DISTRIBUTED SYSTEMS

Communication Basics (Networking Review)

Some slides adapted from material by Kurose and Ross for their book: *Computer Networking: A Top Down Approach Featuring the Internet*

Overview of Lecture

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

Who recognizes this?

```
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);
```

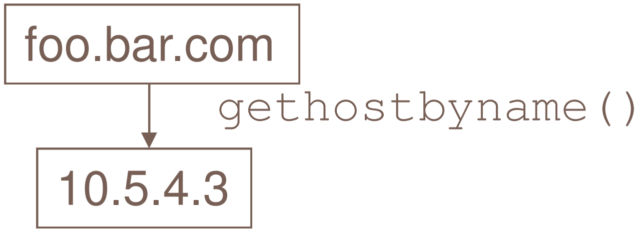
Classic view of network API

- Start with host name (maybe)

foo.bar.com

Classic view of network API

- □ Start with host name
- □ Get an IP address



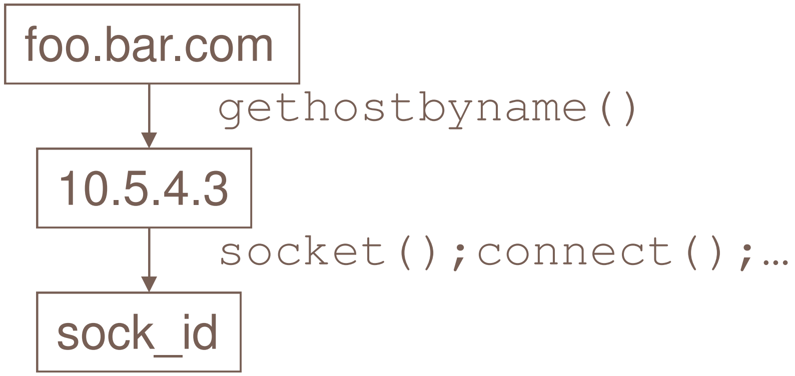
foo.bar.com

`gethostbyname()`

10.5.4.3

Classic view of network API

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



foo.bar.com

`gethostbyname()`

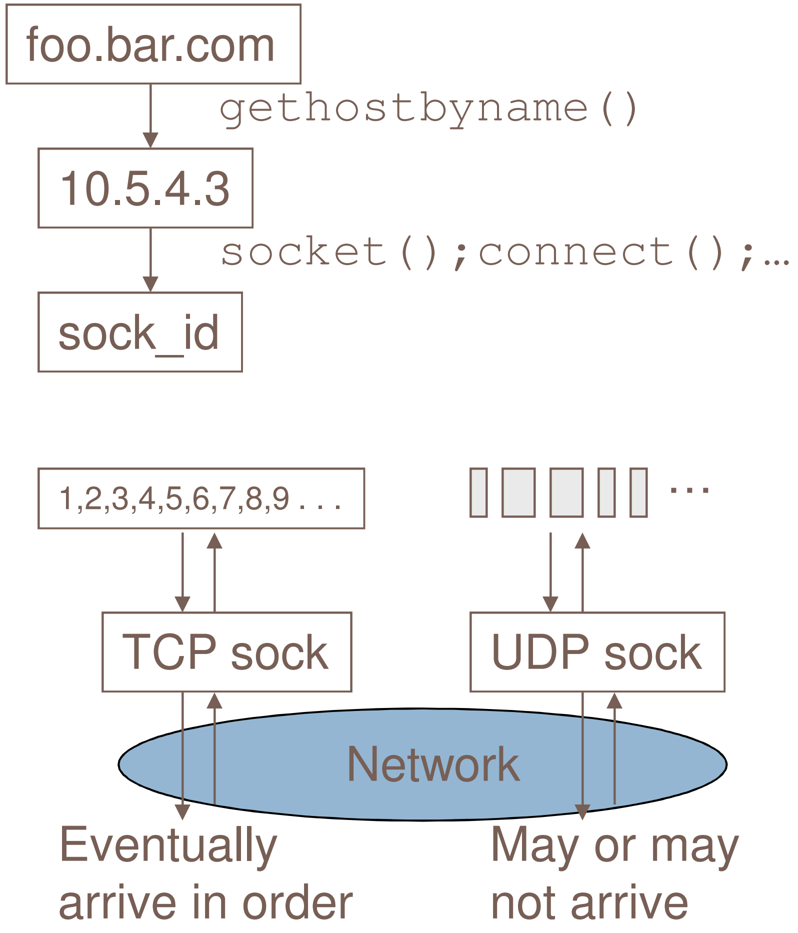
10.5.4.3

`socket();connect();…`

sock_id

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)



