SeRViTR: A Framework, Implementation, and a Testbed for a Trustworthy Future Internet

Shingo Ata\textsuperscript{a}, Dijiang Huang\textsuperscript{b}, Xuan Liu\textsuperscript{c}, Akira Wada\textsuperscript{d}, Tianyi Xing\textsuperscript{a}, Parikshit Juluri\textsuperscript{e}, Chun-Jen Chung\textsuperscript{b}, Yasuhiro Sato\textsuperscript{d}, Deep Medhi\textsuperscript{c,e}\textsuperscript{*}

\textsuperscript{a}Osaka City University, Osaka, Japan
\textsuperscript{b}Arizona State University, Tempe, AZ, USA
\textsuperscript{c}University of Missouri–Kansas City, Kansas City, MO, USA
\textsuperscript{d}Japan Coast Guard Academy, Japan
\textsuperscript{e}Indian Institute of Technology–Guwahati, India

Abstract

A flexible, scalable, and robust framework that enables fine-grained flow control under fixed or dynamic policies while addressing trustworthiness as a built-in network level functionality is a desirable goal of the future Internet. Furthermore, the level of trustworthiness may possibly be different from one network to another. It is also desirable to provide user-centric or service-centric routing capabilities to achieve service-oriented traffic controls as well as trust and policy management for security. Addressing these aspects, we present the SeRViTR (Secure and Resilient Virtual Trust Routing) framework. In particular, we discuss the goal and scope of SeRViTR, its implementation details, and a testbed that enables us to demonstrate SeRViTR. We have designed protocols and mechanisms for policy and trust management for SeRViTR and show a validation on the functional implementation of several SeRViTR components to illustrate virtual domains and trust level changes between virtual domains that are achieved under SeRViTR protocols. Going from implementation to testbed, we demonstrate SeRViTR in a virtual network provisioning infrastructure called the Geo-distributed Programmable Layer-2 Networking Environment (G-PLaNE) that connects three institutions spanning the US and Japan.

Keywords: Future Internet, Internet Trustworthiness, Architecture, Prototype, Geo-Distributed Networks, Network Virtualization

1. Introduction

An important factor that has led to the success of the current Internet is its flexible routing functionality. However, with the rapid growth of the Internet, many issues have risen. One area of interest is trustworthiness, especially at the architectural level. For instance, the current Internet has a limited capability on trustworthiness at the network layer level. For example, a secure tunnel (e.g., IPSec [28, 29]) at the network layer can be established to exchange information, or access control lists that are used for filtering traffic from unreliable networks by adding network prefixes to their blacklist. An approach such as IPSec is a point-to-point approach, rather than being a network-wide holistic architectural solution. From a routing standpoint, the current Internet provides a simple and network-centric packet forwarding function by only referring to the destination address of the packet where packets are forwarded in the shortest-path manner. However, in the future Internet, it is strongly desirable to have user- or service-centric routing capabilities to achieve service-oriented traffic controls. Additionally, in the future Internet, to handle various network services’ flexibilities and dynamics, routing will be required to be more flexible and have fine-grained flow controls based on a policy while addressing trustworthiness.

Current routing policies provide a functionality on limiting traffic to control what may be allowed. Border Gateway Protocol (BGP) made policy decisions at the AS level so that the AS has the control on redistributing routing information. Another protocol called Inter-Domain Routing Protocol (IDRP) [31] supports access restriction through policies to deny particular traffic transitions. Both BGP and IDRP use path-vector routing, while Inter-Domain Policy Routing (IDPR) [43, 42] uses link-state routing to distribute routing policies over managed domains. Here, the managed domains refer to any collection of contiguous networks [33] including gateways, links, and hosts whose intra-domain routing manner and service restrictions are controlled by a single administrator. IDPR aims to enforce policies to direct traffic under the users’ service requirements, such as bandwidth, acceptable latency, and the paths to avoid. Although those routing policies can provide network resources respecting users’ requirements or restrict traffic to some certain path, a more standard and rigorous access control framework at the edges of a routing domain is needed to ensure that the traffic could be trusted.

Policy-based management on the routing framework that provides secure routing is a significant plus on realizing trustworthiness in a virtualized network environment. Having se-
curity and trust specification policies enforces access control at the
domain level to filter out anomaly traffic so that more secure
and better services can be guaranteed from the users’ perspec-
tive. An adaptive policy-based routing management framework
is able to select a managed domain with a proper trust level to
direct transit traffic by reviewing the historical behavior of the
traffic and the users’ service requirements. Note that the traffic
behavior refers to the impact on the network performance.

