Chapter 23

Process-to-Process Delivery: UDP, TCP, and SCTP
The transport layer is responsible for process-to-process delivery—the delivery of a packet, part of a message, from one process to another. Two processes communicate in a client/server relationship, as we will see later.

**Topics discussed in this section:**
- Client/Server Paradigm
- Multiplexing and Demultiplexing
- Connectionless Versus Connection-Oriented Service
- Reliable Versus Unreliable
- Three Protocols
The transport layer is responsible for process-to-process delivery.
Figure 23.1  *Types of data deliveries*

Node to node: Data link layer
Host to host: Network layer
Process to process: Transport layer
Figure 23.2 Port numbers
Figure 23.3  *IP addresses versus port numbers*
Figure 23.4  IANA ranges

Well known

Registered

Dynamic

0 1023 1024

49,152 49,151

65,535
Figure 23.5 *Socket address*

- IP address: 200.23.56.8
- Port number: 69
Figure 23.6  *Multiplexing and demultiplexing*
Figure 23.7 Error control

Error is checked in these paths by the data link layer
Error is not checked in these paths by the data link layer

Transport
Network
Data link
Physical

LAN
WAN
LAN
Figure 23.8  Position of UDP, TCP, and SCTP in TCP/IP suite
The User Datagram Protocol (UDP) is called a connectionless, unreliable transport protocol. It does not add anything to the services of IP except to provide process-to-process communication instead of host-to-host communication.

**Topics discussed in this section:**
- Well-Known Ports for UDP
- User Datagram
- Checksum
- UDP Operation
- Use of UDP
<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>Nameserver</td>
<td>Domain Name Service</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>
Example 23.1

In UNIX, the well-known ports are stored in a file called /etc/services. Each line in this file gives the name of the server and the well-known port number. We can use the grep utility to extract the line corresponding to the desired application. The following shows the port for FTP. Note that FTP can use port 21 with either UDP or TCP.

```
$ grep ftp /etc/services
ftp 21/tcp
ftp 21/udp
```
**Example 23.1 (continued)**

SNMP uses two port numbers (161 and 162), each for a different purpose, as we will see in Chapter 28.

```
$ grep snmp /etc/services
snmp  161/tcp       #Simple Net Mgmt Proto
snmp  161/udp       #Simple Net Mgmt Proto
snmptrap 162/udp    #Traps for SNMP
```
Figure 23.9 User datagram format
Note

UDP length
= IP length – IP header’s length
### Figure 23.10  Pseudoheader for checksum calculation

<table>
<thead>
<tr>
<th>Pseudoheader</th>
<th>32-bit source IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pseudoheader</td>
<td>32-bit destination IP address</td>
</tr>
<tr>
<td>All 0s</td>
<td>8-bit protocol (17)</td>
</tr>
<tr>
<td>Header</td>
<td>Source port address 16 bits</td>
</tr>
<tr>
<td></td>
<td>UDP total length 16 bits</td>
</tr>
<tr>
<td>Data</td>
<td>(Padding must be added to make the data a multiple of 16 bits)</td>
</tr>
</tbody>
</table>
Figure 23.11 shows the checksum calculation for a very small user datagram with only 7 bytes of data. Because the number of bytes of data is odd, padding is added for checksum calculation. The pseudoheader as well as the padding will be dropped when the user datagram is delivered to IP.
Figure 23.11  **Checksum calculation of a simple UDP user datagram**

<table>
<thead>
<tr>
<th>153.18.8.105</th>
<th>17</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>171.2.14.10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>All 0s</td>
<td>17</td>
<td>15</td>
</tr>
</tbody>
</table>

1087
15

<table>
<thead>
<tr>
<th>T</th>
<th>E</th>
<th>S</th>
<th>T</th>
<th>I</th>
<th>N</th>
<th>G</th>
<th>All 0s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

10011001 00010010 153.18
00001000 01101001  8.105
10101011 00000010 171.2
00001110 00001010 14.10
00000000 00010001  0 and 17
00000000 00001111  15
00001000 00111111 1087
00000000 00001101  13
00000000 00001111  15
00000000 00000000  0 (checksum)
01010100 01000101 T and E
01010011 01010100 S and T
01001001 01001110 I and N
01000111 00000000 G and 0 (padding)

10010110 11101011 Sum
01101001 00010100 Checksum
Figure 23.12 *Queues in UDP*

- **Daytime client**
  - Outgoing queue
  - Incoming queue
- **Daytime server**
  - Outgoing queue
  - Incoming queue

*UDP*}

Port 52000

Port 13
TCP is a connection-oriented protocol; it creates a virtual connection between two TCPs to send data. In addition, TCP uses flow and error control mechanisms at the transport level.