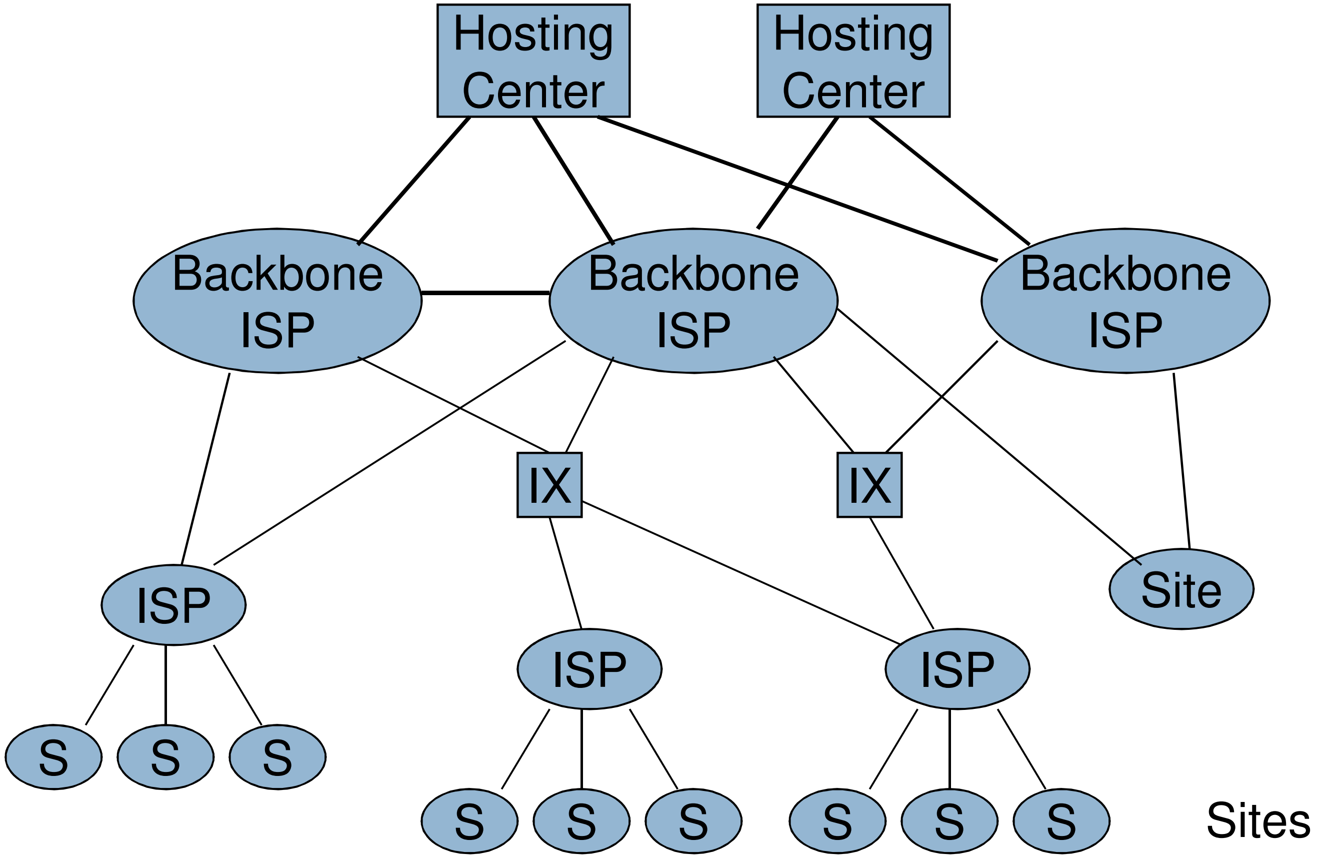
Classic approach “broken” in many ways

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

Network components

- **Network:** Collection of hosts, links, and routers
- **Site:** Stub (or edge) network, typically in one location and under control of one administration
- **ISP:** Internet Service Provider. Transit network that provides IP connectivity for sites
- **Firewall/NAT:** Box between the site and ISP that provides filtering, security, and Network Address Translation
- **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

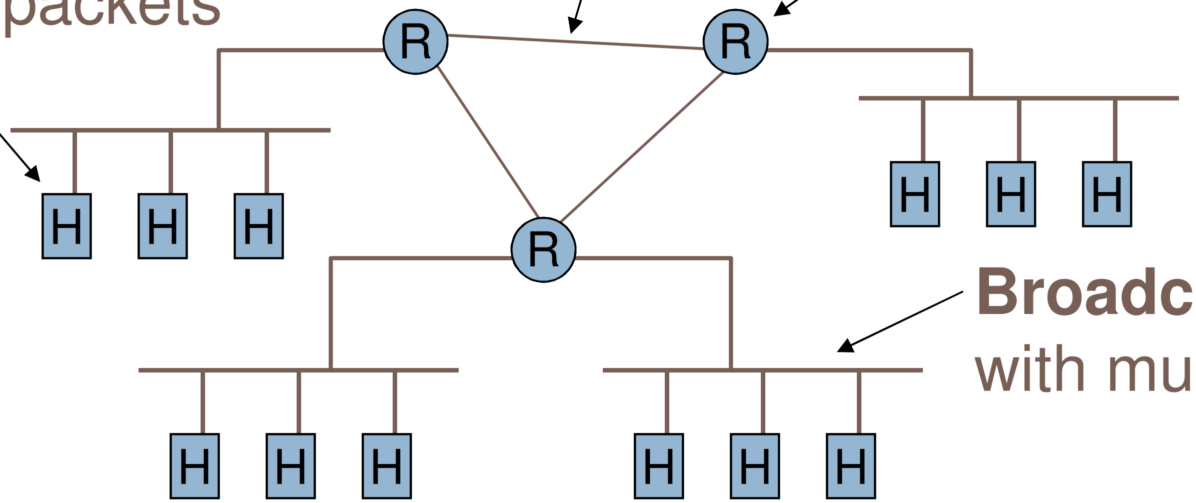


IXs came first

IXs tend to be performance bottlenecks

Hosting centers and bilateral peering are a response to poor IXs

Network components



Point to point link: link with two nodes (router or host)

Router: Forwards IP packets

Host: Source and sink of IP packets

Broadcast link: link with multiple nodes

Protocol "Layers"

Networks are complex!

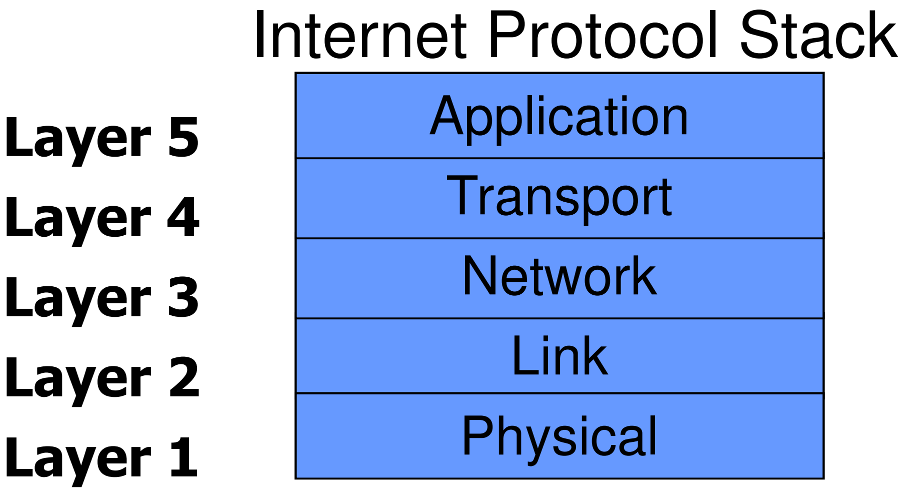
- Many "pieces":
  - hosts
  - routers
  - links of various media
  - applications
  - protocols
  - hardware, software

Question:

Is there any hope of *organizing* structure of network?

Or at least our discussion of networks?

Internet Layering Model



Internet Protocol Stack

Layer 5	Application
Layer 4	Transport
Layer 3	Network
Layer 2	Link
Layer 1	Physical

- Each layer uses the function of the layer below
- Each layer exports functionality to layer above
- This layering of protocol behavior called a "protocol stack"
- Aka, the "TCP/IP stack"

Why layering?

