GROUP NAME - TITANS.

MEMBERS

Ndekere kiritu - 17g01acs036.

Kelvin oduor - 17g01acs029.

Emmanuel kosgei - 17g01acs023.

Stephen mutual - 17g01acs028

Brian oyollo - 17g01acs030.

Arthur odera - 12m01abt070.

Samantha achieng - 17g01acs002.

APPLICATIONS OF DISTRIBUTED SYTEMS IN REAL LIFE

1) **MPESA**

a) *Structure*

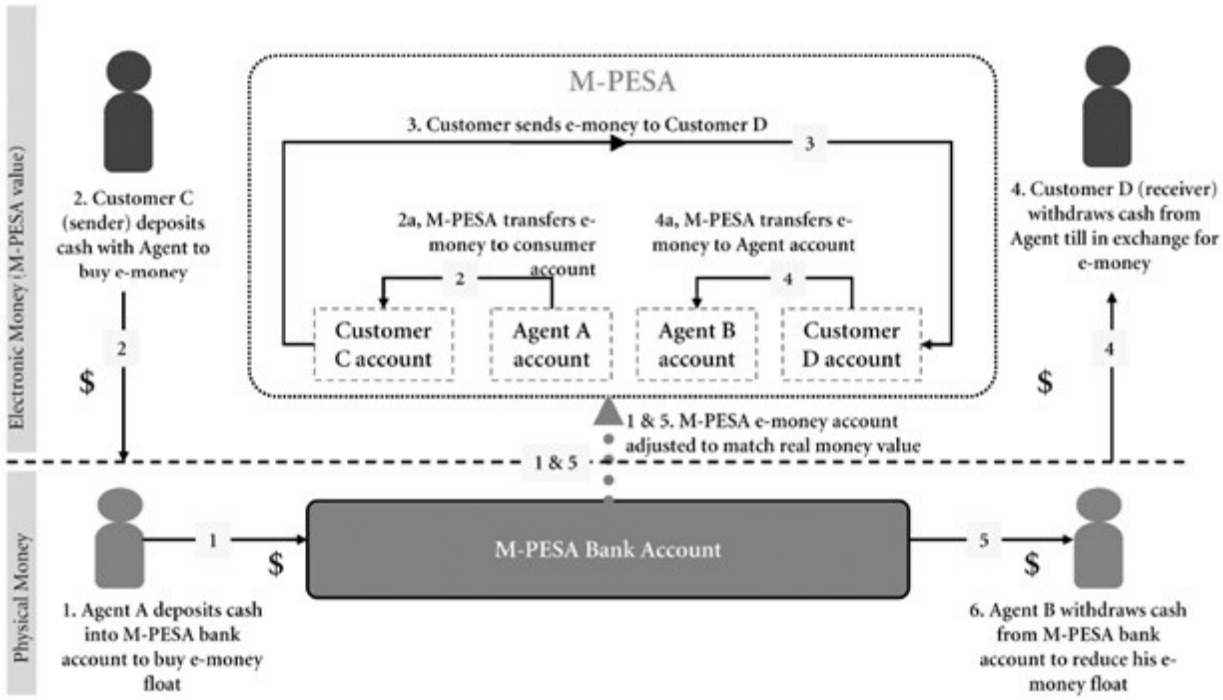


Figure 1.0

b) *How it works*

In order to start using M-PESA, all you need are a Safaricom mobile phone and a national ID card. As M-PESA is much more informal than traditional banking services, you do not need to go through tedious registration process as you would in a bank. Once you are registered for the service, you can visit local M-PESA agents. Once you deposit cash, you'll get cyber money called "e-float" in exchange. Then using SMS technology, you can exchange that "e-float" with another M-PESA subscriber. Or you can retrieve cash from the agent in exchange of "e-float" you have in your M-PESA account.

c) *What components does it use?*

M-PESA uses synchronous system to communicate between users from different geographical regions without delays and does not depend on a global clock, rather, it uses a logical clock.

d) *REFERRENCES*

i) Vodafone to shut down M-Pesa in Albania - Capital Business. *Capital Business*. 5 July 2017. Retrieved 13 August 2018.

ii) *Safaricom M-Ledger - Android Apps on Google Play". play.google.com. Retrieved 23 August 2017.*

