Regression-Based Analytics for Response Dynamics of SDN Solutions and Components

Nageswara S. V. Rao, Qiang Liu, and Satyabrata Sen
Computational Sciences and Engineering Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831
{raons.liuq1.sens}@ornl.gov

Rajkumar Kettimuthu and Joshua Boley
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439
{kettimut,jboley}@anl.gov

Bradley W. Settlemyer and Hsing-Bung Chen
High Performance Computing Division
Los Alamos National Laboratory
Los Alamos, NM 87545
{bws,hbchen}@lanl.gov

Dimitrios Katramatos
Computational Science Initiative
Brookhaven National Laboratory
Upton, NY 11973
dkat@bnl.gov

Abstract—We consider Software-Defined Network (SDN) solutions that dynamically setup and teardown connections, and also their response dynamics of connectivity and TCP throughput. There is a wide variety of choices for controllers, switches, and software modules, and objective analytics of measurements are needed to compare the response dynamics of solutions based on their combinations. Throughput and ping measurements during connection changes contain significant statistical variations due to the complexities of controllers, northbound scripts, and network devices. We propose analytical methods to estimate response regression functions, which provide a rigorous and objective comparison of different SDN solutions and their components, namely, controllers, switches, and OpenFlow versions. We apply this approach in two scenarios: (a) switching of long-haul connections emulated in hardware by ANUE devices over a testbed consisting of HP and Cisco switches, and (b) multi-site science infrastructure emulated using VMs and Mininet. The results provide useful practical insights including: (i) the dpctl method responds seconds faster than the OpenDayLight (ODL) method on average under OpenFlow 1.0, but performs similar to ODL under OpenFlow 1.3, and (ii) the ping response of dpctl is about 0.6 sec faster than ODL for the second scenario.

Index Terms—Software-defined networks; OpenFlow; OpenDaylight; controller; long-haul connection; switching response; testbed.

I. INTRODUCTION

Advances in Software-Defined Network (SDN) solutions [3], [10] promise fast and automatic provisioning of connections within seconds for a variety of scenarios, for example, switching of long-haul connections [6] and dedicated data-plane connections for scientific environments [7]. Individual SDN solutions composed of switches, together with controllers and their corresponding custom northbound codes, can be quite complex and varied. In particular, there is a wide spectrum of choices for controllers, including custom dpctl scripts [6], OpenDayLight (ODL) [4], Floodlight [2], ONOS [5], and others. Also, there is a wide variety of network devices, including switches with native OpenFlow implementations and others with OpenFlow implemented using software enhancement modules. Furthermore, the quick update cycles of software tools used in the controllers, switches, and northbound modules lead to rapidly changing SDN solutions. As a consequence, it is becoming extremely challenging to assess the performance of SDN solutions based on these compositions, and hence there is a critical need to develop objective analytics based on measurements to compare and select effective options.

We consider the response dynamics of connectivity and TCP throughput in two different scenarios, in which SDN solutions have been developed to dynamically setup and teardown connections:

A Long-Haul Connection Switching: Two sites are connected over a long-haul dedicated connection, which needs to be switched to a backup connection upon degradation, as shown in Fig. 1. Detection of path degradations and fail-over tasks are performed by software modules at the sites without relying on the connections between them [6].

B Multi-Site Science Infrastructure: Executing scientific flows using multi-site infrastructures requires dynamic provisioning of high-performance data-plane connections between the sites. Unlike the Scenario A, the sites are connected via a separate persistent shared network, which can be used as a control-plane to coordinate the provisioning of data-plane connections [7].

In both scenarios, it is not practical to install and test various SDN components (e.g., a new switch or controller) on production deployments due to cost, availability, and security considerations. Hence, initial solutions are developed using testbed and emulation environments with a goal of transitioning them upon maturity to physical systems. However, SDN solutions in these two scenarios are developed using different approaches. For Scenario A, hardware-emulated testbed connections between pairs of host and controller workstations are utilized [6], whereas for Scenario B, a collection of Virtual Machines (VMs) are utilized to emulate sites consisting of
hosts and Mininet emulations of site networks [7]. Under these test environments, multiple SDN solutions can be implemented and tested, and then appropriate SDN controllers and northbound scripts are selected for transparently transitioning to deployments.

