**All About TCP**

**Transmission Control Protocol (TCP)**

When you open a web page on your browser, your device sends TCP packets to the server’s address. It’s a request to the server to send the data and information for the website. The web server replies by sending a course of TCP packets. **These packets are stitched together by the browser to display a web page on your screen**.

When you click on a link given on that page, or go to a different website, the browser again sends TCP packets to the server and the server responds by sending back more TCP data packets.

In essence, **TCP isn’t a one-way communication**. These **packets are sent back and forth by your browser to the server and from the server to the browser**.

**TCP numbers the packets so the recipient can get them in order**. When the recipient gets a packet, it sends an acknowledgement to the sender. If the sender doesn’t get the acknowledgement, it will assume that the packet was not received so it will resend it.

**TCP is serious about reliability**. The packets are checked for errors to make sure the request is fulfilled correctly. TCP packets are **tracked to make sure that no data is lost in between**. Packets are also checked for corruption. This is why when you download files using TCP, they are received perfectly even if there are network issues in between.

## ****All About UDP****

**User Datagram Protocol (UDP)**

As discussed earlier, UDP works similar to TCP in the sense that it transmits data packets. However, **it doesn’t do any kind of error-checking**. To make it lightweight and easy, UDP lets go of the reliability factor.

Putting packets in sequence, sending acknowledgements, and requesting resends takes a lot of time and it slows things down. **UDP makes things fast by removing all the overhead steps**.

With UDP, the sender sends the packets to the recipient. The sender doesn’t care if the recipient received them or not. **It will continue sending the data packets**. As a receiver, if you missed a couple of UDP packets in between, you can’t ask for them again.

Did you get all the packets? There’s no way to know. And you cannot get back any missing packets either. This sounds like a terrible deal but there’s a good thing about this communication – it’s **faster than TCP**.

As you may have guessed, UDP is not used in connections where reliability is a big concern. It’s used only when speeds are more important than a few lost packets. For example, if you want a network for online gaming, video conferencing, or broadcasts, UDP will be better than TCP.

## ****TCP and UDP: What are the differences?****

|  |  |  |
| --- | --- | --- |
|  | TCP | UDP |
| **Connection** | Connection-oriented | Connectionless |
| **Sequencing** | TCP numbers each packet so they can be arranged in a sequence by the recipient | UDP sends the packets without numbering |
| **Speed** | Slower | Faster |
| **Reliability** | High | Low |
| **Header size** | Packets are heavy because of overheads | Lightweight packets with minimal headers |
| **Error detection/correction** | Error checking and error recovery | Error checking but no recovery. Corrupted packets are simply discarded and not requested again |
| **Acknowledgement** | Acknowledgement sent by the recipient | No acknowledgement is sent |
| **Transfer method** | Stream | Individual packets |
| **Applications** | File transfer, email, web browsing | Video conferencing, gaming, broadcasts |
| **Congestion control** | Yes | No |

**Let’s get into the details.**

As you now know, TCP and UDP both are used in online communications between the client and the server. Here are some primary differences between them.

**1. Connection**

TCP is connection-oriented and UDP is connectionless. This means **that before sending TCP packets, a connection is established between the server and the client**. This process of setting up a connection is called TCP handshaking. The stream of packets is then sent over this connection.

**In UDP, there is no such connection**. Each packet is sent individually and directly from the sender to the receiver without a reliable data channel.

**2. Sequencing**

TCP is a reliable protocol that adds a sequence number to the data packets as it sends out a stream. This helps the recipient arrange and stitch back the message together. UDP doesn’t add a number to its header, which means **the recipient has no way of knowing if it received all the packets** and in the right order.

**3. Speed**

Since UDP doesn’t have many requirements, it offers a faster connection. TCP, on the other hand, is slower but more reliable. **If you need speed more than reliability, you should use UDP instead of TCP.**

**4. Reliability**

TCP has provisions for data packet sequencing, acknowledgements, error detection, and correction. This makes it **a reliable protocol**. On the other hand, **UDP doesn’t have sequencing or acknowledgements**. While UDP has error detection mechanism, it does nothing to correct the error. The erroneous packets are simply discarded.

**5. Header size**

Since TCP has more details (sequence of the packet, error detection, acknowledgement field etc.), **the header of TCP packets is larger than that of UDP packets**. This makes each packet heavy. This is why TCP connections are slower than UDP connections.

**6. Error detection/correction**

**TCP has error detection and correction methods**. When a packet is found to be corrupted, TCP doesn’t send an acknowledgement for it. This prompts the sender to resend the packet. This way, the complete message is delivered without errors.

**In UDP, there is error detection** via checksum but there is no error correction. If a given packet is found to be erroneous, it is simply discarded.

**7. Acknowledgement**

When TCP packets are received by the recipient, it sends back an acknowledgement to the sender. If the sender doesn’t receive the acknowledgement, it will assume that the packets were not delivered or were delivered corrupted. It will then proceed to resend the packets.

On the other hand**, UDP doesn’t send an acknowledgement** so the sender will not know if the packets were received or not.

