User Datagram Protocol

User Datagram Protocol

Topics Covered:

1. Introduction
2. User Datagram
3. Checksum
4. UDP Operation
    1. Connectionless Services
    2. Flow and Error Control
    3. Encapsulation and Decapsulation
    4. Queuing
        1. Queues at Client Side.
        2. Queues at Client Side.
5. Use of UDP

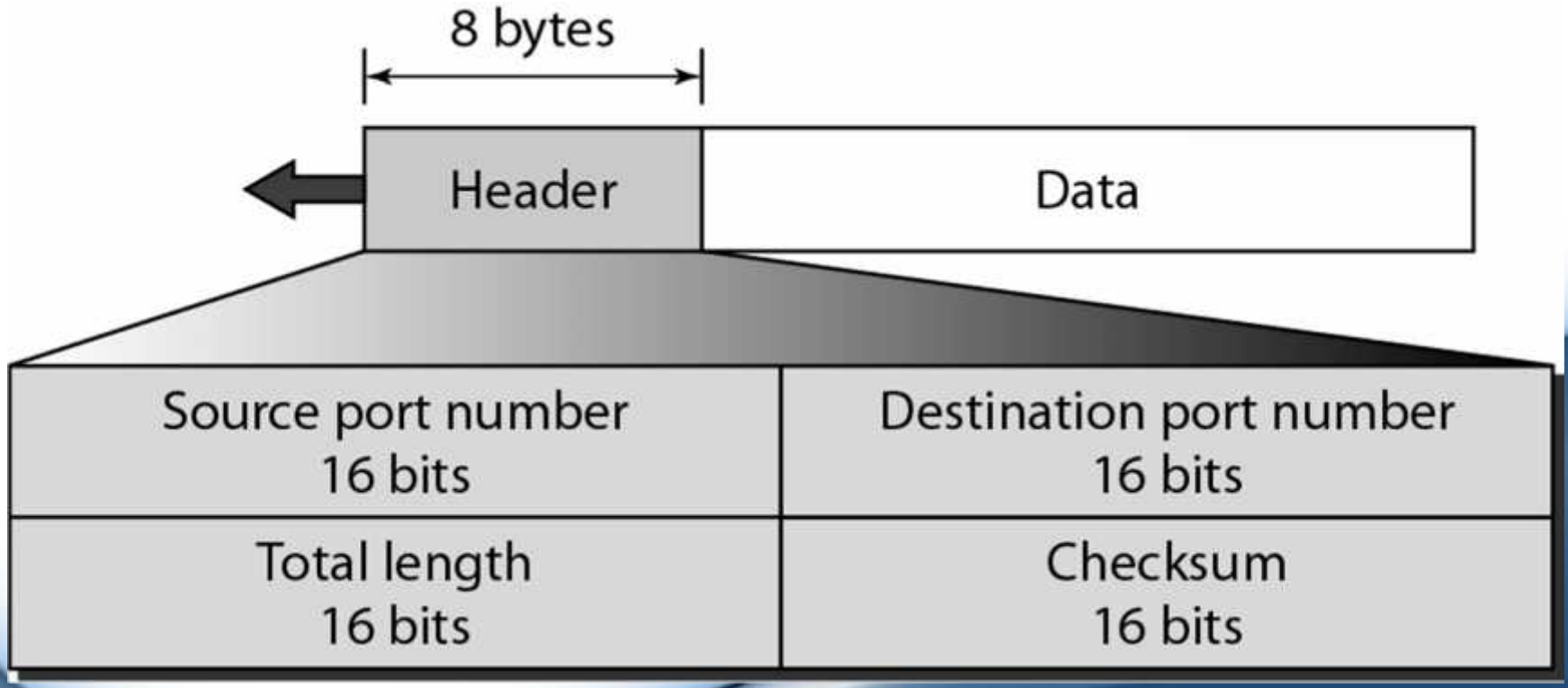
USER DATAGRAM PROTOCOL (UDP):

- 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.
- Also, it performs very limited error checking. If UDP is so powerless, why would a process want to use it?
- With the disadvantages come some advantages. UDP is a very simple protocol using a minimum of overhead.
- If a process wants to send a small message and does not care much about reliability, it can use UDP. Sending a small message by using UDP takes much less interaction between the sender and receiver than using TCP or SCTP.

