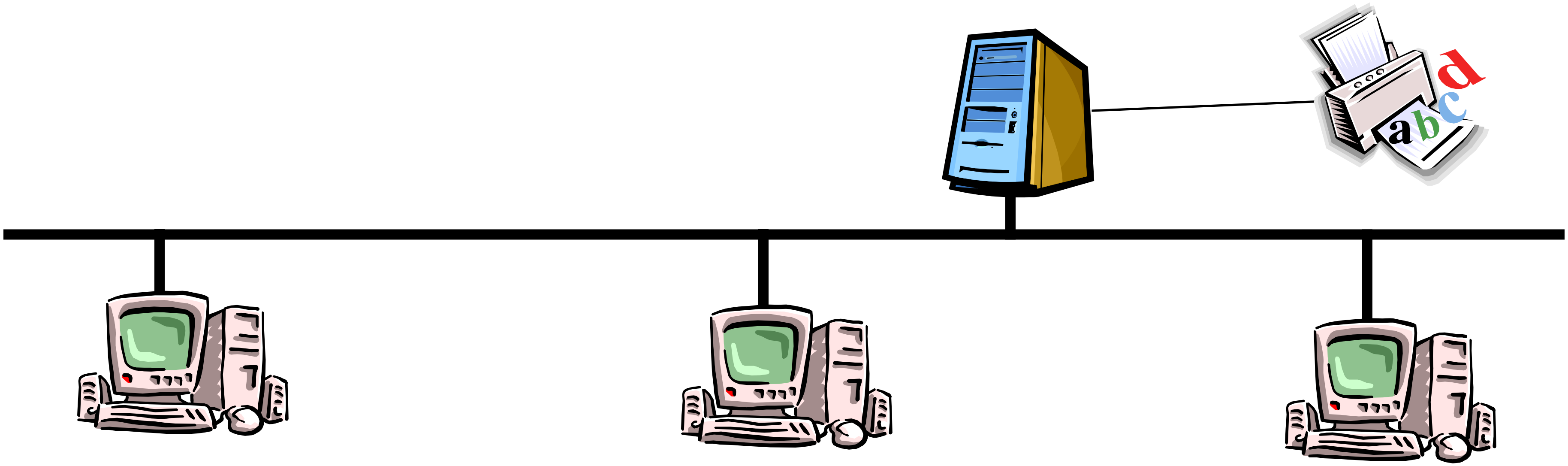
CS 360

Networking Intro: TCP/ip

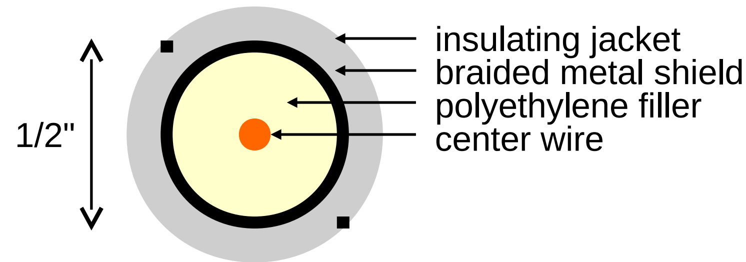
Ethernet LAN Technology



- **Connections:**
  - ✔ each computer has an Ethernet card (special HW with a unique *address*)
  - ✔ every computer's card is connected to the same single piece of wire (a *cable*)
- **Messages:**
  - ✔ sender broadcasts a structured chunk of bytes (a *packet*) onto cable
  - ✔ packet has address of sender and receiver, plus body of message (the *payload*)
  - ✔ only the card with the receiver address accepts the message
  - ✔ if two senders attempt to broadcast at same time (a *collision*), they both wait a random length of time and then try again

Ethernet = Medium Speed & Distance

- **History:**
  - ✔ packet switched LAN technology
  - ✔ invented at Xerox PARC 1970's
  - ✔ now IEEE standard 802.3



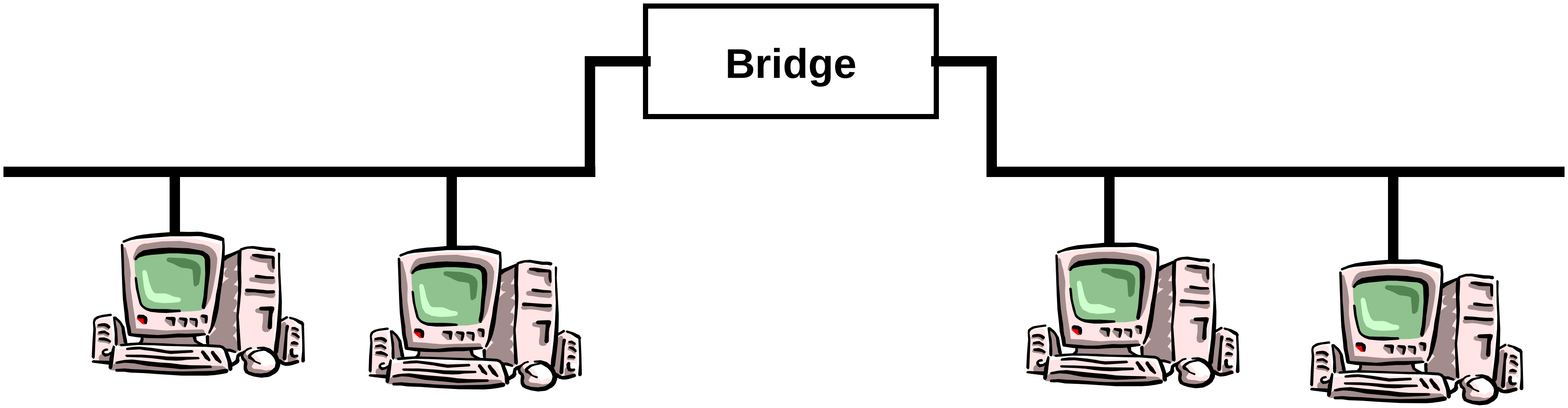
Ethernet thick coax
cable cross section

- **Hardware:**
  - ✔ max cable length about 500 meters (can be extended with *repeaters*)
  - ✔ approximately 10 Mbps max transfer rate (= few fast or several slow computers)
  - ✔ several cabling schemes (thick & thin coax, twisted pair 10Base-T)
- **Addressing:**
  - ✔ addresses are 48 bits long and are fixed at the factory
  - ✔ this is a *physical* address, which stays with a computer where ever it moves

Preamble	Destination Address	Source Address	Frame Type	Frame Data	CRC
8 octets	6 octets	6 octets	2 octets	64 .. 1500 octets	4 octets

