Feedback Control Flow Management in OpenFlow-based Data Center Networks

HAO Shuai
Department of Computer Science
The College of William and Mary
haos@cs.wm.edu

***Abstract*—The emerging area of Software-Defined Networking (SDN) provides an exciting research methodology to network design and evolution. The separation of control-plane and data-plane in SDN enables network devices to process traffic based on the operations defined by control-plane program. The OpenFlow, as a SDN platform, purposes a standardized protocol interface in OpenFlow-enabled switches and controller by implementing a new flow-table design. In this paper, we explore the feasibility of exploiting the statistics information in OpenFlow flow-table and control channel to construct a basic feedback control module in OpenFlow-based Data Center Networks. We present our system analysis and design, and also the preliminary implantation method in OpenFlow platform.**

Keywords—OpenFlow; Feedback; Flow Management; Data Center

I. INTRODUCTION

Today's Data Centers host more and more computing and storage resources to provide public services, such as Cloud Computing. The emerging data center solutions with clusters of commodity servers accelerate service delivery and deployment, improve quality of service, and reduce risk for new business model. However, the interconnections between a large number of servers and high-volume data transmission require a scalable, efficient and robust underlying network design. Also, the structured storage, virtual machine deployment and distributed computing engine (*e.g.MapReduce*) exacerbate the requirement for networking infrastructure in Data Centers. The OpenFlow, as a Software-Defined Networking platform, is receiving wide consideration to explore new architecture design and facilitate test of prototype for Data Center Networking with controllable network infrastructure.

In this paper, we purpose a basic feedback mechanism in OpenFlow-based Data Center Networks without any additional installment of monitoring tools in servers. With the controllable switches, the OpenFlow Data Center has intrinsically facility on feedback control of flow management: 1) the switches have flow state information that can be collected by controller to decide how to adjust switch behaviors; 2) the control channel provided by OpenFlow protocol can be used to collect/send feedback and control information. This preliminary feedback-based flow management can be used to implement performance improvement in Data Center, such as overall throughput, QoS and load-balance.

The rest of this report is organized as follows. In Section 2, we summarize some relevant features of OpenFlow and related work on design and management for data center networking. In Section 3, we present the algorithm and system design on feedback control mechanism. In Section 4, we propose the pre-prototype design with case study. We conclude this paper and discuss future work in Section 5.

II. BACKGROUND AND RELATED WORK

A. Background

OpenFlow [1][2] defines a standardized interface to provide an interaction channel between switches and controller. The controller program installs or removes *flow-entries* in the *Flow-Table* of OpenFlow switches. The flow-entry is defined as a triple <*Rules*, *Actions*, *Statistics*>, shown in Fig.1:

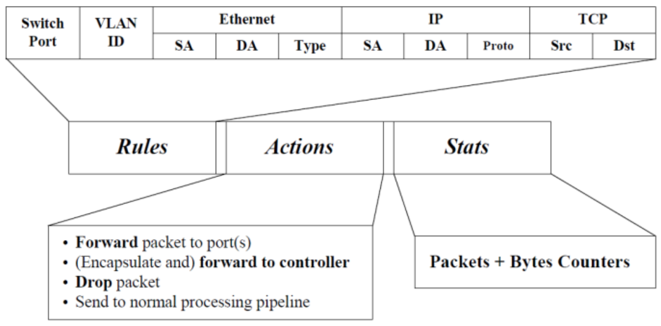

Figure 1. Rules matching and flow process in OpenFlow flow-table

- *Rules* match certain packet-head fields to determine specific flows. In this context, with the *wildcard* in some fields, the *flows* can be broadly defined, such as TCP connection, VLAN ID, IP source and destination address or MAC address.
- *Actions* field specifies the methods of processing each flow's packets: 1) *forward* this flow's packets to given ports, which allows the packets to be routed through network; 2) encapsulate and *forward the flow's packets*

This paper is a report for course project of CSci634: Advanced Computer Networking in Fall 2011, instructed by Dr. Gang ZHOU.

