CS514: Intermediate Course in Computer Systems

Lecture 3: Sept. 8, 2003

"Introduction to the Network"

Overview of Lecture

CS514

Introduction to the network layer

- Classic view of network layer
  - OSI stack
- Classic view no longer (never was?) accurate
- End-to-end argument
- Internet components (hosts, routers, links, etc.)
- Protocol layering fundamentals
- IP, UDP, TCP, pros and cons, SCTP
- Ethereal---nice protocol monitoring and debugging tool
- Naming: Taxonomy, DNS, URIs

Who recognizes this?

CS514

```
int sockfd;
struct sockaddr_in addr;

addr.sin_family = AF_INET;
addr.sin_addr.s_addr =
          inet_addr(SERV_HOST_ADDR);
addr.sin_port = htons(SERV_TCP_PORT);

sockfd = socket(AF_INET, SOCK_STREAM, 0);
connect(sockfd, (struct sockaddr *) &addr,
          sizeof(serv_addr));
do_stuff(stdin, sockfd);
```

Classic view of network API

CS514

- Start with host name (maybe)

foo.bar.com

Classic view of network API

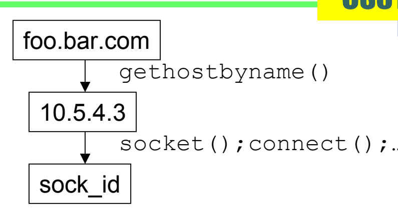
CS514

- Start with host name
- Get an IP address

Classic view of network API

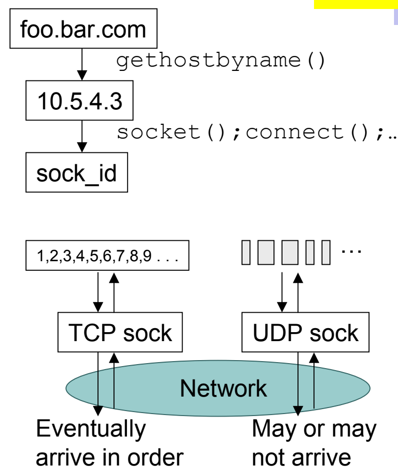
CS514

- Start with host name
- Get an IP address
- Make a socket (protocol, address)



Classic view of network API

- Start with host name
- Get an IP address
- Make a socket (protocol, address)
- Send byte stream (TCP) or packets (UDP)

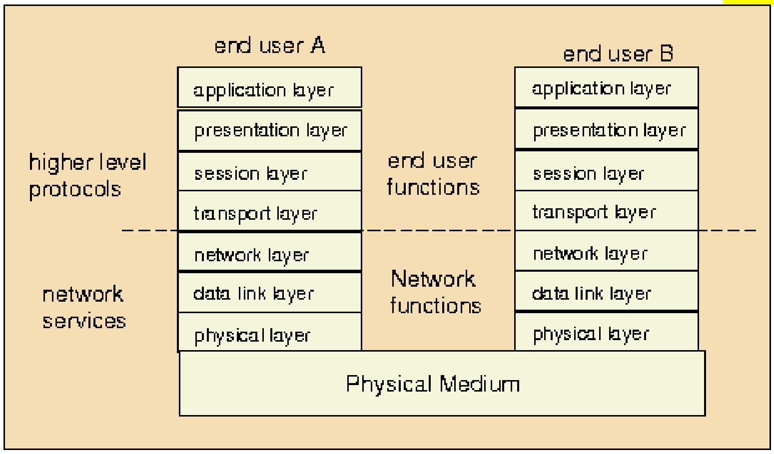


Classic approach “broken” in many ways

- IP address different depending on who asks for it
- Address may be changed in the network
- IP address may not be reachable (even though destination is up and attached)
  - Or may be reachable by you but not another host
- IP address may change in a few minutes or hours
- Packets may not come from who you think (network caches)

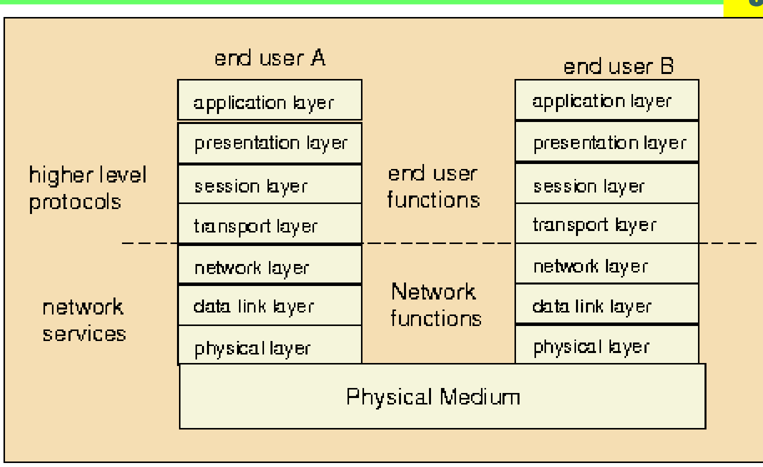
Classic OSI stack

CS514



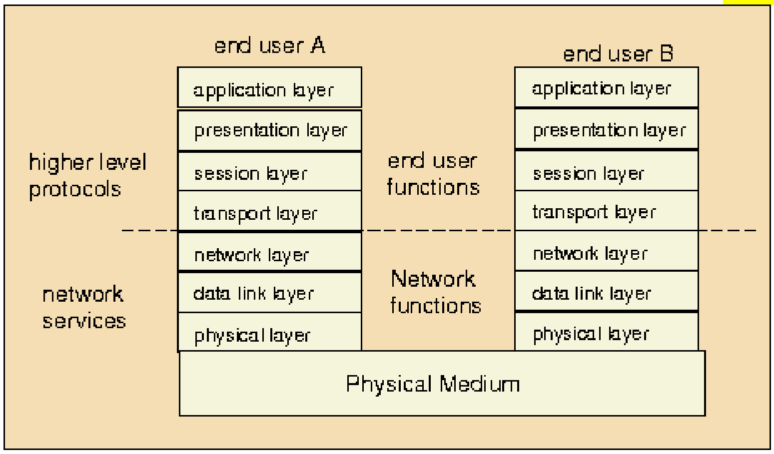
Useful abstraction or out-dated and misleading?

CS514



Ethernet? Bridged Ethernet? XML? HTTP?

