**User Datagram Protocol (UDP)**

* The User Datagram Protocol (UDP) is simplest Transport Layer communication protocol available of the TCP/IP protocol suite.
* It involves **minimum amount of communication** mechanism.
* UDP is said to be an unreliable transport protocol but it uses IP services which provides best effort delivery mechanism.
* In UDP, the receiver does not generate an acknowledgement of packet received and in turn, the sender does not wait for any acknowledgement of packet sent. This shortcoming makes this **protocol unreliable as well as easier on processing.**
* It is an **unreliable and connectionless protocol.**So, there is no need to establish a connection before data transfer.
* The UDP helps to **establish low-latency and loss-tolerating connections** over the network.
* The UDP enables **process-to-process** communication.
* It is a communication protocol used across the internet for **time-sensitive transmissions such as video playback or**[**DNS lookups**](https://www.geeksforgeeks.org/dns-look-up/).
* UDP is **connectionless and does not guarantee delivery, order, or error checking**, making it a lightweight and efficient option for certain types of data transmission.

## Requirement of UDP

A question may arise, why do we need an unreliable protocol to transport the data?

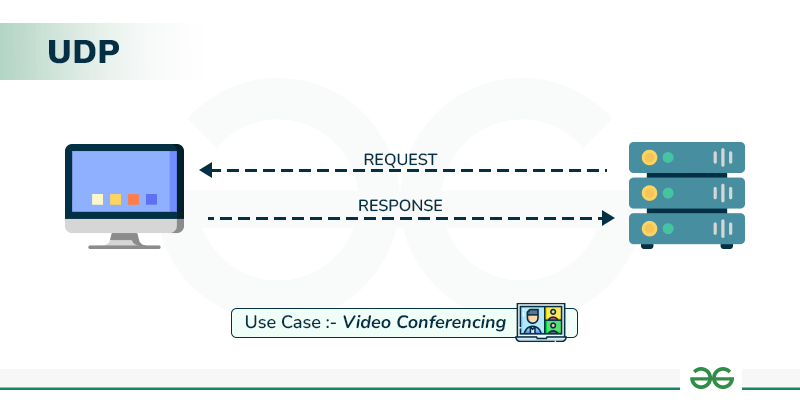
We deploy UDP where the acknowledgement packets share significant amount of bandwidth along with the actual data.

For example, in case of video streaming, thousands of packets are forwarded towards its users. Acknowledging all the packets is troublesome and may contain huge amount of bandwidth wastage. The best delivery mechanism of underlying IP protocol ensures best efforts to deliver its packets, but even if some packets in video streaming get lost, the impact is not calamitous and can be ignored easily.

Loss of few packets in video and voice traffic sometimes goes unnoticed.

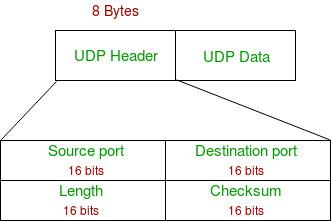
## Features

* UDP is used when acknowledgement of data does not hold any significance.
* UDP is good protocol for data flowing in one direction.
* UDP is simple and suitable for query based communications.
* UDP is not connection oriented.
* UDP does not provide congestion control mechanism.
* UDP does not guarantee ordered delivery of data.
* UDP is stateless.
* UDP is suitable protocol for streaming applications such as VoIP, multimedia streaming.



**UDP Header**

* UDP header is an **8-byte**fixed and simple header.
* The first 8 Bytes contain all necessary header information and the remaining part consists of data.
* UDP port number fields are each 16 bits long, therefore the range for port numbers is defined from 0 to 65535; port number 0 is reserved.
* Port numbers help to distinguish different user requests or processes.



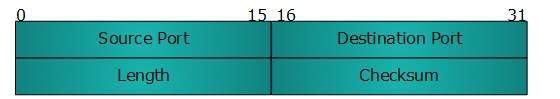
*UDP Header*

* **Source Port:** Source Port is a 2 Byte long field used to identify the [port number](https://www.geeksforgeeks.org/what-is-ports-in-networking/)of the source.
* **Destination Port:** It is a 2 Byte long field, used to identify the port of the destined packet.
* **Length:** Length is the length of UDP including the header and the data. It is a 16-bits field.
* **Checksum:** Checksum is 2 Bytes long field. It is the 16-bit one’s complement of the one’s complement sum of the UDP header, the pseudo-header of information from the IP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

**Notes –**Unlike TCP, **the Checksum calculation is not mandatory in UDP**. No **Error control or flow control** is provided by UDP. Hence UDP depends on IP and ICMP for error reporting. Also **UDP provides port numbers** so that is can differentiate between users requests.

