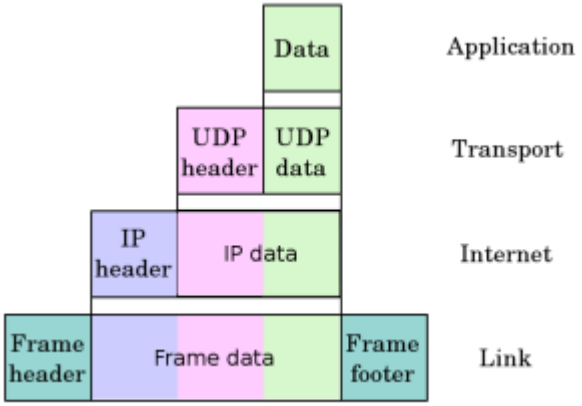
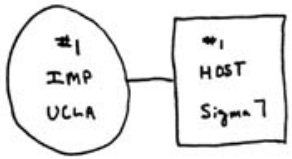
# Network layers, attacks and history

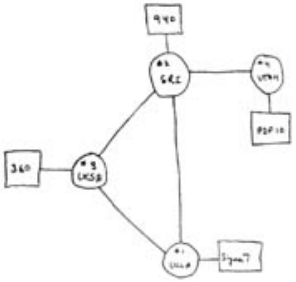



THE ARPA NETWORK

SEPT 1969

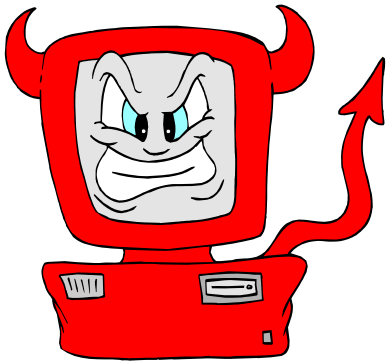
1 NODE



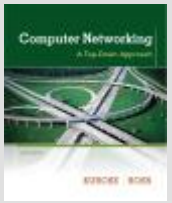
THE ARPA NETWORK

DEC 1969

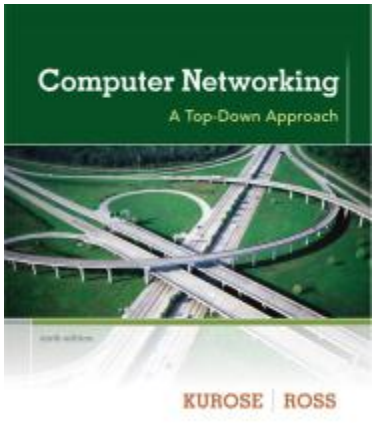
4 NODES



*Computer Networking: A Top Down Approach*
6th edition
Jim Kurose, Keith Ross
Addison-Wesley



# Overview




- Chapter 1: Introduction
  - Quick overview of field
  - Learn some terminology
- Network layers
  - How we break up a very complicated system
- Attacking networks
- History

# Protocol layers

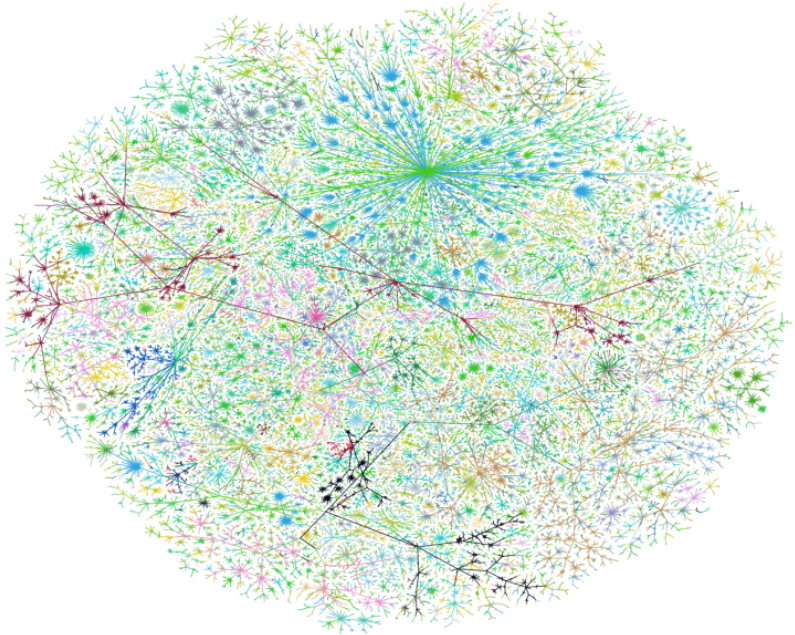
*Networks are complex, with many "pieces":*

- 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?



# 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?
  - Duplicated functionality in layers
  - Sometimes layers may really need to "peek" at information in another layer

# Letters to the Editor

## Go To Statement Considered Harmful

Key Words and Phrases: go to statement, jump instruction, branch instruction, conditional clause, alternative clause, repetitive clause, program intelligibility, program sequencing
*CR* Categories: 4.22, 5.23, 5.24

EDITOR:

For a number of years I have been familiar with the observation that the quality of programmers is a decreasing function of the density of **go to** statements in the programs they produce. More recently I discovered why the use of the **go to** statement has such disastrous effects, and I became convinced that the **go to** statement should be abolished from all "higher level" programming languages (i.e. everything except, perhaps, plain machine code). At that time I did not attach too much importance to this discovery; I now submit my considerations for publication because in very recent discussions in which the subject turned up, I have been urged to do so.

dynamic progress is only characterized when we also give to which call of the procedure we refer. With the inclusion of procedures we can characterize the progress of the process via a sequence of textual indices, the length of this sequence being equal to the dynamic depth of procedure calling.

Let us now consider repetition clauses (like, **while** *B* **repeat** *A* or **repeat** *A* **until** *B*). Logically speaking, such clauses are now superfluous, because we can express repetition with the aid of recursive procedures. For reasons of realism I don't wish to exclude them: on the one hand, repetition clauses can be implemented quite comfortably with present day finite equipment; on the other hand, the reasoning pattern known as "induction" makes us well equipped to retain our intellectual grasp on the processes generated by repetition clauses. With the inclusion of the repetition clauses textual indices are no longer sufficient to describe the dynamic progress of the process. With each entry into a repetition clause, however, we can associate a so-called "dynamic index," inexorably counting the ordinal number of the

Edsger Dijkstra
March 1968 Communications of the ACM

## "GOTO Considered Harmful" Considered Harmful

The most-noted item ever published in *Communications* was a letter from Edsger W. Dijkstra entitled "Go To Statement Considered Harmful" [1] which attempted to give a reason why the **GOTO** statement might be harmful. Although the argument was academic and unconvincing, its title seems to have become fixed in the mind of every programming manager and methodologist. Consequently, the notion that the **GOTO** is harmful is accepted almost universally, without question or doubt. To many people, "structured programming" and "**GOTO**-less programming" have become synonymous.

This has caused incalculable

## "'GOTO Considered Harmful' Considered Harmful" Considered Harmful?

I enjoyed Frank Rubin's letter ("'**GOTO** Considered Harmful' Considered Harmful," March 1987, pp. 195–196), and welcome it as an opportunity to get a discussion started. As a software engineer, I have found it interesting over the last 10 years to write programs both with and without **GOTO** statements at key points. There are cases where adding a **GOTO** as a quick exit from a deeply nested structure is convenient, and there are cases where revising to eliminate the **GOTO** actually simplifies the program.

Rubin's letter attempts to "prove" that a **GOTO** can simplify the program, but instead proves to me that his implementation language is deficient. In the first solution example the **GOTO** programmers got the answer very effectively with no wasted effort:

# On a somewhat disappointing correspondence

I did not react to Frank Rubin's letter [0], confident that all my potential comments would be made by others, but in the five letters published two months later [1], I found none of them expressed. So, I reluctantly concluded that I had better record my concerns, big and small.

(0) The problem statement refers to an N by N matrix X ; Rubin's programs refer to an n by n matrix x . In other contexts this might be considered a minor discrepancy, but I thought that by now professional programmers had learned to be more demanding on themselves and not to belittle the virtue of accuracy. I shall stick to the capital letters.

(1) Rubin still starts indexing the rows and the columns at 1 . I thought that by now professional programmers knew how much more preferable it is to let the natural numbers start at 0 . I shall start indexing at 0 .

(2) Rubin's third program fails for N=0 (in which case his second program succeeds only by accident -see below-). I thought that by now professional programmers would know the stuff the silly bugs are made of.