Thus, a flexible, scalable, and robust routing framework that
enables fine-grained flow control under fixed or dynamic poli-
cies while addressing trustworthiness is a desirable goal. Fur-
thermore, the level of trustworthiness may possibly be differ-
ent from one network to another, which is also important to
allow coexistence. To support flexible traffic control according to
various service or user requirements, diversification of rout-
ning functionality is also desirable. For this, the network should
have virtualization and slicing capabilities according to con-
trol policies (for example, for trustworthiness) in place, but that
may change dynamically. Delivery of critical traffic in a secure
manner by using an isolated slice from other traffic and which
is controlled independently should be possible. In addition,
the integrated routing framework should support both user- or
service-centric traffic controls and provide secured communication with
differentiated security requirements. These need to also warrant
that the network programmability is a desirable functionality.

Considering the above needs, we present the SeRViTR (Se-
cure and Resilient Virtual Trust Routing) approach. This com-
prehensive paper is built on our earlier conference and work-
shop papers [26, 32, 49]. SeRViTR encompasses three aspects: 1)
a framework for a future secure Internet that addresses trust-
worthiness at the network level, 2) a discussion on how this
is implemented, and 3) a geo-distributed testbed that connects
three universities between the US and Japan where this func-
tionality is being tested. The core framework builds on the
notion of a Virtual Trusted Routing and Provisioning Domain
(VTRouPD) concept, at both the network level and the service
level. A VTRouPD is constructed by a collection of networking
resources including routers and switches based on virtualiza-
tion techniques. Within one or spanning multiple VTRouPDs,
we can further create user-centric virtual routing domains that
are denoted as µVTRouPDs.

The paper is organized as follows. We first describe related
works in Section 2. We present the scope and the goal of
SeRViTR in Section 3. Our design model of SeRViTR with
descriptions on detailed operations of key components is pre-
scribed in Section 4 and Section 5. We next discuss the geo-
distributed programmable Layer-2 network environment for the
SeRViTR experiment deployment in Section 6 and Section 7. In
Section 8, we will briefly introduce the next phase on improving
SeRViTR features. We conclude this paper with future research
topics in Section 9.

2. Related Work

Research on the future Internet has been active for years.
There are several projects exploring the future Internet infras-
structure to provide a large scale programmable networking
testbed. GENI [46, 18], Global Environment for Network In-
novations, is a program exploring the future global network-
ing infrastructure in the United States, where different types of
resource provisioning platforms resides. GENI platforms
such as PlanetLab [6], ProtoGENI [7], and OpenFlow Net-
works [34] have different concentrations in terms of provision-
ing resources, network architecture, programmable networks,
and so on. For example, ProtoGENI has integrated a large
group of resources available from the world to provide re-
sources with network programmability and sensing features.
All GENI-related projects [19, 20, 38, 41, 17, 44, 35] are sum-
marized in Table 1. In particular, DETERLab [17] is a public
Emulab-based cyber security research testbed, which supports
traffic generation, attack generation, and data analysis capabil-
ities [40]; whereas, Seattle [20] has an efficient design that can
easily make spare nodes join their available resource pool to be
further utilized to provide Python based experiments.

In Europe, FIRE [2] program is an initiative on the Future
Internet architecture design. OneLab [4] is a GENI-like testbed
that supports research on the future Internet, as well as the fed-
eration between networking research testbeds, in order to es-
tablish the international relationship with the future Internet
researches in other countries around the world. It is known that
GENI has deployed large-scale OpenFlow Networks in the
US, and OFELIA [3, 45] is the first large-scale programmable
OpenFlow Network research environment in Europe. It sup-
ports network virtualization capability, new controller testing
and customization. Another related effort is Bonfire [1].

Network virtualization has been actively involved in the fu-
ture Internet research for several years. A recent survey [23]
states network virtualization may occur at four different layers:
the physical layer, link layer, network layer, and application
layer. For example, PlanetLab [6] is an application-layer virtu-
alization, whereas VINI [10] and VNET [11] are network and
link-layer virtualizations, respectively. In [22], a customized
VINI framework for network virtualization in GpENI [44, 35]
has been presented. With virtualization techniques, various new
services arise such as network management and resource man-
agement. A programmable hardware platform to construct vir-
tual data planes that focus on hardware implementation can be
found in [15]. VROOM [47] (Virtual ROuters On the Move) is a
recent network management primitive that avoids unnecessary
changes to the logical topology by allowing (virtual) routers
freely move from one physical node to another. The work
presented in [27] focuses on the accountability in a virtualized
hosting environment. [36] introduces a policy-based resource
management function into a virtual network environment based
on a two-phase resource distributive model. [21] is an early
work on trust/admission control in software-defined networks.

3. SeRViTR: The Goal and The Scope

SeRViTR address an architectural approach with a built-in
trustworthy network level functionality. While there have been
a number of approaches on testbed and new architecture for the
future Internet as highlighted in Section 2, there is none that
addresses this scope. As we briefly discussed in the Introduction, an approach such as IPSec is not an architectural solution; it only provides a tunnel level functionality for encrypted information if viewed from an architectural perspective.

