Managing Failures in IP Networks Using SDN Controllers by Adding Module to OpenFlow

Vivek S [1], Karthikayini T [2]

[1]PG Scholar, Department of Computer Science and Engineering, New Horizon College of Engineering, Bangalore, Karnataka, India

[2]Assistant Professor, Department of Computer Science and Engineering, New Horizon College of Engineering, Bangalore, Karnataka, India

Abstract: ***The creation of networks based on software-defined networking (SDN) are becoming complex to distribute to all cities and it is very difficult to configure individual routing components and routing path using traditional components. So an easy way to manage the network components and routing is needed. SDN is the latest area which separates the network components to two planes – Data Plane and Control Plane. By this separation the logic of operation of network devices can be far separated at different place in Control plane. By considering the above situation a new configuration is made possible to centralized control over the global view of network with improved architecture will be considered. This paper produces a solution on a distributed hierarchical control plane as a Orion to verify the feasibility of the hybrid hierarchical approach.***

Keywords: Software defined networking (SDN), central controller, scalability, Orion, hybrid hierarchical, super-linear and large-scale networks

1. Introduction

In Software Defined Networking (SDN), the control and data planes are decoupled, and the complex control and management functions are stripped out of the network device[1]. Meanwhile, SDN supports a flow-based management to enable highly programmable and flexible of Networks. OpenFlow switches are mainly used to achieve the fine-grained flow control in SDN[2]. However, such decoupled architecture and fine-grained flow control feature bring a large amount of communication messages between the data plane and the control plane which limit the scalability of the SDN network [3, 4, and 5].

The scalability of Software-Defined Networks is so important that many researchers have been trying to solve this problem. Maestro exploits parallelism in single-threaded controller [6]. Beacon employs multi-threaded techniques to improve the scalability of a single controller [7]. But the two studies [6] and [7] do not yet provide communication between multiple controllers. DevoFlow considers the SDN controller handling too many micro-flows, which creates excessive load on the controller and switches [4]. Then it proposes a way that the control plane maintains a useful amount of visibility without imposing unnecessary costs. DIFANE employs authority switches to store necessary rules to share the work load of the control plane [8]. However, both studies [4] and [8] require to modify the Open Flow switch. Then some researchers design different control plane structures to extend the control planes processing ability.

Though the above two kinds of control plane architecture can improve the scalability of SDN networks, they still have unresolved issues. 1) **The flat control plane architecture** cannot solve the super-linear computational complexity growth of the control plane when SDN network scales to large size. We argue that the fine-grained flow control feature of SDN will lead to the super-linear computational complexity growth of the control plane and that limits the scalability of SDN networks. We take an example to illustrate the problem. Assuming that a SDN controller manages M network devices; it uses the to identify the data flow; the SDN controller adopts Dijkstra algorithm to compute routing paths.

In the beginning, the computational complexity of the Dijkstra algorithm is O(M2). When the network size increases N times, there will be N ∗ M nodes. Then the computational complexity of the routing algorithm increases to O(N2M2). If we use N SDN controllers to share the work load of the Thus, if we use N SDN controllers to share the work load of the control plane, the processing capacity will increase N times, but the computational complexity will increase N2 times.

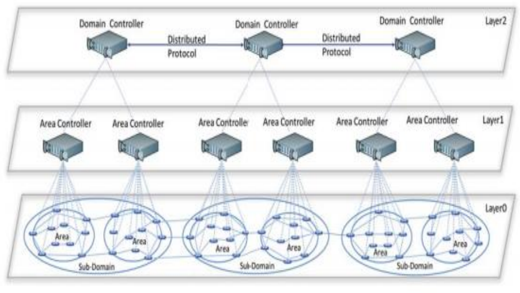

Fig. 1. Orion's Hybrid Hierarchical Architecture

A typical centralized hierarchical control plane, such as Kandoo, cannot solve the issue of the super-linear computational complexity growth of the control plane. 2) **The centralized logical hierarchical** control plane architecture brings path stretch problem. In a network graph, Stretch (u, v) represents the stretch of path from node u to node v. It is defined as Stretch(u, v) = Path(u,v) ShortestPath(u,v), where Path(u, v) is the length of the path from node u to node v and the ShortestPath(u, v) is the corresponding shortest path length. To provide scalable SDN control plane, but the method used brings path stretch problem. The more layers it abstracts, the bigger the path stretch is.

