15-213

Internetworking I

November 19, 1998

Topics

- Bridged networks
- Internets
- IP datagram delivery
- IP addresses

The internetworking problem

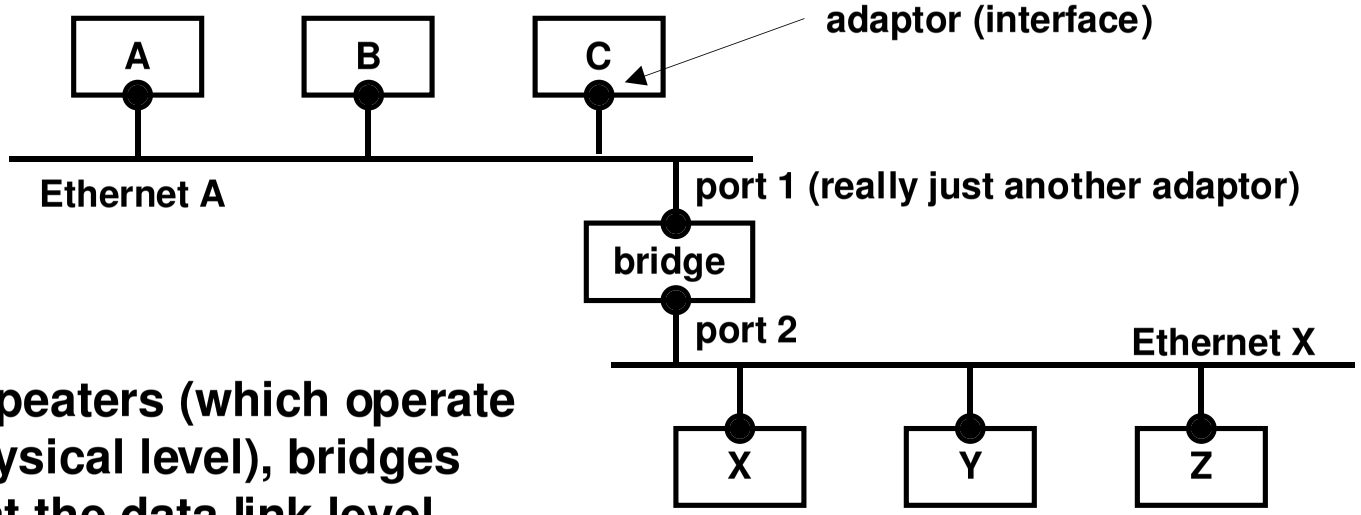
Want to build a single network (an interconnected set of networks, or *internetwork*, or *internet*) out of a large collection of separate networks.

Challenges:

- **heterogeneity**
  - lots of different kinds of networks (Ethernet, FDDI, ATM, wireless, point-to-point)
  - how to unify this hodgepodge?
- **scale**
  - how do we provide uniques names for potentially billions of nodes? (naming)
  - how do we find all these nodes? (forwarding and routing)

Note: *internet* refers to a general idea, *Internet* refers to a particular implementation of that idea (IP).

Bridges



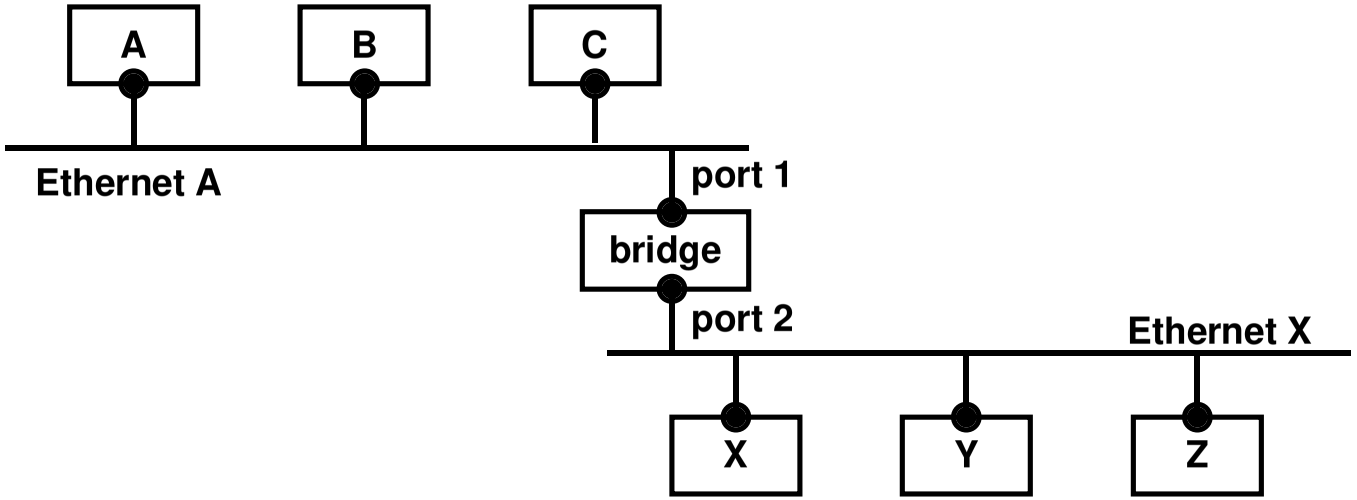
Unlike repeaters (which operate at the physical level), bridges operate at the data link level (or link level).

By link level, we mean that they can parse and understand e.g. ethernet frames (as opposed to IP packets).

Basic forwarding algorithm (*flooding)*: copy each received frame to all other ports.

Learning bridges

Problem: Flooding is wasteful



Optimization: Forward packets only when necessary by learning and remembering which hosts are connected to which bridge ports.

Learning bridges (cont)

