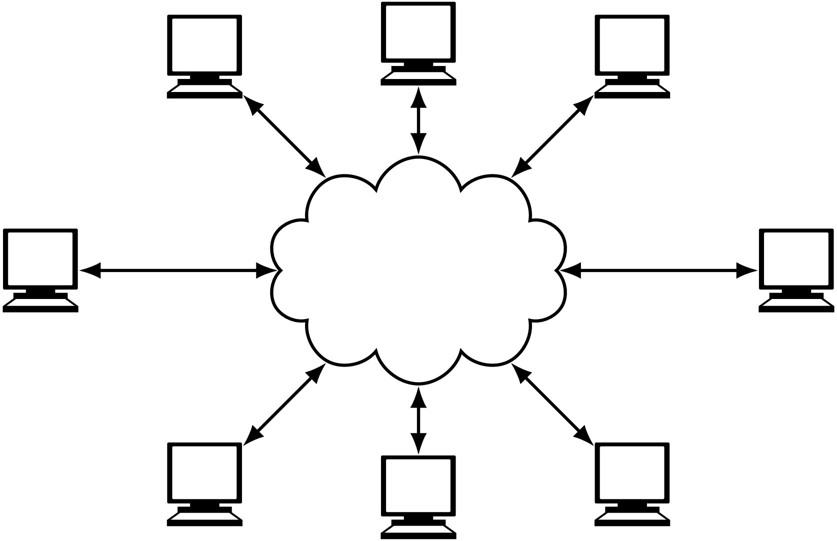
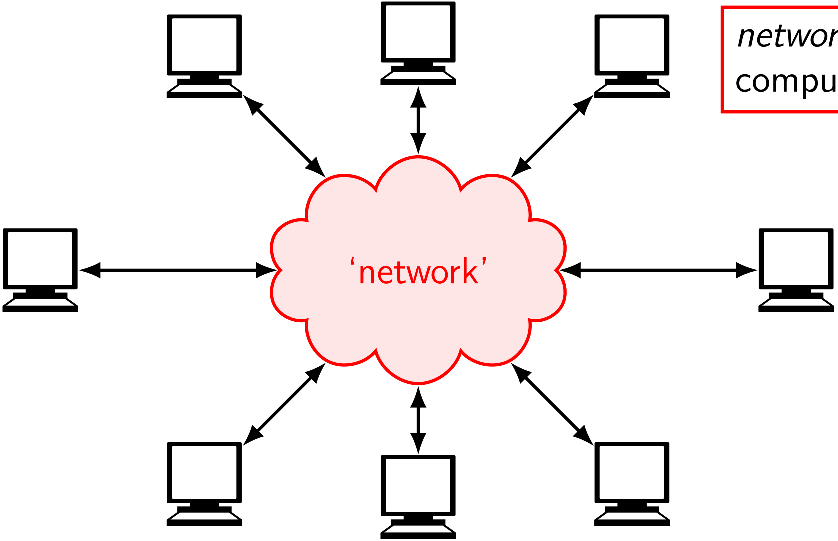
networks / hosts aka end systems

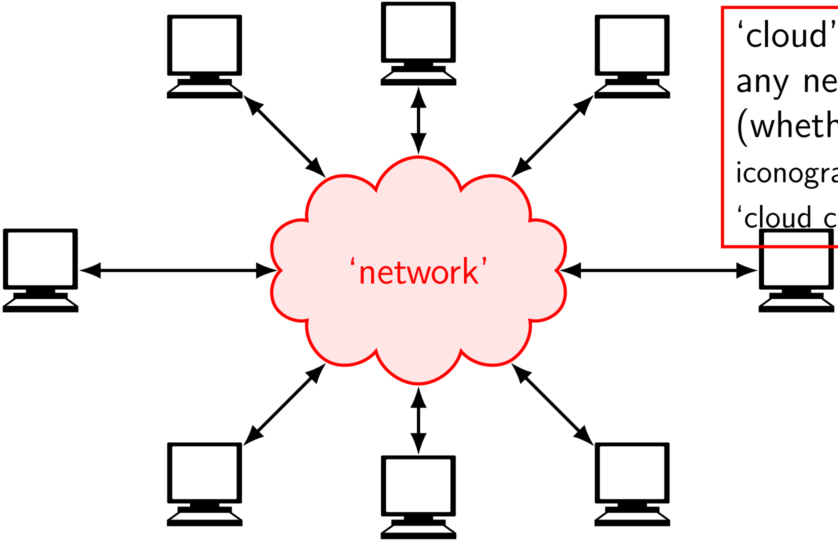


networks / hosts aka end systems



networks connect computers

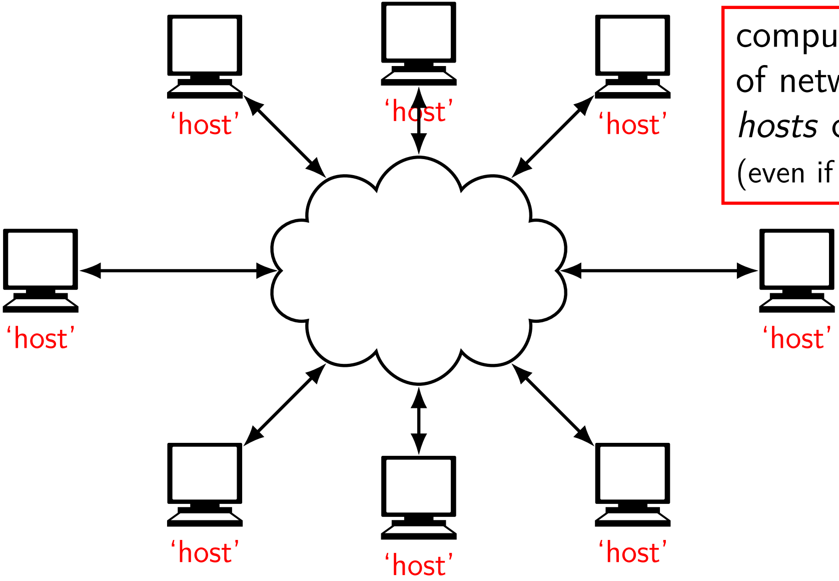
networks / hosts aka end systems



‘cloud’ represents any network (whether local or not)

