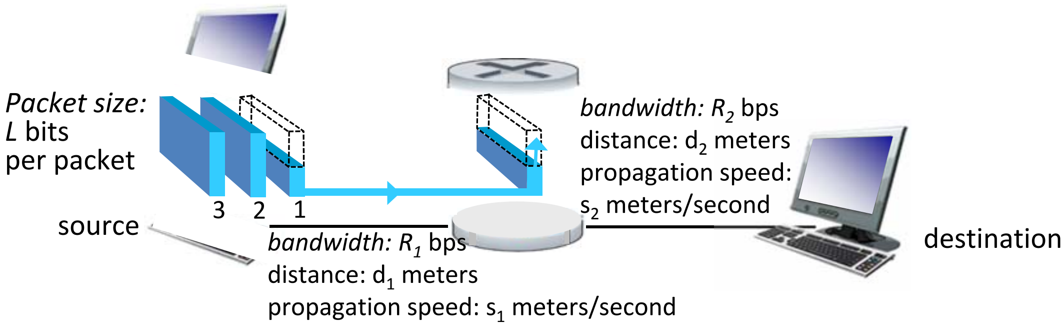


The network core: Packet-Switching

- mesh of interconnected routers
- packet-switching: hosts break application-layer messages into *packets*
  - forward packets from one router to the next, across links on path from source to destination
  - each packet transmitted at full link capacity

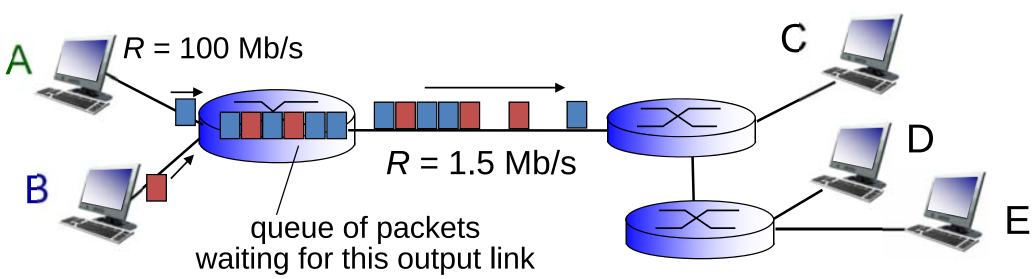


Packet-switching: store-and-forward



- One-hop example: 1 router and 2 links between source and destination
- bandwidth is also called transmission rate
  - bps: bit per second, 1 Byte = 8 bits
- transmission delay on a link = packet size / bandwidth
- propagation delay on a link = distance / propagation speed
- ***store and forward:*** entire packet must arrive at router before it can be transmitted on next link
- Link delay = transmission delay + propagation delay
- Assuming zero queuing delay and the packet switch processing delay is d_{proc} sec,
  end-to-end delay = $(L/R_1 + d_1/s_1 + d_{proc} + L/R_2 + d_2/s_2)$ seconds

Packet Switching: queueing delay, loss



queuing and loss:

- If arrival rate to an output link exceeds transmission rate of this output link:
  - packets will queue in a buffer, wait to be transmitted on this output link
  - packets can be dropped (lost) if memory (buffer) fills up
- queuing delay: the time that a packet waits in the queue for being transmitted
  - Example: when a packet arrives at the queue, one packet is halfway done being transmitted, and 5 packets have been in the queue, the packet size is 300 Bytes

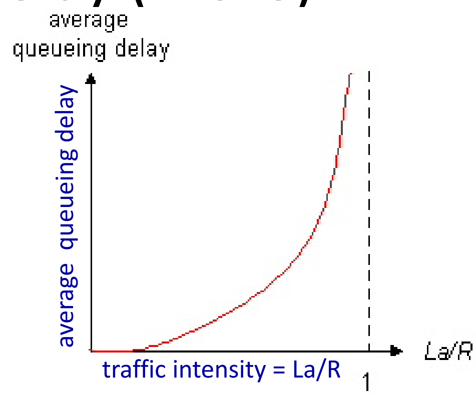
  queuing delay = $(0.5 \times 300 + 5 \times 300) \times 8$ bits / $(1.5 \times 10^6$ bits per second$) = 0.0088$ second
  - Normally, average queuing delay is interested by researchers