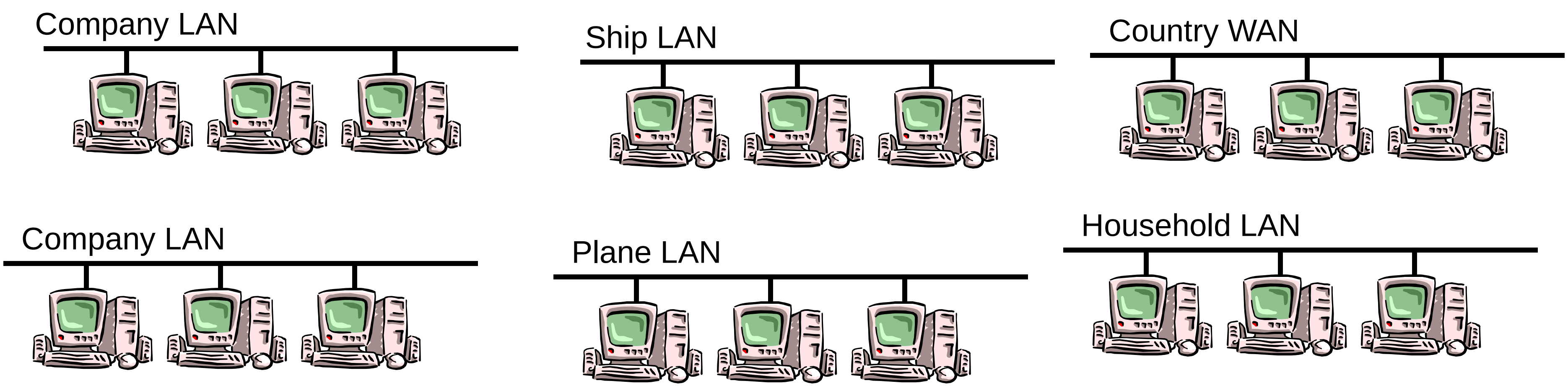
Ethernet packet layout

Bridges Connect Separate Ethernets



- **Operation:**
  - ✔ the bridge has a list of the Ethernet addresses on each separate net
  - ✔ packets are accepted from one side and forwarded to the other side, as needed
  - ✔ the bridge is invisible (*transparent*) to the Ethernet cards
- **Attributes:**
  - ✔ an adaptive bridge learns each side's addresses and forwards only as needed
  - ✔ bridges do not propagate electrical noise (cf. repeaters) & follow collision rules, so they can be used to extend an Ethernet almost indefinitely

Still, We Have A Problem of Scale



- **How to connect large numbers of completely separate networks?**
  - ✔ the networks may have incompatible physical addresses & data representations
  - ✔ there may be multiple paths from destination to source
  - ✔ machines assigned to various functions may change
  - ✔ no single entity can know where every thing is
  - ✔ response times cannot be guaranteed
- **Solution = "internetworking":**
  - ✔ agreements (*protocols*) that networks obey in communicating with each other
  - ✔ most successful protocol is TCP/IP ("<u>the</u> Internet")

protocol = treaty:
binds behavior,
hides local peculiarities

The Original Idea

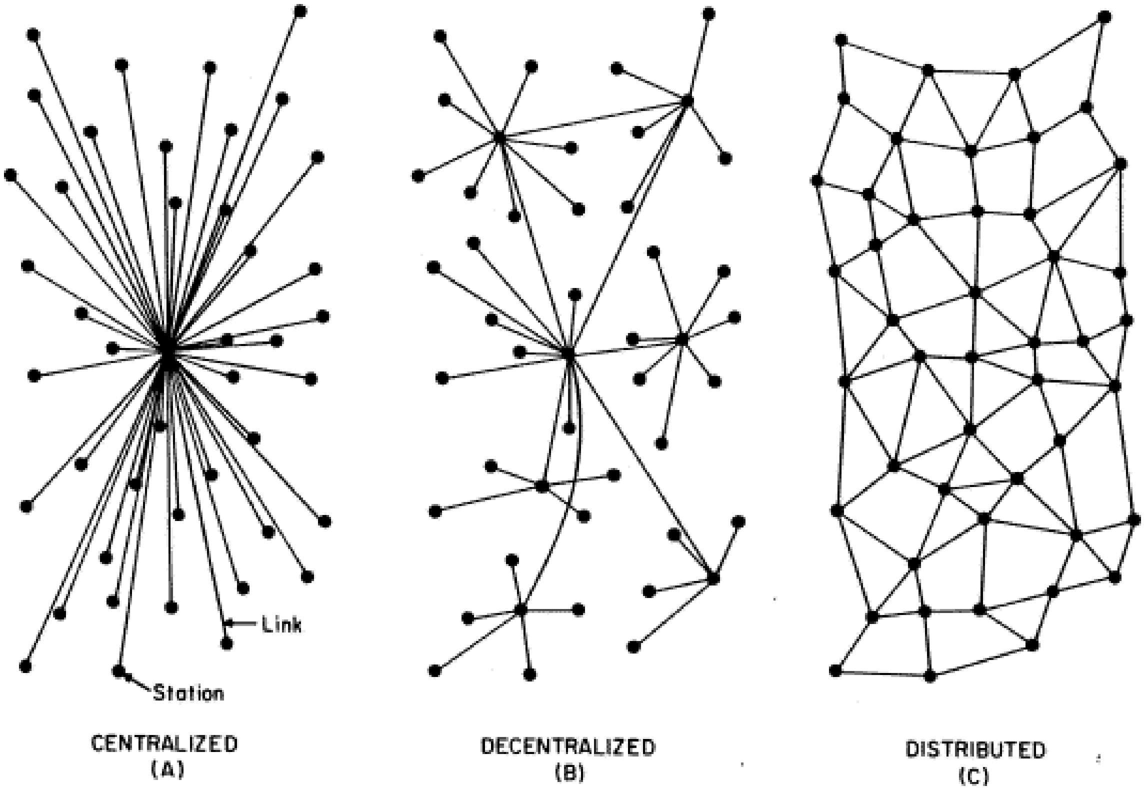


FIG. I — Centralized, Decentralized and Distributed Networks

The pioneering research of Paul Baran in the 1960s, who envisioned a communications network that would survive a major enemy attacked. The sketch shows three different network topologies described in his RAND Memorandum, "On Distributed Communications: 1. Introduction to Distributed Communications Network" (August 1964). The distributed network structured offered the best survivability

