A Review of Key Networking Concepts

Dr. Arjan Durresi
Louisiana State University

Durresi@CSC.LSU.Edu

These slides are available at:
http://www.csc.lsu.edu/~durresi/csc7702-06/

- Telecommunications – tools to improve communications
- ISO/OSI Reference Model
- Ethernet/IEEE 802.3 LANs
- Interconnecting Devices
  All these concepts are taught in other networking courses.

Communication

- Exchange of Information (Communication), makes possible the Human society and the civilization
- Improvements in communication - milestones in the history of civilization
  - Language
  - Writing
  - Books
  - Electronic communication, Internet

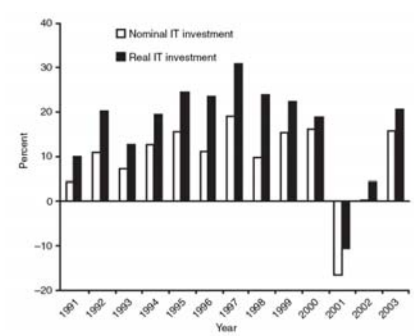
Communication with computers

- Tools created to increase and enhance our capabilities:
  - Cars, Airplanes, Microscopes, Telescopes
  - Telegraph , Telephone to communicate
  - Computers born to store and process information
  - Computers to communicate; Network - more than two computers
- Each epoch in human history is dominated by one tool:
  - Industrial Revolution: Steam engine
  - Information Age: Computers and networks
    - The Internet is the universal medium of communication

The New Economy

- Fundamental transformation in economy as businesses and individuals capitalize on new technologies, new opportunities, and national investments in computing, information, and communication technologies
- New telecommunication technologies have contributed significantly to the New Economy
  - New product capabilities for businesses and consumers
  - More efficient forms of industrial organization made possible by cheaper and more efficient communications
- While telecom sector accounts for about one percent of US economy, it is responsible for generating about ten percent of the nation's economic growth
  - How to sustain or improve on this factor of ten?

The Bubble

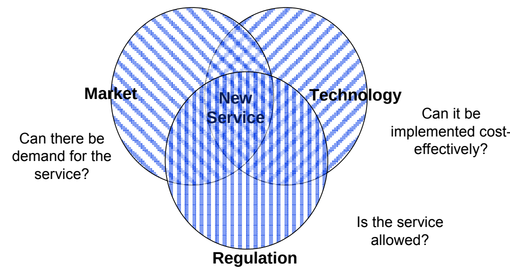


Is Technology only technical stuff ?

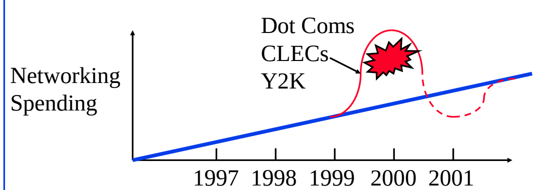
- Technology depends on the Socio-technical System
  - Social, Political, Economic, Institutional
- Not simply the rational product of scientists and engineers.
- Technology makes sense when seen as part of the society
- Examples:
  - Automobile engines: Internal combustion vs. steam
  - Network technologies:
    - OSI vs. TCP/IP vs. ATM, Ethernet vs. Token Ring, ISDN vs. fax
    - Future: Quality of Service mechanisms over the Internet

Success Factors for New Services

- Technology not the only factor in success of a new service
- Three factors considered in new telecom services



The Bubble



- Sidgmore: Internet Traffic doubling every 40 days, 30 days, …⇒ Over-projection data networking equipment
- Nearly 1/3 of all tech IPOs over the last 21 years happened in 1999 and 2000. Source:Morgan Stanely/Chi at Opticomm
- CLEC - Competitive Local Exchange Carrier
- ILEC - Incumbent Local Exchange Carriers

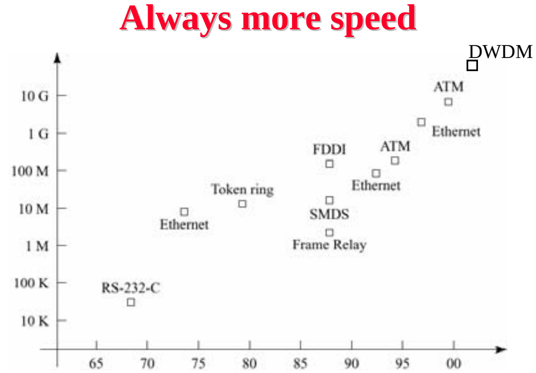
Trends in Network Evolution

- It's all about services
  - Building networks involves huge expenditures
  - Services that generate revenues drive the network architecture
- Current trends
  - Packet switching vs. circuit switching
  - Multimedia applications
  - More versatile signaling
  - End of trust
  - Many service providers and overlay networks
  - Networking *is* a business

Networking Trends

- Networking Bottleneck – Bandwidth bps more important that CPU speed
- Networking Age - Internet-based Economy
- Internet Growth - High
- Data > Voice ⇒ Networking and Telecom Merger
- Quality of Service – "Holy Grail" for the research
- Optical Networks – The promise for unlimited bandwidth
- The Internet - the universal medium of communication

Always more speed



Trend: Convergence

Entertainment
Video Games
Publishing
News
Advertising — Cable TV — Telephone — Computer

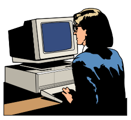
Digital Media Production	Video Transport	Voice Transport	Digital Media Storage/ Handling

Convergence (Cont)

Content
Computing — Communications

- Merging of Content Providers and Content transporters
- Phone companies, cable companies, entertainment industry, and computer companies
- Single department for telephone and computer networking
- LAN/WAN convergence

Social Impact of Networking




From : "To be important should be online"
to: " To exist should be online"

- No need to get out for
  - Office
  - Shopping
  - Entertainment
  - Education
- Virtual Schools
- Virtual Cash
- Virtual Workplace
- More than 170 Million US online users

Cave Persons of 2050

Packet vs. Circuit Switching

- Architectures appear and disappear over time
  - Telegraph (message switching)
  - Telephone (circuit switching)
  - Internet (packet switching)
- Trend towards packet switching at the edge
  - IP enables rapid introduction of new applications
  - New cellular voice networks packet-based
  - Soon IP will support *real-time* voice and telephone network will gradually be replaced
  - However, large packet flows easier to manage by circuit-like methods

Optical Circuit Switching

- Optical signal transmission over fiber can carry huge volumes of information (Tbps)
- Optical signal processing very limited
  - Optical logic circuits bulky and costly
  - Optical packet switching will not happen soon
- Optical-to-Electronic conversion is expensive
  - Maximum electronic speeds << Tbps
  - Parallel electronic processing & high expense
- Thus trend towards optical circuit (packet ?) switching in the core

Multimedia Applications

- Trend towards digitization of *all* media
- Digital voice standard in cell phones
- Music cassettes replaced by CDs and MP3's
- Digital cameras replacing photography
- Video: digital storage and transmission
  - Analog VCR cassettes largely replaced by DVDs
  - Analog broadcast TV to be replaced by digital TV
  - VCR cameras/recorders to be replaced by digital video recorders and cameras
- High-quality network-based multimedia applications now feasible

More Versatile Signaling

- Signaling inside the network
  - Connectionless packet switching keeps network simple & avoids large scale signaling complexity
  - Large packet flows easier to manage using circuit-like methods that require signaling
  - Optical paths also require signaling
  - Generalized signaling protocols being developed
- End-to-End Signaling
  - Session-oriented applications require signaling between the endpoints (not inside the network)
  - Session Initiation Protocol taking off

End of Trust

- Security Attacks
  - Spam
  - Denial of Service attacks
  - Viruses
  - Impersonators
- Firewalls & Filtering
  - Control flow of traffic/data from Internet
- Protocols for privacy, integrity and authentication

Servers & Services

- Many Internet applications involve interaction between client and server computers
  - Client and servers are at the edge of the Internet
  - SMTP, HTTP, DNS, …
- Enhanced services in telephone network also involve processing from servers
  - Caller ID, voice mail, mobility, roaming, . . .
  - These servers are inside the telephone network
  - Internet-based servers at the edge can provide same functionality
- In future, multiple service providers can coexist and serve the same customers

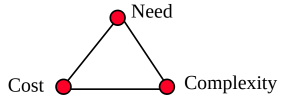
P2P and Overlay Networks

- Client resources under-utilized in client-server
- Peer-to-Peer applications enable sharing
  - Napster, Gnutella, Kazaa
  - Processing & storage (SETI@home)
  - Information & files (MP3s)
  - Creation of virtual distributed servers
- P2P creates transient overlay networks
  - Users (computers) currently online connect directly to each other to allow sharing of their resources
  - Huge traffic volumes a challenge to network management
  - Huge opportunity for new businesses

Operations, Administration, Maintenance, and Billing

- Communication like transportation networks
  - Traffic flows need to be monitored and controlled
  - Tolls have to be collected
  - Roads have to be maintained
  - Need to forecast traffic and plan network growth
- Highly-developed in telephone network
  - Entire organizations address OAM & Billing
  - Becoming automated for flexibility & reduced cost
- Under development for IP networks

The right Trade off in Networking



- User is the King => Pays the bill
- What does the user really need?
- Killer applications are key for the success of a particular technology
- In today's Internet the driving need is connectivity
  - Email and web browser – killer applications, which don't need more QOS
- Future Internet, new applications + more QOS ?

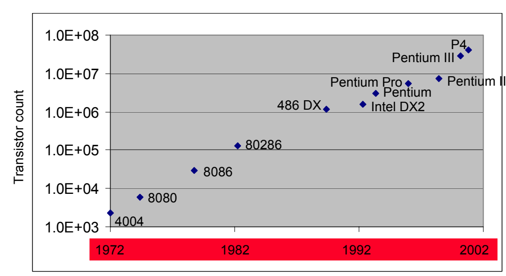
Transmission Technology

- Relentless improvement in transmission
- High-speed transmission in copper pairs
  - DSL Internet Access
- Higher call capacity in cellular networks
  - Lower cost cellular phone service
- Enormous capacity and reach in optical fiber
  - Plummeting cost for long distance telephone
- Faster and more information intensive applications

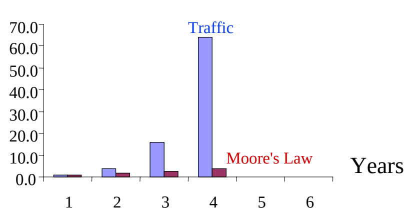
Processing Technology

- Relentless improvement in processing & storage
- Moore's Law: doubling of transistors per integrated circuit every two years
- RAM: larger tables, larger systems
- Digital signal processing: transmission, multiplexing, framing, error control, encryption
- Network processors: hardware for routing, switching, forwarding, and traffic management
- Microprocessors: higher layer protocols and applications
- Higher speeds and higher throughputs in network protocols and applications

Moore's Law

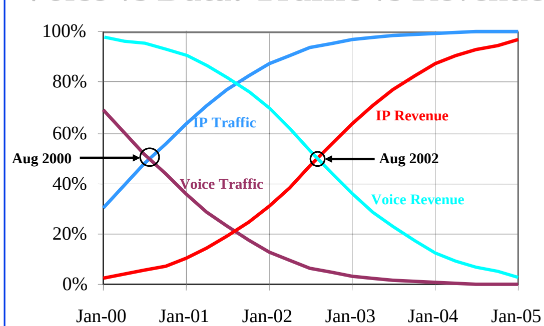


Moore's Law is Too Slow



- Moore's Law: Factor of 2 every 1.5 years ⇒ 60%/year
- Internet Traffic: Factor of 4 per year
  ⇒ Need for more Networks, QoS, Optical Switching

Voice vs Data: Traffic vs Revenue



Source: L. Roberts at Opticomm 2001

Trend: Traffic > Capacity

"You can never be too rich, too thin, or have too much bandwidth"

Expensive Bandwidth	Cheap Bandwidth
❑ Sharing	❑ No sharing
❑ Multicast	❑ Unicast
❑ Virtual Private Networks	❑ Private Networks
❑ Need QoS	❑ QoS less of an issue
❑ Likely in WANs	❑ Possible in LANs

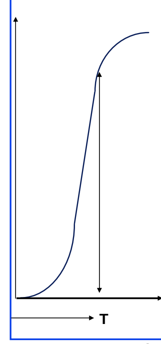
Software Technology

- Greater functionality & more complex systems
- TCP/IP in operating systems
- Java and virtual machines
- New application software
- Middleware to connect multiple applications
- Adaptive distributed systems

Market

- *The network effect*: usefulness of a service increases with size of community
  - Metcalfe's Law: usefulness is proportional to the square of the number of users
  - Phone, fax, email, ICQ, …
- *Economies of scale*: per-user cost drops with increased volume
  - Cell phones, PDAs, PCs
  - Efficiencies from multiplexing
- *S-curve*: growth of new service has S-shaped curve, challenge is to reach the critical mass

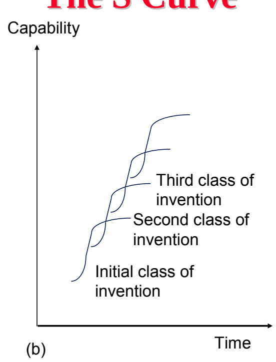
The S Curve



Service Penetration & Network Effect

- Telephone: T=30 years
  - city-wide & inter-city links
- Automobile: T=30 years
  - roads
- Others
  - Fax
  - Cellular & cordless phones
  - Internet & WWW
  - Napster and P2P

The S Curve



Regulation & Competition

- Telegraph & Telephone originally monopolies
  - Extremely high cost of infrastructure
  - Profitable, predictable, slow to innovate
- Competition feasible with technology advances
  - Long distance cost plummeted with optical tech
  - Alternative local access through cable, wireless
  - Radio spectrum: auctioned vs. unlicensed
- Basic connectivity vs. application provider
  - Tussle for the revenue-generating parts

Standards

- New technologies very costly and risky
- Standards allow players to share risk and benefits of a new market
  - Reduced cost of entry
  - Interoperability and network effect
  - Compete on innovation
  - Completing the value chain
    - Chips, systems, equipment vendors, service providers
- Example
  - 802.11 wireless LAN products

Standards Bodies

- Internet Engineering Task Force
  - Internet standards development
  - Request for Comments (RFCs): www.ietf.org
- International Telecommunications Union
  - International telecom standards
- IEEE 802 Committee
  - Local area and metropolitan area network standards
- Industry Organizations
  - MPLS Forum, WiFi Alliance, World Wide Web Consortium

Dealing with Network Complexity

- Network complexity:
  - Many technologies with different features
  - Not all standards are compatible, from different organizations
  - Multiple technologies to interconnect the networks
  - No single underlying theory that explains the relationship among the parts
- How to learn about the networking ?
  - Focus on the concepts, go beyond the details
  - When needed is easy to go from concepts to details
  - Concepts are “borrowed” among technologies.

Transformations in Communications

- Diffusion of Broadband
- Voice over Internet Protocol – VoIP
- Grid Computing
- Seamless Wireless

VoIP

VoIP has the potential to undermine the business model underpinning the telecommunications industry. Factors such as the length of the call or the distance between callers, key determinants of cost today, are irrelevant with VoIP. In addition, VoIP augers more widespread use of videoconferencing as well as new applications such as unified messaging and television over Internet Protocol (IPTV).

Many analysts believe that the question is not whether VoIP will displace traditional telephony, but how quickly. This disruptive potential of VoIP is a challenge for telephone, mobile, and cable incumbents—with some attempting to block the new technology and others moving to embrace it.[a]

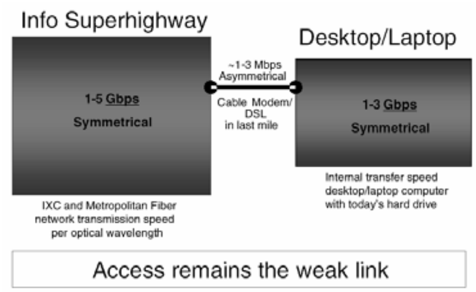
VoIP

- Two new opportunities:
  - Sharing infrastructure – packets over existing networks
  - Using software to provide a variety of services for customers

Grid Computing

- Allows users to share data, software, and computation power over fiber optics networks
- SETI (Search for Extraterrestrial Intelligence)
  - About 500,000 people have downloaded the program, generating an amount of computing power that would have cost $100 million to purchase
- "Holy Grid" – where everything is connected to everything, running common software, able to tackle a wide range of problems
- *Autonomic Computing* – where integrated computer systems are not only able to self-protecting, self-configuring, and self-healing, but also come closer to self-managing.
- *Pervasive Computing* – where sensors embedded in a variety of devices and products would gather and analyze data.
  - Soon trillion of sensors
- With telecommunication firms becoming more dependent on information technology, and vice versa, the two industries are becoming more interwined.