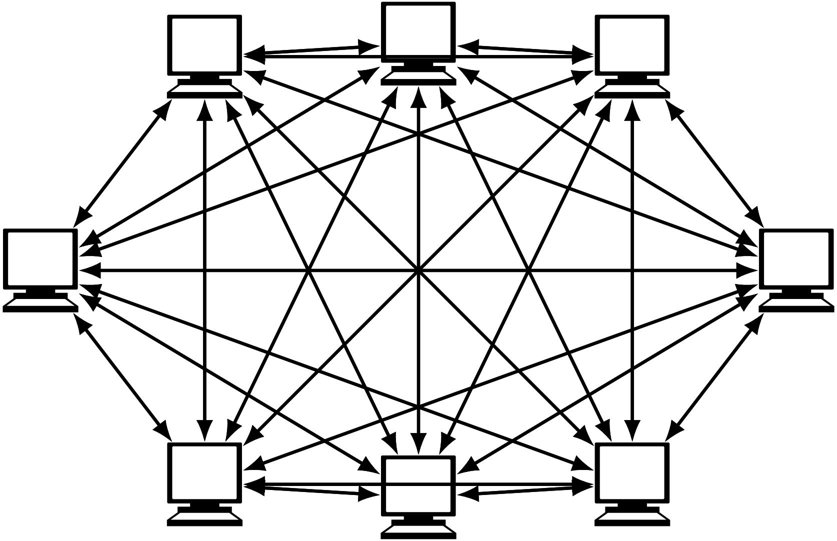
iconography predates ‘cloud computing’

networks / hosts aka end systems

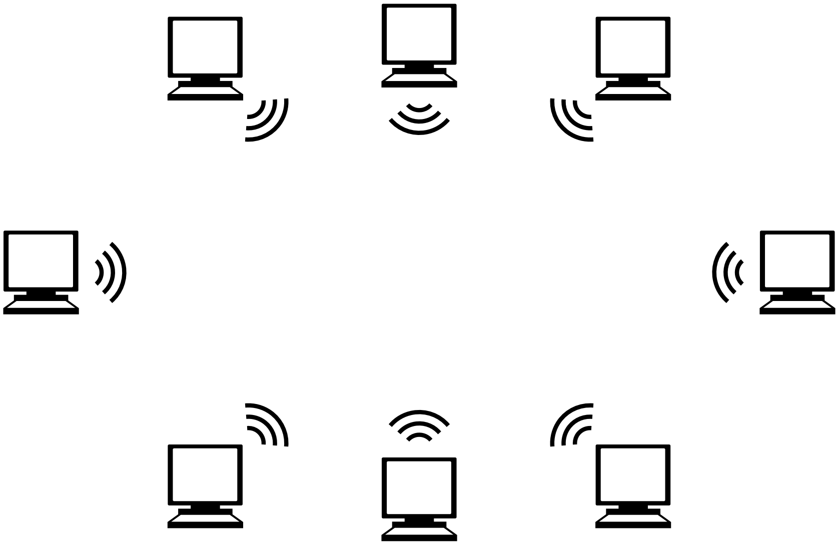


computers on edge of network called *hosts* or *end systems* (even if also ‘servers’)

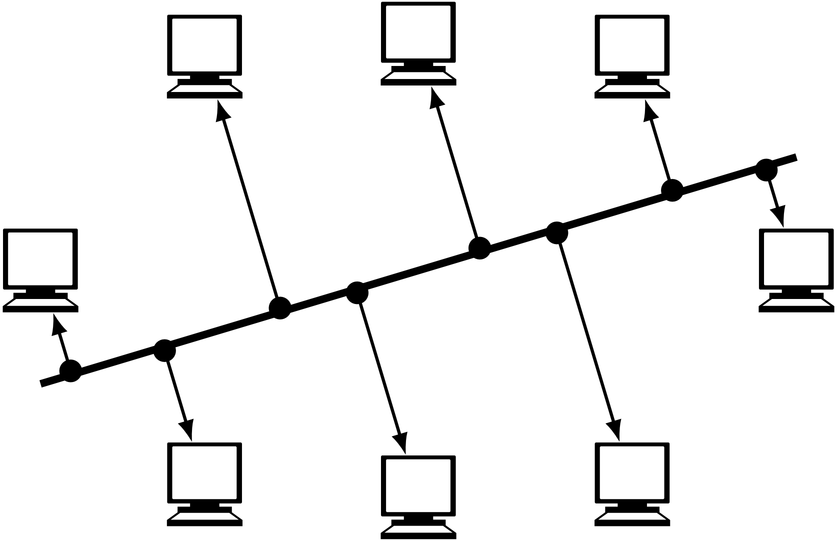
direct connections?



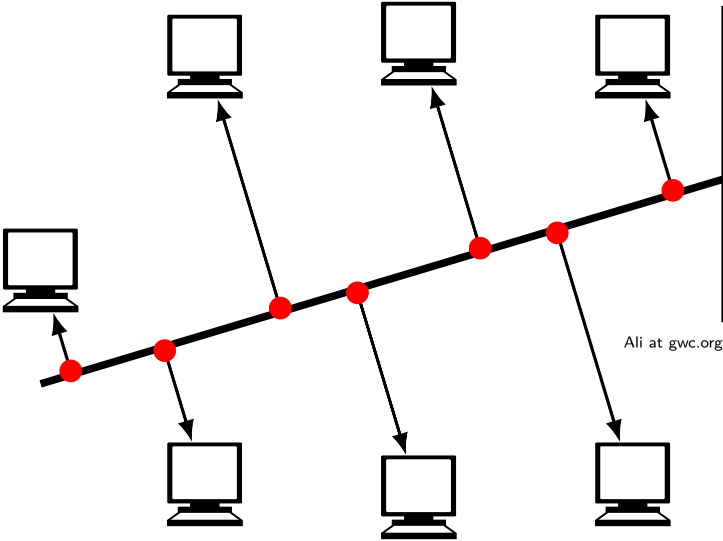
shared medium: radio?

