TCP, UDP and QUIC

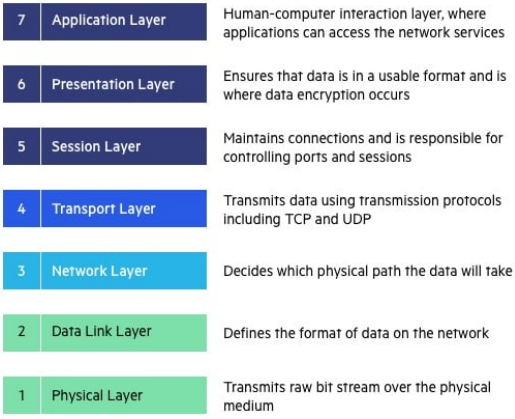
Luca Campa

University of Klagenfurt – Introduction to Cybersecurity

Learning Goals

- Understand how TCP and UDP exchanges differ at protocol level.
- Analyze common security issues and attack techniques.
- Introduction to QUIC.
- Explain why design trade-offs create specific weaknesses.
- Apply practical mitigations at host, network, and application layers.

Transport Layer in Context



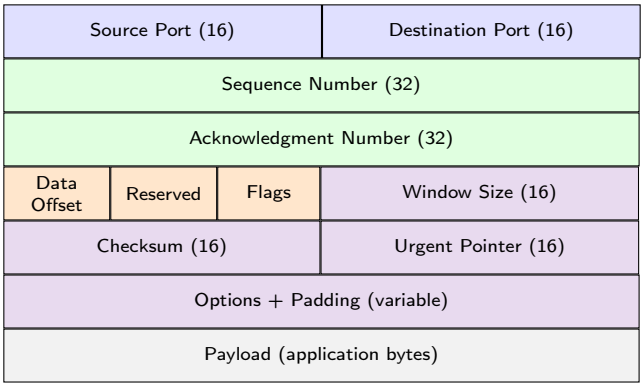
- The transport layer defines endpoint communication semantics.
- Security controls above and below it are constrained by these semantics.

TCP vs UDP at a glance

Property	TCP	UDP
Connection model	Connection-oriented	Connectionless
Reliability	Built-in (SYN-ACK, retransmit)	None by default
Ordering	In-order byte stream	Message/datagram based
Flow/congestion control	Yes	No (application-defined)
Header size	Larger (20+ bytes)	Small (8 bytes)
Typical security risk	State exhaustion, hijack/reset	Replay attacks, spoofing, amplification

TCP Protocol

TCP Segment Format (Header Overview)

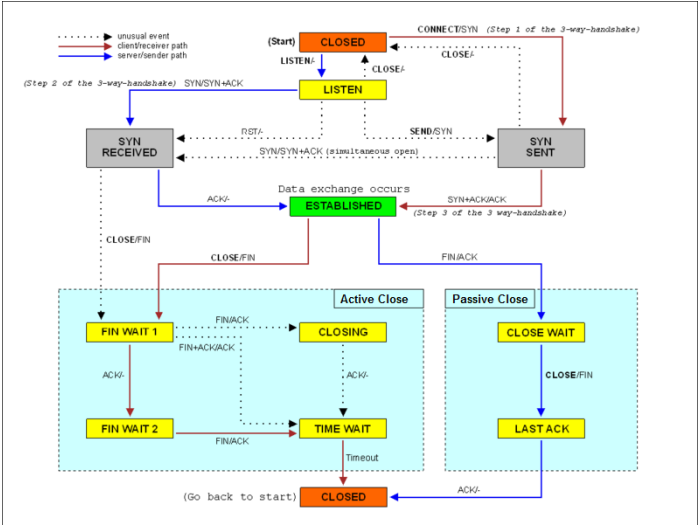


Minimum TCP header is 20 bytes; options increase size and can affect performance and fingerprinting.
Flags: CWR ECE URG ACK PSH RST SYN FIN