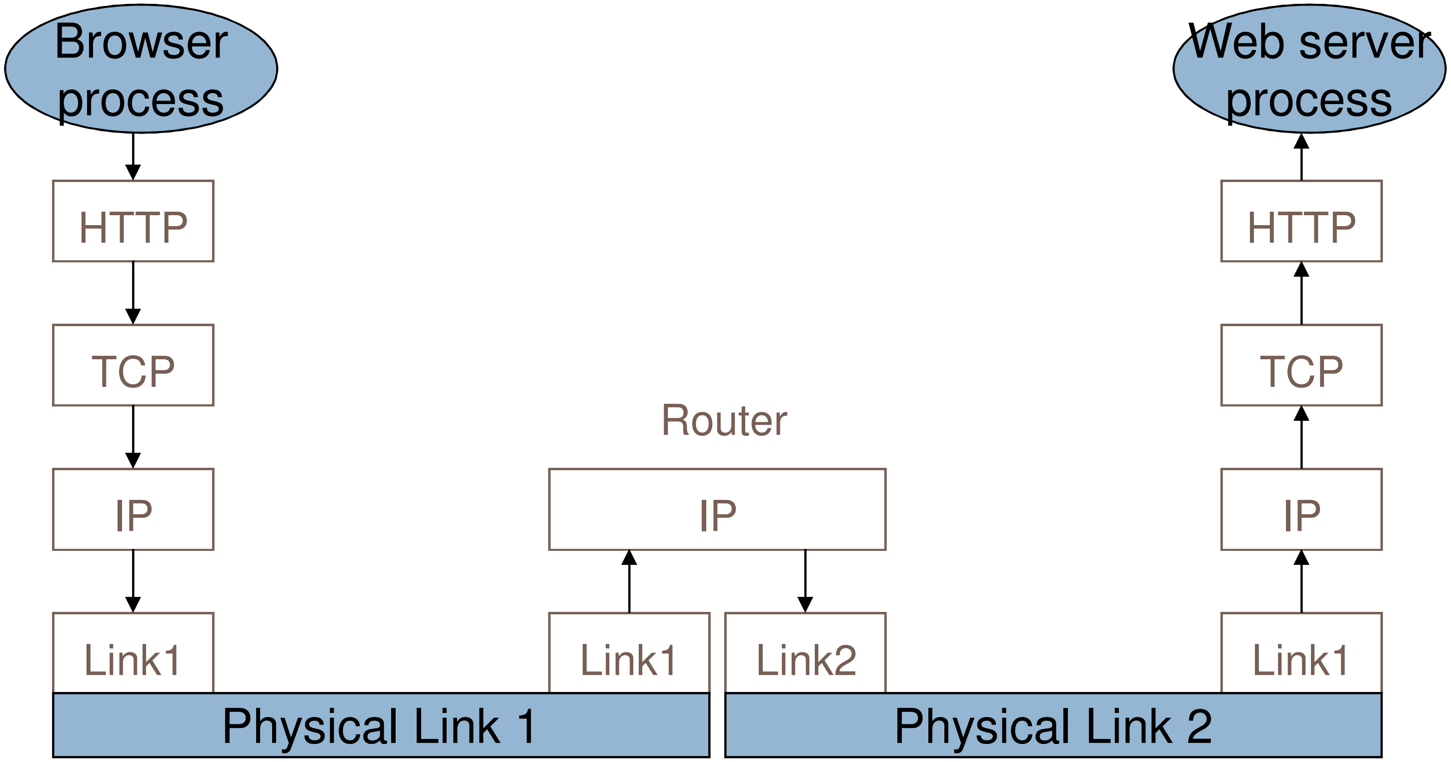
Dealing with complex systems:

- explicit structure allows identification, relationship of complex system's pieces
  - layered reference model for discussion
- modularization eases maintenance, updating of system
  - change of implementation of layer's service transparent to rest of system
- layering considered harmful?

Protocol layering (cont'd)

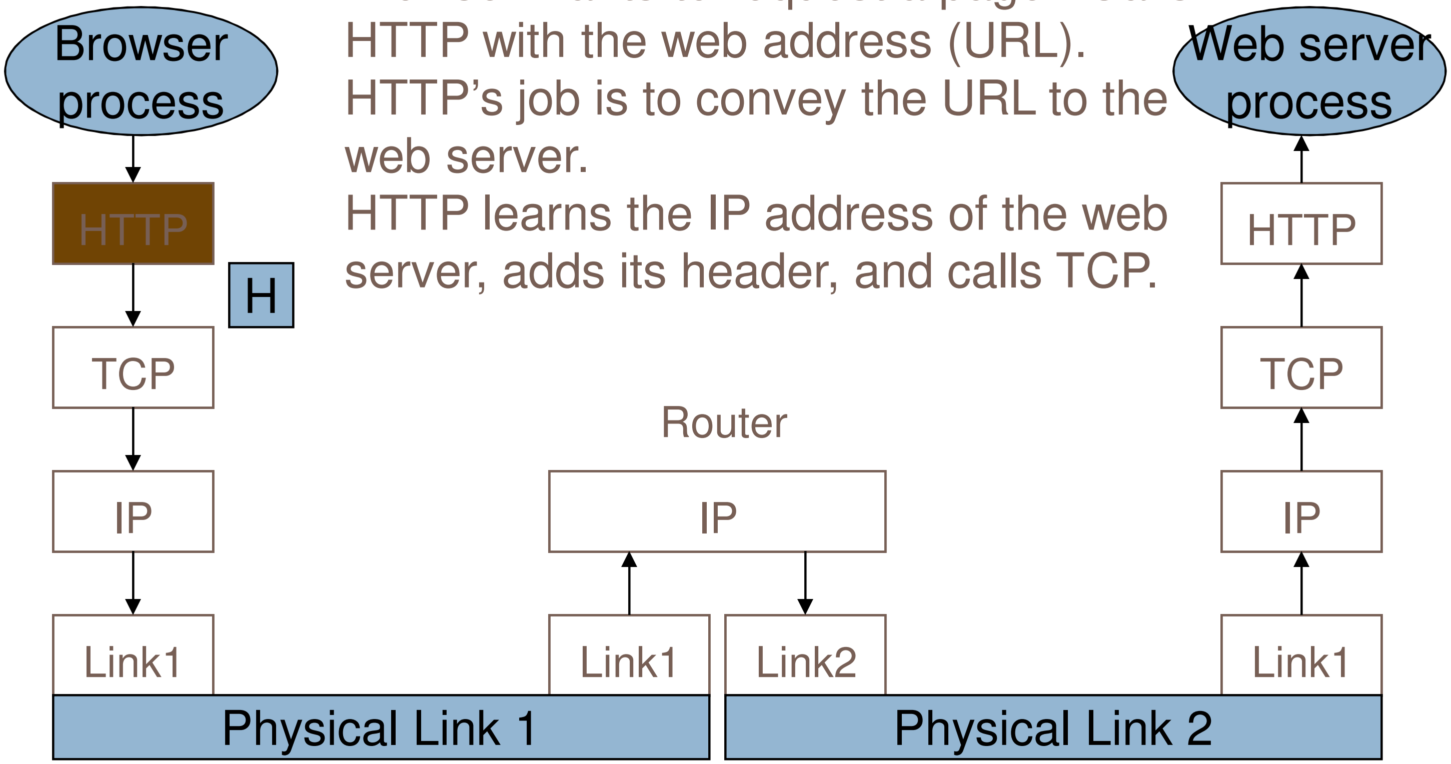
- Communications stack consists of a set of services, each providing a service to the layer above, and using services of the layer below
  - Each service has a programming API, just like any software module
- Each service has to convey information to one or more peers across the network
- This information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

