The SOFTmon Network for Software-Defined Networks

K.Chaitanya [1], Ch.Ambedkar [2],
Assistant Professor [1],[2],
Department of CSE, SRK INSTITUTE OF TECHNOLOGY ENIKEPADU VIJAYAWADA
Mail Id : Vishnu kilaru.chaitanya84@gmail.com, Mail id : rahul59985@gmail.com,

ABSTRACT

The foundational enabler for network virtualization is the software-defined networking (SDN) substrate. They provide many opportunities but need novel approaches to addressing established, long-standing problems. Accordingly, in this research, we provide a novel network monitoring tool that is compatible with the standard, commercially-available Open Flow controllers. The provided instrument offers usage charts and data up to a flow level, hence extending the controller monitoring capabilities. The tools' architecture and implementation will be presented with the feature set.

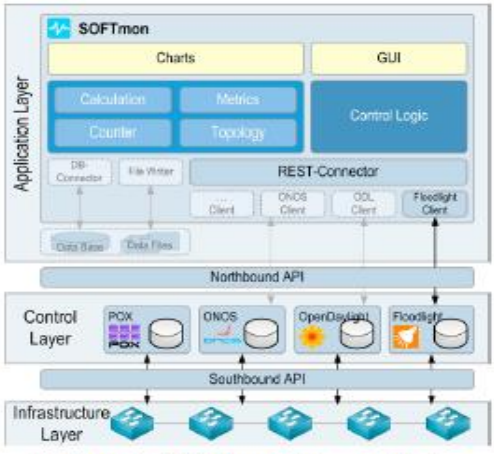
1. Introduction

The monitoring of computer networks has been an essential part of performance management ever since their inception. To assess the state of a network, it is required to isolate critical status parameters. Monitoring a network entails keeping close tabs on a number of metrics, such as the percentage of available bandwidth and the delay between individual nodes. In addition, node-based characteristics, like as connection failures and packet losses, are critical in determining whether or not the network is operating normally. Avoiding congestion and finding architectural bottlenecks are both made easier with network monitoring. Physical problems, such as severed cables or down computing and network nodes, may also be detected and fixed rapidly. These instances highlight why network monitoring is crucial. The complexity and size of modern networks need constant monitoring. In particular, it is crucial to have a cutting-edge monitoring solution due to the increasing complexity of data centers and networks generally. Moreover, virtualization based on CC and the SDN paradigm is constantly redefining the difficulties of network monitoring. Consequently, we'd like to provide SOFTmon, our free and open-source SDN monitoring application. With the help of this utility, you may get an advanced monitoring solution that works with any Network Operating System (NOS). As a result, it enhances the capabilities of traditional NOS-based monitoring by adding graphical transmission charts that provide a wide range of usage data at the switch, port, and flow levels. For example, it enables the possibility of differentiating between many IP-related flows and their burden in the context of the entire network's use and capabilities. This Section of the paper is organized as follows. Some Context is provided and relevant work is described in Section 2. Section 3 describes the structure of the tool, while Section 4 details a prototype implementation. Section 5 provides an assessment, while Section 7 provides a quick summary. Second, context and related studies In general, monitoring is a subject that should not be taken lightly. Multiple mechanisms are often used to keep an eye on a network. Internet Control Message Protocol (ICMP) host-based latency measures and Simple Network Management Protocol (SNMP) network-node-based inquiries are two such examples. These applications, however, need decentralized configuration and testing. As a result, a centralized monitoring server component is essential, such as Zabbix1 or Nagios2. This monitoring server compiles, analyzes, and displays data on a regular basis. Technologies like the Open Flow protocol give and support direct access to the network nodes and other statistics, allowing for the use of these methods in SDN networks as well. Therefore, an SDN-based monitoring system would be more robust and would provide a great deal of additional opportunities to capture network information, such as various flow statistics that can be directly accessed from the flow tables of the switches. Moreover, Open Flow managed switches typically report any network status changes like e.g. a failed link status, instantly to the NOS. They also exchange frequent keep alive messages with the NOS in order to determine the network status as a whole. On the other hand, the NOS are frequently using a mechanism similar to the Link Layer Discovery Protocol (LLDP) to obtain the current network topology and the regarding interconnects. Furthermore, the NOS can be triggered to query the network nodes via the Open Flow protocol in order to obtain the flow tables, flow entries, as well as their counters and statistics. This particular mechanism is utilized by SOFTmon to provide a very fine granular flow-based monitoring solution. There are several open source Open Flow based SDN NOS available. OpenDaylight3 and Floodlight4 for instance, are very common at the moment. As previously mentioned, they generally support basic monitoring

