

Switching & Multiplexing

- Network is a **shared** resource
 - Provide services for many people at same time
 - Carry bits/information for many people at same time
- How do we do it?
 - **Switching**: how to deliver information from point A to point B?
 - **Multiplexing**: how to share resources among many users

Think about postal service and telephone system!

Switching and multiplexing are closely related!

Switching Strategies

- Circuit switching
 - set up a dedicated route (“circuit”) first
 - carry all bits of a “conversation” on one circuit
 - original telephone network
 - Analogy: railroads and trains/subways
- Packet switching
 - divide information into small chunks (“packets”)
 - each packet delivered independently
 - “store-and-forward” packets
 - Internet

(also Postal Service, but they don’t tear your mail into pieces first!)

 - Analogy: highways and cars
- Pros and Cons?
 - think taking subways vs. driving cars, during off-peak vs. rush hours!

Circuit Switching Analogy: railroad and train

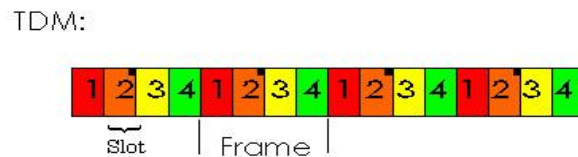
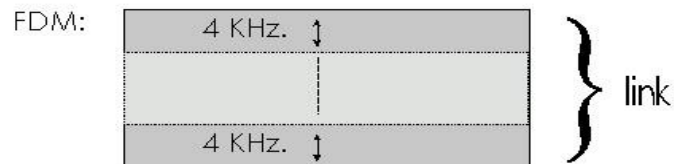


Packet Switching Analogy: Hiahway and cars



Circuit Switching

- network resources
(e.g., bandwidth)
divided into "pieces"
- pieces allocated to calls
 - resource piece **idle** if not used by owning call
(no sharing)



All slots labelled  are dedicated to a specific sender-receiver pair.

- dividing link bandwidth into "pieces"
 - ❖ frequency division
 - ❖ time division
 - ❖ code division

□ Trivia Q:

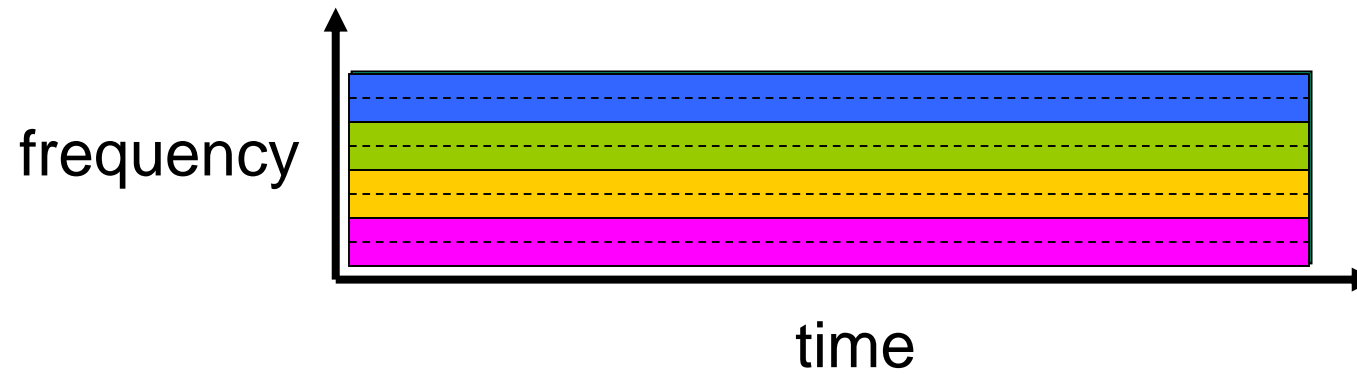
You must have heard of the term "CDMA" (think the company Qualcomm, for which it is most associated with), what does "CD" in CDMA stand for?

