CHAPTER 1

INTRODUCTION

Software-Defined Network (SDN) is a new networking paradigm in which hardware is separated and forwarded by controllers. Traditional networks consist of hosts, switches, routers, and others. Network functionality and management in traditional networks are implemented in every network device. It is intended to facilitate network management enabling future innovation. Multi-controller architecture in network design provides more resistance to failure. The integration of network forwarding behavior and controls on individual devices makes networking time-consuming and error-prone. The master controller functions as a load balancer for all local controllers so that each local controller must work together with the master controller so that the load distribution is more even. Another thing will happen if the controller crashes, the failed controller will need another controller to take over its role. In the event of a controller failure, recovery must be performed by redirecting the path to another controller.

Hardware is an important part of the system in the software-defined network. The system used has more than one core device, which will be divided into load balancing and failover. On the network, it is possible to separate the control and data fields. Based on this, it provides scalability, programmability, and centralized control. Moreover, by using this device to achieve ubiquitous connectivity. the existing concept of a software-defined network does not offer this advantage without cost. By utilizing a centralized controller and when one controller fails, service disruptions can occur. This can happen due to the failure of the communication network link and also this kind of situation is unavoidable. This study proposes a failover mechanism for the controller. The mechanism for handling link failures that occur will provide the lowest delay time efficiency.

The advantage of the failure mechanism is that the controller can break the link by changing the data path. This change in flow path occurs instantly and does not take too long. When the main controller fails, the connected device will continue to communicate with the main controller, and when the device does not get any information then the communication is lost. It may take some time to restart communication between the controller and the device, so the time that occurs when the controller changes. A failure mechanism using a heartbeat is implemented to react to failures and monitor the connectivity of the entire network and minimize disruption in the event of a controller failure. SDN approach to heart rate problem can be a better solution for link failure in critical network infrastructure.

1.1 Rationale

The approach is taken in minimizing flow entries for the backup path and in delaying the recovery of the required controller failure. When the recovery controller succeeds in moving to another controller, the results show that the time required is approximately 45ms and 70-130ms, respectively. And this

does not consider the situation where the controller or switch when crashes [1]. In failure detection, the failure detection time is defined as the period time of the last controller message communication time. Using a fake controller is defined as the percentage of task assignments sent. In moving the controller has a delay switch-controller an average time of 8.1568 ms [2]. The controller here is a critical point when a failure occurs and therefore, the controller reduces overall network availability. However, most of the controllers do not support the control field restoration mechanism in case of a controller failure, such as NOX, POX, Beacon, MUL. The recovery phase starts after the detection of the main controller failure. This includes steps to specify a new main controller with a delay time of 150 ms [3].

When a failure interrupts the main work path. The affected OpenFlow will notify the OpenFlow controller about failure (Out of sync message to inform the controller of network events and changes). failure, the OpenFlow Center (OFC) controller will calculate the shortest path back up from the source and requires a network recovery time of around 200 ms [4]. Failover algorithm, no matter how big the topology is, the amount of failure detection time and the flow adjustment time is more or less the same. Therefore, it makes sense to deduce the total recovery time in a large-scale topology by referring to the results obtained in a small-scale topology. Through experiments, that the time to detect a link failure is relatively long (around 335 ms) due to inefficient implementation flows, and the time required for 15 ms to resolve a downlink failure [5].

The controller can be critical because it might lose the connection between OpenFlow and the controller and might not be able to recover failure with the controller or the OpenFlow switch alone. During the test, it was shown that the OpenFlow switch lost its master controller and could not connect to other slave controllers. Restoring the controller requires a linear round trip time of 1ms, but recovery time increases depending on the size of the network. That way the controller starts the process to become a new master controller. The newly selected master controller sends a message requesting a role change to the OpenFlow switch [6]. However, time to recover the damaged path, in addition to the detection time, including the delay introduced by the propagation time to notify the event to the controller, the path recalculates and network reconfiguration by the controller. As a result, the path restoration initiated by the controller possibly takes more than 100ms to complete, which is considered too long [7]. Fast failover mechanism (sub 50ms) The scheme relies on link failure detection by combining primary and backup paths configured by central OpenFlow [8]. Controller and implement failure detection per link using Bidirectional Forwarding Detection (BFD) [9]. Proper delay in waiting time can reduce failure detection time. Apart from failure detection, we need to do it reassign the switches under the failed controller to a suitable alternative controller [10]. Several data-link layer failure detection protocols exist, such as the Spanning Tree Protocol (STP) or Rapid STP, which are designed to maintain the distribution tree on the network by updating the port state on the switch. This protocol, however, can be classified as slow in performing controller detection [11].

