Evaluating SNMP, NETCONF, and RESTful Web Services for Router Virtualization Management

Paulo Roberto da Paz Ferraz Santos, Rafael Pereira Esteves,
Lisandro Zambenedetti Granville
Federal University of Rio Grande do Sul, Porto Alegre, Brazil
Email: {prpfsantos, rpesteves, granville}@inf.ufrgs.br

***Abstract*— In network virtualization environments (NVEs), the physical infrastructure is shared among different users (or service providers) who create multiple virtual networks (VNs). As part of VN provisioning, virtual routers (VRs) are created inside physical routers supporting virtualization. Currently, the management of NVEs is mostly realized by proprietary solutions. Heterogeneous NVEs (*i.e.,* with different equipment and technologies) are difficult to manage due to the lack of standardized management solutions. As a first step to achieve management interoperability, good performance, and high scalability, we implemented, evaluated, and compared four management interfaces for physical routers that host virtual ones. The interfaces are based on SNMP (v2c and v3), NETCONF, and RESTful Web Services, and are designed to perform three basic VR management operations: VR creation, VR retrieval, and VR removal. We evaluate these interfaces with regard to the following metrics: response time, CPU time, memory consumption, and network usage. Results show that the SNMPv2c interface is the most suitable one for small NVEs without strict security requirements and NETCONF is the best choice to compose a management interface to be deployed in more realistic scenarios, where security and scalability are major concerns.**

I. Introduction

Network virtualization emerged as an alternative to support the development of new network architectures and overcome Internet ossification [1]. In a network virtualization environment (NVE), the underlying physical infrastructure, called substrate, is shared among different users (or service providers) who create multiple virtual networks (VNs), each one employing independent protocols, forwarding schemes, and policies. Unlike server/host virtualization, which usually happens only at the edge nodes of a networked environment, network virtualization takes place in the core of the network [2]. Thus, virtual routers (VRs) are created and hosted within physical routers, enabling multiple independent networks to operate simultaneously on a single physical infrastructure [3].

Current NVEs are mostly manually operated through proprietary command line interfaces (CLIs) or proprietary management tools. The management of NVEs built on top of heterogeneous physical substrates (*i.e.,* with different equipment and technologies) poses difficulties to NVE operators because several independent tools are required to perform a management task, *e.g.,* creating a virtual network. To minimize the hurdle of managing a diversified pool of physical resources, standardized management interfaces must be defined and evaluated to allow interoperability between different virtualization platforms and management tools. In addition, virtual networks may be constructed from resources located at different administrative domains [4] and must be properly managed. Therefore, NVE operators should opt for a management interface/protocol that allows efficient and scalable management of physical routers hosting several virtual ones.

Standardization efforts have been conducted under Internet Engineering Task Force (IETF) to enable the management of VRs in different contexts. For example, the VR-MIB [5] was proposed to manage VRs in the context of L3VPNs. The VRRP-MIB [6] defines a VR as an abstract representation of a physical router and is used to handle failures. Motivated by the IETF efforts, Daitx *et al.* [2] proposed an NVE management solution based on Simple Network Management Protocol version 2c (SNMPv2c) extending the VR-MIB [5]. Although SNMP is not traditionally used for configuration-related tasks, Daitx *et al.* have demonstrated that SNMP can still be used in NVE management.

In this article, we evaluate SNMP version 3 (SNMPv3) [7], Network Configuration Protocol (NETCONF) [8], and RESTful Web services (RWS) [1] as alternatives to SNMPv2c for the management of physical routers supporting virtualization, thus extending the analysis presented by Daitx *et al.* [2]. SNMPv3 has been investigated because of its security and remote configuration features [9] that were not originally present in SNMPv2c. NETCONF and RWS have been considered possible alternatives to SNMP for network management and can overcome the limitations of SNMP in terms of scalability [10], [11]. However, neither NETCONF or RWS have been properly investigated in the context of router virtualization and management issues related to the NVE domain require further analysis. RWS were chosen because they are more lightweight than SOAP-based Web services and are being widely deployed in many Web services implementations.

In addition, we propose an updated data model as alternative to VR-MIB considering that VR-MIB did not progress in the standardization path, remaining as an expired draft since 2006 and leaving the area with no SNMP-based solution. Inspired by the recent VMM-MIB [12], we proposed a new MIB, called NEW-VMM-MIB, which is aligned with current IETF efforts towards the management of virtualized environments. For each protocol, we developed a corresponding management interface and compared them using three basic operations: VR creation, VR retrieval, and VR removal. We evaluated these interfaces with regard to the following metrics: response time, CPU time, memory consumption, and network usage.

[1] From this point we use the acronym RWS to refer to RESTful Web services along the text

978-3-901882-76-0 @2015 IFIP

In our implementations we used Net-SNMP [13] for SNMP, OpenYUMA [14] for NETCONF, and Jersey [15] for RWS. We used the XenServer [16] hypervisor to emulate a physical router supporting virtualization and the Vyatta routing suite [17] to implement VRs.

The remaining of this article is organized as follows. In Section II, we review both VR-MIB and VMM-MIB proposals along with other solutions that rely on the aforementioned protocols in network virtualization management. In Section III, we outline the architecture of a manageable physical router hosting virtual ones, describe the updated data model, and explain the message flow of each proposed management interface. The performance of the proposed management interfaces is evaluated and the results achieved are presented in Section IV. Finally, in Section V we conclude the article with final remarks and directions for future work.

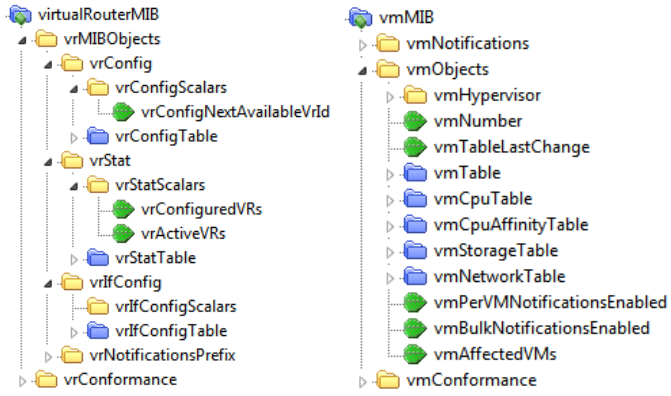
II. Related Work

Network virtualization reduces infrastructure costs by allowing multiple virtual resources to share a single physical one, which is often referred to as network consolidation [18]. Moreover, network virtualization improves the reliability of network infrastructures by offering hardware independence to virtual network components (*e.g.,* virtual routers). Management is considered a crucial aspect to enable NVEs, which is endorsed by many research projects that consider NVE management at the design stage, instead of tackling it after the NVE is already operational [19]. In this section, we discuss a number of proposals related to the management of NVEs.

