Date : 06/05/2021

High Level System Design :

(AlgoExpert ie. SystemExpert)

content :

22 Fundamental Lectures (Algoexpert)

Resources: ① Grokking the system Design (Educative.io) pdf

② System Design Videos (clement)

③ Gaurav Sen

④ Tech Dummies

⑤ codeKode (Yogita's channel) → System Design Primer Course

⑥ GitHub link → donnemartin (System Design Primer)
checkcheck33 (System Design Interviews)
shashank88

Other Top Youtube channels for HLD:

① Codekarle
② Think software
③ Udit Agarwal
④ Success in Tech
⑤ Jackson Gabbard
⑥ Coding simplified
⑦ System Design Interview

Blogs and websites:

highscalability.com
Interviewbit → system Design
Gaurav Sen Answer (Quora)

Syllabus

Main Topics of System Design:

Grokking System Design (Educative.io)

- System Design Basics
- Key characteristics of Distributed systems
- Load Balancing
- Caching
- Data Partitioning
- Indexes
- Proxies
- Redundancy and Replication
- SQL vs NoSQL
- CAP Theorem
- Consistent Hashing
- Long Polling vs Websockets vs Server-Sent Events

SystemExpert (clement)

- Client-server model
- Network Protocols
- Storage
- Latency and Throughput
- Availability
- Hashing
- Relational Databases
- Key-value stores
- Specialized storage Paradigms
- Replication & sharding
- Leader Election
- Peer to peer networks
- Polling & streaming
- Configuration
- Rate Limiting
- Logging and Monitoring
- Publish/Subscribe Pattern
- MapReduce
- Security and Https
- API Design

content of this copy

System Design Problems

educative.io

1. Step by step guide
2. Designing a URL shortening service like TinyURL
3. Designing Pastebin
4. " Instagram
5. " Dropbox
6. " Facebook Messenger
7. " Twitter
8. " Youtube or Netflix
9. " Typeahead suggestion
10. " an API Rate Limiter
11. " Twitter search
12. " a web crawler
13. " Facebook's Newsfeed
14. " Yelp or Nearby Friends
15. " Uber Backend
16. " Ticketmaster
17. Design AlgoExpert

systemExpert (clement)

18. Design a Stockbroker
19. " Google Drive
20. " Reddit API
21. " Netflix
22. " Uber API
23. " Slack
24. " Airbnb
25. " a code deployment system

System Design Fundamentals :

System Expert (By clement - Algoexpert)

System Design Basics

How to build robust, functional and scalable systems?

→ Design fundamentals are pre-requisites for any system design interview in the same way just like the knowledge of data structure & algorithms is the pre-requisite for any coding interview.

→ Following are some of the design fundamentals that you need to know:

SQL	Load Balancer	Availability	client
server	Leader Election	Proxies	HTTP
cache	Peer-to-peer	Nginx	Database
Polling		mapReduce	Hashing
			Replication

Design Fundamentals:

→ underlying / Foundational knowledge of System Design

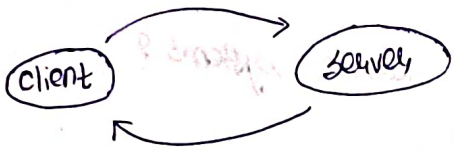
→ key characteristics of system

→ Actual components of system Eg: Load Balancer, Proxies, etc

→ Actual tech

Eg: zookeeper, Amazon S3 etc.

Client - server model



A client is something that speaks to server, whereas server is something that listens to client to speak to it and then speaks back to those clients.

A client sends data or request data to a server.

Eg: when you typed algoexpert.com in your favorite browser, than Browser will act as ~~server~~ client & algoexpert will act as server.

