www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 5 Issue 12 Dec. 2016, Page No. 19648-19652

Multi-threaded QoS Architecture for Multimedia Services over Software Defined Networks

Ms. Pooja A. Baleghate[1], Prof. Sachin B. Takmare[2], Prof. Pramod A. Kharade[3]

[1]Department of Computer Science and Engg., Bharati Vidyapeeth's College of Engineering,
Kolhapur, Maharashtra, India
poojabaleghate@gmail.com
[2]Department of Computer Science and Engg., Bharati Vidyapeeth's College of Engineering,
Kolhapur, Maharashtra, India
sachintakmare@gmail.com
[3]Department of Computer Science and Engg., Bharati Vidyapeeth's College of Engineering,
Kolhapur, Maharashtra, India
pramodkharade@gmail.com

Abstract: ***This paper presents novel multi threaded controller-forwarder architecture to support QoS for multimedia streaming over SDN. We foresee that, large network is partitioned into domains; each domain is managed by a controller, where each controller performs optimal QoS routing and shares the routing information with other domain controllers. To this effect, this paper proposes (i) an algorithm for super controller managing inter domain routing, (ii) an algorithm for controller managing intra domain routing, to find out an optimized QoS routes. We apply these extensions to streaming videos and compare the performance of proposed architecture with single threaded controller-forwarder architecture. Our experimental result shows that the proposed architecture performs faster than the single threaded controller-forwarder architecture.***

Keywords: Software Defined Network (SDN), quality-of-service (QoS), multimedia streaming, OpenFlow.

1. Introduction

Media streaming attempts to overcome the problems associated with file download, and also provides a significant amount of additional capabilities. Typically when you download a file, you must wait for the entire file to finish downloading before you can open and view it. It can be very frustrating for large media files. Streaming media improves the download process by downloading a portion of the media file (say, the first few seconds of a video) and then allow the user to view that bit while it's downloading the next couple of seconds. As the process continues, the user watches a little while the next piece is downloading in the background until the user has seen the entire video. Since the user doesn't have to wait for the entire video to download before viewing it, streaming can produce a less frustrating viewing experience. In current Internet architecture, there are number of basic problems that afflict media streaming [1]. Media streaming over the Internet is difficult because the Internet only offers best effort service. That is, it gives no assurances on bandwidth, loss rate, or delay jitter. Specifically, these characteristics are dynamic and unknown. Therefore, a key goal of media streaming is to design a system that supports some level of QoS to reliably deliver high-quality video over the Internet when dealing with unknown and dynamic Bandwidth, Delay jitter and Loss rate. Many architectures [2], [3], [4], [5], [6], [7] have been proposed in the literature to provide QoS for media streaming, yet none of them is truly successful and globally implemented. OpenFlow is a successful Software Defined Network (SDN) paradigm that decouples the control and forwarding layers in routing [8], [9]. SDN is an emerging and its uniqueness comes by the fact that it provides programmability through decoupling of control and data planes, and ensures simple programmable network devices, instead of making them more complex. With SDN, the control of the network can be done separately on the control plane without impact on the data flows. The intelligence of the network can be removed from the switching devices and placed on the controller. Meanwhile, the switches can be controlled externally by software without need of onboard intelligence. The separation of control from data planes provides not only a simpler programmable environment. Numerous network device merchants have effectively begun to deliver OpenFlow-empowered switches or routers. Therefore, SDN or OpenFlow will incrementally spread all through the world sooner rather than later as new OpenFlow empowered switches are sent. OpenFlow has also attracted the attention of numerous organizations offering cloud administrations, and it will further permit system administration suppliers to offer inventive multimedia administrations with progressively reconfigurable QoS. This is the primary inspiration behind employing OpenFlow architecture in this work. Yet, current OpenFlow specification [10] does not provision communication between different controllers managing separate network domains. It is vital to implement a distributed control plane with multi-threaded controllers and forwarders to manage multi-domain, multi-operator SDNs. This paper presents, a multi threaded controller-forwarder architecture which supports QoS for media streaming over SDN and performs faster than single threaded controller-forwarder architecture.