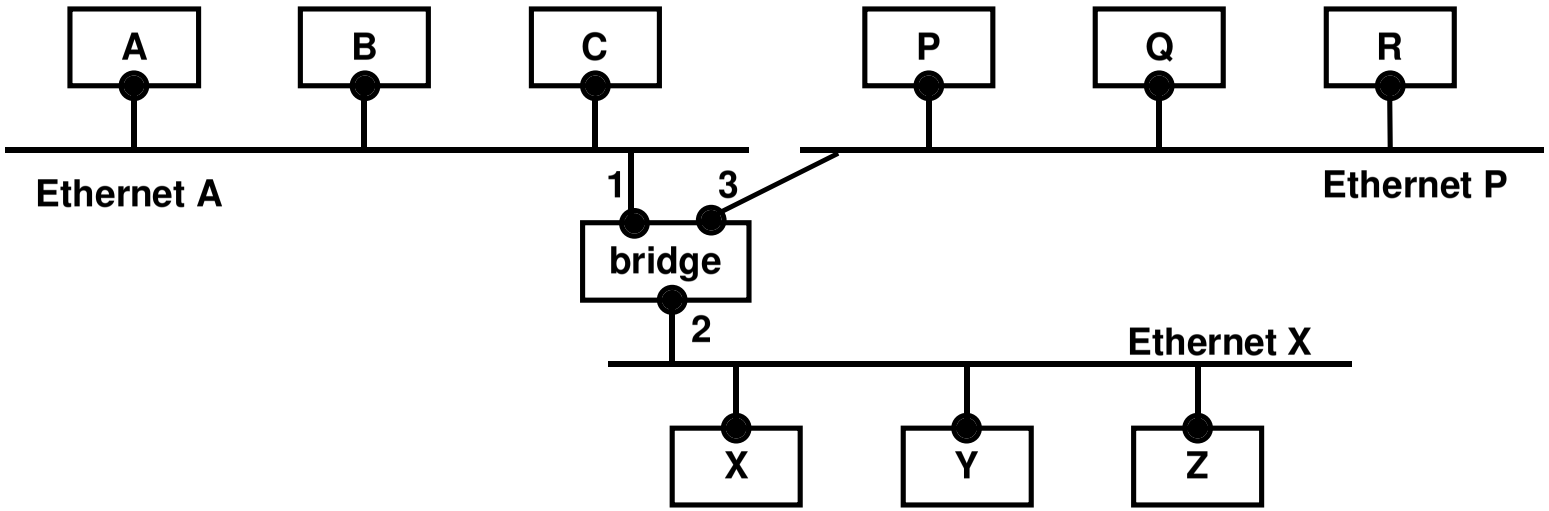
***Learning algorithm*:**

1. **start with empty hash table *T* that maps hosts to ports**
2. **receive frame from host *src* on port *p***
3. **add *(src,p)* to *T***
4. **delete *old* entries**

***Forwarding algorithm*:**

1. **receive frame *f* from host src to host *dst* on port *p***
2. **if *T(dst)* = *n/a* then flood *f*.**
   else if *T(dst)* = *p* then discard *f*
   else forward *f* on port *T(dst).*

Learning bridges (example)



A -> C

host	port
A	1

flood 2 & 3

B -> A

host	port
A	1
B	1

discard

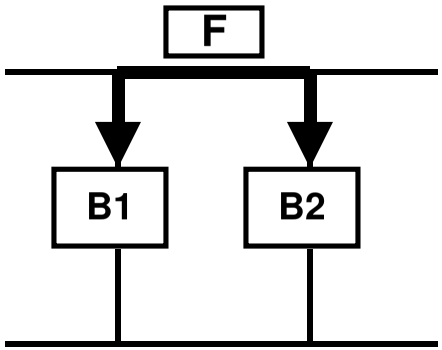
X -> A

host	port
A	1
B	1

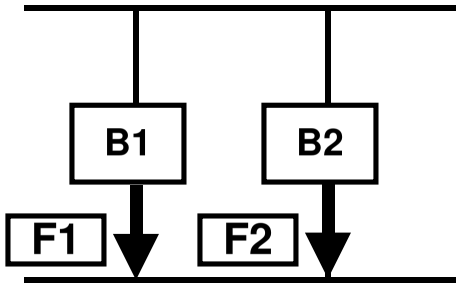
forward on 1

Cycles in bridged networks

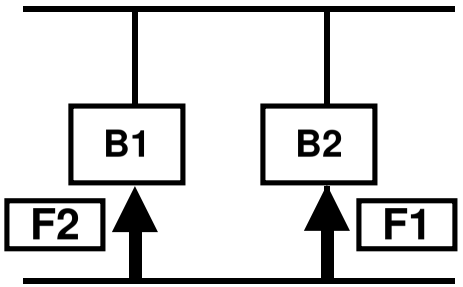
1. host writes frame F
to unknown destination



