Chapter 6

(Week 12)

The Transport Layer (CONTINUATION)

ANDREW S. TANENBAUM
COMPUTER NETWORKS
FOURTH EDITION
PP. 524-574

THE TRANSPORT LAYER’S TASK IS

TO PROVIDE RELIABLE, COST-EFFECTIVE DATA TRANSPORT

FROM SOURCE MACHINE TO DESTINATION MACHINE, INDEPENDENTLY OF THE PHYSICAL NETWORK OR NERWORKS CURRENTLY IN USE.

6.4. The Internet Transport Protocols: UDP

- The Internet has two main protocols in the transport layer, a connectionless protocol and a connection-oriented one.
- UDP is the connectionless protocol.
- TCP is the connection-oriented protocol.

6.4. The Internet Transport Protocols: UDP (User Datagram Protocol)

- UDP is basically just IP with a short header added
- UDP is a connectionless protocol that is mainly a wrapper for IP packets with the additional feature of multiplexing and demultiplexing multiple processes using a single IP address.
- UDP can be used for client-server interactions, for example, using RPC (Remote Procedure Call).
- UDP can also be used for building real-time protocols such as RTP (Real-Time Transport Protocol).

6.4. The Internet Transport Protocols: UDP

- Introduction to UDP
- Remote Procedure Call (RPC)
- The Real-Time Transport Protocol (RTP)

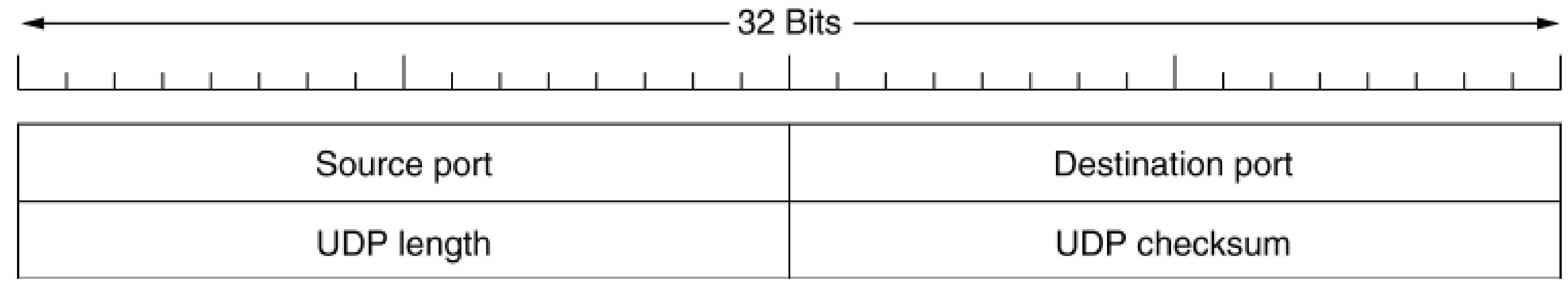
6.4.1. Introduction to UDP

- The Internet protocol suite supports a connectionless transport protocol, UDP (User Datagram Protocol).
- UDP provides a way for applications to send encapsulated IP datagrams and send them without having to establish a connection.

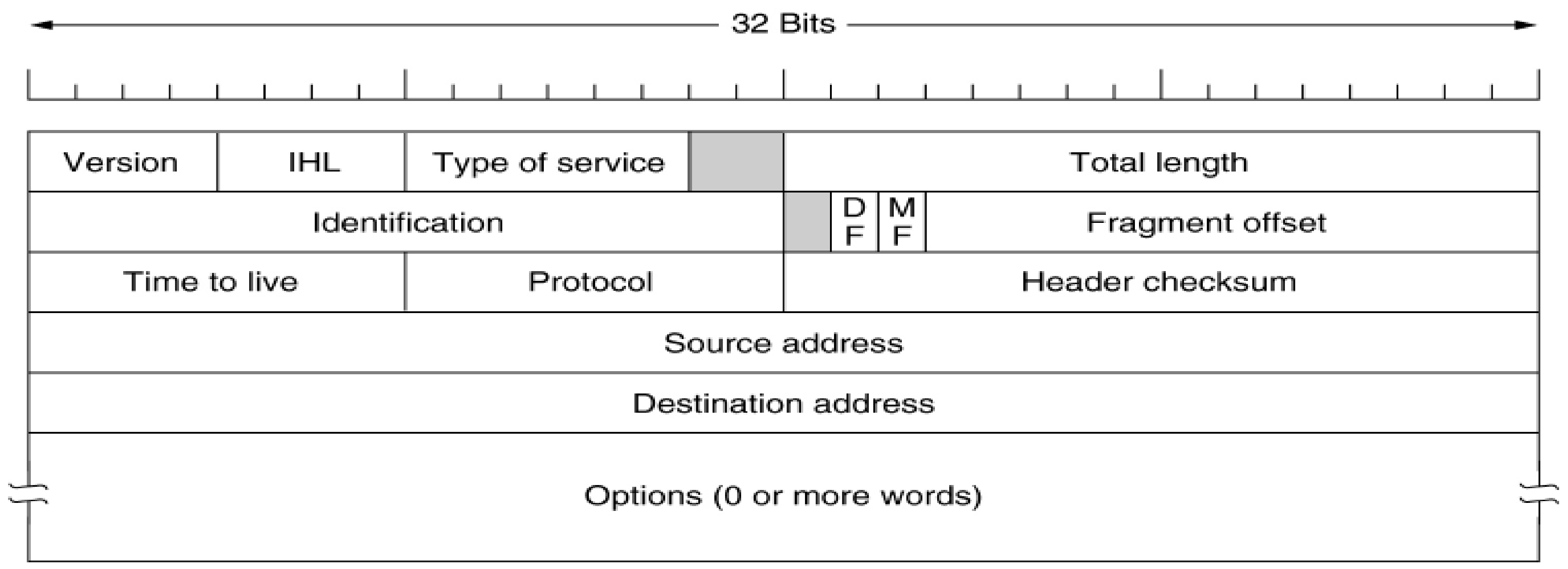
6.4.1. Introduction to UDP

- UDP transmits segments consisting of an 8-byte header followed by the payload.
- The two ports serve to identify the end points within the source and destination machines.
- When a UDP packet arrives, its payload is handed to the process attached to the destination port.

6.4.1. Introduction to UDP



The UDP header.



The IPv4 (Internet Protocol) header.

6.4.1. Introduction to UDP

- UDP does not do flow control, error control, or retransmission upon receipt of a bad segment.
- What UDP does do is provide an interface to the IP protocol with the added feature of demultiplexing multiple processes using the ports.

6.4.1. Introduction to UDP

- One area where UDP is especially useful is in client-server situations.
- Often, the client sends a short request to the server and expects a short reply back.
- If either the request or reply is lost, the client can just time out and try again.
- An application that uses UDP this way is DNS (Domain Name System)

6.4.1. Introduction to UDP

- In brief, a program that needs to look up the IP address of some host name, for example, www.cs.berkeley.edu, can send a UDP packet containing the host name to a DNS server.
- The server replies with a UDP packet containing the host's IP address.
- No setup is needed in advance and no release is needed afterward.
- Just two messages go over the network.

6.4.2. Remote Procedure Call

- In a certain sense, sending a message to a remote host and getting a reply back is a lot like making a function call in programming language.
- In both cases you start with one or more parameters and you get back a result.

6.4.2. Remote Procedure Call

- This observation has led people to try to arrange request-reply interaction on networks to be cast in the form of procedure calls.
- Such an arrangement makes network applications much easier to program and more familiar to deal with.

6.4.2. Remote Procedure Call

- For example, just imagine a procedure named *get_IP_address(host_name)* that works by sending a UDP packet to a DNS server and waiting for the reply, timing out and trying again if one is not forthcoming quickly enough.
- In this way, all the details of networking can be hidden from the programmer.

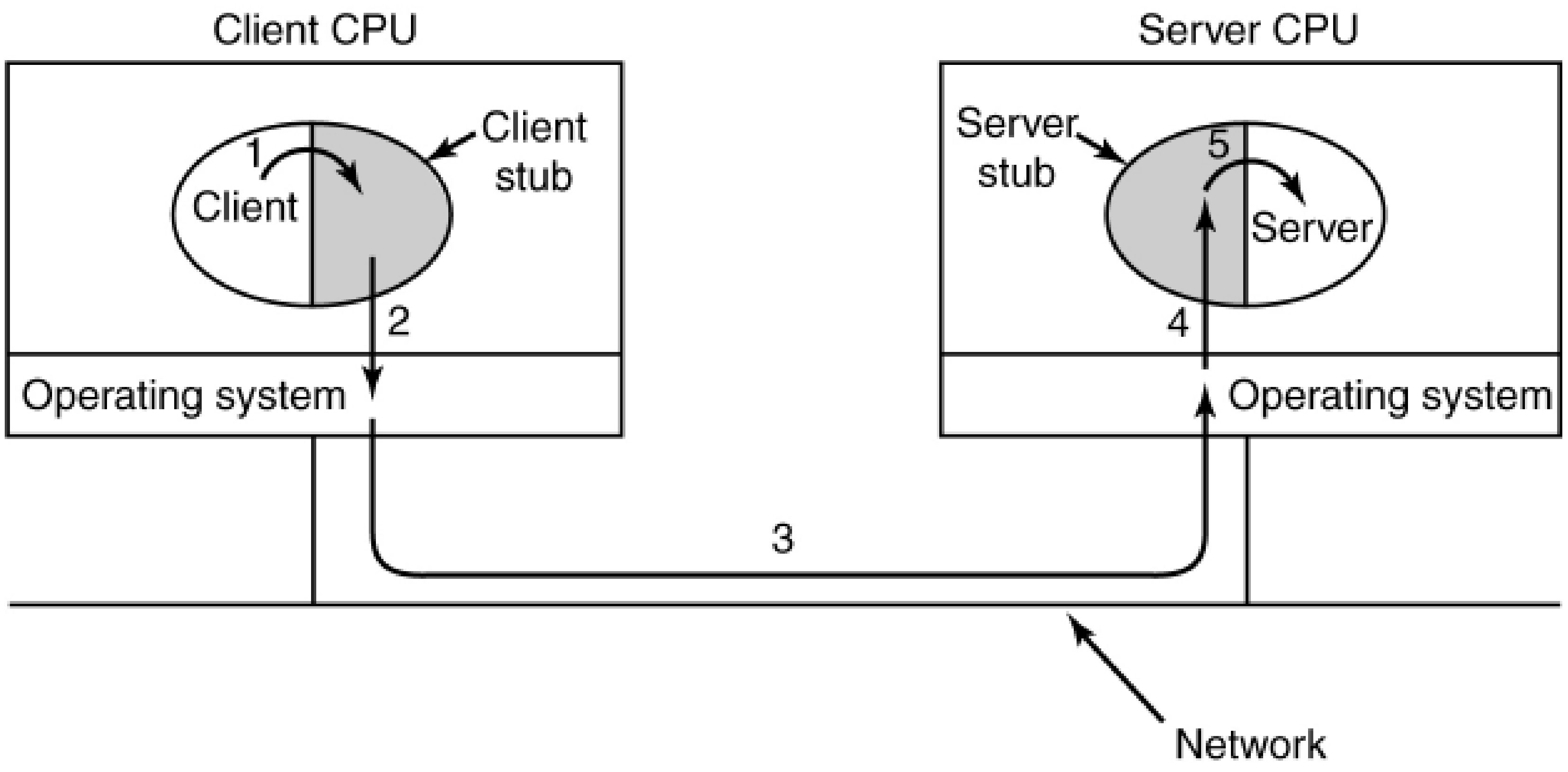
6.4.2. Remote Procedure Call

- Birrell and Nelson: programs can call procedures located on remote hosts.
- Information can be transported from the caller to the called in the parameters and can come back in the procedure result. No message passing is visible to the programmer.

6.4.2. Remote Procedure Call

- This technique is known as RPC (Remote Procedure Call) and has become the basis for many networking applications.
- Traditionally, the calling procedure is known as the client and called procedure is known as the server.

6.4.2. Remote Procedure Call



Steps in making a remote procedure call. The stubs are shaded.