2. Design

In this section, we present the design of Orion, hybrid hierarchical control plane architecture of SDN. Orion focuses on the intra-domain control and management of large-scale networks. In this paper, domain is a whole network which can be controlled and managed by one administrator. It divides into several sub-domains. A sub-domain consists multiple areas that located relatively close to each other. Area refers a region that can be controlled by a single SDN controller.

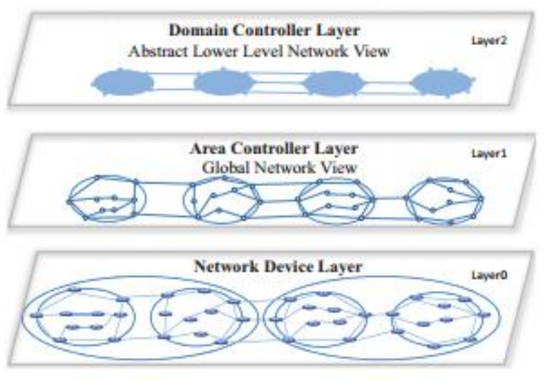

Fig. 2. Network View of the Hybrid Hierarchical Architecture

A. Architecture

The hybrid hierarchical architecture of Orion is shown in Fig. 1. Orion has three layers. 1) The bottom layer of Orion is the physical layer. The physical layer is composed of large amounts of connected Open Flow switches. 2) The middle layer of Orion is the area controller layer. The area controller is responsible for collecting physical device information and link information, managing the intra-area topology and processing intra-area routing requests and updates. Meanwhile, it abstracts its area network view and sends it to the top layer. 3) The top layer of Orion is the domain controller layer. In this layer, the domain controller treats area controllers as devices, and it synchronizes the global abstracted network view through a distributed protocol.

B. Components

There are eight major components in Orion, shown in Figure 3. Among these components, the Host Management, Topology Management, Routing, Storage and Vertical Communication Module have two sub-modules. The two sub-modules are responsible for intra-area information processing and inter-area information processing.

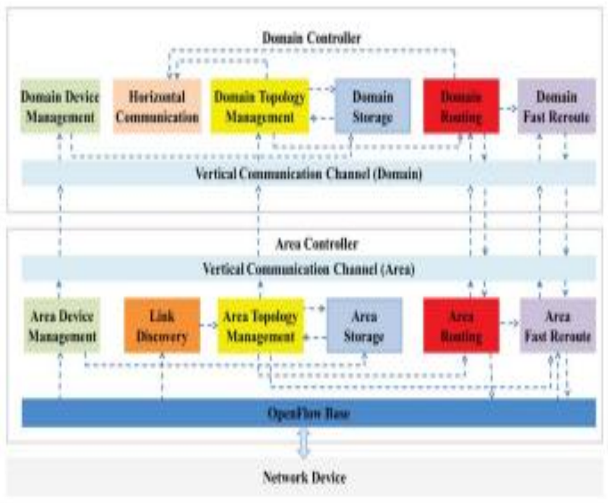

Fig. 3. The modules of Orion.

OpenFlow Base Module: The OpenFlow Base Module is responsible for collecting OpenFlow switch information and receiving messages through the SDN southbound Interfaces. Meanwhile, it provides an interface for the area controller to install rules on OpenFlow switches, such as Flow-Mod or Packet-Out.

Host Management Module: The Host Management Module has two parts to deal with the area host information and domain host information. 1) The Area Host Management SubModule obtains the host information in its area through the ARP packet sent by the host. 2) The inter-area host information is managed by the Domain Host Management Sub-Module. I

Link Discovery Module. The link discovery module obtains the intra-area link information through LLDP protocol. In order to obtain inter-area link information, an area controller sends LLDP packets to all ports of its edge switches with its ControllerID. When the LLDP packet reaches the edge switch in another area, the switch encapsulates the LLDP packet into a Packet-In message and sends the message to its area controller. Then the area controller encapsulates the Packet-In message and gets the TLV message out of the LLDP Packet.

Topology Management Module. The Topology Management module has two sub-modules. 1) The Area Topology Management sub-module manages the physical topology information received from the Link Discovery Module. Meanwhile, it computes the shortest path from every edge switch to other edge switches. Then it sends the switch information and the hops between any two edge switches to the domain controller. When the domain controller receives the above information, it treat an area as a node, and treat the edge switches of the area as a port, the hop between any two edge switches is like the weight of an abstract link between two port.

Storage Module. The Storage Module includes three kinds of information: host information, switch information and link information. Host information includes . Switch information includes . The link information includes abstract link information and real physical link information. There are two kinds of abstract link.