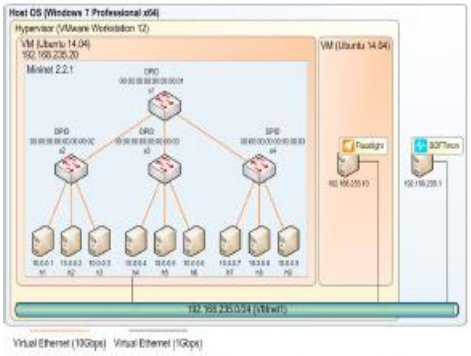
capabilities like the visualization of network topology or the flow statistics in a tabular representation. Nevertheless, the presentation of this information can be evolved, since it is not really human readable neither it provides an appropriate understanding of the current network utilization. Thus, several papers try to address this issue with proposals and approaches for SDN based network monitoring5, 6, and 7. However, almost all of them mainly deal with different measurement approaches and procedures in order to increase the measurement accuracy concerning time. On the other hand, some papers8, 9 present controller module extensions, which are bound to particular NOS and interact directly with the packet forwarding process. Others again, describe just some early work prototypes10, which are not available for testing or downloading. In contrast, the presented SOFTmon tool presented in this paper introduces a method of flow monitoring using the northbound NOS interface. The tool is completely decoupled from other network or software components and acts as an additional utility to observe the network utilization. Furthermore, the prototype implementation including the Floodlight connector is available on GitHub11.

Architecture

SOFTmon's fundamental concept is to provide a NOS-independent traffic monitoring application that delivers supplementary monitoring features and a clear presentation of those features. As a result, SOFTmon employs a technique for traffic measurement that depends only on the switch, port, and flow data established by the Open Flow standard and available through query by any standard NOS. As shown in Figure 1(a) and explained in12, SOFTmon is a business application that operates at the network application layer of the SDN paradigm. It does this by way of the platform's own application programming interface (API), which communicates with the northbound NOS interface.



(a) Conceptual SOFTmon software architecture



(b) Mininet based development environment and topology

Figure 1: Buildings SOFTmon follows the pattern of layered software architecture in its own conceptual design. Database, file I/O, and representational state transfer (REST) capabilities are all part of the data access layer, the foundation of the stack. This layer provides the fundamental capabilities necessary for interaction with the NOS. Northbound API13 that conforms to the REST paradigm is provided by the vast majority of open source NOS implementations, making it possible for network applications to be written in any language. As a result, this interface was chosen to serve as a bridge between SOFTmon and the NOS. Since there is now no common NOS northbound API, REST seems to be the most convenient technique to integrate new NOS connections with different network controllers. In order to describe the methods and data model that must be offered by a certain communication module for the related NOS implementation, SOFTmon's design includes an abstraction layer called REST connector. The data model is part of the next higher level. The NOS statistics are used by the data model to calculate the performance metrics. Once again, the NOS got these details from the network nodes through Open Flow. However, there are primarily three parts to the data model. Let's start with the topology. It consists of all the switches in a network and the wiring between them. It also has the counters that are keeping track of the statistics. Metrics are the last component. In order to see how well a network is doing, they are essential. Open Flow v1.3 provides the foundation for the object model used to describe the network's topology and counter objects. The specific REST client of the data access layer is responsible for providing the functionality necessary to convert the data model retrieved from the NOS non-standardized REST API into the SOFTmons data model. The graphical user interface (GUI) for data visualization and user interaction is located in the uppermost layer. There are tabs where you can choose how you want to measure things, buttons to initiate and terminate the visualization, and a chart component to display the performance metrics in near real time. The SOFTmon team has relied on a simulated testing environment built with the Mininet15 network emulator and the Floodlight4

