User Datagram Protocol

From Wikipedia, the free encyclopedia

The **User Datagram Protocol** (**UDP**) is one of the core members of the Internet protocol suite (the set of network protocols used for the Internet). With UDP, computer applications can send messages, in this case referred to as *datagrams*, to other hosts on an Internet Protocol (IP) network without prior communications to set up special transmission channels or data paths. The protocol was designed by David P. Reed in 1980 and formally defined in RFC 768.

UDP uses a simple transmission model with a minimum of protocol mechanism.[1] It has no handshaking dialogues, and thus exposes any unreliability of the underlying network protocol to the user's program. As this is normally IP over unreliable media, there is no guarantee of delivery, ordering, or duplicate protection. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram.

UDP is suitable for purposes where error checking and correction is either not necessary or is performed in the application, avoiding the overhead of such processing at the network interface level. Time-sensitive applications often use UDP because dropping packets is preferable to waiting for delayed packets, which may not be an option in a real-time system.[2] If error correction facilities are needed at the network interface level, an application may use the Transmission Control Protocol (TCP) or Stream Control Transmission Protocol (SCTP) which are designed for this purpose.

A number of UDP's attributes make it especially suited for certain applications.

- It is *transaction-oriented*, suitable for simple query-response protocols such as the Domain Name System or the Network Time Protocol.
- It provides *datagrams*, suitable for modeling other protocols such as in IP tunneling or Remote Procedure Call and the Network File System.
- It is *simple*, suitable for bootstrapping or other purposes without a full protocol stack, such as the DHCP and Trivial File Transfer Protocol.
- It is *stateless*, suitable for very large numbers of clients, such as in streaming media applications for example IPTV
- The *lack of retransmission delays* makes it suitable for real-time applications such as Voice over IP, online games, and many protocols built on top of the Real Time Streaming Protocol.
- Works well in *unidirectional* communication, suitable for broadcast information such as in many kinds of service discovery and shared information such as broadcast time or Routing Information Protocol

Contents

Service ports

Applications use datagram sockets to establish host-to-host communications. An application binds a socket to its endpoint of data transmission, which is a combination of an IP address and a service port. A port is a software structure that is identified by the port number, a 16 bit integer value, allowing for port numbers 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.

The Internet Assigned Numbers Authority (IANA) has divided port numbers into three ranges.[3] Port numbers 0 through 1023 are used for common, well-known services. On Unix-like operating systems, using one of these ports requires superuser operating permission. Port numbers 1024 through 49 151 are the registered ports used for IANA-registered services. Ports 49 152 through 65 535 are dynamic ports that are not officially designated for any specific service, and may be used for any purpose. They also are used as ephemeral ports, from which software running on the host may randomly choose a port in order to define itself.[3] In effect, they are used as temporary ports primarily by clients when communicating with servers.

Packet structure

UDP is a minimal message-oriented Transport Layer protocol that is documented in IETF RFC 768.

UDP provides no guarantees to the upper layer protocol for message delivery and the UDP protocol layer retains no state of UDP messages once sent. For this reason, UDP sometimes is referred to as *Unreliable* Datagram Protocol.[4]

UDP provides application multiplexing (via port numbers) and integrity verification (via checksum) of the header and payload.[5] If transmission reliability is desired, it must be implemented in the user's application.

UDP Header

Offsets	**Octet**	**0**								**1**								**2**									
Octet	**Bit**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**
0	**0**	Source port																Destination por									
4	**32**	Length																Checksum									

The UDP header consists of 4 fields, each of which is 2 bytes (16 bits).[2] The use of the fields "Checksum" and "Source port" is optional in IPv4 (pink background in table). In IPv6 only the source port is optional

(see below).

