RESCUE: A Resilient Cloud Based IoT System for Emergency and Disaster Recovery

Taher Khan, Saptarshi Ghosh, Muddesar Iqbal, George Ubakanma, Tasos Dagiuklas
Department of Computer Science, School of Engineering
London South Bank University
London, United Kingdom
e-mail: khant6,ghoshs4,m.iqbal,george.ubakanma,tdagiuklas@lsbu.ac.uk

***Abstract*—Modern Disaster management requires technology to manage disaster mitigation, preparedness, response and recovery. Reliable data exchange, interoperability and integration, particularly with the rise of IoT devices is the biggest challenge for disaster management systems, sharing information among civil protection stakeholders working at disaster sites who use different devices and platforms to exchange such information is vital. This requires investigation and design for a disaster Management system to utilize a cloud based IoT framework to address data-interoperability and load balancing issues. In this paper, we have proposed and implemented such cloud based IoT system. Furthermore, a proof of concept has been implemented and evaluated using an android application, an IoT kit and a Cloud based middleware platform that is not only able to exchange data between publishers and subscribers but also able to perform load balancing.**

Keywords-Disaster Management; Smartphones; Internet of Things (IoT); Broker Messaging; Asynchronous messaging; MQTT Protocol.

I. INTRODUCTION

In the field of Civil Protection (CP) and Disaster Management (DM) Blue Light Services and Civil Protection organizations work together to effectively manage disasters. Information systems have become a key part of disaster management strategies leading to cloud services and data centers to be incorporated into disaster management systems. Data integration, exchange and interoperation are vital ingredients in disaster management systems, each stakeholder has specialist information which other stakeholders need access to, the exchange and interoperability of the data between stakeholders becomes a pertinent issue to be addressed in the field of disaster management technology, even more so if a disaster spans more than one country or continent. Furthermore, the rise of IoT and its potential uses in disaster management systems calls for highly accurate, reliable and fast systems as many stakeholders need to work together and need access to the same dataset to facilitate efficient disaster recovery.

In [1] Snyder et al state an effective method for data, integration and interoperation is to use enterprise messaging middleware. As such a messaging architecture has been developed that effectively exchanges data through messaging, the system allows passersby or victims to push a message asynchronously by using the MQTT protocol through a simple IoT device or a smartphone and send data from disaster sites to trained first responders and emergency services. Smartphones, wearable devices and Internet of Things devices have useful features that allow rich and important data such as location, temperature, proximity, motion sensor data to be obtained and utilized by emergency services to manage disasters, these devices also have high availability among the public.

In this paper, we have proposed a system that uses a cloud-based broker messaging platform as a common interface to overcome some data integration and load balancing issues, allowing exchange of data among disaster stake holders using heterogeneous devices. The architecture has taken into consideration the individuality and privacy of stakeholders therefore allowing decoupling, scalability and security of applications.

The remaining parts of this paper is organized as follows, Section II discusses related work, Section III describes the proposed disaster management system, Section IV presents the performance evaluation and lastly the paper conclusion is presented in Section V.

II. RELATED WORK

In 2015 Facebook launched “Safety Check” a feature of Facebook’s “Crisis Response “service. [2] This service was released by Facebook in the response the 2011 Japanese earthquake and tsunami. This system is available as a location based mobile app/feature deployed by Facebook in times of crisis/disaster allowing users to connect with friends, family and emergency services during times of disaster.

More recently the Internet of Things (IoT) and cloud computing paradigms have been the focus of much DM research and development (R&D) activity. Ray, et al [3] conducted a study of the current state of the art IoT based disaster management solutions and future directions and potential challenges.

Taking a related approach Levius et al [4] in 2017 published a case based overview of Information and Communication Technology (ICT) to support DM in the Caribbean Community Region. The investigation was based on interviews conducted with managers of National Disaster Organizations (NDO) in 17 participating countries in the Caribbean region. Their conclusions include recommendations for the R&D of Web Emergency Operation Centre (WebEOC) applications; Also Alert, warning, and notification applications as essential components for DM and recovery solutions.

Alternative approaches to the application of ICT in DM solutions includes state of the art contributions in 2018 from Ali et al [5] whose work on "Disaster Management Using D2D Communication with Power Transfer and Clustering Techniques", proposes development of Simultaneous Wireless Information and Power Transfer (SWIPT). This technology utilizes Device-to-Device communication and Energy Transmission/Harvesting Techniques to ensure mobile devices in remote disaster zones receive both emergency power and communication in a single transmission.