The remainder of the paper is organized as follows. We discuss the literature review in section II. Section III proposes schematics of multi-threaded controller-forwarder mechanism. Section IV presents simulation results comparing the multi-threaded and single threaded mechanisms. Section V draws the conclusion. Future work is described in section VI.

2. Literature Review

Many non-standard distributed control plane structures have been proposed in the literature.

R. Ramanathan, S. Shenker, T. Koponen et al. [2] introduced Onix - a distributed platform that defines a general arrengment of APIs actualize a control plane. In this model, a network-wide control stage, running on at least one server in the network, manages an arrangement of simple switches. The control platform handles state distribution gathering data from the switches and conveying the appropriate control state to them, and in addition planning the state among the different platform servers and gives an automatic interface upon which developers can manufacture a wide variety of management applications.

A. Tootoonchian and Y. Ganjali [3] proposed an event-based appropriated control plane structure, named HyperFlow, permitting proficient distribution of the network events among controllers.

D. Levin, J. Rexford, Feamster et al. [4] presented a software defined Internet exchange (SDX) design whose expansions will permit multi-site arrangements of SDN. The single-node SDX architecture looks to some extent like a traditional route server, however its design makes a couple significant takeoffs from a route server. In the first place, the SDX controller permits each AS to apply a custom route choice procedure to choose at least one best route to each Internet destination. This feature appears differently in relation to existing inter domain routing rehearses, whereby each AS must apply the traditional BGP route determination procedure to choose a single best route to each destination. Second, the SDX controller can straightforwardly influence sending by upgrading switch-table entries, instead of indirectly influencing route control by means of BGP policy mechanisms.

X. Dimitropoulos, Kotronis et al. [5] proposed a control plane design concentrating on advancing inter-domain routing so that the legacy BGP stays good. The paper presents SDN ideas to enhance inter-domain routing. The paper proposes to outsource the routing control plane of an ISP to an external trusted supplier, i.e., the service contractor. The contractor represents considerable authority in routing management.

T. Koponen, S. Shenker Raghavan et al. [6] introduced Software Defined Internet Architecture (SDIA) considering both inter and intra-domain sending tasks. The objective of this paper is basic: to change architectural advancement from a hardware issue into a software one. And the answer is somewhat standard, acquiring intensely from long- standing (e.g., MPLS) and developing (e.g., SDN) deployment practices.

Hilmi E. Egilmez and A. Murat Tekalp [7] proposed novel QoS expansions to distributed control plane designs for interactive media delivery over large-scale, multi-operator Software Defined Networks (SDNs) using single threaded controller-forwarder mechanism.

3. Proposed Work

As of now, it is difficult to dynamically change network routing on a per-flow basis. Ordinarily, when a packet arrives at a router, it checks the packet's source and destination address pair with the entries of the routing table, and forwards it as indicated by generally fixed, predefined rules (e.g., routing protocol) configured by the network operator. OpenFlow offers a new paradigm to mainly remedy this deficiency by permitting network operators to flexibly define different types of flows (i.e., traffic classes) and associate them to some set of forwarding rules (e.g., routing, priority queuing). So as to guarantee ideal end-to-end QoS for multimedia delivery, gathering up-to-date global network state information, such as delay, bandwidth, and packet loss rate is crucial. Yet, over a large-scale multi-domain network, this is a troublesome task because of dimensionality. The problem becomes even more difficult due to the distributed architecture of the current Internet. OpenFlow facilitates this issue by employing a centralized controller. Rather than sharing the state information with all other routers, OpenFlow enabled forwarders directly forward their local state information using the OpenFlow protocol to the controller. The controller processes each forwarder's state information and determines the best forwarding rules using up-to-date global network state information. Nonetheless, the current OpenFlow specification is not appropriate to large scale multi operator telecommunication networks. Consequently, there is requirement for a distributed control plane comprising of multiple controllers each of which is responsible for a part (domain) of the network. Fig. 1 shows the schematics of multi threaded controller-forwarder mechanism.

