Internet Technology Review

Internet Technologies and Applications

Contents

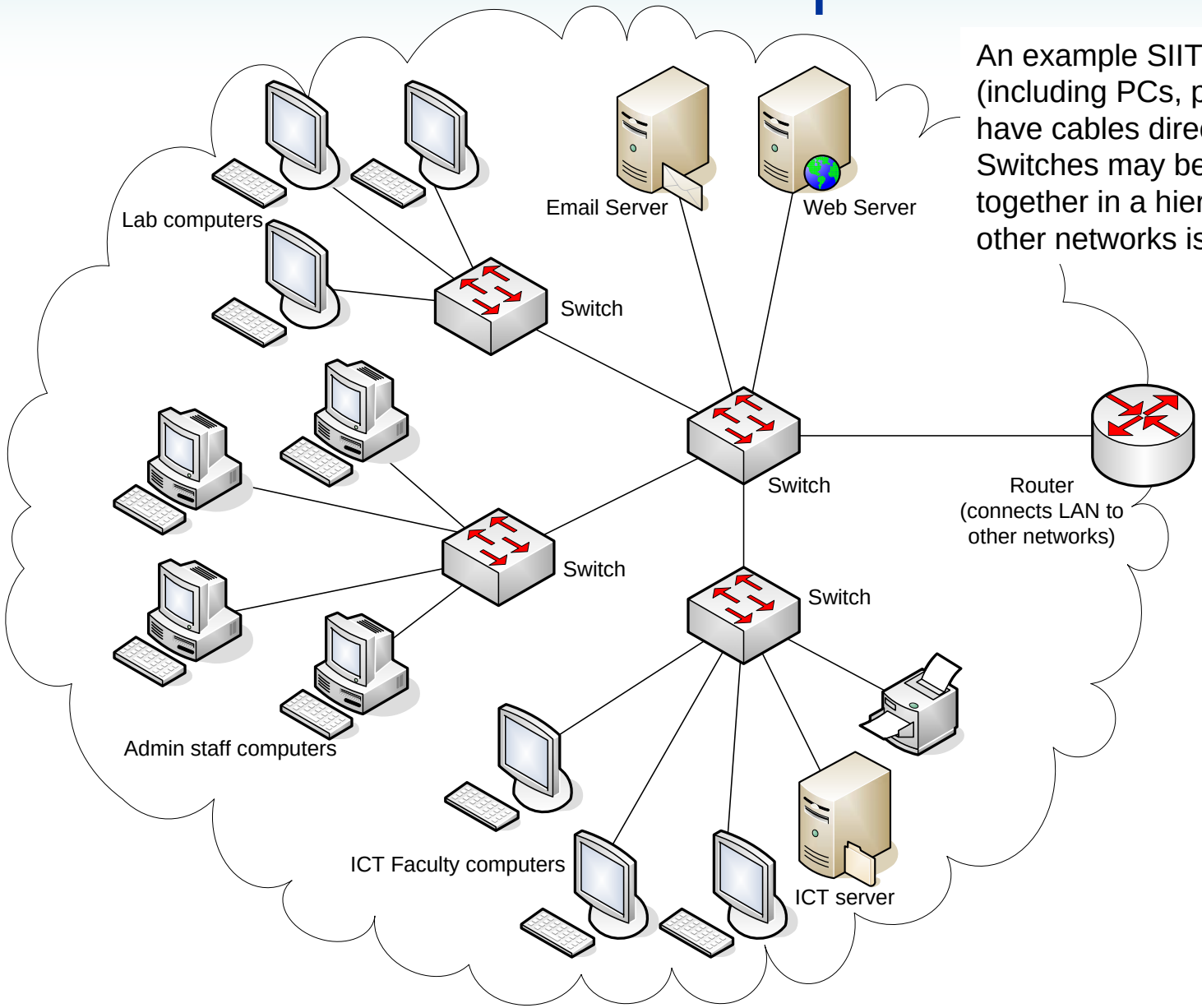
- LANs, WANs and Routers
  - Quick revision of basic network architecture
- Common Protocol Mechanisms
  - Some of the mechanisms used by protocols at all layers
  - (Its important to review these mechanisms, because many are common to Internet applications we consider in this course, e.g. instant messaging, P2P file sharing, …)
- The Internet Protocol (IP)
  - IP datagram
  - IP addresses
- Performance in the Internet

LANs, WANs and Routers

Local Area Networks

- LANs allow computers to communicate over a small distance (“local area”)
  - Home, office, campus, building, …
  - End-users usually access networks via a LAN
  - Data rates to support traffic of 10’s to 100’s of users in an organisation
  - Typically owned and operated by the owner of the end-user computers
    - E.g. SIIT owns and operates all the devices in its LANs
- There are different LAN technologies:
  - Ethernet (IEEE 802.3): 10Mb/s, 100Mb/s, 1Gb/s, 10Gb/s, …
  - Wireless LAN (IEEE 802.11): 11b, 11a, 11g, 11n, …
  - Older technologies like: Token Ring, Novell NetWare, IBM SNA, …
- Example
  - The computers within an SIIT Computer Lab may form one LAN, and also connect to other LANs (e.g. Faculty computers, other Computer Labs)
  - Or, the computers within the IT Building may form on LAN

LAN Example

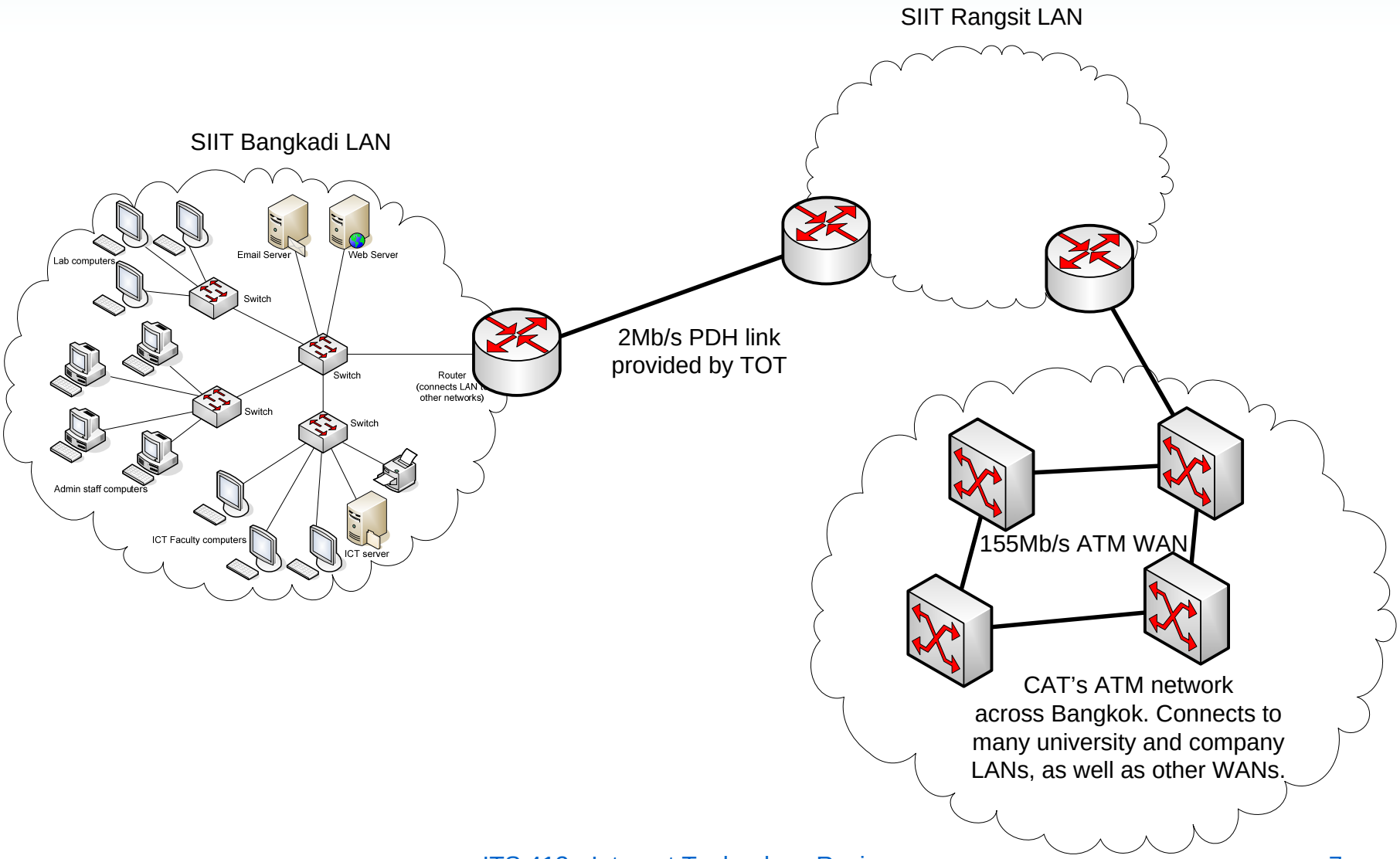


An example SIIT LAN. Hosts (including PCs, printers, servers) have cables directly to Switches. Switches may be connected together in a hierarchy. Access to other networks is via the Router.