Queueing delay (more)

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

La/R ~ 0

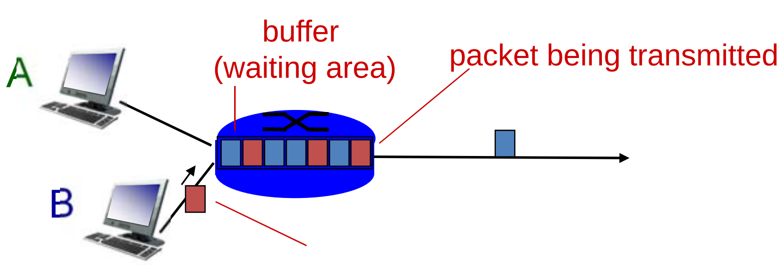
La/R -> 1



- Traffic Intensity = La/R
  - *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!

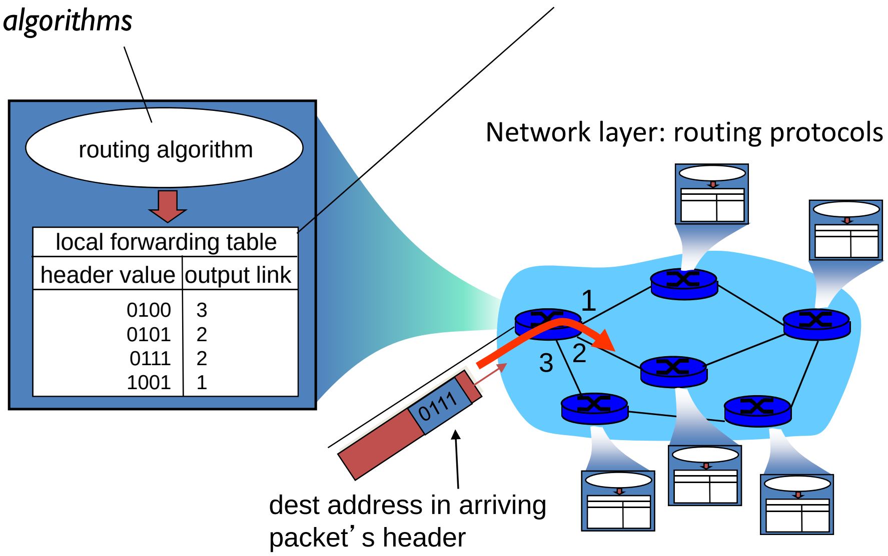
Packet loss

- queue (aka buffer) preceding link in buffer has finite capacity
- packet arriving to full queue dropped (aka lost)
- lost packet may be retransmitted by previous node, by source end system, or not at all

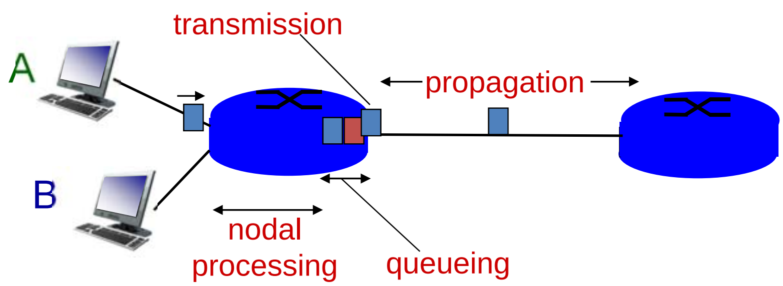


Two key network-core functions

- *routing:* determines source-destination route taken by packets
  - *routing algorithms*
- *forwarding:* move packets from router's input to appropriate router output