The Virtual Router MIB (VR-MIB) [5], originally proposed by the L3VPN IETF Work Group [20] for Layer 3 Virtual Private Networks (L3VPNs), was one of the first initiatives to use SNMP for router virtualization management. The VR-MIB module contains objects related to the high-level configuration of VRs structured in three main tables, as shown in Fig.1(a): the `vrConfigTable`, responsible for the creation and removal of VRs; the `vrStatTable`, which stores statistics of VRs hosted in a physical router; and the `vrIfConfigTable` that manages the mapping between VRs and their virtual network interfaces. The main problem with VR-MIB is the lack of progression in the IETF standardization path and the absence of updates since 2006.

In July 2012, the VMM-MIB [12] was proposed in IETF for managing virtual machines (VMs) controlled by a hypervisor. This MIB module, which the structure is shown in Fig.1(b), consists of managed objects related to the hypervisor (`vmHypervisor` group), VMs (`vmTable`), and virtual resources allocated to VMs, *i.e.,* processors (`vmCpuTable`, `vmCpuAffinityTable`), network interfaces (`vmNetworkTable`), and storage devices (`vmStorageTable`). Despite the VMM-MIB appears to be useful for the management of VRs, because a VR can be implemented by a software-based router running in a VM, most objects of this MIB module are `read-only`, preventing proper VR configuration. Therefore, VMM-MIB requires configurable objects to support, for example, VR creation and removal.

Daitx *et al.* [2] have proposed a network virtualization management interface based on SNMP, extending the VR-MIB module to allow flexible interface binding. Daitx *et al.* used some basic management operations to evaluate the performance of the proposed interface in two virtualization platforms: XenServer and VMWare. Although, they have shown that SNMP performance largely depends on the virtualization platform, they did not investigate others management protocols and used the obsolete VR-MIB as data model.



(a) VR-MIB (b) VMM-MIB

Fig. 1. VR-MIB and VMM-MIB structures

Patricio *et al.* [21] have proposed a NETCONF-based interface for configuring virtual switches in VLAN-based virtual networks. This interface has been tested with Open vSwitch (OVS) [22] in the context of Software Defined Networks (SDNs) and supported five management operations: creation and removal of virtual switches, creation and removal of a port of a switch, and creation of a port with a VLAN tag in a switch. Aside from the fact that this interface did not manage virtual routers but switches, there was no performance evaluation to demonstrate the feasibility of the proposed interface.

Rendon *et al.* [23] have used the Mashup [24] [25] technology to provide a mechanism able to monitor heterogeneous Virtual Nodes. The proposed model enables any Virtual Infrastructure Administrator to adapt, customize, and combine existing monitoring tools in order to improve system and network monitoring tasks in virtualized environments. Rendon *et al.* have used RWS to develop Virtual Node Wrappers. These elements, located at the Adaption Layer, receive service requests from a Composition Layer and provide monitoring operations to a Managed Resources Layer to hide the complexity and heterogeneity of the substrate. Although this work has proposed a flexible and extensible system, it focused on monitoring and did not support configuration of Virtual Nodes (*e.g.,* VR creation).

The previous works addressed several management issues and provided important contributions to the network virtualization area. However, most proposals have not evaluated traditional network management protocols when applied to router virtualization management. This investigation is important to provide a better understanding about the benefits and limitations of such protocols in NVE management and identify requirements of an standardized management interface for physical routers supporting virtualization. Vendors will be encouraged to support such an interface to allow their products to be easily compatible with existing management tools and thereby facilitate the deployment of their equipment in NVEs

with predominance of other manufacturers.

III. Management Interfaces for Router Virtualization

In this section we first introduce the conceptual architecture of a physical router hosting virtual ones [2]. Then, we present the proposed management interfaces for SNMPv2c, SNMPv3, NETCONF, and RWS, respectively, describing the data model and the message flow for each management interface.

A. Architecture

A physical router is conceptually divided in two layers: a substrate layer and a virtualization layer. The substrate layer consists of the physical device itself and a layer of software, called hypervisor, that allows deployment of VRs on top of it. The virtualization layer comprises multiple isolated VRs running their own control planes. In our proposed architecture, depicted in Fig.2, the management interface is located between the VRs and the hypervisor.

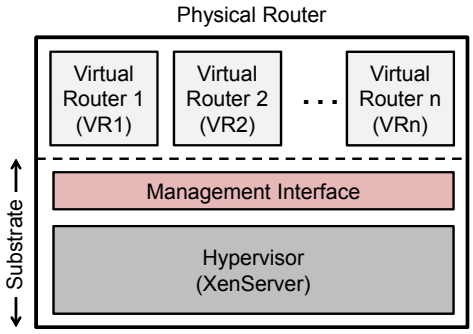


Fig. 2. Architecture of a physical router hosting VRs

The management interface allows the Infrastructure Provider (InP) operator to access, operate, maintain, and administer a physical router that supports virtualization [19]. In our case, the InP operator uses the management interface to perform basic VR management operations, such as creation, removal, and information retrieval. The definition of an appropriate data model is fundamental to support such operations.

B. Data Model

As a starting point to define a data model, we analyzed the VR-MIB structure, objects, and functionality. As mentioned before, VR-MIB is outdated, which hinders its utilization in practice. On the other hand, the VMM-MIB is an ongoing effort that focus on the management of virtual machines controlled by a hypervisor. However, VMM-MIB currently does not support VR configuration. To address these issues, we propose a NEW-VMM-MIB using the VMM-MIB as a basis and incorporating objects from VR-MIB to enable proper VR management. To achieve this, we had to add tables and objects from VR-MIB, changed access permissions for some objects in VMM-MIB, and migrated some objects from their original tables to other tables.

