Jahong Liu

System Design Basics 4: TCP and UDP

TCP and UDP

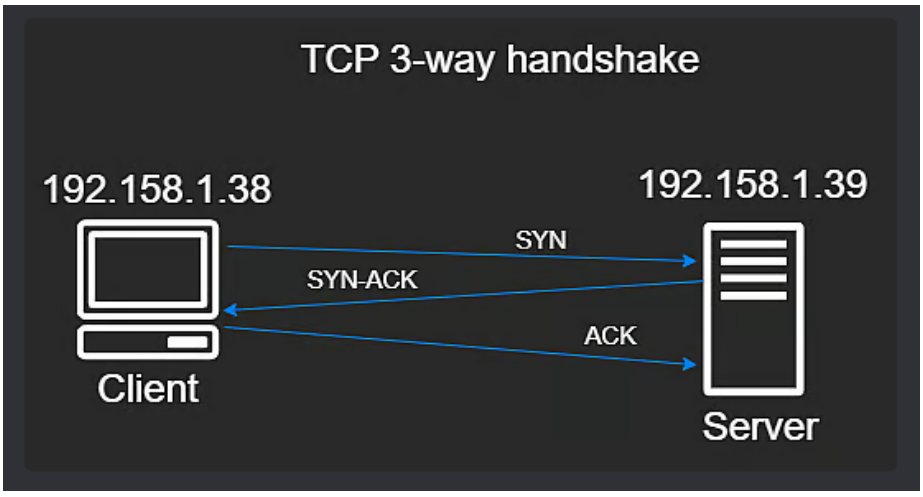
The use cases for TCP

In our prior discussion, we introduced TCP and skimmed over its key features. Now, we're going to delve deeper into TCP and also introduce UDP (User Datagram Protocol). This level of comprehension may not be crucial for developers, but it lays the groundwork necessary for a thorough understanding of the subject matter.

TCP and UDP serve different functions. As we previously stated, when data is transmitted, it's divided into packets. Nonetheless, there's a chance of packet loss during this process. TCP ensures that these lost packets are reliably delivered. It accomplishes this by setting up a two-way connection, commonly referred to as a 3-way handshake, between the devices in communication. As illustrated in the hypothetical diagram below, TCP requires that both the client and server establish a reliable connection before any data exchange take place.

After the connection is established, bidirectional communication can begin. If a data packet is sent and doesn't receive an acknowledgement, TCP assumes it wasn't received and triggers a retransmission. This feature of reliability is a distinguishing characteristic of TCP.

While TCP provides reliability, it does have its drawbacks. It brings about more overhead and tends to be slower due to its rigorous approach to ensuring delivery.



The use cases for UDP

UDP and TCP have distinct qualities when it comes to reliability and speed. UDP, despite being less reliable than TCP, boasts faster data transmittion.

So why opt for UDP instead of TCP, given UDP's lower reliability? UDP finds its purpose in scenarios like online gaming and video straming. In these cases, it is often preferable to skip a dropped frame rather than resend it. This approach ensures smoother gameplay and uninterrupted streaming.

Hence, UDP is selected over TCP in specific situations where speed and efficiency take precedence over reliability and error correction.