Internet architecture

Portland State
Computer Science

Overview

- Packet switching over circuit switching
- End-to-end principle and "Hourglass" design
- Layering of functionality

Packet switching vs. circuit switching

- Analogy
  - Ride sharing vehicles vs. privately owned vehicles
- Zipcar, car2go, Lime/Bird/Skip (packet-switching)
  - Many users share a single car or scooter
  - Large demand causes users to delay usage
  - Car or scooter more efficiently used
- Privately owned vehicles (circuit-switching)
  - Single user
  - Guaranteed access for user
  - Vehicle not used as efficiently

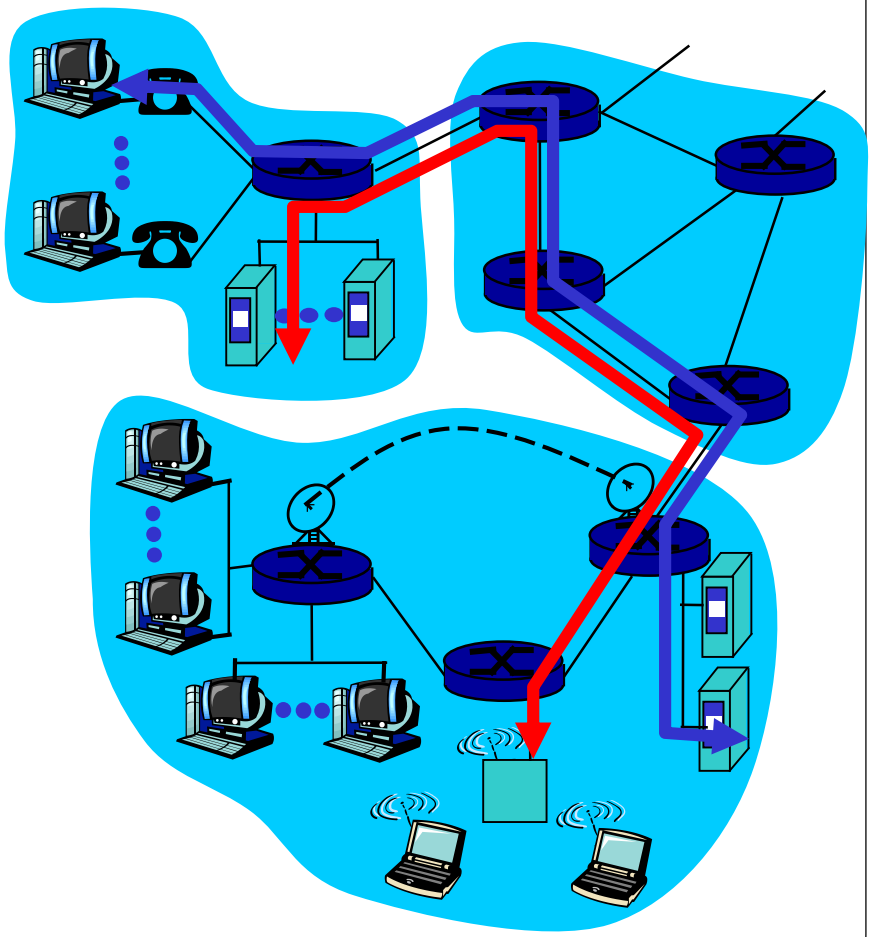
What is this?


gettyimages
Creative
157678264


NORTH AND SOUTH AMERICA
ARGENTINA • BAHAMAS • BERMUDA • BRAZIL • CANADA • CHILE • COLOMBIA • COSTA RICA • CUBA
DOMINICAN REPUBLIC • GUATEMALA • HONDURAS • MEXICO • NICARAGUA • PANAMA • PARAGUAY • PERU
URUGUAY • VENEZUELA
LONDON
PARIS

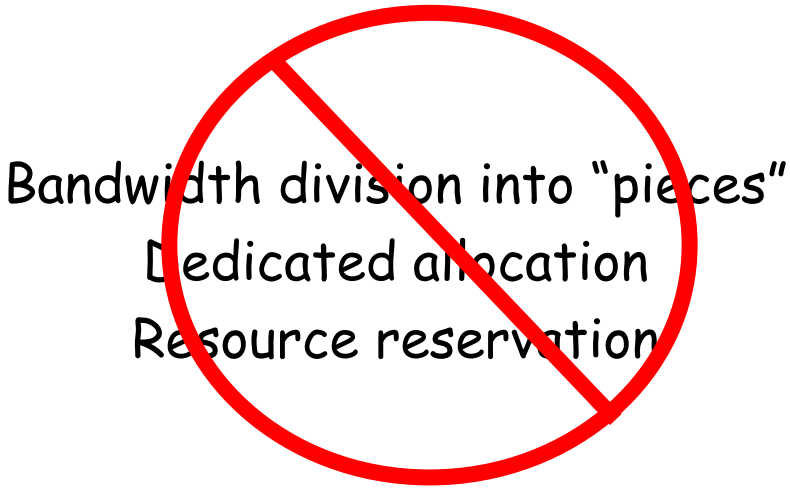
Circuit Switching

- Example
  - Phone network (pre-cellular)
- End-end network resources divided into "pieces" and reserved for call
  - link bandwidth, switch capacity
  - resource piece idle if not used by owning call
    - dedicated resources: no sharing
- Guaranteed performance
- Call setup and admission control required