As depicted in Fig.3, the NEW-VMM-MIB has the same basic structure of VMM-MIB, with the addition of 2 new tables (`vmConfigTable` and `vmNetworkConfigTable`) and 3 new scalar objects (`vmConfigNextAvailableVmIndex`[2], `vmConfiguredVMs`[3], and `vmActiveVMs`[4]) adapted from VR-MIB. The `vmConfigTable`, depicted in Fig.3(c), has been included in NEW-VMM-MIB containing objects with *read-create* permission, inspired by VR-MIB, to allow VR configuration features that were missing in VMM-MIB, *e.g.,* VR creation and removal. The `vmNetworkConfigTable`, depicted in Fig.3(d), has also been included to enable the configuration of VR network interfaces. VR routing information is modeled by 5 objects (Fig.3(b)) added to `vmTable`, based on `vrStatTable` of VR-MIB. Moreover, the objects `vmName` and `vmAdminState` have been migrated from `vmTable` to `vmConfigTable`. As a result, the NEW-VMM-MIB is capable of managing VRs and VMs as well, and consequently, we used it as the data model of the SNMP management interface.

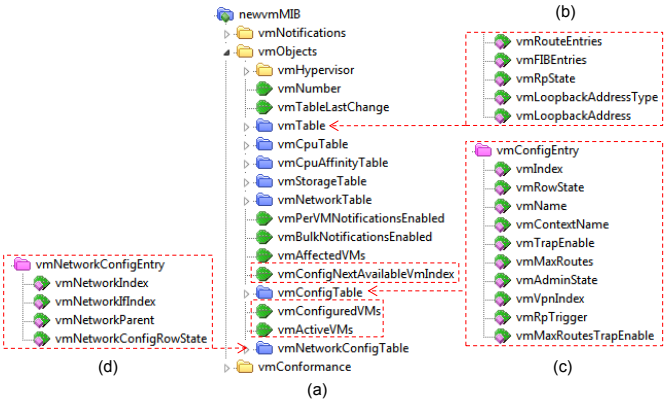


Fig. 3. NEW-VMM-MIB structure with new objects and tables

YANG is a data modeling language used to model device configuration, device state, remote procedure calls, and notifications using the NETCONF protocol [26] and we can translate Structure of Management Information version 2 (SMIv2) MIB modules to YANG modules [27]. Thus, we have defined an YANG-based data model for the NETCONF interface by directly translating the NEW-VMM-MIB module. The tree structure remains the same of NEW-VMM-MIB but *tables* are translated into *lists* and *objects* into *leaves*.

Unlike SNMP and NETCONF, the RWS solution does not have an standardized data model language. In RWS, the data model is highly coupled with the implementation. Because we have implemented a Java solution for the RWS interface, we used the Java Architecture for XML Binding (JAXB) [28] to create the data model. Using JAXB, each row of a table (*e.g.,* `vmConfigTable`) is represented in XML format but is stored as an *Object* in a `Map<K,V>`[5], where the index (*e.g.,* `vmIndex`) refers to the *key*.

Despite the particularities of each data model, the information that is modeled is essentially the same. The key difference between the proposed management interfaces relies

[2] *vmConfigNextAvailableVmIndex* provides the next available VM index value to be used to create an entry in the associated *vmConfigTable*

[3] *vmConfiguredVMs* provides the number VMs configured in the network element.

[4] *vmActiveVMs* provides the number of active VMs in the network element.

[5] *Map<K,V>* is an object that maps a key (K) to one value (V) at most and cannot contain duplicate keys.

on the management protocol, *i.e.,* SNMP, NETCONF, and RWS. Next, the messages exchanged between a manager (or client) and an agent (or server) for each interface (*i.e.,* the message flow) are discussed.

C. Message Flow

The message flow for VR creation or VR removal in the SNMPv2, SNMPv3, NETCONF, and RWS management interfaces is shown in Fig.4. The message flow for VR information retrieval for each management interfaces is shown in Fig.5. We start by describing the message flow of the SNMP interface.

A typical message exchange between an SNMPv2c manager and an agent using the proposed NEW-VM-MIB module to create/remove VR is depicted in Fig.4(a). Initially, the manager sends an SNMP `set-request` carrying the information needed by the agent to perform the desired operation. As a reaction, the agent issues a CLI request to the hypervisor that actually creates/removes the VR. Because the action at the router side may last awhile, the `set-request` is immediately replied in parallel with a `get-response`. When the requested operation finishes, a *trap* message is issued by a trap generator running inside the physical router, informing that the VR was successfully created/deleted. Since the trap message can be lost (because SNMP uses UDP), the manager initiates a timer together with the first `set-request`. If the timer expires and the confirmation trap is not received, the manager cannot guarantee the VR creation/removal. In this case, additional queries to the agent are needed to confirm the operation.

To retrieve information of a VR, the manager sends an SNMP `getbulk-request` carrying the OID of the required objects. The manager must know the `vmIndex` to retrieve information of a specific VR. Otherwise, the manager can discover the VR index by consulting the `vmConfigTable` and matching another attribute, *e.g.,* the VR name (`vmName`). The agent replies with a `get-response` containing the values of the requested objects, as depicted in Fig.5(a).

The SNMPv3 defines a number of security-related capabilities and, among them, the User-based Security Model (USM) [29] is largely used. USM provides authentication and privacy (encryption) services for SNMP. We choose HMAC-SHA-96 as the authentication protocol [29] and AES encryption [30]. Initially, the manager sends an SNMP `get-request` requesting `contextEngineID`[6] and `contextName` [7] [31]. Then, the agent responds with a `report PDU` message carrying, in addition to context information, security-related parameters in the message header. Subsequent packets contain the same messages exchanged by the SNMPv2 protocol, but are encrypted by AES and have USM security parameters in the header. The creation/removal operation messages are shown in Fig.4(b) and the retrieval ones in Fig.5(b).

In the NETCONF management interface, we choose SSH as the transport mechanism [32]. For the sake of simplicity, SSH is not depicted. In order to perform a configuration action, the NETCONF client (manager) establishes a session with the server (agent) only if a previous session is not present. The session establishment involves opening a TCP connection (omitted in the diagrams for simplicity) and the subsequent exchange of `<hello>` messages. After session establishment, the NETCONF client issues an `<edit-config>` message containing VR data (in XML format). After a `<commit>`, this configuration is applied to the *running* datastore triggering the hypervisor to actually create or remove the VR, as shown in Fig.4(c). The action of creating or removing a VR is defined in XML data as *operation="create"* or *operation="delete"*, respectively.