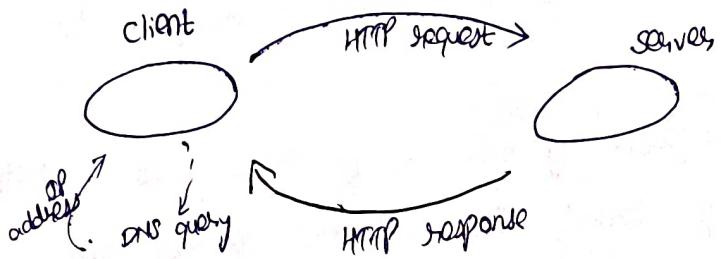
→ Browser will first do DNS query to first know the IP address of algoexpert.com
↓ to know the IP address of any URL.
↓ It is just the unique identifier of the machine.

& then the browser will send the http request to this IP address.
↓ It is a way to send information that your server is able to understand

When we say your browser has sent the http request to algoexpert servers, it means it sends a bunch of bytes or characters that were packed in packets in some special format and that will be sent to over Algoexpert servers.

This request will also contain IP address of the browser, which is also called the source address of the request

So, when a server gets this request, it will know to what IP address it needs to respond back to.



Eg: Browser

Algoexpert.com

→ A server usually listens 4 requests in specific ports :-

→ Any machine that has a distinct IP address has 16,000 ports that program on the machine can listen to.

→ So, when you are communicating with other machine, you actually have to specify to it what port you want to communicate on.

→ So, as a client, you have to specify the port that you want to communicate on.

Eg: Consider ~~mailbox~~ IP address as mailbox & port no as the apartment in a society
mailbox needs to route to.

⇒ For HTTP → Port 80

HTTPS → port 443

→ Now, the server sends the response to client (browser) → letter pay in form of (Algoexpert)

html of algoexpert and your browser receive the html response and will render the page for you.

Network Protocols :

IP
TCP
HTTP
UDP

Protocol : It is some set of rules or procedures for transmitting data b/w computers or electronic devices.

Network Protocol : It consists of the types of messages that are gonna sent over the network, the format of these messages how they are structured, the order of these messages if they have an order, and ~~whether~~ etc.

Eg: IP, TCP, HTTP, etc

ⓐ **IP :** It stands for "Internet Protocol".

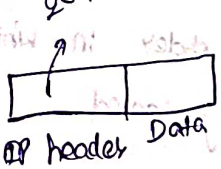
The modern internet runs following Internet Protocol ie IP.

→ **IP packet :** It is a fundamental unit of data that is sent from one machine to another.

IP packets are building blocks of communications b/w machines over the network. → made up of bytes

→ IP packet has 2 main sections :

- IP header
  - It is at beginning of packet.
  - → It contains the source IP address of the packet, destination IP address, total size of packet
- Data



IP packet

IP

→ Header also contains the ~~header~~ version of the Internet protocol that this IP packet is operating by.

→ There are multiple versions of IP., and today, there are 2 versions of IP ⇒

IPv4 IPv6

Based on IP version, the packet might look a little different.

→ Header is very ~~slow~~ small, anywhere bet 20 and 60 bytes.

Data: It contains the information which one machine is trying to sent to other machine over an IP packet.

IP packets are limited in size ie only 2^{16} bytes i.e. 0.055MB i.e. 65000 bytes

→ obviously, your data would be > than 2^{16} byes so your data would might fit it in multiple IP packets.

→ When we have multiple IP packets that is sent from one machine to another and if you are using Internet protocol then you don't actually have a way to guarantee that these packets are actually gonna received. It is quite possible that some of these packets would be lost over the network.

→ Also, the order in which those packets can be read & interpreted can't be guaranted.

→ So, for these disadvantages, IP falls apart. This is where TCP comes into play.

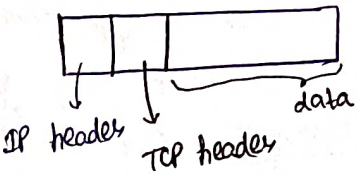
TCP : It stands for "Transmission Control Protocol".

→ It is built on top of Internet Protocol and is meant to solve the issues that the Internet Protocol were having.