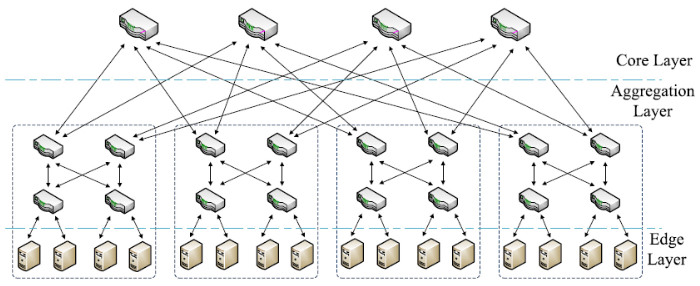

Figure 2. A Fat-tree topology for Data Center Networking

to controller by a Secure Channel, where the controller will decide how to process packets and then install a new rule in switch for this flow; 3) drop the packets, which can be used to prevent from malicious traffic and redirect workload. The OpenFlow also defines an additional action to enable normal switch processing pipeline.

- *Statistics* field records the number of packets and bytes for each flow, and maintains a timeout that triggers the switch to remove inactive flows.

In OpenFlow networks, the network infrastructures are operated by the separating controller program [3] [4]. The controller installs and removes flow-entries from the flow-table to control network behalf. In our solution, the controller program collects the network status information, makes the decision and sends feedback control to switches.

B. *Related Work*

Many works on design, management and optimization for data center networking are proposed recent years [5] [6] [7]. The fat-tree structure [8] (shown in Fig.2) implements a simple and scalable architecture design for Data Center Networks. A framework for understanding and evaluating data center traffic characteristics is presented in [9]. The policy-aware switching layer design [10] facilitates the middleboxes deployment in Data Centers. The DFIANE [11] proposes an effective and scalable flow-based networking architecture with OpenFlow. Also, there are many works that focus on the load-balance in Data Centers [12] [13] [14] [15] [16].

A control-theoretic analysis to flow control was presented in [17]. The DFC [18] proposes a general dynamic feedback control model on Data Center Networks, but based on the status information of each server.

III. SYSTEM DESIGN

A general feedback control system consists of three main components: *Sensor*, *Controller* and *Actuator*:

- *Sensor*. In general feedback control system, the sensor is used to sample the output of the controlled system, and transfer sampling results to the Controller. In our data center solution, we don't need to install any sensor (monitoring program) at switch or server because the flow-table can provide status information.

- *Controller*. Controller receives sensor data and makes the decision to adjust the behavior of controlled system. In OpenFlow platform, a controller adds and removes flow-entries from flow-table.

- *Actuator*. Actuator produces the signal to adjust the controlled system. In our solution, a new flow-entry that adjusts switch behavior can be regarded as the actuator signal.

A. *System Overview*

Instead of traditional data center with high-performance servers, today's data centers are constructed by the commodity computers that are interconnected by the underlying switching networks to gain scalability and better price-performance trade-off. Therefore, to provide the stable services, a dynamic and prompt response mechanism that adjusts network configuration to guarantee reliability is required.

The flow-granularity status information and the distributed controllable switches with one (or more) centralized controller in OpenFlow network enlighten us to design a feedback-based flow control module to improve the overall performance of data center network:

- The flow state information can be used by controller to decide how to adjust switch behaviors by estimating the throughput.

- The secure channel provided by OpenFlow protocol can be directly used as feedback control channel by warping standard OpenFlow operation.

- The controller can administrate the switch behaviors by installing and updating the flow rules on flow-table of each switch.

In our solution for feedback control module, we collect server status information from the *statistics* field in the flow-table of each network device and construct a global flow state information table. To simplify the system architecture, we construct global flow state information by distinguishing the flows based on IP destination address. This is a reasonable assumption because the switches in layer-3 switching network forward the packet based on server IP address Also, we assume all general commodity servers deployed in data center have similar processing ability and performance.

With the packet statistics state from each flow-entry, the controller can estimate current throughput for each switch and server:

- By aggregating the statistics information from single switch, the controller can calculate the throughput for that switch.

