International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 12, No.1 (Jul-2022)

https://dx.doi.org/10.12785/ijcds/120105

Fault Tolerance Mechanism for Software Application Through Fog Computing as Middleware

Mohd Hariz Naim[1], Jasni Mohamad Zain[2] and Kamarularifin Abd Jalil[3]

[1]*Centre for Advanced Computing Technology C-ACT, Universiti Teknikal Malaysia Melaka, Melaka, Malaysia*
[2]*Institute for Big Data Analytics and Artificial Intelligence, Universiti Teknologi MARA, Shah Alam, Malaysia*
[3]*Department of Computer Technology and Network, Universiti Teknologi MARA, Shah Alam, Malaysia*

Received 16 Apr. 2021, Revised 10 Mar. 2022, Accepted 15 Jun. 2022, Published 1 Jul. 2022

Abstract: Fault tolerance is a paradigm for providing high availability in a computerized system where application services are replicated to multiple nodes. The paradigm is widely used in a cloud computing environment where users may benefit from automatic backup when an application such as a clinic support system is deployed in the cloud. However, applications residing in premises such as small clinics are still prone to an outage and would require certain time with human involvement when performing recovery. Thus, this research aims to provide a prototype of a backup mechanism for a health system residing in-premise of a clinic. The study will also act as a feasibility investigation of fault tolerance mechanism in edge of network where we proposed the use of fog computing model that acts as middleware for detecting and failover solution when an outage has temporarily occurred. The middleware will perform failure detection through heartbeat and replicate the services at the same time. When an outage is detected, the middleware will take over as a secondary service provider to ensure applications may be used seamlessly. There are four test simulations for testing the proposed mechanism where three has shown successful recovery actions and only one fail test case due to limitation at the client side.

Keywords: Fault Tolerance, Fog Computing, High Availability

1. Introduction

The computer system needs to ensure three security elements consisting of Confidentiality, Integrity and Availability (CIA) [1] when providing services to end users. The term high availability is part of the Availability element where the system needs to provide 99.999% uptime [2], [3] known as the five nine. The availability of the system to end users may differ due to many factors. As for the reason, it is a challenge for software and system providers to meet the Service Level Agreement (SLA) in term of availability. In critical sectors such as health care, the availability of the computerized system is crucial to provide seamless information so that medical personnel could respond to patient accurately when giving treatment.

Many software providers preferably deploy the applications on the cloud as a popular solution as cloud providers could take care of the infrastructure and availability. The cloud architecture however would require internet connectivity for end users to access the applications. This, however, has created specific issues such as network attacks [4], performance delay [5] and expensive subscription service when using the cloud provision [5], [6], [7]. As for the reasons, some solution providers are looking an opt to deploy their application closer to users such as in-premise server deployment. The term closer to users are associated with fog computing [8], [9] where the machine or devices are located at the edge of network [10] isolated from the cloud computing architecture. Together with the implementation of cloud computing, it will act as a supplementary to the computer system in an event of cloud failure, the fog computing may continuously provide alternative services to the end users.

In ensuring seamless application services running, cloud computing technology has implemented several measures in providing high availability. In general, there are three mechanisms as mentioned by [1]: Fault Tolerance, Protective Redundancy and Overload Protection, which will be explained in detailed in the next section. Even though cloud providers implement all the mechanisms, the outage would still occur unexpectedly, causing interruptions to hundreds of back-end service operations. In addition, for applications running internally within an organization, they would not be able to take the benefit of the cloud and need to set up their own backup and secondary server.

In order to maximize the uptime and utilizing existing resources, middleware applications are used to assist and reuse the current software architecture deployment [11]. The

E-mail address: mohdhariz@utem.edu.my, jasni@tmsk.uitm.edu.my, kamarul@tmsk.uitm.edu.my

middleware is beneficial for cloud architecture where it may interact and integrate with various platform to apply the fault tolerant mechanisms [12]. Through middleware, the replication strategy can be implemented into different cloud providers where the transition between different cloud is faster with the use of middleware.

In the next section, the paper will discuss the literature review of related research projects including fault tolerant and recovery techniques. In Section 3, the research methodology will be explained where the proposed method is composed into failure detection, service replication and the switching over to delegate failed service. The result and discussion based on the simulation are explained on Section 4, followed by conclusion and future work in the last section.

2. Related Works

Cloud computing is widely used nowadays as the popularity and advancement of internet connectivity and the wide use of computerized devices. Mobile phones, small peripherals including Internet of Things rely heavily on the internet to connect to the cloud. As for the reason, software providers prefer to deploy software services, especially back-end services such as web services on the cloud servers. The term cloud computing is referred to as a pool of computerized system resources [13] that are used by end-users regardless of different operating system and hardware devices. With cross-platform and less dependency on hardware-centric, cloud computing able to integrate and exchange communication between multiple devices.

The cloud services are also considered as distributed computing [14] which consist of categories, namely as Infrastructure as a Service (IaaS), Platform as a Service (Paas) and Software as a Service (Saas) [1], [14], [15]. These services have different characteristics in providing daily business activities and operation. As mention by [1], the SaaS cloud provides a complete and ready to use application for end-users where users may buy the services according to subscription such as Dropbox, Netflix and Office 365. On the other hand, the PaaS focuses on deploying end-user's solution through a semi-complete and less hectic server setup. In contrast, the IaaS provides virtual or hardware resources, including network connectivity collaboration between different geographical location. With different scope of services, users of all levels may flexibly customize their cloud subscriptions based on their requirements and budget.