Fundamentally, if we wish to think about whether a unit of information is trustworthy, we can go to the packet level, which serves as the atomic unit of communication. At the packet level, an elementary sanity check can be performed such as through the checksum function. On the other hand, trying to do a check on each packet that enters a router to see whether it is trustworthy (analogous to checksum checking) is not a pragmatic choice due to the high packet processing cost for checking trustworthiness; in addition, there are additional systems functionalities needed if we have to push different trustworthy levels to all routers much like a routing protocol does in pushing routing table information to the routers. This then raises the issue of how and where we can fundamentally inject trustworthiness. Certainly, another possible place is at the flow level on a flow by flow basis. This is still not a complete architectural solution as this is not being addressed at the network level. Thus, we propose to build trustworthiness at the network level where different networks (both physical or virtual) may have different levels of trustworthiness. Note that this may, in turn, force some flow level functionality but not the other way around. Using this basic premise, we set out to design SeRViTR.

To briefly summarize, our goal is to first build multiple virtualized routing domains through a comprehensive approach by using logical virtual routers to partition the physical networking environment into multiple virtual networks while having trustworthiness as an intrinsic property. Second, we aim to provide a secure inter/intra-routing domain that ensures safe end-to-end services. Therefore, we provide a fine-grained virtualization for user-centric networking services by allowing to set security policies. Third, we design a mechanism to dynamically negotiate trust levels between domains so that it is possible to differentiate applications or services behaving abnormally. We discuss how these aspects come together as a framework and are implemented in our SeRViTR approach. We also present G-PLaNE, consisting of multiple clusters that are geographically distributed, and serves as the testbed platform for SeRViTR. As a proof of concept testbed, we have been able to create a Layer-2 virtualization among multiple virtual domains and across geographically distributed clusters to achieve intra and inter-domain communications in the SeRViTR framework by using the OpenFlow switch and XEN.

To further elaborate, trustworthiness is a fundamental architectural requirement in SeRViTR. We posit that trustworthiness is needed to be realized for a trustable future Internet as the needs and the requirements may vary based on different viewpoints from end users, to network operators, or to service providers. Consider the following:

- From the viewpoint of users, trustworthiness of communication is based on the degree to which users can trust a communication peer. Generally, trustworthiness is often related to the importance of the information to be communicated. For example, none of the end users may like a situation where their private information is delivered over the untrusted network. To achieve this, the network should be isolated based on the degree of trustworthiness, and the communications in the lower trustworthiness network should be more restricted;
- From the viewpoint of the network, trustworthiness of the network is determined by the condition in which routing in the network is secured and safe. That is, any routing information (e.g., routing table exchanges, signalling traffic, MIB) must be confidential, secured, and protected;
- From the viewpoint of service providers, trustworthiness of services is a situation where the service is protected, secured, and exclusive of anonymous users. Not only users but also the paths to users, might need to be managed by a service provider.

To realize the above stated diverse needs in regard to trustworthiness, adding a special routing layer is not sufficient. A network should have the capability to be sliced, isolated from other networks, managed by an integrated security policy, and have a secure routing protocol along with flexible traffic engineering. Our aim with SeRViTR is to present a proof-of-concept model to achieve these goals and to demonstrate it through a testbed.

<table>
<thead>
<tr>
<th>Project</th>
<th>Major Resource</th>
<th>Programmable Networks</th>
<th>Extension Simplicity</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlanetLab [6]</td>
<td>Fedora VM</td>
<td>No</td>
<td>Difficult</td>
<td>Access Control (PKI)</td>
</tr>
<tr>
<td>ProtoGENI [7]</td>
<td>PC and VM</td>
<td>Yes</td>
<td>Difficult</td>
<td>Access Control (PKI)</td>
</tr>
<tr>
<td>OpenFlow Networks [34]</td>
<td>OF Switch</td>
<td>Yes</td>
<td>DD</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>GENICloud [19]</td>
<td>Physical node</td>
<td>No</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>OFELIA [3, 45]</td>
<td>Physical node</td>
<td>No</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>Seattle</td>
<td>Experimenter Software</td>
<td>No</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>DETERlab [17]</td>
<td>PC</td>
<td>Yes</td>
<td>NA</td>
<td>Set proper operation mode for different threaten level of security experiments; ABAC</td>
</tr>
<tr>
<td>GpENI [44, 35]</td>
<td>Fedora VM</td>
<td>Yes</td>
<td>DD</td>
<td>Access Control (PKI)</td>
</tr>
</tbody>
</table>

DD: Dedicated Device, NA: Not Allowed, PKI: Public Key Infrastructure, ABAC: Attribute Based Access Control
4. SeRViTR: Overview, Models, and Specification

The SeRViTR framework is designed on the conceptual model presented in Fig. 1, in order to realize the Virtual Trust Routing and Provisioning Domain (VTrouPD) and construct user-centric VTrouPD as so-called µVTrouPD. In this section, we first introduce the conceptual model of the VTrouPD and the µVTrouPD, and then we describe the SeRViTR framework design to achieve a trusted routing procedure in a virtualized routing environment.