# Internet protocol stack

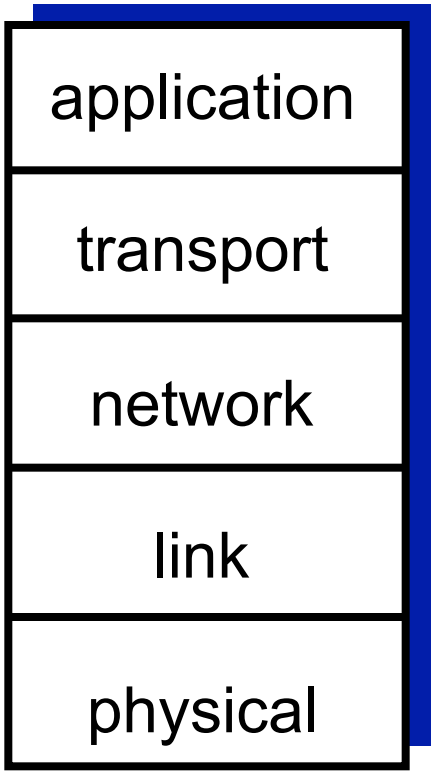
- *Application:*
  - Supporting network applications
  - e.g. FTP, SMTP, HTTP
- *Transport:*
  - Process-process data transfer
  - e.g. TCP, UDP
- *Network:*
  - Routing of datagrams from source to dest
  - e.g. IP, routing protocols
- *Link:*
  - Transfer between neighboring elements
  - e.g. Ethernet, 802.11 (WiFi), PPP
- *Physical:*
  - Bits "on the wire"

MESSAGE

SEGMENT

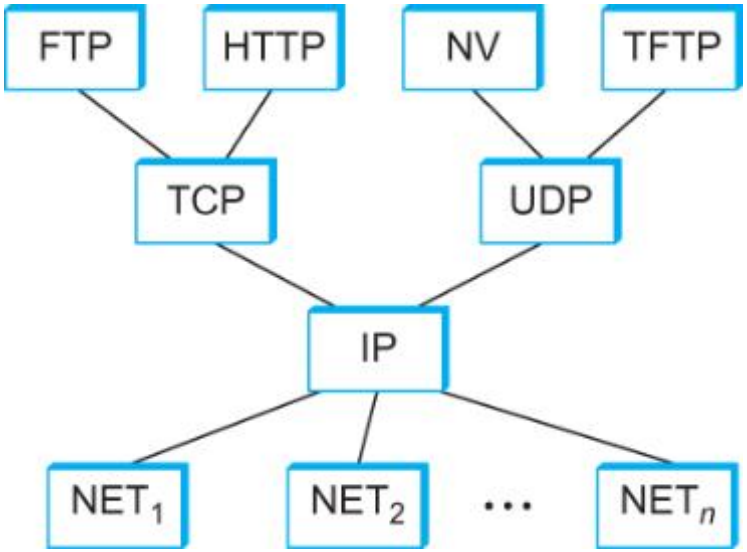
DATAGRAM

FRAME

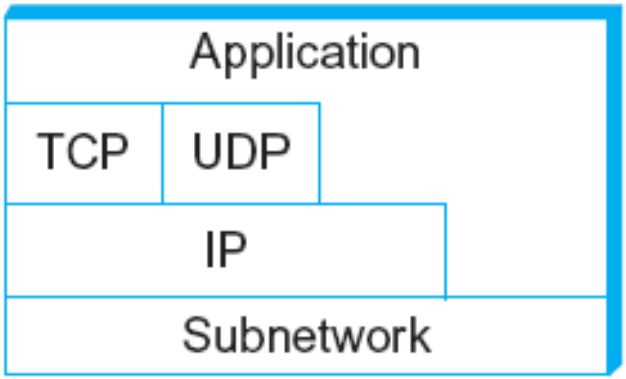


# Internet architecture

- All roads go through IP



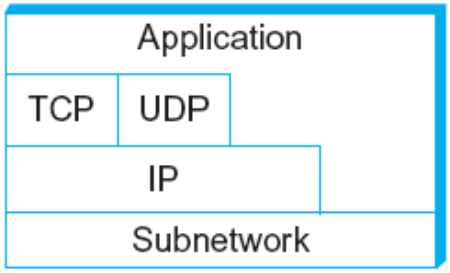

*Internet protocol graph*




*Another view of the Internet architecture. Subnetwork is often called network or link layer.*

# Internet architecture

- No strict layering
  - Application can bypass layers if need be

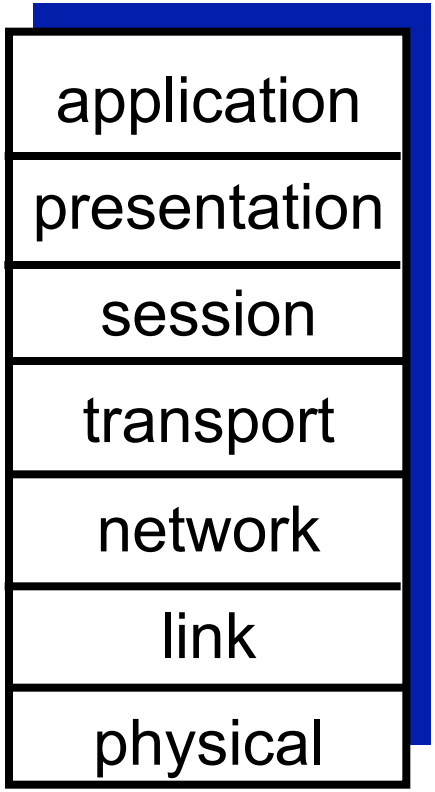



- New protocols:
  - Approval requires specification and working code
  - Internet Engineering Task Force (IETF)

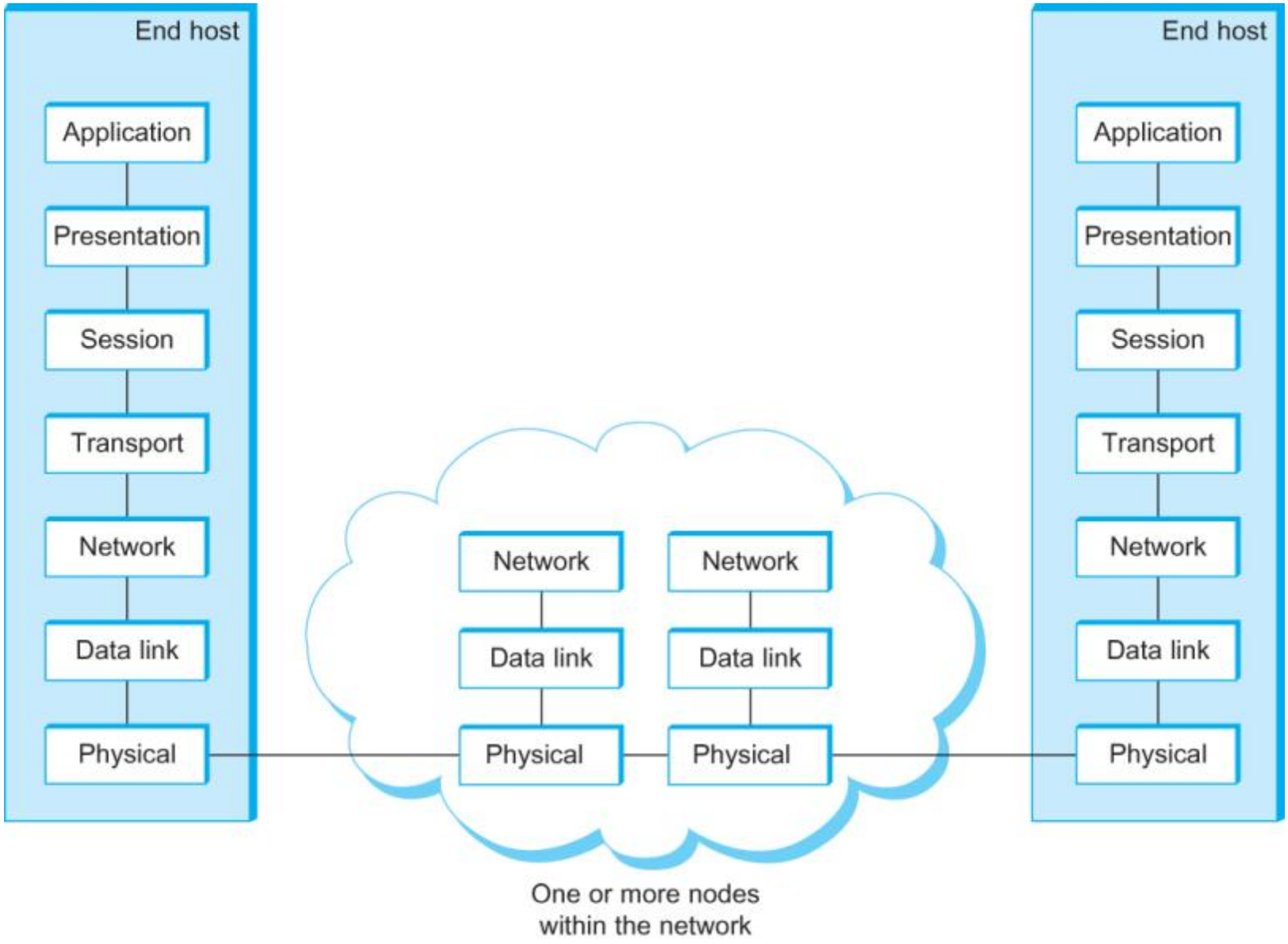
"We reject kings, presidents and voting. We believe in rough consensus and running code."
*-David Clark*