Source port number

This field identifies the sender's port when meaningful and should be assumed to be the port to reply to if needed. If not used, then it should be zero. If the source host is the client, the port number is likely to be an ephemeral port number. If the source host is the server, the port number is likely to be a well-known port number.[3]

Destination port number

This field identifies the receiver's port and is required. Similar to source port number, if the client is the destination host then the port number will likely be an ephemeral port number and if the destination host is the server then the port number will likely be a well-known port number.[3]

Length

A field that specifies the length in bytes of the UDP header and UDP data. The minimum length is 8 bytes since that's the length of the header. The field size sets a theoretical limit of 65,535 bytes (8 byte header + 65,527 bytes of data) for a UDP datagram. The practical limit for the data length which is imposed by the underlying IPv4 protocol is 65,507 bytes (65,535 − 8 byte UDP header − 20 byte IP header).[3]

In IPv6 Jumbograms it is possible to have UDP packets of size greater than 65,535 bytes.[6] RFC 2675 specifies that the length field is set to zero if the length of the UDP header plus UDP data is greater than 65,535.

Checksum

The checksum field is used for error-checking of the header *and* data. If no checksum is generated by the transmitter, the field uses the value all-zeros.[7] This field is not optional for IPv6.[8]

Checksum computation

The method used to compute the checksum is defined in 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.*[7]

In other words, all 16-bit words are summed using one's complement arithmetic. The sum is then one's complemented to yield the value of the UDP checksum field.

If the checksum calculation results in the value zero (all 16 bits 0) it should be sent as the one's complement (all 1s).

The difference between IPv4 and IPv6 is in the data used to compute the checksum.

IPv4 Pseudo Header

When UDP runs over IPv4, the checksum is computed using a "pseudo header"[9] that contains some of the same information from the real IPv4 header. The pseudo header is not the real IPv4 header used to send an IP packet, it is used only for the checksum calculation.

IPv4 Pseudo Header Format

Offsets	Octet	0								1								2								3				
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
0	0	Source IPv4 Address																												
4	32	Destination IPv4 Address																												
8	64	Zeroes								Protocol								UDP Length												
12	96	Source Port																Destination Port												
16	128	Length																Checksum												

The source and destination addresses are those in the IPv4 header. The protocol is that for UDP (see *List of IP protocol numbers*): 17 (0x11). The UDP length field is the length of the UDP header and data.

UDP checksum computation is optional for IPv4. If a checksum is not used it should be set to the value zero.

IPv6 Pseudo Header

When UDP runs over IPv6, the checksum is mandatory. The method used to compute it is changed as documented in 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.*[8]

When computing the checksum, again a pseudo header is used that mimics the real IPv6 header:

IPv6 Pseudo Header Format

Offsets	Octet	0								1								2								3				
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
0	0	Source IPv6 Address																												
4	32																													
8	64																													
12	96																													
16	128	Destination IPv6 Address																												
20	160																													
24	192																													
28	224																													
32	256	UDP Length																												
36	288	Zeroes																								Next Head				
40	320	Source Port																Destination Port												
44	352	Length																Checksum												

The source address is the one in the IPv6 header. The destination address is the final destination; if the IPv6 packet does not 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 value of the Next Header field is the protocol value for UDP: 17. The UDP length field is the length of the UDP header and data.

Reliability and congestion control solutions

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.[3]

Most often, UDP applications do not employ 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. In these particular applications, loss of packets is not usually a fatal problem. If an application requires a high degree of reliability, a protocol such as the Transmission Control Protocol may be used instead.

Applications

Numerous key Internet applications use UDP, including: the Domain Name System (DNS), where queries must be fast and only consist of a single request followed by a single reply packet, the Simple Network Management Protocol (SNMP), the Routing Information Protocol (RIP)[2] and the Dynamic Host Configuration Protocol (DHCP).

Voice and video traffic is generally transmitted using UDP. Real-time video and audio streaming protocols are designed to handle occasional lost packets, so only slight degradation in quality occurs, rather than large delays if lost packets were retransmitted. Because both TCP and UDP run over the same network, many businesses are finding that a recent increase in UDP traffic from these real-time applications is hindering the performance of applications using TCP, such as point of sale, accounting, and database systems. When TCP detects packet loss, it will throttle back its data rate usage. Since both real-time and business applications are important to businesses, developing quality of service solutions is seen as crucial by some.[10]

Comparison of UDP and TCP

Transmission Control Protocol is a connection-oriented protocol, which means that it requires handshaking to set up end-to-end communications. Once a connection is set up, user data may be sent bi-directionally over the connection.

