CO7219:
Internet and Cloud Computing

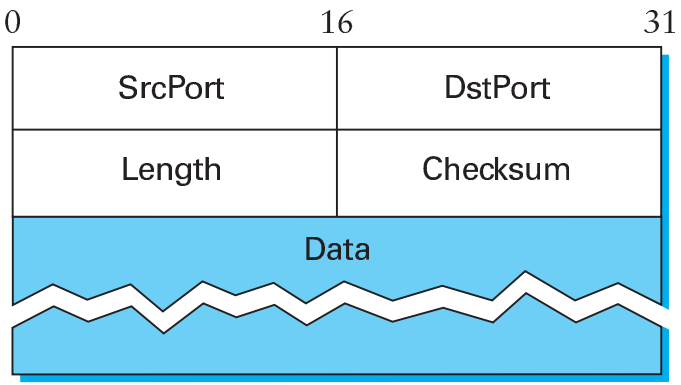
4. End-to-End Protocols

End-to-End Protocols

- The **transport** level of the network architecture turns the best-effort host-to-host packet delivery service into a process-to-process communication channel.
- Examples of desirable properties:
  - guarantees message delivery.
  - delivers messages in the same order they are sent.
  - delivers at most one copy of each message.
  - supports arbitrary-length messages.
  - supports synchronisation between sender and receiver.
  - allows receiver to apply flow control to the sender.
  - supports multiple application processes on each host.

4.1 Simple Demultiplexer (UDP)

- Simplest possible transport protocol: extends host-to-host delivery service into process-to-process communication service, adds no further functionality.
- The Internet's **User Datagram Protocol (UDP)** is an example of such a transport protocol.
- UDP uses the concept of **ports** to indirectly identify an application process on a host (cf. socket interface): A ⟨port,host⟩ pair is used as demultiplexing key.
- UDP header:

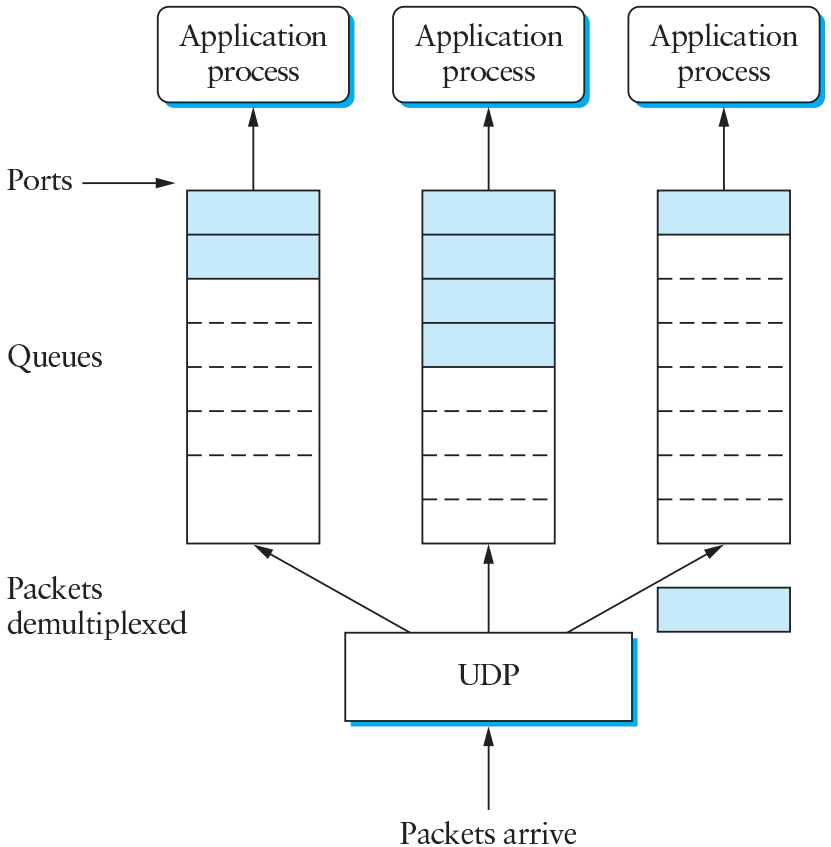


Port Numbers

- How does a process learn the port for the process to which it wants to send a message?
  - **Well-known port** approach: Specific services always use the same port (e.g. DNS: 53, mail: 25, talk: 517), cf. /etc/services.
    Sometimes the port is just the starting point for communication, to agree on a port to be used for subsequent communication.
  - **Port Mapper**: Client sends message to Port Mapper's well-known port asking for the port it should use to to talk to the service it requires.
- As a UDP packet contains the port number of the sending application, the recipient simply sends replies to that port.

Port Implementation

- Operating systems typically implement ports as message queues:



Remark: Multimedia applications often employ the real-time transport protocol (RTP) that runs on top of UDP.

4.2 Reliable Byte Stream (TCP)

- The Internet's **Transmission Control Protocol (TCP)** provides a reliable, connection-oriented, byte-stream service.
