An Inter-AS Routing Component for Software-Defined Networks

Ricardo Bennesby, Paulo Fonseca, Edjard Mota and Alexandre Passito
Laboratory of Intelligent and Autonomous Computing (LabCIA)
Institute of Computing (IComp)
Federal University of Amazonas (UFAM)
Manaus - AM, Brazil
Email: {ricardo.bennesby, paulo.cesar, edjard, passito}@icomp.ufam.edu.br

Abstract—**Network management is a challenging problem of wide impact with many enterprises suffering significant monetary losses, that can be of millions per hour, due to network issues, as downtime cost. The Software Defined Networks (SDN) approach is a new paradigm that enables the management of networks with low cost and complexity. The SDN architecture consists of a control plane, a forwarding plane and a protocol that enables communication with both planes. The control plane is composed by an Operating System and applications that run on top of it. The forwarding plane contains the switches, routers, and other network equipment. Nowadays, inter-domain routing system presents some critical problems, mainly due to its fully distributed model. Recent research has showed that it would be beneficial to apply the SDN approach to address some of those problems. But it became necessary to build a new mechanism to allow inter-domain routing using SDN. The aim of this paper is to present an inter-domain routing solution using a NOX-OpenFlow architecture, based on some characteristics of the today largely used inter-domain protocol, keeping the SDN architectural principles. Although NOX-OpenFlow was originally created for routing only in enterprise networks, we propose routing beyond those, lifting this undesirable restriction of the original architecture. Our tests show that the built of this kind of application provides a much less complex, less prone to errors, and scalable solution.**

I. Introduction

The SDN approach proposes to solve some of the most well-known network problems [1] and it is composed by a very simple and straightforward architecture. The control plane present in SDN is comprised of a Network Operating System (NOX) [3] and applications that run on top of it and manage the network. All the elements in a network, such as switches and routers, compose the forwarding plane. A protocol, for example OpenFlow [1], [4], is required to enable communication between these two planes.

This approach is independent of hardware vendors, because the OpenFlow protocol obtains common information of all switches and creates an abstraction to enable the Network Operating System and its applications to manage the network. In SDN, such as NOX, the applications that manage the network are called components. The components can not access the forwarding plane directly, but only the abstraction of the programmatic interface provided by the network OS.

978-1-4673-0269-2/12/$31.00 © 2012 IEEE

The Internet is composed by a large number of groups of networks under the same administration control and policies, called Autonomous Systems (AS). Autonomous Systems, or domains, run a protocol to communicate and provide reachability among other ASes or domains. The largest deployed protocol used to exchange reachability information between ASes in the Internet is BGP.

The aim of this paper is to propose a component that runs on top of a network OS and provides inter-Autonomous System reachability exchange to endow one host of an AS with the ability to send and receive information (packets) to a host in another AS. We call this solution an Inter-AS Component. The main contributions of this work are: (1) development of a novel inter-domain routing solution, based on the SDN approach, that propose to solve some BGP problems; (2) extension of two NOX components to support Inter-AS Component features; and (3) positive test results from experiments performed in the Mininet environment which demonstrate the feasibility of our proposed solution.

This paper is organized as follows: Section II defines the NOX/OpenFlow architecture including important concepts involved on its implementation. Section III describes the BGP protocol currently used on the Internet and some of its shortcomings. Section IV depicts our solution for inter-domain routing using the SDN approach. Section V describes the simulation environment used to run our solution and presents the scenario analyzed and the test results. Section VI discusses related work. Section VII presents our conclusions and future work.

II. NOX-OpenFlow Architecture

Enterprise networks are complex and difficult to manage. One approach being increasingly recognized today as able of making the task of managing those networks easier is Software Defined Networks (SDN) [2]. SDNs are composed by a Network Operating System, applications (components) and a protocol that communicates the Network Operating System with the datapaths in the network. The Network Operating System called NOX [3] was created by a group of researchers at Stanford University as a proposal to solve network management problems. It provides a programmatic

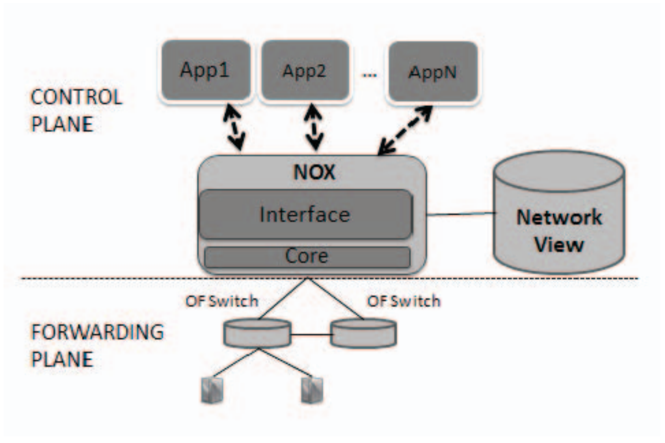

Fig. 1. Components of a NOX/OpenFlow Network

interface where applications can run and perform management tasks. With NOX, the applications can be developed with high-level terms, e.g. users and host names, instead of low-level terms, e.g. IP and MAC addresses, due to the abstraction generated by the programmatic interface. To afford this sophisticate concept, NOX maintains bindings between the physical network elements and the abstractions provided to applications. NOX can be run from a single machine and is also called the network controller.