Packet Switching

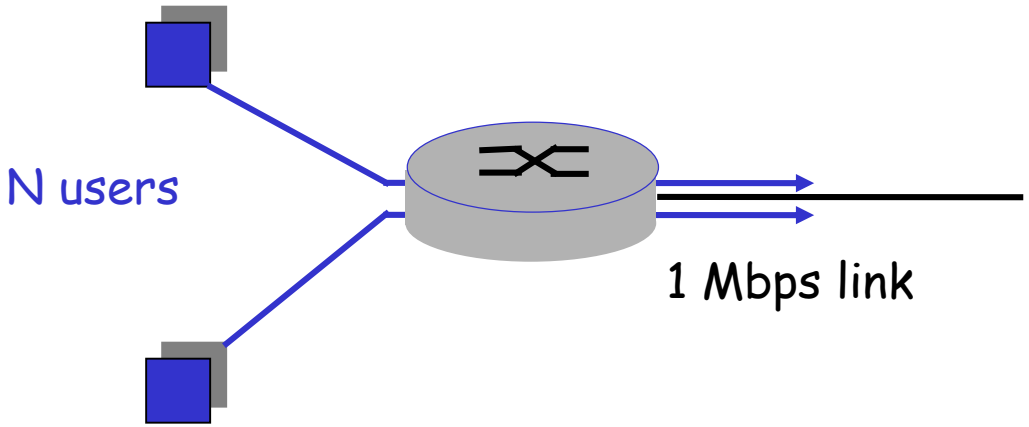
- Data divided into packets (Kleinrock 1960)
  - Packets from users share network resources
  - Each packet uses full link bandwidth
  - Packets stored and forwarded one hop at a time
  - Resources used as needed



- But…congestion possible
  - aggregate resource demand can exceed amount available
  - packets queue, wait for link use

Packet switching versus circuit switching

- N users over 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
  - Packet switching allows more users to use network
  - "Statistical multiplexing gain"
    - The basis for the cloud
    - Amazon with an enormous cluster to handle Christmas season (active < 10% of the year)



Packet switching versus circuit switching

Is packet switching a "slam dunk winner?"

- Great for bursty data
  - resource sharing
  - simpler, no call setup
- Bad for applications with hard resource requirements
  - Excessive congestion: packet delay and loss
  - Need protocols and applications that can deal with packet loss/congestion
- Basis for the Internet

