Network Overview (cont’d)

- Circuit vs. Packet Switching
- Best Effort Internet Model

Circuit Switching
(e.g., Phone Network)

- Step 1: Source establishes \textit{connection} to destination
  - Connection setup signaling from src to dst
  - Routers en route store path connection info (state)
- Step 2: Source sends data over the connection
  - (Sometimes packets are called \textit{cells - cells are fixed size})
  - No dst address in packets, since routers know path
- Step 3: When done, source tears down connection

Circuit Switching Diagram: [Diagram placeholder]
Virtual Circuits (VC)

- Need call setup/teardown for each call before data can flow
- Each packet carries VC identifier (not destination host address)
- Every router on src-dst path maintains “state” for each passing connection
- Link, router resources (bandwidth, buffers) may be allocated to VC
- Example network: ATM (Asynchronous Transfer Mode)

VC Implementation

A VC consists of:
1. Path from source to destination
2. VC numbers, one number for each link along path
3. Entries in forwarding tables in routers along path

- Packet belonging to VC carries a VC number in the header
- VC number may change (and usually does) on each link.
  - VCs are negotiated between neighboring routers

Forwarding Tables

Routers maintain connection state information!

What is the forwarding state in R2 for C1?
Advantages of Circuit Switching

- Easy to provide performance guarantees
  - Bandwidth can be reserved along the entire path
  - Fixed path means virtually constant latency

- Simple abstraction
  - Reliable communication channel between hosts
  - No worries about lost or out-of-order packets

- Simple forwarding
  - Based on time slot, frequency or label
  - No need for complex packet header
  - Low per-packet overhead

Disadvantages of Circuit Switching

- With reservations, wasted BW (need peak BW)
  - Idle resources during silent periods
  - Unable to achieve gains from statistical multiplexing

- Blocked connections
  - Connection refused when resources are not sufficient
  - Unable to offer “okay” service to everybody

- Connection set-up delay
  - No communication until a connection is set up (1 RTT)
  - Unable to avoid extra latency for small data transfers

- Network state
  - Network nodes must store per-connection state
  - Unable to avoid per-connection storage and state

Packet Switching (e.g., Internet)

- Messages divided into globally addressable packets (Datagrams)
  - Each packet’s header contains a dst address

- Packets may travel separately through network
  - Packet forwarding based on the header
  - Network nodes may store packets temporarily

- Destination reconstructs the message
IP Service Model: Why Packet Switching?

- In one word: Flexibility!
- Data traffic is bursty
  - Remote login, email, video, voice, etc.
- Packets don’t waste reserved bandwidth
  - No traffic exchanged during idle periods
- Packets better for multiplexing
  - Different transfers share access to same links
- Packets can be delivered by almost anything
  - Best effort service
  - RFC 2549: IP over Avian Carriers (aka birds)
- … still, packet switching can be inefficient
  - Extra header bits

Network Architecture: Internet vs. POTS*

- There is a fundamental architectural difference between Internet and telephone network
- POTS: Intelligent network, dumb terminals
  - Reliable, in-sequence, guaranteed delivery (bandwidth and delay)
- Internet: Dumb network, intelligent endpoints
  - Best effort delivery (unreliable, packets may arrive out of sequence and duplicated, no bandwidth or delay guarantees)

(*POTS: Plain Old Telephone System)

IP Service Model: Why Best-Effort?

- Flexibility: Network does not dictate applications
- IP means never having to say you’re sorry…
  - Don’t need to reserve bandwidth and memory
  - Don’t need to do error detection & correction
  - Don’t need to remember from one packet to next
  - Can’t get any simpler than that!
- Easier to survive failures
  - Transient disruptions are okay during failover
- … but, applications do want efficient, accurate transfer of data in order, in a timely fashion
  - IP pushes these to the higher layers
IP Service: Is Best-Effort Enough?

- No error detection or correction
  - Higher-level protocol can provide error checking
- Successive packets may not follow the same path
  - Usually not a problem as long as packets get there
- Packets can be delivered out-of-order
  - Number packets so they can be put back in order
- Packets may be lost or arbitrarily delayed
  - Sender can send the packets again (if desired)
- No network congestion signal (beyond “drop”)
  - Sender can slow down in response to loss or delay (but this is a really hard problem..)

To Think About..

- Think about the diametrically opposing architectural difference:
  - Smart Network, dumb endpoints, vs. Stupid Network, Intelligent Endpoints.
- Which one provides more flexibility?
- Which one allows more future innovation?
- What do you think about the KISS principle? (Keep It Simple, Stupid)