COMP 3721
Introduction to Data Communications

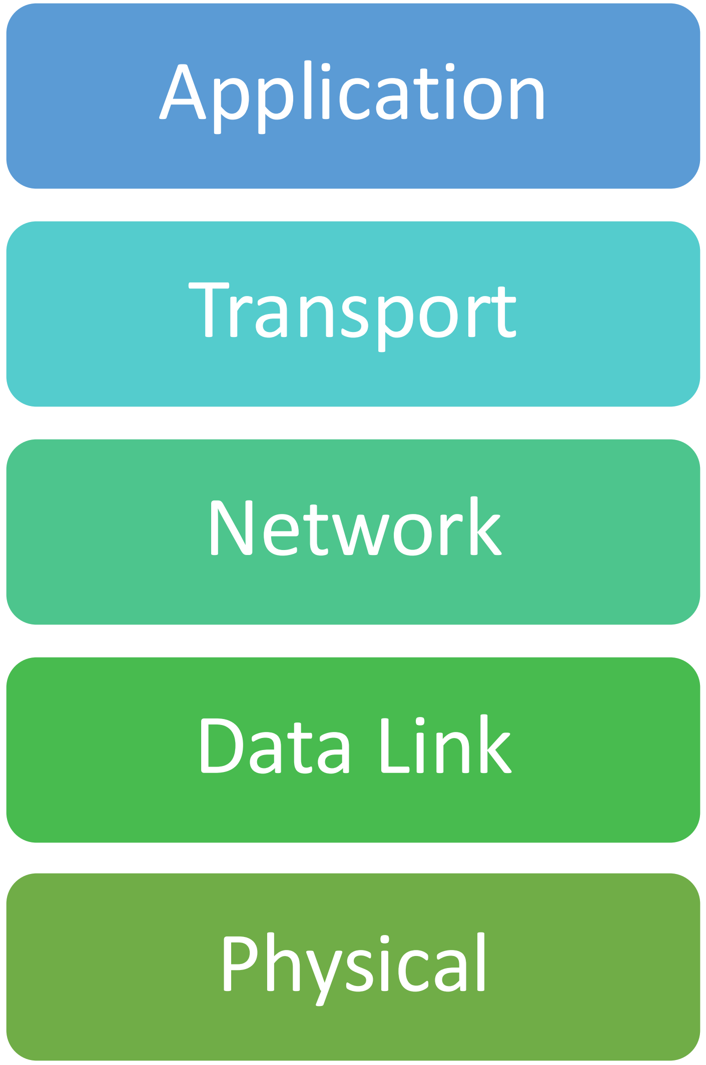
11b - Week 11 - Part 2

Learning Outcomes

- By the end of this lecture, you will be able to
  - Explain what are the transport-layer services.
  - Explain how UDP works and what are its characteristics.