Wide Area Networks

- Connect computers (and networks) over a large ("wide") area
  - Interconnect offices, campuses, homes across cities, countries and the world
  - Typically owned and operated by telecommunications companies or Internet Service Providers (ISPs)
    - E.g. SIIT pays TOT a monthly fee for connecting Bangkadi to Rangsit via a WAN
- There are different WAN technologies:
  - PDH, SDH, SONET for dedicated links
  - ATM, Frame Relay, X.25 for networks
  - Microwave, satellite and cellular for wireless/mobile access
  - ADSL, Cable for links and networks
- Example
  - The SIIT LANs at Bangkadi and Rangsit are connected via a WAN
  - Similarly, SIIT/Thammasat are connected to other networks (universities, companies, ISPs) via another WAN

WAN Example

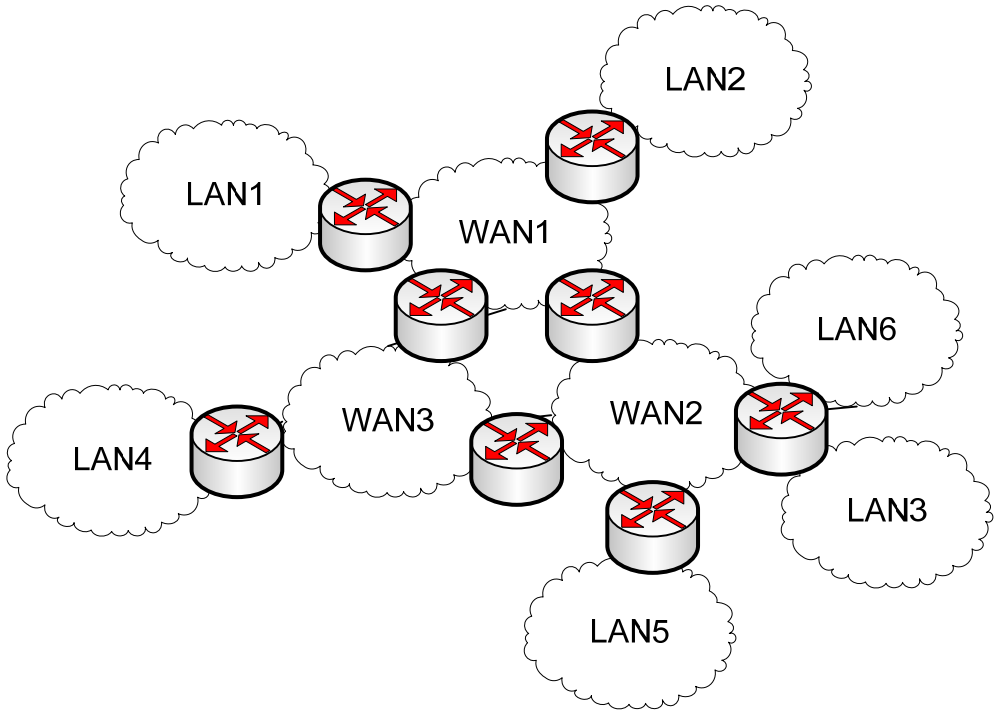

SIIT Rangsit LAN
SIIT Bangkadi LAN
Lab computers
Email Server
Web Server
Switch
Switch
Switch
Switch
Router
(connects LAN
other networks)
Admin staff computers
ICT Faculty computers
ICT server
2Mb/s PDH link
provided by TOT
155Mb/s ATM WAN
CAT's ATM network
across Bangkok. Connects to
many university and company
LANs, as well as other WANs.

Connecting Different LANs and WANs

- Many different technologies are used for LANs and WANs
  - A computer on the SIIT Bangkadi LAN can communicate with any other computer on the SIIT Bangkadi LAN since they all use Ethernet protocol
  - But how does a computer at SIIT Bangkadi LAN communicate with computer at SIIT Rangsit LAN?
    - Both computers may use Ethernet, but they are separated by a PDH link
    - What “translates” from Ethernet to PDH and back to Ethernet?
      - Answer: the Internet Protocol
- Since computers across the world use different LAN/WAN protocols, to enable communication with any other computer, the Internet Protocol is used
  - All computers talk a “common language” which is IP; even if they use different LAN/WAN protocols
  - The devices that connect different LANs/WANs together are routers

Routers Connect Networks

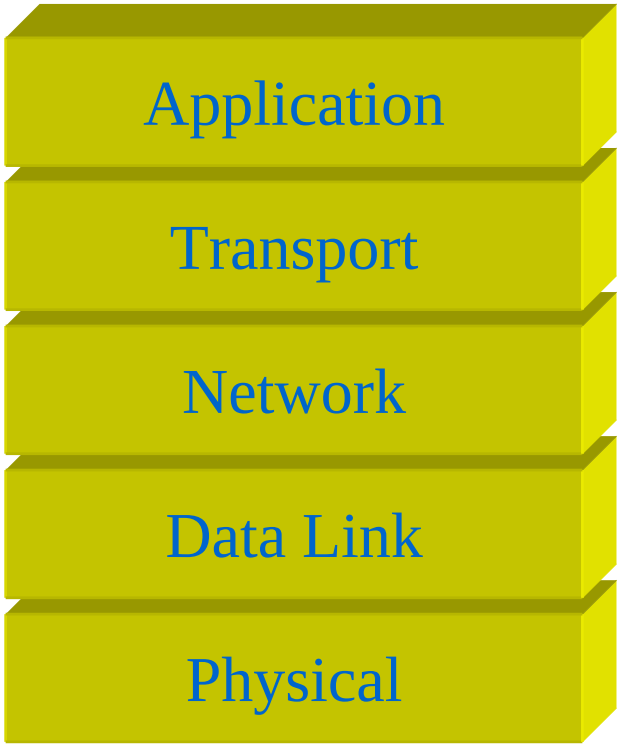
- Routers in the Internet provide the “translation” from one LAN/WAN protocol to another LAN/WAN protocol
- All computers (also referred to as hosts) and routers implement IP
  - To communicate between any two computers on the Internet, a source host sends an IP datagram to the destination host, via a set of routers



Layers and Protocol Stacks

- Computer communications is hard!
  - Many tasks to be performed for one computer to communicate with another, including:
    - Transform bits (0's and 1's) into electrical signals to be sent across cables
    - Detect and correct transmissions when errors occur (nothing is perfect; errors are a part of life)
    - Identify and locate computers using addresses
    - Synchronise sender and receiver so they are ready to communication
    - And many more …
  - Hence the "divide-and-conquer" principle is applied by splitting the tasks amongst *layers*
- Layering in the Internet
  - Often a 5 layer Internet model or protocol stack is used
  - Can be referred to as the TCP/IP suite or stack

Our 5 Layer Internet Stack



Contains functionality that is common to various applications used on the Internet

Provides connections between applications (processes) running on the end hosts

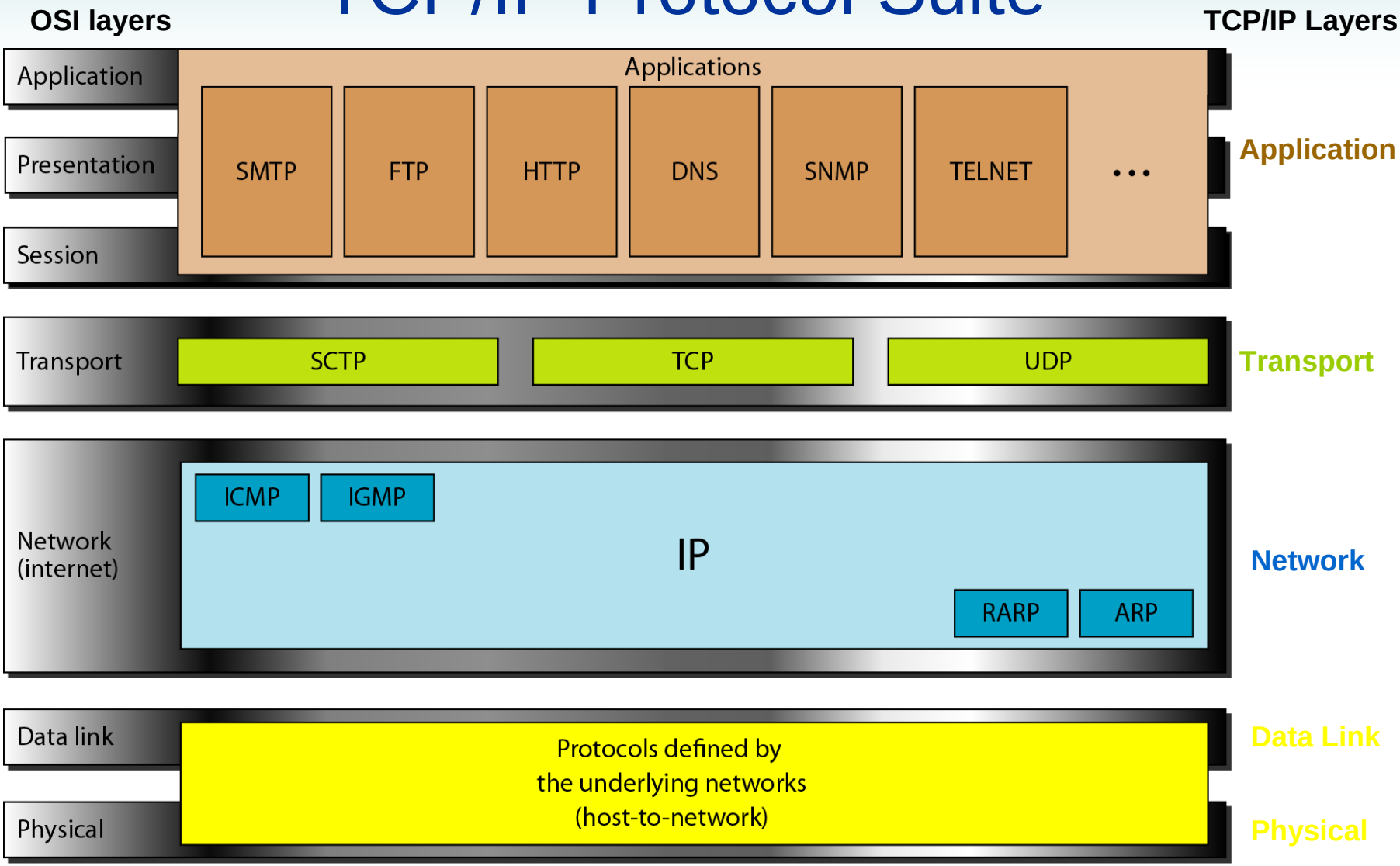
Allows hosts to communicate across different networks; Provides routing across the Internet; determine the path to take; Provides unreliable, connectionless deliver of packets

Transmission of data over link or network to which the device is attached; Addressing scheme of destination device; Allows layers above to ignore details of links