Well-Known Ports for UDP

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Nameserver	Domain Name Service
67	BOOTPs	Server port to download bootstrap information
68	BOOTPc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

User Datagram



Fields of UDP

- Source port number:
  - This is the port number used by the process running on the source host. It is 16 bits long, which means that the port number can range from 0 to 65,535. If the source host is the client (a client sending a request), the port number, in most cases, is an ephemeral port number requested by the process and chosen by the UDP software running on the source host. If the source host is the server (a server sending a response), the port number, in most cases, is a well-known port number.
- Destination port number:
  - This is the port number used by the process running on the destination host. It is also 16 bits long. If the destination host is the server (a client sending a request), the port number, in most cases, is a well-known port number. If the destination host is the client (a server sending a response), the port number, in most cases, is an ephemeral port number. In this case, the server copies the ephemeral port number it has received in the request packet.

Fields of UDP

- Length.
  - This is a 16-bit field that defines the total length of the user datagram, header plus data. The 16 bits can define a total length of 0 to 65,535 bytes. However, the total length needs to be much less because a UDP user datagram is stored in an IP datagram with a total length of 65,535 bytes. The length field in a UDP user datagram is actually not necessary. A user datagram is encapsulated in an IP datagram. There is a field in the IP datagram that defines the total length. There is another field in the IP datagram that defines the length of the header. So if we subtract the value of the second field from the first, we can deduce the length of a UDP datagram that is encapsulated in an IP datagram.

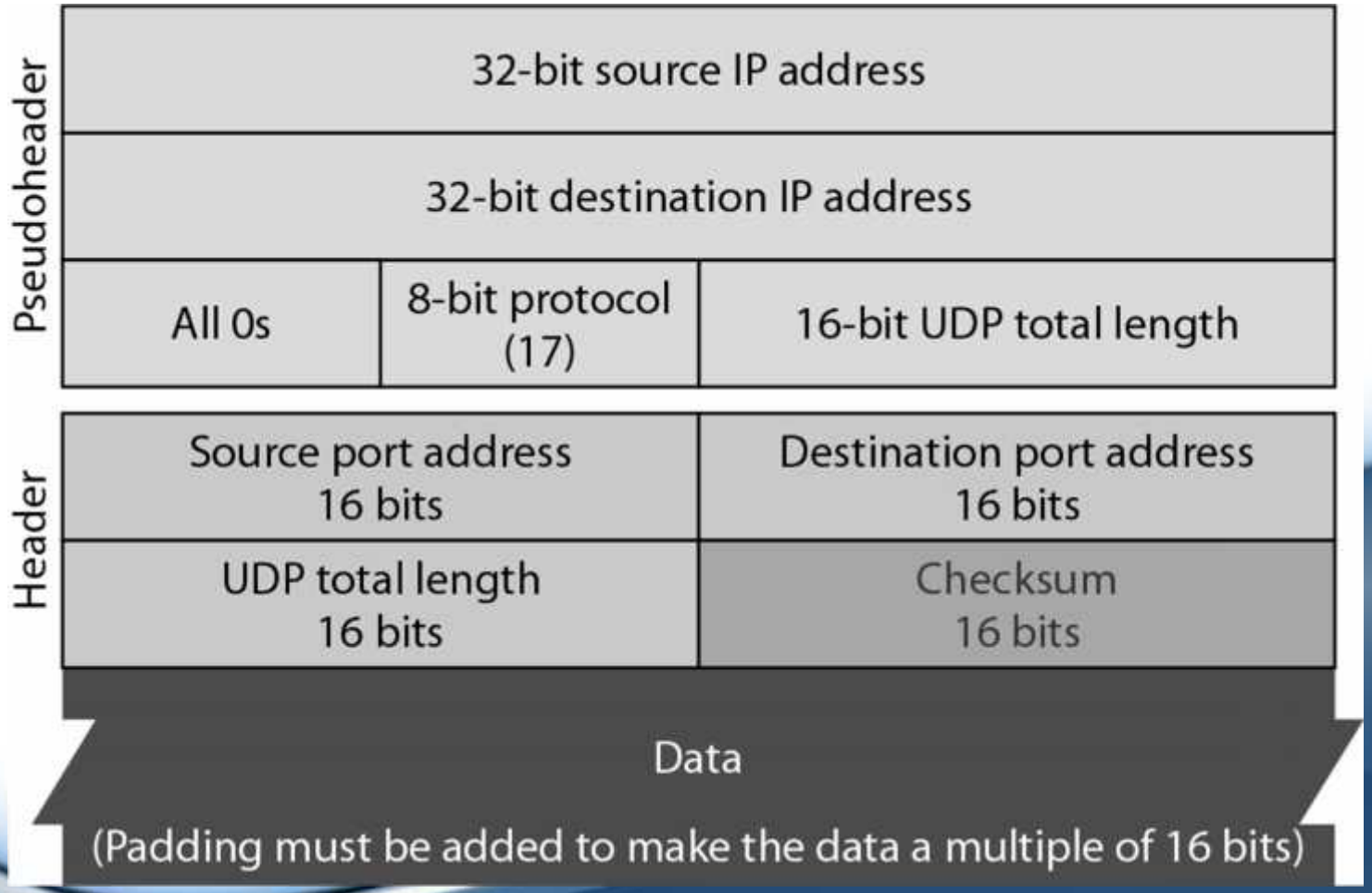
    UDP length = IP length - IP header's length

