Applying Big Data Technologies to Manage QoS in an SDN

Shashwat Jain, Manish Khandelwal, Ashutosh Katkar, and Joseph Nygate
Rochester Institute of Technology, Rochester, NY, USA

***Abstract* - Managing QoS in a telecommunications network is a complex process. Effective network design and sizing in conjunction with load balancing, access control and traffic prioritization need to be orchestrated to optimize CAPEX investment, maximize network utilization and ensure that performance metrics and SLAs are met. This work shows how big data analytics were used to improve the management of QoS in an SDN by performing multi-dimensional analysis of Key Performance Indicators (KPIs) and applying machine learning algorithms to discover new correlations, perform root cause analysis and predict traffic congestion.**

I. INTRODUCTION

Effective QoS management requires accurate and timely collection of large volumes of performance data across the network. KPIs [1, 2] such as delay, round trip time, jitter, dropped packets and utilization need to be analyzed to determine where there are QoS degradations and how they can be best resolved e.g. through network expansion, changing traffic prioritization or applying more restrictive controls.

Telecommunication operators typically use commercial off-the-shelf management systems such as IBM Tivoli, OSI Net Expert or Ericsson/Telcordia to process raw performance data that is collected from the network using interfaces provided by the Element Management Systems (EMS). Challenges to this process include the thousands of Management Information Base (MIB) values that need to be collected from each network element every 15 minutes [3] and sophisticated KPI modelling and analytical requirements such as correlation, root cause analysis and prediction. As these products were developed over 20 years ago they do not provide the scalability and analytical capabilities that modern big data technologies offer.

Previous research on correlation in telecommunication networks focused on fault localization. A comprehensive evaluation of existing methods, including rule based systems [4, 5, 6], codebook analysis [7, 8] and fault propagation [9, 10], was performed in [11]. They concluded that all current approaches are lacking as they cannot support the complexity of modern hierarchical networks and require new logic to be developed for each network. Indeed, to our knowledge, from all the methods analyzed in [11] just three were commercialized [4, 6, 7] and they were only applied in niche areas with limited success. Moreover, none of these systems were used to analyze performance data.

QoS management could be improved if performance data was analyzed and one could learn which KPI(s) impact QoS. For example, if a router had 12 ports and 12 links that were all carrying traffic and we were able to discover that only the traffic volumes on ports 1, 5 and 7 were in a monotonic relationship with the delay on link 6. Moreover, to effectively manage QoS, we would then wish to quantify this relationship and determine by how much the delay would increase as traffic grows. Therefore, if we were able to discover (1), we could then determine how much traffic could be sent on ports 1, 5 and 7 before the delay on link 6 became unacceptable; or if the delay on link 6 increases we could then conclude that port 5 is the probable cause as it has the highest coefficient.

$$\textit{Delay (link6)} = 0.1 * \textit{traffic}(p1) + 2.3 * \textit{traffic}(p5) + 0.2 * \textit{traffic}(p7) \quad (1)$$

In general, what is required, is a method that creates a set of formulae that quantify the relationship between KPIs (e.g. packets received, packet size, queue lengths, etc.) and QoS metrics (e.g. throughput, per link and end-to-end delay, packet loss, etc.). However, none of the methods described in [11] can quantify the impact of performance metrics on QoS nor meet the scalability requirements.

Recent research on correlation in telecommunication has focused on leveraging advancements in big data technologies. In [12], clustering algorithms were used to automatically correlate five classes of faults from a small data set of a few hundreds of alarms. In [13], clustering algorithms were used on a much larger data set of millions of faults, albeit for only 3 pre-defined alarm classes. More extensive use of big data technologies was done in [14] where, instead of analyzing faults, 2.2 billion usage records were data-mined to detect application and cohort clustering.

While these efforts have demonstrated the ability to cluster faults and support very large data sets, big data algorithms have yet to be used to discover the quantitative correlations that are required to manage QoS. Moreover, SDN brings new challenges due to the scalability and elasticity this technology offers and the need to analyze the performance data from both the virtualized network and the under-pinning IT infrastructure.

This work demonstrates how big data technologies were applied to performance data and improved QoS in an SDN by:

- Applying machine learning algorithms to implement multi-dimensional KPI analysis, discover new correlations, perform root cause analysis and predict QoS violations
- Implementing data cleansing techniques to handle incomplete, missing or corrupt data
- Supporting a robust and scalable architecture that collects data from both the virtual and real worlds

We have extended previous work in three directions. First, we applied big data technologies to correlate events in an SDN. Secondly, we correlated performance data as opposed to faults. Thirdly, we implemented a two phase analysis mechanism that first discovers which KPIs are correlated with the QoS and then data mines each KPI's quantitative impact.