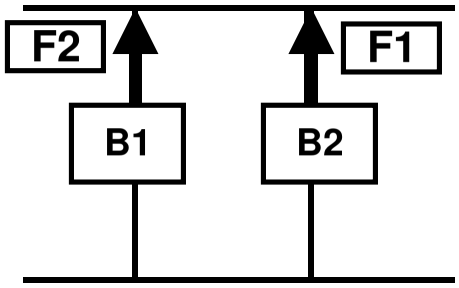
2. B1 and B2 flood



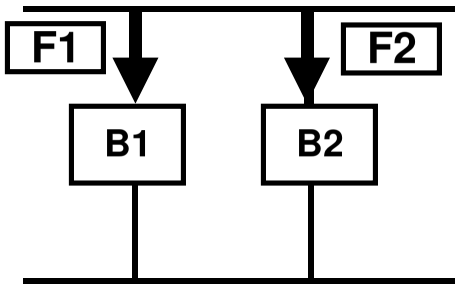
3. B2 reads F1,
B1 reads F2



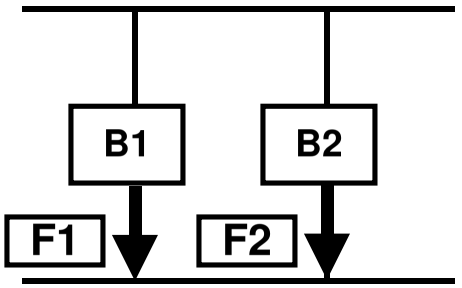
4. B1 and B2 flood



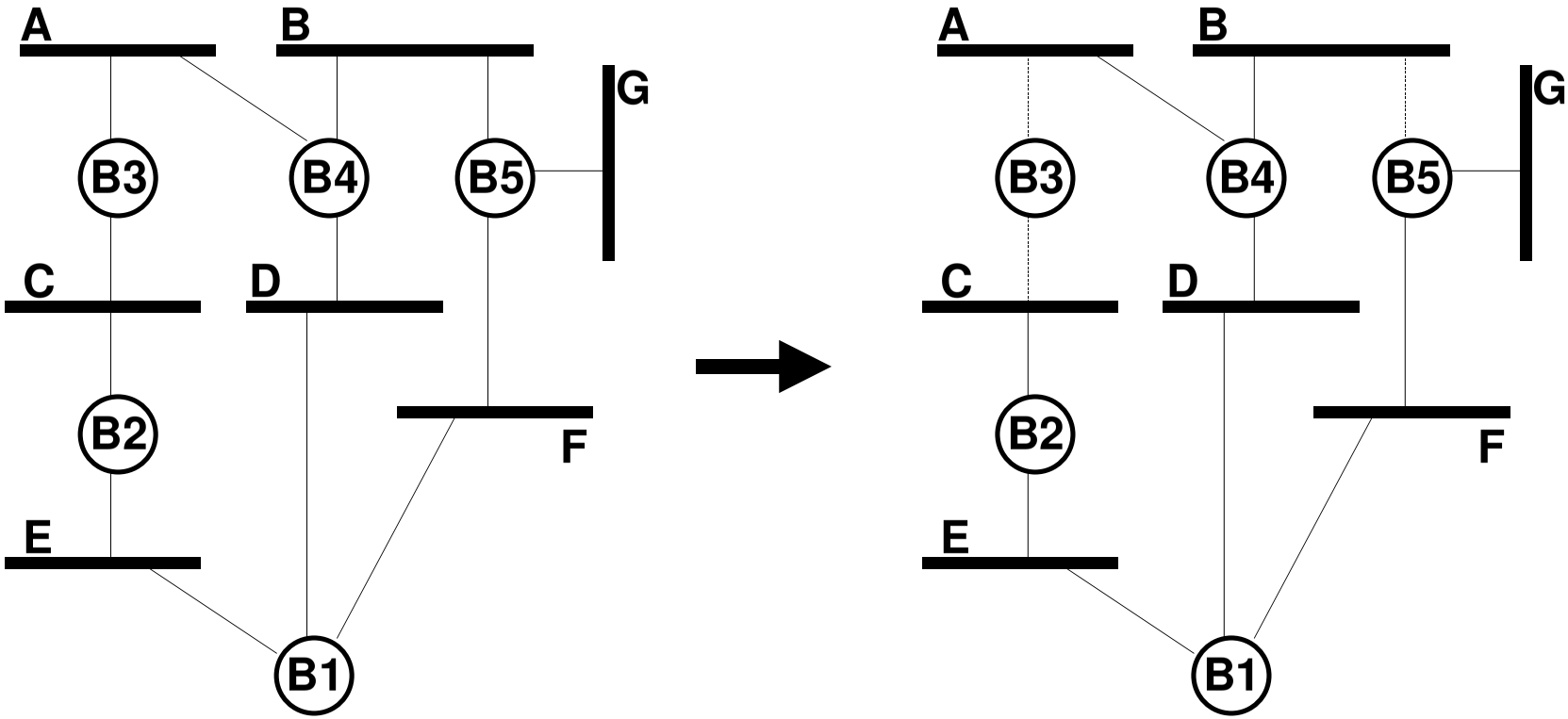
5. B1 reads F1
B2 reads F2



6. B1 and B2 flood



Spanning tree bridges



- Networks are graph nodes, ports are graph edges
- Tree is constructed dynamically by a distributed "diffusing computation" that prunes ports.
- "spanning" refers only to networks, not bridges

Pros and cons of bridges

Pros

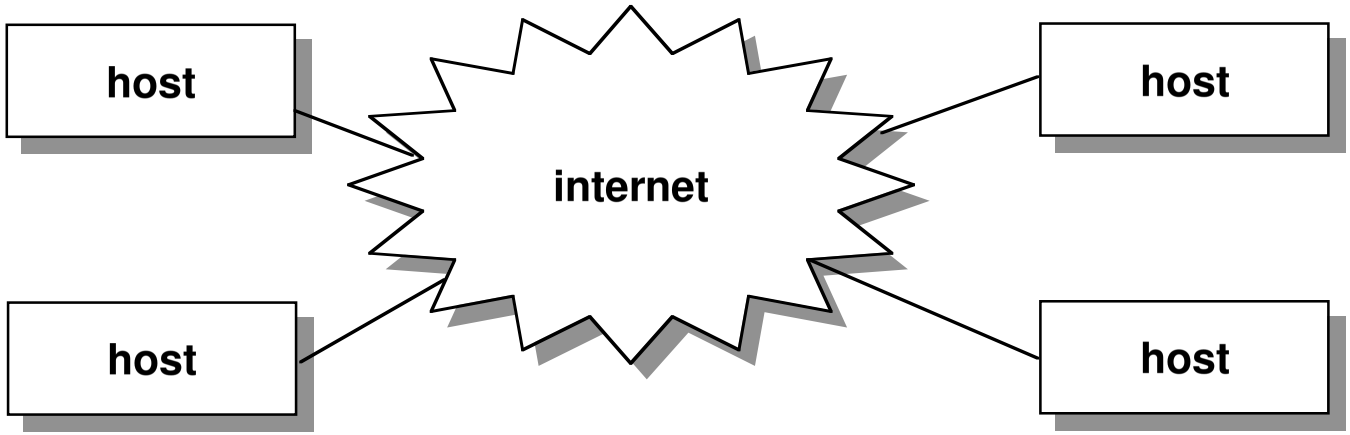
- **transparency**
  - LANS can be connected without any awareness from the hosts

Cons

- **transparency can be misleading**
  - looks like a single Ethernet segment, but really isn't
  - packets can be dropped, latencies vary
- **homogeneity**
  - can only support networks with identical frame headers (e.g., Ethernet/FDDI)
- **scalability**
  - tens of networks only
    - » bridges forward all broadcast frames
    - » increased latency

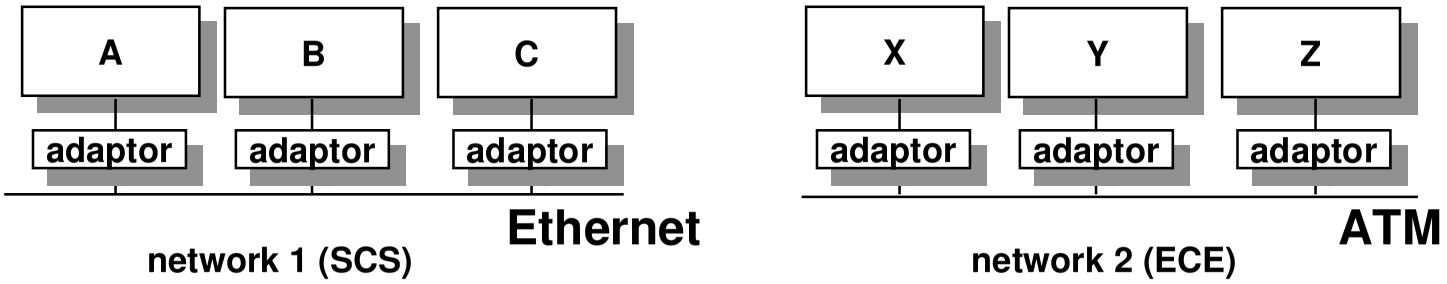
Internetworks

Def: An *internetwork* (internet for short) is an arbitrary collection of *physical networks* interconnected to provide some sort of host-to-host packet delivery service.



Building an internet

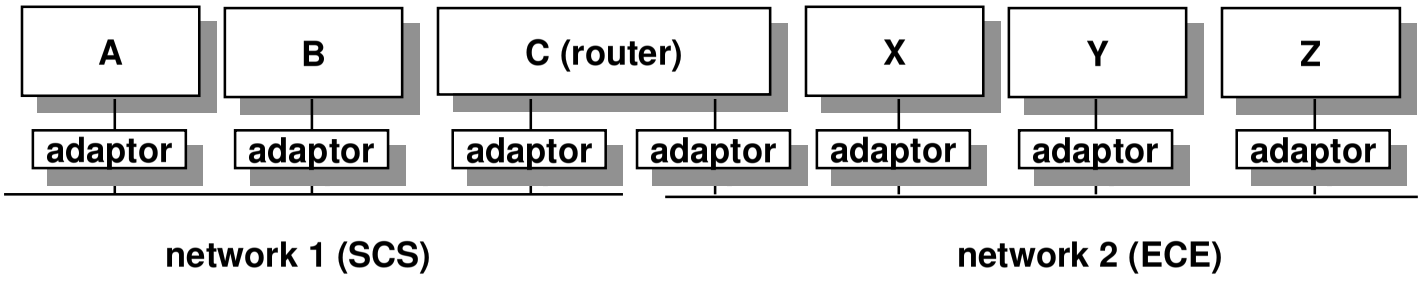
We start with two separate, unconnected computer networks (subnets), which are at different locations, and possibly built by different vendors.