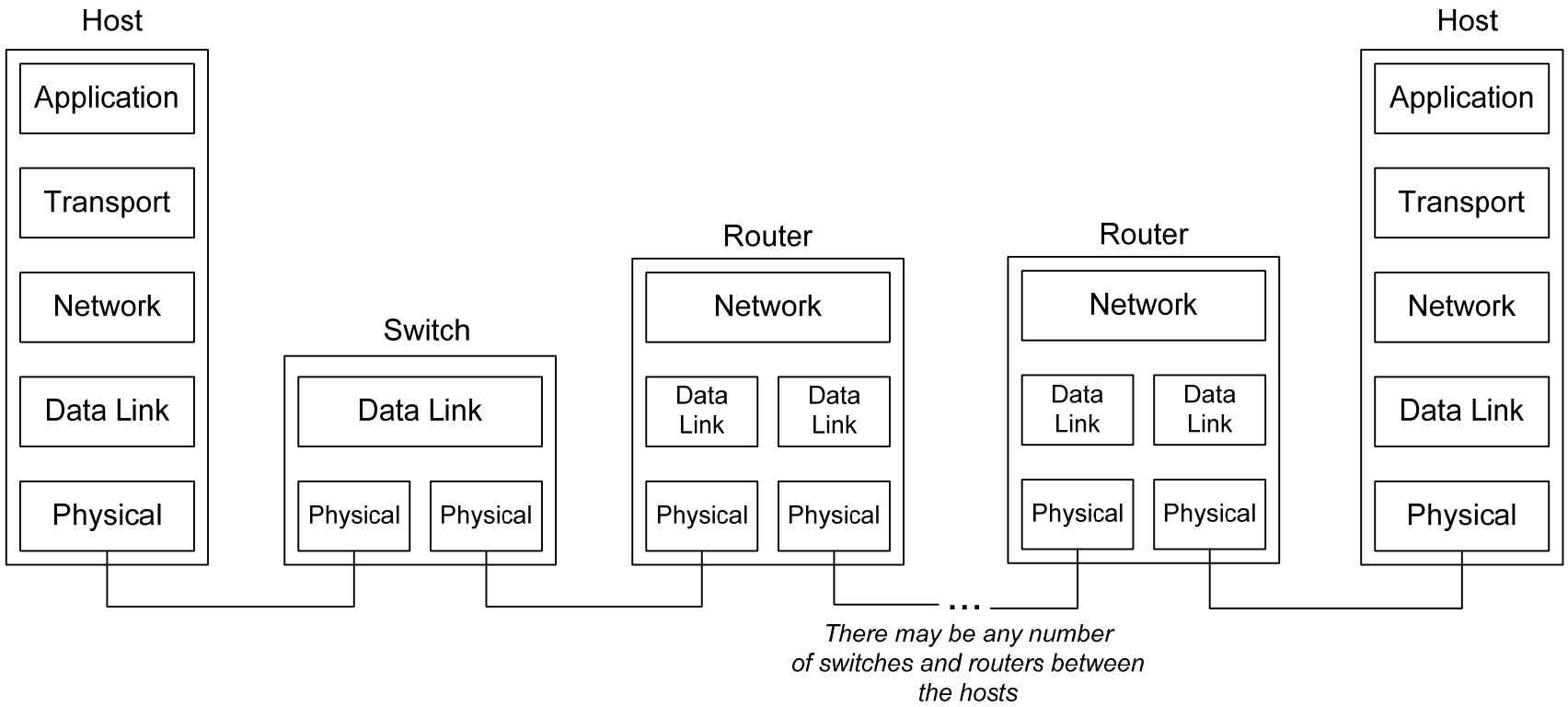
Physical interface between transmission device and medium

How to send bits over transmission medium: data rate, signalling, electrical signals, codecs, modems, …

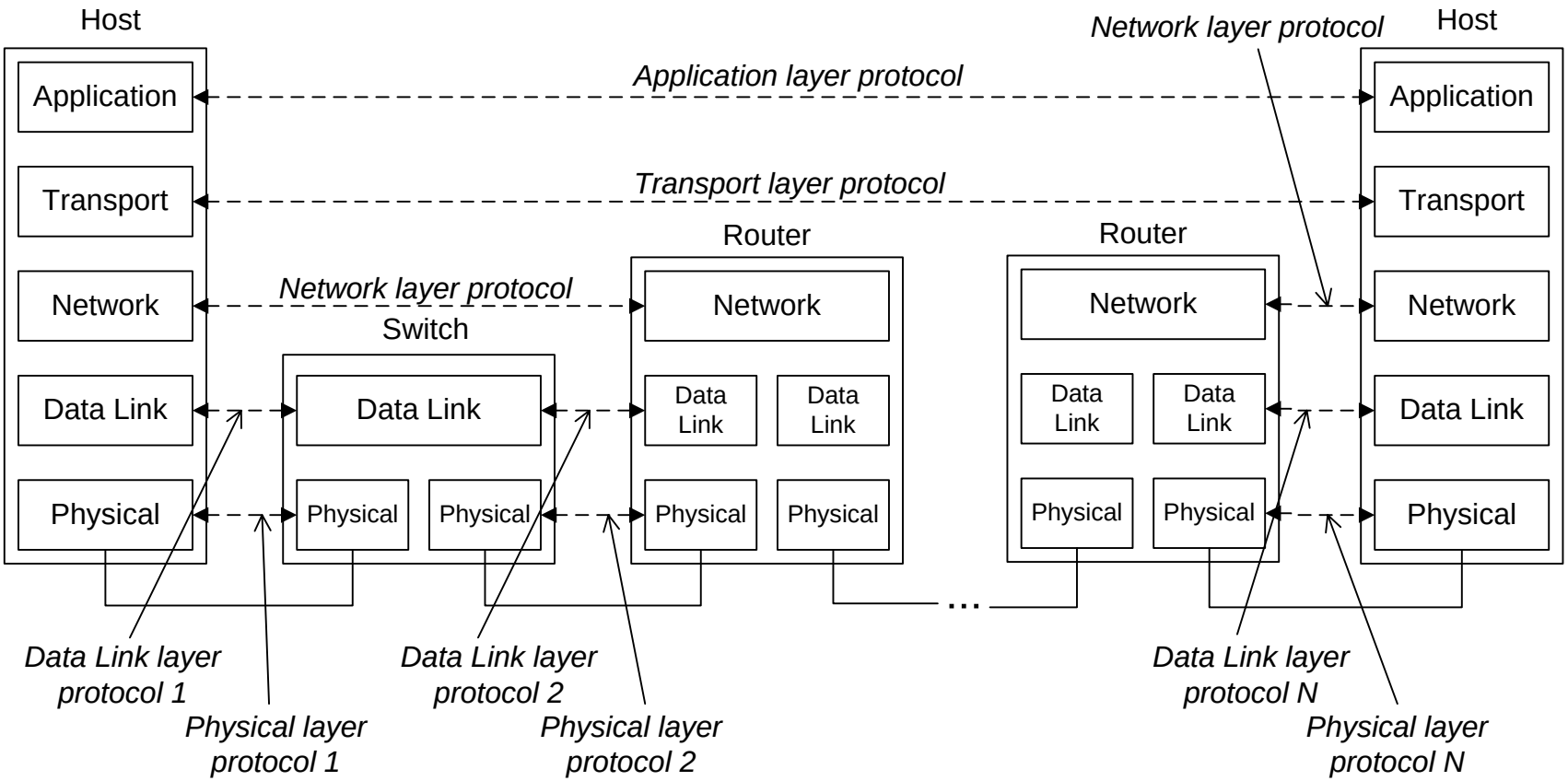
TCP/IP Protocol Suite



Example Protocol Stacks



Protocols at Different Layers



Common Protocol Mechanisms

Protocol Building Blocks

- Protocols at Layers 2 and above use some basic building blocks to perform desired functionality
  - Layer 1 (Physical) deals with transmission of bits as signals
- Common Protocol Mechanisms
  - Encapsulation (by adding Headers)
  - Fragmentation and Re-assembly (Packet Sizes)
  - Ordered Delivery (using Sequence Numbers)
  - Error Control (Reliability)
  - Flow Control
  - Congestion Control
  - Connection Management
  - Routing
  - Addressing

Encapsulation

- Almost all protocols transfer data in blocks
  - A “block” usually contains data plus some control information
    - In fact, some times there is no data, only control information
      - We often refer to this control information as a *header*

  - Control information has various purposes:
    - Addressing: identify the sender/receiver
    - Error detection/correction: detect (and possibly correct) errors in the data
    - Protocol control: used to implement the protocol functions discussed in remainder of this lecture
  - Adding control information to data is called *encapsulation*
    - Removing the control information to retrieve the original data is called *decapsulation*
- Encapsulation usually performed at each layer
  - There are many different names used for the blocks of data and control information

Encapsulation across Layers

1. User (application) generates data

Application

2. Application layer protocol follows protocol rules, adds control information, and sends *message* to Transport layer

Transport

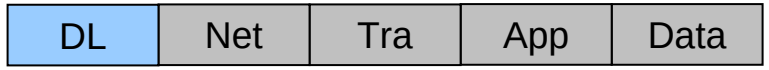
3. Transport layer protocol follows protocol rules, adds control information, and sends *segment* to Network layer

Network

4. Network layer protocol follows protocol rules, adds control information, and sends *datagram* to Data Link layer

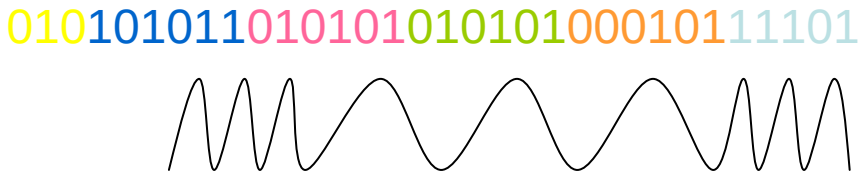
Data Link

5. Data Link layer protocol follows protocol rules, adds control information, and sends *frame* to Physical layer



Physical

6. Physical layer protocol follows protocol rules, adds control information, and sends *bits* over the link (encoded via signal)



Encapsulation: Names

- There are many different names for the “blocks”
- Some common names which we will use for each layer are:
  - *Message*: used by the application layer
  - *Segment*: used by transport layer
  - *Datagram*: used by network layer
  - *Frame*: used by data link layer
  - *Bits*: used for physical layer
- A general name (which will often use) is *packet*
  - A formal name sometimes used is *Protocol Data Unit* (PDU)
- These blocks refer to the data plus control information
  - Example: a *message* refers to the original data plus the Application layer control information (header)
  - Example: a *segment* refers to the original data, plus Application layer header plus Transport layer header