- TCP is the most widely used protocol of its type, and has been very carefully tuned.
- TCP guarantees reliable, in-order delivery of a stream of bytes.
- TCP is a full-duplex protocol, i.e., each TCP connection supports a pair of byte streams, one in each direction.
- TCP includes mechanisms for **flow control** (preventing the sender from overrunning the receiver) and **congestion control** (preventing the sender from overloading the network).

4.2.1 End-to-End Issues

TCP uses the sliding window algorithm, but over the Internet rather than a point-to-point link, so the following difficulties arise:

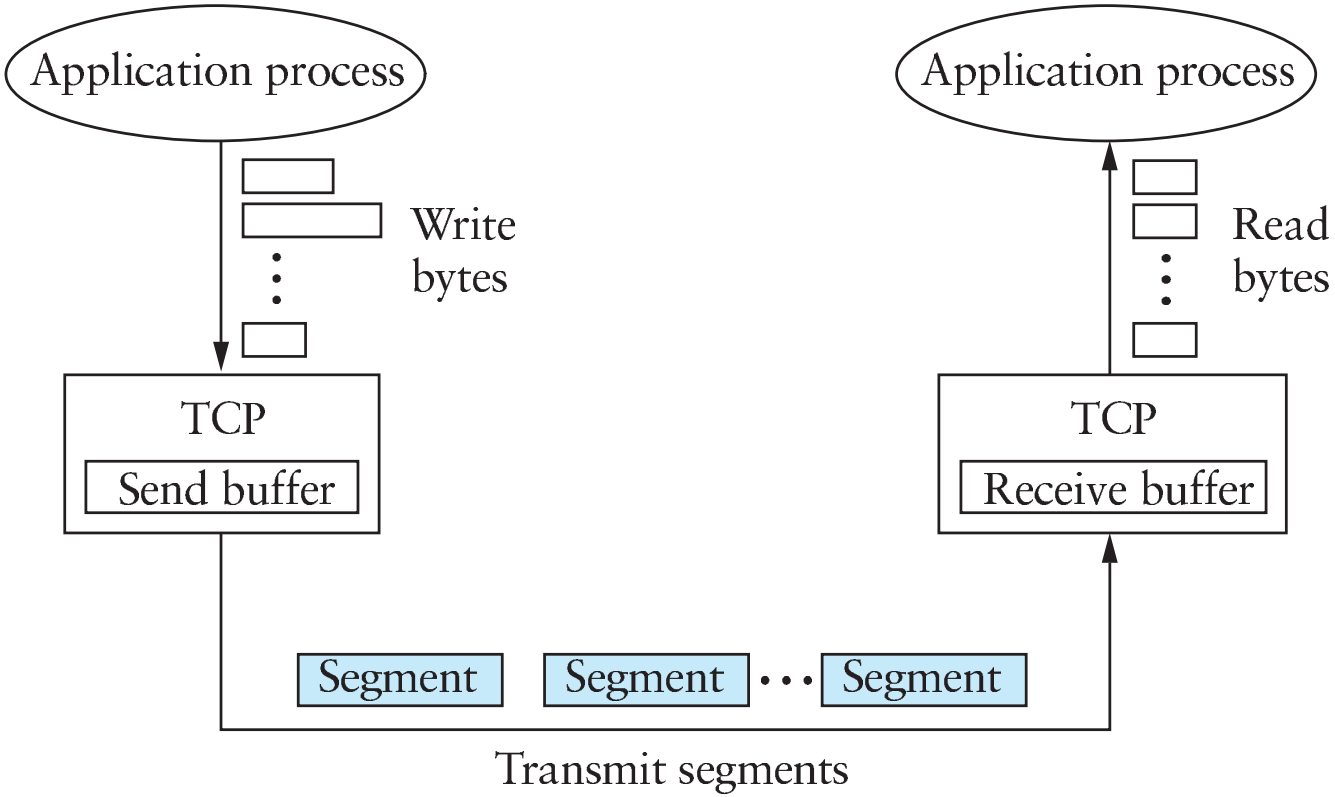
- TCP needs an explicit connection establishment phase during which the two sides agree to exchange data and establish some shared state for the sliding window algorithm to begin.
- The RTT can differ widely across different TCP connections and can vary dynamically ⇛ TCP needs an adaptive timeout mechanism.
- Packet reordering is possible ⇛ TCP assumes a maximum segment lifetime (MSL), typically 120 seconds.

