# GIROFLOW: Openflow virtualized infrastructure management tool

Tarcisio do Nascimento Araujo
Computer System Engineering Departament
Instituto Militar de Engenharia
Rio de Janeiro-RJ, Brazil
Email: tarcisio@ime.eb.br

Ronaldo Moreira Salles
Computer System Engineering Departament
Instituto Militar de Engenharia
Rio de Janeiro-RJ, Brazil
Email: salles@ieee.org

***Abstract*—*FlowVisor* is a special purpose controller inside *OpenFlow* architecture witch allows capacities of network devices to be divided by creating *silces*. This enables multiple virtual networks to run on the same physical infrastructure, based on the deployment of rules for routing the packets together with the *OpenFlow* controllers and datapaths. However, *FlowVisor* has limitations in its architecture related to the absence of a specific model for managing the *silces* as well as a more friendly user interface. Thus, taking into account the limitations mentioned, this paper presents a tool to complement the *FlowVisor* called *GiroFlow*. This tool presents a model for management of the *silces* focusing on the properties of the application running on the controller. And it also uses automated interfaces with *FlowVisor* and network controllers to create and adjust the *silces* and *policies* inside a network managed by *FlowVisor*.**

## I. Introduction

One of the main proposals for future Internet based on the paradigm of virtualization are software defined networks (SDN). The principle characteristic of these networks is the separation of the control plane from the data plane. In other words, the devices have an interface for programming the forwarding table. In this paradigm, the *OpenFlow* architecture stands out as the most popular technology for implementing solutions based on the SDN concept [1].

Another feature of the software defined networks is the existence of a central controller, responsible for determining how each packet is to be processed. And like operating systems, the network controller has an overview of the network, this enables the development of various applications to support the management of the various problems of a network, allowing the application of changes in the way dataflows are routed yielding in a more dynamic approach when compared to the traditional model, based on the individual configuration of each network device [2].

In this sense, *FlowVisor* is a special purpose controller inside the *OpenFlow* architecture witch allows the capacity of the network device to be divided up creating *silces* isolated for various applications (users). As the operating system does with the peripherals, *FlowVisor* enables multiple virtual networks to run on the same physical infrastructure, based on the deployment of rules for routing packets as required by the applications which are running [3].

*FlowVisor* works like a transparent *proxy*, controlling access to the forwarding table of network devices by other controllers. This allows more than one controller to use the same network resources, for this, the tool uses the term *slice*, where each *slice* represents the view controller on the network. The *silces* have different *policies* (rules), which enable *FlowVisor* to broker the exchange of messages between controllers and the device network. The division of control-plane network is implemented in order to keep each slice isolated from the others, thus *FlowVisor* implements the virtualization control-plane network by isolating, but sharing the data-plane devices [3].

*FlowVisor* allows global control of the network to be distributed by controllers, but control of each *slice* in operation is centered in *FlowVisor* itself, enabling the tool to function as a controller of the *silces* of the whole network in operation [3]. However, *FlowVisor* has limitations in its architecture, relating to the absence of a specific model for the management of *silces* as well as a more friendly user interface.

Thus, taking into account the limitations mentioned, this paper presents a tool to complement *FlowVisor* called *GiroFlow*. This tool presents a model for the management of the *silces* focusing on the properties of the application running on the controller, using automated interfaces with *FlowVisor* and network controllers.

The remainder of this paper is structured as follows. Section II evaluates related works. Section III presents the proposed model. Section IV describes and explains the architecture. Section V validation tests are demonstrated. Finally, Section VI discusses the conclusions.

## II. Related Papers

Studies into virtualization highlight the importance of adequately separating available bandwidth, topology and routing table between the *silces* of the network as one of the key points in achieving full network virtualization. In relation to bandwidth, we can also highlight the importance that each slice should have its own share of the bandwidth, and the lack of mechanisms for achieving isolation may result in interference between the different virtual environments, so that the virtual network may compromise the performance of all the existing infrastructure[4] and [3].

Although *FlowVisor* has the ability to function in an environment with several controllers, solutions that implement a complete management for resource allocation in networks with virtualized infrastructure are still a relatively unexplored problem [5].

Among studies that do not use *FlowVisor*, we can cite [6] which introduces a mechanism for managing and controlling virtualized networks, using the isolation of resources used by each virtual environment in the Xen field of control, but is rather a solution specific to the Xen virtualization platform and furthermore requires the use of mechanisms for labeling packets.

