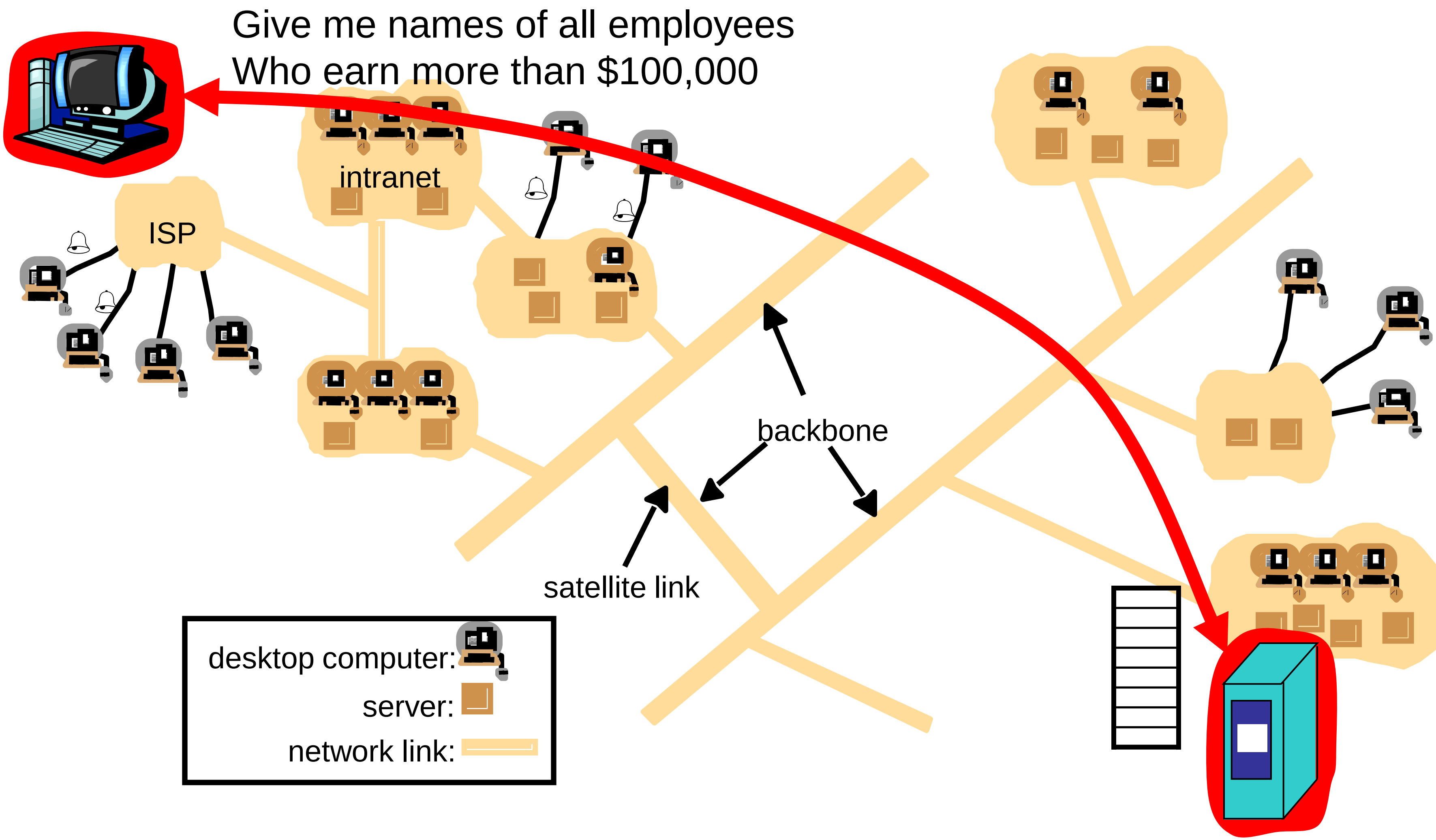
Module 2 – Overview of Computer Networks

Networks and Communication



Internet Connectivity

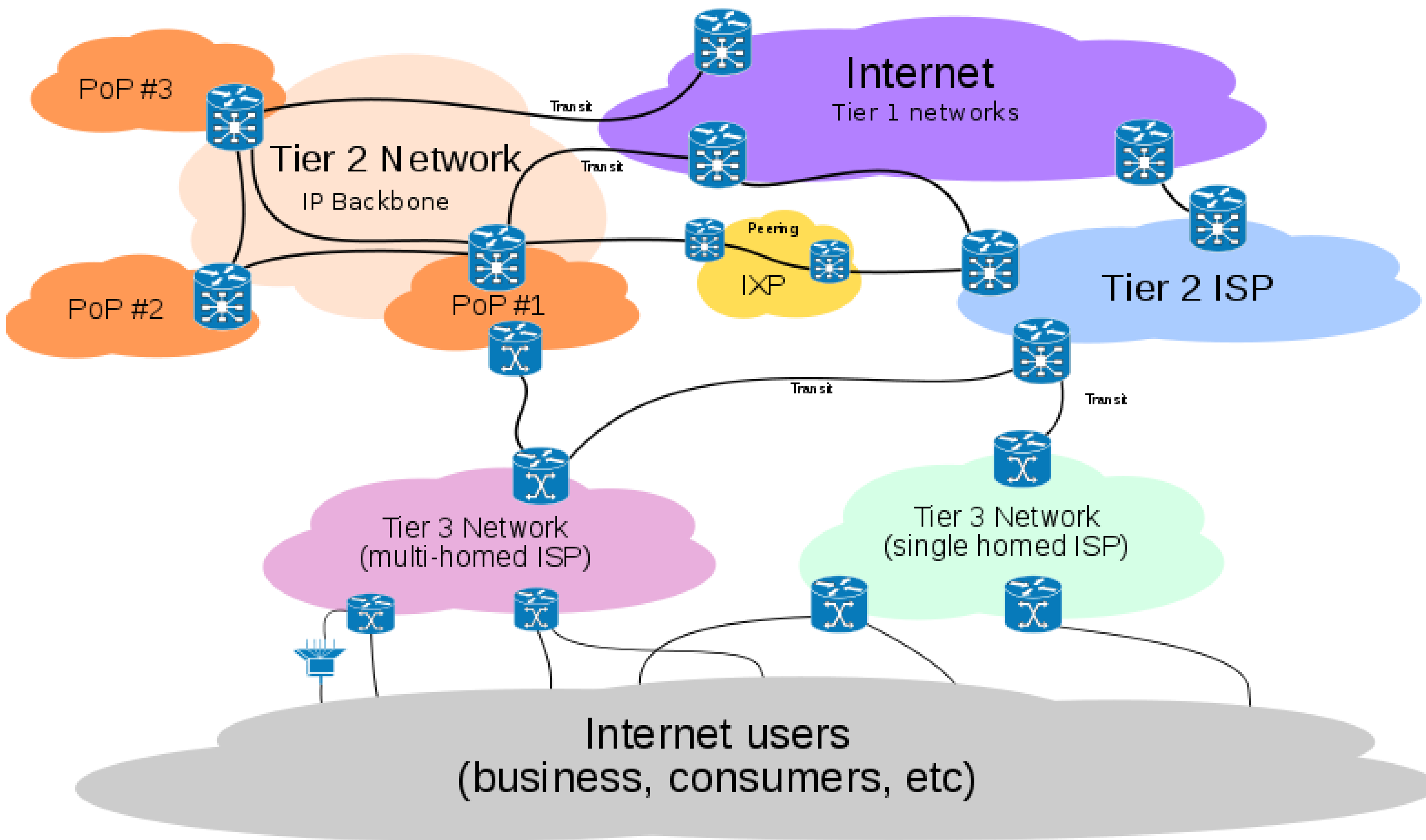


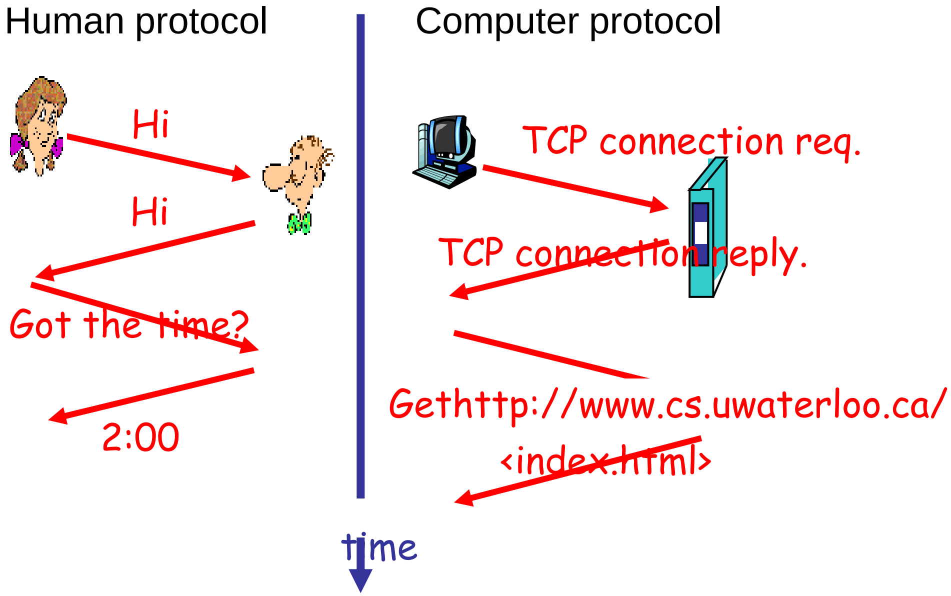
Figure from Wikipedia

Issues

- How do the request and response get transmitted between the requestor and the server?
  - Protocols to facilitate communication
  - Moving the request and reply messages through the network

What is a protocol?