Fragmentation and Reassembly

- Protocols may split a block into smaller blocks: called *fragmentation*
- Typical reasons for fragmentation:
  - Network may only support blocks up to certain size
    - E.g. ATM limited to 53 byte frames; Ethernet limited to 1526 byte frames
  - Error control, such as retransmission schemes, may be more efficient with smaller packet
    - With smaller packet, fewer bits are retransmitted if there is an error
  - Fairer access to a shared medium
    - If one station has a very large packet to send, everyone else must wait a long time until they've finished (unfair); with smaller packets, only wait a short time
  - Smaller buffers at receivers
- Disadvantages of fragmentation:
  - Greater overhead
    - Each packet has control information added; for smaller packets, control information represents larger percentage overhead
  - Greater processing time
    - Usually a receiver must process all the control information; more packet, more processing
- Selecting an practical packet size is difficult, but important
- Opposite is *reassembly* of smaller blocks into original larger block

Fragmentation across Layers

1. User (application) generates data

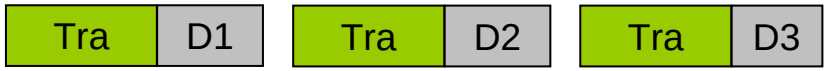
Application

2. Application layer protocol follows protocol rules, adds control information, and sends *message* to Transport layer

Transport

3. Transport layer protocol follows protocol rules, fragments the application message into 3 pieces, adds control information to each, and sends each *segment*, one by one, to Network layer

Fragmentation



Network

4. Network layer protocol follows protocol rules, adds control information, and sends *datagrams* to Data Link layer

and so on …

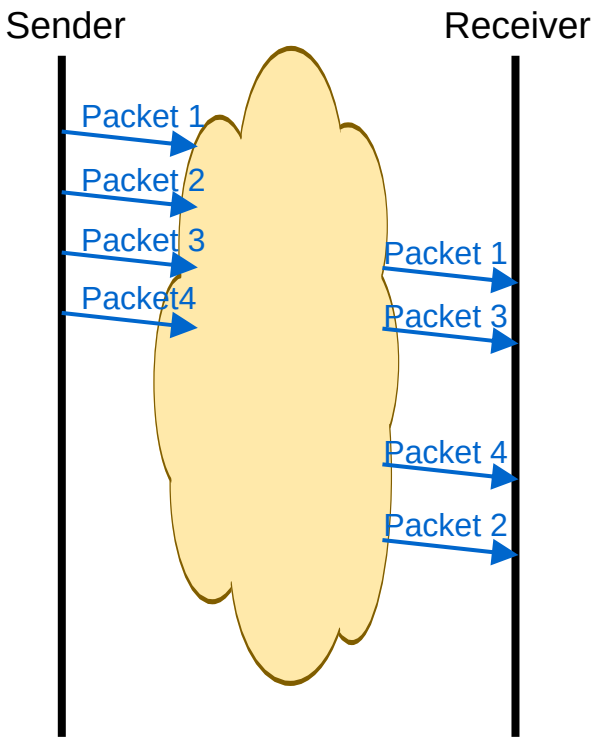
Re-assembly occurs at the Transport layer receiver

Connection Control

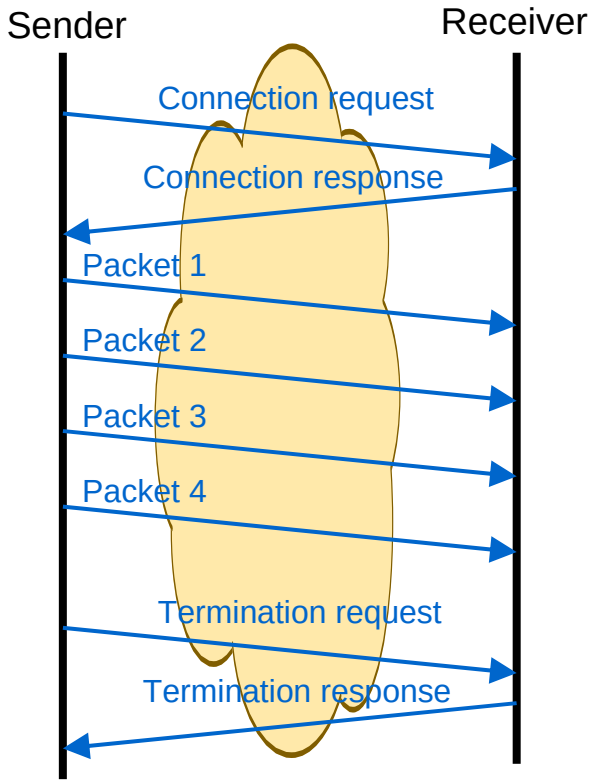
- Two modes of data transfer:
  - Connectionless: each packet is transmitted independently of all prior packets
    - Simple, but makes it difficult for end points to manage the data transfer efficiently
    - Example: IP is connectionless protocol
  - Connection-oriented: sender and receiver create a logical association (or connection) between each other before data transfer
    - Complex, but allows control of the data transfer
    - Involves three phases:
      - Connection setup (or connection establishment)
      - Data transfer
      - Connection teardown (or connection termination)
    - Example: telephone networks; TCP

Connection Control

- Connectionless



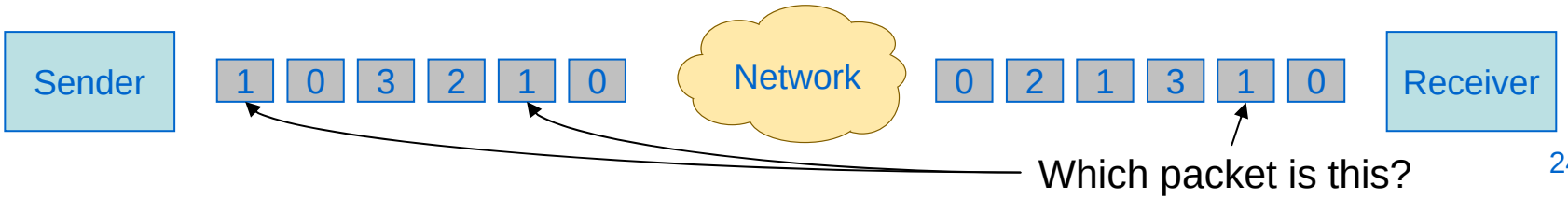
- Connection-Oriented



Ordered Delivery

- Using packet switched networks, it is possible packets may arrive at the receiver out of order

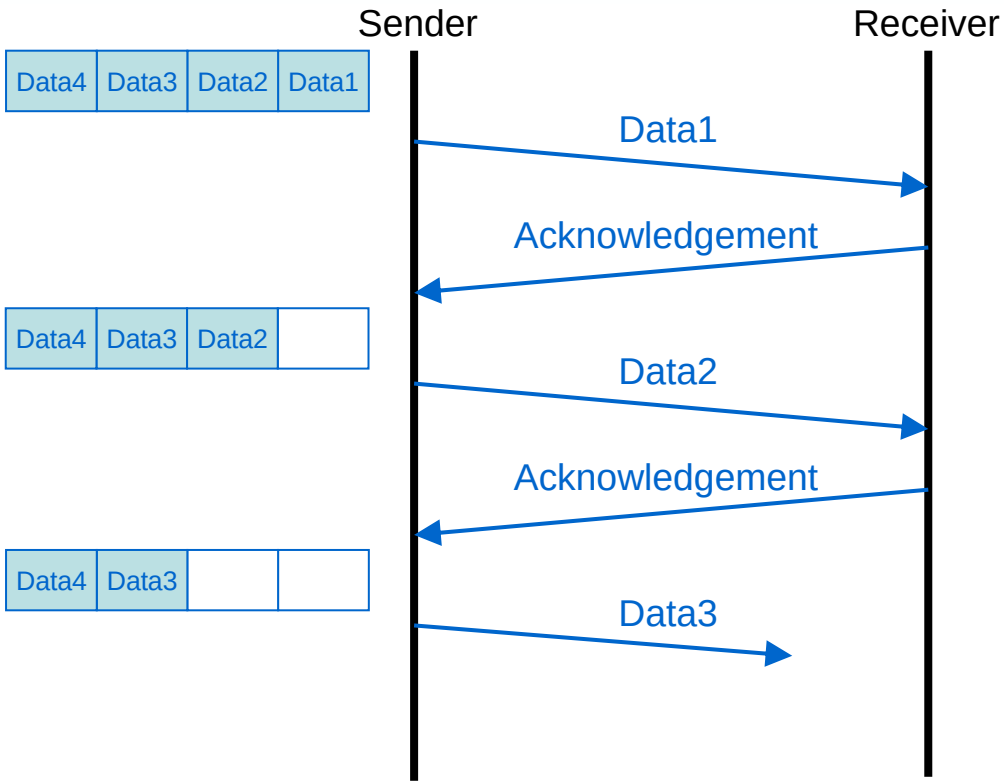
- In connection-oriented protocols, expect to receive the data in order
- Therefore, need to re-order the packets at the receiver
  - How do we know which order the packets should be in?
- Sequence numbers
  - Each packet carries a sequence number in its header
- The problem with sequence numbers
  - The header has a limited size: sequence number field is N bits
    - N bits means sequence number can be from 0 to 2^N-1
  - Must “wrap” the sequence number, which can lead to possibility of two different packets having same sequence number