Broadband Challenge



Broadband Challenge

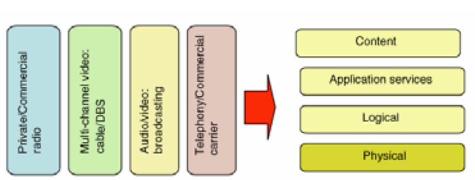
- The US has fallen behind far behind other nations in broadband penetration.
  - US thirteen to fifteen place
  - DSL 4.8% compared to 27.7% of South Korea
- Bandwidth constrain rather than computer hardware frequently dominate the cost of adapting new applications.
- Improved access – high quality video conferencing, gaming, VoIP, immersive gaming etc.
- Flawed market motives of telecom and cable companies

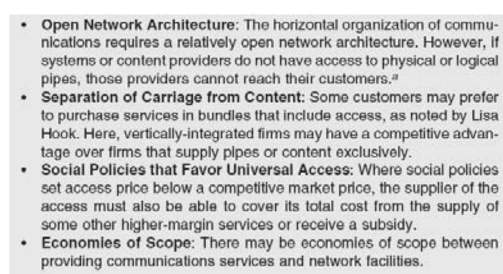
Problems

- Internet ignores whatever is specific about a single network, including features that had formed the basis of or the telephone or cable companies
- Because these companies have little to sell beyond access, they have little incentive in providing broadband access
- Local Loop Unbundling – LLU (1996 Telecommunication Act) may have inhibit the investment in broadband
- Outdated Standards and Regulatory Uncertainty
  - Cable industry is regulated by city-specific rules
- Wireless can help in broadband

The End of Stovepiping

- Transformation from vertical industrial organization in print, radio, entertainment, and broadcast to more horizontal Internet based platforms

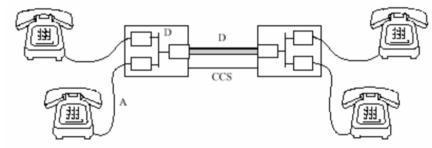




- **Open Network Architecture**: The horizontal organization of communications requires a relatively open network architecture. However, if systems or content providers do not have access to physical or logical pipes, those providers cannot reach their customers.[a]
- **Separation of Carriage from Content**: Some customers may prefer to purchase services in bundles that include access, as noted by Lisa Hook. Here, vertically-integrated firms may have a competitive advantage over firms that supply pipes or content exclusively.
- **Social Policies that Favor Universal Access**: Where social policies set access price below a competitive market price, the supplier of the access must also be able to cover its total cost from the supply of some other higher-margin services or receive a subsidy.
- **Economies of Scope**: There may be economies of scope between providing communications services and network facilities.

Evolution of Networks

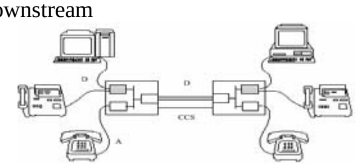
- In 1890 simple telephone networks with manually operated switches – circuit switching
- Operators replaced by mechanical switches and 100 years later by electronic switches



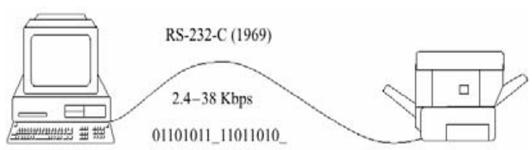
- Electronic switches and exchange control information using the common channel signaling (CCS)

Evolution of Networks

- Since 1980s Synchronous Optical Network (SONET) and Synchronous Digital Hierarchy (SDH) for the transmission links of the telephone networks
- Integrated Services Digital Network (ISDN) – integration of data and voice
- Asymmetric Digital Subscriber Line (ADSL) > 1.5 Mbps downstream

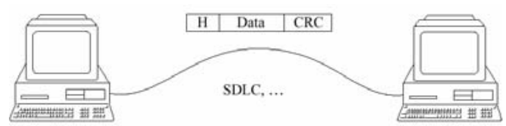


RS-232C



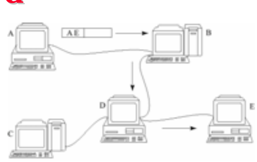
- Up to 38kbps over short distances (less than 30m)
- Serial transmission one character at a time
- Each character => 7 bits + 1bit parity

Synchronous Transmission



- The first computer networks designed to share large-scale computers
- HDLC family
- Packet based => header + data + CRC

Store-and-forward Packet Switching

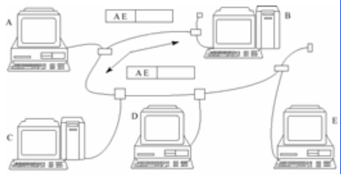


- To interconnect many computers
- Statistical multiplexing – more efficient than time-division multiplexing
- ARPNET late 1960s
  - The network is peripheral

Evolution of Networks

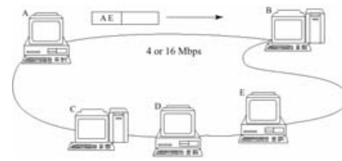
- Emergence of LANs
- In the 1990s Internet becomes a commercial success
- Internet has been doubling is size every nine months
- From "If a company is important it is online" to "If a company is not on line it doesn't exist"
- The impact of Internet:
  - Economic, Education, Government, etc.

Local Area Networks



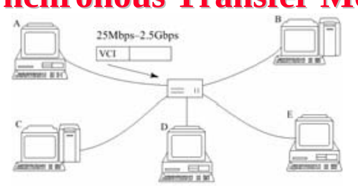
- Emergence of LANs
- Ethernet 10Mbps, 100Mbps, 1Gbps, 10Gbps
- Ethernet everywhere: LAN and WAN

LAN



- Token Ring – better performance than Ethernet
  - Avoid collisions
  - Handle priorities
- FDDI up to 100Mbps
  - Data + Real time traffic

Asynchronous Transfer Mode

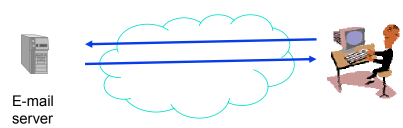


- Switched technology – total throughput much larger than shared medium ones
- 53 bytes cell. Integration of data and voice.
- Very good QOS
- Complex, less scalable than Ethernet and IP, no native applications

Communication Services & Applications

- A communication service enables the exchange of information between users at different locations.
- Communication services & applications are everywhere.

E-mail

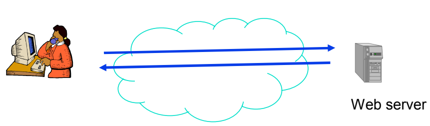


Exchange of text messages via servers

Communication Services & Applications

- A communication service enables the exchange of information between users at different locations.
- Communication services & applications are everywhere.

Web Browsing

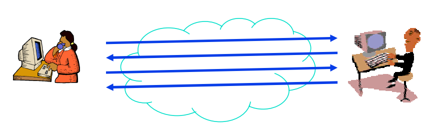


Retrieval of information from web servers

Communication Services & Applications

- A communication service enables the exchange of information between users at different locations.
- Communication services & applications are everywhere.

Instant Messaging

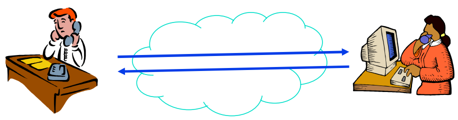


Direct exchange of text messages

Communication Services & Applications

- A communication service enables the exchange of information between users at different locations.
- Communication services & applications are everywhere.

Telephone

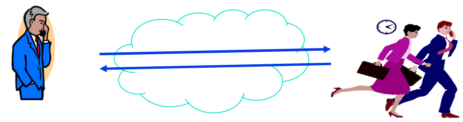


Real-time bidirectional voice exchange

Communication Services & Applications

- A communication service enables the exchange of information between users at different locations.
- Communication services & applications are everywhere.

Cell phone

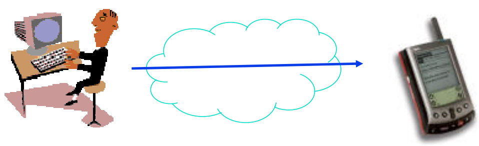


Real-time voice exchange with mobile users

Communication Services & Applications

- A communication service enables the exchange of information between users at different locations.
- Communication services & applications are everywhere.

Short Message Service



Fast delivery of short text messages

Many other examples!

- Peer-to-peer applications
  - Napster, Gnutella, Kazaa file exchange
  - Searching for ExtraTerrestrial Intelligence (SETI)
- Audio & video streaming
- Network games
- On-line purchasing
- Text messaging in PDAs, cell phones (SMS)
- Voice-over-Internet

Services & Applications

- Service: Basic information transfer capability
  - Internet transfer of individual block of information
  - Internet reliable transfer of a stream of bytes
  - Real-time transfer of a voice signal
- Applications build on communication services
  - E-mail & web build on reliable stream service
  - Fax and modems build on basic telephone service
- New applications build on multiple networks
  - SMS builds on Internet reliable stream service and cellular telephone text messaging

What is a communication network?

- The equipment (hardware & software) and facilities that provide the basic communication service
- Virtually invisible to the user; Usually represented by a cloud

- Equipment
  - Routers, servers, switches, multiplexers, hubs, modems, …
- Facilities
  - Copper wires, coaxial cables, optical fiber
  - Ducts, conduits, telephone poles …

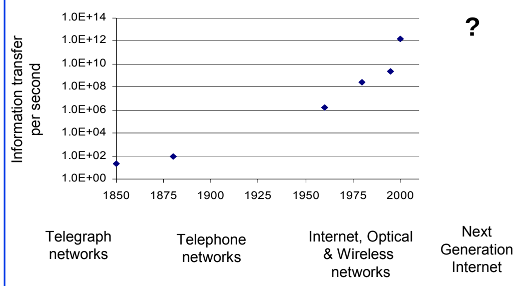
How are communication networks designed and operated?

Communication Network Architecture

- *Network architecture:* the plan that specifies how the network is built and operated
- Architecture is driven by the network services
- Overall communication process is complex
- Network architecture partitions overall communication process into separate functional areas called *layers*

Next we will trace evolution of three network architectures: telegraph, telephone, and computer networks

Network Architecture Evolution



Network Architecture Evolution

- Telegraph Networks
  - Message switching & digital transmission
- Telephone Networks
  - Circuit Switching
  - Analog transmission → digital transmission
  - Mobile communications
- Internet
  - Packet switching & computer applications
- Next-Generation Internet
  - Multiservice packet switching network

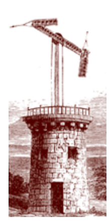
Telegraphs & Long-Distance Communications

Approaches to long-distance communications

- Courier: physical transport of the message
  - Messenger pigeons, pony express, FedEx
- Telegraph: message is transmitted across a network using signals
  - Drums, beacons, mirrors, smoke, flags, semaphores…
  - Electricity, light
- Telegraph delivers message much sooner

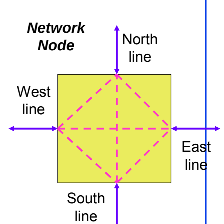
Optical (Visual) Telegraph

- Claude Chappe invented optical telegraph in the 1790's
- Semaphore mimicked a person with outstretched arms with flags in each hand
- Different angle combinations of arms & hands generated hundreds of possible signals
- Code for enciphering messages kept secret
- Signal could propagate 800 km in 3 minutes!



Message Switching

- Network nodes were created where several optical telegraph lines met (Paris and other sites)
- *Store-and-Forward* Operation:
  - Messages arriving on each line were decoded
  - Next-hop in ***route*** determined by destination ***address*** of a message
  - Each message was carried by hand to next line, and stored until operator became available for next transmission



Electric Telegraph

- William Sturgeon Electro-magnet (1825)
  - Electric current in a wire wrapped around a piece of iron generates a magnetic force
- Joseph Henry (1830)
  - Current over 1 mile of wire to ring a bell
- Samuel Morse (1835)
  - Pulses of current deflect electromagnet to generate dots & dashes
  - Experimental telegraph line over 40 miles (1840)
- Signal propagates at the speed of light!!!
  - Approximately 2 x 10^8 meters/second in cable

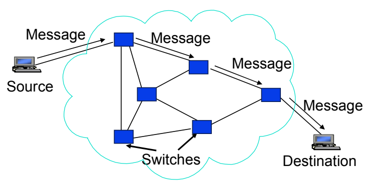
Digital Communications

- Morse code converts text message into sequence of dots and dashes
- Use transmission system designed to convey dots and dashes

	Morse Code		Morse Code		Morse Code		Morse Code
A	· —	J	· — — —	S	· · ·	2	· · — — —
B	— · · ·	K	— · —	T	—	3	· · · — —
C	— · — ·	L	· — · ·	U	· · —	4	· · · · —
D	— · ·	M	— —	V	· · · —	5	· · · · ·
E	·	N	— ·	W	· — —	6	— · · · ·
F	· · — ·	O	— — —	X	— · · —	7	— — · · ·
G	— — ·	P	· — — ·	Y	— · — —	8	— — — · ·
H	· · · ·	Q	— — · —	Z	— — · ·	9	— — — — ·
I	· ·	R	· — ·	1	· — — — —	0	— — — — —

Electric Telegraph Networks

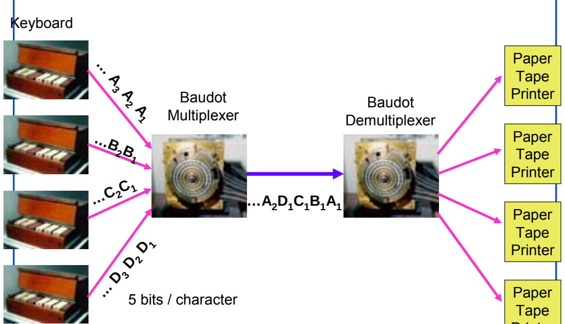
- Electric telegraph networks exploded
  - Message switching & Store-and-Forward operation
  - Key elements: Addressing, Routing, Forwarding
- Optical telegraph networks disappeared



Baudot Telegraph Multiplexer

- Operator 25-30 words/minute
  - but a wire can carry much more
- Baudot multiplexer: Combine 4 signals in 1 wire
  - Binary block *code* (ancestor of ASCII code)
    - A character represented by 5 bits
  - Time division *multiplexing*
    - Binary codes for characters are interleaved
  - *Framing* is required to recover characters from the binary sequence in the multiplexed signal
  - *Keyboard* converts characters to bits

Baudot Telegraph Multiplexer



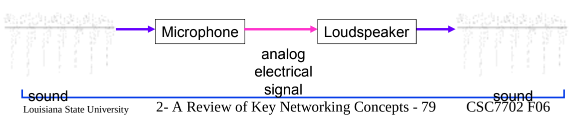
Elements of Telegraph Network Architecture

- Digital transmission
  - Text messages converted into symbols (dots/dashes, zeros/ones)
  - Transmission system designed to convey symbols
- Multiplexing
  - *Framing* needed to recover text characters
- Message Switching
  - Messages contain source & destination *addresses*
  - *Store-and-Forward:* Messages forwarded hop-by-hop across network
  - *Routing* according to destination address

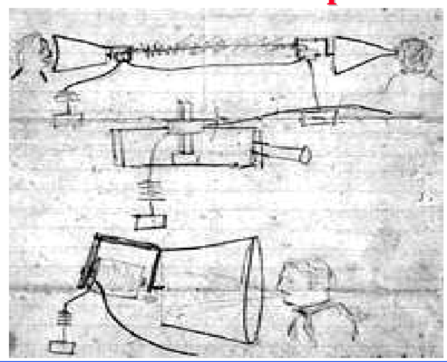
Bell's Telephone

- Alexander Graham Bell (1875) working on harmonic telegraph to multiplex telegraph signals
- Discovered voice signals can be transmitted directly
  - Microphone converts voice pressure variation (sound) into *analogous* electrical signal
  - Loudspeaker converts electrical signal back into sound
- Telephone patent granted in 1876
- Bell Telephone Company founded in 1877

Signal for "ae" as in cat

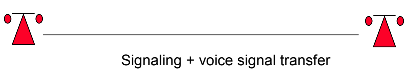


Bell's Sketch of Telephone



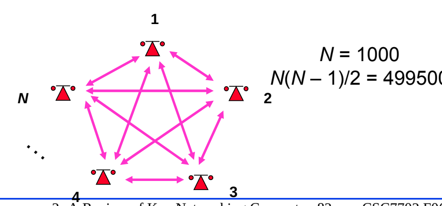
Signaling

- Signaling required to establish a call
  - Flashing light and ringing devices to alert the called party of incoming call
  - Called party information to operator to establish calls

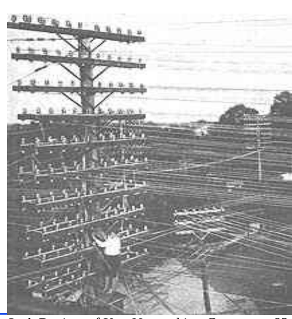


The N^2 Problem

- For N users to be fully connected *directly*
- Requires $N(N-1)/2$ connections
- Requires too much space for cables
- Inefficient & costly since connections not always on

