Understanding TCP

A Wireshark trace to radio.garden

November 2025

Luke Poley

What Stages does TCP go through on radio.garden

Think of it like a phone conversation:

- Picking up the phone → Typing the URL
- Dialing → Requesting initial page (SYN)
- Waiting for answer → Sending initial page (SYN-ACK)
- Talking → Page interaction (ESTABLISHED)
- Saying goodbye → Closing page (FIN)

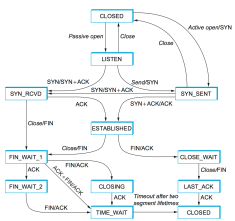


Figure 1: TCP state-transition diagram

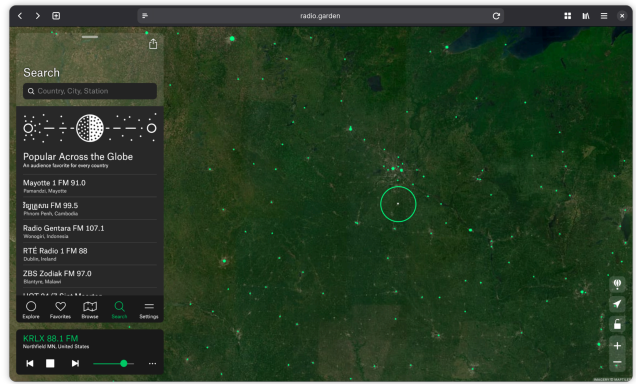


Figure 2: Radio.garden uses a three-tier distribution of: Cloudflare (layout), BunnyCDN (map), Voscast (radio)

1 How does a connection begin? The Three-Way Handshake

After entering the URL, your computer (the client) and the server need to synchronize

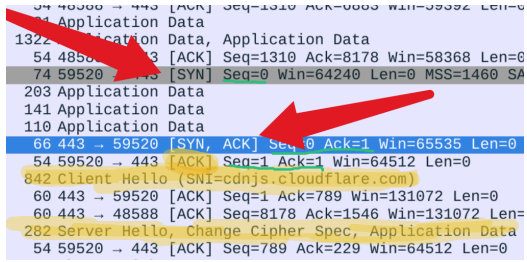


Figure 3: Packets 42-51. This handshake demonstrates TCP's **connection-oriented** service, establishing a session before data is exchanged.

Client Initialization: CLOSED → SYN-SENT:

- Client says "I need to connect to this server"
- TCP creates a SYN (synchronize) packet
- Includes initial sequence number (random for security)
- Sends to server and waits for response

The Three-Way Handshake

Server: LISTEN → SYN-RECEIVED

- Was in LISTEN state (waiting for connection requests)
- Receives SYN packet from client
- Allocates resources for this connection and sequence number
- Sends back SYN-ACK (synchronize + acknowledge)

Client: Sends ACK

- Sends ACK to Server with to verify server's sequence number

Connection will encrypt with TLS

The Handshake will repeat for each server: Cloudflare (Layout), BunnyCDN (Map), Voscast (Radio)

2 The ESTABLISHED State

What Happens in ESTABLISHED State?

This is where TCP spends most of its time - transferring data!

