UNIVERSITY OF WATERLOO

Faculty of Mathematics

Yahoo Groups New Backend: Neomailer

Verizon Media (Yahoo!)

701 1st Ave, Sunnyvale, CA

Prepared by

Haomin Li

4B Computer Science

Student ID

April 20th, 2020

MEMORANDUM OF SUBMITTAL

To: Yung Lin

From: Haomin Li

Date: April 20th, 2020

Re: Yahoo Group New Backend: Neomailer

Dear supervisor,

This report, entitled “Yahoo Group New Backend: Neomailer” was prepared as my 4B term work report at Verizon Media (Yahoo!). This report is the fourth of the four work reports that the Co-operative Education Program requires for successful completion as part of my Computer Science Co-op degree requirements.

Throughout this internship, I have been working in the Yahoo Mail Delivery team mentored by Shashikiran and supervised by Yung. This report is written by me independently and has not received any previous academic credit. I would like to thank you for offering the amazing assistance in the workplace whenever I need help, which I could use to develop this report.

Thank you for your time in assessing this report.

Eric (Haomin) Li

Table of Contents

Executive Summary

Yahoo Groups was launched in January 2001. It aimed to provide a platform for users to look for friends who have the same interest with them. Recently, Yahoo decides to rewrite the entire Yahoo Groups project. It involves the rewrite of the frontend and the backend. During my four months at Yahoo, I was working on the rewrite of its backend component, called Neomailer. The implementation language of the Yahoo Groups backend was changed from C to Java.

In this report, I will introduce about Yahoo Groups and its new backend: Neomailer. As Yahoo Groups has millions of users and high volume daily posts, the regular synchronous implementation was not efficient enough. The concept of asynchronous programming is introduced. Rewriting Yahoo Groups backend involves many up-to-date technologies, such as the CICD tool, Asynchronous HTTP client, and Completable future type. I will further discuss both the technologies used in the Neomailer and the generic design of it. Three Commands of Yahoo Groups: Post command, Subscribe command, and Unsubscribe command was completed during my internship. Those complete commands and the high level of their designs would be demonstrated.

1.0 Introduction

“Yahoo Groups, launched in 2001, is basically a cross between a platform for mailing lists and internet forums. Groups can be interacted with on the Yahoo Groups site itself, or via email. In the 18 years that it existed, numerous niche communities made a home on the platform” (Jordan, 2019). According to Wikipedia, when the Yahoo Groups was at the peak, it has 115 million group members and there were 10 million groups. The web analytics website Quantcast reported around 933 thousand unique users daily to the Yahoo Groups website (https://en.wikipedia.org/wiki/Yahoo!_Group). On October 21, 2019, Yahoo Groups was winded down and users were no longer able to post new content to the site (Jordan, 2019). In 2020, Yahoo decided to rewrite the entire project of the Yahoo Groups.

Initially, Yahoo Group’s backend was written in C. C is an old language developed in 1972 by Bell Lab. In the current industry, no one uses C for software development. Programming in C is very time-consuming as it does not have many frameworks and libraries. Besides, C is not an easy-to-learn language. C codes are hard to be maintained. Thus, Yahoo decided to rewrite the Yahoo Groups project and update it with a higher-level language: Java.

2.0 Content

In the introduction, we have discussed that Yahoo decided to rewrite the whole Yahoo Group backend (Neomailer) with Java. In the following contents, I will introduce part of the design of the Neomailer and the commands that have been implemented during my internship.

2.1 Design of the Neomailer

Neomailer has three different layers: 1. Command layer, 2. Service layer, 3. Application layer. The reason that we use three different layers is because of encapsulation. "Encapsulation is used to hide the values or state of a structured data object inside a class, preventing unauthorized parties' direct access to them" (Dave Braunchweig, 2017).