C. Abstracted Hierarchical Routing Method

As we know the abstracted hierarchical control plane architecture brings the path stretch problem. An abstracted hierarchical routing method is designed to address the problem. The abstracted hierarchical routing method is based on the Dijkstra algorithm[16]. Dijkstra algorithm is a graph search algorithm widely used in network routing protocols, such as IS-IS[17] and OSPF[18]. The core idea of the abstracted hierarchical routing method is similar to IS-IS and OSPF.

The abstracted hierarchical routing method is divided into two parts. 1) As the area controller has the detailed inner information of its area, the Area Routing Management SubModule of the area controller pre-computes the inner hops from every inner switch to all edge switches by Dijkstra algorithm and send the result to the domain controller. 2) The Domain Routing Management Sub-Module of the domain controller computes the global shortest path. Though the domain controller level only has the abstracted lower level network view, it can compute the shortest path for the flow based on the sum of the inner path result sending by the area controller and the inter-area path length.

Area Routing Management Sub-Module: When a PacketIn message reaches the area controller, the area routing management sub-module checks the source address and destination address of the message. 1) If the destination address is in the area, the area controller employs Dijkstra algorithm to compute intra-area path. 2) If the destination address is out of the area, the area controller sends the source address and the destination address of the message to the domain controller, and stores the message to a waiting buffer with index.

Domain Routing Management Sub-Module. At first, the domain routing management sub-module uses Dijkstra algorithm to calculate the inter-area routing path. Next, it collects the intra-area hops from the inner switch to all edge switches which is sending by area controllers, and adds the inter-area hops and the intra-area hops together. The shortest length path determining the final forwarding path.

Routing Example: We give two examples to illustrate how Orion carries out the intra-area routing and inter-area routing. The two routing examples are based on the topology shown in Fig. 4. In the topology, a domain has two sub-domains and each sub-domain has two areas. At the same time there are four host (host A, B, C and D) located in the topology.

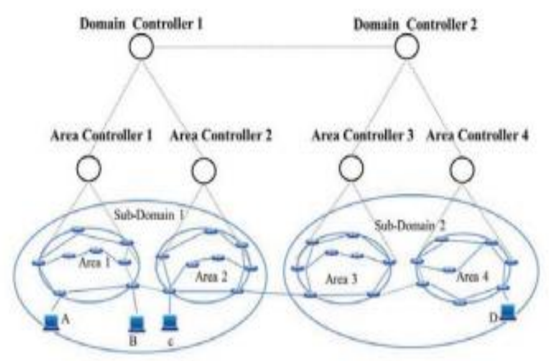

Fig. 4. Example topology

Communicate between two hosts (such as host A and host B) in an area. When the host A sends a data flow to the host B, the switch which connects to the host A generates a PacketIn message and sends the message to area controller1. When area controller1 receives the message, it checks whether the destination address of the data flow is in its area. As host B is located in area1, so area controller1 can find the information of host B. Then it calculates the intra-area routing path from host A to host B based on the intra-area topology. Next, area controller1 sends the routing rules to the switches in the path list, so that the switches can install the rules for the data flow. Finally, when all the switches in the path list are installed routing rules, the data flow sent by host A is forwarded to host B.

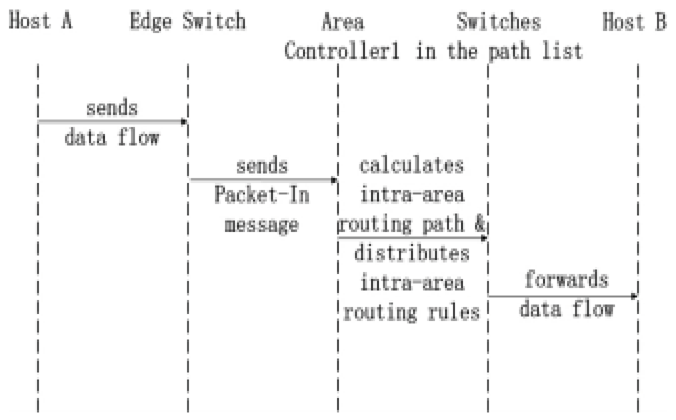
Host A
Edge Switch
Area Controller1
Switches in the path list
Host B
sends data flow
sends Packet-In message
calculates intra-area routing path & distributes intra-area routing rules
forwards data flow

Fig. 5. Intra-area routing sequence of Orion

Inter-area Routing Example. The second example is an example of the inter-area routing. The example illustrates how host C sends data flow to host D with Orion. When host C sends a data flow to host D, the data flow reaches the switch which host C connects to. Then the switch generates a PacketIn message and sends the message to area controller2. As host D is not in area2, when area controller2 receives the message, it extracts the from the Packet-In message and encapsulate it to a simple request, sends the request to domain controller1, and buffers the Packet-In message with an index.

