18TH WORLD CONFERENCE ON EARTHQUAKE ENGINEERING
WCEE2024
MILAN

MILAN, ITALY
30TH JUNE - 5TH JULY 2024

www.wcee2024.it

DESIGN AND IMPLEMENTATION OF THE PHILIPPINE PUBLIC SEISMIC NETWORK FOR RESEARCH AND EDUCATION

C.J. Sanchez[1], J.C. Cabang[2], & A.M.F. Lagmay[2]

[1] University of the Philippines Resilience Institute, Quezon City, Philippines, christopher.jeff.sanchez@eee.upd.edu.ph

[2] University of the Philippines Resilience Institute, Quezon City, Philippines

Abstract: *The reliable detection of earthquake events heavily relies on the efficient transmission of seismic data across a seismic network. Traditional seismic detection networks are mainly composed of specialized agencies and universities that can afford costly equipment. However, with the emergence of low-cost sensors made possible by the Internet of Things (IoT) and the growing Public Seismic Network (PSN, which is now dominated by the Raspberry Shake network instruments), citizen science initiatives have become a feasible approach to broaden seismic detection participation. To further enhance citizen involvement in the Philippines, we have developed software that enables local seismic detection data centers to accept seismic data transmissions from citizen sensors. Our software, powered by open source libraries, is also freely available for use by other data centers. This will allow any interested individuals or small institutions to set up their own seismic detection system, leading to wider citizen participation in seismic detection efforts.*

1. Introduction

Seismic detection networks have traditionally comprised large agencies and universities, relying on expensive sensors, equipment, and proprietary software. However, the emergence of IoT technology has paved the way for cost-effective seismic sensors, empowering citizen scientists in earthquake detection. Additionally, recent strides in the open-source community have granted smaller institutions access to industry-standard seismological software and libraries. Despite these advancements, a crucial component—the seismic data transmission protocol—remains lacking, particularly for citizen science applications. This presents a unique opportunity to upgrade the Philippines' seismic network, adapting existing software, libraries, and protocols to create a seismic network focused on citizen science. In this light, encouraging the involvement of citizens promises to enhance community education which also beneficially increases traditional network density (Lukyanenko, 2019). An approach towards this is to provide timely earthquake notifications and a public portal displaying live data and seismic events. Furthermore, open data access fosters community engagement by providing much needed data for seismological research. Finally, adherence to open-source principles and open data policies furthers the ongoing scientific efforts and collaboration (Lagmay, 2018).

This paper explores the process of creating a citizen science seismic network tailored for research and education. We begin with a comprehensive literature review covering seismic data communication protocols, processing softwares, and an overview of local and international seismic networks. In the Methodology section, software development and deployment specifics are detailed in terms of motivations that guided the design. The current state of the new network and community participation are then examined in the results section.

The discussion culminates with recommendations for the software and network, concluding with the project's key contributions.

The project aims to enhance awareness of the general public to earthquake hazards and promote risk mitigation. By providing an avenue for seismic network engagement, we empower the general public with valuable insights into seismic activity which contributes to improved preparedness and safety in earthquake prone regions, extending the project's impact beyond scientific research.

2. Literature Review

In this section, we examine existing seismic communication protocols with the aim to identify critical feature gaps. We also explore open-source seismological softwares capable of fulfilling key functions such as event detection and data archival. Additionally, we assess the current state of seismic detection networks and sensors in the Philippines, identifying areas for improvement. Finally, drawing inspiration from international seismic networks, we extract valuable lessons that inform our project's direction.

2.1 Communication Protocols

SeedLink is an open-source, near real-time seismic data acquisition protocol. An initiative of the GEOFON Project of GeoForschungsZentrum (GFZ) Potsdam with contributions of several other institutions, it was originally developed to link together different Seismological Communication Processor (SeisComP) nodes (Hanka, Heinloo and Jaeckel, 2000). Since its inception more than two decades ago, it has become the de facto standard protocol for seismic data transmission. Based on TCP, it was designed so that all connections are *initiated by the client* which will receive the seismic data in a FIFO manner (gempa GmbH, GFZ Potsdam, no date). This detail highlights the traditional setup of seismic networks where data center nodes draw data from other openly or institutionally accessible data centers. Meaning, transmission is unidirectional and node authentication is not required due to only involving trusted parties. Many established seismic networks reliably use SeedLink; however, the new requirements of a citizen science tailored networks renders SeedLink to be lacking in certain features.

DataLink is a near real-time, open-source, TCP-based transmission protocol similar to SeedLink. Developed by the Incorporated Research Institutions for Seismology (IRIS), it is notable in that it supports generic payload and bidirectional data transfer. In contrast to SeedLink, a client may send data to the server or receive requested data streams (EarthScope, no date). This means that DataLink may be used to facilitate data transmission from citizen scientists to a central server. Tools such as *slink2dali* and *RingServer* make DataLink highly compatible with SeedLink protocol, effectively easing its adoption within traditional software and hardware. This is further discussed in the Methodology section.

Common Acquisition Protocol Server (CAPS), designed by Gempa GmbH, is a proprietary data acquisition protocol developed as an industry-grade alternative to SeedLink. It has improved features such as bidirectional data transmission, handling of redundant connections, and authentication-secured connections (*gempa GmbH - CAPS*, no date). These are features that are desirable for a citizen science network. Raspberry Shake, S.A. (RShake), a company known for its low-cost seismographs and the RShake Network, utilizes CAPS protocol in its data centers to provide a paid service for real-time streaming of data from their network and into acquisition programs like SeisComP (*Real Time Data - Raspberry Shake*, 2023). Additional discussion of the RShake network can be found in section 2.4.