The First Implementation

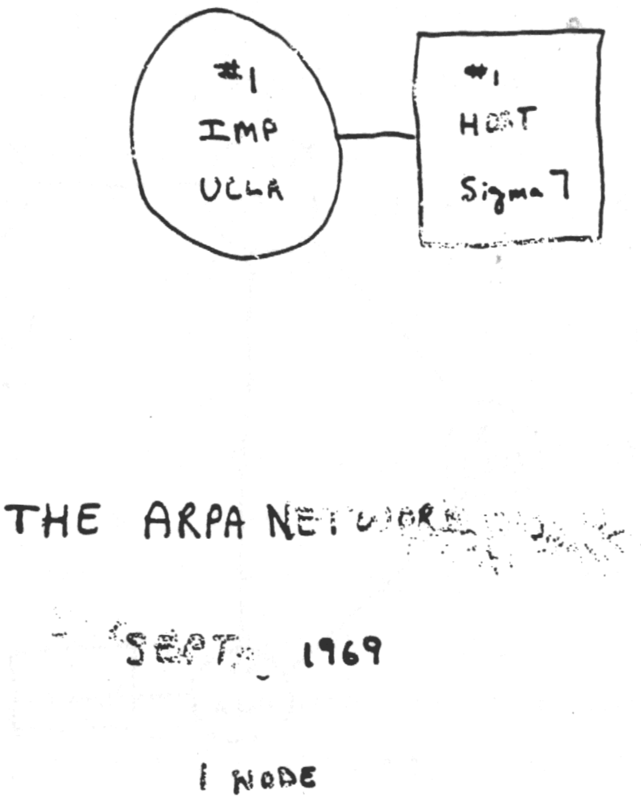
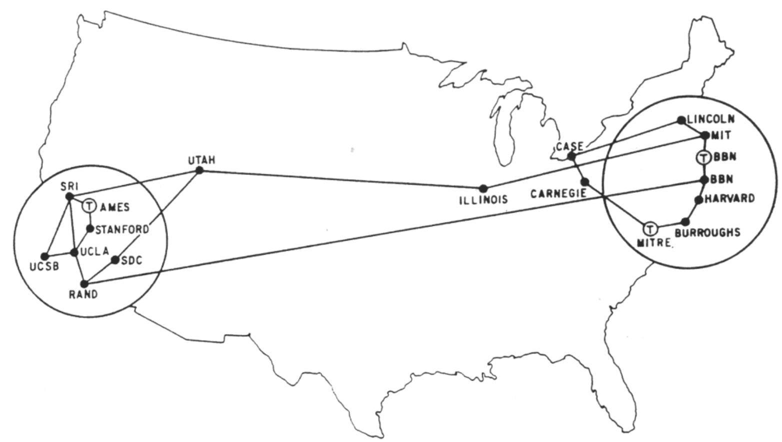


FIGURE 6.1 Drawing of September 1969
(Courtesy of Alex McKenzie)

The first node on ARPANET at University California Los Angeles (UCLA) on the 2nd of September 1969



MAP 4 September 1971

1971

Symbolic Addressing Helps Humans

- **To simplify human use, the authority also assigns a symbolic address**
  - ✔ format is *org.type*,
    where *org* denotes the organization
    and *type* denotes the type of organization

    WSU: `wsu.edu`

  - ✔ this name is called a "*domain*"

com	commercial
edu	educational
gov	governmental
mil	military
net	net resources
org	non-profit
us, fr, …	country code

- **The organization extends the address to name hosts within its network**

WSU:

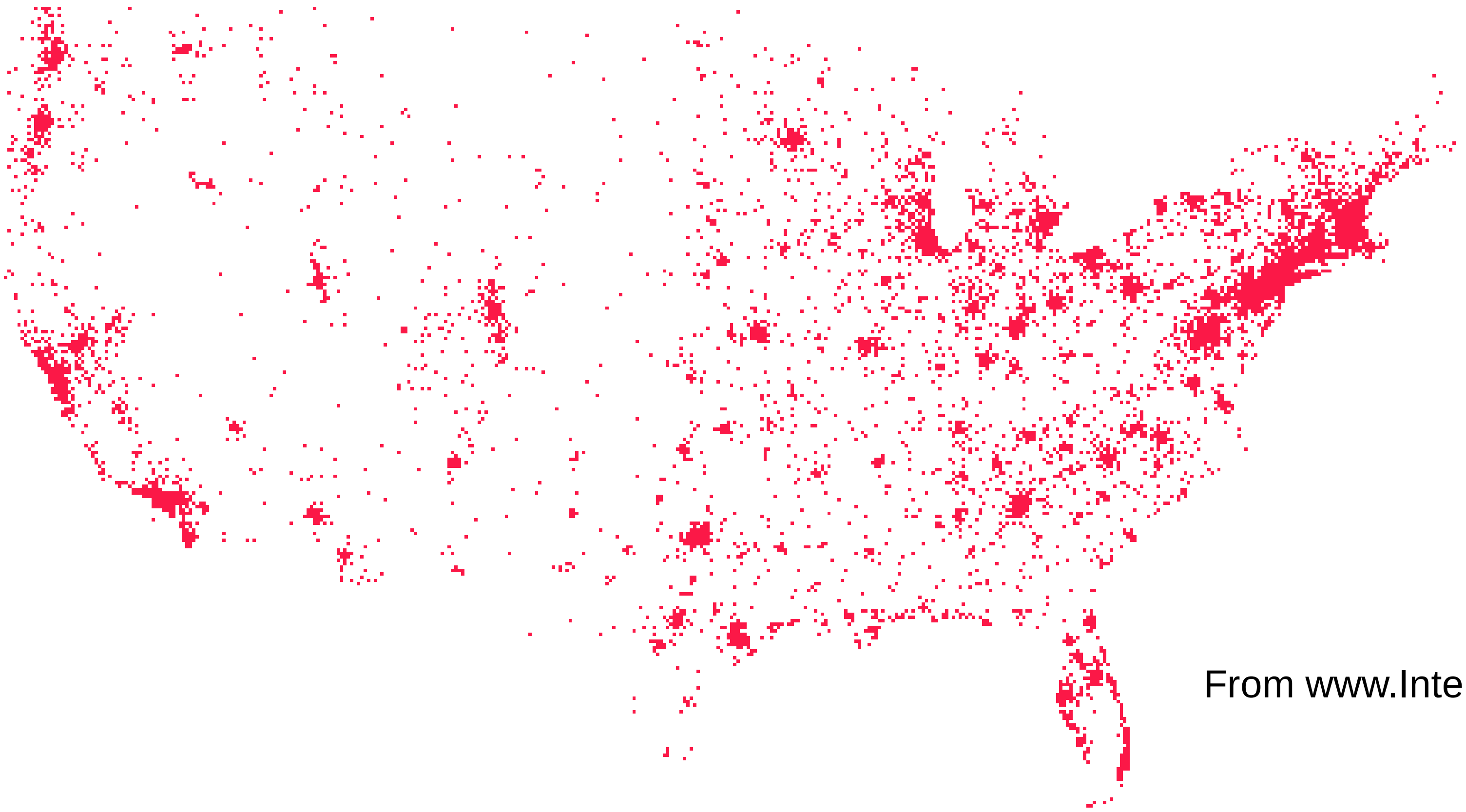
```
vancouver.wsu.edu
encs.vancouver.wsu.edu
lx.encs.vancouver.wsu.edu
```

"subdomains" are defined (~ subdirectories) down to hosts (~ files). Also, hosts may have multiple names (~inodes).