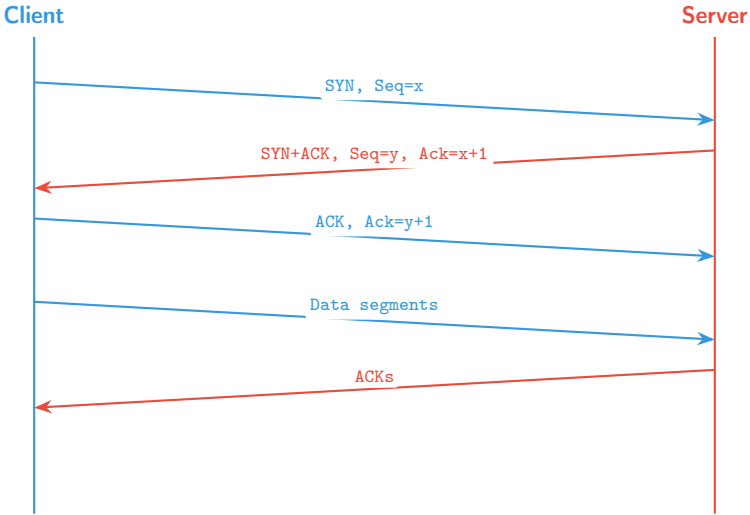
TCP Fields

- **Ports:** map service role and expected policy path.
- **Seq/Ack:** indicate stream progress and forged segment plausibility.
- **Flags:** reveal state-machine actions (setup, data, reset, teardown).
- **Window:** reflects receiver flow-control pressure and tuning behavior.
- **Options:** MSS (Maximum Segment Size), Timestamps.

TCP State Diagram

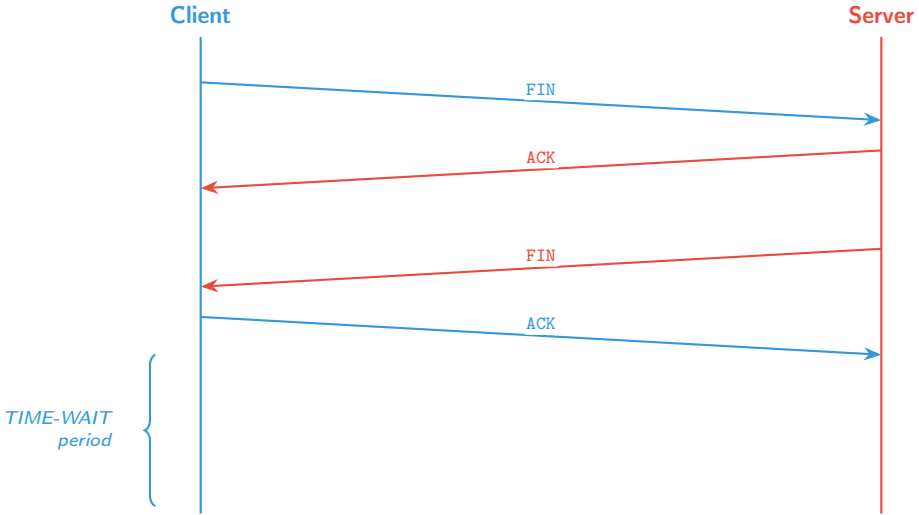


TCP Three-Way Handshake



- Establishes sequence number state and liveness before data transfer.

TCP Connection Termination



- TIME-WAIT helps to prevent delayed packet and old segment reuse.

TCP Security-Relevant Properties

- Stateful protocol: endpoints allocate a certain amount of memory per connection.
- Sequence/ACK numbers protect order, but are **not** a cryptographic authentication mechanism.
- Flags (SYN, ACK, FIN, RST) control state transitions and can be abused.
- Congestion and retransmission logic can be manipulated under adversarial traffic.

TCP Header Fields and Connected Security Implications

Field	Security relevance
Source/Destination Port	Service targeting, lateral movement identification, policy enforcement.
Sequence/Ack Numbers	Window-aware injection attempts.
Flags (SYN, ACK, FIN, RST, PSH, URG)	State transitions, termination abuse, anomaly signatures in IDS (Intrusion Detection System).
Window Size	Flow pressure hints; unusual values may indicate crafted traffic.
Options (MSS, Timestamps)	Fingerprinting, performance behavior.

TCP States and Operational Implications

State	Operational/security implication
LISTEN	Subject to scan and SYN pressure.
FIN-WAIT / CLOSE-WAIT	Mismanagement can leak sockets or lead to app bugs or Denial of Service (DoS) attacks.
TIME-WAIT	Ephemeral-port reuse.

TCP Reliability and Congestion in Practice

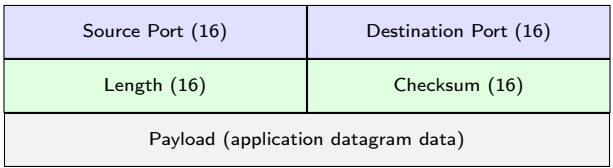
- Reliability: cumulative ACKs and retransmissions recover from packet loss.
- Ordering: receiver reassembles byte stream, missing segments delay delivery.
- Flow control: receiver-advertised window protects endpoint buffers.
- Congestion control: sender adapts rate (slow start, avoidance) to path conditions.