Table 1 summarizes the important considerations in choosing a suitable communication protocol for a citizen science seismic network. CAPS, despite fulfilling the majority of the needed features, cannot be used due to its proprietary license. A necessary feature is the ability to transfer data from client to server which SeedLink doesn't support. Majority of the protocols lack redundancy, which is important for localized detection within a network. Authentication is also an essential feature for security and network data integrity. Given that DataLink already supports the desired client-to-server transmission, and with its being open-source, we decided to improve the protocol by adding authentication and redundancy features to it.

Table 1. Comparison of common communication protocols used in Seismic Networks

	Redundancy Handling	Near Real-time	Open Source	Data Transfer Clnt-to-Srvr	Data Transfer Srvr-to-Clnt	Authentication
Seedlink 3.0	**No**	Yes	Yes	**No**	Yes	**No**
CAPS	Yes	Yes	**No**	Yes	Yes	Yes
DataLink	**No**	Yes	Yes	Yes	Yes	**No**
Needed Protocol	Yes	Yes	Yes	Yes	Yes	Yes

2.2 Seismic Data Processing and Event Detection Software

Two widely used open-source seismic processing softwares are SeisComP and Earthworm. *SeisComP*, initiated in 2001 for the GEOFON program by GFZ and jointly developed by gempa GmbH since 2008, it transitioned to open-source in 2014 and is licensed under GNU AGPLv3. It employs C++ and Python in a modular architecture, encompassing data acquisition, processing, utilities, and GUI modules (Weber *et al.*, 2007; *gempa GmbH*, 2008). SeisComP utilizes SeedLink for real-time seismic data retrieval, SDS structure and miniSEED for archiving, and provides accessibility via an FDSNWS module. *EarthWorm*, originating in 1993 at the USGS and currently managed by Instrumental Software Technologies, Inc., operates under the MIT License. Written in C, EarthWorm employs a distributed module architecture, supporting various data acquisition protocols and processing algorithms (Friberg, 2019). Unlike SeisComP, it lacks a built-in GUI, central database, notification system, and post-processing, with these functions fulfilled by external applications and ANSS Quake Monitoring System or AQMS (Olivieri and Clinton, 2012). Prominent users of SeisComP include GEOFON and the Raspberry Shake Network (*SeisComP FDSNWS - DataSelect*, no date)., while EarthWorm and AQMS find use with Pacific Northwest Seismic Network and the Southern California Seismic Network, adopted as a standard by ANSS and USGS-funded networks for seismic data analysis, reporting, and archiving (Hartog *et al*., 2019).

ObsPy is another open-source seismological software. However, unlike SeisComP or EarthWorm, ObsPy is a library for the usage of Python on seismological programming. At its core, it provides programmatic access to the waveform as a generic time-series data. With a thriving developer community spanning over a decade, ObsPy has expanded with a rich set of tools for signal processing, data analysis, visualization, and convenience scripts (Megies *et al.*, 2011). ObsPy excels in manipulating various file formats, and enabling programmatic connection to data centers worldwide. Currently, it supports 30 waveform formats, and can retrieve data and metadata from 31 FDSN-member data centers (*obspy.core.stream.read — ObsPy 1.4.0 documentation*, 2022). A desktop application based on the ObsPy library is *rsdup*. This application provides continuous trace displays, sudden motion monitoring, and historical data replay, with a specific focus on Raspberry Shake devices. It is also an open-source project with active development (Nesbitt, Boaz and Long, 2021). Its architecture emphasizes extensibility, making it a valuable resource for those seeking to customize and expand its capabilities. rsudp caters more toward hobbyists, offering a beginner-friendly platform for working with seismic data.

SeisComP is an industry-standard choice with a well-designed architecture and comprehensive built-in features. In contrast, EarthWorm boasts a rich history of nearly three decades of open-source development, although it relies on non-standard wrappers for higher-level post processing. Widely adopted by networks worldwide, both packages are mature and reliable solutions for automatic data acquisition and processing. Alternatively, ObsPy is a newer software solution that bridges seismology with programming, enabling the use of Python for specialized seismological tasks. It is the foundation of rsudp, which offers citizen scientists an approachable entry point to seismology.

2.3 Philippine Seismic Network

In 1992, the Philippine Institute of Volcanology and Seismology (PHIVOLCS) took a significant step forward by establishing the Digital Strong Motion Accelerograph Network (Kurita *et al.*, 1999) with 4 strong-motion instruments distributed within Metro Manila (DOST-PHIVOLCS, 2021). As shown in Table 2, PHIVOLCS has consistently advanced its monitoring capabilites over the years. Through partnering with various local and international intitutions, the network has gone from the 4 strong-motion stations in 1992, to 107 seismic stations and 8 volcanological observatories by 2022 (Hernandez and Castillejos, 2010; DOST-PHIVOLCS, 2022).

Table 2. Philippine Seismic Network Milestones (Torregosa, Sugito and Nojima, 2001; Hernandez and Castillejos, 2010; Melosantos, 2015; DOST-PHIVOLCS, 2021; DOST-PHIVOLCS, 2022)

Year	Milestone
1992	PHIVOLCS established the Digital Strong Motion Accelerograph Network, starting with 4 strong-motion instruments
1999	8 sites of digital strong-motion accelerograph deployed at various sites characterized by distinct geological conditions, with each instrument enclosed from seismic noise via a *2-square-meter vault*
2001	34 seismological stations built
2011	30 satellite-telemetered (unmanned) seismic stations, 29 staff-controlled (manned) seismic stations, and 6 volcanological observatories
2012	12 out of 70 seismic stations are utilizing broadband seismometers for the collection of data
2022	78 unmanned seismic stations, 29 manned seismic stations, and 8 volcano observatories

The Philippine Seismic Network is a good example of a traditional seismic network which relies on expensive, large scale equipment to detect and dissimenate seismic information. Illustratively, one such investment is the most recent monitoring establishment of PHIVOLCS amounting to approximately 20 million Philippine pesos or about 353,000 US dollars (Marfal, A.M., 2021). Overall, the network provides reliable seismological and early warning services; however, access to real-time and archived data from these networks remains restricted from the general public.