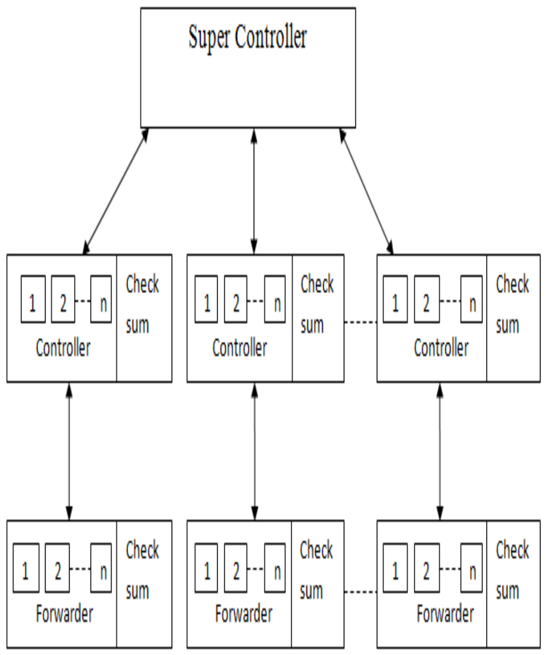

Figure 1: Schematics of Multi-Threaded Controller-Forwarder mechanism

In proposed architecture, the network is partitioned into domains and each domain is managed by a controller. The architecture consists of consists of three main modules: Super Controller, Multi threaded Controllers and Multi threaded forwarders.

A. Super Controller

The super controller is the core of an SDN network. It lies between network devices at one end and applications at one end. Any communications between applications and devices have to go through controller. SDN controller is an application in SDN that manages flow control to enable intelligent networking. Super Controller is responsible for inter-domain routing. Super controller gets aggregated routing information from all controllers through the interface, and based on this knowledge an inter-domain route is determined. Super

controller pushes inter-domain routing decisions to all controllers using controller-controller interface. Controller-Controller interface allows multiple controllers to share necessary information to cooperatively manage the whole network. This interface allows controllers to share aggregated routing information and QoS parameters among themselves to help making inter-domain routing decisions with end-to-end QoS requirements. In case of drastic events such as network failure or congestion, the interface informs other controllers actively. It periodically collects network topology or state information, distributes and keeps them in sync.
Following algorithm is used by super controller managing inter-domain routing in the distributed control plane.

Steps:

1. Procedure main calls procedure route and update, in parallel.
2. Procedure route gets aggregated network information from interface and decides inter-domain routes.
3. The inter-domain routing decisions are sent to each domain's controllers through the interface.
4. Procedure update checks that if there is a route failure. If there is such an event, then the route procedure is restarted.

B. Multi-threaded controller

The Controller is responsible for intra-domain routing. Each controller gets inter-domain route(s) determined by the super controller. It starts 'n' number of threads in single environment to perform faster than a single threaded controller.
Following algorithm is used by each controller managing intra-domain routing in the distributed control plane.

Steps:

1. Procedure main calls procedure route and update, in parallel.
2. Procedure route determines an intra-domain route and pushes necessary routing information to forwarders.
3. If there are no feasible routes, then a route failure event is triggered and sent to the interface to inform the super controller.
4. Update procedure keeps the network state information up-to-date. If a link failure or congestion event is detected, then the route procedure is restarted to re-optimize intra-domain routing.

C. Multi-threaded forwarders

The forwarder is responsible for data forwarding function to the controller. When a packet arrives at a forwarder, first it is compared against the flow table. If matching entry is found, then the packet will be forwarded to the specified port. If no matching is found, then the packet is forwarded to controller. The controller is then responsible for how to handle the packets. This communication between controller and forwarder is managed by OpenFlow protocol. This model forwards their local state information to the controller. It starts 'n' number of threads in single environment to perform faster than a single threaded forwarder.
These algorithms determine an optimized QoS routes from the inter-domain and intra-domain decisions, so as to improve the performance of the system.

4. Result Analysis

In this section, we apply our multi threaded mechanism to streaming of videos. In order to simulate the proposed architecture, we implemented a simulator by using minimum 1GB RAM and 60GB (or above) hard disk. The proposed system is run in simulation environment with JVM heap size 21496k. The simulation results are carried out with different thread counts ranging from 0 to 10. As the thread count increases, the time required for execution decreases. The simulation results are shown in Fig.2, 3 and 4.