SDN controller. This development setup consists of two VMs running Ubuntu 14.04 Linux as guests in VMware Workstation, which is installed on a computer running Windows 7 Professional as the host operating system. Mininet simulator is installed on the first virtual machine. Figure 1(b) depicts the tree topology used to organize Mininet, which has a depth of two and a fan-out of three. This method was used often during development to achieve consistent NOS return results. While development took place on a Windows host system using Java and the Eclipse IDE, the NOS itself was encased in a second virtual machine (VM).

4. Implementation

The complete software system may be developed incrementally and modularly thanks to the design outlined in section 3. SOFTmon's monitoring capabilities are, however, limited by the data available via the NOS's Restful interface. It is important to develop the related REST client in order to incorporate a certain NOS. The prototype of SOFTmon is compatible with Floodlight. While the uniform resource identities (URIs) for each every Floodlight REST call16 are described in full, the data model for the JSON structure that is returned is not. As a result, it was necessary to use reverse engineering and probes in order to ascertain the data model. The REST client implementation may end up being the most difficult and time-consuming aspect of future NOS support. The quality of the documentation for the NOS REST API is crucial to this.

Table 1. Indicators and underlying measurements

Type	Metric	Unit	Counter	Unit
Switch Stats.	Flow Count	n	Active Entries (Tables)	n
	Packet Rate	n/s	Received Packets (Flows)	n
	Byte Rate	Byte/s	Received Bytes (Flows)	Byte
Port Stats.	RX Packet Rate	n/s	Received Packets	n
	TX Packet Rate	n/s	Transmitted Packets	n
	RX Byte Rate	Byte/s	Received Bytes	Byte
	TX Byte Rate	Byte/s	Transmitted Bytes	Byte
	RX Port Usage	%	Received Bytes	Byte
	TX Port Usage	%	Transmitted Bytes	Byte
Flow Stats.	Packet Rate	n/s	Received Packets	n
	Byte Rate	Byte/s	Received Bytes	Byte

The Open Flow v1.3 port and flow data are used in the computation of the performance metrics. The computed metrics and their corresponding counters are shown in Table 4. While the NOS are responsible for aggregating data, the values represented by the switch counters are not. Time-dependent performance metrics m (t) may be derived from their associated time-dependent counters c (t):

$$m(t) = \frac{d}{dt}c(t)$$

Counter values are available only in time-discrete form. Thus, the calculation of a metric can be approximated by using the corresponding time interval _*t*:

$$m(t) = \frac{c(t) - c(t - \Delta t)}{\Delta t}$$

The Open Flow specification defines duration counters for port statistics, as well as flow statistics (since Open Flow version 1.3), which could be used as a time base for the time-dependent counter values. This would help to achieve measurements with a theoretical accuracy up to one nanosecond. However, the counter for the nanosecond portion of the duration is marked as optional in the Open Flow specification. Thus, the maximum feasible and guaranteed time resolution is one second. In addition, the port duration counters do not exist in earlier Open Flow versions.

Unfortunately, one second is not sufficient to achieve a fluent visualization in soft real time. The solution for this issue is to create an additional time base by generating and adding a system time stamp to the counter values based on the arrival time of the corresponding JSON object received from the NOS. This is also necessary for adding the functionality of presenting historical values. The resulting error of the time stamp approach will be analyzed in detail in section V.