Question: How to present the illusion of one network?

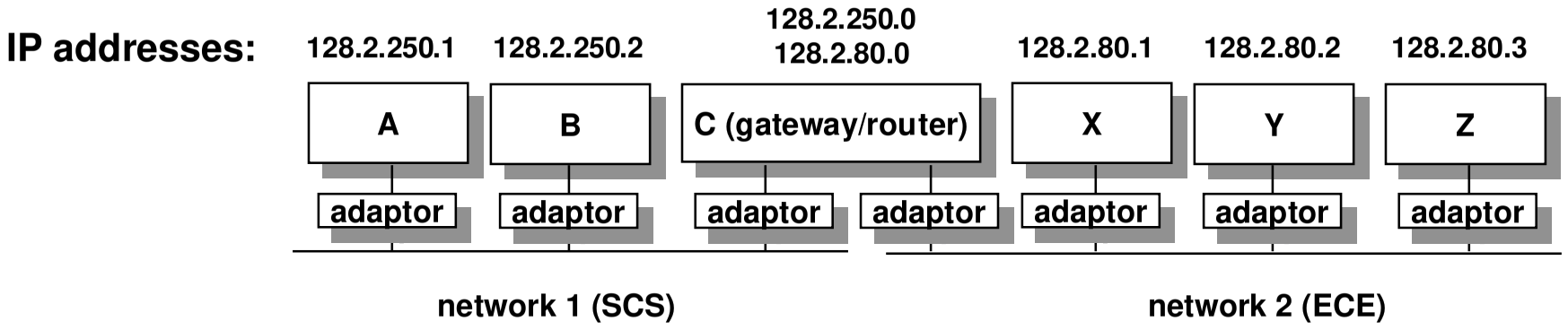
Building an internet (cont)

Next we connect one of the computers (in this case computer C) to each of the networks.



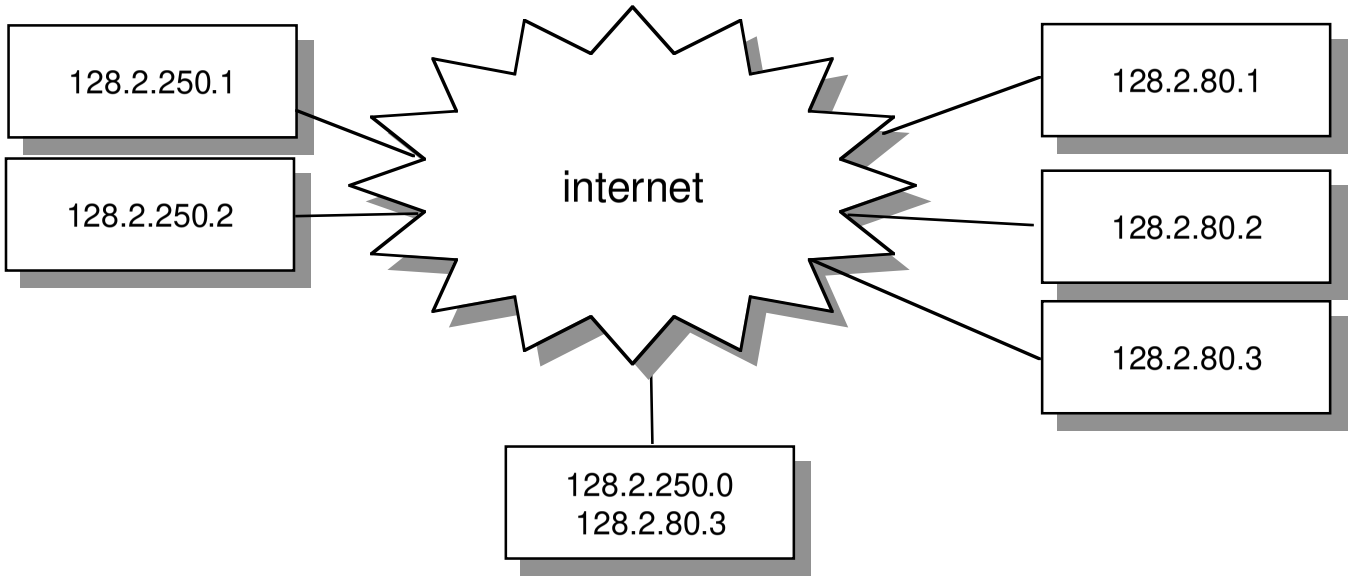
Building an internet (cont)

Finally, we run a software implementation of the Internet Protocol (IP) on each host and router. IP provides a global name space for the hosts, routing messages between network1 and network 2 if necessary.

