Broadcasting Notifications: Integrating a Notification Feature for an Internal Google Tool

A Technical Report

presented to the faculty of the

School of Engineering and Applied Science

University of Virginia

by

Michael Acolatse

May 11, 2023

Technical advisor: Briana Morrison, Department of Computer Science

Broadcasting Notifications: Integrating a Notification Feature for an Internal Google Tool

CS 4991 Capstone Report, 2022
Michael Acolatse
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
msa3ga@virginia.edu

ABSTRACT

An internal workflow team at Google needed a way to mass notify engineers of a technical emergency via a chrome and dashboard notification. I designed and developed a broadcasting system that integrates with a current pipeline of an internal chrome extension to achieve this objective. I created a design document with all the information related to the motivation, architecture and timeline of the project. This system was a backend heavy project with some interaction to a frontend client so most of the implementation involved building services and interacting with APIs using Google's own systems and software. I created an end-to-end working system of the broadcasting system. However, Google has over 100,000 engineers and testing revealed about 20% failed calls due to issues with throttling. Future tasks include fixing the issue of throttling by either catching failed users and retrying the call or sectioning the users and calling the broadcast on those sections. Other stretch goals include adding the functionality to broadcast to subgroups of engineers and UX changes to the frontend.

1. INTRODUCTION

On December 1st, 2021, a huge security vulnerability was found in an open-source java library called Log4j. This called for all Google engineers to ensure none of the technologies were at risk from the technical threat. Google asked the internal notification team, which I had joined as a summer intern, to send a warning message through their system. This internal tool has a user base of approximately 4,000 engineers so the message would have a significant reach. However, the team did not have the functionality to send out mass notifications since it was designed for individual notifications only. This situation prompted the idea of a broadcasting feature for the internal notification tool. The tool was recently launched to be preinstalled in all the Google engineer's machines, increasing the user base from 4,000 to 100,000 engineers, which increases the potential impact of the broadcasting project.

2. RELATED WORKS

Everbridge (2021) explains factors to consider when utilizing a mass messaging system: targeting the individual and not the device; escalating to ensure that the next person or group is notified; broadcasting to virtually any communications device including desktop alerts; setting up templates with predetermined contacts and messages; protecting infrastructure capacity with flexible call-throttling; and automatically publishing to websites, internal systems, and social media. A simple email is not enough to perform an effective notification of a technical emergency that requires immediate action. The broadcast project had to consider all those points in the design of the system.

Vdovin (2022) proposes that the main use case for the broadcasting system is to relay messages of technical emergencies which can give rise to panic and chaos. He explains that notification systems create a centralized command center to bring order to a chaotic situation. From this center, employees can be immediately informed about the situation, as well as measures they must take to ensure technical security.

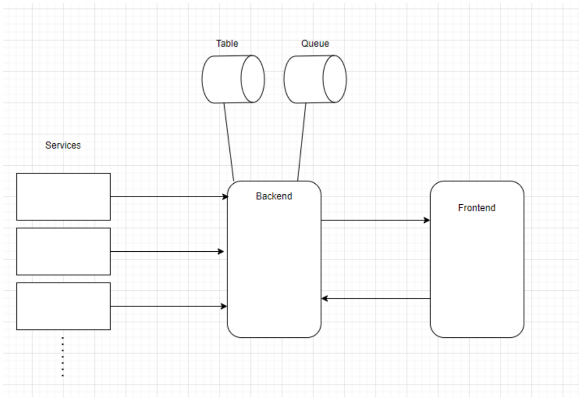
3. SYSTEM DESIGN

The broadcasting feature was built into the existing architecture of the internal notification tool.

3.1 Review of System Architecture

The internal notification tool has a main backend server that is open to receive API or RPC calls from other services. This backend server is responsible for pipelining notifications from the services to the frontend. The backend works with two main databases. The first is a relational database that holds information about the users. The second is a queue responsible for firing the notifications to the users.

Figure 1: System Architecture of Internal Notification Tool



3.2 Design of Broadcast Feature

The broadcasting feature has five main components: storage and approval of broadcast message, initiation of the broadcast, pulling all users from database, creating notification events for all users, and pushing the notifications to the frontend.

The storage and approval of the broadcast message is done through configuration file system. The message is created and pushed to a configuration file with five components: title, summary, details, link, and severity.

Next the configuration file is sent to be reviewed and approved. Once the message is approved, it is ready to be launched. The broadcast is triggered through a command line API call by a verified user. The API call sends an RPC (remote procedure call) to the backend to initiate the broadcast.

4. RESULTS

By the end of my internship, I had an end-to-end functioning broadcasting system. A verified user is able to create a new broadcast message, as well as trigger the message to be sent. Users were able to receive the message as a desktop notification and a dashboard notification. In order to test the new feature, I created a silent broadcasting system which would send the message to all the user but would not be displayed to the users. This test showed that the system had some throttling issues which resulted in about 20% of messages failing to be sent. Due to the time constraint, I was not personally able to fix the issue; however, I wrote documentation on several fixes that can be implemented.

5. CONCLUSION

The broadcasting project made use of a system that has shown to have quicker response metrics by users to notify Google engineers about technical emergencies. This project can be described as a functionality that hopefully does not have to be used but is necessary for emergency situations. This feature eliminates the concern of missing important emails, as email inboxes are usually flooded. With the ability to carefully detail the broadcast message, engineers can quickly understand and iterate on the broadcasted messages. Also, with the security of the initiation of a broadcast message through approval and authentication, it can be ensured that the broadcast messages are safe and reliable.

6. FUTURE WORK

Future iterations of this project include sub-group broadcasting and UX changes. At the time of initial completion of the project, the broadcast sends a message to all Google engineers. Some technical emergencies or update only pertain to certain sub-section of Google engineers. The feature of sub-group broadcasting makes the system more flexible. Regarding UX changes, the broadcast message currently shows up in the dashboard a regular notification with a color related to its severity. Since broadcast messages are supposed to be considered urgent, changing it to a banner-like notification can be more effective.

REFERENCES

1. Everbridge. 2021. What an emergency mass notification system is and why it matters. (April 2021). Retrieved September 23, 2022 from https://www.everbridge.com/blog/what-is-emergency-mass-notification-system-why-it-matters/
2. Anton Vdovin. 2022. The importance of a notification system in the workplace. (September 2022). Retrieved September 23, 2022 from https://www.alert-software.com/blog/2810-2