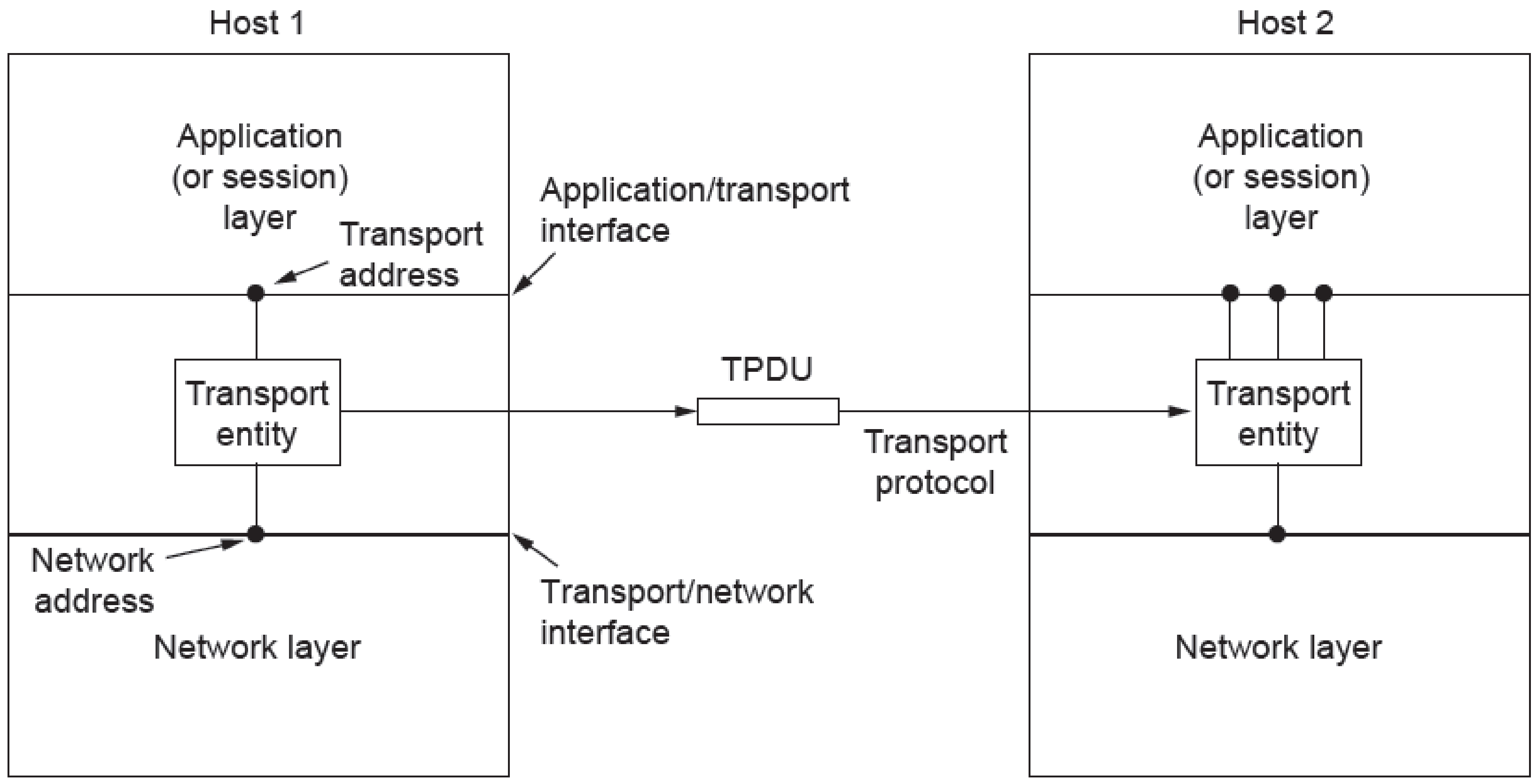
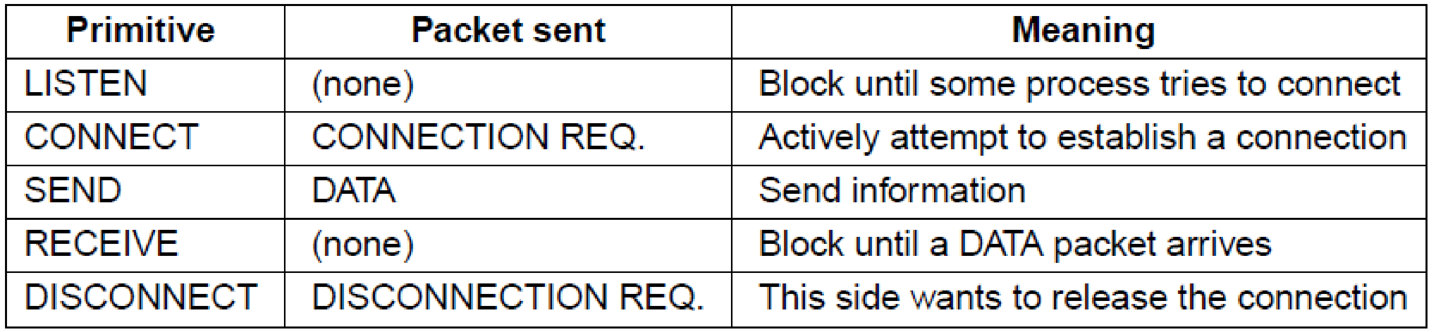
Chapter 6
The Transport Layer