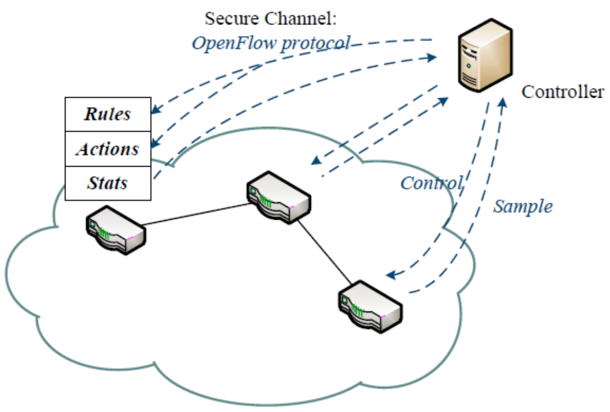

Figure 3. The framework of feedback control module for OpenFlow data center network

- By aggregating the flow-entries that have the same IP destination address, the controller can roughly estimate current workload for certain server which is assigned that IP address.

The framework of feedback control is shown on Fig.3. Next we will present the simple adaptive algorithms for adjusting switch traffic and rebalancing server load respectively.

B. *Switch Throughput*

The throughput for single switch can be represented by its flow-table statistics. The i^{th} flow entry in flow-table of switch s has bytes counter $n_i(s)$, then we estimate current throughput for switch s as:

$$T(s) = \sum_i \frac{n_i(s)}{b_i(s)} \tag{1}$$

where the $b_i(s)$ is the duration of i^{th} flow entry.

The controller maintains a global state table for all network devices, and periodically requests the complete flow-table from them (or all devices periodically send their flow-table to controller).

- In the case that the controller needs to install a new flow entry to network at certain period, it can choose a device with lowest estimated throughput $T(s)$ from available switches.
- Consider the popular Fat-tree topology from Fig.2. If a switch requests to establish a new flow entry, the controller can decide to install this flow to another switch in the same block with lower throughput to keep the data path.

Also, now we assume that there is a throughput threshold $\rho(r)$ for switch r. For deciding whether the throughput $T(r)$ for switch r is overloaded, the controller can simply compare the $\rho(r)$ with current $T(r)$:

$$\alpha(r) = T(r) - \rho(r) \tag{2}$$

If $\alpha(r)$ is greater than zero, the controller can consider the device is overloaded, and then move some flow entries from switch r to other switch. For the Fat-tree topology data center network, we only consider the flow moving at aggregation layer blocks because 1) the core-layer switches have higher performance; 2) the flow change may affect the data path on data center. Also, we don't consider the situation that all the network devices are overloaded; this case is often caused by the mis-configurations.

Obviously, the controller hopes to reduce the switch throughput as efficiently as possible with once flow moving. Thus, the controller moves the flow entry with the highest throughput t_i:

$$t_i = \frac{n_i}{d_i} \tag{3}$$

If the value of $T(r)$ - t_i is less than the throughput threshold $\rho(r)$, the controller moves this flow i from one switch r to another one with lowest throughput T. If the $\alpha(r)$ is still greater than zero, the controller moves the next highest t_j from r until $\alpha(r) < 0$.

C. *Server Workload*

As mentioned previously, an aggregation of flow entries with the same destination IP address from different switches can represent a rough estimate for server workload because the flow aggregation indicates the throughput that flows into the specific server.

1) *Flow-granularity solution*

First we present the simplest solution for the server load-balance with flow-granularity. The workload of a server can be defined as a combination of CPU and memory utilization, and the usage rate of other resources. However, for the online servers, in most cases the workload can be effectively represented by the number of connection requests, e.g. the TCP connections. Thus, it is also reasonable that the controller considers the number of incoming flows as the server workload in our data center networks.

- From the global flow state table, the controller can check the number of flow entries connected to each server.
- The controller can assign the new incoming connection requests or computing tasks to the server with the lowest number of connections (flow-entries).
- With a load threshold λ, which is defined as the certain number of flows for specific server, the controller can determine the server overload and move some flows to other switches.

2) *Byte-granularity solution*

The method based on the number of incoming flows provides a preliminary solution for server load-balance. Here we propose a *flow statistics-based* solution which is rather similar with the switch throughput. With the same definition from switch throughput, the estimated workload *W* for server *u* can be defined as:

$$W(u) = \sum \frac{n_d(u)}{b_d(u)} \tag{4}$$

where the *d* is a specific IP address, and the n_d and b_d are the bytes counter and flow duration for the flow entries in flow-tables of all switches. Then the controller can assign the new incoming connection requests or computing tasks to the servers with the lowest *W*.