End-to-End Issues (cont.)

- Any kind of computer can connect to the Internet, so the amount of resources dedicated to a TCP connection are highly variable ➠ TCP must include a flow control mechanism.
- The sender has no idea what links will be traversed to reach the destination, so it can potentially create network congestion ➠ TCP needs a congestion control mechanism.

➠ Running the sliding window algorithm over the Internet is much more complex than running it over a dedicated point-to-point link.

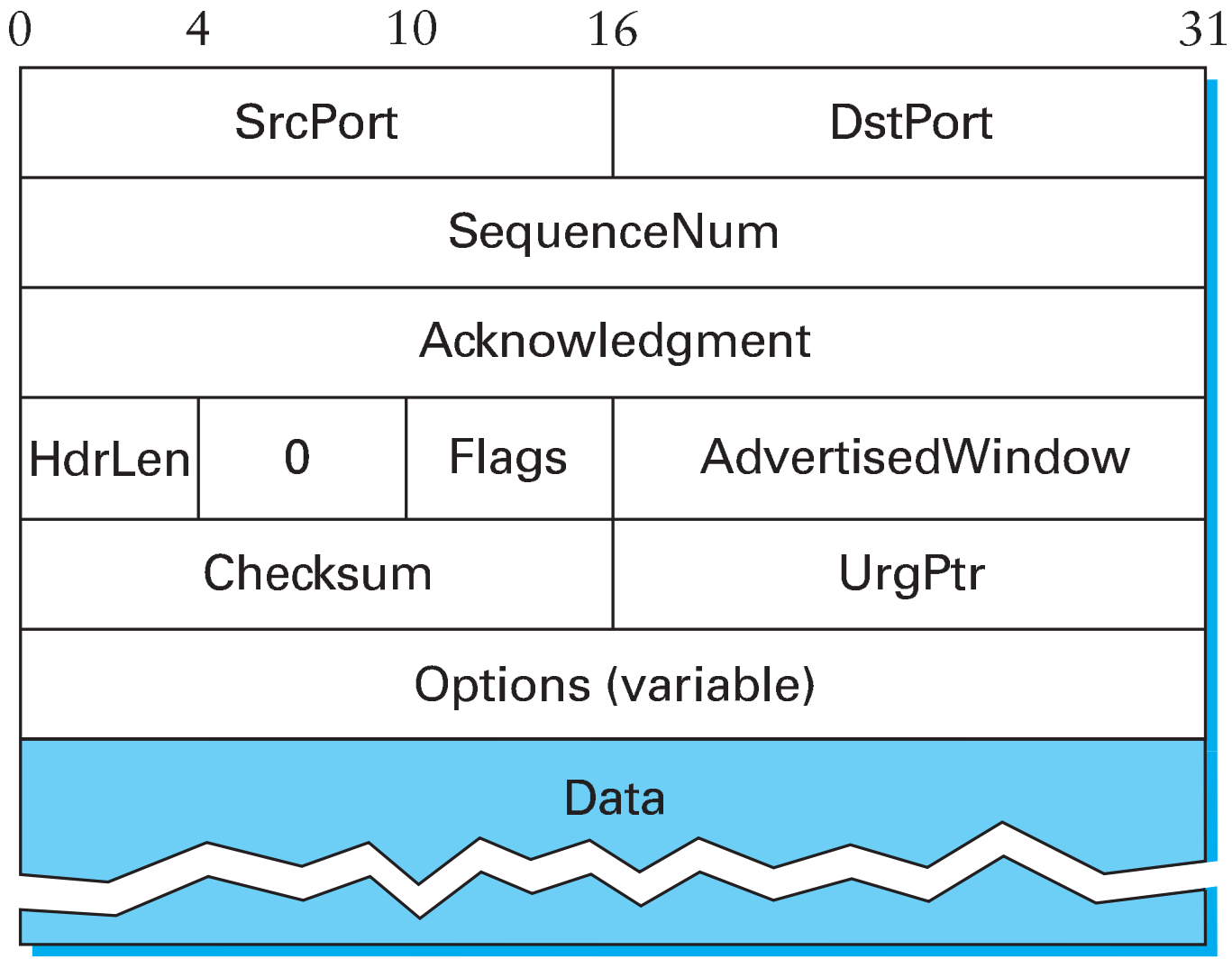
4.2.2 Segment Format



- TCP is byte-oriented, but transmits packets called **segments**.
- TCP's demux key is the 4-tuple:

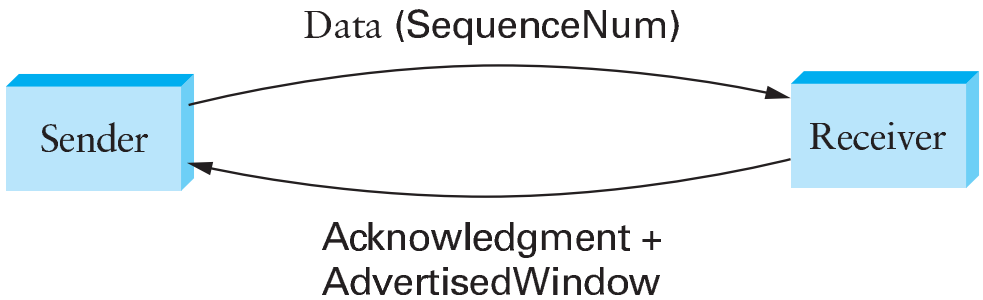
$$\langle SrcPort, SrcIPAddr, DstPort, DstIPAddr \rangle$$

