Load Balancing in SDN Controllers

Submitted in partial fulfillment of
the requirements for the award of the degree of

Bachelor of Technology
in
Computer Science and Engineering

Submitted by

14CS01038 Shailesh Kumar

Under the guidance of

Dr. Padmalochan Bera

Department of Computer Science and Engineering
Indian Institute of Technology Bhubaneswar

B'Tech Project 2017

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY BHUBANESWAR

Certificate

This is the project entitled **Load Balancing in SDN Controllers** was carried out by **Mr. Shailesh Kumar**, Roll No **14CS01038**, a bonafide student of **Indian Institute of Technology Bhubaneswar** in partial fulfilment for the award of **Bachelor of Technology** in Electrical Engineering during year 2017. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

Dr. Padmalochan Bera
Assistant Profesor
(Project Supervisor)

Madhukrishna Priyadarsini
PhD Research Scholar
(Project Guide)

Date:December 11, 2017

Abstract

Software Defined Networking (SDN) has gained significant attention from network researcher community and industries in recent years. Software defined network is an approach to design, build, and manage networks that separates the networks control (brains) and data (muscle) planes enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services. The SDN platform provides various advantages such as programmability, potential for task virtualization and schedulability and easy management of the network. However, it introduces new challenges towards scalability and performances, security hardening, cross-layer communication protocols, etc. It is important to understand and analyze the performances and limitations of SDN for implementation and deployment in live network environments and applications. To increase scalability and availability in SDN distributed controllers are placed in the network, each controlling a its domain of network. This paper reports different load balancing algorithms, their advantages and disadvantages, which is better in which case and proposes a new load balancing algorithm for SDN controllers.

Acknowledgments

With deep regards and profound respect, I avail this opportunity to express my deep sense of gratitude and indebtedness to my supervisor **Dr. Padmalochan Bera**, School of Electrical Sciences, Indian Institute of Technology, Bhubaneswar for his inspiring guidance, constructive criticism and valuble suggestion throughout this work. It would have not been possible for me to bring out this thesis without his help and constant encouragement. I express my heartfelt thanks to **Madhukrishna Priyadarsini** for their constant support in helping me in the successful completion of my project.

Contents

List of Figures

Chapter 1

Introduction

The recent trends of digitization of data in large scale calls significant changes or disruption in the communication technologies and network platforms towards scalable, configurable, performance-centric, pay-per-use and demand driven application execution environment. The traditional network, computing infrastructure, and protocol stack may not be suitable to provide adequate solutions to such growing and heterogeneous demands. This triggered the emergence of a different approach to network systems architecture, called Software-Defined Networking (SDN). Software Defined Networking is a layered network architecture that provides unprecedented programmability, automation, and network control by decoupling the control plane and the data plane of the network. In SDN architecture, network intelligence and states are logically centralized, and the underlying network infrastructure is abstracted for network applications. Network architectures in which the control plane is decoupled from the data plane have been gaining popularity with scope of research and developments. One of the major features of this approach is that it provides a more structured software environment for developing network-wide abstractions while potentially simplifying the data plane.

SDN offers many advantages, such as centralized and decentralized control of multiple cross-vendor network elements, mainly data plane platforms with a common API abstraction layer for all SDN-enabled equipment. It also reduces the complexity of network configuration and operation that is achieved by automation high level configuration is translated into specific forwarding behaviour of network elements. SDN allows easy deployment of new protocols and network-services as a result of high operation abstraction. Increased control granularity in SDN allows a per flow definition with a high granularity policy level. SDN infrastructure can adjust to the specific user application running on it via the control plane, which greatly improves the user experience.

1.1 Motivation

Traditional IP networks are complex and hard to to configure the network according to predefined policies, and to reconfigure it to respond to faults, load, and changes. To express the desired high-level network policies, network operators need to configure each individual network device separately using low-level and often vendor-specific commands. Automatic reconfiguration and response mechanisms are virtually nonexistent in current IP networks. Enforcing the required policies in such a dynamic environment is therefore highly challenging. To make it even more complicated, current networks are also vertically integrated. The control plane (that decides how to handle network traffic) and the data plane (that forwards traffic according to the decisions made by the control plane) are bundled inside the networking devices, reducing flexibility and hindering innovation and evolution of the networking infrastructure. The transition from IPv4 to IPv6, started more than a decade ago and still largely incomplete, bears witness to this challenge, while in fact IPv6 represented merely a protocol update. Due to the inertia of current IP networks, a new routing protocol can take five to ten years to be fully designed, evaluated, and deployed.