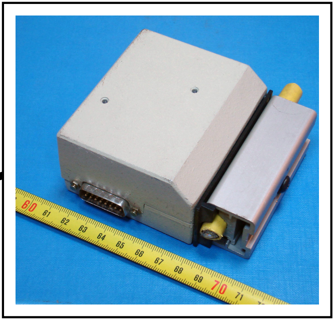


shared medium: wires



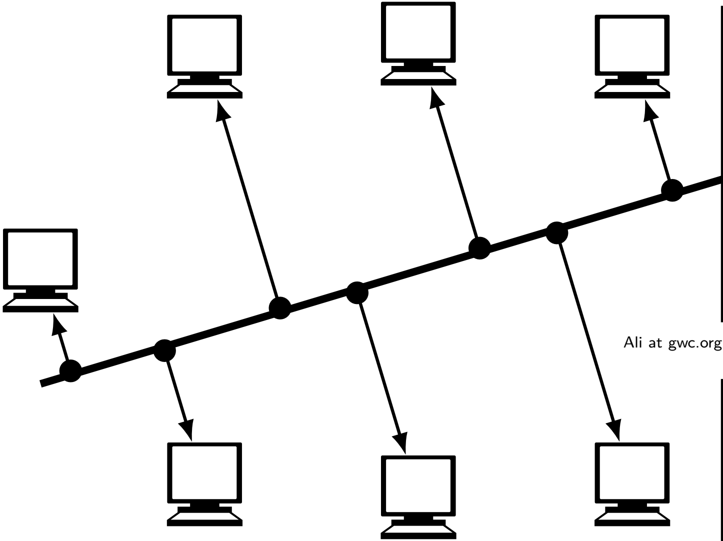
shared medium: wires

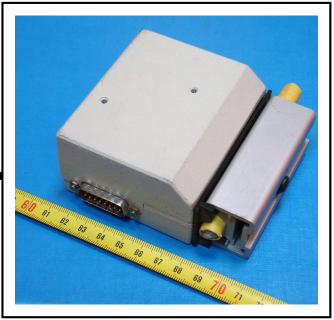




Ali at gwc.org.uk / Alistair1978 via Wikimedia commons / CC-BY-SA 2.5

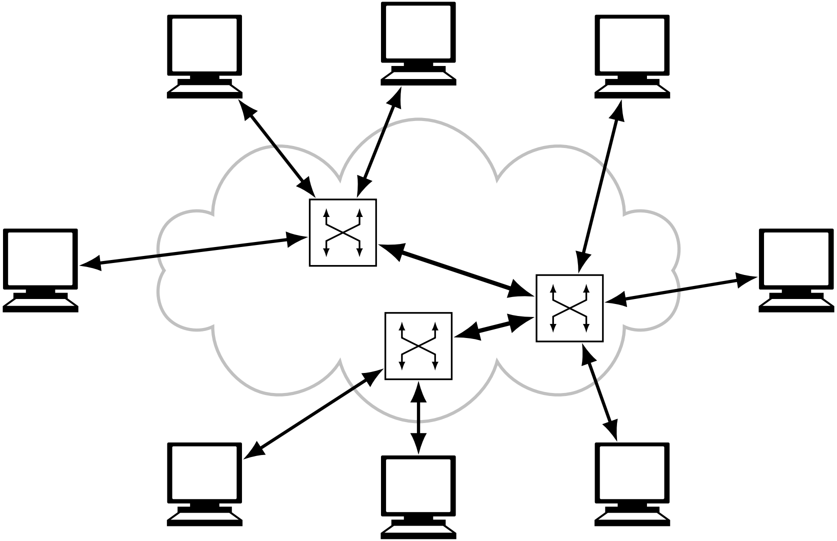
shared medium: wires



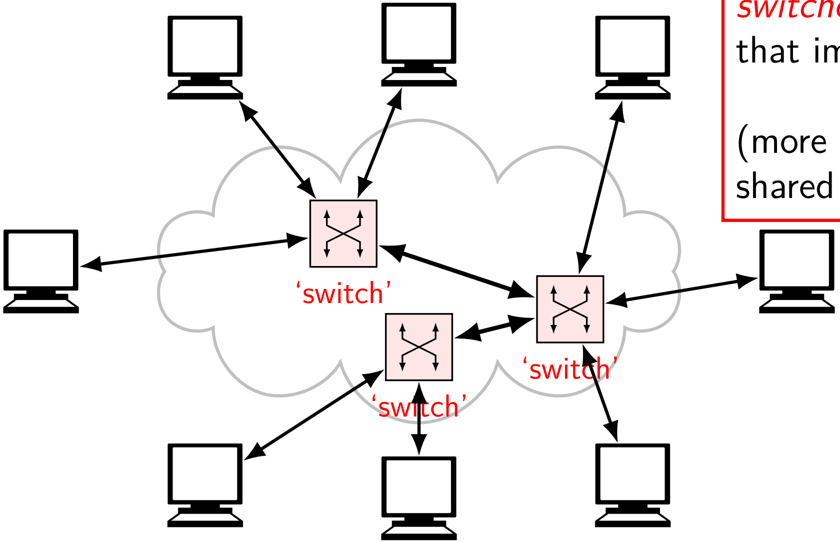


Ali at gwc.org.uk / Alistair1978 via Wikimedia commons / CC-BY-SA 2.5

switches / nodes / links



switches / nodes / links

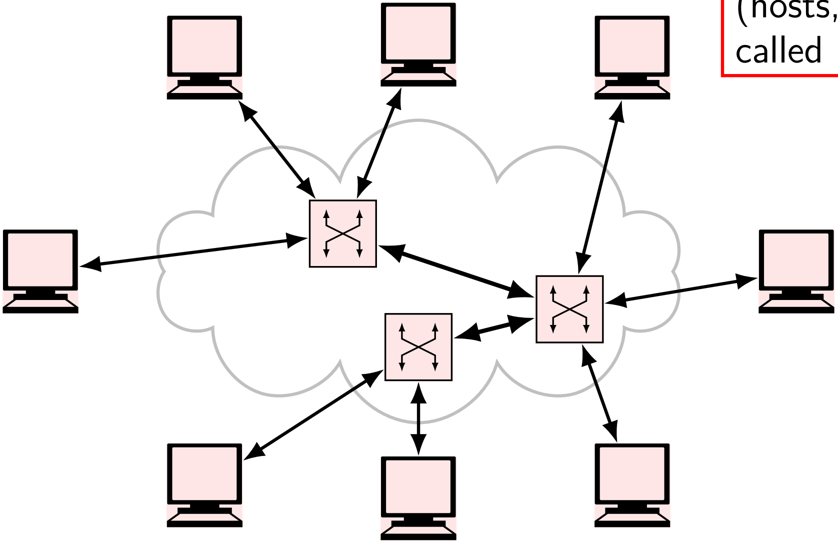


hosts directly connected to *switches*
that implement network

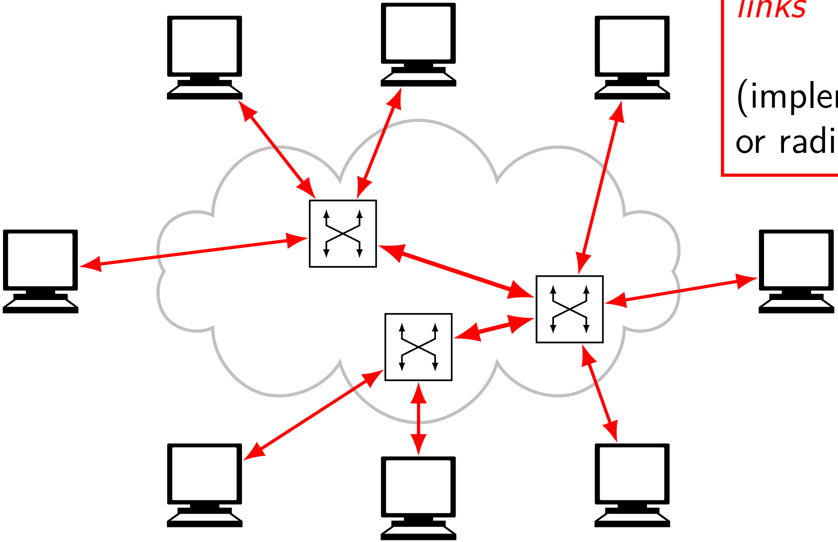
(more efficiently than shared medium)

