PROTOCOLS AND LAYERING

BY NTALASHA

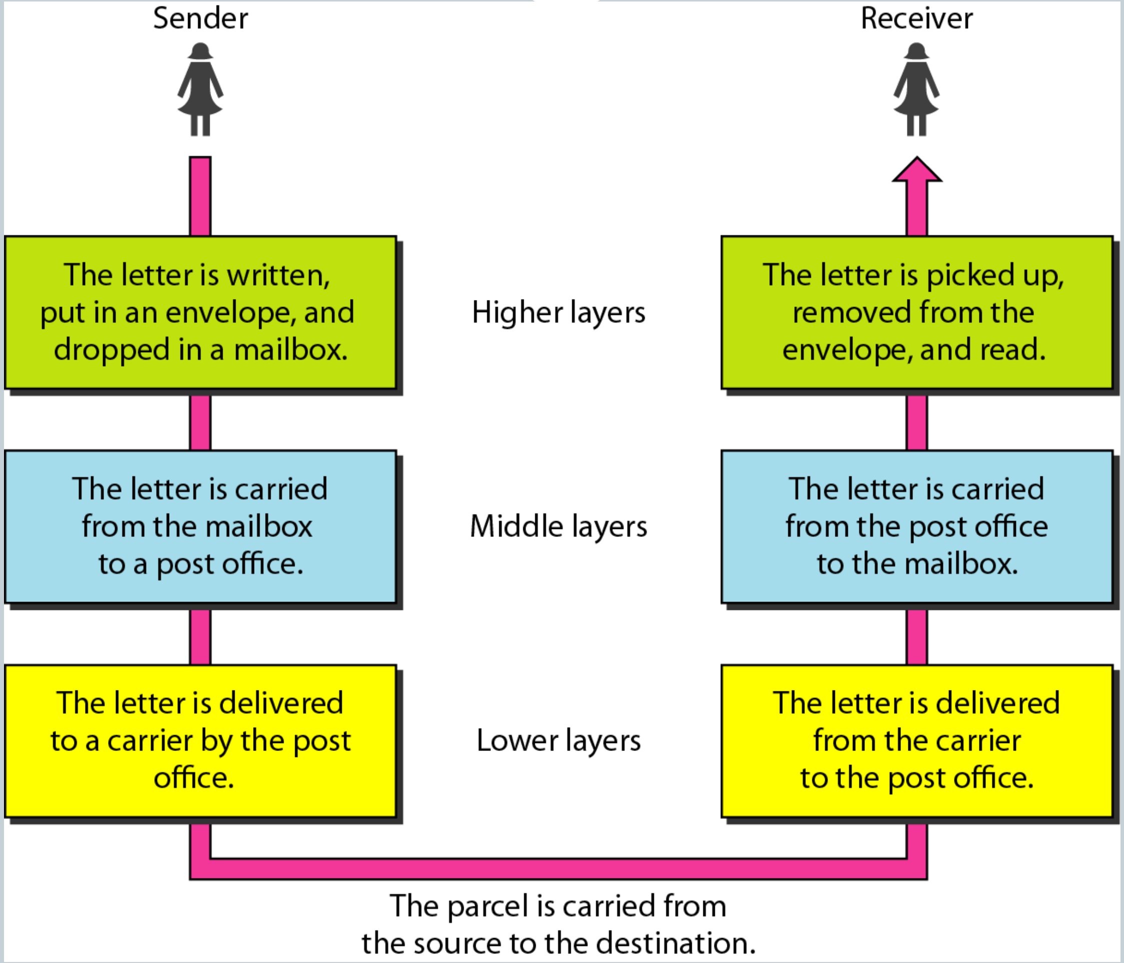
Introduction

- LAN/WAN hardware can't solve all computer communication problems
- Software for LAN and WAN systems is large and complicated
- *Layering* is a structuring technique to organize networking software design and implementation

LAYERED TASKS

- *We use the concept of layers in our daily life. As an example, let us consider two friends who communicate through postal mail.*
- *The process of sending a letter to a friend would be complex if there were no services available from the post office*

Tasks involved in sending a letter



Why network software?

- Sending data through raw hardware is awkward and inconvenient - doesn't match programming paradigms well
- Equivalent to accessing files by making calls to disk controller to position read/write head and accessing individual sectors
- May not be able to send data to every destination of interest without other assistance
- Network software provides high-level interface to applications

Why protocols?

- Name is derived from the Greek *protokollen*, the index to a scroll
- Diplomats use rules, called protocols, as guides to formal interactions
- A *network protocol* or *computer communication protocol* is a set of rules that specify the format and meaning of messages exchanged between computers across a network
  - Format is sometimes called *syntax*
  - Meaning is sometimes called *semantics*
- Protocols are implemented by *protocol software*

One or many protocols?

- Computer communication across a network is a very hard problem
- Complexity requires multiple protocols, each of which manages a part of the problem
- May be simple or complex; must all work together

Protocol suites

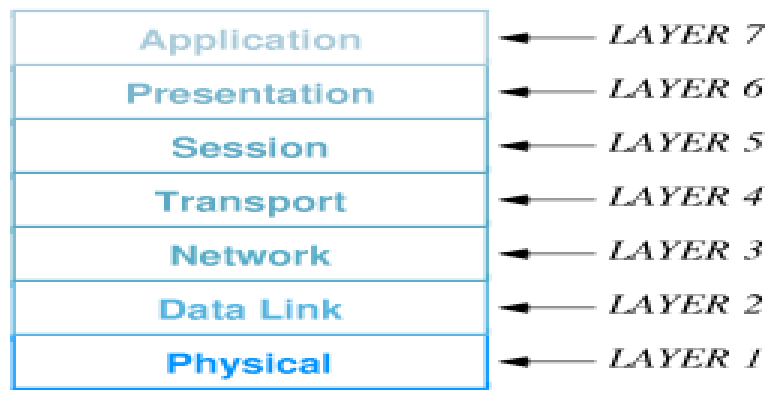
- A set of related protocols that are designed for compatibility is called a *protocol suite*
- Protocol suite designers:
  - Analyze communication problem
  - Divide problems into subproblems
  - Design a protocol for each subproblem
- A well-designed protocol suite
  - Is efficient and effective - solves the problem without redundancy and makes best use of network capacity
  - Allows replacement of individual protocols without changes to other protocols

Layered Protocol Design

- *Layering model* is a solution to the problem of complexity in network protocols
- Model suggests dividing the network protocol into *layers*, each of which solves part of the network communication problem
- These layers have several constraints, which ease the design problem
- Network protocol designed to have a protocol or protocols for each layer

The ISO 7-layer reference model

- *International Organization for Standards* (ISO) defined a *7-layer reference model* as a guide to the design of a network protocol suite
- Layers are named and numbered; reference to ``layer *n*'' often means the n^{th} layer of the ISO 7-layer reference model



The OSI Model

- *Established in 1947, the International Standards Organization (ISO) is a multinational body dedicated to worldwide agreement on international standards.*
- *An ISO standard that covers all aspects of network communications is the Open Systems Interconnection (OSI) model.*
- *It was first introduced in the late 1970s*

ISO is the organization.
OSI is the model.

