Working of Protocols

1. Working of TCP (Transmission Control Protocol)

A. Connection Establishment – Three-Way Handshake:

1. Client → Server: Sends a message with a special flag called SYN to say "I want to start a connection."
2. Server → Client: Replies with a message that says SYN-ACK, meaning "I received your request, and I agree to start."
3. Client → Server: Sends an ACK back to say "Thanks, ready to communicate."

Connection is now open.

B. Data Transmission:

1. The client breaks its data (e.g., a file or web page request) into segments.
2. Each segment is given a sequence number.
3. The server receives a segment and sends back an acknowledgment number (ACK) to confirm receipt.
4. If any data segment is missing or damaged:
   - The server doesn't acknowledge it.
   - The client resends that segment.

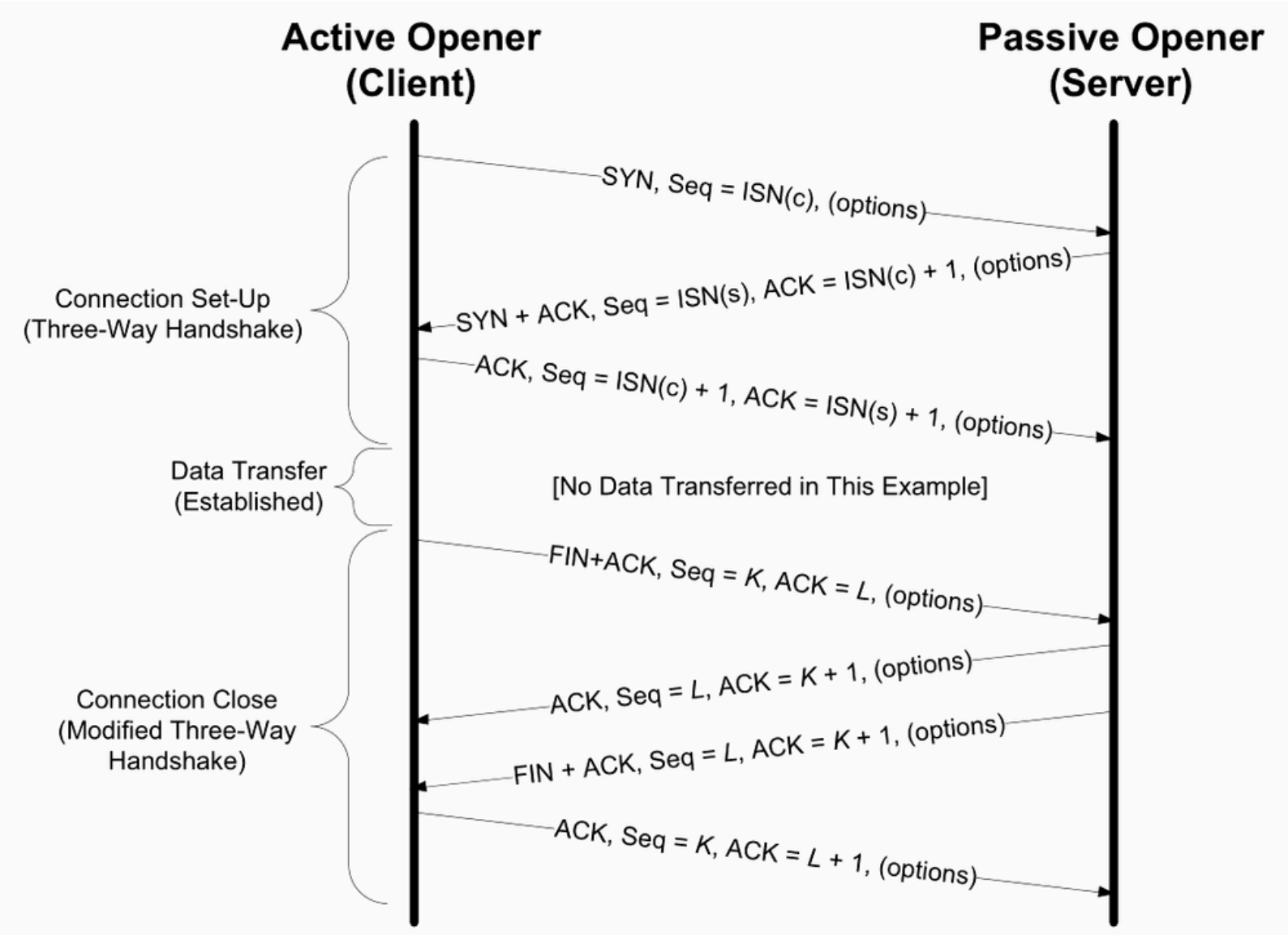
This continues until all data is successfully received and acknowledged.