978-3-901882-85-2 (c) 2016 IFIP

II. Methodology

For our research, we constructed an SDN simulation and analysis environment based on open off-the-shelf products as shown in Fig. 1.

During each simulation we collected 24 Key Performance Indicators (KPI) from the virtualized network, traffic simulator and IT infrastructure. These KPIs were then analyzed using big data algorithms to discover correlations between KPIs that lead to poor QoS.

A. Functional Architecture

We created an SDN simulation and analysis environment using off-the-shelf products as shown in Fig. 1.

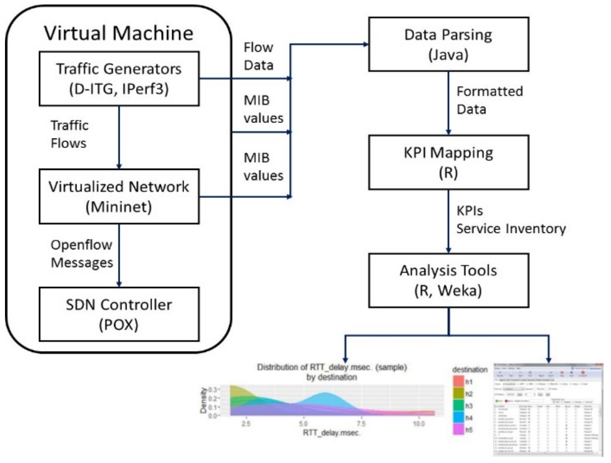

Fig. 1. SDN Research Architecture

Distributed Internet Traffic Generator (D-ITG) [15] - generates IPv4 and IPv6 traffic including TCP, UDP and VoIP. Transaction logs make it possible to capture fine-grained metrics such as bit rate, jitter, packet loss and round trip time.

IPerf3 [16] provides active measurement of IP networks and supports tuning of various parameters related to timing, protocols and buffers. For each test it reports the bandwidth, loss and other parameters.

Mininet [17, 18] is a leading SDN emulator that creates a virtual network of hosts, switches, controllers and links. Mininet enables complex topology testing without the need to wire up a physical network.

POX is an open source SDN controller and is the default controller for Mininet.

During the simulations, we collected performance and resource data from MIBs [19] modelled in the virtual network elements and IT infrastructure, as well as traffic information from D-ITG.

The *data parsing* module, developed in Java, converted the raw MIB values into a CSV format that represented the metrics we wish to analyze.

"*R*" is a leading big data tool for data cleansing, manipulation, mining, calculation and display [20]. The CSV data was cleansed using the following built-in methods [20]:

1. Scaling metrics that ranged between two values to the range 0 to 1 – for example if the POX controller utilization varied between 19% and 23% this was mapped between 0 and 1 to expand the range and make the results easier to compare
2. Mapping metrics to their order of magnitude – for example, delay took on thousands of different values that varied between 0.001 msec and 10 seconds. By mapping the delay to the order of magnitude we were able to discover correlations that would not be detected due to the variability of the raw numbers
3. Cleaning observations that had missing or extreme values due to instrumentation error

B. Measurements

During our simulation, the following 24 KPIs were collected:

- Port KPIs from MIBs in Mininet using SNMP
    - Bytes – sent, received
    - Packets – sent, received
    - Packet loss – incoming, outgoing
- Queue KPIs using commands on the virtual switches: transmission rate, packets/bytes sent, packets dropped
- KPIs from D-ITG
    - Flows – delay, jitter, throughput
    - Packet – rate, size, loss
- Round trip time KPI: calculated by timing messages sent between each flow's source and destination
- IT infrastructure KPIs using MIBs and UNIX utility commands on the virtual machine:
    - CPU – controller, virtual machine
    - RAM – used, buffered, cached

The data we collected from our simulations was first analyzed using Spearman's algorithm to correlate time series data for each KPI. The strength of association was calculated between variables having a monotonic relationship. The values ranged from +1 (positive correlation) to -1 (negative correlation). The correlation coefficient between the KPIs was then used to determine the importance of each KPI during root cause analysis.

We also used a linear regression machine learning algorithm, M5Rules that is pre-integrated with Weka. We chose M5Rules as it combines decision trees and linear regression for prediction. The list of decision rules derived from the decision trees are formulated as linear equations which provide more insight as compared to a simple decision tree model. This mechanism was used to implement the two phased analysis described above to quantify which KPI had the biggest overall impact on the QoS. The rules and correlations from R and Weka could then be utilized to provide recommendations on where and how to improve QoS in a network.