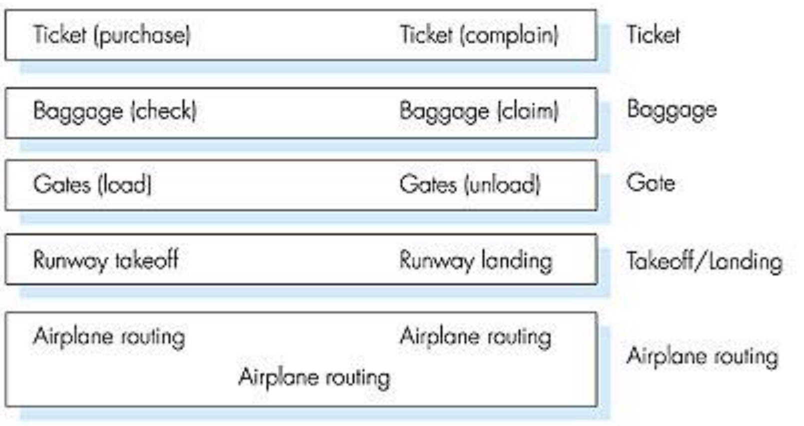
- A protocol defines the format and the order of messages sent and received among network entities, and the actions taken on message transmission and receipt
- Human protocols:
  - “What’s the time?”
  - “I have a question”
  - introductions
- Network protocols:
  - machines rather than humans
  - all communication activity in Internet governed by protocols



Layered Architecture

- Can describe an airline system sequentially
  - Lacks structure
  - Any changes to one component requires re-describing the whole procedure

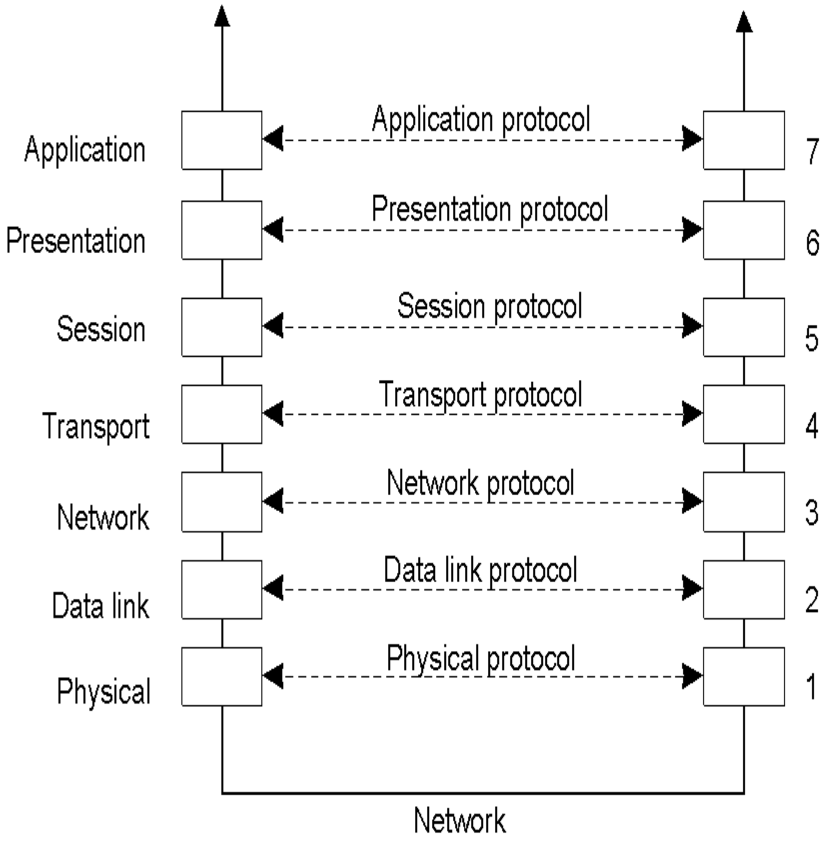
- We can instead recognize that there complementary functions at each end
  - Can divide the airline functionality into layers
  - Allows us to discuss a well-defined, specific part of a large and complex system.



Example taken from *Computer Networks A Top-Down Approach*

Layered Protocols

- Divide functionality to different layers and let each layer provide one function.
- ISO OSI Layered network architecture (International Organization for Standardization, Open Systems Interconnections)
  - Function layers
  - Interfaces
  - Protocols at each layer



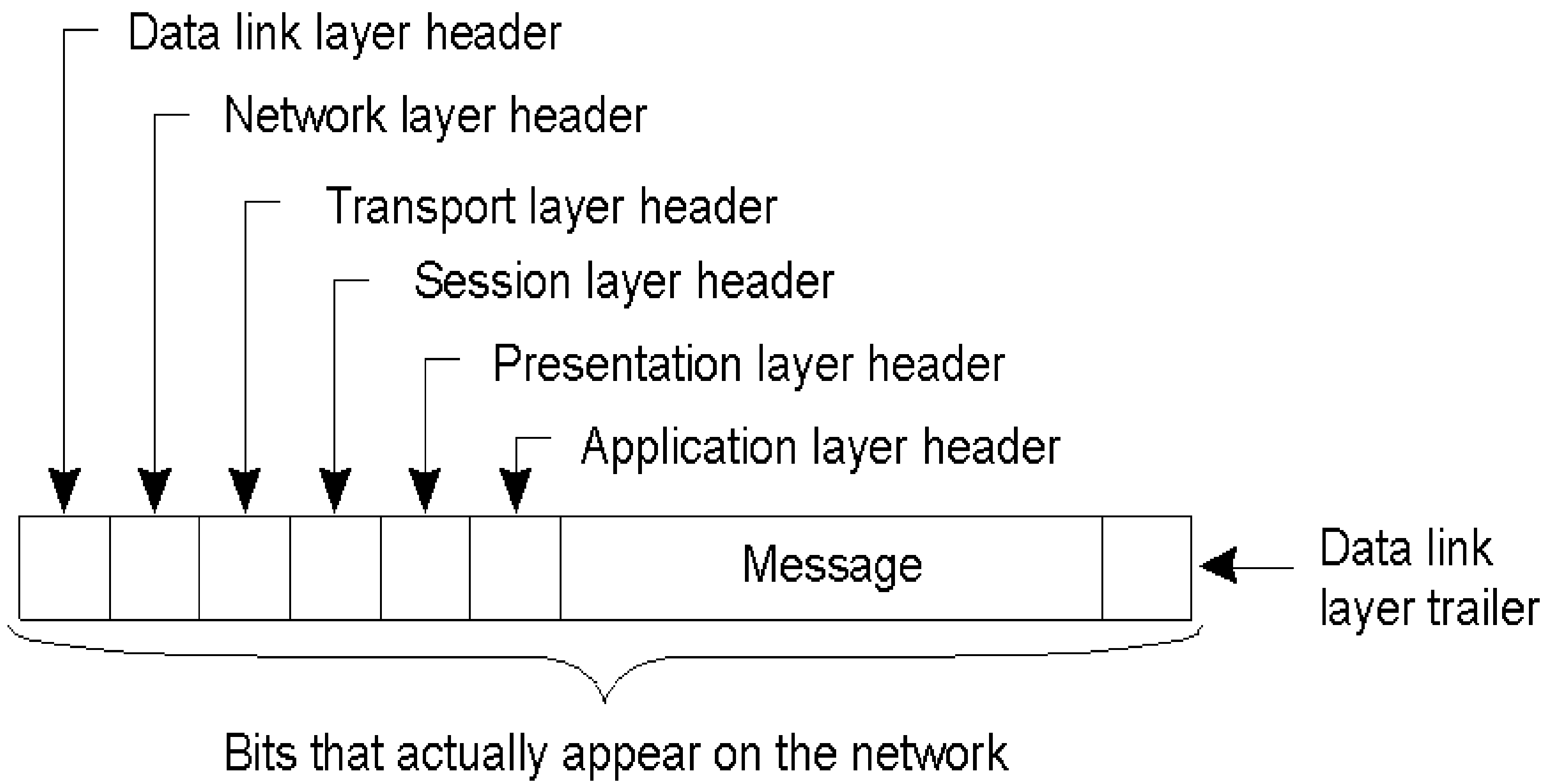
From Tanenbaum and van Steen, Distributed Systems: Principles and Paradigms