Four sources of packet delay



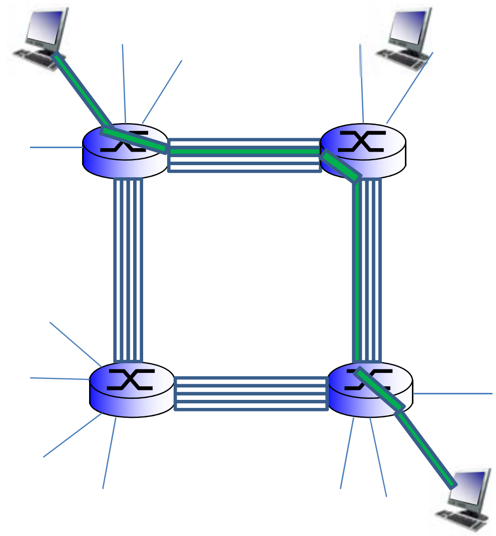
End-to-end delay = all d_{proc} + all d_{queue} + all d_{trans} + all d_{prop} along the path from source to destination

- 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_{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

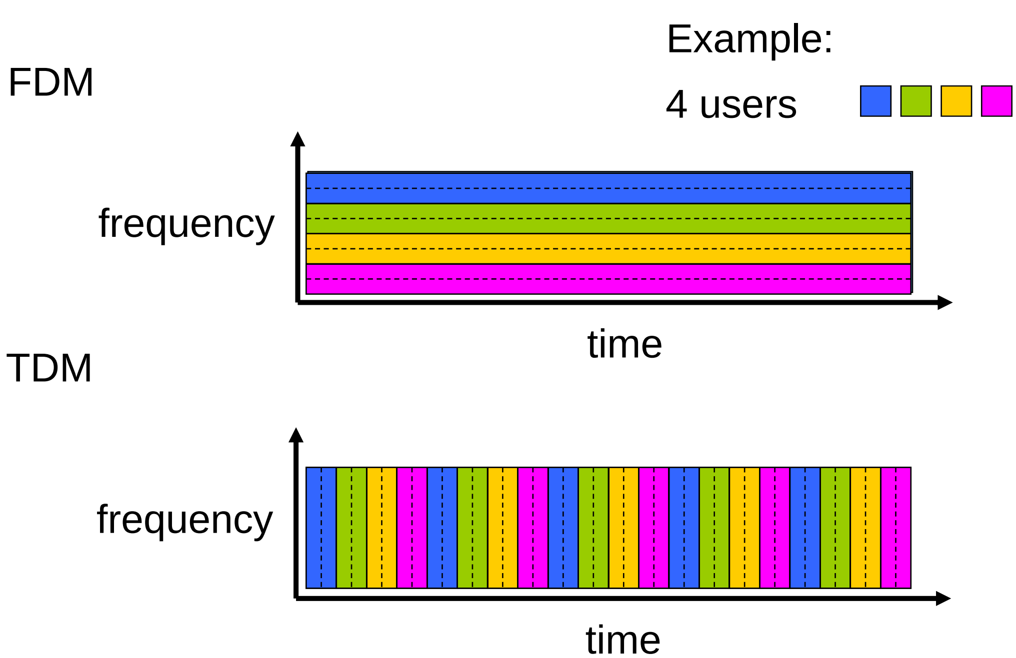
Alternative network core: circuit switching

end-end resources allocated to, reserved for "call" between source & dest:

- In diagram, each link has four circuits.
  - call gets 2nd circuit in top link and 1st circuit in right link.
- dedicated resources: no sharing
  - circuit-like (guaranteed) performance
- circuit segment idle if not used by call *(no sharing)*
- Commonly used in traditional telephone networks



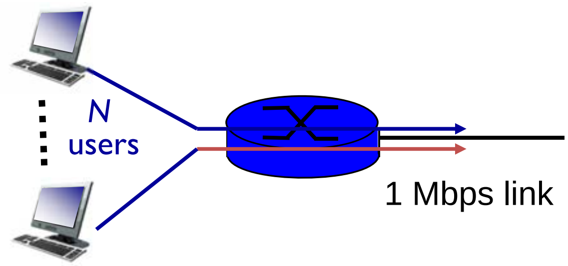
Circuit switching: FDM versus TDM



Packet switching versus circuit switching

packet switching allows more users to use network!