CS514



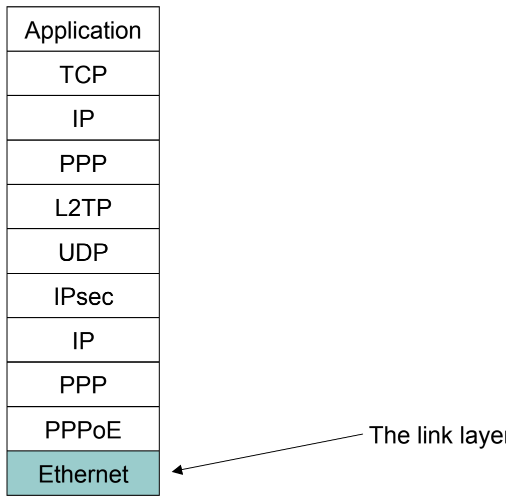
Example Microsoft VPN stack (PPTP)

CS514

Application
TCP
IP
PPP
L2TP
UDP
IPsec
IP
PPP
PPPoE
Ethernet

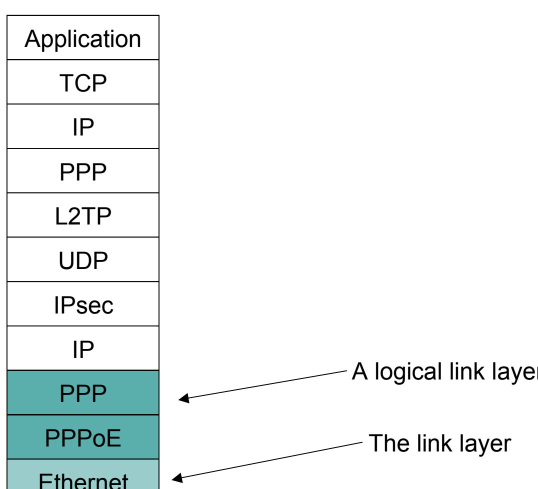
Example Microsoft VPN stack

CS514



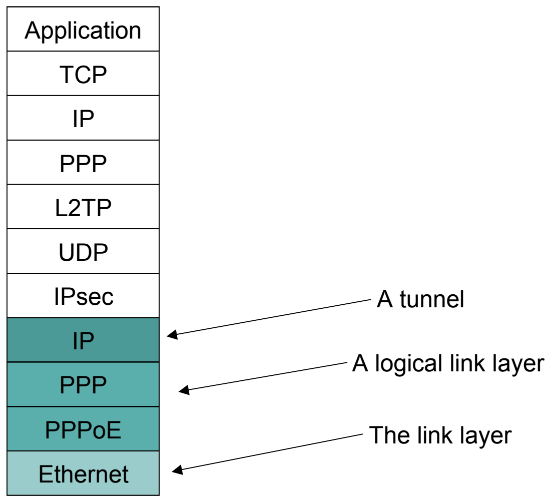
Example Microsoft VPN stack

CS514



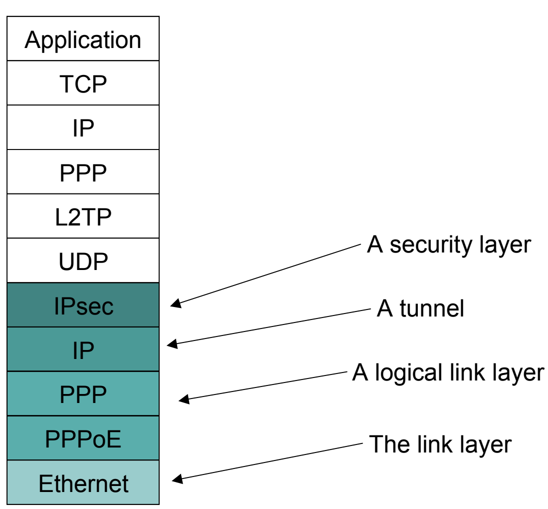
Example Microsoft VPN stack

CS514



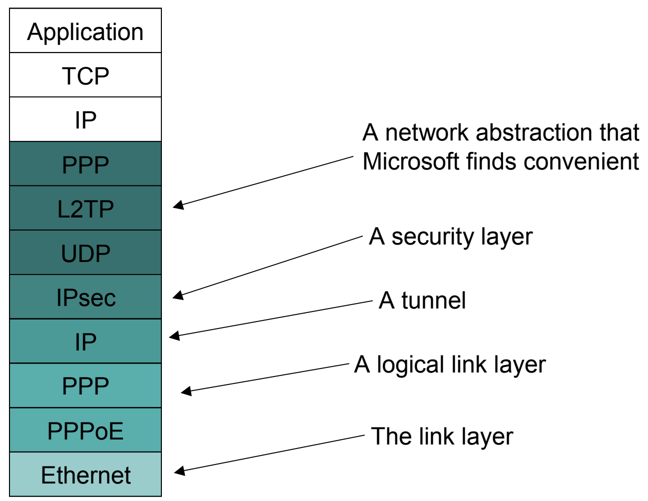
Example Microsoft VPN stack

CS514



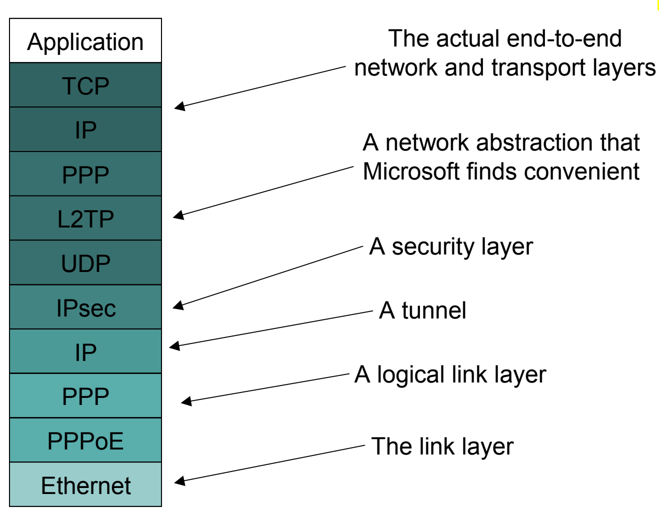
Example Microsoft VPN stack

CS514



Example Microsoft VPN stack

CS514



Example Microsoft VPN stack

CS514

Application
TCP
IP
PPP
L2TP
UDP
IPsec
IP
PPP
PPPoE
Ethernet

TCP: Transport Control Protocol
IP: Internet Protocol
PPP: Point-to-Point Protocol
L2TP: Layer 2 Tunneling Protocol
UDP: User Datagram Protocol
IPsec: Secure IP
PPPoE: PPP over Ethernet

What can we learn from this?

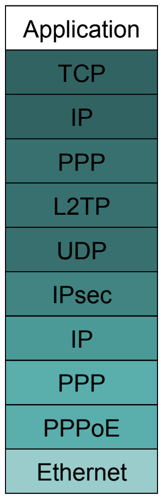
CS514

Application
TCP
IP
PPP
L2TP
UDP
IPsec
IP
PPP
PPPoE
Ethernet

- That the internet is a mature technology
  - Kludges on kludges

What can we learn from this?

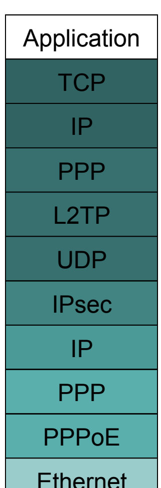
CS514



- That the internet is a mature technology
  - Kludges on kludges
- Having the biggest company isn't good enough for Bill

What can we learn from this?

CS514



- That the internet is a mature technology
  - Kludges on kludges
- That having the biggest company isn't good enough for Bill

That the end-to-end argument actually works!

What is the end-to-end argument?