# ISO OSI reference model

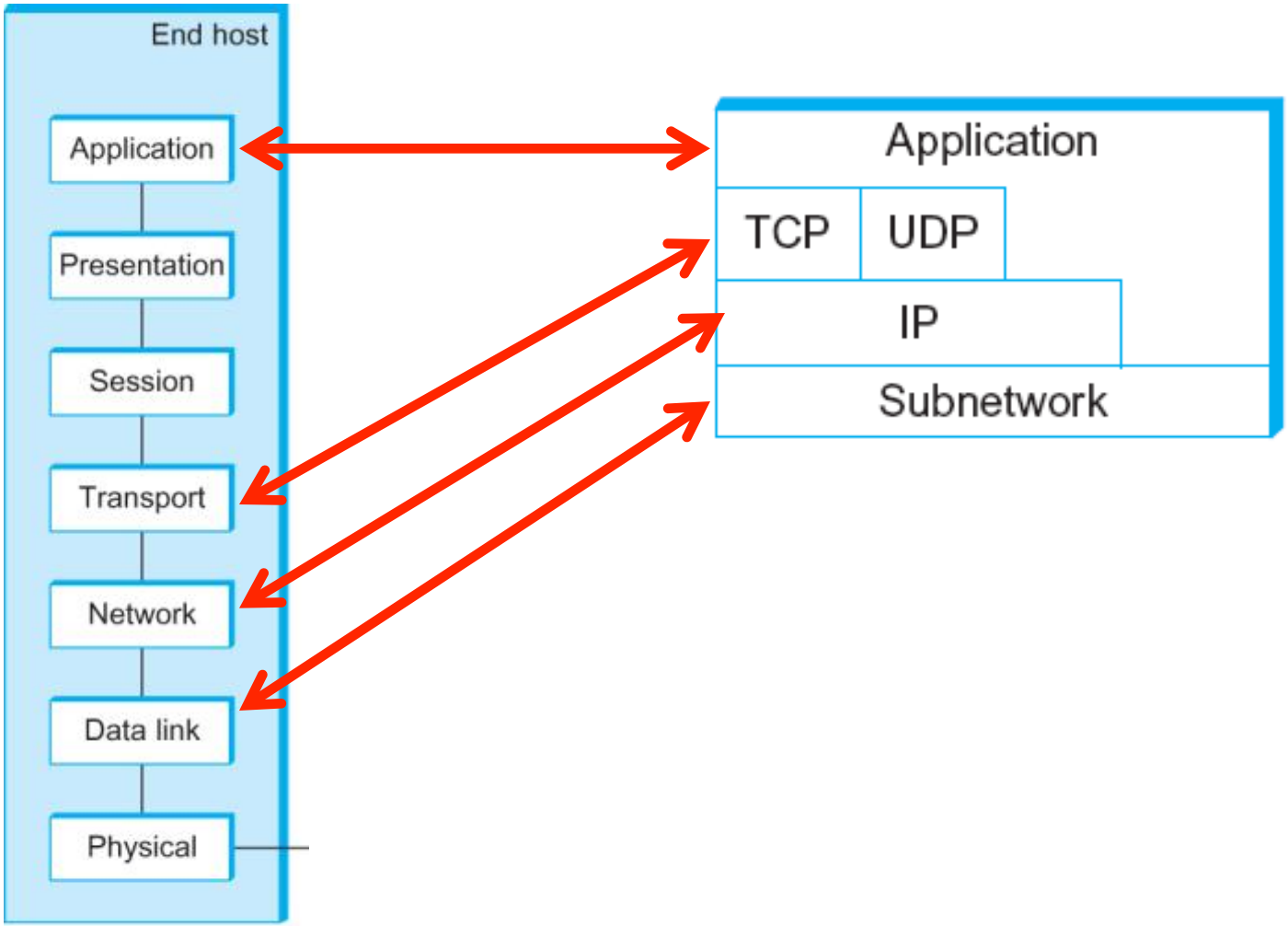
- *Presentation:*
  - Allow applications to interpret data
  - e.g., encryption, compression, machine-specific conventions
- *Session:*
  - Synchronization, checkpointing, recovery of data exchange
- Internet stack "missing" these layers!
  - These services, *if needed,* must be implemented in application