UDP Header

UDP header is as simple as its function.



UDP header contains four main parameters:

* **Source Port**  - This 16 bits information is used to identify the source port of the packet.
* **Destination Port**  - This 16 bits information, is used identify application level service on destination machine.
* **Length**  - Length field specifies the entire length of UDP packet (including header). It is 16-bits field and minimum value is 8-byte, i.e. the size of UDP header itself.
* **Checksum**  - This field stores the checksum value generated by the sender before sending. IPv4 has this field as optional so when checksum field does not contain any value it is made 0 and all its bits are set to zero.

**Applications of UDP**

* **Used for simple request-response communication** when the size of data is less and hence there is lesser concern about flow and error control.
* It is a suitable protocol **for multicasting as UDP supports packet switching**.
* UDP is used for some **routing update protocols like**[**RIP(Routing Information Protocol).**](https://www.geeksforgeeks.org/routing-information-protocol-rip/)
* Normally used for real-time applications which can not tolerate uneven delays between sections of a received message.
* [**VoIP (Voice over Internet Protocol)**](https://www.geeksforgeeks.org/voice-over-internet-protocol-voip/) services, such as Skype and WhatsApp, use UDP for real-time voice communication. The delay in voice communication can be noticeable if packets are delayed due to congestion control, so UDP is used to ensure fast and efficient data transmission.
* [**DNS (Domain Name System)**](https://www.geeksforgeeks.org/domain-name-system-dns-in-application-layer/) also uses UDP for its query/response messages. DNS queries are typically small and require a quick response time, making UDP a suitable protocol for this application.
* [**DHCP (Dynamic Host Configuration Protocol)**](https://www.geeksforgeeks.org/dynamic-host-configuration-protocol-dhcp/) uses UDP to dynamically assign IP addresses to devices on a network. DHCP messages are typically small, and the delay caused by packet loss or retransmission is generally not critical for this application.
* UDP takes a datagram from[Network Layer](https://www.geeksforgeeks.org/network-layer-services-packetizing-routing-and-forwarding/), attaches its header, and sends it to the user. So, it works fast.

**Advantages of UDP**

* **Speed:** UDP is faster than TCP because it does not have the overhead of establishing a connection and ensuring reliable data delivery.
* Lower latency: Since there is no connection establishment, there is lower latency and faster response time.
* **Simplicity:** UDP has a simpler protocol design than TCP, making it easier to implement and manage.
* **Broadcast support:** UDP supports broadcasting to multiple recipients, making it useful for applications such as video streaming and online gaming.
* **Smaller packet size:** UDP uses smaller packet sizes than TCP, which can reduce network congestion and improve overall network performance.
* User Datagram Protocol (UDP) is more efficient in terms of both latency and bandwidth.

**Disadvantages of UDP**

* **No reliability:** UDP does not guarantee delivery of packets or order of delivery, which can lead to missing or duplicate data.
* **No congestion control:** UDP does not have congestion control, which means that it can send packets at a rate that can cause network congestion.
* **Vulnerable to attacks:** UDP is vulnerable to [denial-of-service attacks](https://www.geeksforgeeks.org/deniel-service-prevention/), where an attacker can flood a network with UDP packets, overwhelming the network and causing it to crash.
* **Limited use cases:** UDP is not suitable for applications that require reliable data delivery, such as email or file transfers, and is better suited for applications that can tolerate some data loss, such as video streaming or online gaming.

## UDP applications

Applications of UDP include the following:

1. Lossless data transmission.
2. Gaming, voice and video.
3. Services that don't need fixed packet transmission.
4. Multicasting and routing update protocols.
5. Fast applications.

### Lossless data transmission

Applications that require lossless data transmission can make use of UDP. For example, an application that is configured to manage the process of retransmitting lost packets and correctly arrange received packets might use UDP. This approach can help to improve the data transfer rate of large files compared to TCP.

In the [Open Systems Interconnection](https://www.techtarget.com/searchnetworking/definition/OSI) communication model, UDP is in Layer 4, the transport layer. UDP works in conjunction with higher-level protocols to help manage data transmission services, including Trivial File Transfer Protocol, Real Time Streaming Protocol and Simple Network Management Protocol.

