Feature-based Comparison and Selection of Software Defined Networking (SDN) Controllers

Rahamatullah Khondoker, Adel Zaalouk, Ronald Marx, Kpatcha Bayarou
Fraunhofer Institute for Secure Information Technology
Rheinstr. 75, Darmstadt, Germany
Email: {rahamatullah.khondoker, adel.zaalouk, ronald.marx, kpatcha.bayarou}@sit.fraunhofer.de

***Abstract*—Software Defined Networking (SDN) is seen as one way to solve some problems of the Internet including security, managing complexity, multi-casting, load balancing, and energy efficiency. SDN is an architectural paradigm that separates the control plane of a networking device (e.g., a switch / router) from its data plane, making it feasible to control, monitor, and manage a network from a centralized node (the SDN controller). However, today there exists many SDN controllers including POX, FloodLight, and OpenDaylight. The question is, which of the controllers is to be selected and used? To find out the answer to this question, a decision making template is proposed in this paper to help researchers choose the SDN controller that best fits their needs. The method works as follows; first, several existing open-source controllers are analyzed to collect their properties. For selecting the suitable controller based on the derived requirements (for example, a "Java" interface must be provided by the controller), a matching mechanism is used to compare the properties of the controllers with the requirements. Additionally, for selecting the best controller based on optional requirements (for example, GUI will be extremely preferred over the age of the controller), a Multi-Criteria Decision Making (MCDM) method named Analytic Hierarchy Process (AHP) has been adapted by a monotonic interpolation / extrapolation mechanism which maps the values of the properties to a value in a pre-defined scale. By using the adapted AHP, the topmost five controllers have been compared and "Ryu" is selected to be the best controller based on our requirements.**

I. Introduction

Imagine the Internet as a person traveling through time from the sixties, to where we are now. The person would be shocked to find things flying in the air (planes) and strange looking cars, complex electronics, strange clothes, and everything looks unfamiliar and complex, the person cannot survive this sudden shock unless it starts to learn about all of this. This is exactly how the Internet survived till today. When the Internet was originally designed, it was not architected to be mobile, to communicate simultaneously with many nodes, to be secure and private, to coexist with different types of networks, or to be omnipresent. It was rather designed for exchanging information between end-hosts. Therefore, to cope with the changes, additional services are integrated to the Internet in the form of add-ons, which increases the complexity of management and control.

Software Defined networking (SDN) is a technology to keep increased complexity manageable through the use of abstractions, i.e, SDN is a clear separation of concerns by refactoring the relationship between the control plane and the data plane and decoupling them from one another.

The SDN architecture consists of a control plane and a data plane. The control plane is handled separately inside a controller. However, to be able to realize the SDN concept, one must choose a suitable controller. This decision problem can be troublesome as it is difficult to define the right metrics, and the number of controllers keeps increasing.

To solve this issue, we surveyed literatures, websites, talks, blogs, and any available resource providing information about the existing SDN controllers. Furthermore, the controllers were ranked based on their current deployment and utilization. Among them, the top five controllers have been selected for the survey to gather their properties. These controllers are: POX [1], Ryu [2], Trema [3], FloodLight [4], and OpenDaylight [5]. To derive the properties, we also modeled the SDN architecture and its approaches in a simple layered view on the whole architecture.

SDN controllers can have different properties. Selecting a controller using a single property is trivial. However, selecting a controller based on multiple properties is a Multi-Criteria Decision Making (MCDM) problem [6]. For solving such a problem, several methods exist in management science like Multiple Attribute Utility Theory (MAUT), Analytic Hierarchy Process (AHP), etc.

We used AHP to select the best controller for two reasons. Firstly, it uses pairwise prioritization. Secondly, it has an integrated consistency checking mechanism. However, to select the best controller automatically, the AHP needs to be adapted as it does not provide a mechanism to map the value of properties to the pairwise prioritization scale. The adaptation is done here by using a monotonic interpolation/extrapolation mechanism.

The topmost five controllers have been compared by utilizing the adapted AHP. The "Ryu" controller is selected to be the best controller based on our requirements.

In terms of related work, Amin Tootoonchian et. al. compared three controllers (NOX-MT, Beacon, and Maestro) corresponding to their performance [7]. However, these controllers have already been superseded by other controllers like POX, Ryu, FloodLight, and OpenDaylight.