  - However, the designers of the UDP protocol felt that it was more efficient for the destination UDP to calculate the length of the data from the information provided in the UDP user datagram rather than ask the IP software to supply this information. We should remember that when the IP software delivers the UDP user datagram to the UDP layer, it has already dropped the IP header.
- Checksum:
  - This field is used to detect errors over the entire user datagram (header plus data).

Checksum

- The UDP checksum calculation is different from the one for IP and ICMP. Here the checksum includes three sections: a pseudoheader, the UDP header, and the data coming from the application layer. The pseudoheader is the part of the header of the IP packet in which the user datagram is to be encapsulated with some fields filled with Os
- If the checksum does not include the pseudoheader, a user datagram may arrive safe and sound. However, if the IP header is corrupted, it may be delivered to the wrong host. The protocol field is added to ensure that the packet belongs to UDP, and not to other transport-layer protocols.
- We will see later that if a process can use either UDP or TCP, the destination port number can be the same. The value of the protocol field for UDP is 17. If this value is changed during transmission, the checksum calculation at the receiver will detect it and UDP drops the packet. It is not delivered to the wrong protocol.

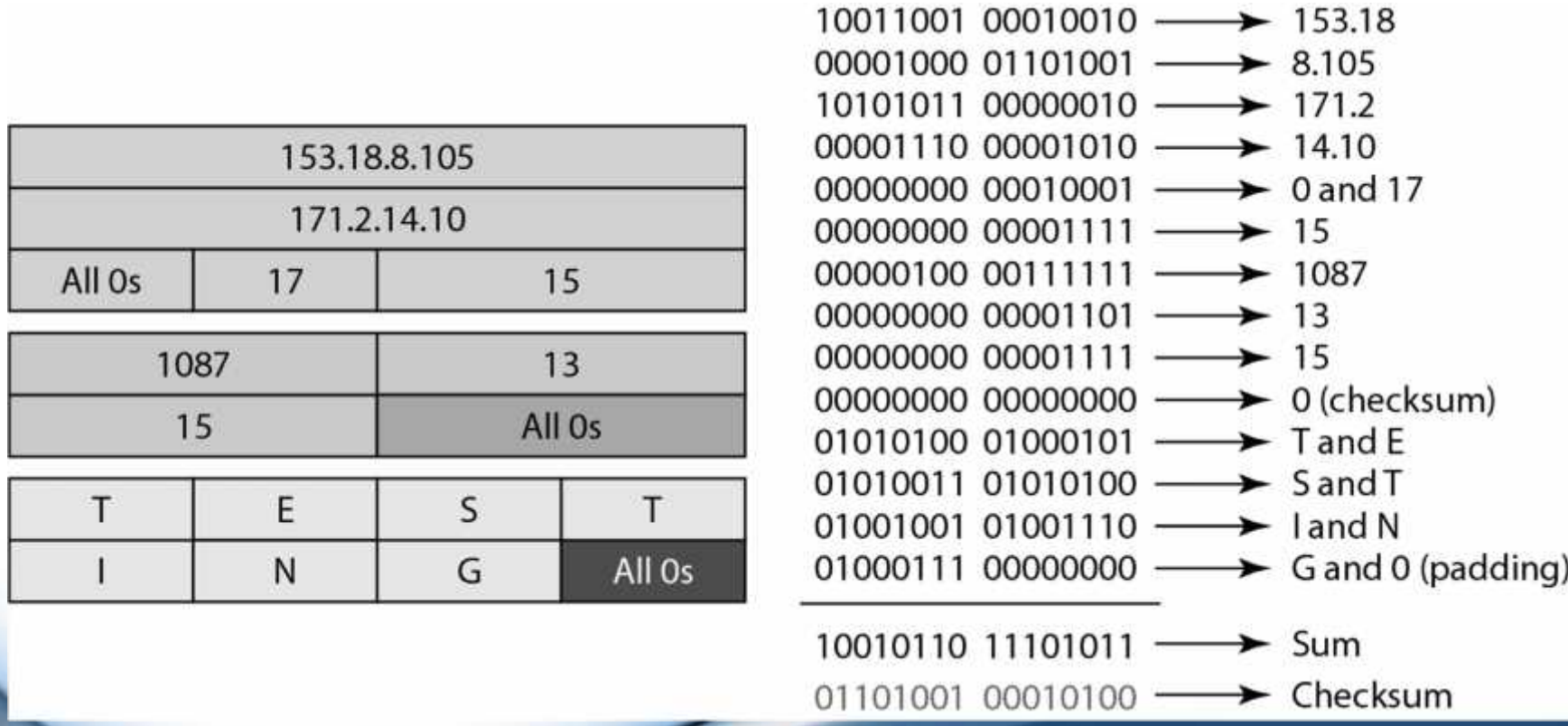
CheckSum


Pseudoheader
32-bit source IP address
32-bit destination IP address
All 0s
8-bit protocol (17)
16-bit UDP total length
Header
Source port address 16 bits
Destination port address 16 bits
UDP total length 16 bits
Checksum 16 bits
Data
(Padding must be added to make the data a multiple of 16 bits)

Optional Use of the Checksum

- The calculation of the checksum and its inclusion in a user datagram are optional.
- If the checksum is not calculated, the field is filled with 1s.
- Note that a calculated checksum can never be all 1s because this implies that the sum is all Os, which is impossible because it requires that the value of fields to be Os.

Optional Use of CheckSum



UDP Operation

- Connectionless Services
- Flow and Error Control
- Encapsulation and Decapsulation
- Queuing

Connectionless Services

- As mentioned previously, UDP provides a connectionless service. This means that each user datagram sent by UDP is an independent datagram.
- There is no relationship between the different user datagrams even if they are coming from the same source process and going to the same destination program.
- The user datagrams are not numbered.
- Also, there is no connection establishment and no connection termination, as is the case for TCP. This means that each user datagram can travel on a different path.