C. *Use Cases*

For our simulations we built the network shown in Fig 2.

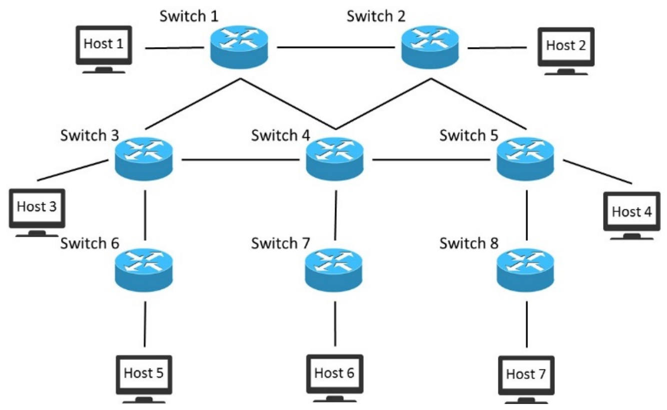

Fig. 2. Network Architecture

The topology followed a hierarchical model of core, distribution and access layer switches. Physical loops were removed by enabling the spanning tree protocol in the POX controller. Queues were installed on ports that had hosts connected to them so that the flows from these hosts could be provided with different Quality of Service (QoS) values. This topology allowed us to simulate multiple flows with varying combinations of clients and servers so that each link could be utilized and there would be considerable changes in the flow metrics and corresponding KPI values. SNMP was installed on the virtual machine and an agent was created for each of the network elements and their ports.

Using this environment, multiple simultaneous unidirectional flows were sent for one hour between 6 sets of hosts. Transmission rates and packet size were varied to model low, average and high network loads by varying the queue configuration and traffic flows as follows:

- Queue minimum transmission (1 and 10 MB/sec)
- Queue maximum transmission rate (4, 8, 12, 16, 40, 60, 80 MB/sec)
- Packet size (512 and 1024 Bytes)

We also ran simulations with simultaneous bi-directional flows using D-ITG and IPerf3 where each host acted both as a client and a serve and the traffic was sent via UDP.

During each simulation the 24 KPIs described above were collected every second.

III. Findings

The simulations and analysis uncovered tens of correlations. A subset of these findings have been divided into 3 logical groups

- *Expected correlations*
- *Discovered correlations* that we were able to explain, i.e. these are new learnings or insights
- *Unexpected correlations* that have not yet been able to explain, i.e. these are areas of future investigation

A. *Expected Correlations*

These findings were important as they demonstrated the stability of our platform. The first example given in Fig. 3 shows that jitter is correlated with delay with a Spearman coefficient of +0.95.

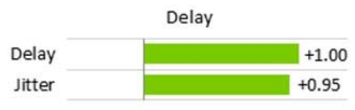

Fig. 3. Jitter/delay correlation

This is quite obvious as if the delay increases, one would expect the jitter to increase as well. Many other expected correlations were found, including:

1. Increasing the packet size, increases the delay
2. Increasing the queue transmission rate, decreases average delay
3. Increasing the queue transmission rate, increases the CPU utilization

While not particularly insightful, they did show that our methodology was sound.

B. *Discovered Correlations*

These correlations provided insights into how the network was performing. For example, Fig. 4 shows that CPU utilization is correlated with the number of transmitted packets in queues 1, 3 and 7. This was because most of the traffic was using the queue and not the entire allocated port's bandwidth.

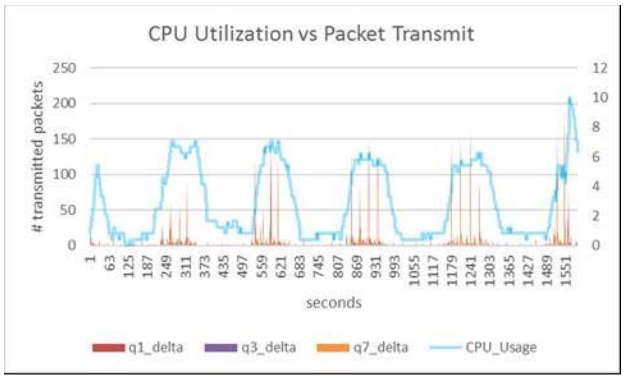

Fig. 4. CPU/Queue Transmitted Packets

Other discovered correlations included that packet loss is correlated with the average delay with a Spearman coefficient of +0.38. This was interesting as it occurred even when the average delay was low.