**8. Transfer method**

**TCP sends out a stream of data packets** while UDP packets are sent individually. Data stream doesn’t have a defined boundary but individual packets possess proper boundaries.

**9. Congestion control**

TCP has provisions for congestion or flow control. Since **TCP is connection-oriented**, it ensures that there is no congestion on the data channel that’s been setup. **UDP is connectionless**and doesn’t care much about congestion. Each packet is sent separately and if a packet is lost due to congestion, the recipient can’t do much about it.

**10. Applications**

**TCP is used in applications** **where reliability is more important**, such as file transfer, emails, and web browsing.**UDP is used in applications where speed is more important** such as video conferencing, live streaming, and online gaming.

# **Transport Layer Security (TLS)**

Transport Layer Security (TLS) is an Internet Engineering Task Force ([IETF](https://www.techtarget.com/whatis/definition/IETF-Internet-Engineering-Task-Force)) standard protocol that provides authentication, privacy and data integrity between two communicating computer applications.

It's the most widely deployed security protocol in use today and is best suited for web browsers and other applications that require data to be securely exchanged over a network.

### How does Transport Layer Security work?

TLS uses a client-server handshake mechanism to establish an encrypted and secure connection and to ensure the authenticity of the communication. Here's a breakdown of the process:

1. Communicating devices exchange encryption capabilities.
2. An authentication process occurs using [digital certificates](https://www.techtarget.com/searchsecurity/definition/digital-certificate) to help prove the server is the entity it claims to be.
3. A session key exchange occurs. During this process, clients and servers must agree on a key to establish the fact that the secure session is indeed between the client and server -- and not something in the middle attempting to hijack the conversation.
4. TLS uses a public key exchange process to establish a shared secret between the communicating devices. The two handshake methods are the Rivest-Shamir-Adleman (RSA) handshake and the [Diffie-Hellman handshake](https://www.techtarget.com/searchsecurity/definition/Diffie-Hellman-key-exchange). Both methods result in the same goal of establishing a shared secret between communicating devices so the communication can't be hijacked. Once the keys are exchanged, data transmissions between devices on the encrypted session can begin.

### 

### **The benefits of Transport Layer Security**

* Security is built directly into each application, as opposed to external software or hardware to build IPsec tunnels.
* There is true end-to-end encryption (E2EE) between communicating devices.
* There is granular control over what can be transmitted or received on an encrypted session.
* Since TLS operates within the upper layers of the Open Systems Interconnection ([OSI](https://www.techtarget.com/searchnetworking/definition/OSI)) model, it doesn't have the network address translation ([NAT](https://www.techtarget.com/searchnetworking/definition/Network-Address-Translation-NAT)) complications that are inherent with IPsec.
* TLS offers logging and auditing functions that are built directly into the protocol.

### **Differences between TLS and SSL**

As mentioned previously, SSL is the precursor to TLS. Thus, most of the differences between the two are evolutionary in nature, as the protocol adjusts to address vulnerabilities and to improve implementation and integration capabilities.

Key differences between SSL and TLS that make TLS a more secure and efficient protocol are **message authentication**, **key material generation** and the supported [cipher](https://www.techtarget.com/searchsecurity/definition/cipher) suites, with TLS supporting newer and more secure algorithms.

## What is HTTP?

HTTP stands for Hypertext Transfer Protocol, and it is a [protocol](https://www.cloudflare.com/learning/network-layer/what-is-a-protocol/) – or a prescribed order and syntax for presenting information – used for transferring data over a network. Most information that is sent over the Internet, including website content and API calls, uses the HTTP protocol. There are two main kinds of HTTP messages: requests and responses.

## What is HTTPS?

The S in HTTPS stands for "secure." HTTPS uses TLS (or SSL) to encrypt HTTP requests and responses, so in the example above, instead of the text, an attacker would see a bunch of seemingly random characters.

## In HTTPS, how does TLS/SSL encrypt HTTP requests and responses?

TLS uses a technology called [public key encryption](https://www.cloudflare.com/learning/ssl/how-does-public-key-encryption-work/): there are two [keys](https://www.cloudflare.com/learning/ssl/what-is-a-cryptographic-key/), a public key and a private key, and the public key is shared with client devices via the server's SSL certificate. When a client opens a connection with a server, the two devices use the public and private key to agree on new keys, called [session keys](https://www.cloudflare.com/learning/ssl/what-is-a-session-key/), to encrypt further communications between them.

All HTTP requests and responses are then encrypted with these session keys, so that anyone who intercepts communications can only see a random string of characters, not the plaintext.

## What is SSL?

[SSL](https://www.cloudflare.com/learning/ssl/what-is-ssl/) stands for Secure Sockets Layer, and it refers to a protocol for encrypting and securing communications that take place on the Internet. Although SSL was replaced by an updated protocol called [TLS (Transport Layer Security)](https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/) some time ago, "SSL" is still a commonly used term for this technology.

The main use case for SSL/TLS is securing communications between a client and a server, but it can also secure email, VoIP, and other communications over unsecured networks.