Development of a Gateway Server to Enhance Early Warning in Digital Terrestrial Television

Gonzalo Olmedo(✉) and Alejandro Salas

WiCOM-Energy Research Group, Universidad de Las Fuerzas Armadas ESPE, Sangolquí, Ecuador
{gfolmedo,jlsalas}@espe.edu.ec

Abstract. This research addresses the analysis, design, development, and implementation of a Gateway Server aimed at decoding and retransmitting early warning messages within the context of digital terrestrial television services. It is crucial to highlight that Ecuador is a disaster-prone area susceptible to natural calamities. The Emergency Warning Broadcasting System is highly beneficial in disseminating information. However, due to its low popularity and certain limitations, developing a Gateway server becomes essential to notify users or clients through various mobile and desktop applications. The system is designed to be versatile and compatible with multiple platforms, including desktop and mobile devices, utilizing iOS and Android native services for notifications. Furthermore, a public server hosted on Digital Ocean, a user-friendly platform offering accessible Cloud Computing services, has been implemented. This server manages the logic related to users, tokens, device identifiers, and alert messages. These services result in a Gateway Server capable of interpreting early warning messages and administering alert distribution through APNs and Google FCM for mobile clients and through direct API consumption toward the server for desktop clients.

Keywords: EWBS · ISDB-T · Gateway Server · TDT

1 Introduction

Disaster prevention and the swift dissemination of early warnings are crucial aspects for the safety and well-being of the population. Throughout history, we have witnessed numerous devastating events, such as the earthquake that struck Tokyo in September 1983, resulting in the loss of 100,000 lives [1]. This tragic event gave rise to Disaster Prevention Day in Japan, commemorated on the first of September. To stress the importance of early warning and disaster prevention, Japan implemented the Emergency Warning Broadcasting System (EWBS) in 1985, marking the beginning of a new era in emergency response [2].

In the international context, natural disasters, such as the devastating earthquake that occurred in Ecuador in April 2016, reaching a magnitude of 7.8 on the Richter scale and resulting in the tragic loss of 655 lives, remind us of the importance of early warning and the need to expand the dissemination of alerts to a broader audience. In this regard,

M. J. Abásolo et al. (Eds.): jAUTI 2023, CCIS 2140, pp. 18–30, 2024.
https://doi.org/10.1007/978-3-031-70439-0_2

technological solutions have been sought, such as the implementation of alert systems in the IP telephony network of the Universidad de las Fuerzas Armadas- ESPE, aiming to reach as many people as possible and reduce loss of life and damage [3].

The transition from analog to digital terrestrial television (DTT) in Ecuador, driven by the government since 2010, has been a catalyst for improving the dissemination of early warnings through this medium. Adopting the ISDB-Tb standard (International et al. - Terrestrial) has allowed for modernizing television broadcasts and incorporating additional services, such as emergency alerts [4].

The ESPE has played a crucial role in researching and developing technologies related to early warning in digital terrestrial television services. Previous projects, such as Software Defined Radio (SDR), were used to implement a terrestrial digital television transmitter and an emergency signal receiver. These have laid the groundwork for the work presented in this article [5, 6].

This research's significance lies in its ability to communicate early warnings to the population through multiple end devices, such as mobile phones, considering that a significant portion of the Ecuadorian population has access to these devices and the Internet. The rapid dissemination of information amid a natural disaster can save lives and mitigate the negative impact on society.

A project similar to the one presented in this article was developed in Peru by experts and manufacturers from Japan in collaboration with the Japan International Cooperation Agency (JICA) to enhance the transmission of emergency alerts. The development of the EWBS Gateway, which plays a vital role in the system, was explicitly requested by the National Peruvian Organization for Disaster Prevention. This device is designed to send emergency notifications to personal computers and mobile devices within internal networks, leveraging existing communication infrastructure to optimize the effectiveness and reach of the system in response to emergencies [7].