- **Symbolic addresses are mapped to domain addresses by a distributed technique called Domain Name Services (DNS)**
  - ✔ look in a local table (on Unix = /etc/hosts) for a match
  - ✔ if none, ask a DNS server to find one of the domain's servers → i.e. follow ".." up & then go down
  - ✔ then, ask that server to completely resolve the symbolic address

There Are Many Domain Names

- **Registration location of Internet domain names in the USA:**

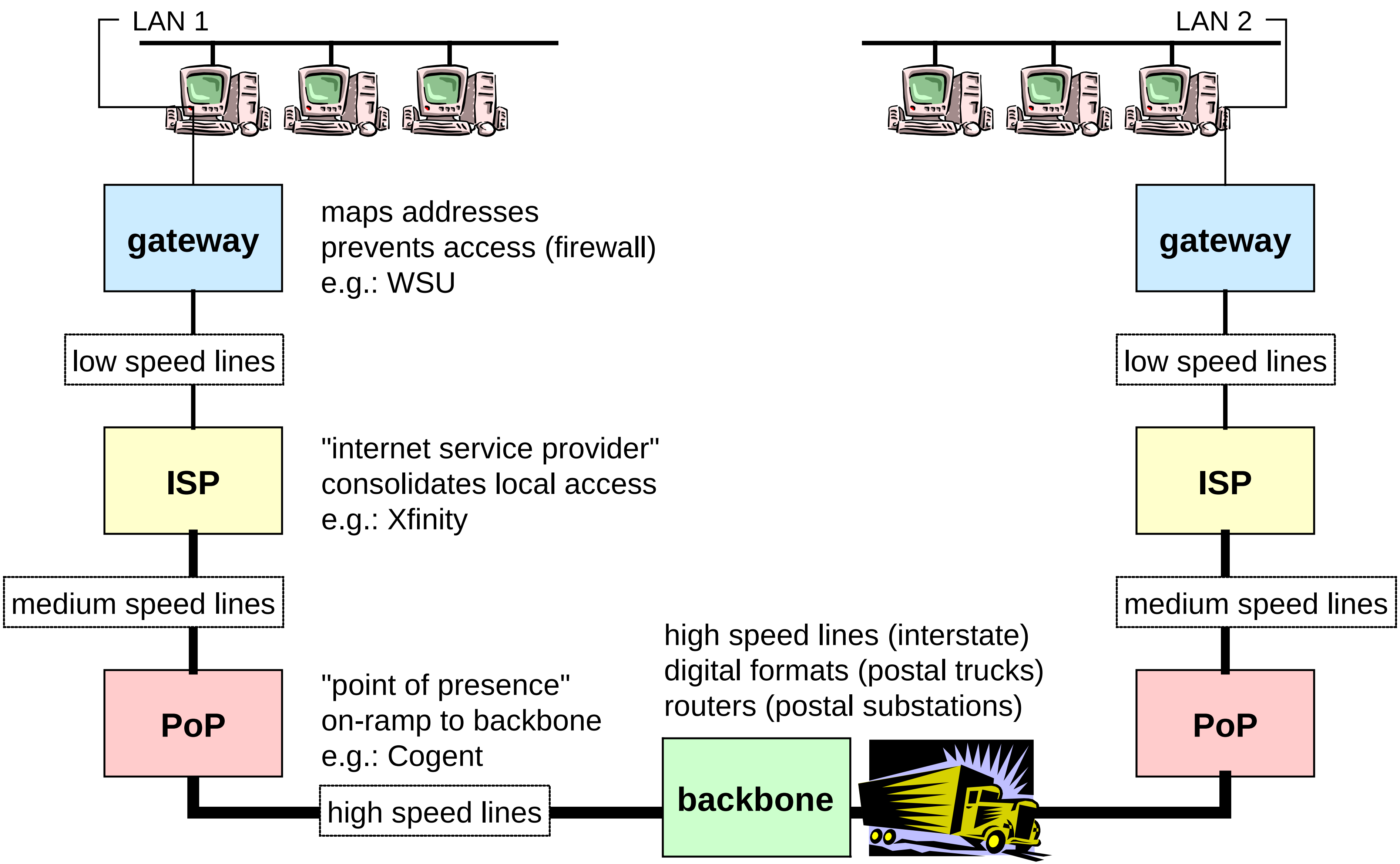


From www.Internet.Org

- **... and growing rapidly worldwide!**

How Are the Networks Connected?*

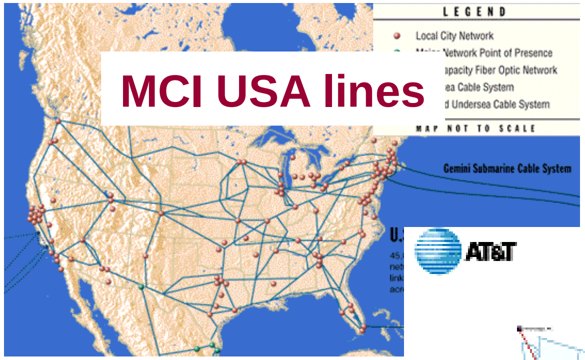
* picture shows a typical situation



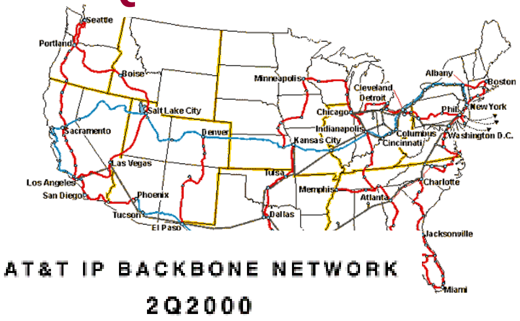
The Backbone Makes the Internet Global

- Many providers, who cooperate in moving the packets
- Your packet may go a round about way, depending on traffic
- Most backbone lines carry both data & voice