We also discovered that the VM CPU, some queues and jitter are correlated with the delay between hosts 4 and 5. Using R to analyze this correlation, a decision tree was created that indicating the following:

For 99% of all the traffic between hosts 4 and 5 (2)
if (jitter<0.046) then delay = 724×10^{-6}

Note that this rule does not include the VM CPU as the decision tree algorithm determined that its overall quantitative impact was minimal. We then fed this data into Weka to generate an even more detailed predictive rules shown below:

*If (jitter <= 0) then Delay = 0.0045 * Jitter + 0.0002* (3)
If (jitter <= 0.007) and (jitter > 0.001) then
*delay = -0.0013 * CPU_POX + 0.5131 * Jitter + 0.0162*
*If (jitter <= 0.019) then delay = 0.0001 * q7_delta + 0.04 * Jitter + 0.0073*
*else delay = 0.0026 * CPU Utilization - 0.0052 * q2_delta +*
*0.0047 * q7_delta + 0.5645 * Jitter - 0.0317*

These rules indicate that jitter, q2, q7 and CPU utilization are the key factors in predicting delay and we do not have to consider q1 and q3. That is, although q1 and q3 are correlated with delay, they do not contribute much in absolute values.

Another discovered correlation depicted by the decision tree shown in Fig.5 shows that the order of magnitude of delay and jitter are correlated with the queue length.

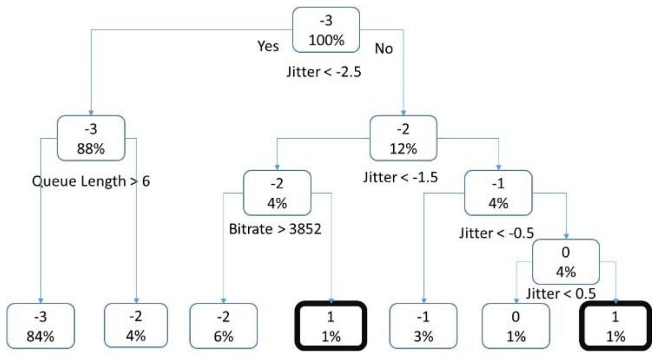

Fig. 5. Order of Magnitude Delay - Decision Tree

The correlation rule derived from the highlighted nodes is:

The order of magnitude of delay = 1 when queue lengths are < 6, bit rate > 3852 and jitter's order of magnitude is < 1 (4)

This observation could not have been found without big data analytics which can yield patterns that are not otherwise obvious.

C. Unexpected Correlations

As mentioned above, high queue transmission rates were indeed discovered to be correlated with delay. However, while this indeed occurred at low transmission rates, we discovered that when the rate was increased from 60 MB/sec to 80 MB/sec the delay actually increased. It is not clear why this was the opposite of findings at lower transmission rates.

Other correlations that we have yet explained were that bidirectional flows increased delay by a factor of 4. While we expected that sending data in both directions simultaneously would increase the delay, we were surprised to see it increase by a factor 4. Another example was the negative correlation between the amount of RAM used on the VM and the amount of cached memory. That is, why an increase in RAM utilization on the VM decrease the amount of cached memory.

IV. Discussion

The use of off-the-shelf products and tools allowed rapid development and automation. Mininet and POX provided a stable and scalable simulation environment. However, collecting time series data from POX was challenging as it required installation of SNMP agents, performing MIB walks and running UNIX utilities. One of the weaknesses of Mininet is that it does not provide any built-in traffic generation tools. D-ITG was chosen as it provided many useful flow metrics and was able to generate over 10 simultaneous flows. However, when queues were configured it could not support more than 3 simultaneous flows. We partially addressed this by running 3 D-ITG flows and the remaining using IPerf3.

Many of the correlations that were found can be applied to improve QoS management. Examples of the derived rules we discovered and how they could be used are:

1. *Root cause analysis* – "if the delay between host 4 and host 5 is high then check the transmission rates for queues 2 and 7" (see Fig. 6)
2. *Qualitative predictions* – "if the number of transmitted queue packets is high, then the VM CPU utilization will increase" (see Fig. 5). Or, "if the number of transmitted packets in queues 2 or 7 are high then the delay will increase" (see Fig. 6 and (3))
3. *Quantitative predictions* – "to predict delay apply (2) (3) and (4)"

Future extensions to this work could include performing the same analysis on other network configurations, collecting additional performance KPIs and analyzing fault data as well.