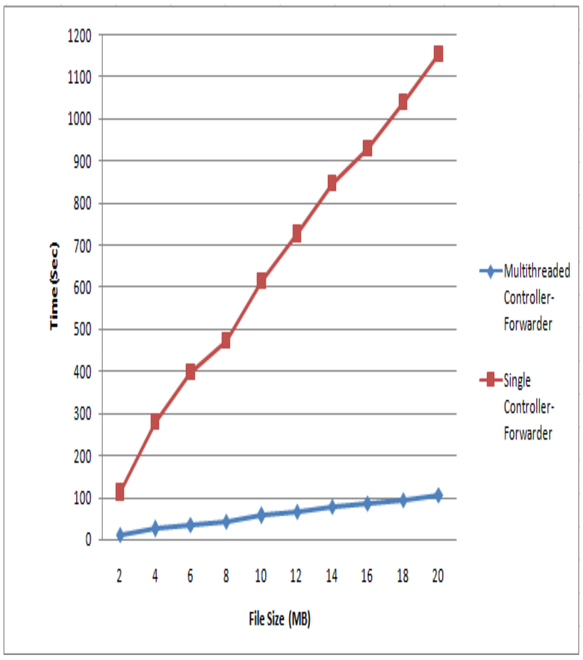

Figure 2: Time comparison for single-threaded and multi threaded controller-forwarder mechanism
(Thread count 10)

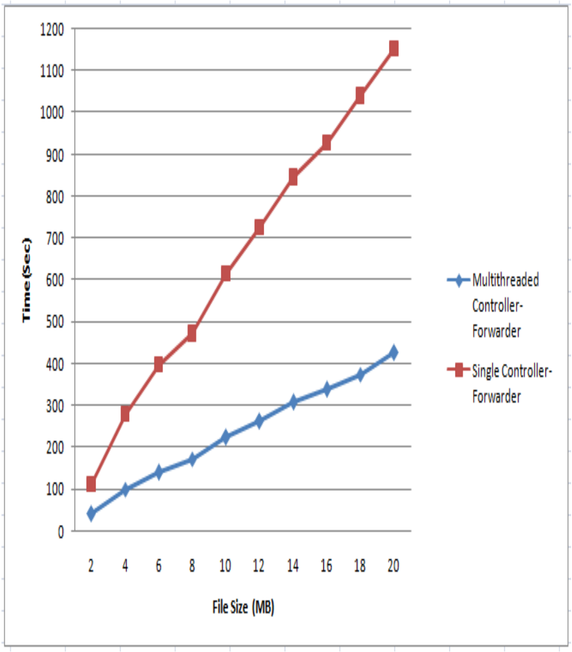

Figure 3: Time comparison for single-threaded and multi-threaded controller-forwarder mechanism
(Thread count 7)

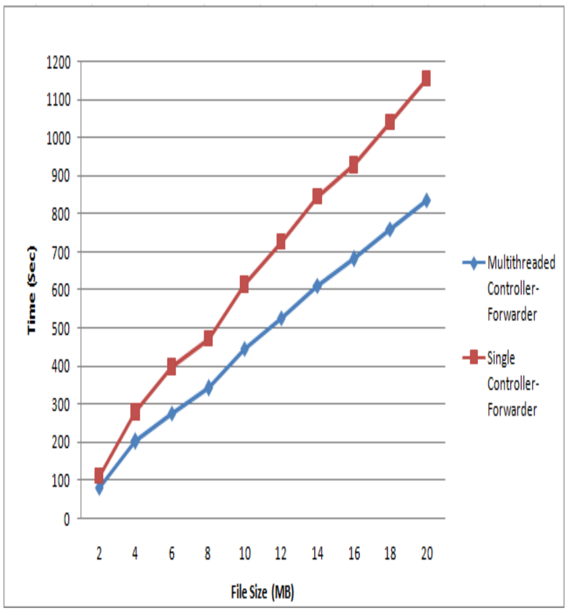

Figure 4: Time comparison for single-threaded and multi-threaded controller-forwarder mechanism (Thread count 3)

In Fig. 2 we compare, the time required for single threaded controller-forwarder mechanism and multi threaded controller-forwarder mechanism (with thread count 10) to transfer media files sizes from 2 MB to 20 MB from source to destination. Similarly, In Fig. 3 we compare, the time required for single threaded controller-forwarder mechanism and multi threaded controller-forwarder mechanism (with thread count 7) to transfer media files sizes from 2 MB to 20 MB from source to destination. In Fig. 4 we compare, the time required for single threaded controller-forwarder mechanism and multi threaded controller-forwarder mechanism (with thread count 3) to transfer media files sizes from 2 MB to 20 MB from source to destination.