Services Provided to the Upper Layers



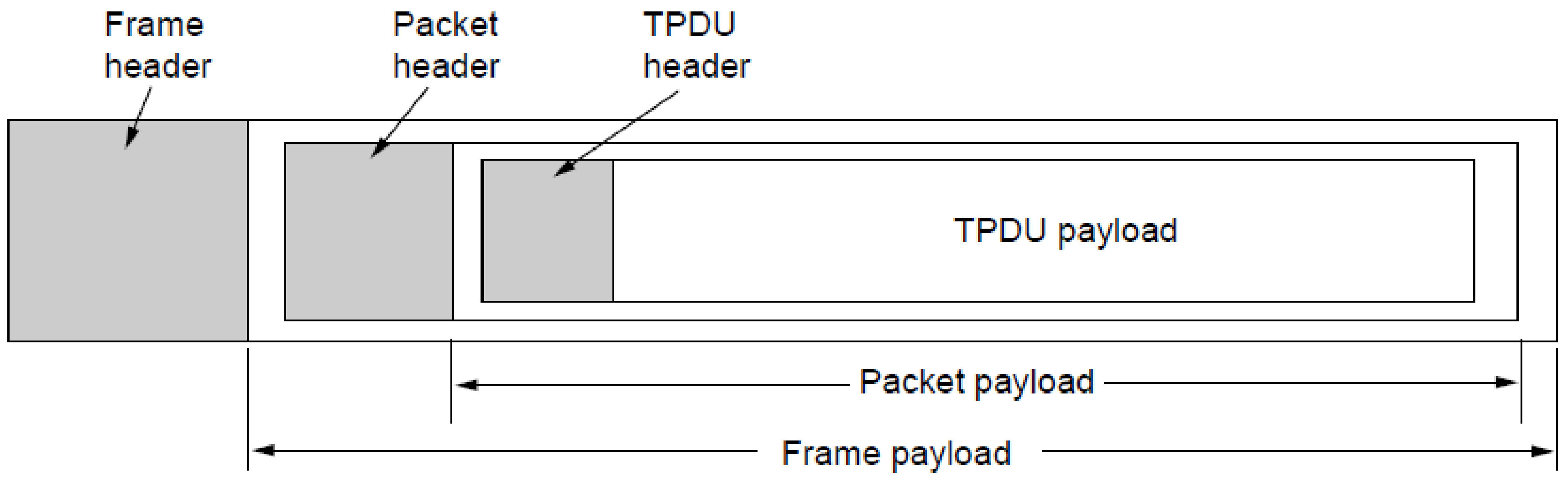
Transport Service Primitives (1)

The primitives for a simple transport service



Primitive	Packet sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	This side wants to release the connection

Transport Service Primitives (2)
Nesting of TPDUs, packets, and frames.