OSI Protocol Summary

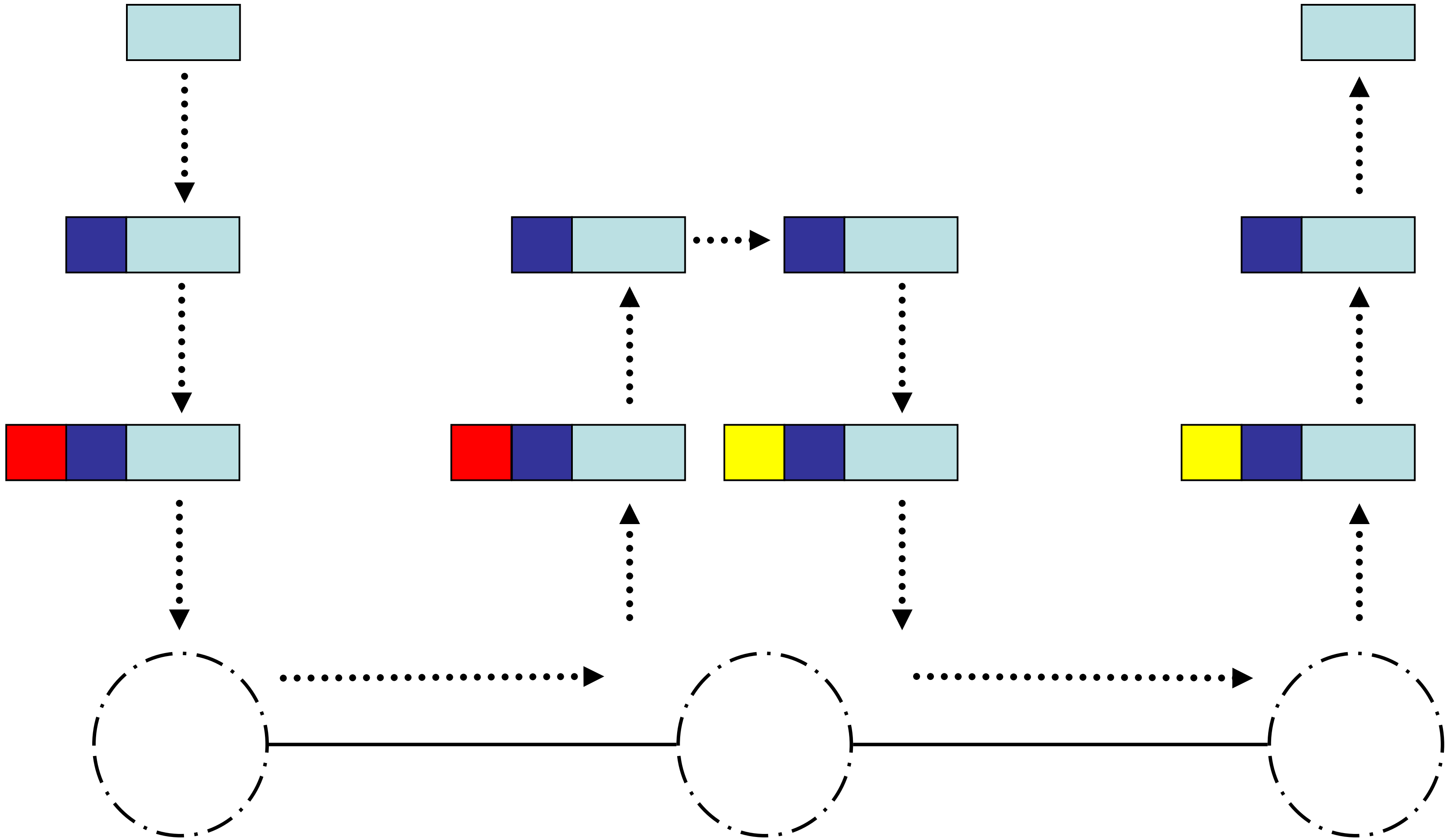
Layer	*Description*	*Examples*
Application	Protocols that are designed to meet the communication requirements of specific applications, often defining the interface to a service.	HTTP, FTP, SMTP, CORBA IIOP
Presentation	Protocols at this level transmit data in a network representation that is independent of the representations used in individual computers, which may differ. Encryption is also performed in this layer, if required.	Secure Sockets (SSL),CORBA Data Rep.
Session	At this level reliability and adaptation are performed, such as detection of failures and automatic recovery.	
Transport	This is the lowest level at which messages (rather than packets) are handled. Messages are addressed to communication ports attached to processes, Protocols in this layer may be connection-oriented or connectionless.	TCP, UDP
Network	Transfers data packets between computers in a specific network. In a WAN or an internetwork this involves the generation of a route passing through routers. In a single LAN no routing is required.	IP, ATM virtual circuits
Data link	Responsible for transmission of packets between nodes that are directly connected by a physical link. In a WAN transmission is between pairs of routers or between routers and hosts. In a LAN it is between any pair of hosts.	Ethernet MAC, ATM cell transfer, PPP
Physical	The circuits and hardware that drive the network. It transmits sequences of binary data by analogue signalling, using amplitude or frequency modulation of electrical signals (on cable circuits), light signals (on fibre optic circuits) or other electromagnetic signals (on radio and microwave circuits).	Ethernet base- band signalling, ISDN

From Coulouris, Dollimore and Kindberg, Distributed Systems: Concepts and Design, 3rd ed.

Message Format



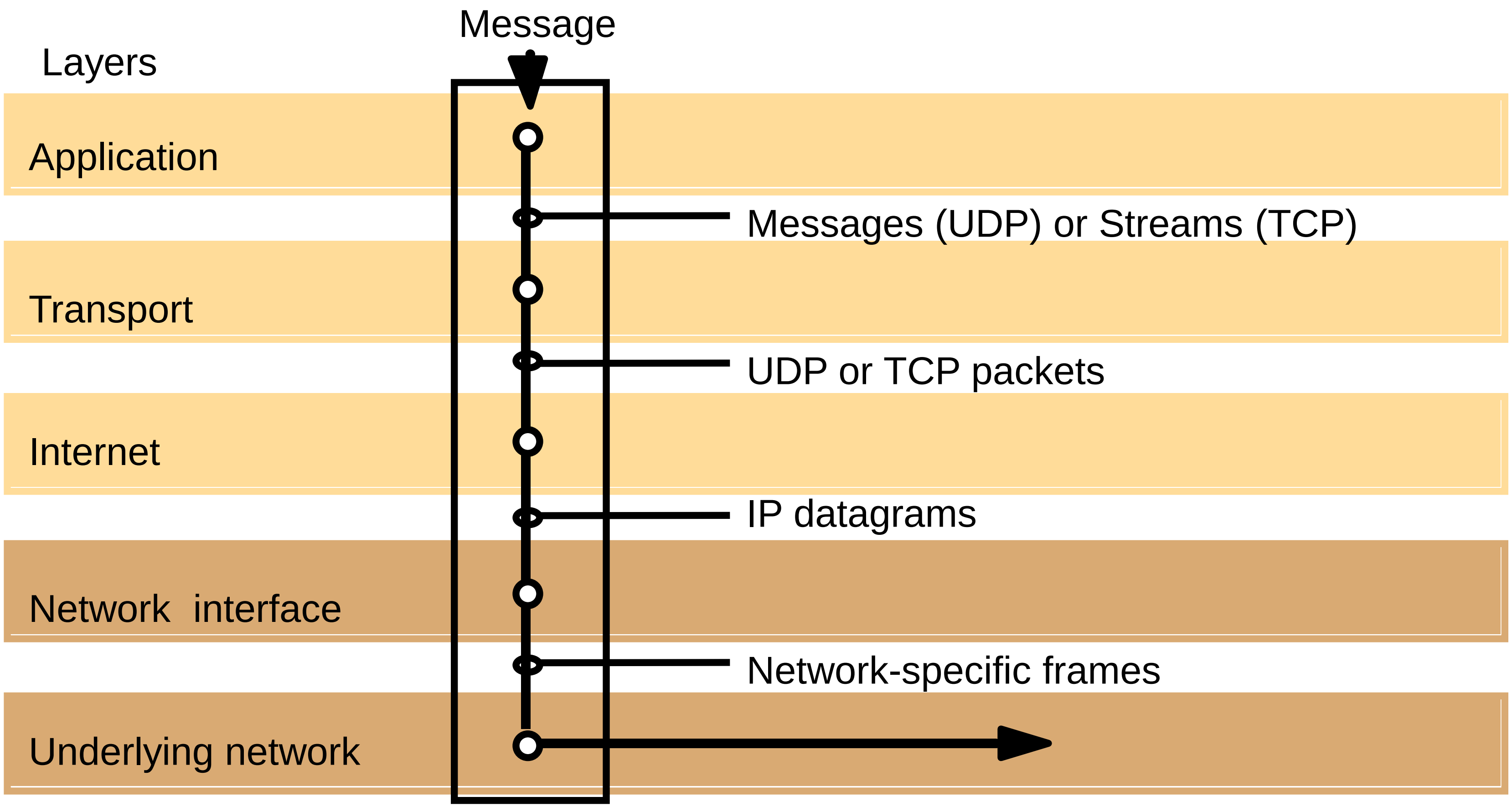
Message Transmission (Assume only 3 layers)



Internet Protocols

Application (7)	FTP Telnet NFS SMTP HTTP …					
Transport (4)	TCP			UDP		
Network (3)	IP					
Data Link (2) Physical (1)	X.25	Ethernet	Packet Radio	ATM	FDDI	…

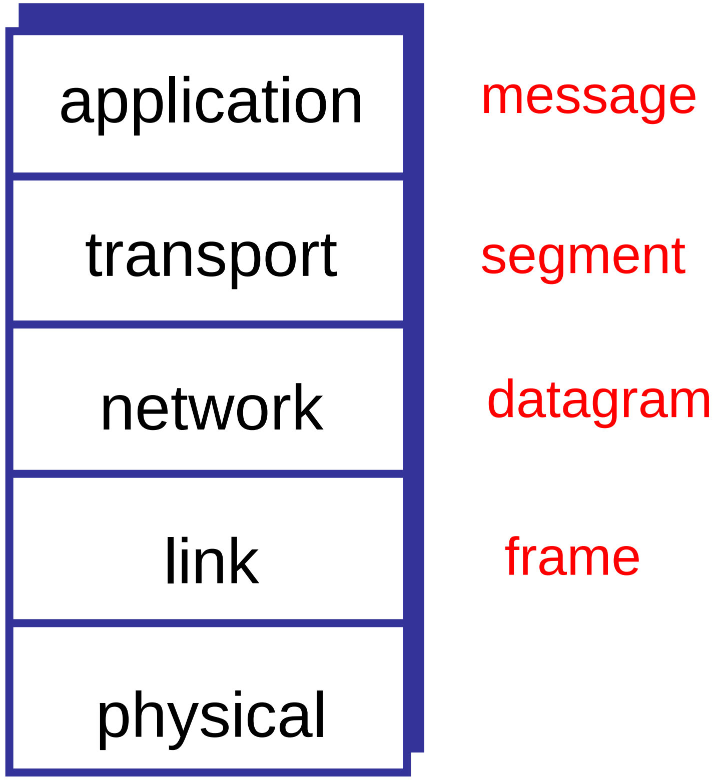
TCP/IP Layers



From Coulouris, Dollimore and Kindberg, Distributed Systems: Concepts and Design, 3rd ed.

