CS 6114
Conventional Networking

Nate Foster
Cornell University

Based on slides from Jen Rexford
Packet Switching
Life in the 1970s...

Multiple unconnected networks
- ARPAnet, data-over-cable, packet satellite (Aloha), packet radio, ...

Heterogeneous designs
- Addressing, max packet size, handling of lost/corrupted data, fault detection, routing, ...
Handling Heterogeneity

Where to handle heterogeneity?
- Application process? End hosts? Packet switches?

Compatible process and host conventions
- Obviate the need to support all combinations

Retain the unique features of each network
- Avoid changing the local network components

Introduce the notion of a gateway
Internetwork Layer and Gateways

Internetwork appears as a single, uniform entity
Despite the heterogeneity of the local networks
Network of networks

“Embed internetwork packets in local packet format or extract them”
Route (at internetwork level) to next gateway

ARPAnet

Gateway

satellite net
Internetwork Packet Format

Internetwork header in standard format
- Interpreted by the gateways and end hosts

Source and destination addresses
- Uniformly and uniquely identify every host

Ensure proper sequencing of the data
- Include a sequence number and byte count

Enable detection of corrupted text
- Checksum for an end-to-end check on the text
Layering
Socket Abstraction

Best-effort packet delivery is a clumsy abstraction
- Applications typically want higher-level abstractions
- Messages, uncorrupted data, reliable in-order delivery

Applications communicate using “sockets”
- Stream socket: reliable stream of bytes (like a file)
- Message socket: unreliable message delivery
Layers and Routers

HTTP message

TCP segment

IP packet
Naming
## Three Kinds of Identifiers

<table>
<thead>
<tr>
<th>Host Name</th>
<th>IP Address</th>
<th>MAC Address</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example</strong></td>
<td><strong><a href="http://www.cs.cornell.edu">www.cs.cornell.edu</a></strong></td>
<td><strong>132.236.204.10</strong></td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>Hierarchical, human readable, variable length</td>
<td>Hierarchical, machine readable, 32 bits</td>
</tr>
<tr>
<td><strong>Read by</strong></td>
<td>Humans, hosts</td>
<td>IP routers</td>
</tr>
<tr>
<td><strong>Allocation, top-level</strong></td>
<td>Domain, assigned by registrar (e.g., for .edu)</td>
<td>Variable-length prefixes, assigned by ICANN, RIR, or ISP</td>
</tr>
<tr>
<td><strong>Allocation, low-level</strong></td>
<td>Host name, local administrator</td>
<td>Interface, by admin or DHCP</td>
</tr>
</tbody>
</table>
Mapping Between Identifiers

Dynamic Host Configuration Protocol (DHCP)
- Given a MAC address, assigns a unique IP address
- … and gives host other information about the local network (e.g., gateway)
- Automates the boot-strapping process

Address Resolution Protocol (ARP)
- Given an IP address, provides the MAC address
- Enables communication within the local network

Domain Name System (DNS)
- Given a host name, provides the IP address
- Given an IP address, provides the host name
Learning a Host’s Address

Who am I?

- Hard-wired: MAC address
- Static configuration: IP interface configuration
- Dynamically learned: IP address configured by DHCP

Who are you?

- Hard-wired: IP address in a URL, or in the code
- Dynamically looked up: ARP or DNS
Address Resolution Protocol (ARP)

Every host maintains an ARP table
- (IP address, MAC address) pair

Consult the table when sending a packet
- Map destination IP address to destination MAC address
- Encapsulate and transmit the data packet

But, what if the IP address is not in the table?
- Sender broadcasts: “Who has IP address 1.2.3.156?”
- Receiver responds: “MAC address 58-23-D7-FA-20-B0”
- Sender caches the result in its ARP table
Dynamic Host Configuration Protocol

Host learns
IP address, Subnet mask, Gateway address, DNS server(s), and a lease time.
Domain Name System

Host at cs.cornell.edu wants IP address for gaia.cs.umass.edu

Recursive query: #1
Iterative queries: #2, 4, 6
Host at cs.cornell.edu wants IP address for gaia.cs.umass.edu

Recursive query: #1
Iterative queries: #2, #4, #6
Host at cs.cornell.edu wants IP address for gaia.cs.umass.edu

Recursive query: #1
Iterative queries: #2, 4, 6
Domain Name System

Host at cs.cornell.edu wants IP address for gaia.cs.umass.edu

Recursive query: #1
Iterative queries: #2, 4, 6
Domain Name System

Host at cs.cornell.edu wants IP address for gaia.cs.umass.edu

Recursive query: #1
Iterative queries: #2, 4, 6
Host at cs.cornell.edu wants IP address for gaia.cs.umass.edu

Recursive query: #1
Iterative queries: #2, 4, 6
Host at cs.cornell.edu wants IP address for gaia.cs.umass.edu

Recursive query: #1
Iterative queries: #2, 4, 6
Host at cs.cornell.edu wants IP address for gaia.cs.umass.edu

Recursive query: #1
Iterative queries: #2, 4, 6
Host at cs.cornell.edu wants IP address for gaia.cs.umass.edu

Recursive query: #1
Iterative queries: #2, 4, 6
Transport
Two Basic Transport Features
Two Basic Transport Features

Demultiplexing: port numbers

Client host

Service request for 128.2.194.242:80 (i.e., the Web server)

Server host 128.2.194.242

Web server (port 80)

Echo server (port 7)
Two Basic Transport Features

Demultiplexing: port numbers

Service request for 128.2.194.242:80 (i.e., the Web server)

Error detection: checksums

detect corruption
Two Transport Protocols
Two Transport Protocols

User Datagram Protocol (UDP)

- Just provides demultiplexing and error detection
- Header fields: port numbers, checksum, and length
- Low overhead, good for query/response and multimedia
Two Transport Protocols

User Datagram Protocol (UDP)
- Just provides demultiplexing and error detection
- Header fields: port numbers, checksum, and length
- Low overhead, good for query/response and multimedia

Transmission Control Protocol (TCP)
- Provides a “stream of bytes” abstraction
- Retransmits lost or corrupted data
- Puts out-of-order data back in order
- Adapts the sending rate to alleviate congestion
- Higher overhead, good for most stateful applications
Resource Allocation Challenges

Best-effort network easily becomes overloaded
- No mechanism to “block” excess calls
- Instead excess packets are simply dropped

Examples
- Shared Ethernet medium: frame collisions
- Ethernet switches and IP routers: full packet buffers

Quickly leads to congestion collapse

Increase in load that results in a decrease in useful work done.
End Hosts Adjusting to Congestion

End hosts adapt their sending rates
- In response to network conditions

Learning that the network is congested
- Shared Ethernet: carrier sense multiple access
  - Seeing your own frame collide with others
- IP network: observing your end-to-end performance
  - Packet delay or loss over the end-to-end path

Adapting to congestion
- Slowing down the sending rate, for the greater good
- But, host doesn’t know how bad things might be…
TCP Congestion Control

Additive increase, multiplicative decrease
- On packet loss, divide congestion window in half
- On success for last window, increase window linearly

Other mechanisms: slow start, fast retransmit vs. timeout loss, etc.