The present work focuses on implementing a Gateway Server that enables the reception and retransmission of messages from the early warning system in digital terrestrial television services. Mobile and desktop applications will also be developed to allow users to receive these alerts promptly. The project scope includes creating a private virtual server, a Python script, and applications for multiple platforms. Furthermore, future work will utilize technologies like Docker and Kubernetes to enhance the system's scalability.

This work aims to contribute to the safety and well-being of the population by providing an effective early warning system within the context of digital terrestrial television, offering the capability to communicate alerts to many people efficiently. Using cutting-edge technologies and collaboration with various institutions in Ecuador will enhance disaster response and help prevent more significant damages.

2 Methodology

To initiate the project, we developed an ISDB-T receiver with EWBS on SDR, building on the foundation in references [5] and [6]. Initially, we created a Python script to extract the overlay message from the transport stream transmitted via EWBS. Later, we used the message from the Radio on a software-defined radio (SDR) receiver and followed the methodology outlined below to advance the project.

Figure 1 depicts the project's comprehensive block diagram, representing a complete scenario of a terrestrial digital television transmitter and receiver, with its Transport Stream (TS) containing the early warning message. The "Gateway" section is particularly relevant to this research, which involves reading the message and retransmitting it through various mobile or desktop applications used by end clients. This aspect represents three stages of design and implementation.

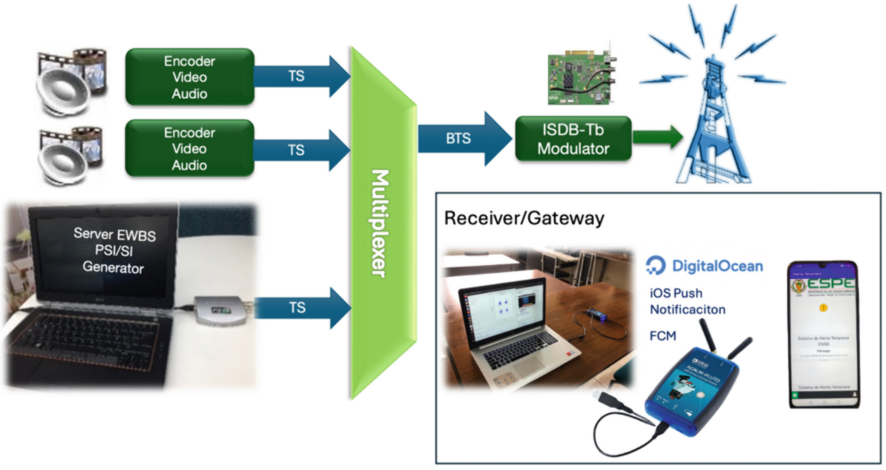

Fig. 1. Comprehensive Block Diagram of Terrestrial Digital Television System with EWBS and Gateway.

For the transmitter, we employed an ISDB-T modulator, specifically the DeKtec DTA-211, which, via the StreamXpress software, transmits the TS signal over VHF to UHF bands. This TS included an early warning message of the EWBS type, generated based on the project presented in [4]. However, any other TS conforming to the EWBS system under the ISDB-T standard could also be utilized.

The receiver, designed using GNU SDR platforms such as Adalm Pluto and based on the project outlined in [5], enables real-time capture of the OneSeg signal of ISDB-T, with the EWBS system configured within the Transport Stream (TS) structure and the design conducted in the UHF bands. We have created a Python script with multiple functions, such as reading early warning messages, triggering the alert signal, and notifying mobile clients. This script mediates between the received message and the mobile applications, ensuring that alerts are immediately disseminated. The script checks the file's content for emptiness and takes no action if the file is empty. However, when the transmitter issues an alarm with an encrypted message, the receiving computer via the SDR detects the alert, decodes the message, and saves the decoded message into a text file named "Alert.txt" at the path "/home/user/Downloads."