In 2017 Tani et al [6] published a model and simulation of DM and recovery of communications in disaster zones utilizing High-Throughput Satellites to respond to sudden increases in mobile communications demand in and around disaster zones. The techniques proposed apply bandwidth manipulation utilizing Radio Spectrum Management and, Frequency Control Techniques.

Also in 2017 Li et al [7] proposed a vehicle-assist resilient information and network system composed of 3 major components: "(1) smartphone apps that provide functions of SOS reporting, life and medical resources request/provision, and safe road navigation; (2) mobile stations that assist data exchange between smartphone apps and servers; (3) geo-distributed servers that collect user data, conduct distributed data analysis, and make disaster management decisions." [7] The work includes simulations for performance evaluation of the proposed algorithms given a range of disaster management tasks.

III. Proposed System

The proposed system is a cloud-based infrastructure that allows heterogeneous devices to send and receive data using a broker-messaging server allowing stakeholders to exchange data using the publish/subscribe pattern. A custom disaster management app was built on the android platform to send and receive different types of data from the broker. The subsequent sub-sections will discuss the layers of the three-tier architecture.

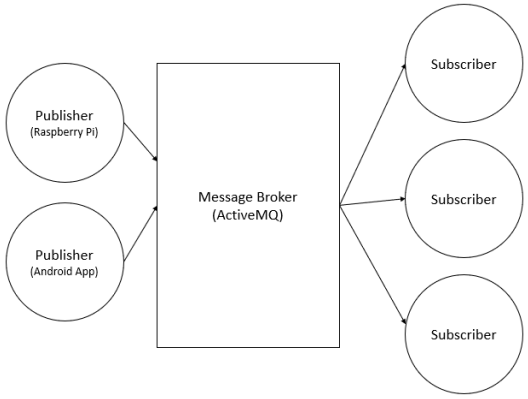

Figure 1. Overview of the pub/sub Architecture

A. *Android Client*

The Android client which is developed using Java is designed to send and receive text, images and location data to the broker messaging server, subscribers are envisaged to receive this data through asynchronous messaging. The Android client consists of a reporter and responder mode from the user perspective, in the reporter mode the app picks up the users location and allows the user to send vital data to the broker messaging server, the responder mode allows local responders to receive an alert and sends the data and presents the disaster type and distance, if the responder accepts then the app redirects the responder to google maps and directs them to the disaster scene, the reporter is able to view the location of responders.

Figure 2. Android client collecting various data formats

The Android client connects, publishes and receives data to a "topic" in the ActiveMQ server using the MQTT protocol. The Eclipse Paho Android Project provides the MQTT support, Eclipse state the Paho Android service "has been created to provide scalable open-source implementations of open and standard messaging protocols aimed at new, existing, and emerging applications for Machine-to-Machine (M2M) and Internet of Things (IoT)" [8]. The android client is also able to subscribe to the topic to receive asynchronous messages from the broker.

B. *Raspberry PI Client*

On the sensor layer of an IoT architecture, many IoT devices are present such as smartphones, motion and touch sensors, to reflect this we have added a Raspberry Pi which can send data to the broker through a topic, then participants such as hospitals, public rescue and response units can receive the data through asynchronous messages from the disaster scene, the ultimate aim of this is so that emergency services are able to produce a better response. The Raspberry PI connects to the same ActiveMQ topic and has been configured with a temperature sensor to send its data to the cloud-based broker messaging server using the MQTT protocol as well. In summary both devices send data using the publish/subscribe model and this allows other services to subscribe and receive data from the broker.

C. *Message Orientated Middleware (MoM)*

Messaging is an effective method for solving system integration challenges, the subject of message-oriented middleware is not understood clearly particularly when it comes to messaging with IoT and mobile devices. Mesnil summarizes in five keywords what messaging is, "an application *produces* a *message* to a *destination* or a broker, an application *subscribes* to this same destination to *consume* the message." [9].

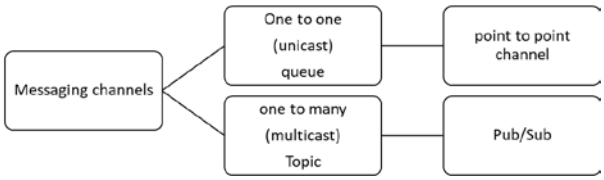

Figure 3. The Pub/Sub Pattern was used in the proposed system.

The Middleware application in our scenario which resides in a local private cloud will act a as a mediator between senders and receivers and mediate events and messages amongst publishers and subscribers. If Message Oriented Middleware (MOMs) can receive data through IoT devices, then using pub/sub patterns, data usage can be improved and interesting applications could be developed in DM Information Systems. To use MoMs in disaster management technology is useful because applications can be decoupled, increase reliability and become more scalable. The ActiveMQ MoM, an Open Source MoM built by Apache was used in the proposed disaster management system. The method of sending messages can vary in patterns, the proposed disaster management system sends messages using a multicast messaging channel, and this allows one publisher to send messages to many subscribers. Figure 3 depicts message queueing patterns which have two main logical patterns in the way it can send and receive messages namely point to point and publish/subscribe.

