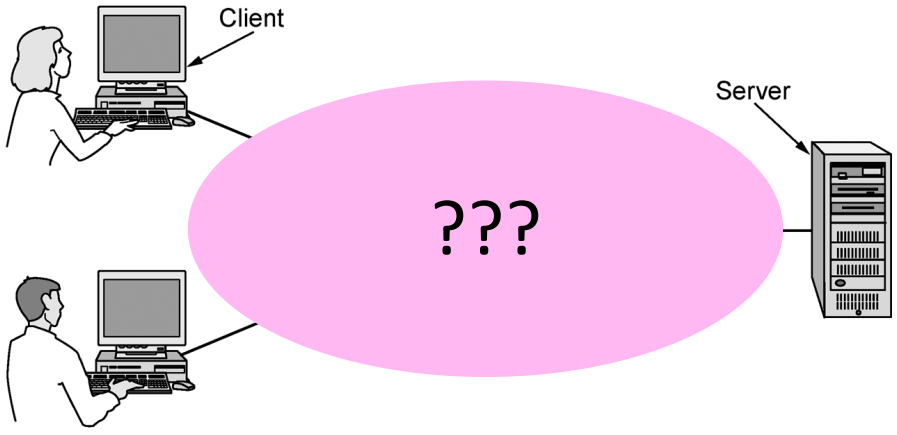
Computer Networks

Arvind Krishnamurthy

Material based on courses at Stanford, Princeton, and MIT

1

Focus of the course



2

Focus of the course (2)

- Three "networking" topics:
  - Communications
  - Networking
  - Distributed systems
- Our focus is on the "middle" layer

3

But, What *is* Networking?

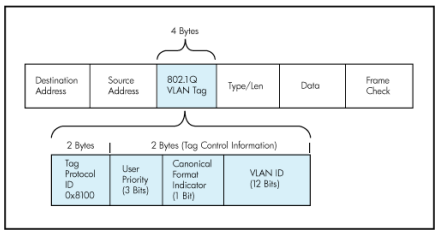
4

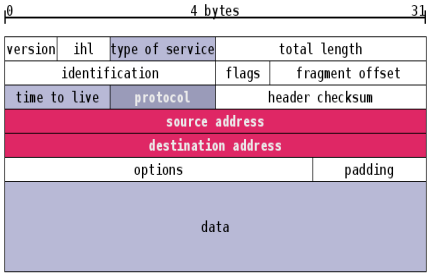
A Plethora of Protocol Acronyms?

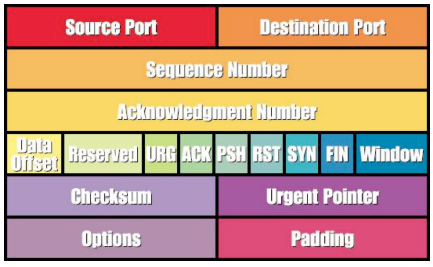
SNMP WAP SIP IPX PPP MAC LLDP FTP UDP OSPF RTP ICMP IMAP IGMP HIP BGP PIM RED HTTP ECN ARP RIP IP MPLS TCP RTCP SMTP RTSP BFD CIDR NNTP TLS NAT STUN SACK DNS SSH DHCP VTP VLAN LISP TFTP LDP POP

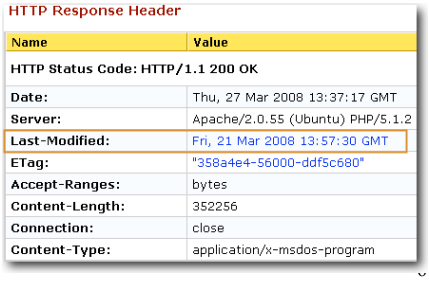
5

A Heap of Header Formats?









TCP/IP Header Formats in Lego

7

A Big Bunch of Boxes?

Router

Label Switched Router

Load balancer

Switch

Scrubber

Repeater

Gateway

Intrusion Detection System

Bridge

Route Reflector

Deep Packet Inspection

DHCP server

Packet shaper

Firewall

NAT

Hub

Packet sniffer

WAN accelerator

DNS server

Base station

Proxy

8

A Ton of Tools?

arpwatch
tcpdump
syslog
wget
nslookup
traceroute
trat
snort
nmap
whois
ipconfig
rancid
ntop
bro
net-snmp
dig
ping
iperf
wireshark
NDT
dummynet
mrtg

9

A Ton of Tools?

arpwatch
tcpdump
syslog
wget
nslookup
traceroute
trat
snort
nmap
whois
ipconfig
rancid
ntop
bro
net-snmp
dig
ping
iperf
wireshark
NDT
dummynet
mrtg

10

The Main Point

1. To learn how the Internet works »
   - What really happens when you "browse the web"?
   - What are TCP/IP, DNS, HTTP, NAT, VPNs, 802.11 etc. anyway?
2. To learn the fundamentals of computer networks

11

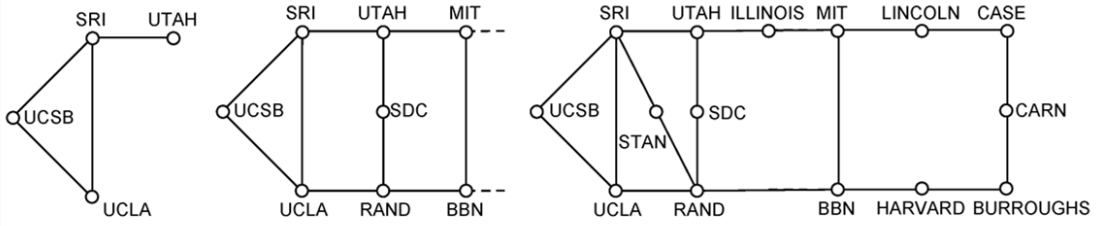
Why learn about the Internet?

1. Curiosity »
2. Impact on our world »

12

From this experimental network …

ARPANET ~1970

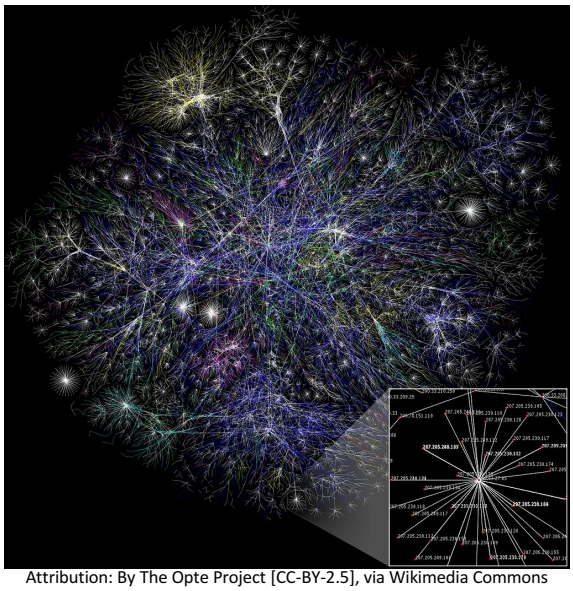


(a) Dec. 1969. (b) July 1970. (c) March 1971.