Despite the fact that the benefit of cloud computing has to offer, the cloud still prone to some issues that users may need to consider. Research done by [5], mentions the latency issues that suffer from geographically distributed cloud servers as the users are physically located far from the servers. This means that the further the users are located, the longer the time would be needed for the service to acknowledge and provide the service. Consecutively, this explains the need for the users to specify which location they would prefer to access the service especially during downloading files from server. Although the cloud architecture has implemented certain high availability mechanism, there is unexpected downtime that experience by certain cloud providers [12]. It was reported that even the renowned cloud provider such as Amazon EC2 and Google Cloud Platform also suffers from sudden outage that would impact high cost to the end users [12]. Additionally, the situations are worsened if the applications hosted on the cloud require real-time respond which need low latency and high availability of service [16] such as providing service related to financial and health care. Any latency or outage of service would have huge impact and loss to the involved users. In ensuring high availability and seamless operation of service at the application layer, some software providers make use of agent and middleware [17], [18] for integrating the service among different cloud providers. However, some cloud providers do not provide platform and features for implementing the middleware making the high availability mechanism limited to the cloud layer. As the cloud computing progress to suit the users demand, the fog computing also emerged as a supplementary to the cloud infrastructure. The term fog is viewed as mass of devices ranging from mobile phones, small peripherals and sensors [19] connected through certain protocols. As reviewed by [20], [21] the fog computing is another paradigm of computing where the logical process such as data collection, calculation, analysis and decision making are executed by devices residing at the edge of network instead of the cloud. All the devices are capable to perform tasks starting from data collection, storing into database and other tasks with minimal dependency from the central or cloud servers. To help visualize the role of fog computing, the Figure 1 show how the fog devices residing in edge of network correlating between application and cloud servers. This proves to be helpful and beneficial in term of latency, better real time response and independent from the internet [20].

In fact, the usage of fog devices such as sensors devices for healthcare usage would bring more flexibility and advantages such as portability and real time processing [21], [22] as they are closer and can be attached directly to patients. This is crucial for patients that require constant monitoring and elderly citizens that lives away from relatives and medical personnel. In the event of emergency occurrences, the IoT fog devices may trigger notifications and alerting emergency responses to the relatives. Additionally, fog devices may also help to observe and perform data collections among patients with better latency.

In term of connectivity, the fog devices are able to perform without internet connection and capable to work solely [21], [23] without depending on cloud centralized servers. This proves to be beneficial for remote clinics or health centres that are isolated and require offline processing rather than depending on centralized cloud servers that need stable internet connectivity. The fog devices however, are still suffers from some issues and challenges especially related to the limited power supplies and resources capabilities

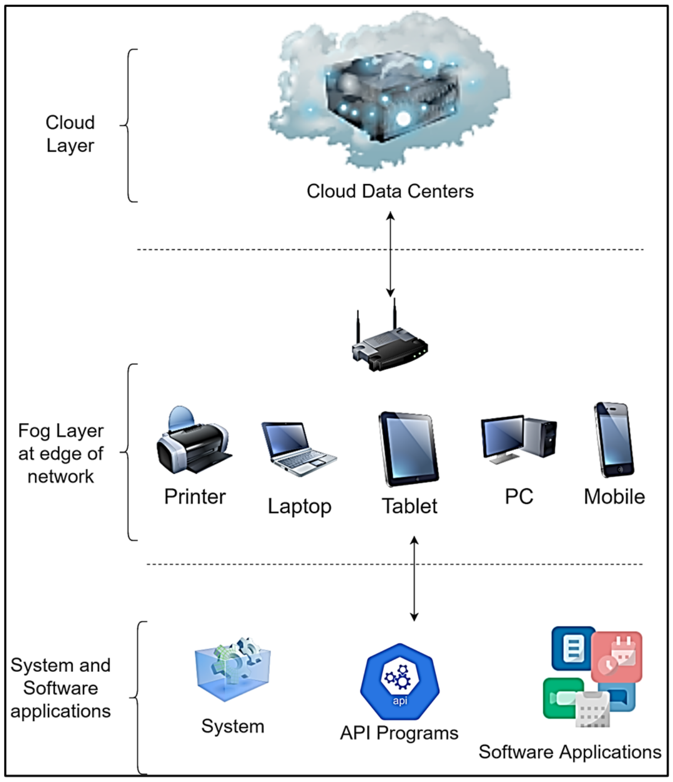

Figure 1. Role of fog layer

[19], [21], [24] which is not at the same par with cloud resources. Moreover, deploying healthcare application on isolated clinics and health centres on single host node as a server is prone to downtime and service interruptions as fault tolerance mechanisms have to be manually imposed at the surrounding environment. This eventually would make deployment expensive and hard to be maintained in the future. In addition, there are many remote clinics are in small size organization and does not have dedicated staff for managing the infrastructure.

In order to ensure the high availability of services, many IT providers would implement certain mechanisms. According to [1], the availability mechanisms namely fault tolerance, protective redundancy and overload protection are widely implemented in cloud environment. These mechanisms are very suitable for clouds as the resources and hardware capabilities are significantly high end [7] where the storage is massive while the processing power are scalable. Cloud providers such as Google, Microsoft Azure and Amazon AWS would implement all the availability mechanisms within their data centres that are distributed around the world which mean that they have abundant of resources to spare promising close-to-zero downtime. Despite the commitment to fulfilling the Service Level Agreement (SLA), the outage of service could still happen as reported by [25] where unexpected service downtime by certain cloud providers has exceeded beyond the accepted downtime. The fault tolerance is defined as the capability of a system to continuously operate even when a fault has occurred where it includes avoiding a failure or failure detection before able to recover [1]. Generally, the fault tolerance mechanisms are categorized into post active [14] and proactive [26], [27] technique. The proactive would mean the mechanism will take precaution measurement before the failure occurs while the post active or reactive method would need to take recovery action after the failure has occurred.

