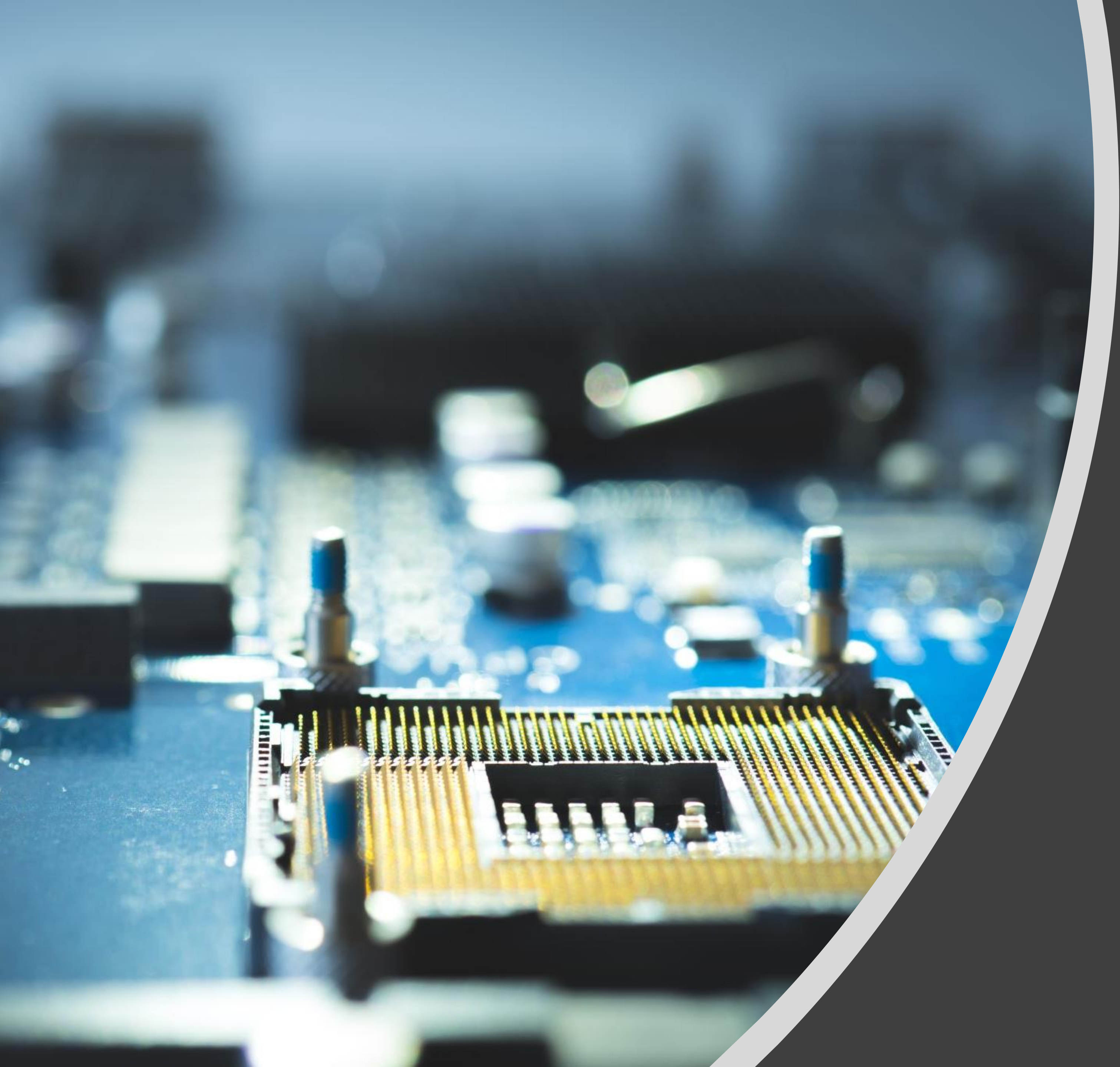


Computer Networks

Transport Layer

جامعة الانبار

قسم الهندسة الكهربائية

مادة شبكات الحاسوب – 2021

م.م. براء سعيد العبيدي

An overview of the Transport Layer

- A transport-layer protocol provides for **logical communication** between application processes running on different hosts. L*ogical communication* means that from an application’s perspective, it is as if the hosts running the processes were directly connected; in reality, the hosts may be on opposite sides of the planet, connected via numerous routers and a wide range of link types.
- Data units are called **segments**.
- On the sending side, the transport layer converts the application-layer messages it receives from a sending application process into transport-layer segments.
- This is done by breaking the application messages into smaller chunks and adding a transport-layer header to each chunk to create the transport-layer segment.
- The transport layer then passes the segment to the network layer at the sending end system, where the segment is encapsulated within a network-layer packet (a datagram) and sent to the destination.
- On the receiving side, the network layer extracts the transport-layer segment from the datagram and passes the segment up to the transport layer. The transport layer then processes the received segment, making the data in the segment available to the receiving application.