4.1. VTrouPD Conceptual Model

The entire network can be divided into multiple routing/provisioning domains, and we refer to every such domain as a Virtual Trust Routing and Provisioning Domain (VTrouPD). Fig. 1 presents a conceptual model design on the VTrouPD. In each VTrouPD, we refer to routers or switches as a generic Routing Service Node (RSN), which can host one more multiple Router Instances residing in different user-centric routing domains. A Node Manager (NM) is responsible for managing the Router Instances’ loading and unloading in the RSN.

For a trustable environment, Trust Management Service is the trust authority for the system. It handles the cryptographic key and the parameters’ distribution and revocation. It also provides identity search and federation services for RSNs within different administrative domains, as well as policy checking and enforcement functions to provide a unified trust management system. The Resource and Application Manager (RAM) is the resource manager directed by the VTrouPD manager and TMS to construct VTrouPDs. Furthermore, from a system’s perspective, traffic management and network resource management components are necessary for monitoring and managing a network. Thus, we add two more components when designing the SeRViTR framework, which are Flow Controller and Traffic Monitor. We will discuss SeRViTR components in detail in the next subsection.

4.2. VTrouPD and µVTrouPD

To establish VTrouPDs, we require network routers to be programmable, i.e., we should be able to create multiple virtual routers on the same physical router, and each virtual router is responsible for a particular virtual domain. We consider two levels of approaches to deploy the virtual routers. At the first level, we create independent virtual router images, where each virtual router has its own protocol stack and independent IP addresses associated with virtualized interfaces. In this way, traffic can be differentiated at the IP layer through routing functions. For example, the routing table in each virtual router can mark the next hop. However, there is a certain roadblock since the router’s hardware configuration restricts the number of virtual routers on each router. This restriction usually makes it impractical to support a large number of virtual routers running on the same router that further restricts the number of supported virtual domains. Second, it is difficult to allow inter-VTrouPD traffic when services provided by two VTrouPDs are highly correlated and may need to merge partial traffic from two VTrouPDs. Thus, the approach using the entire routing
function virtualization is only appropriate for network-centric virtualization.

To address the restriction of network-centric routing virtualization, we introduce a fine-grained routing virtualization at the second level that can take care of the users’ service level demands. To this end, a complementary concept, virtual trust routing and provisioning sub-domain (noted as µVTRouPD) is introduced. A µVTRouPD may contain multiple µVTRouPDs. In other words, a µVTRouPD is a sub-domain consisting of a collection of virtual routers under a certain routing policy, and it can be constructed either within a single VTRouPD or by spanning multiple VTRouPDs. To explain the latter case, we define a µVTRouPD as µ, and there are two VTRouPDs, V₁ and V₂, where both V₁ and V₂ have a set of virtual sub-domains. A spanning µVTRouPD means the µ does not belong to either V₁ or V₂, but µ belongs to V₁ ∪ V₂. In general, an administrative domain consists of a number of VTRouPDs, V₁,..., Vₙ. Each VTRouPD has a different number of virtual sub-domains Vᵢₗ, where i ∈ [1, n], and m is used to differentiate the virtual sub-domains by the policies. A spanning µVTRouPD can be represented as a collection of virtual sub-domains under a certain policy over these n VTRouPDs, {V₁₁, V₂₂, ..., Vₙₙ}, where j ∈ [1, m].

Compared with VTRouPDs, the boundary of µVTRouPDs is not established through virtualizing router functions; instead, we perform virtualization of µVTRouPDs through the following techniques:

- Use of cryptographic packet marking techniques to allow each virtual router to recognize the traffic flows for different µVTRouPDs;
- Invocation of efficient secure group communication solutions to isolate traffic flows that belong to different µVTRouPDs. This approach provides us a fine-grained traffic management and filtering capability to identify malicious traffic and reduce the impact to other services when the malicious traffic flows are blocked. Moreover, through secure group-based communication, we can further merge or diverge traffic belonging to different µVTRouPDs through super- or sub-group communications;
- Use of an efficient security data access control solution that provides trust party verification, traffic access control, and data privacy protection for µVTRouPDs. This capability allows us to provide versatile user-centric network routing services with assured data access policy enforcement and privacy protection.