IV. Performance Results

This section describes the testbed architecture, implementation and experimental results.

A. *Testbed Architecture*

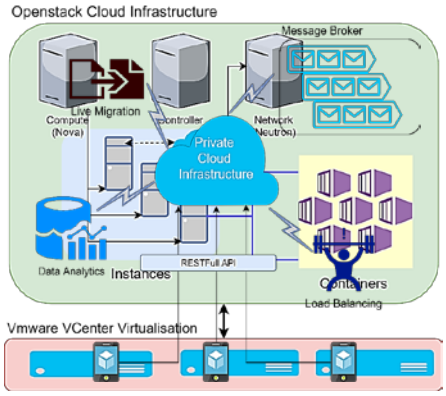

Figure 4. Resource Utilization monitoring by the load balancer

Figure 4 depicts the testbed architecture which is segregated into two layers, virtualization and cloud. The testbed acts as a private cloud infrastructure to accumulate and service requests coming from the apps. The following section describes the architecture in more detail.

i. Virtualization: physical hardware such as workstations are clubbed together to create a scalable resource pool using type 1 (bare metal) hypervisors. We've used VMware ESXI servers for the purpose, and VCenter for a centralized management of them. Each ESXI servers provides virtualization as a service. The hypervisor allows creating Virtual machines (VMs) and the Virtual Network (VNet) provides a bridge access to the physical network. VMWare License was provided by the University for this Experimentation, however an open source alternative, Citrix XEN center is also an option.

ii. Cloud: For the cloud infrastructure we have used Openstack cloud operating system. The fundamental cloud modules (Compute, Controller & Networks) are created as VMs in ESXI servers. The Openstack Compute (Nova) hosts several Linux instances acts as cloud servers, Openstack network (Neutron) connects the instances and the Openstack controller controls the modules. The instances run Docker containers to host services provided by RESCUE. Containers synchronized across Nova compute with load balancing and live migration is enabled.

B. *Implementation*

Figure 4 shows the implementation of the testbed (discussed in the previous section) to integrate with the android client.

- Message Broker: each end device running RESCUE sends data to the cloud where the MQTT message broker is placed. The broker queues the incoming requests from the apps and interleaves with other queues to provide data interoperability
- Data Analytics: Data set gathered by the high volume of requests is stored and analyzed. For the time being only simple clustering of request type versus region is implemented. However, it is a potential area of future work.
- Load Balancing: As discussed in the previous section, the network functions are virtualized into containers and placed in the cloud instances, also termed as Virtual Network Functions (VNFs). Load balancing of the instances are provided by optimal VNF placement and live migration.
- Live Migration: Live Docker migration is implemented using Docker API for python. Instances periodically updates the resource utilization of its VNFs. Also, the utilization of the instances is shared among each other periodically. Whenever an instance gets overloaded the three-step migration protocol gets initiated.

Step 1 Initialization: The load balancer-monitoring agent triggers the migration which reactively selects a VNF that is consuming maximum resource. Step 2 Finding Target: The instance with least utilization is set as a target for the departing VNF. Step 3 State Transfer: The transfer process involves two stages, first, the present state of the running container is snapshot and VNF image is saved a file. Then, the file is transferred to the target using a secure channel. Once transfer complete VNF is stopped on the old instance and fired on the new instance.

C. *Experimental Results*

This section shows some test results gathered during the experiments, stands in support of the claims made in the earlier sections. Figure 5 shows a real-time time domain plot from the load balancer. The plot is of the average utilization data from 4 different cloud instances taking requests from the client devices running RESCUE. IP addresses of the cloud instances are shown on the top left corner. The plot is dynamic, it can be noticed the red line stops plotting somewhere between the interval 10 & 20 as the instance 192.168.122.244 was stopped during the experiment. Also, the same utilization pattern is followed by all 4 instances due to same identical behaviors were executed from the apps.

Figure 6 shows the active load balancing by live migration. The blue line representing an overall utilization (Z value) of a subjected cloud instance, accumulated from other three resource utilization (CPU, Memory & Bandwidth). It can be noticed, in the time interval [40,60] the Z value increase stays above 40% (a cutoff which was set earlier). Near time 60 the red line representing network shows activity, as the migration was triggered, and container was on the fly. Near time 80, the migration process gets completed hence the VNF relinquished the resources and the Z value curve relaxes.