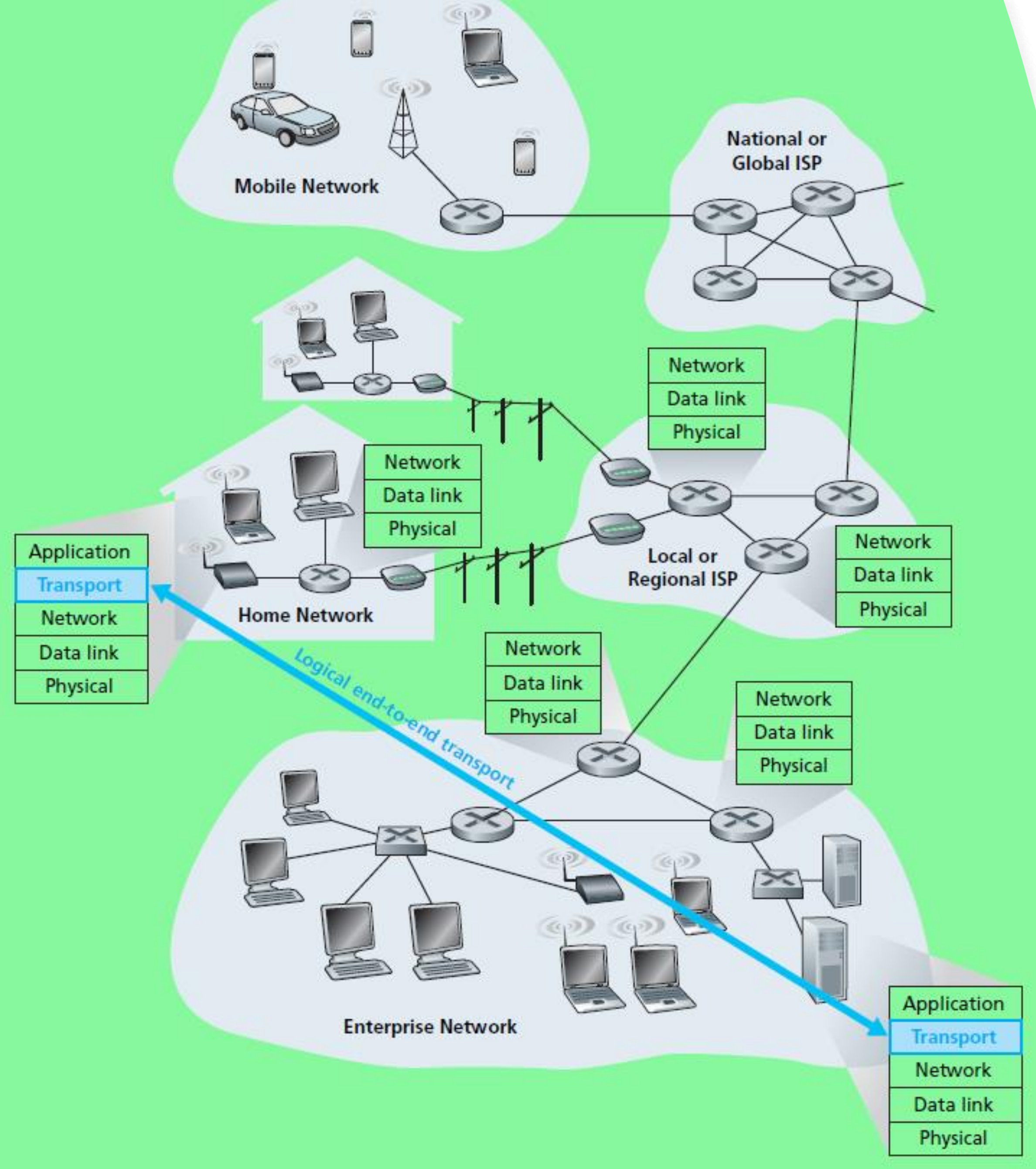


Figure 3.1 ♦ The transport layer provides logical rather than physical communication between application processes

Logical End-to-end Transport

- A transport-layer protocol provides logical communication between *processes* running on different hosts (computers, mobile devices, etc.).