The outline of the paper is as follows: in Section II, we surveyed the different controllers on the stage today to collect their features. To do that, we followed a methodology which is discussed in subsection II-A. In subsection II-B, We showed the results of the analysis in a layered view of the SDN Architecture. We discuss the decision making approach that

we used, namely, Analytic Hierarchy Process (AHP) in Section IV. Finally, we follow up with a conclusion in Section V.

II. Survey of SDN Controllers

The goal of the survey was to capture the properties of topmost five controllers.

A. Methodology for the Survey

For finding out the properties of the controllers, we followed the following steps. In the first step, we searched for the conferences, journals, and workshops papers where those controllers have already been described. If we found a particular property and its value, then we include those information into the table. In the second step, only the websites of each controller have been searched. We entered the properties and their values into the table if we found that on the website. This step helped to get other sources to look at, for example, published papers, other websites, talks and blogs. In the third step, technical talks from the conferences, workshops have been listened. Based on the talks and their corresponding presentations and published papers, the properties and their values are noted in the table. In the fourth step, technical blogs about the controllers have been studied. The rate of the response in the blogs has also be valued during taking notes.

In some cases, the properties and their values have been checked and verified by comparing the information of different sources so that the biased information from the developers can be avoided.

B. Investigating Each Controller

Five topmost open source controllers in terms of their usage have been analyzed here: POX, Ryu, Trema, FloodLight, and OpenDaylight.

We have not considered other SDN controllers like FlowER [8], MUL (C) [9], NOX (C++/Python) [10], Jaxon (Java) [11], Beacon (Java) [12], Maestro (Java) [13], NDDI-OESS [14], NodeFlow (JavaScript) [15], ovs-controller (C) [16] because they are either deprecated, poorly documented, or they do not provide a mature implementation. Some controllers are not considered here as they are constructed to do special tasks. For example, RouteFlow [17], Flowvisor (Java) [18], SNAC (C++) [19], Reasonance [20], and Oflops (C) [21].

Inherited from the NOX controller, the POX, a python-based SDN controller, is used to explore SDN debugging, network virtualization, controller design, and programming models.

Supported by NTT, Ryu (Japanease for "Flow") is a component-based SDN controller. Ryu has a set of pre-defined components. These components can be modified, extended, and composed for creating a customized controller application. Any programming language can be used to develop a new component.

Trema is supported by NEC labs and its most important design goals are easy-to-write-code and performance. The scripting language "Ruby" is used to increase productivity. The compiler language "C" is used to increase performance.

Born from the Beacon, the FloodLight controller consists of a set of modules, where each module provides a service to the other modules and to the control logic application through either a simple Java API or a REST API. The controller can run on the top of Linux, Mac and Windows OS.

OpenDaylight is an open source project under linux distribution. The goal of the project is to create robust code that covers most of the major components of the SDN architecture, to gain acceptance among the vendors and users, and to have a growing community that contributes to the code and uses the code for commercial products.

After doing analysis, we obtained the interfaces of SDN controllers as shown in Figure 1. The service logic of the controlling applications interact with the controllers using northbound (NB) API and the controllers interact with the forwarding plane (dumb switches) using southbound (SB) API. The technologies that are used for the northbound API are REST/JSON [22], Java/RPC [23], OSGi [24], and Quantum [25]. The technologies that are used in the southbound API can be divided into two major categories: management and control. The management technologies are Open vSwitch Database Management Protocol (OVSDB) [26], OpenFlow-Config (OF-Config) [27], Simple Network Management Protocol (SNMP) [28], and Extensible Messaging, and Presence Protocol (XMPP) [29]. The control technologies are: OpenFlow [30], XMPP, Path Computation Element/Path Computation Element Communication Protocol (PCE/PCEP)[31], Forwarding and Control Element Separation (FoRCES) [32], Interface to Routing System (I2RS) [33], Border Gateway Protocol (BGP) [34], and BGP-LS [35]. In the forwarding plane, both physical and virtual switches can be utilized.