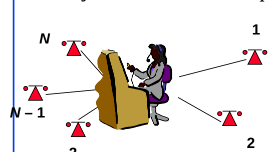


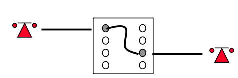
Telephone Pole Congestion



Circuit Switching

- Patchcord panel switch invented in 1877
- Operators connect users on demand
  - Establish *circuit* to allow electrical current to flow from inlet to outlet
- Only N connections required to central office

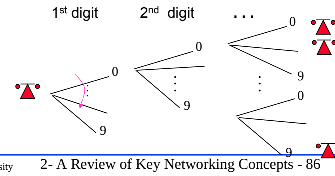




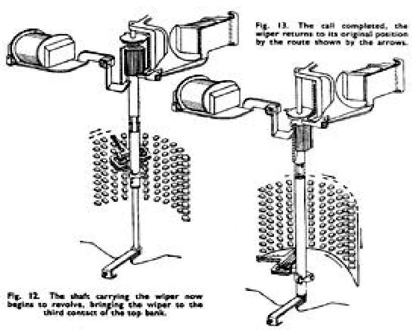
Manual Switching

Strowger Switch

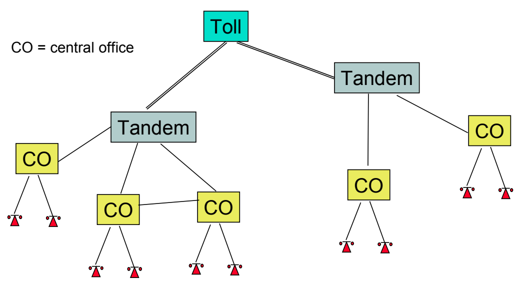
- Human operators intelligent & flexible
  - But expensive and not always discreet
- Strowger invented automated switch in 1888
  - Each current pulse advances wiper by 1 position
  - User dialing controls connection setup
- Decimal telephone numbering system
- Hierarchical network structure simplifies routing
  - Area code, exchange (CO), station number



Strowger Switch



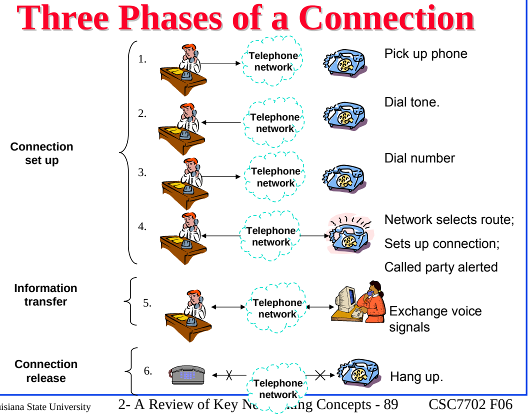
Hierarchical Network Structure



Telephone subscribers connected to local CO (central office)

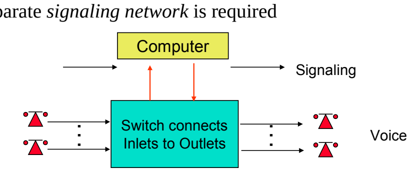
Tandem & Toll switches connect CO's

Three Phases of a Connection



Computer Connection Control

- A computer controls connection in telephone switch
- Computers exchange *signaling messages* to:
  - Coordinate set up of telephone connections
  - To implement new services such as caller ID, voice mail, . . .
  - To enable *mobility and roaming in* cellular networks
- "Intelligence" inside the network
- A separate *signaling network* is required

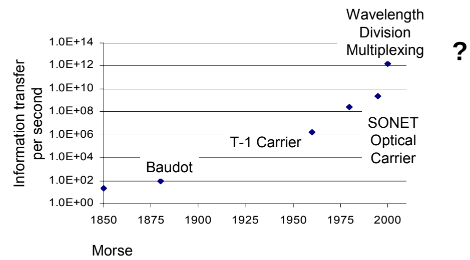


Digitization of Telephone Network

- Pulse Code Modulation digital voice signal
  - Voice gives 8 bits/sample x 8000 samples/sec = 64×10^3 bps
- Time Division Multiplexing for digital voice
  - T-1 multiplexing (1961): 24 voice signals = 1.544×10^6 bps
- Digital Switching (1980s)
  - Switch TDM signals without conversion to analog form
- Digital Cellular Telephony (1990s)
- Optical Digital Transmission (1990s)
  - One OC-192 optical signal = 10×10^9 bps
  - One optical fiber carries 160 OC-192 signals = 1.6×10^{12} bps!

All digital transmission, switching, and control

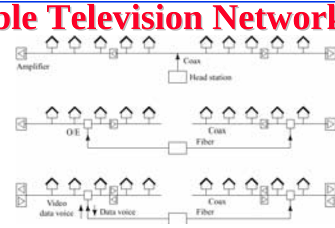
Digital Transmission Evolution



Elements of Telephone Network Architecture

- Digital transmission & switching
  - Digital voice; Time Division Multiplexing
- Circuit switching
  - User signals for call setup and tear-down
  - Route selected during connection setup
  - End-to-end connection across network
  - Signaling coordinates connection setup
- Hierarchical Network
  - Decimal numbering system
  - Hierarchical structure; simplified routing; scalability
- Signaling Network
  - Intelligence inside the network

Cable Television Networks



- CATV delivers TV in more than 50% of US households
- High Speed Internet Access. Cable modems: 45 Mbps downstream, 1.5 Mbps upstream
- Sharing ⇒ Security issues
- No cable in offices
- But new revenue for providers

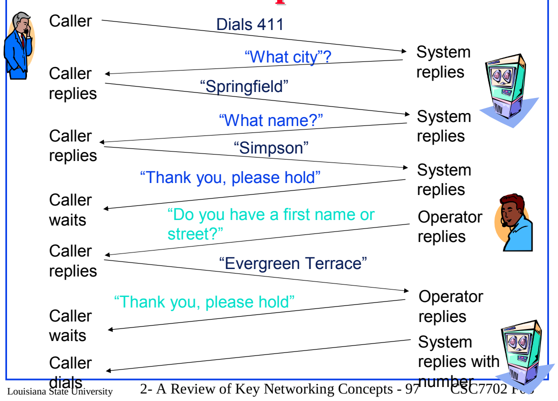
Computer Network Evolution Overview

- *1950s*: Telegraph technology adapted to computers
- *1960s*: Dumb terminals access shared host computer
  - SABRE airline reservation system
- *1970s*: Computers connect directly to each other
  - ARPANET packet switching network
  - TCP/IP internet protocols
  - Ethernet local area network
- *1980s & 1990s*: New applications and Internet growth
  - Commercialization of Internet
  - E-mail, file transfer, web, P2P, . . .
  - Internet traffic surpasses voice traffic

What is a protocol?

- Communications between computers requires very specific unambiguous rules
- A protocol is a set of rules that governs how two or more communicating parties are to interact
  - Internet Protocol (IP)
  - Transmission Control Protocol (TCP)
  - HyperText Transfer Protocol (HTTP)
  - Simple Mail Transfer Protocol (SMTP)

A familiar protocol



Why Layering?

- Layering simplifies design, implementation, and testing by partitioning overall communications process into parts
- Protocol in each layer can be designed separately from those in other layers
- Protocol makes "calls" for services from layer below
- Layering provides flexibility for modifying and evolving protocols and services without having to change layers below
- Monolithic non-layered architectures are costly, inflexible, and soon obsolete

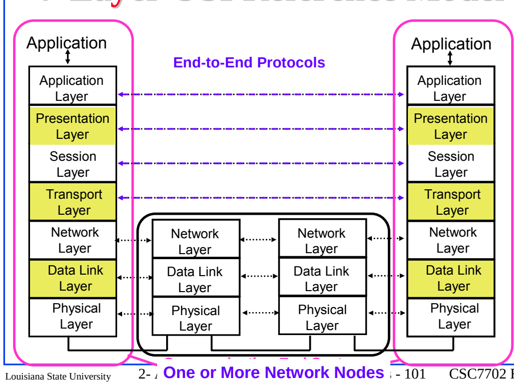
Open Systems Interconnection

- Network architecture:
  - Definition of all the layers
  - Design of protocols for every layer
- By the 1970s every computer vendor had developed its own proprietary layered network architecture
- Problem: computers from different vendors could not be networked together
- Open Systems Interconnection (OSI) was an international effort by the International Organization for Standardization (ISO) to enable multivendor computer interconnection

OSI Reference Model

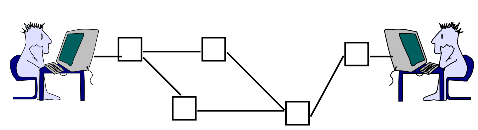
- Describes a seven-layer abstract reference model for a network architecture
- Purpose of the reference model was to provide a framework for the development of protocols
- OSI also provided a unified view of layers, protocols, and services which is still in use in the development of new protocols
- Detailed standards were developed for each layer, but most of these are not in use
- TCP/IP protocols preempted deployment of OSI protocols

7-Layer OSI Reference Model

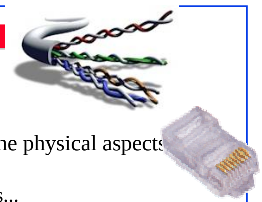


ISO/OSI Reference Model

3	Application	File transfer, Email, Remote Login
	Presentation	ASCII Text, Sound
	Session	Establish/manage connection
2	Transport	End-to-end communication: TCP
	Network	Routing, Addressing: IP
1	Datalink	Two party communication: Ethernet
	Physical	How to transmit signal: Coding

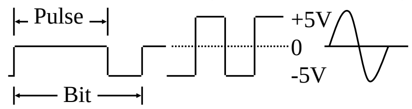


Physical Layer



- Transfers bits across link
- Definition & specification of the physical aspects of a communications link
  - Mechanical: cable, plugs, pins...
  - Electrical/optical: modulation, signal strength, voltage levels, bit times, …
  - functional/procedural: how to activate, maintain, and deactivate physical links…
- Ethernet, DSL, cable modem, telephone modems…
- Twisted-pair cable, coaxial cable optical fiber, radio, infrared, …

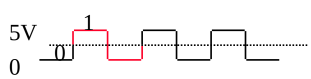
Coding Terminology



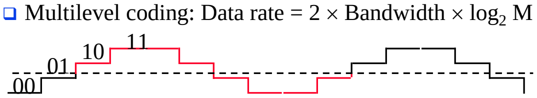
- Signal element: Pulse
- Modulation Rate: 1/Duration of the smallest element =Baud rate
- Data Rate: Bits per second
- Data Rate = Fn(Bandwidth, signal/noise ratio, encoding)

Channel Capacity

- Capacity = Maximum data rate for a channel
- **Nyquist Theorem**:
- Bilevel Encoding: Data rate = 2 × Bandwidth



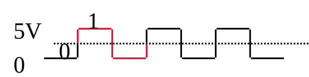
- Multilevel coding: Data rate = 2 × Bandwidth × $\log_2 M$



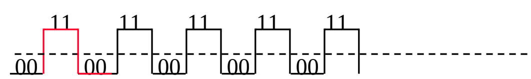
Example: M=4, Capacity = 4 × Bandwidth

Channel Capacity (Cont)

- Bilevel Encoding: Worst case: 1010101010
  Cycle time = 2 × Bit time
  ⇒ Data rate = 2 × Bandwidth

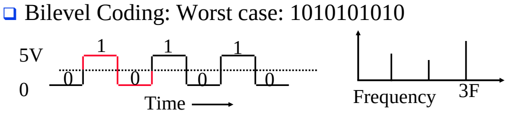


- Multilevel coding: Worst case 0011001100110011
  Cycle time = 4 × Bit time
  ⇒ Data rate = 2 × Bandwidth × $\log_2 M$

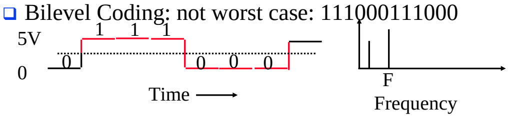


Channel Capacity (cont.)

- Bilevel Coding: Worst case: 1010101010



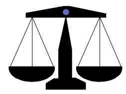
- Bilevel Coding: not worst case: 111000111000



Shannon's Theorem

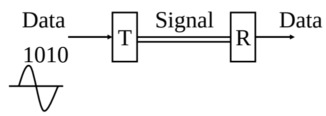
- Bandwidth = B Hz
  Signal-to-noise ratio = S/N
- Maximum number of bits/sec $C_{MAX} = B \log_2 (1+S/N)$
  - Application of Second Law of Thermodynamic
  - When more than C_{MAX}, information becomes noise
- Example: Phone wire bandwidth = 3100 Hz
  S/N = 30 dB, 10 $\log_{10}$ S/N = 30
  $\log_{10}$ S/N = 3, S/N = 10^3 = 1000
  Capacity = 3100 $\log_2$ (1+1000) = 30,894 bps
- Compression : Code repetitive patterns with a shorter data set
  - Example: Code "XXXXXX" (6 bytes) with 2 bytes

Nyquist's vs. Shannon's Theorem



- Nyquist's Theorem:
  - Explore ways to encode bits
  - Clever encoding allows more bits to be transmitted per unit time, for example multilevel encoding
- Shannon's Theorem:
  - No amount of clever encoding can overcome the laws of physics

Data vs Signal

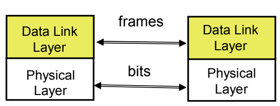


- Data: Analog (Music), Digital (files)
- Signal: Analog (POTS, Radio), Digital (ISDN)

Data	Signal		Examples
Analog	Analog	Modulation	AM, FM
Digital	Analog	Coding/Keying	ASK, FSK, PSK
Analog	Digital	Modulation	PCM, ADPCM
Digital	Digital	Coding	Manchester, NRZ

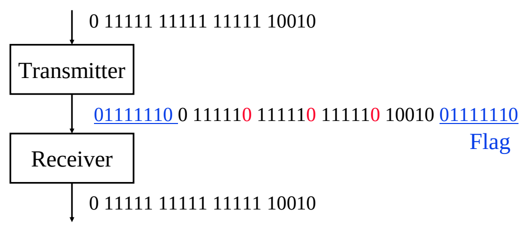
Data Link Layer

- Transfers *frames* across *direct* connections
- Groups bits into frames
- Detection of bit errors; Retransmission of frames
- Activation, maintenance, & deactivation of data link connections
- Medium access control for local area networks
- Flow control



Bit Stuffing

- Delimit with special bit pattern (bit flags)
- Stuff bits if pattern appears in data
- Remove stuffed bits at destination



What Goes Wrong in the Network?

- Bit-level errors (electrical interference)
- Packet-level errors (congestion)
- Link and node failures
- Messages are delayed
- Messages are deliver out-of-order
- Third parties eavesdrop
- The key problem is to fill in the gap between what applications expect and what the underlying technology provides.

Flow Control

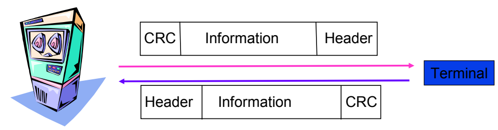
- Flow Control = Sender does not flood the receiver, but maximizes throughput
- Sender throttled until receiver grants permission
- Methods:
  - Stop and wait
  - Sliding window

Error Control

- Error Control = Deliver frames without error, in the proper order to network layer
- Error control Mechanisms:
  - Ack/Nak: Provide sender some feedback about other end
  - Time-out: for the case when entire packet or ack is lost
  - Sequence numbers: to distinguish retransmissions from originals
- ARQ: Stop and Wait, Selective Reject, Go-back n

Error Control Protocol

- Communication lines introduced errors
- Error checking codes used on frames
  - "Cyclic Redundancy Check" (CRC) calculated based on frame header and information payload, and appended
  - Header also carries ACK/NAK control information
- Retransmission requested when errors detected



Multiple Access Protocols

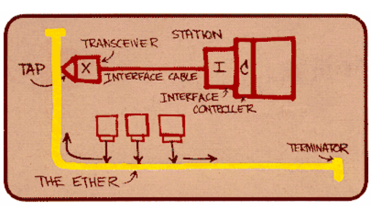
(a) Multiple Access

(b) Carrier-Sense Multiple Access with Collision Detection

Multiple Access Protocols

- Aloha at University of Hawaii:
  Transmit whenever you like
  Worst case utilization = 1/(2e) =18%
- CSMA: Carrier Sense Multiple Access
  Listen before you transmit
- CSMA/CD: CSMA with Collision Detection
  Listen while transmitting.
  Stop if you hear someone else.
- Ethernet uses CSMA/CD.
  Standardized by IEEE 802.3 committee.