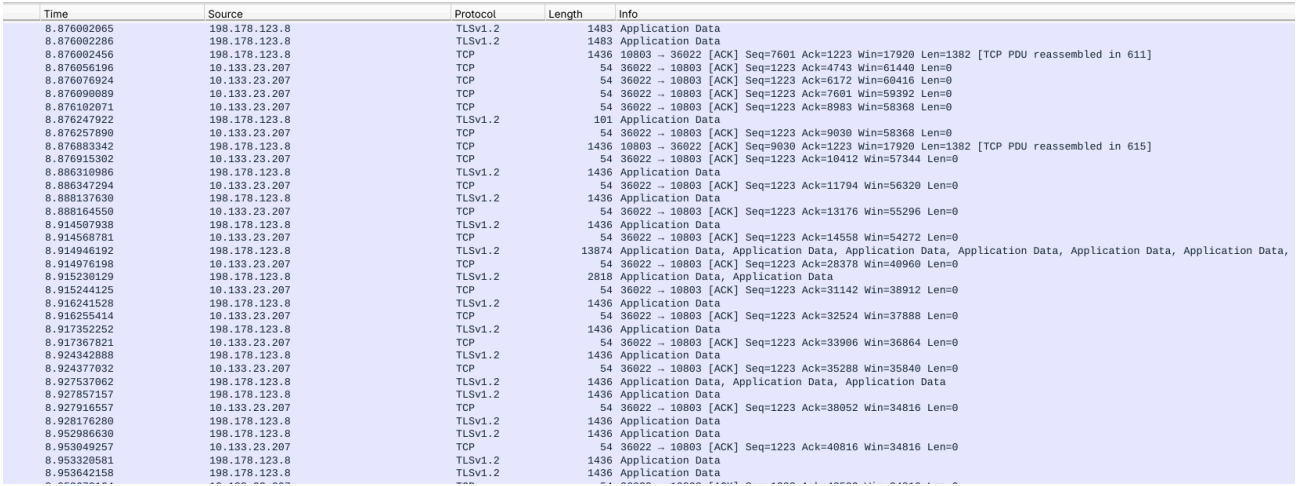
In ESTABLISHED state, both sides can:

- Send data in either direction (full-duplex).
- Exchange data using TCP's core services:
  - **In-order delivery**, ensured by sequence numbers.
  - **Reliable delivery**, ensured by acknowledgements (ACKs).
  - **Flow control**, managed by the sliding window. The advertised receive window limits how much data the sender can have in flight.

What you'll see in Wireshark:

- Many packets flowing back and forth
- All have ACK flag set
- **Sequence (Seq) numbers** incrementing. These are byte-stream counters, not packet counters. The sequence number indicates the first byte of data in the segment. Wireshark uses relative numbers for readability.
- **Window sizes** adjusting. This is established in slow start and prevents the sender's buffer from overflowing. It is separate from the sequence number.

What Happens in ESTABLISHED State? (ii)