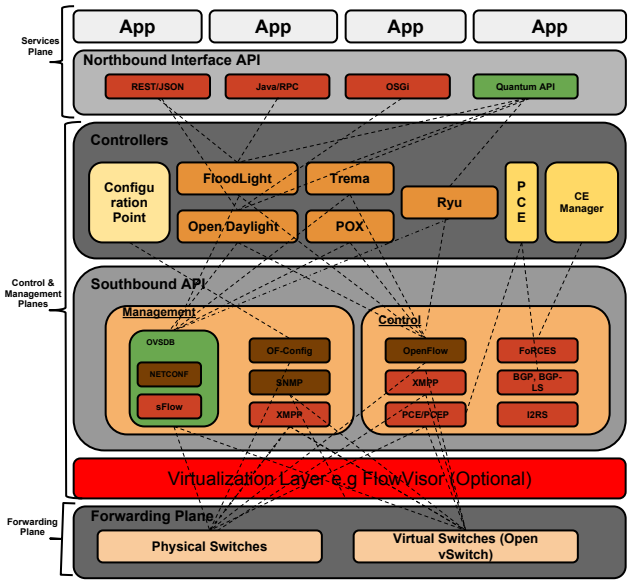

Fig. 1: *Interfaces of SDN Controllers*

C. Results of the Comparison

The controllers have been compared based on their available interfaces, support for virtual switching, Graphical User

Interface (GUI), support for REST API, the controllers productivity in terms of speed of writing codes, writing codes and performance, programming language support, modularity, operating system support, TLS support, maturity, openflow version support, and OpenStack networking support. The result of the comparison is shown in Figure 2. It can be concluded that none of the controllers is seen optimal considering all of its properties.

	POX	Ryu	Trema	FloodLight	OpenDaylight
Interfaces	SB (OpenFlow)	SB (OpenFlow) +SB Management (OVSDB JSON)	SB (OpenFlow)	SB (OpenFlow) NB (Java & REST)	SB (OpenFlow & Others SB Protocols) NB (REST & Java RPC)
Virtualization	Mininet & Open vSwitch	Mininet & Open vSwitch	Built-in Emulation Virtual Tool	Mininet & Open vSwitch	Mininet & Open vSwitch
GUI	Yes	Yes (Initial Phase)	No	Web UI (Using REST)	Yes
REST API	No	Yes (For SB Interface only)	No	Yes	Yes
Productivity	Medium	Medium	High	Medium	Medium
Open Source	Yes	Yes	Yes	Yes	Yes
Documentation	Poor	Medium	Medium	Good	Medium
Language Support	Python	Python-Specific + Message Passing Reference	C/Ruby	Java + Any language that uses REST	Java
Modularity	Medium	Medium	Medium	High	High
Platform Support	Linux, Mac OS, and Windows	Most Supported on Linux	Linux Only	Linux, Mac & Windows	Linux
TLS Support	Yes	Yes	Yes	Yes	Yes
Age	1 year	1 year	2 years	2 years	2 Month
OpenFlow Support	OF v1.0	OF v1.0 v2.0 v3.0 & Nicira Extensions	OF v1.0	OF v1.0	OF v1.0
OpenStack Networking (Quantum)	NO	Strong	Weak	Medium	Medium

Fig. 2: *Comparison among the controllers*

III. Selecting Suitable Controllers

Suitable controllers are chosen by matching the mandatory requirements of an enterprise with the properties of the controller. Our mandatory requirements were

- The controllers must have "TLS Support" and "Virtualization"
- Only "open source" controllers should be used

As all of the controllers in Figure 2 support our mandatory requirements, they are suitable to use.

The best controller is selected by matching the optional requirements of an enterprise. The list of our optional requirements is described in Section IV-D.

IV. Selecting the Best Controller

For selecting the controller, we need to consider multiple selection criteria such as interfaces, modularity, GUI and maturity. That is why, selecting the best controller is a Multi-Criteria Decision Making problem (MCDM). For solving such a problem, several Multi-Criteria Decision Analysis (MCDA) approaches are used in managerial science like Analytic Hierarchy Process (AHP), ELECTREIII, Evamix, Multiple Attribute Utility Theory (MAUT), Multi - Objective - Programming (MOP), Goal Programming (GP), NAIADE and Regime [6].

We used AHP for selecting the best controller. The selection approach which uses AHP has several advantages; first, pairwise prioritization of requirements as an input, second, consistency checking, third, benefits of relative prioritization over linear prioritization.