2.4 Global Seismic Networks

Collaborative efforts across continents have led to the establishment of different global seismic networks that play crucial role in advancing our understanding of Earth's geophysical processes. European and American initiatives, as well as citizen scientist-driven networks, are instrumental in these advancements. In Europe, the GEOFON program, initiated in 1993, operates a global seismic network comprising of distributed stations equipped with broadband and strong-motion sensors. The GFZ data center, part of GEOFON, offers real-time data access and houses a substantial 120 TB seismic data archive (Quinteros *et al.*, 2021). ORFEUS, a collaborative non-profit foundation, promotes seismology in the Euro-Mediterranean area, collecting, archiving, and distributing seismic data and providing strong-motion seismology services through Rapid Raw Strong-Motion and Engineering Strong-Motion Databases (Luzi *et al.*, 2019; ORFEOUS Foundation, no date; *ESM* , no date). In the United States, the United States Geological Survey (USGS) oversees the Advanced National Seismic System and the Global Seismographic Network, providing publicly available earthquake information (Earthquake Hazards Program, no date). University research consortia IRIS and UNAVCO collaborate under the Earthscope Consortium to advance geophysics and geology research, with IRIS operating the SAGE Facility and UNAVCO running the GAGE Facility (*About EarthScope | EarthScope Consortium*, no date). The IRIS Data management Center (IRIS DMC) serves as the central node for real-time data access, supporting global seismographic research. Notable users of IRIS DMC data include the Global Seismographic Network. These networks provide and promote publicly available real-time earthquake information, ground motion parameters, and spectral amplitudes displayed using list and map views through their respective web portals.

The Raspberry Shake (RShake) Network, formerly known as the Public Seismic Network (PSN), stands as a remarkable citizen-scientist-driven earthquake monitoring network (Raspberry Shake, S.A., 2016). The network is built with Raspberry Shake devices which are Raspberry Pi sensors equipped with ground motion sensors and processing software and hardware. Remarkably, these devices are ranging from $175 to $1415, owing to the widespread adoption of cost-effective sensors (*Earthquake and Infrasound Monitors*, no date). Real-time seismic data of each device is locally accessible to its owner via SeedLink, OSOP Wave Server (OWS) or Shake UDP Broadcast. On the other hand, citizen scientists willing to pay the network's streaming service may access real-time data streaming from the entire network, via the CAPS protocol. Earthquake events and information about devices connected to the RShake network are accessible through their portal (*Station View: Raspberry Shake Network & EQ Activity Map*, no date). As of writing, there are about 3400 devices connected (both online and offline) to the RShake network.

2.5 Summary

To create our seismic network for citizen science engagement in the Philippines, essential features must be identified. Our analysis of communication protocols highlighted the need for *authentication, redundancy, and*

bidirectional communication. To implement these features, we have determined that extending DataLink through RingServer presents the most pragmatic approach. Furthermore, we have surveyed various open-source seismological software options, such as SeisComP, Earthworm, and rsudp, and observed that different seismic networks employ different software solutions. This underscores the value of establishing a network infrastructure *capable of seamlessly integrating various processing software*, ensuring flexibility and preventing institutions from being confined to a single option. We have assessed the existing Philippine network, a traditional model characterized by limited public access to ground motion data archives and real-time sensor data. This network relies on costly, bulky sensors, mainly accessible to government-funded institutions. Conversely, many international seismic networks feature *publicly accessible web portals*, typically offering *earthquake lists and interactive earthquake maps*. Additionally, networks often provide web services like *FDSNWS for accessing data archives and station metadata* for real-time data streaming. While most networks offer open access to their real-time data streams, the RShake network, similar to our envisioned network, does not provide open access to its real-time data. Drawing inspiration from well-established networks like IRIS, ORFEUS, USGS, and the innovative citizen science efforts of RShake, we aim to incorporate these key features into our own seismic network.

3. Methodology

This section uncovers the details of the software design for the EarthquakeHub network. Motivated by the previously identified design parameters, we take apart each component of the software suite and discuss its functionalities and peculiarities in its design in the context of a seismic detection network centered around citizen science. Shown in Figure 1 is an overview of the software architecture. The network's primary function is to facilitate the transmission of data from a citizen scientist's device to a central server capable of seismic event detection. On the left is the citizen scientist's device equipped with a sensor and a program enabling data transmission over the Internet. On the opposite side is the central server that receives the ground motion data. This server is equipped with a data acquisition program that interfaces with a seismic data processor and a web application, both harnessing the collected sensor data.

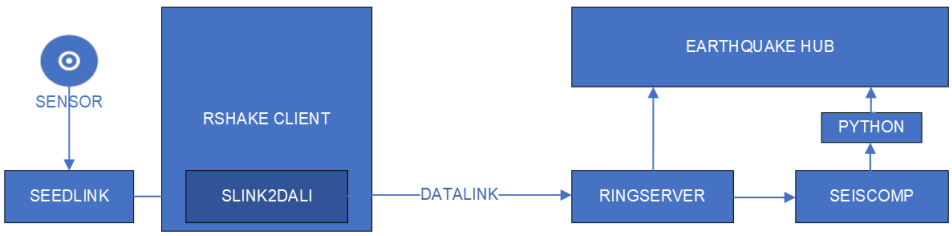

Figure 1. Flow of data from station to server

This section follows a ground-up approach, beginning with the low level components: Slink2Dali, DataLink, and RingServer. These form the bedrock for transmitting data across the network. Moving up, we delve into the processing software SeisComP which performs the seismic event detection using the collected data. Next is the development of web applications, the public interface of the EarthquakeHub network showcasing the products of the network. Lastly, the deployment strategy for the software suite, both on the servers and on the Raspberry Shake devices. This section aims to capture the entire software design and implementation process, presenting a complete picture of the EarthquakeHub network's software ecosystem.