4.3. Realizing VTRouPD in SeRViTR

The conceptual VTRouPD model discussed above is proposed from the perspective of a high-level design. In order to realize the concept described in the VTRouPD model, we designed a comprehensive architecture called Secure and Resilient Virtual Trust Routing (SeRViTR) that crystallizes the components defined in the VTRouPD model. For this, we designed functional components at the implementation level to realize the roles of both TMS and the VTRouPD Manager. We also define the exchange messages for communication between these functional components (see the Appendix). Table 2 presents the VTRouPD components and their roles in the SeRViTR framework, and Table 4 summarizes terminologies for the key fields defined in the message types. These notations will be used through the rest of this paper. It should be noted that to realize VTRouPD, there are two additional components needed in ScRViTR; they are: 1) the flow controller, and 2) the traffic monitor.

A Routing Service Node (RSN) is a physical programmable router within the VTRouPD, and it is in charge of forwarding packets to a specific Virtual Domain. A Routing Service Node forwards packets to the next or Flow Controller by modifying the ForwardingID denoted in the routing table.

4.4. SeRViTR Components

In this part, we illustrate the role of other SeRViTR components, and the interactive relation between one another. Table 2 shows the roles of conceptual VTRouPD elements in the design of the SeRViTR framework, as shown in Fig. 2.

4.4.1. Trust Management Service (TMS)

Trust Management Service is a key service in SeRViTR. We accomplish Trust Management Service through three functional components: Policy Manager, Authentication Manager, and Trust Level Regulator.

A Policy Manager is associated with each VTRouPD. It maintains three tables: the Rule-set Table, the Trust-level Table, and the Virtual Domain Table. The role of the Policy Manager is many-fold, which is depicted in Fig. 3:

- It enforces policies that are assigned by the Administrator
- It is in charge of policy management. We use XACML (eXtensible Access Control Markup Language)[12] to describe the rules, and each rule is identified by <FlowID, Trust>. In order to ensure security, we use a unique 16-bit integer to identify the Trust based on the policy.
- It announces the request for the creation or deletion of Virtual Domains to the VTRouPD Manager. When a new TrustID is obtained, the Policy Manager generates a request and sends it to the VTRouPD Manager through the Virtual Domain Management Message (Fig. 13(a)).
- It sends a Flow Table Update Message (Fig. 13(b)) to the Flow Controller that manages the flow table, informing how incoming packets (flows) should be processed at the Flow Controller.
- It plays a role in negotiating the trust level between managed domains. To do so, it creates an OutboundDomainID, and communicates with the Trust Level Regulator about the Outbound Domain ID Notification.

The Trust Level Regulator behaves as a trust gateway in each VTRouPD through which the trust level is established,
changed, and updated between two VTRouPDs. It relays the Outbound Domain ID notification that is sent from the Policy Manager to other VTRouPD’s Trust Level Regulators.

The Authentication Server is responsible for the generation and distribution of authentication keys, and it manages the HostID and Secret where the HostID indicates an IP address or a user name, and Secret is the password or certificate.

4.4.2. VTRouPD Manager

A VTRouPD Manager manages the information of physical routers within the VTRouPD and it is responsible for the creation or deletion of the VirtualDomainID, as well as resource management in terms of resource information from the Routing Service Nodes. The VTRouPD Manager maintains a Virtual Domain Management Table that stores information of the Virtual Domains, the RSNIDs, and the Resource Information. In order to create Virtual Domains, the VTRouPD Manager assigns a VirtualDomainID and inserts it into the Virtual Domain Management Table. Similarly, in order to delete a Virtual Domain, the VTRouPD Manager deletes the entry from the Virtual Domain Management Table.

For the resource management, the VTRouPD Manager obtains resource information such as bandwidth from the Routing Service Node. The VTRouPD Manager sends the Routing Table Update Message (Fig. 13(c)) to the Routing Service Node to update its routing table. Particularly, the ForwardingID is set as the Layer-2 ID in our implementation.

4.4.3. Flow Controller and Traffic Monitor

The Flow Controller is a component in SeRViTR that is placed at the edge of the VTRouPD. A Flow Controller forwards flows to the appropriate Virtual Domains based on a given policy. For any outgoing packet, the ForwardingID is removed from a data packet; while for any incoming packet, a ForwardingID is attached according to the Flow Table. Also, a Flow Controller encrypts the incoming packet and decrypts the outgoing packet according to the flow table. The Flow Controller updates the flow table based on the Flow Table Update Message.

The Traffic Monitor is used to monitor anomaly behaviors of the flows at the ingress routers. If anomaly traffic is detected,
it communicates the flow information to the Trust Level Regulator. We assume that there is a separate engine for identifying anomaly traffic.

5. Policy and Trust Management in SeRViTR

We now present the enabling techniques to establish VTRouPDs with a systematic approach. We start with a description of the challenges in the policy and trust management, and we then discuss our approach.