Transport Layer Protocols

TCP (Transmission Control Protocol):

- provides a reliable, connection-oriented service to the invoking application.

UDP (User Datagram Protocol):

- provides an unreliable, connectionless service to the invoking application.

Multiplexing and Demultiplexing

- **Multiplexing :** The job of gathering data chunks at the source host from different sockets, encapsulating each data chunk with header information (that will later be used in demultiplexing) to create segments, and passing the segments to the network layer.
- **Demultiplexing :** At the destination host, the transport layer receives segments from the network layer just below. The transport layer has the responsibility of delivering the data in these segments to the appropriate application process running in the host.
- **How multiplexing and demultiplexing are done?**
- Each segment have special fields that indicate the socket to which the segment is to be delivered. These special fields are the **source port number field** and the **destination port number field**.
- Each port number is a 16-bit number, ranging from 0 to 65535.
- The port numbers ranging from 0 to 1023 are called **well-known port numbers** and are restricted, which means that they .are reserved for use by well-known application protocols such as HTTP.
- Each socket in the host is assigned a port number, and when a segment arrives at the host, the transport layer examines the destination port number in the segment and directs the segment to the corresponding socket. The segment’s data then passes through the socket into the attached process.

Segment Format

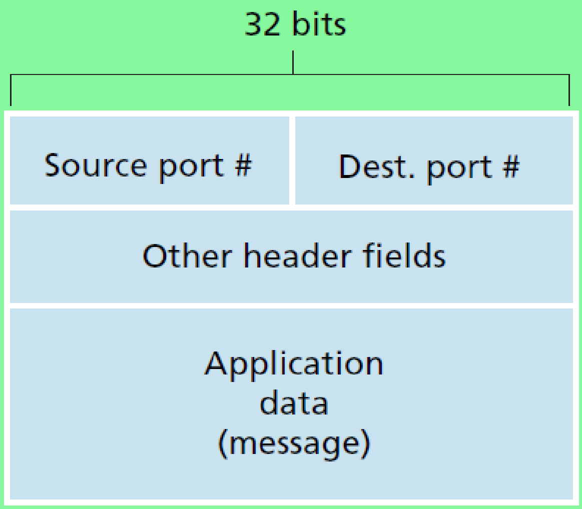
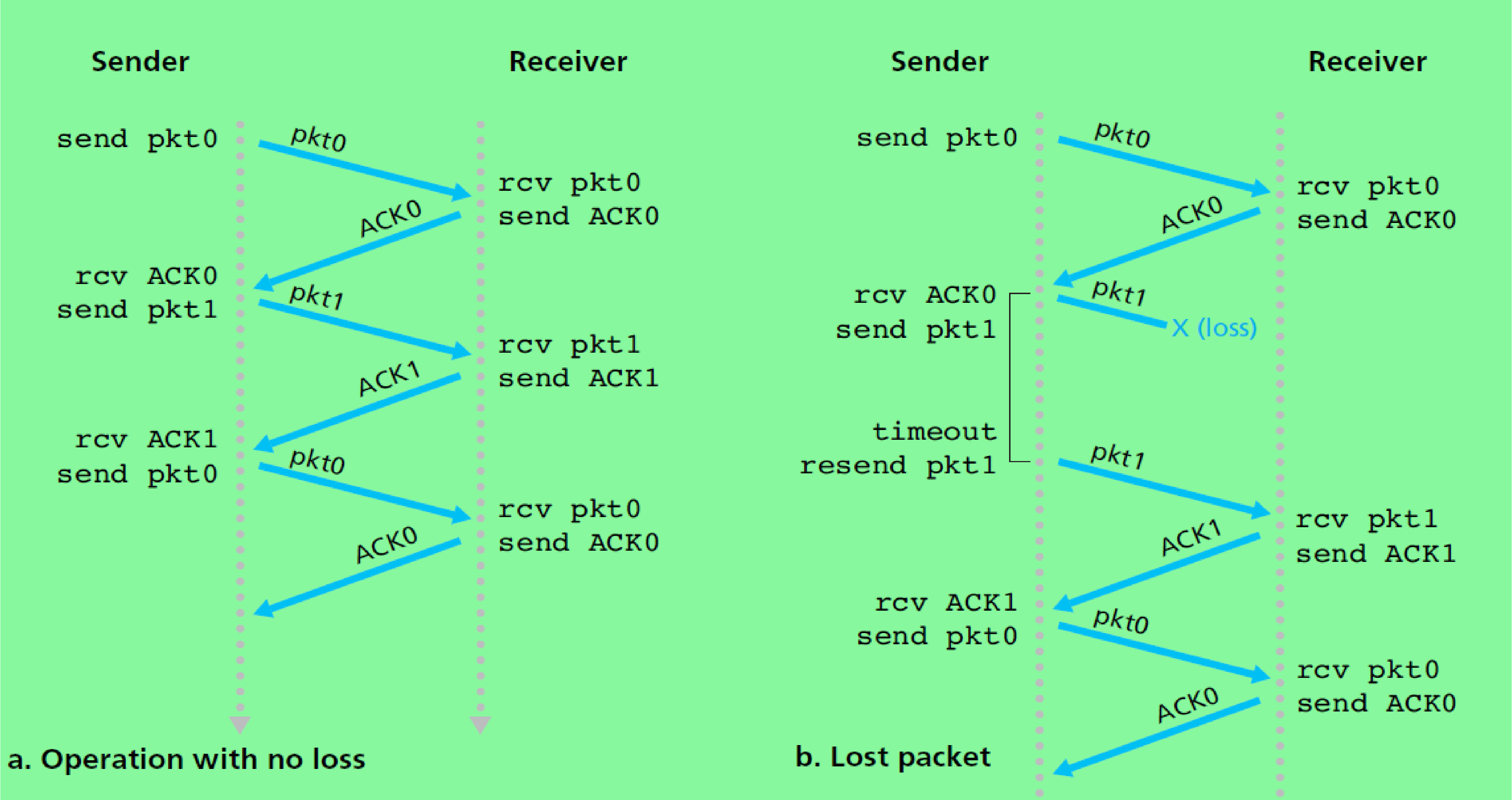


