CodinGurukul />

Follow 304 Followers

WhatsApp-Engineering Inside-2

 Suraj Kumar Jan 27, 2019 · 4 min read

In “WhatsApp-Engineering Inside-1” we talked about the system and network architecture of WhatsApp, in this article we are going to deep dive into the messaging server and other components.

One thing which needs to be known is that the connection is always initiated by the client because the server does not know the address of the client but the client knows the address of the server.

How *Sent, Delivered* and *Seen* Works?

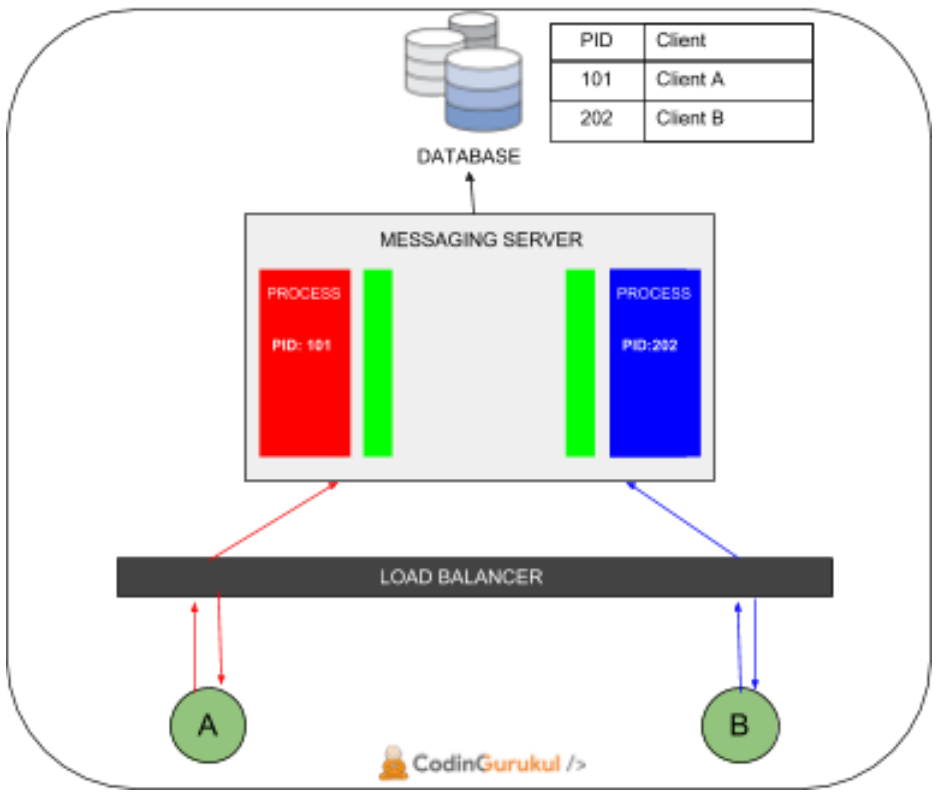
WhatsApp message status symbols

Delivered: When the message is delivered to the receiver from the Whatsapp server.

Seen: When the message is seen/opened by the receiver.

To Incorporate all these status changes, every message has a unique ID to identify each message and acknowledgement from the various events (sent / delivered /seen).

What happens inside the Whatsapp server when a client connects to the server?



When a client connects to the WhatsApp server, a process (or thread) is created with respect to that client. This process is responsible for handling all the operations related to that client.

With every process, a queue(Highlighted with light green colour) is associated which act as a buffer for that process. After process creation, a table is created in the database to maintain the record of PID(Process ID) and the associated Client.

How Last Seen Work?

Implementation of this feature is very simple and straightforward, It is just about maintaining a record with Client ID and Timestamp.

client_ID	last_seen
Client A_ID	2019-01-27 13:15:25
Client B_ID	2019-01-26 12:27:45

A table containing client id and their last seen status

When we open Whatsapp in our smartphone, our application sends a pulse to server every 5 seconds, and with every pulse last seen time is updated in the table. As the client disconnects the last seen time exists in the record that is updated by the last pulse sent before closing the app.

How the media sharing works?

Instead, WhatsApp uses a different server(like HTTP) to share media.

System design with HTTP Server

When we share a media, it gets uploaded to an HTTP Server over a different connection, after successful upload, the HTTP server returns a hash or unique ID associated to that media and that hash value is sent to the WhatsApp server. At the receiver end, the same thing works in a reverse way, the receiver receives the hash value then it downloads the media from HTTP server associated to that hash value.

The Telephony services also work in the same way just like media services, for this, we also use a different server and use a different kind of connection like socket etc. for real-time communication.

This is all about the overview of a real-time messaging system.

Let's Talk about the actually Technology used by Whatsapp :)

-> Programming Language: Erlang

Erlang is a super fast programming language which supports features like Hot Reload/Update on Fly etc. It also has a concept of the lightweight thread which makes it capable of handling millions of connections at a time. This is the reason Erlang is an ideal choice for WhatsApp.

In Actual, Whatsapp handles 10 million connection on a single server, which seems to be impossible but the WhatsApp team able to achieve this. And it is only possible if you know all the things about the system like Server kernel, networking library, infrastructure configuration etc.

-> Operating System on Servers: FreeBSD is the OS used by all the messaging servers of WhatsApp. As it is open source OS and the developer knows all the in and out. so that they can get maximum performance out of it.

-> Database: AMNESIA is the database which is used for storing data, it is also a key-value pair based DB which couples really good with Erlang.

So, This is all about the engineering behind a real-time messaging platform like WhatsApp.

I hope you enjoyed it, feel free to Share and Clap, In future, I will be writing this kind of articles only at Coding Gurukul

Happy Exploring!

WhatsApp Software Engineering Software Architecture System Architecture

Application Architecture