With a load threshold of $W(u)$, the controller can determine the server overload and move the flows to other switches based on the simple flow throughput, defined by n_d / b_d.

D. *Discussion*

Basically, the balancing mechanisms for the switches and servers can be performed respectively. However, in some cases, the feedback control outputs for switch throughput and server workload may conflict: the adjustment of flow entries for balancing the server workload may increase the throughput of specific switch. We don't consider this situation as a serious problem because the reconfiguration of flow entries to balance switch throughput will never change the workload on servers. Thus, it doesn't generate a flow-moving oscillation among the switches.

IV. PROTOTYPE

In this section we present a basic framework of prototype construction for our feedback control module in OpenFlow-based data center network.

A. *Global state table*

To implement the feedback control, the controller needs to aggregate the flow statistics information for each switch and for each server, as shown in Fig. 4:

- For each flow-table from specific switch, we simply add an attribute which denotes the overall throughput, calculating by the formula (1).
- We construct an index table for each server in data center network. The items in this table include the position of each flow entry (which switch this entry is installed on) and the single flow throughput defined by formula (3). Then the controller can locate the entry when it moves the specific flow. Also a server's workload defined by formula (4) is associated on each server's index table.

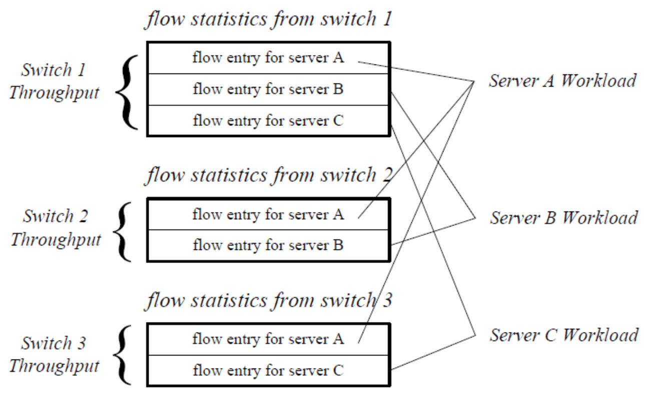

Figure 4. Global flow state information

B. *Control channel*

As mentioned previously, the secure channel provided by OpenFlow can be directly used as feedback control channel. In OpenFlow architecture, the controller is in change for adding and updating the flow-entries from the Flow Table on behalf of network.

In previous discussion, we assume that the controller can periodically request the complete flow-table from switches, or all switches periodically send their flow-table to controller. The OpenFlow protocol defines the rich interaction interface between the switches and controller, and we only use the basic operations for reading switch state and modifying flow-table. By wrapping these functions with a periodical information collection interface, the control channel can be defined within several lines script code.

C. *Controller*

The balancing algorithm will be implemented in Controller. The program obtains the flow-table data from the interface of control channel, constructs the global flow state information, makes the decision for the flow moving based on the switch throughput and the server workload, and returns the control message to the channel.

D. *Extendsions*

Finally, we present some possible function extension for our solution.

- With a slightly modified flow-table function, a switch can monitor its own throughput state only by the flow-table statistics. Thus, it can actively notify the overload to controller and request a flow moving.
- Another consideration is whether the controller will recalculate the global state table when it makes a flow adjustment. Here we consider that this issue revolves the length of period of the state information collection. If the period is short, the recalculation is unnecessary.

If the period is relatively long, recalculating the global state is beneficial for the performance stability of data center. Thus, an adaptive scheme for the period of information collection may be designed. After an occurrence of flow moving, the controller checks the flow statistic within a relatively short time. If there is no flow adjustment at a certain period, the controller can check the global state in a long period.

- This scheme can work together with other server load-balance algorithms. When any monitor program on servers reports an overload state, the high-throughput flow moving will be effective for reducing the server workload.

V. Conclusion

Today's data centers are constructed by exploiting the clusters of commodity servers and robust underlying switching network design. This architecture raises the new challenges for design and management for data center networking. The OpenFlow as a SDN platform purposes a standardized protocol interface between distributed network devices and centralized controller. In this paper, we explore the feasibility of exploiting the statistics information in OpenFlow flow-table and control channel. By defining a simplified measurement mechanism for switch throughput and server workload, we construct a basic feedback control module in OpenFlow data center networks. Also, we present the preliminary prototype construction by wrapping the standard operation of OpenFlow protocol.