switches / nodes / links

machines on network (hosts, switches, …) called *nodes*



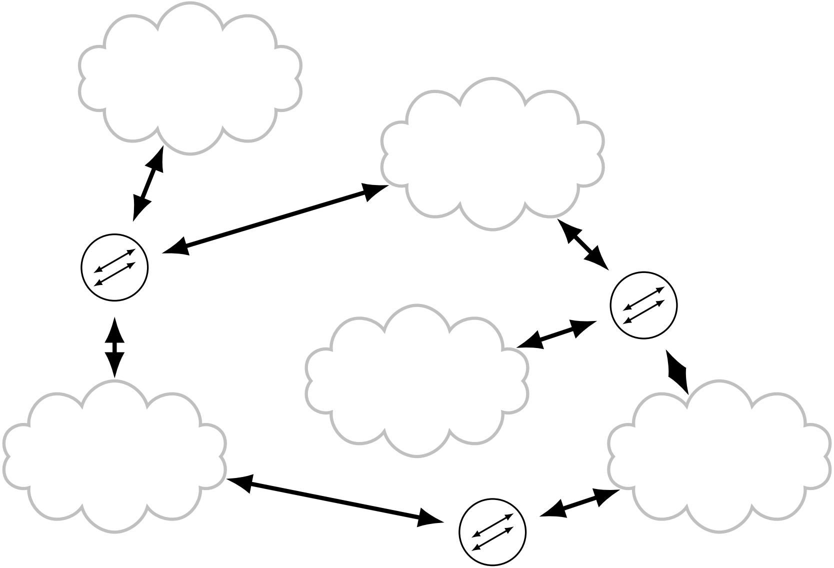
switches / nodes / links



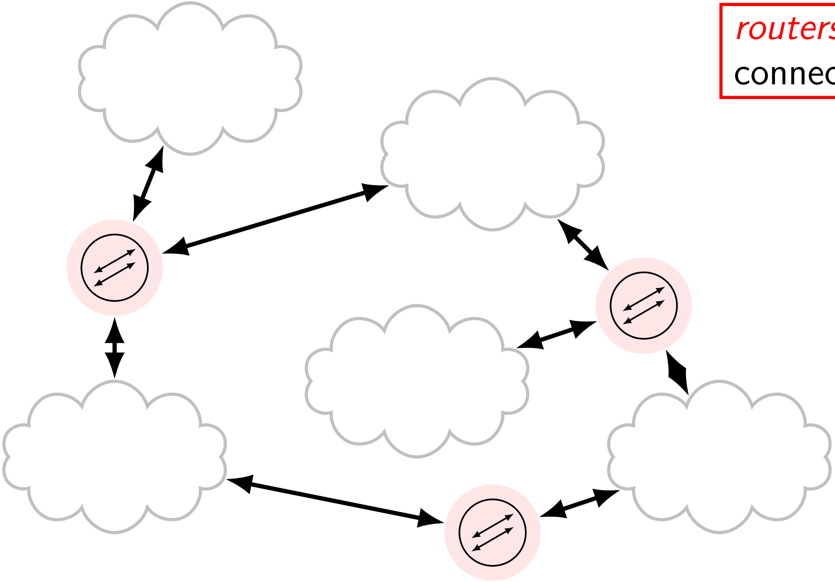
nodes connected by *links*

(implemented by wires or radio or ...)

routers / internetwork

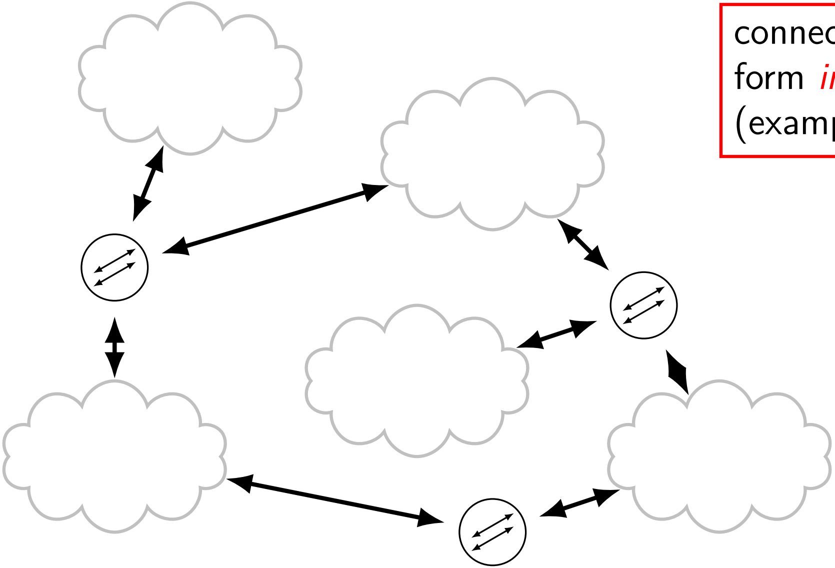


routers / internetwork



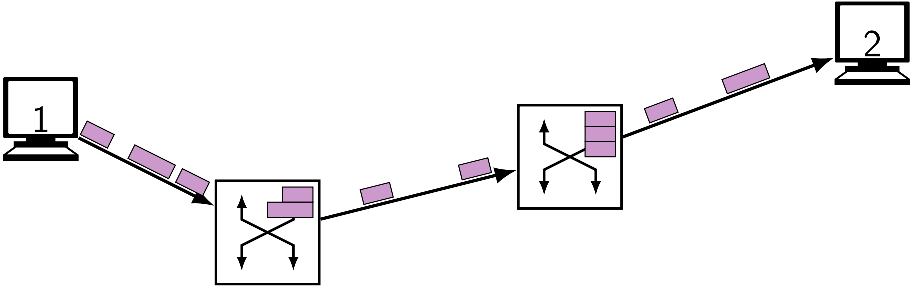
routers or *gateways* connect networks

routers / internetwork



connected networks form *internetwork* (example: the Internet)

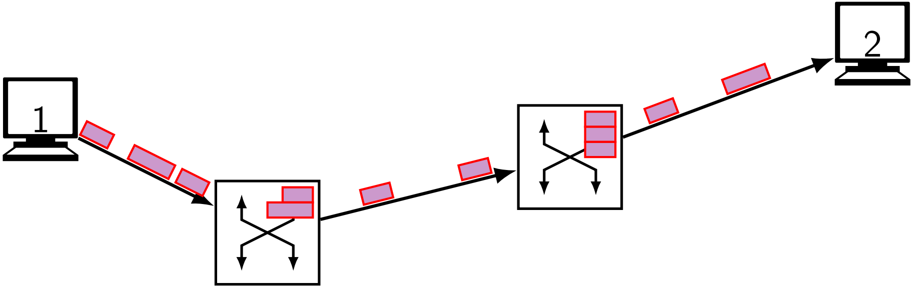
flows / packets



flow of data between two machines

flow is very general term
will depend on context how it relates to
connections, sockets, etc.

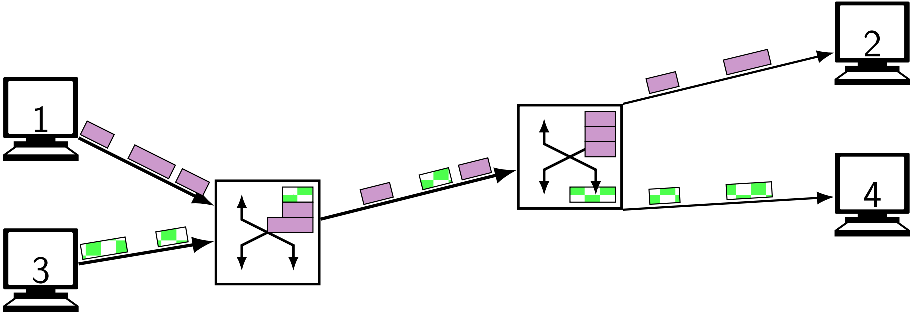
flows / packets



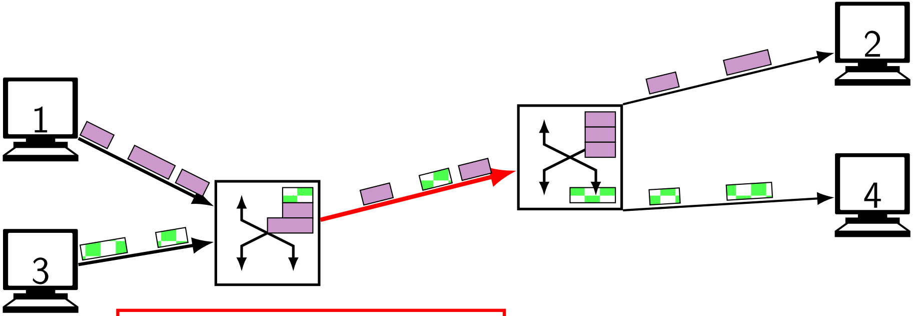
flow of data between two machines

possibly divided up into pieces,
called *packets*, *frames*, *segments*
(which name is best depends on context)