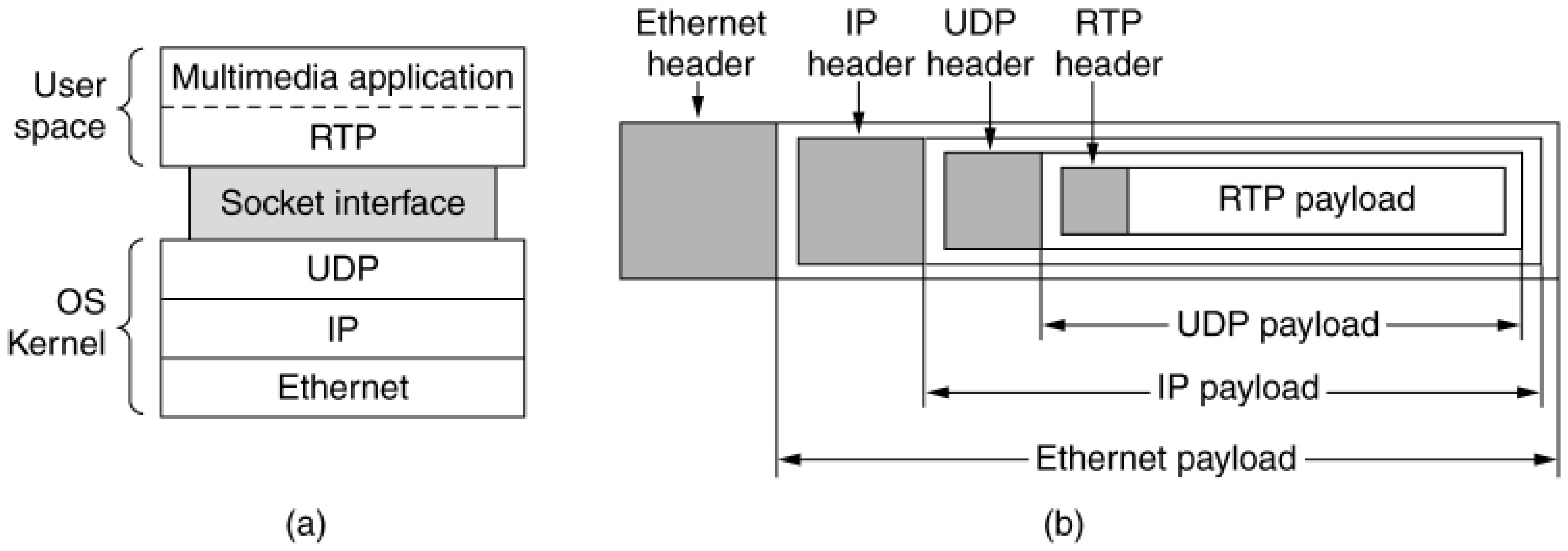
6.4.3. The Real-Time Transport Protocol

- Client-server RPC is one area in which UDP is widely used.
- Another one is real-time multimedia application.
- In particular, as Internet radio, Internet telephony, music-on-demand, videoconferencing, video-on-demand,…

6.4.3. The Real-Time Transport Protocol

- People discovered that each application was reinventing more or less the same real-time transport protocol.
- It gradually became clear that having a generic real-time transport protocol for multiple application would be a good idea.
- Thus was RTP (Real-Time Transport Protocol) born.

6.4.3. The Real-Time Transport Protocol



(a) The position of RTP in the protocol stack. (b) Packet nesting.

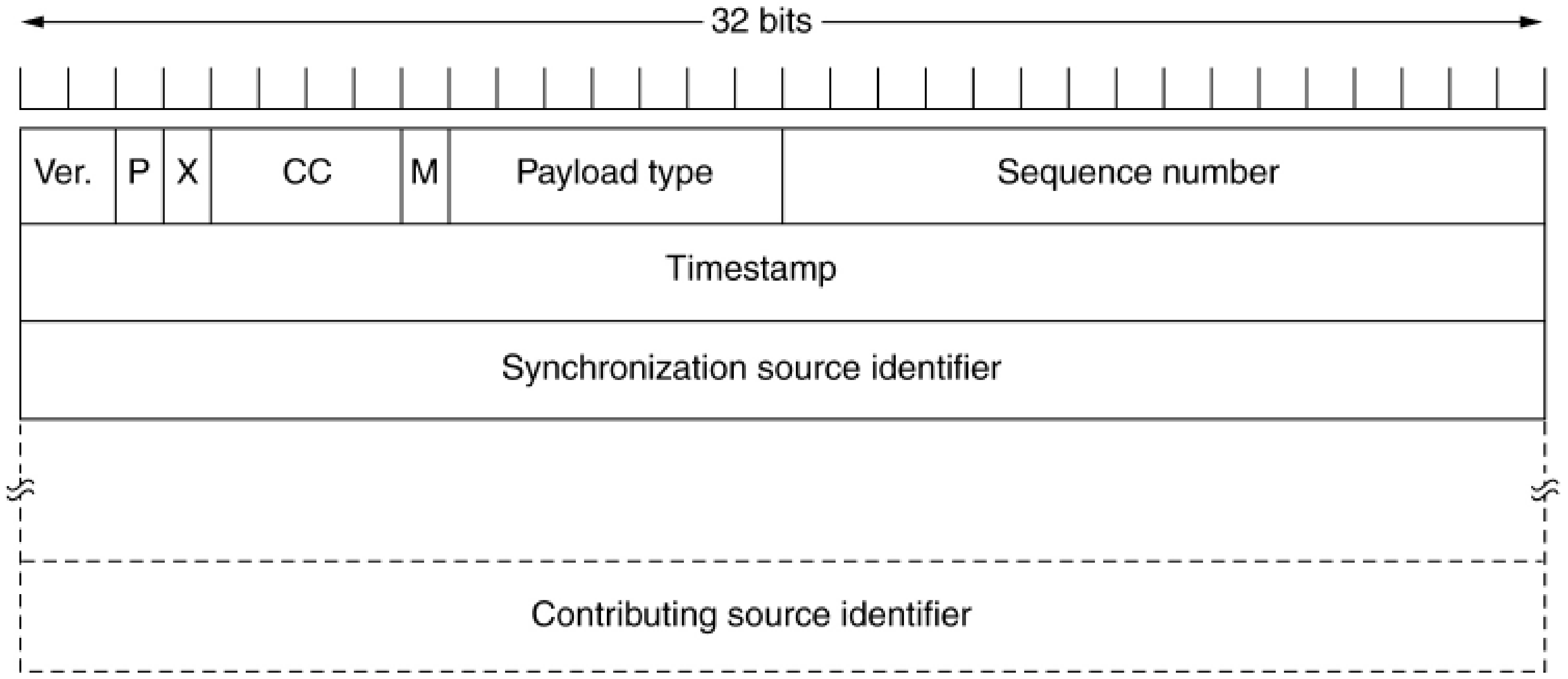
6.4.3. The Real-Time Transport Protocol

- The basic function of RTP is to multiplex several real-time data streams onto a single stream of UDP packets.
- The UDP stream can be sent to a single destination (unicasting) or to multiple destinations (multicasting).

6.4.3. The Real-Time Transport Protocol

- Each packet sent in an RTP stream is given a number one higher than its predecessor.
- RTP has no flow control, no error control, no acknowledgements, and no mechanism to request retransmission.

6.4.3. The Real-Time Transport Protocol


32 bits
Ver.
P
X
CC
M
Payload type
Sequence number
Timestamp
Synchronization source identifier
Contributing source identifier

The RTP header.

6.4.3. The Real-Time Transport Protocol

- RTCP (Real-Time Transport Control Protocol)
- It handles feedback, synchronization, and the user interface but does not transport any data.
- The first function can be used to provide feedback on delay, jitter, bandwidth, congestion, and other network properties to the sources.

6.4.3. The Real-Time Transport Protocol

- RTCP also handles interstream synchronization. The problem is that different streams may use different clocks, with different granularities and different drift rates. RTCP can be used to keep them in sync.

6.4.3. The Real-Time Transport Protocol

- RTCP provides a way for naming the various sources (e.g., in ASCII text). This information can be displayed on the receiver’s screen to indicate who is talking at the moment.

6.5. The Internet Transport Protocols: TCP

- UDP is a simple protocol and it has some niche uses, such as client-server interactions and multimedia.
- But for most Internet applications, reliable, sequenced delivery is needed.
- UDP cannot provide this, so another protocol is required.
- It is called TCP and is the main workhorse of the Internet.

6.5. The Internet Transport Protocols: TCP

- The main Internet transport protocol is TCP.
- It provides a reliable bidirectional byte stream.
- It uses a 20-byte header on all segments.
- Segments can be fragmented by routers within the Internet, so hosts must be prepared to do reassembly.

6.5. The Internet Transport Protocols: TCP

- A great deal of work has gone into optimizing TCP performance, using algorithms from Nagle, Clark, Jacobson, Karn, and others.
- Wireless links add a variety of complications to TCP.
- Transactional TCP is an extension to TCP that handles client-server interactions with a reduced number of packets.

6.5. The Internet Transport Protocols: TCP

- Introduction to TCP
- The TCP Service Model
- The TCP Protocol
- The TCP Segment Header
- TCP Connection Establishment
- TCP Connection Release
- TCP Connection Management Modeling
- TCP Transmission Policy
- TCP Congestion Control
- TCP Timer Management
- Wireless TCP and UDP
- Transactional TCP

6.5.1. Introduction to TCP

- TCP - Transmission Control Protocol
- TCP was specifically designed to provide a reliable end-to-end byte stream over an unreliable internetwork.

6.5.1. Introduction to TCP

- An Internetwork differs from a single network because different parts may have widely different topologies, bandwidths, delays, packet sizes, and other parameters.
- TCP was designed to dynamically adapt to properties of the internetwork and to be robust in the face of many kinds of failures.

6.5.1. Introduction to TCP

- Each machine supporting TCP has a TCP transport entity.
- A TCP transport entity is a library procedure (a user process) or part of the kernel.
- In both cases, it manages TCP streams and interfaces to the IP layer.

6.5.1. Introduction to TCP

- A TCP transport entity accepts user data streams from local processes, breaks them up into pieces not exceeding 64 KB (in practice, often 1460 data bytes in order to fit in a single Ethernet frame with the IP and TCP headers), and sends each piece as a separate IP datagram.

6.5.1 Introduction to TCP

- When datagrams containing TCP data arrive at a machine, they are given to the TCP entity, which reconstructs the original byte streams.
- "TCP" or "TCP transport entity" is a piece of software.
- TCP protocol is a set of rules.

6.5.1. Introduction to TCP

- IP layer gives no guarantee that datagrams will be delivered properly, so it up to TCP to time out and retransmit them as need be.
- Daragrams that do arrive may well do so in wrong order; it is also up to TCP to reassemble them into messages in proper sequence.
- In short, TCP must furnish the reliability that most users want and that IP does not provide.

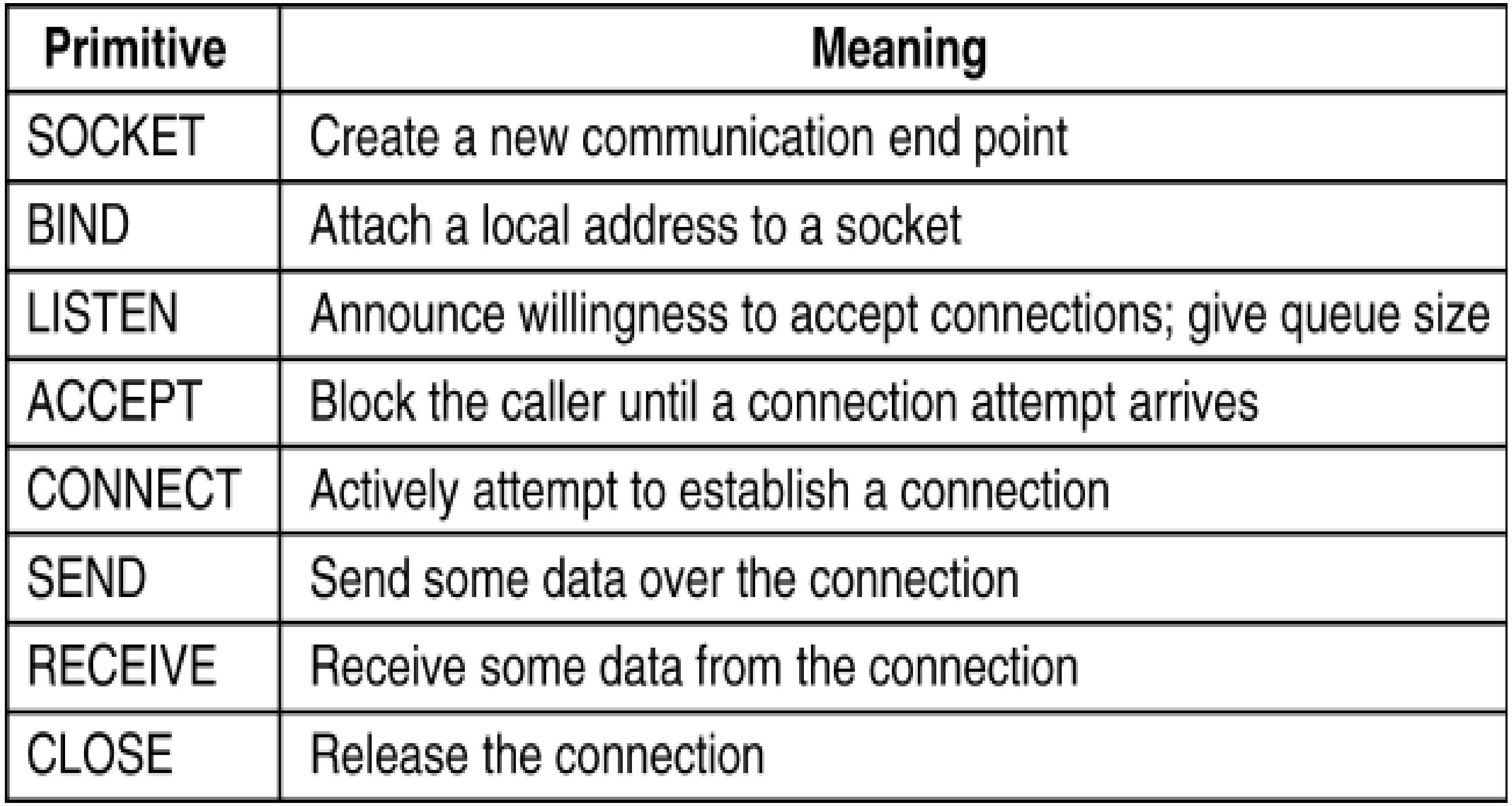
6.5.2. The TCP Service Model

- TCP service is obtained by both the sender and receiver creating end points, called sockets.
- Each socket has a socket number (address) consisting of the IP address of the host and a 16-bit number local to that host, called port.
- A port is the TCP name for a TSAP.