Acknowledgment

This paper is a technical report of student-defined course project for *CSci634: Advanced Computer Networking* in Fall 2011, instructed by Prof. ZHOU Gang. I want to thank for the instruction and helpful discussion from Prof. Zhou.

References

[1] OpenFlow Switch Specification. http://www.openflow.org, Feb. 2011.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turn. "OpenFlow: Enabling Innovation in Campus Network," *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, April 2008.

[3] N. Gude, T. Koponen, J. Pettit, M. Casado, N. McKeown, and S.Shenker. NOX: Towards an Operating System for Networks," *ACM SIGCOMM Computer Communications Review*, 38(3), 2008

[4] Martin Casado, Michael J. Freedman, Justin Pettit, Jianying Luo, Nick McKeown, Scott Shenker. "Ethane: Taking Control of the Enterprise," *ACM SIGCOMM*' 07, August 2007, Kyoto, Japan.

[5] Chuanxiong Guo, Haitao Wu, Kun Tan, Lei Shiy, Yongguang Zhang, Songwu Lu, "Dcell: A Scalable and Fault-Tolerant Network Structure for Data Centers." ACM SIGCOMM, 2008, New York, NY, USA

[6] Haitao Wu, Guohan Lu, Dan Li, Chuanxiong Guo, and Yongguang Zhang, "MDCube: A High Performance Network Structure for Modular Data Center Interconnection," *ACM SIGCOMM CoNEXT 2009*, Dec. 2009

[7] Radhika Mysore, Andreas Pamboris, Nathan Farrington, Nelson Huang, Pardis Miri, Sivasankar Radhakrishnan, Vikram Subramanya, and Amin Vahdat. "PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric," *ACM SIGCOMM*, Barcelona, Spain, August 2009.

[8] Mohammad Al-Fares, Loukissas, Vahdat, "A Scalable Commodity Data Center Network Architecture," *ACM SIGCOMM*, 2008, New York, NY, USA

[9] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," *Proc. 1st ACM workshop on Research on enterprise networking*, pp. 65–72, 2009.

[10] Dilip A. Joseph, Arsalam Tavakoli, and Ion Stoica, "A Policy-aware Switching Layer for Data Centers," *ACM SIGCOMM Computer Communication Review*, 38(4):51–62, 2008.

[11] Minlan Yu, Jennifer Rexford, Michael J. Freedman and Jia Wang, "Scalable Flow-Based Networking with DIFANE," *ACM SIGCOMM*, New Delhi, India, August 2010

[12] Richard Wang, Dana Butnariu and Jennifer Rexford. "OpenFlow-Based Server Load Balancing Gone Wild," *Proc. Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services* (*Hot-ICE*), March 2011

[13] N. Handigol, S. Seetharaman, M. Flajslik, N. McKeown and R.Johari. "Plug-n-Serve: Load Balance Web Traffic using OpenFlow," *Demo on ACM SIGCOMM*, Aug 2009.

[14] Hardeep Uppal and Dane Brandon. "OpenFlow Based Load Balancing," University of Washington, *Course Project Report CSE561:Networking*

[15] Markus Kliegl, Jason Lee, Jun Li, Xinchao Zhang, Chuanxiong Guo, and David Rincon, "Generalized DCell Structure for Load-Balanced Data Center Networks", *IEEE INFOCOM*, March 2010

[16] Laiquan Han, Jinkuan Wang and Cuirong Wang, "A Novel Multipath Load Balancing Algorithm in Fat-Tree Data Center," *1st International Conference on Cloud Computing* (*CloudCom*), Dec.2009, Beijing, China

[17] S. Keshav, "A control-theoretic approach to flow control," *ACM SIGCOMM Computer Communication Review*, 25(1):201, 1995.

[18] Baohua Yang, Guodong Li, Yaxuan Qi, Yibo Xue and Jun Li, "DFC: Towards Effective Feedback Flow Management for Datacenters," *9th International Conference on Grid and Cloud Computing*(*GCC*), Nanjing, China, Novermber 2010.