4. Application Layer

28 January 2025 08:49

Client - Server Paradigm

Server
- Always on
- Permanent IP address
- Often in data centres for scaling

Client
- May have dynamic IP addresses
- Never communicate directly with other clients
- Communicates directly w/ server

Peer - to - Peer Architecture

- No always on server
- Arbitrary end systems directly communicate

Self scalability

New peers → More service capacity
New peers → More service demands

- IP addresses change; devices intermittently connected

Processes

what → program running in a host

type → Client processes: initiates communication; Server processes: waits to be contacted

communication → within the same host: inter-process communication defined by OS; different hosts: sending messages

Socket

Packet sends / receives messages to/from corresponding socket

→ analogous to door
- sending process = throw message out of door
- two doors, one on each side

Addressing processes

IP address of host + port number → identifier for process

Note: Ephemeral / Universal port numbers

HTTP : 80
Mail : 25

Application Layer Protocols

- types of messages exchanged: request / response
- message syntax: what fields
- message semantics: meaning of fields
- rules for when and how processes can respond

Open protocols
- Defined in RFCs, everyone has access to definition

respond

Open protocols

- Defined in RFCs, everyone has access to definition
- Allows for interoperability

Eg: HTTP, SMTP

Proprietary protocols

Eg: Skype

> **NOTE: RFCs**
> Request For Comments → formal document from Internet Engineering Task Force (IETF) containing specifications and organisational notes

TRANSPORT SERVICES NEEDED BY APPLICATIONS

- **data integrity**: some need 100% reliable transfer, others loss tolerant
- **timing**: low delay needed for online games etc.
- **throughput**: elastic app → use whatever throughput available
- **security**: encryption, data security

Application	Data Loss	Throughput	Time Sensitive?
File transfer/download	No loss	Elastic	No
E-Mail	No Loss	Elastic	No
Web Documents	Loss-tolerant	Audio: 5Kbps - 1Mbps Video: 10Kbps - 5Mbps	Yes 10's msec
Real-time audio/video	Loss-tolerant	Same as above	Yes few secs
Streaming Audio/Video	Loss-tolerant	Kbps+	Yes 10's msec
Interactive Games	Loss-tolerant	Elastic	Yes and no

application	application layer protocol	transport protocol
file transfer/download	FTP [RFC 959]	TCP
e-mail	SMTP [RFC 5321]	TCP
Web documents	HTTP 1.1 [RFC 7320]	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary (Skype)	TCP or UDP
streaming audio/video	DASH	TCP
interactive games	WOW, FPS (proprietary)	UDP or TCP

- update/notify mechanisms proposed IETF standard
- RFC 2136

HTTP specifications [RFC 1945; RFC 2616; RFC 7540]

HTTP

- stateless: server maintains no information about past client requests
- uses TCP

client initiates → server accepts → messages → connection closed

> **NOTE: Complexity of stateful connections**
> - Past history needs to be maintained for a huge number of devices

for a huge number of devices

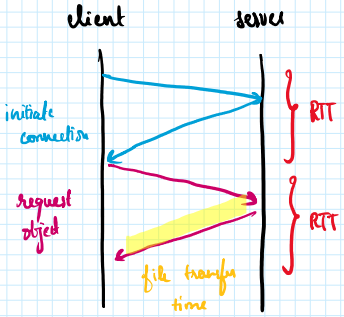
- Server/client crash → views of state become inconsistent → needs to be reconciled

messages exchanged → connection closed

- persistent HTTP / non-persistent HTTP

Non-persistent HTTP

Every object → separate connection



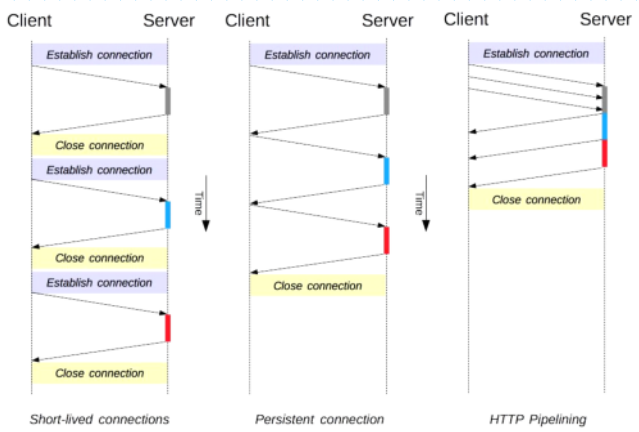
Delay for non-persistent = 2 RTT + file transfer time for each object

- Requires OS overhead (TCP buffer + variables) for every connection

Persistent HTTP (HTTP 1.1)

Many objects → same connection

Delay → as little as 1 RTT + file transfer for all objects (pipelining)



HTTP Requests & Responses

Requests

GET url HTTP 1.1 \r\n

- POST
- HEAD
- GET w/ user input
- PUT (explores file)

w/ ?