(de)multiplexing

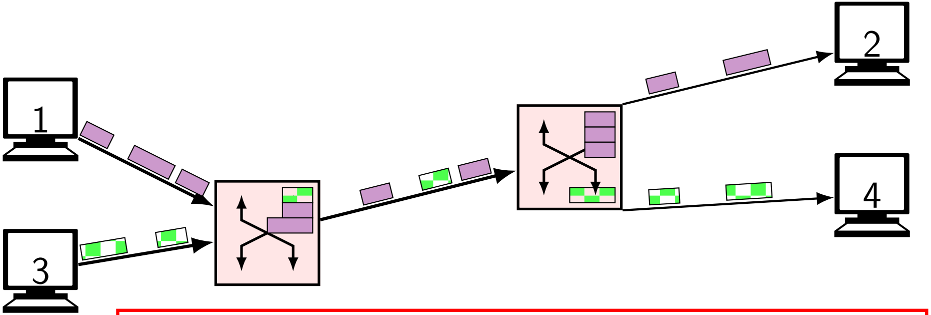


(de)multiplexing



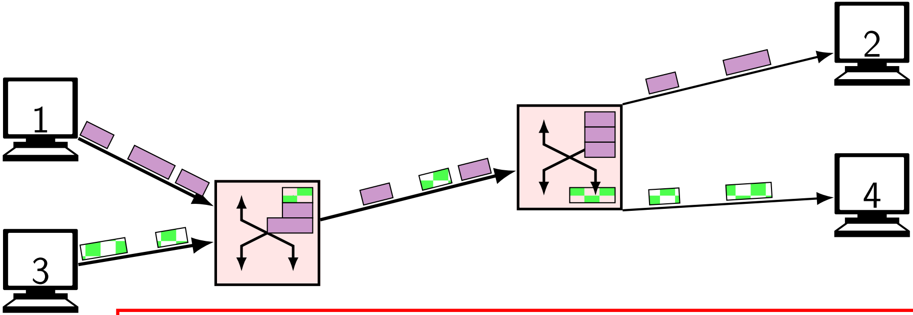
two or more flows can share one or more links

(de)multiplexing



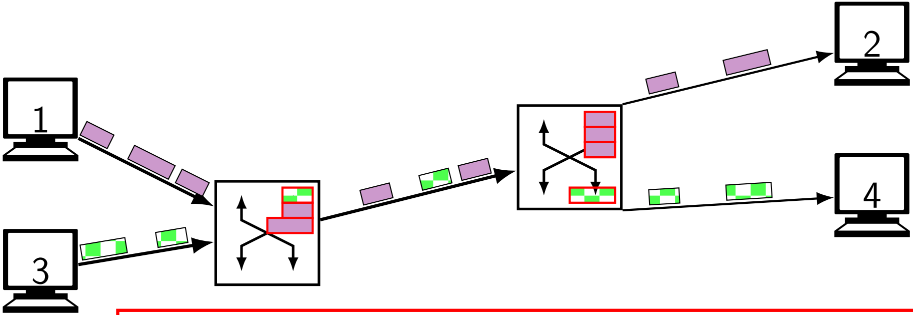
left switch *multiplexes* the two flows onto one link
right switch *demultiplexes* them to separate them

(de)multiplexing



this picture: multiplexed by dividing up *time* on link

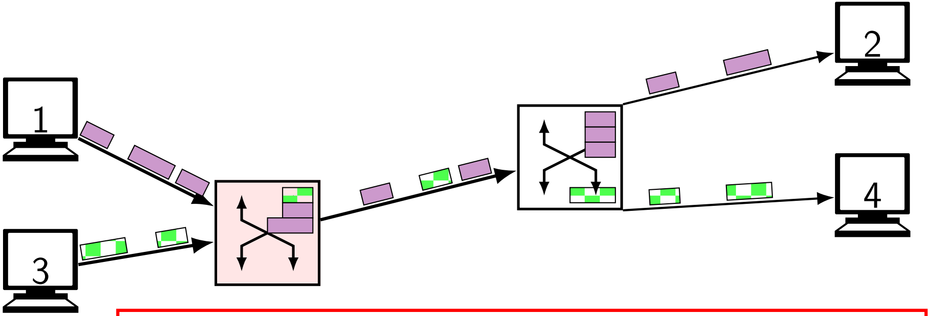
(de)multiplexing



switches usually have *buffers* (also called *queues*) hold waiting packets

absorbs temporary "bursts" where packets come faster than outgoing link can handle

(de)multiplexing

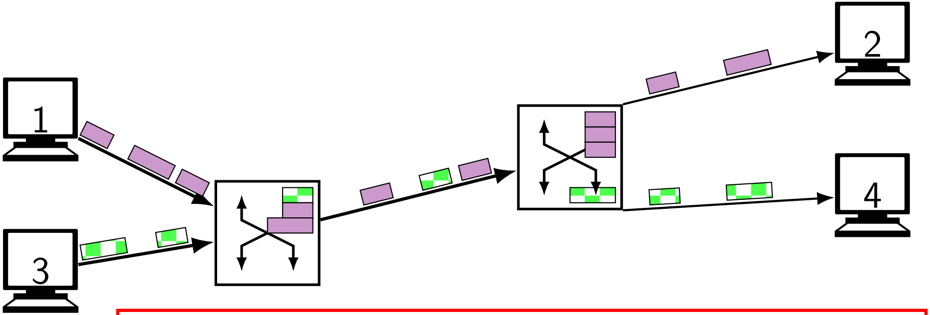


incomplete list of causes of 'bursts':

multiple unsynchronized flows

fast links produce packets faster for slow can send

(de)multiplexing

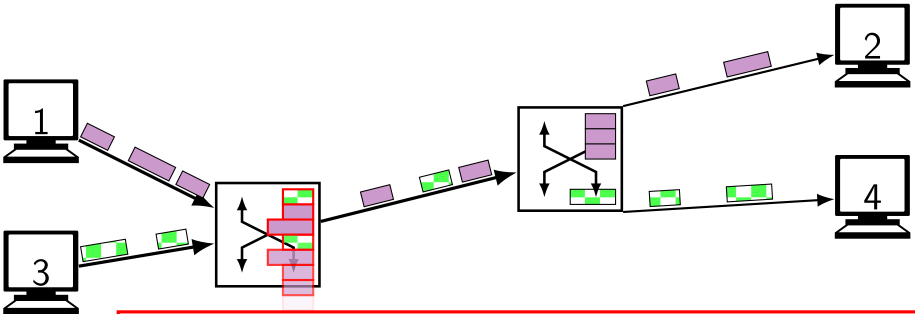


incomplete list of causes of 'bursts':