Figure 3.3 ♦ Source and destination port-number fields in a transport-layer segment

- Source port number and destination port number are used for multiplexing/demultiplexing.

Reliable Data Transfer (rdt)



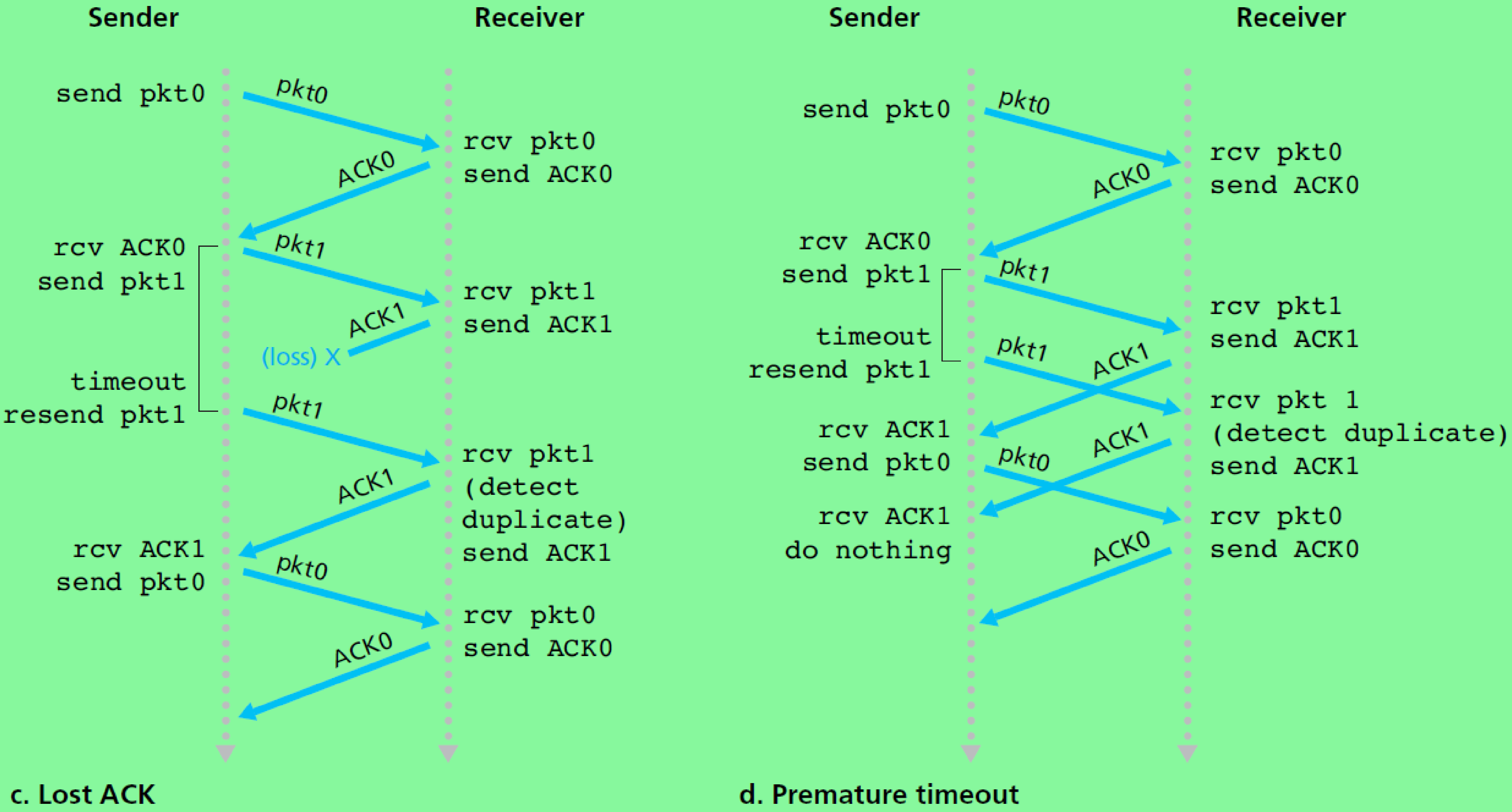
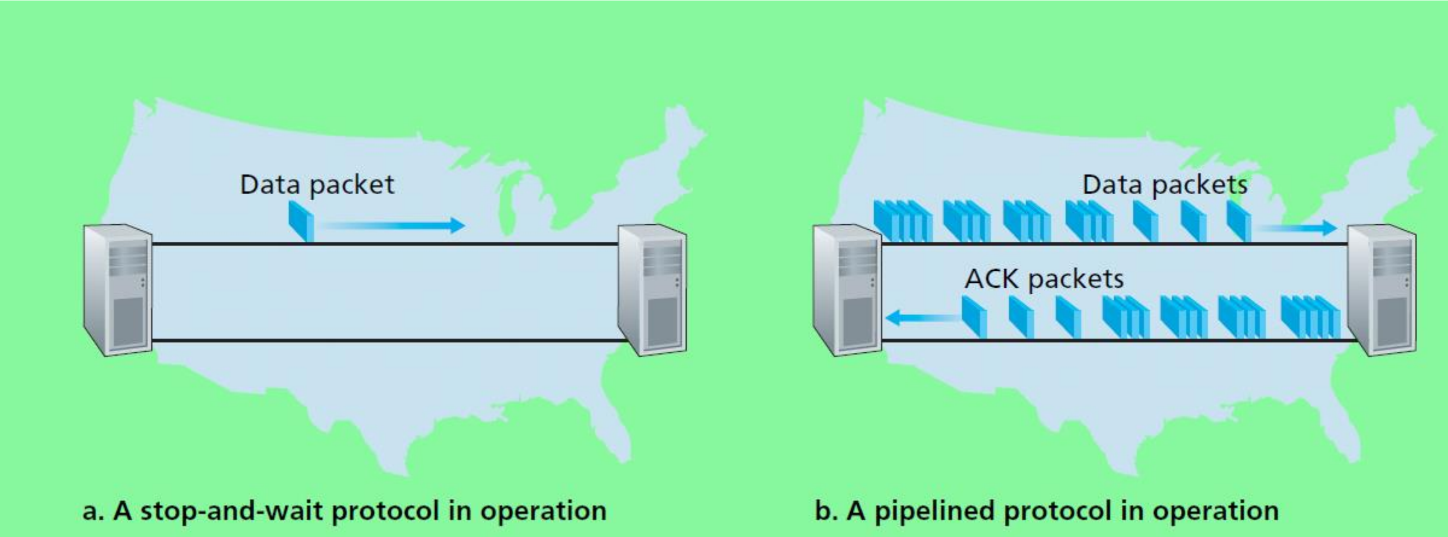


