GET /path/to/resource HTTP/1.1 → Request Line

Host: www.example.com
User-Agent:
Accept:
} Headers

It is not recommended to have body in GET

POST /subject-form HTTP/1.1 → Request Line

Host: www.example.com
Content-type: application/x-www-form-urlencoded
Content-length: 80
} Headers

— Empty line

username=x&subject=y
} Body

Status code:
2xx - success
300 - redirection
4xx - error on client side
5xx - error on server side

GET - Retrieve Data
→ No effect on Server
POST - Create new resource/submit data
PUT - updating
PATCH - (not operation) partial modification
DELETE - Delete the resource

Idempotency

Use this Idempotency logic to tell client which HTTP Method to use.

HTTPS

↓

Secure version of HTTP

Encryption

Symmetric
- same Password
- data fetch

Asymmetric
- Encryption

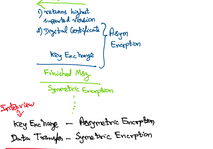
Encryption (Secret key) — decryption — Encryption (Public key) — Decryption

Public → Private

Encrypt — Decrypt

No one can generate publicky from private key
Only private key can decrypt public key
Mathematically both public & private keys are generated at the same time

Asymmetric Encryption
First we will do Asym Encryption
Symmetric exchange Common key
Then everything happens with Sym Encryp

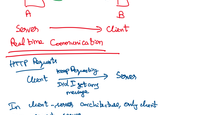


Symmetric Encryption

Subjective

Key Exchange — Asymmetric Encryption
Data Transfer — Symmetric Encryption

WebSocket/Channel



Real time Communication

HTTP Request

Client — Send Request / Send Response → Server

In client-server architecture, only client can request server

This is called Short Polling as client is asking in regular Short Intervals

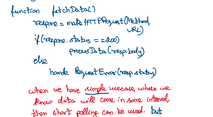
Short Polling

→ Client will keep on requesting server on Regular Intervals
→ These are dull HTTP requests
→ So Not Efficient
→ Response is unnecessary
→ High network traffic
→ Increase server load
→ Not real time, depends on intervals

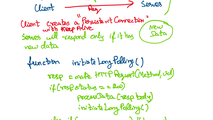
```
function initialShortPolling()
    forever loop:
    {  - fetchData()
       - wait (5 seconds)
    }

function fetchData()
    response = make HTTP Request(Method, URL)
    if(response.status == 200)
        processData(response.body)
    else
        handle Request Error(response.status)
```

when we have simple usecase, where we fetch data will come in some interval, then short polling can be used. but not for complex usecase like whatsapp & discord

Now lets see Real time LONG POLLING




```
function initialLongPolling()
    resp = make HTTP Request(Method, URL)
    if(resp.status == 200)
        processData(resp.body)
        initialLongPolling()
    else if(                    ) // timeout
        initialLongPolling()
    else
        handleRequestError(resp.status)
```



keep-alive — HTTP/1.1 → persistent connection

Client creates persistent connection with server
Server will send data only if receives new data
Immediately Client sends second request to followup

In interview, don't jump directly to 3rd/4th solution. Go from 1st to 4th

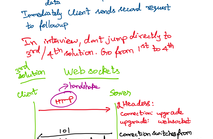
3rd solution — Web sockets

Useful in stockmarket prices

web socket will be one to most intervw
→ Full Duplex Communication / Simplex TCP
→ Bidirectional Communication / Connection
→ Key messages are called Frames
→ Client will be same as TCP & may close
→ Binary / Text support

4th Solution — SSE

Servers Sent Events



web socket is more important in interview

1) Short polling — Not realtime

2) Long polling

(No) 3) Web sockets — ws, wss, net/tcp, Bidirectional, binary/text message

4) SSE - Servers Sent Events — unidirectional

These 4 solutions solve issues to client Real Time Communication

Monolith — Microservices

Monolith
→ Single codebase (ease regression testing)
→ Development
→ Testing
→ Deployment
→ Debug
→ Performance