- One of the ramifications of being connectionless is that the process that uses UDP cannot send a stream of data to UDP and expect UDP to chop them into different related user datagrams.
- Instead each request must be small enough to fit into one user datagram. Only those processes sending short messages should use UDP.

Flow and Error Control

- UDP is a very simple, unreliable transport protocol.
- There is no flow control and hence no window mechanism.
- The receiver may overflow with incoming messages.
- There is no error control mechanism in UDP except for the checksum.
  - This means that 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.
- The lack of flow control and error control means that the process using UDP should provide these mechanisms.

Encapsulation and Decapsulation

- To send a message from one process to another, the UDP protocol encapsulates and decapsulates messages in an IP datagram.

Queuing

- Queues at Client Side.
- Queues at Client Side.

Queues at Client Side.

- In UDP, queues are associated with ports At the client site, when a process starts, it requests a port number from the operating system. Some implementations create both an incoming and an outgoing queue associated with each process. Other implementations create only an incoming queue associated with each process.
- Note that even if a process wants to communicate with multiple processes, it obtains only one port number and eventually one outgoing and one incoming queue. The queues opened by the client are, in most cases, identified by ephemeral port numbers. The queues function as long as the process is running. When the process terminates, the queues are destroyed.
- The client process can send messages to the outgoing queue by using the source port number specified in the request. UDP removes the messages one by one and, after adding the UDP header, delivers them to IP. An outgoing queue can overflow. If this happens, the operating system can ask the client process to wait before sending any more messages.

Queues at Client Side.

- When a message arrives for a client, UDP checks to see if an incoming queue has been created for the port number specified in the destination port number field of the user datagram. If there is such a queue, UDP sends the received user datagram to the end of the queue. If there is no such queue, UDP discards the user datagram and asks the ICMP protocol to send a port unreachable message to the server. All the incoming messages for one particular client program, whether coming from the same or a different server, are sent to the same queue. An incoming queue can overflow. If this happens, UDP drops the user datagram and asks for a port unreachable message to be sent to the server.