The layers in the ISO model

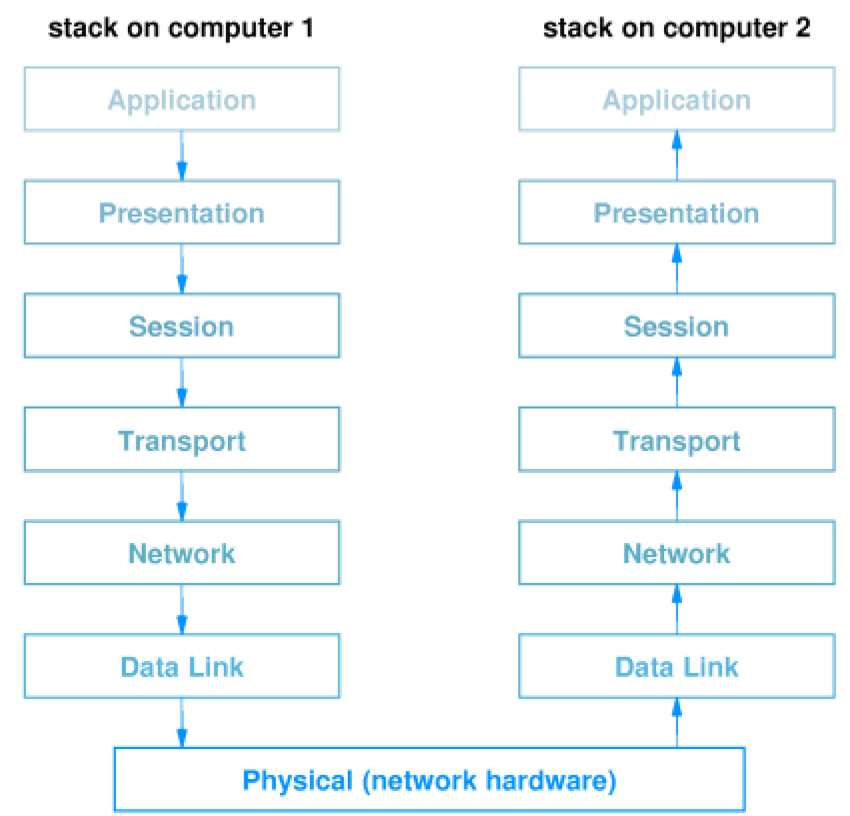
- Caveat - many modern protocols do not exactly fit the ISO model, and the ISO protocol suite is mostly of historic interest
- Concepts are still largely useful and terminology persists

Layers

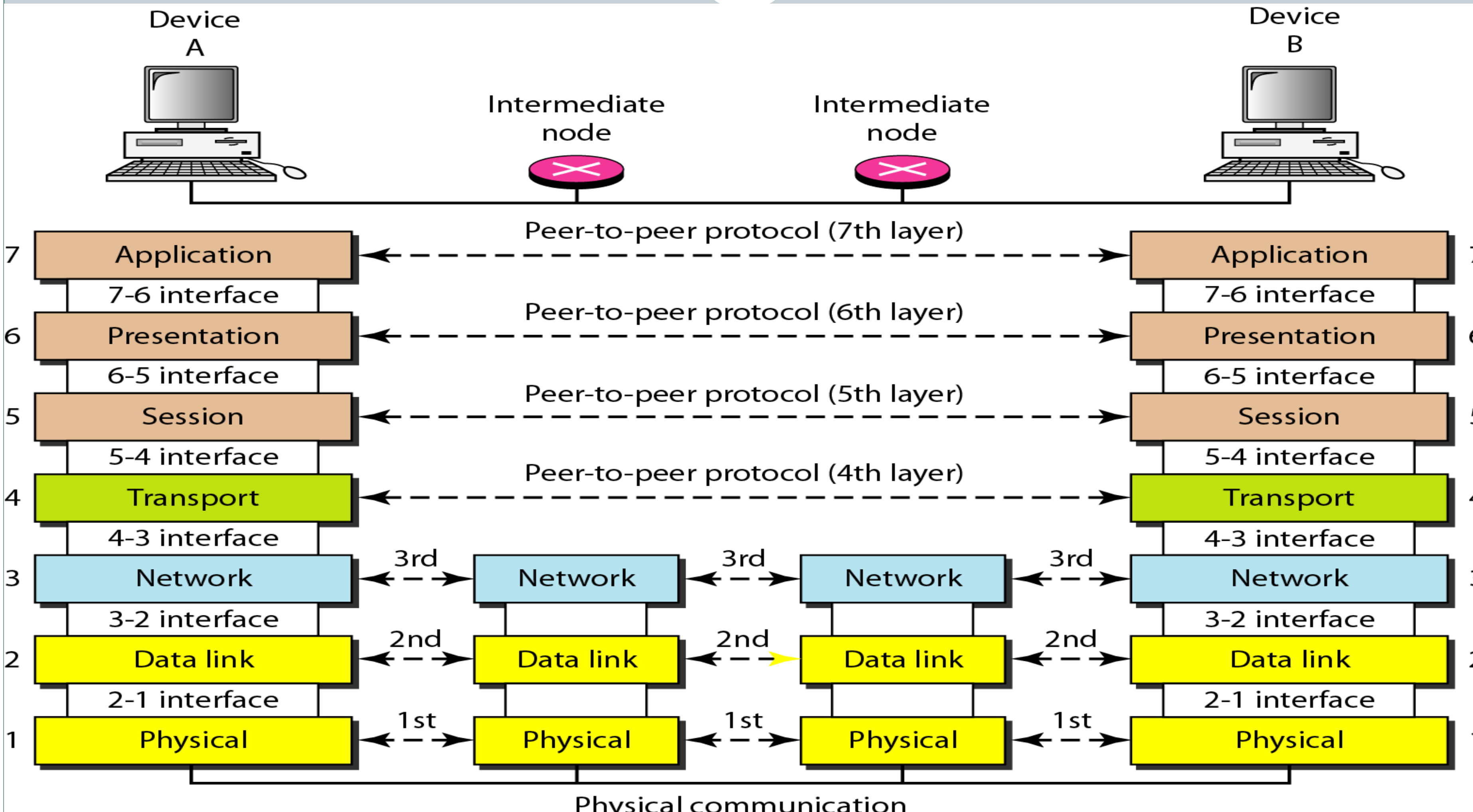
- Layer 7: Application Application-specific protocols such as FTP and SMTP (electronic mail)
- Layer 6: Presentation Common formats for representation of data
- Layer 5: Session Management of sessions such as login to a remote computer
- Layer 4: Transport Reliable delivery of data between computers
- Layer 3: Network Address assignment and data delivery across a physical network
- Layer 2: Data Link Format of data in frames and delivery of frames through network interface
- Layer 1: Physical Basic network hardware - such as RS-232 or Ethernet

Layered software implementation

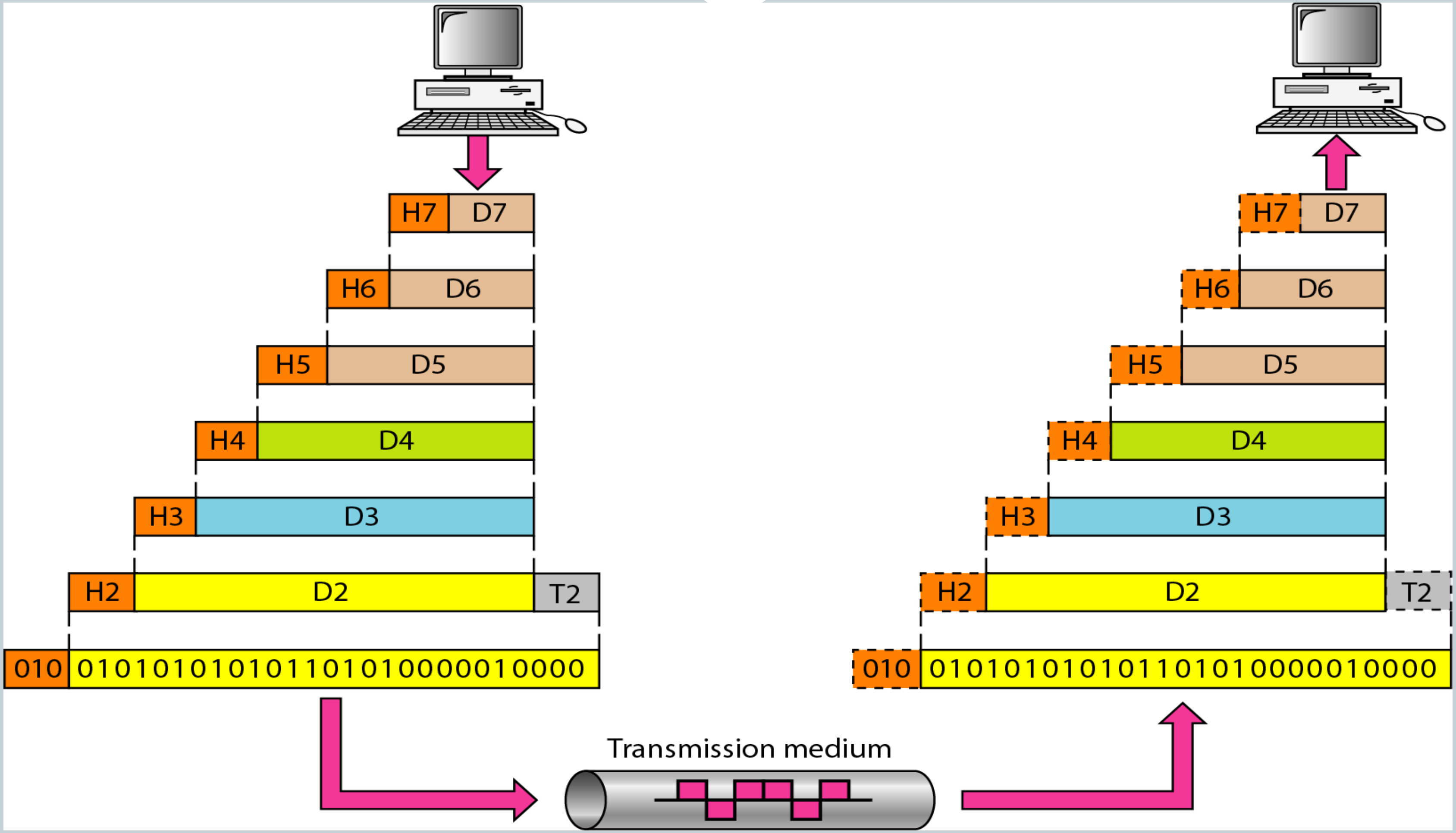
- Software implemented from layered design has layered organization
- Software modules can be viewed as:



The interaction between layers in the OSI model

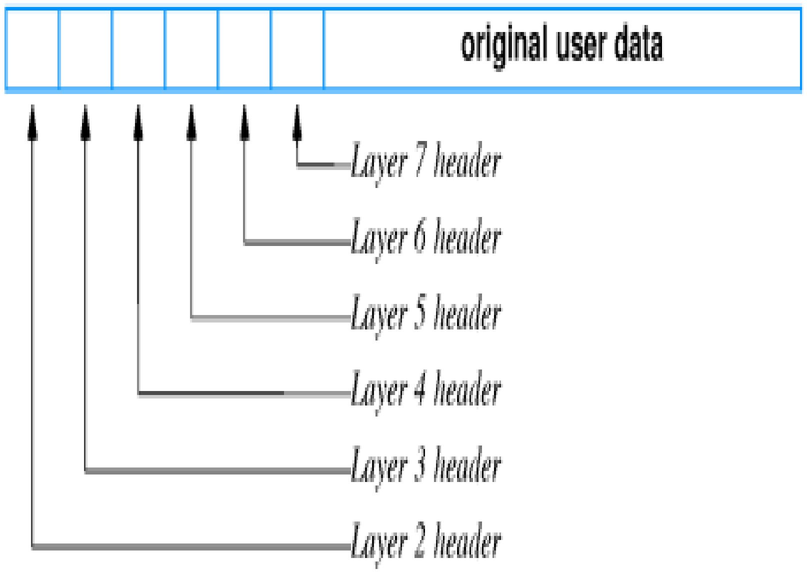


An exchange using the OSI model



Protocol headers

- The software at each layer communicates with the corresponding layer through information stored in headers
- Each layer adds its header to the front of the message from the next higher layer
- Headers are nested at the front of the message as the message traverses the network



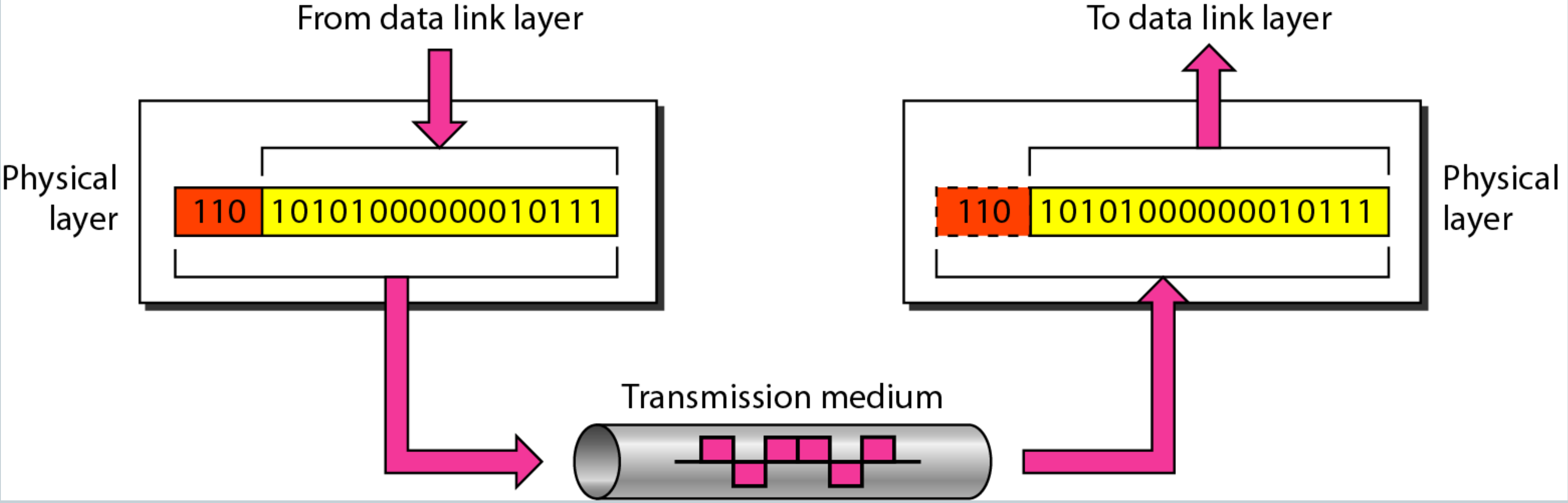
Layered software and stacks

- Related modules from previous figure are called a *protocol stack* or simply a *stack*
- Two constraints:
  - The software for each layer depends only on the services of the software provided by lower layers
  - The software at layer *n* at the destination receives exactly the same protocol message sent by layer *n* at the sender
- These constraints mean that protocols can be tested independently and can be replaced within a protocol stack

LAYERS IN THE OSI MODEL

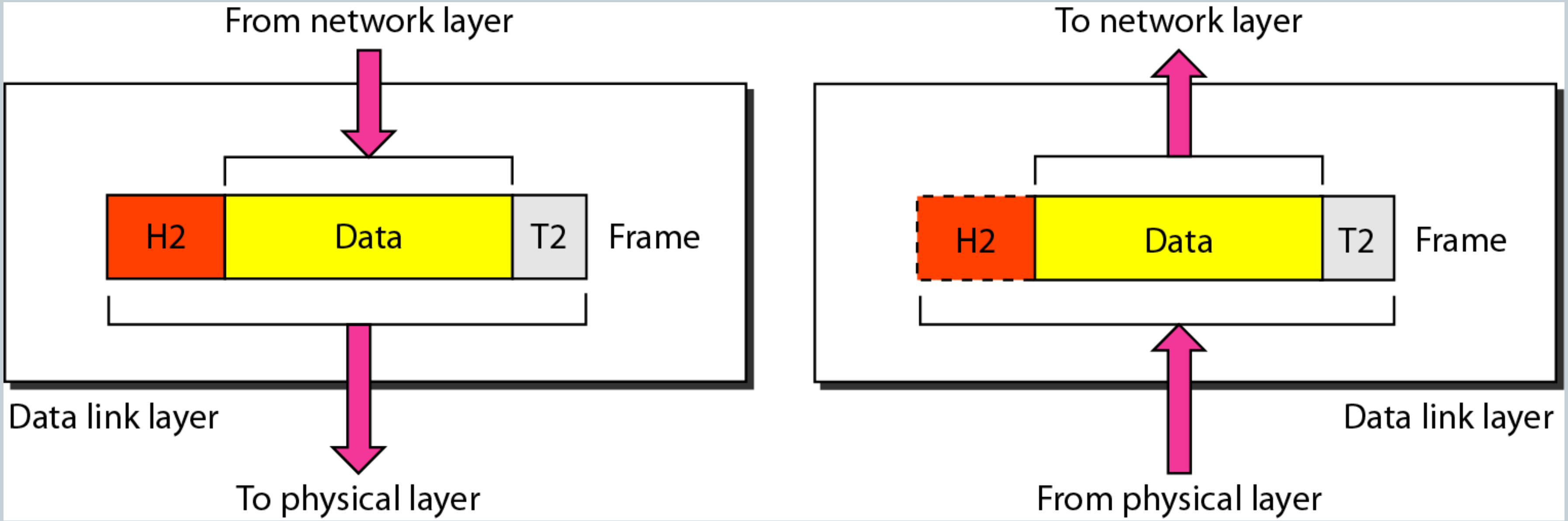
- *In this section we briefly describe the functions of each layer in the OSI model.*
- Physical Layer
- Data Link Layer
- Network Layer
- Transport Layer
- Session Layer
- Presentation Layer
- Application Layer