Example:
- 1 Mb/s link
- each user:
  - 100 kb/s when "active"
  - active 10% of time (active probability p = 0.1)



- *circuit-switching:*
  - 10 users
- *packet switching:*
  - Combination, $\binom{N}{i}$, can be calculated using combin(N, i) in excel if N and i are known
  - with N users in total, probability that exact ly M users are active = $\binom{N}{M}p^M(1-p)^{(N-M)}$
  - with N users in total, probability that <= M (M or less) users are active = $\sum_{i=0}^{M}\binom{N}{i}p^i(1-p)^{(N-i)}$
  - with N users in total, probability that > M (more than M) users are active = $1-\sum_{i=0}^{M}\binom{N}{i}p^i(1-p)^{(N-i)}$
  - E.g., with 35 users, probability > 10 active at same time is 0.000424

Packet switching versus circuit switching (Cont.)

is packet switching a "slam dunk winner?"

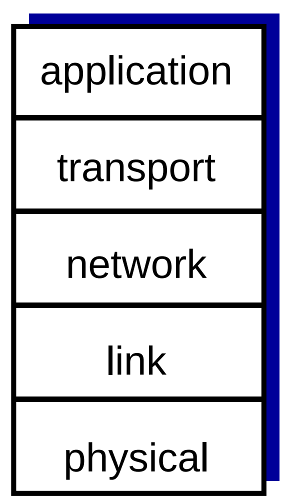
- great for bursty data
  - resource sharing
  - simpler, no call setup
- excessive congestion possible: packet delay and loss
  - protocols needed for reliable data transfer, congestion control
- *Q:* How to provide circuit-like behavior?
  - QoS (Quality of Service)
  - bandwidth guarantees needed for audio/video apps
  - still an unsolved problem (chapter 7)

Protocol "layers"

- *Networks are complex, with many "pieces":*
  - routers
  - links of various media
  - applications
  - protocols
  - hardware, software
- Why layering? dealing with complex systems:
  - explicit structure allows identification, relationship of complex system's pieces
    - layered *reference model* for discussion
  - modularization eases maintenance, updating of system
    - change of implementation of layer's service transparent to rest of system
  - layering considered harmful?

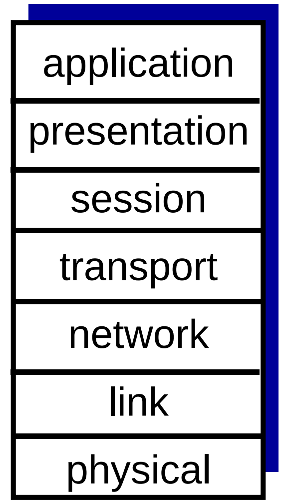
Internet protocol stack

- *application:* supporting network applications
  - FTP, SMTP, HTTP
- *transport:* process-process data transfer
  - TCP, UDP
- *network:* routing of datagrams from source to destination
  - IP, routing protocols
- *link:* data transfer between neighboring network elements
  - Ethernet, 802.111 (WiFi), PPP
- *physical:* bits "on the wire"



ISO/OSI reference model

- *presentation:* allow applications to interpret meaning of data, e.g., encryption, compression, machine-specific conventions
- *session:* synchronization, checkpointing, recovery of data exchange
- Internet stack "missing" these layers!
  - these services, *if needed,* must be implemented in application
  - Most of the time, not really needed



Encapsulation