0-9

TCP 3-Way Handshake: SYN, SYN-ACK, ACK, and Troubleshooting

Understand the TCP three-way handshake, sequence and acknowledgment numbers, connection states, packet-capture examples, failures, and SYN-flood defenses.

PUBLISHER
Gridinsoft Support

UPDATED
Jul 31, 2026

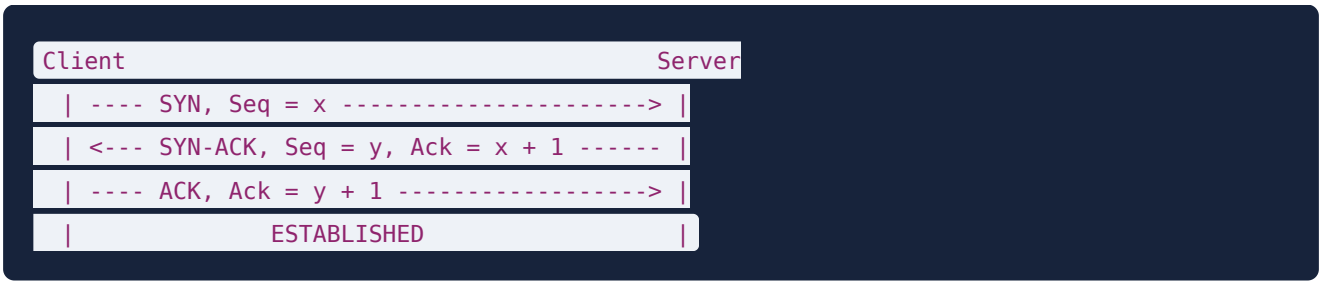
READ TIME
4 min

CONTENTS

In this guide

The **TCP three-way handshake** establishes a reliable connection before application data is exchanged. The client sends **SYN**, the server replies with **SYN-ACK**, and the client completes the handshake with **ACK**. These messages confirm two-way reachability and synchronize the initial sequence numbers used to track bytes.

01 The three steps



1. **SYN:** the active opener, usually the client, proposes its initial sequence number `x` and enters the `SYN-SENT` state.
2. **SYN-ACK:** the listening server acknowledges `x + 1`, proposes its own sequence number `y`, and enters `SYN-RECEIVED`.
3. **ACK:** the client acknowledges `y + 1`. Both endpoints can then enter `ESTABLISHED`.

A SYN consumes one position in TCP sequence space, which is why the acknowledgment number is the received sequence number plus one.

02 Why does TCP use a three-way handshake?

Both endpoints must advertise an initial sequence number and acknowledge the other endpoint's number. The server combines its acknowledgment of the client's SYN with its own SYN in one **SYN-ACK** segment. The client's final ACK confirms that the server's reply arrived, so the server does not treat a one-way or stale SYN as a completed connection.

03 What the handshake proves

A completed handshake confirms that packets and replies can travel between the two TCP endpoints and that both sides agreed on connection state. It does **not** authenticate the user, prove that a website is legitimate, or encrypt the following traffic. Protocols such as TLS perform their own negotiation after the TCP connection is established.

04 Common failure patterns

Capture pattern	Likely meaning
Repeated SYN, no reply	Packet loss, routing failure, silent firewall drop, or unavailable host.
SYN followed by RST	The host is reachable, but the port is closed or the service rejected the connection.
SYN, SYN-ACK, repeated SYN	The client did not receive the reply or its final ACK is being lost or filtered.
Handshake succeeds, application times out	Investigate TLS, authentication, proxy, application, or server processing rather than basic TCP reachability.

05 Inspect the handshake in Wireshark or tcpdump

In Wireshark, this display filter shows initial SYN packets that are not acknowledgments:

```
tcp.flags.syn == 1 && tcp.flags.ack == 0
```

To focus on one TCP conversation, right-click a packet and select **Follow → TCP Stream**, or filter by a stream number such as:

```
tcp.stream == 7
```

On Linux or macOS, an authorized administrator can capture one destination port with:

```
sudo tcpdump -nn -i any 'tcp port 443'
```

Packet captures can contain IP addresses, hostnames, cookies, and application data. Capture only systems and networks you are authorized to inspect, limit the duration, and protect the resulting file.

06 Handshake and connection termination

TCP connection establishment normally uses three messages. Graceful connection termination commonly uses four segments because each direction is closed independently with FIN and ACK. An endpoint can also terminate immediately with RST. Do not describe every TCP close as another three-way handshake.

07 SYN floods and half-open connections

A server normally keeps temporary state after receiving a SYN and before receiving the final ACK. A SYN-flood attack tries to consume that capacity with many incomplete handshakes. Defenses can include SYN cookies, backlog tuning, rate limiting, upstream filtering, load balancers, and connection limits. A burst of incomplete handshakes is not automatically an attack; routing failures and overloaded clients can create similar evidence.

08 Troubleshooting checklist

1. Confirm the destination hostname, resolved IP address, port, and protocol.
2. Verify that the server process is listening on the expected address and port.
3. Compare client, server, load-balancer, and firewall logs using synchronized timestamps.
4. Capture on both sides when possible to determine where a packet disappears.
5. Check NAT, asymmetric routing, security groups, and stateful firewall timeouts.
6. If TCP succeeds, move to the next layer: TLS, proxy, authentication, or the application itself.

NEED MORE HELP?

We are here when the guide is not enough.

Visit the Gridinsoft Support Center for current product guidance or submit a support request with the details of your case.

Open this guide online at support.gridinsoft.com