CS514

In a nutshell:

If you want something done right, you gotta do it yourself

"End-To-End Arguments In System Design", Saltzer, Reed, Clark, ACM Transactions on Computer Systems, 1984

End-to-end argument is mostly about reliability

CS514

- Early 80's: industry assumed that the network should do everything
  - Guaranteed delivery, sequencing, duplicate suppression
  - If the network does it, the end system doesn't have to
  - X.25, for example

The network doesn't always work right

CS514

- Applications had to check to see if the network really did its job…
  - … and repair the problem if the network didn't do its job
- End-to-end insight:

  If the application has to do it anyway, why do it in the network at all?
- Keep the network simple

So when should the network do more?

CS514

- When you get performance gains
  - Link-level retransmissions over a lossy link are faster than E2E retransmissions
- Also
  - When the network doesn't trust the end user
    - Corporation or military encrypt a link because the end user might not do it
  - Some things just can't be done at the end
    - Routing algorithms
    - Billing
    - User authentication

Fate sharing: a stronger statement of end-to-end

CS514

- If the network has no state, network failures won't screw you
- Keep the state in the same box as the application
  - The fate of the communications is shared with the fate of the application

God, Motherhood, Apple Pie, and the E2E *Principle*

CS514

Brief Rant

- E2E followed with religious fervor in IETF
- Often applied to addressing, which has nothing to do with the original E2E argument
  - Reaction to NAT was to fix the network (IPv6), *actively discourage* "fixing" the host
  - Laudable goal, but in a way opposite of E2E "spirit"
- Sometimes performance hurt in deference to E2E
  - Compression of Voice over IP (RTP, Real Time Protocol)
  - Mobile IP

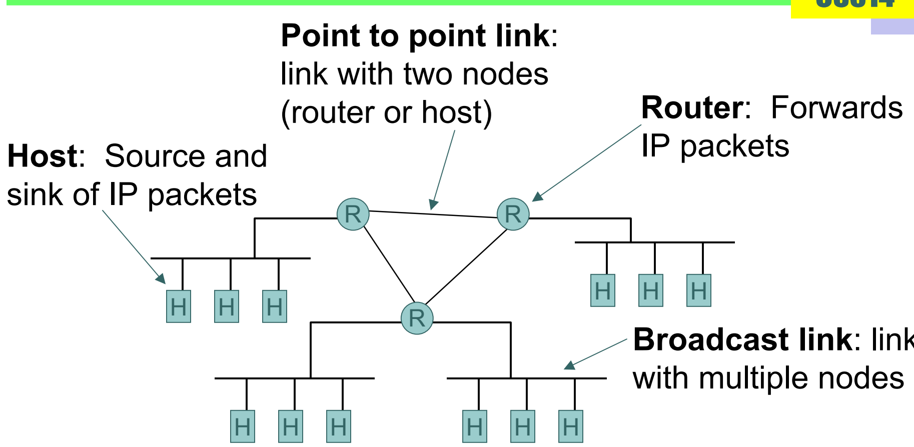
E2E vs. fault tolerance vs. high availability

CS514

- E2E says minimize the number of boxes with state
  - Two endpoints = two boxes with state
- Fault tolerance says maximize the number of boxes with (the same) state
  - Five boxes, four can crash
- High available requires performance, which means fewer stateful boxes
  - While still achieving fault tolerance . . .

Network components

CS514



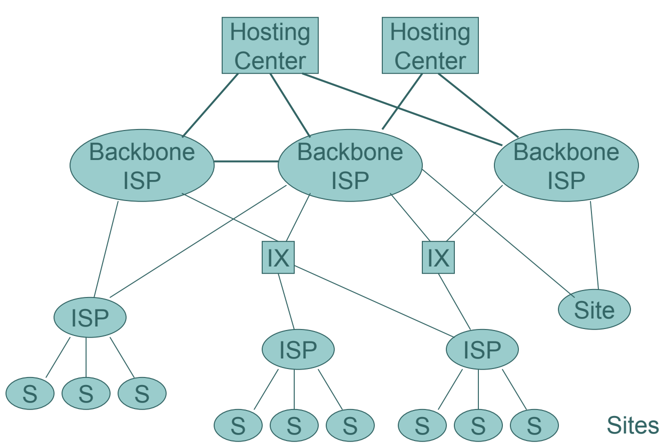
Network components

CS514

- **Network:** Collection of hosts, links, and routers
- **Site**: Stub network, typically in one location and under control of one administration
- **Firewall/NAT**: Box between the site and ISP that provides filtering, security, and Network Address Translation
- **ISP**: Internet Service Provider. Transit network that provides IP connectivity for sites
- **Backbone ISP**: Transit network for regional ISPs and large sites
- **Inter-exchange (peering point)**: Broadcast link where multiple ISPs connect and exchange routing information (peering)
- **Hosting center**: Stub network that supports lots of hosts (web services), typically with high speed connections to many backbone ISPs.
- **Bilateral peering**: Direct connection between two backbone ISPs

Internet topology

CS514



IXs came first

IXs tend to be performance bottlenecks

Hosting centers and bilateral peering are a response to poor IXs

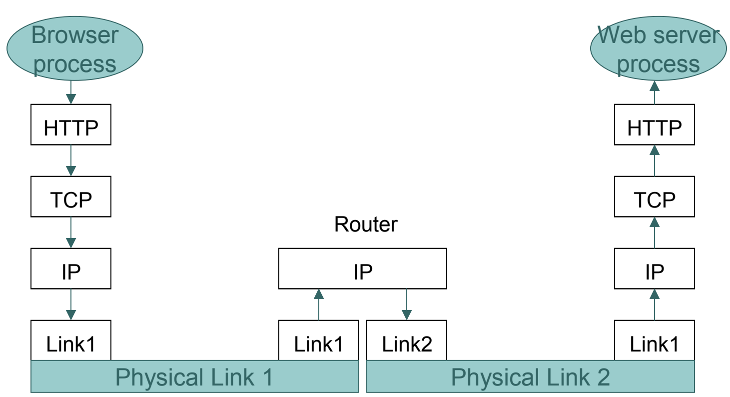
Protocol layering

CS514

- Communications stack consists of a set of *services*, each providing a service to the *layer* above, and using services of the layer below
  - Each service has a programming API, just like any software module
- Each service has to convey information one or more *peers* across the network
- This information is contained in a *header*
  - The headers are transmitted in the same order as the layered services