6.5.2. The TCP Service Model

- For TCP service to be obtained, a connection must be explicitly established between a socket on the sending machine and a socket on the receiving machine.
- A socket may be used for multiple connections at the same time.
- In other words, two or more connections may terminate at the same socket
- socket1, socket2, … at both ends

Berkeley Sockets



Primitive	Meaning
SOCKET	Create a new communication end point
BIND	Attach a local address to a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Block the caller until a connection attempt arrives
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

The socket calls for TCP.

The TCP Service Model

Some assigned ports.

Port	Protocol	Use
21	FTP	File transfer
23	Telnet	Remote login
25	SMTP	E-mail
69	TFTP	Trivial File Transfer Protocol
79	Finger	Lookup info about a user
80	HTTP	World Wide Web
110	POP-3	Remote e-mail access
119	NNTP	USENET news

Port numbers below 1024 are called well-known ports and are reserved for standard services.

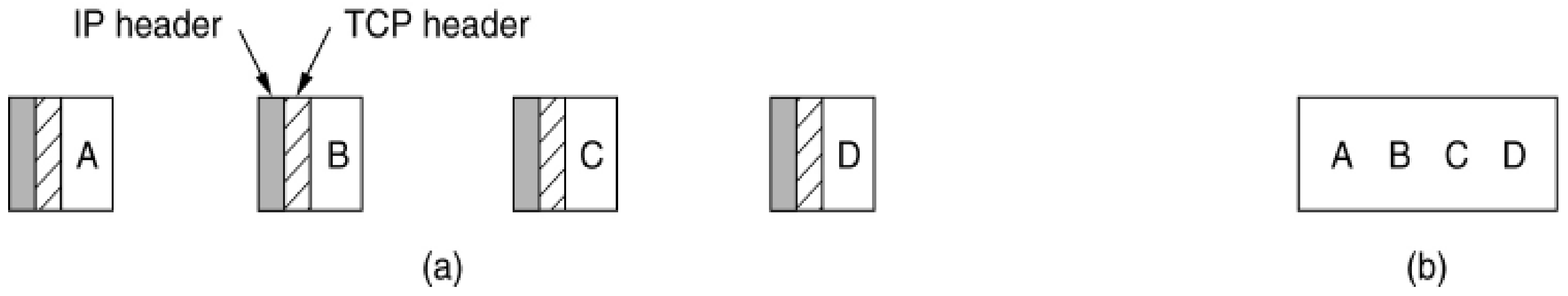
6.5.2. The TCP Service Model

- All TCP connections are full duplex and point-to-point.
- Full duplex means that traffic can go in both directions at the same time.
- Point-to-Point means that each connection has exactly two end points.
- TCP does not support multicasting or broadcasting.

6.5.2. The TCP Service Model

- A TCP connection is a byte stream, not a message stream.
- Message boundaries are not preserved end to end.
- For example, if the sending process does four 512-byte writes to a TCP stream, these data may be delivered to the receiving process as four 512-byte chunks, two 1024-byte chunks, one 2048-byte chunk, or some other way.

The TCP Service Model



(a) Four 512-byte segments sent as separate IP datagrams.

(b) The 2048 bytes of data delivered to the application in a single READ call.

6.5.2. The TCP Service Model

- urgent data:
- This feature of the TCP service causes TCP to stop accumulating data and transmit everything it has for that connection immediately.

6.5.3. The TCP Protocol

- A key feature of TCP is that every byte on a TCP connection has its own 32-bit sequence number.
- Separate 32-bit sequence numbers are used for acknowledgements and for window mechanism.

6.5.3. The TCP Protocol

- The sending and receiving TCP entities exchange data in the form of segments.
- A TCP SEGMENT consists of a fixed 20-byte header (plus an optional part) followed by zero or more data bytes.
- Two limits restrict the segment size.
- First, each segment, including the TCP header, must fit in the 65,515-byte IP payload.

6.5.3. The TCP Protocol

- Second, each network has a maximum transfer unit, or MTU, and each segment must fit in the MTU.
- In practice, the MTU is generally 1500 bytes (the Ethernet payload size) and thus defines the upper bound on segment size.
- The basic protocol used by TCP entities is the sliding window protocol.

6.5.4. The TCP Segment Header

- Every segment begins with a fixed-format, 20-byte header.
- The fixed header may be followed by header options.

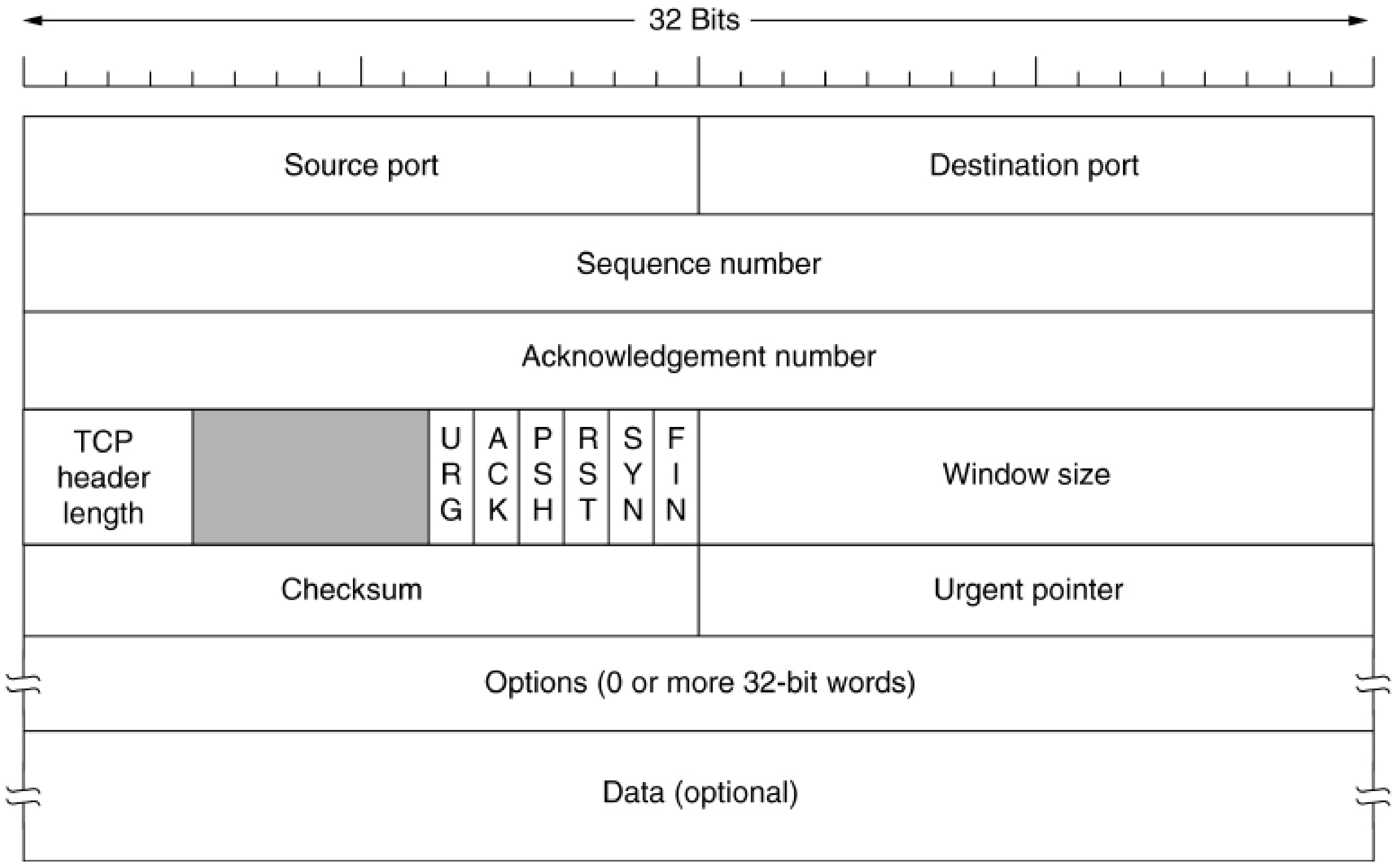
6.5.4. The TCP Segment Header

- After the options, if any, up to 65,535-20-20=65,495 data bytes may follow, where the first 20 refer to the IP header and the second to the TCP header.
- Segments without any data are legal and are commonly used for acknowledgements and control messages.

6.5.4. The TCP Segment Header

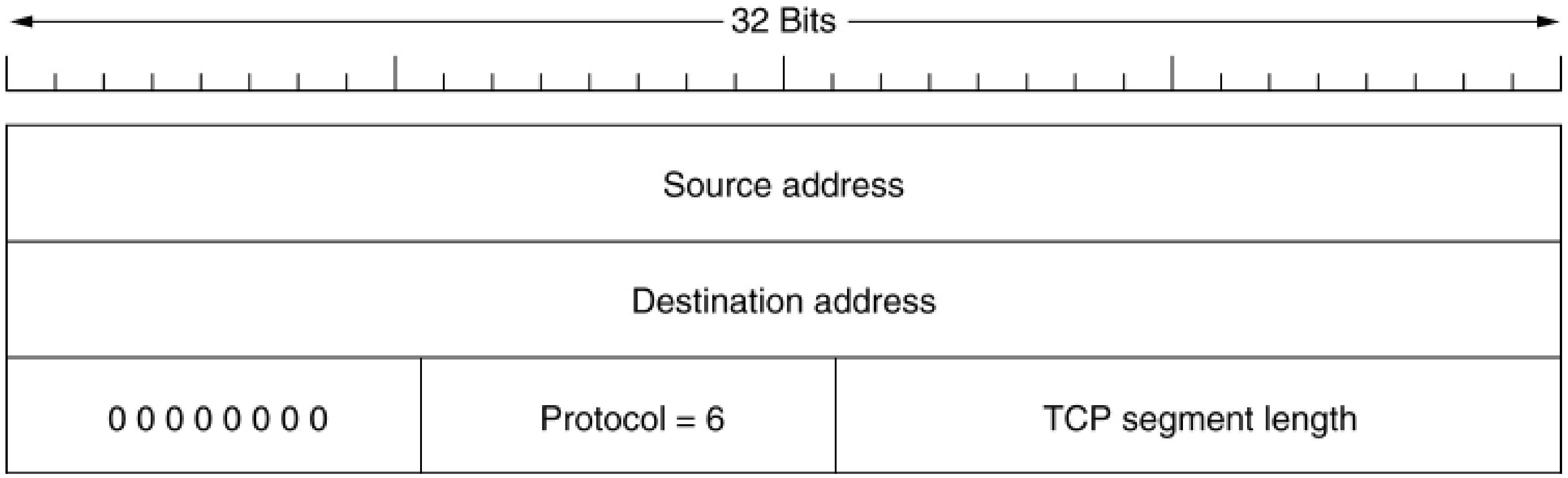
- Source port and destination port fields identify the local end points of the connection.
- The source and destination end points together identify the connection.

6.5.4.The TCP Segment Header



TCP Header.

The TCP Segment Header (2)



The pseudoheader included in the TCP checksum.