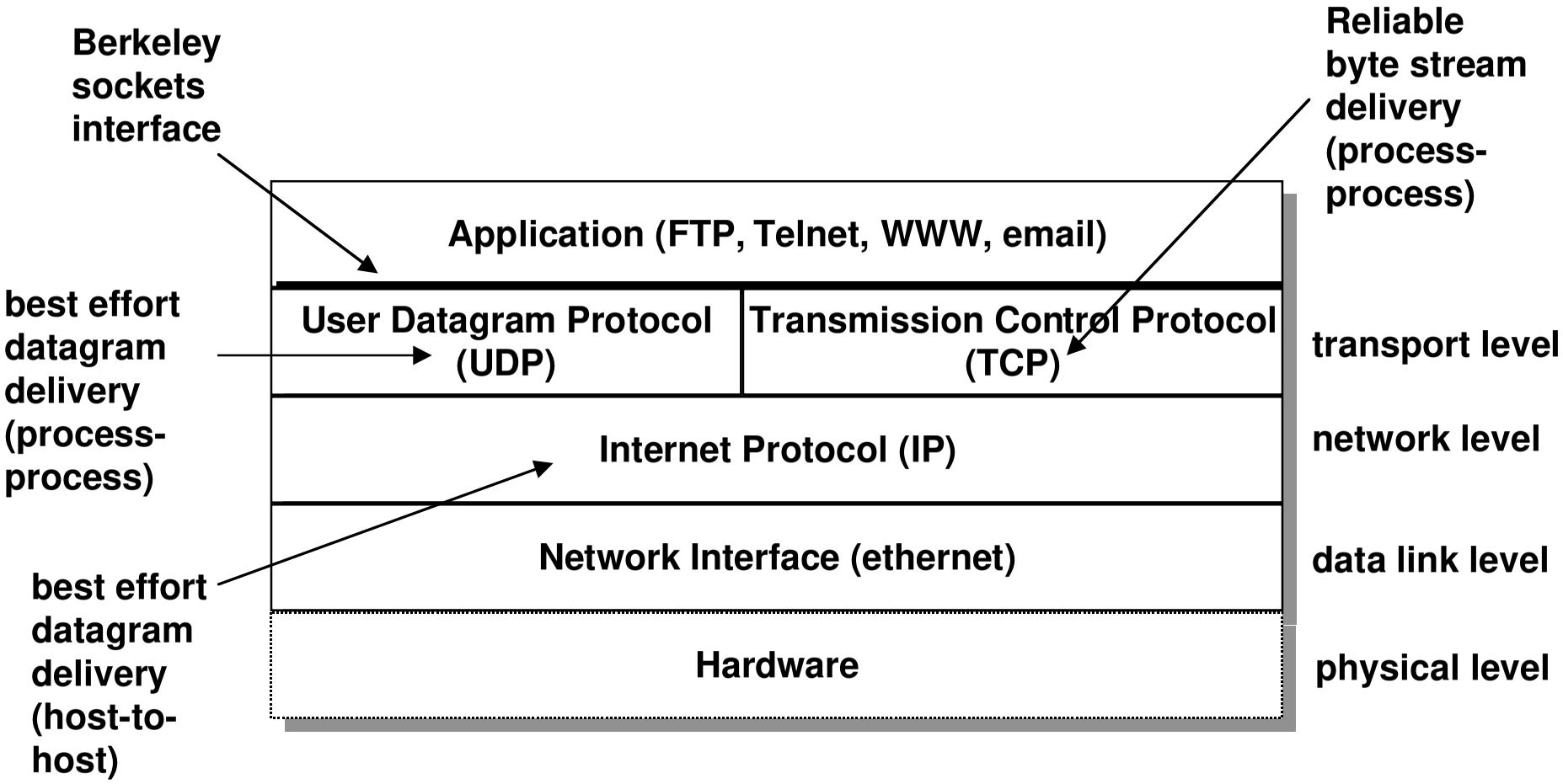


Building an internet (cont)

At this point we have an internet consisting of 6 computers built from 2 original networks. Each computer on our internet can communicate with any other computer. IP provides the illusion that there is just one network.



Internet protocol stack



IP

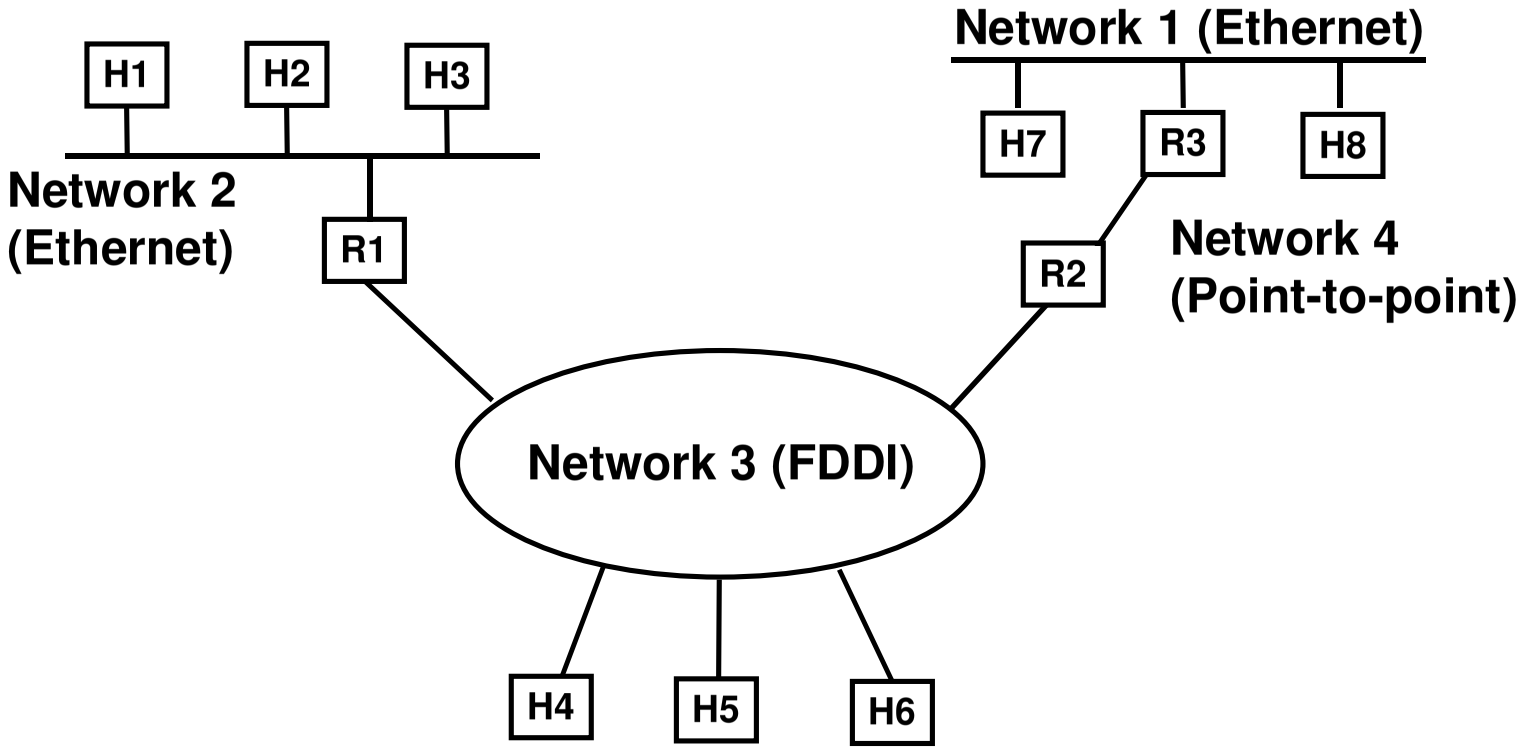
IP service model:

- **IP provides best-effort delivery of datagram (connectionless) packets between two hosts.**
  - IP tries but doesn't guarantee that packets will arrive (best effort)
  - packets can be lost or duplicated (unreliable)
  - ordering of datagrams not guaranteed (connectionless)
- **IP an addressing scheme for all the hosts in the internet**

Why would such a limited delivery model be useful?