In order to execute the proactive measurement, the information regarding the resources is needed such as resource consumption and utilization from previous history where the record is modelled [14], [28] to predict when will the next downtime is expected to occur. Besides prediction, the proactive also involve taking preventative actions such as rebooting and restarting operating system and application services [29] to prevent system aging which can lead to application failure when the resources are not freed up. These proactive measurements however still suffer from downtime as rebooting the operating system and services would take some time [30], [29] to get back up and running especially for big data centres where rebooting the entire systems at the same time is not an effective method. Additionally, modelling and prediction also prone to false positive detection and inaccurate estimation [14] which require deep and more knowledge about the node task activities and resource utilization.

On the other hand, the post active measurement would need to continuously monitor the remote node for any failure by implementing certain such as heartbeat and watch dog timer [28]. Through this technique, a signal is continuously sent to the monitored node which then listened for response from the monitored node. Once the monitored node is not responding to the signal, a failure is indicated and may trigger to perform recovery actions. Since it requires seamless monitoring, this is very costly in term of resources as additional computational overhead is incurred [14]. The recovery actions would need to redirect or provide secondary service provider which is also known as switchover or protective redundancy where another node will act as a backup [7], [31] to take over from the failed node. For application-based fault tolerance, in case of failure occurrence the software provider could employ Roll-Back and Roll-Forward mechanism [1] as part of post active measurement. These would involve in saving the state of application to a save point [27] especially related to data that is stored in database. Whenever the data is corrupted or missing, the application can change back to the saved point to recover the application from failure.

Under the protective redundancy mechanism, there are several models that can be implemented namely 2N, N+M, N-Way and N-Way Active [1]. Here the models would refer to the number of replications that will be created to serve as backup node where the backup would be scattered around either within same or different geographical place. The N indicates the redundant element or replicated node that acts as standby while the M is the active node providing the primary services. According to [1], the 2N model will be having two nodes or elements where one will provide main services while the other element will act as standby node, this way when the main node has failed, the standby node will take over to provide the required services. On the other hand, the N+M model will create standby node depending on the number of main nodes and the standby node must be more than the main node as the main idea is to provide multiple geographically different standby node [14]. In the event of natural disaster such as earthquake and flood, the services can be redirected to another location. Contrarily, the N-Way model would also require the redundant node to provide some services instead of just standing by awaiting to take over the main node. This eventually, will optimize the resources and able to balance the task load between main node and replicated nodes. Similarly, the N-Way Active model also make all the available node as the main node which protect each other and provide all the required services. In other words, there is no standby node as all the nodes are actively providing services to the end client.

From the high availability mechanisms, some researchers have combined the methods such as [2], [3], [5] where they utilized the redundancy model together with switchover technique. They applied several devices or nodes as a backup that replicate with the main node and standby to switchover when failure has occurred. These research however only targeted on the cloud that has the features for fault tolerant mechanism. In term of software failure, there are researchers who have proposed the implementation of rejuvenation technique [29], [30]. The technique is part of proactive measurement where it would estimate the next failure occurrence and then prevent the failure from happening or reduce the time of recovery through several countermeasures such as restarting services and mitigating to other standby node. The Table I summarizes the related works in the field of high availability technique.

In a research done by [33] where the researcher makes use of fog devices to migrate task from cloud server to fog devices. The technique utilized the Docker technology together with the integration with Amazon Web Service (AWS) to implement the switchover mechanism. Based on the Amazon cloud service, application would be transferred to the fog devices on the edge on network when failure has occurred. The technique however, still suffer from latency and migrating the whole tasks to fog device would eventually slow down task processing since the fog device resources are not at the same par with the Amazon AWS cloud. Furthermore, it is not applicable for private cloud that has its own infrastructure that differs from the Amazon AWS. Moreover, the technique requires the use of containerization technology in order to migrate and delegate the task to fog device which is not applicable for a common web-based application host on private cloud.

Another research that also make benefit of fog computing where the research implement switchover technique to nearest fog devices at the edge of network [23]. The researcher main objective is to predict and determine which nodes that suitable to take over tasks in order to reduce network traffic and bandwidth. Thus, the closest fog devices between users and main source are the main concern of the research. In the other words, the research does not focus on improving high availability mechanism between fog device and cloud.

Based on the discussed mechanism, the suitable technique to be implemented in this research is the post active with the combination of fault tolerance and failure detection by using heartbeat signal listener. This is because, for the clinic support system, the end users who are residing at the edge of the network are prone to connectivity failure and unable to access the required services due to application failure when performing daily tasks. For this reason, the research aim is to provide high availability services by utilizing fog devices residing on the edge of network. Further discussion on the implementation will be explained in the next section.

3. Methodology

In this research, there are three proposed modules: failure detection, service replication and task offloading to secondary backup node, as depicted in Figure 2. The first action before applying failover solution is to identify failure occurrences with minimal false positive correctly. At the same time, another dedicated machine or node is configured to become the redundant elements which is called the cloned fog device. The last module is the switchover or offloading the services to the cloned fog devices which resides closer to end users located in a clinic. Here, the proposed components are installed on a dedicated fog device called Raspberry Pi 4 residing at the same network within a health centre. The fog device has a resource specification of 2 gigabytes of memory and extended storage of 128 gigabytes of hard disk storage which is enough to store cloned data for small application.