The Gateway Server component plays a crucial role in our project. It receives the TS from the receiver, efficiently separating the audio, video, and encrypted early warning message data. The EWBS Receiver block then receives and decodes the message, storing

it in a specific path. This data management process is a key factor in the smooth functioning of our system. Subsequently, a Public Server, VPS, or Droplet is implemented. This component handles data persistence and token management, serving as a query interface for desktop clients and ensuring the availability of necessary data, thereby facilitating communication between the server and desktop applications.

Our project was designed with a strong focus on the end client's experience. Web and mobile applications were developed to replicate the early warning message. These applications serve as the user interface through which end clients receive and interact with the alerts, ensuring prompt and effective dissemination of the message. This user-centric approach is a key aspect of our project, highlighting our commitment to effective communication.

EWBS Receiver

This section provides an overview of the processing blocks used for One-seg reception, as shown in Fig. 2 from the GNU Radio Companion environment, which were developed in the work for implementing a One Seg receiver [5]. The reception starts from the PlutoSDR Source block, which tunes and samples the signal from the Adam Pluto device at a specific rate allowed by the software, as indicated in the algorithm developed for the signal recorder since not all devices can sample at any rate.

The OFDM synchronization stage is intended to counteract timing and frequency synchronization effects in the OFDM system and is also responsible for orthogonal demodulation. The synchronization stage searches for the samples at the beginning of each symbol and estimates them to correct the carrier frequency offsets subsequently. The Sync Dem Ofdm 1Seg and OFDM Dem 1seg blocks perform this procedure. The first of these applies a Van de Beek maximum likelihood algorithm in a $2N + L$ window, where N represents the number of samples per symbol per operating mode (e.g., 1024 samples in a mode of operation 3), while L represents the length of samples in time for the guard interval [5]. Additionally, it determines the SNR (Signal Noise Ratio) corresponding to one of the objective metrics for evaluating performance and its variations in reception. It has an output of complex sample signals connected to the input of the OFDM Dem 1seg block.

This second block performs time and frequency synchronization by converting signals from the time domain to carriers in frequency (e.g., mode of operation 3 has 432 active carriers: auxiliary, TMCC, pilot, and data). It comes from the gr-isdbt module.

Subsequently, the most crucial block is the TMCC Decoder 1-seg, as it allows for the extraction of information concerning the activation of bit 26, where the emergency alert system of the TMCC carrier is located, which has 204 symbols. At its input, it has all the active carriers; at its output, only those corresponding to data.

Once the information is processed for one-seg reception, frames or packets of 188 bytes are outputted and received at the blocks' input (Read EWBS, Emergency Message, Flag EWBS). These processing blocks, programmed in Python for the GNU Radio environment, are responsible for extracting the characteristics of the EWBS emergency alert system. Figure 3 illustrates the processing blocks for EWBS. The Read EWBS block processes the data flow of 188 bytes (TS) at the receiver's output to extract detailed information from the PMT table and descriptor1 containing the emergency information.