Ethernet



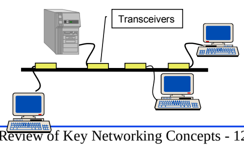
The diagram was drawn by Dr. Robert M. Metcalfe in 1976 to present Ethernet to the National Computer Conference in June of that year

Ethernet Local Area Network

- In 1980s, affordable workstations available
- Need for low-cost, high-speed networks
  - To interconnect local workstations
  - To access local shared resources (printers, storage, servers)
- Low cost, high-speed communications with low error rate possible using coaxial cable
- Ethernet is the standard for high-speed wired access to computer networks

Ethernet Medium Access Control

- Network interface card (NIC) connects workstation to LAN
- Each NIC has globally unique address
- Frames are broadcast into coaxial cable
- NICs listen to medium for frames with their address
- Transmitting NICs listen for collisions with other stations, and abort and reschedule retransmissions

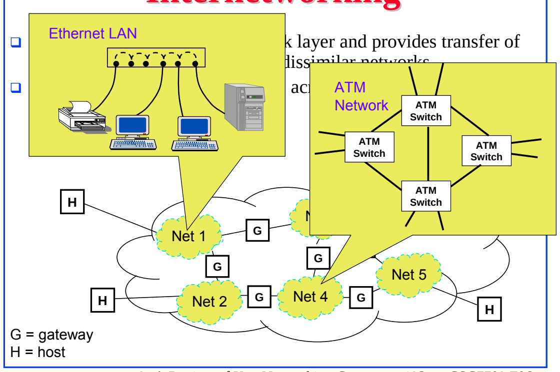


Network Layer

- Transfers *packets* across multiple links and/or multiple networks
- Addressing must scale to large networks
- Nodes *jointly* execute routing algorithm to determine paths across the network
- Forwarding transfers packet across a node
- Congestion control to deal with traffic surges
- Connection setup, maintenance, and teardown when connection-based

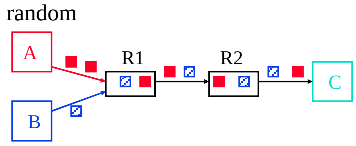
Internetworking

- k layer and provides transfer of dissimilar networks
- ac



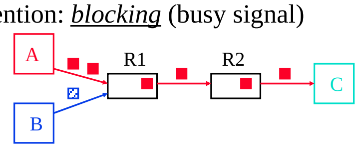
Packet Switched

- Data entering network divided into chunks called "packets" packets traversing network share network resources (e.g., link bandwidth, buffers) with other packets. Resource use: statistical resource sharing
- Resource demands may exceed available resources :
  - A and B packets arrive at R1, destined for C : <u>*resource contention:*</u> queuing (waiting), <u>*delay*</u> are random



Circuit Switched

- All resources (e.g. communication links) needed by the call dedicated to that call for duration example: telephone network
- Resource demands may exceed available resources; A and B want to call C: resource contention: <u>*blocking*</u> (busy signal)

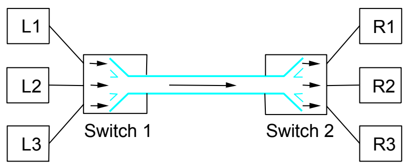


Addressing and Routing

- Address: byte-string that identifies a node
  - usually unique
- Routing: process of forwarding messages to the destination node based on its address
- Types of addresses
  - unicast: node-specific
  - broadcast: all nodes on the network
  - multicast: some subset of nodes on the network

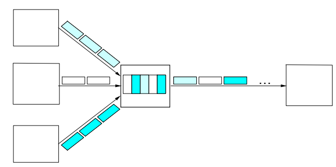
Multiplexing

- Time-Division Multiplexing (TDM)
- Frequency-Division Multiplexing (FDM)



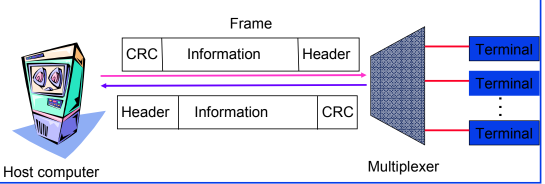
Statistical Multiplexing

- On-demand time-division
- Schedule link on a per-packet basis
- Packets from different sources interleaved on link
- Buffer packets that are *contending* for the link
- Buffer (queue) overflow is called *congestion*



Statistical Multiplexing

- Statistical multiplexer allows a line to carry *frames* that contain messages to/from multiple terminals
- Frames are buffered at *multiplexer* until line becomes available, i.e. store-and-forward
- *Address* in frame header identifies terminal
- Header carries other *control* information

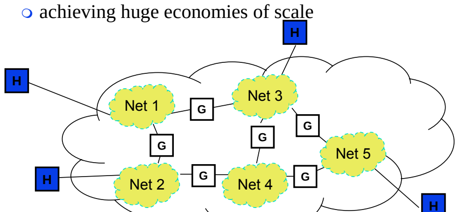


Why statistically share resources?

- Save/make money!
- Example: 1 Mbit/sec link; each user requires 100 Kbits/sec when transmitting; each user has data to send only 10% of time.
  - *circuit-switching:* give each caller 100 Kbits/sec capacity. Can support 10 callers.
  - *packet-switching:* with 35 ongoing calls, probability that 10 or more callers simultaneously active < 0.0004!
- Can support many more callers, with small probability of` "contention."

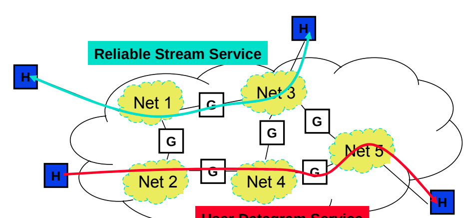
Why Internetworking?

- To build a “network of networks” or internet
  - operating over multiple, coexisting, different network technologies
  - providing ubiquitous connectivity through IP packet transfer
  - achieving huge economies of scale



Why Internetworking?

- To provide *universal communication services*
  - independent of underlying network technologies
  - providing common interface to user applications

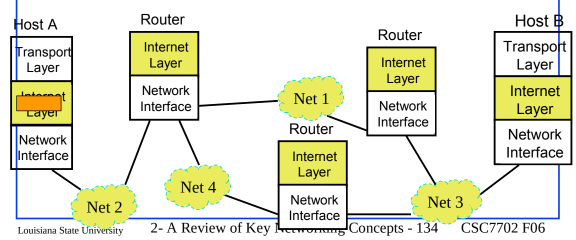


Why Internetworking?

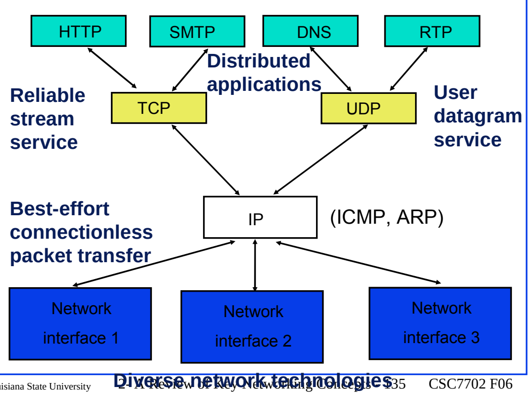
- To provide *distributed applications*
  - Any application designed to operate based on Internet communication services immediately operates across the entire Internet
  - Rapid deployment of new applications
    - Email, WWW, Peer-to-peer
  - Applications independent of network technology
    - New networks can be introduced below
    - Old network technologies can be retired

Internet Protocol Approach

- IP packets transfer information across Internet
  Host A IP → router→ router…→ router→ Host B IP
- IP layer in each router determines next hop (router)
- Network interfaces transfer IP packets across networks

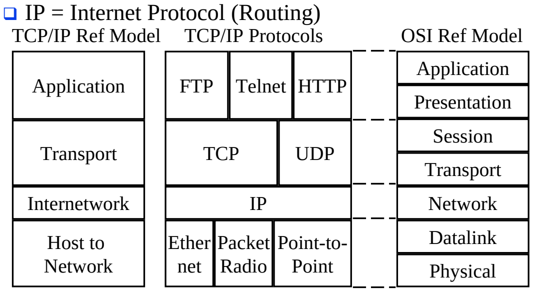


TCP/IP Protocol Suite



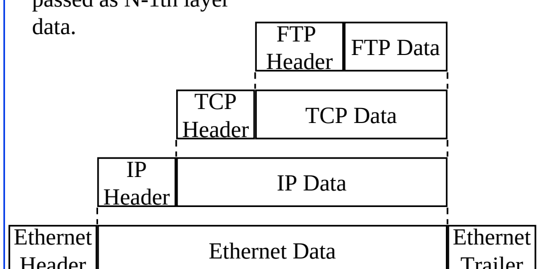
TCP/IP Reference Model

- TCP = Transport Control Protocol
- IP = Internet Protocol (Routing)



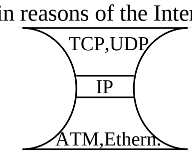
Layered Packet Format

- Nth layer control info is passed as N-1th layer data.



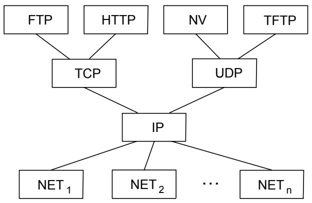
Internet Protocol IP

- Hour-glass model:
  - Glue of the Internet,
  - Everything over IP, and IP over everything
  - The single common language
- Implemented at both hosts and routers
- Accommodating heterogeneity
- Minimalist approach. Best effort datagram service
- One of the main reasons of the Internet's success

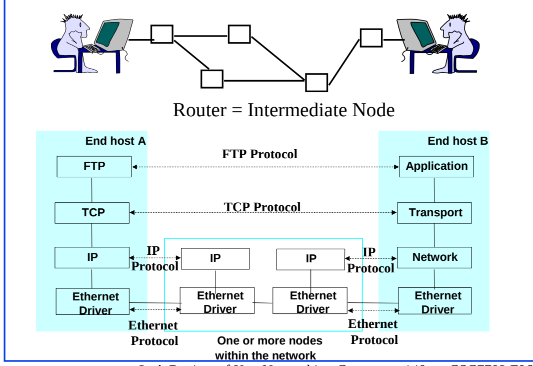


Internet Architecture

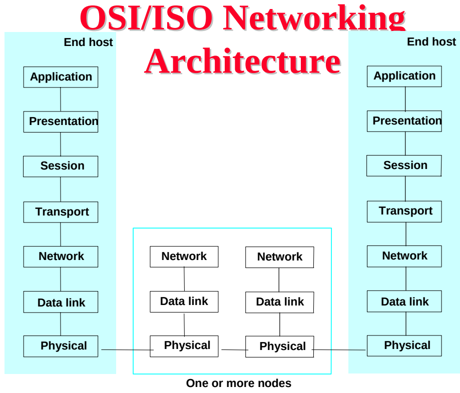
- Defined by Internet Engineering Task Force (IETF)
- Hourglass Design
- Application vs. Application Protocol (FTP, HTTP)



TCP/IP Networking Architecture



OSI/ISO Networking Architecture



OSI vs TCP Reference Models

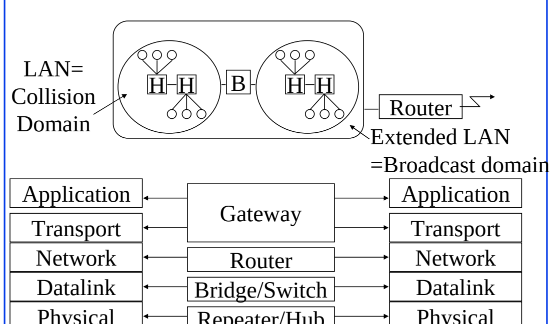
- OSI introduced concept of services, interface, protocols
  These were force-fitted to TCP later
  ⇒ It is not easy to replace protocols in TCP.
- In OSI, reference model was done before protocols.
  In TCP, protocols were done before the model
- OSI: Standardize first, build later
  TCP: Build first, standardize later
- OSI took too long to standardize. TCP/IP was already in wide use by the time.
- OSI became too complex.
- TCP/IP is not general. Ad hoc.

Interconnection Devices

- **Repeater**: PHY device that restores data and collision signals
- **Hub:** Multiport repeater + fault detection and recovery
- **Bridge:** Datalink layer device connecting two or more collision domains. MAC multicasts are propagated throughout "extended LAN."
- **Router:** Network layer device. IP, IPX, AppleTalk. Does not propagate MAC multicasts.
- **Switch**: Multiport bridge with parallel paths

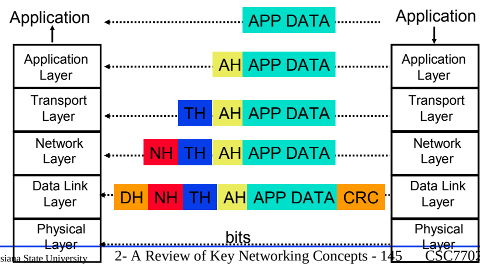
These are functions. Packaging varies.

Interconnection Devices



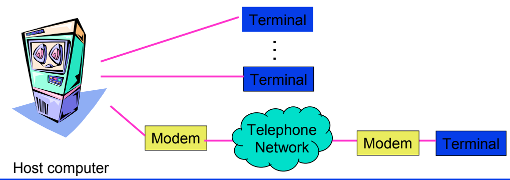
Headers & Trailers

- Each protocol uses a header that carries addresses, sequence numbers, flag bits, length indicators, etc…
- CRC check bits may be appended for error detection



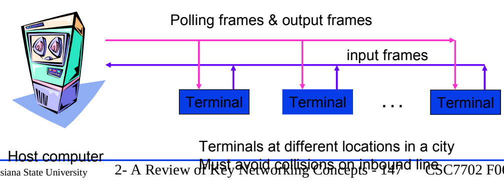
Terminal-Oriented Networks

- Early computer systems very expensive
- Time-sharing methods allowed multiple terminals to share local computer
- Remote access via telephone modems



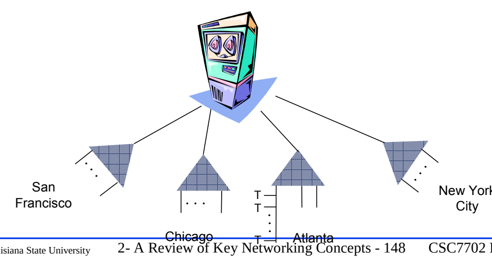
Medium Access Control

- Dedicated communication lines were expensive
- Terminals generated messages sporadically
- Frames carried messages to/from attached terminals
- Address in frame header identified terminal
- *Medium Access Controls* for sharing a line were developed
- Example: Polling protocol on a multidrop line



Tree Topology Networks

- National & international terminal-oriented networks
- Routing was very simple (to/from host)
- Each network typically handled a single application



Computer-to-Computer Networks

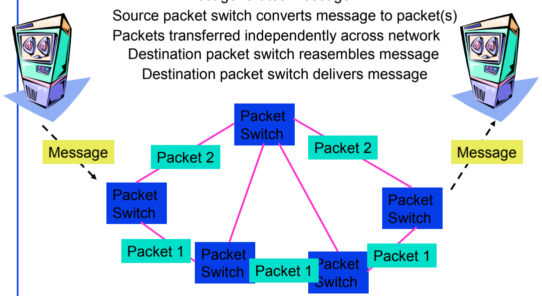
- As cost of computing dropped, terminal-oriented networks viewed as too inflexible and costly
- Need to develop flexible computer networks
  - Interconnect computers as required
  - Support many applications
- Application Examples
  - File transfer between arbitrary computers
  - Execution of a program on another computer
  - Multiprocess operation over multiple computers

Packet Switching

- Network should support multiple applications
  - Transfer arbitrary message size
  - Low delay for interactive applications
  - But in store-and-forward operation, long messages induce high delay on interactive messages
- Packet switching introduced
  - Network transfers packets using store-and-forward
  - Packets have maximum length
  - Break long messages into multiple packets
- ARPANET testbed led to many innovations

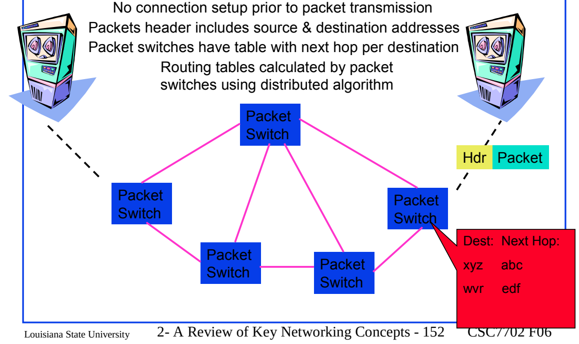
ARPANET Packet Switching

Host generates message
Source packet switch converts message to packet(s)
Packets transferred independently across network
Destination packet switch reasembles message
Destination packet switch delivers message



ARPANET Routing

Routing is highly nontrivial in mesh networks
No connection setup prior to packet transmission
Packets header includes source & destination addresses
Packet switches have table with next hop per destination
Routing tables calculated by packet switches using distributed algorithm