Protocol layering example



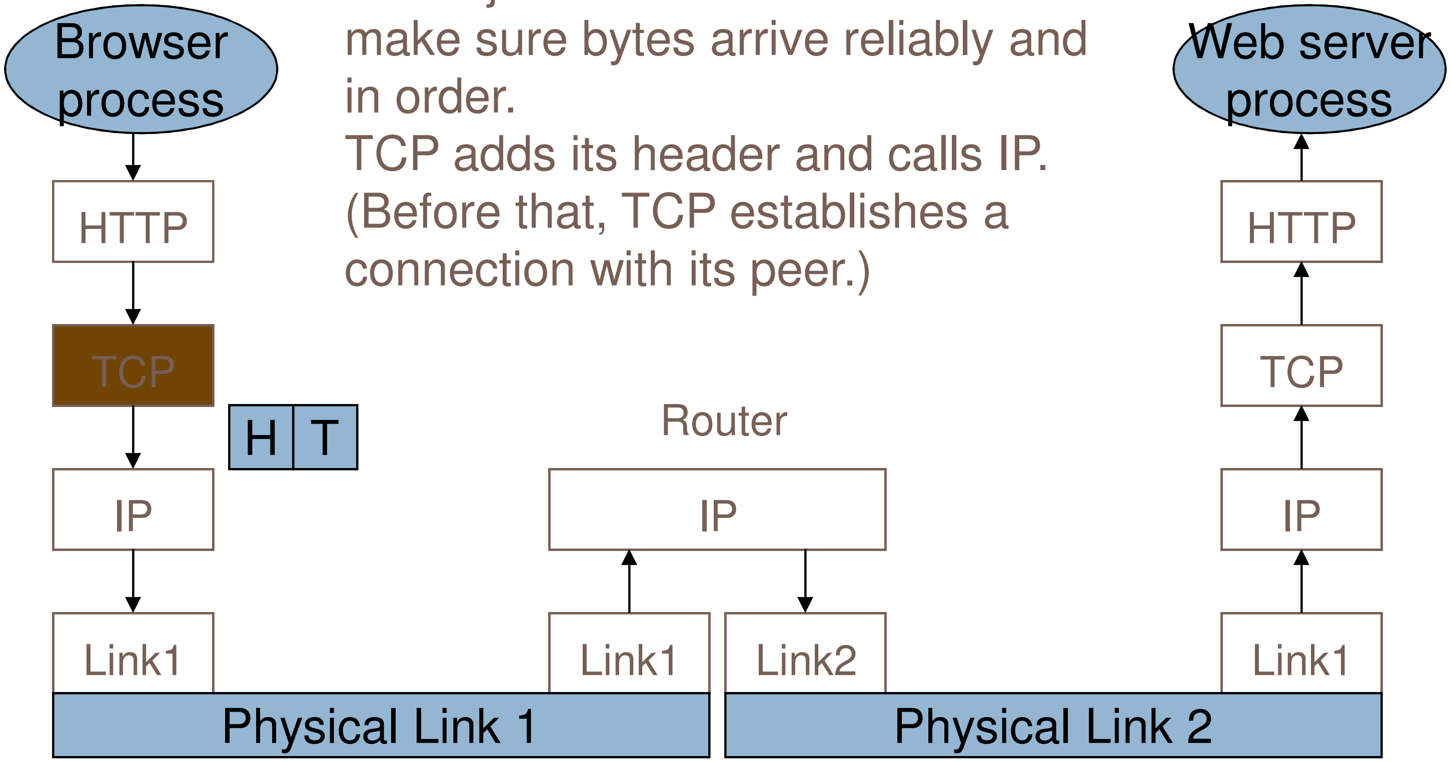
Protocol layering example

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

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.)

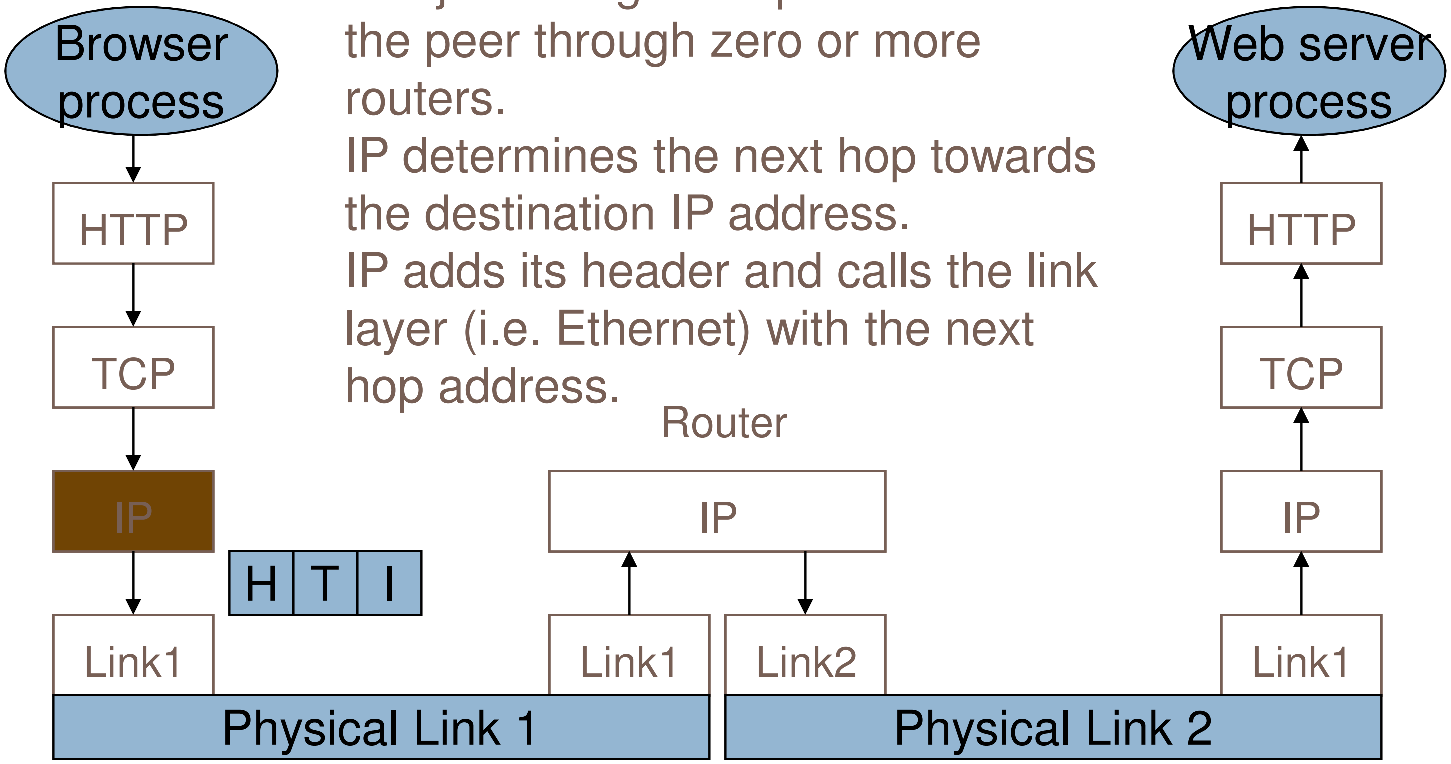


Protocol layering example

IP's job is to get the packet routed to the peer through zero or more routers.