Time	Source	Protocol	Length	Info
8.876002065	198.178.123.8	TLSv1.2	1483	Application Data
8.876002286	198.178.123.8	TLSv1.2	1483	Application Data
8.876002456	198.178.123.8	TCP	1436	10803 → 36022 [ACK] Seq=7601 Ack=1223 Win=17920 Len=1382 [TCP PDU reassembled in 611]
8.876056196	10.133.23.207	TCP	54	36022 → 10803 [ACK] Seq=1223 Ack=4743 Win=61440 Len=0
8.876076924	10.133.23.207	TCP	54	36022 → 10803 [ACK] Seq=1223 Ack=6172 Win=60416 Len=0
8.876090089	10.133.23.207	TCP	54	36022 → 10803 [ACK] Seq=1223 Ack=7601 Win=59392 Len=0
8.876102071	10.133.23.207	TCP	54	36022 → 10803 [ACK] Seq=1223 Ack=8983 Win=58368 Len=0
8.876247922	198.178.123.8	TLSv1.2	101	Application Data
8.876257890	10.133.23.207	TCP	54	36022 → 10803 [ACK] Seq=1223 Ack=9030 Win=58368 Len=0
8.876883342	198.178.123.8	TCP	1436	10803 → 36022 [ACK] Seq=9030 Ack=1223 Win=17920 Len=1382 [TCP PDU reassembled in 615]
8.876915302	10.133.23.207	TCP	54	36022 → 10803 [ACK] Seq=1223 Ack=10412 Win=57344 Len=0
8.886310986	198.178.123.8	TLSv1.2	1436	Application Data
8.886347294	10.133.23.207	TCP	54	36022 → 10803 [ACK] Seq=1223 Ack=11794 Win=56320 Len=0
8.888137630	198.178.123.8	TLSv1.2	1436	Application Data
8.888164550	10.133.23.207	TCP	54	36022 → 10803 [ACK] Seq=1223 Ack=13176 Win=55296 Len=0
8.914507938	198.178.123.8	TLSv1.2	1436	Application Data
8.914568781	10.133.23.207	TCP	54	36022 → 10803 [ACK] Seq=1223 Ack=14558 Win=54272 Len=0
8.914946192	198.178.123.8	TLSv1.2	13874	Application Data, Application Data, Application Data, Application Data, Application Data, Application Data,
8.914976198	10.133.23.207	TCP	54	36022 → 10803 [ACK] Seq=1223 Ack=28378 Win=40960 Len=0
8.915230129	198.178.123.8	TLSv1.2	2818	Application Data, Application Data
8.915244125	10.133.23.207	TCP	54	36022 → 10803 [ACK] Seq=1223 Ack=31142 Win=38912 Len=0
8.916241528	198.178.123.8	TLSv1.2	1436	Application Data
8.916255414	10.133.23.207	TCP	54	36022 → 10803 [ACK] Seq=1223 Ack=32524 Win=37888 Len=0
8.917352252	198.178.123.8	TLSv1.2	1436	Application Data
8.917367821	10.133.23.207	TCP	54	36022 → 10803 [ACK] Seq=1223 Ack=33906 Win=36864 Len=0
8.924342888	198.178.123.8	TLSv1.2	1436	Application Data
8.924377032	10.133.23.207	TCP	54	36022 → 10803 [ACK] Seq=1223 Ack=35288 Win=35840 Len=0
8.927537062	198.178.123.8	TLSv1.2	1436	Application Data, Application Data, Application Data
8.927857157	198.178.123.8	TLSv1.2	1436	Application Data
8.927916557	10.133.23.207	TCP	54	36022 → 10803 [ACK] Seq=1223 Ack=38052 Win=34816 Len=0
8.928176280	198.178.123.8	TLSv1.2	1436	Application Data
8.952986630	198.178.123.8	TLSv1.2	1436	Application Data
8.953049257	10.133.23.207	TCP	54	36022 → 10803 [ACK] Seq=1223 Ack=40816 Win=34816 Len=0
8.953320581	198.178.123.8	TLSv1.2	1436	Application Data
8.953642158	198.178.123.8	TLSv1.2	1436	Application Data

Figure 4: Packets shortly after the audio streaming started, while the map data is still in transit. (Dynamic Window Sizes) Packets 622 - 649

3 Congestion Control

Slow Start

When a TCP connection begins, it doesn't know the available capacity of the network.

- **Start Small:** The client starts with a small **congestion window** (cwnd), which limits the amount of data it can send.
- **Exponential Growth:** For every ACK received, the cwnd increases. This leads to an exponential increase in the sending rate, doubling roughly every round-trip time.
- **Finding the Limit:** This continues until a packet is lost or the window reaches a predefined threshold.
- **Congestion Avoidance:** After this initial "slow start" phase, TCP switches to a more linear growth of the window to avoid creating new congestion. Seen as ACKs in the trace (packets 91,92)

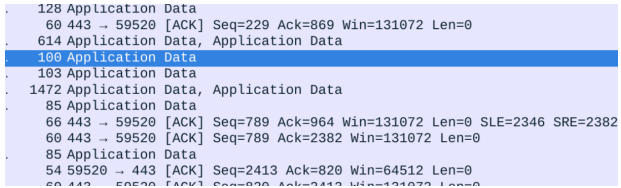


Figure 5: Packets 84-89 sends bursts of data that shows a growing congestion window

Reliable Transport

TCP's reliable transport service ensures data arrives intact and in order, even with packet loss

- Many packets have the same ACK number since it gets lost in transport
- The sender, not knowing yet, keeps sending the later cards (1539, ...)
- The receiver keeps saying it needs packet with ACK=1538

Wireshark Blacklines

- TCP Out-of-Order: Packet arrived before the missing one
- TCP Dup ACK: Receiver is saying "Same ACK again because we are still missing data."