MCI USA lines



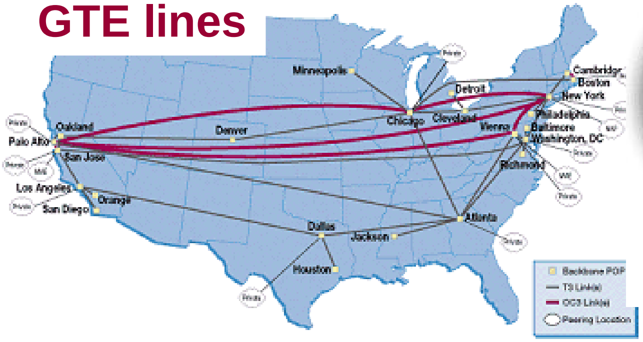
Qwest



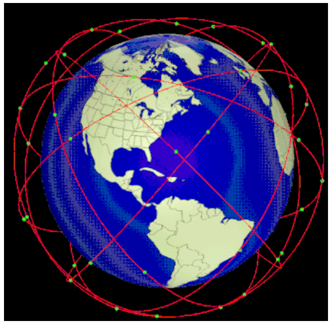
AT&T

IBM global lines

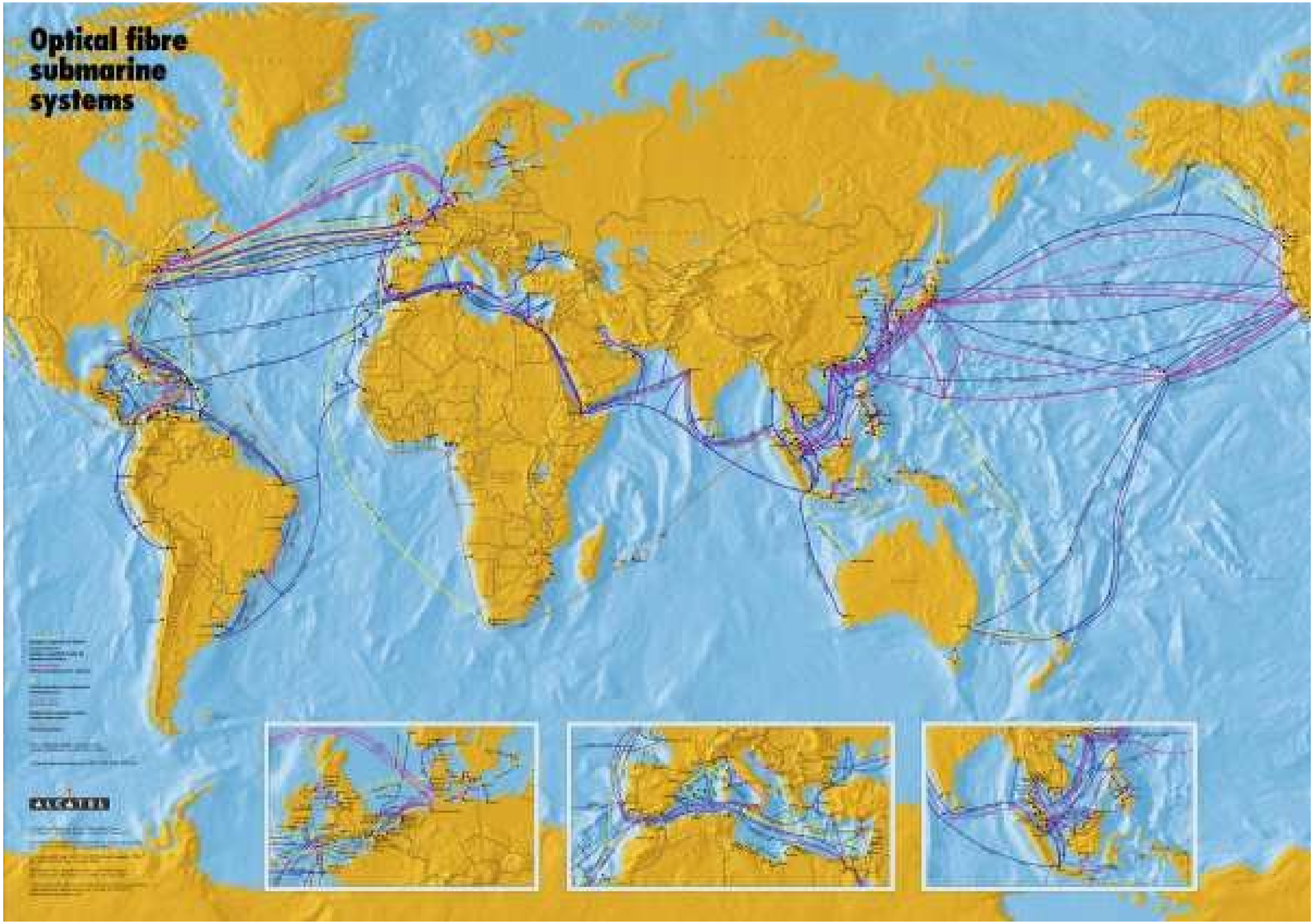
GTE lines



Teledesic satellites

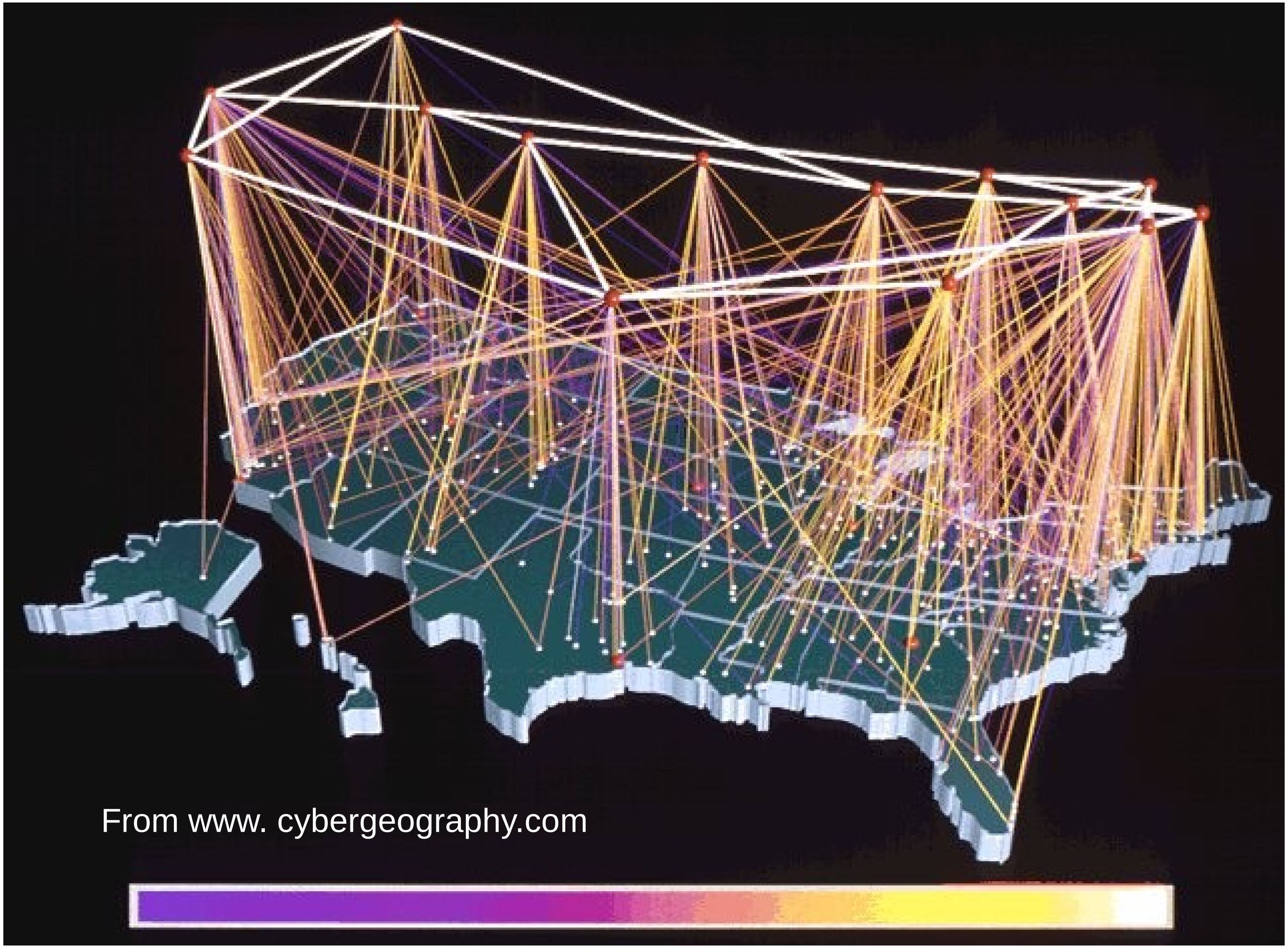


Submarine Routes Connect the World

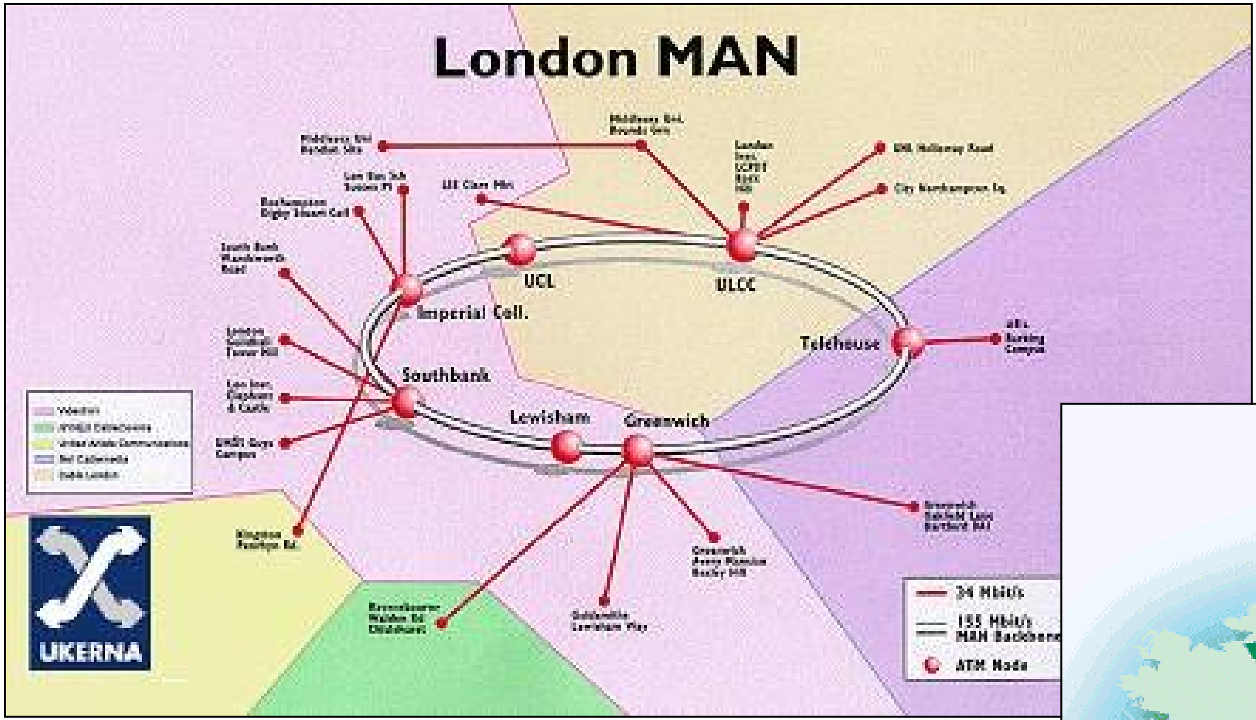


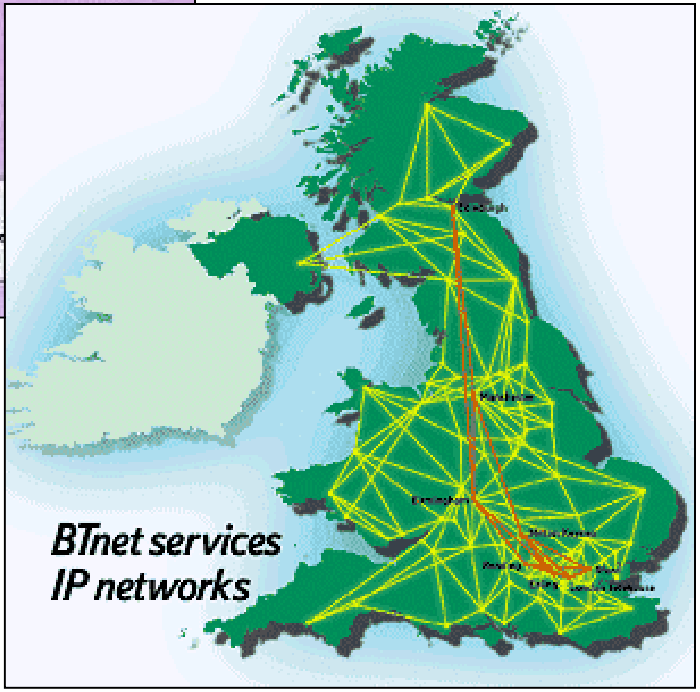
Alcatel

Bandwidth Map of USA

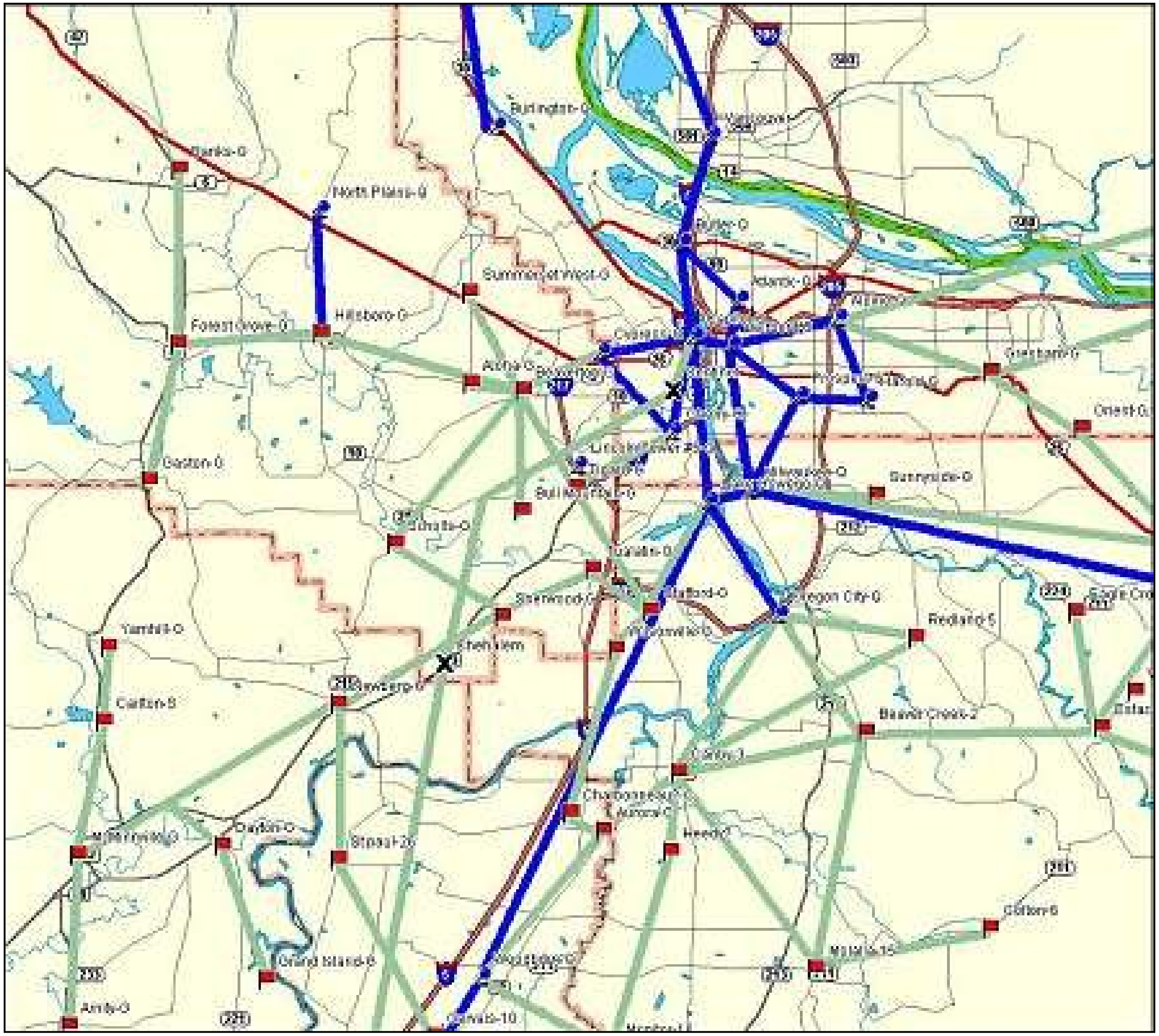


Major Cities Have MAN's





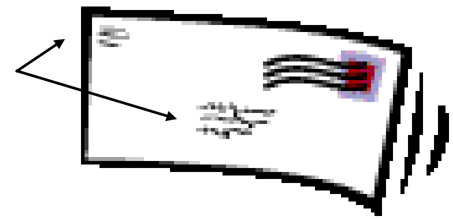
Portland-Vancouver Fiber Optic Map



Oregon Ecomonic & Community Development Department

How is Data Delivered?

- **Hosts break messages up into structured chunks (*packets*)**
  - ✔ the packets are like envelopes
  - ✔ the outside has addresses, which are read during delivery
  - ✔ the inside has data, which is not read during delivery



- **The IPv4 packet format is fixed:**

0	4	8	16	19	24 … 31
version	hlen	service type	total length		
identification			flags	fragment offset	
time to live		protocol	header checksum		
source IP address					
destination IP address					
IP options					padding
data					
...					

these are called "datagrams", and are the basic unit of TCP/IP transfer

- **The IP delivery service guarantees are <u>remarkably weak</u>:**
  - ✔ packets shouldn't be longer than about 1500 characters
  - ✔ packets may be arbitrarily delayed
  - ✔ packets may die and never arrive
  - ✔ packets may arrive out of sequence
  - ✔ packet data may get corrupted

each packet is on it's own, so the service is called "connectionless"

Ping Let's You Watch Packets Mov

- **The Ping program sends a packet to a host which then echoes it back**

```
CS360> ping neon.vancouver.wsu.edu