A. Failure Detection Module

In this module, the failure detection is implemented based on the heartbeat technique [27]. The module will keep on listening to the main cloud server by continuously sending periodic signals and awaiting response. The module is written in Python script which points to the main cloud server address. Here the heartbeat listener will keep on running and trigger the signal for every 5 seconds executed by cron script that is available by the Raspbian operating system of the Raspberry Pi.

B. Redundancy and Replication of Services

The second module is the replication of services that mirrored to the central cloud server. However, due to the constraint of storage and memory resources by the Raspberry Pi device, only essential services are replicated in the

TABLE I. Summary of Related Works

Research Article	Used Technique	Targeted Platform	Failure Level
[5]	Redundancy Model + Failover	Cloud	Cloud
[23]	Switchover	Fog/Edge	None
[29]	Failover + Rejuvenation	Single Node	Application
[30]	Failover + Rejuvenation	Cloud	Application
[12]	Redundancy Model	Cloud	Cloud
[32]	Redundancy Model	Cloud	Microservice
[15]	Redundancy + Roll Back	Virtual Machine	Virtual Machine
[33]	Switchover + Redundancy	Fog/Edge	Application
[2]	Switchover + Redundancy	Cloud	Cloud
[3]	Switchover + Redundancy	Cloud	Virtual Machine

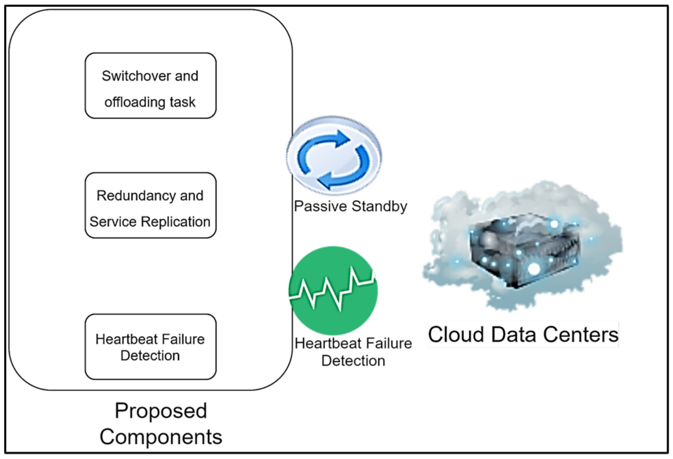
Switchover and offloading task
Redundancy and Service Replication
Heartbeat Failure Detection
Passive Standby
Heartbeat Failure Detection
Cloud Data Centers
Proposed Components

Figure 2. The proposed components

```
Algorithm 1: Example code
  Input: CentralServerAddress,FogAddress
  Output: CentralServerRespond
1 CentralServerRespond ← $fnSendHeartbeatMsg$
2 if $CentralServerRespond == false$ then
3     FogAddress ← $CentralServerAddress$
4 else if $CentralServerRespond == true$ then
5     fnUpdateServices
6     fnFetchData
```

device. Here, the replicated services are the Apache Tomcat service for running back-end application, the MySQL service for providing database and the Clinic Support System, a Java Swing application that runs on the Apache Tomcat. The installed service acts as cold standby which means end users do not access the fog device as primary source instead it would only be accessed when an outage has occurred at the central server. In order to ensure that the data is up to date with the centralized server, the MySQL database is synchronized by enabling data mirroring feature. The replication process follows the generic algorithm as shown in Algorithm 1 where it is necessary to have both central server and fog device address in the middleware program written in Python. Based on the respond from the central server as output, the program will determine the next action to be taken. Generally, there are two output condition, first if the respond is false, then the fog address will be changed to become similar with the central server address. Secondly, if the respond is true meaning that the central server is accessible, then the middleware program will try to obtain the latest data and backend services from the central server by invoking the method **fnUpdateServices** and **fnFetchData**

C. *Switchover and Offloading Task*

Since the Raspberry Pi is a cold standby node, it also known as a slave node that waits to take over from the central server that is the master node. Here, we proposed that both the master and the slave have their own dedicated static IP address rather than dynamic address so that it would be easier to switch the address in the event of failure occurrences. When the heartbeat has triggered the failure connectivity, the IP address of the slave will switch to be the same as the master's IP address. This way, end-users would not notice service interruption and access the clinic support system seamlessly as the desktop application has resolved the inaccessible address. The same Python middleware program is used where it executes the process for switchover and offloading task following the Algorithm 2. The program requires the heartbeat respond as an input and later change the services required to be run on the fog device as output. Conditionally, if the heartbeat respond is false, then the program will automatically start all the required services on the fog device. At the same time, the heartbeat message is continuously sent by invoking fnSendHeartbeatMsg to lookup for central server in case the server has been recovered. Later, when the respond is equal to true, indicating that the central server has been recovered, the services on the fog device is turn down to conserve resources so that other process such as data and service replication can utilize the resources needed.

