User Datagram Protocol

From Wikipedia, the free encyclopedia

User Datagram Protocol (**UDP**) is one of the core protocols of the Internet protocol suite. Using UDP, programs on networked computers can send short messages sometimes known as *datagrams* (using Datagram Sockets) to one another. UDP is sometimes called the **Universal Datagram Protocol**.

UDP does not guarantee reliability or ordering in the way that TCP does. Datagrams may arrive out of order, appear duplicated, or go missing without notice. Avoiding the overhead of checking whether every packet actually arrived makes UDP faster and more efficient, at least for applications that do not need guaranteed delivery. Time-sensitive applications often use UDP because dropped packets are preferable to delayed packets. UDP's stateless nature is also useful for servers that answer small queries from huge numbers of clients. Unlike TCP, UDP supports packet broadcast (sending to all on local network) and multicasting (send to all subscribers).

Common network applications that use UDP include the Domain Name System (DNS), streaming media applications such as IPTV, Voice over IP (VoIP), Trivial File Transfer Protocol (TFTP) and online games.

The five-layer TCP/IP model

5. Application layer

DHCP · DNS · FTP · Gopher · HTTP · IMAP4 · IRC · NNTP · XMPP · POP3 · SIP · SMTP · SNMP · SSH · TELNET · RPC · RTCP · RTSP · TLS · SDP · SOAP · GTP · STUN · NTP · (more)

4. Transport layer

TCP · **UDP** · DCCP · SCTP · RTP · RSVP · IGMP · (more)

3. Network/Internet layer

IP (IPv4 · IPv6) · OSPF · IS-IS · BGP · IPsec · ARP · RARP · RIP · ICMP · ICMPv6 · (more)

2. Data link layer

802.11 · 802.16 · Wi-Fi · WiMAX · ATM · DTM · Token ring · Ethernet · FDDI · Frame Relay · GPRS · EVDO · HSPA · HDLC · PPP · PPTP · L2TP · ISDN · (more)

1. Physical layer

Ethernet physical layer · Modems · PLC · SONET/SDH · G.709 · Optical fiber · Coaxial cable · Twisted pair · (more)

Contents

Ports

UDP uses ports to allow application-to-application communication. The port field is a 16 bit value, allowing for port numbers to range between 0 and 65,535. Port 0 is reserved, but is a permissible source port value if the sending process does not expect messages in response.

Ports 1 through 1023 are named "well-known" ports and on Unix-derived operating systems, binding to one of these ports requires root access.

Ports 1024 through 49,151 are registered ports.

Ports 49,152 through 65,535 are ephemeral ports and are used as temporary ports primarily by clients when communicating to servers.

Packet structure

UDP is a minimal message-oriented transport layer protocol that is currently documented in IETF RFC 768.

In the Internet protocol suite, UDP provides a very simple interface between a network layer below (e.g., IPv4) and a session layer or application layer above.

UDP provides no guarantees to the upper layer protocol for message delivery and a UDP sender retains no state on UDP messages once sent (for this reason UDP is sometimes called the *Unreliable* Datagram Protocol). UDP adds only application multiplexing and checksumming of the header and payload. If any kind of reliability for the information transmitted is needed, it must be implemented in upper layers.

+	**Bits 0 - 15**	**16 - 31**
0	Source Port	Destination Port
32	Length	Checksum
64	Data	

The UDP header consists of only 4 fields. The use of two of those is optional (pink background in table).

Source port
: This field identifies the sending port when meaningful and should be assumed to be the port to reply to if needed. If not used, then it should be zero.

Destination port
: This field identifies the destination port and is required.

Length
: A 16-bit field that specifies the length in bytes of the entire datagram: header and data. The minimum length is 8 bytes since that's the length of the header. The field size sets a theoretical limit of 65,527 bytes for the data carried by a single UDP datagram. The practical limit for the data length which is imposed by the underlying IPv4 protocol is 65,507 bytes.

Checksum
: The 16-bit checksum field is used for error-checking of the header *and data*.

With IPv4

When UDP runs over IPv4, the method used to compute the checksum is defined within RFC 768:

Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data,

> *padded with zero octets at the end (if necessary) to make a multiple of two octets.*

In other words, all 16-bit words are summed together using one's complement (with the checksum field set to zero). The sum is then one's complemented. This final value is then inserted as the checksum field. Algorithmically speaking, this is the same as for IPv6.

The difference is in the data used to make the checksum. Included is a pseudo-header that mimics the IPv4 header:

+	**Bits 0 - 7**	**8 - 15**	**16 - 23**	**24 - 31**
0	Source address			
32	Destination address			
64	Zeros	Protocol	UDP length	
96	Source Port		Destination Port	
128	Length		Checksum	
160	Data			

The source and destination addresses are those in the IPv4 header. The protocol is that for UDP (*see List of IPv4 protocol numbers*): 17. The UDP length field is the length of the UDP header and data.

If the checksum is calculated to be zero (all 0's) it should be sent as negative zero (all 1's). If a checksum is not used it should be sent as zero (all 0's) as zero indicates an unused checksum.