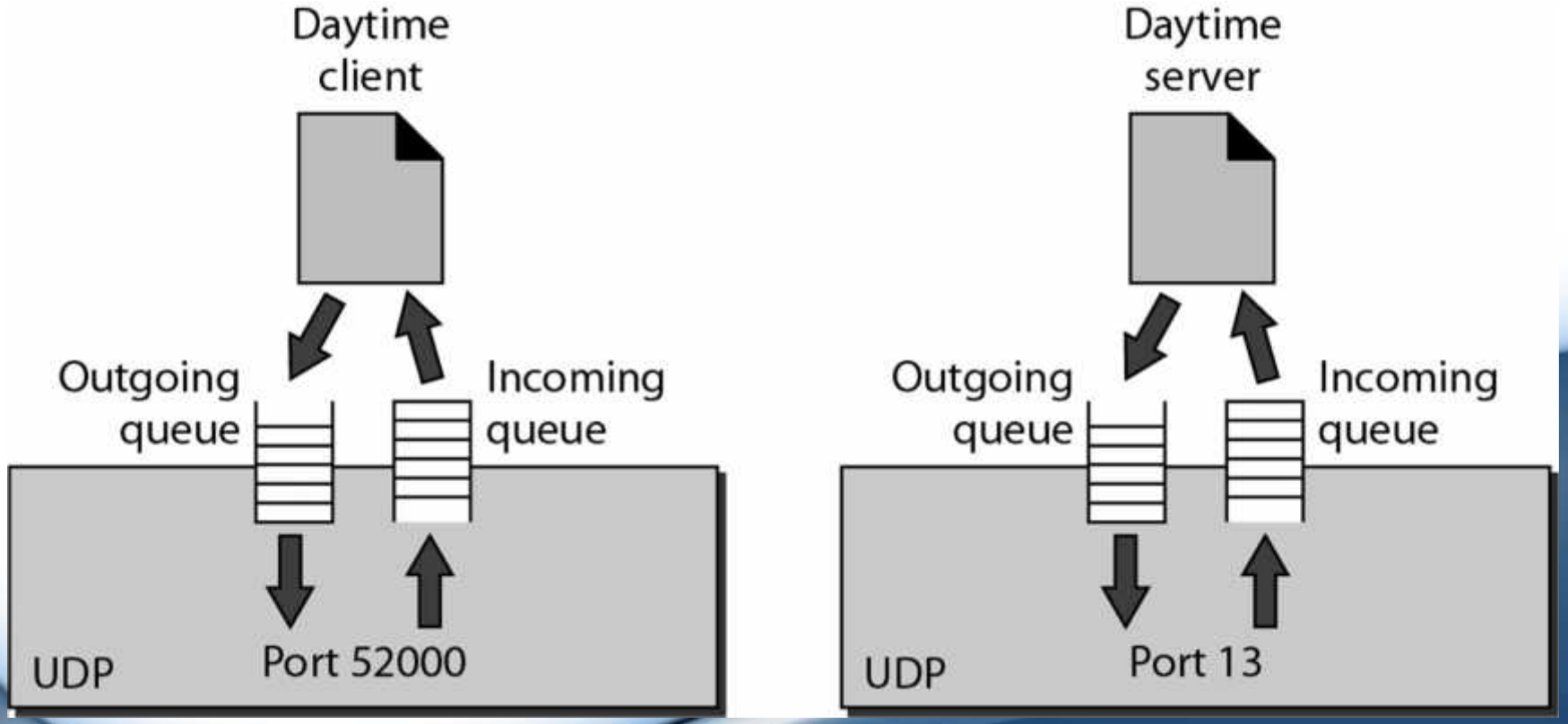
Queues at Server Side.

- At the server site, the mechanism of creating queues is different. In its simplest form, a server asks for incoming and outgoing queues, using its well-known port, when it starts running. The queues remain open as long as the server is running.
- When a message arrives for a server, UDP checks to see if an incoming queue has been created for the port number specified in the destination port number field of the user datagram. If there is such a queue, UDP sends the received user datagram to the end of the queue. If there is no such queue, UDP discards the user datagram and asks the ICMP protocol to send a port unreachable message to the client. All the incoming messages for one particular server, whether coming from the same or a different client, are sent to the same queue.

Queues at Server Side.

- An incoming queue can overflow. If this happens, UDP drops the user datagram and asks for a port unreachable message to be sent to the client. When a server wants to respond to a client, it sends messages to the outgoing queue, using the source port number specified in the request. UDP removes the messages one by one and, after adding the UDP header, delivers them to IP. An outgoing queue can overflow. If this happens, the operating system asks the server to wait before sending any more messages.

Queuing


Daytime client
Outgoing queue
Incoming queue
UDP
Port 52000
Daytime server
Outgoing queue
Incoming queue
UDP
Port 13

Use of UDP

- UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control. It is not usually used for a process such as FTP that needs to send bulk data.
- UDP is suitable for a process with internal flow and error control mechanisms. For example, the Trivial File Transfer Protocol (TFTP) process includes flow and error control. It can easily use UDP.
- UDP is a suitable transport protocol for multicasting. Multicasting capability is embedded in the UDP software but not in the TCP software.
- UDP is used for management processes such as SNMP.
- UDP is used for some route updating protocols such as Routing Information Protocol (RIP).

Thank You