It is easy for people to compare two objects by using their properties. For example, a recruitment manager needs to select the best candidate for the job. One candidate has an excellent education but no working experience and the other one has a good education but has 2 years of experience. The manager will take these two selection criteria of the candidates (education, working experience) and can easily identify which is more important to him. If working experience is more important to him, then he will select the second candidate, otherwise, he will select the first one.

When the number of selection criteria increases, the consistency of the pairwise priority assignment needs to be checked. As discussed earlier, the analytic hierarchy process provides a way to check consistency.

AHP uses relative prioritization rather than linear prioritization. In linear prioritization, the priority value of the requirement is assigned linearly like $(delay > throughput > loss)$ which means that the service with the lowest delay should be selected at first. If two services have the same delay, then the service with the highest throughput is selected. In relative prioritization, the selection criteria is pairwise prioritized. That means, a service is selected based on all of the considered criteria not only a single criteria like in a linear prioritization technique.

A. The Analytic Hierarchy Process (AHP)

AHP is a process designed for assisting human decision making which is used in many application areas like social, personal, education, manufacturing, political, engineering, industry and government [36]. Basically, AHP is used for determining priorities of different alternatives.

The first step of the process is to define a hierarchy. The first and last levels in the hierarchy contain the goal and the alternatives to choose from, respectively. One or more levels in the middle contain evaluation criteria.

The second step is to assign pairwise priority to the criteria. The pairwise priority is the preference or satisfaction feelings of one evaluation criterion over another. For defining pairwise priority, a scale between 1 (equally important) to 9 (absolutely more important) is used. To make the priority assignment easier, 5 levels in the scale are used instead of 9 levels: 1 (equally important), 3 (moderately more important), 5 (strongly more important), 7 (very strongly more important), and 9 (extremely more important). In this step, an $i \times j$ dimensional comparison matrix A is constructed as shown in the following equation

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & ... & a_{1j} \\ a_{21} & a_{22} & a_{23} & ... & a_{2j} \\ . & & & & \\ . & & & & \\ . & & & & \\ a_{i1} & a_{i2} & a_{i3} & ... & a_{ij} \end{bmatrix} \quad (1)$$

where $a_{ij} > 0$, $a_{ij} = 1$ when $i = j$, and $a_{ji} = \frac{1}{a_{ji}}$.

The next step is to calculate the overall priority value, or priority vector, which provides the relative weight among the

n	1	2	3	4	5	6	7	8	9
RCI	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45

TABLE I: Random Consistency Index

things, or criteria, we compare. This is done in three steps. In the first step, column normalization is done according to the following equation

$$a1_{ij} = \frac{a_{ij}}{\sum_{i=1}^{n} a_{ij}} \tag{2}$$

In the second step, a vector is constructed by summing up the elements in each row

$$v_i = \sum_{j=1}^{n} a1_{ij} \tag{3}$$

In the final step, the priority vector w of each criteria i is obtained as follows

$$w_i = \frac{v_i}{j} \tag{4}$$

The subsequent step is to check the consistency of the priority vector by using the method proposed by Saaty, which is done in three steps. First, the largest eigenvalue of the priority matrix A, λ_{max}, is calculated. Second, the consistency index $CI(n)$ is computed by using the formula

$$CI(n) = \frac{(\lambda_{max} - n)}{(n-1)} \tag{5}$$

where n is the number of criteria. Finally, the consistency ratio, $CR(n)$, is the ratio of the consistency index $CI(n)$ and the Random Consistency Index $RCI(n)$

$$CR(n) = \frac{CI(n)}{RCI(n)} \tag{6}$$

The RCI for different different values of n are shown below.

If $CR(n)$ is less than 10%, then the assignment of the pairwise priority is said to be consistent.

If the vector is not consistent, the next step is to change the pairwise priority of the criteria and repeat the process from the second step. When the vector is proved to be consistent, the priority vector of the next levels in the hierarchy is calculated.

Except for the first criteria level (i.e., the start level having criteria) in the hierarchy, the priority vector of subsequent hierarchy levels is calculated by multiplying the priority vector calculated from the nearest upper hierarchy which is consistent, and the priority vector of the hierarchy.

B. Adaptation of Analytic Hierarchy Process (AHP) for Controller Selection

The Analytic Hierarchy Process (AHP) needs to be adapted for selecting the best controller automatically. AHP is used for determining priorities of different alternatives. The details of the AHP process is beyond the scope of this text.