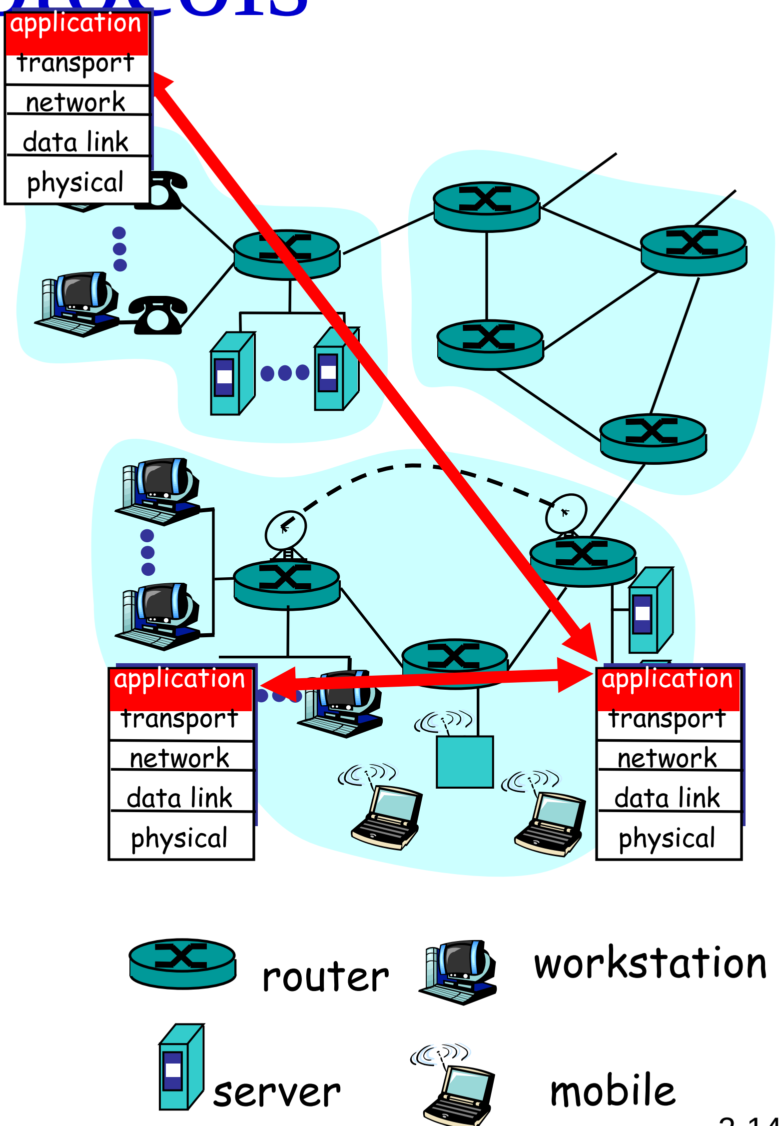
Internet Protocol Stack

- Application: supporting network applications
  - ftp, smtp, http
- Transport: host-host data transfer
  - tcp, udp
- Network: routing of datagrams from source to destination
  - ip, routing protocols
- Link: data transfer between neighboring network elements
  - ppp, ethernet
- Physical: bits “on the wire”



Applications and Application-Layer Protocols

- Application: communicating, distributed processes
  - running in network hosts in "user space"
  - exchange messages to implement app
  - e.g., email, file transfer, the Web
- Application-layer protocols
  - one "piece" of an app
  - define messages exchanged by apps and actions taken
  - use services provided by lower layer protocols



Application-Layer Protocols (cont).

- API: application programming interface
  - Defines interface between application and transport layer
  - socket: Internet API
    - → two processes communicate by sending data into socket, reading data out of socket
- What transport services does an application need?
  - Data loss
    - → some apps (e.g., audio) can tolerate some loss
    - → other apps (e.g., file transfer, telnet) require 100% reliable data transfer
  - Bandwidth
    - → some apps (e.g., multimedia) require a minimum amount of bandwidth to be “effective”
    - → other apps (“elastic apps”) make use of whatever bandwidth they get
  - Timing
    - → some apps (e.g., Internet telephony, interactive games) require low delay to be “effective”

Transport Service Requirements of Common Applications

Application	Data loss	Bandwidth	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5Kb-1Mb video:10Kb-5Mb	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few Kbps up	yes, 100's msec
financial apps	no loss	elastic	yes and no

Internet Transport Protocol Services

TCP service:

- *connection-oriented:* setup required between client, server
- *reliable transport* between sending and receiving process
  - Receipt of packets returned as acknowledgements
- *flow control:* sender won't overwhelm receiver
- *congestion control:* throttle sender when network overloaded
- *does not provide:* timing, minimum bandwidth guarantees

UDP service:

- unreliable data transfer between sending and receiving process
- does not provide: connection setup, reliability, flow control, congestion control, timing, or bandwidth guarantee

Flow Control

- Both flow control and congestion control affect the sending rate
- Flow control aims to eliminate the possibility of overflowing the receiver's buffer.
- Receiver keeps track of two variables:
  - LastByteReceived and LastByteRead
  - LastByteReceived – LastByteRead <= ReceiveBufferSize
- Notifies the sender of its receive window size (rwnd)
- Sender keeps track of two variables
  - LastByteSent and LastByteAcked
  - Delays sending of packets to ensure that
    - → LastByteSent – LastByteAcked <= rwnd

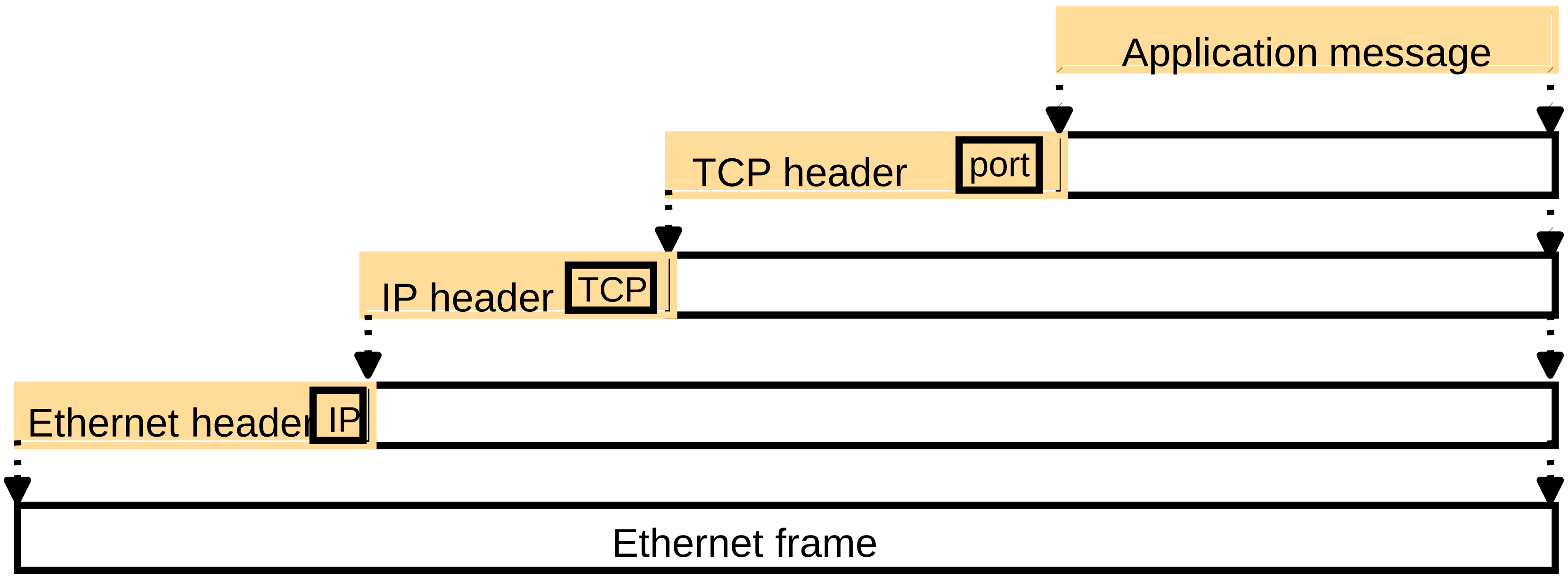
Congestion Control

- Traffic from multiple paths that share the same routers/links can overwhelm capacity within the network.
- In TCP, senders limit the rate at which it sends traffic as a function of the perceived network congestion.
- If there is little to no perceived congestion
  - Additive increase in sending rate
- If there is congestion
  - Multiplicative decrease in sending rate
- Congestion is detected via:
  - Lost packets