5. Conclusion

This paper proposes a multi-threaded controller-forwarder mechanism. The mechanism improves the QoS for multimedia streaming over software defined network and performs faster than a single threaded controller-forwarder mechanism.

6. Future Work

The proposed system supports multi threaded controller-forwarder mechanism for enhancing the QoS of multimedia traffic over Software Defined Networks. There are several other domains where we can do some interesting research. For instance,

1) Buffering at node: in current theory, we don't have any provision for minimizing link delay. These link delays will affect the total performance of the system. So, we can implement buffering at those nodes where high link delays occur.
2) Caching: Caching helps to minimize processing of frequently requested data. Caching will be implemented at the Super Controller so that the frequently required data (i.e. the data which is mostly requested at super controller) can be stored in cache memory so as it will take less amount of time for processing the requests. We leave the study of these problems as our future work.

Acknowledgement

There have been many contributors for this to take shape and authors are thankful to each of them.

References

[1] J. H. Saltzer, D. P. Reed, and D. Clark, "End-to-end arguments in system design," ACM Trans .Comput. Syst.,vol.2,no.4,Nov.1984.

[2] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: A distributed control platform for large-scale production networks", OSDI, pp. 351–364, 2010

[3] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for OpenFlow", in Proc. INM / WREN'10, pp.3–3, 2010.

[4] N. Feamster, J. Rexford, S. Shenker, D. Levin, R. Clark, R. Hutchins, and J. Bailey, "SDX: A software-defined internet exchange", Open Netw. Summit, Apr. 2013.

[5] V. Kotronis, X. Dimitropoulos, and B. Ager, "Outsourcing the routing control logic: Better internet routing based on sdn principles", in Proc. 11th ACM Workshop Hot Topics Netw., pp. 55–60, 2012.

[6] B. Raghavan, M. Casado, T. Koponen, S. Ratnasamy, A. Ghodsi, and S. Shenker, "Software-defined internet architecture: Decoupling architecture from infrastructure", in Proc. 11th ACM Workshop Hot Topics Netw., pp. 43–48, 2012.

[7] Hilmi E. Egilmez, A. Murat Tekalp, "Distributed QoS Architectures for Multimedia Streaming Over Software Defined Networks", IEEE Trans. on Multimedia, vol. 16, no.6, Oct.2014.

[8] Open Networking Foundation (ONF), PaloAlto, CA, USA, "Software defined networking: The new norm for networks" 2012 [Online]. Available: https://www.opennetworking.org/images/stories/downloads/openflow/wp-sdn-newnorm.pdf

[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," SIGCOMMComput.Commun.Rev.,vol.38,no. 2, pp. 69–74, Mar. 2008.

[10] Open Networking Foundation (ONF), Palo Alto, CA, USA, "OpenFlow Switch Specification v1.3.1" [Online]. Available: https://www. opennetworking.org/ Accessed: Sep. 6, 2012.

Author Profile

Ms. Pooja A. Baleghate received the B.E. degree in Information Technology in 2013 from Shivaji University. She is currently pursuing M.E. in Computer Science and Enginnering from Shivaji University.

Prof. Sachin B. Takmare is working as Assistant Professor in Computer Science and Engineering Department of Bharati Vidyapeeth's College of Engineering, Kolhapur with teaching experience of about 10 years. He has published about three International Papers and five National Papers.

Prof. Pramod A. Kharade is working as Assistant Professor in Computer Science and Engineering Department of Bharati Vidyapeeth's College of Engineering, Kolhapur with teaching experience of about 8.5 years. His areas of specialization are Information Security and Discrete Mathematics.