2.1.1 Layers design

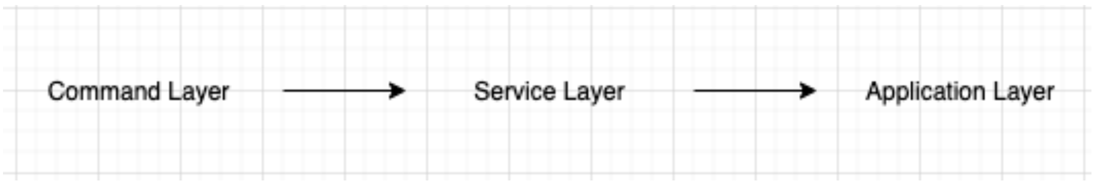


figure 1

1. Command Layer: contains several commands that will be used by users. Users can directly use the function from this layer to achieve functionalities without knowing the implementation. Commands in Command Layer only depend on classes in Service Layer and other commands. All the exceptions thrown from here should all be Command Exception.
2. Service Layer: contains classes that are used by commands in Command Layer or other services. Users cannot get in touch with the Service Layer. Classes on Service

Layer only depend on the Application Layer. All the exceptions thrown from here should all belong to Service Exception.

3. Application Layer: contains the most fundamental classes. There are only used by the Service Layer. Classes in Service Layer only depend on other packages, such as Apache open-source packages and do not depend on other layers. All the exceptions thrown from here should all be API Exception.

2.1.2 CICD tool: Screwdriver

The CICD tool is a very common tool in industrial production. "Continuous integration (CI) and continuous delivery (CD) deliver software to a production environment with speed, safety, and reliability" (Christian Melendez, 2019).

With Continuous Integration (CI), developers are able to integrate code changes continuously with the rest of the team. The integration happens after pushing (git push) to the master branch. Then the CI tool will automatically build the software and run some tests with the updated code.

"Continuous Delivery is the ability to get changes of all types—including new features, configuration changes, bug fixes and experiments—into production, or into the hands of users, safely and quickly in a sustainable way." (Christian Melendez, 2019). The CD tool ensures the quality of deployment.