As depicted in Fig.5(c), to retrieve information of a VR, the client must send a `<get-config>` message to the NETCONF server. The server responds with a `<rpc-reply>` message if the operation is successful, or with a `<rpc-error>` message otherwise. If the retrieval operation is successful, the response message contains a `<data>` element in the query results. After all configurations/queries, the NETCONF session can be closed by sending the `<close-session>` message to the NETCONF server.

The RWS management interface, as the name says, is based on REST architecture [33]. The RWS solution uses HTTP methods (*e.g.,* `GET`, `POST` and `DELETE`) as verbs to request a resource identified by a Uniform Resource Identifier (URI) [34], [35]. In order to create a VR, the RWS client sends a `POST` method with data in HTTP, XML, or JSON formats. We chose XML for fairness because NETCONF data is also based on XML. The RWS server processes the request, stores the new VR information, and calls the hypervisor to create the VR. After successfully creating the VR, the server answers by sending a HTTP response with status code `200 OK` [36], as shown in Fig.4(d). VR removal (Fig.4(d)) is analogous to VR creation, except for the fact that the client sends a `DELETE` method to an URI containing the index of the VR (`vmIndex`) to be removed.

The VR information retrieval is just as simple as VR creation and removal. As shown in Fig.5(d), the client sends a `GET` method to a URI containing the index of the VR to be queried. The server searches in the datastore and, in case of success, sends back a HTTP response containing the retrieved data in XML format. If the queried VR index does not exist in the datastore, the server responds with status code `204 No Content`.

IV. Evaluation

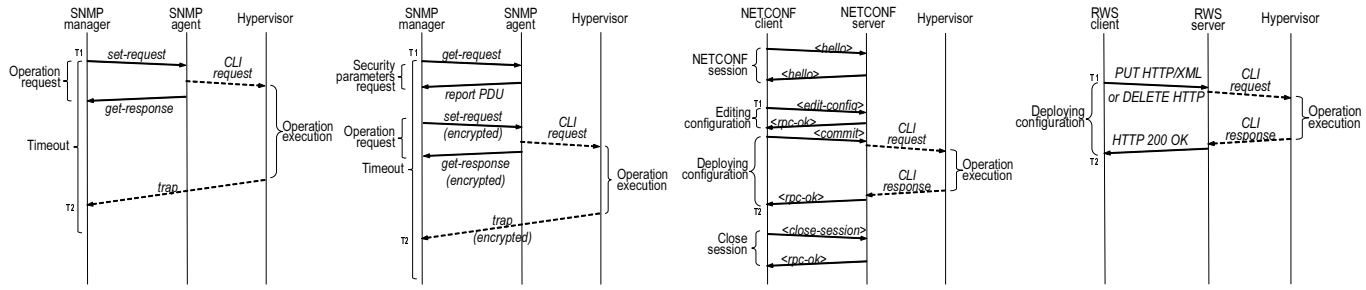
In this section we evaluated management interfaces based on SNMPv2, SNMPv3, NETCONF, and RWS to investigate which is the best solution for managing physical routers hosting VRs with regard to performance and scalability. We start by describing the test environment and pointing out all the tools used in the experiments. Then, we explain the methodology and define the metrics used in the evaluation. Finally, we present and analyze the results.

A. Test Environment

The experiments were performed on a computer with Intel Core 2 Duo 1.86GHz processor and 4GB of RAM. Citrix XenServer 5.6 [16] was used to emulate a physical router supporting virtualization. Vyatta 6.2 [17] was used as a template for VRs.

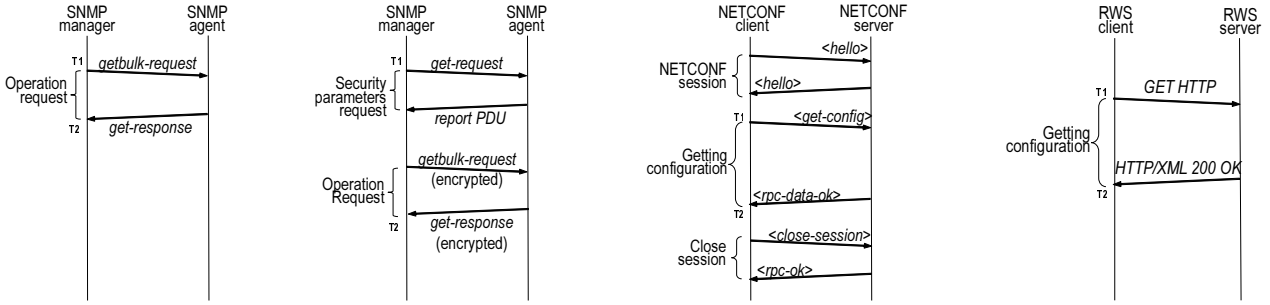
[6] contextEngineID uniquely identifies an SNMP entity that may realize an instance of a context with a particular contextName.

[7] contextName must be unique within an SNMP entity and is used to name a context.



(a) SNMPv2c (b) SNMPv3 (c) NETCONF (d) RWS

Fig. 4. Message flow of VR creation or VR removal operations



(a) SNMPv2c (b) SNMPv3 (c) NETCONF (d) RWS

Fig. 5. Message flow of VR retrieval operation

The proposed NEW-VMM-MIB module, previously discussed in Section III-B, has been compiled, instrumented, and attached to a Net-SNMP 5.7.2 agent (*snmpd*) [13]. The SNMP interface runs SNMP versions 2c and 3. The SNMP interface uses two Net-SNMP applications: *snmpset* to send `SetRequest` messages and *snmpgetbulk* to send `GetBulkRequest` messages. The SNMPv3 interface has been configured to use SHA as the authentication protocol and AES for data encryption.

We implemented the NETCONF interface using the OpenYUMA (YANG Unified Modular Automation) 2.2.5 toolkit [14]. OpenYUMA includes a NETCONF client and server (*yangcli* and *netconfd*), a YANG module compiler (*yangdump*), and a server instrumentation library (*SIL*). We translated the NEW-VMM-MIB from SMIv2 to YANG, according to RFC6643 [27], and compiled the new YANG module using *yangdump*. Then, we instrumented the generated *SIL* code and attached it to a NETCONF agent. We also have configured this interface to use NETCONF over SSH mapping.

Finally, we deployed the RWS interface using Jersey [15] implementation of JAX-RS [37], a Java API for RESTful Web services. We have used Apache Tomcat 8.0.11 [38] with Java SE Runtime Environment 1.8.0_20 [39] as a server to run the Jersey servlet, and cURL 7.37.1 [40] as a client to send messages to a URL using the HTTP methods `POST`, `GET`, or `DELETE`.