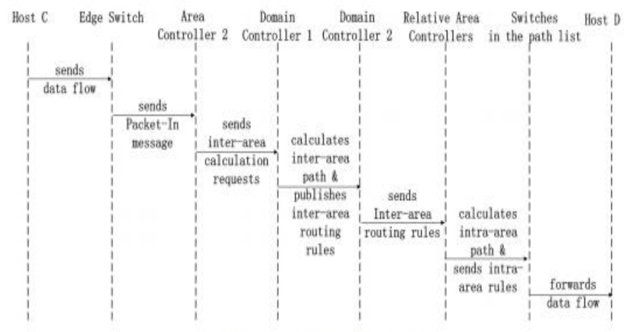
Host C
Edge Switch
Area Controller 2
Domain Controller 1
Domain Controller 2
Relative Area Controllers
Switches in the path list
Host D
sends data flow
sends Packet-In message
sends inter-area calculation requests
calculates inter-area path & publishes inter-area routing rules
sends Inter-area routing rules
calculates intra-area path & sends intra-area rules
forwards data flow

Fig. 6. Inter-area routing sequence of Orion

3. Implementation And Evaluation

In this section, we present the implementation and evaluation of Orion. In this part, we evaluate Orion through both theoretical and experiment ways. A. Evaluation 1) Theoretical Evaluation: We write a simple single threaded Dijkstra algorithm to calculate the path from the source address to the destination address. The algorithm is running under random topology. In the random topology, if there are N nodes in the topology, then the topology has Nx edges, where x is a variable. We run the algorithm on a server with Intel E5645 processors (6cores in total, 2.40GHz) and 64GB memory. We compare the proposed abstracted hierarchical routing method with the traditional Dijkstra algorithm. Under the best conditions, the computational complexity of the abstract hierarchical routing method is O(K2), where K is the number of edge switches. The best conditions refer that the network does not change, the area controllers do not need to re-compute the inner hops from every inner switch to edge switches. Meanwhile, the source host and destination host of the flow are both connected to edge switches. From Fig.7, we can observe that with the increasing number of areas, the computing time of Orion is increased as linear growth, much lower than the traditional Dijkstra routing algorithm.

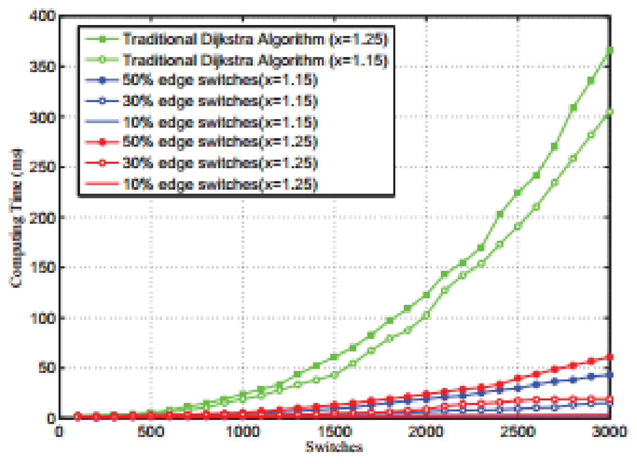

Fig. 7. Theoretical evaluation under the best condition

From Fig.8, we can note that when the network nodes exceed 2900, the abstracted hierarchical routing method proposed by Orion is better than the traditional Dijkstra algorithm.