To use AHP in controller selection, the following steps are performed

1) Define the goal and the selection criteria for achieving the goal
2) Priority assignment of the selection criteria defined by the manager
3) Priority assignment of the criteria for the controllers

The first step is to define the goal, which is to select the best controller, and the selection criteria to achieve that goal. The selection criteria are actually a set of required properties. Examples of the selection criteria are interfaces, virtualization, GUI, productivity, and documentation. Both functional and non-functional criteria can be selected.

After determining the selection criteria, the next step is to assign pairwise priority between the selection criteria. One of the reasons of pairwise priority assignment is that, it is easier for a person to take two criteria and to assign a priority one over the other.

The third step of the process is to assign pairwise priority between the controllers based on those selection criteria. However, as pairwise priority assignment is a time-consuming task, and as offerings are decoupled from the applications, the pairwise priority assignment of the controllers based on those selection criteria needs to automated.

This requires a mapping mechanism to map the measured/calculated values of the controllers to the pairwise priority assignment scale which will be discussed in the next section.

The priority vector coming from the user side is then multiplied by the priority vector from the offering side. The result is then called the overall priority vector. The controller with the highest priority value in the overall priority vector is the best controller.

C. Automated priority assignment for the offerings

Different controllers can have different properties. The value (or attribute) of these properties can be assigned beforehand as shown in Figure 4. An automatic mapping mechanism from measured values to the priority scale (1, 9) is required.

The mapping should have certain properties. First, the mapping must be generic, i.e. not specific to effects or units of measured values. Second, the mapping must be monotonic.

An approach for mapping has been proposed which uses a monotonic interpolation/extrapolation scheme [37] as shown in Figure 3. In this case, the application requirements provide value points for interpolation/extrapolation (must be monotonic) of measured values to the priority scale. A monotonic interpolation/extrapolation of these points is used to define a mapping. In addition, the specific measured values of the

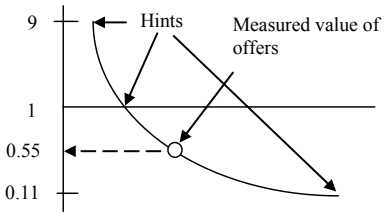

Fig. 3: Mapping mechanism

offerings are then mapped to these priorities. Assuming that $f()$ is a function used to define a mapping. As an example, considering interpolation, the requirements must contain at least the following two points

- $x_0, where f(x_0) = 0.11$
- $x_n, where f(x_n) = 9$

If there are measurement values, y, not within the interval $[x_0, x_n]$, we can extrapolate

- $if y < x_0, then f(y) = 0.11$
- $if y > x_n, then f(y) = 9$

To use inter-/extrapolation, the manager must specify two points but can have as many parameters as he wants to be more precise.

The aforementioned mapping mechanism is used to assign a priority of one controller over another for every selection criteria.

D. The Best Controller Selection

For selecting the best controller, we considered the following ten selection criteria: Interfaces, Platform support, GUI, REST API, Productivity, Documentation, Modularity, Age, OpenFlow support, and Quantum support [25]. We have not considered the criterion "language support". According to our requirements, language support is not important. The list of our optional requirements is given below

- Interfaces are
  - strongly important (7) than Age
  - strongly important (5) than GUI and Documentation
  - as equally important (1) as platform support, REST API, Productivity, Modularity, OpenFlow, and Quantum
- Platform support is
  - extremely important (9) than Documentation and Age
  - strongly important (5) than GUI
  - as equally important (1) as REST API, Productivity, Modularity, OpenFlow, and Quantum
- GUI is
  - extremely important (9) than Age
  - as equally important (1) as REST API, Productivity, Documentation, and Modularity