TCP Header Format



Explanation of TCP Header Fields

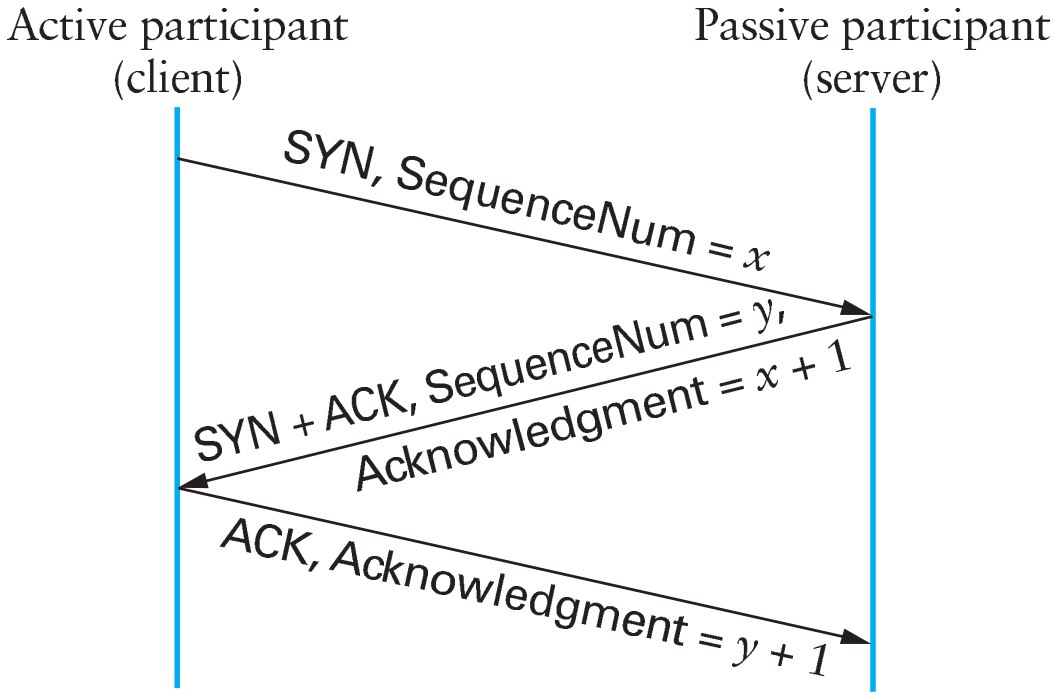
- For sliding window algorithm:



  - SequenceNum: seq. number of first **byte** in segment
  - Acknowledgment, AdvertisedWindow: information about the flow going in the other direction
- HdrLen: Length of header in 32-bit words
- Flags: SYN, FIN, RESET, PUSH, URG, ACK
- UrgPtr: indicates where the nonurgent data in the segment begins (urgent data present if URG flag is set).