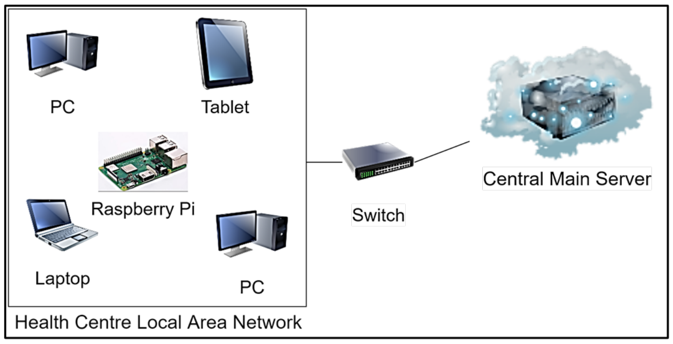

Figure 3. Device connected within same local area netwprk(LAN)

Algorithm 2: Example code

```
Input: HeartbeatRespond
Output: FogDeviceService
1 HeartbeatRespond ←fnSendHeartbeatMsg
2 if CentralServerRespond == false then
3   | FogDeviceService ← UP
4 else if CentralServerRespond == true then
5   | FogDeviceService ← DOWN
```

4. Simulation and Result

From the proposed methodology, the environment takes place in a private health centre where it simulates the edge of the network that has separated network architecture compared to the central cloud server. The fog device is placed within the health centre that belongs to the same network segment with other devices attached to a single switch as depicted in Figure 3.

From the environmental setup, few simulations are executed for testing out the feasibility of the proposed solution. The first simulation case is when the network connectivity is disconnected at the switch level. The following simulation would be disabling the network adapter card at the central server to mimic that the server is experiencing failure. Then, the application services are stopped but the network connectivity is enabled. The application services that are stopped are Apache, MySQL and a Java backend service. These are the core services needed to enable the clinic support system as whole complete package for a web application. Also, we will try to disable the network adapters at one of the end-users' computers. We will also monitor the data integrity to ensure that information is up to date with the master node.

Here, other devices such as desktops and tablets would connect to the same switch. When network outage has occurred, other devices are isolated but still connected as a single local area network (LAN) which means all the devices can communicate with each other. The simulation of failure is done by disconnecting the switch connection towards the central main server which is similar to the situation where the network has failed or that internet connectivity is unavailable.

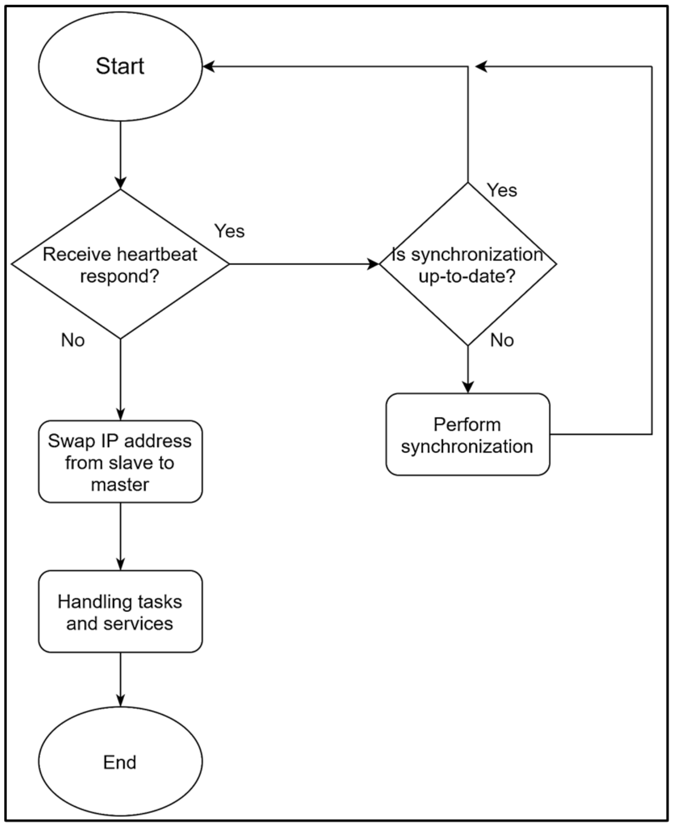
Start
Receive heartbeat respond?
Yes
No
Is synchronization up-to-date?
Yes
No
Perform synchronization
Swap IP address from slave to master
Handling tasks and services
End

Figure 4. Proposed Process Flow

Based on the process flow shown in Figure 6, the heartbeat signals are sent to the central server every 5 seconds and wait for a respond. At the same time if the signal is acknowledged, the fog device will perform data synchronization to update its local database. The data in central server is massive with more than 10 gigabytes of data, thus in order to avoid unnecessary synchronization and avoid network congestion, the synchronization only executed when the binary log is different between the master and slave. A difference in the binary log would indicate that the data has changed, and only certain data is passed to the slave rather than passing the whole database.

On the other hand, if the heartbeat is not getting any respond from the central server, the fog device that act as slave will change its IP address to become similar with the central server. In this simulation, initially the central server has address of 10.102.100.100 while the fog device has address of 10.102.100.102 as depicts in Figure 5. When outage occurred, the fog device address change to become 10.102.100.100 as visualize in Figure 6. By swapping the address, the clinic support system desktop application will automatically refer the tasks and services to the fog devices without having to manually redirect or change service provider for every device in the same network segment.