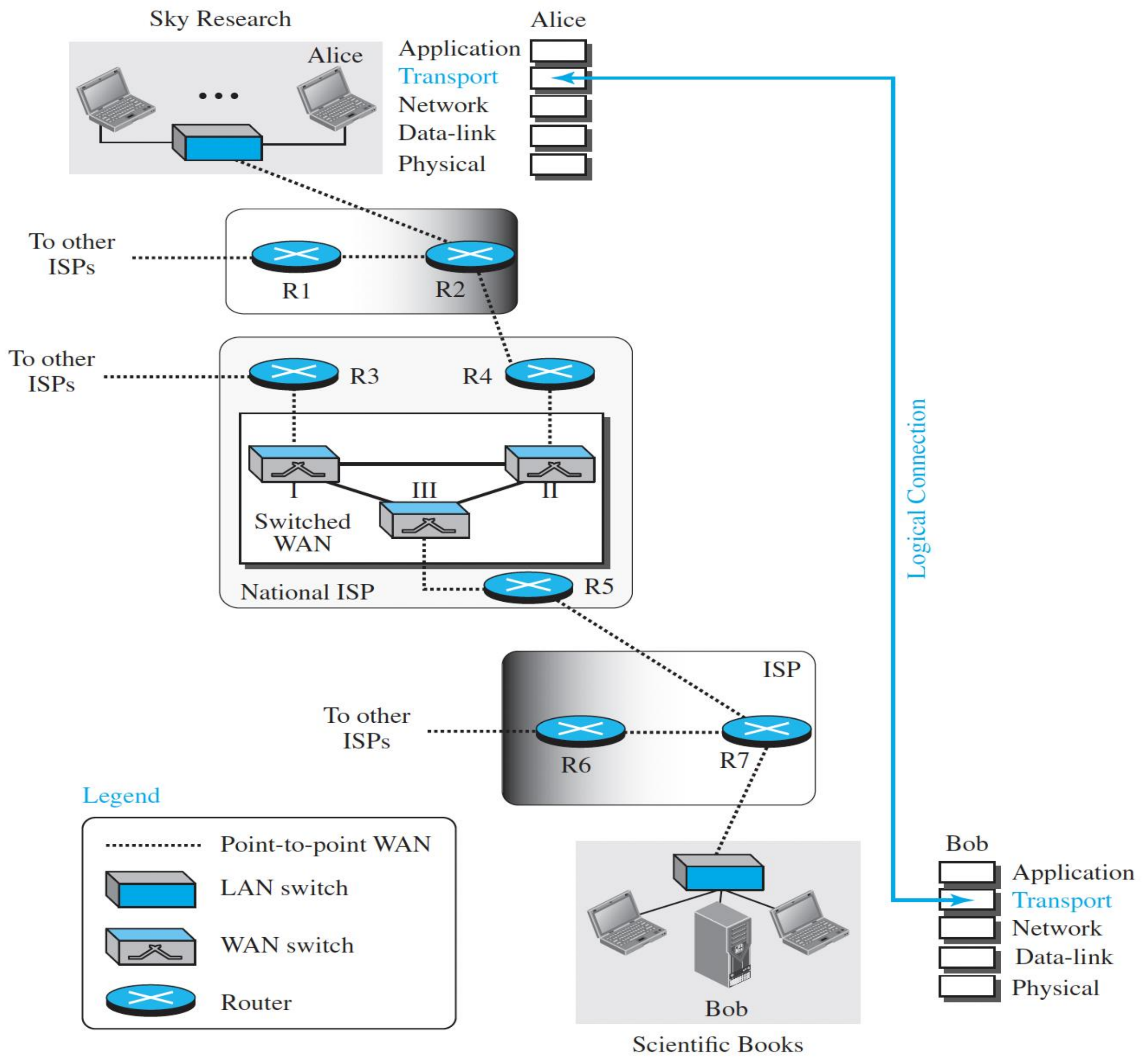
Introduction

- Transport layer **provides services** to the **application layer** and **receives services** from the **network layer**.
- It is the **end-to-end** logical vehicle for transferring data from one point to another in the Internet (i.e., from source to destination).



Logical Connection at the Transport Layer

- Only the two **end systems** (Alice's and Bob's computers) use the services of the **transport layer**.
- All **intermediate routers** use only the **first three layers**.



Transport Layer vs. Network Layer

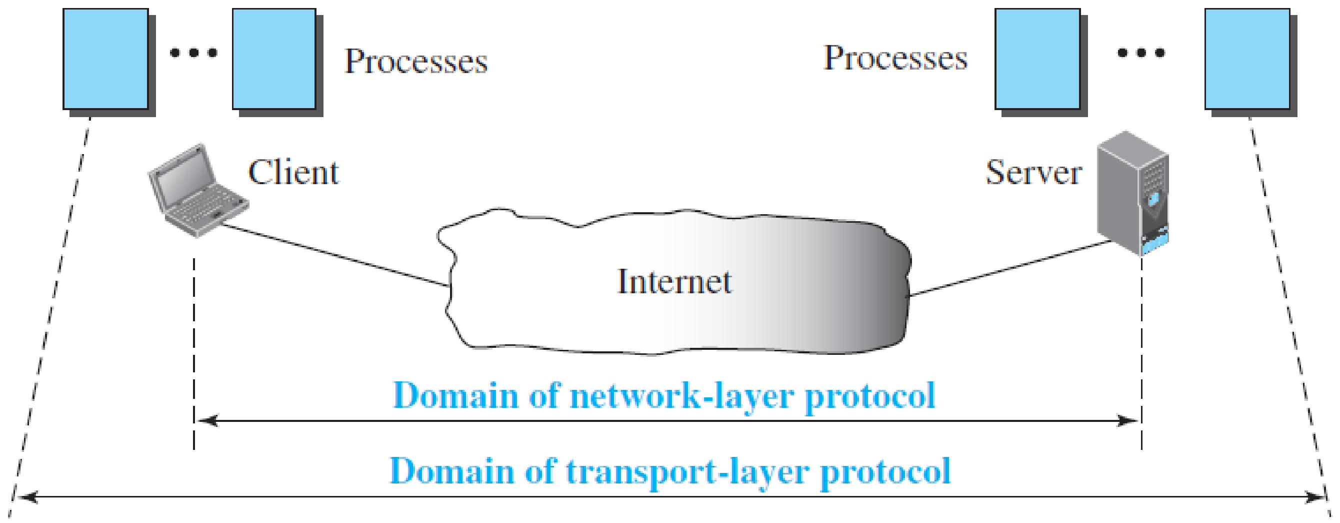
- The network layer is responsible for communication at the computer level (**host-to-host communication**).
  - A network-layer protocol can deliver the message only to the destination computer. However, this is an **incomplete delivery**.
- The message still needs to be handed to the **correct process**.
  - where the transport-layer protocol takes over (**process-to-process communication**).
- A transport-layer protocol is responsible for delivery of the message to the appropriate process.

Transport-Layer Services

- Provide **process-to-process** communication
- Addressing: **Port Numbers**
- Encapsulation and decapsulation
- Multiplexing and demultiplexing
- **Flow control**
- **Error control**