2) **GOOGLE**

a) *Structure*

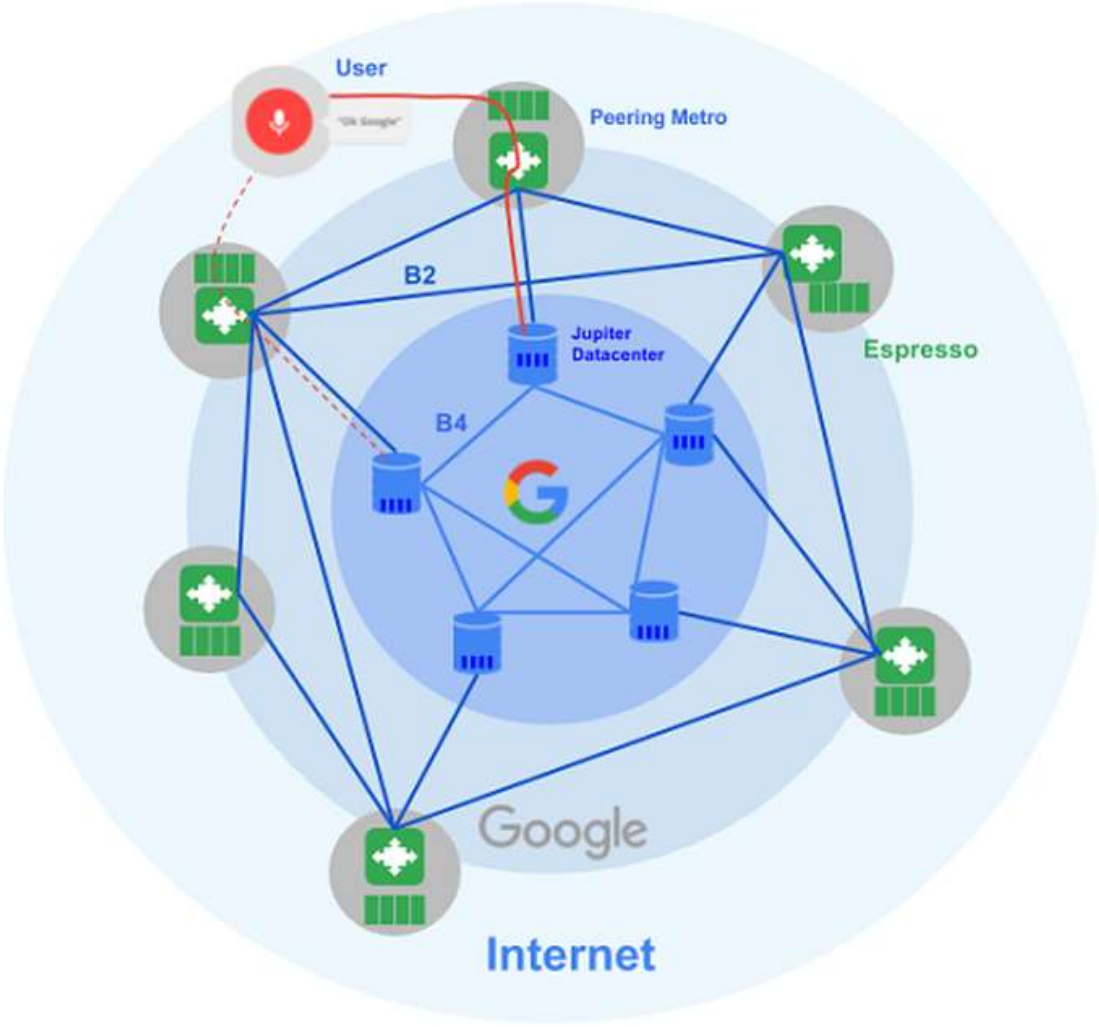


Figure 2.0

b) How it works

The machines that power Google's operations aren't cutting-edge power computers. In fact, they're relatively inexpensive machines running on Linux operating systems. How can one of the most influential companies on the Web rely on cheap hardware? It's due to the **Google File System** (**GFS**), which capitalizes on the strengths of off-the-shelf servers while compensating for any hardware weaknesses.

Google uses the GFS to organize and manipulate huge files and to allow application developers the research and development resources they require. The GFS is unique to Google and isn't for sale. But it could serve as a model for file systems for organizations with similar needs. Some GFS details remain a mystery to anyone outside of Google. For example, Google doesn't reveal how many computers it uses to operate the GFS.

c) What components does it use?