Other papers propose improving the *FlowVisor* tool. In [7] ADVisor implementation of a new mechanism for traffic isolation between virtual topologies. However that proposal also requires manual configuration of the device network, and makes no change in the *OpenFlow* protocol which would enable *FlowVisor* to configure the datapaths, such as defining schedulers and the allocation queue.

Some other studies make use of some of the mechanisms used by different versions of *FlowVisor* with the goal of improving resource management between different *slices* running on the network. In [3] and [8] the *Vlan PCP* field for marking packets is adopted as the solution to bandwidth allocation. However the author himself, reports that the use of *Vlan PCP* is only a temporary solution, requiring that future versions implement a specific control *QoS*. Furthermore, this solution is subject to specific traffic classes being directly configured in the datapaths by command.

*FlowVisor* itself has been improved, such as being upgraded to the v1.0 *OpenFlow* protocol which introduces as an innovation use of the enqueue action to route packets through the queue configured on a datapath port, but has no mechanism to enable resource control.

The 0.10 version of *FlowVisor* [9], features improvements in the treatment of type enqueue messages, enabling the creation of queues along the flowspaces, by defining new parameters for input streams. Output type actions can also be reset as enqueue type actions. However the main limitation of this solution is also related to the fact that the datapath queue definitions must be manually configured by external applications, thus restricting the management of service classes by *FlowVisor*.

In [10] *QoS*Flow is presented, a tool for managing *QoS* in *OpenFlow* domains through an architecture which uses two main components: datapath and QosFlow controller. There is, however, no automated interface with the *FlowVisor* command fvctl.

In all these studies the proposed solutions address the management of virtualized infrastructure from the point of view of the network, not taking into consideration the applications' characteristics in terms of how they transmit their information. In other words, *QoS* is handled in the network layer and not within the application layer.

This article proposes the *GiroFlow* tool witch aims to manage *silces* of *OpenFlow* networks, where the main innovation over previous proposals is a management model which focuses on the application, and the implementation of an automated interface to *FlowVisor* and controllers.

## III. Proposed management model

To perform network management with a focus on application, it is necessary to identify the types of applications, in relation to demand for data transmission. In [11] it is explained that the applications can be classified as *inelastic* and also *elastic* where delays are tolerated, but packet losses are not accepted.

*Inelastic* applications can be subdivided into *intolerant*, characterized by not supporting the delay and not supporting losses either. *Tolerant* applications do not admit losses, but tolerate the delay.

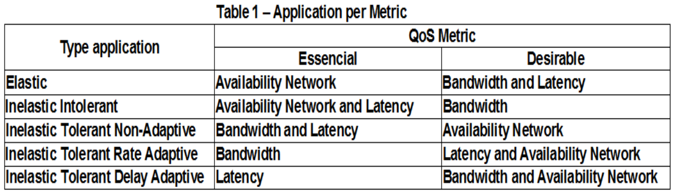
*Inelastic* and *tolerant* applications, themselves can be subdivided into *non-adaptive*, where a guaranteed minimum bandwidth is required for operation, and *adaptive* where loss rate is constant; we can adapt the operation of the application to this situation.

Finally, *inelastic* and *tolerant* and *adaptive* applications, may be classified as according to the type of adaptation. *Delay adaptive*, characterized by applications that utilize mechanism buffers, and so adjust their mode of operation when the delay is constan, and *rate adaptive* for applications that adjust operating parameters according to the available bandwidth, providing service to a greater or worse quality than the available bandwidth. Thus we can identify the existence of five main types of applications (Table 1).

In [12] several types of metrics are cited used for checking the quality of service (*QoS*) provided to an application in a network. It is possible to identify three types of *QoS* metrics which are essential for the application types, bandwidth, latency and network availability (table 1).

The proposed model implemented in the *GiroFlow* tool, lists the type of application with its most important metric for managing the operation of multiple virtual networks (*silces*) (Table 1).