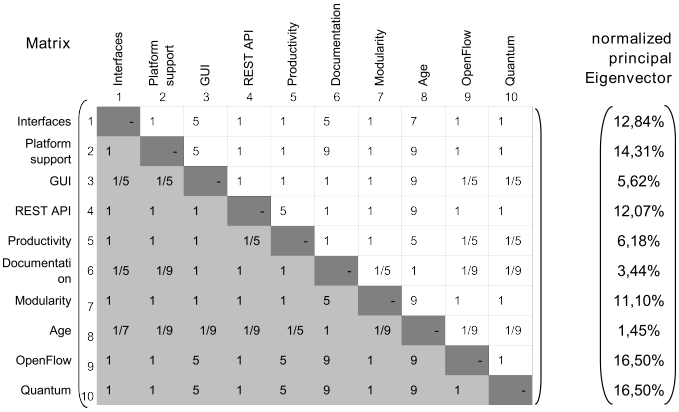

Fig. 4: Requirement matrix

- OpenFlow and Quantum are strongly important (5) than GUI
- REST API is
  - extremely important (9) than Age
  - strongly important (5) than Productivity
  - as equally important (1) as Documentation, Modularity, OpenFlow, and Quantum
- Productivity is
  - strongly important (5) than Age
  - as equally important as Documentation and Modularity
- OpenFlow and Quantum are strongly important (5) than Productivity
- Documentation is as equally important (1) as Age
- OpenFlow and Quantum are extremely important (9) than Documentation
- Modularity is
  - strongly important (5) than Documentation
  - extremely important (9) than Age
  - as equally important (1) as OpenFlow and Quantum
- OpenFlow and Quantum are extremely important (9) than Age
- OpenFlow is as equally important (1) as Quantum

The pairwise prioritization of the criteria is shown in Figure 4. It should be noted that the values in the prioritization matrix can vary based on the use case. For example, as the requirements for an enterprise environment are different from the data-centers, different priorities will be assigned to the same metrics. Consequently, the pairwise prioritization of the criteria shown in Figure 4 serves as a template to guide the users of this method.

The normalized principle Eigenvector and the consistency ratio are calculated according to AHP. As the consistency ratio is 7% which is less than 10%, the pairwise prioritization is supposed to be consistent.

	POX	Ryu	Trema	FloodLight	OpenDaylight
Interfaces	.034	.31	.034	.31	.31
Platform support	.43	.48	.48	.43	.48
GUI	.26	.17	.039	.35	.17
REST API	.038	.23	.038	.35	.35
Productivity	.077	.077	.69	.077	.077
Documentation	.036	.21	.32	.21	.21
Modularity	.17	.17	.17	.25	.25
Age	.2	.2	.3	.3	.03
OpenFlow	.17	.33	.17	.17	.17
Quantum	.03	.32	.16	.24	.24

Fig. 5: Offering matrix

By applying the mapping mechanism and normalization, we have got the offering matrix as shown in Figure 5.

The overall priority vector is calculated by multiplying the requirement vector with the offered matrix. The result is $[POX, RYU, Trema, FloodLight, OpenDaylight] = [0.146, 0.287, 0.211, 0.275, 0.268]$. RYU is selected to be the best controller as it has got the highest value.

V. Conclusion and Future Work

In this paper, we adapted a Multi-Criteria Decision Making (MCDM) method, namely, Analytic Hierarchy Process (AHP) to select the best open source SDN controller. The adaptation of AHP is done by using a mapping mechanism.

SDN controllers have been surveyed to choose the topmost five controllers according to their current deployment and utilization. Those five controllers have been analyzed in order to collect their properties like available interfaces, modularity, and productivity.

Considering our requirements and the properties of those topmost five controllers, we found that "Ryu" is the best one.

The presented work can be used as a template for SDN developers or researchers to help facilitate in the SDN controller choice process.

The future work would be to define an ontology both for the enterprise's requirements and for the offering of properties by the controllers so that no intermediate translation is required during the matching process. Additionally, expert SDN controller users (i.e., experts from enterprises, data-centers, etc.) can be consulted to validate the selection of the best controller. Furthermore, input from the users can be solicited to obtain the ranking of the criteria.

References

[1] "POX Controller," accessed 11-June-2013. [Online]. Available: http://www.noxrepo.org/pox/about-pox/

[2] "Ryu," accessed 11-June-2013. [Online]. Available: http://osrg.github.io/ryu/

[3] "Trema," accessed 11-June-2013. [Online]. Available: http://trema.github.io/trema/

[4] "FloodLight," accessed 11-June-2013. [Online]. Available: http://www.projectfloodlight.org/floodlight/

[5] "OpenDaylight," accessed 11-June-2013. [Online]. Available: http://www.opendaylight.org/

[6] M. Ehrgott and X. Gandibleux, "Multiple Criteria Optimization State of the Art Annotated Bibliographic Surveys," *Kluwer Academic Publishers*, 2003.