During the outage, the fog device will keep on sending

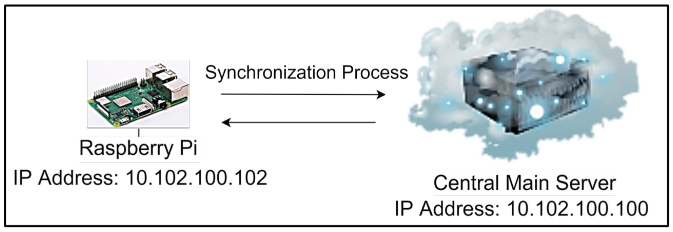

Figure 5. IP address configuration during uptime

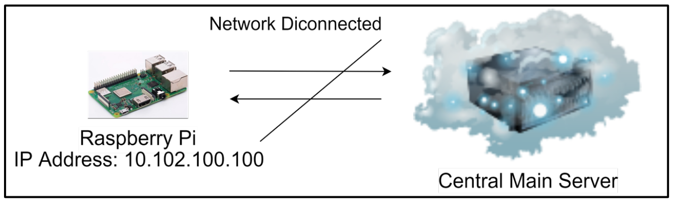

Figure 6. IP address configuration during network failure

heartbeat signal to the central main server and when the network has re-established, the fog device will quickly swap back to its original IP address to avoid address conflict. All the devices will now point to the central main server and the fog device will become the slave node performing cold standby which continue to monitor central main server for any outage by sending heartbeat signal.

A. *Result and Analysis*

In this subsection, the simulation result will be presented based on the simulation test cases. The result is based on time to recover from failure upon disconnection of the switch from the internet source, which is the primary connection to the central server. Here we record the downtime and the time taken to recover until three trials to observe the recovery consistency and the results are shown in Table II. The recovery time taken is recorded by the fog device which is based on the connection to the neighbouring devices when failure has occurred. Figure 7 shows the graphical presentation of the simulation result, indicating that each test has almost similar recovery times with the only deviation of less than one second. Another measurement metric that is taken into consideration is the time taken by application users to gain access after network failure occurrences, where the time is recorded by the Clinic Support System when it has detected inaccessible back-end service to the main cloud server. The recovery time taken is calculated based on the given formula [34], [27]:

$$RecoveryTimeTaken = TimeRecovered - TimeDetected \tag{1}$$

From the several simulation tests, the result is shown in Table III. From four of test case, only one failed which is the failure at the end user level where the network card has been disabled. However, the failure is not occurring on other devices thus, the application may be still accessible. Currently, failure by the end-user itself is beyond the research scope, but it is vital to take note that future research may open to resolve.

TABLE II. Simulation result based on failure condition

No of trial	Recovery Time Taken(seconds)	Time for application to gain Access(seconds)
First	5.1	4.3
Second	5.5	6.1
Third	5.3	4.8

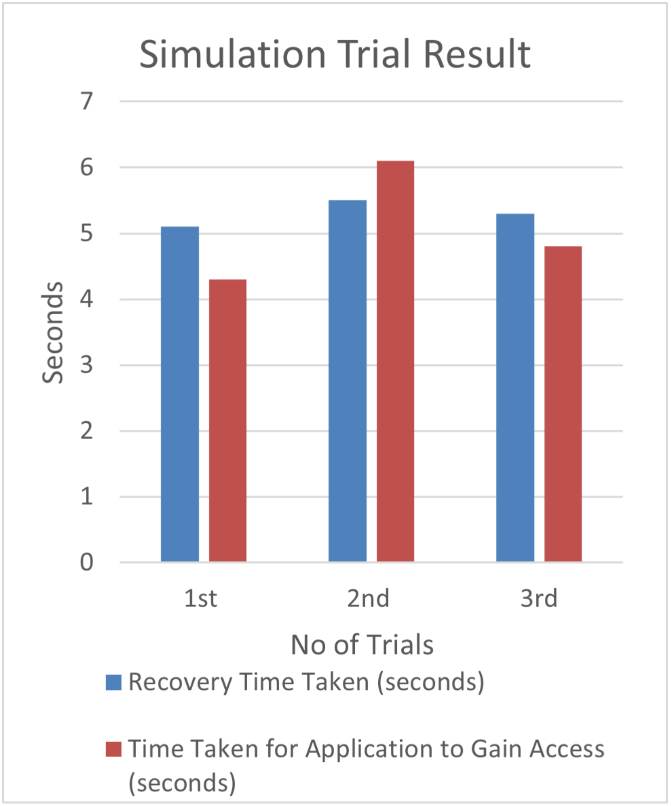

Figure 7. Time taken to recover based on three trials

It is also observed that the time taken for the application to recover is slightly different from the time taken to recover by the fog device. This might be due to the back-end application programming interface (API) call by the Clinic Support System restricted and random by the Java library. Furthermore, optimizing the application for faster service recovery is beyond our research scope.

5. Conclusion and Future Work

In this research, we have proposed the utilization of Raspberry Pi as fog devices to provide fault tolerance mechanism in a health centre environment where the device acts as cold standby as part of redundancy element. From the simulation, the proposed method has proven that the fog device is capable of providing backup service whenever

TABLE III. Test Cases and Result

Failure Test Case	Failure Simulation	Result
Switch	Disconnect cable at switch	Pass
Server	Disabled network card	Pass
Application	Stop app services	Pass
End Users	Disable network card	Failed

a network segment is experiencing connectivity downtime. Additionally, the proposed method would be beneficial to be implemented in the edge of network where internet connectivity is an issue.

Despite the advantages that, the proposed method has some issues that need to be resolved. It is known that fog devices are limited in term of resources which is not at the same par with cloud server resources [33], [35], [36]. As for the reason, the proposed method would need certain services to be installed and running on the fog device. The limitation of storage also hinders the fog device from storing gigantic database and would need data optimization or meticulous data selection before storing into the fog device. In term of fault tolerance, the proposed method is limited to provide recovery for desktop application and does not cater the domain name as well as dynamic address for more flexible IP address assignation. For future work, we aim to provide more supple and wider application that can be protected from failure.