Physical layer



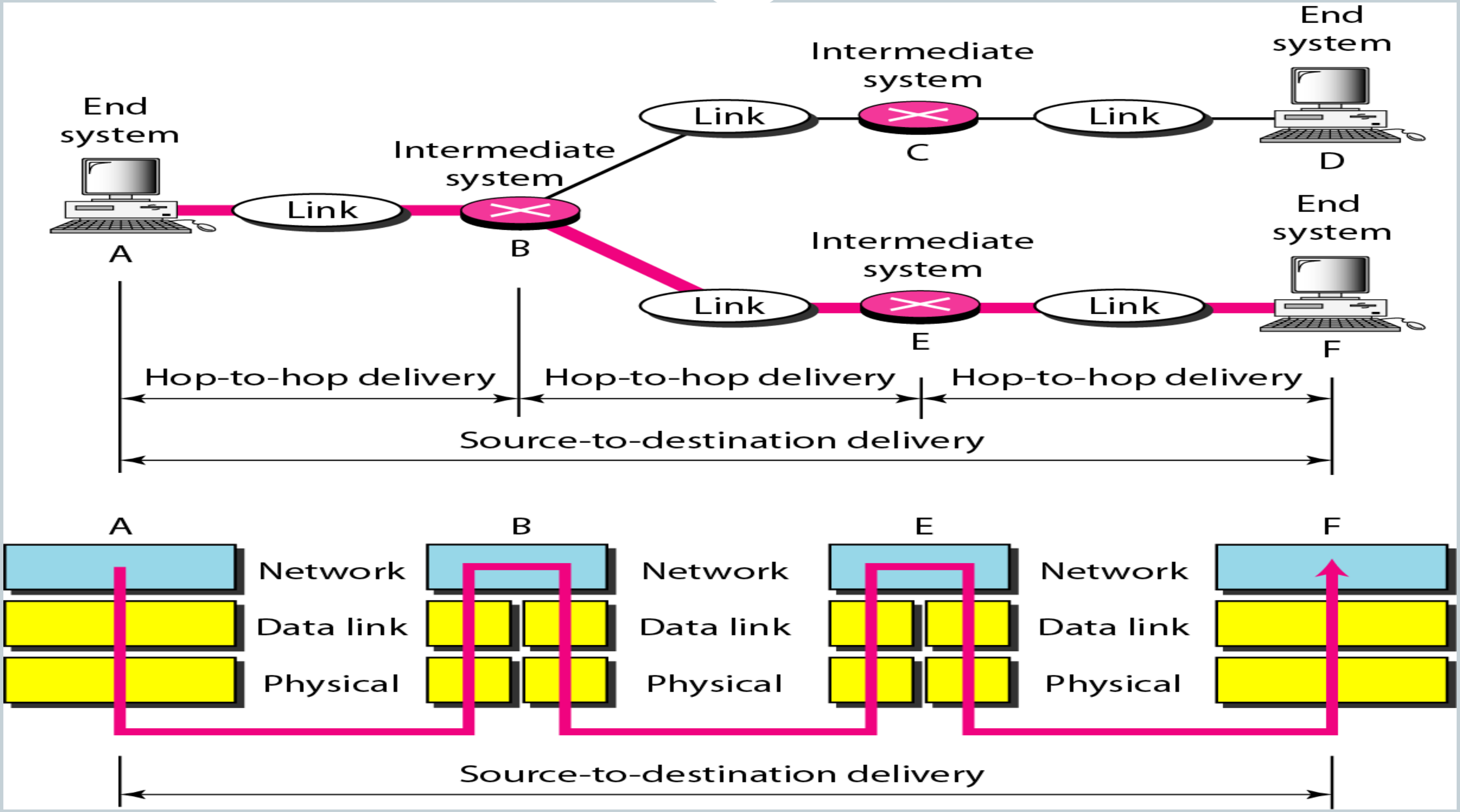
The physical layer is responsible for movements of individual bits from one hop (node) to the next.

Data link layer

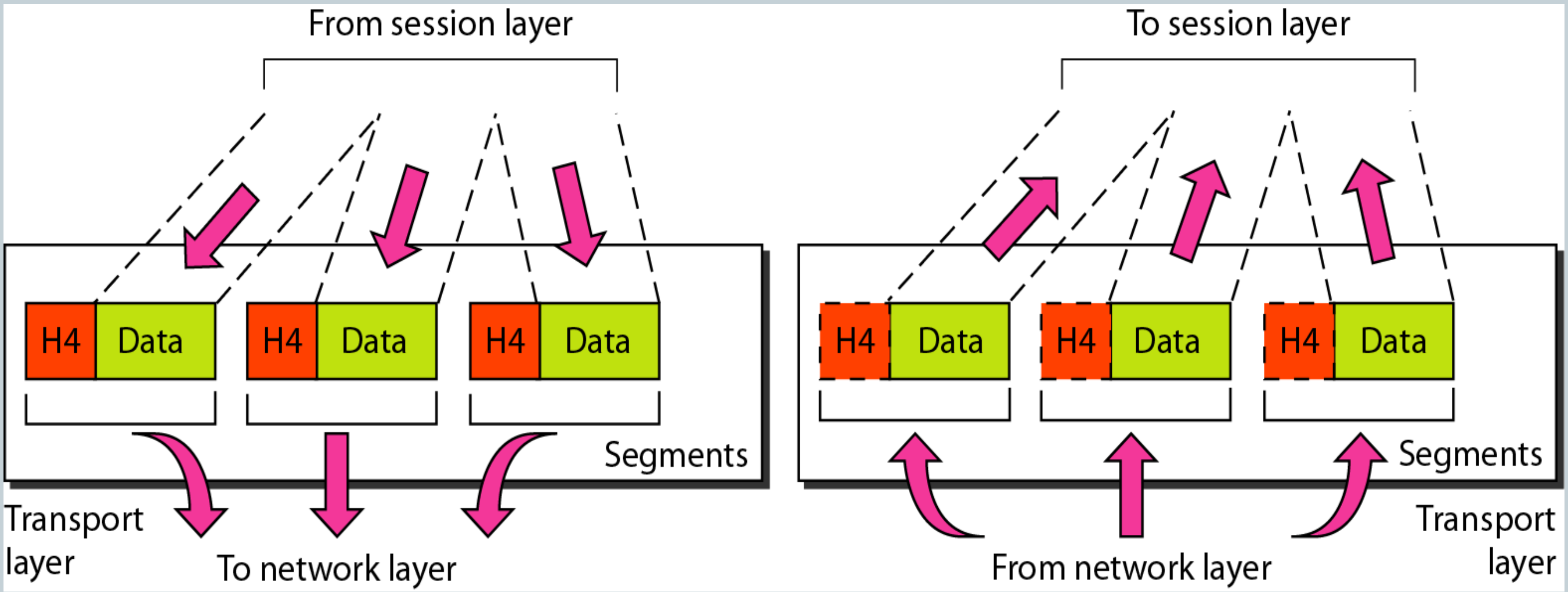


The data link layer is responsible for moving frames from one hop (node) to the next.