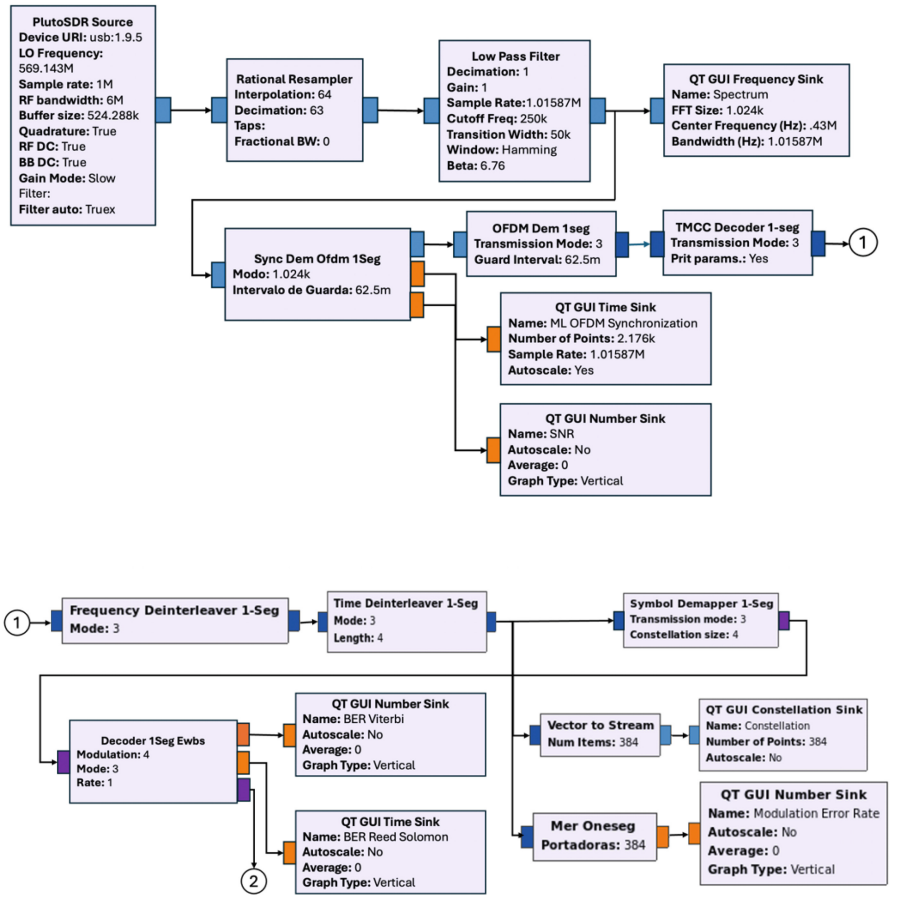

Fig. 2. GNU Radio Companion Environment for One Seg Receiver Implementation.

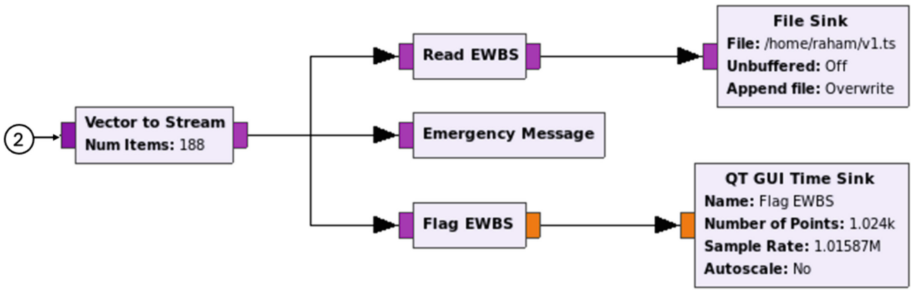

Fig. 3. Processing Blocks for EWBS.

Cloud Server Configuration
The initial stage involved selecting a cloud service provider, such as Digital Ocean [8], to host the server. Subsequently, a Droplet, a virtual machine on Digital Ocean, was created and configured with the Ubuntu 20.04 LTS operating system. A domain name was also associated with the Droplet's public IP address, necessitating DNS records configuration at the domain provider.

Dependency Installation
We have successfully installed all the necessary dependencies on the server, including MySQL 8 for data persistence, Python 3.8, Django, and the Django Rest Framework for creating and managing the web API. Additionally, we installed Nginx as a web server to act as a reverse proxy and Gunicorn to serve the Django application [9].

The database, an instance of MySQL 8 installed on the VPS, plays a crucial role in our system. It is used for data persistence, such as messages and tokens, and through its relational logic, it stores messages read by different clients.

Below are the different tables used in the MySQL relational database "Messages." Their usage will be explained later to ensure the proper functioning of the Gateway Server.