To make the network control possible, the NOX uses the OpenFlow protocol [1], [4], also created at Stanford University. This protocol provides the network controller an interface to communicate with switches or routers and control the network traffic. In the absence of OpenFlow, hardware vendors would have to provide an open programmable interface embedded as part of all network equipment, e.g. switches and routers, they commercialize. In practice, the internal hardware responsible for this communication in such equipment is hidden and no standardization exists among different vendors. The OpenFlow solution identifies the most important set of properties and functions, e.g. flow-tables, present in routers and switches, independently of vendor, which permit it to be programmable and controlled by the applications that run on top of NOX.

The datapath of an OpenFlow Switch consists of a Flow Table and an action applied to each flow entry that describes how the switch must process the flow; a Secure Channel that connects the switch to the controller; and the OpenFlow Protocol that provides an standardized way for switches to communicate with the controller. The flow entries have the form $< header : counters, action >$, where the $header$ field contains some features of each incoming packet and field $action$ expresses the appropriated action that the OpenFlow switch must take. Examples of possible actions are: forward, drop, modify and broadcast. Henceforth, we are going to use the term datapath to designate the OpenFlow switches.

Figure 1 depicts the standard SDN mechanism illustrated through the NOX-OpenFlow architecture, similar to that present in [3]. In this configuration, the Network View is a database used to help make management decisions. The hosts connected to an OpenFlow (OF) switch can not reach NOX because it only communicates with datapaths via Secure Channel. When packets arrive on a datapath, its header is checked. If the header matches any entry in the Flow Table, the appropriated actions are taken. Otherwise, if there is no match, the packets are forwarded to the NOX controller, where applications decide what to do, taking some action and installing it on the flow-table. If the controller does not set any actions, the packets are dropped.

The OpenFlow protocol has achieved more visibility nowadays, due to the advantageous features it provides, and many researchers around the world are testing solutions to implement the Future Internet using OpenFlow. Some NOX/OpenFlow applications recently developed that explore the large potential of this architecture, include mobility control [5] and datacenter network management [6].

III. Internet Routing using BGP

To built our inter-domain component we focused on the the Internet largely used protocol as a base to the proposed method. The goal of this section is show some of its most important characteristics and point out some of its limitations.

A. The BGP protocol description

The Internet is composed by groups of networks connected in a decentralized manner [8]. Each group is called Domain or Autonomous System (AS). An AS can contain other networks with many routing protocols, but it has a single administration and it is viewed by others ASes as an unique entity, as if only one routing protocol was being executed. The inter-domain routing protocol used on the Internet is known as Border Gateway Protocol (BGP) [7]. Its goal is to exchange reachability information between ASes. This information indicates which destinations can be reached through the AS that is announcing the route. When the routing information is exchanged inside the AS it is referred to as iBGP, i.e. internal BGP. When it is exchanged between two ASes it is known as eBGP, or external BGP.

BGP is a path vector protocol, meaning that it chooses a route based on the AS path and on internal policies. BGP, currently in version 4, uses the Transmission Control Protocol (TCP) as its transport protocol, unlike the majority of the other routing schemes, which utilize the User Datagram Protocol (UDP). When two ASes establish a TCP connection they are called BGP peers. TCP port 179 is used to establish connection between two BGP peers. A router announcing the internal routes out of its AS is known as a BGP Speaker. An AS might have more than one BGP speaker. When two ASes want to change routes, their BGP speakers establish a TCP connection with each other, becoming BGP peers. Routing information is exchanged based solely on the destination address in the IP packet header, because BGP is a destination-based forwarding paradigm.

There are four types of messages that might be exchanged between two BGP peers:

1- OPEN: This message is exchanged after the TCP connection is established. The OPEN message contains the whole BGP routing table.

2- UPDATE: Advertises both new routes and routes that do not exist anymore. With this type of message, refreshing of all the contents of the BGP routing table is unnecessary.

3- KEEPALIVE: These are messages periodically sent to check if the connection between the peers persists.

4- NOTIFICATION: Indicates that some error occurred. When it is sent, the connection is closed.

The BGP routing table is called Routing Information Base (RIB) and is divided into three parts: Adj-RIBs-In, Loc-RIB and Adj-RIBs-Out. Adj-RIBs-In stores the routes that are advertised by other BGP speakers. Loc-RIB represents the routes that will be used by local speakers. Adj-RIBs-Out contains the routes that might be advertised to other speakers. Although the RIB is divided into three parts, the draft standard for BGP-4 [7] states that it does not need to be implemented in this way. The programmer might decide to implement the RIB with such division or not.