Process-to-Process Communication

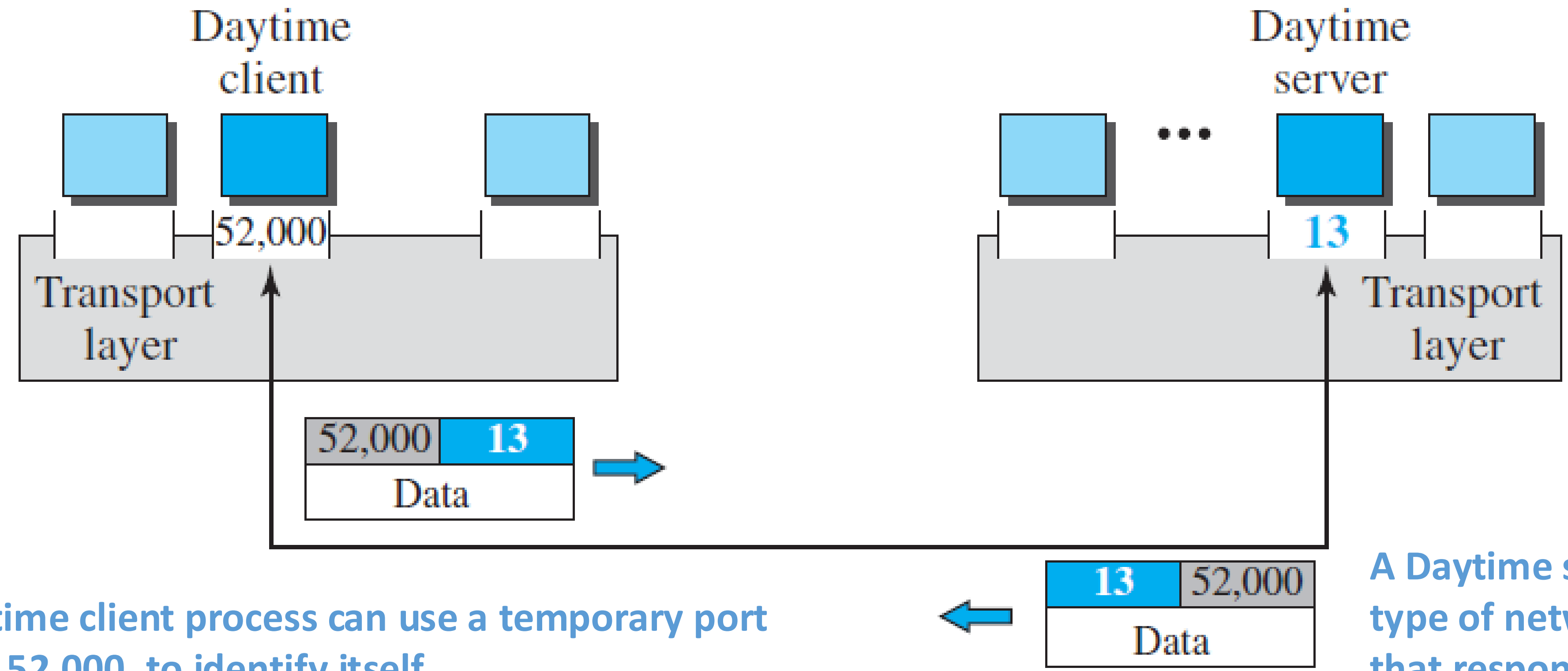
- **Process**: is an application-layer entity (**running program**) that uses the services of the transport layer.
- The most common way to **achieve process-to-process communication** is **client/server paradigm**.
  - A **process-to-process communication** between two application layers is provided, one at the **local host** (**client**) and the other at the **remote host** (**server**).


Processes
Processes
Client
Server
Internet
Domain of network-layer protocol
Domain of transport-layer protocol

Addressing: Port Numbers

- Port numbers provide end-to-end addresses at the transport layer.
- For communication, we must define the local host, local process, remote host, and remote process.
  - Local host and remote host → defined using **IP addresses**
  - Processes → defined using **port numbers**
- **Port numbers**: integers between 0 and 65,535 (16 bits)
- The port number of the **client program** is called the **ephemeral**/**temporary port number**, and it is **greater than 1023**.
- The port number of a **server process** is called a **well-known port number**.
  - Well-known port numbers ≤ 1023