- **auth_user (Framework-specific)**: This table stores information about the superuser, who has permissions to save new messages and query tokens from different devices. An unregistered user cannot send alerts.
- **Messages**: This table stores early warning messages along with their respective timestamps or creation dates.
- **Tokens**: This table stores the token IDs (Device IDs) of each device that has installed the application, whether they are desktop clients or mobile apps. It contains two flags to identify if the token is mobile ('esMovil') and whether it belongs to iOS or Android ('esAndroid').
- **Token_messages**: This table establishes a many-to-many relationship between messages and tokens. By querying this table, we can determine which devices have successfully received the sent message.

Other tables within the DRF framework are also used for logging, sessions, and migrations, which could be more relevant to the project.

Django Application Configuration
This section involves cloning the Django project from a repository onto the server, setting up a Python virtual environment, and executing migrations to create the database tables. Additionally, a Django superuser was created to manage the application.

Django Rest Framework is a Python framework for web applications that optimizes code quantity when building web applications and handles application logic.

Application Deployment in Production
During deployment, the team configured a service to execute Gunicorn and established an Nginx configuration file for the domain. Furthermore, they enabled an SSL certificate through Let's Encrypt to ensure secure connections via HTTPS. Adjustments were made to the firewall to permit traffic on ports 80 and 443 [10]. On port 80, Nginx acted as a

reverse proxy, directing requests to Gunicorn, which can efficiently manage thousands of simultaneous connections, accessible at https://alertaewbs.site.

Gunicorn is an application server that translates HTTP requests into a language that Python can understand using the Web Server Gateway Interface (WSGI), a standard between web servers and web applications. Figure 4 illustrates how the architecture of Nginx, Gunicorn, and Python operates.

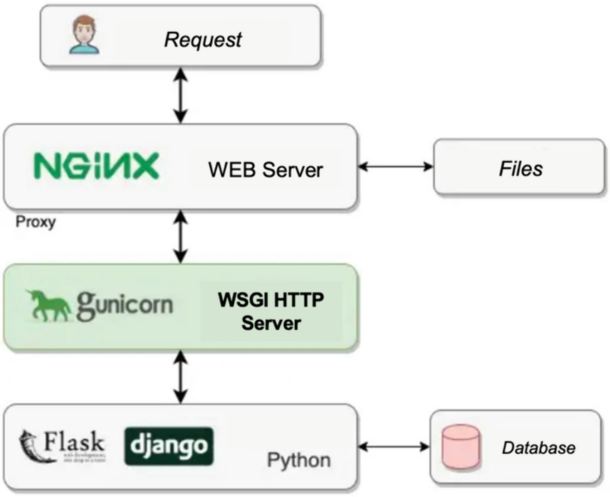

Fig. 4. Architecture of Nginx, Gunicorn, and Python.

Figure 5 shows a flow diagram of the receiver for reading an early warning message for subsequent dissemination on various platforms.

As a second security measure against potential attacks, two endpoints of the system were protected:

- /Insert/: Inserts a new message into the database.
- /Tokens/: Inserts and registers a new device (Device ID) in the database.

We protect the Droplet from potential attacks by using JSON Web Token (JWT) [11], an RFC 7519 standard that ensures secure data exchange between two parties and implements security measures commonly used in most systems and services. Authentication with Apple's APNs is performed in the same manner.

Development of Applications

This stage focused on developing desktop and mobile applications. The desktop application was developed using Spring Boot [12], while the mobile application was developed using React-Native [13] with Expo. Both applications are registered on the server and query unread messages. Push notifications for iOS and Android devices are implemented using APNs [14] and FCM [15].