To assess the responsiveness of various SDN solutions in setting up and tearing down connections, we analyze their performances at the application level, in particular, via ping and TCP throughput measurements. In the ideal case, a connection setup request would receive an immediate ping return, and the TCP throughput would become the maximum instantaneously. However, in practice, we observe that the connectivity and throughput measurements during the connection changes lag the ideal responses and show significant statistical variations due to the complexities of controllers, northbound scripts, and switches; therefore, they are typically not amenable to any simple and direct comparisons.

In this paper, to objectively compare the performances of SDN solutions, we propose regression-based analytics methods that estimate response regression functions based on measurements collected in response to custom-designed periodic connection changes. For Scenario A, these connection changes are triggered by path degradations, and for Scenario B, they are triggered by user requests. The regression response method not only enables us to compare the performance of SDN solutions, but also allows us to quantify them and to choose suitable candidates for a given infrastructure. While our discussion is centered around ping and TCP measurements, the overall approach is generically applicable to other parameters for comparing SDN solutions with different controllers and switches, and it is particularly useful when the measurement traces are not amenable to simple comparisons.

Additionally, for both the scenarios, we utilize the underlying monotonicity properties, namely, increasing and decreasing trends of ping and TCP throughput measurements with the round-trip time (RTT), respectively, to provide confidence estimates for the regression fits. Thus, we present a unified treatment of the underlying estimation process that has been developed separately for Scenario A [6] and Scenario B [7]. In particular, we show that our response regression estimate is a close approximation to the ideal regression with a confidence level that increases with the number of measurements. The confidence bound is distribution-free in that it does not depend on the complex joint distributions of components of SDN solutions, which in general are very hard to estimate accurately. More generally, we show the applicability of machine learning methods based on Vapnik’s theory to the class of problems involving response regression functions. These results provide useful practical insights: (i) for Scenario A, the dpctl method responds faster than the ODL method under OpenFlow 1.0 but performs similarly to ODL under OpenFlow 1.3, and (ii) for Scenario B, the ping response of the dpctl method is about 0.6 sec faster than the ODL method in setting up the paths.

The organization of this paper is as follows. In Section II, we present a general statistical method for estimating response regression functions and its confidence bounds. The measurement analysis and inferences are described in Section III. The paper concludes in Section IV.

II. REGRESSION-BASED ESTIMATION

We characterize the performance of an SDN solution by its ping and TCP throughput responses to periodic “events” that trigger flow manipulations on the switches, in particular, insertion and/or deletion of flow entries. For Scenario A, each event is triggered by degrading the current connection, which is detected by the application daemons at the end sites, as shown in Fig. 1(b). In response, northbound scripts are initiated to switch to the backup connection. The effectiveness of the fail-over process is measured by monitoring the throughput of an ongoing TCP stream. TCP throughput becomes low as the path degrades and is then restored over the backup connection; the width of the “pulse” of throughput loss is an indicator of the responsiveness of an SDN solution (the ideal response is an impulse function). This response in particular captures the detection of degradation by the application monitor, controller response to flow-modifying northbound
messages, and switches’ response to the OpenFlow commands from the controllers. For Scenario B, flow modifications are made in response to user requests to teardown and setup the connections, which are typically separated compared to Scenario A. These two scenarios have common dynamics characteristics, namely, the overall increase and decrease of measurements in response to connection changes, which we capture using the response regression functions.