C. Connection Termination – Four-Step Closure:

1. Client → Server: Sends a FIN message (finished sending).
2. Server → Client: Sends ACK (acknowledges termination request).
3. Server → Client: Then sends its own FIN message.
4. Client → Server: Sends final ACK.

The connection is now fully closed.

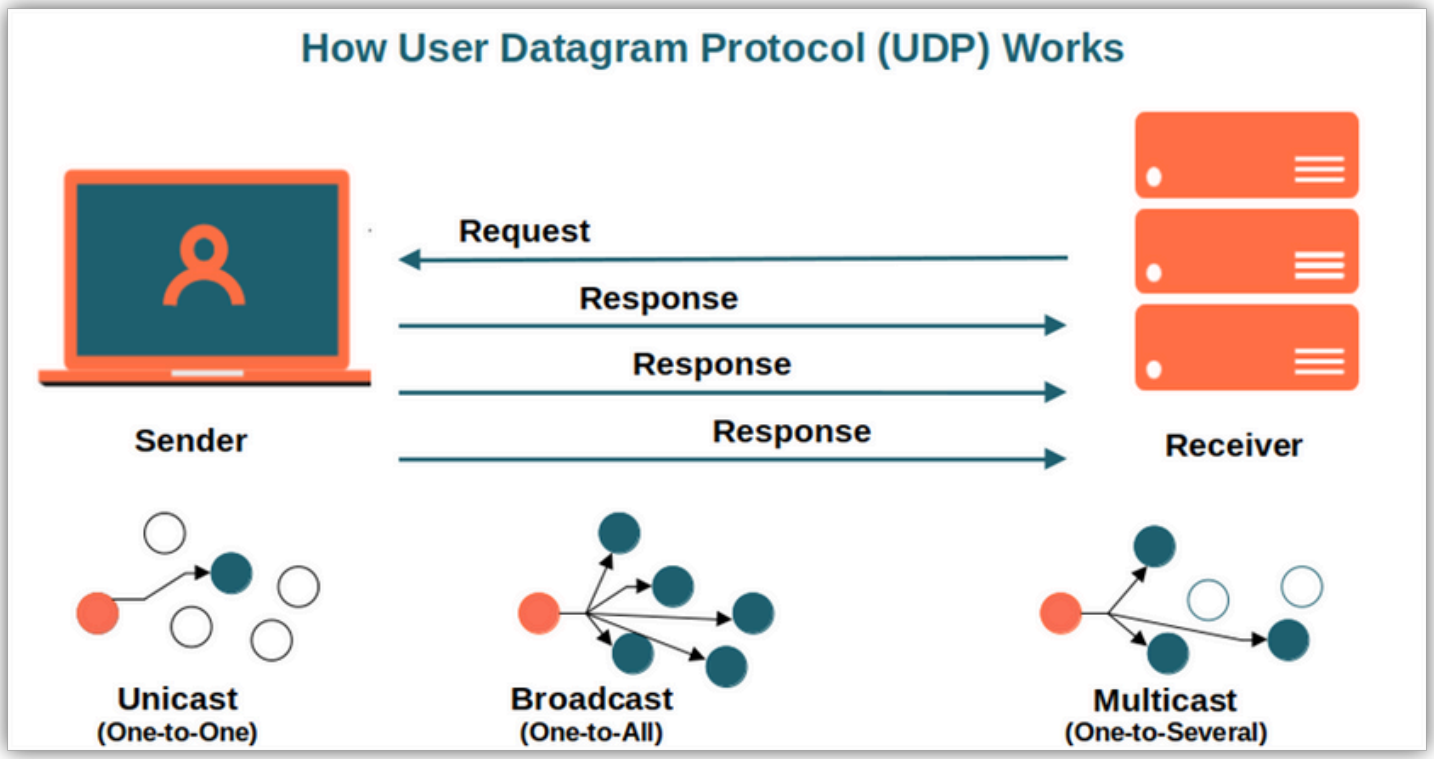
TCP/IP Handshake Diagram



2. Working of UDP (User Datagram Protocol)

1. **The sender creates datagrams (small data chunks).**
2. **Each datagram contains the destination address and data.**
3. **The datagrams are sent directly to the receiver without any setup or confirmation.**
4. **The receiver may get all, some, or none of the datagrams.**
5. **There is no acknowledgment, no error-checking, and no retry from the sender's side.**