Circuit Switching: FDM and TDM

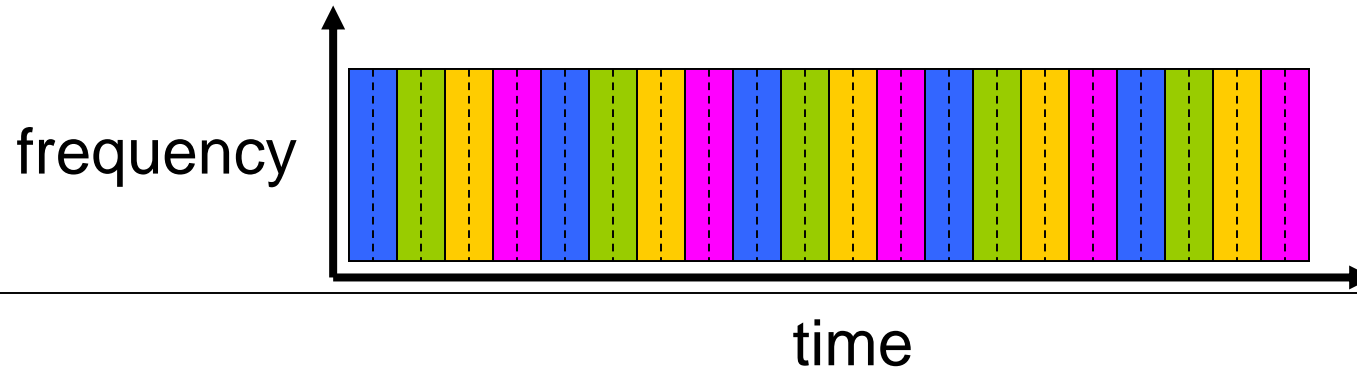
FDM

Example:

4 users



TDM



Numerical example

- How long does it take to send a file of 640,000 bits from host A to host B over a circuit-switched network?
 - All links are 1.536 Mbps
 - Each link uses TDM with 24 slots/sec
 - 500 msec to establish end-to-end circuit

Let's work it out!