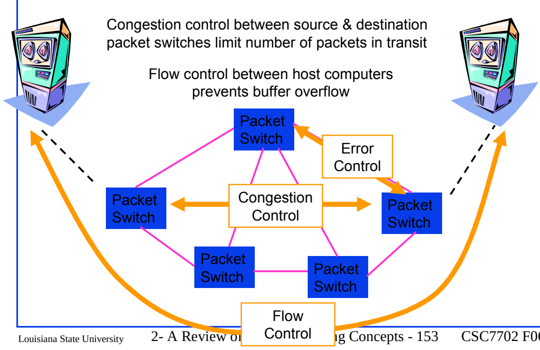


Other ARPANET Protocols

Error control between adjacent packet switches

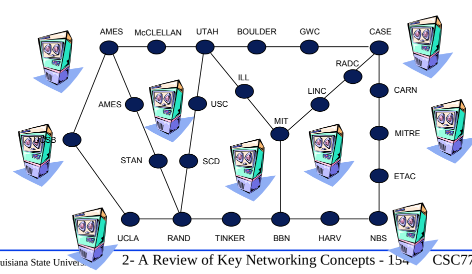
Congestion control between source & destination packet switches limit number of packets in transit

Flow control between host computers prevents buffer overflow



ARPANET Applications

- ARPANET introduced many new applications
- Email, remote login, file transfer, …
- Intelligence at the ***edge***



Historical Maps of Computer Networks

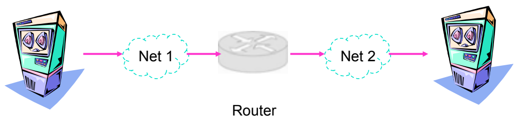
- http://www.cybergeography.org/atlas/historical.html
- The pioneering research of Paul Baran in the 1960s, who envisioned a communications network that would survive a major enemy attacked.
- Donald Davies, a pioneer in networking in the 1960s
- A good book Where Wizards Stay Up Late: The Origins of the Internet, by Katie Hafner and Matthew Lyon

The Internet

- Different network types emerged for data transfer between computers
- ARPA also explored packet switching using satellite and packet radio networks
- Each network has its protocols and is possibly built on different technologies
- *Internetworking protocols* required to enable communications between computers attached to *different* networks
- ***Internet***: a network of networks

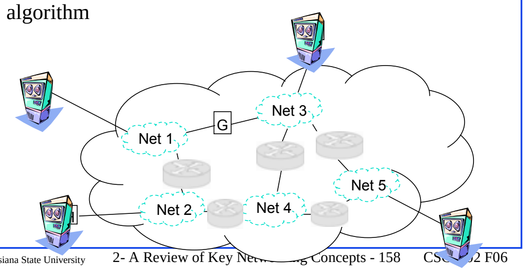
Internet Protocol (IP)

- *Routers (gateways)* interconnect different networks
- Host computers prepare IP packets and transmit them over their attached network
- Routers forward IP packets across networks
- *Best-effort* IP transfer service, no retransmission



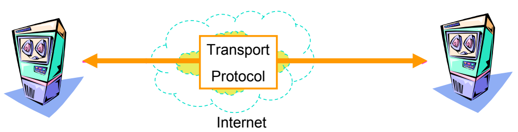
Addressing & Routing

- Hierarchical address: Net ID + Host ID
- IP packets routed according to Net ID
- Routers compute routing tables using distributed algorithm



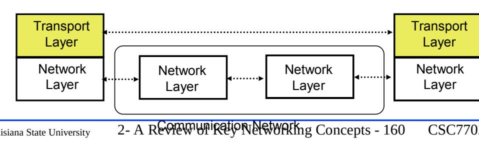
Transport Protocols

- Host computers run two transport protocols on top of IP to enable process-to-process communications
- *User Datagram Protocol* (UDP) enables best-effort transfer of individual block of information
- *Transmission Control Protocol* (TCP) enables reliable transfer of a stream of bytes



Transport Layer

- Transfers data end-to-end from process in a machine to process in another machine
- Reliable stream transfer or quick-and-simple single-block transfer
- Port numbers enable multiplexing
- Message segmentation and reassembly
- Connection setup, maintenance, and release

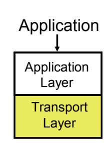


Names and IP Addresses

- Routing is done based on 32-bit IP addresses
- Dotted-decimal notation
  - 128.100.11.1
- Hosts are also identified by name
  - Easier to remember
  - Hierarchical name structure
  - tesla.comm.utoronto.edu
- Domain Name System (DNS) provided conversion between names and addresses

Application & Upper Layers

- Application Layer: Provides services that are frequently required by applications: DNS, web acess, file transfer, email…
- Presentation Layer: machine-independent representation of data…
- Session Layer: dialog management, recovery from errors, …



Incorporated into Application Layer

Internet Applications

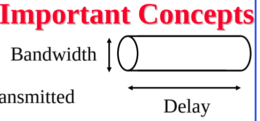
- All Internet applications run on TCP or UDP
- TCP: HTTP (web); SMTP (e-mail); FTP (file transfer; telnet (remote terminal)
- UDP: DNS, RTP (voice & multimedia)
- TCP & UDP incorporated into computer operating systems
- Any application designed to operate over TCP or UDP will run over the Internet!!!

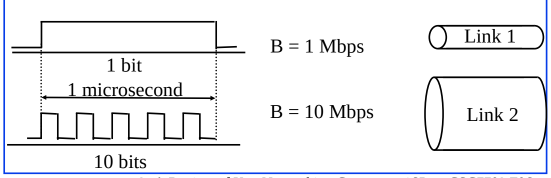
Design Issues for Layers

- Duplexity:
  - Simplex: Transmit or receive
  - Full Duplex: Transmit and receive simultaneously
  - Half-Duplex: Transmit and receive alternately
- Error Control: Make "channel" more reliable; Error detection and recovery
- Flow Control: Avoid flooding slower peer
- Fragmentation: dividing large data chunks into smaller pieces; reassembly
- Multiplexing: several higher level session share single lower level connection
- Addressing/naming: locating, managing identifiers associated with entities

Performance Metrics-Important Concepts

- Throughput
  - Bandwidth vs. throughput
  - Amount of data that can be transmitted per time unit
  - link versus end-to-end
  - Notation: KB = 2^{10} bytes, Mbps = 10^6 bits per second
  - Bandwidth related to "bit width"





Performance Metrics-Important Concepts

- Latency (delay)
  - time to send message from point A to point B
  - one-way versus round-trip time (RTT)
  - Latency = Propagation_time + Transmit_time + +Queuing _time
    - Propagation_time = Distance/SpeedOfLight
    - Transmit_time = SizeOfData/Bandwidth
    - Queuing _time = Processing time in routers, long in case of network congestion when routers buffer the data

Performance Metrics (cont.)

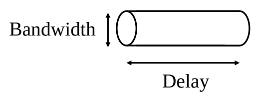
- Speed of light
  - 3.0 x 10^8 meters/second in a vacuum
  - 2.3 x 10^8 meters/second in a cable
  - 2.0 x 10^8 meters/second in a fiber
- No queuing delays in direct link
- Bandwidth not relevant if Size = 1 bit
- Process-to-process latency includes software overhead
- Software overhead can dominate when Distance is small

Performance Metrics (cont.)

- Relative importance of bandwidth and latency
  - small message (e.g., 1 byte): 1ms vs 100ms dominates 1Mbps vs 100Mbps
    With bandwidth = 100Mbs => transmit_time = 0.08 micros
    With bandwidth = 1Mbs => transmit_time = 8 micros
  - large message (e.g., 25 MB): 1Mbps vs 100Mbps dominates 1ms vs 100ms
  - With bandwidth = 1Mbs => transmit_time = 200 s
  - With bandwidth = 100Mbs => transmit_time = 2 s
- High speed networks
  - RTT dominates
    - Throughput = TransferSize / TransferTime
    - TransferTime = RTT + 1/Bandwidth x TransferSize
  - 1-MB *file* to 1-Gbps link as 1-KB *packet* to 1-Mbps link

Performance Metrics (cont.)

- Delay x Bandwidth: amount of data "in the pipe", important concept in network design
  - Example: 100ms x 45Mbps = 560KB



- Application Needs
  - Bandwidth requirements: burst versus peak rate
  - Jitter: variance in latency (inter-packet gap)

Elements of Computer Network Architecture

- *Digital transmission*
- Exchange of *frames* between adjacent equipment
  - Framing and error control
- *Medium access control* regulates sharing of broadcast medium.
- *Addresses* identify attachment to network or internet.
- Transfer of *packets* across a packet network
- Distributed calculation of *routing tables*

Elements of Computer Network Architecture

- *Congestion control* inside the network
- *Internetworking* across multiple networks using routers
- *Segmentation and reassembly* of messages into packets at the ingress to and egress from a network or internetwork
- *End-to-end transport protocols* for process-to-process communications
- *Applications* that build on the transfer of messages between computers.
- *Intelligence is at the edge of the network.*

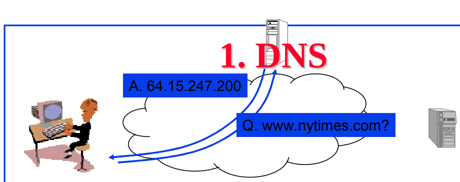
Layers, Services & Protocols

- The overall communications process between two or more machines connected across one or more networks is very complex
- ***Layering*** partitions related communications functions into groups that are manageable
- Each layer provides a ***service*** to the layer above
- Each layer operates according to a ***protocol***
- Let's use examples to show what we mean

Web Browsing Application

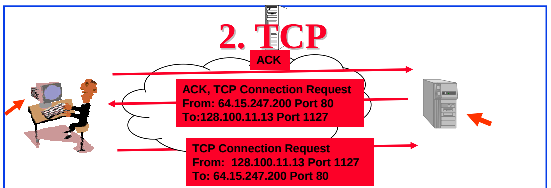
- World Wide Web allows users to access resources (i.e. documents) located in computers connected to the Internet
- Documents are prepared using HyperText Markup Language (HTML)
- A browser application program is used to access the web
- The browser displays HTML documents that include *links* to other documents
- Each link references a *Uniform Resource Locator* (URL) that gives the name of the machine and the location of the given document
- Let's see what happens when a user clicks on a link

1. DNS



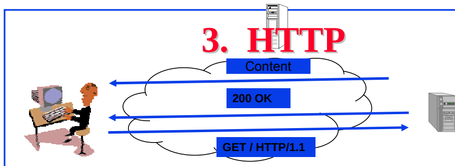
- User clicks on http://www.nytimes.com/
- URL contains Internet name of machine (www.nytimes.com), but not Internet address
- Internet needs Internet address to send information to a machine
- Browser software uses Domain Name System (DNS) protocol to send query for Internet address
- DNS system responds with Internet address

2. TCP



- Browser software uses HyperText Transfer Protocol (HTTP) to send request for document
- HTTP server waits for requests by listening to a well-known port number (80 for HTTP)
- HTTP client sends request messages through an "ephemeral port number," e.g. 1127
- HTTP needs a Transmission Control Protocol (TCP) connection between the HTTP client and the HTTP server to transfer messages reliably

3. HTTP



- HTTP client sends its request message: "GET …"
- HTTP server sends a status response: "200 OK"
- HTTP server sends requested file
- Browser displays document

- Clicking a link sets off a chain of events across the Internet!
- Let's see how protocols & layers come into play…

Protocols

- A *protocol* is a set of rules that governs how two or more communicating entities in a layer are to interact
- *Messages* that can be sent and received
- *Actions* that are to be taken when a certain event occurs, e.g. sending or receiving messages, expiry of timers
- **The purpose of a protocol is to provide a service to the layer above**

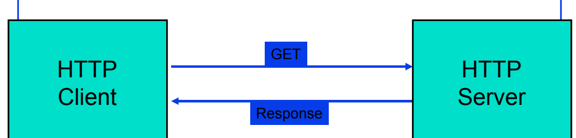
Layers

- A set of related communication functions that can be managed and grouped together
- Application Layer: communications functions that are used by application programs
  - HTTP, DNS, SMTP (email)
- Transport Layer: end-to-end communications between two processes in two machines
  - TCP, User Datagram Protocol (UDP)
- Network Layer: node-to-node communications between two machines
  - Internet Protocol (IP)

Example: HTTP

- HTTP is an application layer protocol
- Retrieves documents on behalf of a browser application program
- HTTP specifies fields in request messages and response messages
  - Request types; Response codes
  - Content type, options, cookies, …
- HTTP specifies actions to be taken upon receipt of certain messages

HTTP Protocol



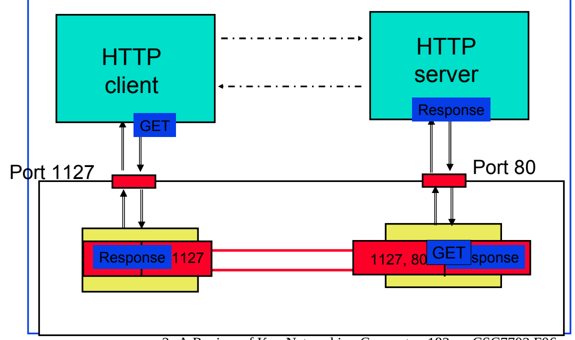
- HTTP assumes messages can be exchanged directly between HTTP client and HTTP server
- In fact, HTTP client and server are processes running in two different machines across the Internet
- HTTP uses the reliable stream transfer service provided by TCP

Example: TCP

- TCP is a transport layer protocol
- Provides *reliable byte stream service* between two processes in two computers across the Internet
- Sequence numbers keep track of the bytes that have been transmitted and received
- Error detection and retransmission used to recover from transmission errors and losses
- TCP is *connection-oriented*: the sender and receiver must first establish an association and set initial sequence numbers before data is transferred
- Connection ID is specified uniquely by

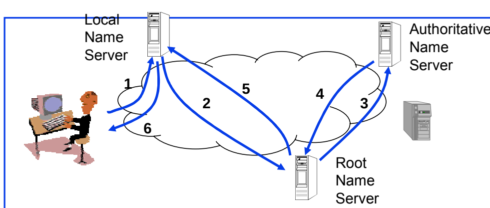
(send port #, send IP address, receive port #, receiver IP address)

HTTP uses service of TCP



Example: DNS Protocol

- DNS protocol is an application layer protocol
- DNS is a distributed database that resides in multiple machines in the Internet
- DNS protocol allows queries of different types
  - Name-to-address or Address-to-name
  - Mail exchange
- DNS usually involves short messages and so uses service provided by UDP
- Well-known port 53



- Local Name Server: resolve frequently-used names
  - University department, ISP
  - Contacts Root Name server if it cannot resolve query
- Root Name Servers: 13 globally
  - Resolves query or refers query to Authoritative Name Server
- Authoritative Name Server: last resort
  - Every machine must register its address with at least two authoritative name servers

Example: UDP

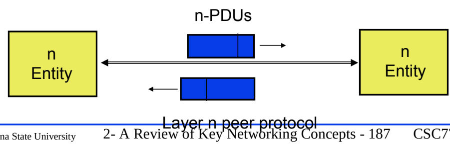
- UDP is a transport layer protocol
- Provides *best-effort datagram service* between two processes in two computers across the Internet
- Port numbers distinguish various processes in the same machine
- UDP is *connectionless*
- Datagram is sent immediately
- Quick, simple, but not reliable

Summary

- Layers: related communications functions
  - Application Layer: HTTP, DNS
  - Transport Layer: TCP, UDP
  - Network Layer: IP
- Services: a protocol provides a communications service to the layer above
  - TCP provides connection-oriented reliable byte transfer service
  - UDP provides best-effort datagram service
- Each layer builds on services of lower layers
  - HTTP builds on top of TCP
  - DNS builds on top of UDP
  - TCP and UDP build on top of IP

OSI Unified View: Protocols

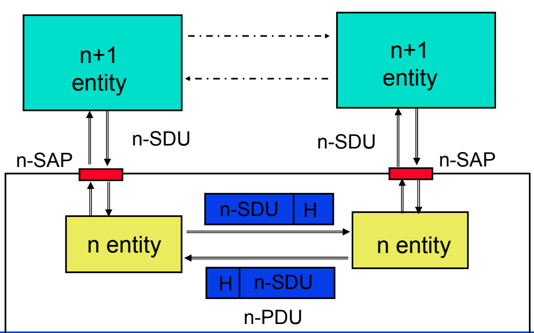
- Layer n in one machine interacts with layer n in another machine to provide a service to layer n +1
- The entities comprising the corresponding layers on different machines are called *peer processes.*
- The machines use a set of rules and conventions called the *layer-n protocol*.
- Layer-n peer processes communicate by exchanging *Protocol Data Units* (PDUs)