3.1. DataLink, RingServer, and Slink2Dali

At the lowest level of the client-server data transmission system are the Slink2Dali and RingServer programs. Both programs are written in C and interact directly with data packets. Originating as open-source software from IRIS, we modified these programs to fit our architecture. Slink2Dali resides on the client device, and forwards the sensor data to the RingServer. The RingServer, deployed on the central server, receives and manages this data. Data is exchanged between the two programs using the DataLink protocol.

DataLink is chosen as the communication protocol due to its being an open-source protocol supporting a write command from the client, e.g. the sensor sending data to the server. In contrast, the widely recognized SeedLink protocol solely accommodates streaming from the server to the client. With SeedLink, a client initiates a connection, requests a specific stream, and receives either real-time or archived data. Conversely, DataLink can stream bidirectionally. After connection establishment, a client negotiates the desired stream

and opts to either receive or transmit the data. In citizen science applications, the usual connection setup necessitates the capability for participants to push data from their devices to the server, as opposed to the server drawing data from the citizen scientists' devices. This is because citizen scientists typically operate within residential networks secluded behind a Network Address Translation (NAT) service. This renders the actual devices inaccessible to the central server for data retrieval. Therefore, these constraints dictate that the central server, with its public accessibility, be the recipient of data pushed by the client devices.

RingServer was chosen as the data-acquisition program for its being open-source and its utilization of the DataLink protocol along with SeedLink and HTTP protocols. This fits well in our design to be the first receiver of data since this data will be processed by SeisComP (which uses SeedLink) and our web applications (which uses HTTP). To further adapt the program to the design specifications, we had to modify the original source code made available by the IRIS organization and add two important additional features.

The first modification is the addition of the authorization command. When allowing clients to write data into the server, it is important to be able to discriminate among clients based on their write permissions. Originally, RingServer can discriminate between client connections via manually written IP addresses in the configuration file. However, in citizen science applications, there is no way to identify user IP address pre-connection since it is often dynamic and hidden behind NAT. To solve this problem we made use of JSON web tokens (JWT) and our own external Authentication API to know if a client requesting to send data into the RingServer has permission. The way this works within the DataLink protocol is as follows:

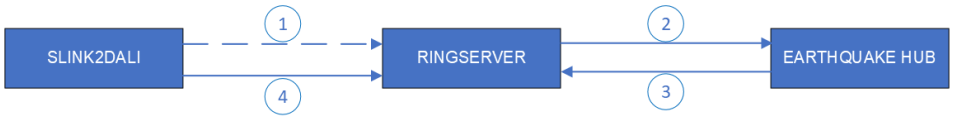

Figure 2. Slink2Dali-RingServer authorization procedure.

1. After the negotiation commands, the client sends an authorization request to the RingServer: `AUTHORIZATION size\r\n [token]`
2. The RingServer forwards this token to the external Authentication API
3. The external Authentication API replies with information on the writing privileges of the client
4. Once authorized, the RingServer accepts the succeeding WRITE commands from the client. Otherwise, the connection is closed.

This approach necessitates client pre-registration to acquire authorized tokens for writing to the server. Such a service is designed to be provided by the external authentication API, discussed under the Web Applications section. The reason for outsourcing the authentication service is twofold: first, it keeps the RingServer program from having to maintain a local database of users; and second, it allows for the network to have multiple instances of RingServers with only a single centralized database of users. This means that not only can contributors participate as sources of sensor data, but small institutions may be set up to participate as localized detection systems within the network.

The second modification is the handling of redundant connections. Due to the possibility of having multiple localized detection systems within the network, each having their own RingServer, the sources of sensor data may choose to establish parallel connections to these servers. This means that a set of connections as shown in Figure 3 is possible.

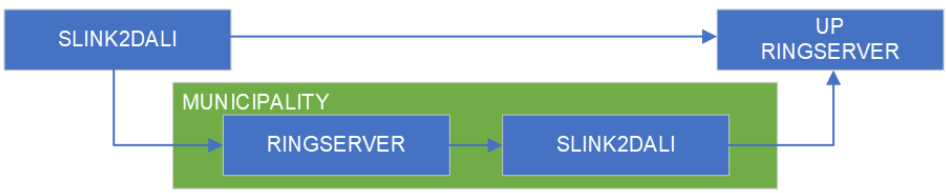

Figure 3. A station establishes a connection to both the central server and a local municipality server. The latter however also forwards their data to the University of the Philippines (UP) server. This creates a redundant connection where the data of the source station is received by the UP server twice.

Although this creates a problem where the last station receives duplicate data, it beneficially improves network resiliency and allows better localized event detection. In the context of data processing softwares, with all things being equal, a RingServer receiving data from a client geographically closer to it should be able to more quickly provide this data to a processor as compared to another RingServer farther from the client. To allow this, RingServer must be able to drop duplicate packets, effectively avoiding multiple copies of the same packet written on a single stream. The way this is incorporated within a stream that uses the DataLink protocol is that upon receiving a write command, the new packet's *start-time* is compared to the *end-time* of the packet that was written before it. The new packet will only be written if its start-time is later or equal to the end-time of the latest written packet. The rationale is that the DataLink packet timestamps depend on the payload, i.e. a miniseed packet from the Raspberry Shake which uses a single clock timer for all packets. Thus, it is reasonable to perform comparison between timestamps and use this to decide packet ordering. Notably, identical packets shall have identical start and end times and hence can be identified on the receiving end to be duplicates. Note that similar to the original RingServer, no packet re-ordering is performed. The packets are all received and written in the order they arrive, with only the difference that packets failing the duplicate check are dropped.