Security angle:

- induced loss/reordering can degrade throughput and inflate latency
- congestion collapse can be triggered by malicious traffic patterns
- reliability mechanisms can be exploited for amplification (e.g., ACK storms)
- stateful design can be abused for resource exhaustion (e.g., SYN floods)
- lack of encryption means metadata and payload are visible to on-path attackers

UDP Protocol

UDP Datagram Format (Header Overview)



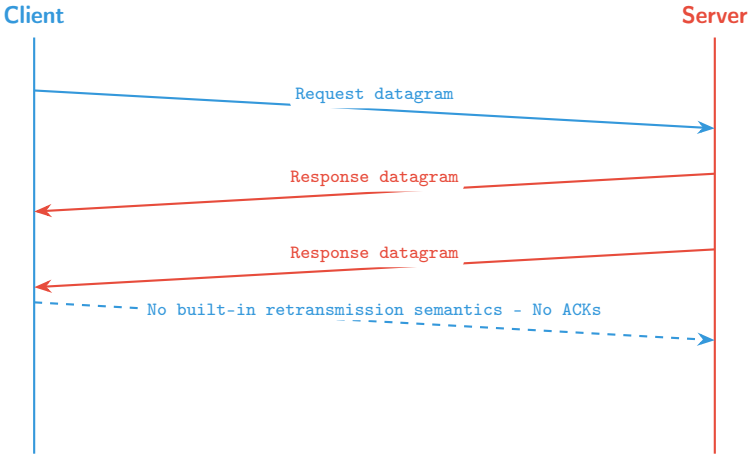
UDP header is fixed at 8 bytes; minimal transport logic means application protocol design is security-critical.

UDP Fields

- **Ports:** identify service type and role.
- **Length:** helps detect malformed packets and suspicious payload asymmetry.
- **Checksum:** integrity signal (especially relevant in IPv6 processing paths).

Security consequence: small fixed headers reduce overhead but shift reliability and validation burden to the application layer.

UDP Exchange Pattern



- No handshake, no built-in reliability, no connection state.
- Great for low latency, but trust must be added at the application layer.

Why UDP Is Still Widely Used

- DNS, DHCP, NTP, SNMP, media streaming, gaming, QUIC bootstrapping.
- Lower overhead and no head-of-line blocking at transport level.
- Works well when applications can tolerate loss/jitter or implement custom reliability.

Security implication: stateless speed can amplify spoofing/replay attacks abuse.

How Applications add Reliability on UDP

- Sequence numbers and app-level ACKs for selective retransmission.
- Jitter buffers and forward error correction in real-time media.
- Adaptive rate control to avoid self-induced congestion.
- Stateless retry tokens/cookies to reduce spoofed request abuse.

Trade-off: each application reinvents reliability/security logic, increasing implementation risk.

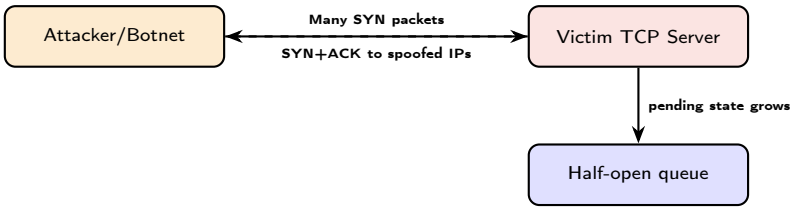
UDP Fragmentation and MTU (Maximum Transmission Unit)

- Large UDP responses may trigger IP fragmentation: UDP does not handle reassembly.
- Fragment loss leads to full datagram loss; attackers can exploit fragility.
- Fragment reassembly at application level can increase CPU/memory pressure on targets.
- Safer design: prefer small responses and path-aware sizing.

Note: oversized datagrams are harder to monitor reliably.

TCP Attacks

SYN Flood (TCP State Exhaustion)



- **Goal**: exhaust backlog, so legitimate clients cannot connect (Denial of Service).

TCP Reset Injection and Session Disruption

- Attacker injects forged RST segments with plausible sequence window.
- If accepted by the endpoint, connection is terminated immediately.
- Historically observed in censorship/disruption and weakly protected paths.

Defenses:

- sequence number hardening and stricter validation
- end-to-end encryption/authentication (e.g., TLS, SSH)
- anti-spoofing and path protection where possible

TCP Session Hijacking

- On-path attacker can observe sequence progress and inject data.
- Off-path hijack is harder today due to randomization but still studied.
- If application layer is not authenticated/encrypted, injected payload can succeed.