13

To this!
Internet ~2005

- An everyday institution used at work, home, and on-the-go
- Visualization contains millions of links



Attribution: By The Opte Project [CC-BY-2.5], via Wikimedia Commons

14

Question

- What do you think are the issues that one has to tackle to grow from a small network to an extremely large network?

15

Internet – Societal Impact

- An enabler of societal change
  - Easy access to knowledge
  - Electronic commerce
  - Personal relationships
  - Discussion without censorship

16

Internet – Economic impact

- An engine of economic growth
  - Advertising-sponsored search
  - "Long tail" online stores
  - Online marketplaces
  - Crowdsourcing

17

The Main Point (2)

1. To learn how the Internet works
2. To learn the fundamentals of computer networks
   - What hard problems must they solve?
   - What design strategies have proven valuable?

18

Not a Course Goal

- To learn IT job skills
  - How to configure equipment
    - e.g., Cisco certifications
  - But course material is relevant, and we use hands-on tools

19

Course Mechanics

- Course Administration
  - Everything you need to know will be on the course web page:
    http://www.cs.washington.edu/csep561/
- Teaching Assistants:
  - Yuchen & Ming

Course Logistics

1. Readings
2. Weekly reports: 25%
3. Projects/Homeworks: 75%
   - In groups of two
   - If you want to work individually, then you can just do assignments 1 and 2 (with different deadlines)
   - Details on website

21

Assignment Framework

- Mininet – emulation system from Stanford
- Allows you to create nodes, switches, links all on a single machine
- Assignment 1: TCP dynamics
- Assignment 2: Build a simple router
- Assignment 3: Build a NAT

22

Topics

Introduction

- Intro material, protocol layers, reference models, history
- Optional: The Design Philosophy of the DARPA Internet Protocols

TCP

- Congestion Avoidance and Control
- Optional: Analysis and simulation of a fair queueing algorithm

TCP Congestion control

- Design, implementation and evaluation of congestion control for multipath TCP
- Optional: PCP: Efficient Endpoint Congestion Control

23

Topics

Apr 17	**Routing** • An algorithm for the distributed computation of a spanning tree in an extended LAN • MIT course notes (BGP)
Apr 24	**Datacenters** • VL2: A Scalable and Flexible Data Center Network • Optional: Data Center TCP (DCTCP) • Optional: Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network
May 1	**Software defined networking** • The Road to SDN • Optional: P4: Programming Protocol-Independent Packet Processors

Topics

May 8	**Peer-to-peer systems** • Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications • (Optional) Do incentives build robustness in BitTorrent?
May 15	**DNS/Web** • Demystifying Page Load Performance with WProf
May 22	**Security** • DNS security, BGP security, HTTPS

5

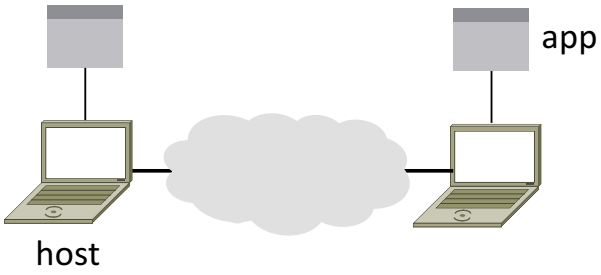
Exercise

- Consider a web object fetch “index.html” from “nytimes.com”
  - What are all the steps involved in performing the fetch?

26

Key Interfaces

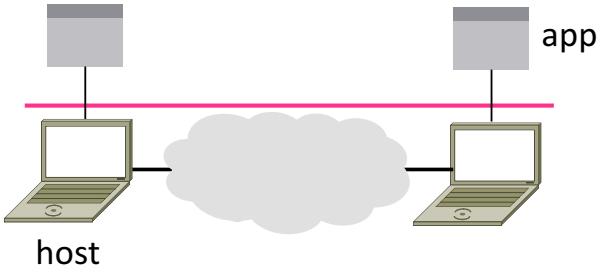
- Between (1) apps and network, and (2) network components
  - More formal treatment later on



27

Key Interfaces (2)

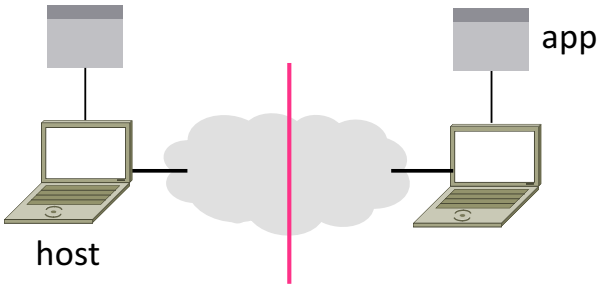
1. Network-application interfaces define how apps use the network
   - Sockets are widely used in practice



28

Key Interfaces (3)

2. Network-network interfaces define how nodes work together
   - Traceroute can peek in the network

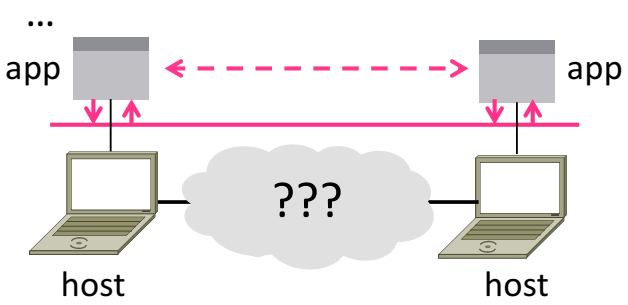


29

Peeking inside the Network with Traceroute

Network Service API Hides Details

- Apps talk to other apps with no real idea of what is inside the network
  - This is good! But you may be curious …



31

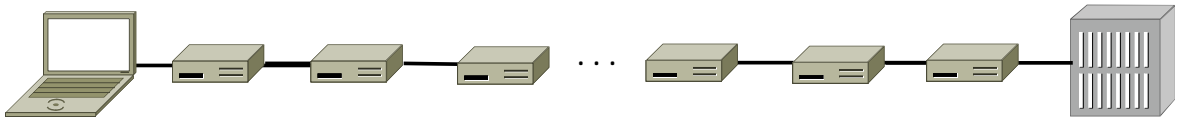
Traceroute

- Widely used command-line tool to let hosts peek inside the network
  - On all OSes (tracert on Windows)
  - Developed by Van Jacobson ~1987
  - Uses a network-network interface (IP) in ways we will explain later

32

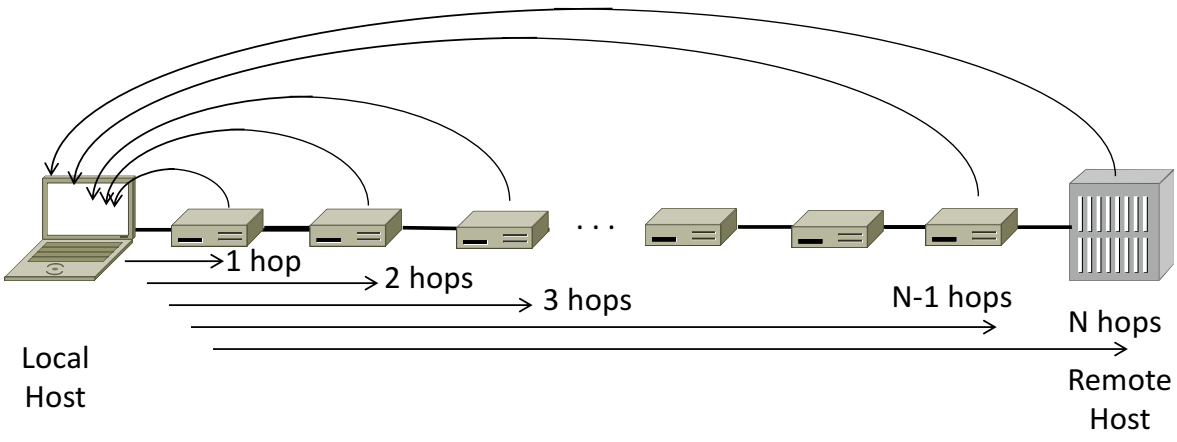
Traceroute (2)