# ISO OSI reference model

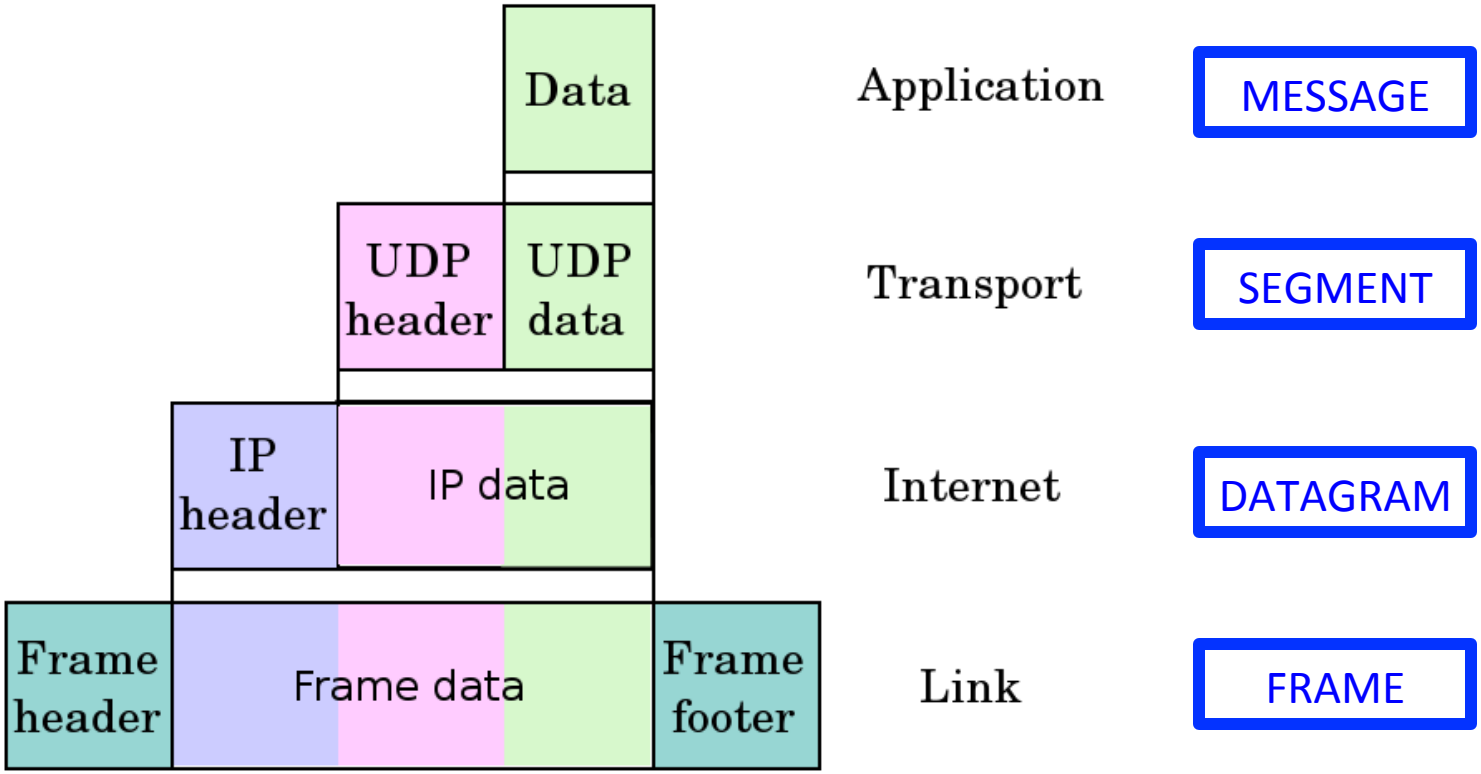


# Layers compared

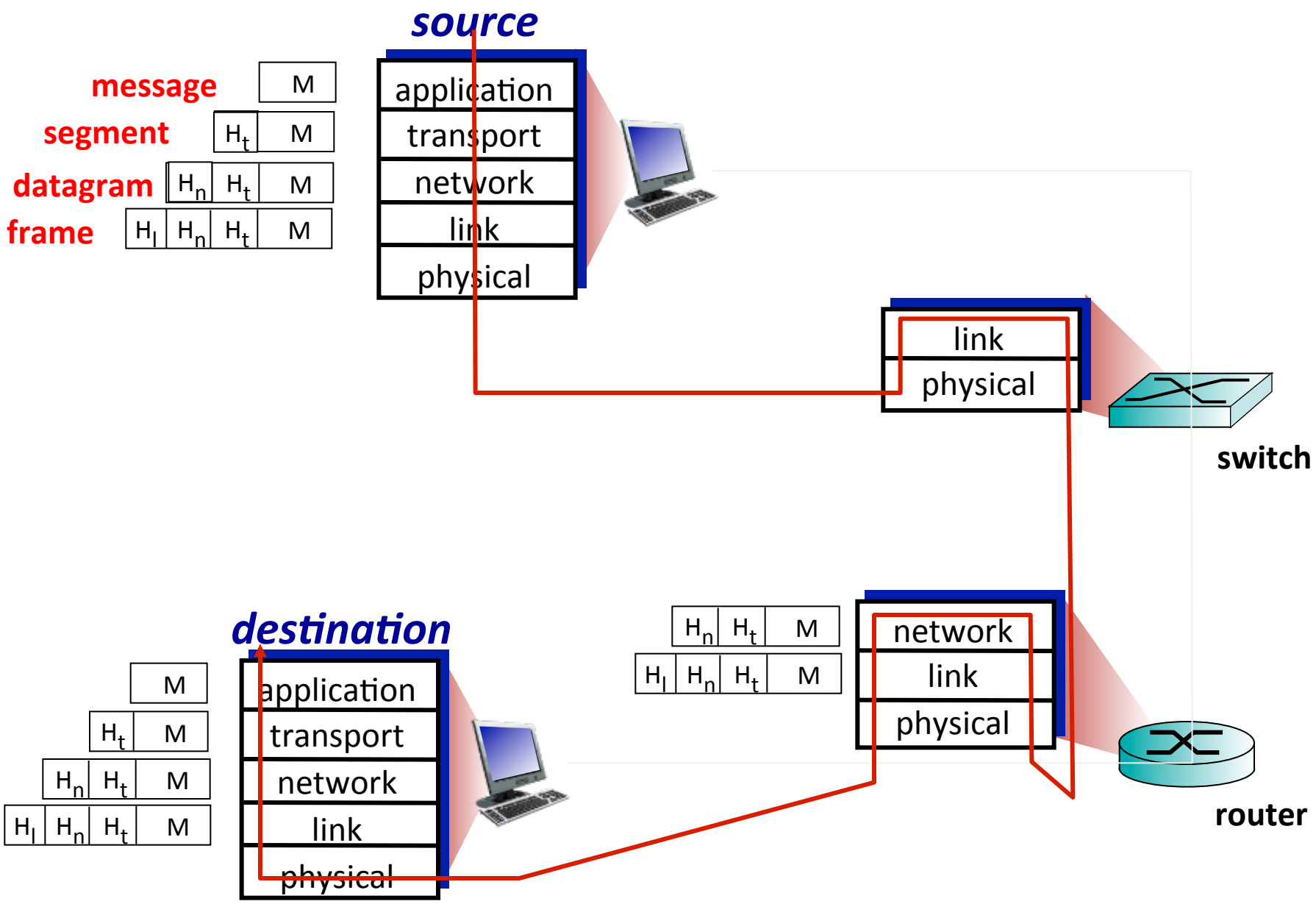


# Encapsulation

- High-level messages ***encapsulated*** in low-level messages
  - Headers/footer get added by each layer



# Encapsulation



# Network security

- Field of network security:
  - How can bad guys attack computer networks?
  - How can we defend networks against attacks?
  - How to design architectures immune to attacks?
- Internet not originally designed with (much) security in mind
  - *Original vision:* "a group of mutually trusting users attached to a transparent network" ☺
  - Internet protocol designers playing "catch-up"
  - Security considerations in all layers!

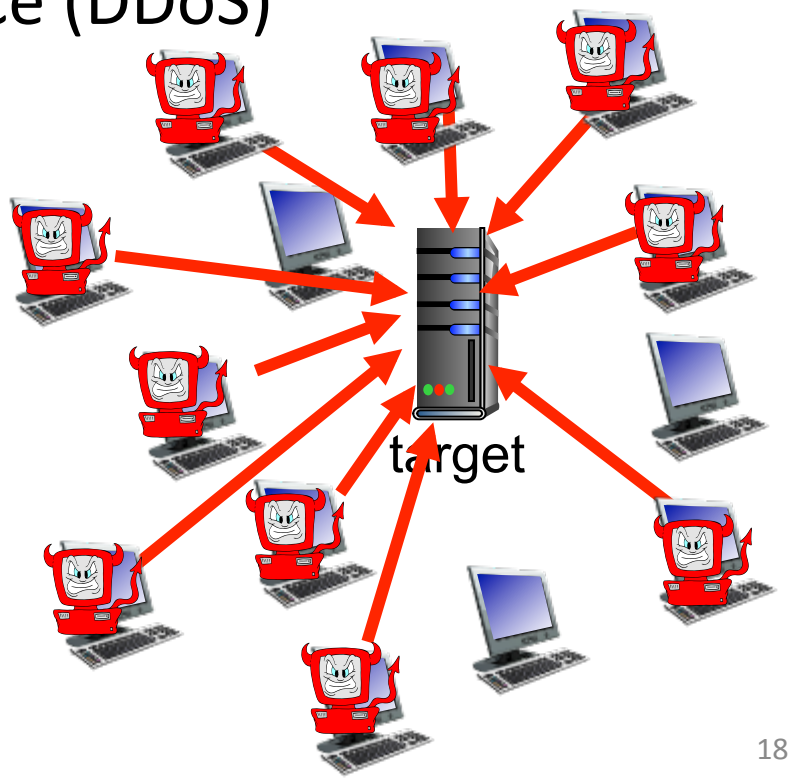
# Bad guys: malware

- Malware can get into host from:
  - *Virus:* self-replicating infection by receiving/ executing object (e.g., e-mail attachment)
  - *Worm:* self-replicating infection by passively receiving object that gets itself executed
- Spyware malware
  - Record keystrokes, web sites visited, upload info to collection site
- Infected host can be enrolled in botnet
  - Used for spam, Distributed Denial of Service (DDoS) attacks

# Bad guys: attacking servers/network

- Denial of Service (DoS) attack
  - Attackers make resources (e.g. server, bandwidth) unavailable to legitimate traffic by overwhelming resources with bogus traffic
  - Distributed Denial of Service (DDoS)

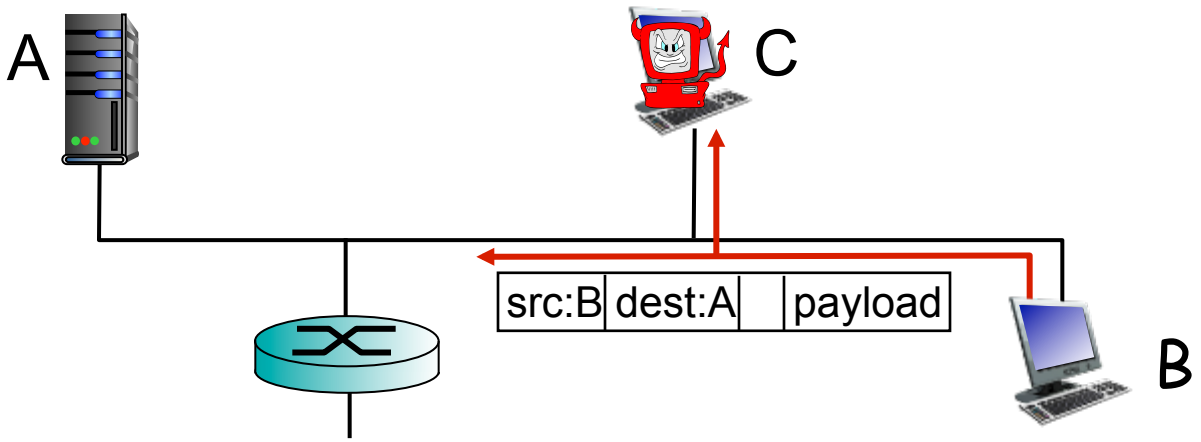
1) Select target
2) Break into hosts around the network (see botnet)
3) Send packets to target from compromised hosts



# Bad guys can sniff packets

## Packet "sniffing":

- Broadcast media (shared Ethernet, wireless)
- Promiscuous network interface
  - Reads/records all packets (e.g. including passwords!) passing by

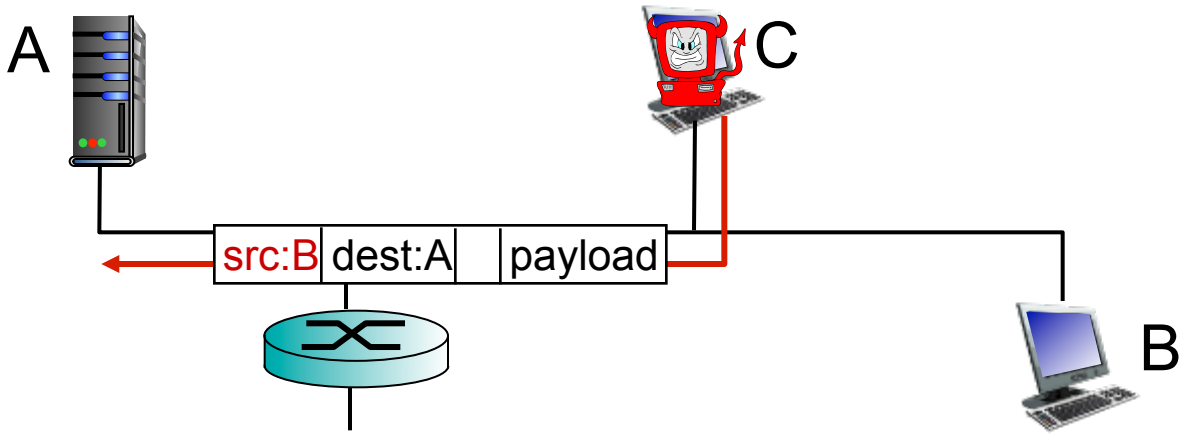



- wireshark software used for end-of-chapter labs is a (free) packet-sniffer

# Bad guys can use fake addresses

IP spoofing:

– Send packet with false source address



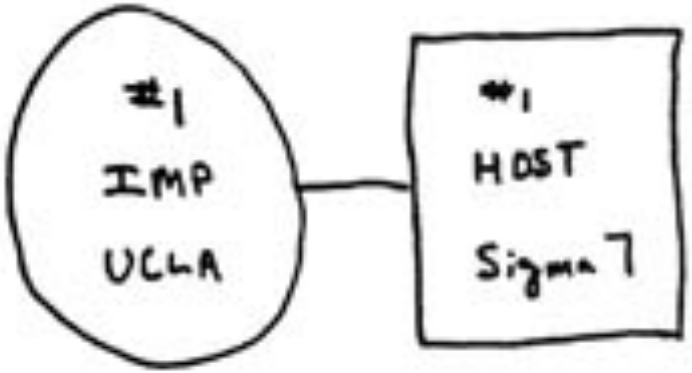

*… lots more on security (throughout, Chapter 8)*

# Internet history

## *1961-1972: Early packet-switching principles*

- **1961:**
  - Kleinrock, queueing theory shows effectiveness of packet-switching
- **1964:**
  - Baran, packet-switching in military nets
- **1967:**
  - ARPAnet conceived by Advanced Research Projects Agency
- **1969:**
  - First ARPAnet node operational
- **1972:**
  - ARPAnet public demo
  - NCP (Network Control Protocol) first host-host protocol
  - First e-mail program

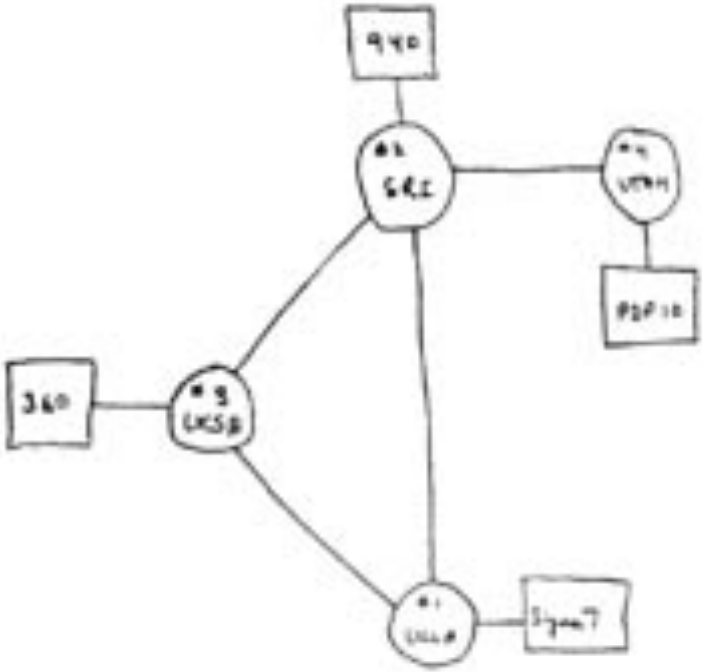
# The Internet: 1969




THE ARPA NETWORK

SEPT 1969

1 NODE

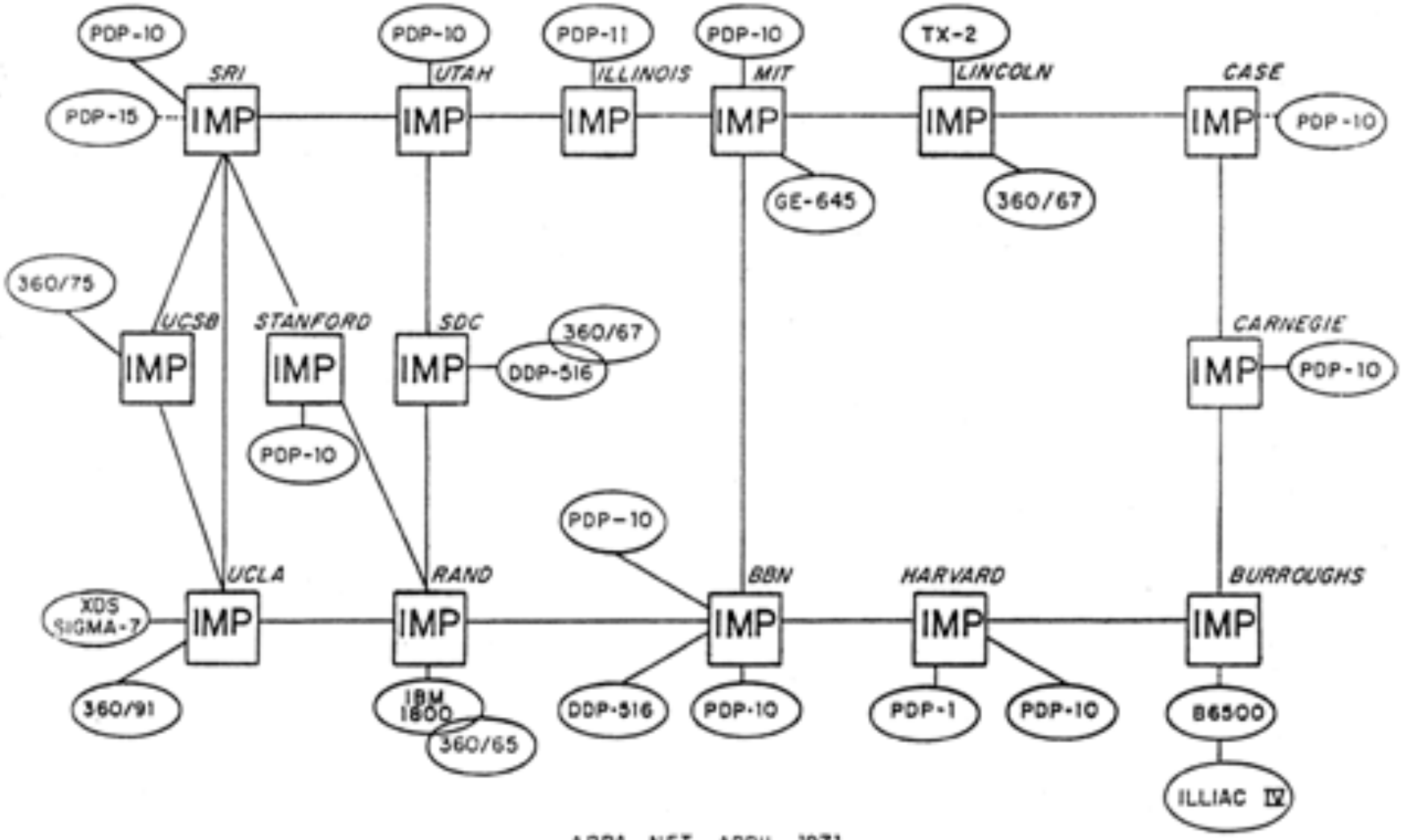



THE ARPA NETWORK

DEC 1969

4 NODES

# The Internet: 1971




ARPA NET, APRIL 1971

# Internet history

## *1972-1980: Internetworking, new and proprietary nets*

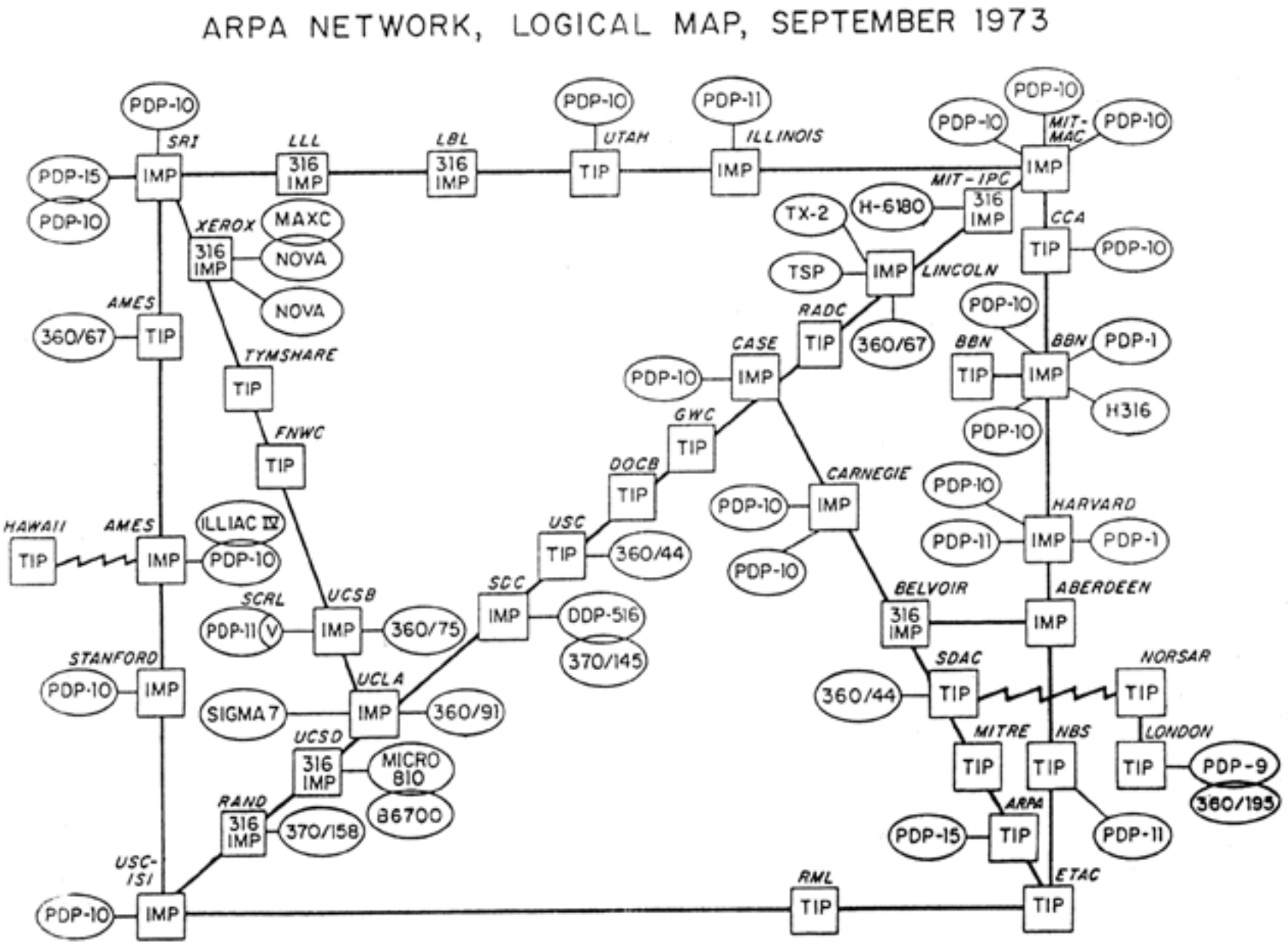
- **1970:**
  - ALOHAnet satellite network in Hawaii
- **1974:**
  - Cerf and Kahn, architecture for interconnecting networks
- **1976:**
  - Ethernet at Xerox PARC
- **Late 70's:**
  - Proprietary architectures: DECnet, SNA, XNA
  - Switching fixed length packets (ATM precursor)
- **1979**
  - ARPAnet has 200 nodes