Pinging neon.vancouver.wsu.edu [199.237.80.9] with 32 bytes of data:
Reply from 199.237.80.9: bytes=32 time=267ms TTL=49
Reply from 199.237.80.9: bytes=32 time=269ms TTL=49
Reply from 199.237.80.9: bytes=32 time=241ms TTL=49
Reply from 199.237.80.9: bytes=32 time=211ms TTL=49
Ping statistics for 199.237.80.9:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 211ms, Maximum =  269ms, Average =  247ms

CS360> ping utdallas.edu
Pinging utdallas.edu [129.110.10.1] with 32 bytes of data:
Reply from 129.110.10.1: bytes=32 time=354ms TTL=238
Request timed out.
Reply from 129.110.10.1: bytes=32 time=347ms TTL=238
Reply from 129.110.10.1: bytes=32 time=354ms TTL=238
Ping statistics for 129.110.10.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 347ms, Maximum =  354ms, Average =  263ms
```

The Backbone Works Hard to Find a Route

3 round trip times

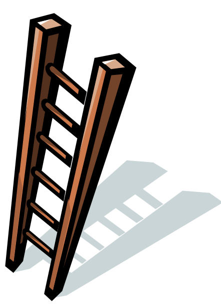
```
% traceroute utdallas.edu
 1  pul1-gw (199.237.80.254)  1 ms  1 ms  1 ms
 2  murr-rtr-vanc.net.wsu.edu (192.220.207.178)  17 ms  17 ms  18 ms
 3  subnet-e8-gw.net.wsu.edu (134.121.239.254)  16 ms  17 ms  16 ms
 4  wsu-gw.net.wsu.edu (134.121.2.10)  16 ms  16 ms  16 ms
 5  seauw1-H4-1-0.wa-k20.net (205.175.104.17)  39 ms  56 ms  38 ms
 6  seauwbr1-FE1-0-0.wa-k20.net (198.32.171.21)  44 ms  42 ms  69 ms
 7  hssi8-0.sea-br1.nw.verio.net (204.200.241.2)  56 ms  60 ms  55 ms
 8  fe2-1.sea-br2.nw.verio.net (204.200.8.6)  121 ms  35 ms  40 ms
 9  hssi3-0.wes-br4.nw.verio.net (198.104.194.50)  34 ms  37 ms  45 ms
10  e0.sea0.verio.net (204.202.46.20)  59 ms  63 ms  32 ms
11  sea0.sjc0.verio.net (129.250.2.17)  68 ms  57 ms  71 ms
12  sjc0.nuq0.verio.net (129.250.3.98)  63 ms  85 ms  81 ms