Slink2Dali is chosen as the client program as a result of selecting DataLink as the communication protocol, and the fact that the Raspberry Shake devices that serve as clients in the network, come with a SeedLink server built-in. Slink2Dali fits this requirement for its being a program that converts SeedLink packets into DataLink packets and sends them to a remote server via the DataLink protocol. The modifications performed on the original program are made simply in accordance with the changes on the RingServer: the new authorization command, and the message exchange for when a packet is dropped due to redundant connections.

3.2. Seismic Data Processor

SeisComP functions as the central processing software utilized by the EarthquakeHub network. Having been open-sourced by gempa and GEOFON, SeisComP fits well with the other open-source programs used in the network architecture. Its primary role revolves around the automatic processing of seismic data obtained from the Raspberry Shake devices. This encompasses a range of tasks, including the triggering of picks, the calculation of earthquake event magnitudes, triangulation of its precise origins, and organization of the seismic data into the archive server. The seismic data used to perform these calculations are acquired via a seedlink *chain* connection to the RingServer process running alongside SeisComP's seedlink module. Subsequently, the identified picks and seismic events are routed to the backend of the EarthquakeHub web application for archival and simultaneously sent to the frontend for visualization. The communication between the EarthquakeHub backend and SeisComP is facilitated through the execution of Python plugin scripts, which are running as background services, designed to actively monitor and capture picks and seismic events published by the SeisComP messaging system (seiscomp.de, no date)

It should be acknowledged that the EarthquakeHub network's reliance on SeisComP as its seismic data processing software is not exclusive. The network remains flexible to adopt alternative seismic data processing solutions, such as the RSUDP and Earthworm, in lieu of SeisComP. This showcases the adaptable nature and modularity of the network's architecture, accommodating diverse software options while maintaining its core functionality.

3.3. Web Applications: EarthquakeHub and RShake Client

The web applications serve as the public facing component of the citizen science network. With the data transmission over the Internet made possible by the RingServer and Slink2Dali programs, and with this data processed via SeisComP, the results are made publicly available through the EarthquakeHub web application. It is also where other services concerning citizen scientist participation such as account registration and data streaming monitoring are done. On the other hand, configurations and controls specific to a sensor device are done through a second web application called the Raspberry Shake (RShake) client. This latter web application is accessed by connecting to the Raspberry Shake device itself.

The EarthquakeHub is implemented following the frontend-backend software pattern and is written using the MongoDB, ExpressJS, ReactJS, and NodeJS (MERN) stack. Utilizing the modular approach for the entire software suite, the frontend and backend components allows for a separation of concerns between the user interface and the system logic, respectively. *The frontend* is centered around a map view of the citizen science network showcasing the information provided in Figure 4. It is focused on real-time capabilities of event

detection and trace viewing to enhance user engagement. It also features web-push notifications for near real-time alerts of earthquakes with magnitude greater than 5.5. A citizen scientist may also register to the network to have their devices included on the map view.

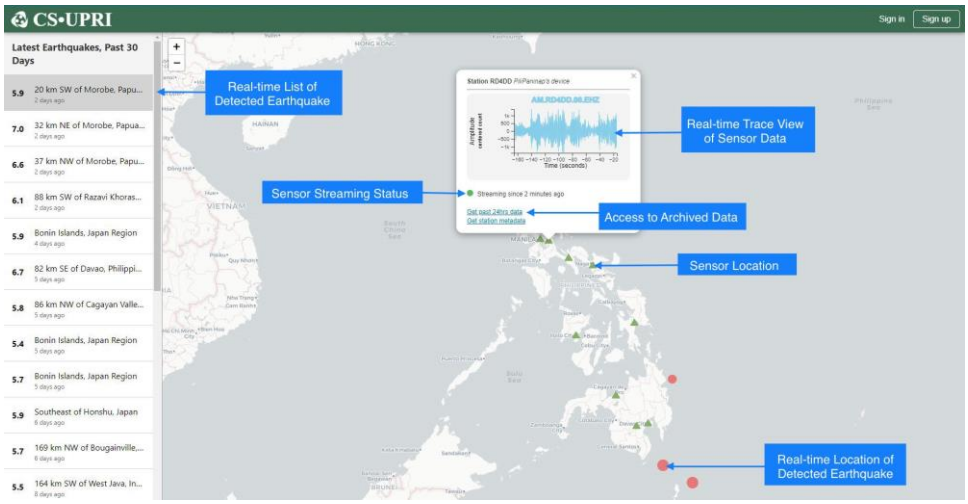

Figure 4. EarthquakeHub Frontend.

The backend provides all the information presented on the user-interface. More specifically, the backend has two major roles: first, as the communication link between the EarthquakeHub frontend and the server-side processes which covers SeisComP, RingServer, and the database; second, as the entire network's authentication and authorization server. For the first role, the backend uses HTTP endpoints for static information and server-sent-events (SSE) for real-time data and updates. Specifically, SeisComP sends its processing results to the backend via HTTP POST requests, while RingServer streams data and its diagnostics via SSE. These information are both stored into the database and forwarded directly to frontend clients that are online. Conversely, the real-time trace view on the frontend is implemented via a direct connection to the WebSocket endpoint of the RingServer. The second role of the backend concerns the membership of the citizen scientists into the network. To be able to share sensor data, a citizen scientist must first register their devices into the network. This service requires a database of registered users, and consequently another service to authenticate the identity of these users or clients. This should be done for different types of clients; namely, the data sources sending the seismic data, the browsers accessing the user accounts, and the RingServers receiving the seismic data.