### Gaming, voice and video

UDP is an ideal protocol for network applications in which perceived latency is critical, such as in gaming, voice and video communications. These examples can suffer some data loss without adversely affecting perceived quality. In some cases, however, forward error correction techniques are used in addition to UDP to improve audio and video quality, despite some loss.

### Services that don't need fixed packet transmission

UDP can also be used in applications that depend on the reliable exchange of information but should have their own methods to answer packets. These services are advantageous because they're not bound to fixed patterns to guarantee the completeness and correctness of the data packets sent. Users can decide how and when to respond to incorrect or unsorted information.

### Multicasting and routing update protocols

Multicasting can use UDP because it supports packet switching. In addition, some routing update protocols, such as [Routing Information Protocol](https://www.techtarget.com/searchnetworking/definition/Routing-Information-Protocol), can also use UDP.

### Fast applications

UDP can be used in applications where speed -- rather than reliability -- is critical. For example, it might be prudent to use UDP in an application that sends data from a fast acquisition where it's OK to lose some data points

# Java Socket Programming

Java Socket programming is used for communication between the applications running on different JRE.

Java Socket programming can be connection-oriented or connection-less.

Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

The client in socket programming must know two information:

**IP Address of Server, and**

**Port number.**

**Here, we are going to make one-way client and server communication. In this application, client sends a message to the server, server reads the message and prints it. Here, two classes are being used: Socket and ServerSocket. The Socket class is used to communicate client and server. Through this class, we can read and write message. The ServerSocket class is used at server-side. The accept() method of ServerSocket class blocks the console until the client is connected. After the successful connection of client, it returns the instance of Socket at server-side.**

**Socket Programming in Java**

**Socket class**

**A socket is simply an endpoint for communications between the machines. The Socket class can be used to create a socket.**

**Important methods**

**Method**

**Description**

**1) public InputStream getInputStream()**

**returns the InputStream attached with this socket.**

**2) public OutputStream getOutputStream()**

**returns the OutputStream attached with this socket.**

**3) public synchronized void close()**

**closes this socket**

**ServerSocket class**

**The ServerSocket class can be used to create a server socket. This object is used to establish communication with the clients.**

**Important methods**

**Method**

**Description**

**1) public Socket accept()**

**returns the socket and establish a connection between server and client.**

**2) public synchronized void close()**

**closes the server socket.**

**Example of Java Socket Programming**

**Creating Server:**

**To create the server application, we need to create the instance of ServerSocket class. Here, we are using 6666 port number for the communication between the client and server. You may also choose any other port number. The accept() method waits for the client. If clients connects with the given port number, it returns an instance of Socket.**

**ServerSocket ss=new ServerSocket(6666);**

**Socket s=ss.accept();//establishes connection and waits for the client**

**Creating Client:**

**To create the client application, we need to create the instance of Socket class. Here, we need to pass the IP address or hostname of the Server and a port number. Here, we are using "localhost" because our server is running on same system.**

**Socket s=new Socket("localhost",6666);**

**Let's see a simple of Java socket programming where client sends a text and server receives and prints it.**

**File: MyServer.java**

**import java.io.\*;**

**import java.net.\*;**

**public class MyServer {**

**public static void main(String[] args){**

**try{**

**ServerSocket ss=new ServerSocket(6666);**

**Socket s=ss.accept();//establishes connection**

**DataInputStream dis=new DataInputStream(s.getInputStream());**

**String str=(String)dis.readUTF();**

**System.out.println("message= "+str);**

**ss.close();**

**}catch(Exception e){System.out.println(e);}**

**}**

**}**

**File: MyClient.java**

**import java.io.\*;**

**import java.net.\*;**

**public class MyClient {**

**public static void main(String[] args) {**

**try{**

**Socket s=new Socket("localhost",6666);**

**DataOutputStream dout=new DataOutputStream(s.getOutputStream());**

**dout.writeUTF("Hello Server");**

**dout.flush();**

**dout.close();**

**s.close();**

**}catch(Exception e){System.out.println(e);}**

**}**

**}**

**To execute this program open two command prompts and execute each program at each command prompt as displayed in the below figure.**

**After running the client application, a message will be displayed on the server console.**

**Example of Java Socket Programming (Read-Write both side)**

**In this example, client will write first to the server then server will receive and print the text. Then server will write to the client and client will receive and print the text. The step goes on.**