Google uses synchronous system to communicate with different users across the world without depending on global clock, instead it uses a logical clock to synchronize time in different nodes.

d) REFERRENCES

i. https://computer.howstuffworks.com/internet/basics/google-file-system.htm. Retrieved on 02June09 How the Google File System Works

ii. Google Architecture. High Scalability. 2008-11-22. Retrieved 2012-02-17.

3) **Netflix**

Components

Netflix has 3 main components;

1.OC or Netflix CDN:

Open Connect stores Netflix video in different locations throughout the world. When you press play the video streams from Open Connect, into your device, and is displayed by the client.

CDN — A content delivery network (CDN) is a system of distributed servers (network) that deliver pages and other Web content to a user, based on the geographic locations of the user, the origin of the webpage and the content delivery server.

Advantages of OC

- Less expensive
- Better quality
- More Scalable

2.Backend

Netflix uses a number of open-source software at the backend, including Java, MySQL, Gluster, Apache Tomcat, Hive, Chukwa, Cassandra, and Hadoop.

3.Client

How Netflix works

When the user loads Netflix app All requests are handled by the server in AWS Eg: Login, recommendations, home page, users history, billing, customer support etc.

Your app automatically figures out the best OC server, best format and best bitrate for you and then the video is streamed from a nearby Open Connect Appliance (OCA) in the Open Connect CDN.

The Netflix apps are so intelligent that they constantly check for best streaming server and bitrate for you and switches between formats and servers to give the best viewing experience for you.

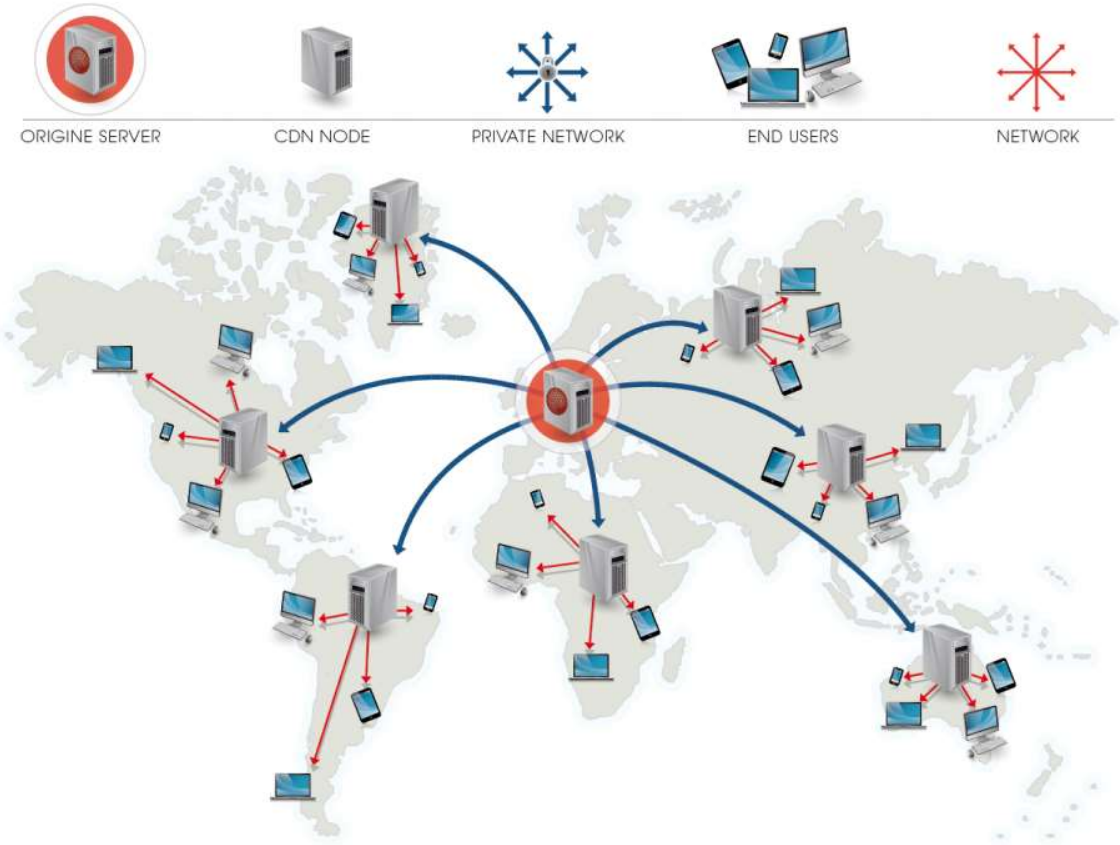
Now what Netflix does is with all of your searches, viewing, location, device, reviews and likes data on AWS it uses Hadoop | Machine learning models to recommend new movies which you might like.