5.1. Challenges for Policy and Trust Management

There are limits on creating virtual routers on a physical programmable router (refer to Section 4.2). When using the routing function alone, virtualization cannot fully meet the requirement of merging traffic from two Virtual Domains at the edge of one VTRouPD and send it to another. To address these issues, a flexible and lightweight virtual routing policy management and enforcement mechanism is required. On the other hand, the trust management is managed independently in different VTRouPDs. Hence, different administrative domains can have different compositions of VTRouPDs. To federate trust management among VTRouPDs created by different administrative domains, we need to construct a trust negotiation system to address the incurred inconsistency and incompatibility issues.

The second critical trust management issue is how to initiate trust among routers (and virtual routers). To address this, a reputation based approach can be used. Trust Management Service, which involves the Trust Level Regulator and the Policy Manager, can collect feedback from the system to rank the trust of a router, Virtual Domain, and the VTRouPD, and in turn, they can calculate trust ranking and provide a recommended trust level for the corresponding party at the initial trust level. The trust level can be measured using metrics such as the percentage of good traffic transited, the reliability of the routing system, trust levels of ingress and egress neighboring domains, and so on.

5.2. Trust Management Service Sequence Diagram

Considering the above challenges, we present sequence diagrams for a number of situations of VTRouPD Trust Management Services.

5.2.1. Policy Setting

Policy setting involves a series of steps that includes creating Virtual Domains and updating routing tables and flow tables. The sequence diagram for policy setting is shown in Fig. 4. As we can see from this figure, the Administrator inputs a policy and on receiving it, the Policy Manager assigns a FlowID and unique TrustID. As we mentioned earlier, once the Policy Manager sets the TrustID and there is no VirtualDomainID that corresponds to the TrustID in the Virtual Domain Table at the Policy Manager, it generates and sends a request to create a Virtual Domain to the VTRouPD Manager through the Virtual Domain Management Message. In turn, the VTRouPD Manager will assign the VirtualDomainID that is used for updating the Virtual Domain Management Table and sends Routing Table Update Messages to the Routing Service Node. The Routing Service Node replies with an Update Complete Notification and the VTRouPD Manager sends a response back to the Policy Manager. Next, the Policy Manager sends the Flow Table Update Message to the Flow Controller that responds with an Update Complete Notification to indicate that it has already updated the routing policies. Finally, the Policy Manager notifies the Administrator that the policy setting is completed.

5.2.2. Outbound Trust Level Notification

Two VTRouPDs can negotiate trust levels through their own Trust Level Regulators. The sequence diagram for outbound trust level notification is shown in Fig. 5. In this case, the Policy Manager first creates the OutboundDomainID list, and then notifies the Trust Level Regulator with the OutboundDomainID list. In turn, the Trust Level Regulator is responsible for communication with another Trust Level Regulator associated with a corresponding NotifyDomainID. Once this is received by the other Trust Level Regulator, it first verifies the validation. If the NotifyDomainID is acceptable, the Trust Level Regulator communicates with its own Policy Manager about the NotifyDomainID, so that data communication is possible between these two VTRouPDs based on the established level of trust.
5.2.3. Trust Level Change Notification

Periodically, there is a requirement to communicate a trust level change within a Managed Domain in regard to communication with another VTRouPD. The sequence diagram for such a change is shown in Fig. 6. In this case, when the other VTRouPD’s Trust Level Regulator is aware of the trust level change, it communicates the change to the first VTRouPD’s Trust Level Regulator. Then, internally, this change is notified to its Policy Manager, which in turn, informs its Traffic Monitor to request detecting these changes.

6. SeRViTR Functionality Implementation

SeRViTR introduces a mechanism to forward flows to the corresponding µVTRouPD according to their trust level, and the µVTRouPD creation based on the trust level is the critical part in the SeRViTR functionality deployment. In this section, we discuss its implementation on the Policy Manager, VTRouPD Manager, Flow Controller, and Routing Service Node, followed by a multiple domain scenario.

6.1. Implementation Platform

The SeRViTR implementation is OpenFlow based. The VTRouPD Manager is based on NOX 0.9.1 on an Ubuntu 10.10 platform where each function is implemented in C++. As we described before in section 7.3, we chose to use an OpenFlow switch over NetFPGA to achieve better performance in terms of handling packets. The Flow Controller and Routing Service Nodes are implemented using NetFPGA 2.1.3 full on CentOS 5.3 where the OpenFlow Switch was installed. As we mentioned earlier, the Flow Controller assigns a different ForwardingID to the flows according to their trust level. When a Virtual Domain is created, the Flow Controller will update its flow table and the Routing Service Node updates its routing table, so that the end systems can communicate with one another.

We have implemented four SeRViTR components, and Fig. 7 presents the work flows between these components. The entire process involves six procedures:

(1) Start the four SeRViTR components: VTRouPD Manager, Routing Service Node, Policy Manager and Flow Controller, successively. Note that since we have not implemented the Trust Level Regulator yet, we presume that the Policy Manager already knew the change of trust level when it created the virtual domains. In our implementation, the Routing Service Node (step 1.b) and Policy Manager (Step 1.c) are started after the VTRouPD Manager (step 1.a), and the Flow Controller (step 1.d) is started after the Policy Manager and under the Policy Manager’s control.