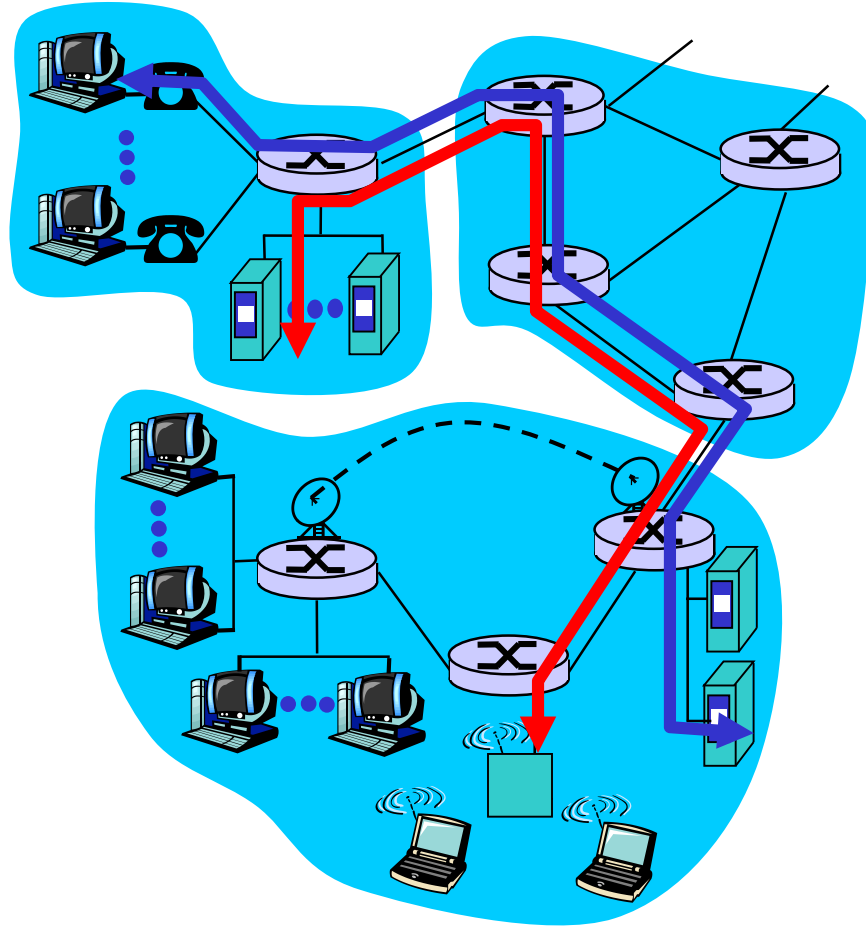
10.5 seconds

Networks with Circuit Switching

e.g., conventional (fixed-line) telephone networks

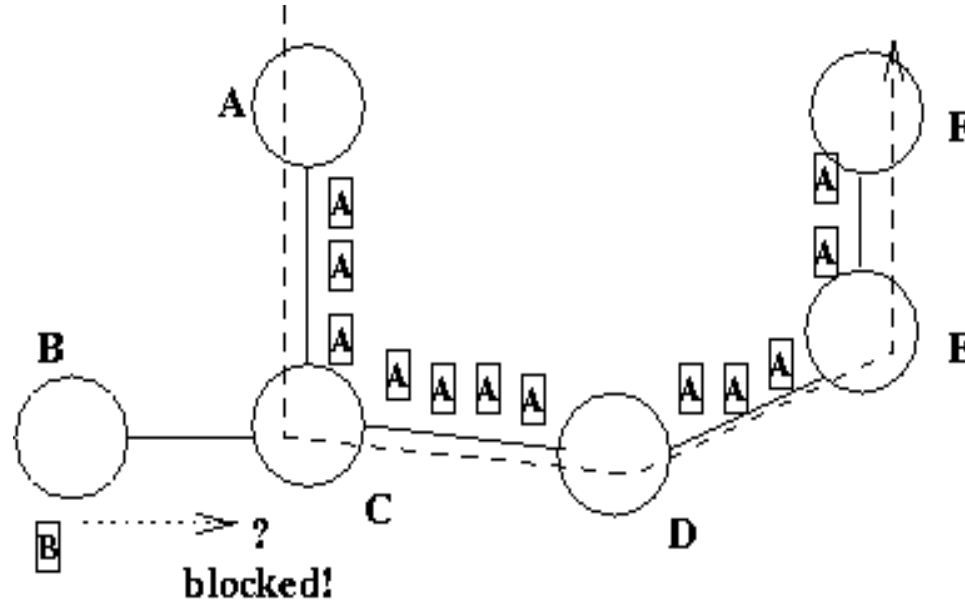
End-end resources
reserved for "call"

- link bandwidth, switch capacity
- dedicated resources:
no sharing
- circuit-like
(guaranteed)
performance
- call setup required



Circuit Switched Networks

- All resources (e.g. communication links) needed by a call dedicated to that call for its duration
 - Example: telephone network
 - Call blocking when all resources are used



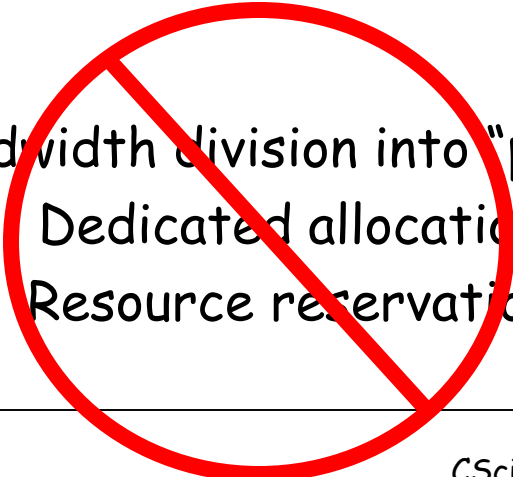
Note: if one circuit/link, A-to-F call blocks B-to-E call

Packet Switching

Each end-end "data stream"
divided into *packets*

- users A, B packets share network resources
- each packet uses full link bandwidth
- resources used *as needed*

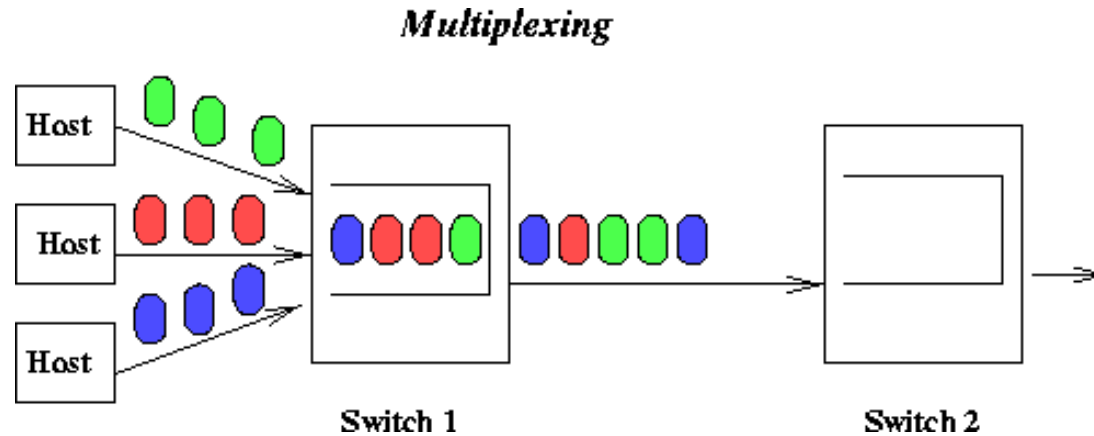
Bandwidth division into "pieces"
Dedicated allocation
Resource reservation