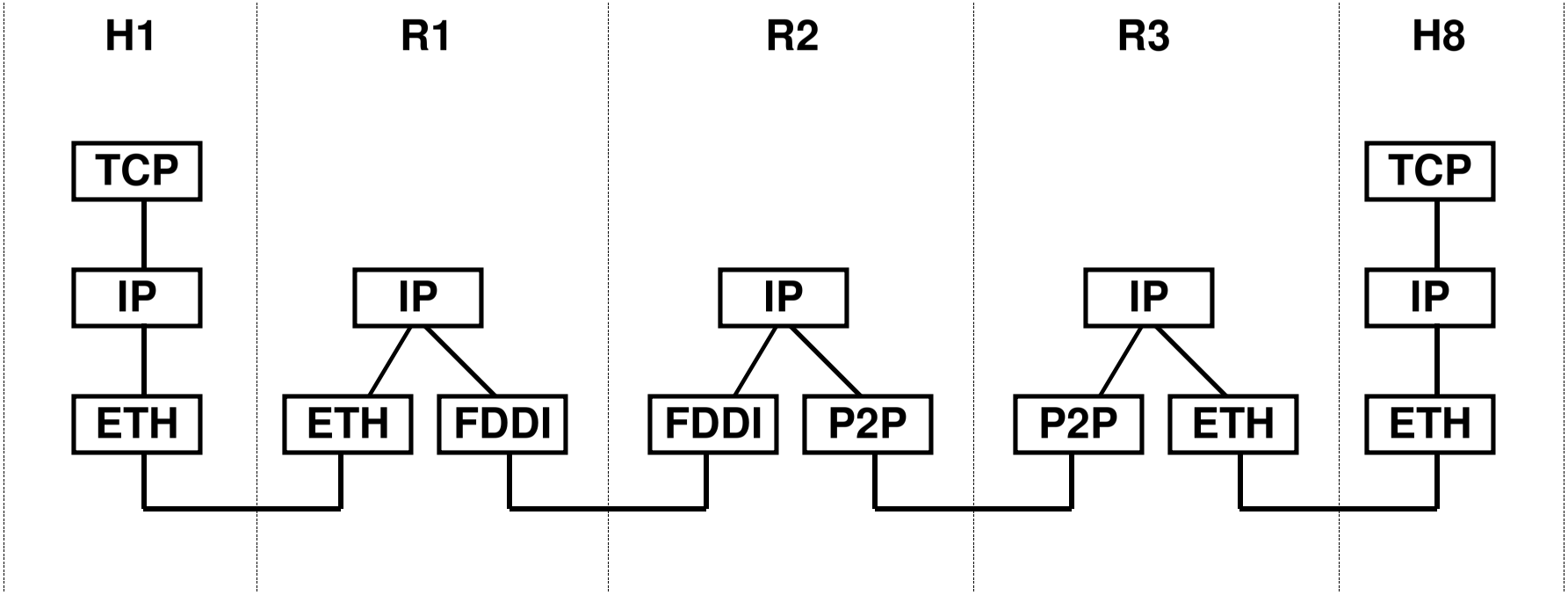
- **simple, so it runs on any kind of network**
- **provides a basis for building more sophisticated and user-friendly protocols like TCP and UDP**

Example internet

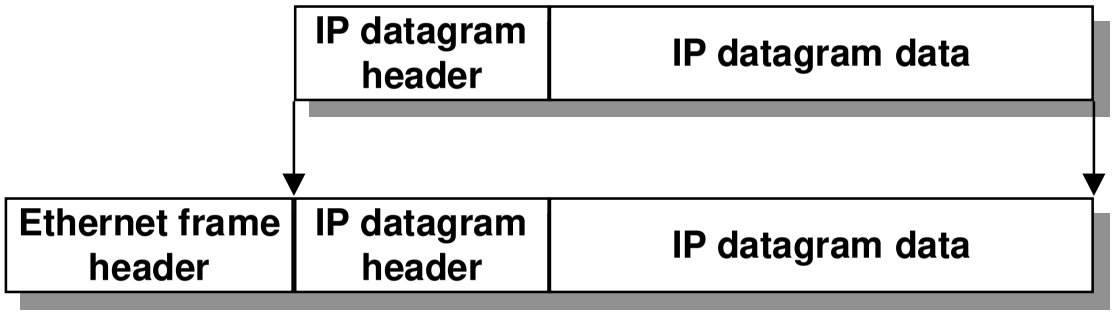


IP layering

Protocol layers used to connect host H1 to host H8 in example internet.



Encapsulating IP datagrams in Ethernet

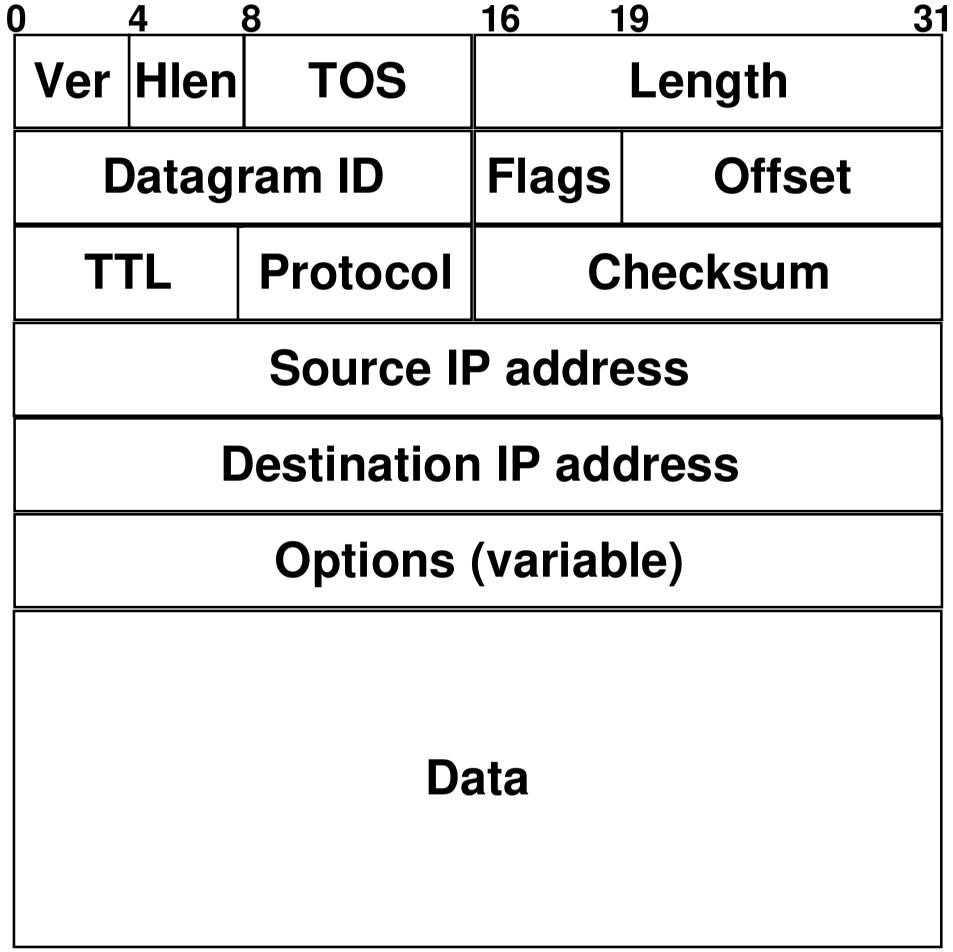


IP datagram

Ethernet frame

The same idea is used for other types of physical networks

IP packet format



VER	IP version
HL	Header length (in 32-bit words)
TOS	Type of service (unused)
Length	Datagram length (max 64K B)
ID	Unique datagram identifier
Flags	xxM (more fragmented packets)
Offset	Fragment offset
TTL	Time to Live
Protocol	Higher level protocol (e.g., TCP)

Fragmentation and reassembly

Different networks have a different maximum transfer unit (MTU).