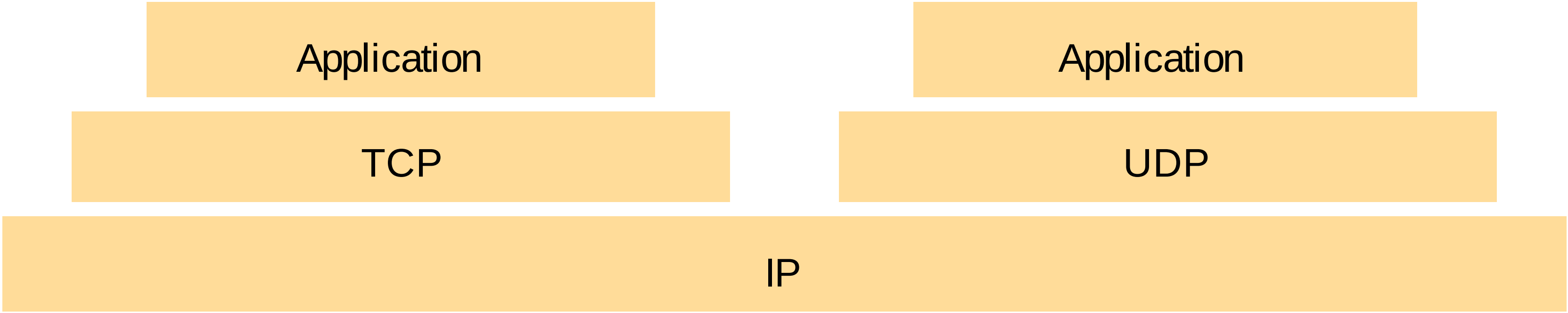
Internet Apps: Their Protocols and Transport Protocols

Application	Application layer protocol	Underlying transport protocol
e-mail	smtp [RFC 821]	TCP
remote terminal access	telnet [RFC 854]	TCP
Web	http [RFC 2068]	TCP
file transfer	ftp [RFC 959]	TCP
streaming multimedia	proprietary (e.g. RealNetworks)	TCP or UDP
remote file server	NFS	TCP or UDP
Internet telephony	proprietary (e.g., Vocaltec)	typically UDP

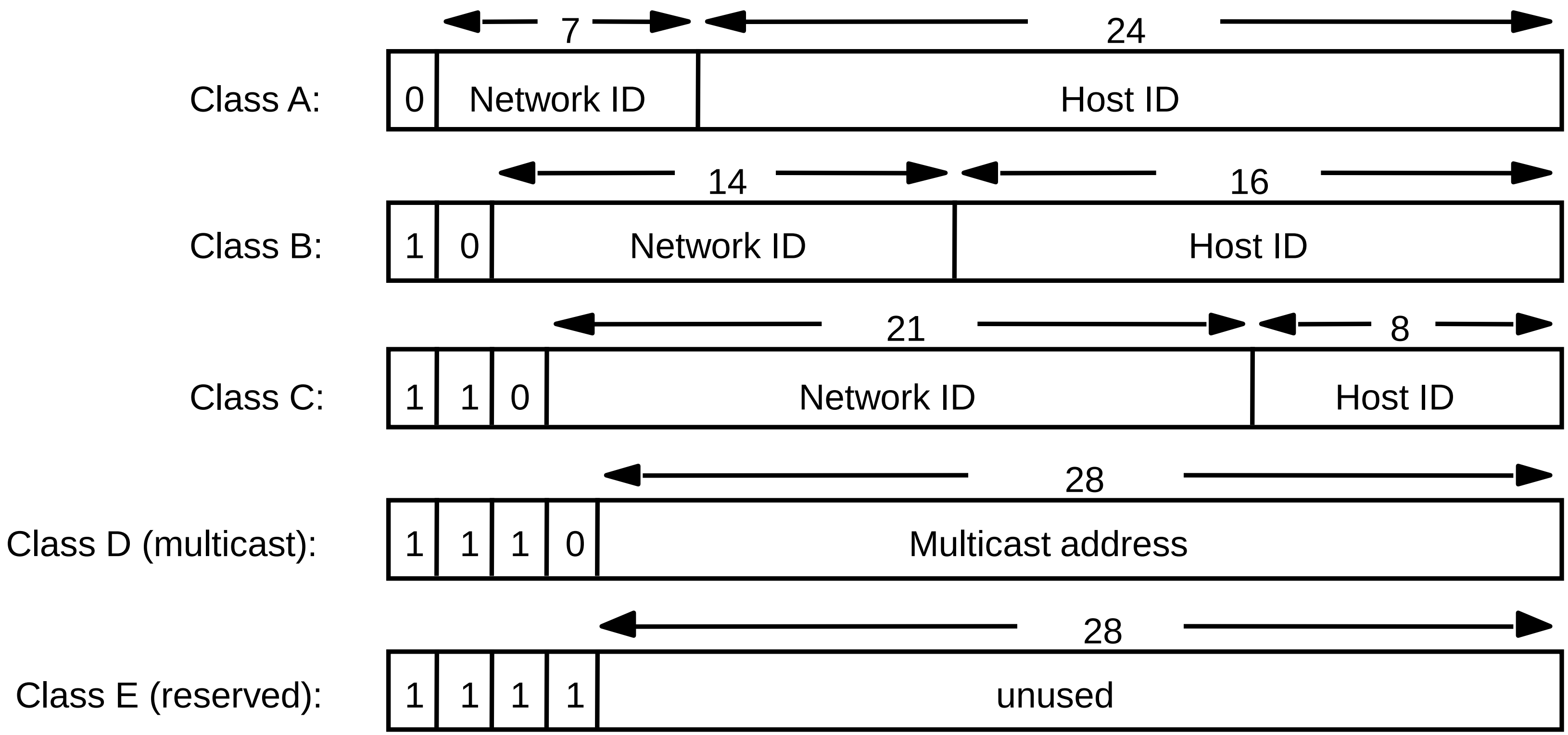
TCP Message Encapsulation over an Ethernet



The programmer's conceptual view of a TCP/IP Internet



Internet Address Structure



From Coulouris, Dollimore and Kindberg, Distributed Systems: Concepts and Design, 3rd ed.

Decimal Representation of Internet Addresses

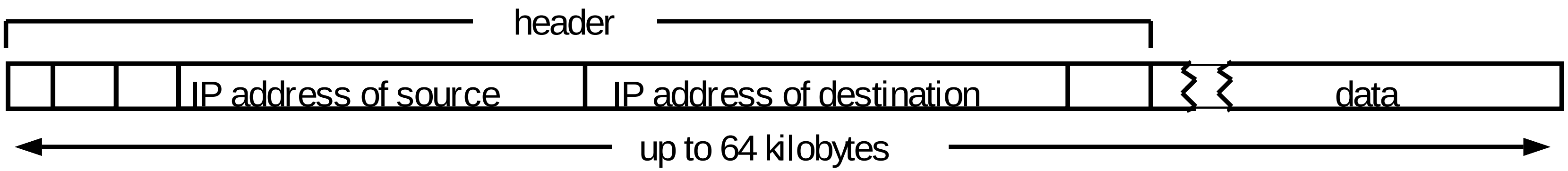
	octet 1	octet 2	octet 3		Range of addresses
	Network ID	Host ID			
Class A:	1 to 127	0 to 255	0 to 255	0 to 255	1.0.0.0 to 127.255.255.255
	Network ID		Host ID		
Class B:	128 to 191	0 to 255	0 to 255	0 to 255	128.0.0.0 to 191.255.255.255
	Network ID			Host ID	
Class C:	192 to 223	0 to 255	0 to 255	1 to 254	192.0.0.0 to 223.255.255.255
	Multicast address				
Class D (multicast):	224 to 239	0 to 255	0 to 255	1 to 254	224.0.0.0 to 239.255.255.255
Class E (reserved):	240 to 255	0 to 255	0 to 255	1 to 254	240.0.0.0 to 255.255.255.255

From Coulouris, Dollimore and Kindberg, Distributed Systems: Concepts and Design, 3rd ed.

Classless Interdomain Routing (CIDR)

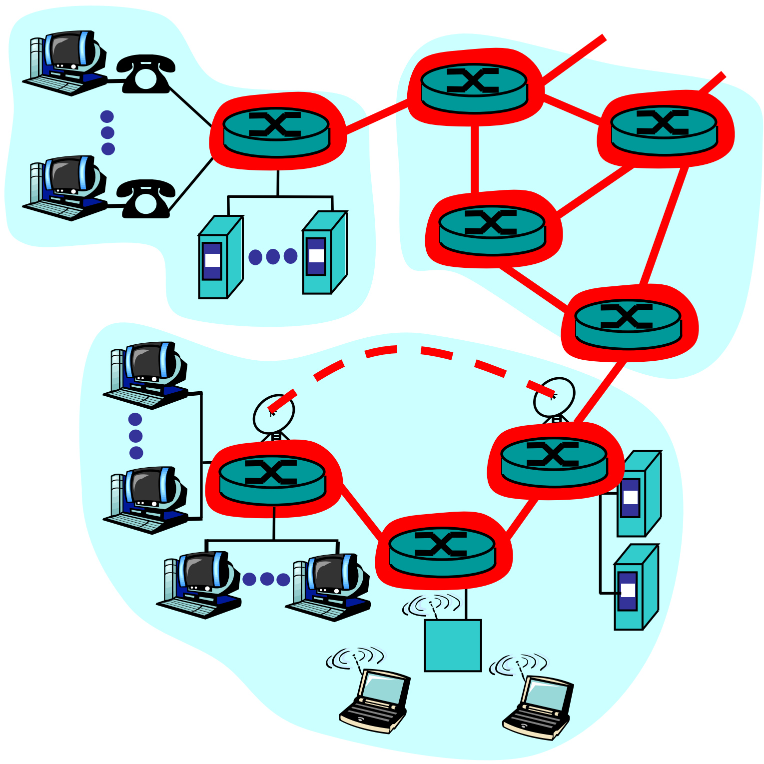
- Breaking the IP address space into fixed size classes is space inefficient
- CIDR generalizes the notion of subnet addressing
  - 32-bit address is divided into two parts
  - a.b.c.d/x, where x indicates the number of bits in the first part of the address
    - → The x most significant bits signify the network portion (prefix)
    - → The remaining bits are for identifying individual hosts
  - IP addresses are managed by ICANN (Internet Corporation for Assigned Names and Numbers)
- CIDR creates more subnets
  - Each subnet needs to be advertised for routing
  - We will see later that this increases the routing table size