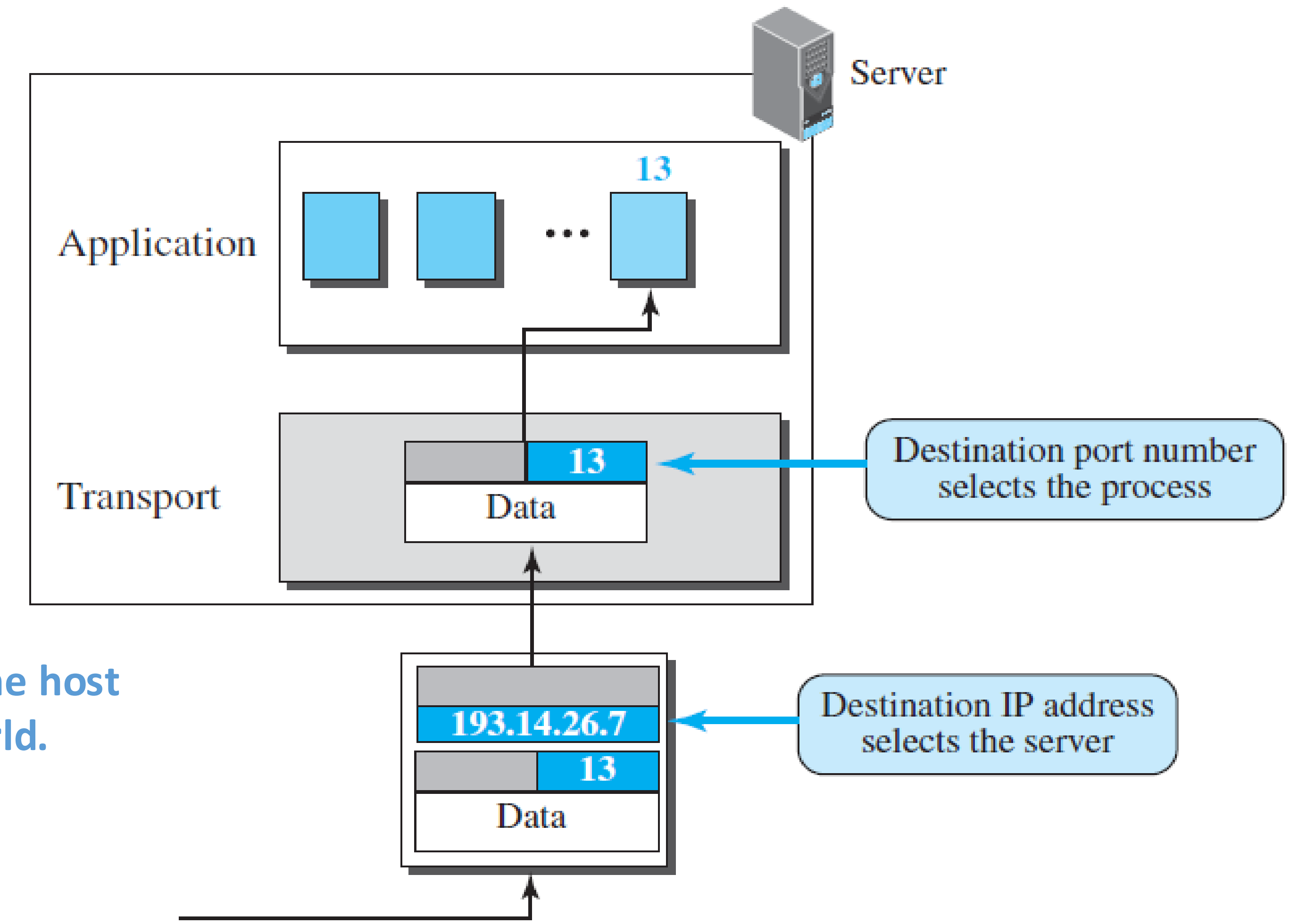
Addressing: Port Numbers



The daytime client process can use a temporary port number, 52,000, to identify itself.
The daytime server process must use the well-known (permanent) port number 13.

A Daytime server is a type of network server that responds to requests from a Daytime client with the current date and time.

IP Addresses vs. Port Numbers

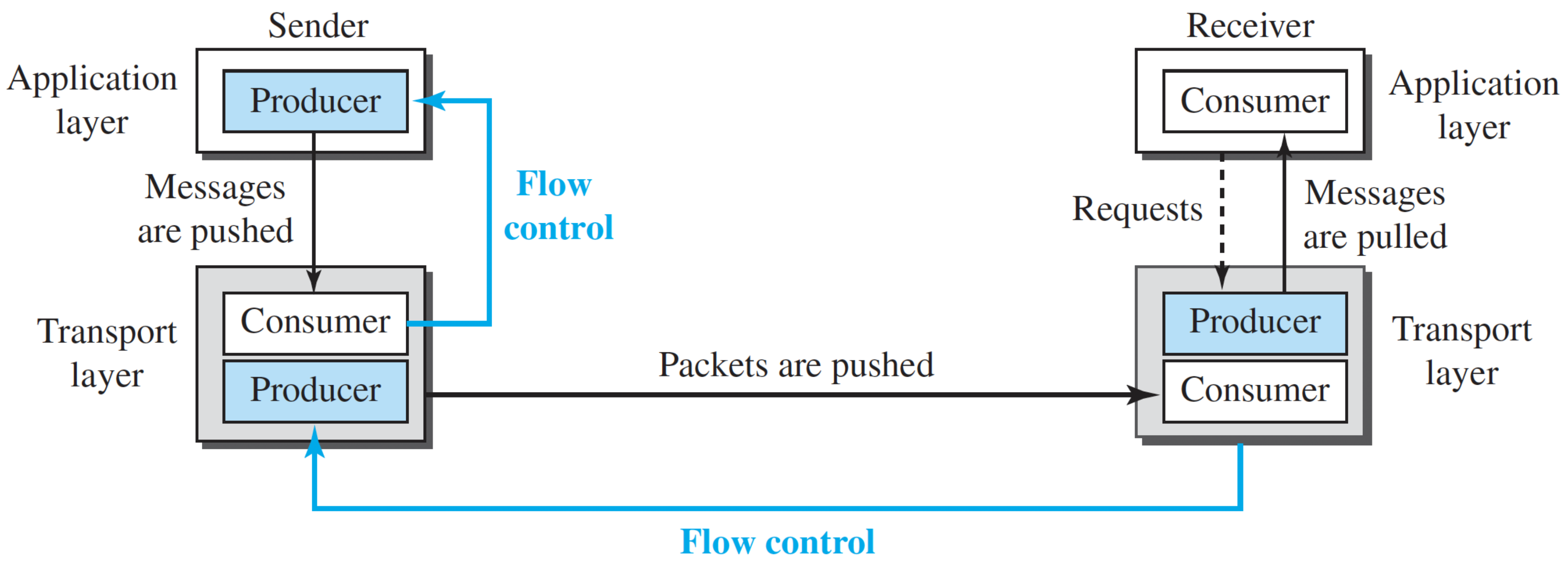

Server
Application
13
Transport
13
Data
Destination port number selects the process
193.14.26.7
13
Data
Destination IP address selects the server

The destination IP address defines the host among the different hosts in the world. The port number defines one of the processes on this particular host.