6.5.5. TCP Connection Establishment

- Connections are established in TCP by means of the three-way handshake.
- To establish a connection:
- The server passively waits for an incoming connection by executing the LISTEN AND ACCEPT primitives, either specifying a specific source or nobody in particular.

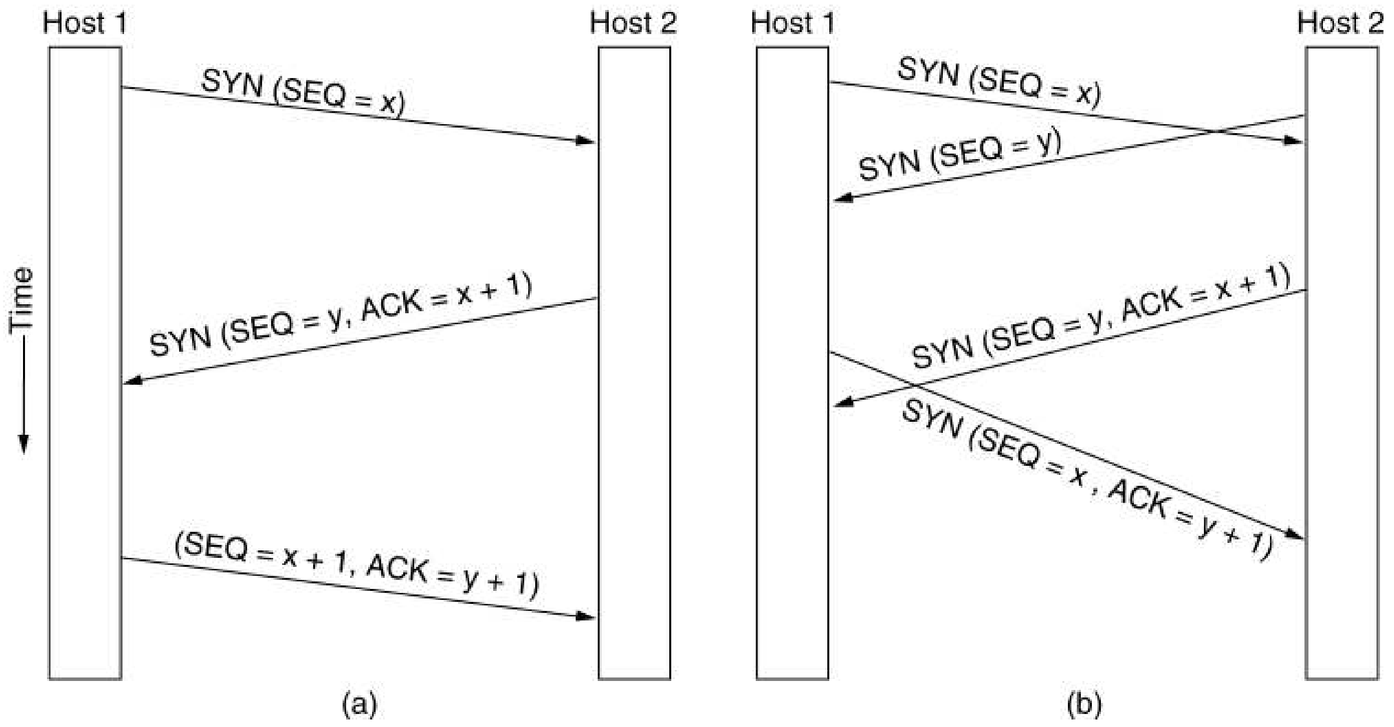
6.5.5. TCP Connection Establishment

- The client executes a CONNECT primitive, specifying the IP address and port to which it wants to connect, the maximum TCP segment size it is willing to accept, and optionally some user data (e.g., a password). The CONNECT primitive sends a TCP segment with the SYN bit ON and ACK bit OFF and waits for a response.

6.5.5. TCP Connection Establishment

- When this segment arrives at the destination, the TCP entity there check to see if there is a process that has done a LISTEN on the port given in the Destination port field. If not, it sends a reply with the RST bit on to reject the connection.
- If some process is listening to the port, that process is given the incoming TCP segment. It can then either accept or reject connection. If it accepts, an acknowledgement segment is sent back.

6.5.5. TCP Connection Establishment


Host 1
Host 2
Time
SYN (SEQ = x)
SYN (SEQ = y, ACK = x + 1)
(SEQ = x + 1, ACK = y + 1)
(a)
Host 1
Host 2
SYN (SEQ = x)
SYN (SEQ = y)
SYN (SEQ = y, ACK = x + 1)
SYN (SEQ = x , ACK = y + 1)
(b)

(a) TCP connection establishment in the normal case.
(b) Call collision.

6.5.6. TCP Connection Release

- Although TCP connections are full duplex, to understand how connections are released it is best to think of them as a pair of simplex connections.
- Each simplex connection is released independently of its sibling.
- To release a connection, either party can send a TCP segment with the FIN bit set, which means that it has no more data to transmit.

6.5.6. TCP Connection Release

- When FIN is acknowledged, that direction is shut down for new data.
- Data may continue to flow indefinitely in the other direction, however.
- When both directions have been shut down, the connection is released.

6.5.6. TCP Connection Release

- Normally, four TCP segments are needed to release a connection, one FIN and one ACK for each direction.
- However, it is possible for the first ACK and the second FIN to be contained in the same segment, reducing the total count to three.

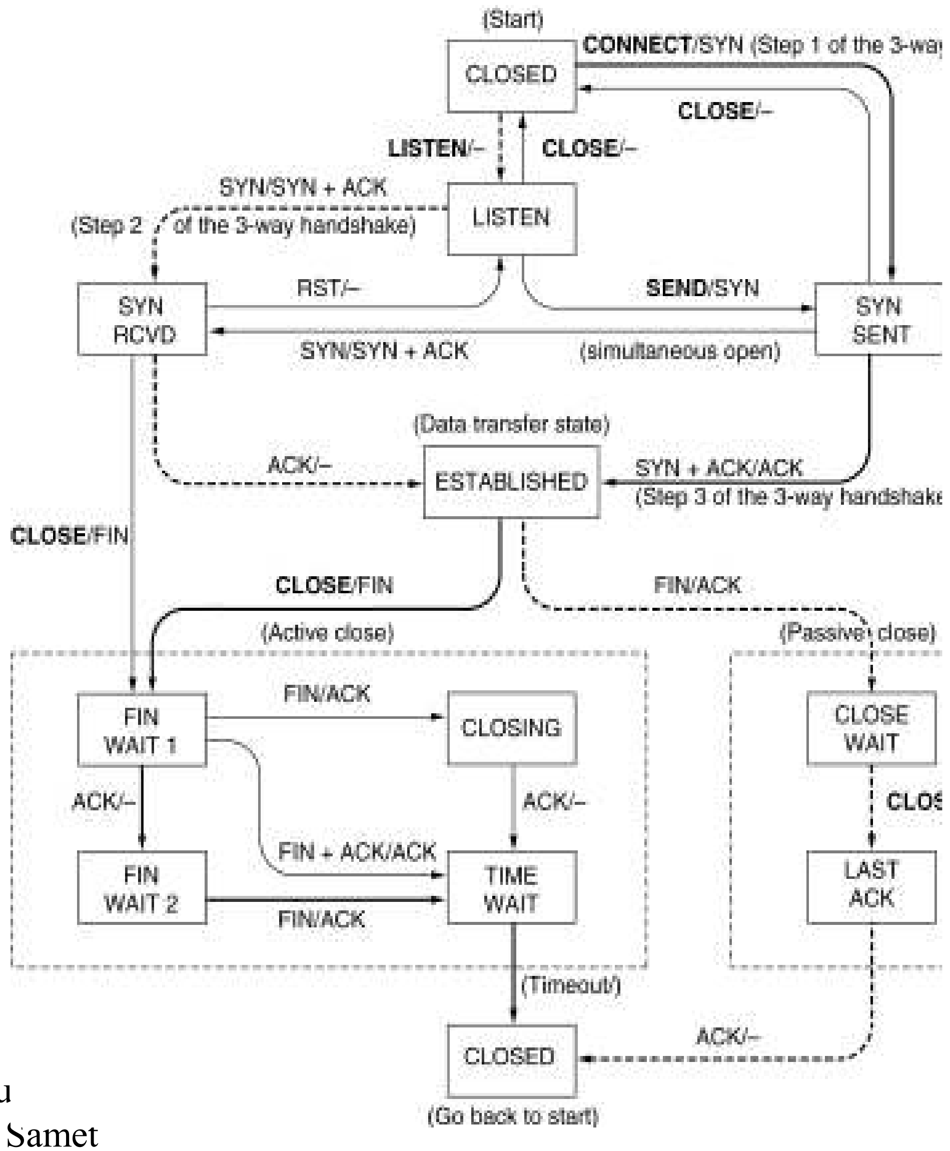
6.5.7 TCP Connection Management Modeling

State	Description
CLOSED	No connection is active or pending
LISTEN	The server is waiting for an incoming call
SYN RCVD	A connection request has arrived; wait for ACK
SYN SENT	The application has started to open a connection
ESTABLISHED	The normal data transfer state
FIN WAIT 1	The application has said it is finished
FIN WAIT 2	The other side has agreed to release
TIMED WAIT	Wait for all packets to die off
CLOSING	Both sides have tried to close simultaneously
CLOSE WAIT	The other side has initiated a release
LAST ACK	Wait for all packets to die off

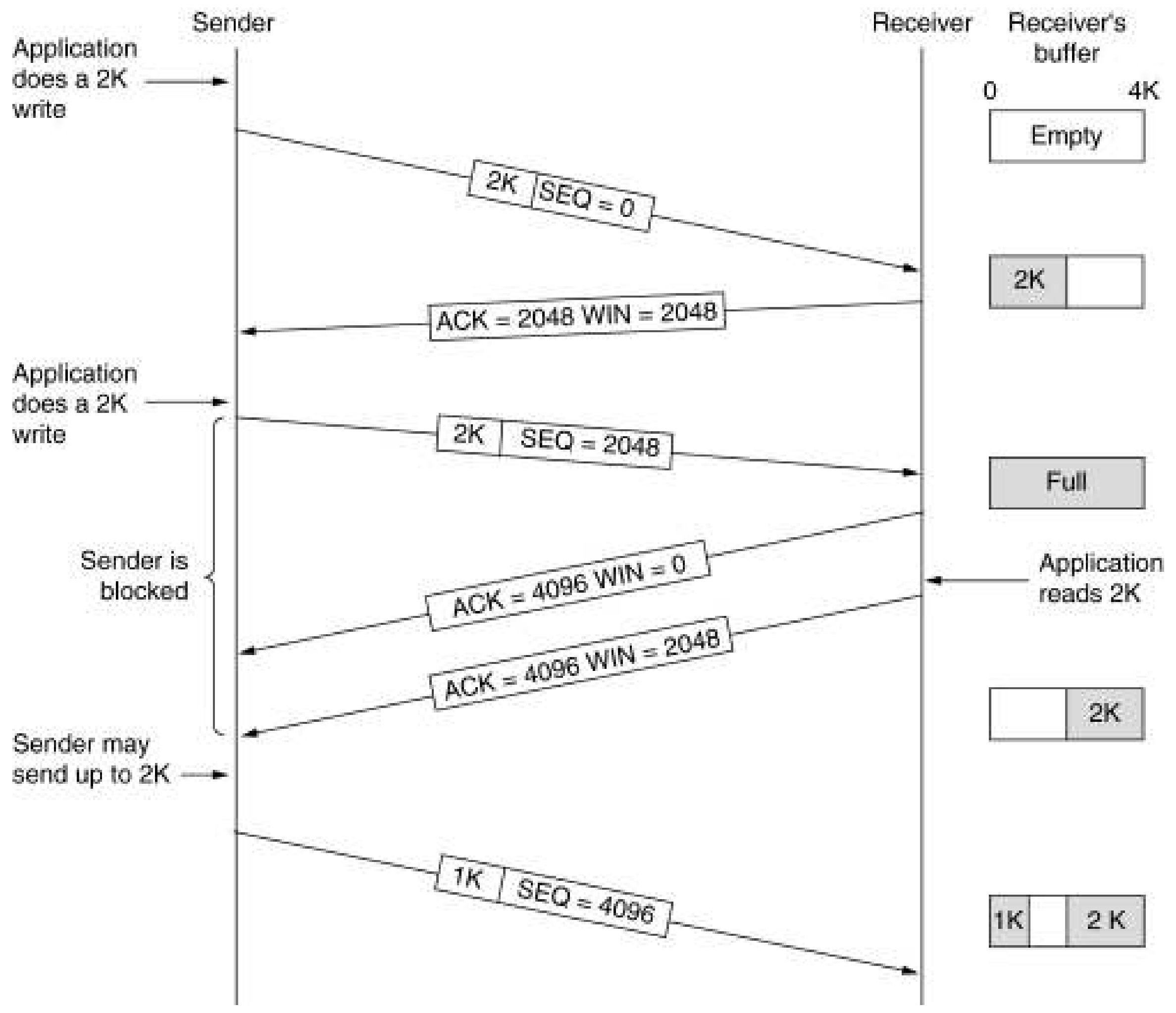
The states used in the TCP connection management finite state machine.