**Topics discussed in this section:**
- TCP Services
- TCP Features
- Segment
- A TCP Connection
- Flow Control
- Error Control
Table 23.2  **Well-known ports used by TCP**

<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>20</td>
<td>FTP, Data</td>
<td>File Transfer Protocol (data connection)</td>
</tr>
<tr>
<td>21</td>
<td>FTP, Control</td>
<td>File Transfer Protocol (control connection)</td>
</tr>
<tr>
<td>23</td>
<td>TELNET</td>
<td>Terminal Network</td>
</tr>
<tr>
<td>25</td>
<td>SMTP</td>
<td>Simple Mail Transfer Protocol</td>
</tr>
<tr>
<td>53</td>
<td>DNS</td>
<td>Domain Name Server</td>
</tr>
<tr>
<td>67</td>
<td>BOOTP</td>
<td>Bootstrap Protocol</td>
</tr>
<tr>
<td>79</td>
<td>Finger</td>
<td>Finger</td>
</tr>
<tr>
<td>80</td>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>111</td>
<td>RPC</td>
<td>Remote Procedure Call</td>
</tr>
</tbody>
</table>
Figure 23.13 *Stream delivery*
Figure 23.14 Sending and receiving buffers
The bytes of data being transferred in each connection are numbered by TCP. The numbering starts with a randomly generated number.
Example 23.3

The following shows the sequence number for each segment:

<table>
<thead>
<tr>
<th>Segment</th>
<th>Sequence Number:</th>
<th>Range:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10,001</td>
<td>10,001 to 11,000</td>
</tr>
<tr>
<td>2</td>
<td>11,001</td>
<td>11,001 to 12,000</td>
</tr>
<tr>
<td>3</td>
<td>12,001</td>
<td>12,001 to 13,000</td>
</tr>
<tr>
<td>4</td>
<td>13,001</td>
<td>13,001 to 14,000</td>
</tr>
<tr>
<td>5</td>
<td>14,001</td>
<td>14,001 to 15,000</td>
</tr>
</tbody>
</table>
Note

The value in the sequence number field of a segment defines the number of the first data byte contained in that segment.
The value of the acknowledgment field in a segment defines the number of the next byte a party expects to receive. The acknowledgment number is cumulative.
### Figure 23.16  TCP segment format

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source port address</td>
<td>16</td>
</tr>
<tr>
<td>Destination port address</td>
<td>16</td>
</tr>
<tr>
<td>Sequence number</td>
<td>32</td>
</tr>
<tr>
<td>Acknowledgment number</td>
<td>32</td>
</tr>
<tr>
<td>HLEN</td>
<td>4</td>
</tr>
<tr>
<td>Reserved</td>
<td>6</td>
</tr>
<tr>
<td>URG</td>
<td>1</td>
</tr>
<tr>
<td>ACK</td>
<td>1</td>
</tr>
<tr>
<td>PSH</td>
<td>1</td>
</tr>
<tr>
<td>RST</td>
<td>1</td>
</tr>
<tr>
<td>SYN</td>
<td>1</td>
</tr>
<tr>
<td>FIN</td>
<td>1</td>
</tr>
<tr>
<td>Window size</td>
<td>16</td>
</tr>
<tr>
<td>Checksum</td>
<td>16</td>
</tr>
<tr>
<td>Urgent pointer</td>
<td>16</td>
</tr>
<tr>
<td>Options and Padding</td>
<td></td>
</tr>
</tbody>
</table>
Figure 23.17  *Control field*

URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection
Table 23.3  Description of flags in the control field

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URG</td>
<td>The value of the urgent pointer field is valid.</td>
</tr>
<tr>
<td>ACK</td>
<td>The value of the acknowledgment field is valid.</td>
</tr>
<tr>
<td>PSH</td>
<td>Push the data.</td>
</tr>
<tr>
<td>RST</td>
<td>Reset the connection.</td>
</tr>
<tr>
<td>SYN</td>
<td>Synchronize sequence numbers during connection.</td>
</tr>
<tr>
<td>FIN</td>
<td>Terminate the connection.</td>
</tr>
</tbody>
</table>
Figure 23.18  *Connection establishment using three-way handshaking*
A SYN segment cannot carry data, but it consumes one sequence number.
A SYN + ACK segment cannot carry data, but does consume one sequence number.
An ACK segment, if carrying no data, consumes no sequence number.
Figure 23.19  Data transfer
Figure 23.20  *Connection termination using three-way handshaking*
The FIN segment consumes one sequence number if it does not carry data.
The FIN + ACK segment consumes one sequence number if it does not carry data.
Figure 23.21  *Half-close*
Figure 23.22  *Sliding window*