[7] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *Hot-ICE'12 Proceedings of the 2nd USENIX conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, 2012.

[8] "FlowER," accessed 11-June-2013. [Online]. Available: https://github.com/travelping/flower/blob/master/README.md

[9] "MUL SDN Controller," accessed 11-June-2013. [Online]. Available: http://sourceforge.net/projects/mul/

[10] "NOX," accessed 11-June-2013. [Online]. Available: http://www.noxrepo.org/

[11] "Jaxon," accessed 11-June-2013. [Online]. Available: http://jaxon.onuos.org/

[12] "Beacon," accessed 11-June-2013. [Online]. Available: https://openflow.stanford.edu/display/Beacon/Home

[13] "MAESTRO," accessed 08-June-2013. [Online]. Available: http://code.google.com/p/maestro-platform/

[14] "Network Development and Deployment Initiative Open Exchange Software Suite (NDDI OESS)," accessed 08-June-2013. [Online]. Available: http://code.google.com/p/nddi/wiki/README

[15] "NodeFlow," accessed 11-June-2013. [Online]. Available: https://github.com/dreamerslab/node.flow

[16] "Open vSwitch Controller (ovs-controller)," accessed 08-June-2013. [Online]. Available: http://openvswitch.org

[17] "RouteFlow)," accessed 08-June-2013. [Online]. Available: https://sites.google.com/site/routeflow/

[18] "Flowvisor," accessed 08-June-2013. [Online]. Available: https://openflow.stanford.edu/display/DOCS/Flowvisor

[19] "Simple Network Access Control," accessed 08-June-2013. [Online]. Available: http://www.openflow.org/wp/snac/

[20] "Resonance," accessed 08-June-2013. [Online]. Available: http://www.cc.gatech.edu/ sburnett/posts/2010-05-20-resonance-netkit.html

[21] "OpenFlow Operations Per Second (Oflops)," accessed 08-June-2013. [Online]. Available: http://www.openflow.org/wk/index.php/Oflops

[22] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," in *DISSERTATION*, 2000.

[23] A. M. White, "A High-Level Framework for Network-Based Resource Sharing, IETF," January 1976.

[24] "OSGi Alliance Specifications," accessed 13-June-2013. [Online]. Available: http://www.osgi.org/Specifications/HomePage

[25] "OpenStack Networking (Quantum)," accessed 12-June-2013. [Online]. Available: https://wiki.openstack.org/wiki/Quantum

[26] B. Pfaff and B. Davie, "The Open vSwitch Database Management Protocol draft-pfaff-ovsdb-proto-02, IETF," March 2013.

[27] J. Quittek, "ONF Configuration and Management WG," ONF, Tech. Rep.

[28] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A Simple Network Management Protocol (SNMP)," May 1990.

[29] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP): Core," March 2011.

[30] N. McKeown, G. Parulkar, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," Stanford University and University of Washington and MIT and Princeton University University of California Berkeley and Washington University in St. Louis, Tech. Rep., March 2008.

[31] Y. Lee, J. L. Roux, D. King, and E. Oki, "Path Computation Element Communication Protocol (PCEP) Requirements and Protocol Extensions in Support of Global Concurrent Optimization," July 2009.

[32] A. Doria, J. H. Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern, "Forwarding and Control Element Separation (ForCES)," March 2010.

[33] "Interface to the Routing System (I2RS), ietf," accessed 12-June-2013. [Online]. Available: http://datatracker.ietf.org/wg/i2rs/charter/

[34] H. Gredler, J. Medved, and A. Farrel, "North-Bound Distribution of Link-State and TE Information using BGP draft-gredler-idr-ls-distribution-00, IETF," September 2011.

[35] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4), IETF," September 2011.

[36] T. L. Saaty, "Decision making with the analytic hierarchy process," *Int. J. Services Sciences*, vol. 1, no. 1, pp. 83–98, 2008.

[37] R. Khondoker, B. Reuther, D. Schwerdel, A. Siddiqui, and P. Müller, "Describing and Selecting Communication Services in a Service Oriented Network Architecture," in *the proceedings of the 2010 ITU-T Kleidoscope event, Beyond the Internet? Innovations for future networks and services, Pune, India*, December 2010.