- *Reliable* – TCP manages message acknowledgment, retransmission and timeout. Multiple attempts to 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 over a connection in sequence, the first message will reach the receiving application first. When data segments arrive in the wrong order, TCP buffers delay the out-of-order data until all data can be properly re-ordered and delivered to the application.
- *Heavyweight* – TCP requires three packets to set up a socket connection, before any user data can be sent. TCP handles reliability and congestion control.
- *Streaming* – Data is read as a byte stream, no distinguishing indications are transmitted to signal message (segment) boundaries.

UDP is a simpler message-based connectionless protocol. Connectionless protocols do not set up a dedicated end-to-end connection. Communication is achieved by transmitting information in one direction from source to destination without verifying the readiness or state of the receiver. However, one primary benefit of UDP over TCP is the application to VoIP where latency and jitter are the primary concerns. It is assumed in VoIP UDP that the end users provide any necessary real time confirmation that the message has been received.

- *Unreliable* – When a message is sent, it cannot be known if it will reach its destination; it could get lost along the way. There is no concept of acknowledgment, retransmission, or 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 is a small transport layer designed on top of IP.
- *Datagrams* – Packets are sent individually and are checked for integrity only if they arrive. Packets have definite boundaries which are honored upon receipt, meaning a read operation at the receiver socket will yield an entire message as it was originally sent.
- *No congestion control* – UDP itself does not avoid congestion, unless they implement congestion control measures at the application level.

See also

- List of TCP and UDP port numbers
- Reliable User Datagram Protocol (RUDP)
- SCTP
- Comparison of transport layer protocols
- UDP flood attack
- UDP Data Transport
- UDP Lite, a variant that will deliver packets even if they are malformed
- UDP Helper Address
- µTP (Micro Transport Protocol)

Notes and references

Notes

1. ^ RFC 768 p1
2. ^ ***a b c*** Kurose, J. F.; Ross, K. W. (2010). *Computer Networking: A Top-Down Approach* (5th ed.). Boston, MA: Pearson Education. ISBN 978-0-13-136548-3.
3. ^ ***a b c d e f*** Forouzan, B.A. (2000). *TCP/IP: Protocol Suite, 1st ed*. New Delhi, India: Tata McGraw-Hill Publishing Company Limited.
4. ^ content@ipv6.com. "UDP Protocol Overview" (http://ipv6.com/articles/general/User-Datagram-Protocol.htm). Ipv6.com. Retrieved 17 August 2011.
5. ^ Clark, M.P. (2003). *Data Networks IP and the Internet, 1st ed*. West Sussex, England: John Wiley & Sons Ltd.

6. ^ RFC 2675
7. ^ ***a b*** "Postel, J. (August 1980). RFC 768: User Datagram Protocol. *Internet Engineering Task Force*" (http://tools.ietf.org/html/rfc768).
8. ^ ***a b*** "Deering S. & Hinden R. (December 1998). RFC 2460: Internet Protocol, Version 6 (IPv6) Specification. *Internet Engineering Task Force*." (http://tools.ietf.org/html/rfc2460).
9. ^ RFC 768, p2
10. ^ "The impact of UDP on Data Applications" (http://www.networkperformancedaily.com/2007/08/whiteboard_series_nice_guys_fi.html). Networkperformancedaily.com. Retrieved 17 August 2011.

RFC references

- RFC 768 – User Datagram Protocol
- RFC 2460 – Internet Protocol, Version 6 (IPv6) Specification
- RFC 2675 – IPv6 Jumbograms
- RFC 4113 – Management Information Base for the UDP
- RFC 5405 – Unicast UDP Usage Guidelines for Application Designers

External links

- IANA Port Assignments (http://www.iana.org/assignments/port-numbers)
- The Trouble with UDP Scanning (PDF) (http://condor.depaul.edu/~jkristof/papers/udpscanning.pdf)
- Breakdown of UDP frame (http://www.networksorcery.com/enp/protocol/udp.htm)
- UDP on MSDN Magazine Sockets and WCF (http://msdn.microsoft.com/en-us/magazine/cc163648.aspx)
- UDP connections (http://www.faqs.org/docs/iptables/udpconnections.html)

Retrieved from "http://en.wikipedia.org/w/index.php?title=User_Datagram_Protocol&oldid=615518774"
Categories: Internet protocols | Internet Standards | Transport layer protocols

---