multiple unsynchronized flows

fast links produce packets faster for slow can send

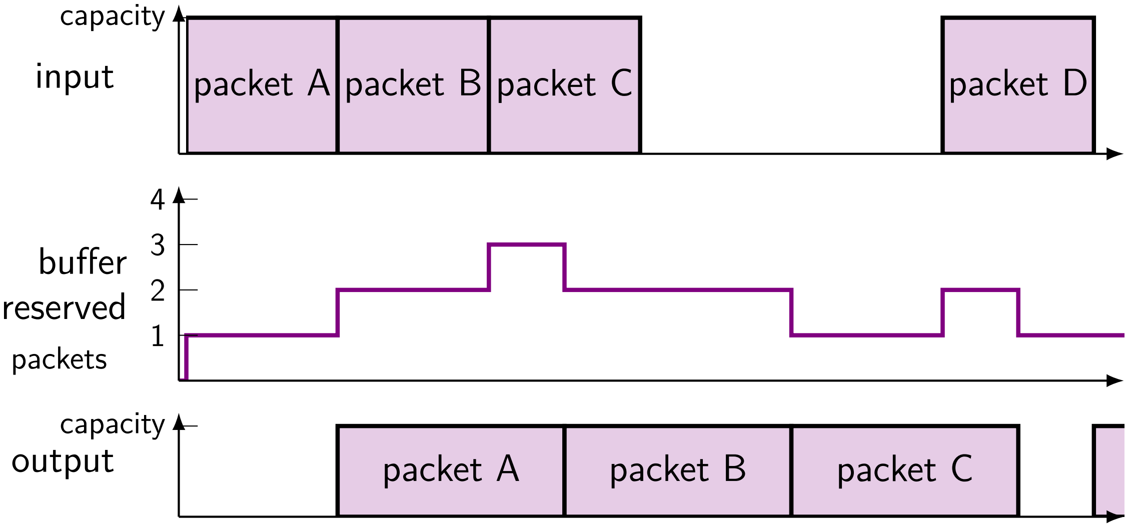
(de)multiplexing



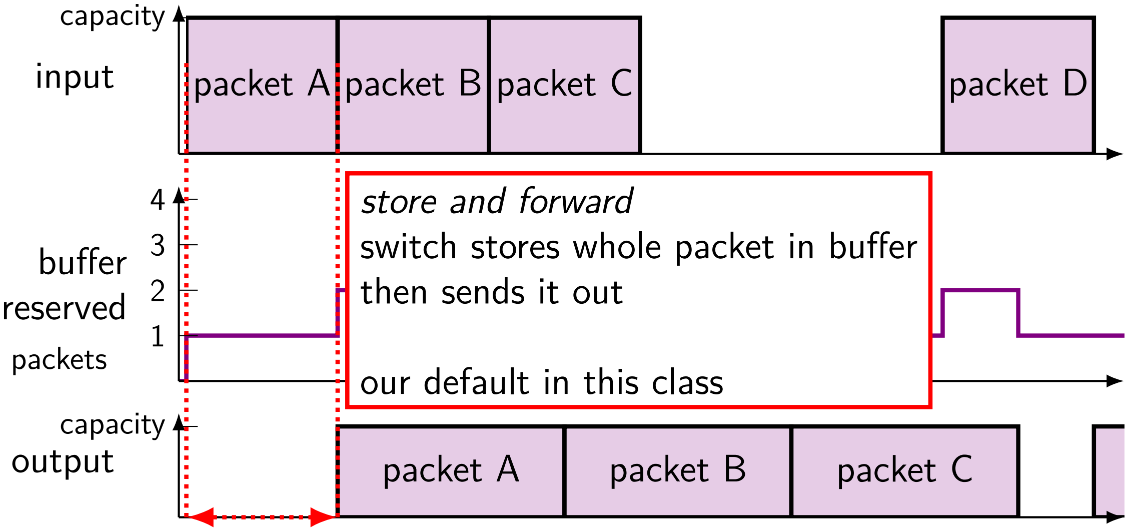
if buffer full, switch must *drop* packets
will happen eventually if overall rate faster than outgoing link

scenario is called *congestion*

buffer usage: fast to slow, store + forward



buffer usage: fast to slow, store + forward

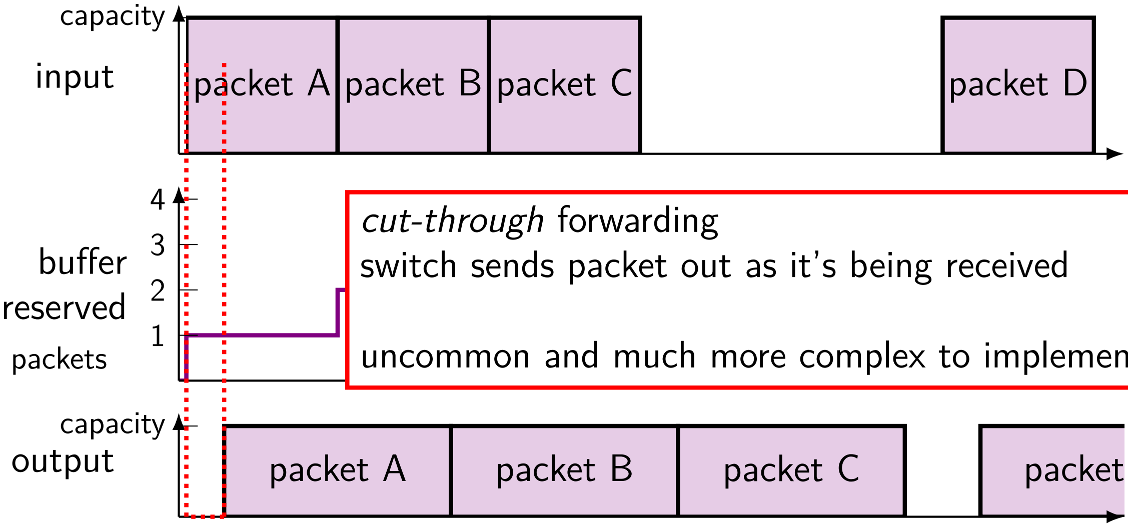


store and forward

switch stores whole packet in buffer then sends it out

our default in this class

buffer usage: fast to slow, cut-through



cut-through forwarding

switch sends packet out as it's being received

uncommon and much more complex to implement

channel abstractions

want to avoid custom network for each application

but applications have different needs

→ multiple application interfaces to networks

common implementation of common patterns

some abstractions

stream

- continuous stream of bytes from one program to another
- ‘connection’ from one program to another

datagrams

- send small messages (*datagrams*)
- each datagram’s destination independently set

remote procedure calls

- make function calls that run on remote machine

remote memory access

- read/write bytes of data in remote memory

…

some abstractions

stream

continuous stream of bytes from one program to another
'connection' from one program to another

datagrams

send small messages (*datagrams*)
each datagram's destination independently set

remote procedure calls

make function calls that run on remote machine

remote memory access

read/write bytes of data in remote memory

…

focus on streams

this class: focus on implementing *streams of bytes*

why?