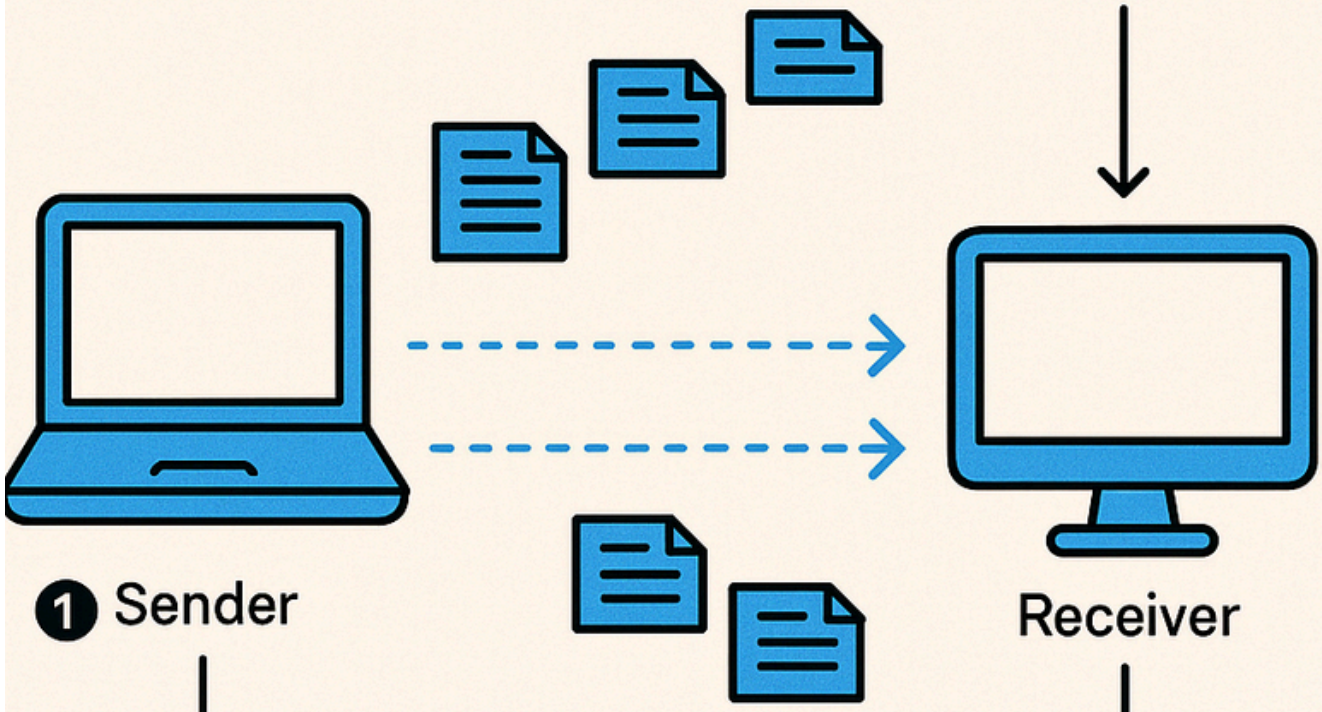
UDP sends data as fast as possible and doesn't wait to see if the data was received.



Working of UDP
(User Datagram Protocool)