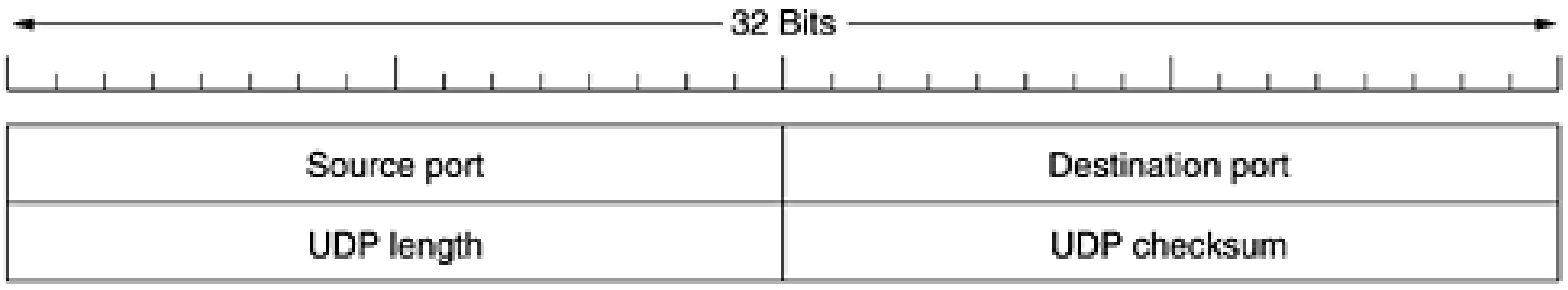
The Internet Transport Protocols

- The Internet has two main protocols in the transport layer, a **connectionless protocol** and a **connection-oriented** one.
- The connectionless protocol is **UDP**. The connection-oriented protocol is **TCP**.
- **TCP** (Transmission Control Protocol) to ensure destination received segments
- **UDP** (User Datagram Protocol) to send segments without assurance of delivery

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.
- UDP transmits **segments** consisting of an 8-byte header followed by the payload.

Figure 6-23. The UDP header.



- 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.
- The main value of having UDP over just using raw IP is the addition of the source and destination ports.
- Without the port fields, the transport layer would not know what to do with the packet. With them, it delivers segments correctly.
- The source port is primarily needed when a reply must be sent back to the source. By copying the *source port* field from the incoming segment into the *destination port* field of the outgoing segment, the process sending the reply can specify which process on the sending machine is to get it.
- The *UDP length field* includes the 8-byte header and the data. The UDP checksum is optional and stored as 0 if not computed

- Some of the things that UDP does *not do. It does not* do flow control, error control, or retransmission upon receipt of a bad segment.
- 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 (the Domain Name System.
- 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.