With IPv6

When UDP runs over IPv6, the checksum is no longer considered optional, and the method used to compute the checksum is changed, as per RFC 2460:

> *Any transport or other upper-layer protocol that includes the addresses from the IP header in its checksum computation must be modified for use over IPv6, to include the 128-bit IPv6 addresses instead of 32-bit IPv4 addresses.*

When computing the checksum, a pseudo-header that mimics the IPv6 header is included:

+	**Bits 0 - 7**	**8 - 15**	**16 - 23**	**24 - 31**
0	Source address			
32				
64				
96				
128	Destination address			
160				
192				
224				

256	UDP length		
288	Zeros		Next Header
320	Source Port	Destination Port	
352	Length	Checksum	
384	Data		

The source address is the one in the IPv6 header. The destination address is the final destination; if the IPv6 packet doesn't contain a Routing header, that will be the destination address in the IPv6 header; otherwise, at the originating node, it will be the address in the last element of the Routing header, and, at the receiving node, it will be the destination address in the IPv6 header. The Next Header value is the protocol value for UDP: 17. The UDP length field is the length of the UDP header and data.

If the checksum is calculated to be zero (all 0's) it should be sent as negative zero (all 1's).

Lacking reliability, UDP applications must generally be willing to accept some loss, errors or duplication. Some applications such as TFTP may add rudimentary reliability mechanisms into the application layer as needed. Most often, UDP applications do not require reliability mechanisms and may even be hindered by them. Streaming media, real-time multiplayer games and voice over IP (VoIP) are examples of applications that often use UDP. If an application requires a high degree of reliability, a protocol such as the Transmission Control Protocol or erasure codes may be used instead.

Lacking any congestion avoidance and control mechanisms, network-based mechanisms are required to minimize potential congestion collapse effects of uncontrolled, high rate UDP traffic loads. In other words, since UDP senders cannot detect congestion, network-based elements such as routers using packet queuing and dropping techniques will often be the only tool available to slow down excessive UDP traffic. The Datagram Congestion Control Protocol (DCCP) is being designed as a partial solution to this potential problem by adding end host TCP-friendly congestion control behavior to high-rate UDP streams such as streaming media.

While the total amount of UDP traffic found on a typical network is often in the order of only a few percent, numerous key applications use UDP, including the Domain Name System (DNS), the simple network management protocol (SNMP), the Dynamic Host Configuration Protocol (DHCP) and the Routing Information Protocol (RIP), to name just a few.

Sample code (Python)

The following, minimalistic example shows how to use UDP for client/server communication:

The server:

```
import socket

PORT = 10000
BUFLEN = 512

server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
server.bind(('', PORT))

while True:
        (message, address) = server.recvfrom(BUFLEN)
        print 'Received packet from %s:%d' % (address[0], address[1])
        print 'Data: %s' % message
```

The client (replace "127.0.0.1" by the IP address of the server):

```
import socket

SERVER_ADDRESS = '127.0.0.1'
SERVER_PORT = 10000

client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)

for i in range(3):
        print 'Sending packet %d' % i
        message = 'This is packet %d' % i
        client.sendto(message, (SERVER_ADDRESS, SERVER_PORT))

client.close()
```

Voice and Video Traffic

UDP is generally the protocol used in transmitting voice and video across a network. This is because there is no time to re-send lost packets when listening to someone or watching a video in real time. Because both TCP and UDP run over the same network, many businesses are finding that the increase in UDP traffic (VoIP and Video) is hurting the performance of their TCP applications, which could be their order entry system, accounting system, etc. By default TCP will rev down to let the real-time data use most of the bandwidth. The problem is that both are important for most businesses, so finding the right balance is crucial.[1]

Difference between TCP and UDP

TCP is a connection-oriented protocol; a connection can be made from client to server, and from then on any data can be sent along that connection.

- **Reliable** - TCP manages message acknowledgment, retransmission and timeout. Many attempts to reliably deliver the message are made. If it gets lost along the way, the server will re-request the lost part. In TCP, there's either no missing data, or, in case of multiple timeouts, the connection is dropped.
- **Ordered** - if two messages are sent along a connection, one after the other, the first message will reach the receiving application first. When data packets arrive in the wrong order, the TCP layer holds the later data until the earlier data can be rearranged and delivered to the application.
- **Heavyweight** - TCP requires three packets just to set up a socket, before any actual data can be sent. It handles connections, reliability and congestion control. It is a large transport protocol designed on top of IP.
- **Streaming** - Data is read as a "stream," with nothing distinguishing where one packet ends and another begins. Packets may be split or merged into bigger or smaller data streams arbitrarily.

UDP is a simpler message-based connectionless protocol. With UDP messages (packets) cross the network in independent units.

- **Unreliable** - When a message is sent, it can't be known if it will reach its destination; it could get lost along the way. There's no concept of acknowledgment, retransmission and timeout.
- **Not ordered** - If two messages are sent to the same recipient, the order in which they arrive cannot be predicted.
- **Lightweight** - There is no ordering of messages, no tracking connections, etc. It's a small transport layer designed on top of IP.
- **Datagrams** - Packets are sent individually and are guaranteed to be whole if they arrive. Packets have definite bounds and no split or merge into data streams may exist.

Notes

1. ^ The impact of UDP on Data Applications

See also

- TCP and UDP port numbers for a partial (growing) listing of ports/services
- Connectionless protocol
- UDP flood attack
- UDP Data Transport
- UDP Lite, a variant that will deliver packets even if they are malformed
- Reliable User Datagram Protocol (RUDP)
- Transmission Control Protocol
- IP or Internet Protocol, on top of which rests UDP
- Transport protocol comparison table

External links

- RFC 768
- IANA Port Assignments
- The Trouble with UDP Scanning (PDF)
- Breakdown of UDP frame
- UDP on MSDN Magazine Sockets and WCF

Retrieved from "http://en.wikipedia.org/wiki/User_Datagram_Protocol"

Categories: Internet protocols | Internet standards | Transport layer protocols