Cerf and Kahn's internetworking principles:

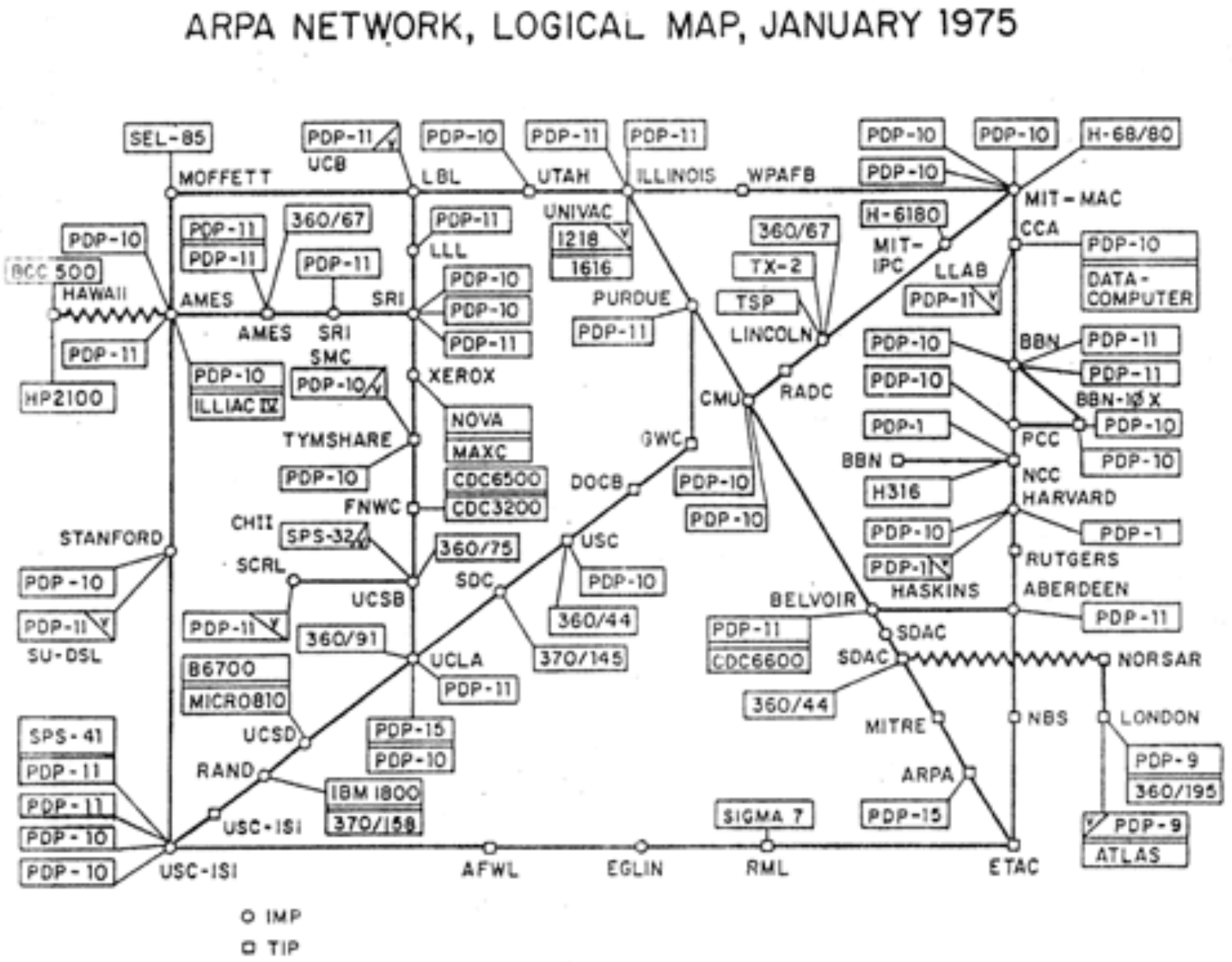
- Minimalism, autonomy - no internal changes required to interconnect networks
- Best effort service model
- Stateless routers
- Decentralized control