IP Packet Layout



How to Transfer Data Through the Network

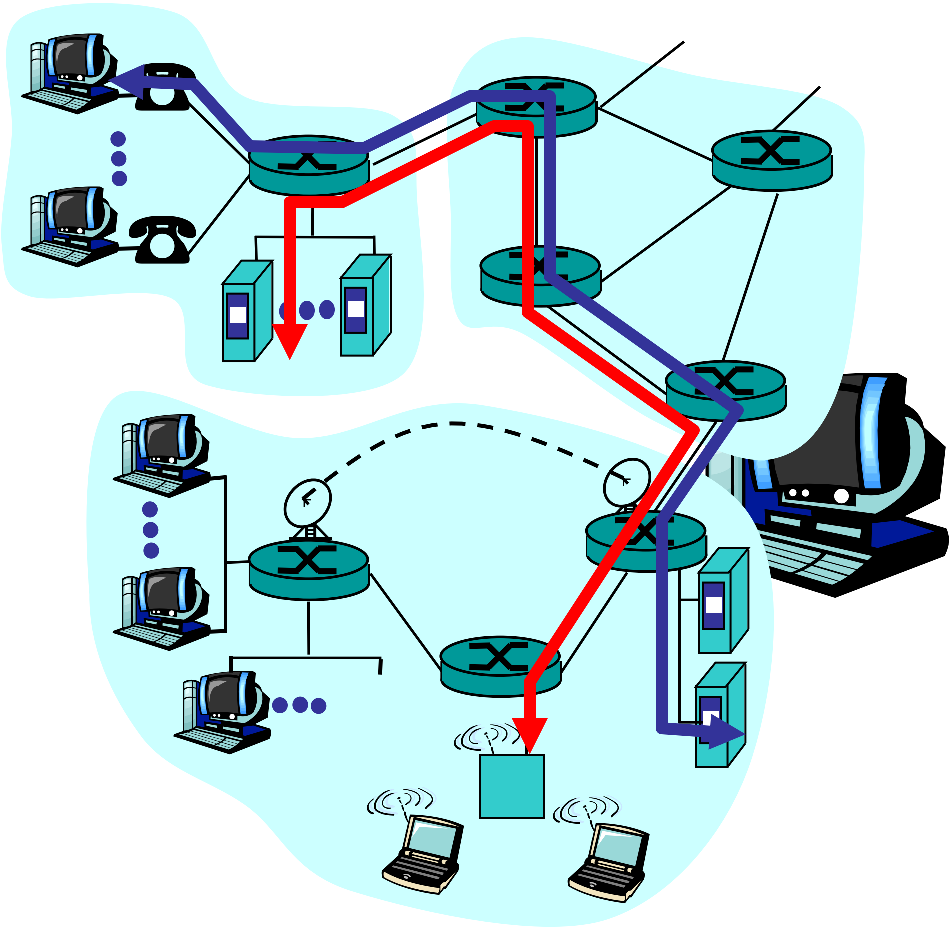
- Network is a mesh of interconnected routers
- Two ways of setting up a connection between two computers
  - circuit switching: dedicated circuit per call: telephone net
  - packet-switching: data sent thru net in discrete “chunks”



Circuit Switching

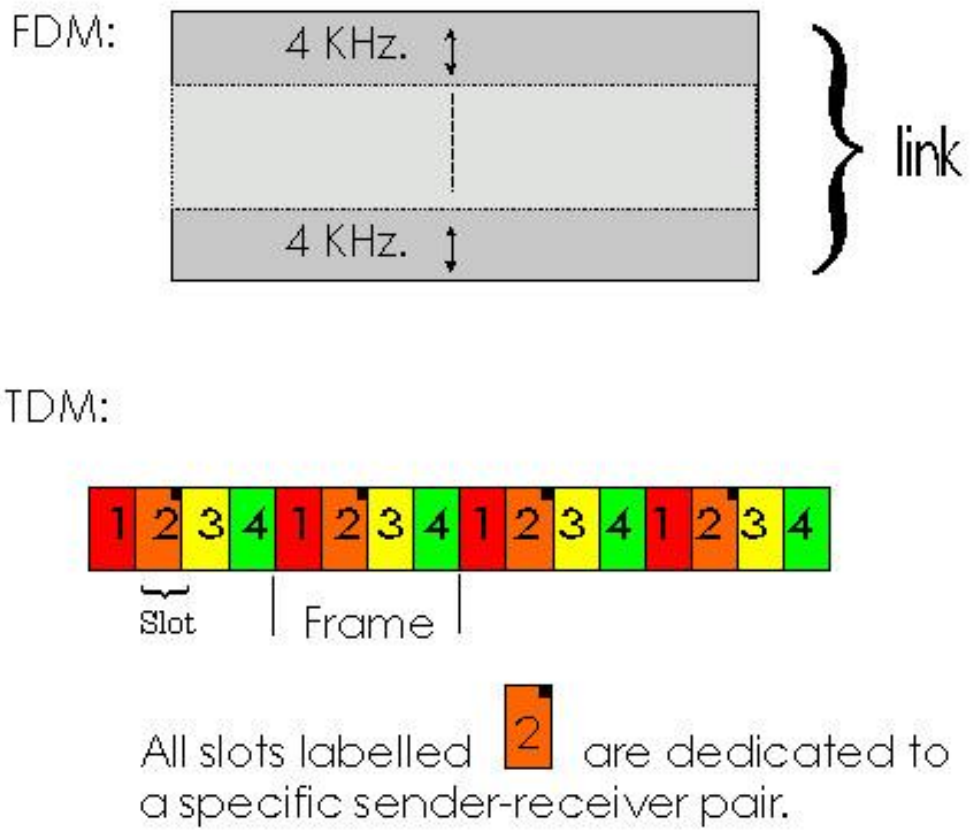
End-end resources reserved for "call"

- link bandwidth, switch capacity
- dedicated resources: no sharing
- circuit-like (guaranteed) performance
- call setup required



Circuit Switching

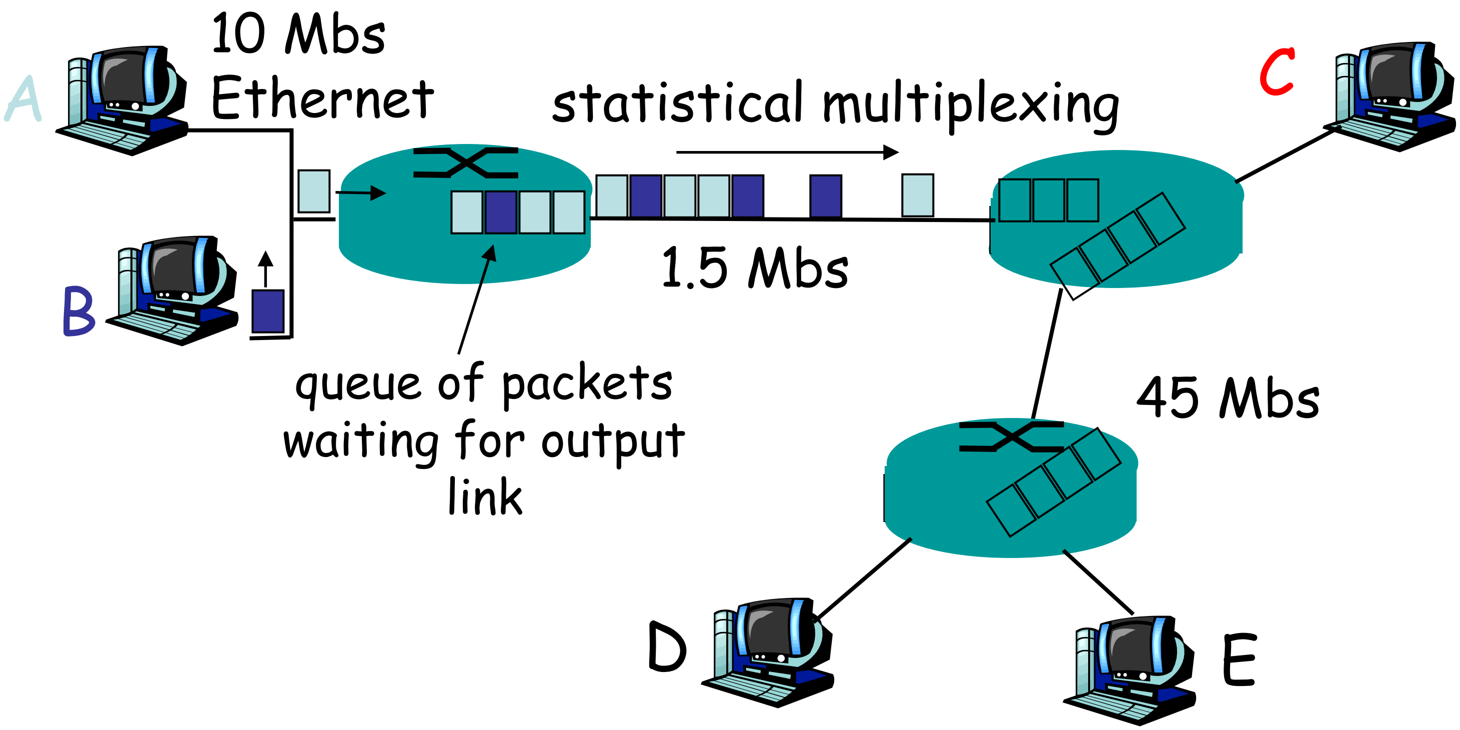
- Network resources (e.g., bandwidth) divided into "pieces"
  - pieces allocated to calls
  - resource piece idle if not used by owning call (no sharing)
  - dividing link bandwidth into "pieces"
    - → frequency division
    - → time division