1.1.1 Need of Load Balancing

To increase the scalability and availability of SDN network more than one controllers are added to the network. Although we can improve the scalability and availability with the help of multiple controllers, another issue is inevitable. How to maintain the map between a switch and a controller in case of one of the controllers is not overwhelmed? Even if we deploy the switches and controllers very carefully, its difficult for controllers to adapt to changeful traffic load. This problem could be explained from another perspective. Real networks show the characteristics from two aspects: temporal and spatial [7]. For instance, from the temporal angle, there may be less traffic during the night. However, there could be a large scale of flows generated by applications of routing calculation in a very short time. Besides, from the point of spatial, applications running on different controllers possibly compute and generate different numbers of flows, and some switches can get lots of flows compared to other domains of the network. So its essential for us to balance load dynamically among distributed controller cluster instead of static network configuration. Overloaded controller should be detected and high-load switches mapped to this controller ought to be smoothly migrated to the under-load controllers so as to improve resource utilization of distributed controller cluster.

Chapter 2

Work Done

2.1 Centralized Load Balancing

One way to solve uneven load problem among distributed controllers is that deploying a super controller, which is responsible for balancing the load of all controllers. Typically, as Fig. 2.1 shows, the centralized decision controller node collects the load information of other controllers and then decides whether a load balancing action should be launched or not. This is a good way to monitor the load change of all controllers from a global view. However, this algorithm may own two limitations. (1) The performance of a centralized node is limited by memory, CPU power and bandwidth. Besides, a centralized node collects load information periodically and it exchanges lots of messages frequently with other controllers, which will lead to performance reduction of the whole system. Moreover, if the central node collapses, the whole load balancing strategy is down. This goes against availability among distributed SDN controllers. (2) As Fig. 2.1 exhibits, each load balancing action need two network transmissions: one for collecting load and the other is for sending commands. In this situation, the aggregated load information may be past due and the command is lag behind the real load condition.

2.1.1 Improvement in Centralized Controller

If the network is not heavily loaded and only some individual is heavily loaded then load of a single controller can be distributed into the network. In such case control messages will not impact much on performance of SDN. But in case of high load on whole network, the load on master controller will increase, and these control message overhead will degrade the SDN performance.

The problem of node failure can be served by electing a new master(load balancer) controller. All controllers knows the interval after which load is

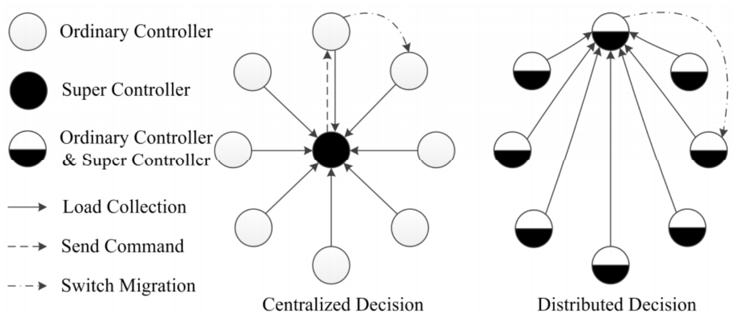

Figure 2.1: **Centralized and Distributed Load Balancing**

requested by the master controller. But if in case it does not happen rest of the controller can elect the next master controller. One approach is CSMA based in which once a controller doesn't receive any load request message it will start a random timer for contention window and when timer expires it will become the new master controller and broadcast this event to all other controllers.

The only problem in above approach is collision, that is two controllers becoming master controller at a time. An alternative to above problem is to assign the each controller an id varying from 0 to n-1 and in case of failure of k^{th} master controller, $(k+1)mod(n)^{th}$ controller will become the next master controller.

2.2 Distributed Load Balancing

As Fig. 2.1 shows, the algorithm based on distributed decision allows every SDN controller collect other controllers load, make its own decision locality. In other words, every distributed controller acts as not only ordinary controller but also super controller. By this way, we will get two advantages. (1) This algorithm is totally based on distributed architecture, which guarantees the scalability and availability of distributed SDN controllers. (2) This algorithm just needs one network transmission for gathering load and doesnt need push command to other controllers. As a result, the decision delay will be reduced in this case. In addition, in order to avoid all controllers collect the load information in cluster too frequently, leading to large numbers of messages transmitted for load collection in SDN environment, we adopt a dynamic and adaptive threshold to prevent frequent load collection.

Fig. 2.2 illustrates the flowchart of Algorithm. The controller will periodically collect its own load information. Then it will check whether the load is beyond the threshold. If the load exceeds the threshold, the controller will

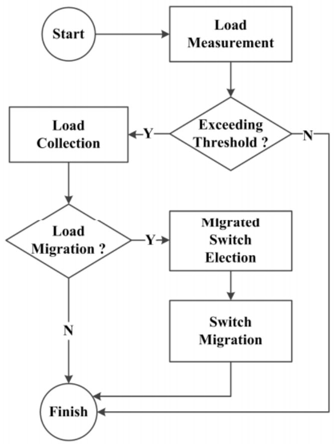

Figure 2.2: **Flow Chart of Distributed Load Balancing**

gather other controllers load information. After aggregating all load information from the controller cluster, this high load controller will make migration decision and select which switch will be migrated to a low load controller.