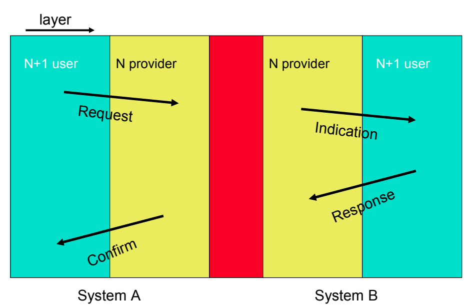
OSI Unified View: Services

- Communication between peer processes is virtual and actually indirect
- Layer n+1 transfers information by invoking the services provided by layer n
- Services are available at *Service Access Points (*SAP's)
- Each layer passes data & control information to the layer below it until the physical layer is reached and transfer occurs
- The data passed to the layer below is called a *Service Data Unit* (SDU)
- SDU's are *encapsulated* in PDU's

Layers, Services & Protocols



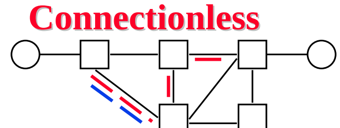
Interlayer Interaction



Connectionless & Connection-Oriented Services

- Connection-Oriented
  - Three-phases:
    1. Connection setup between two SAPs to initialize state information
    2. SDU transfer
    3. Connection release
  - E.g. TCP, ATM
- Connectionless
  - Immediate SDU transfer
  - No connection setup
  - E.g. UDP, IP
- Layered services need not be of same type
  - TCP operates over IP
  - IP operates over ATM

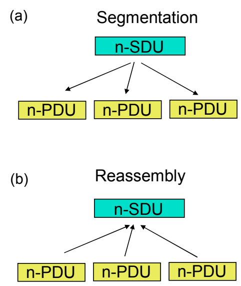
Connection-Oriented vs Connectionless



- Connection-Oriented: Telephone System
  - Path setup before data is sent
  - Data need not have address. Circuit number is sufficient.
- Connectionless: Postal System.
  - Complete address on each packet
  - The address decides the next hop at each router

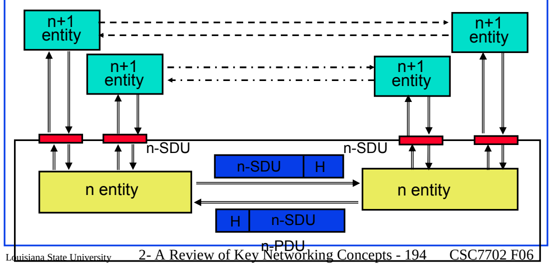
Segmentation & Reassembly

- A layer may impose a limit on the size of a data block that it can transfer for implementation or other reasons
- Thus a layer-n SDU may be too large to be handled as a single unit by layer-(n-1)
- Sender side: SDU is segmented into multiple PDUs
- Receiver side: SDU is reassembled from sequence of PDUs



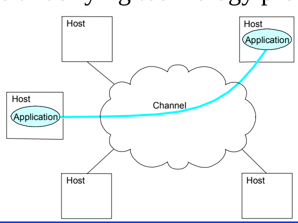
Multiplexing

- Sharing of layer n service by *multiple* layer n+1 users
- Multiplexing tag or ID required in each PDU to determine which users an SDU belongs to



Inter-Process Communication

- Turn host-to-host connectivity into process-to-process communication.
- Fill gap between what applications expect and what the underlying technology provides.



Network Support for Applications

- Network supports common process-to-process channels;
- Request/Reply: for file access and digital libraries
- Message Stream: for video applications
  - video: sequence of frames
  - resolution: 1/4 TV-size image = 352 x 240 pixels;
  - 24-bit color: frame = (352 x 240 x 24)/8 = 247.5KB;
  - frame rate: 30 fps = 7500KBps = 60Mbps
  - video on-demand versus video-conferencing

Summary

- Layers: related communications functions
  - Application Layer: HTTP, DNS
  - Transport Layer: TCP, UDP
  - Network Layer: IP
- Services: a protocol provides a communications service to the layer above
  - TCP provides connection-oriented reliable byte transfer service
  - UDP provides best-effort datagram service
- Each layer builds on services of lower layers
  - HTTP builds on top of TCP
  - DNS builds on top of UDP
  - TCP and UDP build on top of IP

Internet Names & Addresses

Internet Names

- Each host a a unique name
  - Independent of physical location
  - Facilitate memorization by humans
  - Domain Name
  - Organization under single administrative unit
- Host Name
  - Name given to host computer
- User Name
  - Name assigned to user

leongarcia@comm.utoronto.ca

Internet Addresses

- Each host has globally unique *logical* 32 bit IP address
- Separate address for each physical connection to a network
- Routing decision is done based on destination IP address
- IP address has two parts:
  - *netid* and *hostid*
  - *netid* unique
  - *netid* facilitates routing
- Dotted Decimal Notation:

  int1.int2.int3.int4

  (intj = jth octet)

  128.100.10.13

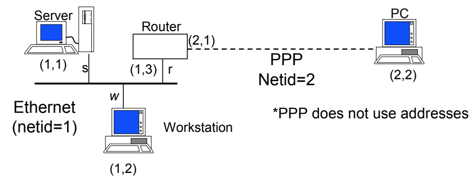
DNS resolves IP name to IP address

Physical Addresses

- LANs (and other networks) assign physical addresses to the physical attachment to the network
- The network uses its own address to transfer packets or frames to the appropriate destination
- IP address needs to be resolved to physical address at each IP network interface
- Example: Ethernet uses 48-bit addresses
  - Each Ethernet network interface card (NIC) has globally unique Medium Access Control (MAC) or physical address
  - First 24 bits identify NIC manufacturer; second 24 bits are serial number
  - 00:90:27:96:68:07 12 hex numbers

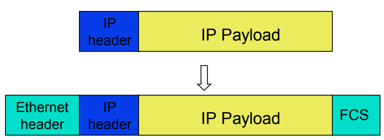
Intel

Example internet



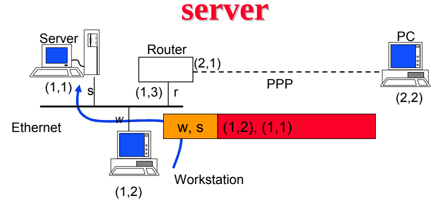
	netid	hostid	Physical address
server	1	1	s
workstation	1	2	w
router	1	3	r
router	2	1	-
PC	2	2	-

Encapsulation



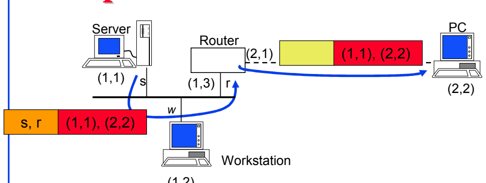
- Ethernet header contains:
  - source and destination physical addresses
  - network protocol type (e.g. IP)

IP packet from workstation to server



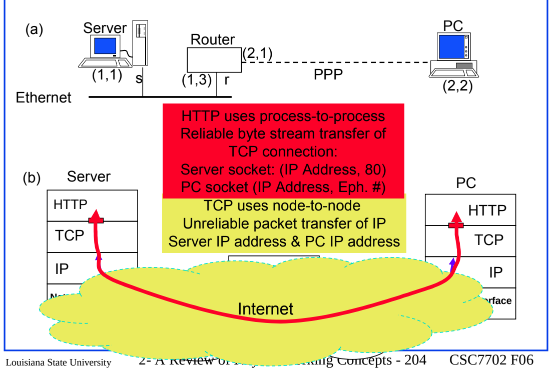
1. IP packet has (1,2) IP address for source and (1,1) IP address for destination
2. IP table at workstation indicates (1,1) connected to same network, so IP packet is encapsulated in Ethernet frame with addresses w and s
3. Ethernet frame is broadcast by workstation NIC and captured by server NIC
4. NIC examines protocol type field and then delivers packet to its IP layer

IP packet from server to PC

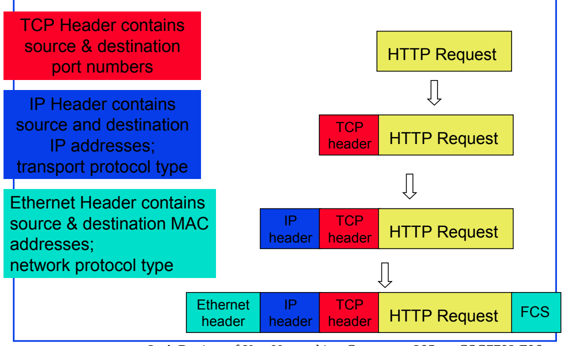


1. IP packet has (1,1) and (2,2) as IP source and destination addresses
2. IP table at server indicates packet should be sent to router, so IP packet is encapsulated in Ethernet frame with addresses s and r
3. Ethernet frame is broadcast by server NIC and captured by router NIC
4. NIC examines protocol type field and then delivers packet to its IP layer
5. IP layer examines IP packet destination address and determines IP packet should be routed to (2,2)
6. Router's table indicates (2,2) is directly connected via PPP link
7. IP packet is encapsulated in PPP frame and delivered to PC
8. PPP at PC examines protocol type field and delivers packet to PC IP layer

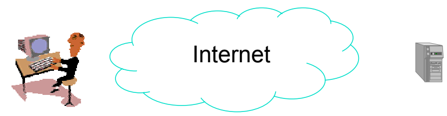
How the layers work together



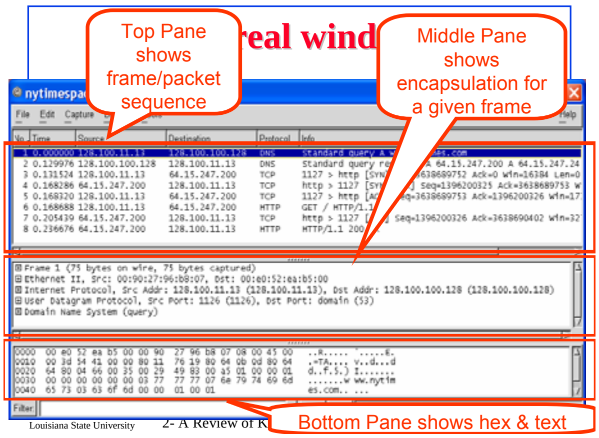
Encapsulation



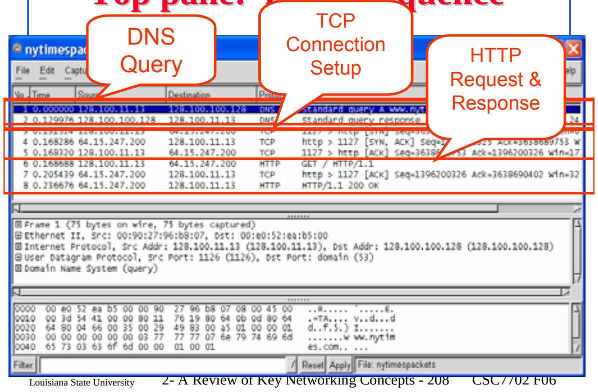
How the layers work together: Network Analyzer Example



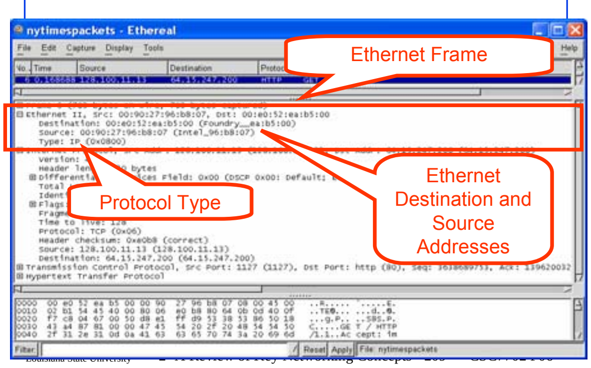
- User clicks on http://www.nytimes.com/
- *Ethereal* network analyzer captures all frames observed by its Ethernet NIC
- Sequence of frames and contents of frame can be examined in detail down to individual bytes



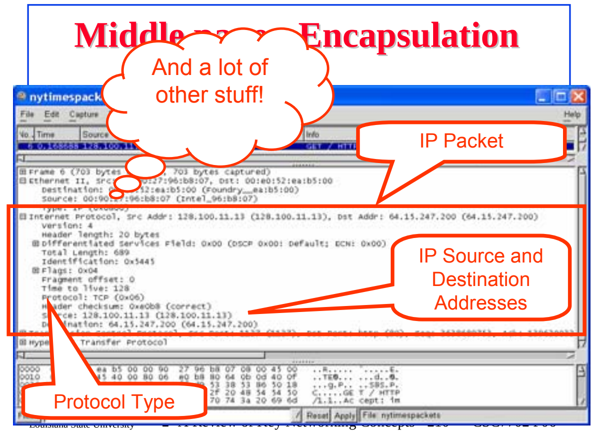
Top pane: frame sequence



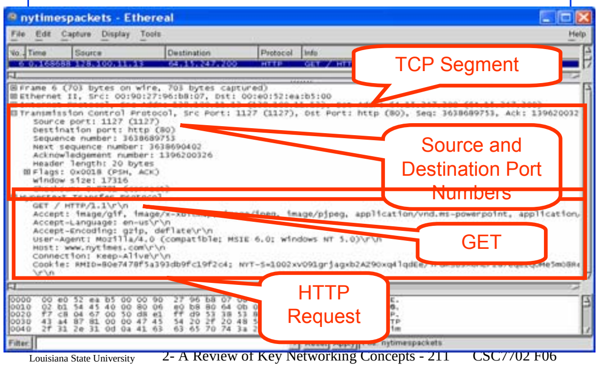
Middle pane: Encapsulation



Middle pane: Encapsulation



Middle pane: Encapsulation



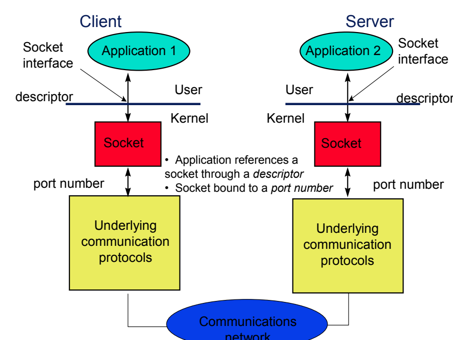
Summary

- Encapsulation is key to layering
- IP provides for transfer of packets across diverse networks
- TCP and UDP provide universal communications services across the Internet
- Distributed applications that use TCP and UDP can operate over the entire Internet
- Internet names, IP addresses, port numbers, sockets, connections, physical addresses

Socket API

- API (Application Programming Interface)
  - Provides a standard set of functions that can be called by applications
- Berkeley UNIX Sockets API
  - Abstraction for applications to send & receive data
  - Applications create sockets that "plug into" network
  - Applications write/read to/from sockets
  - Implemented in the kernel
  - Facilitates development of network applications
  - Hides details of underlying protocols & mechanisms
- Also in Windows, Linux, and other OS's

Communications through Socket Interface



Stream mode of service

Connection-oriented

- First, setup connection between two peer application processes
- Then, reliable bidirectional in-sequence transfer of *byte stream* (boundaries not preserved in transfer)
- Multiple write/read between peer processes
- Finally, connection release
- Uses TCP

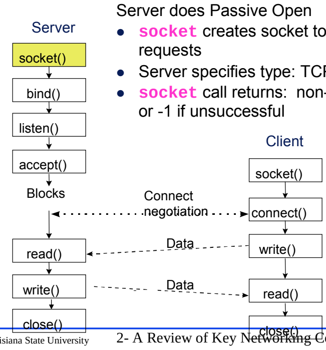
Connectionless

- Immediate transfer of one block of information (boundaries preserved)
- No setup overhead & delay
- Destination address with each block
- Send/receive to/from multiple peer processes
- Best-effort service only
  - Possible out-of-order
  - Possible loss
- Uses UDP

Client & Server Differences

- Server
  - Specifies well-known port # when creating socket
  - May have multiple IP addresses (net interfaces)
  - Waits passively for client requests
- Client
  - Assigned ephemeral port #
  - Initiates communications with server
  - Needs to know server's IP address & port #
    - DNS for URL & server well-known port #
  - Server learns client's address & port #

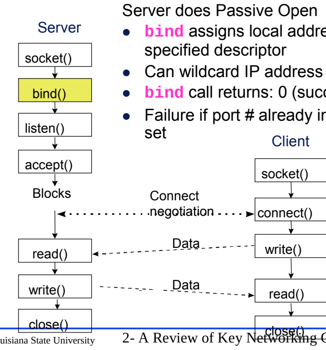
Socket Calls for Connection-Oriented Mode



Server does Passive Open

- **socket** creates socket to *listen* for connection requests
- Server specifies type: TCP (stream)
- **socket** call returns: non-negative integer *descriptor*; or -1 if unsuccessful

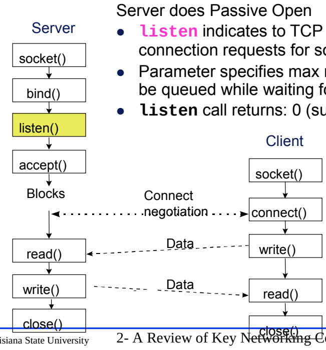
Socket Calls for Connection-Oriented Mode