Figure 3.16 ♦ Operation of `rdt3.0`, the alternating-bit protocol

Stop-and Wait Vs. Pipeline Protocols

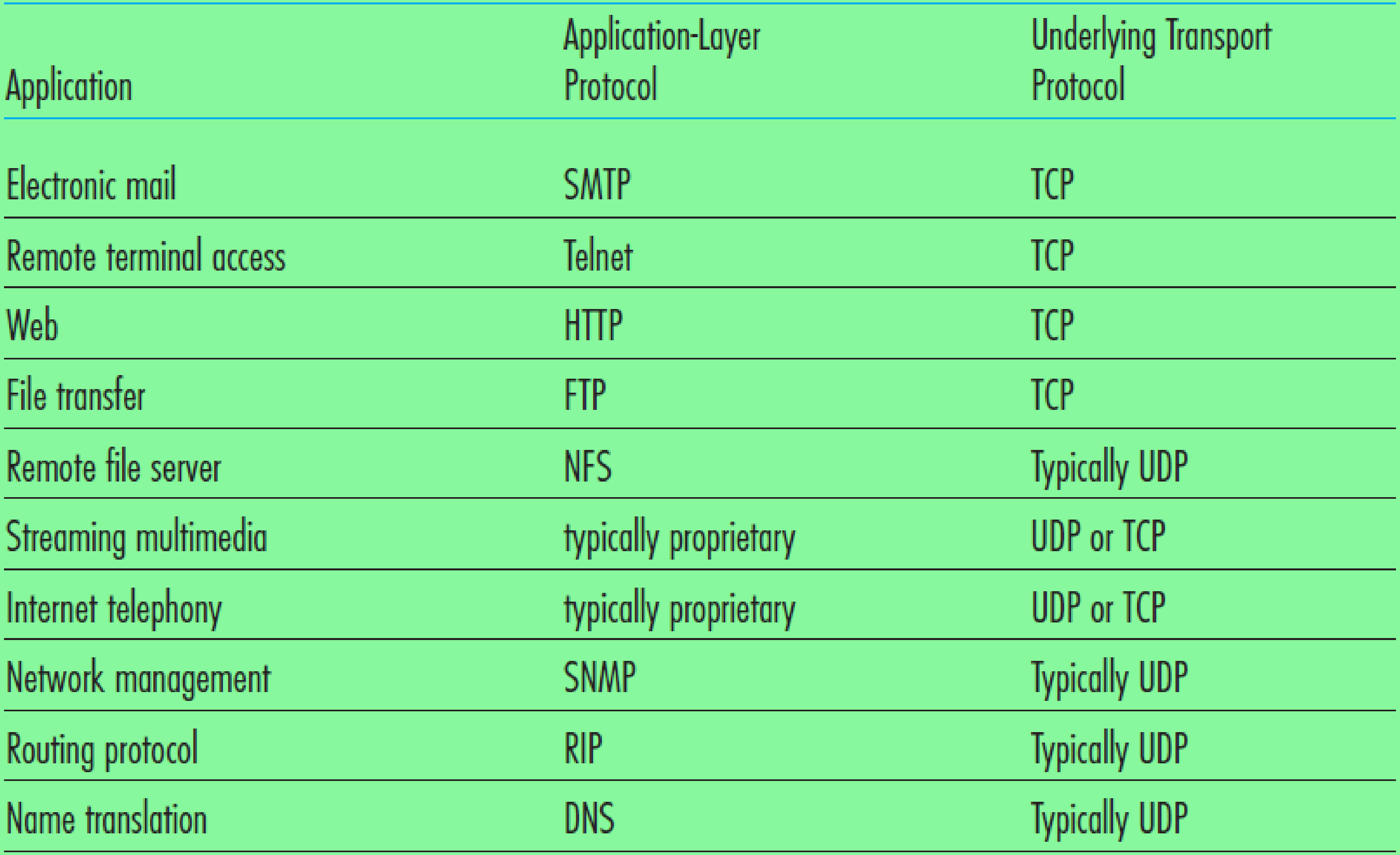
- Send one segment and wait until an ACK is received is called stop-and-wait.
- Sending a bundle of segments and wait for an ACK for all the sent segments is called pipeline.



a. A stop-and-wait protocol in operation

b. A pipelined protocol in operation

Figure 3.17 ♦ Stop-and-wait versus pipelined protocol



Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP	TCP
Remote terminal access	Telnet	TCP
Web	HTTP	TCP
File transfer	FTP	TCP
Remote file server	NFS	Typically UDP
Streaming multimedia	typically proprietary	UDP or TCP
Internet telephony	typically proprietary	UDP or TCP
Network management	SNMP	Typically UDP
Routing protocol	RIP	Typically UDP
Name translation	DNS	Typically UDP

Figure 3.6 ♦ Popular Internet applications and their underlying transport protocols