In general, each event engages: (i) software components including SDN controllers, northbound scripts, and site daemons, and (ii) hardware components including switches, connections, and host systems. We propose the response regression method that utilizes averages from ping or TCP throughput traces collected under a switching sequence, wherein the connection is periodically switched at the beginning of a fixed interval period \( T \), and after duration \( T_D \leq T \), which corresponds to Scenario B. The switching sequence corresponds to periodic path setup and teardown events as shown in Fig. 2(a), along with the expected ping and iperf responses shown in Fig. 2(b) and 2(c) respectively. The Scenario A is the special case with \( T_D = T \) such that path restoration is expected to immediately follow the degradation represented by path teardown. In both scenarios, the ping response is fairly fast, resulting in rectangular-shaped responses illustrated in Fig. 2(b). On the other hand, TCP throughput reacts somewhat slower to connection changes as it recovers, leading to trapezoidal-shaped responses abstracted in Fig. 2(c), and more complicated time traces shown in Fig. 3 corresponding to a sequence of path degradations under Scenario A.

Both TCP throughput and ping estimates are overall monotone functions of time around the switching events, although they vary in opposite directions. We utilize this underlying monotonicity property to apply the statistical methods for estimating the response regressions in the following subsection. Such an analytical justification is important in view of various hardware and software factors that contribute to complex traces shown in Fig. 3.

A. Response Regression

A configuration \( X \) is specified by its software components consisting of SDN controllers and northbound scripts, orchestrator modules, and hardware consisting of switches, routers, and host systems. Let \( \delta(t - iT), i = 0, 1, \ldots, n \) denote the sequence of connection setup and teardown commands, where \( t \) represents time and \( T \) is the period. Let \( T^X(t) \) denote the parameter of interest at time \( t \), namely, TCP throughput as shown in Fig. 3. Let \( R^X(t) = B - T^X(t) \) denote the parameter trace that captures the “unused” portion of the peak parameter value \( B \). For TCP traces over a connection with capacity \( B \), it is the residual bandwidth at time \( t \) above TCP throughput \( T^X(t) \). For data transfers, the desirable TCP response is to achieve throughput close to \( B \) during the \([0, T_D]\) interval in each period \( T \). To unify the analysis of ping and TCP throughput measurements, we denote the RTT over a connection by a small negative number, and let \( B \) represent the RTT of the connection that has been set up. The parameter trace is close to zero when throughput is close to the peak or when the RTT estimate is accurate, and close to \( B \) during the \([T_D, T]\) portion of each cycle.

We define the response function as

\[
R_i^X(t) = R^X(t - iT), t \in [0, T_D)
\]

which is the response to the \( i \)th setup \( \delta(t - iT) \), for \( i = 0, 1, \ldots, n \). An ideal response function is an impulse function that represents instantaneous connection restoration and path setup in Scenarios A and B, respectively, followed by instantaneous ping/TCP throughput recovery. As seen in measured traces, \( R_i^X(t) \) is a somewhat “flattened” impulse function with significant statistical variations. In general, the narrower this function, the quicker is the connection recovery or setup.

The response regression of configuration \( X \) is defined as

\[
\bar{R}^X(t) = E[R_i^X(t)] = \int R_i^X(t)dP_{R_i^X(t)},
\]

for \( t \in [0, T_D] \). The underlying distribution \( P_{R_i^X(t)} \) is in general quite complex since it depends on the dynamics of...
controllers, switches, northbound scripts and TCP or ping mechanisms. As shown in Fig. 3, these variations are different between dpctl and ODL controllers with the same HP switches, and also between HP and Cisco switches with the same ODL controller. Furthermore, in addition to the distributions of individual components, $P_{R^X(t)}$ also depends on the interactions between the components, which often lead to complex joint distributions. In general, we assume that $P_{R^X(t)}$ exhibits an overall decreasing profile for $t \in [0, T_D]$. For example, following the teardown period $[T_D, T]$, TCP throughput increases as it recovers from zero. For ping measurements, the transition is sharper as it becomes RTT $B$ at $t = 0$ and drops to 0 when ping returns an accurate RTT estimate. The duration in which it stays near $B$ is a measure of the time needed for the flows to be installed.

