Архитектурата на WhatsApp

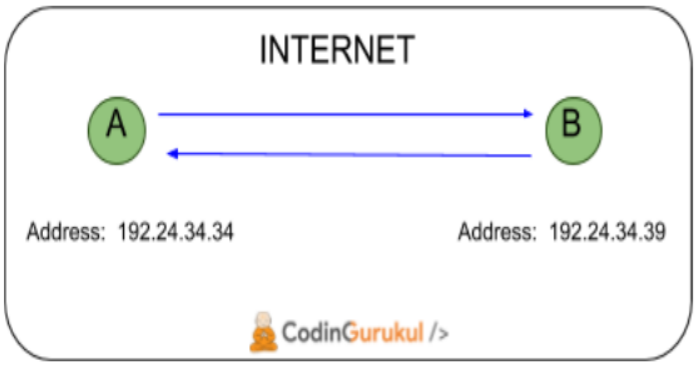
Комуникацията в реално време е една неизменна част от ежедневието на почти всеки човек. Едва ли са много хората обаче, които са се замисляли над това- как функционира приложението, което използват, какво стои зад “обвивката”, наречена потребителски интерфейс, благодарение на която без много усилия всеки бързо и лесно пише съобщения, разговаря с приятели и други.

В тази статия ще се опитам да представя проучването си върху архитектурата на приложението *WhatsApp* конкретно, но то със сигурност би могло да бъде съотнесено и към други приложения за комуникация между голям брой потребители.

Ще разгледаме функционирането на приложението *WhatsApp* поетапно, обяснявайки всяка стъпка, за да са ясни отделните опорни точки, върху които се крепи изграждането на архитектурата.

На първо място е важно да се разбере как се случва комуникацията между отделните потребители (клиенти, **CLIENTS**):

Когато двама клиенти (в случая на **фигура 1** това са A и B) искат да разговарят или да изпращат съобщения помежду си, това, което трябва да им е известно, е адресът на другия (той може да е IP, MAC или какъвто и да персонализиран уникален идентификатор). Така те обменят съобщения по мрежата, която в случая е Интернет.



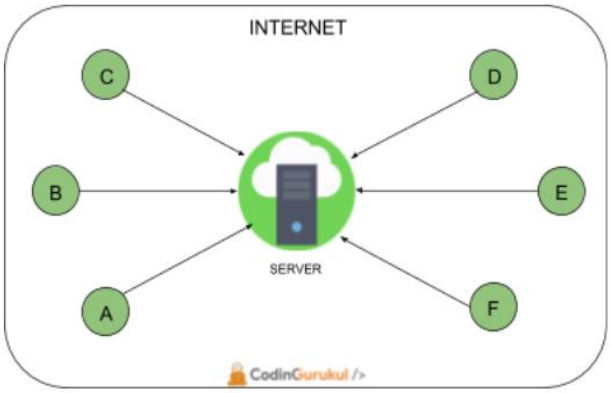
Фигура 1

Какво се случва обаче ако мрежата е много голяма и броят на клиентите не е двама или трима, ами милиони или дори повече?

В този случай възможността всеки клиент да знае адреса на всички останали клиенти в мрежата е граничеща с невъзможното. Тогава се налага да се включи допълнителен компонент между отделните клиенти, към който те се свързват и чиято цел е да направи системата високо достъпна и издръжлива на натоварване, както и да “координира” клиентите. Този компонент се нарича **SERVER**.

В този сценарий комуникацията се случва по следния начин (**фигура 2**):

Когато двама клиенти (A и B) искат да изпратят съобщение до друг клиент (например D), съобщението първо се изпраща до сървъра, който “държи” адресите на всички клиенти, и от своя страна сървърът препраща съобщението към клиентът получател (D) и обратно.

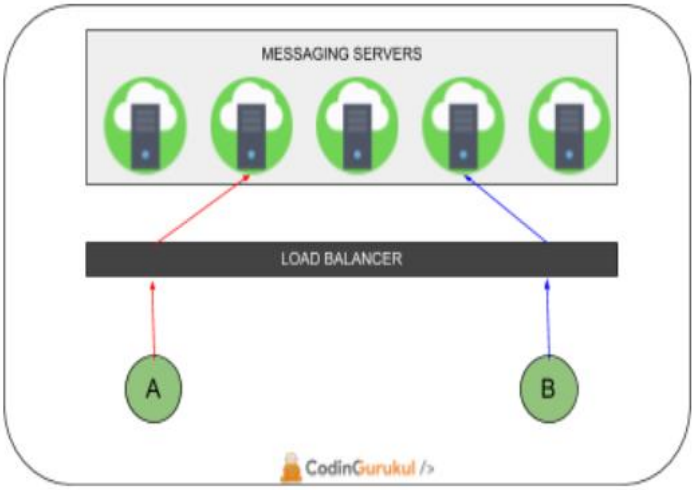


Фигура 2

Сега нека да преминем към самото изграждане на дизайна на системата.

