What’s the Internet: “nuts and bolts” view

- millions of connected computing devices: *hosts, end-systems*
  - pc’s workstations, servers
  - PDA’s phones, toasters running *network apps*
- *communication links*
  - fiber, copper, radio, satellite
- *routers*: forward packets (chunks) of data thru network
What’s the Internet: “nuts and bolts” view

- **protocols**: control sending, receiving of messages
  - e.g., TCP, IP, HTTP, FTP, PPP
- **Internet**: “network of networks”
  - loosely hierarchical
  - public Internet versus private intranet
- **Internet standards**
  - RFC: Request for comments
  - IETF: Internet Engineering Task Force
What’s the Internet: a service view

- Communication infrastructure enables distributed applications:
  - WWW, email, games, e-commerce, database, voting,
  - more?
- Communication services provided:
  - connectionless
  - connection-oriented

What’s a protocol?

**Human protocols:**
- “what’s the time?”
- “I have a question”
- introductions

... specific msgs sent
... specific actions taken when msgs received, or other events

**Network protocols:**
- machines rather than humans
- all communication activity in Internet governed by protocols

*Protocols define format, order of msgs sent and received among network entities, and actions taken on msg transmission, receipt*
A closer look at network structure:

- **network edge:** applications and hosts
- **network core:**
  - routers
  - network of networks
- **access networks, physical media:** communication links

The network edge:

- **end systems (hosts):**
  - run application programs
  - e.g., WWW, email
  - at “edge of network”

- **client/server model**
  - client host requests, receives service from server
  - e.g., WWW client (browser)/ server; email client/server

- **peer-peer model:**
  - host interaction symmetric
  - e.g.: teleconferencing
Network edge: connection-oriented service

**Goal:** data transfer between end sys.
- **handshaking:** setup (prepare for) data transfer ahead of time
  - Hello, hello back human protocol
  - *set up “state”* in two communicating hosts
- TCP - Transmission Control Protocol
  - Internet’s connection-oriented service

**TCP service** [RFC 793]
- *reliable, in-order* byte-stream data transfer
  - loss: acknowledgements and retransmissions
- **flow control:**
  - sender won’t overwhelm receiver
- **congestion control:**
  - senders “slow down sending rate” when network congested

Network edge: connectionless service

**Goal:** data transfer between end systems
- same as before!
- **UDP** - User Datagram Protocol [RFC 768]:
  - Internet’s connectionless service
  - unreliable data transfer
  - no flow control
  - no congestion control

**App’s using TCP:**
- HTTP (WWW), FTP (file transfer), Telnet (remote login), SMTP (email)

**App’s using UDP:**
- streaming media, teleconferencing, Internet telephony
The Network Core

- mesh of interconnected routers
- *the fundamental question:* how is data transferred through net?
  - circuit switching: dedicated circuit per call: telephone net
  - packet-switching: data sent thru net in discrete “chunks”

Network Core: Circuit Switching

End-end resources reserved for “call”

- link bandwidth, switch capacity
- dedicated resources: no sharing
- circuit-like (guaranteed) performance
- call setup required
Network Core: 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)
• dividing link bandwidth into “pieces”
  • frequency division
  • time division

suppose 8000 frames/s with 8bits slots
how much BW per connection?

Network Core: Packet Switching

each end-end data stream divided into packets
• user 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
  • transmit over link
  • wait turn at next link
Network Core: Packet Switching

Packet-switching versus circuit switching: human restaurant analogy
• other human analogies?
Packet switching versus circuit switching
Packet switching allows more users to use network!

- 1 Mbit link
- each user:
  - 100Kbps when “active”
  - active 10% of time
- circuit-switching:
  - 10 users
- packet switching:
  - with 35 users, probability > 10 active less that .004

Packet switching versus circuit switching
Is packet switching a “slam dunk winner?”

- Great for bursty data
  - resource sharing
  - 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 open problem
Packet-switched networks: routing

- **Goal**: move packets among routers from source to destination
  - we’ll study several path selection algorithms (chapter 4)
- **datagram network**:
  - *destination address* determines next hop
  - routes may change during session
  - analogy: driving, asking directions
- **virtual circuit network**:
  - each packet carries tag (virtual circuit ID), tag determines next hop
  - fixed path determined at *call setup time*, remains fixed thru call
  - routers maintain per-call state

**virtual circuits**
virtual circuits

link 1 vci 5 link 2 vci 9
link 3 vci 9 link 5 vci 7
link 1 vci 7 link 3 vci 8
link 1 vci 7 link 3 vci 8
Delay in packet-switched networks

packets experience **delay** on end-to-end path

- **four** sources of delay at each hop
  - nodal processing:
    - check bit errors
    - determine output link
  - queueing
    - time waiting at output link for transmission
    - depends on congestion level of router

Transmission delay:
- \( R = \) link bandwidth (bps)
- \( L = \) packet length (bits)
- time to send bits into link = \( \frac{L}{R} \)

Propagation delay:
- \( d = \) length of physical link
- \( s = \) propagation speed in medium (~\(2 \times 10^8\) m/sec)
- propagation delay = \( \frac{d}{s} \)

Note: \( s \) and \( R \) are **very different quantities**!
Queueing delay (revisited)

- $R$ = link bandwidth (bps)
- $L$ = packet length (bits)
- $a$ = average packet arrival rate

Traffic intensity = $La/R$

- $La/R \sim 0$: average queuing delay small
- $La/R \rightarrow 1$: delays become large
- $La/R > 1$: more “work” arriving than can be serviced, average delay infinite!