Response

HTTP 1.1 200 OK \r\n

301 → moved permanently

301 → moved permanently
505 → wrong HTTP version
404 → not found
400 → Bad request

HTTPS

- HTTP + TLS (SSL) → Secure Socket Layer → port 443

Transport Layer Security

- Public key → encryption
  Private key → decryption

SSL certificate → verified owner → 3rd party Certificate Authority (CA)
↳ links public key with identity of owner

SSL Process

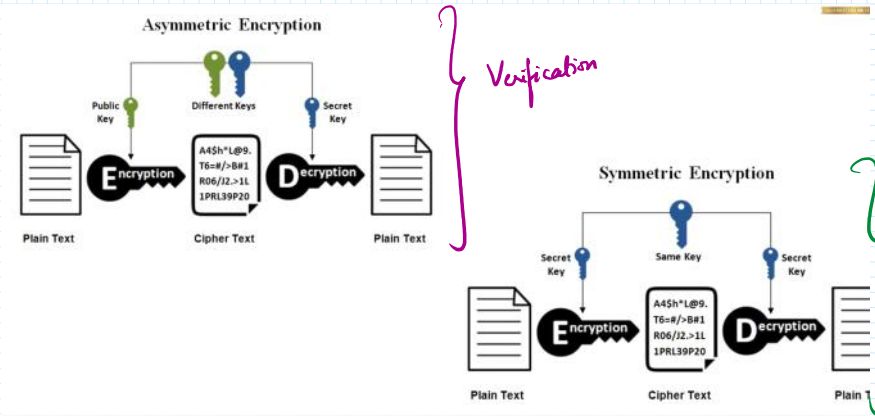
Client requests secure pages → Server sends public key & digital certificate → Client verifies certificate → once verified → Client creates shared/symmetric key. Encrypts it with server public key and sends to server

↓

Server decrypts using its own private key to get shared key → Symmetric key then used to encrypt & decrypt

TSL Process

Built on TCP/IP → 3 way handshake has to be done first



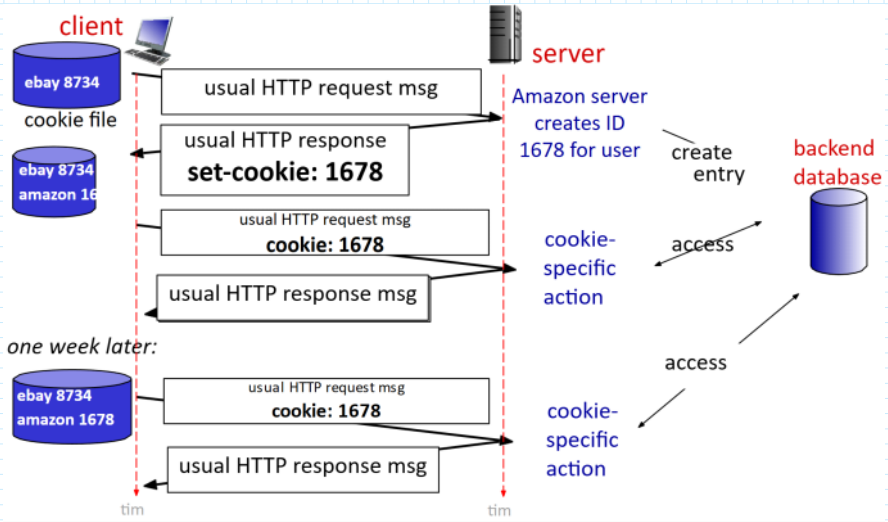
MAIN DIFFERENCE TO SSL: Server & client authentication + 3 way handshake

Feature	SSL	TLS
Stands For	Secure Sockets Layer	Transport Layer Security
Purpose	To provide secure communication over the internet	To provide secure communication over the internet, replacing SSL
Version	SSL 3.0	TLS 1.0 and higher
Encryption Strength	40-bit and 128-bit encryption	Up to 256-bit encryption
Authentication	Server-only authentication	Server and client authentication
Handshake	Two-step handshake process	Three-step handshake process

Web sites and client browser use *cookies* to maintain some state between transactions

four components:

1) cookie header line of HTTP *response* message
2) cookie header line in next HTTP *request* message
3) cookie file kept on user's host, managed by user's browser
4) back-end database at Web site



Web cache: exists

Conditional Get

Goal: don't send object if cache has up-to-date cached version

- no object transmission delay
- lower link utilization

- *cache:* specify date of cached copy in HTTP request
  If-modified-since: <date>

- *server:* response contains no object if cached copy is up-to-date:
  HTTP/1.0 304 Not Modified