Protocol layering example

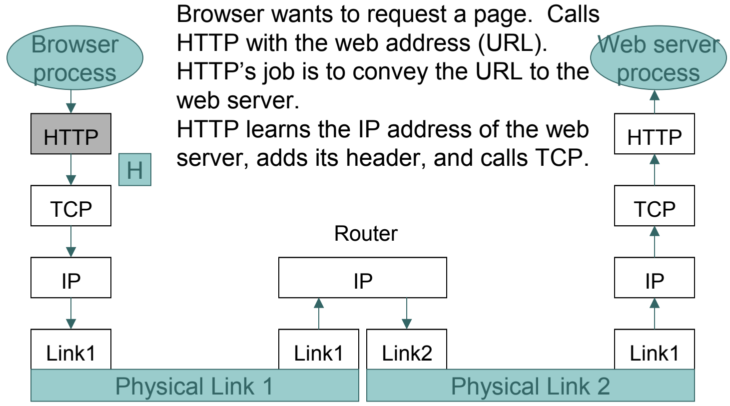
CS514



Protocol layering example

CS514

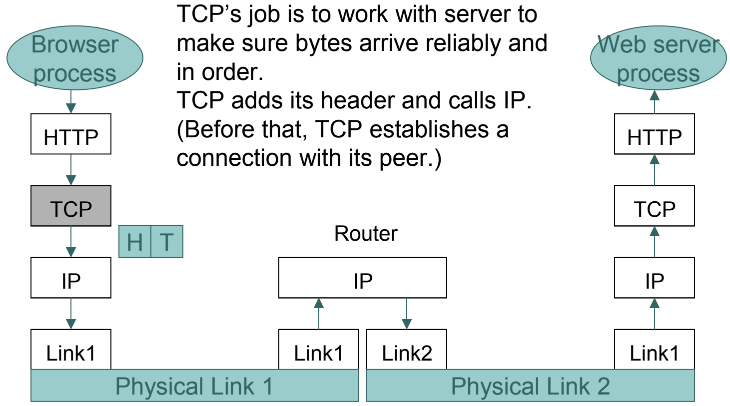
Browser wants to request a page. Calls HTTP with the web address (URL). HTTP's job is to convey the URL to the web server.
HTTP learns the IP address of the web server, adds its header, and calls TCP.



Protocol layering example

CS514

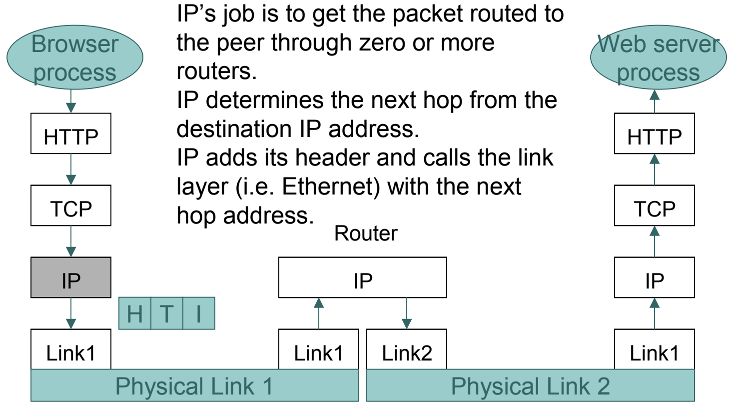
TCP's job is to work with server to make sure bytes arrive reliably and in order.
TCP adds its header and calls IP.
(Before that, TCP establishes a connection with its peer.)



Protocol layering example

CS514

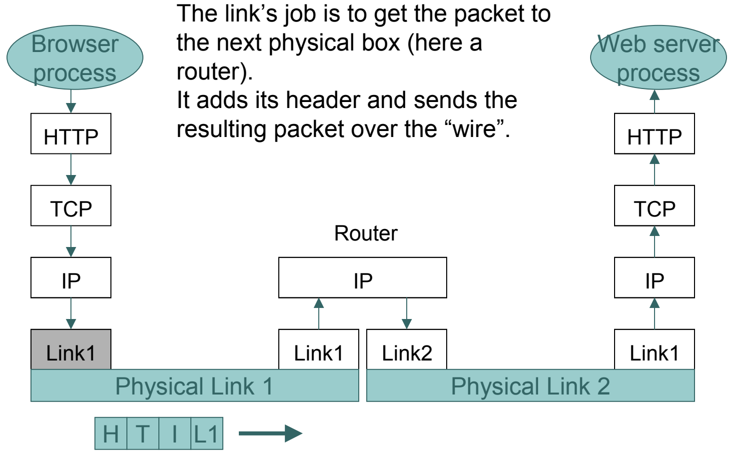
IP's job is to get the packet routed to the peer through zero or more routers.
IP determines the next hop from the destination IP address.
IP adds its header and calls the link layer (i.e. Ethernet) with the next hop address.



Protocol layering example

CS514

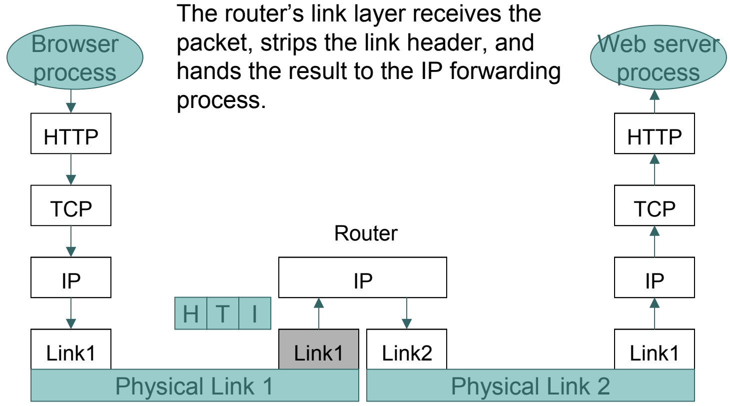
The link's job is to get the packet to the next physical box (here a router).
It adds its header and sends the resulting packet over the "wire".



Protocol layering example

CS514

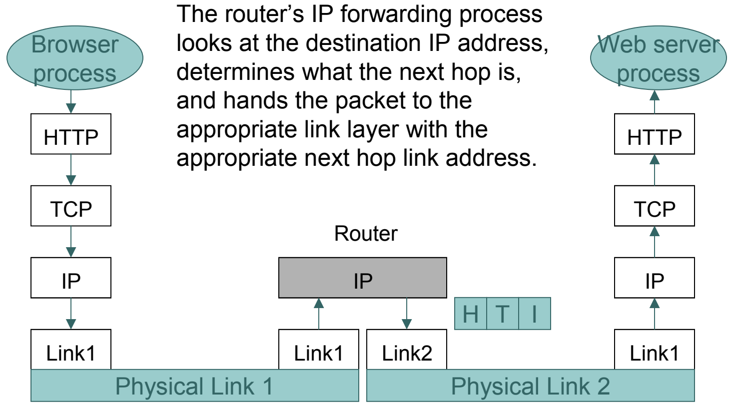
The router's link layer receives the packet, strips the link header, and hands the result to the IP forwarding process.



Protocol layering example

CS514

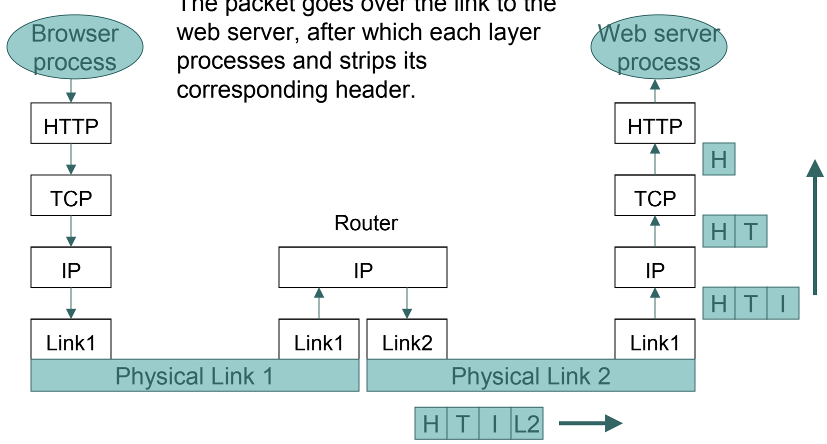
The router's IP forwarding process looks at the destination IP address, determines what the next hop is, and hands the packet to the appropriate link layer with the appropriate next hop link address.