1. The sender creates datagrams (small data chunks).

4. The receiver may get all, some, or none of the datagrams.



1 Sender

Receiver

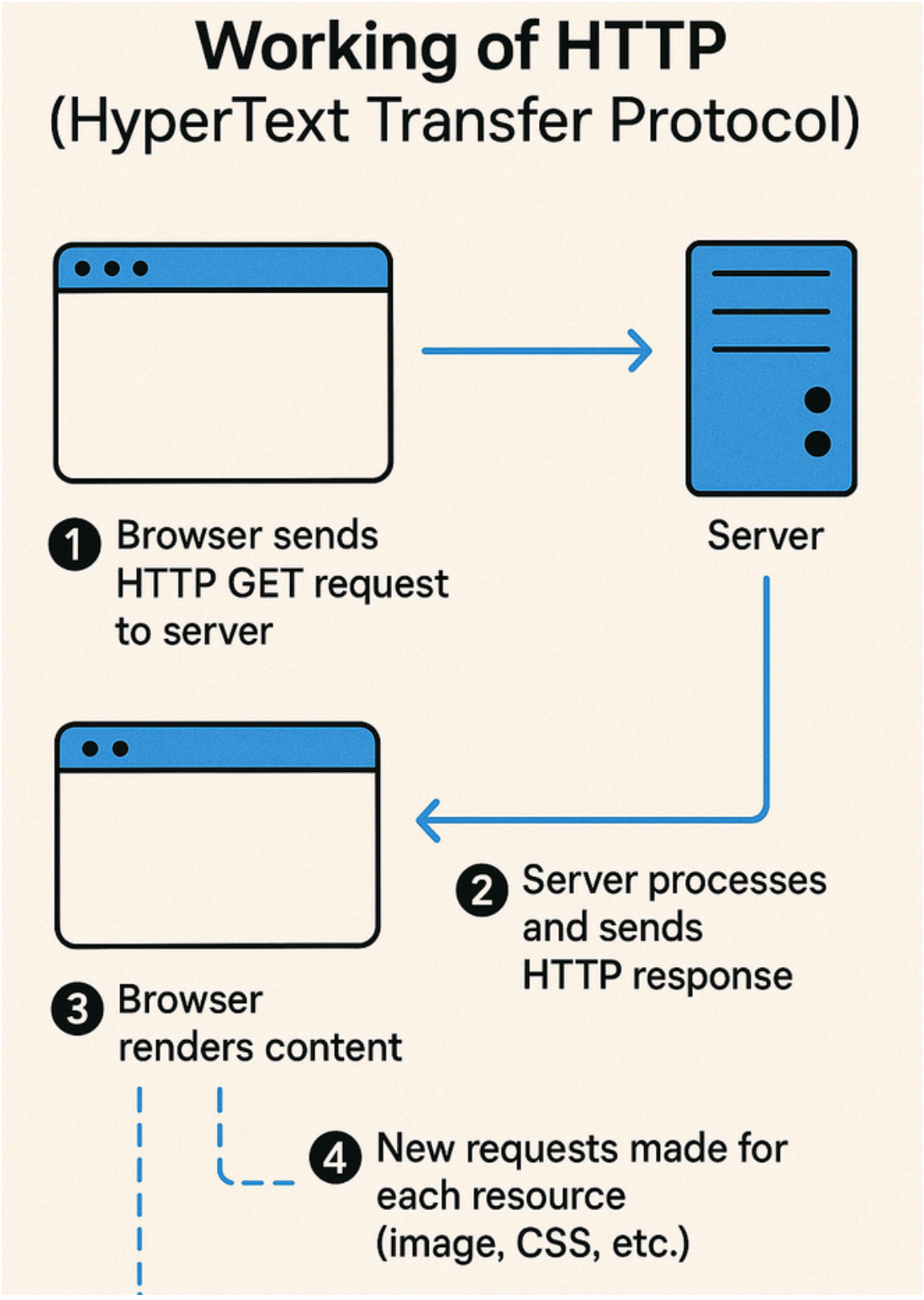
2. Each datagram contains the destination address and data.

3. There ifs no acknowledgement, no error-checking, and no retry from the sender's side.

5. UDP sends data as fast as possible and doesn't wait to see if the data was received.

3. Working of HTTP (HyperText Transfer Protocol)

1. **Browser sends HTTP GET request to server.**
2. **Server processes and sends HTTP response.**
3. **Browser renders content.**
4. **New requests made for each resource (image, CSS, etc.).**



4.Working of HTTPS (Hypertext Transfer Protocol Secure)