A problem can occur if packet is routed onto network with a smaller MTU.

- e.g. FDDI (4,500B) onto Ethernet (1,500B)

Solution: break packet into smaller fragments.

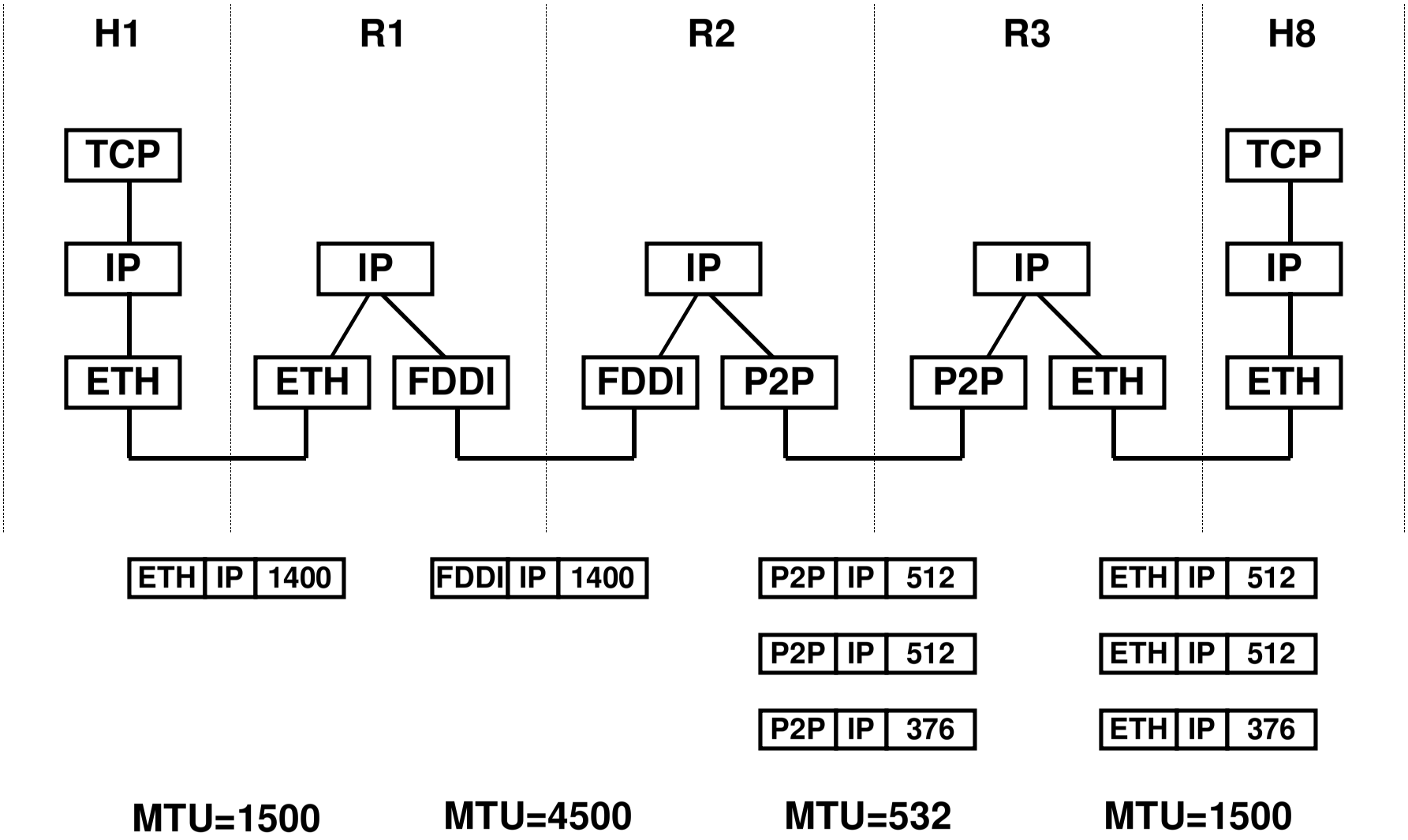
- each fragment has identifier and sequence number

Destination reassembles packet before handing it up in the stack.

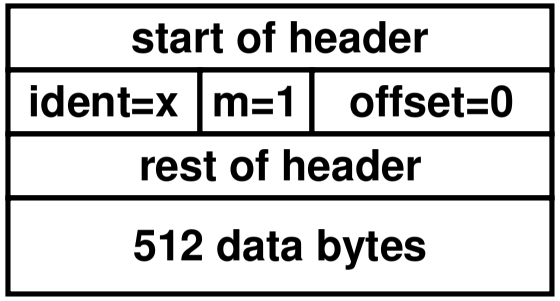
- alternative would be to reassemble when entering network with larger MTU

Sender can disable fragmentation using flag.

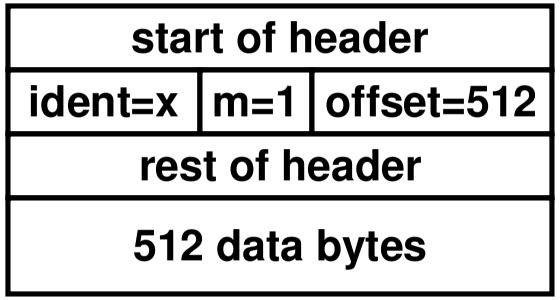
Fragmentation example



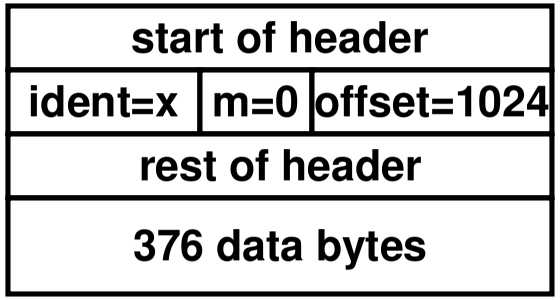
Fragmentation example (cont)



