Zero-Footprint Data Center Network Monitoring How to Monitor Your Data Center Networks for Free

[Extended Abstract]

Author One[*]
Institute One
Address One
author.one@emails.com

Author Two
Institute Two
Address Two
author.two@emails.com

Author Three
Institute Three
Address Three
author.three@emails.com

ABSTRACT

(We should give a cool name to refer this system)

This paper present a network monitoring architecture which take advantages of Software Defined Network and traditional monitor methods. Our architecture support the a number of features: 1)monitoring both large and small flows with low cost, 2)adjust the work condition automatically according to the flow status, 3)accept instructions from users and other applications to control the flow, 4)combined with traditional detection methods.

optional: 1)support long-term flow analysis... TODO: make features clearly

General Terms

Application

Keywords

network monitor, openflow

1. INTRODUCTION

Network monitoring system is considered an indispensable part of the network infrastructure. By probing the instant network states and status, various network monitoring tasks provide useful information for network debugging, health check, intrusion detection, and application identification. More important, it is the most critical link for a closed loop control over the networks. For example, security policy enforcement such as intrusion prevention, traffic engineering for optimized network utilization, and access control for quality of services all rely on the information acquired from a network monitoring system. The advent of Software Defined Networking (SDN) offers fresh opportunities to such a closed loop system, in which a logical central controller provides a unique vantage point to oversee the entire network, analyze the data collected, and apply fine-grained controls on routing and policies based on the network states.

Data center is central to modern ICT (Information and Communication Technology) clouds. Within it, network acts as the artery of the infrastructure to provide connectivity for computing and storage nodes. The networkâĂŹs health and performance is therefore critical to the service offered by the data center. However, data centers also face some challenges to support effective and efficient network monitoring.

First, data center is constantly under the pressure of power and cost. Meanwhile, the monitoring tasks are of great quantity and diverse a lot. Specialized and heterogeneous boxes dedicated for network monitoring significantly increase the data center cost and power budget.

Second, data center can support multiple tenants or present complex work load patterns which makes the network data hard to analyze and the network behavior hard to predict. An agile network monitoring system needs to be able to collect and analyze the network traffic, and react in real time. The monitoring must also not negatively interfere with the normal network operation.

Third, data center is evolving into a âĂIJsoftware-defined everythingâĂİ age. The tools involved in network monitoring need to be unified and streamlined with other tools across networking and computing. In order to lower and the operation cost and avoid incurring sheer learning curve, it is ideal to be able to integrate popular and open-source based software tools into the monitoring system with ease.

Last but not the least, data centers grow at a very fast pace. The monitoring system needs to be stable and scalable, and have a clear growth path as well. It would be counterproductive and even detrimental to the business if the monitoring subsystem upgrades always require a system-wide overhaul.

Traditionally, there are several different ways to handle the network monitoring tasks within data centers.

- Using standard or proprietary protocols such as SNMP[1], NetFlow[6], and sFlow[7]. Most of the existing network switches have built-in support for some popular network monitoring and management protocols. They are useful but not general enough to meet all monitoring requirements. For example, they are not accurate enough for many measurement tasks due to the randomly sampling nature. They are also unreliable for most of the security-related monitoring tasks since most of the packets are basically invisible to the monitoring system.

[*]sample titlenote

- Tapping network and using specialized hardware-based fabric to deliver the traffic to different tools for analyzing (e.g. packet broker[?], BigTap[?], Gigamon[?]). The so-called NPM (Network Performance Monitoring) and NPB (Network Packet Broker) systems are purposely built and can offer very high performance. The monitoring subsystem often operates in a separate shadow network but it does need the network switches to provide dedicated TAP (Test Access Point) or SPAN (Switch Port Analyzer) ports. What can be seen is limited by the switch capability. The extra hardware also incurs high capital and operating cost.
- Using specialized hardware boxes either sits in line (e.g. hardware firewall[?] and ADC appliances[?, ?]) or replacing the commodity off-the-shelf switches with more powerful and sophisticated ones with enhanced processing power for in-device monitoring (e.g. Servone[?] and Pluribus[?]). While no shadow network is needed, this one asks for new hardware and is fundamentally a proprietary network monitoring solution.