As a common characteristic, all implementations have been integrated into the virtualization platform through the native management interface of XenServer, *xe* CLI.

B. Methodology and Metrics

The experiments have been conducted according to the following sequence of operations: create VR, start VR, retrieve information about the created VR, and, finally, remove VR. Each step (except for 'start VR') was a management operation measured with regard to response time, CPU time, memory consumption, and network usage.

The response time is the total time that each VR operation takes to be completed. This evaluation intends to discover if there are significant differences between the interfaces regarding response time. Here, we consider response time as the interval between the first (`T1`) and the last (`T2`) timestamps, indicated in Fig.4 and 5. All management operations were performed locally, *i.e.,* managers (clients) and agents (servers) were executed in the same computer, to avoid network delays.

CPU time and memory consumption metrics check the processing time and the amount of memory used by the server/agent when performing a management operation, *e.g.,* VR creation. Both metrics were obtained using the `ps` program from *procps* package [41] to gather CPU time and memory consumption of the server/agent processes, *i.e., snmpd* (SNMP), *netconfd* (NETCONF), and *java* (RWS). CPU time and memory consumption were collected at the beginning (`T1`) and at the end (`T2`) of every manager operation, as indicated in Fig.4 and 5. CPU time is calculated as the difference between the final CPU time and the initial one, and memory consumption is the last memory measurement returned by `ps` because it reflects a percentage of the host total memory. In XenServer, the host memory is limited by

the memory of *Domain-0* (dom0[8]). Therefore, we calculated memory consumption by multiplying the percentage of used memory (in decimals) by the total memory of dom0.

After 30 cycles of measures, we increased the number of pre-existing active VRs. Each cycle is a sequence of management operations for a single VR, *i.e.,* creation, start, retrieval, and removal. In order to obtain statistically sound results, we used a confidence level of 95%.

The last evaluated metric reflects the network usage when management traffic is generated to create, retrieve, and remove VRs. The goal of this evaluation is to verify the impact of each management interface on the network. Network usage was evaluated using tcpdump [42] program to capture network packets exchanged between clients and servers (*i.e.,* managers and agents), including protocol overhead and session establishment/termination. Then, we calculated the amount of traffic (in bytes), by summing the length of exchanged packets for each type of operation, *i.e.,* creation, retrieval, and removal. As in the previous experiment, this one was performed locally to avoid other network-related issues, such as packet loss.

Unlike the previous experiments, for network usage we increased the number of VRs to be created, retrieved, or removed after 30 cycles of measure, instead of increasing the number of pre-existing active VRs. Network usage results were also presented with a confidence level of 95%.

C. Results

As a first observation, we found out that our XenServer did not allowed more than 6 active VRs simultaneously. This happens because our XenServer allocates 752MB of RAM for *dom0* (by default) and each Vyatta-based VR allocates 512MB (the total RAM is 4GB). Using the xentop utility we could get information about the memory allocated to *dom0* and to the active VRs. Fig.6 shows the total memory consumption in our setup when VRs are activated.

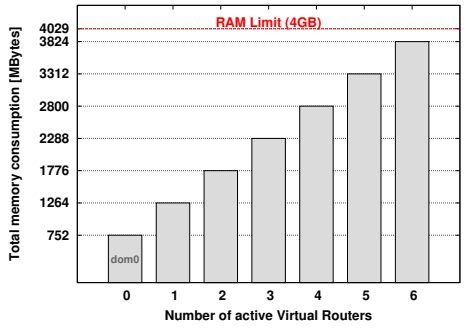


Fig. 6. Memory consumption of our XenServer and active virtual routers

Regarding response time in VR creation, depicted in Fig.7(a), the SNMPv2c and RWS interfaces presented the lowest response times, with statistically equal results. NETCONF and SNMPv3 also presented similar results but slightly higher than the previous two. For VR retrieval, the SNMPv2c interface presented a faster response, followed by NETCONF, SNMPv3, and RWS, as shown in Fig.7(b). For VR removal, SNMPv2c interface presented lower response time than the other three interfaces, whose results were statistically equal in most measurements, as shown in Fig.7(c). However, VR removal presented a far greater response time than VR creation because XenServer has to shut down a VR before deleting it, and this process takes a considerable time.

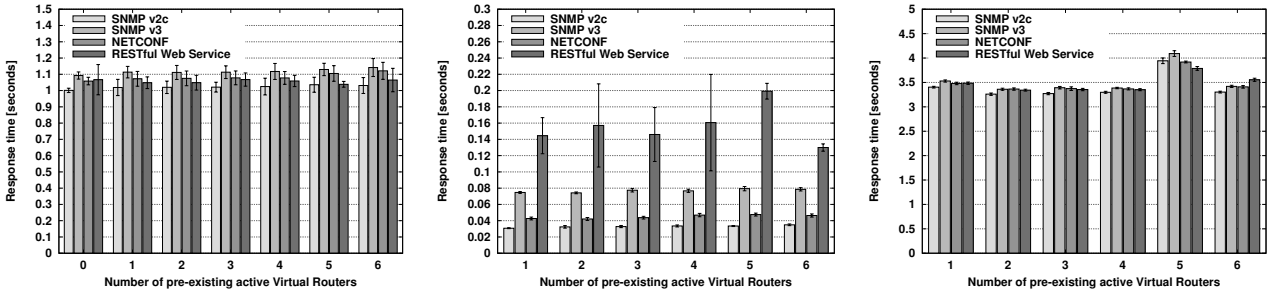
Fig.8 shows that CPU time varied irregularly as we have increased the workload. In spite of these variations, the RWS interface presented higher CPU time than the others, showing that the Java process needs more CPU compared to the others server/agents processes. The SNMPv2c interface presented the lowest CPU time in most measurements for VR creation and removal (Fig.8(a) and Fig.8(c)). For VR retrieval, the NETCONF interface resulted in the lowest CPU time when the number of pre-existing active VRs is two or higher, as shown in Fig.8(b).

With respect to memory consumption, Fig.9 shows that the RWS solution consumes much more (around 70MB) memory than the others. This result was expected because the Java process is more robust and heavier than both snmpd and netconfd processes. The SNMPv2c interface had the lowest memory consumption, followed closely by SNMPv3 and NETCONF. Memory consumption presented slight linear growth for SNMPv2c, SNMPv3, and NETCONF as the workload increased. However, results for RWS showed some variation.