To meet these requirements, the open standard JSON Web Token (JWT) is used to perform the authorization and authentication. A JWT is an object that contains a set of *claims* and a *digital signature* encoded into a Base64URL string. The claims represent the information that will be used to infer what the JWT owner is authorized to, and the digital signature contains the signature of the authorizing figure (in this case the backend) to securely verify that the token is not tampered with and that it was originally created by the backend itself. In short, the JWT can be used to prove that the client is who they say they are, and the authorizing figure can use this accordingly to give them permissions. Having the compact format of a Base64URL string, the JWT can be used easily within the use case of the three clients:

1. Slink2Dali authenticating itself to a RingServer via the AUTHORIZATION command
2. Citizen scientist accessing their EarthquakeHub profile using HTTPS cookies
3. RingServer authenticating itself to the EarthquakeHub backend via HTTPS Bearer header

The RShake client is developed mirroring the EarthquakeHub web application. It adheres to the same frontend-backend pattern, and is also written using the MERN stack. The primary function of the RShake client is to provide a web application that acts as a user-friendly presentation layer to the execution and management of the Slink2Dali processes on a single sensor device. *Its frontend* is a single-page interface showcasing the device configurations and a list of RingServer addresses. The left panel showing the device configurations highlights an important link between the EarthquakeHub and the RShake client. Namely, to be able to send data within the network, the Raspberry Shake device must be linked to an account by registering or signing-in with a citizen scientist account on the EarthquakeHub. Then, on the RShake client, they shall register their device by using their EarthquakeHub credentials and physical information about the device.

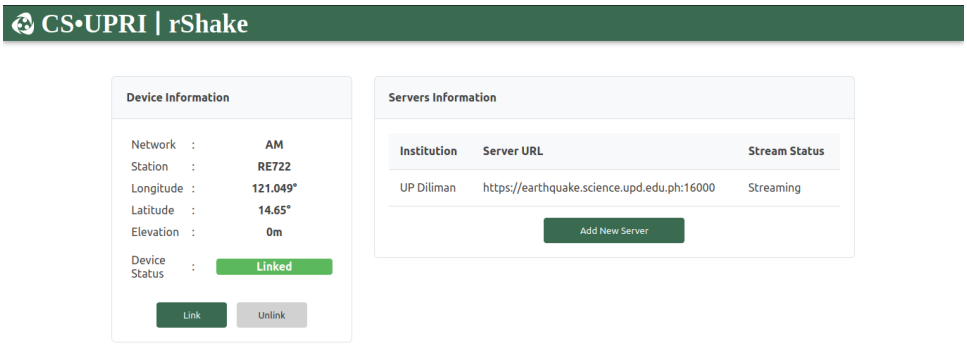

Figure 5. RShake Client Frontend Interface.

The other panel showing the destination RingServers hosts represents the RShake client *backend*. This is because for each host on the list, the backend spawns an instance of Slink2Dali, monitors its status, and reports it to the frontend. This means that Slink2Dali is embedded as a subprocess in the backend component. To access the client application, the user simply has to ensure that the Raspberry Shake is connected to the same network as their computer. Through a browser, they can visit the application at *http://rs.local:3000*.

3.5. Deployment

In general, installing programs into any computer machine or device would require careful consideration of its software dependencies and hardware compatibility. This task proves to be tricky and laborious specially when done for a software suite with a number of microservices. Monitoring each of these processes, once deployed, is also a tedious task. One deployment strategy befitting these complications is to use a *containerization technology* such as *Docker*. The way containerization solves these complications is by providing application-level virtualization in the form of containers. A container simulates a complete and isolated computing environment within which a program can be encapsulated with its configuration files and software dependencies. It can be thought of, simply, as a virtual computer within a computer. A container can be deployed to any hardware system supporting the technology. Additionally, containerization solutions provide tools to orchestrate, monitor, and automate the deployment of containers. Hence, we can containerize each program and deploy all of it as a cluster of containers within a single hardware. This results in a more streamlined approach to the deployment and management of the microservices.

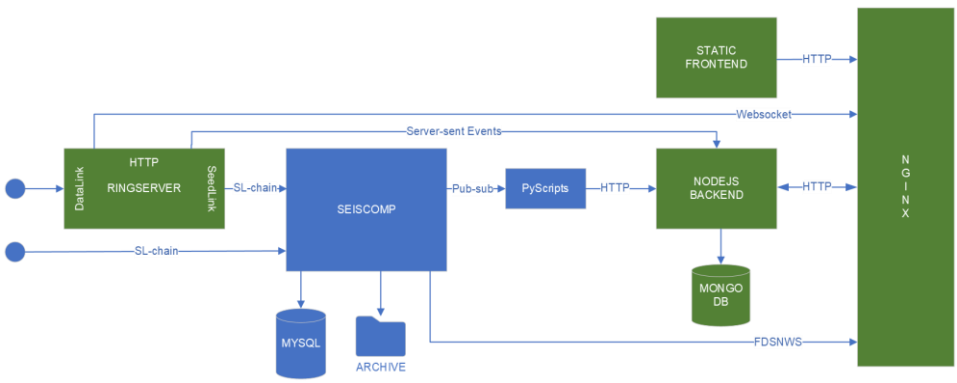

Figure 6. Deployed processes on the server. Those in green are run as containers, while those in blue are run on server OS. Notably, the cluster may still be used with a detection software other than SeisComP.

The diagram illustrates the deployment of the container cluster on the central earthquake detection server, alongside parallel-running processes. It should be noted that the NGINX container serves as an HTTPS reverse proxy for security and ease of SSL certificate management. This means that all incoming HTTP/S requests first come through the NGINX server. Subsequently, it forwards the request to the correct microservice that shall handle the request, otherwise the NGINX server responds with the correct HTTP error message. Using the same approach, the RShake client programs are deployed on the Raspberry Shake using Docker containers.