B. Some BGP drawbacks

The fully distributed model used today on the Internet brings some problems [14]. The scalability and reliability are in fact achieved distributing the inter-domain protocol among the routers, but the lack of a central control makes the policy expression more difficult. Besides that, distributed path selection can make one router depend on configuration of others; if one router is configured incorrectly the entire domain can loose optimal routes to a destination.

As one router does not know the state of the others in a domain, anomalies can happen, as one speaker has a preferred route different from another in the same domain or the occurrence of loops. Such problems could be avoided with the introduction of a central domain control having a network-wide view. With such approach the routers would forward the data traffic without concern about how the routing decision is processed.

IV. An Inter-AS Component for SDNs

Although the SDN approach was designed to minimize the problems in network management, the SDN NOX/OpenFlow architecture has so far being focused on the management of enterprise networks. Given the many benefits obtained when using SDN to solve several network problems, we consider that it is very important to have SDN widely deployed on the Internet. To help achieve this goal, we propose a new component that is able to enable routing between two or more of those enterprise networks, and more generally and perhaps most importantly among Internet domains. We call it Inter-AS Component. As the Internet is organized in domains and BGP is the main protocol used on the Internet, we developed our component based on some BGP features. Many BGP issues remain unsolved [8], [9].

Due to the fundamental differences between SDN and traditional Internet we had to focus only on some BGP attributes that are necessary to make the inter-AS routing possible in SDN. As it will be shown in this Section, we have to leave out BGP features that conflict with SDN principles, e.g. the fully distributed model in a domain, that became centralized, and the routing decision process from the routers, that moved to the network controller applications.

A. Implementation Decisions

For performance reasons, our Inter-AS Component was developed in C++. The NOX component called switch connects the network datapaths to the controller and performs the routing in this network.

When a host tries to reach another host in the network, the packets are sent first to the datapath directly connected to the sender host. Then this host checks if the information in the packet headers match any actions of its Flow Table. If the matching holds, the action described in the Flow Table is taken. If the flow does not match any flow entry, the datapath forwards it to the NOX controller. There are applications running on top of the controller, due to the programmatic interface provided. In this case the only application necessary is the switch. It analyzes the header of incoming packets, chooses which action should be taken, and inserts it in the datapath. As mentioned before, the OpenFlow Switches are simple switches with a flow-table that can communicate with a network controller using the OpenFlow protocol.

We decided to extend the switch component to support the Inter-AS routing, because it is one of the NOX core components. The switch extension checks if the destination IP belongs to its network. When destination is not part of its network, the switch invokes the Inter-AS Component to discover how to reach the destination host located in other domain. Hence, the extended switch is Inter-AS component-dependent.

B. Detailed Description

When a new SDN Domain is activated, it has to connect to its neighbors to setup its RIBs tables. The connections are made in pairs, i.e. through peers. The incoming AS tries to open a ssh connection with a neighbor and asks for the routes it knows. Then, they exchange routes. This can be done to all neighbors the incoming AS has, depending on policies. But as exposed in Section II, NOX can only communicate with the datapaths of networks it controls, via secure channel. To achieve this necessary communication we used a NOX component called messenger that enables the exchange of information between several NOX components. The messenger component has a handler that treats the messages received from another component. Specifically, messenger is used by some NOX component to open a connection on port 2603 with other component, creating a channel dedicated to messenger information. We use messenger as a dependency of the Inter-AS component.

The impact caused by introduction of this new feature can be considered of low cost to the switch component, as its behavior did not change drastically. The switch just adds a call to our Inter-AS component, in case of an outside destination, and chooses to which datapath and port the flow should be forwarded to in order to reach that destination. The process consists of the following steps:

1) The source host sends the packets to a specific destination.
2) The datapath that receives the packet checks if the packet matches any entry of its flow table.
   a) If there is a match, the actions contained in the matching Flow Entry are taken.
   b) If there is no match, the switch component is called and it checks if the destination prefix is from its domain.
      i) If the prefix is in its domain, the packets are delivered to the destination, dropped or other local action is taken.
      ii) If the prefix is from other domain, Inter-AS component is called and the RIB table is checked.
         A) If the destination prefix is in the RIB table, the controller adds a Flow Entry in the datapath that sends the flow. Then the packets are forwarded through the appropriated port.
         B) If the destination is not reachable, no action is inserted in the Flow Table of the sender datapath and the packets are dropped.

Similar to BGP, our Inter-AS component is based on the destination paradigm, because destination prefix is necessary to discover if the host belongs to the receiver domain or to another domain, and to choose which port to forward the packets to in order to reach the destination. Figure 2 shows the sequence of steps followed when packets are sent to the controller.