- most commonly used by applications on the Internet
- many common tasks with other abstractions

some challenges for streams

separating data into pieces network can handle

putting pieces back together

getting network to send piece to correct remote network

getting network to send piece to correct machine

getting machine to send data to correct program

getting pieces into format wires/radio/fiber/etc. can handle

handling transmission errors

some challenges for streams

separating data into pieces network can handle

putting pieces back together

getting network to send piece to correct remote network

getting ne

lots of work! don't want to implement all at once!

getting machine to send data to correct program

getting pieces into format wires/radio/fiber/etc. can handle

handling transmission errors

some challenges for streams

separating data into pieces network can handle

puttin

some parts need to be different for different local networks

getting network to send piece to correct remote network

getting network to send piece to correct machine

getting machine to send data to correct program

getting pieces into format wires/radio/fiber/etc. can handle

handling transmission errors

some challenges for streams

separa

some parts should not concern local network implementors

putting pieces back together

getting network to send piece to correct remote network

getting network to send piece to correct machine

getting machine to send data to correct program

getting pieces into format wires/radio/fiber/etc. can handle

handling transmission errors

some challenges for streams

separating

some parts should be same for different abstraction

putting pieces back together

getting network to send piece to correct remote network

getting network to send piece to correct machine

getting machine to send data to correct program

getting pieces into format wires/radio/fiber/etc. can handle

handling transmission errors

layered model

networking implemented in ‘layers’

upper layers implemented by making calls to lower layers

example: network implements ‘send data to (remote) machine’ function (“network layer”)

stream implementation calls this to implement ‘send stream to remote application’

OSI model

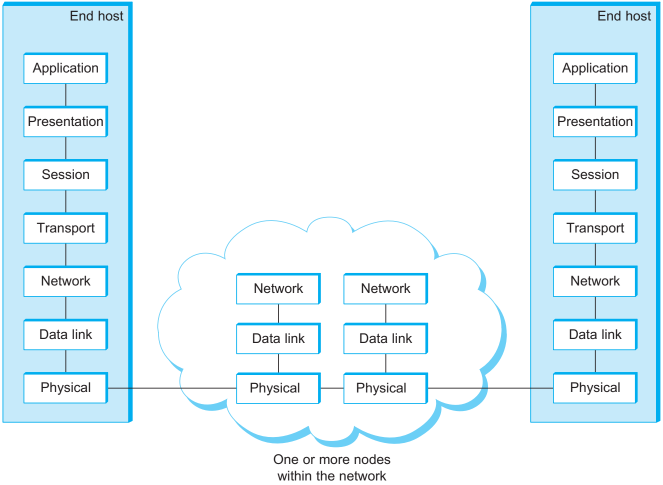
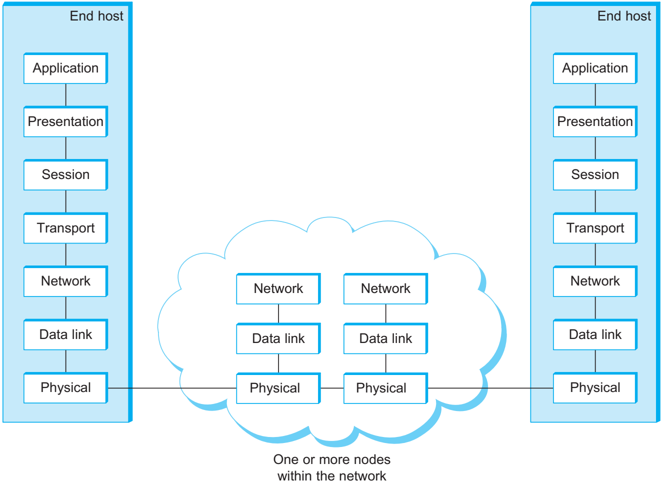


Figure 13 of Chapter 1 of Computer Networks: A Systems Approach (6th ed) (Peterson and Davie)

OSI model

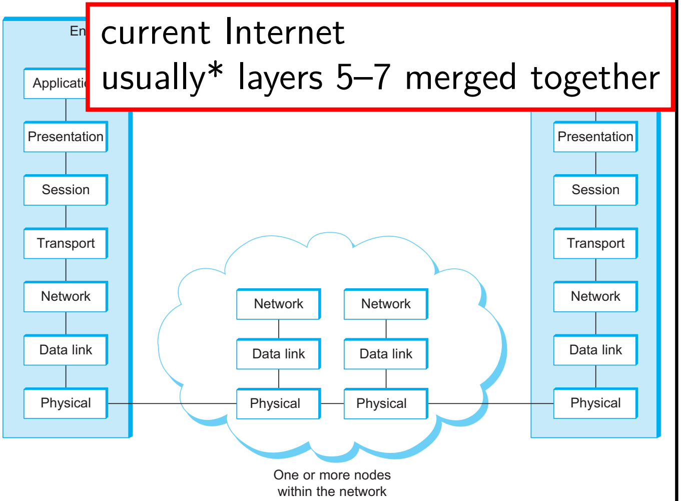


(7) application:
what requests/etc.
(6) presentation:
data format
(5) session:
manage group of streams
(4) transport:
streams of data
(3) network:
message to correct network
(2) data link:
message → bits
message to correct machine
(1) physical:
send bits/...

Figure 13 of Chapter 1 of Computer Networks: A Systems Approach (6th ed) (Peterson and Davie)

OSI model

current Internet
usually* layers 5–7 merged together



(7) application:
what requests/etc.

(6) presentation:
data format

(5) session:
manage group of streams

(4) transport:
streams of data

(3) network:
message to correct network

(2) data link:
message → bits
message to correct machine

(1) physical:
send bits/...

Figure 13 of Chapter 1 of Computer Networks: A Systems Approach (6th ed) (Peterson and Davie)

OSI model

standardized by ISO (International Standards Organization)

full set of protocols...

- file transfer, message sending, directory lookups ...

that were often implemented and sometimes used...

but mostly lost out to IETF-standardized Internet protocols

- Internet Engineering Task Force

OSI influence (1)

term ‘layer 7’, ‘layer 4’, ‘layer 3’, etc. almost always refer to OSI model

…even though most of Internet does not follow it

early Internet protocols predate OSI

OSI influence (2)

are a lot of Internet protocols influenced by OSI protocols

OSI's DAP (directory access protocol)
- adapted into IETF's LDAP (lightweight directory access protocol)

OSI presentation layer ASN.1 used in...
- telephony (between telephone companies)
- inter-bank messaging
- lots of cryptography-related protocols
- ...