Packet Switching

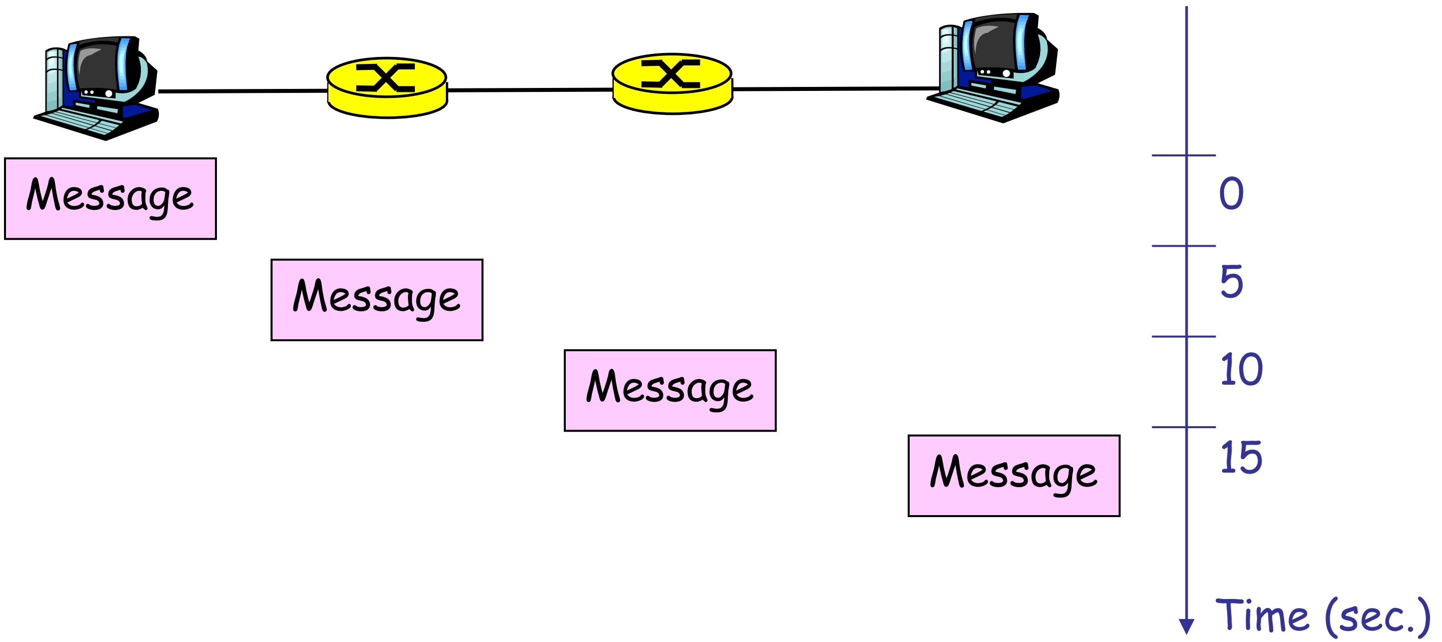
- Each end-end data stream divided into packets
  - User A, B packets share network resources
  - Each packet uses full link bandwidth
  - Resources used as needed
- Resource contention:
  - Aggregate resource demand can exceed amount available
  - Congestion: packets queue, wait for link use
- Store and forward: packets move one hop at a time
  - Transmit over link
  - Wait turn at next link

Packet Switching



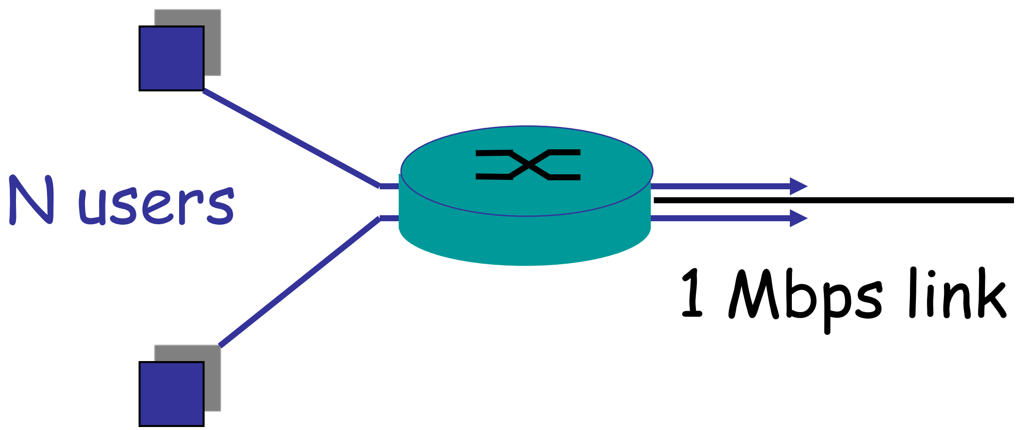
Message Switching

- Message switching = Packet-switching without segmentation
- Packet switching: Store and Forward
- Message remains intact as it traverses the network



Packet Switching vs Circuit Switching

- Packet switching allows more users to use network!
- 1 Mbit link; each user:
  - 100Kbps when "active"
  - active 10% of time



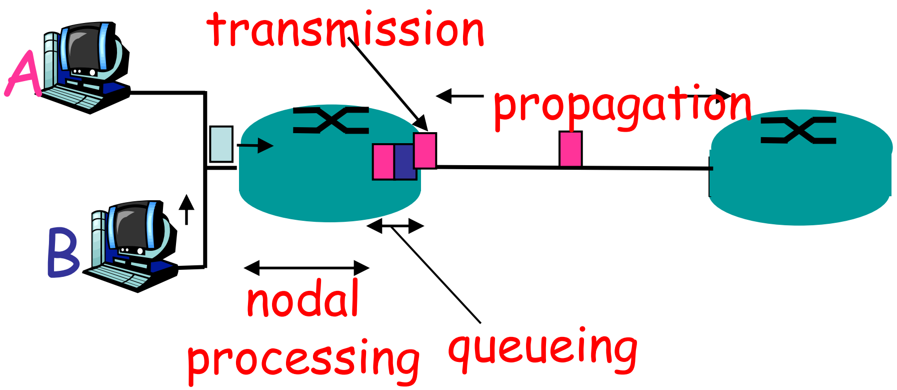
- circuit-switching:
  - 10 users
- packet switching:
  - 35 users, probability > 10 active less than .004

Packet Switching vs Circuit Switching (2)

- Packet switching is great for bursty data
  - resource sharing
  - no call setup
- It incurs excessive congestion: packet delay and loss
  - protocols needed for reliable data transfer, congestion control
- How to provide circuit-like behavior?
  - bandwidth guarantees needed for audio/video apps
  - still an unsolved problem, but solutions such as ATM have been developed

Delays in Packet Switching

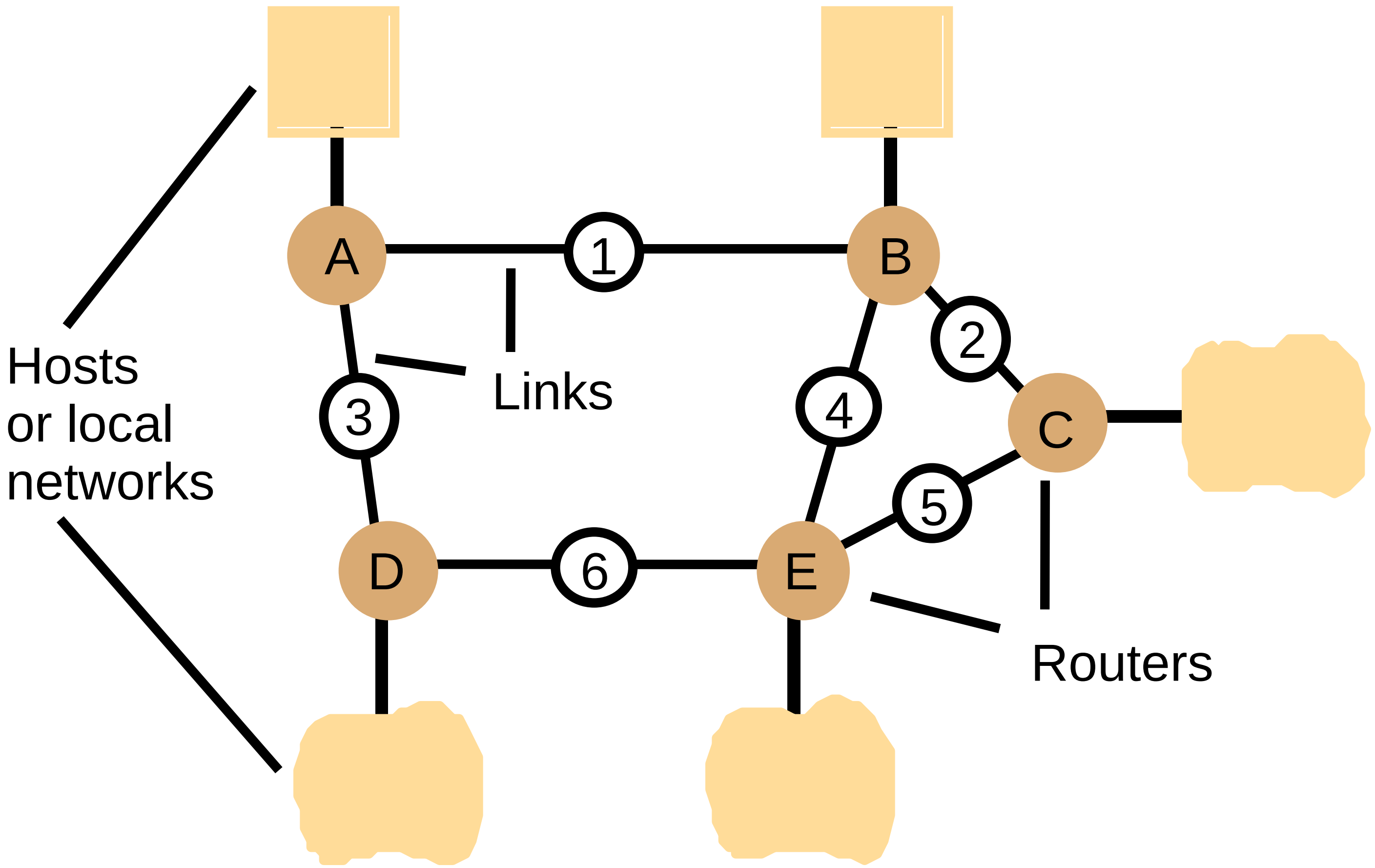
- Packets experience delay on end-to-end path
- Sources of delay at each hop:
  - Nodal processing
    - check bit errors
    - determine output link
  - Queuing
    - time waiting at output link for transmission; depends on congestion level of router
  - Transmission delay
    - R=link bandwidth (bps), L=packet length (bits)
    - time to send bits into link = L/R
  - Propagation delay
    - d = length of physical link, s = propagation speed in medium (~2×10^8 m/sec)
    - propagation delay = d/s