Table 1 – Application per Metric



| Type application | QoS Metric | |
|---|---|---|
| | Essencial | Desirable |
| Elastic | Availability Network | Bandwidth and Latency |
| Inelastic Intolerant | Availability Network and Latency | Bandwidth |
| Inelastic Tolerant Non-Adaptive | Bandwidth and Latency | Availability Network |
| Inelastic Tolerant Rate Adaptive | Bandwidth | Latency and Availability Network |
| Inelastic Tolerant Delay Adaptive | Latency | Bandwidth and Availability Network |

Another important factor is to evaluate network service using *QoS* metrics for the application. For this, the concept of Quality of Experience (*QoE*) was used, which can be summarized as the acceptability of an application or service as subjectively perceived by the user [13]. Thus, the model proposed by *GiroFlow* seeks to perform a mapping based on the reference *QoE* values (defined in Table 2 as *x, y, z*) with the *QoS* metric value (defined in Table 2 as $\alpha$).

Then, the tool calculates the median for each metric ($\alpha$), using the history of metric measurements in the time interval set by the user. The result is then compared with the reference value of the metric defined for the implementation (*x, y, z*), obtaining the mathematical relation between the variables $\alpha$ (*x, y, z*) which indicates the state of the metric (Table 2).

The classification system of states developed by *GiroFlow* used as a reference the Mean Opinion Score (*MOS*) [14], and consists of five states. The state *underused* indicates that the application is not fully utilizing the available resource for the metric. In other words, the resource can be reduced without compromising the performance of the application on the network. The state *right* indicates that the application is using the resource appropriately, it is the state at which all applications must converge. The *warning*, *critical* and *down* states indicate, at different levels of concern, that the resource begin used does not meet the demand of the application for the evaluated metrics, requiring adjustment to be made in the network *silce* (Table 2).

The values set for the limits of each state were defined according to the authors' experience and may be adjusted within the code of the proposed tool.

Table 2 – QoS State Metric

| State | QoS Metric | | |
|---|---|---|---|
| | X = Reference QoE value Bandwidth | Y = Reference QoE value Latency | Z = Reference QoE value Availability Network |
| Underused | α ≤ 0,4 X | α ≤ 0,4Y | 0,5Z > α |
| Right | 0,4X < α ≤ 0,8X | 0,4Y < α ≤ 0,8Y | 0,2Z < α ≤ 0,5Z |
| Warning | 0,8X < α ≤ 0,9X | 0,8Y < α ≤ 0,9Y | 0,1Z < α ≤ 0,2Z |
| Critical | 0,9X < α ≤ X | 0,9Y < α ≤ Y | Z < α ≤ 0,1Z |
| Down | X > α | Y > α | α < Z |
| α=Median QoS Metric | | | |

Based on the classification of each metric relative to the application's *silce* on the network, the proposed model establishes a application's service index, which serves as a reference for evaluating the level of service for the application in general (Table 3).

The index is calculated based on the importance of metrics for the application type. In other words, the required metrics are weighted as 2 and preferred ones are weighted as 1. Each state which the metric has receives a value ranging from one for the *down* state to five for *underused* state. Thus,the highed the index, the better the service of the network *silce* for the application (Table 3). The index serves to prioritize the search for solutions that improve network performance metrics for *QoS* priority for the type of application.

Table 3 – Application's Service Index

| Type application | Formula |
|---|---|
| Elastic | [(State Availability Network *2) + (State Bandwidth) + (State Latency)]/4 |
| Inelastic Intolerant | [(State Availability Network *2) + (State Bandwidth) + (State Latency *2)]/5 |
| Inelastic Tolerant Non-Adaptive | [(State Availability Network) + (State Bandwidth *2) + (State Latency *2)]/5 |
| Inelastic Tolerant Rate Adaptive | [(State Availability Network) + (State Bandwidth *2) + (State Latency)]/4 |
| Inelastic Tolerant Delay Adaptive | [(State Availability Network) + (State Bandwidth) + (State Latency *2)]/4 |

Upon completion of the classification status of the metrics and having calculated the application's service index, *GiroFlow* enables the *FlowVisor silce* parameters to be ajusted in related to the *QoS* metrics used by it; all with the goal of improving the network performance for the application.

## IV. Architecture

*GiroFlow* architecture is supported by using interfaces to the network controller and *FlowVisor* (Fig. 1).