- Probes successive hops to find network path



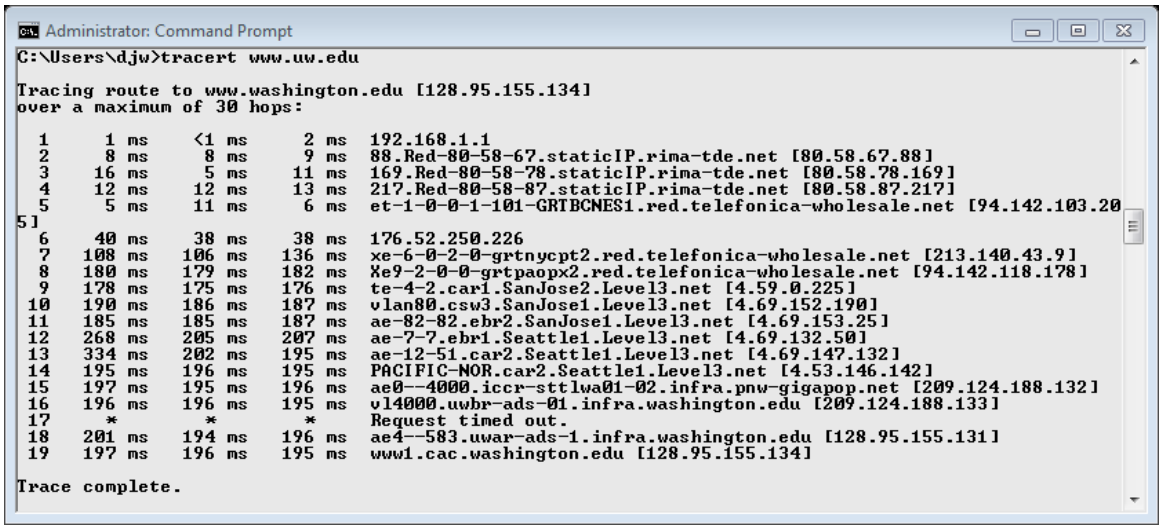
33

Traceroute (3)



34

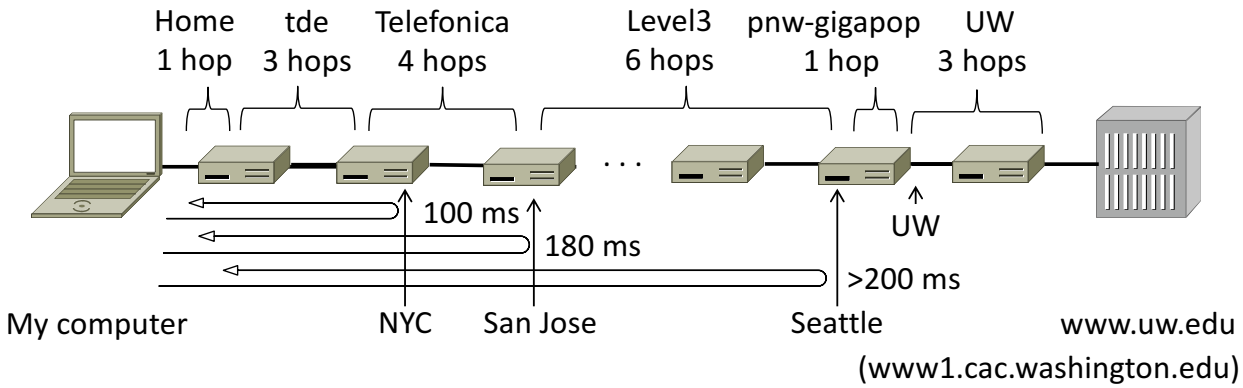
Using Traceroute



```
Administrator: Command Prompt
C:\Users\djw>tracert www.uw.edu

Tracing route to www.washington.edu [128.95.155.134]
over a maximum of 30 hops:

  1     1 ms    <1 ms     2 ms  192.168.1.1
  2     8 ms     8 ms     9 ms  88.Red-80-58-67.staticIP.rima-tde.net [80.58.67.88]
  3    16 ms     5 ms    11 ms  169.Red-80-58-78.staticIP.rima-tde.net [80.58.78.169]
  4    12 ms    12 ms    13 ms  217.Red-80-58-87.staticIP.rima-tde.net [80.58.87.217]
  5     5 ms    11 ms     6 ms  et-1-0-0-1-101-GRTBCNES1.red.telefonica-wholesale.net [94.142.103.20
5]
  6    40 ms    38 ms    38 ms  176.52.250.226
  7   108 ms   106 ms   136 ms  xe-6-0-2-0-grtnycpt2.red.telefonica-wholesale.net [213.140.43.9]
  8   180 ms   179 ms   182 ms  Xe9-2-0-0-grtpaopx2.red.telefonica-wholesale.net [94.142.118.178]
  9   178 ms   175 ms   176 ms  te-4-2.car1.SanJose2.Level3.net [4.59.0.225]
 10   190 ms   186 ms   187 ms  vlan80.csw3.SanJose1.Level3.net [4.69.152.190]
 11   185 ms   185 ms   187 ms  ae-82-82.ebr2.SanJose1.Level3.net [4.69.153.25]
 12   268 ms   205 ms   207 ms  ae-7-7.ebr1.Seattle1.Level3.net [4.69.132.50]
 13   334 ms   202 ms   195 ms  ae-12-51.car2.Seattle1.Level3.net [4.69.147.132]
 14   195 ms   196 ms   195 ms  PACIFIC-NOR.car2.Seattle1.Level3.net [4.53.146.142]
 15   197 ms   195 ms   196 ms  ae0--4000.iccr-sttlwa01-02.infra.pnw-gigapop.net [209.124.188.132]
 16   196 ms   196 ms   195 ms  vl4000.uwbr-ads-01.infra.washington.edu [209.124.188.133]
 17     *        *        *     Request timed out.
 18   201 ms   194 ms   196 ms  ae4--583.uwar-ads-1.infra.washington.edu [128.95.155.131]
 19   197 ms   196 ms   195 ms  www1.cac.washington.edu [128.95.155.134]

Trace complete.
```


Using Traceroute (2)

- ISP names and places are educated guesses



Traceroute to another commercial webserver

```
-bash-3.1$ traceroute www.nyse.com
traceroute to www.nyse.com (209.124.184.150), 30 hops max, 40 byte packets
 1  acar-hsh-01-vlan75.cac.washington.edu (128.208.2.100)  0.327 ms  0.353 ms  0.392 ms
 2  uwcr-hsh-01-vlan3904.cac.washington.edu (205.175.110.17)  0.374 ms  0.412 ms  0.443 ms
 3  uwcr-hsh-01-vlan1901.cac.washington.edu (205.175.103.5)  0.595 ms  0.628 ms  0.659 ms
 4  uwbr-ads-01-vlan1902.cac.washington.edu (205.175.103.10)  0.445 ms  0.472 ms  0.501 ms
 5  ccar1-ads-ge-0-0-0-0.pnw-gigapop.net (209.124.176.32)  0.679 ms  0.747 ms  0.775 ms
 6  a209.124.184.150.deploy.akamaitechnologies.com.184.124.209.in-addr.arpa (209.124.184.150)  0.621 ms  0.456 ms  0.419 ms
```

What is going on?