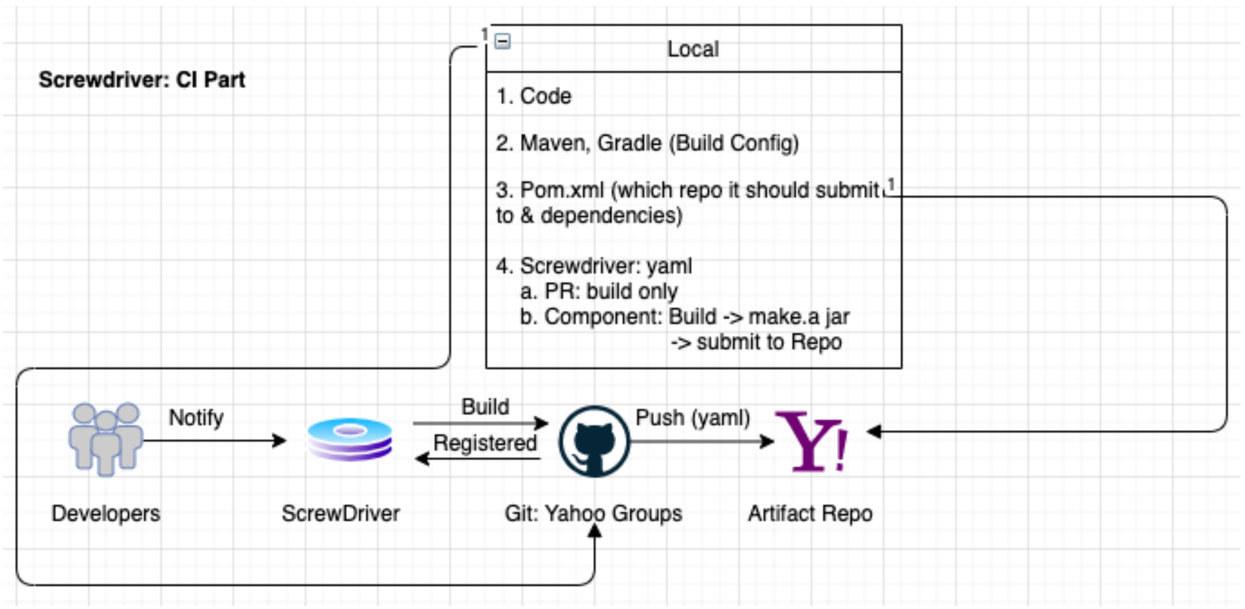


figure 2

Figure 2 shows the pipeline of Screwdriver's Continuous Integration. Screwdriver is an internal CICD tool for Yahoo. Its current fourth version has been integrated with all the Yahoo's projects. After a developer makes a pull request, the Screwdriver, registered with the YAML file, will be notified. It will use the build configuration to build the entire project. Then, it will show the status whether if the build is successful or not. If it is a success, the manager can push the code changes to the artifact repository.

2.1.3 Asynchronous HTTP Client

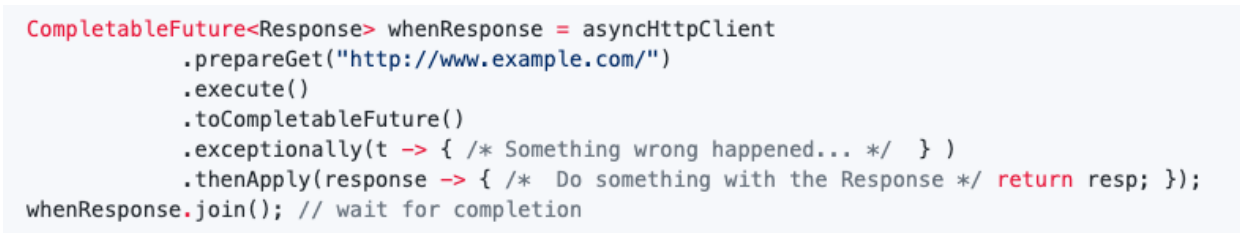
"Asynchronous Http Client (AHC) library allows Java application to easily execute Http request and asynchronously process Http response" (https://github.com/AsyncHttpClient/async-http-client).

```
Future<Response> whenResponse = asyncHttpClient.prepareGet("http://www.example.com/").execute();
Response response = whenResponse.get();
```

figure 3

Figure 3 is an example of Http Get using Asynchronous Http Client (AHC). Note that the 'prepareGet' method of AHC returns a type called the Future. This is a promise that the program will get the response information. The Future will run in the back and wait for the actual response. In the second line, by using the Get method, the user can fetch the response from the Future. However, it takes a lot of time for the response to arrive, this line of code may block the calling thread to get the response. Then, it will be as similar as synchronous Http Client.

We can improve it by using Continuations (figure 4):



```
CompletableFuture<Response> whenResponse = asyncHttpClient
            .prepareGet("http://www.example.com/")
            .execute()
            .toCompletableFuture()
            .exceptionally(t -> { /* Something wrong happened... */  } )
            .thenApply(response -> { /*  Do something with the Response */ return resp; });
whenResponse.join(); // wait for completion
```

figure 4

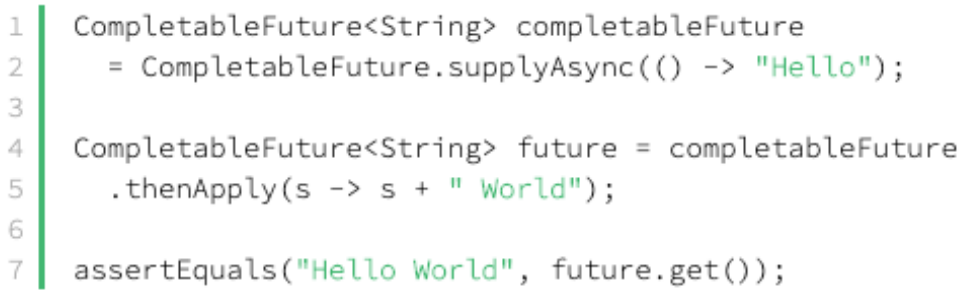
In Continuation, the 'execute' method returns a ListenableFuture, which can configure listeners to be notified of the Future's completion. ListenableFuture has a 'toCompletableFuture' method that returns a CompletableFuture. This can prevent the thread from blocking. In the next section, we will discuss CompletableFuture.