Window size = minimum (rwnd, cwnd)
A sliding window is used to make transmission more efficient as well as to control the flow of data so that the destination does not become overwhelmed with data. TCP sliding windows are byte-oriented.
What is the value of the receiver window (rwnd) for host A if the receiver, host B, has a buffer size of 5000 bytes and 1000 bytes of received and unprocessed data?

Solution

The value of \( rwnd = 5000 - 1000 = 4000 \). Host B can receive only 4000 bytes of data before overflowing its buffer. Host B advertises this value in its next segment to A.
Example 23.5

What is the size of the window for host A if the value of rwnd is 3000 bytes and the value of cwnd is 3500 bytes?

Solution
The size of the window is the smaller of rwnd and cwnd, which is 3000 bytes.
Example 23.6

Figure 23.23 shows an unrealistic example of a sliding window. The sender has sent bytes up to 202. We assume that cwnd is 20 (in reality this value is thousands of bytes). The receiver has sent an acknowledgment number of 200 with an rwnd of 9 bytes (in reality this value is thousands of bytes). The size of the sender window is the minimum of rwnd and cwnd, or 9 bytes. Bytes 200 to 202 are sent, but not acknowledged. Bytes 203 to 208 can be sent without worrying about acknowledgment. Bytes 209 and above cannot be sent.
Figure 23.23  Example 23.6

Window size = minimum \( (20, 9) = 9 \)

- Sent, not acknowledged
- Can be sent immediately

- Sent and acknowledged
- Next byte to be sent
- Can't be sent until window opens
Some points about TCP sliding windows:

- The size of the window is the lesser of rwnd and cwnd.
- The source does not have to send a full window’s worth of data.
- The window can be opened or closed by the receiver, but should not be shrunk.
- The destination can send an acknowledgment at any time as long as it does not result in a shrinking window.
- The receiver can temporarily shut down the window; the sender, however, can always send a segment of 1 byte after the window is shut down.
Note

ACK segments do not consume sequence numbers and are not acknowledged.

23.50
In modern implementations, a retransmission occurs if the retransmission timer expires or three duplicate ACK segments have arrived.
Note

No retransmission timer is set for an ACK segment.
Note

Data may arrive out of order and be temporarily stored by the receiving TCP, but TCP guarantees that no out-of-order segment is delivered to the process.
Figure 23.24  Normal operation
Figure 23.25  *Lost segment*
The receiver TCP delivers only ordered data to the process.
Figure 23.26  *Fast retransmission*

Diagram showing fast retransmission in a network flow. The diagram illustrates the sequence of packets and acknowledgments between a sender and a receiver, highlighting how lost packets are retransmitted quickly.
Stream Control Transmission Protocol (SCTP) is a new reliable, message-oriented transport layer protocol. SCTP, however, is mostly designed for Internet applications that have recently been introduced. These new applications need a more sophisticated service than TCP can provide.

**Topics discussed in this section:**

- SCTP Services and Features
- Packet Format
- An SCTP Association
- Flow Control and Error Control
SCTP is a message-oriented, reliable protocol that combines the best features of UDP and TCP.
Table 23.4  Some SCTP applications

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IUA</td>
<td>9990</td>
<td>ISDN over IP</td>
</tr>
<tr>
<td>M2UA</td>
<td>2904</td>
<td>SS7 telephony signaling</td>
</tr>
<tr>
<td>M3UA</td>
<td>2905</td>
<td>SS7 telephony signaling</td>
</tr>
<tr>
<td>H.248</td>
<td>2945</td>
<td>Media gateway control</td>
</tr>
<tr>
<td>H.323</td>
<td>1718, 1719, 1720, 11720</td>
<td>IP telephony</td>
</tr>
<tr>
<td>SIP</td>
<td>5060</td>
<td>IP telephony</td>
</tr>
</tbody>
</table>
Figure 23.27 Multiple-stream concept
An association in SCTP can involve multiple streams.
Figure 23.28 *Multihoming concept*
SCTP association allows multiple IP addresses for each end.
In SCTP, a data chunk is numbered using a TSN.
To distinguish between different streams, SCTP uses an SI.
To distinguish between different data chunks belonging to the same stream, SCTP uses SSNs.
Note

