Chapter 14

User Datagram Program (UDP)
Network layer versus transport layer
Port numbers

Ephemeral port number

Well-known port number
The well-known port numbers are less than 1,024.
IP addresses versus port numbers

Destination IP address selects the server

Destination port number selects the process

TCP/IP Protocol Suite
✓ Server socket address defines the server process uniquely.
Encapsulation and Decapsulation

- Encapsulation happens at the sender site.
- The packets at the transport layers are called *user datagrams, segments, or packets*.

- Decapsulation happens at the receive site.
The transport layer at the source performs **multiplexing** (many to one).

The transport layer at the destination performs **demultiplexing** (one to many).
Pushing or pulling

✓ Producer-Consumer paradigm
✓ One host produce items and another host consume them
✓ Balance between production and consumption rates
✓ Production rate is faster then consumption rate. The consumer needs to discard some items --- flow control
✓ Production rate is slower than consumption rate. The consumer should wait.

![Diagram]

a. Pushing

b. Pulling

✓ Pushing --- producer delivers items whenever they are produced
✓ Pulling --- producer delivers the items after consumer has requested them
Flow control at the transport layer

TCP/IP Protocol Suite
Error control at the transport layer

Sender
Transport layer

Packets

Receiver
Transport layer

Error Control
Congestion Control

✓ Congestion
  ✓ The load on the network (the number of packets sent to the network) is greater than the capacity of the network (the number of packages a network can handle).

✓ Congestion Control
  ✓ The mechanism to keep the load below the capacity
Connectionless service
Connection-oriented service

1. Client open-request packet
2. Acknowledgment for packet 1
3. Server open-request packet
4. Acknowledgment for packet 3
5. Client close-request packet
6. Acknowledgment for packet 5
7. Server close-request packet
8. Acknowledgment for packet 7
14.1 Introduction
14.2 User Datagram
14.3 UDP Services
14.4 UDP Application
14.5 UDP Package
Figure 14.1 shows the relationship of the User Datagram Protocol (UDP) to the other protocols and layers of the TCP/IP protocol suite: UDP is located between the application layer and the IP layer, and serves as the intermediary between the application programs and the network operations.
Figure 14.1  Position of UDP in the TCP/IP protocol suite
**14-2 USER DATAGRAM**

- UDP is a **connectionless, unreliable** transport protocol.
  - No flow control, no error control, no congestion control
  - Simple protocol using a minimum of overhead

- UDP packets, called **user datagrams**, have a fixed-size header of 8 bytes. Figure 14.2 shows the format of a user datagram.
Figure 14.2  User datagram format

8 to 65,535 bytes

8 bytes

Header  Data

a. UDP user datagram

0 16 31

<table>
<thead>
<tr>
<th>Source port number</th>
<th>Destination port number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total length</td>
<td>Checksum</td>
</tr>
</tbody>
</table>

b. Header format
Example 14.1

The following is a dump of a UDP header in hexadecimal format.

[Hexadecimal code: CB84000D001C001C]

a. What is the source port number?
b. What is the destination port number?
c. What is the total length of the user datagram?
d. What is the length of the data?
e. Is the packet directed from a client to a server or vice versa?
f. What is the client process?
**Example 14.1 Continued**

**Solution**

a. The source port number is the first four hexadecimal digits \((\text{CB84})_{16}\) or 52100.

b. The destination port number is the second four hexadecimal digits \((\text{000D})_{16}\) or 13.

c. The third four hexadecimal digits \((\text{001C})_{16}\) define the length of the whole UDP packet as 28 bytes.

d. The length of the data is the length of the whole packet minus the length of the header, or \(28 - 8 = 20\) bytes.

e. Since the destination port number is 13 (well-known port), the packet is from the client to the server.