(2) Interconnect the four components (Step 2.a - 2.d) and Host Registration. Note that the step ”Host Registration” is not presented in Fig. 7, but it is necessary for the following procedures. Before two hosts setup communications, they have to register at the Policy Manager within their own VTRouPDs, respectively. According to the type of communication between these two hosts, the Policy Manager determines the trust level for them, in terms of TrustID.

(3) The Policy Manager sends the Virtual Domain Management Message. When the Policy Manager receives the host registration information, it will generate the TrustID and FlowID (step 4). Then it looks up the Virtual Domain Table that maps the TrustID into the corresponding VirtualDomainID. Fig. 7 shows two implementation flows after step 4, which are represented in red and green, respectively. The red flow path indicates a shorter flow path to trigger a flow table update, which requires the Virtual Domain Table to have the TrustID already mapped to a VirtualDomainID. If the TrustID does not exist in the trust table, the Policy Manager will then send the Virtual Domain Management Message to the VTRouPD Manager to request the VirtualDomainID, as shown in steps 5-6 in the longer flow path that is represented in green. The longer path is a normal process for a new flow that arrives at the domain, because there is no priori TrustID-VirtualDomainID mapping pre-created. The next procedure is only for the normal process.

(4) The VTRouPD Manager creates the Routing Table Update Message. As soon as the VTRouPD Manager heard messages from the Routing Service Nodes, it decides members for each virtual domain, as shown by step 7 (7-a & 7-b) with both red and green paths. When the VTRouPD Manager has decided the VirtualDomainID, it will send a Routing Table Update Message to all the Routing Service Nodes in the VTRouPD to request a routing table update (steps 8-11). At the same time, the VTRouPD Manager also sends the new created VirtualDomainID to the Policy Manager (steps 12-13).

(5) The Policy Manager sends the Flow Table Management Message. Once the Policy Manager receives the newly assigned VirtualDomainID from the VTRouPD Manager, it will first update its Virtual Domain Table, and then sends a Flow Table Management Message to the Flow Controller to update the flow table.

(6) The flow table and routing table get updated.

6.2. Testing and Validation

We setup a connection between two end-hosts that reside in two different VTRouPDs as shown in Fig. 8. In this scenario, our goal is to validate the work flows between the four components we have discussed above. With each VTRouPD
setup, there are two Flow Controllers, at the ingress and egress points, respectively; a VTRouPD Manager is connected to three Routing Service Nodes; a Policy Manager is connected to both Flow Controllers and the VTRouPD Manager. We considered two different types of applications: SSH and HTTP, where the HTTP flow is set to the lowest trust level and the SSH flow is assigned at the highest trust level.

In VTRouPD A, before sending out SSH and HTTP packets, the end-host PC1 needs to register at the Policy Manager PM1 with the information about the type of flows. For example, providing the port numbers (i.e., SSH: 22, HTTP: 80). Based on the application type, PM1 uses pre-determined policy rule \(<\text{FlowID}, \text{Trust}\rangle\) to generate TrustIDs for both types of applications. By querying its Virtual Domain Table \(<\text{TrustID}, \text{VirtualDomainID}\rangle\), PM1 will decide the VirtualDomainID and request the Flow Controller FC1 and FC2 to
update their flow tables. Note that the VirtualDomainID can be used as the ForwardingID attached to the packet sent from the Flow Controller (i.e., packet with ForwardingID 11, 12, 13 sent by FC1 in VTRouPD A).

In VTRouPD B, the end-host PC2 has to register at Policy Manager PM2. The two Flow Controllers FC3 and FC4 will update their flow tables via the same procedure. Once the VirtualDomainID has been determined for each application, the VTRouPD Managers will request all Routing Service Nodes to update their routing tables.

The second validation is the degradation of the trust level when anomaly traffic is detected in the flow. In our current experimental setup, we assume that there is anomaly traffic mixed in the SSH flow that is detected in VTRouPD A. Because we have not implemented the Trust Level Regulator yet, we manually change the TrustID for the application at PM2, and in turn, the flow tables at FC3 and FC4 are updated. Then, when receiving new SSH packets from VTRouPD A, FC3 attaches a different ForwardingID and forwards the packets to a different Virtual Domain that is under a lower trust level. The degraded SSH flow is represented as a dotted arrow in Fig. 8.