Server does Passive Open

- **bind** assigns local address & port # to socket with specified descriptor
- Can wildcard IP address for multiple net interfaces
- **bind** call returns: 0 (success); or -1 (failure)
- Failure if port # already in use or if reuse option not set

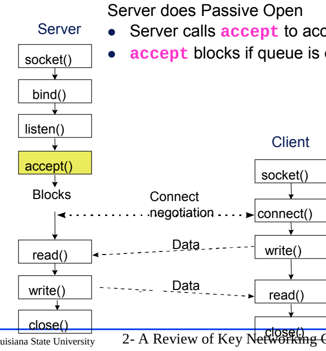
Socket Calls for Connection-Oriented Mode



Server does Passive Open

- **listen** indicates to TCP readiness to receive connection requests for socket with given descriptor
- Parameter specifies max number of requests that may be queued while waiting for server to accept them
- **listen** call returns: 0 (success); or -1 (failure)

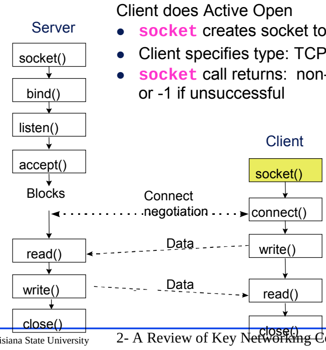
Socket Calls for Connection-Oriented Mode



Server does Passive Open

- Server calls **accept** to accept incoming requests
- **accept** blocks if queue is empty

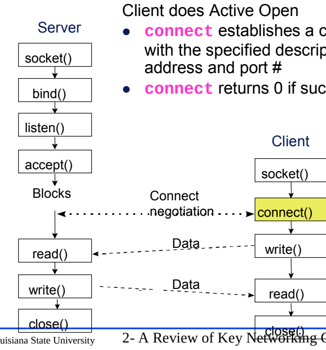
Socket Calls for Connection-Oriented Mode



Client does Active Open

- **socket** creates socket to connect to server
- Client specifies type: TCP (stream)
- **socket** call returns: non-negative integer *descriptor*; or -1 if unsuccessful

Socket Calls for Connection-Oriented Mode



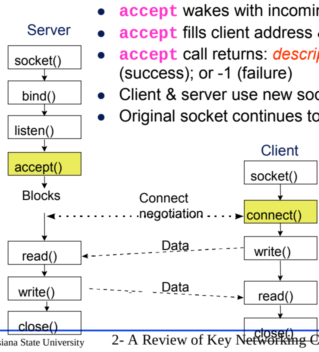
Client does Active Open

- **connect** establishes a connection on the local socket with the specified descriptor to the specified remote address and port #
- **connect** returns 0 if successful; -1 if unsuccessful

Note: **connect** initiates TCP three-way handshake

Socket Calls for Connection-Oriented Mode

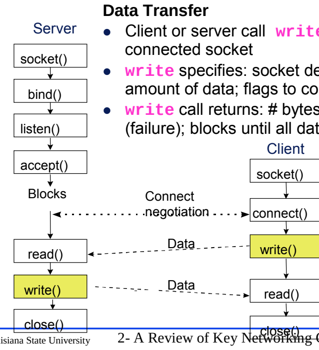
- **accept** wakes with incoming connection request
- **accept** fills client address & port # into address structure
- **accept** call returns: *descriptor of **new** connection socket* (success); or -1 (failure)
- Client & server use new socket for data transfer
- Original socket continues to listen for new requests



Socket Calls for Connection-Oriented Mode

Data Transfer

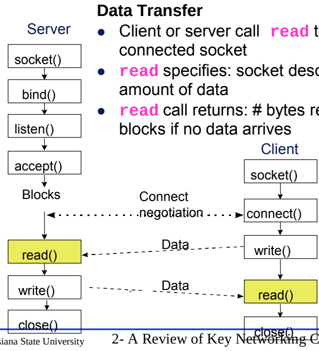
- Client or server call **write** to transmit data into a connected socket
- **write** specifies: socket descriptor; pointer to a buffer; amount of data; flags to control transmission behavior
- **write** call returns: # bytes transferred (success); or -1 (failure); blocks until all data transferred



Socket Calls for Connection-Oriented Mode

Data Transfer

- Client or server call **read** to receive data from a connected socket
- **read** specifies: socket descriptor; pointer to a buffer; amount of data
- **read** call returns: # bytes read (success); or -1 (failure); blocks if no data arrives

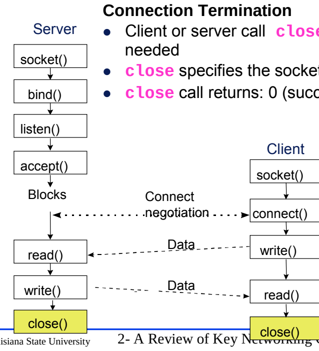


Note: **write** and **read** can be called multiple times to transfer byte streams in both directions

Socket Calls for Connection-Oriented Mode

Connection Termination

- Client or server call **close** when socket is no longer needed
- **close** specifies the socket descriptor
- **close** call returns: 0 (success); or -1 (failure)



Note: **close** initiates TCP graceful close sequence

Example: TCP Echo Server

```
/* A simple echo server using TCP */
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define SERVER_TCP_PORT                3000
#define BUFLEN                         256

int main(int argc, char **argv)
{
        int     n, bytes_to_read;
        int     sd, new_sd, client_len, port;
        struct  sockaddr_in    server, client;
        char    *bp, buf[BUFLEN];

        switch(argc) {
        case 1:
                port = SERVER_TCP_PORT;
                break;
        case 2:
                port = atoi(argv[1]);
                break;
        default:
                fprintf(stderr, "Usage: %s [port]\n", argv[0]);
                exit(1);
        }

        /* Create a stream socket */
        if ((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
                fprintf(stderr, "Can't create a socket\n");
                exit(1);
        }

        /* Bind an address to the socket */
        bzero((char *)&server, sizeof(struct sockaddr_in));
        server.sin_family = AF_INET;
        server.sin_port = htons(port);
        server.sin_addr.s_addr = htonl(INADDR_ANY);
        if (bind(sd, (struct sockaddr *)&server,
        sizeof(server)) == -1) {
                fprintf(stderr, "Can't bind name to socket\n");
                exit(1);
        }

        /* queue up to 5 connect requests */
        listen(sd, 5);

        while (1) {
                client_len = sizeof(client);
                if ((new_sd = accept(sd, (struct sockaddr *)&client,
                &client_len)) == -1) {
                        fprintf(stderr, "Can't accept client\n");
                        exit(1);
                }

                bp = buf;
                bytes_to_read = BUFLEN;
                while ((n = read(new_sd, bp, bytes_to_read)) > 0) {
                        bp += n;
                        bytes_to_read -= n;
                }
                printf("Rec'd: %s\n", buf);

                write(new_sd, buf, BUFLEN);
                printf("Sent: %s\n", buf);
                close(new_sd);
        }
        close(sd);
        return(0);
}
```

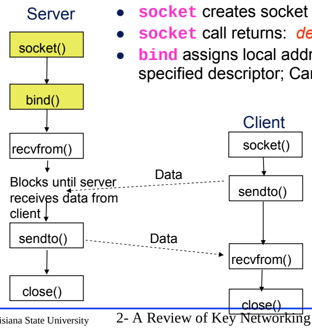

Example: TCP Echo Client

```
/* A simple TCP client */
#include <stdio.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define SERVER_TCP_PORT                3000
#define BUFLEN                         256

int main(int argc, char **argv)
{
        int     n, bytes_to_read;
        int     sd, port;
        struct  hostent        *hp;
        struct  sockaddr_in    server;
        char    *host, *bp, rbuf[BUFLEN], sbuf[BUFLEN];

        switch(argc) {
        case 2:
                host = argv[1];
                port = SERVER_TCP_PORT;
                break;
        case 3:
                host = argv[1];
                port = atoi(argv[2]);
                break;
        default:
                fprintf(stderr, "Usage: %s host [port]\n", argv[0]);
                exit(1);
        }

        /* Create a stream socket */
        if ((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
                fprintf(stderr, "Can't create a socket\n");
                exit(1);
        }

        bzero((char *)&server, sizeof(struct sockaddr_in));
        server.sin_family = AF_INET;
        server.sin_port = htons(port);
        if ((hp = gethostbyname(host)) == NULL) {
                fprintf(stderr, "Can't get server's address\n");
                exit(1);
        }
        bcopy(hp->h_addr, (char *)&server.sin_addr, hp->h_length);

        /* Connecting to the server */
        if (connect(sd, (struct sockaddr *)&server,
        sizeof(server)) == -1) {
                fprintf(stderr, "Can't connect\n");
                exit(1);
        }
        printf("Connected: server's address is %s\n", hp->h_name);

        printf("Transmit:\n");
        gets(sbuf);
                write(sd, sbuf, BUFLEN);

        printf("Receive:\n");
        bp = rbuf;
        bytes_to_read = BUFLEN;
        while ((n = read(sd, bp, bytes_to_read)) > 0) {
                bp += n;
                bytes_to_read -= n;
        }
        printf("%s\n", rbuf);

        close(sd);
        return(0);
}
```

Socket Calls for Connection-Less Mode

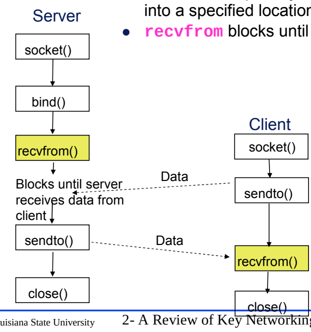
Server started

- `socket` creates socket of type UDP (datagram)
- `socket` call returns: *descriptor*; or -1 if unsuccessful
- `bind` assigns local address & port # to socket with specified descriptor; Can wildcard IP address



Socket Calls for Connection-Less Mode

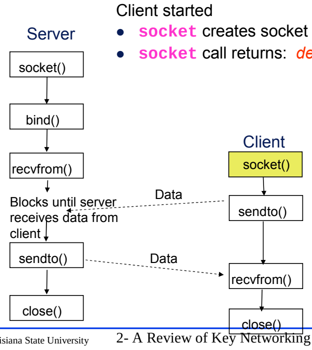
- `recvfrom` copies bytes received in specified socket into a specified location
- `recvfrom` blocks until data arrives



Socket Calls for Connection-Less Mode

Client started

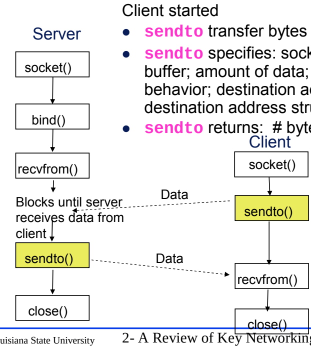
- `socket` creates socket of type UDP (datagram)
- `socket` call returns: *descriptor*; or -1 if unsuccessful



Socket Calls for Connection-Less Mode

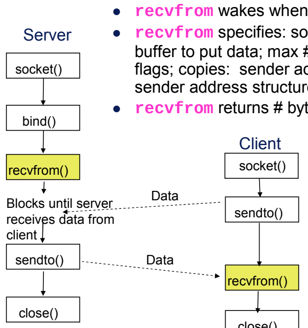
Client started

- `sendto` transfer bytes in buffer to specified socket
- `sendto` specifies: socket descriptor; pointer to a buffer; amount of data; flags to control transmission behavior; destination address & port #; length of destination address structure
- `sendto` returns: # bytes sent; or -1 if unsuccessful



Socket Calls for Connection-Less Mode

- `recvfrom` wakes when data arrives
- `recvfrom` specifies: socket descriptor; pointer to a buffer to put data; max # bytes to put in buffer; control flags; copies: sender address & port #; length of sender address structure
- `recvfrom` returns # bytes received or -1 (failure)

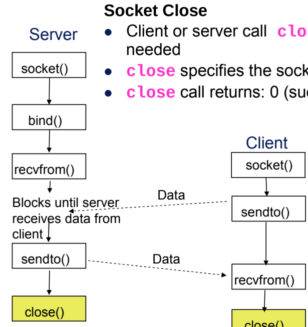


Note: `receivefrom` returns data from at most one `send`, i.e. from one datagram

Socket Calls for Connection-Less Mode

Socket Close

- Client or server call `close` when socket is no longer needed
- `close` specifies the socket descriptor
- `close` call returns: 0 (success); or -1 (failure)



Example: UDP Echo Server

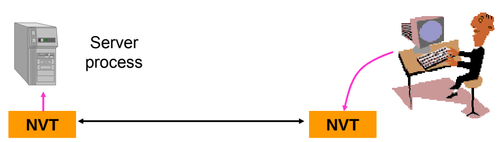
```
/* Echo server using UDP */
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define SERVER_UDP_PORT                 5000
#define MAXLEN                          4096

int main(int argc, char **argv)
{
        int     sd, client_len, port, n;
        char    buf[MAXLEN];
        struct  sockaddr_in     server, client;

        switch(argc) {
        case 1:
                port = SERVER_UDP_PORT;
                break;
        case 2:
                port = atoi(argv[1]);
                break;
        default:
                fprintf(stderr, "Usage: %s [port]\n", argv[0]);
                exit(1);
        }

        /* Create a datagram socket */
        if ((sd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
                fprintf(stderr, "Can't create a socket\n");
                exit(1);
        }

        /* Bind an address to the socket */
        bzero((char *)&server, sizeof(server));
        server.sin_family = AF_INET;
        server.sin_port = htons(port);
        server.sin_addr.s_addr = htonl(INADDR_ANY);
        if (bind(sd, (struct sockaddr *)&server,
        sizeof(server)) == -1) {
                fprintf(stderr, "Can't bind name to socket\n");
                exit(1);
        }

        while (1) {
                client_len = sizeof(client);
                if ((n = recvfrom(sd, buf, MAXLEN, 0,
                (struct sockaddr *)&client, &client_len)) < 0) {
                        fprintf(stderr, "Can't receive datagram\n");
                        exit(1);
                }

                if (sendto(sd, buf, n, 0,
                (struct sockaddr *)&client, client_len) != n) {
                        fprintf(stderr, "Can't send datagram\n");
                        exit(1);
                }
        }
        close(sd);
        return(0);
}
```


Example: UDP Echo Client

```
#include <stdio.h>
#include <string.h>
#include <sys/time.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define SERVER_UDP_PORT         5000
#define MAXLEN                  4096
#define DEFLEN                  64

long delay(struct timeval t1, struct timeval t2)
{
        long d;
        d = (t2.tv_sec - t1.tv_sec) * 1000;
        d += ((t2.tv_usec - t1.tv_usec + 500) / 1000);
        return(d);
}
int main(int argc, char **argv)
{
        int     data_size = DEFLEN, port = SERVER_UDP_PORT;
        int     i, j, sd, server_len;
        char    *pname, *host, rbuf[MAXLEN], sbuf[MAXLEN];
        struct  hostent         *hp;
        struct  sockaddr_in     server;
        struct  timeval         start, end;
        unsigned long address;

        pname = argv[0];
        argc--;
        argv++;
        if (argc > 0 && (strcmp(*argv, "-s") == 0)) {
                if (--argc > 0 && (data_size = atoi(*++argv))) {
                        argc--;
                        argv++;
                }
                else {
                        fprintf(stderr,
                        "Usage: %s [-s data_size] host [port]\n", pname);
                        exit(1);
                }
        }
        if (argc > 0) {
                host = *argv;
                if (--argc > 0)
                        port = atoi(*++argv);
        }
        else {
                fprintf(stderr,
                "Usage: %s [-s data_size] host [port]\n", pname);
                exit(1);
        }

        if ((sd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
                fprintf(stderr, "Can't create a socket\n");
                exit(1);
        }
        bzero((char *)&server, sizeof(server));
        server.sin_family = AF_INET;
        server.sin_port = htons(port);
        if ((hp = gethostbyname(host)) == NULL) {
                fprintf(stderr, "Can't get server's IP address\n");
                exit(1);
        }
        bcopy(hp->h_addr, (char *) &server.sin_addr, hp->h_length);

        if (data_size > MAXLEN) {
                fprintf(stderr, "Data is too big\n");
                exit(1);
        }
        for (i = 0; i < data_size; i++) {
                j = (i < 26) ? i : i % 26;
                sbuf[i] = 'a' + j;
        }
        gettimeofday(&start, NULL); /* start delay measurement */
        server_len = sizeof(server);
        if (sendto(sd, sbuf, data_size, 0, (struct sockaddr *)
                &server, server_len) == -1) {
                fprintf(stderr, "sendto error\n");
                exit(1);
        }
        if (recvfrom(sd, rbuf, MAXLEN, 0, (struct sockaddr *)
                &server, &server_len) < 0) {
                fprintf(stderr, "recvfrom error\n");
                exit(1);
        }
        gettimeofday(&end, NULL); /* end delay measurement */
        if (strncmp(sbuf, rbuf, data_size) != 0)
                printf("Data is corrupted\n");
        close(sd);
        return(0);
```