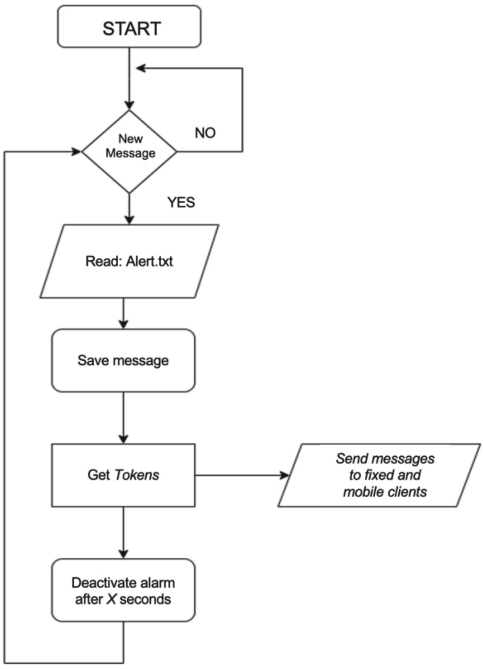

Fig. 5. Flow Diagram of Receiver for Early Warning Message Reading and Dissemination on Various Platforms.

Testing and Deployment
Exhaustive testing was conducted on all system parts to ensure its correct operation. Once all tests are successful, the system will be ready to enter production and provide early warning services to end users.

Maintenance and Updates
Finally, a continuous maintenance plan is established. The system is continuously monitored and maintained to ensure optimal operation. Software and hardware updates are applied as necessary.

3 Results

The tests conducted at the Digital Television Laboratory of the Universidad de las Fuerzas Armadas - ESPE yielded satisfactory results in the context of early warnings through digital terrestrial television. The devices involved in the test scenario included a transmitter utilizing a computer with StreamXpress software, a DeKtec DTA-211 modulator, and an antenna, alongside a receiver consisting of a computer and an Adalm Pluto SDR device. End clients were distributed across Windows, Linux, iOS, and Android systems, allowing for the evaluation of the system's applicability on various platforms.

Figures 6 and 7 document the testing carried out in a laboratory environment and demonstrate the system's successful operation. Notifications were effectively received

on both desktop and mobile devices, indicating the viability and robustness of the implemented infrastructure. The collaboration of various devices and components, such as the transmitter, modulator, antenna, receiver, computers, and the Software Defined Radio (SDR) Adalm Pluto [16], was essential in achieving this outcome.

Fig. 6. Video Demonstration in the Digital Television Laboratory of UFA - ESPE: Real-World Environment Showcase.

Fig. 7. Real-world Testing Environment at the Digital Television Laboratory of the Universidad de las Fuerzas Armadas – ESPE.

Java's JFrame notifications were utilized in the desktop environment, as depicted in Fig. 8. This enabled user alerts on both Windows and Linux operating systems.

For mobile devices, applications were developed for both Android and iOS systems, featuring intuitive and functional user interfaces. In the case of Android, two Tab Navigation layouts were created, offering a list of early warning messages and the ability to customize the alarm. Figure 9 displays the received notifications.

A similar approach was followed in the iOS environment, leveraging the versatility of cross-platform technology. Figure 10 showcases notifications on iOS devices.

Stress tests of the system and server were conducted using Apache JMeter, enabling evaluation of the server's performance under various workloads. The results obtained

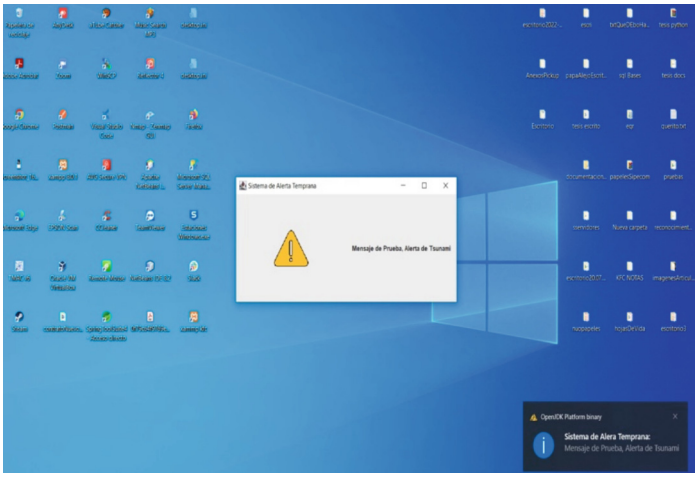

Fig. 8. Notification in a Windows 10 Desktop Environment.