We define the response mean $\bar{R}_t(t)$ of $R_t(t)$ using the discrete measurements collected at times $t = j \delta$, $j = 0, 1, \ldots, T/\delta$, as

$$\bar{R}^X(j \delta) = \frac{1}{n} \sum_{i=1}^{n} \left( R^X(i \delta) \right)$$

which captures the average profile. Based on the mean differences in TCP traces for physical connection in Fig. 4, the dpctl method responds about a second faster than the ODL method. A similar relationship can be drawn from the ping traces collected over connections emulated using Mininet in VMs [7]. From an engineering perspective, the above performance comparisons and those in Section III based on the measurements seem intuitively justified. In the next subsection, we provide a statistical justification for the use of response mean $\bar{R}_t(t)$ for these comparisons by exploiting the underlying monotonic properties of the performance parameter, namely, the recovery of ping or TCP throughput following connection setup.

![Figure 4. Comparison of response means of dpctl and ODL methods for Scenario A using HP switches with 100 path degradations with $T = 50$ seconds.](image)

**B. Finite Sample Statistical Analysis**

A generic empirical estimate $\hat{R}^X(t)$ of $\bar{R}^X(t)$ based on discrete measurements collected at times $t = j \delta$, $j = 0, 1, \ldots, T_D/\delta$, is given by

$$\hat{R}^X(j \delta) = \frac{1}{n} \sum_{i=1}^{n} \left[ g \left( R^X(i \delta) \right) \right]$$

for an estimator function $g(\cdot)$ from function class $M$ of non-decreasing, namely, monotone functions. For ease of notation, we also denote $\hat{R}^X(\cdot) \in M$. The expected error $I(f)$ of the estimator $f$ is given by

$$I(f) = \int [f(t) - \bar{R}^X(t)]^2 dP_{R^X(t), t}.$$

The best expected estimator $f^* \in M$ minimizes the expected error $I(\cdot)$; that is, $I(f^*) = \min_{f \in M} I(f)$. The empirical error of an estimator $\hat{f}$ is given by

$$\hat{I}(f) = \frac{\delta}{nT_D} \sum_{j=1}^{n} \left( \frac{T_D}{\delta} \sum_{i=1}^{T_D/\delta} [f(j \delta) - \bar{R}^X(j \delta)]^2 \right).$$

The best empirical estimator $\hat{f} \in M$ minimizes the empirical error $\hat{I}(\cdot)$; that is, $\hat{I}(f^*) = \min_{f \in M} \hat{I}(f)$. Since the response mean $\bar{R}(t)$ is the mean at each observation time $j \delta$, it achieves zero mean error, which in turn leads to zero empirical error, i.e., $\hat{I}(\bar{R}) = 0$; thus, it is the best empirical estimator. By ignoring minor variations, $\bar{R}$ is closely approximated as a non-decreasing function, as indicated by the response means of dpctl and ODL methods for Scenarios A and B.

The response mean $\bar{R}(t)$ is shown to be a good approximation of the response regression $\bar{R}(t)$ in [7] based on Vapnik-Chervonenkis theory [8]. This performance guarantee is distribution-free, i.e., independent of the underlying joint distributions of controllers and switches, and is valid under very general conditions [9] on the variations of performance parameter (such as, TCP throughput or ping RTT estimate) measurements. We emphasize that the underlying distributions are quite complicated and generally unknown, since they depend on complex interactions between controller software and switches.

First, the error of estimator $\bar{R}$, given by $I(\bar{R})$, is within $\epsilon$ of the optimal error $I(f^*)$ such that

$$P \left\{ I \left( \bar{R} \right) - I(f^*) > \epsilon \right\} \leq \delta,$$

which decreases with $n$, independent of the distributions of the controllers, switches, and orchestrator modules. We now provide an outline to derive a closed-form expression for $\delta$ (details can be found in [7]). By applying the uniform bound for convergence of the expected and empirical errors [1] provided by Vapnik-Chervonenkis theory, we obtain

$$P \left\{ I \left( \bar{R} \right) - I(f^*) > \epsilon \right\} \leq 16N_{\infty} \left( \frac{\epsilon}{B} \cdot M \right) ne^{-\epsilon^2 n/(4B)^2}$$