Socket Address

- A **socket address** = a **combination** of an **IP address** and a **port number**.
- The **client socket address** defines the **client process** uniquely just as the **server socket address** defines the **server process** uniquely.
- To use the services of the transport layer on the Internet, we need **a pair of socket addresses**: the client socket address and the server socket address.

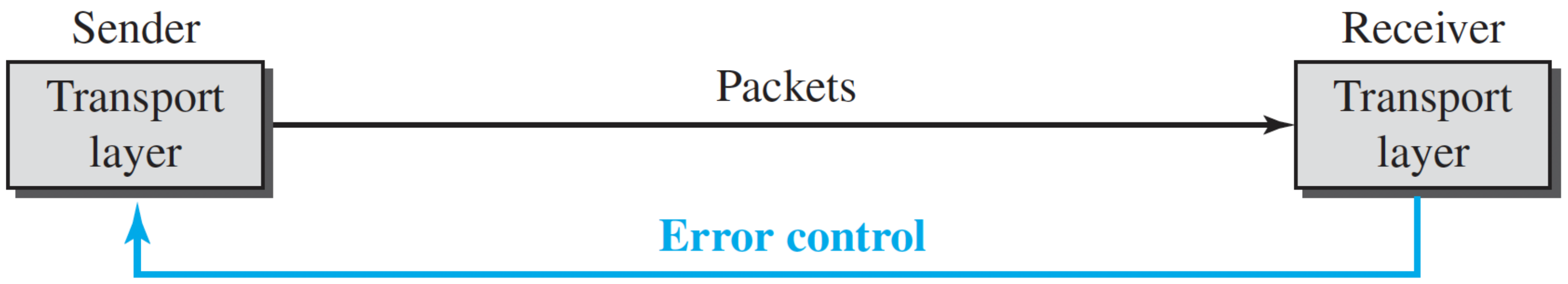
Flow Control



- Main goal of flow control: **Avoid overwhelming the receiver**.
- Flow control can be implemented by using **two buffers**: one at the sending transport layer and the other at the receiving transport layer.
- **Buffer**: a set of memory locations that can hold packets at the sender and receiver.

Error Control

- The underlying **network layer** (IP) is **unreliable**.
- We need to make the transport layer reliable if the application requires reliability.
- **Error control** involves **only** the sending and receiving transport layers.
- The **receiving transport layer** manages error control, most of the time, by **informing** the **sending transport layer** about the problems.
- Error control at the transport layer is responsible for handling:
  - **Corrupted packets**, **duplicate packets**, **lost/discarded packets** and **out-of-order packets**.
- Error control can be achieved by using **sequence numbers** and **acknowledgements**.



Error Control – Sequence Numbers

- **Sequence numbers** are assigned to packets (a field in the header of the transport layer packet).
- The sequence numbers are modulo 2^m, where m is the size of the sequence number field in bits.
- Why are sequence numbers needed?
  - Error control requires that the **sending transport layer** knows **which packet** is to be resent.
    - **Corrupted** or **lost packets**: the receiving transport layer can somehow inform the sending transport layer to resend that packet using the sequence number.
  - Error control requires that the **receiving transport layer** knows **which packet** is a **duplicate**, or which packet has arrived **out of order**.
    - **Duplicate packets**: if two received packets have the same sequence number.
    - **Out-of-order packets**: by observing gaps in the sequence numbers.

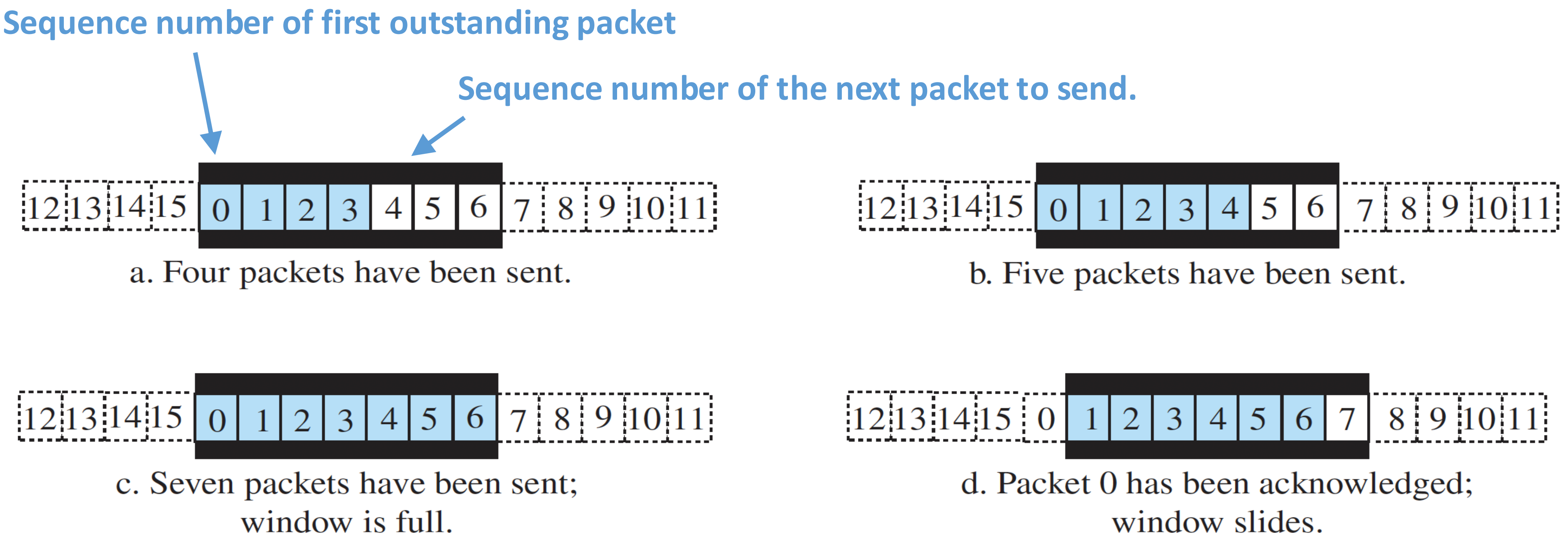
Error Control – Sequence Numbers (Cont.)