2.1.4 CompletableFuture

CompletableFuture is an extension to Java's Future API which is mentioned in 2.1.3. There are many limitations of the Future type, however it can be achieved by CompletableFuture. While implementing Neomailer, we used two features of CompletableFuture: 1. Processing Results of Asynchronous Computations. 2. Combining Futures.

1. Processing Results of Asynchronous Computations:

   When the data is not complete, developers have to wait for the data so that they can modify it. Instead, as shown in figure 5, developers can use the ‘thenApply’ method to give a rule to the CompletableFuture, which can modify the pending data even if it is not complete. In this way, the working thread would not be blocked. The modified result can be fetched later in the code when it is needed.

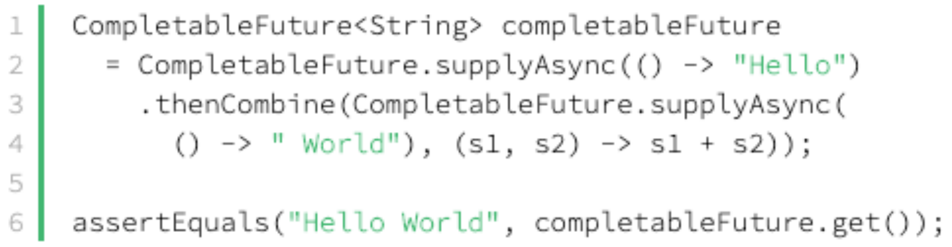


```
CompletableFuture<String> completableFuture
  = CompletableFuture.supplyAsync(() -> "Hello");

CompletableFuture<String> future = completableFuture
  .thenApply(s -> s + " World");

assertEquals("Hello World", future.get());
```

figure 5

2. Combining Futures:

   If developers would like to combine two different Futures and make some changes to their result, the ‘thenCombine’ method can do the work without fetching each of their results as in figure 6. Fetching the result of each Future can block the working thread.



```
CompletableFuture<String> completableFuture
  = CompletableFuture.supplyAsync(() -> "Hello")
    .thenCombine(CompletableFuture.supplyAsync(
      () -> " World"), (s1, s2) -> s1 + s2));

assertEquals("Hello World", completableFuture.get());
```

figure 6

2.2 Commands of the Neomailer

During my internship in the Yahoo mail team, four fundamental commands of Neomailer have been implemented: Post, Post Owner, Subscribe, and Unsubscribe. Each command represents one functionality.

Figure 7 demonstrates the dependencies of the four commands. There are two clients involved: Group API client (GAPI) and Sonic client. They are both implemented with Asynchronous Http Client, which is discussed in 2.1.3. Since the Post Owner command was integrated into the Post command, we will introduce the two clients and the three commands in the following.