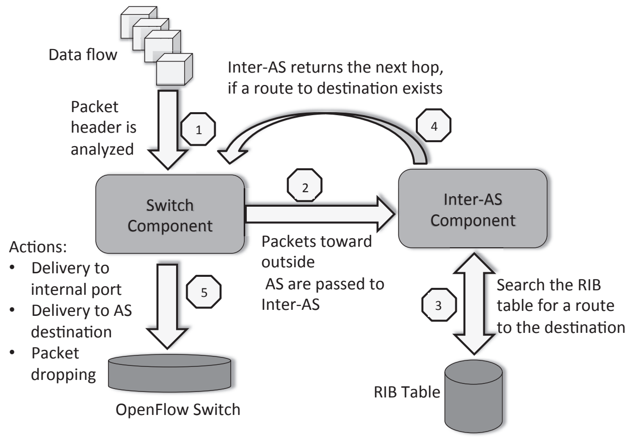
Data flow
Inter-AS returns the next hop, if a route to destination exists
Packet header is analyzed
1
4
2
Switch Component
Inter-AS Component
Actions:
• Delivery to internal port
• Delivery to AS destination
• Packet dropping
Packets toward outside AS are passed to Inter-AS
5
3
Search the RIB table for a route to the destination
RIB Table
OpenFlow Switch

Fig. 2. Routing architecture for packets arriving in the network OS

The Inter-AS component contains a RIB table, but unlike BGP, where RIB is subdivided into three parts, the Inter-AS has one sole RIB. When an AS connects to another, they exchange Open messages, identifying their AS Numbers and sending one another their entire RIB tables (depending on adopted policies). When a new route exists, or when a route is not available anymore, this route is advertised to its peers, according to the administration policies, using Update messages. If some error occurs, e.g. there is some inconsistency in the message sent, Notification messages are used. There was no need for us to develop Keepalive messages, because our socket function already checks if the connection between peers is alive, advertising when a connection is not available anymore.

The switch component interacts with the Inter-AS component using the methods provided by the latter, such as the *sendFlow* and *search* methods. The RIB table also provides information about which ASes the packets will pass until they reach their destination, through an attribute called AS Path. This information is very important because normally more than one route to the same destination exist. In this case, the shortest path is chosen, which generally happens with BGP. To perform this selection, the Inter-AS component calls the *choose_route* method, which returns the next hop AS Number. When the AS peers advertise routes, they become the Next Hops to their AS peers. The IP information of each peer is important to achieve the connection through the messenger component and perform the message exchange. Having this information, Inter-AS uses the peers' AS Number as Next Hop, instead of IPs—as traditionally happens in routing tables, as well as the datapath ID and Out Port to determine where to forward the packets. Besides the methods mentioned above, there are others that make the inter-domain routing possible. Table I depicts an instance of the RIB of an AS.

TABLE I
ROUTING TABLE STRUCTURE

Destination Prefix	AS_PATH	Next Hop AS	DPID	Port
200.0.0.0/24	2,3	2	3	3
192.168.47.0/24	2	2	3	3
173.70.46.0/24	4,5	4	3	4
190.0.0.0/24	4	4	3	4

This is an instance of a domain connected to two other ASes. This example was generated with a topology of five ASes which is the topology used in the simulation explained in Section V. The first column of Table I contains the destination network prefixes the AS can reach; the second lists the ASes that the flows will pass to reach the destination, according to the chosen route which is described by the AS_PATH attribute; the third column informs about the Next Hop AS the packets will be sent to, according to the chosen route; the forth indicates the Datapath ID (DPID), i.e., the datapath Flow Table in the AS where the Flow Entry should be installed; the fifth column shows the datapath Out Port number through which packets will be forwarded.

As part of the implementation of this example, a new set of actions was installed in the switch component to forward the packets according to the next hop indicated by the Inter-

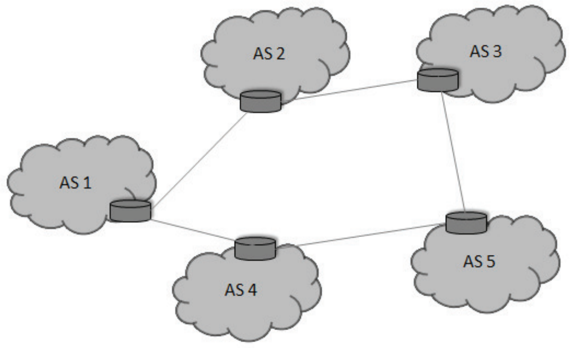

Fig. 3. Topology with five ASes

AS component. In the case that packets would have to be dropped because the destination is not of an internal AS, the actions force the internal AS datapath to forward the packets to another datapath, and this process is repeated until the destination is reachable or the controller of the sender AS notices that a chosen path is no longer available.

An important attribute of the Inter-AS component is the AS_PATH attribute. Besides informing which route can be chosen based on the size of candidate paths to a destination, this attribute can help in loop detection. For instance, consider the topology depicted in Figure 3. Suppose AS1 advertises AS2 that AS4 can be reached by flows passing by AS1, since AS2 does not have a direct connection with AS4. AS2 gets advertisements of AS3 and while advertising AS1 about its RIB table updates, AS2 could advertise AS1 of a route previously advertised by AS1. Such repetitions could generate loops and inconsistencies on the RIB table, e.g. duplicated entries. To prevent this from happening, when a route is advertised, the Inter-AS component checks its AS_PATH attribute. If the receiver AS Number is already present in the AS_PATH, the route is not even inserted in the RIB table. The inclusion of this straightforward step ensures the elimination of several occurrences of loops and inconsistencies.