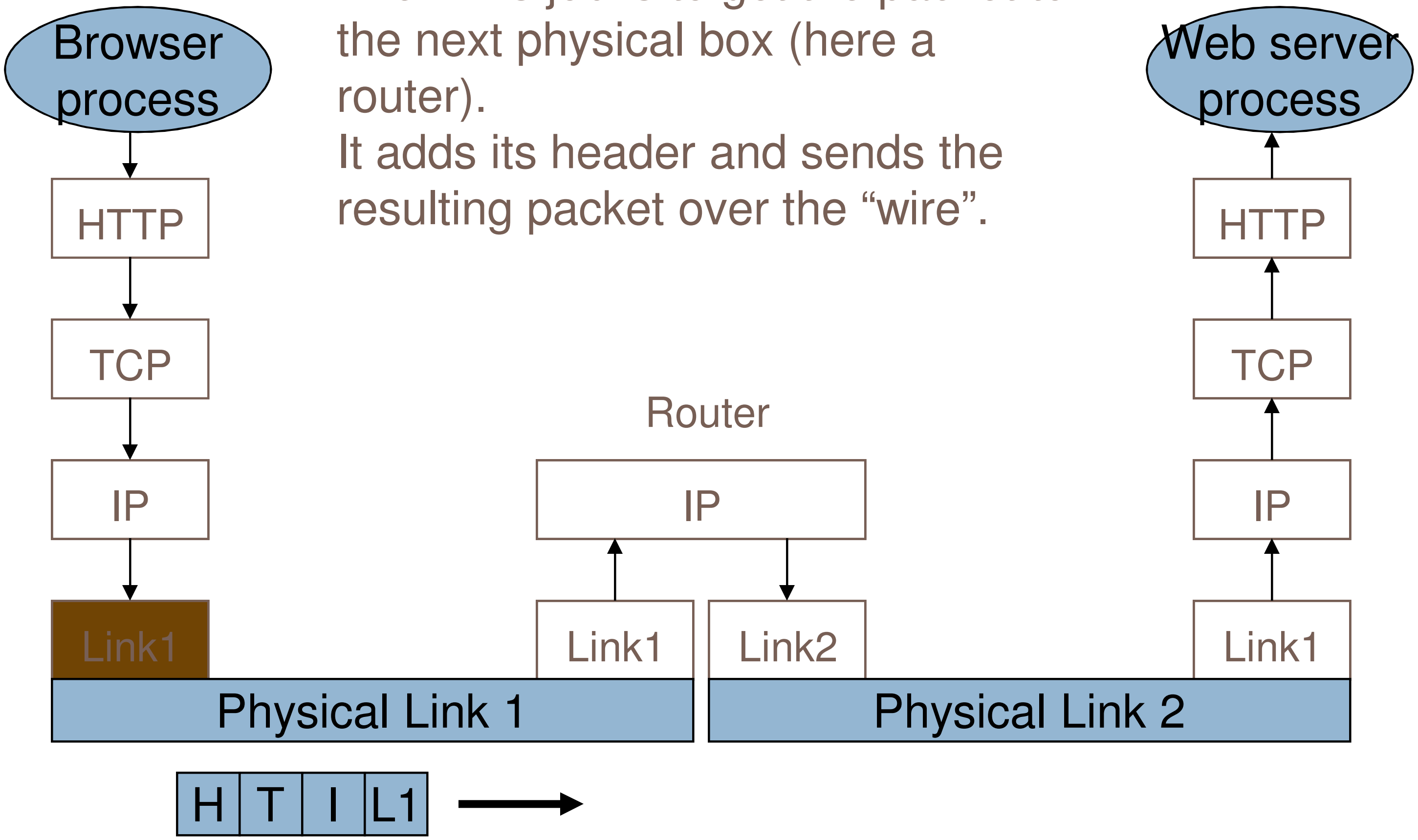
IP determines the next hop towards the destination IP address.

IP adds its header and calls the link layer (i.e. Ethernet) with the next hop address.



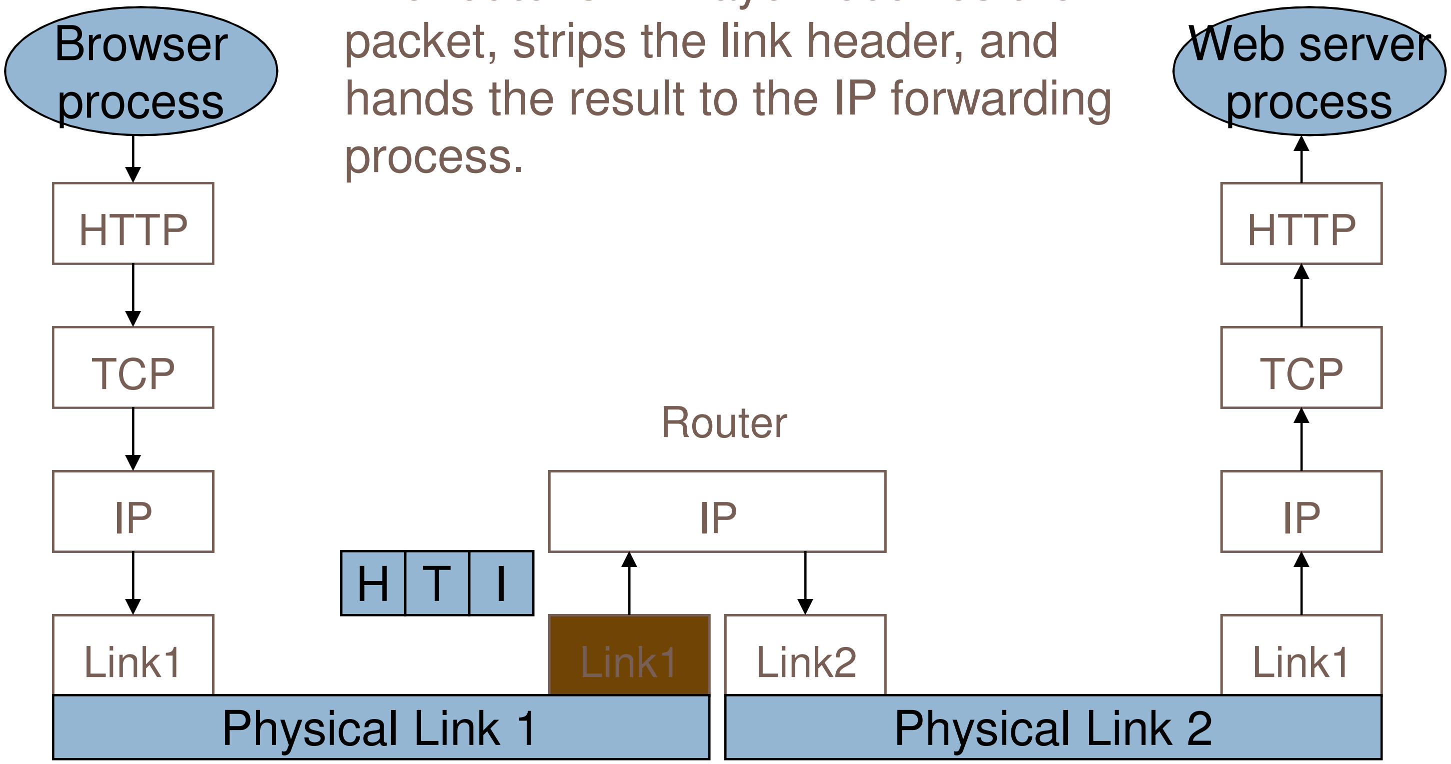
Protocol layering example

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".