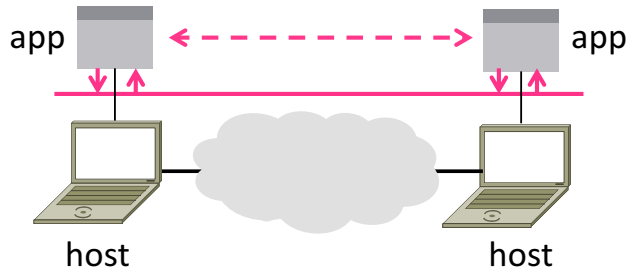
```
-bash-3.1$ nslookup www.nyse.com
Name:   a789.g.akamai.net
Address: 209.124.184.137
```

The Socket API

(§1.3.4, 6.1.2-6.1.4)

Network-Application Interface

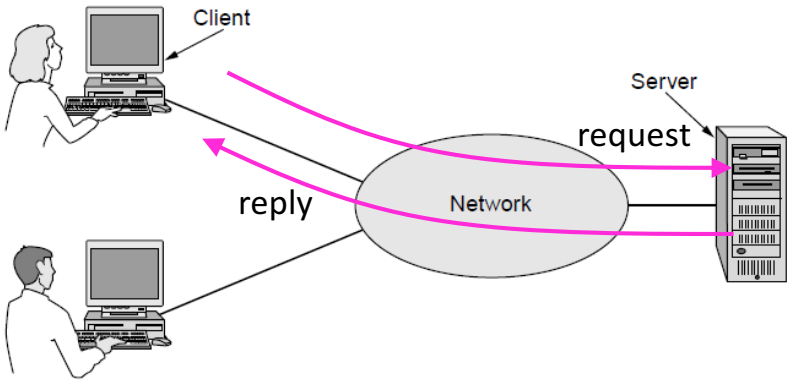
- Defines how apps use the network
  - Lets apps talk to each other via hosts; hides the details of the network



39

Motivating Application

- Simple client-server setup



40

Motivating Application (2)

- Simple client-server setup
  - Client app sends a request to server app
  - Server app returns a (longer) reply
- This is the basis for many apps!
  - File transfer: send name, get file (§6.1.4)
  - Web browsing: send URL, get page
  - Echo: send message, get it back
- Let's see how to write this app …

41

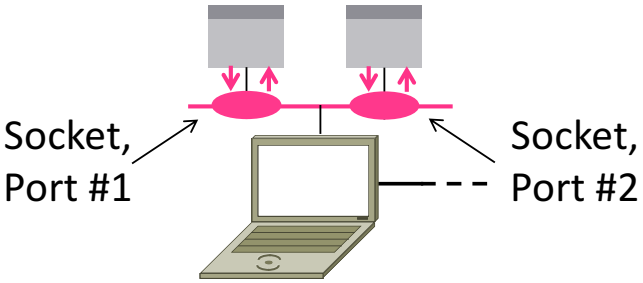
Socket API

- Simple abstraction to use the network
  - The network service API used to write all Internet applications
  - Part of all major OSes and languages; originally Berkeley (Unix) ~1983
- Supports two kinds of network services
  - Streams: reliably send a stream of bytes »
  - Datagrams: unreliably send separate messages. (Ignore for now.)
  - Question: when would you use streams vs. datagrams?

42

Socket API (2)

- Sockets let apps attach to the local network at different ports



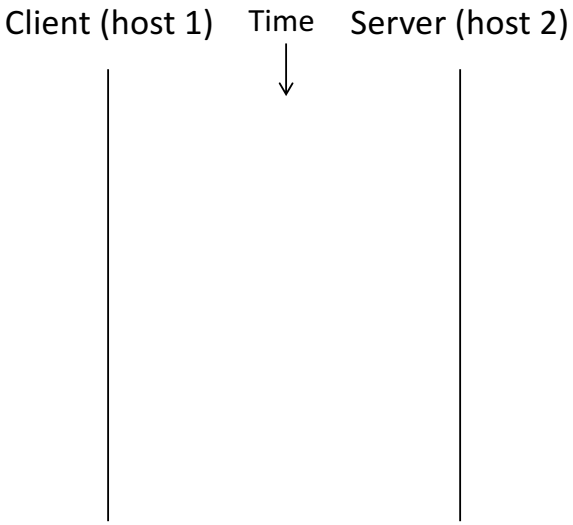
43

Socket API (3)

Primitive	Meaning
SOCKET	Create a new communication endpoint
BIND	Associate a local address with a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Passively establish an incoming connection
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

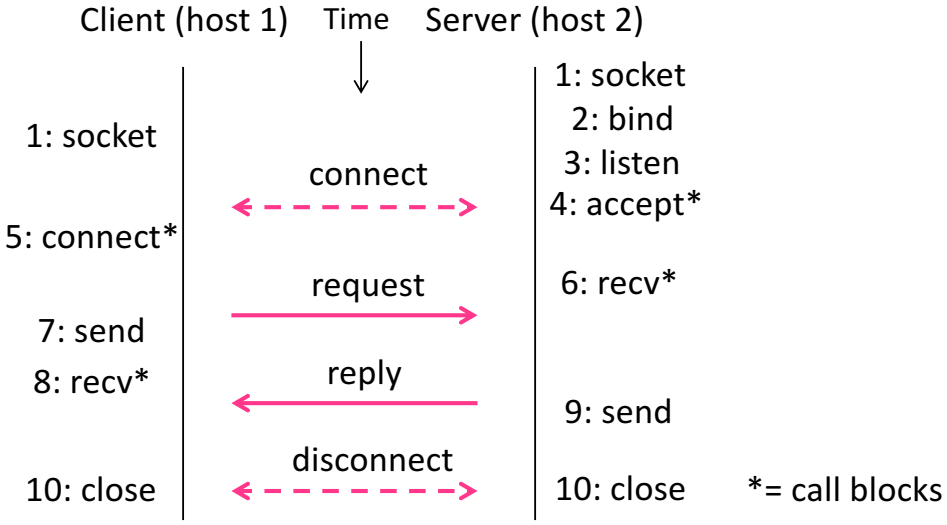
44

Using Sockets



45

Using Sockets (2)



46

Client Program (outline)

```
socket()          // make socket
getaddrinfo()     // server and port name
                  // www.example.com:80
connect()         // connect to server [block]
…
send()            // send request
recv()            // await reply [block]
…                 // do something with data!
close()           // done, disconnect
```