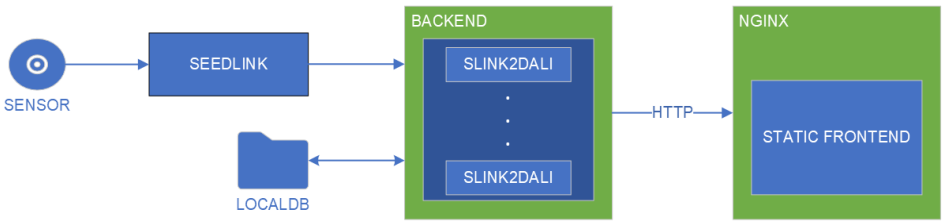

Figure 7. Deployed processes on the Raspberry Shake. The processes shown in green are run as containers, while those in blue are built-in processes in the Raspberry Shake device.

4. Results and Recommendations

In the course of this work, we have developed a comprehensive software package exclusively utilizing open-source tools. This software package is able to perform ground sensor data acquisition, seismic event detection, and the public showcasing of network products. As of writing, the network is actively undergoing public testing and refinement. It currently incorporates 12 citizen science Raspberry Shake stations that are registered and operational, utilizing our client software to stream data within the network. This data, coupled with publicly accessible real-time data from international networks such as GEOFON and IRIS/USGS, allows the network to detect earthquakes exceeding a magnitude of M5 in the local and neighboring regions, subsequently issuing timely notifications. Network data is readily accessible through the web application and our FDSN web service. Looking ahead, several improvements and extensions are expected. Firstly, the existing data acquisition system, RingServer, shall benefit from security enhancements, particularly in the adoption of JWT for authorization within the DataLink protocol and the incorporation of encryption to solidify security. Secondly, data integrity across the network can be elevated by introducing quality assurance and control stages into the archival pipeline. Lastly, the web interface can be further enhanced by incorporating additional features such as seismic event replays, spectrogram views, fault line map displays, and various convenience services catering to both researchers and hobbyists.

5. Conclusion

In summary, this software suite represents a significant contribution to the field of seismology and community-driven scientific endeavors. It was motivated by the unique opportunity to enhance the Philippine seismic network through the proliferation of low-cost IoT sensors and the availability of open-source software solutions. Leveraging citizen science efforts, we aimed to create a more densely interconnected network while fostering community awareness in disaster risk reduction against earthquake hazards. We examined various communication protocols, seismological software systems, including the Philippine network, and those employed by larger institutions. This comprehensive analysis informed the architecture and setup of our network. Subsequently, we have provided a wealth of information, enabling other institutions to replicate our network while equipping developers with the resources to contribute to the source code or build upon our foundation.

Key features of our software suite include near-real-time event notifications, accessible event lists, station locations, and stream traces via a user-friendly web application, *earthuake.science.upd.edu.ph,* accessible to the general public. Researchers and hobbyists gain access to near-real-time and archived data, along with opportunities to participate by deploying their own ground motion sensors. Lastly, engineers and developers may benefit in our commitment to openness that is reflected in our open-source, modular, containerized design approach, supported by a detailed system documentation available at *upri-earthquake.github.io*.

As we look to the future, this project holds the potential to ignite further advancements in seismology, offering new avenues for research, education, and collaborative open-source development. It encourages citizen science efforts not only in ground motion detection but also in earthquake systems engineering. In the Philippine context, this modern network design greatly complements traditional systems and paves the way for forthcoming software and hardware enhancements. In a nation consistently threatened by seismic disasters, initiatives like this one serve as invaluable resources in fortifying societal and economic resiliency.

6. Acknowledgements

We extend our sincere appreciation to the individuals and open-source projects that have been instrumental in making this project a reality. Special thanks to Chad Trabant of IRIS, whose DataLink protocol have served as the foundation of our networking system. We also acknowledge Philip Crotwell of the University of South Carolina for his initial work on JWT-based authorization in the DataLink protocol, upon which our implementation is built. Additionally, his *seisplotjs* library has significantly enhanced our web interface. Lastly, we express gratitude to the University of the Philippines, Diliman, College of Science, for providing us with the necessary resources for the development and deployment of our system.

7. References

About EarthScope | EarthScope Consortium (no date). Available at: https://www.earthscope.org/about/earthscope/ (Accessed: 30 Sep. 2023).

DOST-PHIVOLCS (2021). *Press Release: DOST-PHIVOLCS Establishes its 100th Strong Motion Station.* Available at: https://www.phivolcs.dost.gov.ph/index.php/news/11325-press-release-dost-phivolcs-establishes-its-100th-strong-motion-station (Accessed: 30 Sep. 2023).

DOST-PHIVOLCS (2022). *Earthquake Monitoring System.* Available at: https://www.phivolcs.dost.gov.ph/index.php/earthquake/earthquake-monitoring (Accessed: 30 Sep. 2023).

Earthquake and Infrasound Monitors (no date). Available at: https://shop.raspberryshake.org/all-products/ (Accessed: 30 Sep. 2023).

Earthquake Hazards Program (no date). *Monitoring | U.S. Geological Survey.* Available at: https://www.usgs.gov/programs/earthquake-hazards/monitoring (Accessed: 30 Sep. 2023).

EarthScope (no date). *libdali: the DataLink client library*. Available at: https://earthscope.github.io/libdali/index.html (Accessed: 29 Jan. 2024).

ESM (no date). Available at: https://esm-db.eu/#/about/overview (Accessed: 20 Sep. 2023).