Protocol layering example

CS514

The packet goes over the link to the web server, after which each layer processes and strips its corresponding header.



Basic elements of any protocol header

CS514

- *Demuxing* field
  - Indicates which is the next higher layer (or process, or context, etc.)
- *Length* field or header *delimiter*
  - For the header, optionally for the whole packet
- Header format may be *text* (HTTP, SMTP (email)) or *binary* (IP, TCP, Ethernet)

Demuxing field examples

CS514

- Ethernet: Protocol Number
  - Indicates IPv4, IPv6, (old: Appletalk, SNA, Decnet, etc.)
- IP: Protocol Number
  - Indicates TCP, UDP, SCTP
- TCP and UDP: Port Number
  - Well known ports indicate FTP, SMTP, HTTP, SIP, many other applications
  - Dynamically negotiated ports indicate specific processes (for these and other protocols)
- HTTP: Host field
  - Indicates “virtual web server” within a physical web server
  - (Well, more like an identifier than a demuxing field)

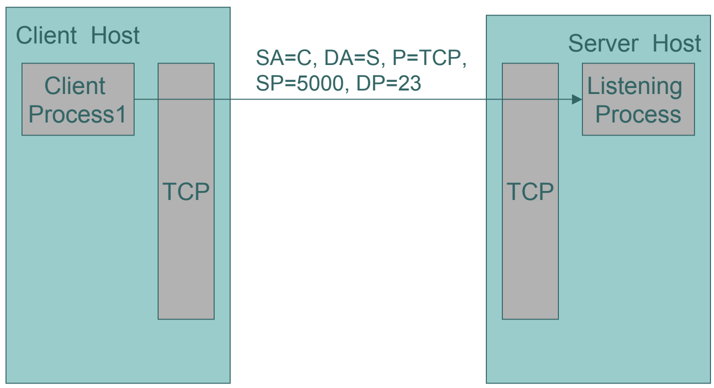
UDP/TCP application process selection

CS514

- Unicast application process is selected by the complete 5-tuple, consisting of:
  - *Source and Dest IP address*
  - *Source and Dest port*
  - *IP protocol*
  - Ex: an FTP server may have concurrent transfers to the same client. Only the source port will differ.
- *Multicast* application process is selected by a 3-tuple: *Dest IP address and UDP port, and IP protocol*
  - Because it is multicast, UDP may select multiple processes

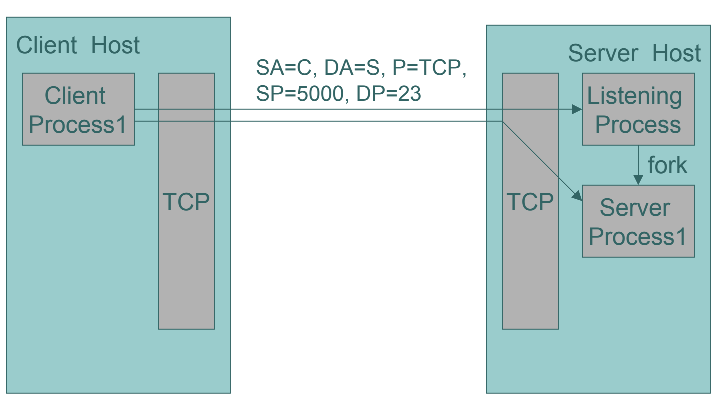
Typical server incoming connection processing

CS514



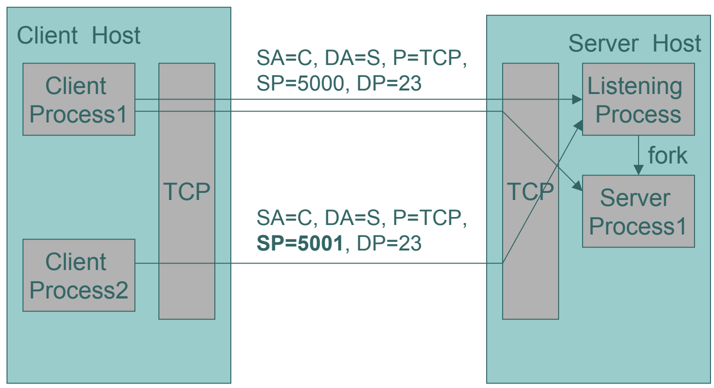
Typical server incoming connection processing

CS514



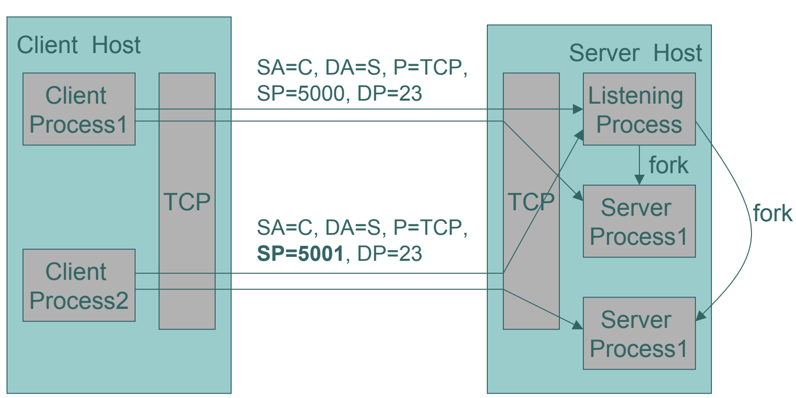
Typical server incoming connection processing

CS514



Typical server incoming connection processing

CS514



IP (Internet Protocol)

CS514

- Three services:
  - *Unicast*: transmits a packet to a specific host
  - *Multicast*: transmits a packet to a group of hosts
  - *Anycast*: transmits a packet to one of a group of hosts (typically nearest)
- Destination and source identified by the IP address (32 bits for IPv4, 128 bits for IPv6)
- All services are unreliable
  - Packet may be dropped, duplicated, and received in a different order

IP address

CS514

- The raison d'être for the IP packet
- Both source and destination address may be modified in transit
  - By NAT boxes
  - But even so, sending a packet back to the source IP address will get the packet there
  - Unless source address is spoofed, which can easily be done
- IP (unicast) address is hierarchical, but host can treat it as a flat identifier
  - (almost…needs to know network mask)
  - Can't tell how close or far a host is by looking at its IP address

IP(v4) address format

CS514

- In binary, a 32-bit integer
- In text, this: "128.52.7.243"
  - Each decimal digit represents 8 bits (0 – 255)
- "Private" addresses are not globally unique:
  - Used behind NAT boxes
  - 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
- Multicast addresses start with 1110 as the first 4 bits (Class D address)
  - 224.0.0.0/4
- Unicast and anycast addresses come from the same space

UDP (User Datagram Protocol)

CS514

- Runs above IP
- Same unreliable service as IP
  - Packets can get lost anywhere:
    - Outgoing buffer at source
    - Router or link
    - Incoming buffer at destination
- But adds port numbers
  - Used to identify "application layer" protocols or processes
- Also a checksum, optional