2.2.1 Load Measurement

Load on a controller can be defined as average packet arrival rate.

$$L_c = Packet\ arrival\ rate$$

2.2.2 Load Collection

One controller can get the loads of other controller by sending a request message which is responded by the other controller. However, frequently sending or receiving messages may decrease the performance of the whole system. In order to address the issue above, a load collection threshold (denoted with LT(load threshold), short for controller's load threshold) is adopted. Still, if all distributed controller nodes are overloaded (which means the load of each controller exceeds LT), setting the LT equal to 1000 will result in the communication overhead is n^2 (n represents the number of controllers). This will also degrade the performance of the distributed SDN controllers. In order to tackle this problem, an adaptive load collection threshold adjusting algorithm named AdaptiveLT is adopted. This will reduce the overhead to $\frac{n^2}{2}$.

Algorithm 1 AdaptiveLT

```
Require: {L1,L2,..Ln}
1: $\alpha = \frac{1}{n} \sum_{i=1}^{i=n} (L_i)$
2: ILT(initial load threshold) = 1000
3: LT = max(ILT,$\alpha$)
4: return LT
```

2.2.3 Decision Maker

We define a load balancing rate ρ as.

$$\rho = \frac{\sum_{i=1}^{n}(L_i)}{n * max_{i=1}^{n} L}$$

The value of ρ is always between 0 to 1. Value of ρ closer to 1 means load is uniformly distributed throughout the network and vice-versa. Hence no need to balance load if ρ is closer to 1. Assume for the time being threshold for ρ is 0.7, that is if $\rho < 0.7$ then load transfer will happen.

In case two controller's load > LT and both having $\rho < 0.7$ then both will migrate their switch to same controller and thus increasing the load on the target controller. Therefore one more constraint is imposed that a controller will go for load balancing only if it's load is maximum of all controller's load.

2.2.4 Switch Election

Now if ρ is less than certain threshold we may go for migrating a switch to a low-load controller. If a switch's control(with load L_s) is transferred to another controller, then there will be decrease in this controller's load by L_s and increase in target controller by L_s. Hence difference between this controller's load and lowest-load controller's load should be greater than twice the load of migrating switch.

$$L_{overloaded} - L_{target} >= 2L_s$$

2.3 Hybrid Approach

Distributed Load Balancing works very much fine, under low load condition in the network. But in case of high load, LT is set to average load of the network, and some of them will have load less than threshold and others will have more. On an average there will be $\frac{n}{2}$ nodes whose L_c will be greater than LT and thus going for load collection. So in case of high load on network there will always be load collection. But how frequent the load balancing algorithm is run in controller is an important factor. It must be less than the period of Centralized load balancing otherwise it will not be better than Centralized load balancing technique in high load condition.

More over if we see, LT value only increases when L_c is greater than LT. In case if high LT value is reached, then controller will not opt for load balancing unless L_c is greater than LT. This will cause overall performance degradation. Hence we must find some ways to lower the LT as network load reduces. One solution is when to recollect the loads when LT - L_c is greater than δ(load-gap, fixed constant).

Operation cost of Distributed algorithm is more than that of centralized one. There is not much more difference in performance of SDN under high load in both algorithms, but distributed one seems have higher performance degradation. Therefore we can use distributed approach in case low load in network and centralized approach in case of high load in the network.

Chapter 3

Conclusion and Future Work

In this report I presented the different load balancing technique, their advantages and disadvantages, which one will work better in which case. My solution is adopt the hybrid approach, that is, to use distributed algorithm in case of high load and centralized algorithm in case of high load.

Future Work includes validation of proposed theory that centralized approach works better in high load condition, by implementing the above proposed works in Opendaylight controller and evaluate the performance with mininet simulation. Performance evaluation will include finding out the optimal value of ILT, threshold for ρ(load balancing rate) and δ(load gap) for updating LT in case of load reduction form high to low.

References

[1] Yuanhao Zhou, Mingfa Zhu, Limin Xiao, Li Ruan, Wenbo Duan, Deguo Li, Rui Liu, Mingming Zhu ***A Load Balancing Strategy for SDN Controller based on Distributed Decision***, `http://ieeexplore.ieee.org/document/7011337/`

[2] Diego Kreutz, Fernando M. V. Ramos, Paulo Esteves Verssimo, Christian Esteve Rothenberg, Siamak Azodolmolky, Steve Uhlig ***Software-Defined Networking: A Comprehensive Survey*** `http://ieeexplore.ieee.org/document/6994333/`