V. Conclusion

We have presented a systematic approach to improve QoS in an SDN by collecting a large number of metrics from the virtual and physical worlds, and applying machine learning algorithms to automatically discover formulae that quantify the relationship between. We were able to

- Discover correlations – for example, KPI_3, KPI_5 and KPI_{14} are correlated with KQI_6
- Perform root cause analysis – for example, high KQI_7 is probably due to low KPI_2
- Make predictions – for example KQI_3 will be greater than 90 when $KPI_2 + 3*KPI_7 > 12$
- Support trend analysis – for example, given the current traffic growth on host 7, in 3 weeks the delay on link 6 will exceed 50 msec.

As the correlations are network and architecture dependent, rules that are good for one network might not be the best for another. Therefore, if an operator wishes to manage QoS more effectively, they could implement this mechanism in their own operations. The *expected correlations* would probably already part of their QoS best practices; d*iscovered correlations* would be new rules they may like to add, and *unexpected correlations* would provide them with insights how their own specific network is performing and where to investigate more. Once understood, these correlations, too, could be added to their QoS best practices.

We believe that adoption of this approach by service providers will improve the management of QoS and can be applied to legacy, next generation and software defined networks.

REFERENCES

[1] TeleManagement Forum, Technical Report: Managing the Quality of Customer Experience, TR 148 , v0.9, November 2009

[2] TeleManagement Forum, GB923A - Wireless Service Measurement Key Quality Indicators Best Practices, from https://www.tmforum.org/resources/standard/gb923-a-wireless-service-measurement-key-quality-indicators/

[3] Plevyak, T., & Sahin, V. (2010). Next generation telecommunications networks, services, and management. Hoboken, NJ, Wiley

[4] Nygate, Y. A. (1995). Event Correlation using Rule and Object Based Techniques. Integrated Network Management IV, 278-289

[5] Bouloutas, A., Calo, S., & Finkel, A. (1994). Alarm correlation and fault identification in communication networks. IEEE Transactions on Communications IEEE Trans. Commun., 42(2/3/4), 523-533

[6] Jakobson, G., & Weissman, M. (1993). Alarm correlation. IEEE Network, 7(6), 52-59

[7] Kliger, S., Yemini, S., Yemini, Y., Ohsie, D., & Stolfo, S. (1995). A Coding Approach to Event Correlation. Integrated Network Management IV, 266-277

[8] Automating Root-Cause Analysis: EMC Ionix Codebook ... (n.d.)., from http://www.emc.com/collateral/software/white-papers/h5964-automating-root-cause-analysis-wp.pdf

[9] Kätker, S., & Paterok, M. (1997). Fault Isolation and Event Correlation for Integrated Fault Management. Integrated Network Management V, 583-596

[10] Katzela, I., & Schwartz, M. (1995). Schemes for fault identification in communication networks. IEEE/ACM Transactions on Networking, 733–764

[11] Steinder, M. Ł, & Sethi, A. S. (2004). A survey of fault localization techniques in computer networks. Science of Computer Programming, 53(2), 165-194

[12] Kim, D. S., Shinbo, H., & Yokota, H. (2011). An alarm correlation algorithm for network management based on root cause analysis. Paper presented at the 13th International Conference on Advanced Communication Technology, 1233-1238

[13] Man, Y., Chen, Z., Chuan, J., Song, M., Liu, N., & Liu, Y. (2016). The Study of Cross Networks Alarm Correlation Based on Big Data Technology. Human Centered Computing Lecture Notes in Computer Science, 739-745

[14] Jun, L., Tingting, L., Gang, C., Hua, Y., & Zhenming, L. (2013). Mining and modelling the dynamic patterns of service providers in cellular data network based on big data analysis. China Communications China Commun., 10(12), 25-36

[15] Botta, A., Dainotti, A., & Pescapè, A. (2012). A tool for the generation of realistic network workload for emerging networking scenarios, Computer Networks (Elsevier), Volume 56, Issue 15, 3531-3547

[16] IPerf - The network bandwidth measurement tool Active measurements in TCP, UDP and SCTP. (n.d.). from https://iperf.fr/

[17] Jha, R. k., Kharga, P., Bholebawa, I. Z., Satyarthi, S., Anuradha, & Kumari, S. (2014). OpenFlow technology: A journey of simulation tools. International Journal of Computer Network and Information Security, 6(11), 49-55

[18] Lantz, B., Heller, B., & Mckeown, N. (2010). A network in a laptop. Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks - Hotnets '10

[19] Teltumde, P. S., Dr. B. B. Meshram, & Bansod, T. M. (2012). Management of networks using SNMP. International Journal of Engineering Innovations and Research, 1(2), 135-138

[20] R Core Team (2013). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. From http://www.R-project.org/

[21] Hall M., Eibe F., Holmes G., Pfahringer B., Reutemann P. & Witten I. (2009). The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1