TCP (Transmission Control Protocol)

CS514

- Runs above IP
  - Port number and checksum like UDP
- Service is in-order byte stream
  - Application does not absolutely know how the bytes are packaged in packets
- Flow control and congestion control
- Connection setup and teardown phases
- Can be considerable delay between bytes in at source and bytes out at destination
  - Because of timeouts and retransmissions
- Works only with unicast (not multicast or anycast)

UDP vs. TCP

CS514

- UDP is more real-time
  - Packet is sent or dropped, but is not delayed
- UDP has more of a “message” flavor
  - One packet = one message
  - But must add reliability mechanisms over it
- TCP is great for transferring a file or a bunch of email, but kind-of frustrating for messaging
  - Interrupts to application don’t conform to message boundaries
  - No “Application Layer Framing”
- TCP is vulnerable to DoS (Denial of Service) attacks, because initial packet consumes resources at the receiver

SCTP (Stream Control Transmission Protocol)

CS514

- IETF standard
- Overcomes many limitations of TCP
  - Motivation is SS7 signaling over IP
  - Probably over-designed
- Message oriented---supports message framing
- Multiple streams for a given session
  - Interruption in one stream does not effect the others
- Cookie mechanism for DoS attacks
- By no means universally available

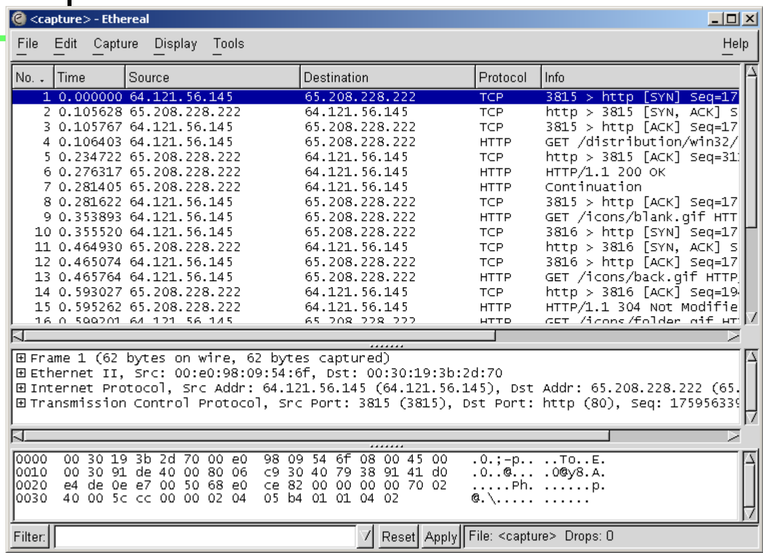
Ethereal

CS514

- Great open-source tool for understanding and debugging protocol behavior
- www.ethereal.com
- Features:
  - Trace packets over the wire
  - Sophisticated filtering language
  - Display contents of each protocol
  - Dump contents into file
  - Display TCP conversation

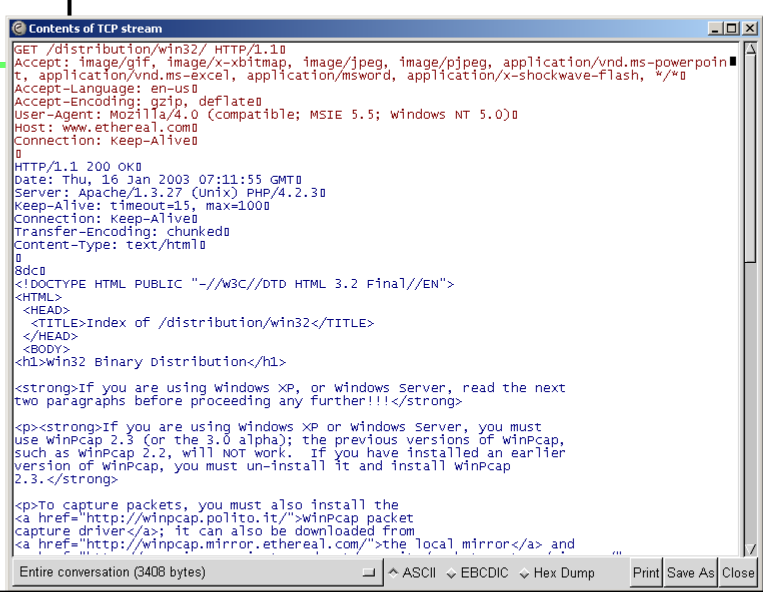
Captured Frames

CS514



TCP conversation

CS514



Supports these 340 protocols

CS514

802.11 MGT, AARP, AFP, AFS (RX), AH, AIM, AJP13, AODV, AODV6, ARCNET, ARP/RARP, ASAP, ASP, ATM, ATM LANE, ATP, AVS WLANCAP, Auto-RP, BACapp, BACnet, BEEP, BGP, BOOTP/DHCP, BOOTPARAMS, BOSSVR, BROWSER, BVLC, CDP, CDS_CLERK, CFLOW, CGMP, CHDLC, CLEARCASE, CLNP, CLTP, CONV, COPS, COTP, CPHA, CUPS, CoSine, DCCP, DCERPC, DCERPC_NT, DCE_DFS, DDP, DDTP, DEC_STP, DFS, DHCPv6, DLSw, DNS, DNSSERVER, DSI, DTSPROVIDER, DTSSTIME_REQ, DVMRP, Data, Diameter, EAP, EAPOL, EIGRP, EPM, ESIS, ESP, Ethernet, FC, FC ELS, FC-SWILS, FCIP, FCP, FDDI, FIX, FLDB, FR, FTP, FTP-DATA, FTSERVER, FW-1, Frame, GIOP, GMRP, GNUTELLA, GRE, GSS-API, GTP, GTPv0, GTPv1, GVRP, H.261, H1, HCLNFSD, HSRP, HTTP, HyperSCSI, IAPP, IB, ICAP, ICMP, ICMPv6, ICP, ICQ, IEEE 802.11, IGMP, IGRP, ILMI, IMAP, IP, IPComp, IPFC, IPP, IPX, IPX MSG, IPX RIP, IPX SAP, IPv6, IRC, ISAKMP, ISDN, ISIS, ISL, ISUP, IUA, KLM, KRB5, KRB5RPC, L2TP, LACP, LANMAN, LAPB, LAPBETHER, LAPD, LDAP, LDP, LLAP, LLC, LMI, LMP, LPD, LSA, LSA_DS, Lucent/Ascend, M2PA, M2TP, M2UA, M3UA, MAPI, MGMT, MMSE, MOUNT, MPEG1, MPLS, MRDISC, MS Proxy, MSDP, MSNIP, MTP2, MTP3, Mobile IP, Modbus/TCP, NBDS, NBIPX, NBNS, NBP, NBSS, NCP, NDMP, NDPS, NETLOGON, NFS, NFSACL, NFSAUTH, NIS+, NIS+ CB, NLM, NMPI, NNTP, NSPI, NTLMSSP, NTP, NetBIOS, Null, OSPF, OXID, PCNFSD, PFLOG, PGM, PIM, POP, PPP, PPP BACP, PPP BAP, PPP CBCP, PPP CCP, PPP CDPCP, PPP CHAP, PPP Comp, PPP IPCP, PPP IPV6CP, PPP LCP, PPP MP, PPP MPLSCP, PPP PAP, PPP PPPMux, PPP PPPMuxCP, PPP VJ, PPPoED, PPPoES, PPTP, Portmap, Prism, Q.2931, Q.931, QLLC, QUAKE, QUAKE2, QUAKE3, QUAKEWORLD, RADIUS, RANAP, REMACT, REP_PROC, RIP, RIPng, RMI, RPC, RPC_BROWSER, RPC_NETLOGON, RPL, RQUOTA, RSH, RSTAT, RSVP, RS_ACCT, RS_ATTR, RS_PGO, RS_REPADM, RS_REPLIST, RS_UNIX, RTCP, RTMP, RTP, RTSP, RWALL, RX, Raw, Rlogin, SADMIND, SAMR, SAP, SCCP, SCCPMG, SCSI, SCTP, SDP, SECIDMAP, SGI MOUNT, SIP, SKINNY, SLARP, SLL, SMB, SMB Mailslot, SMB Pipe, SMPP, SMTP, SMUX, SNA, SNAETH, SNMP, SPNEGO-KRB5, SPOOLSS, SPRAY, SPX, SRVLOC, SRVSVC, SSCOP, SSL, STAT, STAT-CB, STP, SUA, Serialization, SliMP3, Socks, Spnego, Syslog, TACACS, TACACS+, TAPI, TCP, TDS, TELNET, TFTP, TIME, TKN4Int, TNS, TPKT, TR MAC, TSP, Token-Ring, UBIKDISK, UBIKVOTE, UCP, UDP, V.120, VLAN, VRRP, VTP, Vines, Vines FRP, Vines SPP, WCCP, WCP, WHO, WINREG, WKSSVC, WSP, WTLS, WTP, X.25, X11, XDMCP, XOT, XYPLEX, YHOO, YPBIND, YPPASSWD, YPSERV, YPXFR, ZEBRA, ZIP, cds_solicit, cprpc_server, dce_update, iSCSI, roverride, rpriv, rs_misc, rsec_login,