where $N_{\infty}(\epsilon/B \cdot M)$ is the $(\epsilon/B)$-cover size of function class $M$ under $L_{\infty}$ norm. The $(\epsilon/B)$-cover size is a deterministic quantity that depends entirely on the function class, which
in turn makes the above probability bounds distribution-free. Then, the monotonicity of functions in $\mathcal{M}$ establishes that their total variation is upper-bounded by $\mathcal{B}$, which provides an upper bound for $N_{\infty}(\mathcal{R}, \mathcal{M})$ [1]. By using this bound, the following condition is derived in [7]

$$
P \left\{ I \left( \hat{R}_t \right) - I(f^+) > \epsilon \right\} < 32 \left( \frac{n}{\epsilon^2} \right)^{1+\mathcal{B}/\epsilon} \log_2(4\epsilon/\mathcal{B}) n e^{-\epsilon^2 n/(4\mathcal{B})^2}.
$$

This bound provides qualitative insights into this approach when a “sufficient” number of measurements are available. The exponential term on the right-hand side decays faster in $n$ than the growth in other terms, and hence for sufficiently large $n$ it can be made smaller than a given probability $\alpha$. Thus, the expected error $I(\hat{R})$ of the response mean used in the previous subsection is within $\epsilon$ of the optimal error $I(f^+)$ with a probability that increases with the number of observations.

III. MEASUREMENTS AND ANALYTICS

A. Scenario A

The experimental testbed for Scenario A consists of two site LANs, each consisting of multiple hosts connected via 10GigE NICs to the site’s border switch. The border switches are connected to each other via local fiber connections of a few meters in length and ANUE devices that emulate long-haul connections in hardware, as shown in Fig. 5. Tests are performed in configurations that use pairs of HP 5064zl and Cisco 3064 devices as border switches. These switches are OpenFlow-enabled, but only HP switches support the dpctl interface. We utilize OC192 ANUE device in our tests, which can emulate RTTs in the range of [0,800] milliseconds with a peak capacity of 9.6 Gbps.

TCP throughput measurements across the long-haul connection and the RTT between end hosts are constantly monitored respectively using iperf and ping, and path switching is triggered when it crosses a set threshold. Path degradations are implemented as periodic impulses and the responses are assessed using the recovery profiles of TCP throughput captured at one-second intervals. TCP throughput traces in our tests capture the performances of: (a) controllers, namely, dpctl and ODL, in response to fail-over triggers from monitoring codes, and in modifying the flow entries on switches, typically by deleting the current flows and inserting the ones for the standby path, and (b) border switches in modifying the flow entries in response to controller’s messages and re-routing the packets as per new flow entries.

Examples of TCP throughput traces are shown in Fig. 3 with the fiber connection as the primary path, and using the path switching degradation method. The connection RTT is degraded at a period of 50 sec by externally switching to the longer ANUE path, and the change is detected, which in turn triggers the fail-over. TCP throughput parameters on the hosts are tuned to achieve 10 Gbps for the default RTT, and it degrades once the connection RTT is increased to 30 ms after path switching. Upon the detection of increased RTT, the default path is restored, which in turn restores TCP throughput to the original value.

The restoration profiles of TCP throughput in these tests reflect the detection of connection degradation and fail-over, followed by the recovery response of the non-linear dynamics of TCP CUBIC congestion control mechanism. The delayed response of the ODL method compared to dpctl method can be seen in Fig. 4. Since the packets in transit are simply lost during path switching, the instantaneous TCP throughput rapidly drops to zero. Another aspect is that, while TCP throughput recovers to 10 Gbps when the direct fiber connection is used between the switches, it only peaks around 9 Gbps when packets are sent via ANUE connections with zero delay, as shown in Fig. 3. Also, the recovery profiles are different between HP and Cisco switches in otherwise identical configurations. Thus, TCP dynamics depend both on the controller, primarily in terms of recovery times, and on the switches, in terms of the peak throughput achieved and its temporal stability.