TCP has segments; SCTP has packets.
Figure 23.29 Comparison between a TCP segment and an SCTP packet
In SCTP, control information and data information are carried in separate chunks.
Figure 23.30  Packet, data chunks, and streams

Flow of packets from sender to receiver
Data chunks are identified by three items: TSN, SI, and SSN. TSN is a cumulative number identifying the association; SI defines the stream; SSN defines the chunk in a stream.
Note

In SCTP, acknowledgment numbers are used to acknowledge only data chunks; control chunks are acknowledged by other control chunks if necessary.
### Figure 23.31  *SCTP packet format*

<table>
<thead>
<tr>
<th>General header</th>
<th>(12 bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chunk 1</td>
<td>(variable length)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Chunk N</td>
<td>(variable length)</td>
</tr>
</tbody>
</table>
Note

In an SCTP packet, control chunks come before data chunks.
Figure 23.32  *General header*

<table>
<thead>
<tr>
<th>Source port address</th>
<th>Destination port address</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 bits</td>
<td>16 bits</td>
</tr>
<tr>
<td>Verification tag</td>
<td>32 bits</td>
</tr>
<tr>
<td>Checksum</td>
<td>32 bits</td>
</tr>
<tr>
<td>Type</td>
<td>Chunk</td>
</tr>
<tr>
<td>------</td>
<td>----------------------</td>
</tr>
<tr>
<td>0</td>
<td>DATA</td>
</tr>
<tr>
<td>1</td>
<td>INIT</td>
</tr>
<tr>
<td>2</td>
<td>INIT ACK</td>
</tr>
<tr>
<td>3</td>
<td>SACK</td>
</tr>
<tr>
<td>4</td>
<td>HEARTBEAT</td>
</tr>
<tr>
<td>5</td>
<td>HEARTBEAT ACK</td>
</tr>
<tr>
<td>6</td>
<td>ABORT</td>
</tr>
<tr>
<td>7</td>
<td>SHUTDOWN</td>
</tr>
<tr>
<td>8</td>
<td>SHUTDOWN ACK</td>
</tr>
<tr>
<td>9</td>
<td>ERROR</td>
</tr>
<tr>
<td>10</td>
<td>COOKIE ECHO</td>
</tr>
<tr>
<td>11</td>
<td>COOKIE ACK</td>
</tr>
<tr>
<td>14</td>
<td>SHUTDOWN COMPLETE</td>
</tr>
<tr>
<td>192</td>
<td>FORWARD TSN</td>
</tr>
</tbody>
</table>
A connection in SCTP is called an association.
No other chunk is allowed in a packet carrying an INIT or INIT ACK chunk. A COOKIE ECHO or a COOKIE ACK chunk can carry data chunks.
Figure 23.33 *Four-way handshaking*
In SCTP, only DATA chunks consume TSNs; DATA chunks are the only chunks that are acknowledged.
Figure 23.34 Simple data transfer
The acknowledgment in SCTP defines the cumulative TSN, the TSN of the last data chunk received in order.
Figure 23.35  Association termination
Figure 23.36  Flow control, receiver site

Received

Receiving queue

winSize

To process

26  25  24  23  22

26
1000
winSize
20
lastACK

cumTSN
Figure 23.37  Flow control, sender site
Figure 23.38  Flow control scenario

Sender

4 3 2 1
curTSN: rwnd: inTransit
1 2000 0

DATA
TSN: 1
ISN: 1
1000 bytes

4 3 2 1
curTSN: rwnd: inTransit
2 2000 1000

DATA
TSN: 2
ISN: 2
1000 bytes

4 3 2 1
curTSN: rwnd: inTransit
3 2000 2000

ACK: 2
rwnd: 0

6 5 4 3
curTSN: rwnd: inTransit
3 0 0

ACK: 2
rwnd: 2000

Process writes 5 and 6

Receiver

cumTSN
winSize
lastACK
2000

cumTSN
winSize
lastACK
1 1000

cumTSN
winSize
lastACK
2 0

cumTSN
winSize
lastACK
2 2000

Process reads 1 and 2
Figure 23.39  Error control, receiver site
Figure 23.40  

Error control, sender site

From process

Sending queue

Outstanding chunks

To send

Add when timer expires or three SACKs are received.

Retransmission queue

curTSN
2000
1400
inTransit

22 21
To send