**File: MyServer.java**

**import java.net.\*;**

**import java.io.\*;**

**class MyServer{**

**public static void main(String args[])throws Exception{**

**ServerSocket ss=new ServerSocket(3333);**

**Socket s=ss.accept();**

**DataInputStream din=new DataInputStream(s.getInputStream());**

**DataOutputStream dout=new DataOutputStream(s.getOutputStream());**

**BufferedReader br=new BufferedReader(new InputStreamReader(System.in));**

**String str="",str2="";**

**while(!str.equals("stop")){**

**str=din.readUTF();**

**System.out.println("client says: "+str);**

**str2=br.readLine();**

**dout.writeUTF(str2);**

**dout.flush();**

**}**

**din.close();**

**s.close();**

**ss.close();**

**}}**

**File: MyClient.java**

**import java.net.\*;**

**import java.io.\*;**

**class MyClient{**

**public static void main(String args[])throws Exception{**

**Socket s=new Socket("localhost",3333);**

**DataInputStream din=new DataInputStream(s.getInputStream());**

**DataOutputStream dout=new DataOutputStream(s.getOutputStream());**

**BufferedReader br=new BufferedReader(new InputStreamReader(System.in));**

**String str="",str2="";**

**while(!str.equals("stop")){**

**str=br.readLine();**

**dout.writeUTF(str);**

**dout.flush();**

**str2=din.readUTF();**

**System.out.println("Server says: "+str2);**

**}**

**dout.close();**

**s.close();**

**}}**

**TCP vs UDP**

| **Basis** | **Transmission Control Protocol (TCP)** | **User Datagram Protocol (UDP)** |
| --- | --- | --- |
| **Type of Service** | [TCP](https://www.geeksforgeeks.org/what-is-transmission-control-protocol-tcp/)is a connection-oriented protocol. Connection orientation means that the communicating devices should establish a connection before transmitting data and should close the connection after transmitting the data. | [UDP](https://www.geeksforgeeks.org/user-datagram-protocol-udp/)is the Datagram-oriented protocol. This is because there is no overhead for opening a connection, maintaining a connection, or terminating a connection. UDP is efficient for broadcast and multicast types of network transmission. |
| **Reliability** | TCP is reliable as it guarantees the delivery of data to the destination router. | The delivery of data to the destination cannot be guaranteed in UDP. |
| **Error checking mechanism** | TCP provides extensive error-checking mechanisms. It is because it provides flow control and acknowledgment of data. | UDP has only the basic error-checking mechanism using checksums. |
| **Acknowledgme nt** | An acknowledgment segment is present. | No acknowledgment segment. |
| **Sequence** | Sequencing of data is a feature of Transmission Control Protocol (TCP). this means that packets arrive in order at the receiver. | There is no sequencing of data in UDP. If the order is required, it has to be managed by the application layer. |
| **Speed** | TCP is comparatively slower than UDP. | UDP is faster, simpler, and more efficient than TCP. |
| **Retransmission** | Retransmission of lost packets is possible in TCP, but not in UDP. | There is no retransmission of lost packets in the User Datagram Protocol (UDP). |
| **Header Length** | TCP has a (20-60) bytes variable length header. | UDP has an 8 bytes fixed-length header. |
| **Weight** | TCP is heavy-weight. | UDP is lightweight. |
| **Handshaking Techniques** | Uses handshakes such as SYN, ACK, SYN-ACK | It’s a connectionless protocol i.e. No handshake |
| **Broadcasting** | TCP doesn’t support Broadcasting. | UDP supports Broadcasting. |
| **Protocols** | TCP is used by [HTTP, HTTPs](https://www.geeksforgeeks.org/difference-between-http-and-https-2/), [FTP](https://www.geeksforgeeks.org/file-transfer-protocol-ftp/), [SMTP](https://www.geeksforgeeks.org/simple-mail-transfer-protocol-smtp/)and [Telnet](https://www.geeksforgeeks.org/introduction-to-telnet/). | UDP is used by DNS, DHCP, TFTP, [SNMP](https://www.geeksforgeeks.org/simple-network-management-protocol-snmp/), RIP, and VoIP. |
| **Stream Type** | The TCP connection is a byte stream. | UDP connection is a message stream. |
| **Overhead** | Low but higher than UDP. | Very low. |
| **Applications** | This protocol is primarily utilized in situations when a safe and trustworthy communication procedure is necessary, such as in email, on the web surfing, and in military services. | This protocol is used in situations where quick communication is necessary but where dependability is not a concern, such as VoIP, game streaming, video, and music streaming, etc. |