The experiments for network usage were conducted without VR activation because VR activation does not generate network traffic. Therefore, it was possible to evaluate network usage for more than 6 VRs. VR configuration results (Fig.10(a) and 10(c)) show that for few VRs (up to 16 for creation, 20 for removal), the NETCONF interface generated the highest amount of traffic, followed by RWS, SNMPv3, and SNMPv2c. When the number of managed VRs was increased, NETCONF traffic grew less compared to the other interfaces. NETCONF became better than RWS when 16 or more VRs were created (Fig.10(a)) or 20 or more VRs were removed (Fig.10(c)). When 22 VRs were created or 30 or more VRs were removed, NETCONF became better than SNMPv3. Regarding retrieval operations, Fig.10(b) shows that network traffic grows uniformly when the number of retrieved VRs increases for all interfaces. However, the NETCONF interface generated more traffic compared to the others. The SNMPv2c interface presented the lowest network usage for all operations.

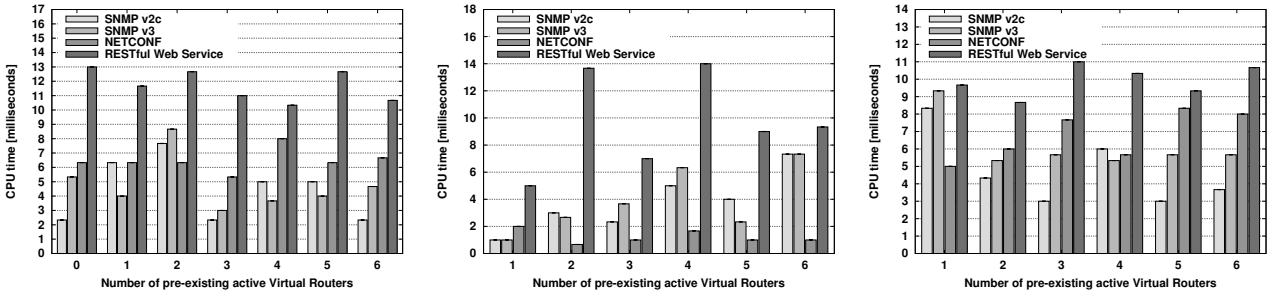
The network usage results reflects protocol overhead. SNMPv2c uses UDP and therefore has the lowest overhead. SNMPv3, despite using UDP as well, is configured to use authentication and encryption, which results in a higher overhead than version 2. RWS is based on HTTP, which uses TCP, and therefore has higher overhead than SNMP. NETCONF has the highest overhead, because it uses an XML-based data encoding for both configuration data and protocol messages, and uses SSH as the transport mechanism. In spite of that, the NETCONF interface presents a lower traffic growth when the number of created/removed VRs increases. This can be explained by the fact that NETCONF first opens the session,

[8] *Domain-0*, *dom0*, or Domain Zero is the initial domain started by the Xen hypervisor on boot and contains the host operating system.



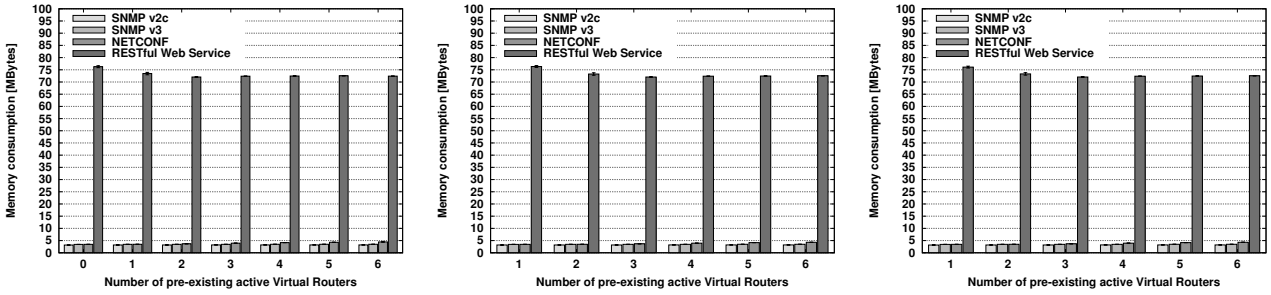
(a) Creation (b) Retrieval (c) Removal

Fig. 7. Response time of the management interfaces



(a) Creation (b) Retrieval (c) Removal

Fig. 8. CPU time of the management interfaces



(a) Creation (b) Retrieval (c) Removal

Fig. 9. Memory consumption of the management interfaces

deploys all VRs configurations and, at last, closes the session. In contrast, the interfaces based on SNMP and RWS configure multiple VRs by configuring each VR independently. On the other hand, for VR retrieval, SNMP and RWS are more efficient because SNMP relies on `GetBulkRequest` messages to get all objects at once and RWS retrieves all objects stored in the *Map* with a single `HTTP GET` message.

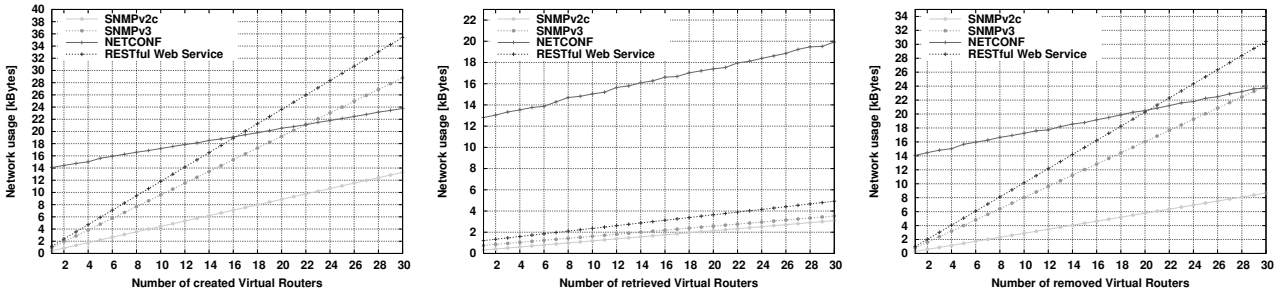
D. Comparison

Regarding response time, the SNMPv2c interface presented the best results. Considering the similarity of the results, we analyzed the average response time of the management interfaces, as shown in Table I. For VR creation, the RWS interface was better than SNMPv3, followed closely by NETCONF. However, RWS and NETCONF interfaces presented equivalent average response times to remove VRs. In retrieval operations, the average response time of the NETCONF interface was very close to SNMPv2c, but in this case, the RWS interface resulted in the highest times.