13  sl-w1-mae-0-0-0-100M.sprintlink.net (198.32.136.11)  92 ms  80 ms  99 ms
14  sl-bb2-stk-2-0-T3.sprintlink.net (144.228.10.45)  109 ms  81 ms  85 ms
15  sl-bb22-stk-3-2.sprintlink.net (144.232.4.9)  97 ms  86 ms  90 ms
16  sl-bb10-fw-0-0.sprintlink.net (144.232.8.69)  165 ms  189 ms  210 ms
17  sl-gw11-fw-0-0-0.sprintlink.net (144.232.11.54)  168 ms  156 ms  164 ms
18  sl-uoftx-1-0-0-T3.sprintlink.net (144.228.135.34)  138 ms  167 ms  171 ms
19  utx1-h4-0.tx-bb.net (192.12.10.13)  154 ms  157 ms  168 ms
20  ut5-fe1-0-0.tx-bb.net (192.12.10.29)  144 ms  170 ms *
21  utd6-h5-0-0.the.net (129.117.24.18)  185 ms  221 ms  194 ms
22  129.117.39.226 (129.117.39.226)  198 ms  163 ms  174 ms
23  shot.utdallas.edu (129.110.5.4)  158 ms  155 ms  157 ms
24  poteidaia.utdallas.edu (129.110.10.1)  163 ms * *
```

Higher Level Protocols Are Built Upon IP

- **The IP protocol provides the most basic service**
  - ✔ similar to Unix philosophy: build simplest sufficient base possible
  - ✔ ... but, hard to write applications using unreliable delivery

- **So, <u>other protocols are defined above IP</u>:**
  - ✔ TCP - Transmission Control Protocol (reliable transport service)
  - ✔ FTP - File Transfer Protocol (file transfer using TCP)
  - ✔ TELNET - remote login protocol
  - ✔ SMTP - Simple Mail Transfer Protocol
  - ✔ SNMP - Simple Network Management Protocol (find and query hosts)
  - ✔ DHCP - Dynamic Host Configuration Protocol (assign IP address at boot)
  - ✔ NTP - Network Time Protocol
  - ✔ UDP - User Datagram Protocol (almost raw IP packets)

FTP ...
TCP
IP



the Internet Engineering Task Force (IETF) administers protocol proposals & analysis; the Internet Architecture Board (IAB) approves protocols and allocates addresses