Netflix uses Amazons Elastic Load Balancer (ELB) service to route traffic to our front-end services. ELB's are set up such that load is balanced across zones first, then instances. This is because the ELB is a two-tier load balancing scheme.

The first tier consists of basic DNS based round robin load balancing. This gets a client to an ELB endpoint in the cloud that is in one of the zones that your ELB is configured to use.

The second tier of the ELB service is an array of load balancer instances (provisioned directly by AWS), which does round-robin load balancing over our own instances that are behind it in the same zone.

Netflix also uses Microservices architecture to power all of the APIs needed for applications and Web apps. Each API calls the other micro-services for required data and then responds with the complete response



4) **EMAIL**

The email application is the pinnacle of distributed applications. Each computer accomplishes one small step in the process in an autonomous fashion. Individual components may fail while the system remains available to most of the users. The architecture is asynchronous, limited-shared-state, highly available, elaboration tolerance, message passing, with a dynamic protocol.

How it works

Each machine has an application server program that understands the email protocol of SMTP. When two machines want to exchange email, first they synchronize to a known initial shared state, then the email message header control information is exchanged. If the second machine agrees to the transfer, the contents of the email is then delivered. The receiving machine places the email in a special spool directory for later reading by the end user. Alternatively, the receiving machine may reject the email message or provide forwarding information.

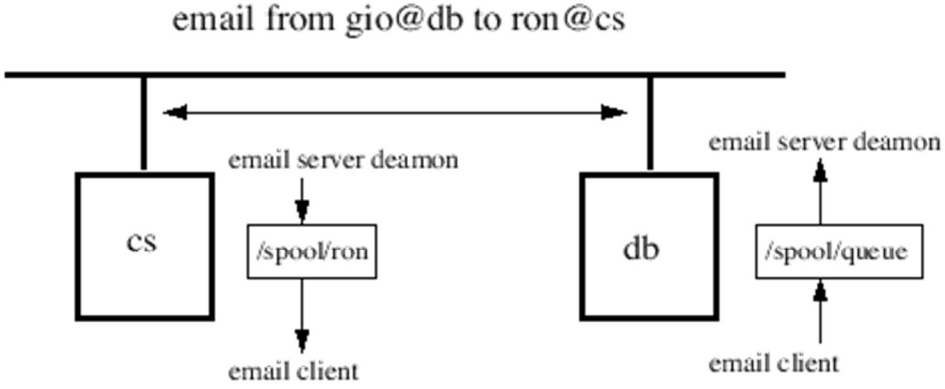
The next piece of the big picture are the email clients. When the end user wants to send email, a email client hands the email to the machine's email server. The server exchanges the email with another machine's email server. The recipient machine places the email message in the spool file belonging to the recipient. The email is ready for the final delivery stage. When the recipient wants to read email, the email client reads the spool file and displays the message.

If a server is not available the message is queued for later delivery. If an error occurs during transmission, the message is resent. Messages sent to unknown recipients are bounced back to the sender. An email address may include routing information, in which case, the receiver machine accepts the message, removes it's name from the routing list, and then processes the message to the next machine.

Disadvantages

1. There is no authentication making the email system autonomous and easily spoofed.
2. There is no data privacy of the message with many opportunities for an outside third party to discover the contents of a message.
3. There is no data integrity of the message allowing for the substitution the message, in transit, without knowledge of the recipient.
4. There is no digital signature on the message that guarantees that the sender really sent this message. There are no guarantees on delivery or performance and no way of specifying different levels of service.

SMTP based Electronic Email



References

NETFLIX system design: Narendra L (https://medium.com/@narengowda/netflix-system-design-dbec30fede8d)

A 360 Degree View Of The Entire Netflix Stack; MONDAY, NOVEMBER 9, 2015: http://highscalability.com/blog/2015/11/9/a-360-degree-view-of-the-entire-netflix-stack.html

Current Distributed Architectures: Ronald LeRoi Burback (http://infolab.stanford.edu/~burback/dadl/node116.html)