resource contention:

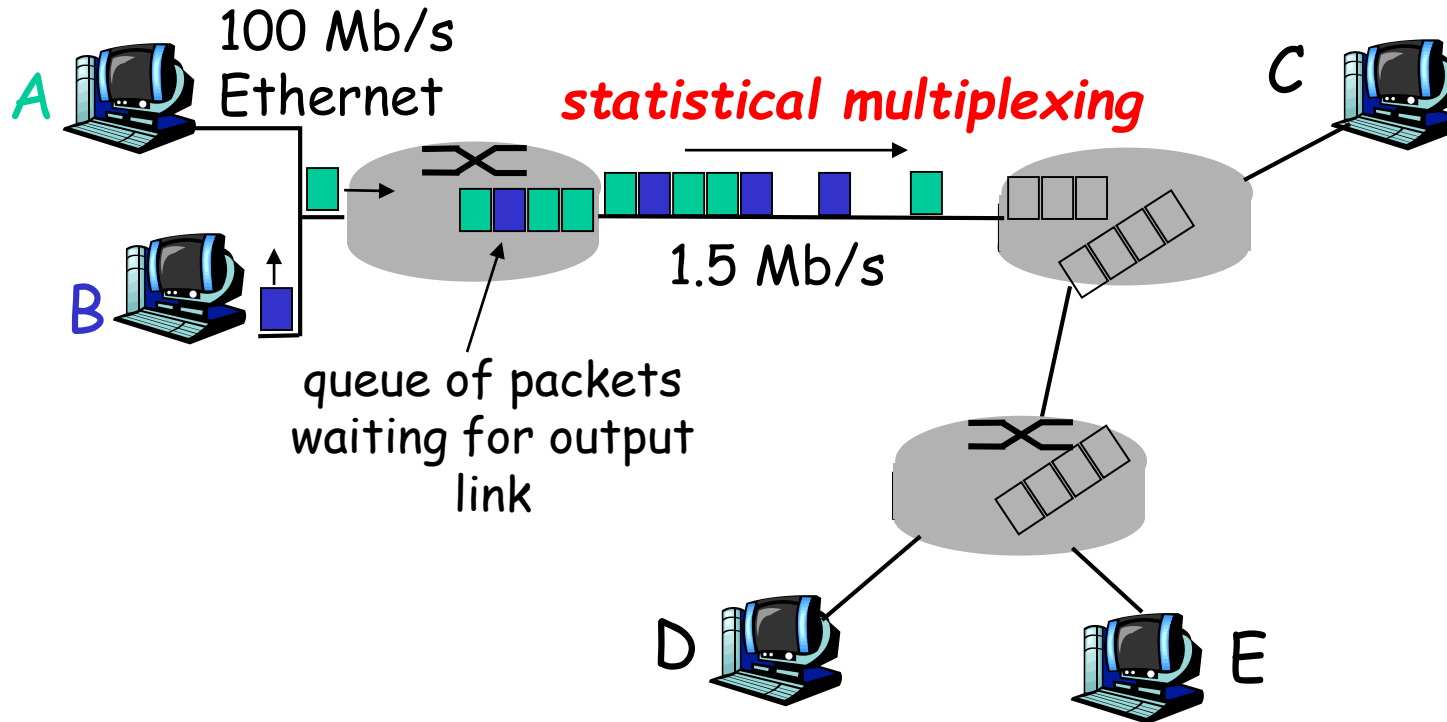
- ❑ aggregate resource demand can exceed amount available
- ❑ congestion: packets queue, wait for link use
- ❑ store and forward: packets move one hop at a time
 - ❖ Node receives complete packet before forwarding
 - ❖ *Packets may suffer delay or losses!*