TABLE I. AVERAGE RESPONSE TIME OF MANAGEMENT INTERFACES

Interfaces	VR Creation [s]	VR Retrieval [s]	VR Removal [s]
SNMPv2c	1.02	0.03	3.41
SNMPv3	1.12	0.08	3.53
NETCONF	1.08	0.04	3.48
RWS	1.06	0.16	3.48

With regard to resource consumption, RWS achieved the highest CPU time and memory consumption. Analyzing maximum CPU time for configuration operations, there is a similarity between SNMPv2c and NETCONF, as shown in Table II. For retrieval operations, NETCONF was the interface that consumed less CPU. Regarding memory consumption, the SNMPv2c, SNMPv3, and NETCONF interfaces had much lower memory consumption than the RWS interface. Table III shows that, with respect to average memory consumption, the SNMPv2c interface was slightly better than SNMPv3,



(a) Creation (b) Retrieval (c) Removal

Fig. 10. Network usage of the management interfaces

followed by NETCONF.

TABLE II. MAXIMUM CPU TIME OF MANAGEMENT INTERFACES

Interfaces	VR Creation [ms]	VR Retrieval [ms]	VR Removal [ms]
SNMPv2c	7.7	9.3	8.3
SNMPv3	8.7	7.3	9.3
NETCONF	8.0	2.0	8.3
RWS	13.0	14.0	11.0

TABLE III. AVERAGE MEMORY CONSUMPTION OF MANAGEMENT INTERFACES

Interfaces	VR Creation [MB]	VR Retrieval [MB]	VR Removal [MB]
SNMPv2c	3.2	3.2	3.2
SNMPv3	3.5	3.5	3.5
NETCONF	3.9	3.8	3.8
RWS	73.1	73.2	73.1

The experiments related to network usage showed that the SNMPv2c management interface generated less traffic compared to the others. For retrieval operations, NETCONF resulted in a higher network usage than the other interfaces, which had similar results. For configuration operations, NETCONF started with a high network usage, but outperformed both RWS and SNMPv3 as the number of managed VRs increased. Table IV classifies the management interfaces in a descending order according to the achieved results for each evaluated metric.

TABLE IV. CLASSIFICATION OF MANAGEMENT INTERFACES ACCORDING TO THE EVALUATED METRICS

Metrics	VR Creation	VR Retrieval	VR Removal
Response time	SNMPv2c RWS NETCONF SNMPv3	SNMPv2c NETCONF SNMPv3 RWS	SNMPv2c RWS≡NETCONF SNMPv3
CPU time	SNMPv2c NETCONF SNMPv3 RWS	NETCONF SNMPv3 SNMPv2 RWS	SNMPv2c≡NETCONF SNMPv3 RWS
Memory consumption	SNMPv2c SNMPv3 NETCONF RWS	SNMPv2c SNMPv3 NETCONF RWS	SNMPv2c SNMPv3 NETCONF RWS
Network usage	SNMPv2c NETCONF SNMPv3 RWS (#VR >22)	SNMPv2c SNMPv3 RWS NETCONF	SNMPv2c NETCONF SNMPv3 RWS (#VR >30)

V. CONCLUSION

In this article, we investigated management interfaces based on SNMPv2c, SNMPv3, NETCONF, and RWS for managing physical routers supporting virtualization. We compared the interfaces by evaluating the performance of each one in terms of response time, CPU time, memory consumption, and network usage for three basic VR management operations: creation, retrieval, and removal.

Comparing the management interfaces, it is possible to observe that the SNMPv2c presented the best results for most evaluated metrics. The second best one was the NETCONF interface, which achieved results quite close to the SNMPv2c interface. Although NETCONF has presented higher network usage for few managed VRs, it demonstrated to be more scalable than SNMPv3 and RWS solutions. In addition, unlike SNMPv2c, NETCONF provides security since it uses SSH as the transport mechanism. Therefore, the SNMPv2c interface is appropriate for managing small NVEs without strict security requirements, while NETCONF appears as a promising alternative to compose a management interface to be deployed in more realistic scenarios, *i.e.,* large-scale NVEs with many VRs, where security and scalability are important concerns.

As future work, we intend to evaluate a management interface based on SOAP Web services. We also plan to investigate NETCONF combined with compression support (*zlib*) and RWS running over HTTPS. Furthermore, we look for an alternative implementation for the RWS interface, because using Java/Tomcat resulted in high resource consumption.

REFERENCES

[1] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, Apr. 2005. [Online]. Available: http://dx.doi.org/10.1109/MC.2005.136

[2] F. Daitx, R. Esteves, and L. Granville, "On the use of SNMP as a management interface for virtual networks," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, 2011, pp. 177–184. [Online]. Available: http://dx.doi.org/10.1109/INM.2011.5990689

[3] N. M. M. K. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *Communications Magazine*, vol. 47, no. 7, pp. 20–26, Jul. 2009. [Online]. Available: http://dx.doi.org/10.1109/MCOM.2009.5183468

[4] ——, "A survey of network virtualization," *Computer Network*, vol. 54, no. 5, pp. 862–876, Apr. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2009.10.017

[5] E. Stelzer, S. Hancock, B. Schliesser, and J. Laria, "Virtual Router Management Information Base Using SMIv2," http://tools.ietf.org/html/draft-ietf-l3vpn-vr-mib-04, Jul. 2005, internet draft. [Online]. Available: http://tools.ietf.org/html/draft-ietf-l3vpn-vr-mib-04

[6] K. Tata, "Definitions of Managed Objects for Virtual Router Redundancy Protocol Version 3 (VRRPv3)," RFC 6527 (Proposed Standard), Internet Engineering Task Force, Mar. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6527.txt

[7] J. Case, R. Mundy, D. Partain, and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework," RFC 3410 (Informational), Internet Engineering Task Force, Dec. 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3410.txt

[8] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network Configuration Protocol (NETCONF)," RFC 6241 (Proposed Standard), Internet Engineering Task Force, Jun. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6241.txt

[9] W. Stallings, "Snmpv3: A security enhancement for snmp," *Communications Surveys & Tutorials, IEEE*, vol. 1, no. 1, pp. 2–17, 1998.