Source-to-destination

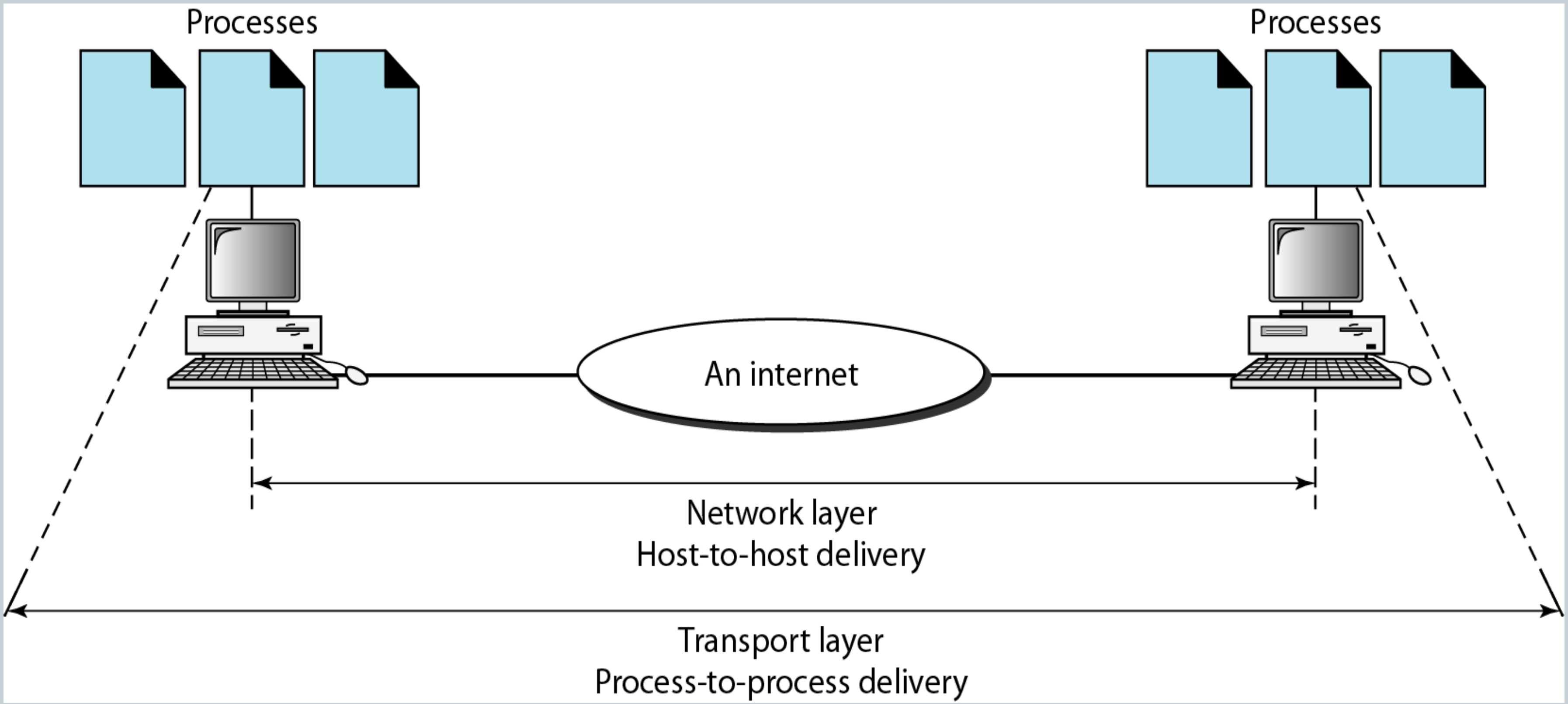


Transport layer

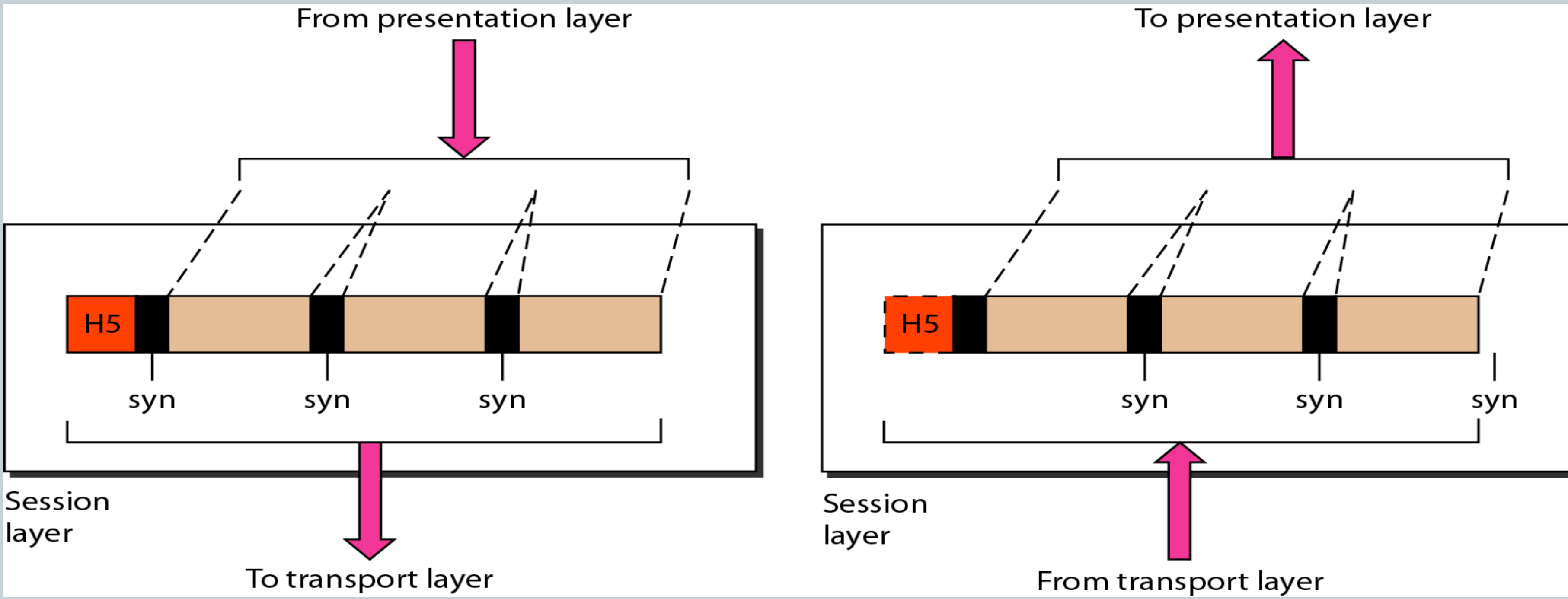


The transport layer is responsible for the delivery of a message from one process to another.