f. The client process is the Daytime (see Table 14.1).
<table>
<thead>
<tr>
<th>Port</th>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Echo</td>
<td>Echoes a received datagram back to the sender</td>
</tr>
<tr>
<td>9</td>
<td>Discard</td>
<td>Discards any datagram that is received</td>
</tr>
<tr>
<td>11</td>
<td>Users</td>
<td>Active users</td>
</tr>
<tr>
<td>13</td>
<td>Daytime</td>
<td>Returns the date and the time</td>
</tr>
<tr>
<td>17</td>
<td>Quote</td>
<td>Returns a quote of the day</td>
</tr>
<tr>
<td>19</td>
<td>Chargen</td>
<td>Returns a string of characters</td>
</tr>
<tr>
<td>53</td>
<td>Domain</td>
<td>Domain Name Service (DNS)</td>
</tr>
<tr>
<td>67</td>
<td>Bootps</td>
<td>Server port to download bootstrap information</td>
</tr>
<tr>
<td>68</td>
<td>Bootpc</td>
<td>Client port to download bootstrap information</td>
</tr>
<tr>
<td>69</td>
<td>TFTP</td>
<td>Trivial File Transfer Protocol</td>
</tr>
<tr>
<td>111</td>
<td>RPC</td>
<td>Remote Procedure Call</td>
</tr>
<tr>
<td>123</td>
<td>NTP</td>
<td>Network Time Protocol</td>
</tr>
<tr>
<td>161</td>
<td>SNMP</td>
<td>Simple Network Management Protocol</td>
</tr>
<tr>
<td>162</td>
<td>SNMP</td>
<td>Simple Network Management Protocol (trap)</td>
</tr>
</tbody>
</table>
TCP/IP Protocol Suite

14-3 UDP Services

✓ Process-to-Process communication
✓ Connectionless service
✓ No flow control
✓ No congestion control
✓ No error control except for checksum
  ✓ The sender does not know if a message has been lost or duplicated
  ✓ When the receiver detects an error through the checksum, the user datagram is silently discarded
Figure 14.5 *Encapsulation and decapsulation*

Sender Process

1. Message
   - UDP header
   - UDP data
   - IP header
   - IP data
   - Frame header
   - Frame data

   a. Encapsulation

Receiver Process

1. Message
   - UDP header
   - UDP data
   - IP header
   - IP data
   - Frame header
   - Frame data

   b. Decapsulation
Figure 14.6  Queues in UDP
Figure 14.7  Multiplexing and demultiplexing
UDP is a connectionless simple protocol with an optional checksum for error detection.

- If the request and response can each fit in one single user datagram, UDP may be preferable.
- Less overhead, less delay
A client-server application such as DNS (see Chapter 19) uses the services of UDP because a client needs to send a short request to a server and to receive a quick response from it. The request and response can each fit in one user datagram. Since only one message is exchanged in each direction, the connectionless feature is not an issue; the client or server does not worry that messages are delivered out of order.
A client-server application such as SMTP (see Chapter 23), which is used in electronic mail, cannot use the services of UDP because a user can send a long e-mail message, which may include multimedia (images, audio, or video). If the application uses UDP and the message does not fit in one single user datagram, the message must be split by the application into different user datagrams. Here the connectionless service may create problems. The user datagrams may arrive and be delivered to the receiver application out of order. The receiver application may not be able to reorder the pieces. This means the connectionless service has a disadvantage for an application program that sends long messages.
Example 14.6

Assume we are downloading a very large text file from the Internet. We definitely need to use a transport layer that provides reliable service. We don’t want part of the file to be missing or corrupted when we open the file. The delay created between the delivery of the parts are not an overriding concern for us; we wait until the whole file is composed before looking at it. In this case, UDP is not a suitable transport layer.
Example 14.7

Assume we are watching a real-time stream video on our computer. Such a program is considered a long file; it is divided into many small parts and broadcast in real time. The parts of the message are sent one after another. If the transport layer is supposed to resend a corrupted or lost frame, the synchronizing of the whole transmission may be lost. The viewer suddenly sees a blank screen and needs to wait until the second transmission arrives. This is not tolerable. However, if each small part of the screen is sent using one single user datagram, the receiving UDP can easily ignore the corrupted or lost packet and deliver the rest to the application program. That part of the screen is blank for a very short period of the time, which most viewers do not even notice. However, video cannot be viewed out of order, so streaming audio, video, and voice applications that run over UDP must reorder or drop frames that are out of sequence.
Summaries:

- To introduce UDP and show its relationship to other protocols in the TCP/IP protocol suite.
- To explain the format of a UDP packet and discuss the use of each field in the header.
- To discuss the services provided by the UDP such as process-to-process delivery, multiplexing/demultiplexing, and queuing.
- To show how to calculate the optional checksum and the sender the needs to add a pseudoheader to the packet when calculating the checksum.
- To discuss how some application programs can benefit from the simplicity of UDP.
- To briefly discuss the structure of the UDP package.