47

Server Program (outline)

```
socket()          // make socket
getaddrinfo()     // for port on this host
bind()            // associate port with socket
listen()          // prepare to accept connections
accept()          // wait for a connection [block]
…
recv()            // wait for request
…
send()            // send the reply
close()           // eventually disconnect
```

48

Protocols and Layering (§1.3)

Networks Need Modularity

- The network does much for apps:
  - Make and break connections
  - Find a path through the network
  - Transfers information reliably
  - Transfers arbitrary length information
  - Send as fast as the network allows
  - Shares bandwidth among users
  - Secures information in transit
  - Lets many new hosts be added
  - …

50

Networks Need Modularity

- The network does much for apps:
  - Make and break connections
  - ork
  - rmation
  - ows
  - s
  - Lets many new hosts be added
  - …

We need a form of modularity, to help manage complexity and support reuse

51

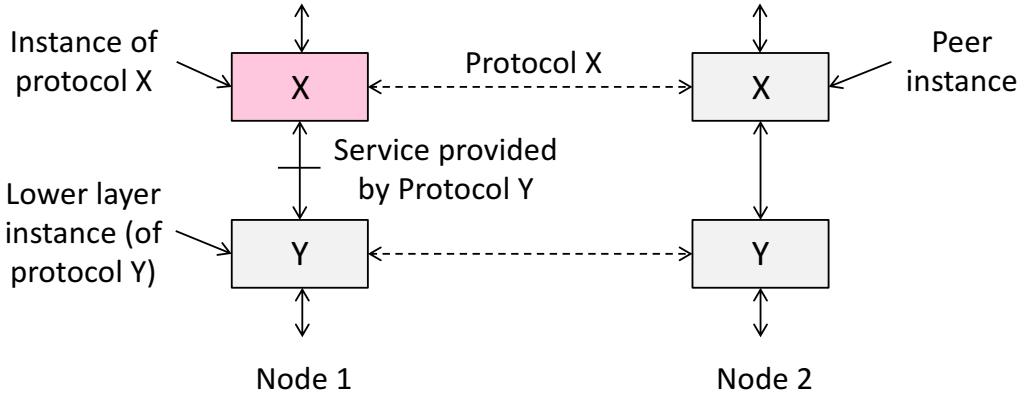
Protocols and Layers

- Protocols and layering is the main structuring method used to divide up network functionality
  - Each instance of a protocol talks virtually to its peer using the protocol
  - Each instance of a protocol uses only the services of the lower layer

52

Protocols and Layers (2)

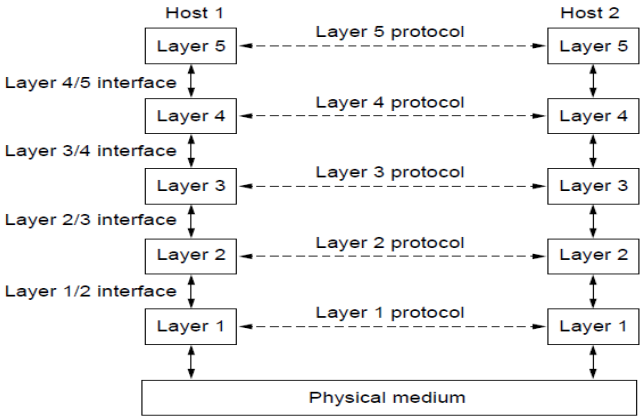
- Protocols are horizontal, layers are vertical



Question: what is an example of a protocol and what abstraction does it provide/rely on? 53

Protocols and Layers (3)

- Set of protocols in use is called a protocol stack



54

Protocols and Layers (4)

- Protocols you've probably heard of:
  - TCP, IP, 802.11, Ethernet, HTTP, SSL, DNS, … and many more
- An example protocol stack
  - Used by a web browser on a host that is wirelessly connected to the Internet

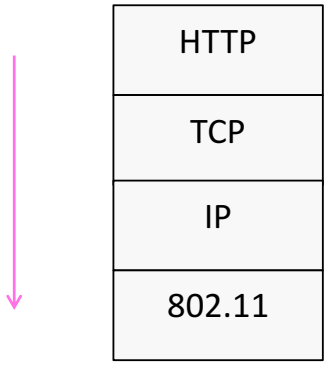
55

Encapsulation

- Encapsulation is the mechanism used to effect protocol layering
  - Lower layer wraps higher layer content, adding its own information to make a new message for delivery
  - Like sending a letter in an envelope; postal service doesn't look inside

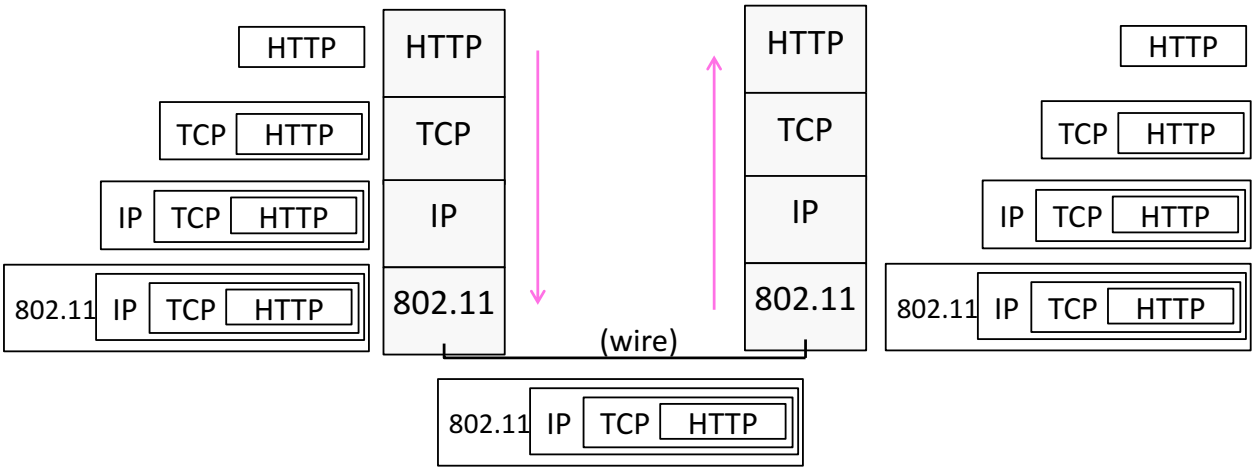
56

Encapsulation (2)



57

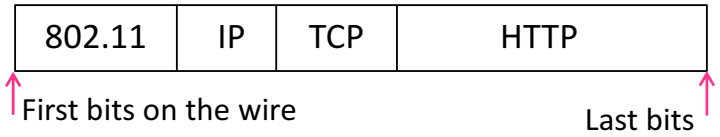
Encapsulation (3)



58

Encapsulation (4)