Instead of following old suit, our approach is drastically different. We provides a general DCN monitoring framework which overcomes the drawbacks of the previous approaches. The resulting monitoring system is non-intrusive, elastic, and scalable. It requires no extra investment on hardware and allows both open source and proprietary software to be seamlessly integrated together. Our approach is motivated by several recent trends in networking domain.

- SDN and OpenFlow[4]: SDN advocates open programmability. It allows new network applications to be written in software and deployed into the forwarding plane devices through an open interface. This is unthinkable before for the vertically integrated network devices. ItâĂŹs believed that Data Center will be the first place to apply SDN technologies. As a de factor south bound interface standard, OpenFlow can configure and control the switchesâĂŹ behavior and pull their status and statistics at arbitrary flow level. The multi-table feature provides us a convenient way to embed monitoringrelated functions and features without interfering with the normal packet forwarding functions.
- OCP and white-box switches[?]: Open Compute Project aims to standardize the data center hardware components. Specifically, the OCP networking project promotes the white-box switches with open operating system and network stacks. These switches are widely available, low cost, and programmable. It gives the network operators great flexibility to customize the switch functions. When an OpenFlow agent is installed, these switches are easily converted into SDN switches.
- NFV[2]: First initiated by telecommunication service providers, Network Function Virtualization tries to realize various network functions using commodity servers and IT vitualization technologies to replace proprietary network devices. The significant performance boost of servers and their cost advantage make this idea plausible. By doing so, network operators can avoid vendor lock-in, roll out new services more quickly, and scale services smoothly. The proposal has created considerable traction in industry. The specialized middle-boxes such as firewall and IPS are among the first group of network gears which are losing ground.

To summary, our approach uses OpenFlow to control white-box switches and realizes NFV-based network monitoring functions by orchestrating the white-box switches and commodity servers.
In a bigger picture, our system can be a monitoring-as-a-service framework in a software-defined data center[?] setup which is responsible for providing various services such as policy enforcement, performance tuning, intrusion detection, fault diagnosis, and other data mining tasks.
(Does it has anything to do with OpenStack?)
Section 2 details the architecture of our system.

2. ARCHITECTURE

As we have emphasized, our data center monitoring system incurs no extra hardware investment and totally relies on the existing infrastructure, such as the surplus computing and networking resources which are in idle otherwise. [We need to gather some data to show that typical data center has enough redundant or idle resources to allow such a system] Therefore, effectively our data center monitoring system has zero footprint. Figure 1 illustrates the high level system overview.

We assume the data center network is built with SDN-enabled white-box switches. The SDN controller uses OpenFlow protocol to control the switches to conduct some monitoring related work in addition to normal packet forwarding. The system can dynamically designate some unused servers (or virtual machines in case a virtualized environment presents) as monitoring nodes. The TOR switch ports