B. Scenario B

1) Testbed Measurements: Two sets of experiments are run using the testbed configuration shown in Fig. 5. The first test set utilizes OpenFlow 1.0, and the traces alternate between 50 sec local connection periods and 10 sec no connection periods. For both dpctl and ODL methods, the throughput quickly reaches the connection capacity for the local connection setup and drops to zero after teardown. However, the recovery process in the ODL case runs about 1 sec slower, and these are consistent with Scenario A results in Fig. 3. The second set of experiments are run using OpenFlow 1.3 along with dpctl and Floodlight and ONOS controllers, where the paths are switched between 0 ms and 20 ms connections. Table I displays the time epochs that correspond to the change-points in the 500-second TCP iperf traces (in 0.1 sec resolution) in response to path setup and teardown events scheduled at 50-
Table I

<table>
<thead>
<tr>
<th>Options</th>
<th>1st ON</th>
<th>1st OFF</th>
<th>2nd ON</th>
<th>2nd OFF</th>
<th>3rd ON</th>
<th>3rd OFF</th>
<th>4th ON</th>
<th>4th OFF</th>
<th>5th ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>dpctl</td>
<td>50.4</td>
<td>100.4</td>
<td>150.8</td>
<td>200.8</td>
<td>251.2</td>
<td>301.2</td>
<td>351.6</td>
<td>401.6</td>
<td>451.9</td>
</tr>
<tr>
<td>Floodlight</td>
<td>50.3</td>
<td>100.2</td>
<td>150.4</td>
<td>200.3</td>
<td>250.5</td>
<td>300.4</td>
<td>350.7</td>
<td>400.6</td>
<td>450.8</td>
</tr>
<tr>
<td>ONOS</td>
<td>50.3</td>
<td>100.2</td>
<td>150.6</td>
<td>200.5</td>
<td>250.8</td>
<td>300.8</td>
<td>351.1</td>
<td>401.0</td>
<td>451.3</td>
</tr>
</tbody>
</table>

2) VM Measurements: The connection events here comprise of 5 periods with \( T = 30 \) s and \( T_D = 20 \) s. In contrast to hardware traces in Fig. 3, the switching responses under VMs here exhibit significant delay and variations as shown in Fig. 6. The latency for the iperf traces to recover after the flows are restored is at least 6 sec, and in some cases 10 sec or longer. In addition, iperf traces exhibit a large degree of instability with throughput rates in 10-15 Mbps range over 20 Mbps connection using the dpctl method implemented in the Open vSwitches, namely, the ovs-ofctl method. The ICMP packets in the ping test, on the other hand, yield much more well-behaved traces as shown in Fig. 7. From the response regression plots, we observe: (i) the trailing edge of the ovs-ofctl response curve, as when the flows are deleted, largely coincides with that of ODL; and (ii) the leading edge of the ovs-ofctl curve, shortly after the flows are added, precedes that of the ODL curve, demonstrating the shorter response time of the former. It takes over 1 sec to recover using ODL, more than twice the amount of time with the ovs-ofctl option for most scenarios.

IV. Conclusions

We considered SDN solutions for fail-over of long-haul connections and setting up dedicated connections needed for multi-site high-performance scientific workflows. We presented analytical methods to estimate response regression functions that provide a rigorous and objective comparison of different SDN solutions and components, namely, controllers, switches and OpenFlow versions.

It would be of future interest to generalize the approach of this paper to develop a baseline test harness wherein a controller or a switch can be plugged into a known, fixed configuration. The general approach will be to develop canonical configurations, each with fixed components of the harness, such as application trigger modules, physical network connections, and others. Switching responses of different controllers, switches, or other SDN components will then be generated in these configurations, and they will be objectively compared to assess their relative performances.

ACKNOWLEDGMENTS

This work is funded by the High-Performance Networking Program and the Applied Mathematics Program, Office of Advanced Computing Research, U.S. Department of Energy, and by Extreme Scale Systems Center, sponsored by U. S. Department of Defense, and performed at Oak Ridge National Laboratory managed by UT-Battelle, LLC for U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

REFERENCES