Reliable process-to-process delivery of a message

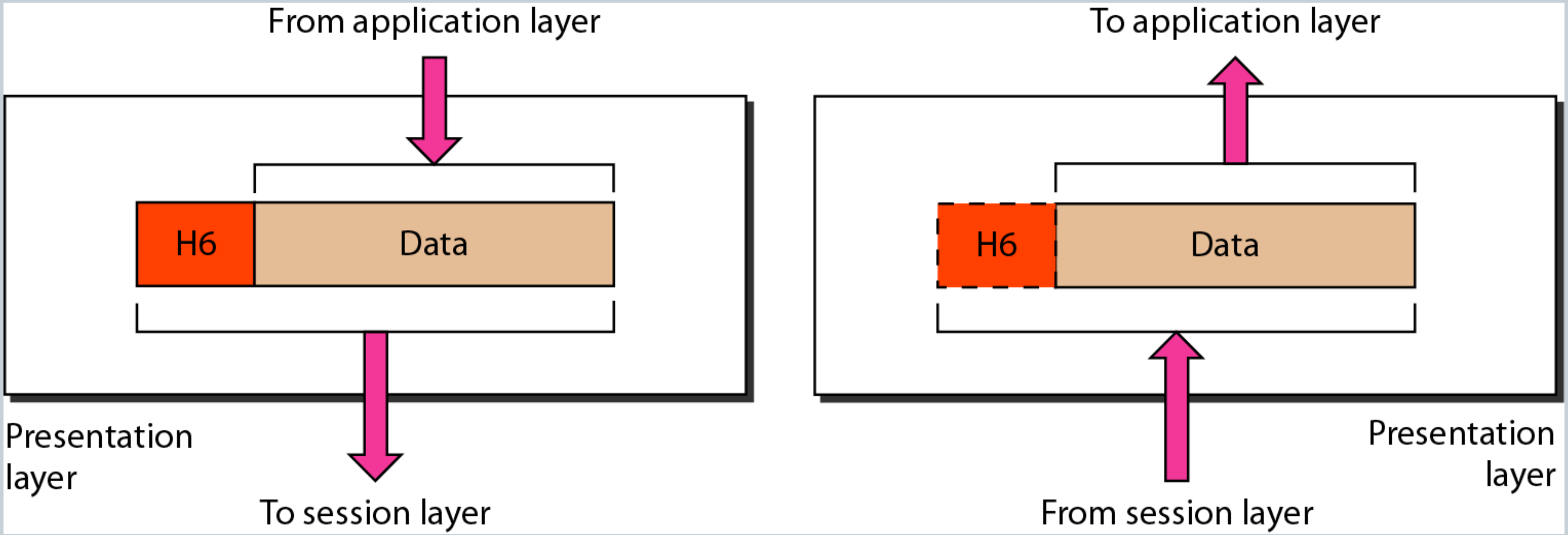


Session layer



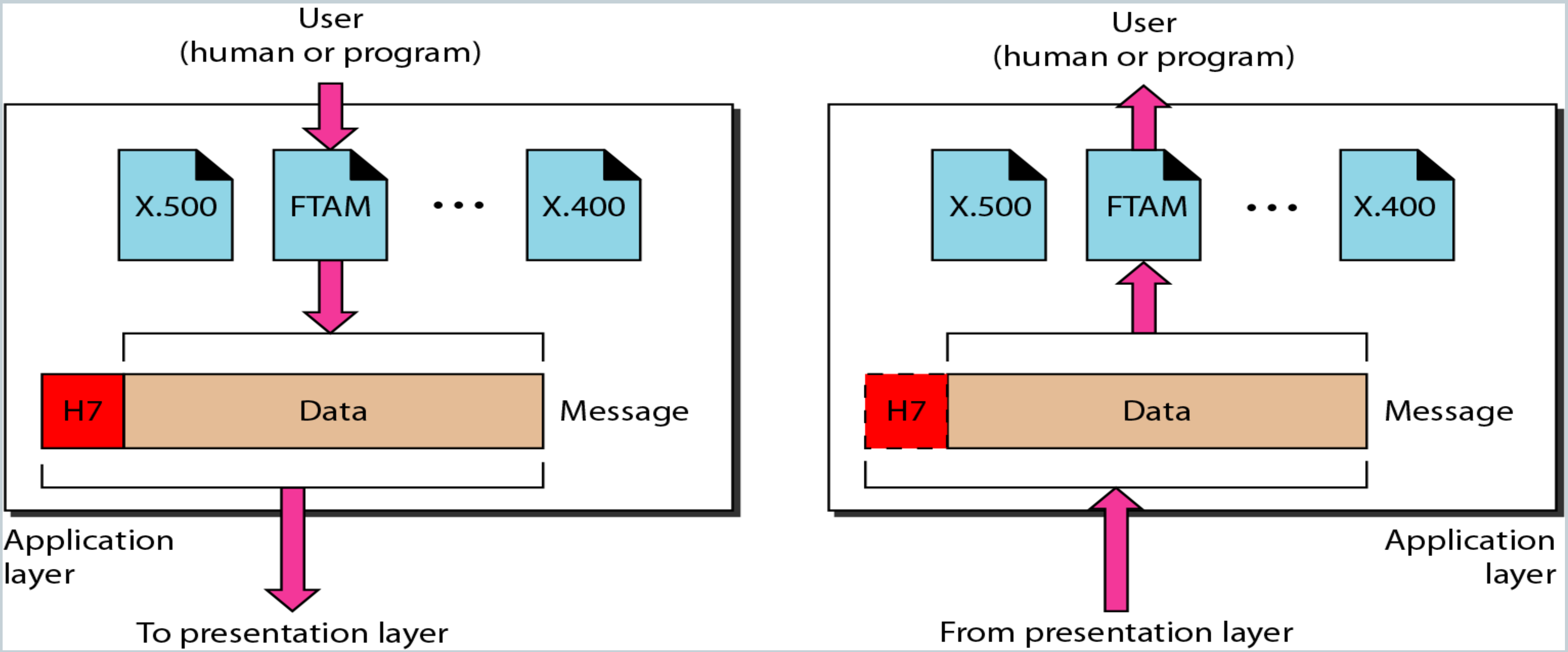
The session layer is responsible for dialog control and synchronization.

Presentation layer



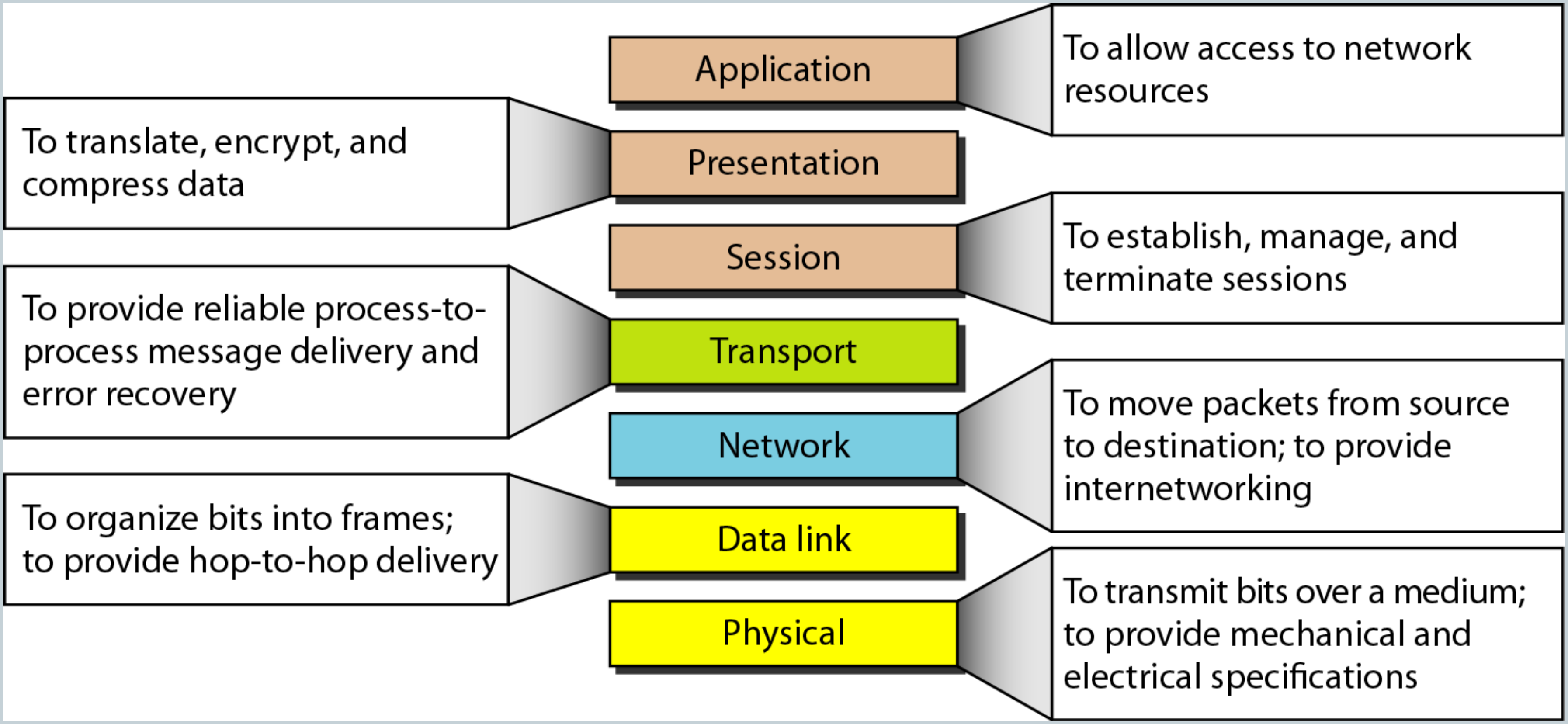
The presentation layer is responsible for translation, compression, and encryption.

Application layer



The application layer is responsible for providing services to the user.