Summary

CS514

- TCP, UDP, IP provide a nice set of basic tools
- But problems/limitations exist
  - IP has been compromised by NAT, can’t be used as a stable identifier
  - Firewalls can block communications
  - TCP has vulnerabilities
  - Network performance highly variable
- Next lecture we’ll look at other forms of naming and identification
  - Help overcome limitations of IP

CS514

- **“Any problem in computer science can be solved with another layer of indirection”**

  David Wheeler

Naming is a layer of indirection

CS514

- What problems does it solve?
  - Makes objects human readable
  - Hides complexity and dynamics
    - Multiple lower-layer objects can have one name
    - Changes in lower-layer objects hidden
  - Allows an object to be found in different ways
    - One object can have multiple names

Names map to objects through a resolution service

CS514

Identifiers and Locators

CS514

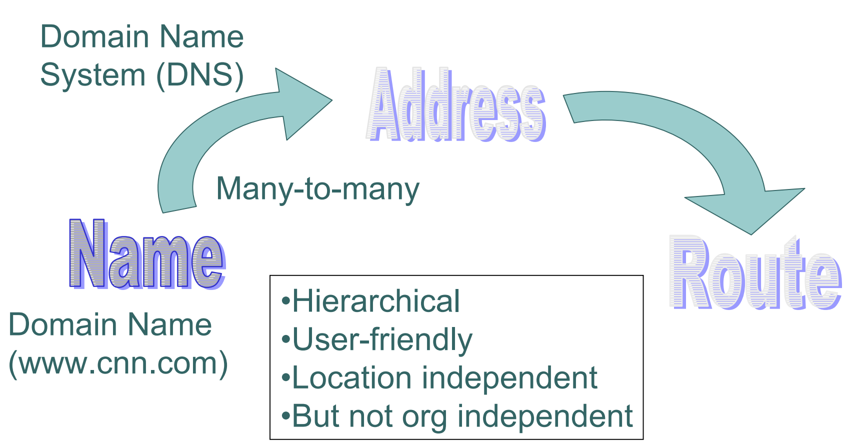
- A name is always an *identifier* to a greater or lesser extent
  - Can be persistent or non-persistent
  - Can be globally unique, locally unique, or even non-unique
- If a name has structure that helps the resolution service, then the name is also a *locator*

Naming in networks

CS514

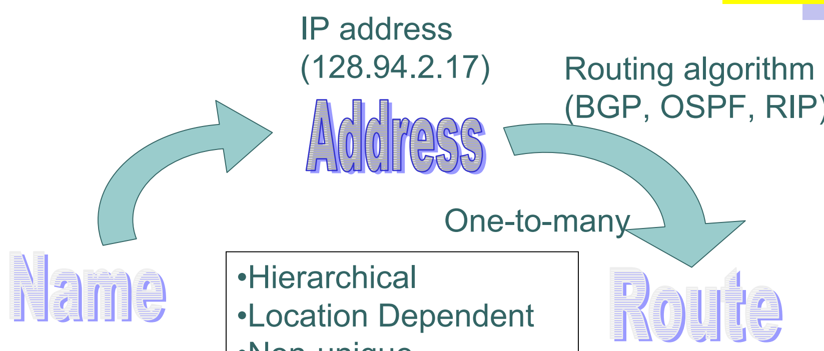
DNS names map into addresses

CS514



Addresses map into routes

CS514



- Hierarchical
- Location Dependent
- Non-unique
- Can change often
- Refers to an interface, not a host

Routes get packets to interfaces

CS514

- A path
- Source dependent
- Can change often

DNS names and IP addresses are identifiers and locators

CS514

- Both are typically non-persistent
- Private IP addresses identify only in the context of an IP realm
- Domain names are good identifiers
  - woodstock.cs.cornell.edu identifies a host
  - www.cnn.com identifies a service
- URLs are good identifiers

Domain Name System (DNS)

CS514

- Distributed directory service
- Hierarchical name space
- Each level separated by '.'
  - Analogous to '/' separator in file systems
- One global root
  - Replicated across <20 root servers!
  - There have been Denial of Service (DoS) attacks on these root servers, none real successful
  - Because of caching, queries to root servers relatively rare
- DNS maybe only global directory service???