Важно е да се отбележи, че е съществено да се уточни до каква степен ще се наложи на софтуерния пазар самото приложение. Тъй като както *WhatsApp*, така и други такива приложения за обмен на информация между потребители, са широко използвани, за да може да бъде овладян успешно големият трафик, вместо един се използват няколко сървъра. Естествен е въпросът, разбира се, как клиентът ще знае към кой сървър да се свърже, като той не може да бъде избран на случаен принцип.

(**Фигура 3**) И тук намира своето приложение още един софтуерен компонент, чиято сериозна задача е да разпределя натоварването равномерно към всички сървъри, за да може системата да оправдае заложените си качествени характеристики. Този компонент се нарича **LOAD BALANCER**. Благодарение на него системата е способна да се справи с огромна потребителска база. Сега, когато клиентът пожелае свързване към сървъра, неговата заявка първо се обработва от ***load balancer***-а, след което именно той я препраща към най-подходящия сървър, като го избира на база различни параметри, например ***load on individual servers.***



Фигура 3

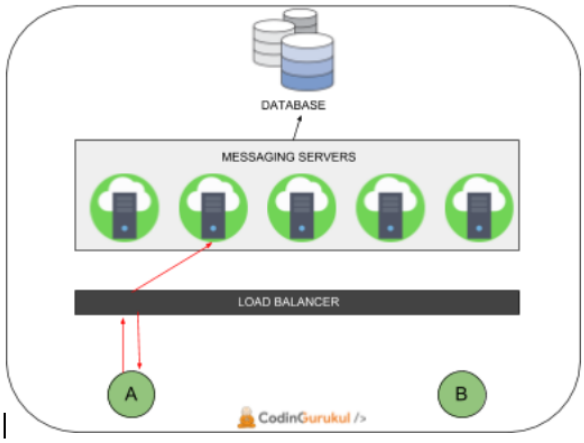
В процеса на изграждане на архитектурата става ясно, че важен детайл е нуждата от механизъм за съхранение на данните. За да се изпълни това изискване към системата, се добавя база от данни, която е достъпна за всички сървъри.

Естествено и тук се появява логичният въпрос какъв е видът на използваната връзка. Най-общо казано *WhatsApp* използва **DUPLEX Connection** (позната още **като BiDirectional Connection**), тъй като съобщенията биват генерирани не само от потребителите, но и от сървърите.

Преди да заключим как точно се случва свързването между клиентите и сървърите в *WhatsApp*, накратко ще разгледаме няколко различни сценария на свързването между потребителите чрез системата с цел придобиване на по-ясна представа и по-детайлно разбиране.

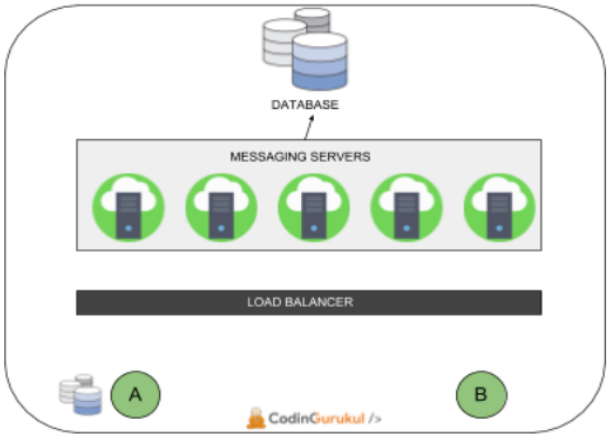
1. **Адресантът (A) е свързан към подходящ сървър от мрежата сървъри, но адресатът (B) не е. (Фигура 4)**

В този случай получателят (B) не е свързан към сървъра и затова съобщенията, които предавателят (A) изпраща се съхраняват в базата данни и когато адресатът (B) се свърже към сървъра, съобщението, пазено в базата, се “взима” от нея и се препраща към него (B).



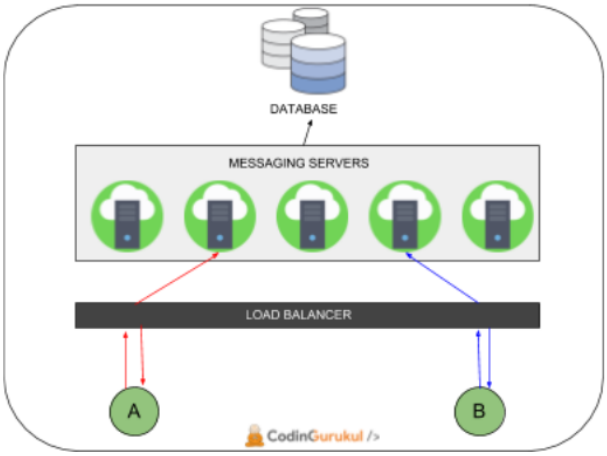
Фигура 4

2. **Нито адресантът (A), нито адресатът (B) са свързани към подходящ сървър от мрежата сървъри. (Фигура 5)**
   В този случай освен че адресатът (B) не е свързан към сървъра, адресантът (A) също не е. В такава ситуация съобщението, изпратено от предавателя (A) се съхранява в така наречената ***локална база данни*** (например ***SQLite***). Когато адресантът (A) премине към онлайн режим или се свърже към сървъра, съобщението е "взето" от локалната база данни е бива изпратено на сървъра, избран от ***load balancer***-а.