Summary of layers



Messages and protocol stacks

- On the sender, each layer:
  - Accepts an outgoing message from the layer above
  - Adds a header and other processing
  - Passes resulting message to next lower layer
- On the receiver, each layer:
  - Receives an incoming message from the layer below
  - Removes the header for that layer and performs other processing
  - Passes the resulting message to the next higher layer

Commercial stacks

Vendor	Stack
Novell Corporation	Netware
Banyan System Corporation	VINES
Apple Computer Corporation	AppleTalk
Digital Equipment Corporation	DECNET
IBM	SNA
(many vendors)	TCP/IP

Control packets

- Protocol layers often need to communicate directly without exchanging data
  - Acknowledge incoming data
  - Request next data packet
- Layers use *control packets*
  - Generated by layer *n* on sender
  - Interpreted by layer *n* on receiver
  - Transmitted like any other packet by layers *n-1* and below

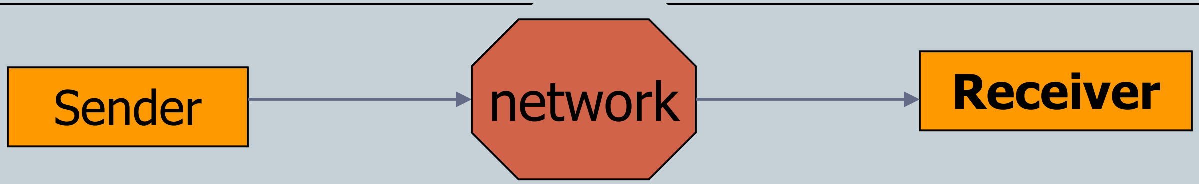
Techniques for reliable network communication

- Model - *reliable* delivery of a block of data from one computer to another
  - Data values unchanged
  - Data in order
  - No missing data
  - No duplicated data
- Example - parity bit, checksum and CRC used to ensure data is unchanged

Protocol Techniques

- For **bit corruption**: parity, LRC, checksum, CRC
- For **out-of-order** delivery: adding sequence numbers to packets
- For **duplicated packets**: using the sequence numbers
- For **lost packets**: sending acknowledge (ACK) information and retransmission
- For **reply delay** (excessive delay): unique message ID
- For **data overrun**: **flow control**
- * Stop-and-go flow control
- * Sliding window flow control

$P_5 P_4 P_3 P_2 P_1$ $P_5 P_5 P_2 P_3 P_1$



Rate **Ds** in sending

Rate **Dr** receiver can process

Data overrun if **Ds > Dr**

Network throughput **Dn** (**> Ds**)

Out-of-order delivery

- Packets may be delivered out of order - especially in systems that include multiple networks
- Out of order delivery can be detected and corrected through *sequencing*
  - Sender attaches sequence number to each outgoing packet
  - Received uses sequence numbers to put packets in order and detect missing packets

Duplicate delivery

- Packets may be duplicated during transmission
- Sequencing can be used to…
  - Detect duplicate packets with duplicated sequence numbers
  - Discard those duplicate packets

Lost packets

- Perhaps the most widespread problem is lost packets
- Any error - bit error, incorrect length - causes receiver to discard packet
- Tough problem to solve - how does the receiver decide when a packet has been lost?

Retransmission

- Protocols use *positive acknowledgment with retransmission* to detect and correct lost packets Receiver sends short message acknowledging receipt of packets
- Sender infers lost packets from missing acknowledgments
- Sender *retransmits* lost packets
- Sender sets timer for each outgoing packet Saves copy of packet
- If timer expires before acknowledgment is received, sender can retransmit saved copy
- Protocols define upper bound on retransmission to detect unrecoverable network

Replay

- To prevent replay, protocols mark each session with a unique ID (e.g., the time the session was established), and require the unique ID to be present in each packet.
- The protocol software discards any arriving packet that contains an incorrect ID.
- To avoid replay, an ID must not be reused until a reasonable time has passed (e.g., hours).

Flow Control

- Data overrun occurs when a computer sends data across a network faster than the destination can absorb it.
- Consequently, data is lost. Several techniques are available to handle data overrun.
- Collectively, the techniques are known as *flow control* mechanisms.
- The simplest form of flow control is a *stop-and-go* system in which a sender waits after transmitting each packet.
- When the receiver is ready for another packet, the receiver sends a control message, usually a form of acknowledgement.

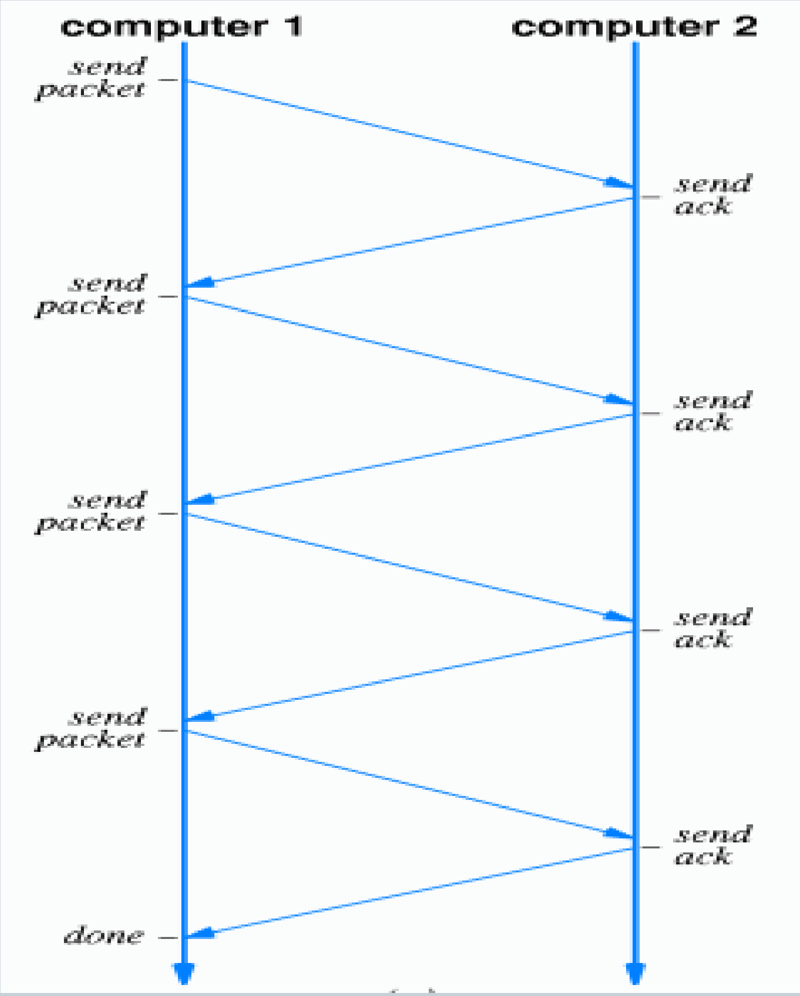
Stop and Go Control flow

- Allow sender sends one packet each time and waits

- Sending side
- * Transmits one packet
- * Waits for acknowledgement from receiver

- Receiving side
- * Receives and consumes packet
- * Transmits acknowledgement to sender

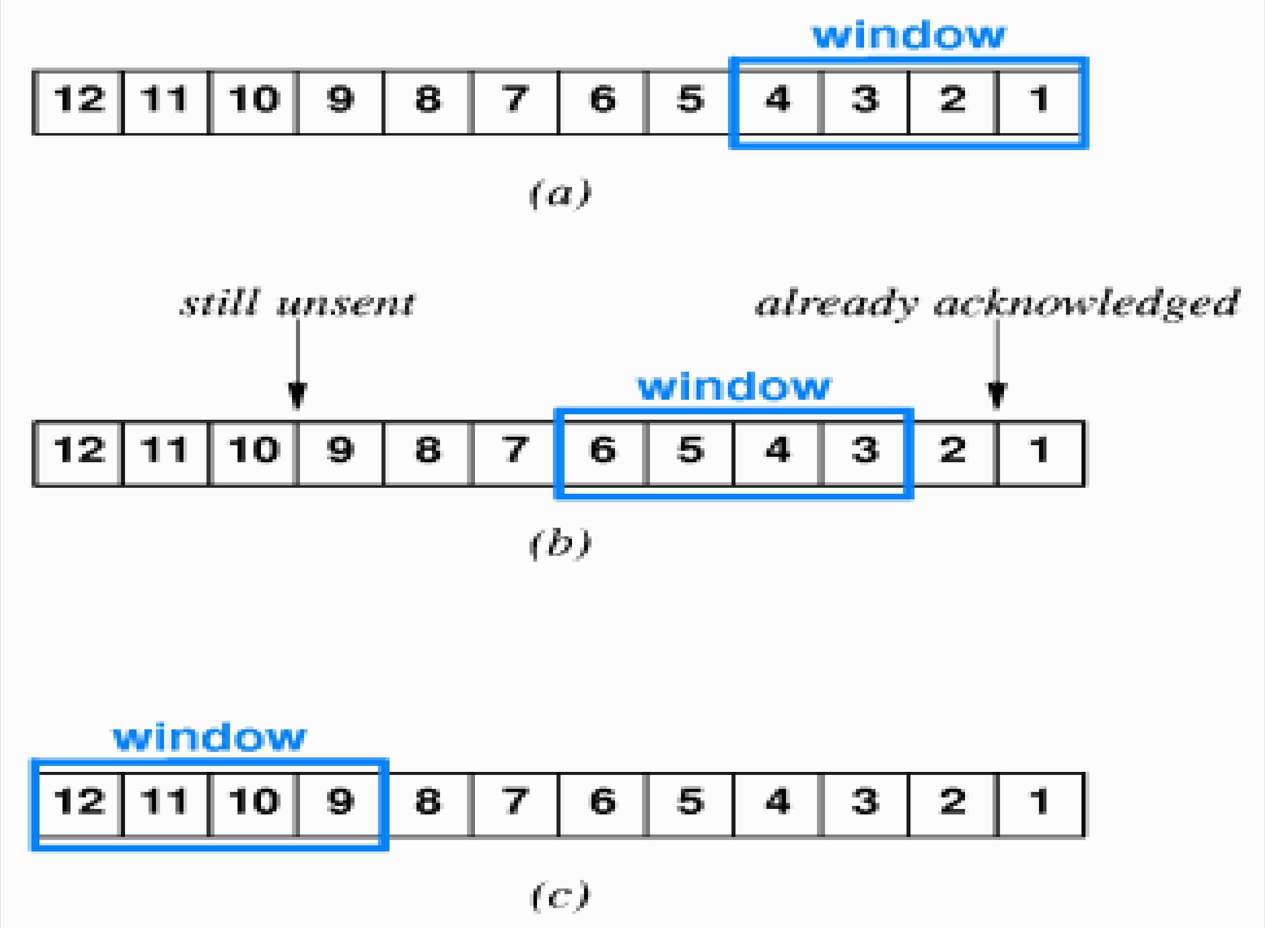
- Inefficient
- * No data sending in waiting time