A. Initial Contact:

Browser connects using https://

B. TLS/SSL Handshake:

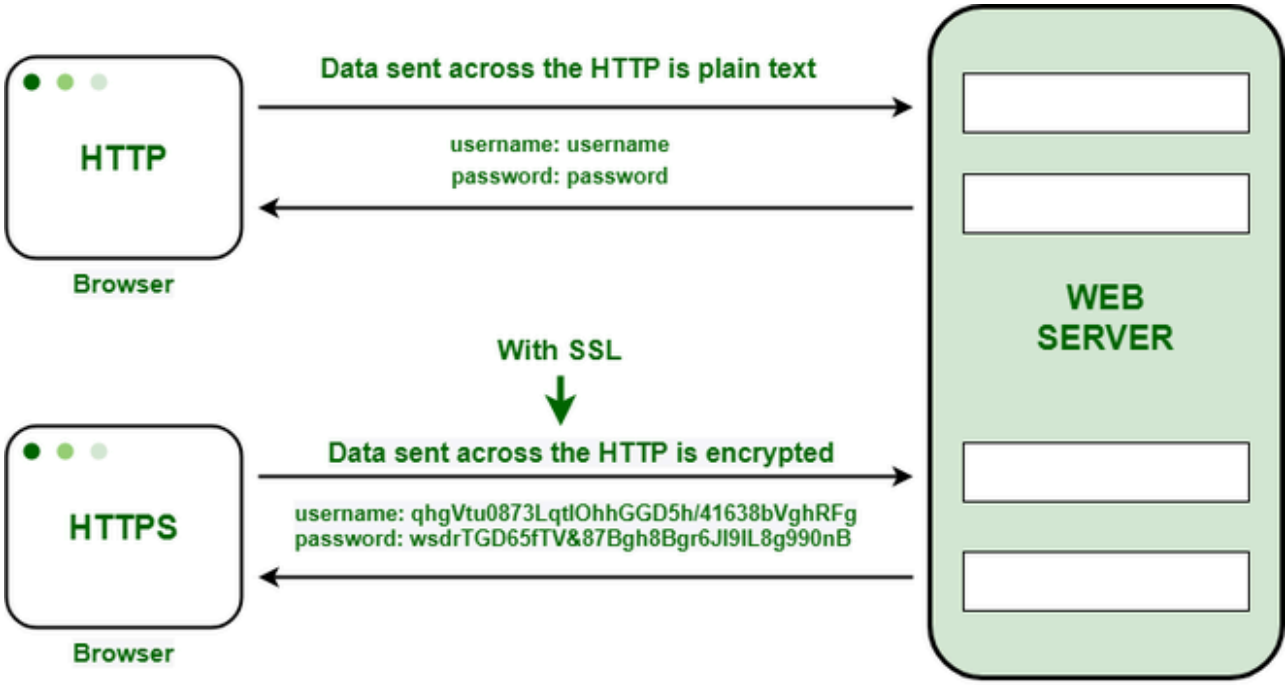
1. Server sends SSL certificate.
2. Client verifies certificate.
3. Client and server agree on encryption method.
4. Session key is generated.

This continues until all data is successfully received and acknowledged.

C. Secure Data Exchange:

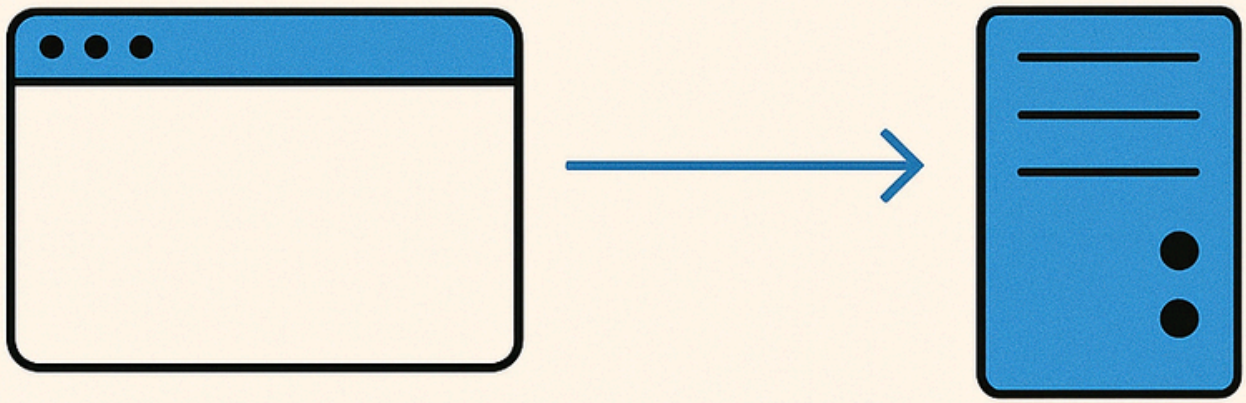
1. Client → Server: Sends a FIN message (finished sending).
2. Server → Client: Sends ACK (acknowledges termination request).
3. Server → Client: Then sends its own FIN message.
4. Client → Server: Sends final ACK.