Statistical Multiplexing



- Time division, but **on demand** rather than fixed
- Reschedule link on a per-packet basis
- Packets from different sources interleaved on the link
- Buffer packets that are **contending** for the link
- Buffer buildup is called **congestion**
- This is **packet switching**, used in computer networks

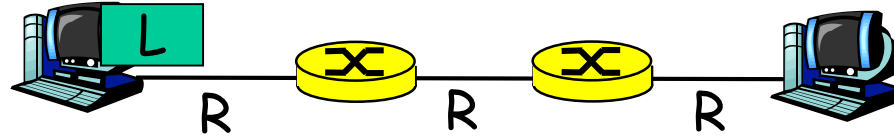
Packet Switching: Statistical Multiplexing



Sequence of A & B packets does not have fixed pattern,
shared on demand □ **statistical multiplexing**.

TDM: each host gets same slot in revolving TDM frame.

Packet-switching: store-and-forward



- Takes L/R seconds to transmit (push out) packet of L bits on to link of R bps
- Entire packet must arrive at router before it can be transmitted on next link: *store and forward*
- delay = $3L/R$ (assuming zero propagation delay)

Example:

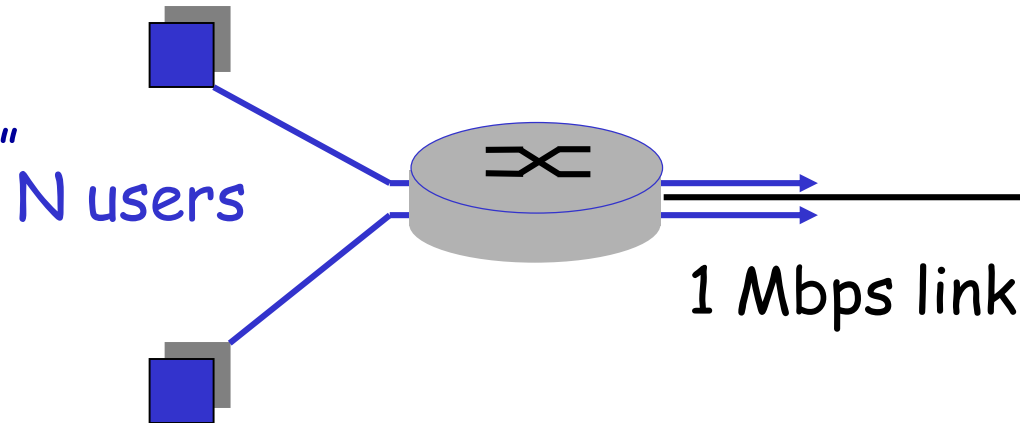
- $L = 7.5$ Mbits
- $R = 1.5$ Mbps
- delay = ? 15 sec

} more on delay later ...

Packet switching versus circuit switching

Packet switching allows more users to use network!

- 1 Mb/s link
- each user:
 - 100 kb/s when "active"
 - active 10% of time



- circuit-switching:
 - 10 users
- packet switching:
 - with 35 users,
probability > 10 active
less than .0004

Q: how did we get value 0.0004?

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

Circuit Switching vs Packet Switching

Item	Circuit-switched	Packet-switched
Dedicated “copper” path	Yes	No
Bandwidth available	Fixed	Dynamic
Potentially wasted bandwidth	Yes	No (not really!)
Store-and-forward transmission	No	Yes
Each packet/bit always follows the same route	Yes	Not necessarily
Call setup	Required	Not Needed
When can congestion occur	At setup time	On every packet
Effect of congestion	Call blocking	Queuing delay

Packet switching vs. circuit switching

Is packet switching a "slam dunk winner?"

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

Q: human analogies of reserved resources (circuit switching) versus on-demand allocation (packet-switching)?

What's so special about the Internet?

- Internet is based on the notion of “**packet switching**”
 - enables **statistical multiplexing**
 - better utilization of network resources for transfer of “bursty” data traffic
- Internet's key organizational/architectural principle: “smart” end systems + “dumb” networks
 - architecture: functional division & function placement
 - **hourglass Internet architecture**: enables diverse applications and accommodates evolving technologies
 - “**dumb**” **network (core)**: simple packet-switched, store-forward, connectionless “datagram” service, with core functions: global addressing, routing & forwarding
 - “**smart**” **end systems/edges**: servers, PCs, mobile devices, ...; diverse and ever-emerging new applications!