- **Duplicate packets** and **corrupted packets** can be silently **discarded** by the receiver.
- **Out-of-order packets** can be either discarded (to be treated as lost packets by the sender) or stored until the missing one arrives.

Error Control – Acknowledgement

- The receiver side can send an **acknowledgment** (**ACK**) for each or a collection of **packets that have arrived safe and sound**.
- The sender can detect **lost packets** if it uses a **timer**. When a packet is sent, the sender starts a timer. If an ACK does not arrive before the timer expires, the sender **resends** the packet.

Combination of Flow and Error Control – Sliding Window Linear Format


Sequence number of first outstanding packet
Sequence number of the next packet to send.
12 13 14 15 0 1 2 3 4 5 6 7 8 9 10 11
a. Four packets have been sent.
12 13 14 15 0 1 2 3 4 5 6 7 8 9 10 11
b. Five packets have been sent.
12 13 14 15 0 1 2 3 4 5 6 7 8 9 10 11
c. Seven packets have been sent; window is full.
12 13 14 15 0 1 2 3 4 5 6 7 8 9 10 11
d. Packet 0 has been acknowledged; window slides.

An outstanding packet is a packet for which no Ack has been received yet.

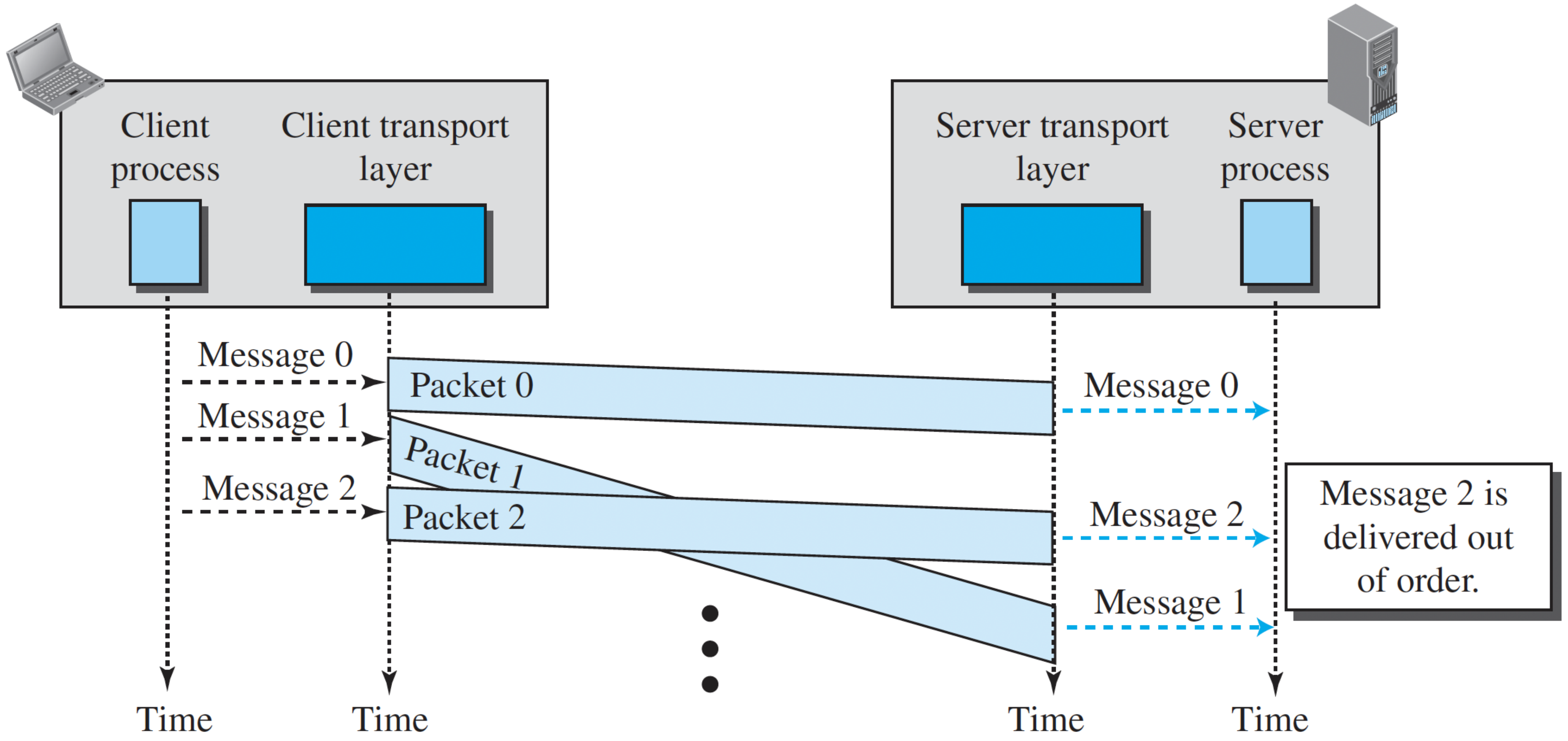
Congestion Control

- **Congestion** in a network may occur if the **load on the network** (i.e., the number of packets sent to the network) is greater than the **capacity of the network** (i.e., the number of packets a network can handle).
- Congestion happens in any system that involves waiting.
  - Routers and switches have queues/buffers that hold the packets **before** and **after** processing.
  - If a router cannot process the packets at the same rate at which they arrive, the queues become overloaded, and congestion occurs.
- **Congestion control** refers to the mechanisms and techniques that control the congestion and keep the load below the capacity.
- Goal of congestion control: **Avoid overwhelming the network**.

Connectionless and Connection-oriented Protocols

- A transport-layer protocol can provide two types of services:
    - **Connectionless service**: no dependency between packets.
    - **Connection-oriented service**: dependency between packets.

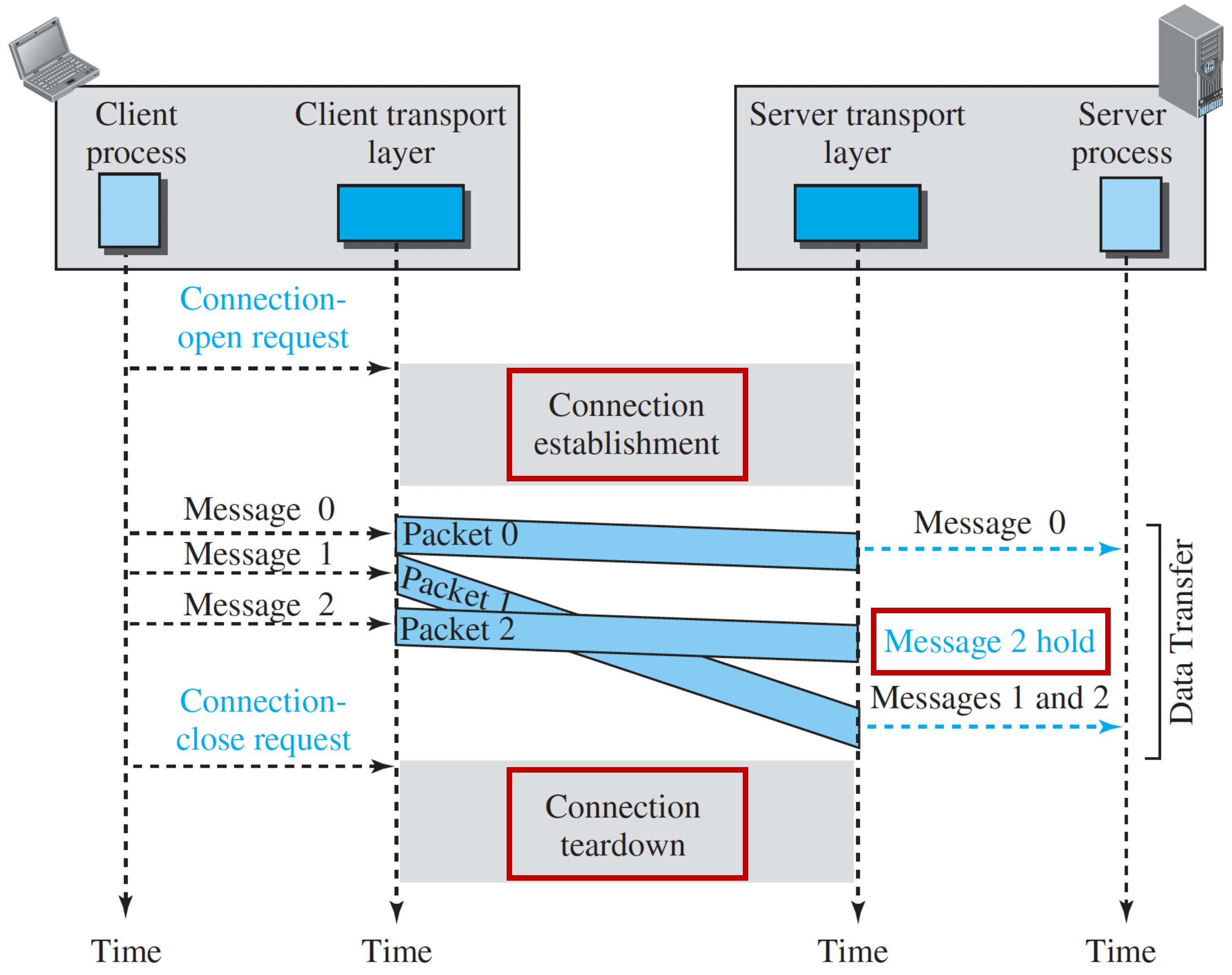
Connectionless Service



No flow control, error control, or congestion control.