Figure 4 shows the architecture and component interaction for the Inter-AS routing. Among the different types of messages that are used for communication between the ASes, a fundamental type of UPDATE message is the Withdrawal. It advertises that one or more routes previously advertised are no longer available and should be erased from the RIB table. This measure is generally used when a connection with some AS peer, or the routes to reach the destinations, are lost. When the AS notices that it cannot reach some destination, it advertises to its peers via an UPDATE message containing routes to be withdrawal. The time to discover that a destination is not available, choose a new path and use it is called convergence time.

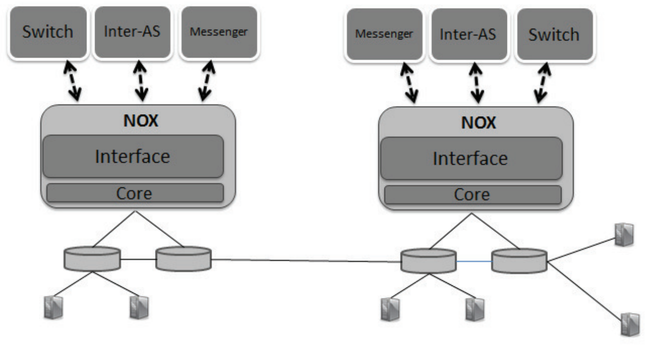

Fig. 4. Inter-AS routing and component communication

C. Discussion: Centralized x Distributed

The inter-domain routing model is fully distributed. This model is used today but brings some problems, as discussed in section III. Our solution takes advantage of those two approaches. We have a centralized controller, particularly a NOX controller, that has a network-wide view and provides an interface to the applications control the domain. This feature eliminates the inconsistence generated inside the domain until the convergence happens, as when one router in domain announces one route that is not reachable anymore because it was not updated yet. The switch component has a complete network view and knows how to reach each node inside its domain.

One may assert that the biggest shortcoming of this approach is that scalability and reliability are injured, arguing that in a huge domain one unique point of control cannot deal with the high processing rate or that the entire network domain might be down if the central controller suffers an attack.

The NOX/OpenFlow researchers have proved that the solution proposed is scalable [1], [3], [4], [5], [6], [10]. Large tests between Stanford and other universities have been done to prove several aspects, as the scalability, obtaining good results. It works well with large networks because it does not process every packet that arrive on its switches. Only the first in a flow of packets are analyzed; the other packets with the same header are are treated in the same way. The Network OS can deal well with large networks and the flexibility is maintained [3]. It leads the switches in the network with the only tasks of compare the income headers with its flow-table, apply the actions in the one that matches or forward it to the controller. The process of discover a router to a destination or other decisions are out of the switches (or routers). Initially, when the switch flow-tables are empty, the packets can take a longer time to be delivered, but after the most rules are installed in the flow-tables, this is much faster and the solution scales well, with thousands of nodes in a network.

The protocol OpenFlow creates an exclusive communication between the switches and the controller called secure channel. With this, no host in the network can reach the controller directly. But to provides more resilience, once this approach can be a point of failure, replication techniques can be developed and deployed in the architecture. As this is not the focus of the paper we are not going deeper on this field.

D. Additional discussion

The Inter-AS routing component actually runs only over NOX Operating System, but this is completely possible to implement components with the same features to other controllers, as Beacon [15], SNAC [16] or Maestro [17], since they are OpenFlow-based. The main efforts are map the implementation in one language to another and change the way some methods or functions must be implemented depending on programming interface the network OS provide. So, it is a specific solution that can be extended to other controllers based on OpenFlow protocol.

An inter-domain protocol or component must support policy expression. But the policy support was not implemented yet. But NOX and other OpenFlow controllers have a high power to express policy to control network. It is possible to drop or forward packets with a certain IP, MAC address or other characteristic, for example, in a very easy way. The development of an policy expression feature is one of next implementation steps.

The Inter-AS routing component is a propose for the future Internet. It continues being incrementally developed and tested. It is not focused on replace BGP soon because this protocol is largely deployed on the Internet domains and has been developed for years. We think that with the SDN approach growing the inter-domain routing component can be increasingly used in some domains in next years.

V. Simulations

A. Simulation Environment: Mininet

The Inter-AS component, as well as the switch and the messenger components, run on top of the NOX controlling a given network. In our experiments, this network was provided by Mininet, an environment developed to support large scale network simulation in a lightweight manner [10].