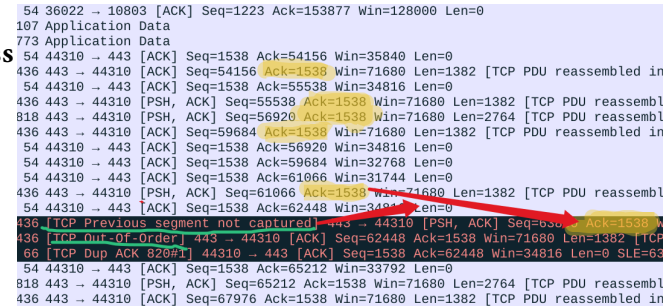
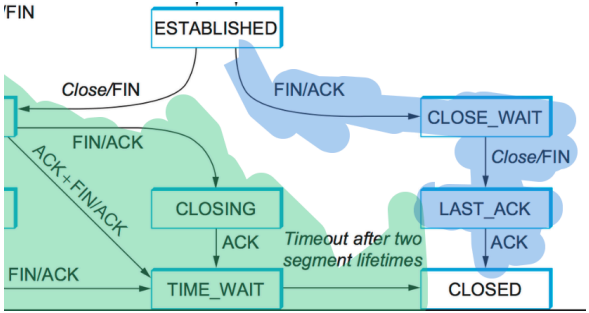


Figure 6: Packets 807-825

4 Connection Termination

Termination: Ending the Session



TCP provides a graceful **connection termination** service.

Blue: The Normal 4-Way Handshake termination from the layout and map servers.

Green: Server sends RST flag for an abrupt termination of the audio stream

1043	198.178.123.8	12.826634729	10.133.23.207	TCP	54 36022 → 10803 [ACK] Seq=1223 Ack=202640 Win=130048 Len=0
1046	198.178.123.8	12.906694853	10.133.23.207	TLSv1.2	85 Encrypted Alert
1047	198.178.123.8	12.906752330	10.133.23.207	TCP	54 36022 → 10803 [FIN, ACK] Seq=1254 Ack=202640 Win=130048 Len=0
1048	10.133.23.207	12.925477437	198.178.123.8	TCP	101 [TCP Previous segment not captured] 10803 → 36022 [PSH, ACK] Seq=204022 Ack=1223 Win=17920 Len=47
1049	198.178.123.8	12.925546816	10.133.23.207	TCP	54 36022 → 10803 [RST] Seq=1223 Win=0 Len=0
1050	10.133.23.207	12.925752438	198.178.123.8	TCP	1436 [TCP Out-Of-Order] 10803 → 36022 [ACK] Seq=202640 Ack=1223 Win=17920 Len=1382
1051	198.178.123.8	12.925762988	10.133.23.207	TCP	54 36022 → 10803 [RST] Seq=1223 Win=0 Len=0
1052	10.133.23.207	12.945030002	198.178.123.8	TCP	60 10803 → 36022 [ACK] Seq=204069 Ack=1254 Win=17920 Len=0
1053	198.178.123.8	12.945086026	10.133.23.207	TCP	54 36022 → 10803 [RST] Seq=1254 Win=0 Len=0
1054	10.133.23.207	12.974677170	198.178.123.8	TCP	1436 10803 → 36022 [ACK] Seq=204069 Ack=1255 Win=17920 Len=1382 [TCP PDU reassembled in 1056]
1055	198.178.123.8	12.974739376	10.133.23.207	TCP	54 36022 → 10803 [RST] Seq=1255 Win=0 Len=0
1056	10.133.23.207	12.976148754	198.178.123.8	TLSv1.2	101 Application Data
1057	198.178.123.8	12.976177918	10.133.23.207	TCP	54 36022 → 10803 [RST] Seq=1255 Win=0 Len=0
1058	10.133.23.207	13.077295964	198.178.123.8	TCP	60 10803 → 36022 [RST, ACK] Seq=205498 Ack=1255 Win=17920 Len=0
1060	38.142.94.218	13.329642961	10.133.23.207	TLSv1.3	93 Application Data

Figure 8: The (Green) Reset Termination as it appears in Wireshark