Error Control

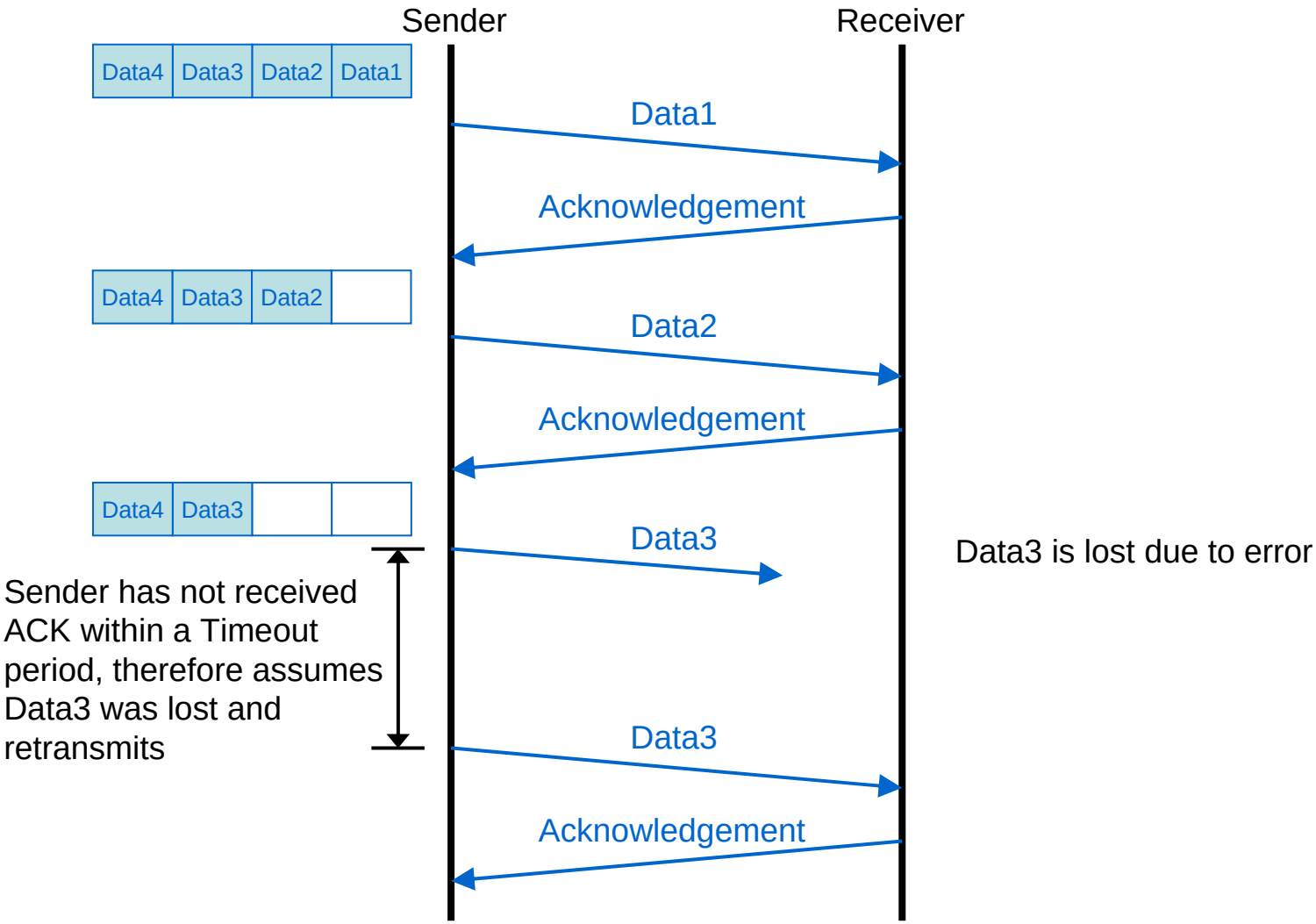
- Errors can occur in networks due to:
  - Imperfect communication links (send the bits 01101, receive the bits 01111)
  - Failed devices: a switch, router or interface card stops working (crashes)
  - Overuse of resources: the memory (buffer) on a router is full, and can handle no more packet
- From a receivers point of view, errors can be grouped as:
  - Data sent, but received in error (damaged packet)
  - Data sent, but not received (lost packet)
- In most applications, errors cannot be tolerated
- Error control involves:
  - Error Detection
    - The receiver must determine if the received packet is correct or in error
    - The sender may determine if a packet has been sent, but not successfully received
  - Error Correction
    - Once an error is detected, need to fix it
  - A common scheme for error detection and correction involves a combination of:
    - Sending error detecting codes with the packet (Frame Check, CRC-32)
    - Apply “Acknowledgement and Retransmission” (or Automatic Repeat Request, ARQ)

Error Control: Stop and Wait



After sending a Data packet, must stop and wait for an acknowledgement before sending the next Data packet

Error Control: Stop and Wait



Flow Control

- A receiver has limits at which it can receive/process data
  - The receiver computer/interface may be slow compared to sender
  - The receiver may have small amount of memory to store received data
- If a sender sends too fast for the receiver, then data will be lost
  - Leads to retransmissions, which significantly reduces performance
- *Flow control* involves the receiver controlling the rate at which the sender sends
  - Stop-and-Wait Flow Control
    - Sender can only send next packet after the previous packet is acknowledged
  - Sliding Window Flow Control
    - Sender can send up to W packets, after which it must wait for an acknowledgement
  - Example: TCP uses a sliding window flow control algorithm
- In the Internet, flow control is usually implemented on some data link layers and the transport layer (TCP)

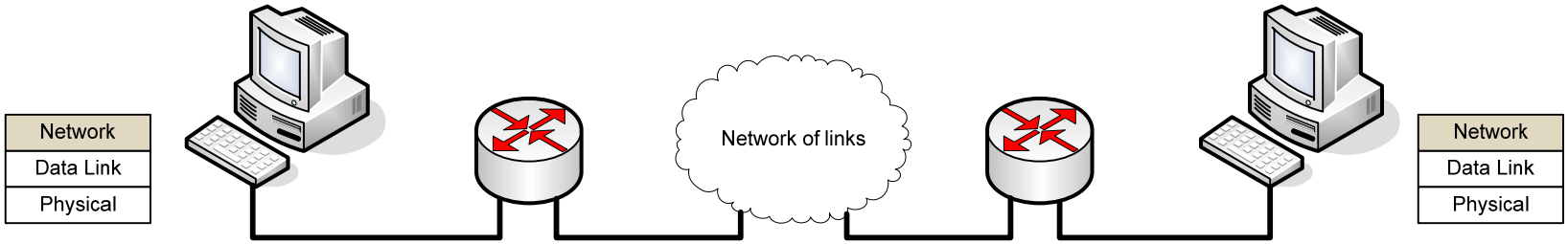
Congestion Control

- Routers in the Internet have limited memory resources, limited processing power and limited sending ability
  - If a router receives too many packets that overflow these resources, then data will be dropped
  - Dropped data leads to retransmissions
  - Retransmissions lead to more packets at the router
  - This leads to more dropped data, and so on, until the network “collapses”: all the time is spent sending retransmissions, and no useful data is received
- Congestion control involves managing the rate at which senders sends such that they do not overflow the routers (that is, avoid congestion at the routers)
- In the Internet, since there is no congestion control in IP, TCP implements congestion control

Addressing

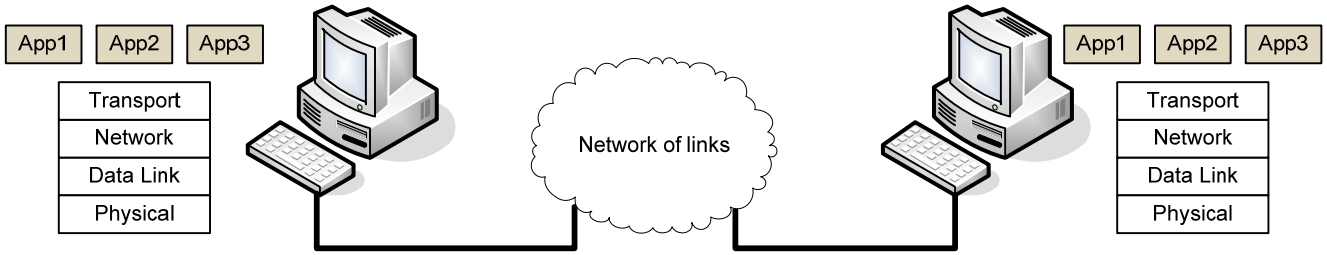
- In the Internet, addresses are used at different levels
- Hardware Addresses
  - When two devices communicate over a link (either wired or wireless) they need addresses to identify the end-points
  - Also called: physical address, Layer 2 address, link address, MAC address

- Logical Addresses
  - In networks, hosts communicate via one or more links, using different link technologies (and addresses)
  - A common address, independent of the hardware address, is needed
  - Most commonly used is IP address

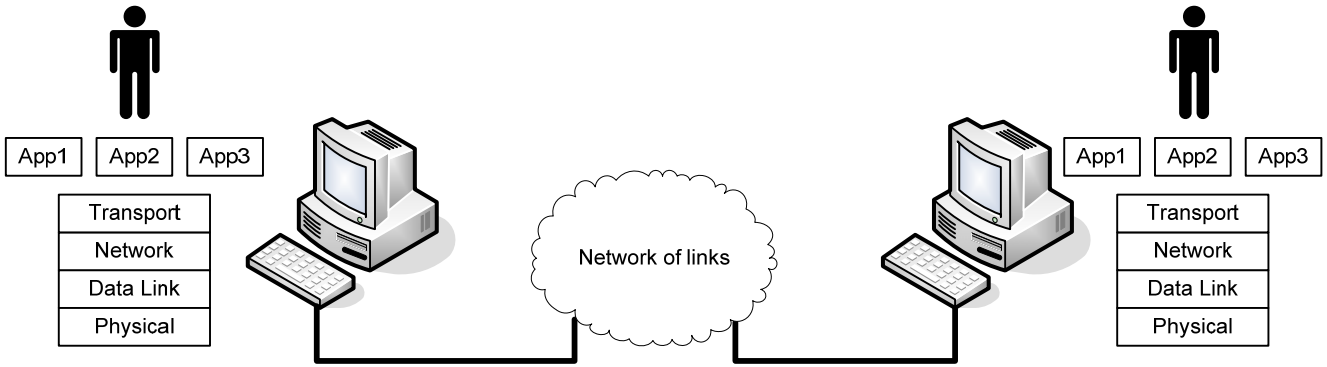


Addressing

- Port numbers
  - A computer may have multiple applications running at the same time
  - A logical address identifies an interface on a computer: how do we identify the application on that computer? Port numbers are a common approach



- User Friendly Addresses
  - Computers use Hardware, Logical and Port addresses to communicate
  - But most cases, humans use the computers, therefore a more “user-friendly” address is needed
  - Domain names and URLs are examples of user-friendly addresses