TCP Connection Management Modeling (2)

TCP connection management finite state machine. The heavy solid line is the normal path for a client. The heavy dashed line is the normal path for a server. The light lines are unusual events. Each transition is labeled by the event causing it and the action resulting from it, separated by a slash.

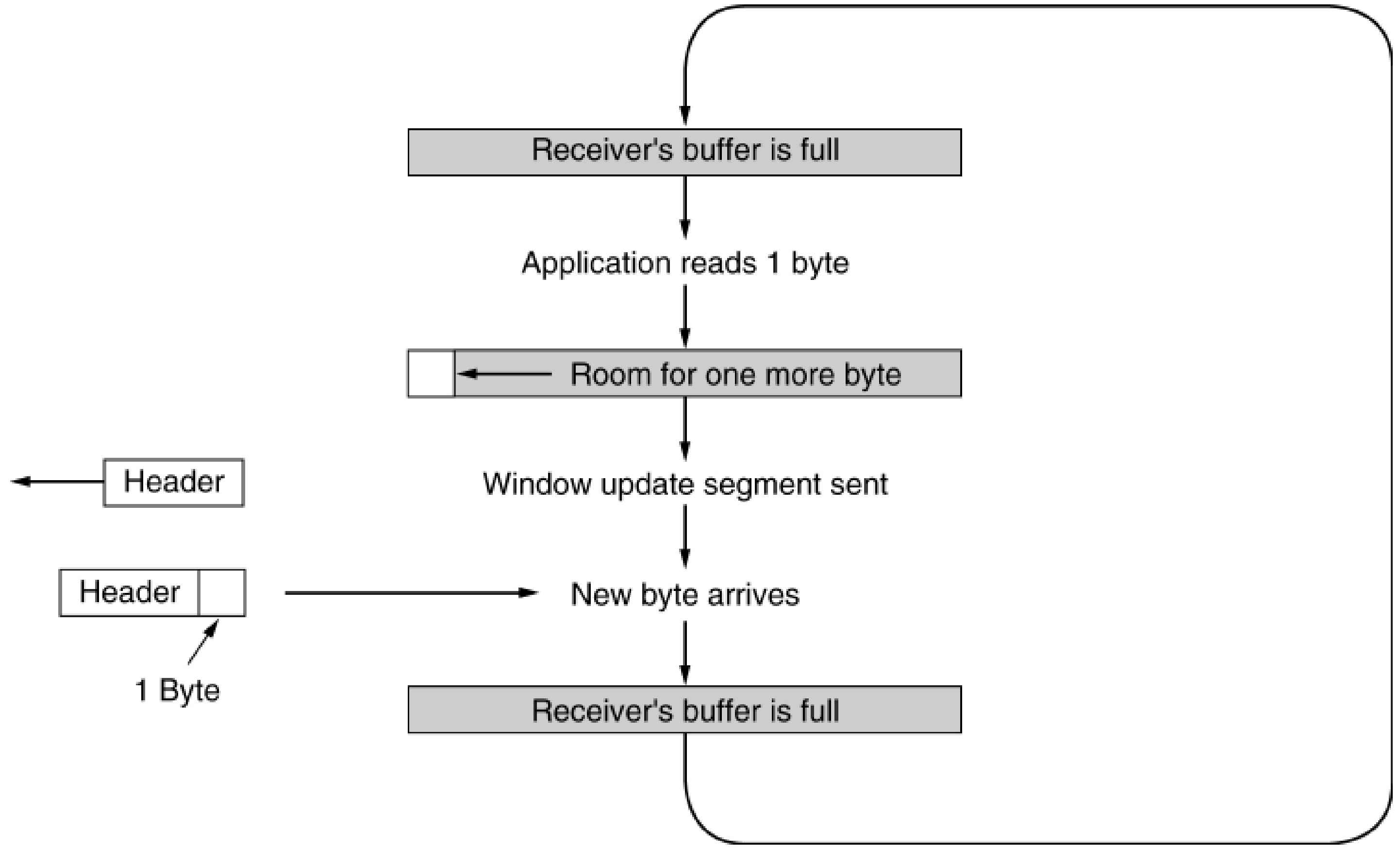


6.5.8. TCP Transmission Policy



Window management in TCP.

TCP Transmission Policy (2)

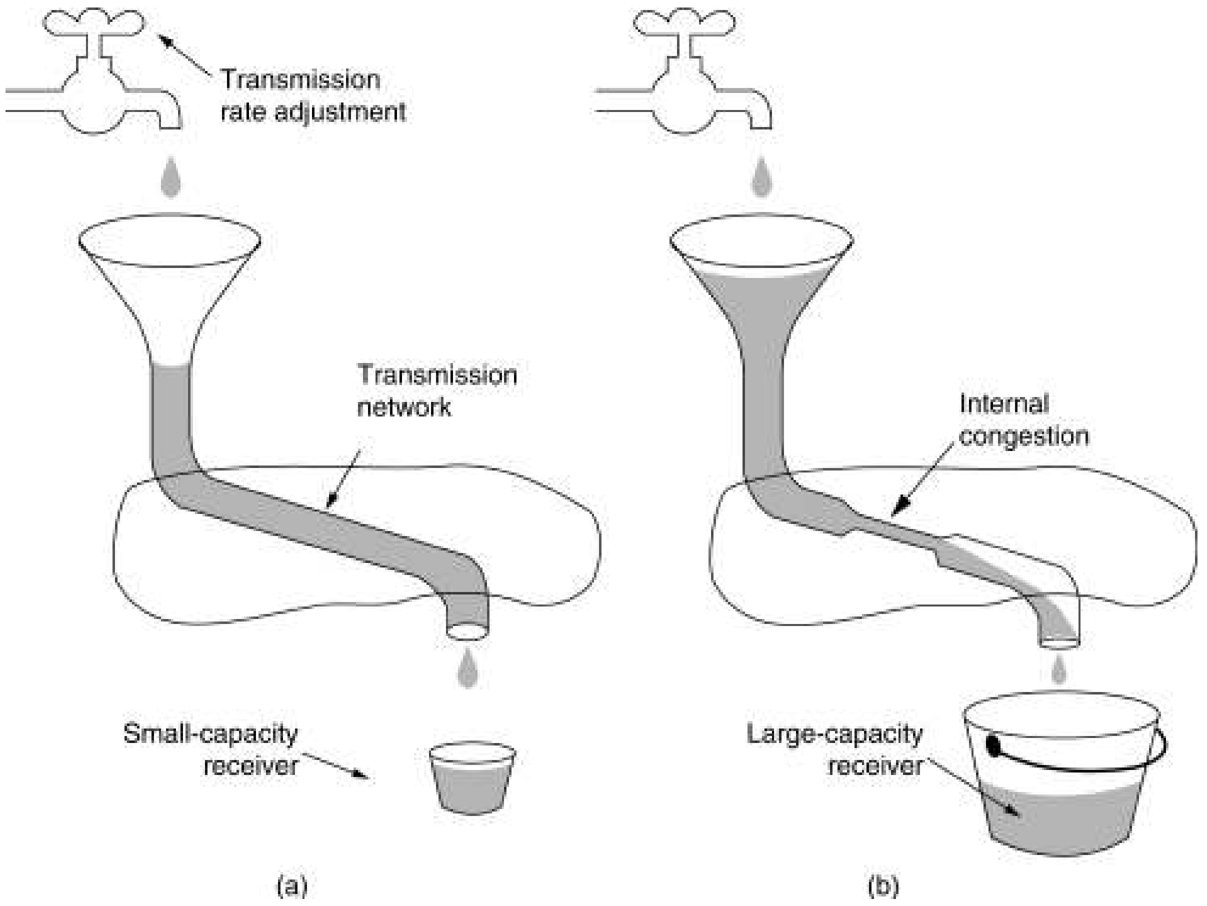


Silly window syndrome.

6.5.9. TCP Congestion Control

- When the load offered to any network is more than it can handle, congestion builds up.
- The Internet is no exception.
- The law of conservation of packets:
- To refrain from injecting a new packet into the network until an old one leaves (i.e., is delivered).
- TCP attempts to achieve this goal by dynamically manipulating the window size.

6.5.9. TCP Congestion Control



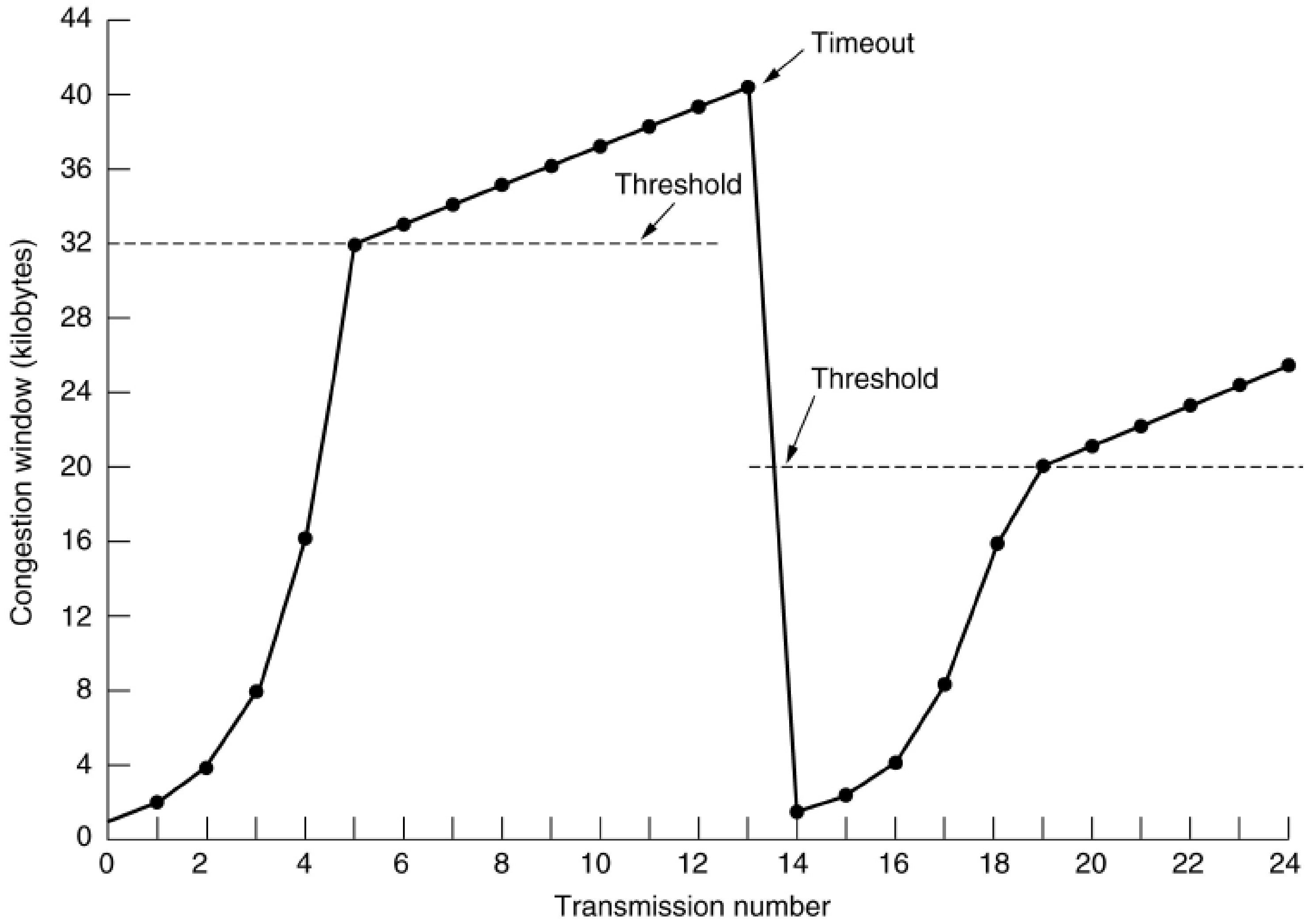
(a) A fast network feeding a low capacity receiver.

(b) A slow network feeding a high-capacity receiver.

6.5.9. TCP Congestion Control

- Most transmission timeouts on the Internet are due to congestion.
- The Internet solution is to realize that two potential problems exist – network capacity and receiver capacity – and to deal with each of them separately.
- To do so, each sender maintains two windows: the window the receiver has granted and a second window, the congestion window.