→ It is meant to send the IP packets in an ordered way meaning that you are guaranting the order in which those IP packets will be read by the destination.

→ You will know if some packets gets failed in getting received in this TCP protocol, and in a error-free way.

→ TCP is used in all web applications and it allows you to send arbitrary long pieces of data to other machine.



→ TCP header contains the information which ensures that multiple packets should be sent in an ordered fashion.

→ Let's say :
If your browser wants to communicate with a website (Eg: algoexpert.io) servers, it first going to create a TCP connection with a destination computer with a destination server. And, it happens through handshake mechanism.

Handshake :
A handshake is a special TCP interaction where one computer basically contacts the other by sending the packets saying 'Hey, I wanna connect with you'. The other computer responds & says " We can connect". And then the client ie. the machine who is trying to establish the connection re-responds again and says " We have got a open connection".

→ Once a connection is established, both machines can freely send data to one another.

If one of the machine doesn't send data in a given period, the connection can be timed out.

Or if one of the machine wants to end the connection for whatever reason, it can do so by sending some kind of special message to let the other one know that "Hey, I'm gonna end the connection".

→ TCP lacks a really robust framework that developers can use to really define meaningful and easy to use communication channel for clients and servers in a system.

Because in TCP, all you are sending is arbitrary data that fits in underlying IP packets.

This is where HTTP comes into play.

HTTP : It stands for "HyperText Transfer Protocol".

It is built on top of TCP. It uses request-response paradigm where one machine sends a request to other machine and the other machine returns a response to the 1st machine.

This request-response paradigm with a company's set of rules makes it really easy for developers to create a robust and easy to maintain system.

→ This is why, most modern day system rely on http protocol for communication.

Here, we deal with http request and response.

Eg of http request and response →

```
const httpRequest = {
    host: 'localhost',
    port: 8080
    method: 'POST',
    path: '/payments'
    headers: {
        ...
    }
    body: ...
}

const httpResponse = {
    statusCode: 200,
    headers: {
        ...
    },
    body: '...'
}
```

Tries to describe the destination servers

describe purpose of request

// GET, PUT, DELETE, etc
- GET → retrieve data from server
- PUT → for providing data to the server
- DELETE → asking the server to delete some data

This could mean that you're making an actual payment.

It describes the type of response it is.

They are collection of key-value pairs that contain important metadata or info about request.

→ A server can have multiple paths for different services and the clients are issuing requests to these various paths

And depending on the path, different business logic is gonna occur.

→ curl command is used to send data to servers and received data from servers.

→ Why http is so useful ?

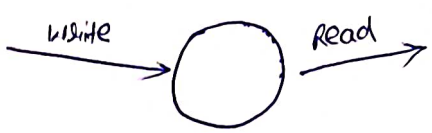
Soln :

IP and TCP protocols were more just for transportation of data.

But, http on the other hand introduces the opportunity to add a lot of business logic in your system.

Storage :

A database is used to → store data
→ retrieve data



→ A database is just a server.

Disk — Persistence — Memory

Persistence of database through server crashes and power outage

→ If you have a database server, and that database writes data to the disk, that data persists even if the database server goes down.

Writing data to disk is basically saving file on your computer.

→ If you shut down your computer or if your computer crashes, that file which you saved is still gonna be there, unless there is some catastrophic issue with the hardware.

→ In contrast, if your database writes data to memory, and if your database server goes down and then it booted back up, the array or hash table in which you might have stored data is no longer have that data.
(Eg: storing data in an array or hash table)

→ ~~The~~ If you are wondering why to store data in memory instead of disk ?

The simple reason is that reading data from memory / writing data to memory } are much faster than { reading from disk / writing to disk

→ So, if you store data in disk, that data is going to persist through a ~~outage~~ outage or issue that your system might go through.

But if you store data in memory, then that data in memory will not gonna persist if your server goes down.

→ Eg:

Google cloud Platform alone offers 8 different storage products.