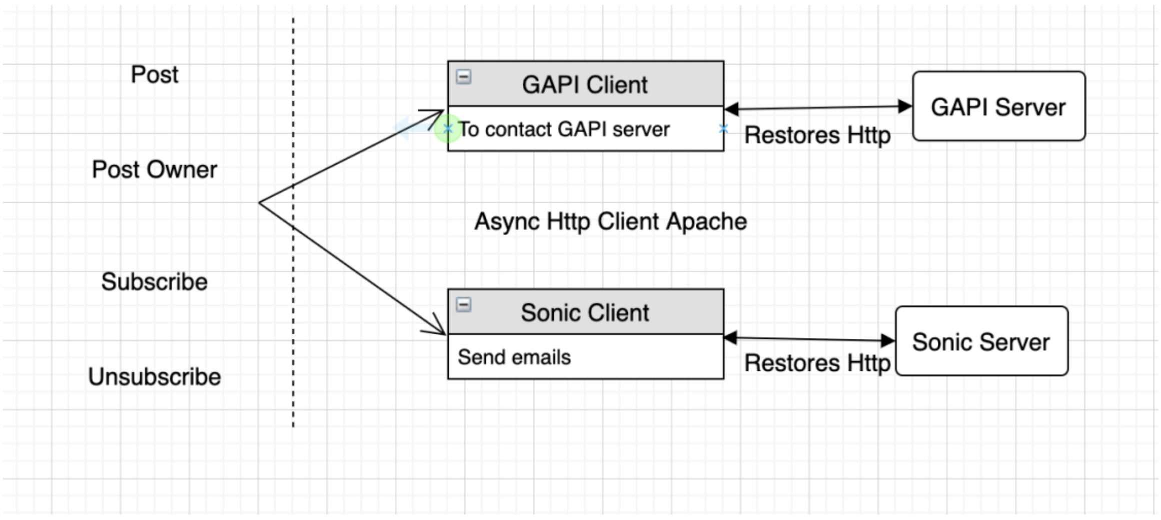


figure 7

2.2.1 Two Clients

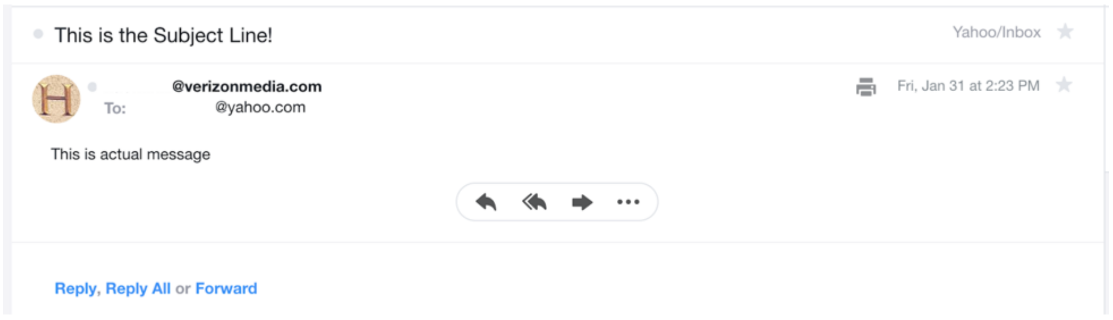
1. GAPI client: In Neomailer, a GAPI Client has been built to interact with GAPI server. The GAPI server is an existing server that has all groups' information, such as group members, owners, and the group's details. With GAPI client, a program can retrieve the members in a group, as well as the email address of a member. In addition, GAPI clients can also update certain groups' information.

2. Sonic client: A Sonic client is a media between the Neomailer and Sonic server. The Sonic server was an internal tool built and maintained by the Yahoo Mail team to perform accurate and fast email delivery service.

Sonic client and GAPI client are both used in the four commands.

2.2.2 Post Command

With Post Command, users can send messages to all the members in a group. With Post Owner Command, users can send messages to all the owners in a group. Figure 8 is an example of the Post Command.


This is the Subject Line!

Yahoo/Inbox

@verizonmedia.com

To: @yahoo.com

Fri, Jan 31 at 2:23 PM

This is actual message

Reply, Reply All or Forward

figure 8

We just need to provide such information: From Email address, group id, MIME data. By GAPI client, all the members of the group can be added to the recipient list. Through Sonic Client, all the recipients will receive the same MIME data from the email address. Note that MIME stands for Multipurpose Internet Mail Extensions. It uses headers and separators that tell the recipient's mail application how to re-create the message, including the contents. While using Post Owner Command, only the owners of the group will receive the message.

2.2.3 Subscribe & Unsubscribe Commands

Subscribe and Unsubscribe Commands are very similar. In terms of the flow of Unsubscribe Command, we can change the following words ‘subscribe’ to ‘unsubscribe’.