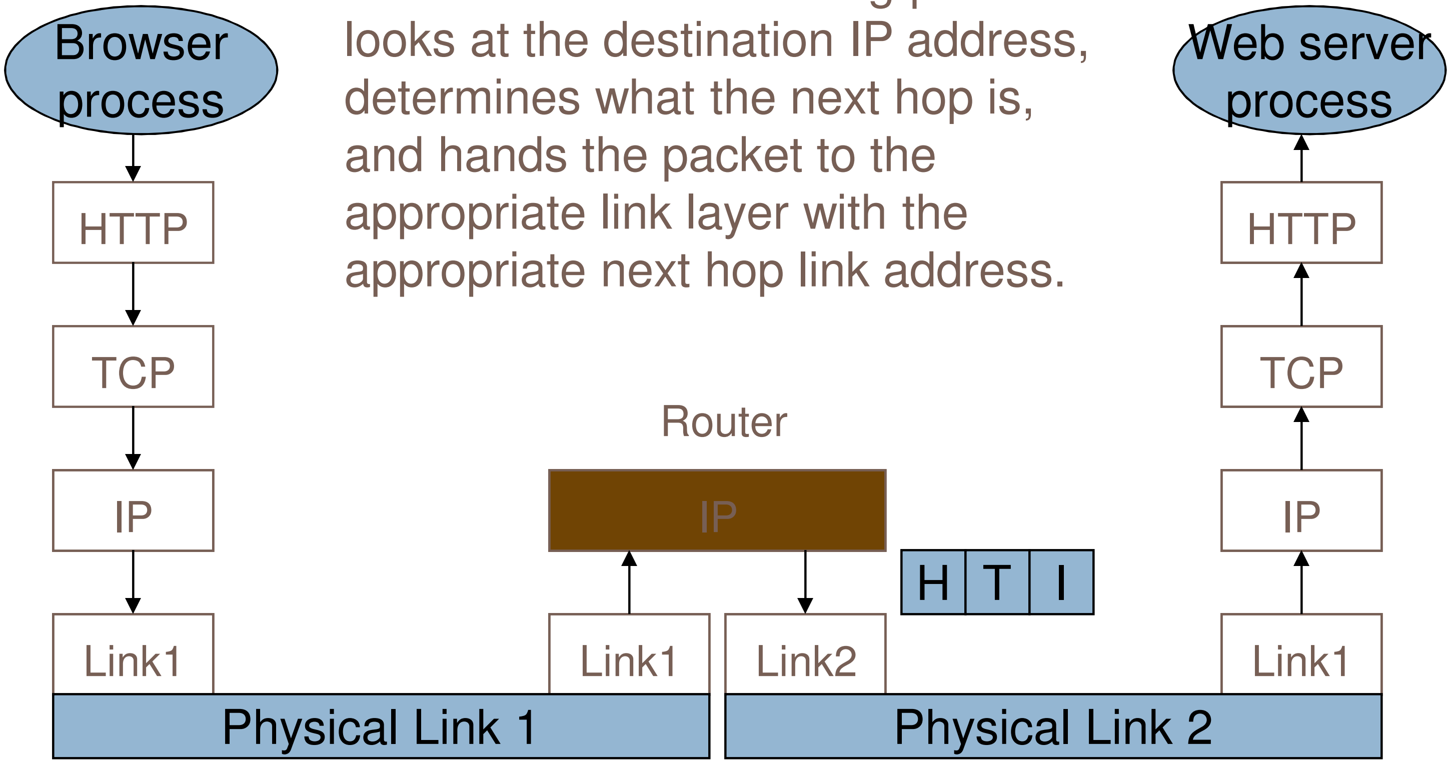
Protocol layering example

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



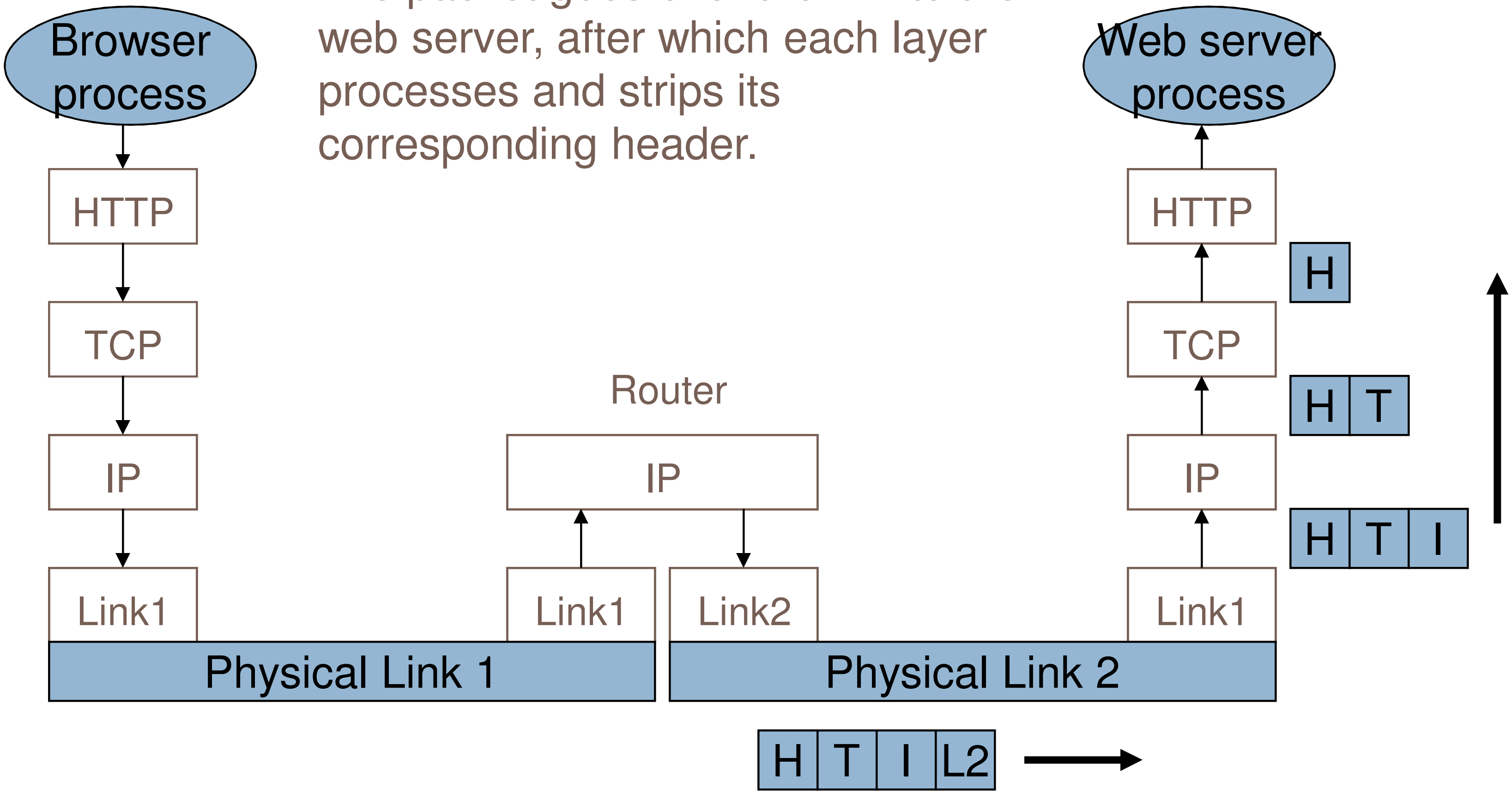
Protocol layering example

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

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

- 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 fields

- 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 others
  - Dynamically negotiated ports indicate specific processes (for these and other protocols)
- HTTP: Host field
  - Indicates "virtual web server" within a physical web server
  - More like an identifier than a demuxing field