OSI's routing protocol IS-IS still common in large Internet-connected networks
- (adapted to work alongside IETF protocols)

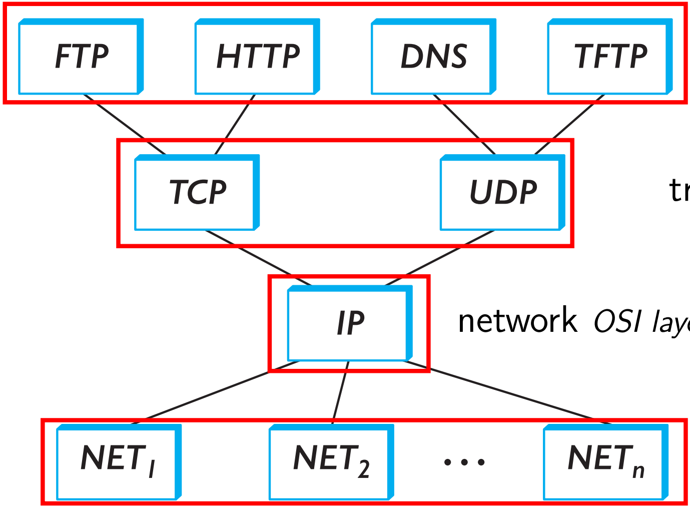
Internet layers

OSI layer	name	examples	purpose
7	application	HTTP, SSH, SMTP, DNS, …	application-defined meanings
4	transport	TCP, UDP, …	reach correct program, reliablity/streams
3	network	IPv4, IPv6, …	reach correct machine (across networks)
2	link	Ethernet, Wi-Fi, …	coordinate shared wire/radio
1	physical	…	encode bits for wire/radio

Internet layers

OSI layer	name	examples	purpose
7	application	HTTP, SSH, SMTP, DNS, …	application-defined meanings
4	transport	TCP, UDP, …	reach correct program, reliablity/streams
3	network	IPv4, IPv6, …	reach correct machine (across networks)
2	link	Ethernet, Wi-Fi, …	coordinate shared wire/radio
1	physical	…	encode bits for wire/radio

Internet protocols and layers (non-exhaustive)



FTP HTTP DNS TFTP

application *OSI layer 7*

TCP UDP

transport *OSI layer 4*

IP

network *OSI layer 3*

NET_1 NET_2 ... NET_n

data link *OSI layer 2*

Figure 14 of Chapter 1 of Computer Networks: A Systems Approach (6th ed) (Peterson and Davie)

Internet protocols and layers (non-exhaustive)

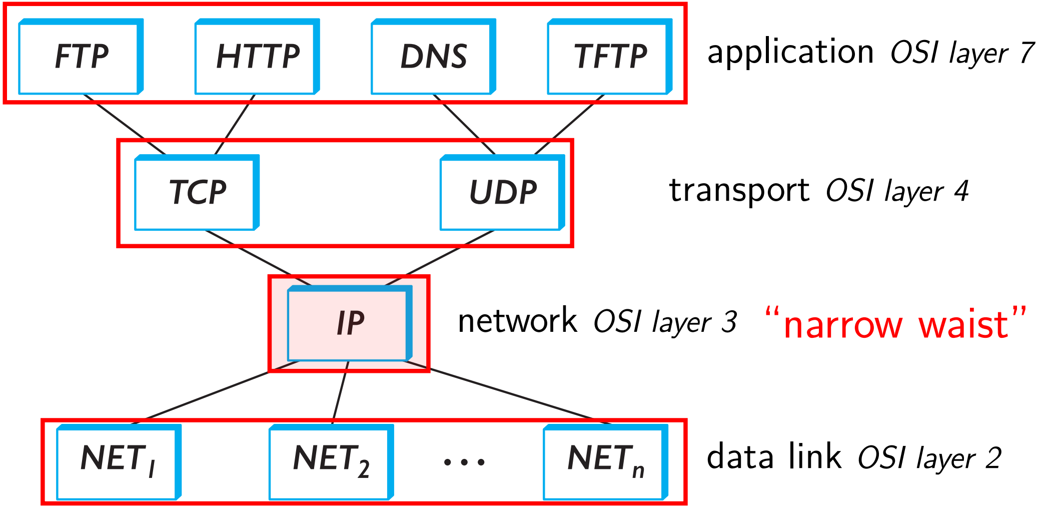


Figure 14 of Chapter 1 of Computer Networks: A Systems Approach (6th ed) (Peterson and Davie)

fuzzy layers (1)

ICMP (Internet Control Message Protocol)...

implemented using a network layer...

so seems like a transport layer protocol?

fuzzy layers (1)

ICMP (Internet Control Message Protocol)...

implemented using a network layer...

- so seems like a transport layer protocol?

used to send errors/control messages about routing...

- routing is the network layer's job
- so ICMP is part of network layer?

fuzzy layers (1)

ICMP (Internet Control Message Protocol)...

implemented using a network layer...

- so seems like a transport layer protocol?

used to send errors/control messages about routing...

- routing is the network layer's job
- so ICMP is part of network layer?

I think saying network layer is probably better...

but we're not going to be picky about it

fuzzy layers (2)

TLS (Transport Control Protocol)...

implemented on top of TCP...

- so seems like a application layer protocol?

fuzzy layers (2)

TLS (Transport Control Protocol)...

implemented on top of TCP...

- so seems like a application layer protocol?

used to send other application layer protocols

- so maybe a transport layer?
- or presentation layer?

I'll call it an application layer...

‘extra’ layers

layer terminology doesn’t always work cleanly

- often “extra” layers in practice

e.g. HTTPS:

- HTTP (app layer) on TLS (another app layer) on TCP (network) on ...

e.g. DNS over HTTPS:

- DNS (app layer) on HTTP on on TLS on TCP on ...

e.g. SFTP:

- SFTP (app layer??) on SSH (another app layer) on TCP on ...

e.g. HTTP over OpenVPN:

- HTTP on TCP on IP on OpenVPN on UDP on different IP on ...

‘extra’ layers

layer terminology doesn’t always work cleanly

  often “extra” layers in practice

e.g. HTTPS:

  HTTP (app layer) on TLS (another app layer) on TCP (network) on ...

e.g. DNS over HTTPS:

  DNS (app layer) on HTTP on on TLS on TCP on ...

e.g. SFTP:

  SFTP (app layer??) on SSH (another app layer) on TCP on ...

e.g. HTTP over OpenVPN:

  HTTP on TCP on IP on OpenVPN on UDP on different IP on ...

protocols usually over HTTP

SOAP (Simple Object Access Protocol) — messaging/remote procedure calls

gRPC (originally form Google) — remote procedure calls

HLS (HTTP Live Streaming) — video streaming

DASH (Dynamic Adaptive Streaming over HTTP) — video streaming

…

end-to-end argument

Saltzer, Reed, Clark, “End-to-End Arguments in System Design”

“The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible. (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.)”

end-to-end argument

Saltzer, Reed, Clark, “End-to-End Arguments in System Design”

“The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible. (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.)”

example: reliable file transfer

want to make sure correct data transferred

want to protect against:

- error in hardware/software on sending machine reading file
- bits being flipped in memory on forwarding machine
- communication system flipping bits in data
- hosts crashing during communication

example: reliable file transfer

want to make sure correct data transferred

want to protect against:

- error in hardware/software on sending machine reading file
- bits being flipped in memory on forwarding machine
- communication system flipping bits in data
- hosts crashing during communication

communication system can't help a lot of these things

example: reliable file transfer

want to make sure correct data transferred

want to protect against:

- error in hardware/software on sending machine reading file
- bits being flipped in memory on forwarding machine
- communication system flipping bits in data
- hosts crashing during communication

communication system can't help a lot of these things

authors experienced router with bad memory/processor

solution: end-to-end checks

want reliable transfer: compare final files (with hash or similar)

“end-to-end” — doesn’t care what middle systems do

end-to-end argument

Saltzer, Reed, Clark, “End-to-End Arguments in System Design”

“The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible. (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.)”

end-to-end in practice

"narrow waist" of IP doesn't provide many gaurnetees

- no gaurentees about reliable transmission, duplicate suppression, message order, ...

but try to provide good service ("best effort")

in design: typically middle systems won't know/care about what's forwarded

- but many exceptions

backup slides