TCP Congestion Control (2)

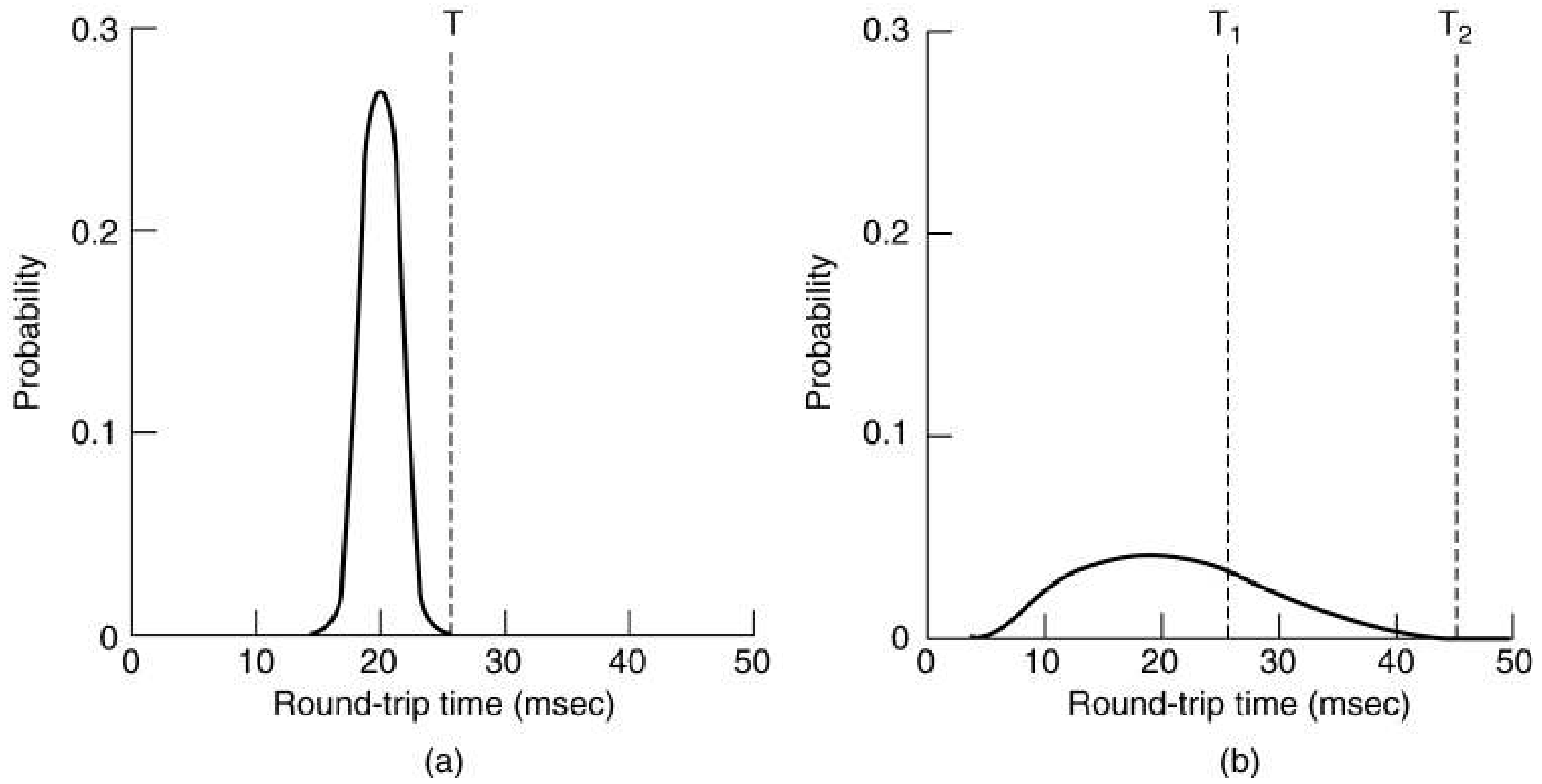


An example of the Internet congestion algorithm.

6.5.10. TCP Timer Management

- TCP uses multiple timers to do its work.
- The most important of these is the retransmission timer.
- When a segment is sent, a retransmission timer is started.
- If the segment is acknowledged before the timer expires, the timer is stopped.
- If, on the other hand, the timer goes off before the acknowledgement comes in, the segment is retransmitted (and the timer started again).

6.5.10. TCP Timer Management



(a) Probability density of ACK arrival times in the data link layer.

(b) Probability density of ACK arrival times for TCP.

6.5.10. TCP Timer Management

- How long should the timeout interval be?
- The solution is to use a highly dynamic algorithm that constantly adjusts the timeout interval, based on continuous measurements of network performance.
- The algorithm generally used by TCP is due to Jacobson.
- For each connection, TCP maintains a variable, RTT, that is the best current estimate of round-trip time to destination in question.

6.5.11. Wireless TCP and UDP

- Most TCP implementations have been carefully optimized based on assumptions that are true for wired networks but that fail for wireless networks.
- The principal problem is the congestion control algorithm.
- Nearly all TCP implementation nowadays assume that timeouts are caused by congestion, not by lost packets.

6.5.11. Wireless TCP and UDP

- Consequently, when a timer goes off, TCP slow down and sends less vigorously (e.g., Jacobson's slow start algorithm)
- The idea behind this approach is to reduce the network load and thus alleviate the congestion.

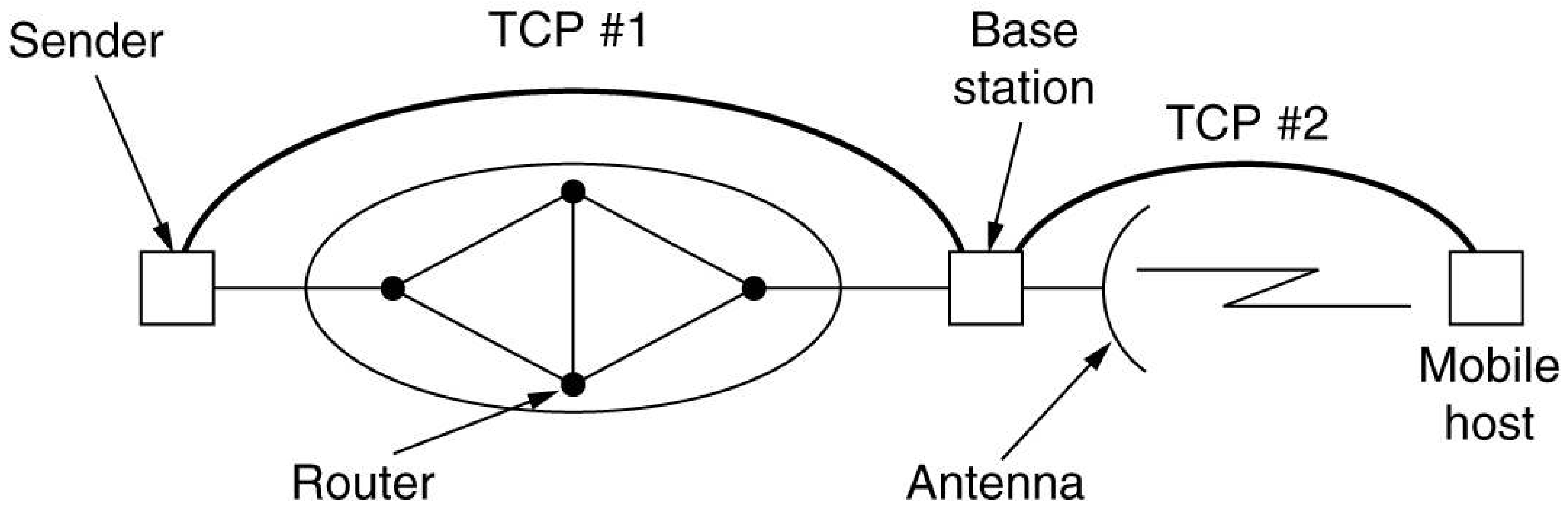
6.5.11. Wireless TCP and UDP

- Unfortunately, wireless transmission links are highly unreliable.
- They lose packets all the time.
- The proper approach to dealing with lost packets is to send them again, and as quickly as possible.
- Slowing down just makes matters worse.

6.5.11. Wireless TCP and UDP

- Indirect TCP – is to split the TCP connection into two separate connections.
- The first connection goes from the sender to the base station.
- The second one goes from the base station to the receiver.
- The base station simply copies packets between the connections in both directions.

6.5.11. Wireless TCP and UDP

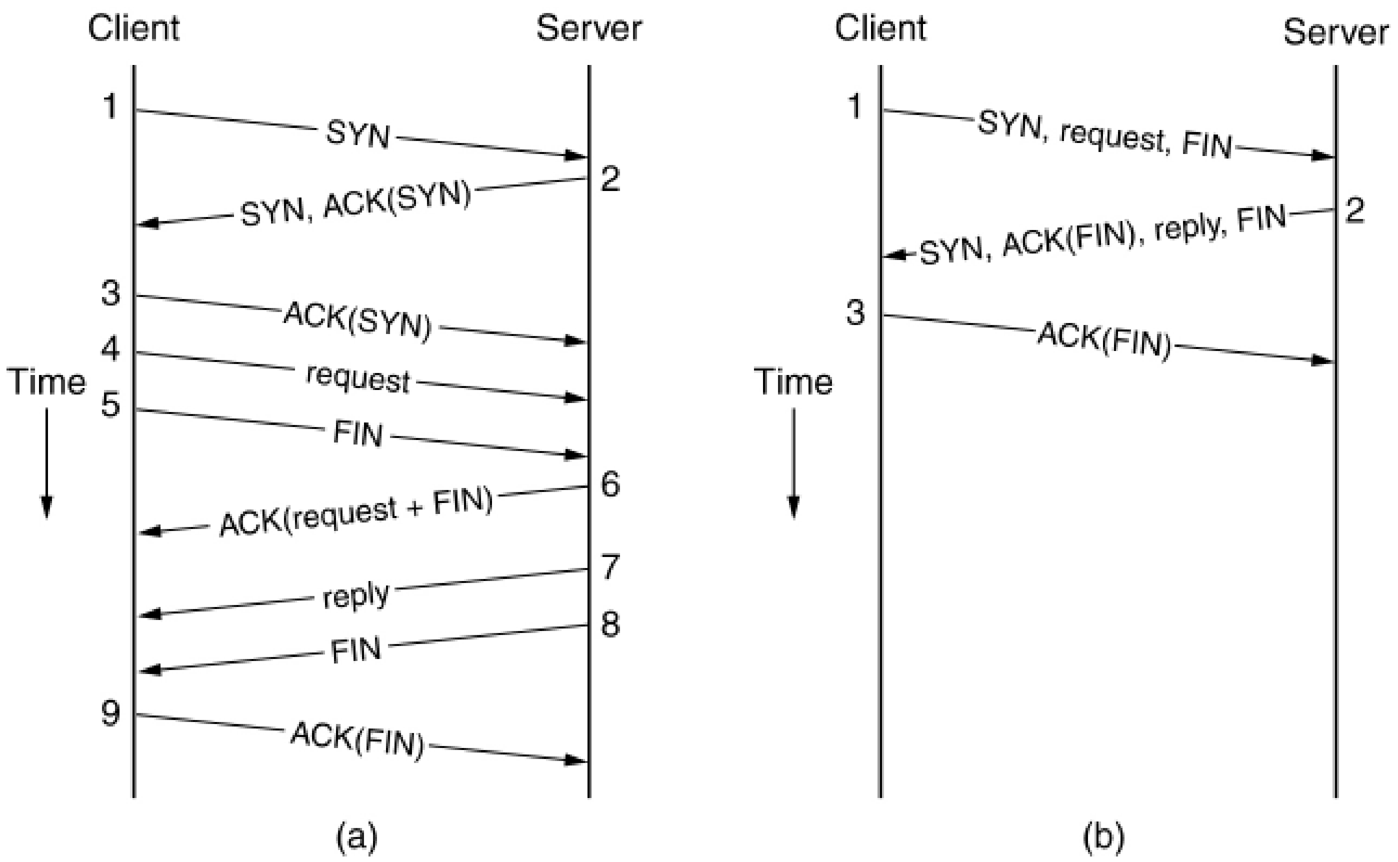


Splitting a TCP connection into two connections.

6.5.12. Transitional TCP

- The problem is the efficiency.
- The normal sequence of packets for doing an RPC over TCP consists of nine packets in the best case.
- The question quickly arises of whether there is some way to combine the efficiency of RPC using UDP with the reliability of TCP.
- ALMOST. It can be done with an experimental TCP variant called T/TCP.