Mininet is (a) highly flexible - allowing a wide range of types of topologies and new functionalities; (b) scalable - it is possible to have an environment with hundreds or thousands of switches in one laptop; (c) realistic - it represents the true behavior of its physical counterpart, achieving high confidence results when compared to real networks. Developed at Stanford University, Mininet has been tested by more than 100 users from 18 institutions. To create a network, Mininet emulates links, switches, hosts and even controllers, using lightweight virtualization.

Our AS or domain testbed is made of one NOX controller connected to a Mininet network containing one switch and two hosts in each domain. Being able to emulate a network (or groups of networks in our case) in one single laptop, creates large test possibilities, while consuming fewer resources that might be later deployed in real networks.

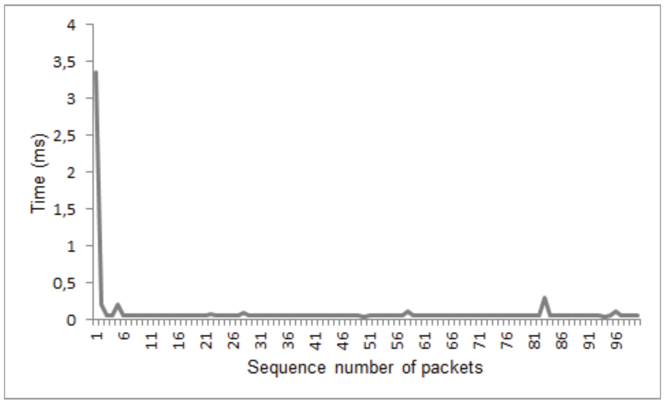

Fig. 5. Response Time of a Sequence of Packets inside AS1

B. Simulation Results

Consider the scenario depicted in Figure 3. Our simulation tests consist in verifying if the hosts of one domain can be reached by any other domain and evaluate the time it takes to send packets and receive other packets as response from the destination. To achieve this goal we utilized the ping program and statistical information collected during its execution. Ping transmits packets using the Internet Control Message Protocol (ICMP).

Given that NOX only receives the first packets of a flow matching a flow entry for analysis and installation of actions in datapaths, while the other packets that match the flow table in the switch are not forwarded to the NOX controller, we decided not to monitor the packets that arrive in the NOX controller, but monitor those in the datapath. For each response ICMP packet that arrives in the datapath, its time statistics were collected and stored in a file. For each test, a resulting graphic was generated from statistical data collected in this file.

We measured the average time of ping at the following points: (a) inside one AS; (b) between two ASes not directly connected passing through another AS; (c) passing through two ASes; and (d) passing through three ASes. We did those tests under the assumption that 100 packets were sent to the destination. The bandwidth used was of 1.8 Gbits inside one AS and of 1 Gbits between two connected ASes.

We chose AS1 for our case study, but we could have chosen any other AS, since distance and bandwidth between ASes are the same. Initially we ran ping inside AS1. Figure 5 shows the graphic of the time spent to get a response from destination and the sequence number of packets received. The Y axis represents the time of response in milliseconds and the X axis the sequence of ICMP packets. The average response time inside AS1 was 0.08 ms.

The second measurement was taken on ping's communication between an AS1 host and an AS2 host, as depicted in Figure 6. The average time was 3.44 ms. The response time between AS1 and AS3 was of 4.71 ms. Figure 7 shows the graphic of the response times of each packet passing through