DNS is simple but powerful

CS514

- Only one type of query
  - Query(domain name, RR type)
    - Resource Record (RR) type is like an attribute type
  - Answer(values, additional RRs)
- Limited number of RR types
- Hard to make new RR types
  - Not for technical reasons…
  - Rather because each requires global agreement

DNS is the core of the Internet

CS514

- Global name space
  - Can be the core of a naming or identifying scheme
- Global directory service
  - Can resolve a name to nearly every computer on the planet

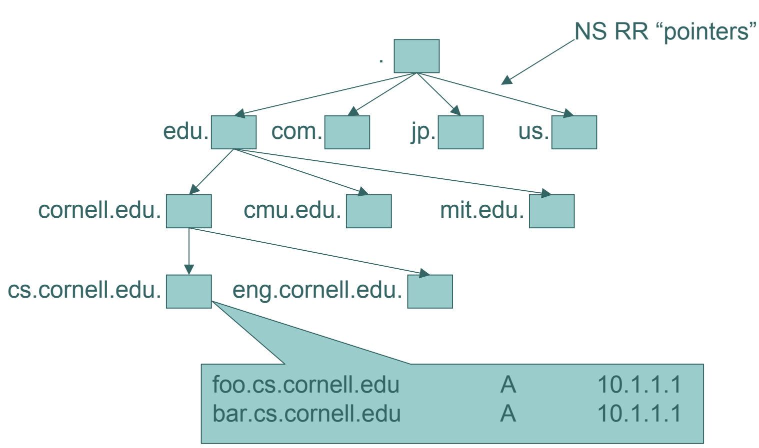
Important DNS RR types

CS514

- **NS**: Points to next Name Server down the tree
- **A**: Contains the IP address
  - **AAAA** for IPv6
- **MX**: Contains the name of the mail server
- Service-oriented RR types
  - **SRV**: Contains addresses and ports of services on servers
    - One way to learn what port number to use
  - **NAPTR**: Essentially a generalized mapping from one name space (i.e. phone numbers) to another (i.e. SIP URL)

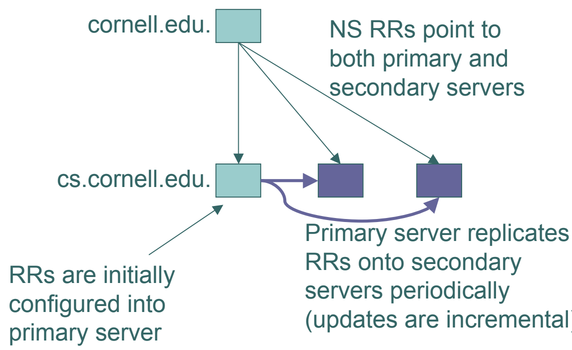
DNS tree structure

CS514



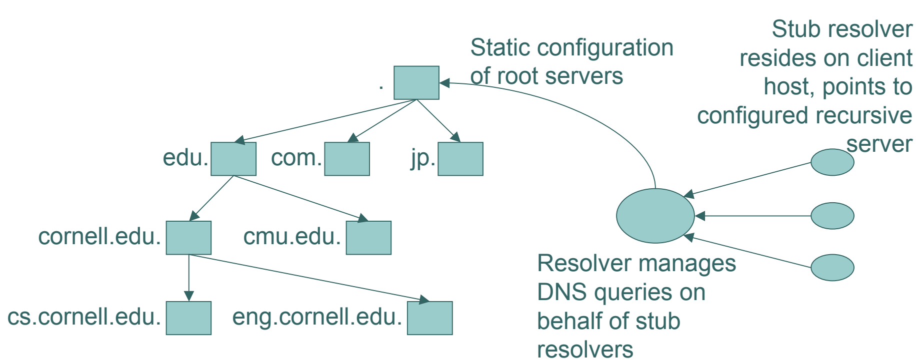
Primary and secondary servers

CS514



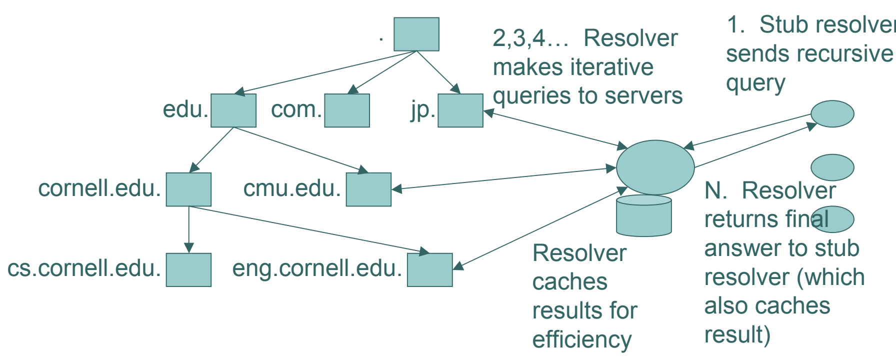
Resolver structure and configuration

CS514



Resolver structure and configuration

CS514



DNS cache management

CS514

- All RRs have Time-to-live (TTL) values
- When TTL expires, cache entries are removed
- NS RRs tend to have long TTLs
  - Cached for a long time
  - Reduces load on higher level servers
- A RRs may have very short TTLs
  - Order one minute for some web services
  - Order one day for typical hosts

Why is DNS iterative and not recursive?

CS514

- Recursive can be more efficient
  - Better caching characteristics
    - Caches in servers, not just resolvers
  - Shorter paths
- However, high-performance recursive server much harder to implement
  - Maintain state for thousands of concurrent queries
  - Manage cache
- Recursive server prone to DoS attacks

LDAP is another popular distributed directory service

CS514

- Richer and more general than DNS
  - Has generalized attribute/value scheme
  - Can search on attribute, not just name
    - But this doesn't scale well
- Simpler and more efficient than a full relational database
- Not a global directory service, though namespace is global
  - Its predecessor, X.500, was meant to be
  - But "local" LDAP services can point to each other
- Commonly used for personnel databases, subscriber databases

URLs, URNs, and URIs

CS514

- Uniform Resource <Locator, Name, Identifier>
- URL tells a computer where and how to reach a resource
  - These came first
- URN is a true identifier
  - Unique, persistent
- URI refers to both URLs and URNs
  - Defines syntax for current and future URLs and URNs
- *For now we only really care about URLs*

URL

CS514

- Consists of:

<scheme>:<scheme-specific-part>

URL

CS514

- Consists of:

<scheme>:<scheme-specific-part>

A protocol

Information the protocol needs

URL examples

CS514

- HTTP (web)
  - http://www.cnn.com/news/story.html
- Email
  - mailto://francis@cs.cornell.edu
- Newsgroups
  - news:cornell/class/cs514
- SIP (Session Initiation Protocol)
  - sip://service@phone.verizon.com

Note the central role of DNS

CS514

- HTTP (web)
  - http://***www.cnn.com***/news/story.html
- Email
  - mailto://francis@***cs.cornell.edu***
- Newsgroups
  - news:cornell/class/cs514
- SIP (Session Initiation Protocol)
  - sip://service@***phone.verizon.com***

How to identify in the application?

CS514

- Obviously you don't want to use an IP address to identify a service
  - Can change for many reasons
  - Person who manages the IP address has no knowledge of the application

How to identify in the application?

CS514

- DNS is much better, but not perfect
  - Can return out of date information
  - Because of caches or stale secondaries
    - CDNs use very small timeouts
    - High-end DNS products have fast replication
  - DNS itself often incurs substantial delay (several second retry timeout)
  - DNS prone to misconfiguration
  - Ultimately, application itself not able to insure that DNS is working

How to identify in the application?

CS514

- What about middleware solutions?
  - A complete middleware solution may be more robust
  - Application talks to the middleware service (i.e. can register itself)
  - We'll look at this type of solution next lecture

Summary of Naming Lecture

CS514

Introduction to Naming

- Naming basics:
  - Names, Addresses, Routes
  - Identifiers and Locators
- DNS is *the* global directory service
  - LDAP is a popular local directory service
- URLs build on DNS (and also URIs and URNs)
- IP is lousy for naming. DNS is better, but not perfect.