- Normally draw message like this:
  - Each layer adds its own header

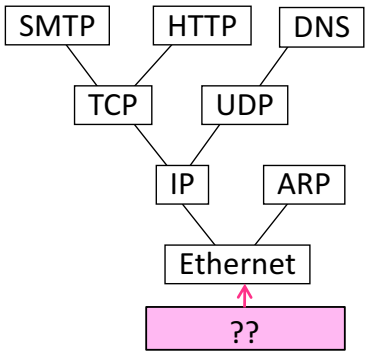


- More involved in practice
  - Trailers as well as headers, encrypt/compress contents
  - Segmentation (divide long message) and reassembly

59

Demultiplexing

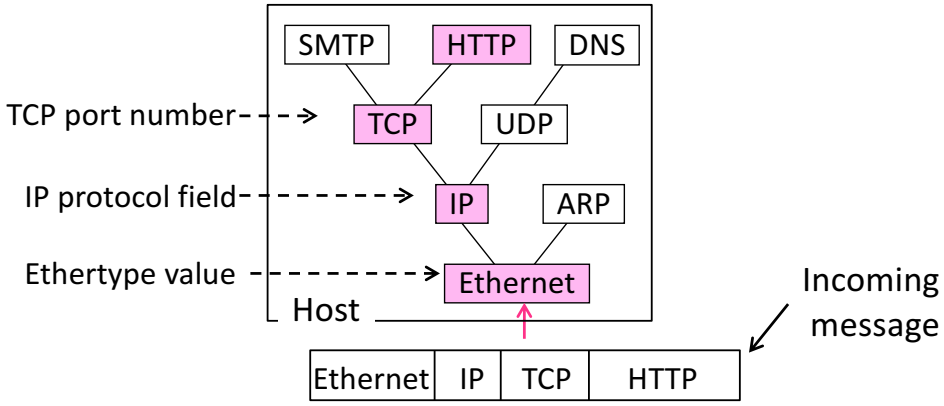
- Incoming message must be passed to the protocols that it uses



60

Demultiplexing (2)

- Done with demultiplexing keys in the headers



61

Protocols

- What are the implications of protocols and layering?
  - Are there any downsides to protocols/layering?
  - Are there any performance costs?

62

Advantage of Layering

- Using information hiding to connect different systems

63

Guidance

- What functionality should we implement at which layer?
  - This is a key design question
  - Reference models provide frameworks that guide us »

64

OSI "7 layer" Reference Model

- A principled, international standard, to connect systems
  - Influential, but not used in practice. (Woops)

7	Application	– Provides functions needed by users
6	Presentation	– Converts different representations
5	Session	– Manages task dialogs
4	Transport	– Provides end-to-end delivery
3	Network	– Sends packets over multiple links
2	Data link	– Sends frames of information
1	Physical	– Sends bits as signals

65

Internet Reference Model

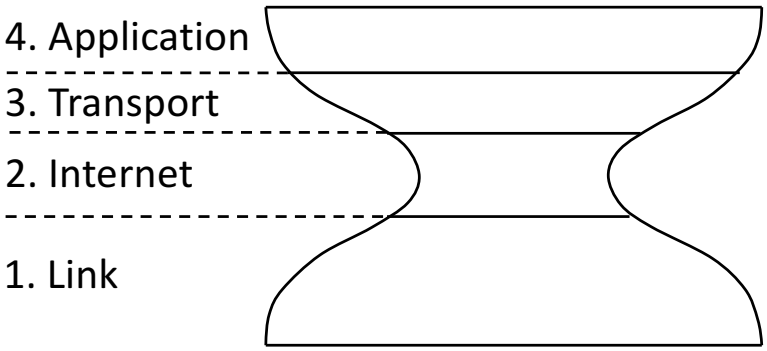
- A four layer model based on experience; omits some OSI layers and uses the IP as the network layer.

4.	Application	– Programs that use network service
3.	Transport	– Provides end-to-end data delivery
2.	Internet	– Send packets over multiple networks
1.	Link	– Send frames over a link

66

Internet Reference Model (2)

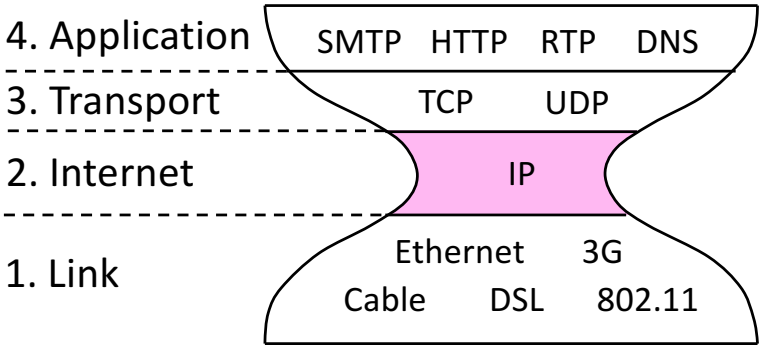
- What are examples of common protocols in each layer?



67

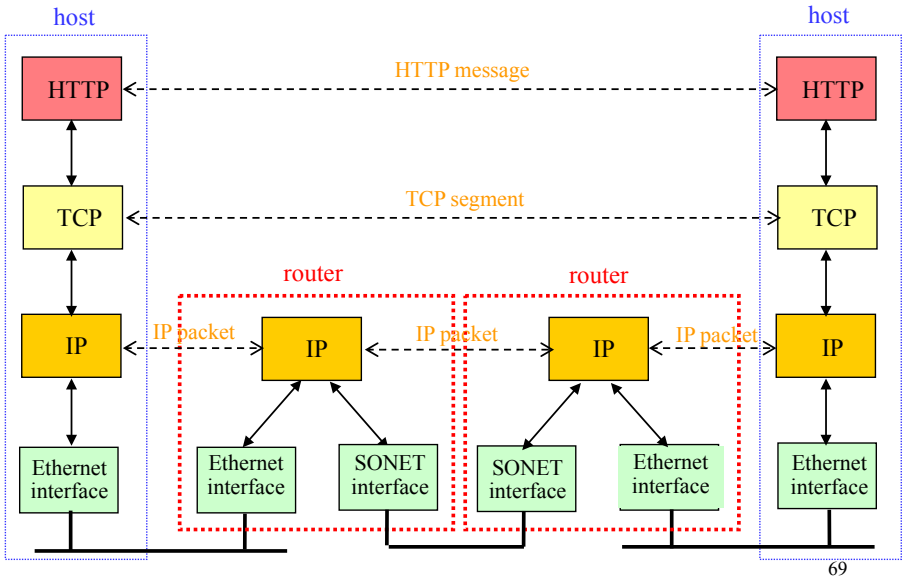
Internet Reference Model (3)

- IP is the "narrow waist" of the Internet
  - Supports many different links below and apps above



68

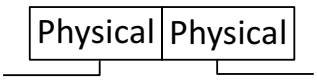
IP Suite: End Hosts vs. Routers



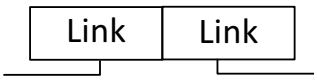
Layer-based Names (2)

- For devices in the network:

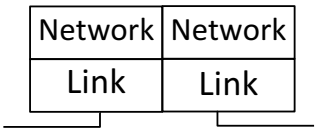
Repeater



Switch (or bridge)



Router

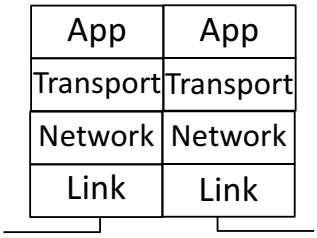


70

Layer-based Names (3)

- For devices in the network:

Proxy or middlebox or gateway



But they all look like this!