Telnet (RFC 854)

- Provides general bi-directional byte-oriented TCP-based communications facility (Network Virtual Terminal)
- Initiating machine treated as local to the remote host
- Used to connect to port # of other servers and to interact with them using command line



Network Virtual Terminal

- *Network Virtual Terminal*
- Lowest common denominator terminal
- Each machine maps characteristics to NVT
- Negotiate options for changes to the NVT
- Data input sent to server & echoed back
- Server control functions : interrupt, abort output, are-you-there, erase character, erase line
- Default requires login & password

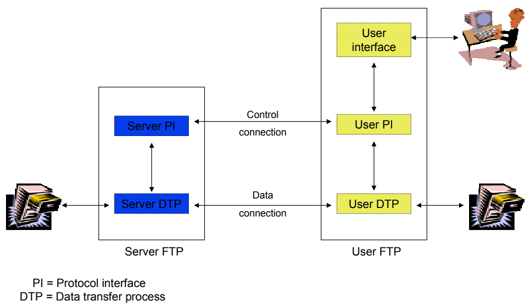
`telnet`

- A program that uses the Telnet protocol
- Establishes TCP socket
- Sends typed characters to server
- Prints whatever characters arrive
- Try it to retrieve a web page (HTTP) or to send an email (SMTP)

File Transfer Protocol (RFC 959)

- Provides for transfer of file from one machine to another machine
- Designed to hide variations in file storage
- FTP parameter commands specify file info
  - File Type: *ASCII*, EBCDIC, image, local.
  - Data Structure: *file*, record, or page
  - Transmission Mode: *stream*, block, compressed
- Other FTP commands
  - Access Control: USER, PASS, CWD, QUIT, …
  - Service: RETR, STOR, PWD, LIST, …

FTP File Transfer



Two TCP Connections

Control connection

- Set up using Telnet protocol on well-known port 21
- FTP commands & replies between protocol interpreters
- PIs control the data transfer process
- User requests close of control connection; server performs the close

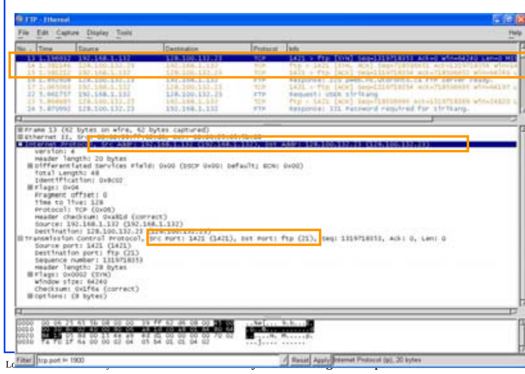
Data connection

- To perform file transfer, obtain lists of files, directories
- Each transfer requires new data connection
- Passive open by user PI with ephemeral port #
- Port # sent over control connection
- Active open by server using port 20

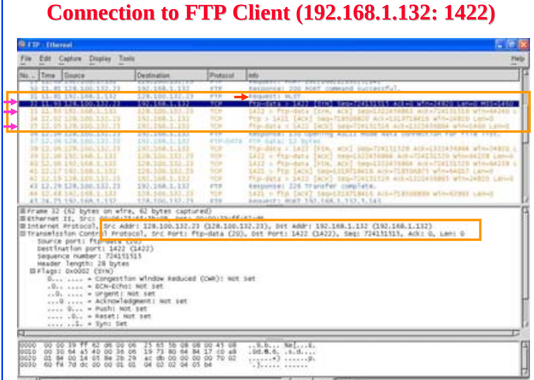
FTP Replies

Reply	Meaning
1yz	Positive preliminary reply (action has begun, but wait for another reply before sending a new command).
2yz	Positive completion reply (action completed successfully; new command may be sent).
3yz	Positive intermediary reply (command accepted, but action cannot be performed without additional information; user should send a command with the necessary information).
4yz	Transient negative completion reply (action currently cannot be performed; resend command later).
5zy	Permanent negative completion reply (action cannot be performed; do not resend it).
x0z	Syntax errors.
x1z	Information (replies to requests for status or help).
x2z	Connections (replies referring to the control and data connections).
x3z	Authentication and accounting (replies for the login process and accounting procedures).
x4z	Unspecified.
x5z	File system status.

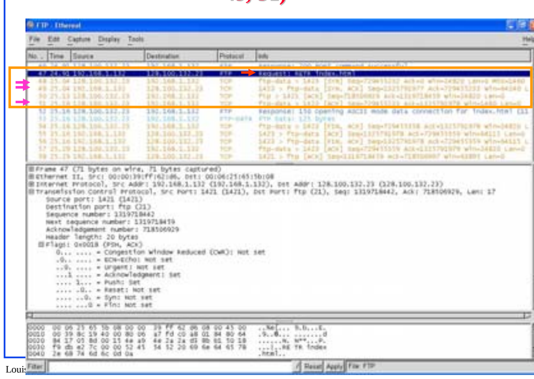
FTP Client (192.168.1.132: 1421) establishes Control Connection to FTP Server (128.100.132.23: 21)



control) FTP Server (128.100.132.23: 20) establishes Data Connection to FTP Client (192.168.1.132: 1422)



control connection (frame 47 request); File transfer on new data connection (port 1423, fr. 48, 49, 51)



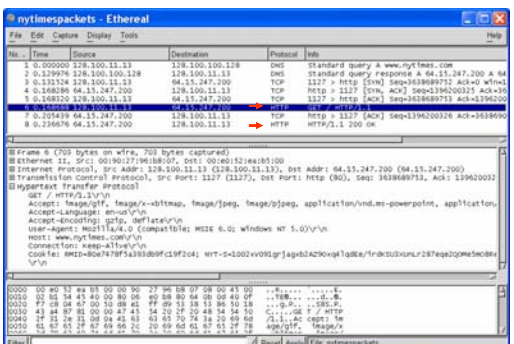
Hypertext Transfer Protocol

- RFC 1945 (HTTP 1.0), RFC 2616 (HTTP 1.1)
- HTTP provides communications between web browsers & web servers
- Web: framework for accessing documents & resources through the Internet
- Hypertext documents: text, graphics, images, hyperlinks
- Documents prepared using Hypertext Markup Language (HTML)

HTTP Protocol

- HTTP servers use well-known port 80
- Client request / Server reply
- Stateless: server does not keep any information about client
- HTTP 1.0 new TCP connection per request/reply (non-persistent)
- HTTP 1.1 persistent operation is default

HTTP Typical Exchange



HTTP Message Formats

- HTTP messages written in ASCII text
- Request Message Format
  - *Request Line (Each line ends with carriage return)*
    - *Method URL HTTP-Version \r\n*
    - Method specifies action to apply to object
    - URL specifies object
  - *Header Lines (Ea. line ends with carriage return)*
    - *Attribute Name: Attribute Value*
    - E.g. type of client, content, identity of requester, …
    - Last header line has extra carriage return)
  - *Entity Body (Content)*

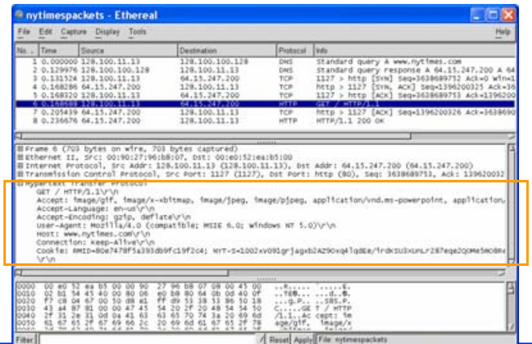
HTTP Request Methods

Request method	Meaning
GET	Retrieve information (object) identified by the URL.
HEAD	Retrieve meta-information about the object, but do not transfer the object; Can be used to find out if a document has changed.
POST	Send information to a URL (using the entity body) and retrieve result; used when a user fills out a form in a browser.
PUT	Store information in location named by URL
DELETE	Remove object identified by URL
TRACE	Trace HTTP forwarding through proxies, tunnels, etc.
OPTIONS	Used to determine the capabilities of the server, or characteristics of a named resource.

Universal Resource Locator

- Absolute URL
  - scheme://hostname[:port]/path
  - http://www.nytimes.com/
- Relative URL
  - /path
  - /

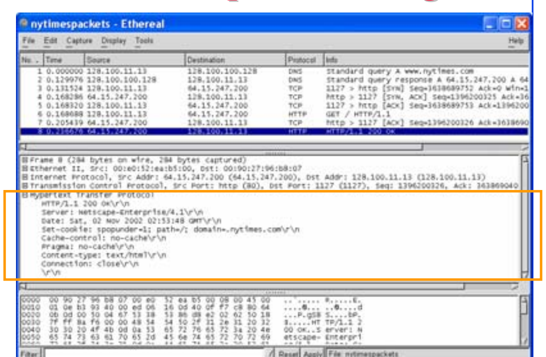
HTTP Request Message



HTTP Response Message

- Response Message Format
  - *Status Line*
    - *HTTP-Version Status-Code Message*
    - Status Code: 3-digit code indicating result
    - E.g. HTTP/1.0 200 OK
  - *Headers Section*
    - Information about object transferred to client
    - E.g. server type, content length, content type, …
  - *Content*
    - Object (document)

HTTP Response Message



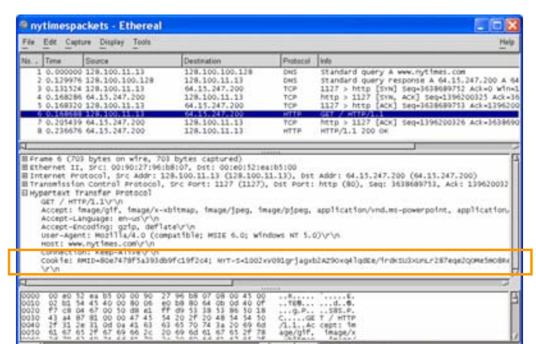
HTTP Proxy Server & Caching

- Web users generate large traffic volumes
- Traffic causes congestion & delay
- Can improve delay performance and reduce traffic in Internet by moving content to servers closer to the user
- Web proxy servers cache web information
  - Deployed by ISPs
  - Customer browsers configured to first access ISPs proxy servers
  - Proxy replies immediately when it has requested object or retrieves the object if it does not

Cookies and Web Sessions

- Cookies are data exchanged by clients & servers as header lines
- Since HTTP stateless, cookies can provide context for HTTP interaction
- *Set cookie* header line in reply message from server + unique ID number for client
- If client accepts cookie, cookie added to client's cookie file (must include expiration date)
- Henceforth client requests include ID
- Server site can track client interactions, store these in a separate database, and access database to prepare appropriate responses

Cookie Header Line; ID is 24 hexadecimal numeral



PING

- Application to determine if host is reachable
- Based on Internet Control Message Protocol
  - ICMP informs source host about errors encountered in IP packet processing by routers or by destination host
  - ICMP Echo message requests reply from destination host
- PING sends echo message & sequence #
- Determines reachability & round-trip delay
- Sometimes disabled for security reasons

PING from NAL host

```
Microsoft(R) Windows DOS
(c)Copyright Microsoft Corp 1990-2001.

C:\DOCUME~1\1>ping nal.toronto.edu

Pinging nal.toronto.edu [128.100.244.3] with 32 bytes of data:

Reply from 128.100.244.3: bytes=32 time=84ms TTL=240
Reply from 128.100.244.3: bytes=32 time=110ms TTL=240
Reply from 128.100.244.3: bytes=32 time=81ms TTL=240
Reply from 128.100.244.3: bytes=32 time=79ms TTL=240

Ping statistics for 128.100.244.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 79ms, Maximum = 110ms, Average = 88ms

C:\DOCUME~1\1>
```


Traceroute

- Find route from local host to a remote host
- Time-to-Live (TTL)
  - IP packets have TTL field that specifies maximum # hops traversed before packet discarded
  - Each router decrements TTL by 1
  - When TTL reaches 0 packet is discarded
- Traceroute
  - Send UDP to remote host with TTL=1
  - First router will reply ICMP Time Exceeded Msg
  - Send UDP to remote host with TTL=2, …
  - Each step reveals next router in path to remote host

Traceroute from home PC to university host

```
Tracing route to www.comm.utoronto.ca [128.100.11.60]
over a maximum of 30 hops:

  1     1 ms   <10 ms   <10 ms  192.168.2.1
  2     3 ms     3 ms     3 ms  10.202.128.1
  3     4 ms     3 ms     3 ms  gw04.ym.phub.net.cable.rogers.com [66.185.83.142]
  4     *        *        *     Request timed out.
  5    47 ms    59 ms    66 ms  gw01.bloor.phub.net.cable.rogers.com [66.185.80.230]
  6     3 ms     3 ms    38 ms  gw02.bloor.phub.net.cable.rogers.com [66.185.80.242]
  7     8 ms     3 ms     5 ms  gw01.wlfdle.phub.net.cable.rogers.com [66.185.80.2]
  8     8 ms     7 ms     7 ms  gw02.wlfdle.phub.net.cable.rogers.com [66.185.80.142]
  9     4 ms    10 ms     4 ms  gw01.front.phub.net.cable.rogers.com [66.185.81.18]
 10     6 ms     4 ms     5 ms  ra1sh-ge3-4.mt.bigpipeinc.com [66.244.223.237]
 11    16 ms    17 ms    13 ms  rx0sh-hydro-one-telecom.mt.bigpipeinc.com [66.244.223.246]
 12     7 ms    14 ms     8 ms  142.46.4.2
 13    10 ms     7 ms     6 ms  utorgw.onet.on.ca [206.248.221.6]
 14     7 ms     6 ms    11 ms  mcl-gateway.gw.utoronto.ca [128.100.96.101]
 15     7 ms     5 ms     8 ms  sf-gpb.gw.utoronto.ca [128.100.96.17]
 16     7 ms     7 ms    10 ms  bi15000.ece.utoronto.ca [128.100.96.236]
 17     7 ms     9 ms     9 ms  www.comm.utoronto.ca [128.100.11.60]

Trace complete.
```

Home Network
Rogers Cable ISP
Shaw Net
Hydro One
Ontario Net
University of Toronto

ipconfig

- Utility in Microsoft® Windows to display TCP/IP information about a host
- Many options
  - Simplest: IP address, subnet mask, default gateway for the host
  - Information about each IP interface of a host
    - DNS hostname, IP addresses of DNS servers, physical address of network card, IP address, …
  - Renew IP address from DHCP server

netstat

- Queries a host about TCP/IP network status
- Status of network drivers & their interface cards
  - #packets in, #packets out, errored packets, …
- State of routing table in host
- TCP/IP active server processes
- TCP active connections

`netstat` protocol statistics

```
IPv4 Statistics

  Packets Received                   = 71271
  Received Header Errors             = 0
  Received Address Errors            = 9
  Datagrams Forwarded                = 0
  Unknown Protocols Received         = 0
  Received Packets Discarded         = 0
  Received Packets Delivered         = 71271
  Output Requests                    = 70138
  Routing Discards                   = 0
  Discarded Output Packets           = 0
  Output Packet No Route             = 0
  Reassembly Required                = 0
  Reassembly Successful              = 0
  Reassembly Failures                = 0
  Datagrams Successfully Fragmented  = 0
  Datagrams Failing Fragmentation    = 0
  Fragments Created                  = 0

UDP Statistics for IPv4

  Datagrams Received    = 6810
  No Ports              = 15
  Receive Errors        = 0
  Datagrams Sent        = 6309

ICMPv4 Statistics

                            Received    Sent
  Messages                  10          6
  Errors                    0           0
  Destination Unreachable   8           1
  Time Exceeded             0           0
  Parameter Problems        0           0
  Source Quenches           0           0
  Redirects                 0           0
  Echos                     0           2
  Echo Replies              2           0
  Timestamps                0           0
  Timestamp Replies         0           0
  Address Masks             0           0
  Address Mask Replies      0           0

TCP Statistics for IPv4

  Active Opens                          = 798
  Passive Opens                         = 17
  Failed Connection Attempts            = 13
  Reset Connections                     = 467
  Current Connections                   = 0
  Segments Received                     = 64443
  Segments Sent                         = 63724
  Segments Retransmitted                = 80
```


`tcpdump` and Network Protocol Analyzers

- `tcpdump` program captures IP packets on a network interface (usually Ethernet NIC)
- Filtering used to select packets of interest
- Packets & higher-layer messages can be displayed and analyzed
- `tcpdump` basis for many network protocol analyzers for troubleshooting networks
- We use the open source Ethereal analyzer to generate examples
  - www.ethereal.com

Summary

- ISO/OSI reference model has seven layers. TCP/IP Protocol suite has four layers.
- Ethernet/IEEE 802.3 uses CSMA/CD.
- Addresses: Local vs Global, Unicast vs Broadcast.

Thank You!