Acknowledgment

The author would like to express gratitude Center of Advanced Computing Technology (C-ACT) Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka and Center of Computer Technology and Network (CTN) Faculty of Computer Science and Mathematics Universiti Teknologi Mara, Malaysia for offering facilities in term of research accommodations.

References

[1] M. Nabi, M. Toeroe, F. Khendek, M. Nabi, M. Toeroe, and F. Khendek, "Availability in the Cloud : State of the Art," *Journal of Network and Computer Applications*, no. 60, pp. 54–67, 2016.

[2] R. Moreno-Vozmediano, R. S. Montero, E. Huedo, and I. M. Llorente, "Orchestrating the deployment of high availability services on multi-zone and multi-cloud scenarios," *Journal of Grid Computing*, vol. 16, no. 1, pp. 39–53, 2017.

[3] M. R. Mesbahi, A. M. Rahmani, and M. Hosseinzadeh, "Highly reliable architecture using the 80/20 rule in cloud computing datacenters," *Future Generation Computer Systems*, vol. 77, pp. 77–86, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.future.2017.06.011

[4] N. Baharudin, F. H. M. Ali, M. Y. Darus, and N. Awang, "Wireless intruder detection system (WIDS) in detecting de-authentication and disassociation attacks in IEEE 802.11," *2015 5th International Conference on IT Convergence and Security, ICITCS 2015 - Proceedings*, pp. 1–5, 2015.

[5] Y. Aldwyan and R. O. Sinnott, "Latency-aware failover strategies for containerized web applications in distributed clouds Cloud Failover Techniques :," *Future Generation Computer Systems*, vol. 101, pp. 1081–1095, 2019. [Online]. Available: https://doi.org/10.1016/j.future.2019.07.032

[6] P. Alves Lima, A. Sá Barreto Neto, and P. Romero Martins MacIel, "Data Centers Service Restoration Based on Distributed Agents Decision," *Proceedings - 2018 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2018*, pp. 1611–1616, 2019.

[7] G. Estácio, G. Patricia, T. Endo, M. Rodrigues, D. H. Sadok, J. Kelner, and C. Curescu, "Resource allocation based on redundancy models for high availability cloud," *Computing*, vol. 102, no. 1, pp. 43–63, 2020. [Online]. Available: https://doi.org/10.1007/s00607-019-00728-1

[8] Y. Jin and H. J. Lee, "On-demand computation offloading architecture in fog networks," *Electronics (Switzerland)*, vol. 8, no. 10, 2019.

[9] A. A. Mutlag, M. K. Abd Ghani, N. Arunkumar, M. A. Mohammed, and O. Mohd, "Enabling technologies for fog computing in healthcare IoT systems," *Future Generation Computer Systems*, vol. 90, pp. 62–78, 2019. [Online]. Available: https://doi.org/10.1016/j.future.2018.07.049

[10] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," *Journal of Network and Computer Applications*, vol. 98, no. September, pp. 27–42, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.jnca.2017.09.002

[11] S. N. Srirama, F. M. S. Dick, and M. Adhikari, "Akka framework based on the Actor model for executing distributed Fog Computing applications," *Future Generation Computer Systems*, vol. 117, pp. 439–452, 2021. [Online]. Available: https://doi.org/10.1016/j.future.2020.12.011

[12] L. Acquaviva, I. Informatica, S. Monti, and I. Informatica, "NoMISHAP : A Novel Middleware Support for High Availability in Multicloud PaaS," *IEEE Cloud Computing*, vol. 4, no. 4, pp. 60–72, 2017.

[13] K. Syed and K. Vijaya, "Cloud Computing: Review on Recent Research Progress and Issues," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 8, no. 3, pp. 959–962, 2019. [Online]. Available: https://doi.org/10.30534/ijatcse/2019/96832019

[14] A. S. Ebenezer, E. B. Rajsingh, and B. Kaliaperumal, "A novel proactive Health Aware Fault Tolerant (HAFT) scheduler for computational grid based on resource failure data analytics A novel proactive Health Aware Fault Tolerant (HAFT) scheduler for computational grid based on resource failure data analyti," *International Journal of Computers and Applications*, vol. 7074, pp. 1–11, 2018. [Online]. Available: http://doi.org/10.1080/1206212X.2018.1440339

[15] M. Torquato, L. Torquato, P. Maciel, and M. Vieira, "IaaS cloud availability planning using models and genetic algorithms," *2019 9th Latin-American Symposium on Dependable Computing, LADC 2019 - Proceedings*, 2019.

[16] A. Sharif, M. Nickray, and A. Shahidinejad, "Fault-tolerant with load balancing scheduling in a fog-based IoT application," *IET Communications*, vol. 14, no. 16, pp. 2646–2657, 2020.

[17] P. Alves, L. Antônio, S. Barreto, and N. Paulo, "Data centers ' services restoration based on the decision-making of distributed agents," *Telecommunication Systems*, 2020. [Online]. Available: https://doi.org/10.1007/s11235-020-00660-2

[18] M. H. Naim, M. K. A. Ghani, A. S. H. Basari, B. Aboobaider, L. Salahuddin, and W. N. A. Rashid, "Synchronization technique via raspbery Pi as middleware for hospital information system," *Advances in Intelligent Systems and Computing*, vol. 734, pp. 262–271, 2018.

[19] S. Svorobej, P. T. Endo, M. Bendechache, C. Filelis-Papadopoulos, K. M. Giannoutakis, G. A. Gravvanis, D. Tzovaras, J. Byrne, and T. Lynn, "Simulating fog and edge computing scenarios: An overview and research challenges," *Future Internet*, vol. 11, no. 3, pp. 1–15, 2019.