Define today's Internet architecture

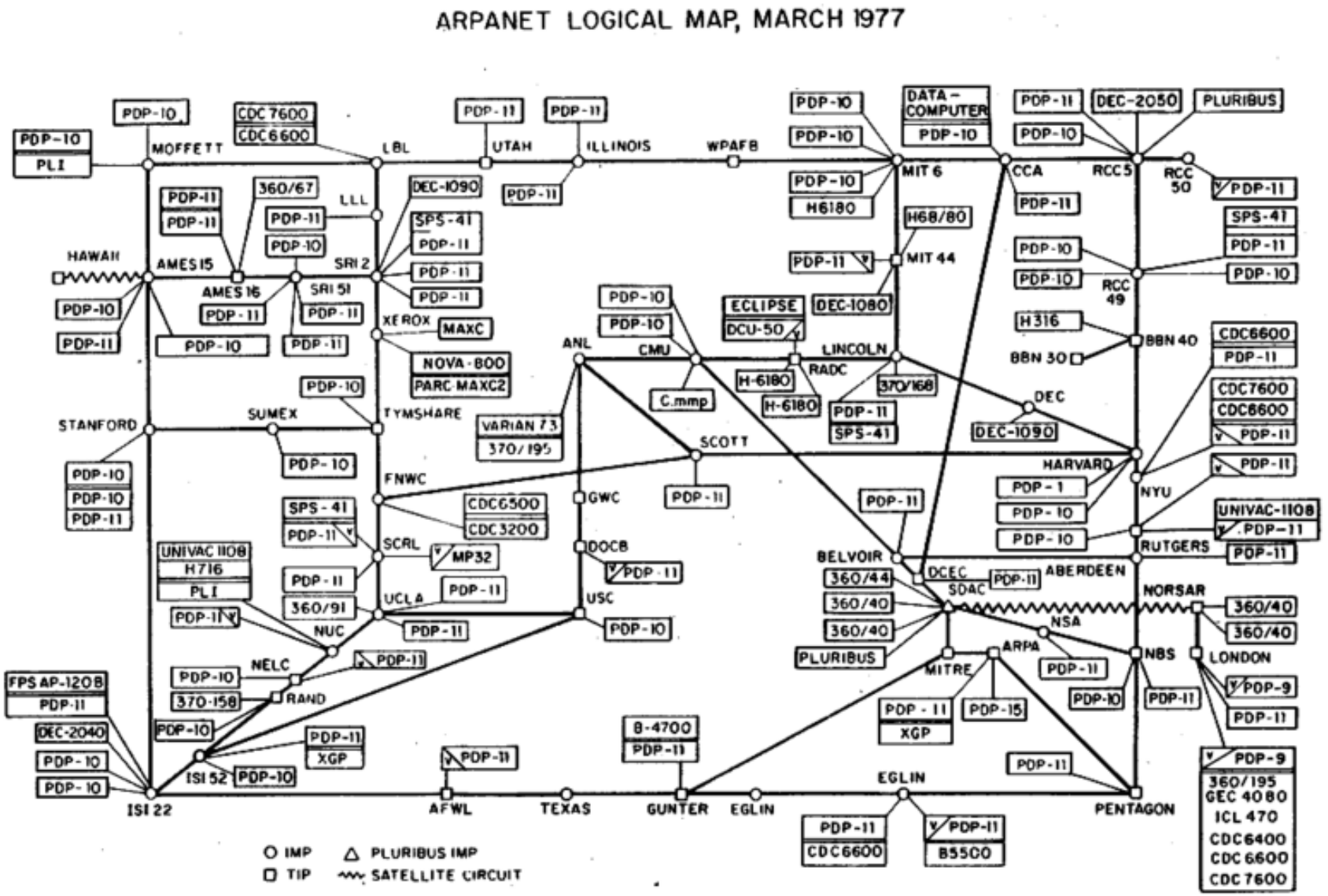
# The Internet: 1973



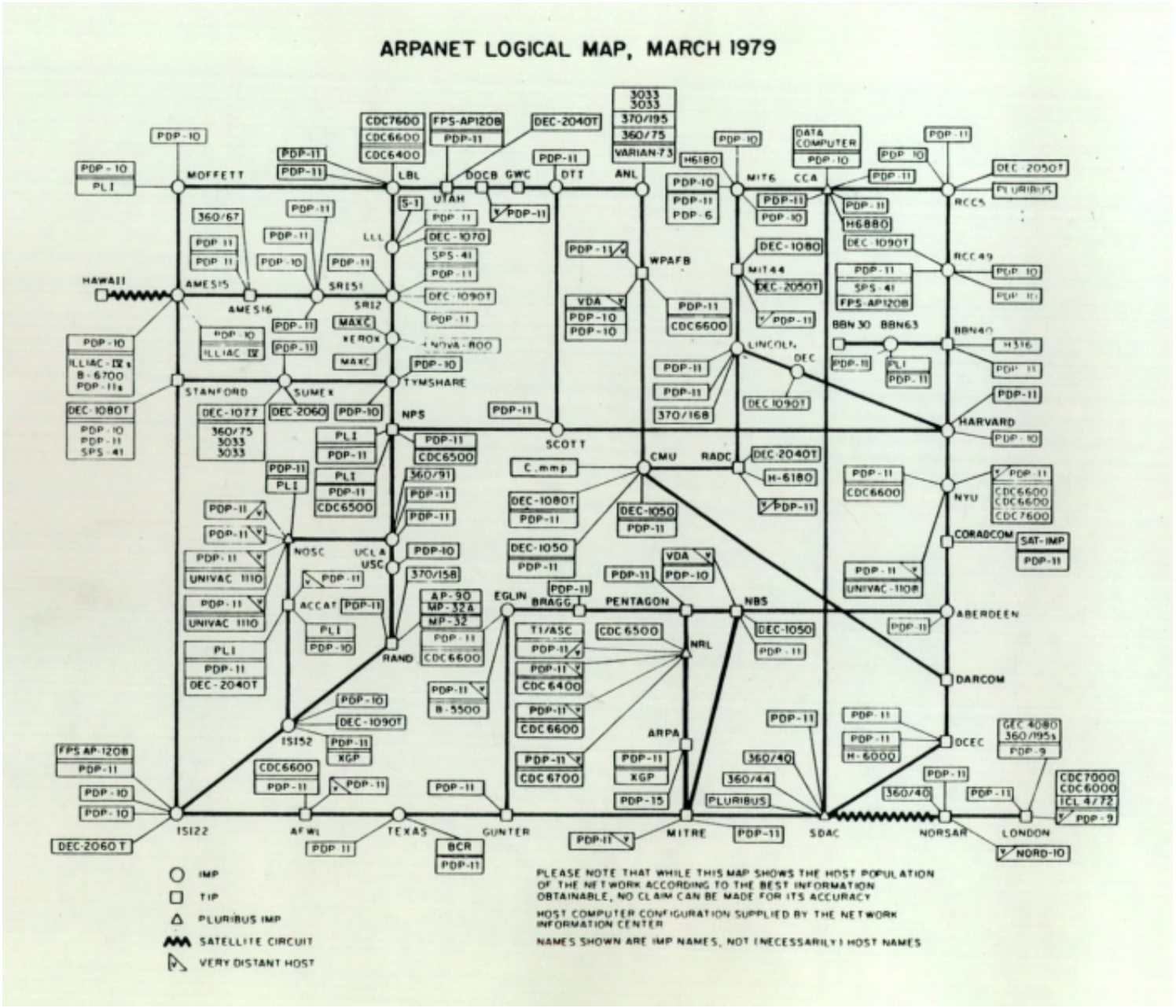
# The Internet: 1975



# The Internet: 1977



# The Internet: 1979



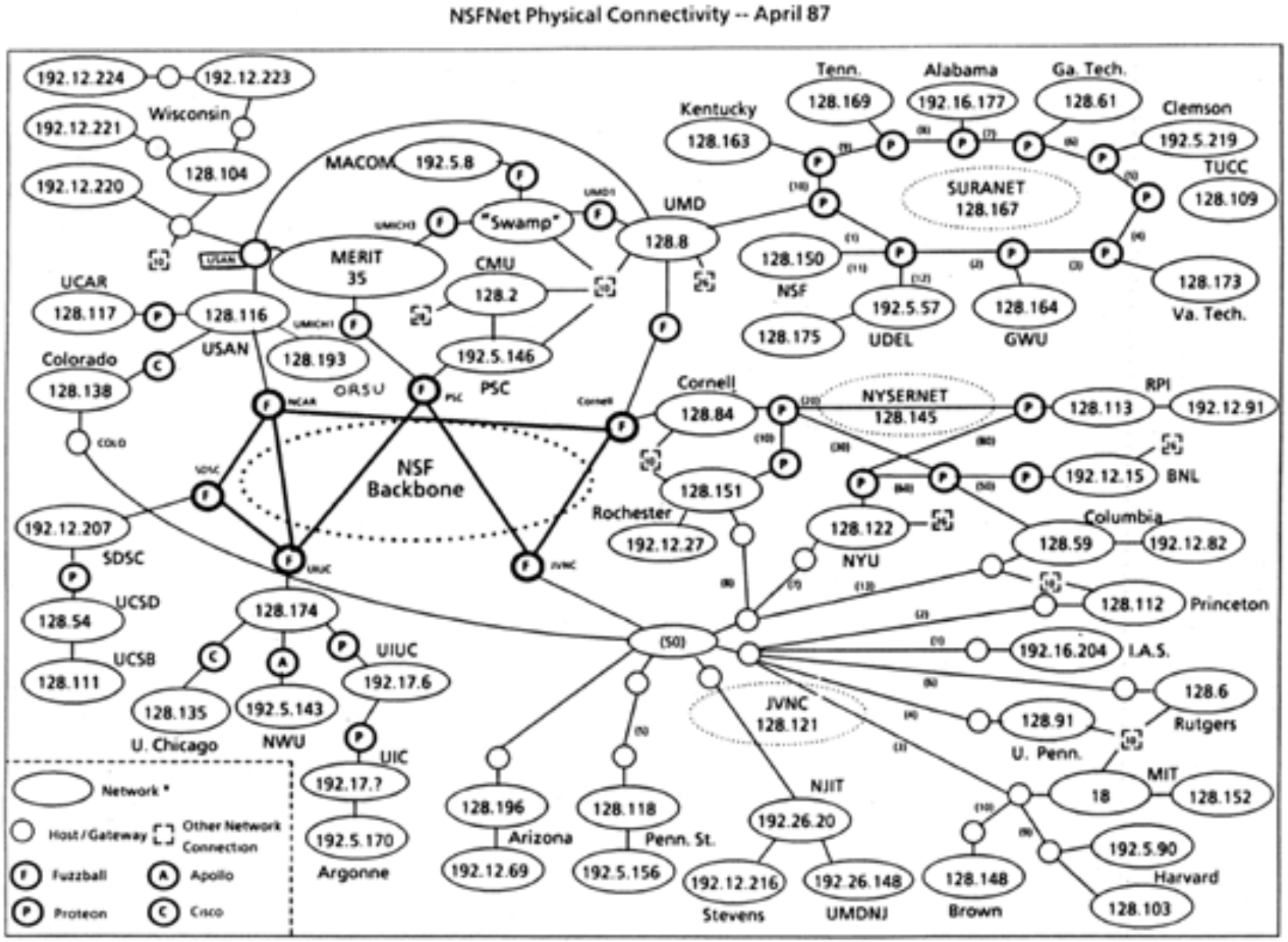
# Internet history

*1980-1990: new protocols, a proliferation of networks*

- **1982:**
  - SMTP e-mail protocol defined
- **1983:**
  - Deployment of TCP/IP
  - DNS defined for name-to-IP address translation
- **1985:**
  - FTP protocol defined
- **1986:**
  - Congestion collapse
  - NSF backbone, 32 kbps -> 40 bps
- **1988:**
  - TCP congestion control

- New national networks:
  - CSNET, BITNET, NSFNET, Minitel
- 100,000 hosts connected to confederation of networks