First packet



Second packet



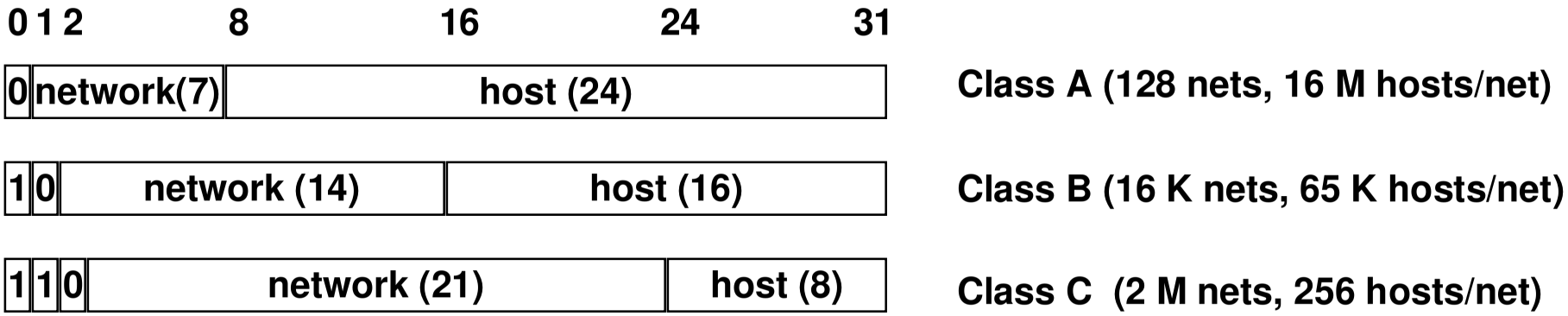
Third packet

Internet addresses

Each host h has a physical address P(h) and a unique IP address I(h).

IP addresses contain a network part and a host part:

3 classes of addresses:



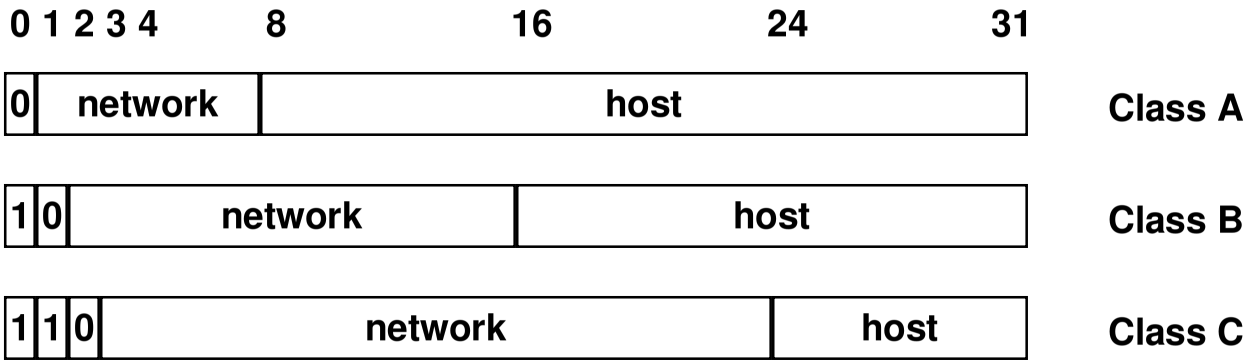
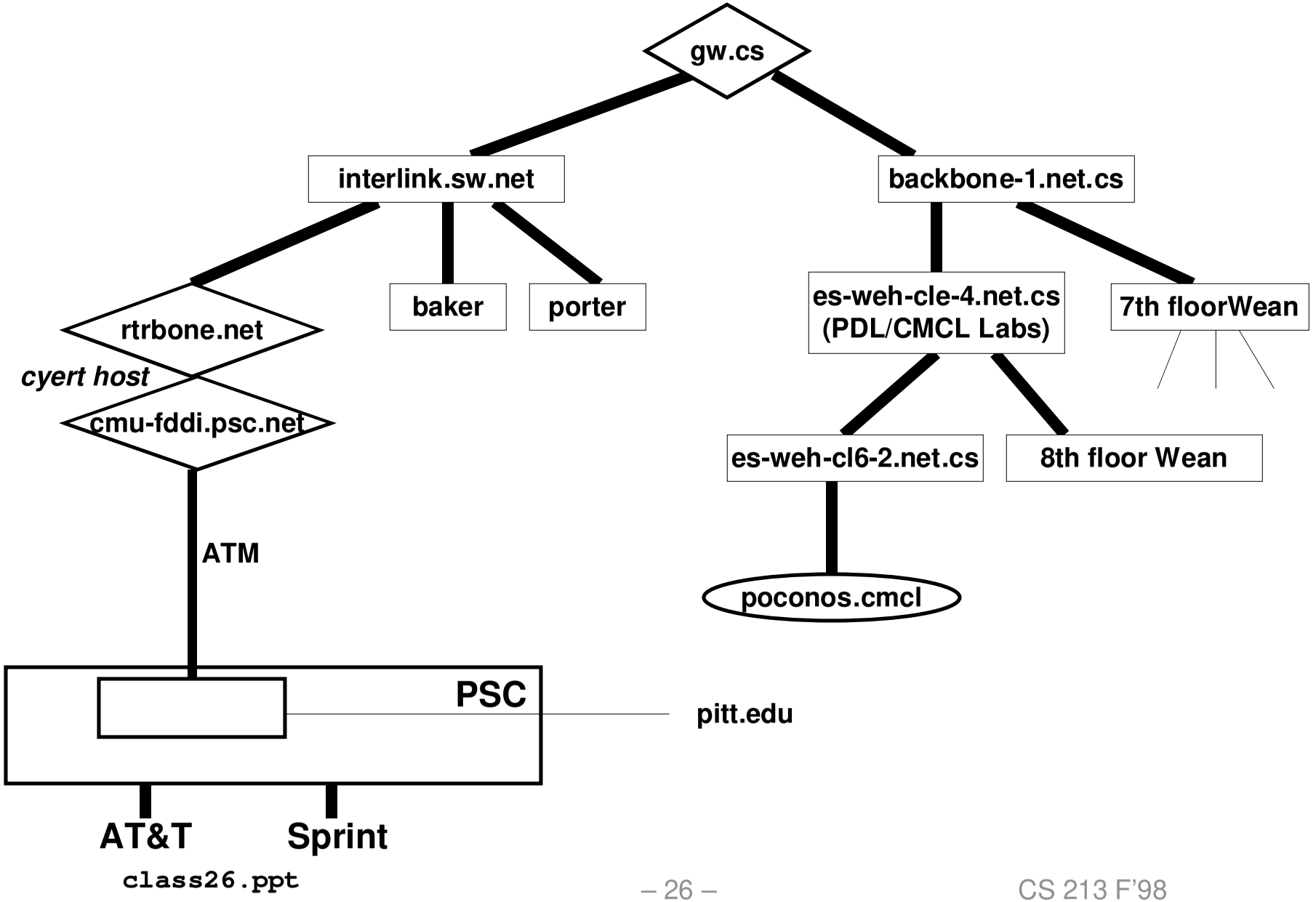
Class A (128 nets, 16 M hosts/net)

Class B (16 K nets, 65 K hosts/net)

Class C (2 M nets, 256 hosts/net)

Example Internet addresses

Host	IP Number	Class	Network
`cs.cmu.edu`	`128.2.222.173`	B	0x0002
`cmu.edu`	`128.2.35.186`	B	0x0002
`cs.stanford.edu`	`171.64.64.64`	B	0x2640
`att.com`	`192.128.133.151`	C	0x008085



CMU internet