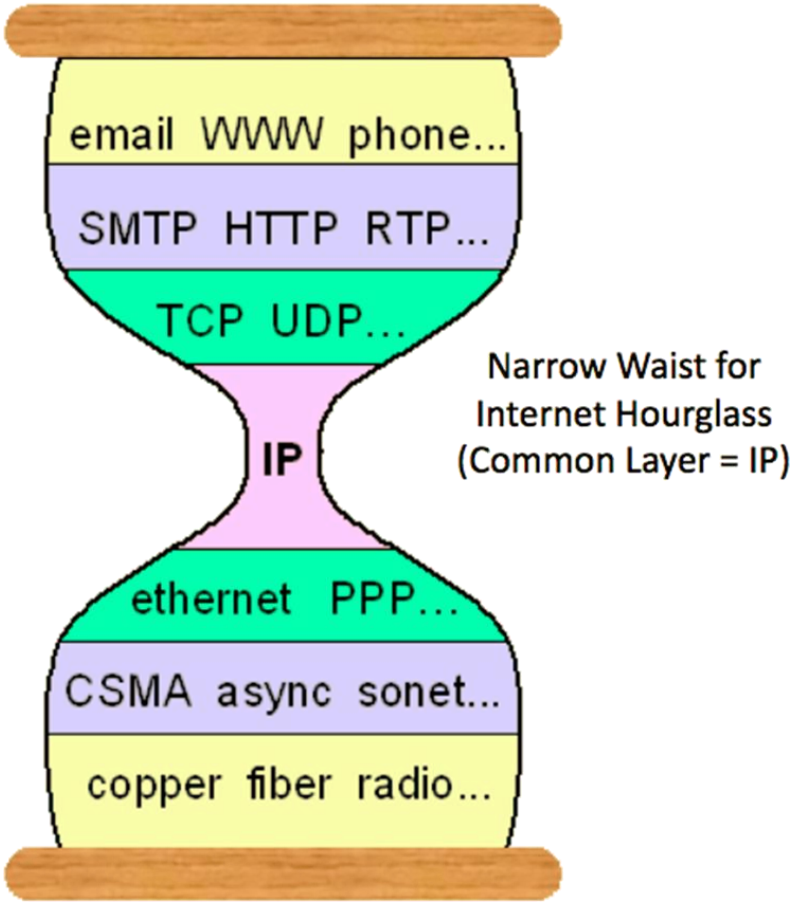
Overview

- Packet switching over circuit switching
- End-to-end principle and "Hourglass" design
- Layering of functionality

End-to-end principle and Hourglass design

- One, simple protocol to run it all

  "Perfection is achieved not when there is nothing more to add, but when there is nothing left to take away" -- Antoine de Saint-Exupery



End-to-end principle

- Where to put the functionality?
  - In the network? At the edges?
- End-to-end functions best handled by end-to-end protocols
  - Network provides basic service: data transport
  - Intelligence and applications located in or close to devices at the edge
- Leads to innovation at the edges
  - Phone network: dumb edge devices, intelligent network
  - Internet: dumb network, intelligent edge devices

Leads to Hourglass design

- Only one protocol at the Internet level
  - Minimal required elements at narrowest point
- IP – Internet Protocol (RFC 791 and 1812)
  - Unreliable datagram service
  - Addressing and connectionless connectivity
  - Like the post office of old!

Hourglass design of IP

- Simplicity allowed fast deployment of multi-vendor, multi-provider public network
  - Ease of implementation
  - Limited hardware requirements (important in 1970s)
  - Rapid development leads to eventual economies of scale
- Designed independently of hardware
  - No link-layer specific functions
  - Hardware addresses decoupled from IP addresses
  - IP header contains no data/physical link specific information (e.g. Ethernet, WiFi, 5G, etc.)
  - Allows IP to run over any fabric
- Translation to the cloud
  - What technology might allow applications to run on any cloud provider (e.g. AWS, GCP, Azure)?
  - Possible answer later on…

End-to-end principle, hourglass design

- The good
  - Basic network functionality allowed for extremely quick adoption and deployment using simple devices
- The bad
  - New network features and functionality are impossible to deploy, requiring widespread adoption within the network
  - IP Multicast, QoS

Overview

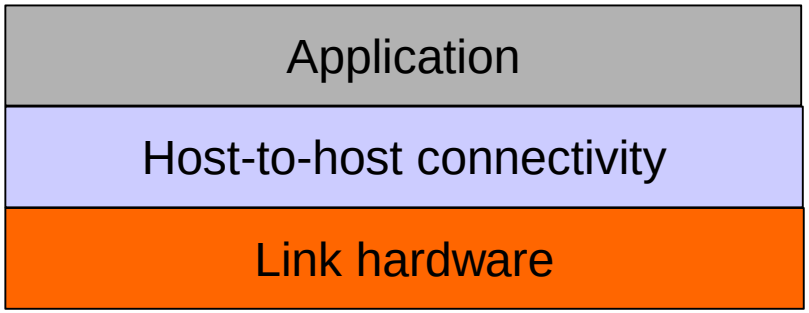
- Packet switching over circuit switching
- End-to-end principle and "Hourglass" design
- Layering and abstractions

Layering

- Modular approach to organizing functionality
- Applied to networks
  - Set of rules governing communication between elements (applications, hosts, routers)
  - Each layer relies on services from layer below and exports services to layer above
  - Each layer specifies format of messages to peer and actions taken based on messages
- Simplifies complex networked systems making them easier to maintain and update
  - Layer implementations can change without disturbing other layers (black box)
  - But, can come with a performance hit (motivates QUIC)