Here is the flow for subscribing a group:

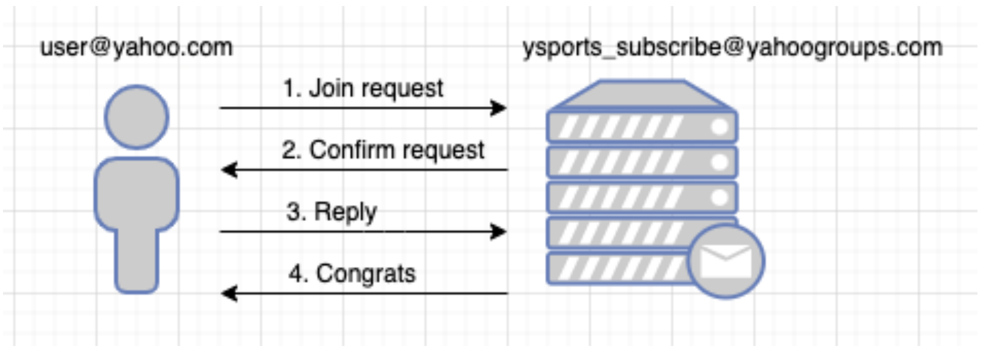
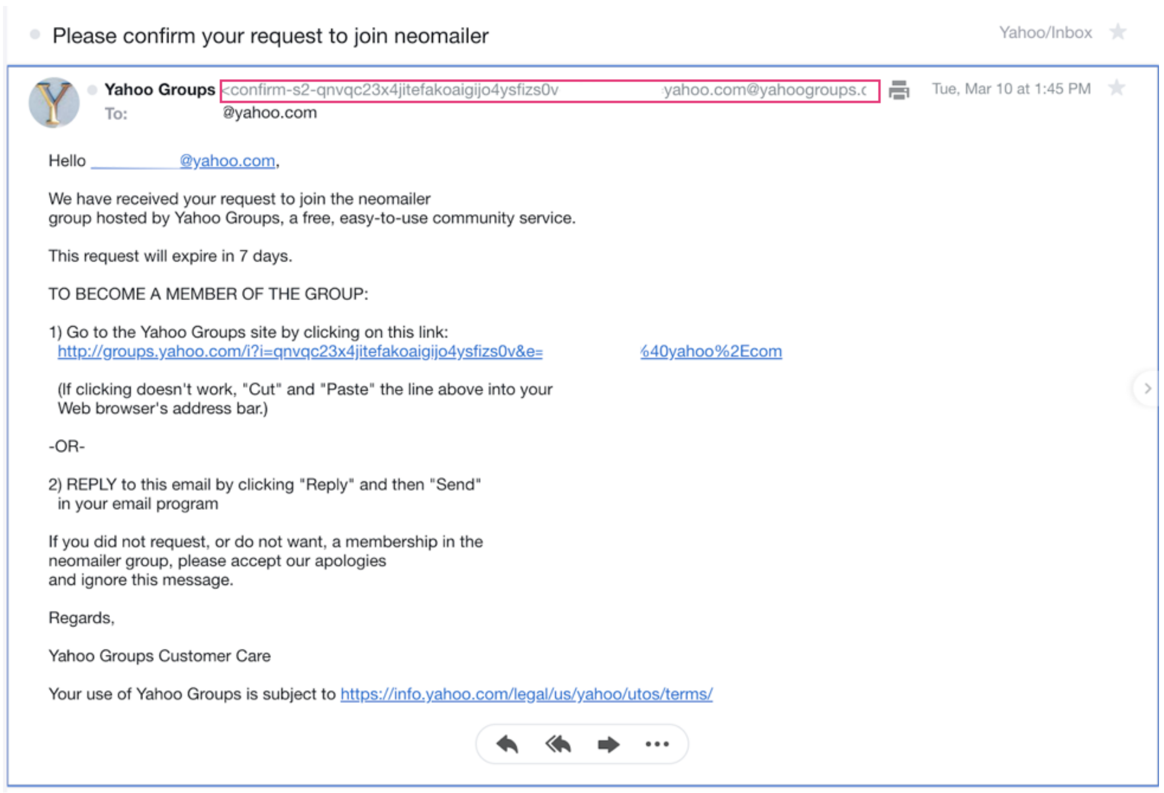


Figure 9

If a user would like to subscribe to an existing group called Ysports, the user can send a subscribing request email to the group subscribe email address. Then, the client generating email address would confirm the user’s request if it is true. The user would reply to the email and finally being added to the Ysports group.