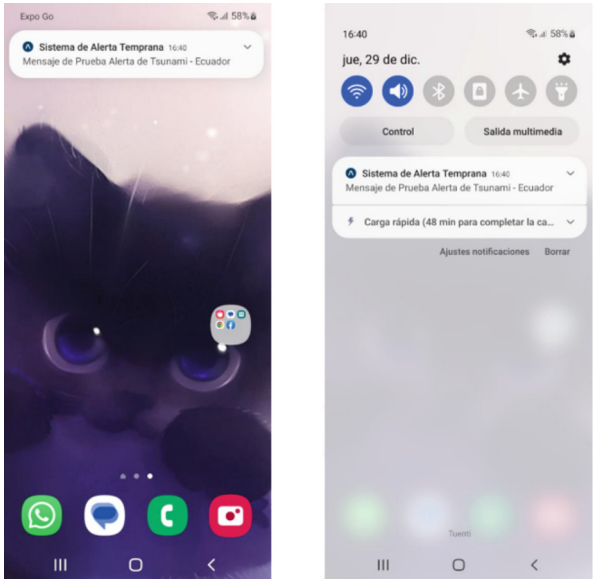

Fig. 9. Android Notifications.

are presented in Table 1. The server demonstrated excellent performance, even under significant loads, with minimal error rates.

Fig. 10. iOS Notifications.

Table 1. Stress Tests with Apache JMeter.

Number of Requests	Error Percentage [%]	Time [s]
100	0	1
200	0	3
300	0	4
400	0	5
500	3,2	7

4 Discussion

The implementation project of the Emergency Warning Broadcasting System (EWBS) in the context of terrestrial digital television represents a significant advancement in providing early warnings to the population. The integration of various components such as transmitters, modulators, antennas, and receivers has been demonstrated as crucial to ensure the effective transmission and reception of alerts in the terrestrial digital television environment. The collaboration of desktop and mobile devices allows notifications to reach a broad audience, increasing the likelihood that people will receive alerts in critical situations.

The server stress tests conducted with Apache JMeter have revealed the system's robustness and efficiency under increasing workloads. The results have shown minimal error rates even under significant loads, supporting the system's scalability and performance in a real-world scenario. Additionally, it was observed that the infrastructure used is highly effective in disseminating alerts on desktop and mobile devices.

5 Conclusions

In the context of terrestrial digital television, implementing the EWBS system has proven to be a promising technological solution for rapidly disseminating early warnings. The ability to notify through both desktop and mobile devices increases the accessibility of alerts, considering that a significant portion of the Ecuadorian population has access to these devices and the internet. The successful implementation of mobile applications for Android and iOS systems allows users to customize their notifications and access critical information promptly.

The server stress tests conducted with Apache JMeter have not just revealed, but strongly affirmed the system's robustness and efficiency under increasing workloads. The results have shown minimal error rates even under significant loads, providing a solid foundation for the system's scalability and performance in a real-world scenario. Additionally, it was observed that the infrastructure used is highly effective in disseminating alerts on desktop and mobile devices.

One of the significant changes in the Gateway design developed by Japan for Peru is the ability to upload area codes and emergency alert messages to the cloud. This structure allows for the extension of emergency alert information beyond the Gateway's coverage area and facilitates the integration of advanced options, such as displaying maps of the affected areas. Additionally, it incorporates the use of the Common Alerting Protocol (CAP). CAP is an international standard designed for the uniform transmission of alerts and warnings across various technologies, enabling effective propagation of emergency information to other devices. This scenario maximizes the existing digital terrestrial television infrastructure, ensuring broad dissemination and timely access to crucial information during emergencies.