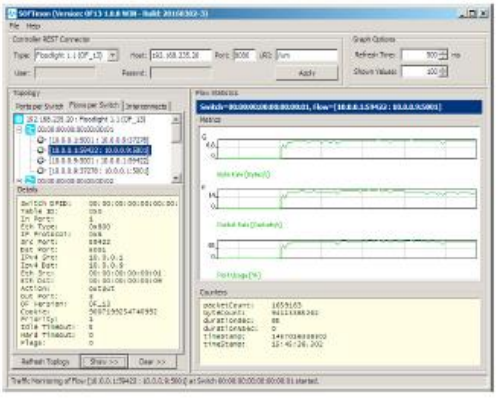


Fig. 2. SOFTmon GUI Figure 2 presents an overview of SOFTmon's graphical user interface on a Windows 7 OS. The parameters and credentials for the REST connection and the regarding NOS can be configured in the upper left area. To the right is another area where the refreshing time and the amount of values for the visualization can be adjusted. The tree view on the left allows choosing different points of measurement dependent on the selected tab. The tree view also reflects the network

topology, while the tabs allow switching the presentation according to the ports per switch, the flows per switch or the switch interconnects. Further details of the selected sample are displayed on the lower left side, whereas the sample is presented as a chart on the right side. The Open Flow based statistics values are monitored in soft real time. In order to measure the network utilization caused by a particular flow, some effort has to be spent in filtering the flow tables of a certain switch and locating the statistic entries of interest. The Floodlight REST interface only allows querying the complete list of all flows in all flow tables of a certain switch at once. This list is arranged by the table ID and the processing sequence of each table regarding the switch's matching process. The current Softmax prototype only supports flow monitoring for the network layer. This means in order to become a selectable item in the SOFtmon GUI, a flow needs to have valid entries in the fields IPv4 source and destination address as well as Ethernet source and destination address. Further, the instruction field must contain a valid action. However, flows are installed and deleted dynamically by the Floodlight's Learning Switch module. Thus, flow list obtained by the NOS and its flow statistics can differ in length and sequence from one measurement cycle to the subsequent one. Therefore, the flow that is selected for monitoring has to be identified in the list through an internal matching process in which the following fields are compared: flow table ID, IPv4 source and destination address, Ethernet source and destination address, Ethernet type, IP protocol, transport protocol source and destination port and physical input port. Not mandatory

Values (e.g. transport protocol ports) are substituted by a wildcard for the search process. In order to not disrupt a flow's utilization visualization metrics when it has been deleted, the statistic values of a missing flow are marked as invalid. This causes the calculation module to return zero as a value. Therefore the graph of the measured metric drops also to zero, but is continued to be drawn until the selected flow is probably active again. The GUI elements that can be selected for monitoring are: switches, switch ports, and flows. They are presented in a tree structure corresponding to the network's topology. Since flows can be installed and deleted within short time frames, this tree structure for selecting the measurement has to be updated manually by the user. The visual presentation of a metric is implemented with the JChart2D library17. It is intended especially for engineering tasks and therefore optimized for the dynamic and precise visualization of data with a minimal configuration overhead. The user can configure the duration of a measurement cycle *dM*, as well as the amount of values displayed in a graph *NM* via the GUI.

5. Evaluation

In order to determine the error that emerges from the proposed and implemented system time stamp approach to label the probes, as described in section IV, the deviation of a switch port metric *m* (*_its)* is measured. This metric is calculated for a time interval *_its* based on the time stamps for the metric *m* (*_tC*), which again is calculated for the time interval *_tC* of the time counters. As shown in table 2, the experimental evaluated and calculated relative deviation of the time interval increases slightly with a decreasing duration of the measurement cycle *dM*. In contrast, the mean deviation of the calculated metric is constantly lower than 0,005 percent.

Table 2. Empirical identified error with time stamp approach

d_M	1000 ms	500 ms	250 ms	50 ms
d_O	27.24 ms	21.60 ms	22.39 ms	27.49 ms
d_R	6.28 ms	6.32 ms	7.11 ms	5.13 ms
	Mean relative Deviation from Δt_C based Values			
Δt_S	-0.002 %	0.001 %	0.004 %	0.110 %
$m(\Delta t_S)$	0.002 %	0.002 %	0.000 %	-0.001 %