TCP Provides Robust Delivery

- **TCP handles:**
  1. decomposition of messages into datagrams
  2. assurance that all arrive and are in good condition (via ack/nak & checksums)
  3. assembly back into proper order (via sequence numbers)

- **Most applications and many other protocols use TCP:**

important to remember!

	applications or other protocols
TCP:	**reliable virtual circuit transport service**
IP:	**unreliable connectionless packet delivery**

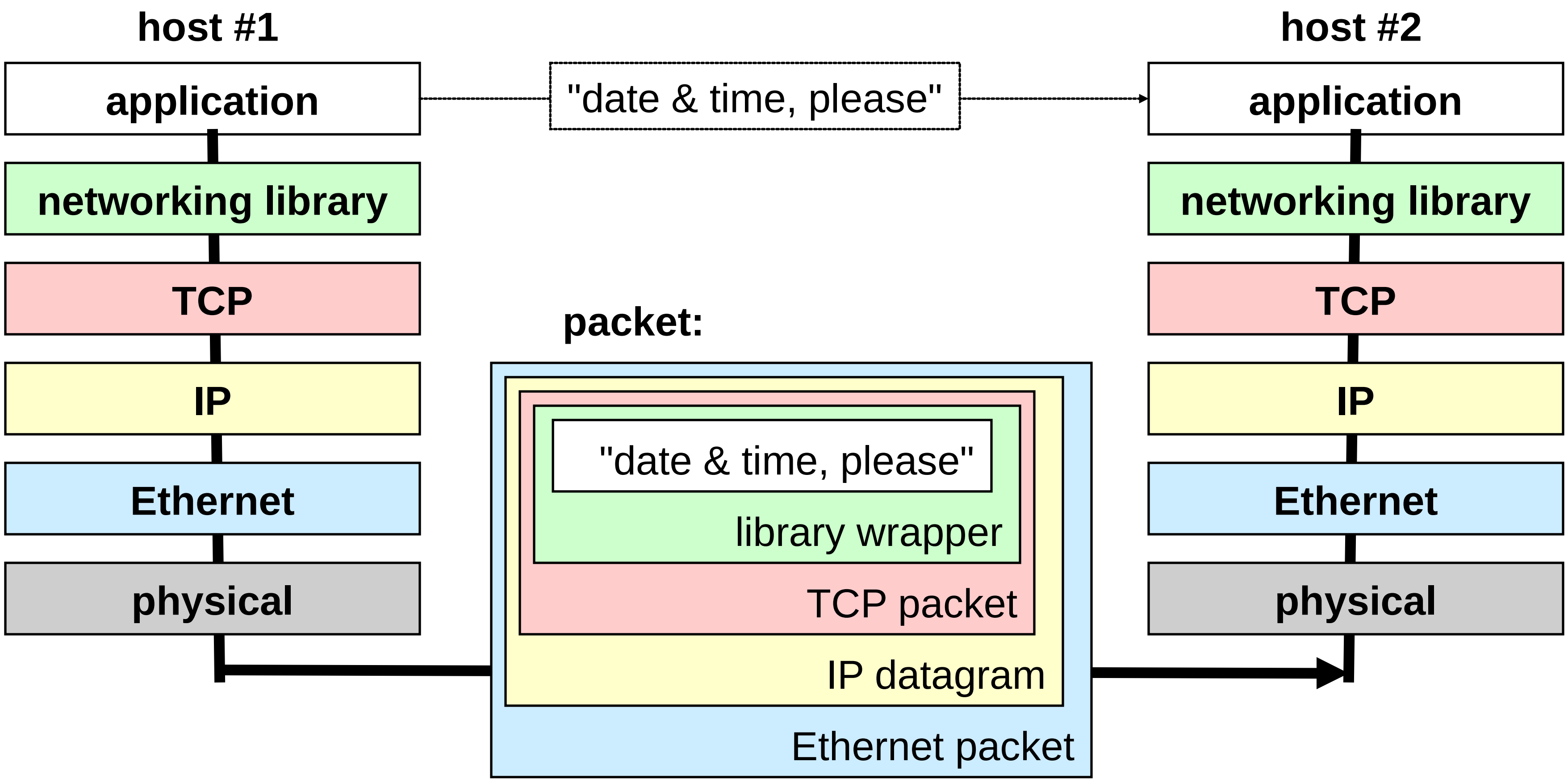
- **Notes:**
  - ✔ as a shorthand, the whole set of Internet protocols is called "**TCP/IP**"

Unix Provides Useful Shell Tools

command	action
whois	display NIC information about a host
ping	time packets from your host to another
traceroute	display path to another host
hostname	display your host's name
nslookup	map symbolic name to numeric address
netstat	display which clients are connected
ftp	begin a file transfer session
telnet	begin a remote login session
mail	send electronic mail
rcp	copy files to another Unix host
rlogin	remotely login to another Unix host
rsh	run a shell command on another Unix host
rusers	display user accounts on a host
rwho	display who is logged in on a host
rwall	send a message to everyone on a host

- **Read the man pages for details and perform experiments**

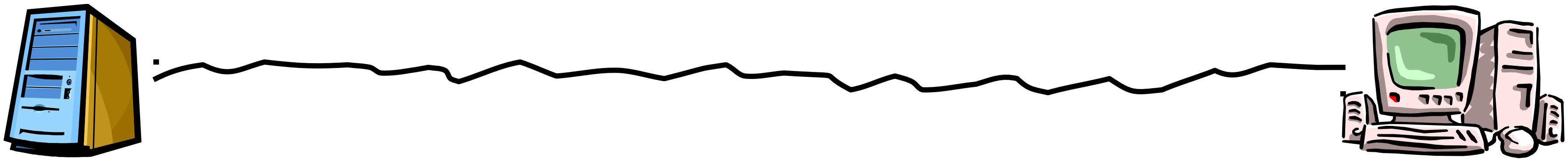
Protocols Use Other Protocols



- **Layers use other layers**
- **Each layer has it's own protocol**

Note: The Open Systems Interconnection (OSI) model defines 7 layers, from app down to physical

Servers Use Ports to Announce Services



server host, who provides services

client host, who uses services

service	port
echo	7
discard	9
systat	11
daytime	13
ftp	21
telnet	23
smtp	25
http	80

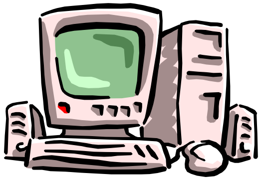
- **<u>Ports</u> are how a server presents its services**
- **Some standard ports are listed below**

The ports created at boot are in /etc/services

E.g.: "oxygen" has 135 ports active

- **The client connects to a specific port**
- **On the server, you can open a new port at any time**

Client/Server Programming Model



client host, who uses services

server host, who provides services

"I'm available!" 1. Create a port

2. Attempt connection to host/port "Will you help me?"

"Yes, I will" 3. Accept connection request

4. Use the service:

4. Fork a child process:

Send message "Please, do ..." Receive message

Receive message "Answer is ..." Send message

Repeat until done

Continue or exit

Repeat until done

Exit

5. Begin again at step 1 (parent)