71

A Note About Layers

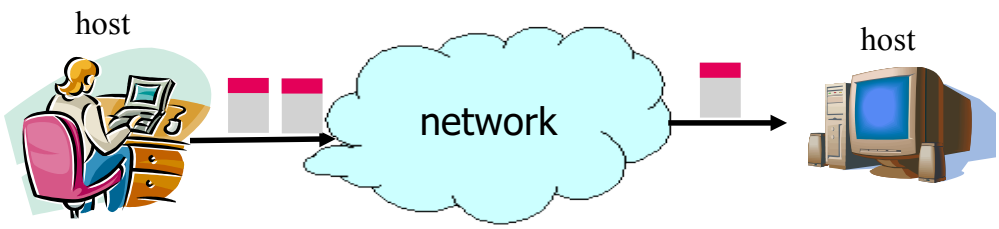
- They are guidelines, not strict
  - May have multiple protocols working together in one layer
  - May be difficult to assign a specific protocol to a layer

72

Best-Effort Packet-Delivery Service

Host-Network Division of Labor

- Packet switching
  - Divide messages into a sequence of packets
  - Headers with source and destination address
- Best-effort delivery
  - Packets may be lost
  - Packets may be corrupted
  - Packets may be delivered out of order



Host-Network Interface: Why Packets?

- Data traffic is bursty
  - Logging in to remote machines
  - Exchanging e-mail messages
- Don't want to waste bandwidth
  - No traffic exchanged during idle periods
- Better to allow multiplexing
  - Different transfers share access to same links
- Packets can be delivered by most anything
  - RFC 1149: IP Datagrams over Avian Carriers

Host-Network Interface: Why Best-Effort?

- Never having to say you're sorry…
  - Don't reserve bandwidth and memory
  - Don't do error detection & correction
  - Don't remember from one packet to next
- Easier to survive failures
  - Transient disruptions are okay during failover
- Can run on nearly any link technology
  - Greater interoperability and evolution

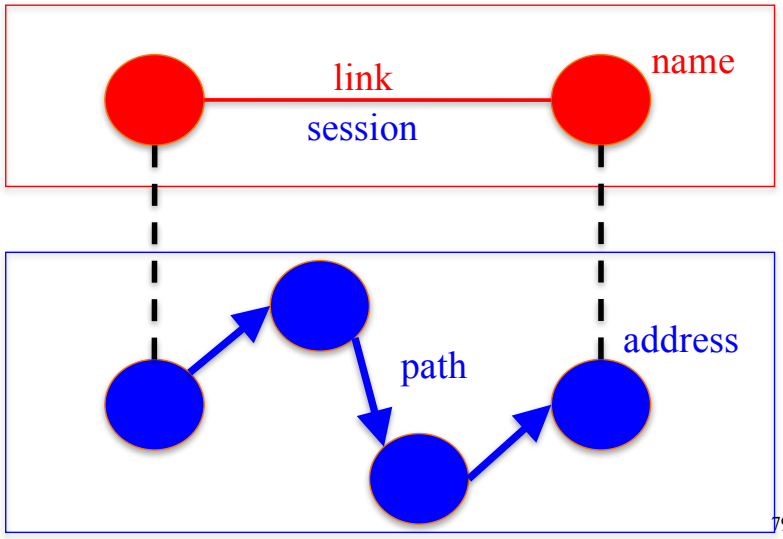
Intermediate Transport Layer

- But, *applications* want efficient, accurate transfer of data in order, in a timely fashion
  - Let the end hosts handle all of that
  - (An example of the "end-to-end argument")
- Transport layer can optionally…
  - Detect and retransmit lost packets
  - Put out-of-order packets back in order
  - Detect and handle corrupted packets
  - Avoid overloading the receiver
  - <insert your requirement here>

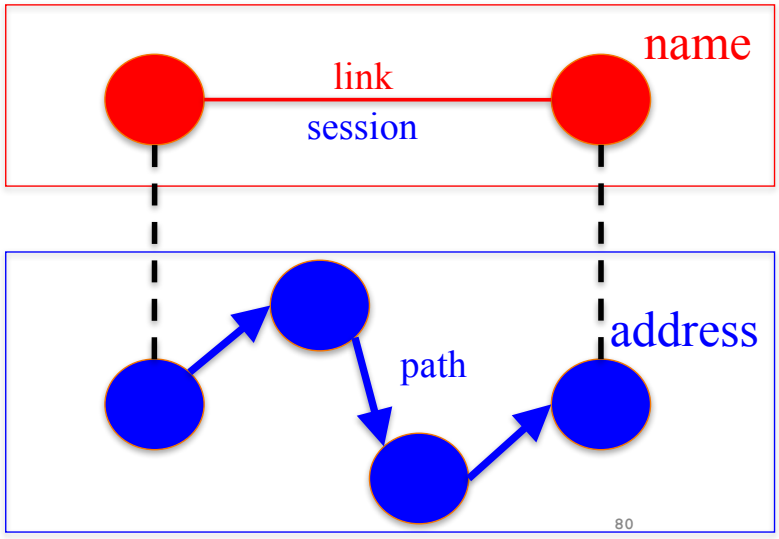
Directories and Routing

78

Relationship Between Layers



Directories: Mapping Name to Address

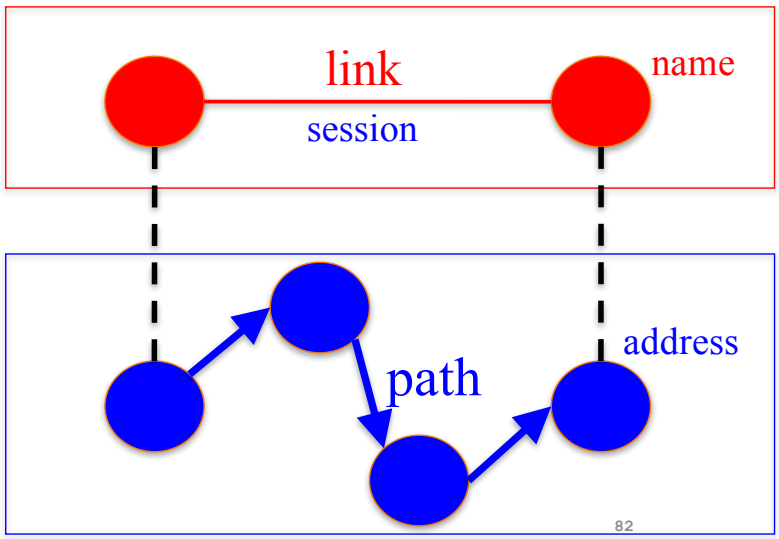


Types of Directories

- Simplistic designs
  - Ask everyone (e.g., flooding in ARP)
  - Tell everyone (e.g., pushing /etc/hosts)
  - Central directory
- Scalable distributed designs
  - Hierarchical namespace (e.g., DNS)
  - Flat name space (e.g., Distributed Hash Table)