HTTP requests and responses are encrypted and transmitted securely.



HTTPS

(HyperText Transfer Protocol Secure)



Server

1 Initial Contact

Browser connects using https.//

2 TLS/SSL Handshake

- Cerver vends SSL certificate
- Client verifies certificate
- Client and server agree on encryption method
- Session key is generated

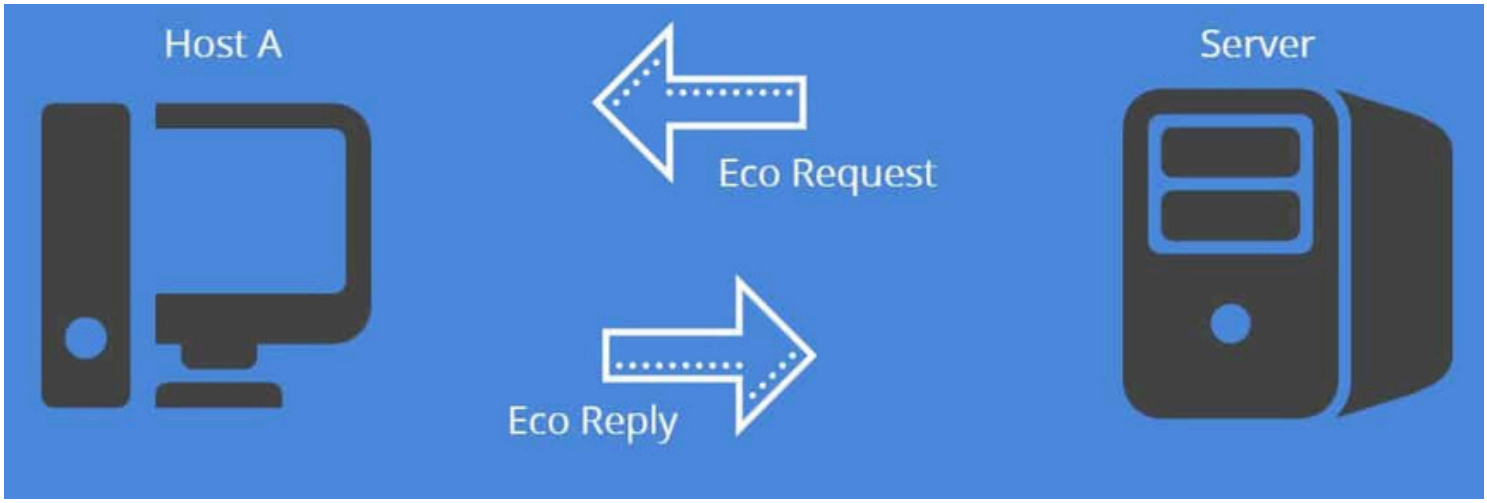
This continues until all data is successfully received and acknowledged.