# The Internet: 1987

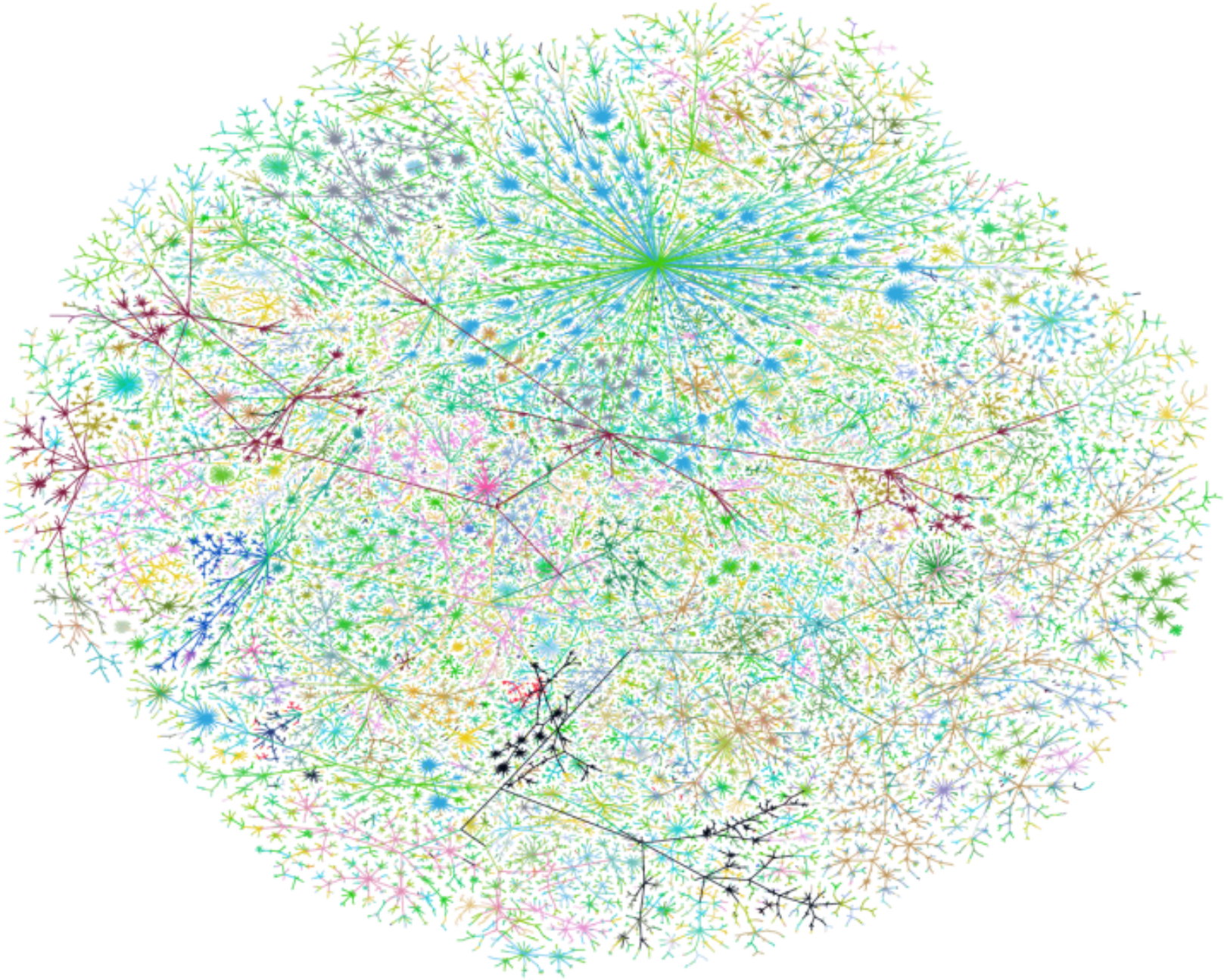


# Internet history

*1990, 2000's: commercialization, the Web, new apps*

- **Early 1990's:**
  - ARPAnet decommissioned
- **1991:**
  - NSF lifts restrictions on commercial use of NSFnet
- **Early 1990's:**
  - Web based on hypertext
  - [Bush 1945, Nelson 1960's]
- **Late 1990's:**
  - Commercialization of the web
- **2000's:**
  - More killer apps: instance messaging, P2P file sharing
  - Network security becomes important
  - Estimated 50 million hosts, 100+ million users
  - Backbone links running at Gbps

# The Internet: 1999



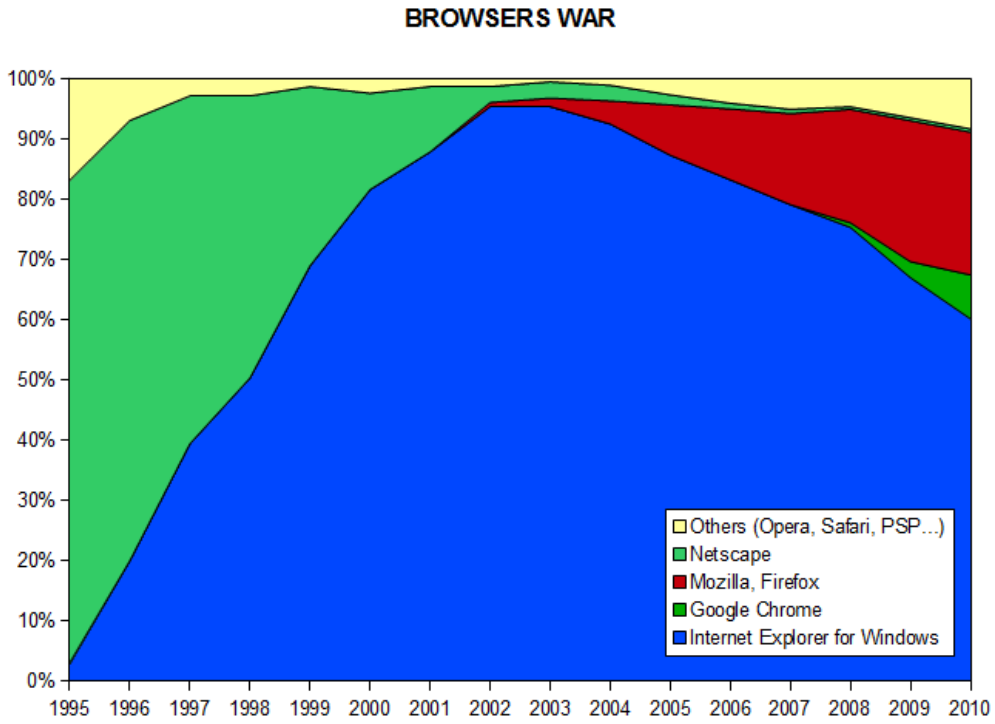

PROPRIETE CERN
PROPRIETE CERN

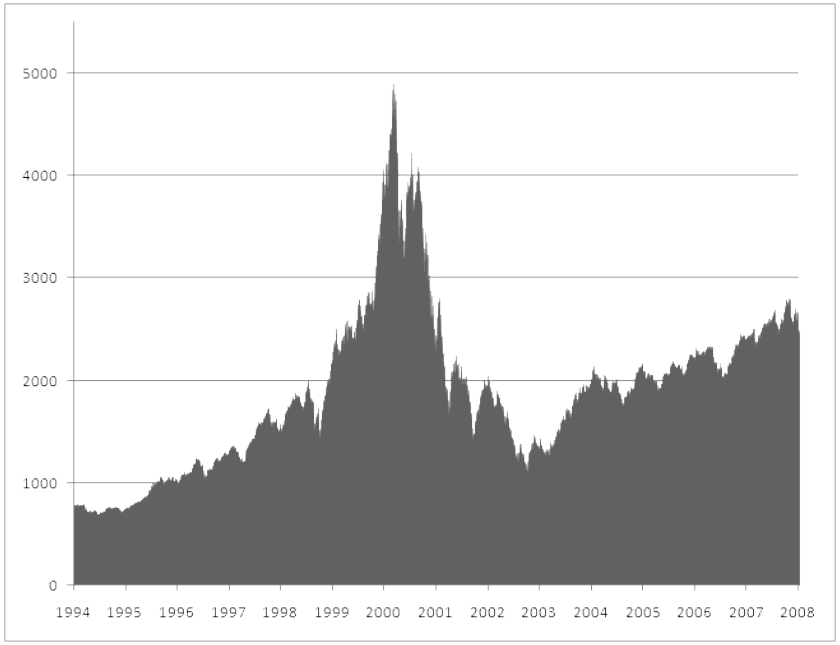
# A short history of the web

- 1989 Tim Berners-Lee at CERN
- 1990 HTTP/0.9, HTML, URLs, first text-based browser
- 1993 Marc Andreesen releases NCSA Mosaic, graphical browser
- 1993 CERN agrees to release protocol royalty-free
- 1994 Andreesen forms Netscape
- 1994 W3C formed, standardizing protocols, encouraging interoperability

# A short history of the web

- 1994+ Browser wars between Netscape and IE
- 1990s-2000 Dot com era

"In the Web's first generation, Tim Berners-Lee launched the Uniform Resource Locator (URL), Hypertext Transfer Protocol (HTTP), and HTML standards with prototype Unix-based servers and browsers.

A few people noticed that the
Web might be better than Gopher.

In the second generation, Marc Andreessen and Eric Bina developed NCSA Mosaic at the University of Illinois.

Several million then suddenly noticed that the
Web might be better than sex.

In the third generation, Andreessen and Bina left NCSA to found Netscape..."

*Microsoft and Netscape open some new fronts in escalating Web Wars*
*By Bob Metcalfe, InfoWorld, August 21, 1995, Vol. 17, Issue 34.*

# Internet history

## *2005-present*

- ~ 750 million hosts
  - Smartphones and tablets
- Aggressive deployment of broadband access
- Increasing ubiquity of high-speed wireless access
- Emergence of online social networks:
  - Facebook: soon one billion users
- Service providers (Google, Microsoft) create their own networks
  - Bypass Internet, providing fast access to search, email, …
- E-commerce, universities, enterprises running their services in the "cloud" (e.g. Amazon EC2)

# Chapter 1: summary

- Internet overview
- What's a protocol?
- Network edge, core, access networks
  - Packet-switching vs. circuit-switching
  - Internet structure
- Performance
  - Loss, delay, throughput
- Layering, service models
- Security
- History

*You now have:*

- Context, overview, "feel" of networking
- More depth and details *to follow!*