Note: transport reliability does not imply endpoint identity trust.

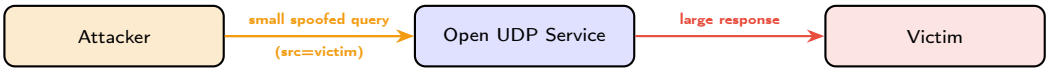
Slow Application-Layer Attacks over TCP

A malicious client opens many TCP connections and sends data very slowly, preventing the server from freeing resources.

Mitigations: connection limits, idle/read timeouts, reverse proxies, adaptive admission control.

UDP Attacks

UDP Replay attacks and Amplification



- Common abused protocols: DNS, NTP, SSDP, CLDAP, memcached-UDP.
- Attack power comes from spoofing + response size asymmetry.

Approximate Amplification Ratios

Protocol	Typical abuse pattern	Relative amplification
DNS	ANY-like/large DNSSEC responses	Moderate to high
NTP	monlist/history queries (legacy)	High
SSDP	UPnP discovery responses	Moderate
Memcached (UDP)	tiny request, huge cached response	Very high

- **Network Time Protocol (NTP)** is an internet protocol used to synchronize the clocks of computer systems and network devices across the globe. The "monlist" command in NTP is a legacy feature that returns a list of the last 600 hosts that have connected to the NTP server.
- **DNS amplification** attacks often involve sending small queries that trigger large responses, especially when DNSSEC is involved, which can significantly increase the size of the response due to cryptographic signatures.
- **SSDP amplification** attacks exploit the Simple Service Discovery Protocol used in UPnP (Universal Plug and Play) for device discovery.
- **Memcached amplification** attacks abuse the distributed memory caching system when it is exposed over UDP. Large responses if the memcached server has a large amount of cached data.

UDP Service Abuse Beyond DDoS

- UDP scanning is fast and low-cost for attackers.
- Spoofed UDP can bypass weak IP-based trust assumptions.
- Application parsers may be exposed to malformed datagram attacks.

Design rule: never trust source IP alone for sensitive authorization logic.

Comparative Security Analysis

Risks comparison

TCP risks:

- state exhaustion
- control flag abuse
- long-lived resource pinning

UDP risks:

- spoofing and replay attacks
- amplification
- weak application-layer trust models

TCP Defensive Controls

- SYN cookies and backlog tuning.
- Rate limiting for new connection attempts.
- Per-IP and global connection quotas.
- Strict timeout policies for idle/incomplete sessions.
- End-to-end encryption and authentication (e.g., TLS) to prevent hijacking and reset abuse.

UDP Defensive Controls

- Restrict service exposure.
- Response rate limiting and payload size control.
- Ingress/egress anti-spoofing.
- Protocol hardening: disable legacy high-amplification features.
- Use authenticated/encrypted application protocols where possible.

Monitoring and Detection Signals

- TCP: check for SYN rate spikes, backlog saturation, abnormal reset rates.
- UDP: check for unexpected source IPs, high request/response ratios, large response sizes.

Best practice: correlate PCAP telemetry (network packet captures) with application logs and host metrics.

Where Encryption Fits

- TCP commonly paired with TLS (HTTPS, SMTPS, IMAPS, etc.).
- UDP increasingly paired with QUIC/TLS 1.3 (HTTP/3).
- Encryption helps confidentiality/integrity but does not stop all DoS patterns.

Key point: cryptography protects data semantics; capacity controls protect availability.

QUIC and TLS

Why QUIC Was Introduced

- Reduce web latency compared to TCP+TLS+HTTP/2 setup.
- Integrate transport control and cryptographic mechanisms.
- Improve connection migration (e.g., Wi-Fi to mobile) via stable connection IDs.

Note: QUIC keeps UDP's deployment flexibility while adding modern reliability and security.

QUIC Packet Format: Long vs Short Headers

Long Header (setup phase)

- Header Form + Fixed Bit + Type
- Version
- Destination Connection ID (DCID)
- Source Connection ID (SCID)
- Token (for Initial)
- Length + Packet Number
- Protected payload (cryptographic data, encrypted frames)

Short Header (1-RTT data)

- Header Form + Key Phase
- Destination Connection ID (optional/implicit)
- Packet Number
- Protected payload (STREAM, ACK, control)

Long headers are used during version negotiation and handshake.
Short headers are used in data traffic.