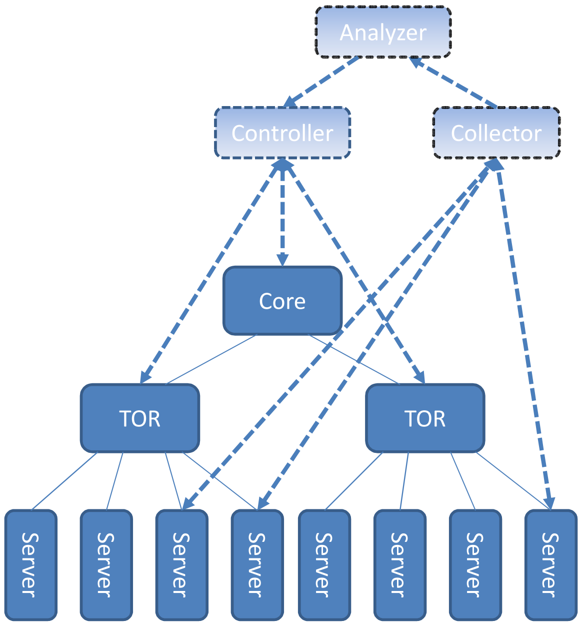

Figure 1: Latency-L2

connecting to these servers are configured as mirror ports which can forward the selected traffic to the monitoring node for further investigation. The traffic filtering criteria for mirror ports are generated by the analyzer and deployed by the SDN controller. The monitoring nodes form a distributed system. Software running on the monitoring nodes performs different functions and returns results to the collector. The collector is responsible for configuring the functions on the monitoring nodes, and collecting and aggregating the results from the monitoring nodes. The aggregated results are then passed to the analyzer to consume. Based on the analysis results, different actions can be taken. For example, alerts can be raised, new rules can be generated, or new functions can be enabled. If new rules are generated, they are passed to the SDN controller through the controllerâĂŹs north bound API. The controller can then reconfigure the switches to modify their behavior. We can see the components form a closed loop system. Note that the dashed boxes in Figure 1 are not specialized devices. They are also implemented in normal data center servers in racks. Each box can represent a single server or a server cluster, depending on the scale of the data center network and the monitoring system. Multiple boxes can also collocate in a single server.

2.1 Switch configuration

Each switch is configured as an OpenFlow switch. It should support OpenFlow 1.2 or newer versions in order to enable multi-table capability. In front of the normal forwarding pipeline, we insert one or more new flow tables dedicated for the monitoring subsystem. In addition to counting the matching packets, each flow entry can be configured to execute a few actions on matching packets including forwarding them to some mirror port or dropping them in place. Unless the packets are configured to be dropped, they will still be submitted to the normal forwarding pipeline. We will show that this simple mechanism allows a wide spectrum of monitoring applications to be efficiently conducted.

2.2 Server configuration

2.3 Traffic collector and analyzer

2.4 Controller and Query language

3. PROTOTYPE AND TEST BED

4. USE CASES

5. EXPERIMENTS AND EVALUATION

6. RELATED WORK

(TBA)Related Work
Sample Citation:
openflow: [4]
B4: [3]
csamp: [5]
SIGCOMM14 switch mirroring Minlan YuâĂŹs group Diagnosis as a Service[?] The structure of the rest of this paper.

7. OUR METHOD

(TBA)Brief summary of related work. Discuss the advantages and disadvantages of Sample, Mirror and so on.
(TBA, motivation)Now we consider the cost when we try to get the network flow information. If we use openflow table to monitor an point-to-point flow, the cost is one flow entry. If we mirror a flow to a server, the cost is the bandwidth. In our system, we use openflow entry to monitor large flow, and mirror small flows to the server.
The rest of the sections is arranged as follows. The system architecture part described the hardware architecture and flow table design, and why we build our system like this. The flow bandwidth monitor describes the algorithm and feedback control on filtering large flows and monitor small flows. The ddos attack detection describe how to monitor long-term ddos attack as well as short-term attack. The deep packetage information detection describe how we detect the package content by using other tools. And the operation for uses command part describe how can we follow users' instructions.

7.1 System Architecture

Hardware Structure

(image)
user –> controller
controller <–> switch
switch —> servers
switch –> mirror server
switch –> snort node(third-part tools)
mirror server –> controller
snort node –> controller
(add port information, and link to controller is inband)

Briefly introduce the structure of different part and their relationships.
Briefly intrudoce the detailed structure of the controller.
Flow table Struction:
(image)
flow table 1:
for high level monitor (user different priority)
flow table 0:
for determinate flow monitor while can make routing decision by oneself (high priority)
fow learning switch(low priority), inlucding the lowest priority packet-in

(TBA)Explain why use the high level part to monitor and routing, but not add another table. (Due to the hardware implentation of multi-table on Pica8...–Can we write like this?–)
(TBA)Explain why use table one for high level monitor. (It can't make routing decisions by one self. eg. all flow come from a given IP.)

7.2 Flow Bandwidth Monitoring

Bandwidth Detection:
Use flow entris to filter elephant flow.
Use feedback to adjust monitoring entries. If the mirror bandwidth extend the port bandwidth, times the factor for feedback result

Flow Entry Replacement Algorithm
All use generated flow matrix to test. Easy to reappear.

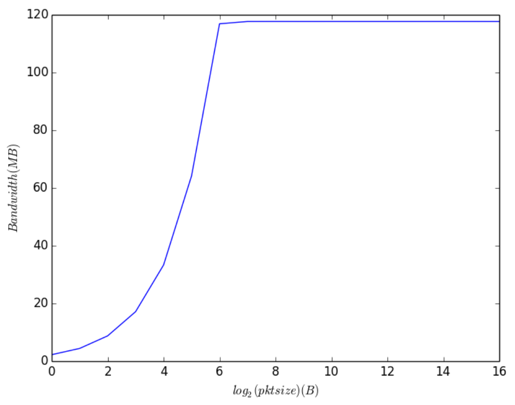

Figure 3: Bandwidth-L2

Random vs Greedy vs Damping using long-term history and dft
compare target: mirror bandwidth, monitor precision, entry change time and so on.
Use dft to detect whether the flow is mutable or stable. Combine it with the average flow size to determine whether to add the corresponding monitor entry.
Store long-term information also help to ddos detection.