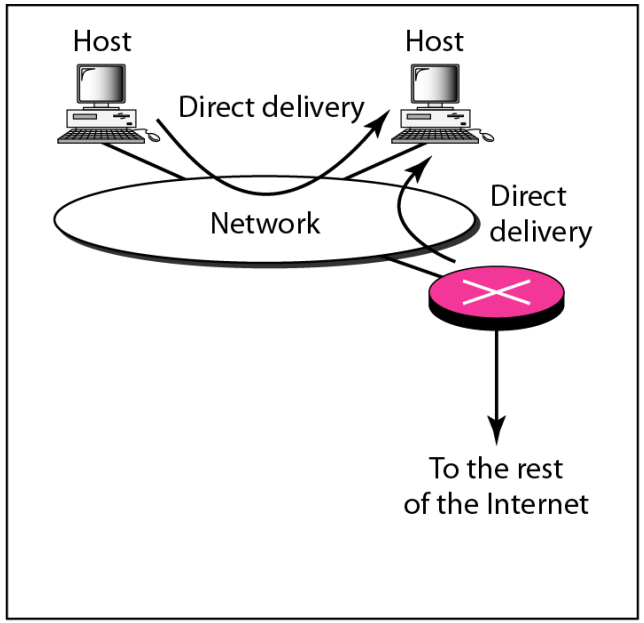
Routing

Forwarding and Routing

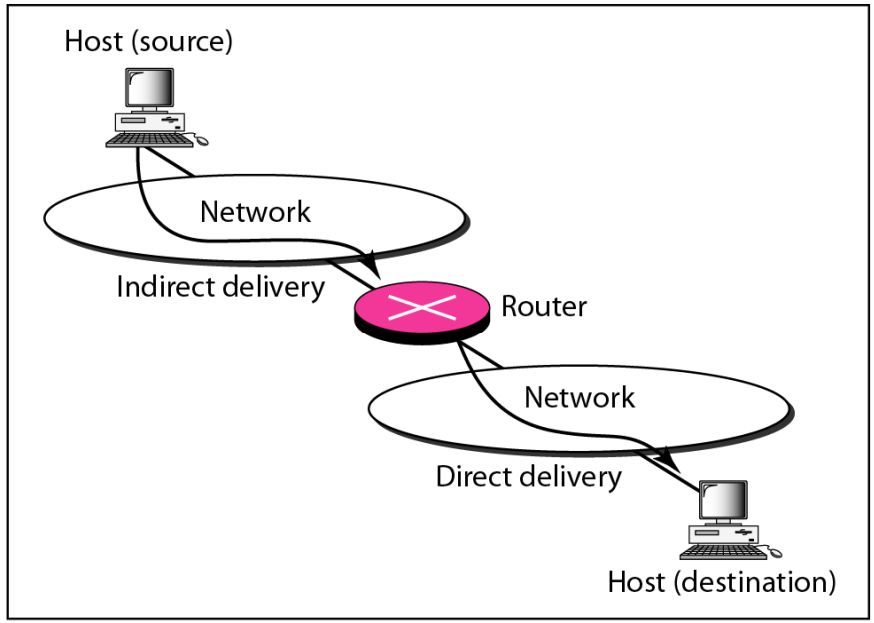
- Routers are a key part of packet switched networks, such as the Internet
- The tasks of routers can be categorised as:
  - Routing: determining a route (or path) from source to destination
  - Forwarding: sending packets along the selected path
- As a result, packets may be directly or indirectly delivered between source and destination

Direct versus Indirect Delivery

- Sending a packet to the destination involves two different delivery methods:
  - Direct delivery: if the final destination is on the same physical network as the "deliverer", send the packet direct to the destination
  - Indirect delivery: if the final destination is NOT on the same physical network as the "deliverer", send the packet indirectly to the destination, via a router (which will become the "deliverer")



a. Direct delivery



b. Indirect and direct delivery

Forwarding

- Routing information (that is, information to determine the path from source to destination) is stored in *routing tables* at each node
- Forwarding: when a node has a packet to send, the node looks up the routing table to determine where to send the packet, and then sends (or forwards) the packet
- In the Internet, the following strategies are used:
  - Next-hop Forwarding: the routing table stores the next-hop that the packet is to be sent to in order to reach the destination
  - Network-specific Routing: the routing table stores paths to *networks*, rather than to individual hosts (its only the last router that deals with individual hosts)
  - Default routes: instead of routers storing routes to all possible networks, they maintain default routes for “all other networks”

Next Hop Forwarding

a. Routing tables based on route

Destination	Route
Host B	R1, R2, host B

Routing table for host A

Destination	Route
Host B	R2, host B

Routing table for R1

Destination	Route
Host B	Host B

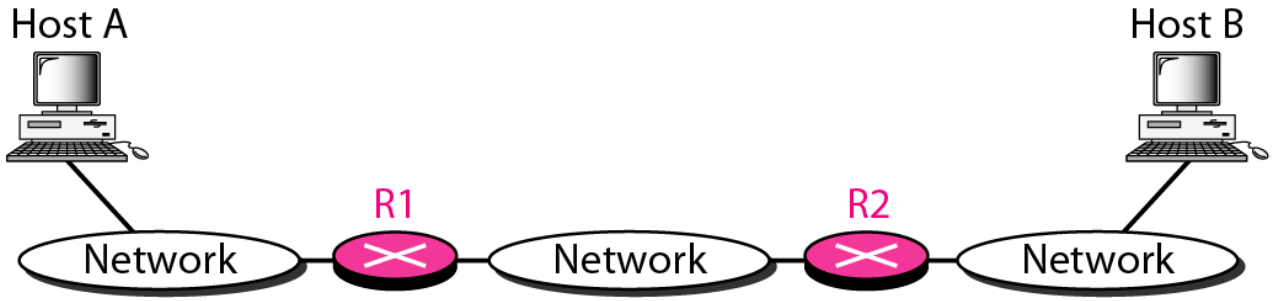
Routing table for R2

b. Routing tables based on next hop

Destination	Next hop
Host B	R1

Destination	Next hop
Host B	R2

Destination	Next hop
Host B	---



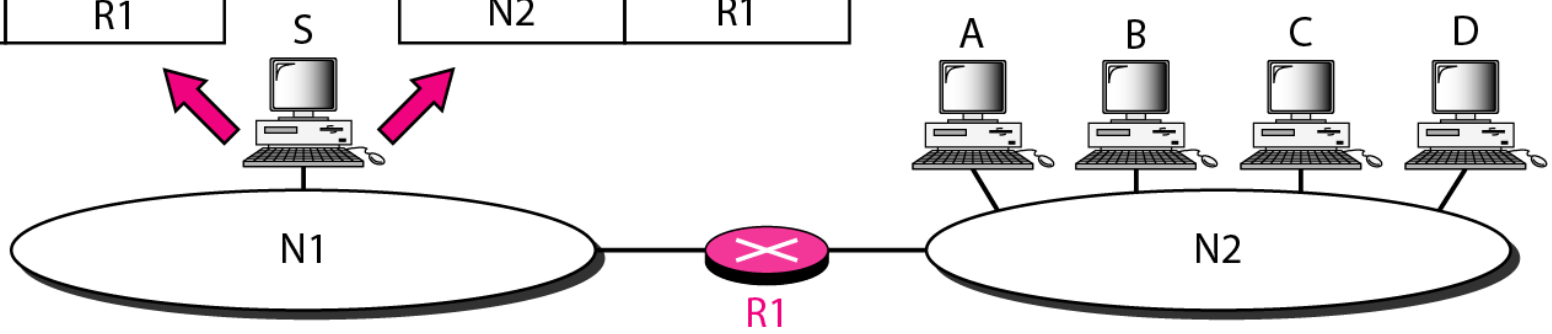
Network-Specific Routing

Routing table for host S based on host-specific method

Destination	Next hop
A	R1
B	R1
C	R1
D	R1

Routing table for host S based on network-specific method

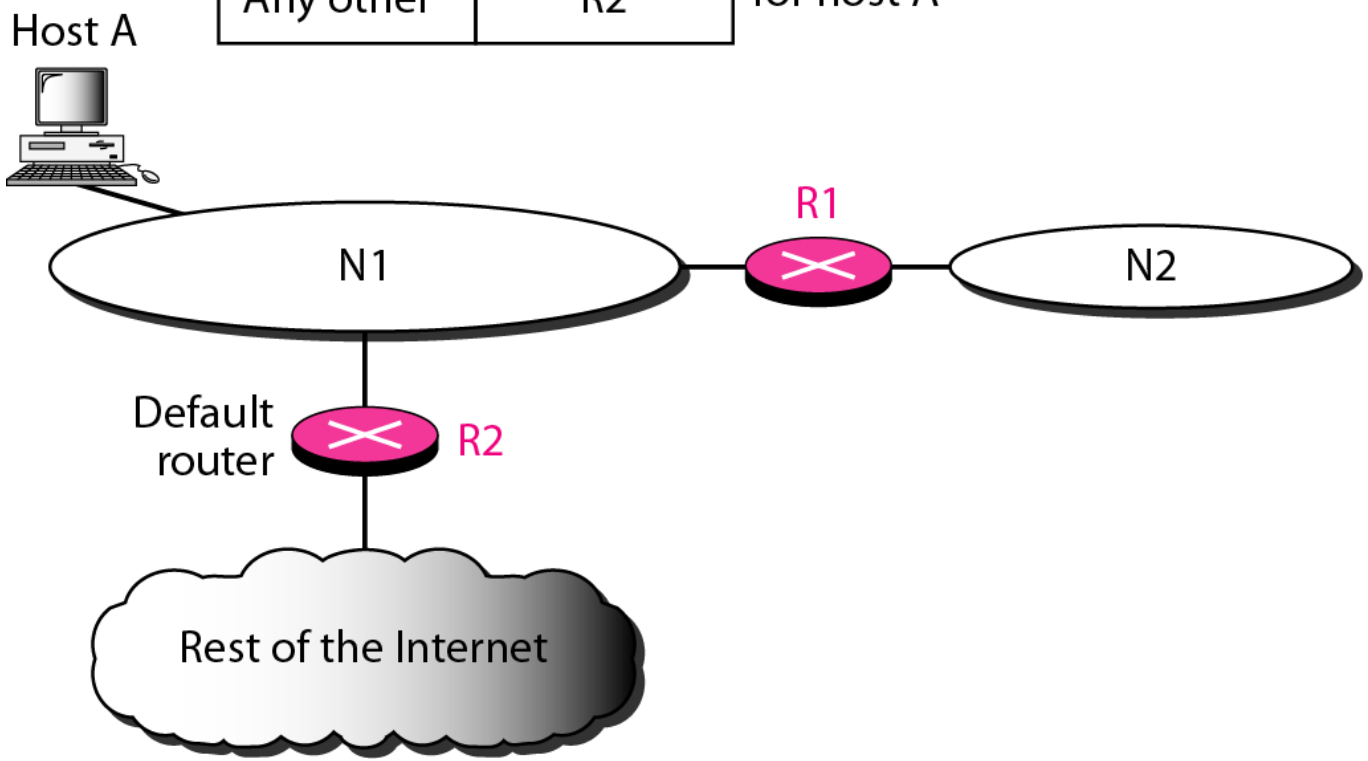
Destination	Next hop
N2	R1



Default Routes

Destination	Next hop
N2	R1
Any other	R2

Routing table for host A



Routing

- How do we choose a route through the network?
  - In other words, how do we create the routing tables at routers?
- In the Internet:
  - Routers are constantly exchanging information with each other to learn about the network connectivity and conditions
  - Given this information, a routing algorithm (in each router) calculates the best routes between networks
  - What are the best paths? Metrics for “best” can include:
    - Number of hops, delay, capacity, monetary cost, commercial agreements, politics, geography, …
  - What are the routing algorithms/protocols?
    - Distance vector routing: RIP
    - Link state routing: OSPF (e.g. use Dijkstra’s algorithm)
    - Path vector routing: BGP
  - Most protocols are quite complex, but can adapt well to changes in the network
- Other approaches:
  - Flooding and variants