IP (Internet Protocol)

- 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
  - Packets may be dropped, duplicated, and received in a different order
  - Best-effort service

IP address

- 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

- 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)

- 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
  - Why?
- Also a checksum, optional

TCP (Transmission Control Protocol)

- 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
  - Why are these needed?
- 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

- 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
  - Why?

SCTP (Stream Control Transmission Protocol)

- 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

Revisiting the end-to-end argument

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

- 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

- 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?

- 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
    - (IP) Routing algorithms
    - Billing
    - Membership agreement

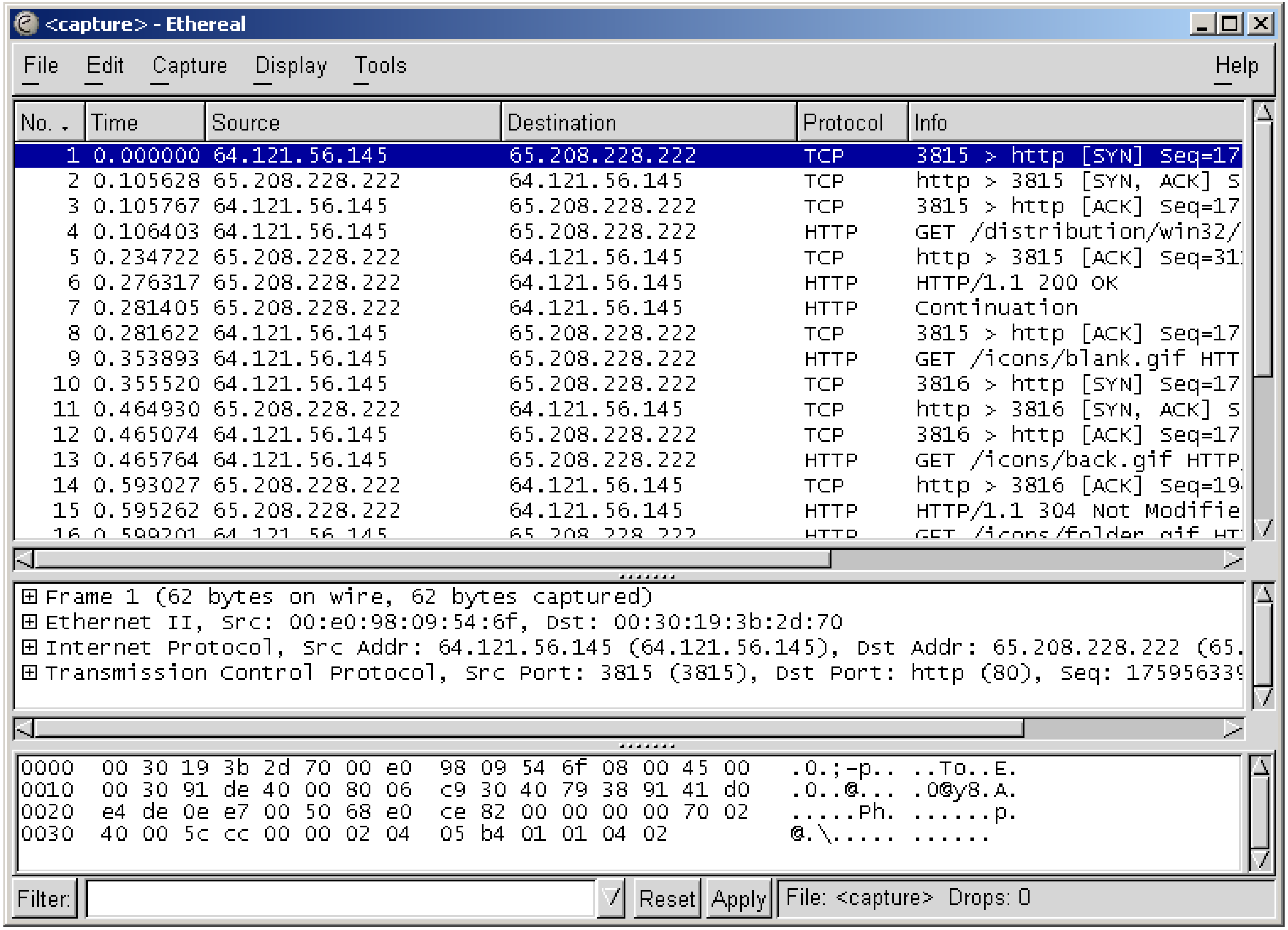
The E2E Debate

- 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"
- Mis-applied in network neutrality debate
  - (treat all network content equally, do not discriminate)
  - Purported E2E Claim: network should be "dumb" and not muck with app data
- Sometimes performance hurt in deference to E2E
  - Compression of Voice over IP (RTP, Real Time Protocol)
  - Mobile IP