7.3 DDos attack detection

Store all the flow statistics information of a period time. For each src node and dst node, use the relavent flows to determine whether it is a potential attacker or a target.

7.4 Deep Package Information Detection

The shortage of our SDN monitor is we can't look up into the packets on the switch. Even we can look into the information in a packet when packet-in event happens, we can't affort the cost of packet-in. A general controller could only deal with 1k-10k packet-in request, which we will test later.
snort

7.5 Operation for User Command

Monitor Flows, Block Ports
Change work status.

8. EVALUATION

8.1 Environment Setup

Hardware parameters introduction. Pica8-3295 and server models
Real System connection struction.

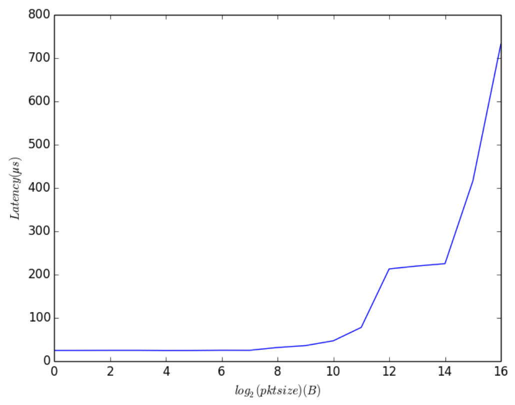

Figure 2: Latency-L2

(Test is based on the features our method support, and including performance test.)

8.2 Basic Environment Test

Test on Switch and Controller:
Basic Test.
Use cbench or other tools. Test on standard learning switch between our switch.
(optional)Test the basic parameters of our system.

8.3 Compared with L2

Compare the Bandwidth and Latency under L2 and Openflow:
use scripts (got help from liyiran)
Test bandwidth between servers under L2 model and Openflow model. If these's little difference, change it to the test between L2 and openflow with our controller
(image)
Test bandwidth between servers under Openflow model with insert/delete monitor flows on different frequency.
(image)

Test the Bandwidth and Latency influence of inserting and deleting monitor flows.
Also test whether it lead to packet-loss when insert and delete flow entries.

8.4 Precision

Bandwidth(Elephant FLow):
Compared with sFlow.
Use generated flows, Compare the precision.
When bandwidth is not full, when mirror port is full.
absolute precision, precision compared with sflow.

DDoss:
Compared with? absolute precision long-term feature react time

8.5 Reaction Time

The attack start time and detected time DPI user's command

8.6 Cost, Cost includes flow entry, mirror bandwidth

Use generated flows
Use real cluster flows(eg. Spark). Depends on the flow characteristc.

9. CONCLUSION

Conclusion

10. REFERENCES

[1] J. Case, M. Fedor, M. Schoffstall, and C. Davin. A simple network management protocol (snmp), 1989.

[2] R. Guerzoni et al. Network functions virtualisation: an introduction, benefits, enablers, challenges and call for action, introductory white paper. In *SDN and OpenFlow World Congress*, 2012.

[3] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, et al. B4: Experience with a globally-deployed software defined wan. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pages 3–14. ACM, 2013.

[4] 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*, 38(2):69–74, 2008.

[5] V. Sekar, M. K. Reiter, W. Willinger, H. Zhang, R. R. Kompella, and D. G. Andersen. csamp: A system for network-wide flow monitoring. In *NSDI*, volume 8, pages 233–246, 2008.

[6] R. Sommer and A. Feldmann. Netflow: Information loss or win? In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 173–174. ACM, 2002.

[7] M. Wang, B. Li, and Z. Li. sflow: Towards resource-efficient and agile service federation in service overlay networks. In *Distributed Computing Systems, 2004. Proceedings. 24th International Conference on*, pages 628–635. IEEE, 2004.