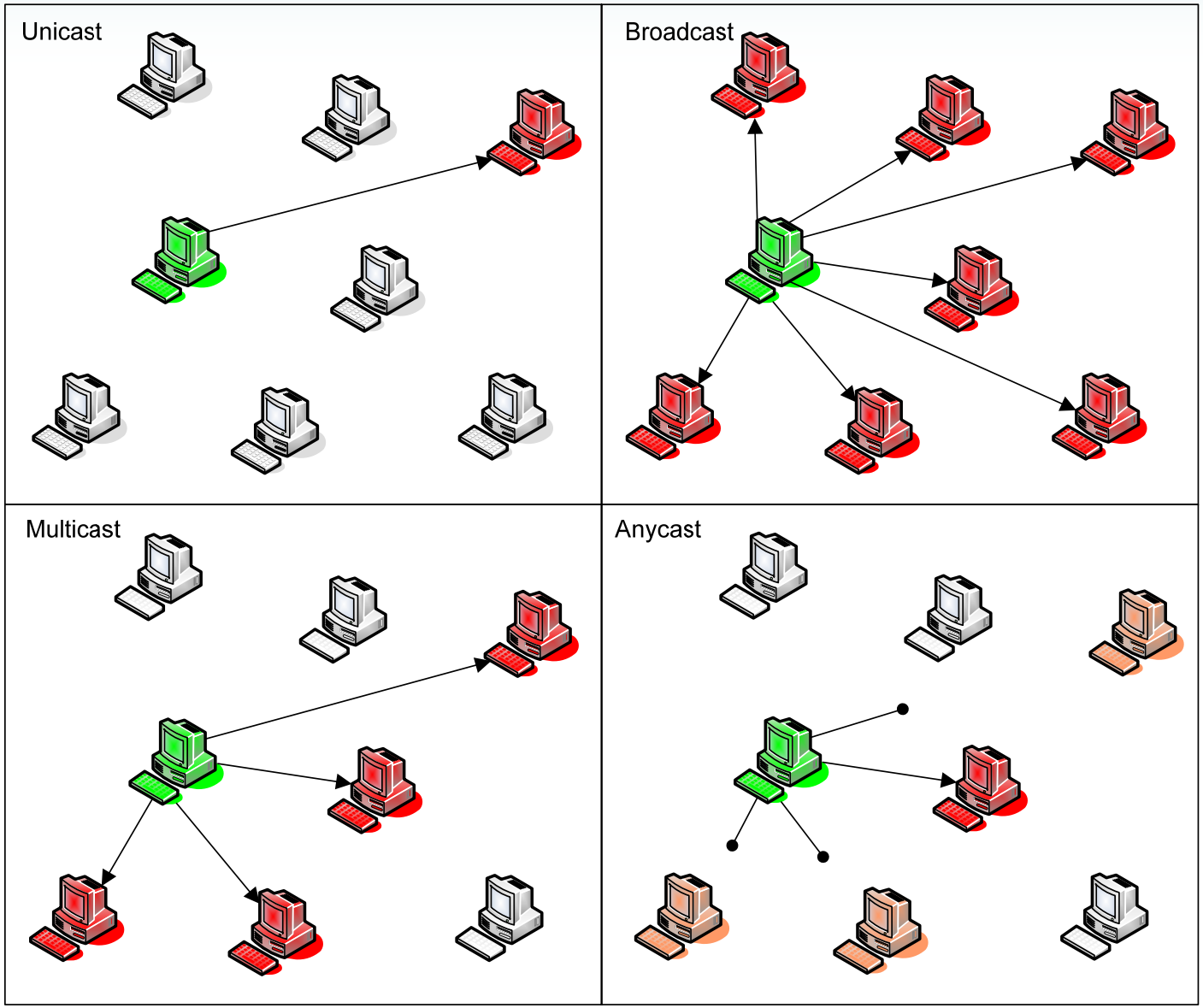
Addressing and Delivery Mechanisms

- Unicast
  - One-to-one association between network address and host
  - The most common method of data delivery
  - A host sends a packet to destination address X – the packet should be delivered to X (and only X will process the packet)
- Broadcast
  - One-to-many association between network address and hosts
  - Many = every host on the network
  - Examples:
    - IP network broadcast address (all 1's in host portion): a host with IP address 192.168.1.2/24 sends a datagram to the network broadcast address 192.168.1.255/24. All other hosts on the IP network will receive the datagram
    - MAC broadcast address (all 1's): a host with MAC address 00:17:31:5A:E5:89 sends a frame to the broadcast address FF:FF:FF:FF:FF:FF. All other hosts on the network will receive the frame
  - Applications:
    - Distributing control/management information, e.g. routing information, address requests
      - Used by many routing protocols, DHCP, ARP, …

Addressing and Delivery Mechanisms

- Multicast
  - One-to-many association between network address and hosts
  - Many = selected group of hosts
  - A more general form of broadcast, where only a selected set of the hosts in the network receive the packet
  - Requires management protocols for the hosts to “subscribe” or “join” the multicast destination group (much more complex than broadcast)
  - Example:
    - IP multicast: a set of hosts on the Internet “subscribe” to the multicast group with address 225.70.8.20. When a host sends a datagram to 225.70.8.20, the datagram is delivered to all hosts subscribed to that multicast group
  - Applications:
    - Multimedia and collaborative applications that involves many users
      - Audio/video/TV streaming, presentations, video/audio conferencing, shared document editing, ..
- Anycast
  - One-to-many association between network address and hosts
  - Many = any host in a group
  - Similar to multicast, but a packet sent to an anycast address is delivered to only one (any one) of the subscribed hosts
  - Applications:
    - Used in advanced DNS implementations: there are multiple, replicated copies of DNS servers in the world; a DNS query will be sent to an anycast address; anycast routing will deliver that query to one of the replicated DNS servers

Addressing and Delivery Mechanisms



Flooding

- Flooding
  - The simplest routing algorithm (but least efficient)
    - There is no separation of forwarding and routing: performed together
    - No use of routing tables
  - Send a copy of the packet to all your neighbours
    - Each neighbour sends a copy of the packet to their neighbours
    - And so on … eventually the packet will reach the destination
  - Advantages:
    - If a path exists between source and destination, a packet will always be delivered to the destination, with at least one copy going via the shortest path
    - All other nodes in the network receive a copy: useful for distributing information to multiple nodes
    - Simple to implement
  - Disadvantages:
    - Inefficient!
  - There are ways to improve the efficiency of flooding

Improvements on Flooding

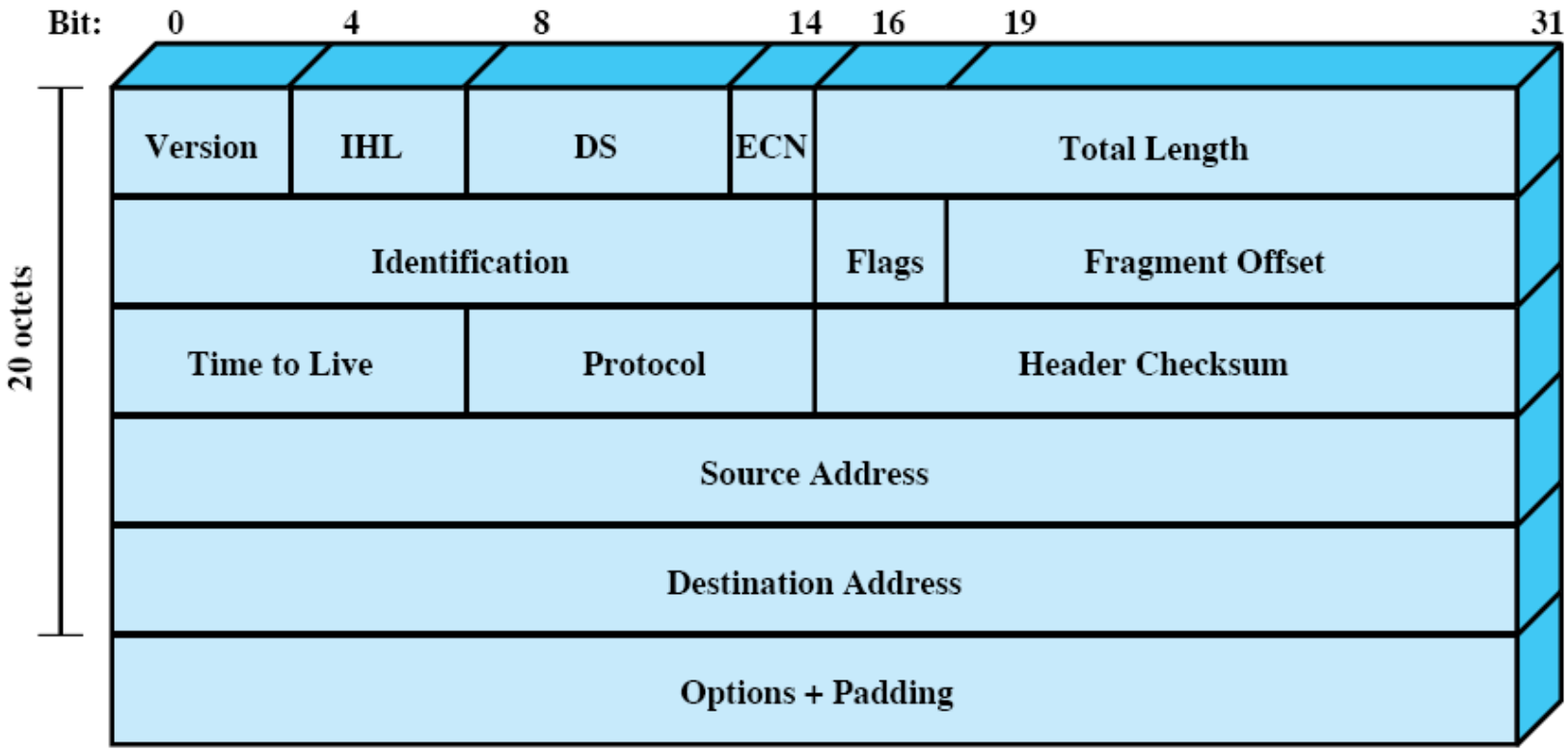
- Do not send a copy of the packet to a neighbour who already has a copy
  - Requires a sequence number in the packet so that each node knows if it has received a copy of this before
- Limit the number of hops a packet can be sent
  - Reduces the number of messages, but not guaranteed of reaching the destination
  - Requires a hop count, or Time To Live, in the packet
    - TTL starts as a value X; each node that receives the packet reduces the value by 1. If TTL = 0, do not send; otherwise send to neighbours
- Do not send to all your neighbours
  - E.g. select N of your M neighbours, either randomly or by some algorithm
  - Reduces the number of messages, but the path taken will not be optimal