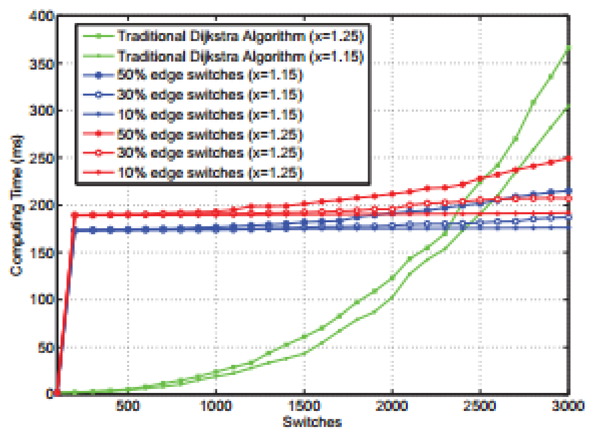

Fig. 8. Theoretical evaluation under the worst condition

2) Experiments Evaluation: In this part, we build a prototype system to verify the feasibility and effectiveness of Orion. The implementation of Orion is in Java. The area controller of Orion is build based on the Floodlight controller [19]. The intra-area OpenFlow Base module, Link Discovery module and Storage module are based on Floodlight. We extend the Floodlight controller to construct the other modules. The domain controller of Orion is not an SDN controller with OpenFlow protocol. The communication between the domain controller and the area controller is through the Vertical Communication Module of Orion, and it is not based on OpenFlow protocol.

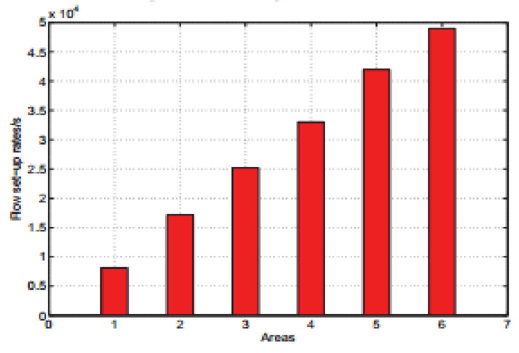

Fig. 9. Flow set-up rate of Orion

we can see that with the increasing number of areas, the delay time gradually increased.

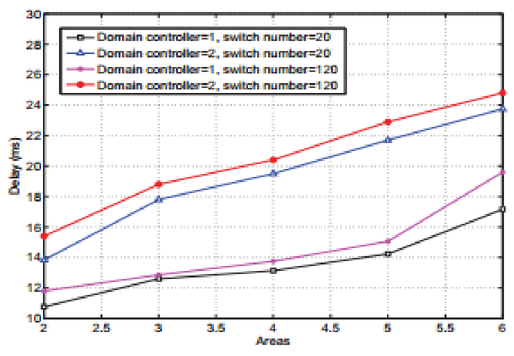

Fig. 10. The average delay time of Orion

4. Conclusion

In this paper, we design and implement Orion, a hybrid hierarchical control plane for large-scale networks. Orion addresses the super-linear computational complexity growth of the control plane when SDN network scales to large size, and solves the path stretch problem brought by the abstracted hierarchical control plane architecture. Further, we evaluate the effectiveness of Orion through theoretical and experiment aspects. Our evaluation results show the efficiency and feasibility of Orion.

References

[1] ONF White Paper, Software-Defined Networking: The New Norm for Networks, Open Networking Foundation, 2012.
[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, OpenFlow: enabling innovation in campus networks, ACM SIGCOMM Computer Communication Review, Vol.38, No.2, pp.69-74, 2008.
[3] A. Tootoocian and Y. Ganjali, HyperFlow: A distributed control plane for OpenFlow, In Proc. ACM INM/WREN, 2010.
[4] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, DevoFlow: Scaling Flow Management for High-Performance Networks, In Proc. ACM SIGCOMM, 2011.
[5] J. McCauley, A. Panda, M. Casado, T. Koponen, S. Shenker. Extending SDN to Large-Scale Networks, In Proc. ONS, 2013.
[6] Z. Cai, A. L. Cox, and T. S. E. Ng, Maestro: A System for Scalable OpenFlow Control, Technical Report, Rice University, 2010.
[7] D. Erickson, The Beacon OpenFlow Controller, In Proc. ACM SIGCOMM HotSDN, 2013.
[8] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, Scalable Flow-Based Networking with DIFANE, In Proc. ACM SIGCOMM, 2010.
[9] 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, In OSDI, 2010.
[10] B.Lantz, B. Connor, J. Hart, P. Berde, P. Radoslavov, M. Kobayashi, T. Koide, Y. Higuchi, M. Gerola, W. Snow, G. Parulkar, ONOS: Towards an Open, Distributed SDN OS, In Proc. ACM SIGCOMM HotSDN, 2014.
[11] S. H. Yeganeh, Y. Ganjali, Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications, In Proc. ACM SIGCOMM HotSDN, 2012.
[12] J. McCauley, A. Panda, M. Casado, T. Koponen, S. Shenker, Extending SDN to Large-Scale Networks, In ONS, 2013.