[10] B. Hedstrom, A. Watwe, and S. Sakthidharan, "Protocol Efficiencies of NETCONF versus SNMP for Configuration Management Functions," Ph.D. dissertation, PhD thesis, MasterâAZs thesis, University of Colorado, 2011. [Online]. Available: http://morse.colorado.edu/~tlen5710/11s/

[11] A. Pras, T. Drevers, R. van de Meent, and D. Quartel, "Comparing the performance of SNMP and Web services-based management," *Network and Service Management, IEEE Transactions on*, vol. 1, no. 2, pp. 72–82, Dec 2004. [Online]. Available: http://dx.doi.org/10.1109/TNSM.2004.4798292

[12] H. Asai, M. MacFaden, J. Schoenwaelder, Y. Sekiya, K. Shima, T. Tsou, C. Zhou, and H. Esaki, "Management Information Base for Virtual Machines Controlled by a Hypervisor," Jul. 2014, draft-ietf-opsawg-vmm-mib-01. [Online]. Available: http://tools.ietf.org/html/draft-ietf-opsawg-vmm-mib-01

[13] NET-SNMP, accessed: Aug. 2014. [Online]. Available: http://www.net-snmp.org/

[14] OpenYUMA, accessed: Aug. 2014. [Online]. Available: https://github.com/OpenClovis/OpenYuma

[15] Jersey, "Restful web services in java," accessed: Aug. 2014. [Online]. Available: https://jersey.java.net/

[16] XenServer, "Open Source Virtualization," accessed: Aug. 2014. [Online]. Available: http://www.xenserver.org/

[17] Brocade, "SDN+NFV Develop and Enhance," accessed: Aug. 2014. [Online]. Available: http://community.brocade.com/t5/SDN-NFV/ct-p/SdnNfv

[18] N. Lippis III, "Network Virtualization: The New Building Blocks of Network Design," Oct. 2007, white Paper. [Online]. Available: http://www.cisco.com/c/dam/en/us/solutions/collateral/enterprise-networks/network-fabric/net_implementation_white_paper0900aecd80707cb6.pdf

[19] R. P. Esteves, L. Z. Granville, and R. Boutaba, "On the Management of Virtual Networks," *Communications Magazine, IEEE*, vol. 51, no. 7, pp. 2–10, Jul. 2013. [Online]. Available: http://dx.doi.org/10.1109/MCOM.2013.6553682

[20] Internet Engeneering Task Force (IETF), "Layer 3 Virtual Private Networks (l3vpn)," accessed: Aug. 2014. [Online]. Available: http://datatracker.ietf.org/wg/l3vpn/charter/

[21] R. Patricio, B. Batista, J. Junior, and A. Patel, "Proposal for a NETCONF Interface for Virtual Networks in Open vSwitch Environments," in *AICT 2014, The Tenth Advanced International Conference on Telecommunications*, 2014, pp. 57–62.

[22] Open vSwitch, "An open virtual switch." [Online]. Available: http://openvswitch.org/

[23] O. M. C. Rendon, C. R. P. dos Santos, A. S. Jacobs, and L. Z. Granville, "Monitoring virtual nodes using mashups," *Computer Networks*, vol. 64, no. 0, pp. 55 – 70, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128614000498

[24] J. Yu, B. Benatallah, F. Casati, and F. Daniel, "Understanding mashup development," *Internet Computing, IEEE*, vol. 12, no. 5, pp. 44–52, 2008.

[25] C. dos Santos, R. Bezerra, J. Ceron, L. Granville, and L. Rockenbach Tarouco, "On using mashups for composing network management applications," *Communications Magazine, IEEE*, vol. 48, no. 12, pp. 112–122, December 2010. [Online]. Available: http://dx.doi.org/10.1109/MCOM.2010.5673081

[26] M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)," RFC 6020 (Proposed Standard), Internet Engineering Task Force, Oct. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc6020.txt

[27] J. Schoenwaelder, "Translation of Structure of Management Information Version 2 (SMIv2) MIB Modules to YANG Modules," RFC 6643 (Proposed Standard), Internet Engineering Task Force, Jul. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6643.txt

[28] P. JAXB, accessed: Aug. 2014. [Online]. Available: https://jaxb.java.net/

[29] U. Blumenthal and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)," RFC 3414 (INTERNET STANDARD), Internet Engineering Task Force, Dec. 2002, updated by RFC 5590. [Online]. Available: http://www.ietf.org/rfc/rfc3414.txt

[30] U. Blumenthal, F. Maino, and K. McCloghrie, "The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model," RFC 3826 (Proposed Standard), Internet Engineering Task Force, Jun. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3826.txt

[31] D. Harrington, R. Presuhn, and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks," RFC 3411 (INTERNET STANDARD), Internet Engineering Task Force, Dec. 2002, updated by RFCs 5343, 5590. [Online]. Available: http://www.ietf.org/rfc/rfc3411.txt

[32] M. Wasserman, "Using the NETCONF Protocol over Secure Shell (SSH)," RFC 6242 (Proposed Standard), Internet Engineering Task Force, Jun. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6242.txt

[33] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[34] R. Fielding and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing," RFC 7230 (Proposed Standard), Internet Engineering Task Force, Jun. 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7230.txt

[35] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," RFC 3986 (INTERNET STANDARD), Internet Engineering Task Force, Jan. 2005, updated by RFCs 6874, 7320. [Online]. Available: http://www.ietf.org/rfc/rfc3986.txt

[36] R. Fielding and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content," RFC 7231 (Proposed Standard), Internet Engineering Task Force, Jun. 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7231.txt

[37] Java.net, accessed: Aug. 2014. [Online]. Available: https://jax-rs-spec.java.net/

[38] Apache, "Tomcat," accessed: Aug. 2014. [Online]. Available: http://tomcat.apache.org/

[39] Oracle, "Java SE at a Glance," accessed: Aug. 2014. [Online]. Available: http://www.oracle.com/technetwork/java/javase/

[40] cURL, "cURL groks URLs," accessed: Aug. 2014. [Online]. Available: http://curl.haxx.se/

[41] PROCPS, "The /proc file system utilities," accessed: Aug. 2014. [Online]. Available: http://procps.sourceforge.net/

[42] TCPDUMP, accessed: Aug. 2014. [Online]. Available: http://www.tcpdump.org/