Layering example

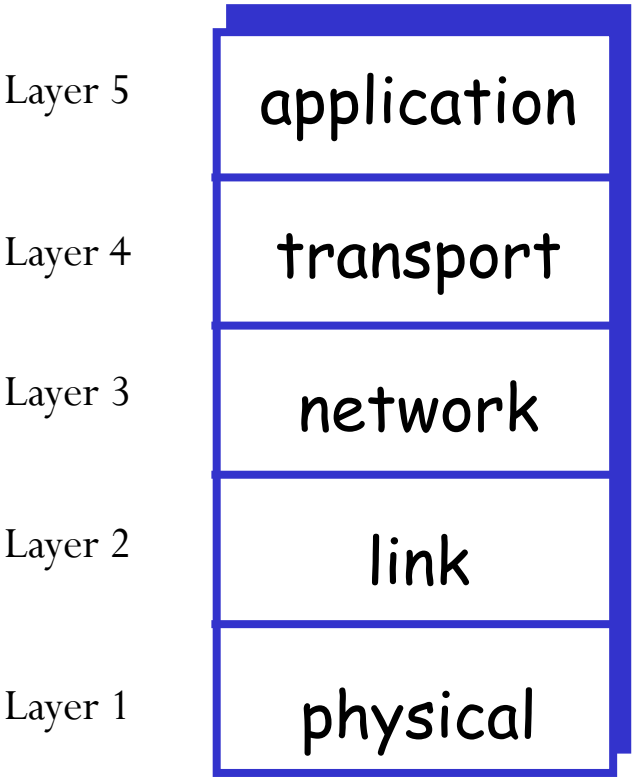
- Topology and physical configuration hidden by network-layer
  - Applications require no knowledge of routes
    - e.g. web servers do not need to calculate routes to clients
    - Abstracts out the network



  - New applications deployed without coordination with network operators or operating system vendors compared to phone network
- Layering and abstraction extends all the way up to the machine, operating system, applications, and collections of all of them!
  - Found all over Computer Science and the cloud
  - Basis for modern serverless cloud applications
    - Cloud platform abstracts out the physical servers, networks, and CDN!

Layering: Internet protocols

- Application:
  - SMTP, HTTP
  - e.g. URL requests and responses
- Transport: process-process data transfer
  - TCP, UDP
  - e.g. how those requests and responses are broken up into network packets
- Network: routing of datagrams from source to destination
  - IP
- Link: data transfer between neighboring network elements
  - Ethernet, 802.11
  - e.g. delivery to next hop router
- Physical: bits "on the wire"



Russian doll analogy

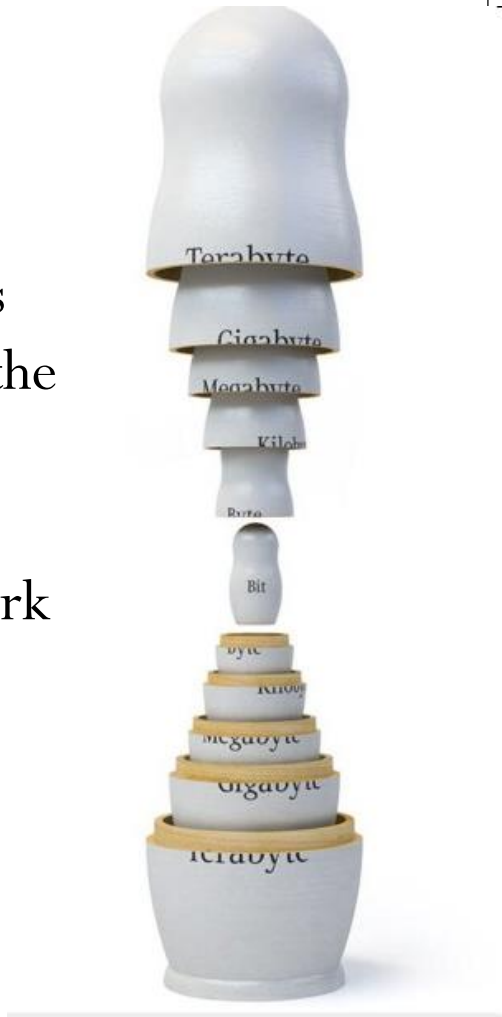
- Packets over the Internet
  - Innermost doll = Application data (i.e. URL request or web page)
  - Next layer = Transport information (i.e. process address or packet sequence number)
  - Next layer = Network information (i.e. network source and destination addresses)
  - Outermost doll = Data-link layer information (i.e. hardware source and destination addresses)

Russian doll analogy

- US Mail analogy
  - Application data (i.e. URL request or web page)
    - Contents of a letter
  - Transport information (i.e. process address or packet sequence number)
    - Recipient: Person, Dorm room #, Apt. #
    - Carrier: USPS, UPS, DHL, FedEx
  - Network information (i.e. network source and destination addresses)
    - Street address, City, State, Zip code
  - Data-link layer information (i.e. hardware source and destination addresses)
    - Vehicle or person transporting the mail

Russian doll analogy

- Operation
  - End host (web client) creates entire doll (app, transport, network, data-link) and sends it to "next hop"
  - Router pulls off outermost doll (data-link), examines destination address of "network layer", and looks up the "next hop" based on it
  - Creates another outer, data-link layer doll, places the packet within it, and sends it to the next hop's network interface.
  - Eventually reaches other end system (web server) which processes all layers to obtain the request



Next 2 weeks

- Crash course on Internet protocol stack
  - Where computing is coming from
  - Review if you've taken CS 494/594