The Internet Transport Protocols: TCP

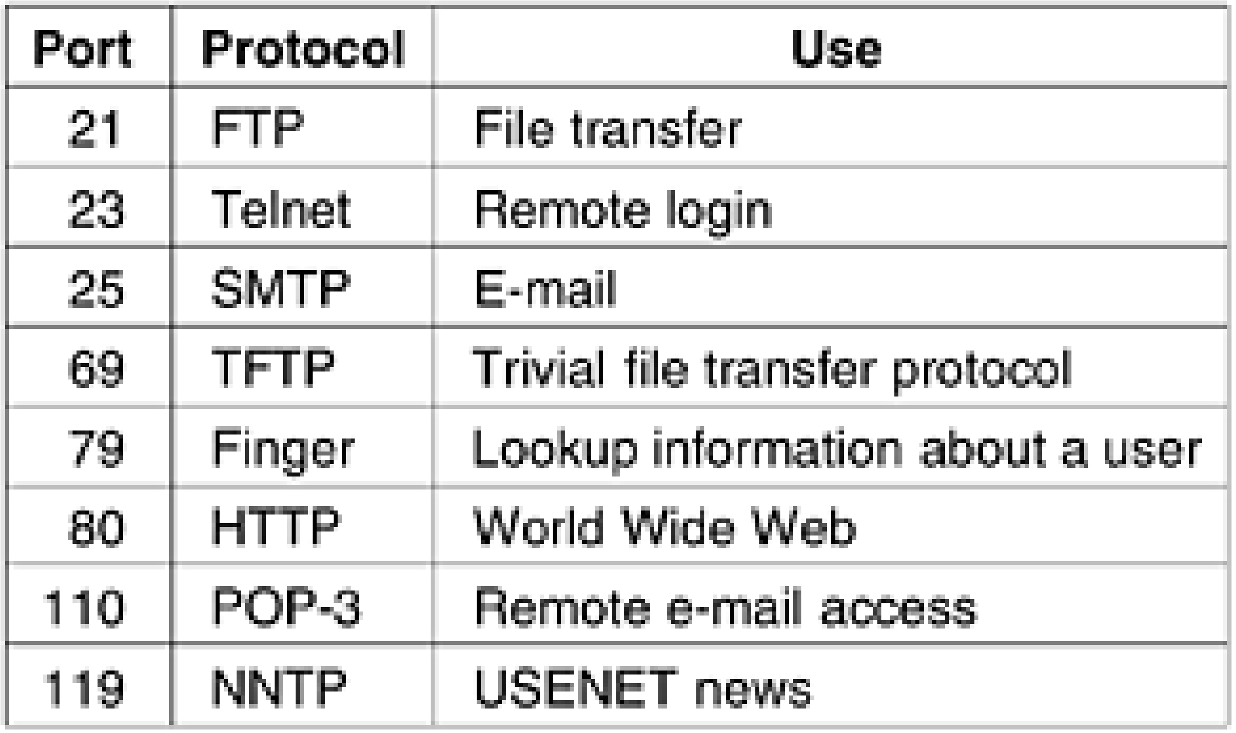
- UDP is a simple protocol and it has some niche uses, such as client-server interactions and multimedia, but for most Internet applications, reliable, sequenced delivery is needed.
- UDP cannot provide this, so another protocol is required. It is called TCP and is the main workhorse of the Internet.

- **TCP (Transmission Control Protocol)** was specifically designed to provide a reliable end-to-end byte stream over an unreliable internetwork.
- An internetwork differs from a single network because different parts may have wildly 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.

- Each machine supporting TCP has a TCP transport entity.
- A TCP 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.
- When datagrams containing TCP data arrive at a machine, they are given to the TCP entity, which reconstructs the original byte streams.
- The IP layer gives no guarantee that datagrams will be delivered properly, so it is up to TCP to time out and retransmit them as need be.
- It is also up to TCP to reassemble Datagrams into messages in the proper sequence.
- In short, TCP must furnish the reliability that most users want.