81

Routing: Mapping Link to Path



Path Computation

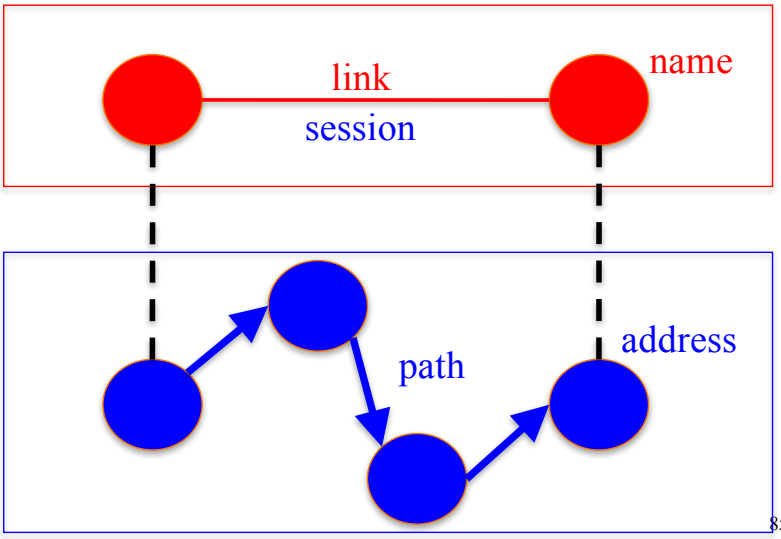
- Spanning tree (e.g., Ethernet)
  - One tree that connects every pair of nodes
- Shortest paths (e.g., OSPF, IS-IS, RIP)
  - Shortest-path tree rooted at each node
- Locally optimal paths (e.g., BGP)
  - Each node selects the best among its neighbors
- End-to-end paths (e.g., source routing)
  - Each node picks the best end-to-end path

83

Network Discovery and Bootstrapping

84

Relationship Between Layers



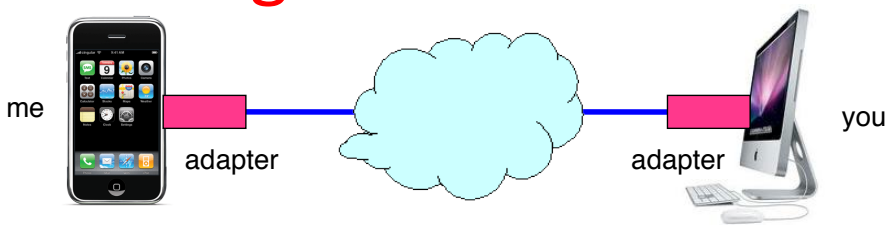
85

Three Kinds of Identifiers

	Host Name	IP Address	MAC Address
Example	www.cs.princeton.edu	128.112.7.156	00-15-C5-49-04-A9
Size	Hierarchical, human readable, variable length	Hierarchical, machine readable, 32 bits (in IPv4)	Flat, machine readable, 48 bits
Read by	Humans, hosts	IP routers	Switches in LAN
Allocation, top-level	Domain, assigned by registrar (e.g., for .edu)	Variable-length prefixes, assigned by ICANN, RIR, or ISP	Fixed-sized blocks, assigned by IEEE to vendors (e.g., Dell)
Allocation, low-level	Host name, local administrator	Interface, by DHCP or an administrator	Interface, by vendor

86

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

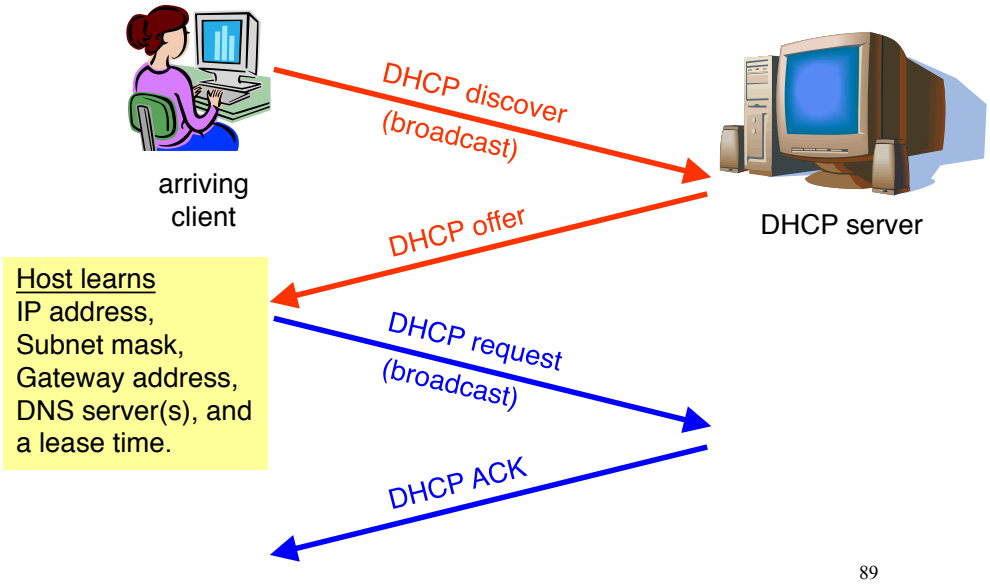
87

Mapping Between Identifiers

- Dynamic Host Configuration Protocol (DHCP)
  - Given a MAC address, assign a unique IP address
  - … and tell host other stuff about the Local Area Network
  - To automate the boot-strapping process
- Address Resolution Protocol (ARP)
  - Given an IP address, provide the MAC address
  - To enable communication within the Local Area Network
- Domain Name System (DNS)
  - Given a host name, provide the IP address
  - Given an IP address, provide the host name

88

Dynamic Host Configuration Protocol



89

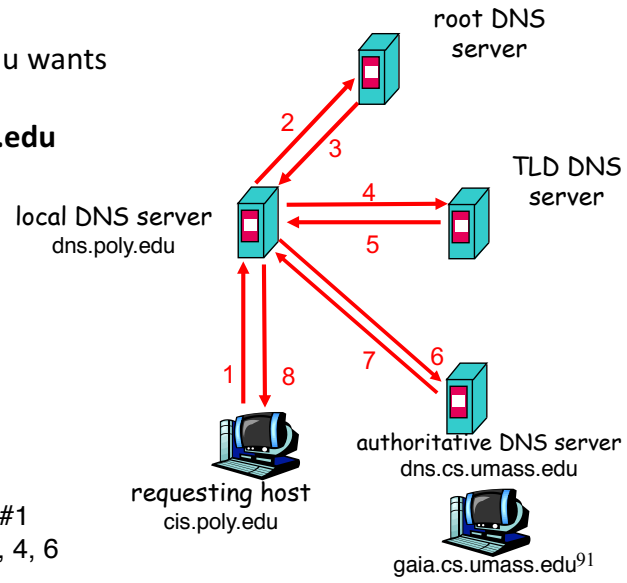
Address Resolution Protocol (ARP)

- 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

90

Domain Name System

Host at cis.poly.edu wants IP address for **gaia.cs.umass.edu**



Recursive query: #1
Iterative queries: #2, 4, 6

Questions

- Should addresses correspond to the interface (point of attachment) or to the host?
- Why do we have all three identifiers? Do we need all three?
- What should be done to prevent spoofing of addresses?