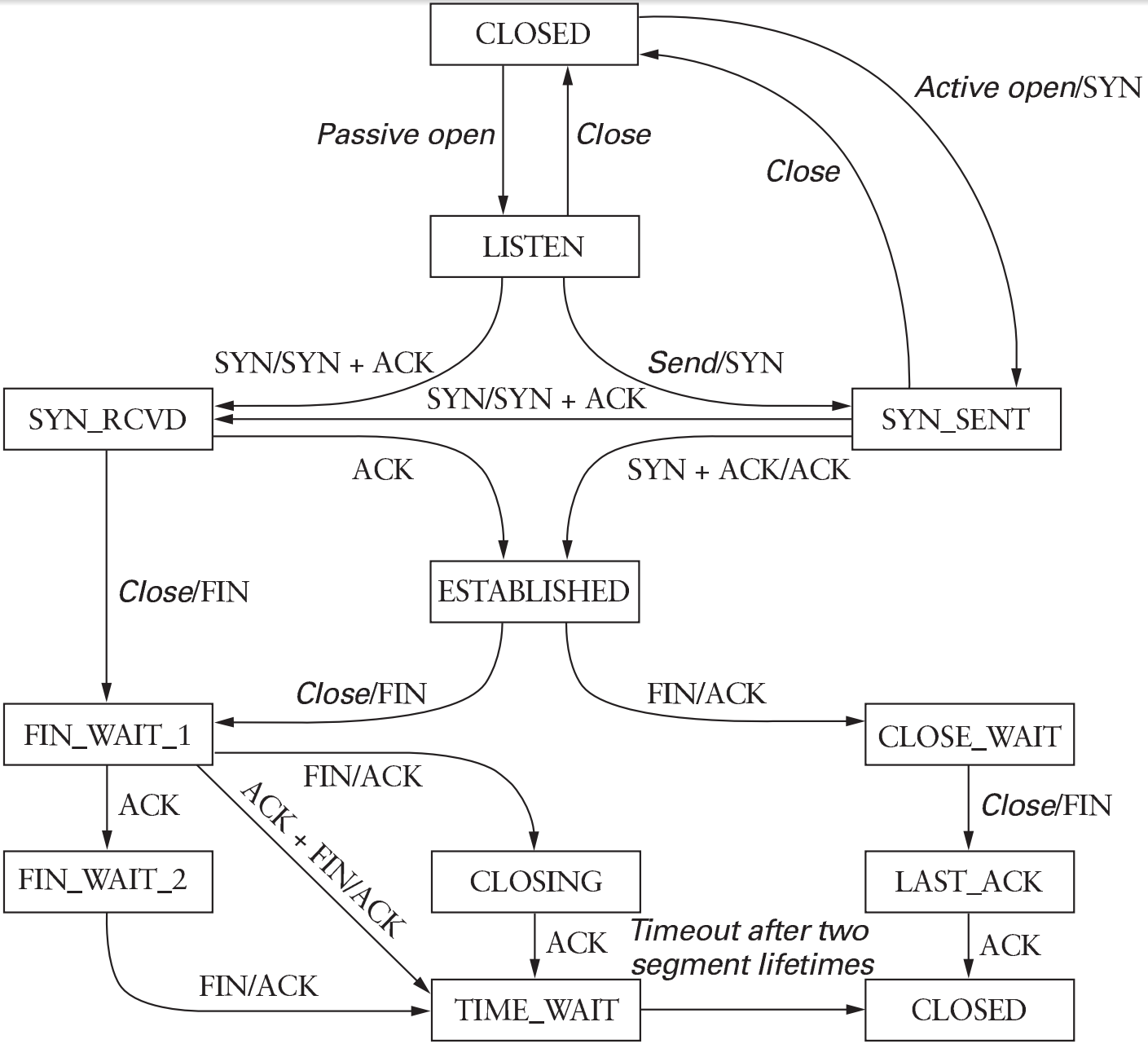
4.2.3 Connection Establishment and Termination

- Three-way handshake to establish connection:



  - Acknowledgment field represents "next sequence number expected," so value is $x + 1$ ($y + 1$).
  - x and y are chosen randomly (to avoid confusion with sequence numbers of a previous incarnation of the same TCP connection).