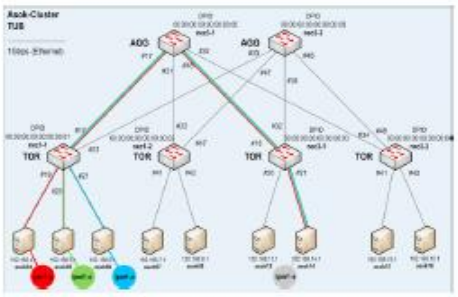
The mean REST call execution time *dR* of the test system results in comparatively constant values between ap-proximately five and seven milliseconds with regard to the measurement cycle *dM*. However, there is an offset *dO* from the instant of time *tS* based on time stamps to the instant of time *Tc* based on time counters. This offset has an averages time of around 25 milliseconds. That means, the metrics that are obtained from the NOS are visualized and displayed around 25 milliseconds later than they actually occur. This is negligible for the applicability as network monitoring tool. In a nutshell, the obtained results demonstrate that even commodity hardware is able to deliver a sufficient sample rate and resolution for the usage of SOFTmon. In addition to the Mininet based development environment presented in section III, SOFTmon was also intensively evaluated on a local SDN research cluster. This cluster is named Asok and has a typical SDN enabled data center fat tree network topology. The SDN network is composed out of dedicated Open Flow switches from NEC. Table 3 lists all components and their hardware and software specifications as used for the cluster based evaluation.

Table 3. Asok Cluster Hardware Configuration

System	Hardware	OS/Firmware
Cluster Node	2 Intel Xeon Quad-Core 2,66 GHz, 32GB RAM	Ubuntu Server 14.04.3 LTS 64 bit
NOS Node	Pentium Dual-Core E5500 2,80GHz, 4GB RAM	Ubuntu Desktop 14.04.3 LTS 64 bit
Monitoring PC	Intel Core i7 2,8 GHz, 8GB RAM	Windows 7 Professional 64 bit
Switches	NEC IP8800/S3640-48T	OS-F3L Ver. 11.1.C.Af

In order to evaluate the monitoring performance with SOFTmon, network traffic was generated using the iperf tool18. Figure 3(a) depicts the evaluation

deployment as well as the iperf server and client configuration. The NOS



(a) Asok Research Cluster - Evaluation configuration

(b) Port traffic on cluster switches

Fig. 3. Evaluation on a SDN cluster and the SOFTmon application are running on dedicated nodes, which are not directly part of the cluster. They are not connected to the SDN data network, but to the separated management network via 1Gbps Ethernet. This network is used for the NOS to switch communication and vice versa. Figure 3(b) shows port traffic probes that were collected with SOFTmon on the cluster. It shows the throughput as byte and packet rate. This particular example was generated with the iperf setup that previously has been introduced and described. The iperf clients were configured to use a to 100Mbit/s limited transmission rate in order to avoid traffic congestion. The graph depicted in figure 3(b), shows the measured and visualized throughput on port 19 of switch nec1-1. This is the incoming (RX) traffic from client asok04, which reaches an averages of 12, 5 MByte/s. This correlates with the configured 100 Mbit/s transmission rate. Moreover, the graph on the right shows the outgoing (TX) throughput of port 18 of switch nec3-1 which is the sum of the iperf traffic of all three clients (asok04 to asok06) that was started successively. The traffic that was limited to 100 Mbit/s per client reaches an average overall amount of 37, 5 MByte/s, which again correlates to the configured 300 Mbit/s transmission rate. For further evaluation of SOFTmon under real traffic conditions, the development environment, as introduced by fig. 1(b), was used for video streaming experiments. The charts that are presented in 4 were collected while retrieving a video live stream with a web browser that was started on a virtual host in the Mininet environment. The curves are showing data bursts which are typical for video streaming.