6.5.12. Transitional TCP



(a) RPC using normal TCP.
(b) RPC using T/TCP.

6.5.12. Transitional TCP

- 1: SYN, request, FIN
    - I want to establish a connection, here is the data, and I am done.
- 2: SYN, ACK(FIN), reply, FIN:
    - I acknowledge your FIN, here is the answer, and I am done.
- 3: ACK(FIN)
    - The client then acknowledges the server's FIN and the protocol terminates in three messages.

6.6. Performance Issues

- Performance issues are very important in computer networks.
- When hundreds or thousands of computers are interconnected, complex interaction, with unforeseen consequences, are common.
- Frequently, this complexity leads to poor performance and no one knows why.

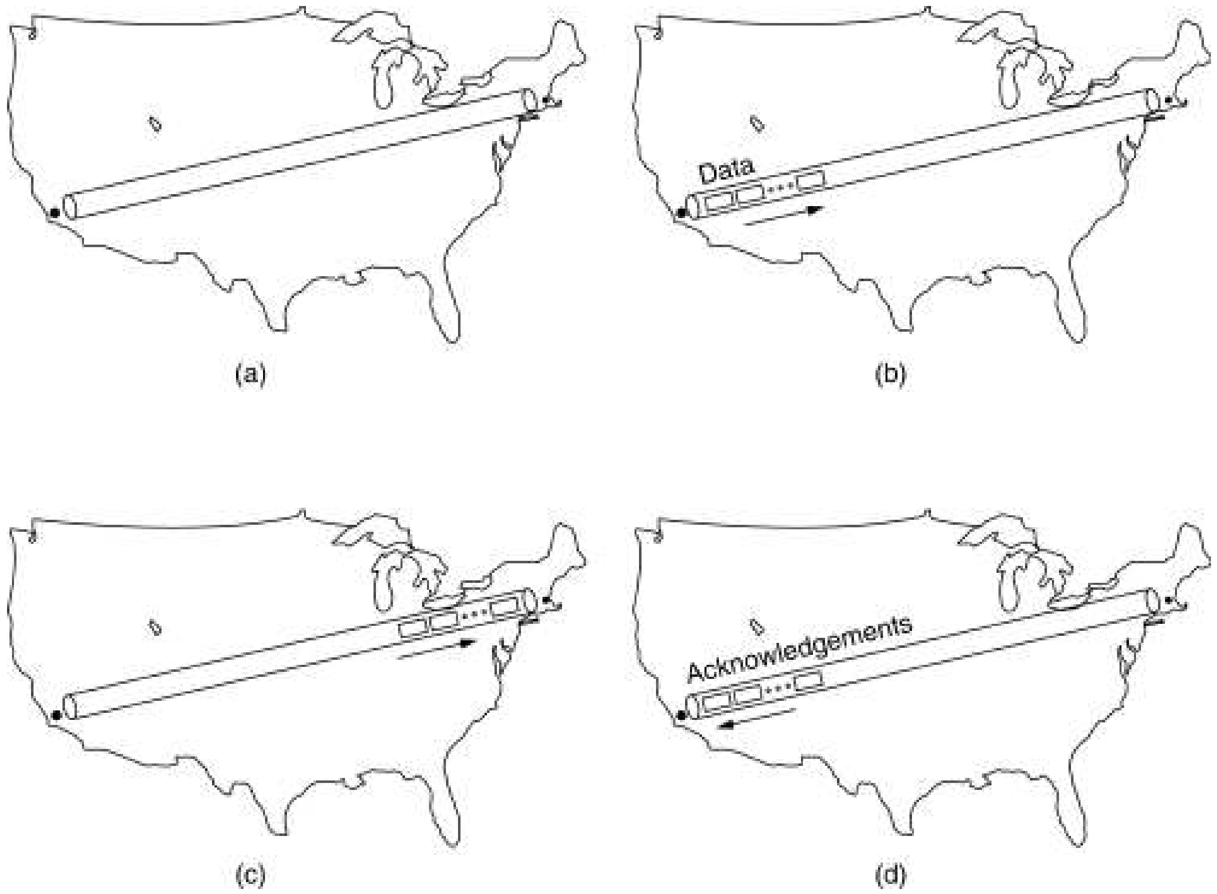
6.6. Performance Issues

- Network performance is typically dominated by protocol and TPDU processing overhead, and this situation gets worse at higher speeds.
- Protocols should be designed to minimize the number of TPDUs, context switches, and times each TPDU is copied.
- For gigabit networks, simple protocols are called for.

6.6. Performance Issues

- Performance Problems in Computer Networks
- Network Performance Measurement
- System Design for Better Performance
- Fast TPDU Processing
- Protocols for Gigabit Networks

Performance Problems in Computer Networks



The state of transmitting one megabit from San Diego to Boston
(a) At t = 0, (b) After 500 µsec, (c) After 20 msec, (d) after 40 msec.

Network Performance Measurement

The basic loop for improving network performance.

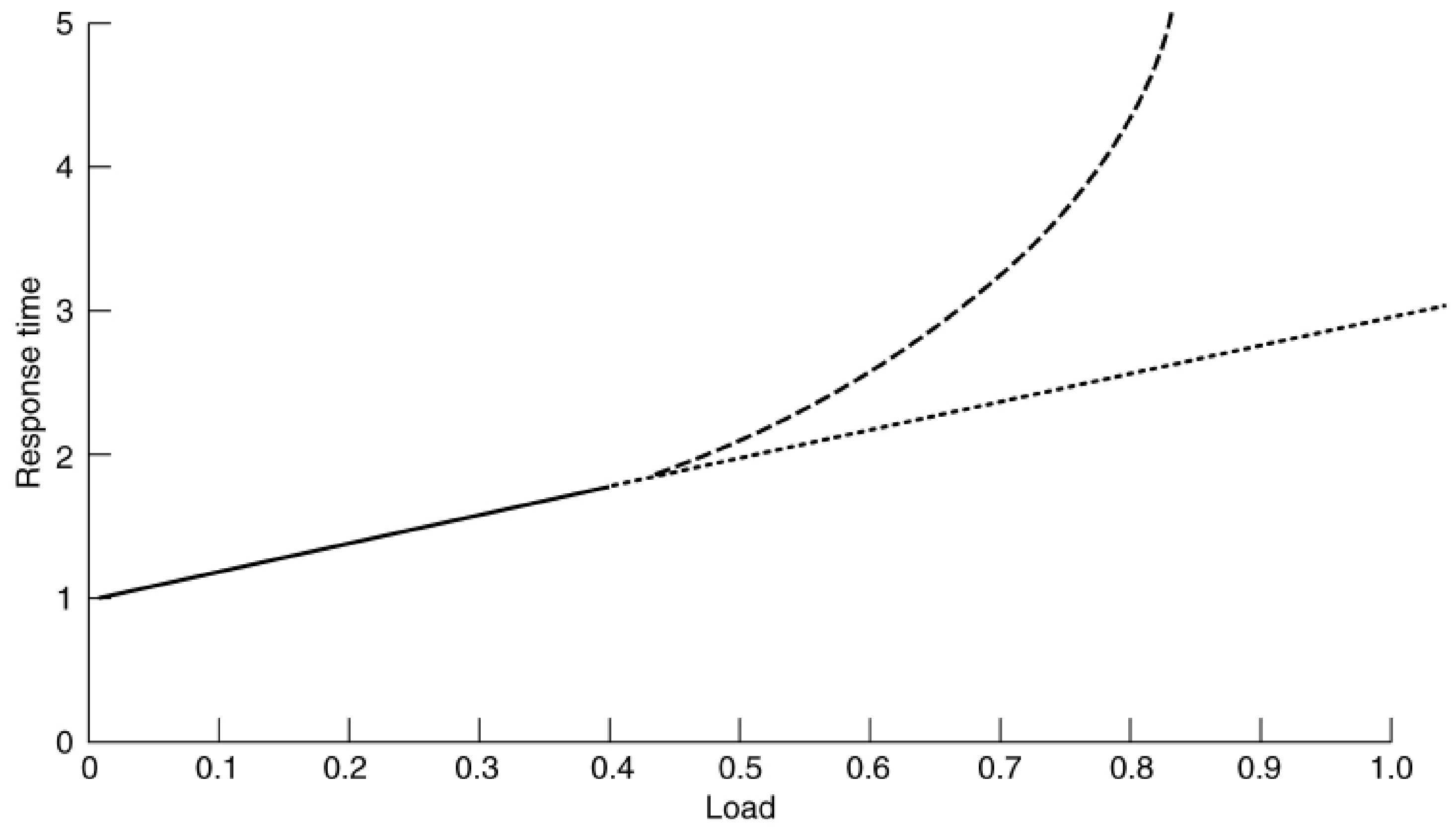
1. Measure relevant network parameters, performance.
2. Try to understand what is going on.
3. Change one parameter.

System Design for Better Performance

Rules:

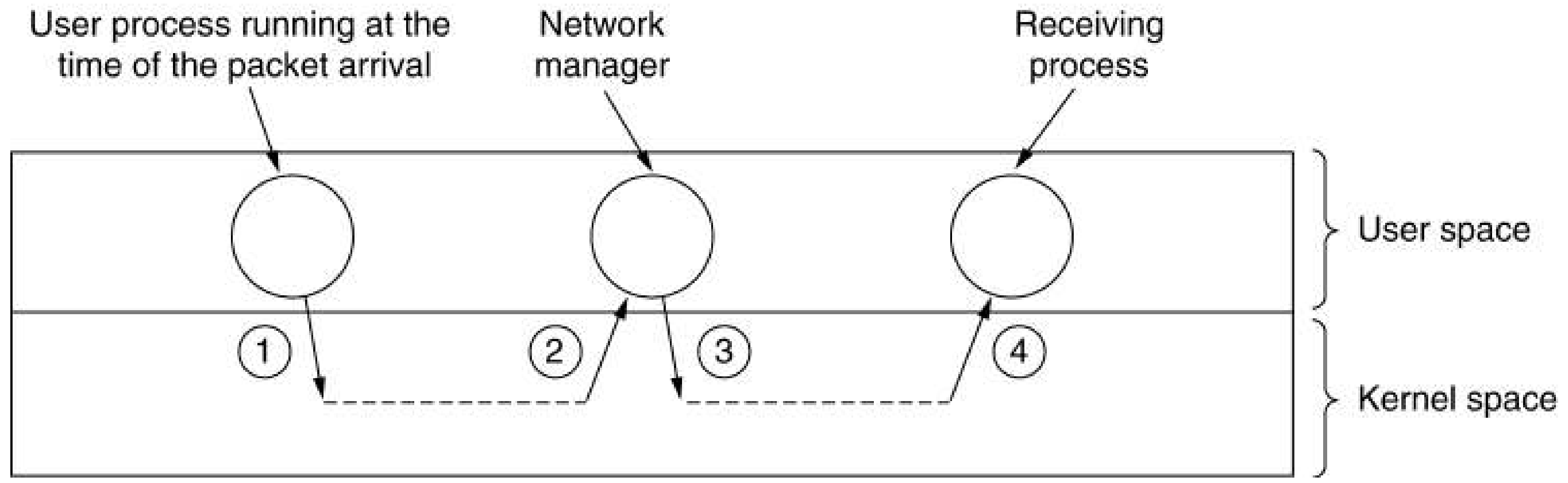
1. CPU speed is more important than network speed.
2. Reduce packet count to reduce software overhead.
3. Minimize context switches.
4. Minimize copying.
5. You can buy more bandwidth but not lower delay.
6. Avoiding congestion is better than recovering from it.
7. Avoid timeouts.