1.2 Theoretical

In previous studies, Master controller actively releases its master role to the disconnected OpenFlow switch and notify slave controller to change its role. When the slave controller continues to receive a port status message from an OpenFlow switch, have up-to-date information about connectivity (i.e., network topology) to the underlying OpenFlow switch. If the slave controller has a connection to the OpenFlow switch, it sends the role change confirmation message to the original master controller so the original master can change its role from master to slave. Meanwhile, the slave controller changed its role from slave to master so it works the previously disconnected OpenFlow switch.

Master controller failure can occur due to several things, namely software failure, interference with the network connected to the master controller, or the master controller hardware is a failure. Software failures can occur through maintenance issues, bugs, or attacks. Maintenance issues usually occur while the software is being updated or restarted. A hardware failure occurs due to a lack of maintenance or power supply. Hardware needs to be monitored to avoid dust because too much dust on a device left unattended will cause hardware components to overheat. Power failure can occur when there is a hardware problem, as well as insects, as it can cause a short circuit in the hardware [12]. Meanwhile, Failure detection is important in determining controller failure with heartbeat message and failure message. Failure messages are a graceful way for controllers to fail. If the controller is shut down due to maintenance, it can send a failure message to its nearest neighbors informing them of its status. Heartbeat messages are a way for a neighboring controller to determine if it is on [13].

1.3 Conceptual Framework/ Paradigm

This section describes the basic concepts of SDN, as well as combined them into a distributed SDN controller. The description is followed by an analysis of the type of failure and how to detect a failed controller, as well as the recovery time that the controller performs when a failure occurs in the master controller.

1.4 Statement of the Problem

The outline of the problem raised in this research is how to find out how long it will take to recover the controller when the master controller is down. This is taken from previous research, which has discussed the failure of the master controller and how to move from slave controller to master controller.

Based on the main problems above, the following are the sub-problems that are the focus of work in this research:

- How to detect the master controller failure?
- How if the master controller is off and some come back on?
- How to measure the recovery time of slave controller to master controller?

1.5 Objective and Hypothesis

The purpose of this experiment is to calculate the link failure delay on a specified network software using a failure recovery mechanism. One failure detection method that they implement is to send a heartbeat message in which if the controller misses 3 consecutive heartbeat messages, the failover procedure starts and requires a recovery time. good to allow the device to reconnect normally.

The purpose of this experiment is to detect failures that occur in the master controller, the time required for recovery from the slave controller to the master controller, and analyze if the failed master controller suddenly revives. One of the failure detection methods they implement is sending successive heartbeat messages, a failover procedure is used to allow the devices to reconnect normally. Based on the previous points, this study analyzes several studies related to recovery time. The failure detection method and algorithm that we use can reduce the recovery time when the master controller fails and the backup controller can take over the role of the new master controller.

This proposal hypothesizes is that if one of the controllers fails, it takes the smallest delay time to be able to determine which controller will be the master controller when the controller failure occurs.

1.6 Scope and Delimitation

The scope of this research is the controller installs two main and backup lines on the network using flow table entries. When the mainline fails, the controller deletes the mainline flow entry and network traffic is routed through the backup line. The controller installs new primary and backup lines on the network and the old lines are eventually removed using the flow expiration mechanism. The series of processes carried out in the proposed research are controller recovery time, switching from slave controller to master controller.

1.7 The Contributions

There are several challenges and problems in analyzing SDN. One is about handling controller failures. The purpose of this analysis is how to find out how long it will take for the controller to recover when the master controller is down. There are several answers, one of which is how to detect the failure of the master controller, then what if the master controller is dead and some time back up, and also how to measure the recovery time of the slave controller to the master controller. To answer this problem, we raised it as motivation and contributed to this research. This study conducted an analysis that usually occurs in SDN, to determine the recovery time later.