Connection-Oriented Service

We can implement flow control, error control, and congestion control in a connection-oriented protocol.



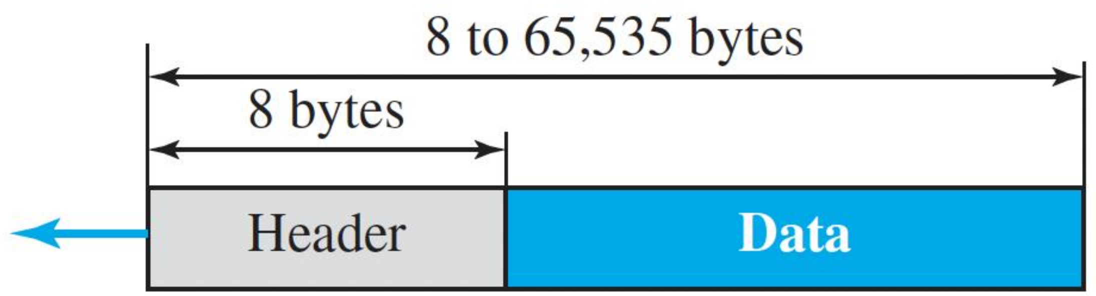
UDP (User Datagram Protocol)

- An **unreliable connectionless** transport-layer protocol.
- It provides simplicity and efficiency:
  - Minimum overhead
  - Good for sending small messages
  - Faster than TCP (Transmission Control Protocol)
- UDP packets are called **user datagrams**.
- The **protocol field** in the header of IP packet (network-layer packet) holds the value of **17** if the payload (data) belongs to **UDP**.
- If a process uses UDP, its data must be small enough to fit into one user datagram (no stream of data can be sent).

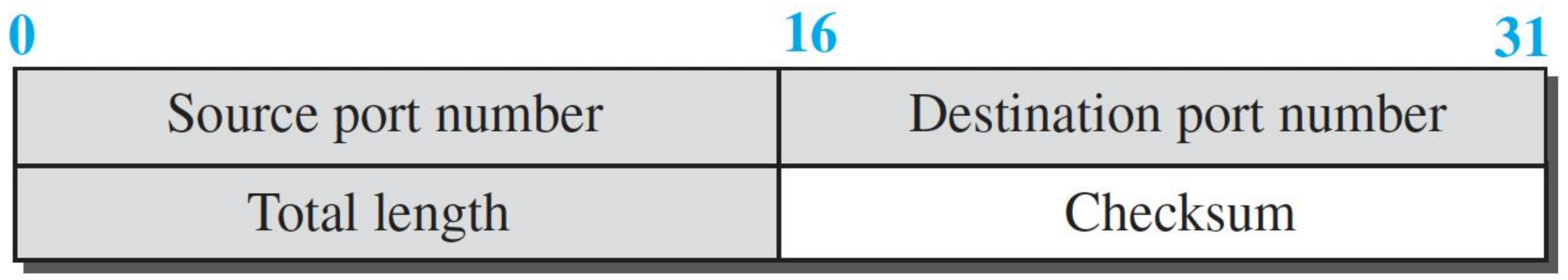
UDP

- **No flow control** → No window mechanism
- **No congestion control**
- **No error control** (except for the optional checksum field in the header)

UDP – Format of a User Datagram



a. UDP user datagram



b. Header format

header plus data

optional

UDP – A Connectionless Transport-Layer Protocol

- Each user datagram sent by UDP is an **independent** datagram.
- **No relationship** between the **different user datagrams** (even if they are coming from the same source process and going to the same destination program).
- The user datagrams are **not numbered** (**no sequence number**).
- No **connection establishment** and no **connection termination**.
  - Each user datagram can travel on a different path .

UDP – Applications

- UDP is suitable for a process that needs simple request-response communication (with little concern for flow and error control), e.g., DNS.
- Management processes
- Online Games like PUBG.
- Interactive real-time applications, e.g., Skype, Zoom, and Discord.
  - Zoom Encryption and used protocols: https://explore.zoom.us/docs/doc/Zoom%20Encryption%20Whitepaper.pdf
  - Some applications use both TCP and UDP protocols, e.g., Discord uses **UDP** for **voice and video calling** as packet loss is acceptable, while using **443 HTTPS over TCP** for text chats.
  - https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

Summary

- **Process-to-process** communication at transport layer
- **Port numbers** are needed to define processes
- **Flow**/**error**/**congestion control**
- **Connectionless** vs. **connection-oriented** service
- **UDP**
  - UDP packets are called **user datagram**
  - unreliable
  - connectionless
  - little overhead
  - fast delivery
  - no flow/error control
  - half-duplex

References

[1] Behrouz A.Forouzan, Data Communications & Networking with TCP/IP Protocol Suite, 6th Ed, 2022, McGraw-Hill companies.

Reading

- Chapter 9 of the textbook, sections 9.1 – 9.3.
- Chapter 9 of the textbook, section 9.7 (Practice Test)