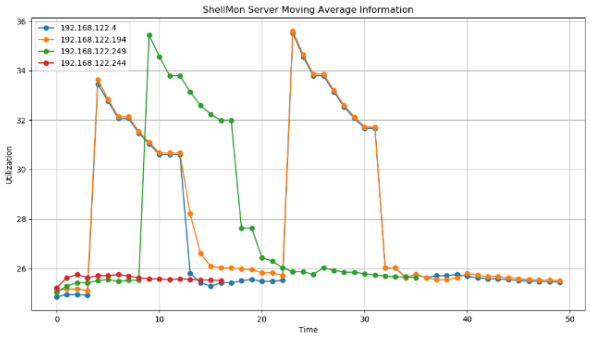

Figure 5. Resource Utilization monitoring by the load balancer

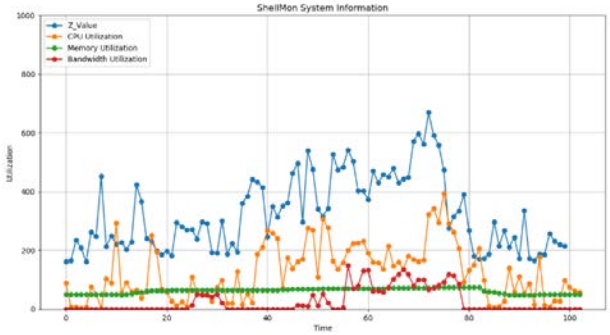

Figure 6. Active load balancing using live migration of VNF containers across the cloud instances

V. Conclusion and Future Work

Today, data integration is key for DM systems in order to improve communication between stakeholders, using middleware applications such as ActiveMQ or Mosquitto are reliable ways to achieve integration. Particularly with the plethora of protocols that are used today in IoT applications such as AMQP, MQTT, COAP and Stomp, DM systems need reliable ways of data exchange and integration. This paper has endeavored to present an effective cloud based platform architecture to make data integration, load balancing and exchange in a disaster management scenario easier. The performance evaluation is

presented to show the load balancing during live migration process. Application developers can utilize this architecture and sample implementation to provide useful disaster management applications for the IoT platform, however, IoT applications require lower latency and need to take into consideration of the user mobility and need to address interoperability between protocols such as MQTT and COAP, with edge platforms such as Fog and MEC it will be interesting to incorporate the message orientated middleware on an edge platform with the aim of understanding user mobility in DM Information Systems and achieving lower latency for DM systems.

ACKNOWLEDGEMENTS

The authors are grateful for the support received from the transnational CiProVoT (Civil Protection Volunteers Training) project. The project is co-funded by ERAMUS+ programme of the European Union.

REFERENCES

[1] Bruce Snyder, Active MQ In Action, First ed., Stamford CT: Manning, 2011.

[2] Facebook,"Facebook SafetyCheck".,https://www.facebook.com/about/crisisresponse/,Accessed May, 2018.

[3] P. Ray; M. Mukherjee; L. Shu, “Internet of Things for Disaster Management: State-of-the-Art and Prospects”, IEEE Transactions on Emerging Topics in Computing. Volume: 5, Issue: 3, July-Sept. 2017, pp 438 – 448.

[4] S. Levius; M. Safa; K. Weeks, “Use of information and communication technology to support comprehensive disaster management in the Caribbean countries”, Journal of Information Technology Case and Application Research Volume 19, 2017 - Issue 2, pp 102-112.R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[5] K. Ali ; H. Nguyen; Q.Vien; P. Shah; Z. Chu, “Disaster Management Using D2D Communication With Power Transfer and Clustering Techniques”, IEEE Access (Volume: 6), Jan 2018, pp 14643 – 14654

[6] S. Tani ; K. Motoyoshi; H. Sano; A. Okamura; H. Nishiyama; N. Kato, “Flexibility-Enhanced HTS System for Disaster Management: Responding to Communication Demand Explosion in a Disaster”, IEEE Transactions on Emerging Topics in Computing, (Early Access) March 2017.

[7] P. Li; T. Miyazaki; K. Wang; S. Guo ; W. Zhuang, “Vehicle-assist resilient information and network system for disaster management”, IEEE Transactions on Emerging Topics in Computing, Volume: 5, Issue: 3, July-Sept. 1 2017, pp 438 – 448.

[8] I. Craggs, "Paho," 2015 November 9 . [Online]. Available: http://wiki.eclipse.org/Paho. [Accessed 2018 May 21].

[9] J. Mesnil, Mobile and Web Messaging, CA: O'Reailly, 2014.