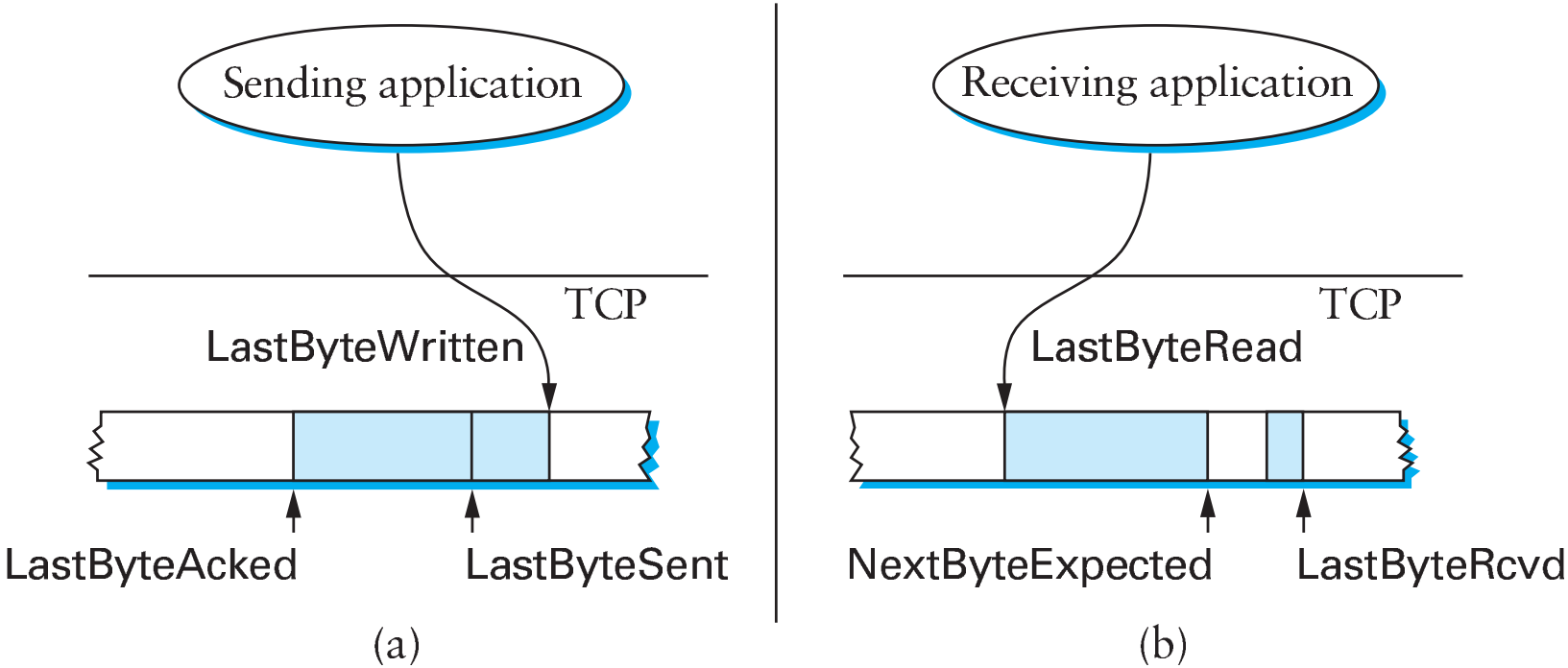
State Transition Diagram



4.2.4 Sliding Window Revisited

- TCP differs from the sliding window algorithm discussed earlier by adding a **flow control** mechanism.
- Rather than having a fixed-size sliding window, the receiver **advertises** a window size to the sender.
- The sender is limited to having at most AdvertisedWindow many bytes of unacknowledged data at any time.
- The receiver selects a suitable value of AdvertisedWindow based on the amount of memory allocated to the connection for buffering data.
- The value of AdvertisedWindow can change dynamically.
- The idea is to keep the sender from overrunning the receiver's buffer.

TCP Send Buffer and Receive Buffer



- Receive buffer is of size MaxRcvBuffer
- Receiver advertises AdvertisedWindow as:
  MaxRcvBuffer – ((NextByteExpected – 1) – LastByteRead)
- Sender ensures:
  LastByteSent – LastByteAcked $\leq$ AdvertisedWindow

Further Remarks

- The receiver can reduce the AdvertisedWindow all the way down to 0 and effectively stop the sender from sending data (but the sender will keep trying to send a 1-byte segment periodically if the AdvertisedWindow is 0).
- TCP uses **adaptive retransmission**: Timeouts are selected based on estimated RTT and RTT variance.
- TCP uses certain rules to trigger the transmission of segments (e.g. if both the window and the available data are larger than MSS, a segment of size MSS is sent, where MSS = maximum segment size).
- TCP extensions allows larger advertised windows by using a **scaling factor**.
  - STS-12 (622 Mbps) link with RTT 100 ms has delay $\times$ bandwidth product 7.4 MB, but the 16-bit field for AdvertisedWindow would limit window size to 64 KB.