QUIC Fields

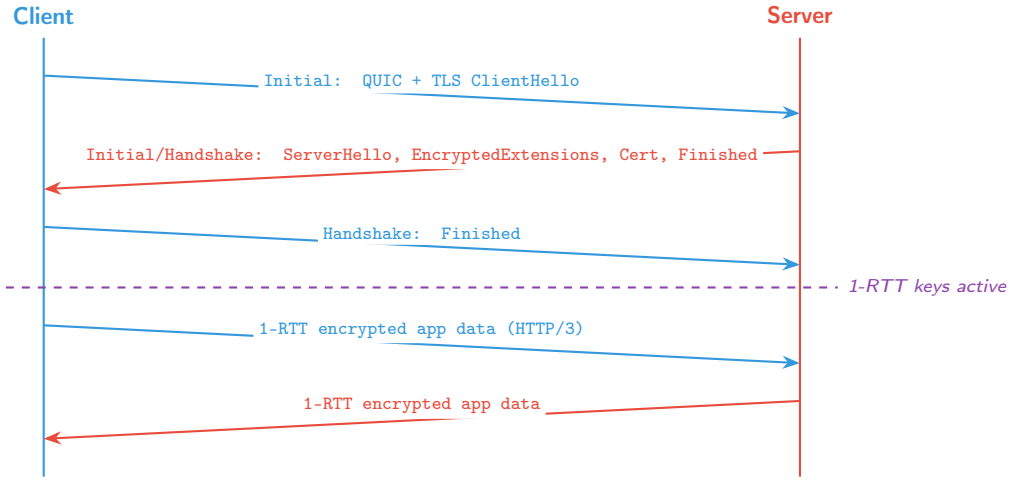
- **Connection IDs:** enable migration and load-balancer routing, but can complicate flow correlation.
- **Packet Number:** strictly increasing per space; gaps can indicate loss/reordering.
- **Header protection and encryption:** most metadata is hidden, reducing fingerprints.
- **Frame types** (STREAM, ACK, CRYPTO, PATH_CHALLENGE): reveal transport behavior when decrypted at endpoints.

TCP+TLS vs QUIC Handshake Cost

Mode	Steps before app data	Latency tendency
TCP + TLS 1.2	TCP 3-way + TLS handshake	higher
TCP + TLS 1.3	TCP 3-way + shorter TLS handshake	moderate
QUIC + TLS 1.3	integrated transport+crypto handshake	lower (1-RTT typical)
QUIC resumed	ticket-based resumption	can send 0-RTT app data

Exact latency depends on network path and server policy, but QUIC usually reduces setup round trips.

QUIC Handshake and TLS 1.3 Integration



Remark: TLS 1.3 runs inside QUIC cryptographic packets; QUIC does not use TCP.

How QUIC Is Connected with TLS

- QUIC mandates TLS 1.3 handshake semantics for authentication and key agreement.
- QUIC encrypts most transport metadata earlier than classic TCP/TLS stacks.
- Packet protection keys are derived from TLS handshake secrets.
- HTTP/3 depends on QUIC transport features (streams, loss recovery, migration).

Remark: QUIC is not "TLS over UDP"; it is a transport protocol with built-in TLS 1.3 cryptography.

QUIC Security Issues and Trade-Offs

- 0-RTT data can be replayed (application must enforce anti-replay semantics).
- Encrypted headers reduce visibility for some network monitoring tools.
- UDP path filtering/rate-limits can unintentionally block QUIC traffic.
- New implementation complexity (user-space) introduces fresh bug surface.

Operational mitigations:

- disable 0-RTT for non-idempotent operations
- maintain robust telemetry at endpoints and load balancers
- keep QUIC/TLS libraries patched and tested

Command Line Interface (CLI) Tools

CLI examples for TCP/UDP monitoring

Observe TCP/UDP sockets:

- `ss -tuna`
- `netstat -tuln`

Capture handshake/reset traffic

- `sudo tcpdump -n 'tcp[tcpflags] & (tcp-syn|tcp-rst) != 0'`

Capture DNS or NTP UDP bursts

- `sudo tcpdump -n 'udp port 53 or udp port 123'`

- Use in isolated lab environments only.

References (Selected)

- RFC 793: Transmission Control Protocol.
- RFC 768: User Datagram Protocol.
- RFC 9293: TCP specification update.
- RFC 8085: UDP usage guidelines.
- RFC 9000: QUIC transport.
- RFC 9001: Using TLS 1.3 with QUIC.
- RFC 9002: QUIC loss detection and congestion control.
- RFC 4987: TCP SYN flooding attacks and mitigations.
- BCP 38 / RFC 2827: Network ingress filtering (anti-spoofing).

Questions?