Friberg, P. (2019). *Earthworm Documentation Table of Contents.* Available at: http://www.earthwormcentral.org/documentation4/index.html (Accessed: 14 Sep. 2023).

gempa GmbH (2008). 'The SeisComP seismological software package'. doi: https://doi.org/10.5880/GFZ.2.4.2020.003

gempa GmbH - CAPS (no date). Available at: https://www.gempa.de/products/caps/ (Accessed: 28 Sep. 2023).

Hanka, W., Heinloo, A. and Jaeckel, K. (2000). 'Networked Seismographs: GEOFON Real-Time Data Distribution', *ORFEUS Newsletter*, 2(3), pp.1–24.

Hartog, J.R. *et al*. (2019) 'Open-Source ANSS Quake Monitoring System Software', *Seismological Research Letters*, 91(2A), pp. 677–686. doi: https://doi.org/10.1785/0220190219

Hernandez, M. and Castillejos, V. (2010). *Seismic Observation of the Philippines*. Available at: https://iisee.kenken.go.jp/net/shiva/update/Philippines.pdf (Accessed 20, Sep. 2023).

Kurita, K. *et al*. (1999). 'Strong Motion Observation In Metro Manila, Philippines', *12WCEE 2000: The 12th World Conference on Earthquake Engineering*. Auckland, New Zealand, 30 January – 4 February 2000. Available at: https://www.iitk.ac.in/nicee/wcee/article/1931.pdf (Accessed: 30 Sep. 2023).

Lagmay, A. (2018). *An Open Data Law for Climate Resilience and Disaster Risk Reduction*. Available at: https://www.researchgate.net/publication/324121945_An_Open_Data_Law_for_Climate_Resilience_and_Disaster_Risk_Reduction (Accessed: 29 Jan. 2024).

Lukyanenko, R. (2019). 'Citizen Science 3.0: A vision for the future of digital citizen science', *International Conference on IT & Computer Science (ICITCS – 2019).* London, UK, 28-29 Dec. 2019. Available at: https://www.researchgate.net/publication/338421212_Citizen_Science_30_A_vision_for_the_future_of_digital_citizen_science (Accessed: 29 Jan. 2024).

Luzi, L. *et al*. (2019). 'ORFEUS Strong-Motion services and products for engineering seismology', *21st EGU General Assembly.* Vienna, Austria, 7-12 April 2019. Available at: https://ui.adsabs.harvard.edu/abs/2019EGUGA..2113374L/abstract (Accessed: 30 Sep. 2023).

Marfal, A.M. (2021). *Davao earthquake and tsunami monitoring facility strengthens PH disaster preparedness efforts*. PTV News. Available at: https://ptvnews.ph/davao-earthquake-and-tsunami-monitoring-facility-strengthens-ph-disaster-preparedness-efforts/ (Accessed: 20 Sep. 2023).

Megies, T. *et al*. (2011). 'ObsPy – What can it do for data centers and observatories?', *Annals of Geophysics*, 54(1). doi: https://doi.org/10.4401/ag-4838.

Melosantos, A. *et al*. (2015). 'Performance of Broadband Seismic Network of the Philippines', *Journal of Disaster Research*, 10(1), pp. 8-17. doi: https://doi.org/10.20965/jdr.2015.p0008.

Nesbitt, I., Boaz, R. and Long, J. (2021). 'rsudp: A Python package for real-time seismic monitoring with Raspberry Shake instruments', *Journal of Open Source Software*, 6(68), p.2565. doi: https://doi.org/10.21105/joss.02565.

obspy.core.stream.read — ObsPy 1.4.0 documentation (2022). Available at: https://docs.obspy.org/packages/autogen/obspy.core.stream.read.html#supported-formats (Accessed: 14 Sep. 2023).

Olivieri, M. and Clinton, J. (2012). 'An Almost Fair Comparison Between Earthworm and SeisComp3', *Seismological Research Letters*, 83(4), pp.720–727. doi: https://doi.org/10.1785/0220110111.

ORFEOUS Foundation (no date). *RRSM.* Available at: https://orfeus-eu.org/rrsm/about/ (Accessed: 21 Sep. 2023).

gempa GmbH, GFZ Potsdam (no date). *Overview — SeisComP Release documentation.* Available at: https://www.seiscomp.de/doc/base/overview.html (Accessed: 14 Sep. 2023).

Quinteros, J. *et al*. (2021). 'The GEOFON Program in 2020', *Seismological Research Letters 2021,* 92(3), pp. 1610-1622. doi: https://doi.org/10.1785/0220200415

Raspberry Shake, S.A. (2016). Raspberry Shake [Data set]. International Federation of Digital Seismograph Networks. doi: https://doi.org/10.7914/SN/AM

Real Time Data - Raspberry Shake (2023) Available at: https://raspberryshake.org/data-center/ (Accessed: 28 Sep. 2023).

SeisComP FDSNWS - DataSelect (no date). Available at: https://data.raspberryshake.org/fdsnws/dataselect/1/ (Accessed: 14 Sep. 2023).

Station View: Raspberry Shake Network & EQ Activity Map (no date). Available at: https://stationview.raspberryshake.org/ (Accessed: 30 Sep. 2023).

Torregosa, R.F., Sugito, M., and Nojima, N. (2001). 'Strong Motion Simulation for the Philippines Based on Seismic Hazard Assessment', *Journal of Natural Disaster Science*, 23(1), pp.35-51. Available at: https://www.framece.com/uploads/3/9/0/1/3901564/torregosa_nds_paper.pdf (Accessed: 30 Sep. 2023).

Weber, B. *et al*. (2007) 'SeisComP3 - Automatic and Interactive Real time Data Processing', *Geophysical Research Abstracts*, Vol. 9. Vienna, Austria, April 2007. Available at: https://www.researchgate.net/publication/323735085_SeisComP3_-_Automatic_and_Interactive_Real_Time_Data_Processing (Accessed: 30 Sep. 2023).