Фигура 5

3. **И адресантът (A), и адресатът (B) са свързани към подходящ сървър от мрежата сървъри. (Фигура 6)**
   В този случай, в който и адресантът (A), и адресатът (B) са се свързали към подходящ сървър от мрежата сървъри, адресантът (A) изпраща своето съобщение и избраният от ***load balancer***-а сървър прерпраща съобщението към адресата (B), без то да бъде съхранявано в отдалечената база данни или в локалната такава.

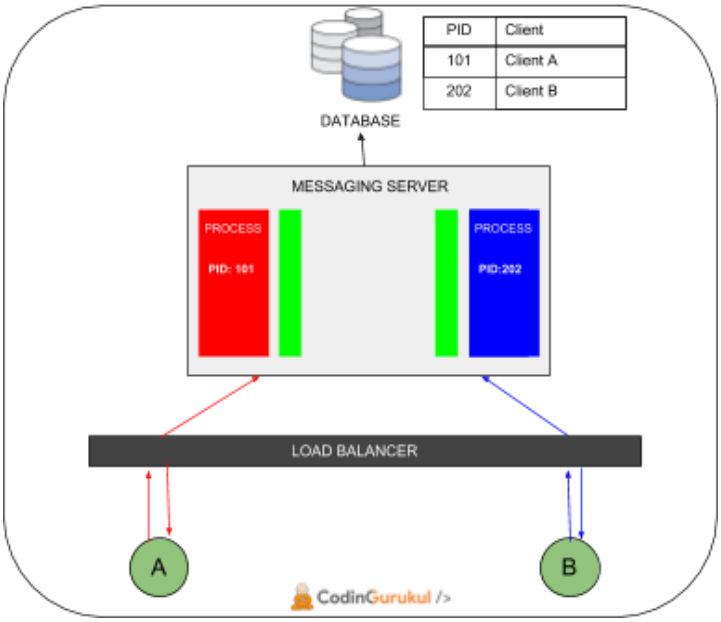


Фигура 6

Именно представената на **Фигура 6** структура по абстрактен начин най-точно и най-пълно описва архитектурата, стояща зад приложението WhatsApp. Според мен тя представлява ***Структура на употреба на модулите***, тъй като представя отделните софтуерни елементи като ***Database***, ***Messaging servers и Load balancer***, чиито функционалности подробно описахме дотук в статията, като модули на системата, които комуникират помежду си чрез предаване на съобщения, идващи от крайните потребители (В случая на **фигура 6** това са A и B, за които също така по-горе в статията обяснихме как се свързват към системата). На **фигура 6** ясно се вижда и кой модул кой друг използва с цел правилно изпълнение на своята функционалност както и до какво имат и нямат достъп крайните потребители.

В допълнение може да се каже че на **фигура 6** се наблюдава и един от доста широко използваните архитектурни стилове, а именно ***Client-Server***. Затова се надявам тази статия да послужи като препратка към по-задълбочено запознаване с темата за Архитектурните стилове и като основа за по-лесното им разбиране.

Фигура 7, от друга страна, може да се каже, че представя един малко по-задълбочен поглед върху това- как точно се обработват заявките от сървъра, избран ***от load balancer***-а. Според мен на нея виждаме и до известна степен ***Структура на процесите***, тъй като ясно се виждат процесите (още наричани нишки), които всъщност са елемнтите на една ***структура на процесите*** (обозначени на фигурата с червено и синьо респективно за адресанта и адресата), които се създават, за да отговарят за съобщенията от клиентите. (Тук искам да подчертая употребеното от мен "до известна степен", тъй като самата структура, представена на **фигура 7**, не показва толкова детайлно всички процеси,случващи се в системата).



Фигура 7

Други полезни структури, които биха могли да бъдат включени в описанието на архитектурата на приложението WhatsApp, според мен са:

-***Декомпозиция на модулите***- би било полезно да се визуализира съдържанието на всеки модул, с други думи явно да се покажат подмодулите на всеки модул, като това разбиване, естествено, се извършва до момента, в който получим съвсем прости и лесно разбираеми единици.
-***Файлова структура-*** поради естеството на работа на приложението *WhatsApp*, а именно изпращане, обработване, получаване на съобщения под различна форма (текстови, мултимедийни), тази структура би била полезна за онагледяване на това- по време на работа на приложението кой модул къде се помещава.

Източници:
За запознаване с архитектурата на *WhatsApp*, използвах:
https://medium.com/codingurukul/whatsapp-engineering-inside-1-1ef4845ff784
https://medium.com/codingurukul/whatsapp-engineering-inside-2-bdd1ec354748
За правилното определяне на използваните от източника структури, а също и какви други структури биха били полезни, използвах:
https://learn.fmi.uni-sofia.bg/pluginfile.php/241597/mod_resource/content/4/SARS_2016-17_T1_SA_intro.pdf

Изготвил:

Теодора Иванова, фак.№ 62250

Софтуерно инженерство, 2 курс, 1 група