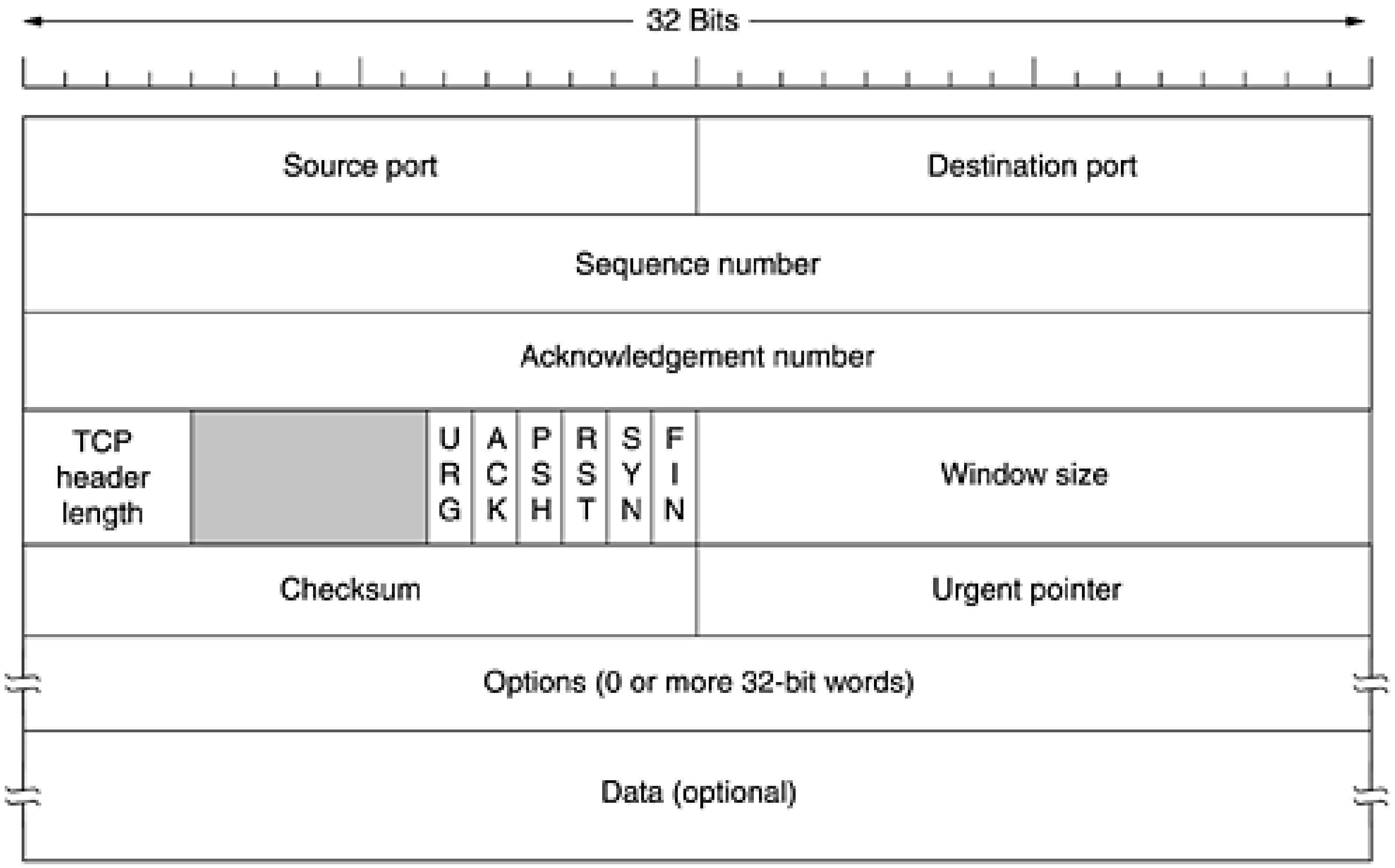
- TCP service is obtained by both the sender and receiver creating end points, called sockets.
- Each socket has a socket number (address) consisting of the IP address of the host and a 16-bit number local to that host, called a **port.**

Figure 6-27. 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 information about a user
80	HTTP	World Wide Web
110	POP-3	Remote e-mail access
119	NNTP	USENET news

Figure 6-29. The TCP header.



- Every segment begins with a fixed-format, 20-byte header.
- The fixed header may be followed by header options.
- After the options, if any, up to 65,495 data bytes may follow.

TCP vs UDP

- Both use **port numbers**
  - application-specific construct serving as a communication endpoint and consist of 16-bit unsigned integer, thus ranging from 0 to 65535 to provide **end-to-end** transport
- UDP: User Datagram Protocol
  - no acknowledgements
  - no retransmissions
  - out of order, duplicates possible
  - connectionless, i.e., app indicates destination for each packet
- TCP: Transmission Control Protocol
  - reliable **byte-stream channel** (in order, all arrive, no duplicates)
  - flow control
  - connection-oriented
  - bidirectional