Please confirm your request to join neomailer

Yahoo/Inbox

Yahoo Groups <confirm-s2-qnvqc23x4jitefakoaigijo4ysfizs0v yahoo.com@yahoogroups.c

To: @yahoo.com

Tue, Mar 10 at 1:45 PM

Hello @yahoo.com,

We have received your request to join the neomailer
group hosted by Yahoo Groups, a free, easy-to-use community service.

This request will expire in 7 days.

TO BECOME A MEMBER OF THE GROUP:

1) Go to the Yahoo Groups site by clicking on this link:
http://groups.yahoo.com/i?i=qnvqc23x4jitefakoaigijo4ysfizs0v&e= %40yahoo%2Ecom

(If clicking doesn't work, "Cut" and "Paste" the line above into your
Web browser's address bar.)

-OR-

2) REPLY to this email by clicking "Reply" and then "Send"
in your email program

If you did not request, or do not want, a membership in the
neomailer group, please accept our apologies
and ignore this message.

Regards,

Yahoo Groups Customer Care

Your use of Yahoo Groups is subject to https://info.yahoo.com/legal/us/yahoo/utos/terms/

figure 10

One tricky part of this command is that the client generating email address need to be unique (figure 10). Yahoo Groups uses this particular email address to record which user and which group the user want to join. Later, the GAPI server can process the request. In terms of the client generating email address, we develop special encrypting and decrypting methods. The encrypting function takes in the user's email address, time, and some other factors on the Yahoo Groups server. The decrypting function decrypts the particular email address on the GAPI server.

3.0 Conclusions

Neomailer has three layers: Command layer, Service layer, and Application layer. In this article, we covered some of the latest technologies used in the application later. CICD tool Screwdriver is used to maintain the project development and delivery. Open-source project: Asynchronous Http Client is used to improve server efficiency. CompletableFuture type and its relevant methods are used to prevent the running thread from blocking. Neomailer uses Sonic and GAPI clients to support the current four commands: Post command, Post Owner command, Subscribe command and Unsubscribe command. Post command allows Neomailer to deliver fast and accurate emails to all group members. Post Owner command allows Neomailer to only deliver emails to group owners. Subscribe and unsubscribe command allows users to subscribe or unsubscribe a group via emails.

Neomailer has been completely redesigned compared to the previous Yahoo Groups backend. We are all looking forward to its future.

Reference Page

1. Pearson, J. (2019, October 16). Yahoo Groups Is Winding Down and All Content Will Be Permanently Removed. Retrieved from https://www.vice.com/en_ca/article/8xwe9p/yahoo-groups-is-winding-down-and-all-content-will-be-permanently-removed
2. Braunschweig, D. (2018, December 15). Encapsulation. Retrieved from https://press.rebus.community/programmingfundamentals/chapter/encapsulation/
3. Meléndez, M. C. (2020, April 6). What Is CICD? What's Important and How to Get It Right. Retrieved from https://stackify.com/what-is-cicd-whats-important-and-how-to-get-it-right/
4. AsyncHttpClient. (2020, April 23). AsyncHttpClient/async-http-client. Retrieved from https://github.com/AsyncHttpClient/async-http-client
5. Baeldung. (2020, February 12). Guide To CompletableFuture. Retrieved from https://www.baeldung.com/java-completablefuture

Acknowledgments

I would like to thank my manager Yung and my mentor Shashi. Yung gave me this opportunity to work in his team and Shashi gave me plenty of mentorship in the project I was working on. They have been very helpful and patiently offered a lot of support since the first day I worked there. They shared a lot of experience as senior engineers for junior engineers working in the industry. I also want to thank the colleagues that I worked with. They gave me a lot of assistance during this internship.