Routing

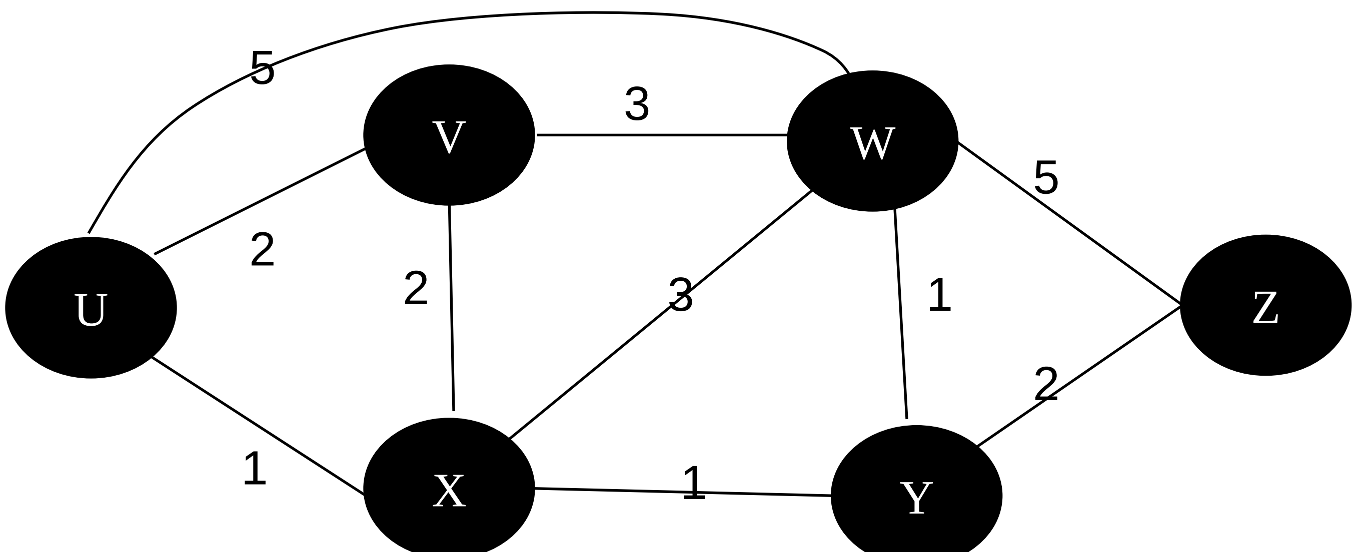
- Goal: move packets among routers from source to destination
- It is an issue in packet switched networks
- datagram network:
  - destination address determines next hop
  - routes may change during session
  - analogy: driving, asking directions
- virtual circuit network:
  - each packet carries tag (virtual circuit ID), tag determines next hop
  - fixed path determined at call setup time, remains fixed thru call
  - routers maintain per-call state

Routing in a Wide Area Network



Routing Algorithms

- End hosts are directly attached to a default router
  - Routers are responsible for forwarding packets from the source to the destination
- How do routers know where to forward the packets?
  - Routers construct routing tables that map destinations to network links
  - Routes should provide "good" properties, such as low latency
    - → Abstractly, the routes should minimize a cost metric



Routing Algorithms

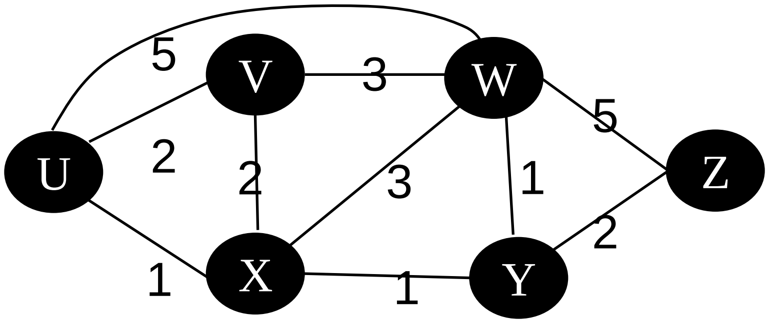
- Two main classifications of routing algorithms
  - Global routing algorithm
    - → e.g. Link-state (LS) algorithm
    - → Connectivity of all nodes and all link costs serve as inputs
  - Decentralized routing algorithms
    - → e.g. Distance-vector algorithm
    - → No node has complete information about the costs of all network links

Link-State Routing

- Network topology and all link costs are known
  - Each node broadcasts its link-state information to all other nodes in the network
  - OSPF is an example of a link-state routing algorithm

- Makes use of Dijkstra's algorithm for finding least-cost paths from one node to all other nodes in the network
  - Iterative algorithm
  - After *k* iterations, the least-cost paths are known to *k* destinations
    - → These *k* paths will have the *k* smallest costs

Link-State Routing



- D(v): cost of the least-cost path from the source node to destination v
- p(v): previous node along the current least-cost path from the source to v
- N': subset of nodes, where v is in N' if the least-cost path from the source to v is definitively known

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	inf	Inf
1	ux	2,u	4,x		2,x	Inf
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

- Example taken from *Computer Networks A Top-Down Approach*

Distance Vector Routing

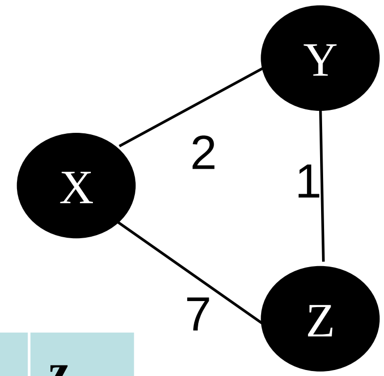
- The distance-vector algorithm is iterative, distributed, and asynchronous
  - Each node receives some information from one or more directly attached neighbors (distributed)
  - It exchanges information until the results converge (iterative)
  - Messages are exchanged at different times between different nodes (asynchronous)

- Based on the Bellman-Ford equation
  - $d_x(y) = \min_v\{c(x,v) + d_v(y)\}$
- $d_x(y)$ is the cost of the least-cost path from x to y
- c(x,v) is the cost of the link between x and v, where v is one of x's neighbors

Distance Vector Routing

- Each node starts off with some estimate of its distance to every node that it knows of, which might only be a subset of all the nodes in the network.
  - Known as its distance-vector
  - $D_x = [D_x(y)\text{: } y \text{ in } N]$
- Exchange its distance-vector with its neighbors
  - Update its own distance vector
  - $D_x(y) = \min_v\{c(x,v) + D_v(y)\}$ for each node y in N

Distance Vector Routing



Node x table

	x	**Y**	**Z**
x	0	2	7
y	Inf	Inf	Inf
z	Inf	Inf	Inf

	x	**y**	**z**
x	0	2	3
y	2	0	1
z	7	1	0

	x	**y**	**z**
x	0	2	3
y	2	0	1
z	3	1	0

Node y table

	X	**y**	**z**
x	Inf	Inf	Inf
y	2	0	1
z	Inf	Inf	Inf

	x	**y**	**z**
x	0	2	7
y	2	0	1
z	7	1	0

	x	**y**	**z**
x	0	2	3
y	2	0	1
z	3	1	0

Node z table

	x	**y**	**z**
x	Inf	Inf	Inf
y	Inf	Inf	Inf
z	7	1	0

	x	**y**	**z**
x	0	2	7
y	2	0	1
z	3	1	0

	x	**y**	**z**
x	0	2	3
y	2	0	1
z	3	1	0

Example taken from *Computer Networks A Top-Down Approach*

Network Taxonomy