The controller interfaces by capturing messages exchanged between network devices and the *OpenFlow* controller. The controller, working in *debug mode*, records messages exchanged with the *datapaths* in a file. This file is constantly checked by the *GiroFlow Module* which detecting a message relating to a failure in the network (*link* or *datapaths*), updates database generating an event in *GiroFlow's Main Module* (Fig. 1). The interface itself to *FlowVisor* is implemented by running the command line *fvctl* issued by *GiroFlow* in different parts of the tool. This is either when initially registering the application in the tool in order to create the initial *silces* and *policies*, or when executing routines related to evaluation of *QoS* metrics from the network to the application (Fig. 1).

A key feature of the architecture is that for *GiroFlow* one *silce*, besides representing a controller, also represents a unique type of application. Thus, it is possible to ensure that *silces* are managed on a per-application basis (Fig. 1).

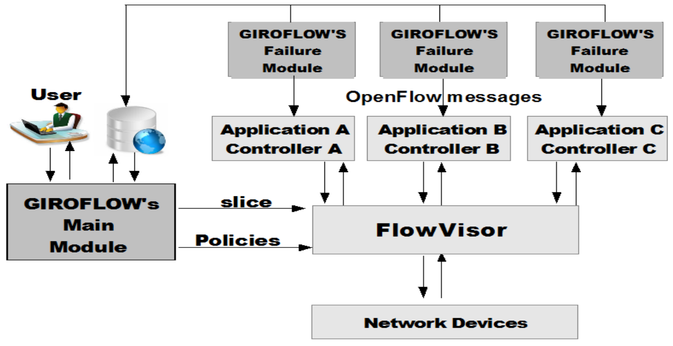

GIROFLOW'S Failure Module
GIROFLOW'S Failure Module
GIROFLOW'S Failure Module
User
OpenFlow messages
Application A Controller A
Application B Controller B
Application C Controller C
GIROFLOW's Main Module
slice
Policies
FlowVisor
Network Devices

Fig. 1. proposed architecture

## V. Validation Test

The tool presented is in the development stage, however it is possible to obtain some partial results as to the validation of the proposed management model and interfaces to *FlowVisor* and network controllers.

In the order to test *GiroFlow* a Fig.2 scenario was constructed representing a network managed by *FlowVisor* in which datapaths have been simulated using the *Mininet* tool [15]. The network management was conducted by *FlowVisor* operating on port 6634, with command line *fvctl* running on port 8083 and *Floodlight* controller running in *debug mode* on port 6642 and its web page being accessed on port 8080.

In the order to make the simulations, a record of an application of *elastic* type called Internet was created inside *GiroFlow* with route using only the gree-colored links between the datapaths in Fig.2, with the following reference values (*QoE*) metrics for *QoS* managed in the tool: bandwidth = 3

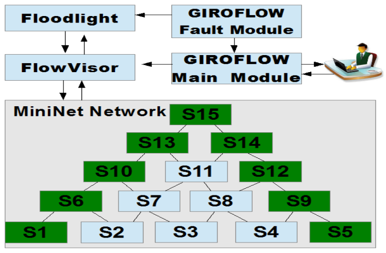


Fig. 2. Test environment

Mbps, latency = 4 ms and network availability = 85% (Table 4). 30 records relating to the measurements history of *QoS* metrics cited were also inseted into *GiroFlow*.

In the first test we tried to verify the correct functioning of interfaces. First with *FlowVisor* by automatically executing *fvctl* commands for creating the *silce* with its *policies* based on data released in the tool. Then the interface *Floodlight* controller was tested by inserting errors in datapaths related to the afore-mentioned slice into *Mininet* simulator.

After running the tests, it was observed that the *Floodlight* controller began to receive messages sent by datapaths as defined in the *policy* of the *silce*, proving the operation of the interface *FlowVisor*. It was also confirmed that *GiroFlow* received device erro messages sent by *Floodlight*, proving the functioning of the interface to the network controller.

In the second test the objective was to prove the model management concepts, this was done by running a routine in the tool for analyzing application performance. After executing the routine, it was observed in the tool that none of the *QoS* metrics were in an ideal performance state (*Right* state, table 2), and that the application's service index was 2.75 (Table 4).