System Design for Better Performance (2)



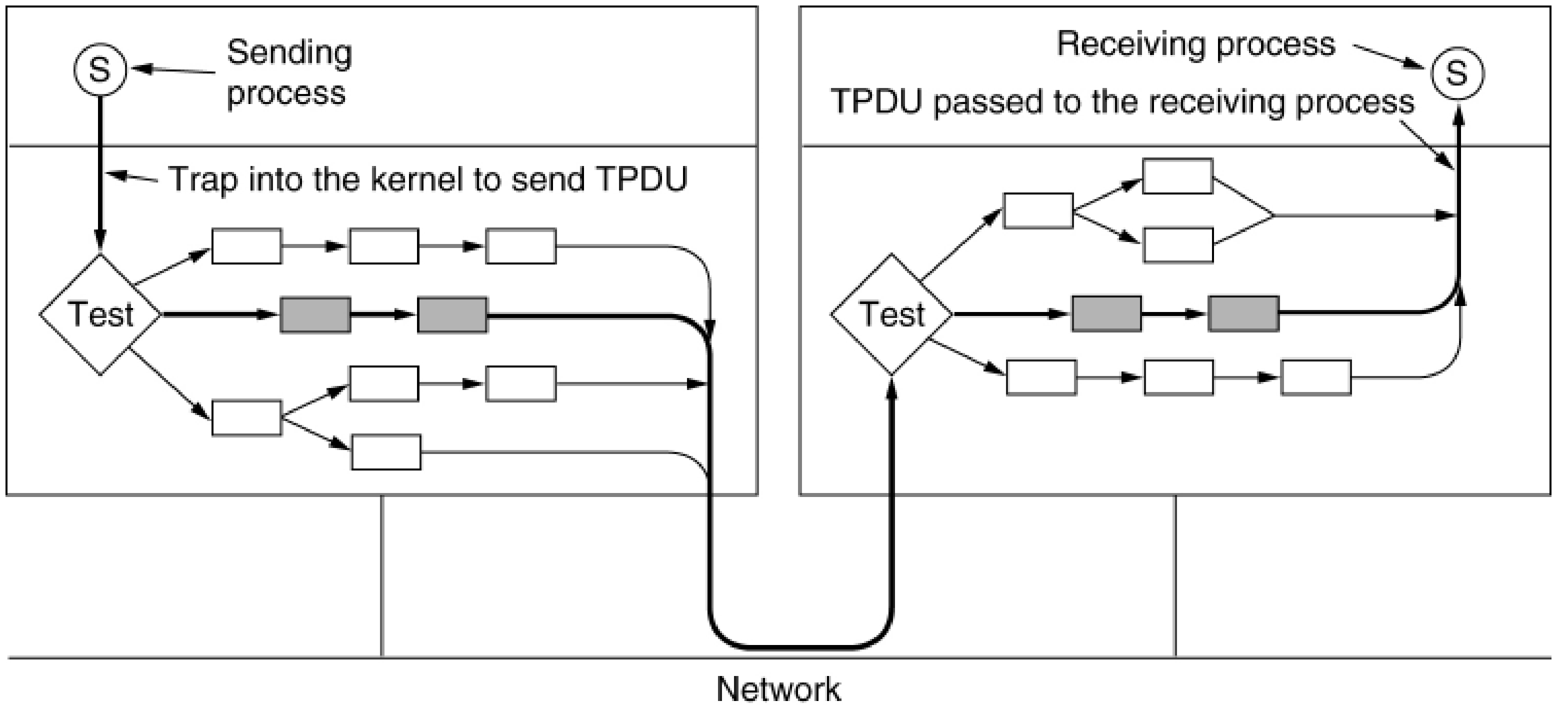
Response as a function of load.

System Design for Better Performance (3)



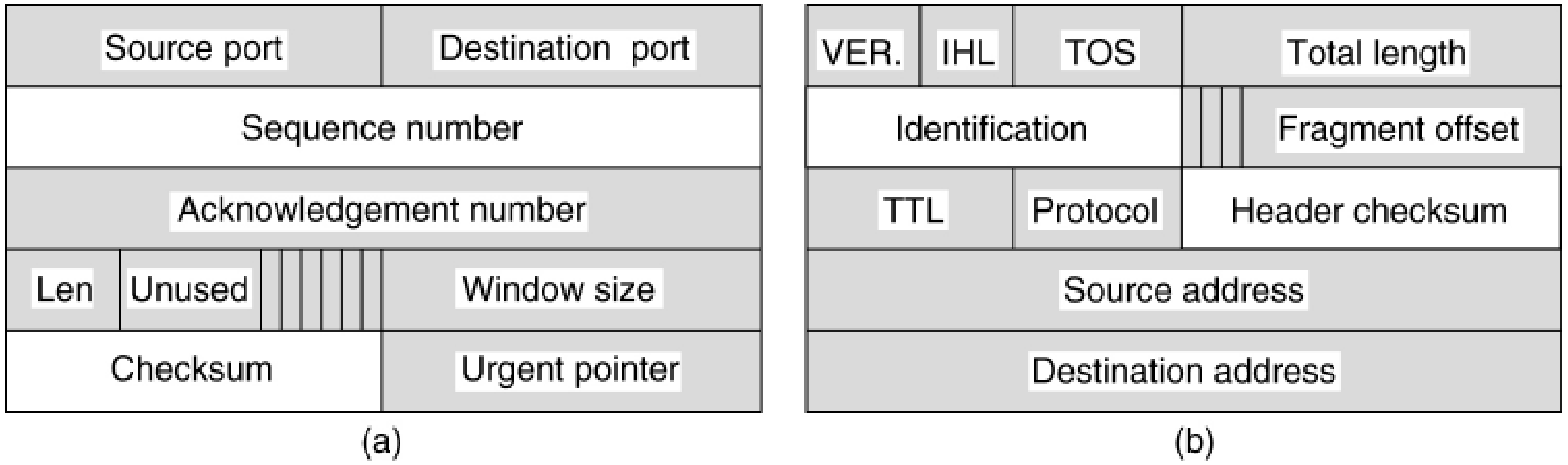
Four context switches to handle one packet with a user-space network manager.

Fast TPDU Processing


S
Sending process
Trap into the kernel to send TPDU
Test
Receiving process
S
TPDU passed to the receiving process
Test
Network

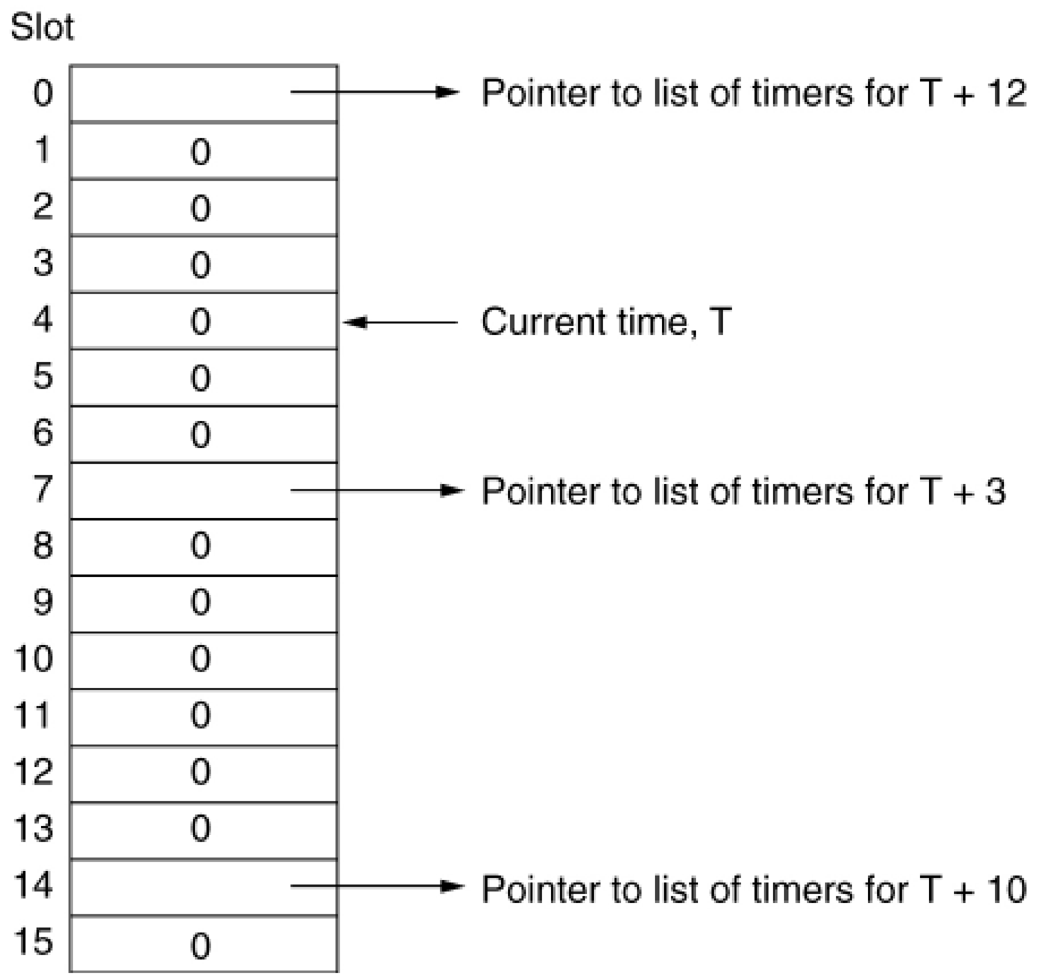
The fast path from sender to receiver is shown with a heavy line.
The processing steps on this path are shaded.

Fast TPDU Processing (2)



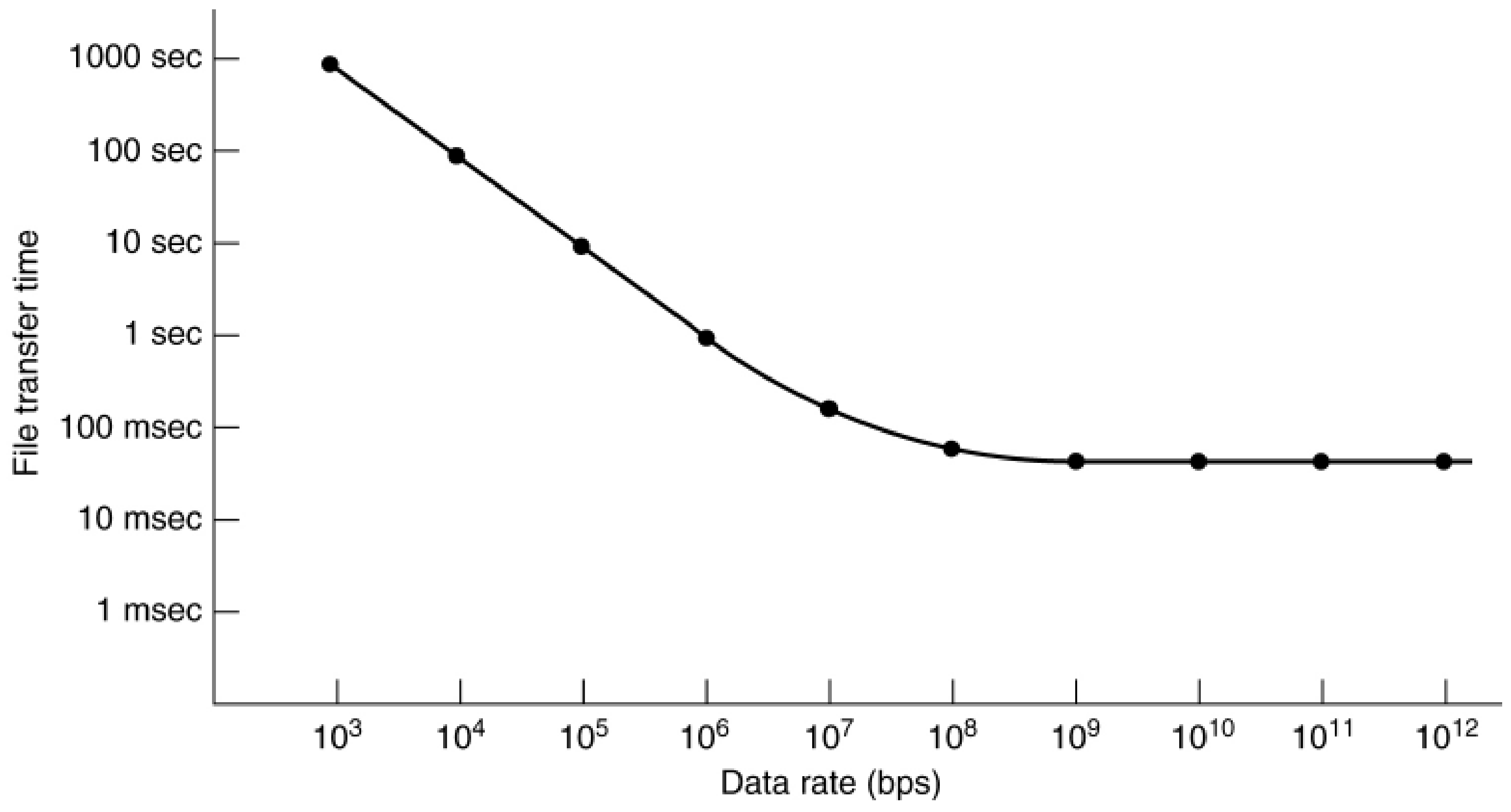
(a) TCP header. (b) IP header. In both cases, the shaded fields are taken from the prototype without change.

Fast TPDU Processing (3)


Slot
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
0
0
0
0
0
0
0
0
0
0
0
0
0
Pointer to list of timers for T + 12
Current time, T
Pointer to list of timers for T + 3
Pointer to list of timers for T + 10

A timing wheel.

Protocols for Gigabit Networks



Time to transfer and acknowledge a 1-megabit file over a 4000-km line.