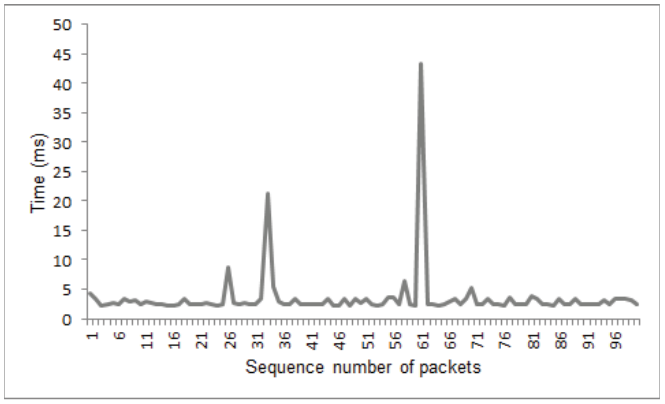

Fig. 6. Response Time of a Sequence of Packets between AS1 and AS2

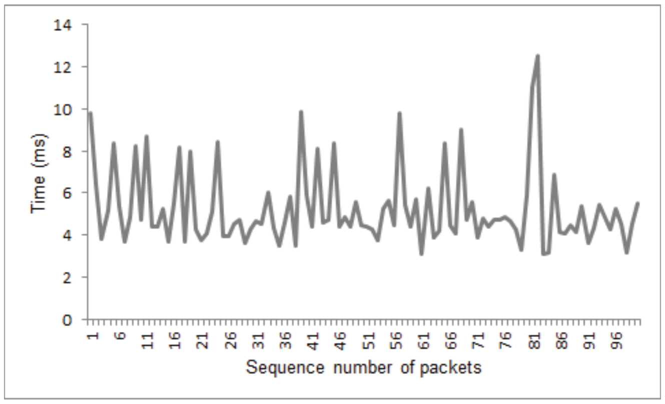

Fig. 8. Response Time of a Sequence of Packets between AS1 and AS4

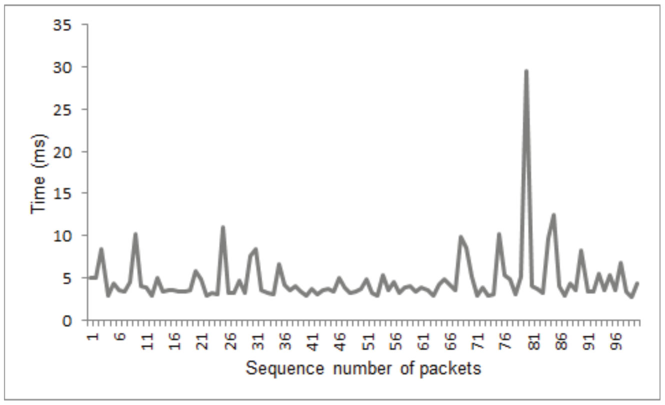

Fig. 7. Response Time of a Sequence of Packets between AS1 and AS3

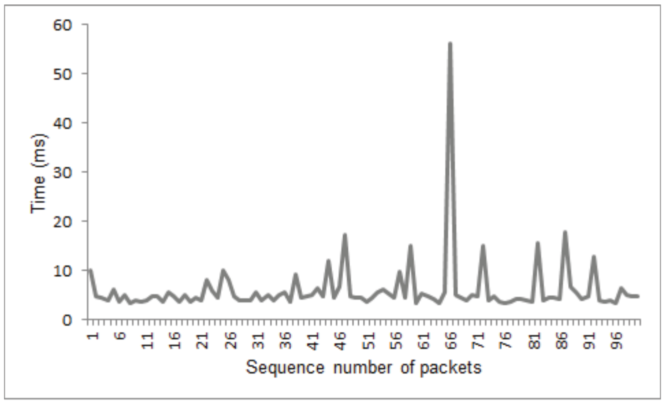

Fig. 9. Response Time of a Sequence of Packets between AS1 and AS5

AS2.

Figure 8 depicts the response time between AS1 and AS4 passing through AS2 and AS3. AS1 could send packets directly to AS4, since they are directly connected. Our primary objective, however, was to measure the response time obtained from AS paths of different lengths. The average response time obtained for this test was 5.25 ms.

Similarly to AS1 and AS4, AS5 and AS4 are directly connected and their response times are not as interesting. Hence, the path from AS5 passing through AS2, AS3 and AS4 was chosen for the tests, instead of the direct one. As depicted in Figure 9, the response time is bigger in this last test. This is expected because larger AS hops cause response time to increase. The average response time obtained was 5.93 ms.

Table II shows the average response time for communicating 100 packets from inside AS1 (reported in first column), and from AS1 to each of the other ASes, reported in the next columns. The goal of these experiments was to obtain some information about the travel time spent by packets to arrive at their destination and the time spent by response packets to arrive at the source host. Even though the number of ASes in our experiment is too small to simulate an inter-domain scenario, it provided us with useful insights on how our approach will work with a large scale test, i.e. requiring high storage and processing resources to handle a large number of ASes.

TABLE II
TESTS RESULTS

Inside AS1	AS2	AS3	AS4	AS5
0.08	3.44	4.71	5.25	5.93

We are already planning large scale experiments for more realistic and more complex scenarios.

VI. RELATED WORK

Although research efforts focusing on SDN technology are quickly gaining momentum and growing rapidly, there are only a small number of published works that provide a solution to the problem of inter-domain routing using SDN. An approach somewhat similar to our is discussed in [11]. This work describes the QuagFlow mechanism, which is based on the Quagga software. The main difference between this approach and ours is that QuagFlow uses an external server to handle packets when using protocols such as BGP. Therefore there is an extra overhead because the entries are processed on SDN controller, converted to BGP messages format and processed by the external server. Our solution requires no additional server. It runs, as a component, directly on top of NOX to avoid high performance loss.

A routing mechanism for SDN based on pathlets is described in [12]. It has a similar goal, but our approach tries to follow the Internet inter-domain routing protocol BGP, adapting its main features to the SDN. Our goal is to provide a smoothly transition from the currently used protocol to the SDN proposal.

The intra-domain vision with a centralized controller is described in [14]. This paper presents the idea of a Routing Control Platform (RCP) that is separated from routers, but observe them and is able to select routes based on the routers behalf. This architecture overcome the problems existing in todays inter-domain routing due to the fully distributed model. Our solution can be thought as the RCP, fully described in the mentioned paper.