Table 4 – Performance analysis of the slice

| | Bandwidth | Latency | Availability Network |
|---|---|---|---|
| Reference QoE values | 4 Mbps | 3 ms | 80% |
| QoS metric values | 3,31 Mbps | 2,97 ms | 89% |
| State of the metric | Warning | Critical | Warning |
| Application's service index = 2,75 | | | |

Then the bandwidth metric adjustment routine was run for the *silce* in the tool, causing the bandwidth value to increase to the ideal state, therefore raising the application's service index value to 3 (Table 5 ). But as it is an *elastic* application, its essential *QoS* metric is network availability. So the previous transaction was rolled back, and then the tool polity adjustment routines were run in the *silce* by analyzing and replacing datapaths with median value below the reference value (*QoE*) established for the metric. After the application's service index was re-generated it was found to be 3.2 (Table 6).

Evaluation of the different tests enables it to be confirmed that the interfaces to the players of the proposed architecture operate correctly. Following the execution of performance analysis routines in the tool, you notice the importance of application's service index to the proposed model. This is because through the index allows the solution of network problems to be prioritized while focusing on the application; which is objective of *GiroFlow*.

Table 5 – Adjust Bandwidth

| | Bandwidth | Latency | Availability Network |
|---|---|---|---|
| New Reference QoE values | 6,4 Mbps | 3 ms | 80% |
| QoS metric values | 3,31 Mbps | 2,97 ms | 89% |
| State of the metric | Right | Critical | Warning |
| Application's service index = 3,00 | | | |

Table 6 – Adjust Availability Network

| | Bandwidth | Latency | Availability Network |
|---|---|---|---|
| New Reference QoE values | 4 Mbps | 3 ms | 100% |
| QoS metric values | 3,31 Mbps | 2,97 ms | 89% |
| State of the metric | Warning | Critical | Right |
| Application's service index = 3,2 | | | |

## VI. Conclusion

*Giroflow* tool, under development, provides networks with *FlowVisor* a management model focusing on automated application management of controller *slices* and *policies* within the network infrastructure.

## Acknowledgment

We thank the Instituto Militar de Engenharia (IME) for giving the necessary conditions for the accomplishment of this work.

## References

[1] K. Greene, "Tr10: Software-defined networking," *Technology Review (MIT)*, 2009.
[2] M. R. Salvador, M. Stanton, F. M. C. UFG, and R. A. Ferreira, "Ii workshop de pesquisa experimental da internet do futuro (wpeif)," 2012.
[3] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech. Rep*, 2009.
[4] Y. Kanada, K. Shiraishi, and A. Nakao, "Network-virtualization nodes that support mutually independent development and evolution of node components," in *Communication Systems (ICCS), 2012 IEEE International Conference on*. IEEE, 2012, pp. 363–367.
[5] M. Takahashi, "Design documents: Enqueue action support," 2012. [Online]. Available: https://openflow.stanford.edu/display/DOCS/ENQUEUE+action+support
[6] N. C. Fernandes and O. Duarte, "Xnetmon: Uma arquitetura com segurana para redes virtuais," *Anais do X Simposio Brasileiro em Segurana da Informao e de Sistemas Computacionais*, pp. 339–352, 2010.
[7] E. Salvadori, R. D. Corin, A. Broglio, and M. Gerola, "Generalizing virtual network topologies in openflow-based networks," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*. IEEE, 2011, pp. 1–6.
[8] S. Min, S. Kim, J. Lee, B. Kim, W. Hong, and J. Kong, "Implementation of an openflow network virtualization for multi-controller environment," in *Advanced Communication Technology (ICACT), 2012 14th International Conference on*. IEEE, 2012, pp. 589–592.
[9] A. Al-Shabibi, "Flowvisor 0.10.0 released," 2012.
[10] A. Ishimori, J. Salvatti, F. Farias, L. Gaspary, L. Granville, and Cerqueira, "Qosflow: Gerenciamento automtico da qualidade de servio em infraestruturas de experimental baseadas em framework openflow," in *III Workshop de Pesquisa Experimental da Internet do Futuro-Simpsio Brasileiro de Redes de Computadores e Sistemas Distribdos, Ouro Preto*, 2012.
[11] L. L. Peterson and B. S. Davie, "Computer networks: A systems approach, 5e," 2010.
[12] J. F. Kurose and K. W. Ross, *Rede de Computadores e a Internet: Uma abordagem Top-Down*. Pearson Education, 2010.
[13] I.-T. P. . A. . (01/07), "New appendix i -definition of quality of experience (qoe)," 2007.
[14] M. Viswanathan, *Measurement error and research design*. Sage, 2005.
[15] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010, p. 19.