Wireshark

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

Captured Frames

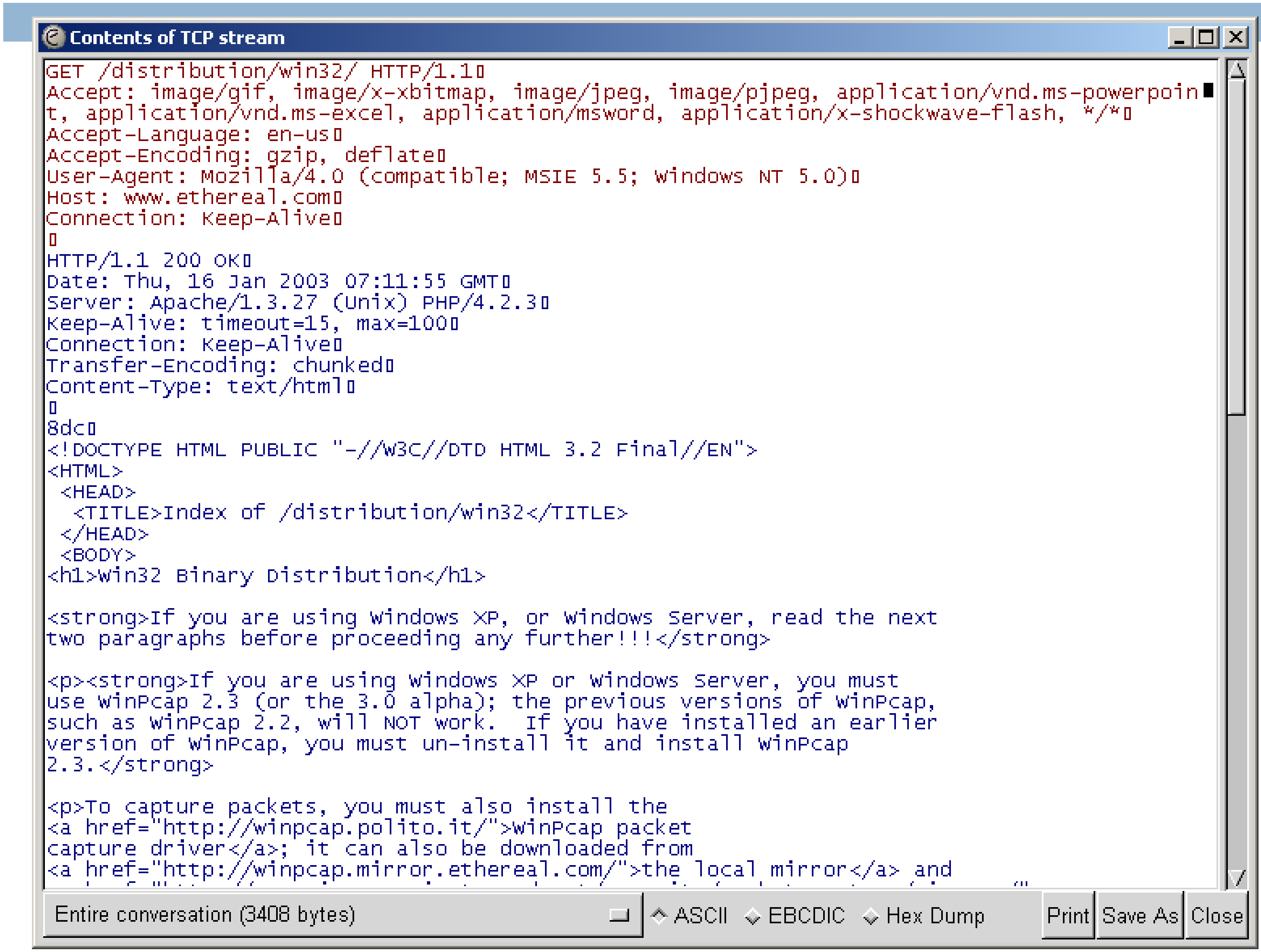


```
No. .  Time     Source          Destination      Protocol  Info
    1 0.000000 64.121.56.145   65.208.228.222   TCP       3815 > http [SYN] Seq=17
    2 0.105628 65.208.228.222  64.121.56.145    TCP       http > 3815 [SYN, ACK] S
    3 0.105767 64.121.56.145   65.208.228.222   TCP       3815 > http [ACK] Seq=17
    4 0.106403 64.121.56.145   65.208.228.222   HTTP      GET /distribution/win32/
    5 0.234722 65.208.228.222  64.121.56.145    TCP       http > 3815 [ACK] Seq=31
    6 0.276317 65.208.228.222  64.121.56.145    HTTP      HTTP/1.1 200 OK
    7 0.281405 65.208.228.222  64.121.56.145    HTTP      Continuation
    8 0.281622 64.121.56.145   65.208.228.222   TCP       3815 > http [ACK] Seq=17
    9 0.353893 64.121.56.145   65.208.228.222   HTTP      GET /icons/blank.gif HTT
   10 0.355520 64.121.56.145   65.208.228.222   TCP       3816 > http [SYN] Seq=17
   11 0.464930 65.208.228.222  64.121.56.145    TCP       http > 3816 [SYN, ACK] S
   12 0.465074 64.121.56.145   65.208.228.222   TCP       3816 > http [ACK] Seq=17
   13 0.465764 64.121.56.145   65.208.228.222   HTTP      GET /icons/back.gif HTTP
   14 0.593027 65.208.228.222  64.121.56.145    TCP       http > 3816 [ACK] Seq=19
   15 0.595262 65.208.228.222  64.121.56.145    HTTP      HTTP/1.1 304 Not Modifie
   16 0.599201 64.121.56.145   65.208.228.222   HTTP      GET /icons/folder.gif HT
```

```
⊞ Frame 1 (62 bytes on wire, 62 bytes captured)
⊞ Ethernet II, Src: 00:e0:98:09:54:6f, Dst: 00:30:19:3b:2d:70
⊞ Internet Protocol, Src Addr: 64.121.56.145 (64.121.56.145), Dst Addr: 65.208.228.222 (65.
⊞ Transmission Control Protocol, Src Port: 3815 (3815), Dst Port: http (80), Seq: 175956339
```

```
0000  00 30 19 3b 2d 70 00 e0  98 09 54 6f 08 00 45 00   .0.;-p.. ..To..E.
0010  00 30 91 de 40 00 80 06  c9 30 40 79 38 91 41 d0   .0..@... .0@y8.A.
0020  e4 de 0e e7 00 50 68 e0  ce 82 00 00 00 00 70 02   .....Ph. ......p.
0030  40 00 5c cc 00 00 02 04  05 b4 01 01 04 02         @.\..... ......
```

TCP conversation



```
GET /distribution/win32/ HTTP/1.1▯
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoin
t, application/vnd.ms-excel, application/msword, application/x-shockwave-flash, */*▯
Accept-Language: en-us▯
Accept-Encoding: gzip, deflate▯
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)▯
Host: www.ethereal.com▯
Connection: Keep-Alive▯
▯
HTTP/1.1 200 OK▯
Date: Thu, 16 Jan 2003 07:11:55 GMT▯
Server: Apache/1.3.27 (Unix) PHP/4.2.3▯
Keep-Alive: timeout=15, max=100▯
Connection: Keep-Alive▯
Transfer-Encoding: chunked▯
Content-Type: text/html▯
▯
8dc▯
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
 <HEAD>
  <TITLE>Index of /distribution/win32</TITLE>
 </HEAD>
 <BODY>
<h1>Win32 Binary Distribution</h1>

<strong>If you are using Windows XP, or Windows Server, read the next
two paragraphs before proceeding any further!!!</strong>

<p><strong>If you are using Windows XP or Windows Server, you must
use WinPcap 2.3 (or the 3.0 alpha); the previous versions of WinPcap,
such as WinPcap 2.2, will NOT work.  If you have installed an earlier
version of WinPcap, you must un-install it and install WinPcap
2.3.</strong>

<p>To capture packets, you must also install the
<a href="http://winpcap.polito.it/">WinPcap packet
capture driver</a>; it can also be downloaded from
<a href="http://winpcap.mirror.ethereal.com/">the local mirror</a> and
```

Supports over 300 protocols

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

- TCP, UDP, IP provide a nice set of basic tools
  - Key is to understand the concept of protocol layering
- 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 we'll look at other forms of naming and identification
  - Help overcome limitations of IP