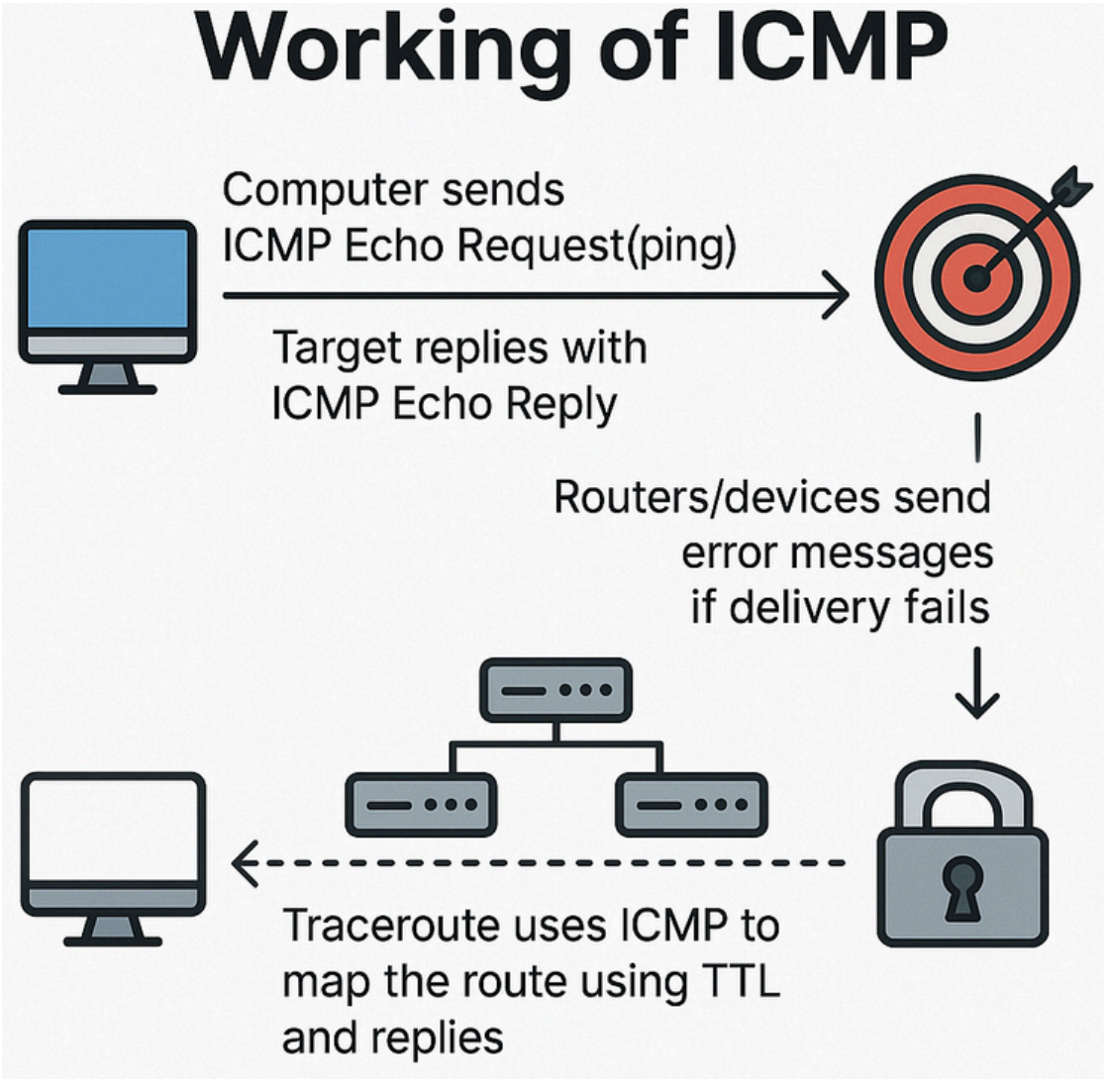
3 Secure Data Exchange

Client → Server: Sends a FIN message (finished sending)
Server → Client: Sends ACK (acknowledges termination request)
Server → Client: Then sends its own FIN message
Client → Server: Sends final ACK

5. Working of ICMP (Internet Control Message Protocol)

1. **Computer sends ICMP Echo Request (ping).**
2. **Target replies with ICMP Echo Reply.**
3. **Routers/devices send error messages if delivery fails.**
4. **Traceroute uses ICMP to map the route using TTL and replies.**





Conclusion

Each protocol examined—TCP, UDP, HTTP, HTTPS, and ICMP—has a distinct operational mechanism suited to specific tasks:

TCP establishes a reliable connection using a three-way handshake, ensures ordered data transfer with acknowledgments, and terminates via a structured four-step process.

UDP sends data without establishing a connection, prioritizing speed by omitting acknowledgments and retries.

HTTP follows a request-response model where each interaction is stateless and independent, transferring data in plain text.

HTTPS extends HTTP by initiating a secure handshake to establish encryption before any data exchange, ensuring confidentiality.

ICMP operates by sending control messages—such as echo requests and error notifications—to report on the status and path of data delivery.

Together, these working processes illustrate how data is transmitted, secured, or diagnosed at different layers of network communication.

References

https://www.khanacademy.org/computing/computers-andinternet/xcae6f4a7ff015e7d:the-internet/xcae6f4a7ff015e7d:transporting-packets/a/transmission-control-protocol--tcp

https://datatracker.ietf.org/doc/html/rfc793

https://datatracker.ietf.org/doc/html/rfc768

https://datatracker.ietf.org/doc/html/rfc2616

https://datatracker.ietf.org/doc/html/rfc2818#page-6

https://datatracker.ietf.org/doc/html/rfc792