VII. Conclusions and Future work

The SDN approach is a potential solution to many network problems. In this paper, we presented a new SDN component that provides inter-domain routing by enabling the communication with two other important components, i.e. the messenger and the switch, which were extended to provide the necessary features to achieve this goal. Our test results show that the solution proposed in this paper enhances the use of NOX-OpenFlow in order to go beyond enterprise networking. Our component is only called when there is no entry in the flow table. When an entry is inserted by the NOX controller, the datapaths start doing the forwarding by itself for matching flows, without the overhead generated by sending the flows (or their headers) to the controller. This fact, and the decision of implementing the Inter-AS component as a NOX component, show that this is an innovative solution which may become increasingly relevant to the Future Internet.

Our plans for future work include the build of a much larger scenario for testing, with hundreds or thousands of ASes, in order to better evaluate several aspects of the Inter-AS component. For example, we intend to measure the time it spends to (1) choose an alternative route when one AS in the path chosen goes down and (2) force all the AS's RIB tables to converge to the new route. We also plan to implement other functionalities to the Inter-AS component to extend it features and evaluate its performance more closely.

In addition, we are continuing to work in parallel on a mechanism to express policies and on a routing approach based on artificial intelligence, previously described in [13]. This work will allow us to extend the functions currently available to our Inter-AS Component in some new promising directions.

Acknowledgment

This project is funded by ASIA project and iOS@UFAM group. The authors Ricardo Bennesby, Paulo Fonseca and Alexandre Passito would like to thank the following Brazilian institutions for the student funding received during the development of this work : CNPq, CAPES, and FAPEAM. We are most thankful to Monica Nogueira for proofreading a draft version of this paper.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[2] S. Das, A. Sharafat, G. Parulkar, and N. McKeown, "Mpls with a simple open control plane," in *Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2011 and the National Fiber Optic Engineers Conference*, pp. 1–3, mar. 2011.

[3] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "Nox: towards an operating system for networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 105–110, Jul. 2008. [Online]. Available: http://doi.acm.org/10.1145/1384609.1384625

[4] "Openflow switch specification. Version 1.1.0," http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf, Feb 2011.

[5] K.-K. Yap, R. Sherwood, M. Kobayashi, T.-Y. Huang, M. Chan, N. Handigol, N. McKeown, and G. Parulkar, "Blueprint for introducing innovation into wireless mobile networks," in *Proceedings of the Second ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures*, ser. VISA '10. New York, NY, USA: ACM, 2010, pp. 25–32. [Online]. Available: http://doi.acm.org/10.1145/1851399.1851404

[6] A. Tavakoli, M. Casado, T. Koponen, and S. Shenker, "Applying nox to the datacenter," in the *Eighth ACM Workshop on Hot Topics in Networks (HotNets-VIII)*, 2009.

[7] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271 (Draft Standard), Internet Engineering Task Force, Jan. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4271.txt

[8] M. Yannuzzi, X. Masip-Bruin, and O. Bonaventure, "Open issues in interdomain routing: a survey," *Network, IEEE*, vol. 19, no. 6, pp. 49 – 56, Nov.-Dec. 2005.

[9] K. Butler, P. McDaniel, and W. Aiello, "Optimizing bgp security by exploiting path stability," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, ser. CCS '06. New York, NY, USA: ACM, 2006, pp. 298–310. [Online]. Available: http://doi.acm.org/10.1145/1180405.1180442

[10] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets '10. New York, NY, USA: ACM, 2010, pp. 19:1–19:6. [Online]. Available: http://doi.acm.org/10.1145/1868447.1868466

[11] M. R. Nascimento, C. E. Rothenberg, M. R. Salvador, and M. F. Magalhães, "Quagflow: partnering quagga with openflow," *SIGCOMM Comput. Commun. Rev.*, vol. 40, pp. 441–442, Aug. 2010. [Online]. Available: http://doi.acm.org/10.1145/1851275.1851252

[12] T. Koponen, S. Shenker, H. Balakrishnan, N. Feamster, I. Ganichev, A. Ghodsi, P. B. Godfrey, N. McKeown, G. Parulkar, B. Raghavan, J. Rexford, S. Arianfar, and D. Kuptsov, "Architecting for innovation," *SIGCOMM Comput. Commun. Rev.*, vol. 41, pp. 24–36. [Online]. Available: http://doi.acm.org/10.1145/2002250.2002256

[13] A. Passito, E. Mota, and R. Braga, "Towards an agent-based NOX/OpenFlow platform for the Internet," in *Workshop on Future Internet Experimental Research*, May 2010.

[14] N. Feamster, H. Balakrishnan, J. Rexford, H. Balakrishnanand, S. Shaikh and J. Merwe, "The Case for Separating Routing from Routers," in *ACM SIGCOMM Workshop on Future Directions in Network Architecture*, 2004.

[15] Beacon Controller. https://openflow.stanford.edu/display/Beacon/Home, last access in december, 2011.

[16] SNAC Controller. http://www.openflow.org/wp/snac/, last access in december, 2011.

[17] Maestro Platform. http://code.google.com/p/maestro-platform/, last access in december, 2011.