The Internet Protocol

IP Overview

- IP is a network layer protocol
  - It is implemented on all hosts and routers in the Internet
    - For a computer to connect to the Internet, it *must* implement IP
- IP provides a simple send/receive service to the transport layer protocol
  - Send:
    - Transport layer sends data to the IP layer
    - IP adds a header, including source and destination address
    - IP sends the datagram (header + data) to Data Link layer
      - The Data Link layer destination is either the destination host or the next router in the path
  - Receive:
    - Data Link layer sends datagram to IP layer
    - IP checks
      - If destination, then IP removes header and sends data to transport layer
      - If not destination, then send datagram to next node in path
  - The protocol itself is quite simple: the main parts are the header format and addressing scheme

IP Header



Note: For our purposes, octet = byte

IP Header Fields

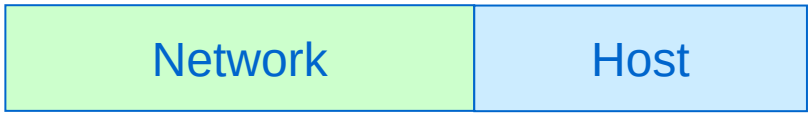
- **Version**: version number of IP; value is 4 (IPv4)
- **Internet Header Length** (IHL): length of header, measured in 4 byte words; minimum value is 5 (20 bytes)
- **DS/ECN** (or ToS): Used for quality of service control. Differentiated Service, Explicit Congestion Notification, Type of Service
- **Total Length**: total length of the datagram, including header, measured in bytes
- **Identification**: sequence number for datagram
- **Flags**: 2 bits are used for Fragmentation and Re-assembly, the third bit is not used
- **Fragment Offset**: Indicates where this fragment belongs in the original datagram (used for Fragmentation and Re-assembly)
- **Time To Live**: how long datagram should remain in internet
- **Protocol**: indicates the next higher layer protocol with a code
  - E.g. TCP = 6; UDP = 17; ICMP = 1

IP Header Fields

- **Header Checksum**: error-detecting code applied to header only (to check for errors in the header); recomputed at each router
- **Source Address**: IP address of source host
- **Destination Address**: IP address of destination host
- **Options**: variable length fields to include options
- **Padding**: used to ensure datagram is multiple of 4 bytes in length
- **Data**: variable length of the data. Maximum length of Data plus Header is 65,535 bytes

IP Addresses

- IP addresses (used to identify source and destination) are 32-bit addresses generally consisting of network portion identifier and host portion identifier



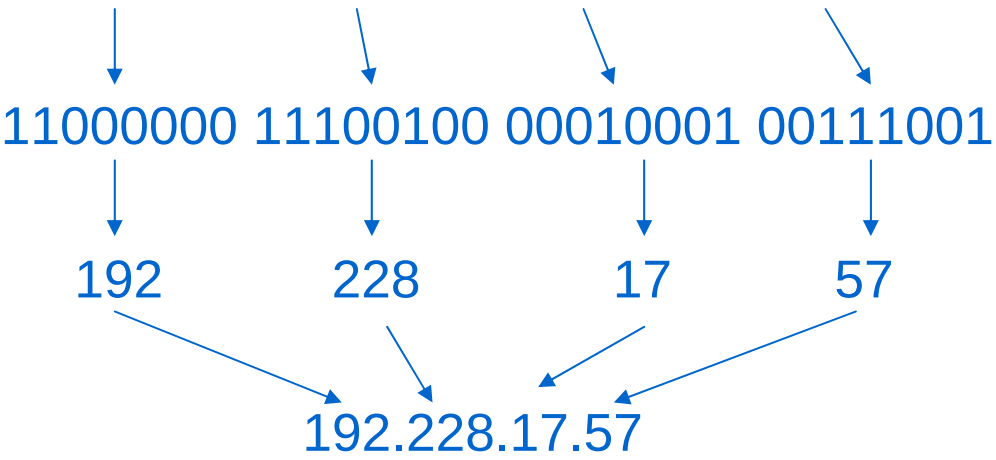
- 32-bits gives 4.2×10^9 possible values
- But IP addresses have different structures (and these have changed over time)
  - First, the set of addresses was separated into five different classes (Classful addressing)
  - Then in 1980's, for organisations to have multiple IP networks (e.g. SIIT Bangkadi is one network, SIIT Rangsit is another), subnet addressing was introduced
  - Then in 1990's, classless addressing was allowed so can fully utilise the address space

Why network and host portion?

- Splitting the IP address into two parts allows for hierarchical addressing. This makes routing in the global Internet possible!
- Example:
  - A reminder: routing protocols provide information to routers about how to reach destinations
    - E.g. “if the destination is D, then send to the next router R”
  - If we did not split the IP address into two ports (that is, flat addressing) then routers must know routes to *hosts*
    - “if the destination is *host H*, then send to the next router R”
      - But on a network, there may be 100’s or 1000’s of hosts
      - Worst case: Routers must know routes to every host on Internet (100,000,000+)
  - But with hierarchical addressing, routers only need to know routes to *networks*
    - “if the destination is *network N*, then send to the next router R”
    - Routers only need to know about hosts on their own network
      - Worst case: routers must know routes to every network on Internet (100,000)

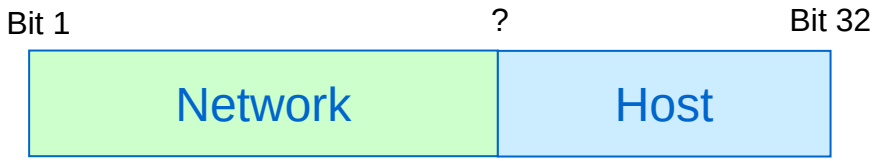
Representing IP Addresses

- Writing (and remembering) 32 bits is difficult!
  - 11000000111001000001000100111001
- IP addresses are usually written in dotted decimal notation
  - Decimal number represents of the bytes of the 32 bit address
  - Decimal numbers are separated by dots



Classless IP Addressing

- IP address has 32 bits: Where is the split between network and host portion?



- In classless addressing, an additional item, called an address mask or subnet mask indentifies where the split is
  - The mask is 32 bits: a bit 1 indicates the corresponding bit in the IP address is the network portion; a bit 0 indicates the corresponding bit in the IP address is the host portion

IP address, 130.17.41.129: 10000010 00010001 00101001 10000001
Subnet mask, 255.255.252.0: 11111111 11111111 11111100 00000000

Network portion | Host portion

Network, 130.17.40.0: 10000010 00010001 00101000 00000000

  - The mask can be given in dotted decimal form or a shortened form, which counts the number of 1 bits
    - The above example can be written as /22, and the IP address as 130.17.41.129/22

Special Cases for IP Addresses

- There are special case addresses that cannot be used to identify a particular host:
  - Network Address
    - The bits of the Host portion are 0
    - Used to identify the network, e.g. for routers to send to a network
    - E.g. host 130.17.41.129/22 is on the network 130.17.40.0
  - Broadcast Address (Directed)
    - The bits of the Host portion are 1
    - Used as a destination for broadcast directed to a specific network
    - E.g. host 130.17.41.129/22 sends to 116.42.2.255/24, then all hosts on network 116.42.2.0/24 will receive the datagram
  - Loopback Address
    - The first 8 bits of Network portion are 01111111 (decimal: 127)
    - Used as a destination address when a host sends to itself
    - E.g. host 130.17.41.129/22 sends to 127.0.0.1, then the datagram will not be sent on the network, but instead to itself (130.17.41.129)
  - Local Broadcast Address
    - All 32 bits are 1 (255.255.255.255)
    - Used as a destination for broadcast to the local network
    - E.g. host 130.17.41.129/22 sends to 255.255.255.255, then all hosts on network 130.17.40.0/22 will receive the datagram
  - Startup Source Address
    - All 32 bits are 0 (0.0.0.0)
    - Used as a source address by a host if the host doesn't know its own IP address
    - E.g. host sends an address to a known server (or local broadcast address) *asking for its own IP address*; 0.0.0.0 is used as the source

Performance in the Internet

Performance Metrics

- How do we know if a network “performs well”?
- Three important metrics are:
  - Throughput [bits per second]
    - Rate at which useful data is delivered to the destination
    - Very important to data and multimedia applications
  - Delay [seconds]
    - Time taken to deliver data from the source to destination
    - Very important to multimedia (real-time, interactive) applications; important for data applications
  - Jitter [seconds]
    - Variance in delay
    - Important to multimedia applications
- Many other related metrics:
  - Overhead, efficiency, fairness, spectral efficiency, computational complexity, energy consumption, …
- Of course, all performance metrics must be considered with cost