Sliding Window

- Allows sender to transmit multiple packets before receiving an acknowledgment
- Number of packets that can be sent is defined by the protocol and called the *window*
- Window size is determined by the empty buffer in receiver
- Receiver tells how many packets can be sent
- Sender transmits a number of packets, specified by available window (buffer) size
- Receiver sends acknowledgements as packets arrive

Sliding Window



Comparison of Stop and Go and Sliding Window

- Sliding widow can send data
- much fast than stop-and-go

- For stop-and-go, each packet takes *2L* time to deliver where *L* is the *latency*, or network delivery time.

- Sliding window can improve by number of packets in window:

  $T_w = T_g * W$

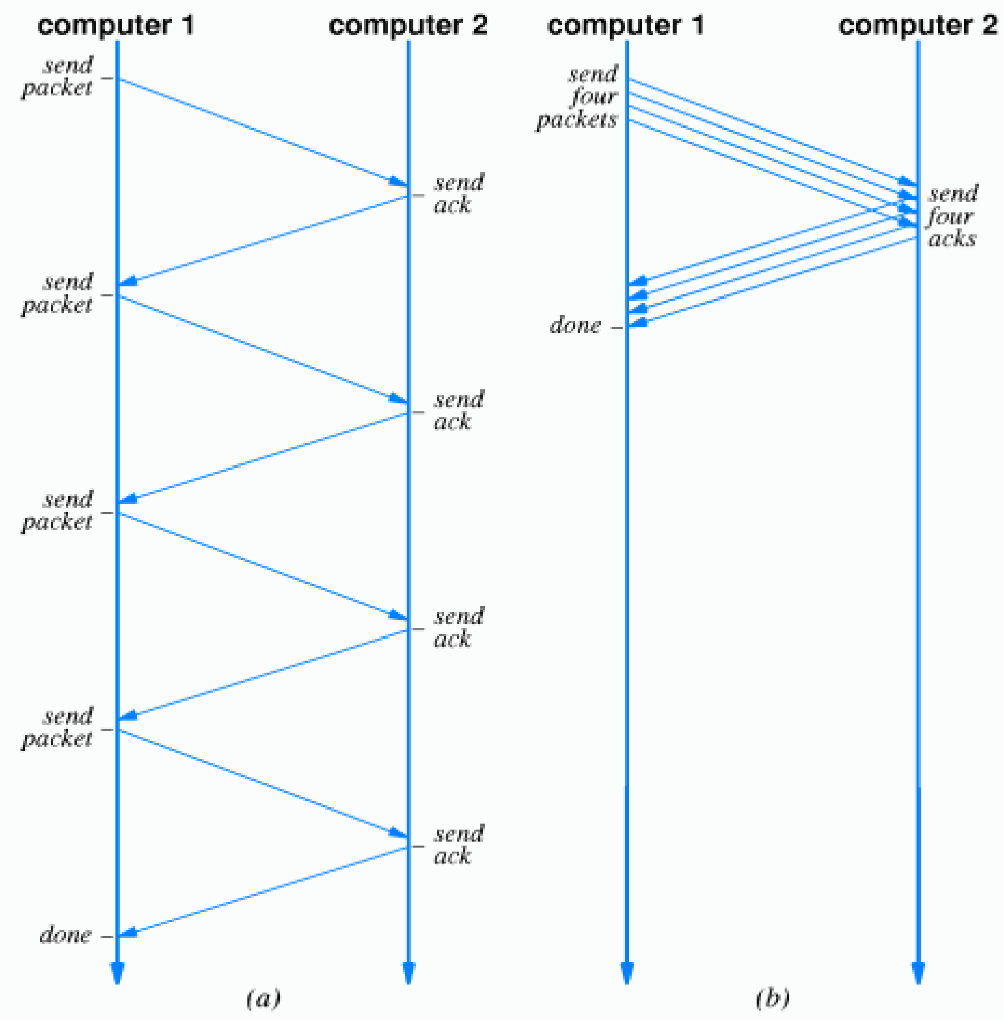
  T_w is sliding window throughput

  T_g is stop-and-go throughput

- Transmission time also limited by network transmission rate:

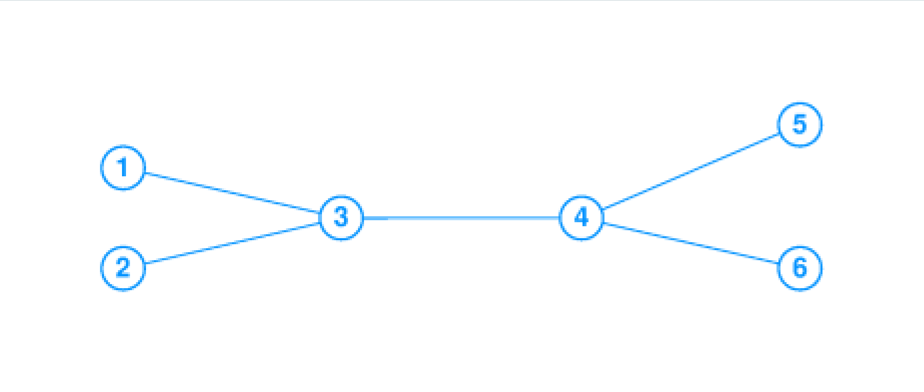
  $T_w = \min(B, T_g * W)$

  B is maximum network bandwidth



Network congestion

- Network congestion arises in network systems that include multiple links
- If input to some link exceeds maximum bandwidth, packets will queue up at connection to that link



Congestion

- Eventually, packets will be discarded and packets will be retransmitted
- Ultimately, network will experience *congestion collapse*
- Problem related to, but not identical to, data overrun

Avoiding and recovering from network congestion

- Protocols attempt to avoid congestion and recover from network collapse by monitoring the state of the network and taking appropriate action
- Can use two techniques:
  - Notification from packet switches
  - Infer congestion from packet loss
- Packet loss can be used to detect congestion because modern networks are reliable and rarely lose packets through hardware failure
- Sender can infer congestion from packet loss through missing acknowledgments
- Rate or percentage of lost packets can be used to gauge degree of congestion

Art, engineering and protocol design

- Protocol design mixes engineering and art
  - There are well-known techniques for solving specific problems
  - Those techniques interact in subtle ways
  - Resulting protocol suite must account for interaction
- Efficiency, effectiveness, economy must all be balanced

Summary

- Layering is a technique for guiding protocol design and implementation
- Protocols are grouped together into related protocol suites
- A collection of layered protocols is called a protocol stack
- Protocols use a variety of techniques for reliable delivery of data