References

1. Ecu911. Servicio Integrado de Seguridad, Sistema de Alerta Temprana. Recuperado de (2018). https://www.ecu911.gob.ec/sat-tsunami/
2. Takahashi, N., Nishida, K., Tsukada, S., Yamanaka, M., Horiuchi, S.: Earthquake early warning (EEW) in Japan: warning to the public and EEW research activities. Soil Dyn. Earthq. Eng. **29**(4), 798–810 (2009). https://doi.org/10.1016/j.soildyn.2008.06.008
3. Acosta, F., Sambrano, Y.: Diseño e Implementación de una Interfaz que Adapte una Señal de Emergencia de Televisión Digital a la red de telefonía IP de la Universidad de las Fuerzas Armadas - ESPE. Universidad de las Fuerzas Armadas - ESPE, Tesis de Ingeniería (2019)
4. Olmedo, G., Acosta, F., Haro, R., Villamarín, D., Benavides, N.: Broadcast Testing of emergency alert system for digital terrestrial television EWBS in Ecuador. In: Abásolo, M., Silva, T., González, N. (eds.) Applications and Usability of Interactive TV. jAUTI 2018. Communications in Computer and Information Science, vol. 1004. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-23862-9_13
5. Olmedo, G., Castillo, R.: . Implementación de un sistema de recepción de una señal de emergencia EWBS utilizando la Plataforma de Radio Definida por Software para el estándar ISDB-Tb. Tesis de Ingeniería, Universidad de las Fuerzas Armadas - ESPE. Recuperado de (2021). http://repositorio.espe.edu.ec/bitstream/21000/25711/1/T-ESPE-044717.pdf
6. Olmedo, G., Chanataxi, B., Benavides, N.: Receiver for ISDB-Tb standard with low-cost SDR and interactive application extractor. In: 2022 IEEE ANDESCON, Barranquilla, Colombia, pp. 1-6 (2022).https://doi.org/10.1109/ANDESCON56260.2022.9989634

7. Sakaguchi, Y., Takahashi, Y., Sakuma, S.: Actividades de Difusión de la Tecnología Japonesa EWBS - Sistema de Alerta de Emergencias por Radiodifusión – (Emergency Warning Broadcast System)," presentado en DiBEG, Tokio, Japón. Disponible (2022). https://www.dibeg.org,
8. Matheus, C.: Rockcontent. Digital Ocean: qué es, cómo usar, ventajas y desventajas (22 de abril de 2020). Recuperado de https://rockcontent.com/es/blog/digital-ocean/
9. FIB, i. (7 de julio de 2016). Django API REST. Recuperado de https://inlab.fib.upc.edu/es/blog/django-api-rest#:~:text=Django%20REST%20Framework%20es%20un,o%20el%20proyecto%20UOC%20%C3%ADndex
10. Gustavo, B.: Hostinger. ¿Qué es SSL, TLS y HTTPS? Recuperado de (22 de noviembre de 2022). https://www.hostinger.es/tutoriales/ssl-tls-https
11. López Magaña, I.: Qué es JSON Web Token y cómo funciona. OpenWebinars (2020). Recuperado de https://openwebinars.net/blog/que-es-json-web-token-y-como-funciona
12. Wigglesworth, A., Duchi, J.: Pro Spring Boot. Springer (2016).https://doi.org/10.1007/978-1-4842-1890-0
13. Facebook, Inc. React Native. GitHub (2013). https://github.com/facebook/react-native
14. Apple Inc. Local and Remote Notification Programming Guide. Apple Developer (2021). https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/
15. Google LLC. Firebase Cloud Messaging. Firebase (2021). https://firebase.google.com/docs/cloud-messaging
16. Analog Devices. Adalm Pluto SDR Active Learning Modulo. Recuperado de (2017). https://www.analog.com/media/en/news-marketing-collateral/product-highlight/ADALM-PLUTO-Product-Highlight.pdf