[20] I. Sittón-Candanedo, R. S. Alonso, J. M. Corchado, S. Rodríguez-González, and R. Casado-Vara, "A review of edge computing reference architectures and a new global edge proposal," *Future Generation Computer Systems*, vol. 99, no. 2019, pp. 278–294, 2019. [Online]. Available: https://doi.org/10.1016/j.future.2019.04.016

[21] K. Bilal, O. Khalid, A. Erbad, and S. U. Khan, "Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers," *Computer Networks*, vol. 130, no. 2018, pp. 94–120, 2018. [Online]. Available: https://doi.org/10.1016/j.comnet.2017.10.002

[22] F. A. Kraemer, A. E. Braten, N. Tamkittikhun, and D. Palma, "Fog Computing in Healthcare-A Review and Discussion," *IEEE Access*, vol. 5, no. 2169, pp. 9206–9222, 2017.

[23] A. Alelaiwi, "An efficient method of computation offloading in an edge cloud platform," *Journal of Parallel and Distributed Computing*, vol. 127, pp. 58–64, 2019. [Online]. Available: https://doi.org/10.1016/j.jpdc.2019.01.003

[24] P. Shroff and A. Bandyopadhyay, "A novel matching framework for one-sided markets in fog computing," *International Journal of Computing and Digital Systems*, vol. 10, pp. 1–10, 2020.

[25] M. R. Mesbahi, A. M. Rahmani, and M. Hosseinzadeh, "Reliability and high availability in cloud computing environments: a reference roadmap," *Human-centric Computing and Information Sciences*, vol. 8, no. 1, 2018. [Online]. Available: https://doi.org/10.1186/s13673-018-0143-8

[26] S. M. Attallah, M. B. Fayek, S. M. Nassar, and E. E. Hemayed, "Proactive load balancing fault tolerance algorithm in cloud computing," *Concurrency Computation*, vol. 33, no. 10, pp. 1–11, 2021.

[27] S. Prakash, V. Vyas, and A. Bhola, "Proactive Fault Tolerance using Heartbeat Strategy for Fault Detection," no. 1, pp. 4927–4932, 2019.

[28] S. J. Ellis, T. H. Klinge, C. College, J. I. Lathrop, J. H. Lutz, R. R. Lutz, and A. S. Miner, "Runtime Fault Detection in Programmed Molecular Systems," *ACM Transactions on Software Engineering and Methodology*, vol. 28, no. 2, pp. 6–20, 2019.

[29] T. I. Azman, N. C. Pa, R. N. H. Nor, and Y. Y. Jusoh, "Risk mitigation for anti software ageing – A systematic literature review," *Journal of Theoretical and Applied Information Technology*, vol. 97, no. 14, pp. 3911–3936, 2019.

[30] J. Bai, X. Chang, F. Machida, K. S. Trivedi, and Z. Han, "Analyzing Software Rejuvenation Techniques in a Virtualized System: Service Provider and User Views," *IEEE Access*, vol. 8, pp. 6448–6459, 2020.

[31] Y.-L. Lee, S. N. Arizky, Y.-R. Chen, D. Liang, and W.-J. Wang, "High-availability computing platform with sensor fault resilience," *Sensors*, vol. 21, no. 2, pp. 1–16, 2021.

[32] L. Abdollahi Vayghan, M. A. Saied, M. Toeroe, and F. Khendek, "Microservice Based Architecture: Towards High-Availability for Stateful Applications with Kubernetes," *Proceedings - 19th IEEE International Conference on Software Quality, Reliability and Security, QRS 2019*, pp. 176–185, 2019.

[33] D. R. Sangolli, N. M. Ravindrarao, P. C. Patil, T. Palissery, and K. Liu, "Enabling high availability edge computing platform," *Proceedings - 2019 7th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, MobileCloud 2019*, pp. 85–92, 2019.

[34] D. Mani, "Availability Modelling of Fault Tolerant Cloud Computing System," *International Journal of Intelligent Engineering Systems*, vol. 10, no. 1, pp. 154–165, 2017.

[35] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *Journal of Systems Architecture*, vol. 98, no. February, pp. 289–330, 2019. [Online]. Available: https://doi.org/10.1016/j.sysarc.2019.02.009

[36] M. Etemadi, M. Ghobaei-Arani, and A. Shahidinejad, "Resource provisioning for IoT services in the fog computing environment: An autonomic approach," *Computer Communications*, vol. 161, no. July, pp. 109–131, 2020. [Online]. Available: https://doi.org/10.1016/j.comcom.2020.07.028

Mohd Hariz Naim is lecturer at Universiti Teknikal Malaysia Melaka(UTeM) in the department of software engineering faculty of information and communication technology (FICT). He is currently pursuing his PhD in field of Software Development and IT Operation (DevOps) related to high availability and fault tolerant. He is also interested in the field of mobile application development.

Jasni Mohamad Zain is a professor and also the director of the Institute of Big Data Analytics and Artificial Intelligence (IBDAAI), Universiti Teknologi MARA (UiTM) Shah Alam Malaysia. She obtained her PhD from Brunel University West London, UK. Her research interests are related to digital watermarking, image processing, data security and data mining.

Kamarularifin Abd Jalil is an associate professor at the department of computer technology and networking, Faculty of Computer and Mathematical Science Universiti Teknologi MARA, Shah Alam, Malaysia. He obtained his PhD at University of Strathclyde, UK. His research interests are mobile communications, mobile networks protocols, wireless networks, digital forensics and information security.