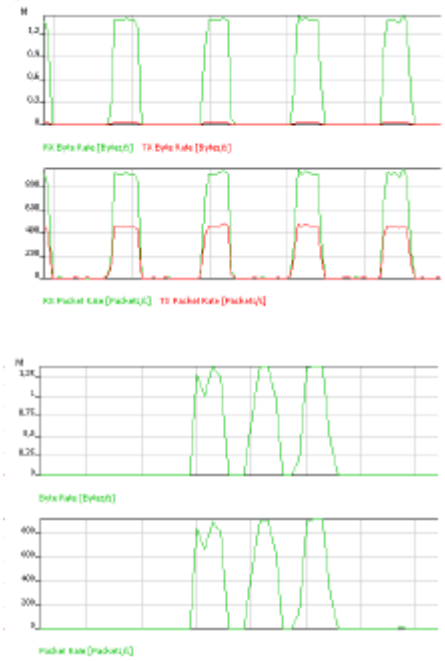


Fig. 4. Port and flow metrics with Youtube traffic evaluated on Mininet All charts that are presented in this paper are screenshots of the current version of the SOFTmon tool. They reflect samples taken during the evaluation and validation process.

6. Disclosure

This work was supported by a fellowship within the FITweltweit programmed of the German Academic Exchange Service (DAAD).

7. Conclusion

This new monitoring tool offers a fresh method for keeping tabs on Open Flow networks. It may be used to identify any form of network activity and expands on the topology-based monitoring capabilities offered by standard NOS. The code for the implementation may be found on GitHub11. The current implementation of the Floodlight REST client might be improved with community contributions, and the software's design is ready to accept them. Because it is written in Java, it can run on any machine. Mininet and Open Flow v1.3 were used to test and verify the functionality of the provided application. Additionally, the standard data center network architecture and Open Flow version 1.0 were used in the evaluation of an SDN research cluster. In addition, a live video streaming was used to test the tool in a realistic network environment. Thus, SOFTmon has already shown that the capital S does not indicate a lack of features or functionality, but rather that it is easy to use. One

of the design constraints was that the tool be very easy to use. A simple, controllable solution like SOFTmon might be extremely beneficial in deploying SDN in productive settings by providing users with a simple, but powerful administrative application, such as Network Operation Control (NOC) operators. Another perk is that SOFTmon may be used independently of the network. The NOC retains primary control over the network, but a local admin, for example, may utilize the tool to investigate a problem.

References

1. Zabbix :: The enterprise-class monitoring solution for everyone. 2016. URL: http://www.zabbix.com.

2. Nagios – the industry standard in it infrastructure monitoring. 2016. URL: https://www.nagios.org.

3. Opendaylight platform. 2016. URL: https://www.opendaylight.org.

4. Project floodlight. 2016. URL: http://www.projectfloodlight.org/floodlight/.

5. Baik, S., Lim, Y., Kim, J., Lee, Y.. Adaptive flow monitoring in sdn architecture. In: Network Operations and Management Symposium (APNOMS), 2015 17th Asia-Pacific. 2015, p. 468–470. doi:10.1109/APNOMS.2015.7275368.

6. Isolani, P.H., Wickboldt, J.A., Both, C.B., Rochol, J., Granville, L.Z.. Interactive monitoring, visualization, and configuration of open flow-based sdn. In: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM). 2015, p. 207–215. doi:10.1109/INM.2015.7140294.

7. Pajin, D., Vuleti, P.V.. Of2nf: Flow monitoring in open flow environment using net flow/infix. In: Network Softwarization (NetSoft), 2015 1st IEEE Conference on. 2015, p. 1–5. doi:10.1109/NETSOFT.2015.7116138.

8. van Adrichem, N.L.M., Doerr, C., Kuipers, F.A.. Opennetmon: Network monitoring in openflow software-defined networks. In: 2014 IEEE Network Operations and Management Symposium (NOMS). 2014, p. 1–8. doi:10.1109/NOMS.2014.6838228.

9. Grover, N., Agarwal, N., Kataoka, K.. liteflow: Lightweight and distributed flow monitoring platform for sdn. In: Network Softwarization (Net Soft), 2015 1st IEEE Conference on. 2015, p. 1–9. doi:10.1109/NETSOFT.2015.7116160.

10. Raumer, D., Schwaighofer, L., Carle, G.. Monsamp: A distributed sdn application for qos monitoring. In: Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on. 2014, p. 961–968. doi:10.15439/2014F175.