6.3. Flow Table Size

In this section, we discuss the size of the flow table at the Flow Controller, in terms of the number of entries in the table. Fig. 9 gives a few typical entries in a flow table. It shows that there are two entries for the ARP protocol, and the rest of the entries are for the end-to-end communication. The first two entries in Fig. 9 give an overview on the basic elements in the complete flow table entries. In our design, we append one more parameter to a standard flow table entry; this parameter has two optional values: "mod_ForwardingID_vid" and "strip_ForwardingID". When a Flow Controller receives a packet from outside of a VTRouPD, the "mod_ForwardingID_vid" option is used in the flow entry with the ForwardingID tag that will be attached to the packet, indicating the Virtual Domain to where the packet has been forwarded. When a Flow Controller forwards a packet out of the VTRouPD, the "strip_ForwardingID" is attached to the flow entry indicating that the ForwardingID tag is removed from the packet. For each host that has registered at the Policy Manager, two flow entries will be created in the flow table for each flow under a particular trust level; here, "two" refers to bi-directional communication.

Table 3 summarizes the parameters that affect the size of a flow table at the Flow Controller. The total number of entries in a flow table is

\[ n_{\text{ARP}} + n_{\text{mod}} + n_{\text{strip}} = n_{\text{arp}} + n_{\text{host}} + 2 \times n_{\text{flow}} + n_{\text{trust}} \]

7. SeRViTR Testbed: G-PLaNE

So far, we discussed SeRViTR components’ implementation from the functional perspective, and our next goal was to deploy the SeRViTR testbed onto a real geo-distributed environment. To support the SeRViTR testbed, we designed the Geo-distributed Programmable Layer-2 Networking Environment (G-PLaNE).

G-PLaNE is established for constructing the SeRViTR system for the following reasons: 1) it is Geo-distributed so that multiple virtual domains can be supported; 2) its model is infrastructure-as-a-service so that lower level resources can be provided to enable the SeRViTR function with more capability; 3) it has a network programmability feature so that the system build on top of it can be developed with an additional network programming capability.

The G-PLaNE currently involves three geographically distributed sites. They are Osaka City University (OCU) in Japan, Arizona State University (ASU) and the University of Missouri–Kansas City (UMKC) in the United States. Thus, the SeRViTR testbed environment consists of three VTRouPDs, one located at each site, respectively. A Trust Level Regulator among these domains will communicate to establish the appropriate trust levels. We started with building the experimental environment and implementing SeRViTR components concurrently. On the one hand, we are building tunnels among these three sites, and we have built GRE tunnels through OpenFlow Switches. On the other hand, we implemented the VTRouPD Manager, the Flow Controller, and the Routing Service Nodes in a two-VTRouPD environment over two sites.

7.1. Geo-Distributed Programmable Layer-2 Networking Environment (G-PLaNE)

G-PLaNE is designed to provide networking, computing, and storage capabilities for terminals that usually have limited resources and capabilities. The system component and architecture can be seen in Fig. 10. We present here a brief description on related components for constructing SeRViTR; for example, components such as storage are not discussed as this is not directly relevant to SeRViTR.

7.1.1. System Components

We partition the G-PLaNE system into a number of components as follows:

- Computing Component: Computing capability is the major provisioning service that the majority of resource provisioning platforms provide. We use Xen [37] to maximally utilize the resource pool (physical XenServers) by creating multiple virtual machines (VMs). With virtualization and programmability enabled, G-PLaNE can provide logically separate resources for end users in terms of a general routing suite, an OpenFlow switch/router, and controllers. A resource pool always has at least one physical

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n_{\text{arp}} )</td>
<td>The number of ARP flow entries, constant value = 2</td>
</tr>
<tr>
<td>( n_{\text{host}} )</td>
<td>The number of hosts registered at the Policy Manager</td>
</tr>
<tr>
<td>( n_{\text{trust}} )</td>
<td>The number of TrustID has been created</td>
</tr>
<tr>
<td>( n_{\text{flow}} )</td>
<td>The number of flows under different trust levels</td>
</tr>
</tbody>
</table>
node, known as the master. Other physical nodes join the existing pool and are described as slaves. Only the master node exposes an administration interface and forwards commands to individual slaves as necessary.

- Administrative Component: We also introduce dedicated management and monitoring servers to administrate the VMs and network resources in the resource pool and monitor network traffic within and across domains. NetFlow [24] and sFlow [9] are both enabled to inspect Layer-2 and Layer-3 networking as well as host performance (i.e., CPU and memory utilization). There is also a set of internal functional servers serving different administrative purposes, i.e., Web server, DHCP, DNS, Authentication Server, DB server, and VPN.

7.1.2. Network Architecture

In the system component discussion above, virtual machines (VMs) are critical when deploying a testbed on G-PLaNE, because they can be deployed into various functional components. In this section, we describe how these VMs communicate with each other in an intra/inter-cluster environment with the network architecture design. Details on how we use VMs in the SeRViTR project will be discussed later in Section 7.3.

In the G-PLaNE system, the data plane and the control plane are isolated. The management network (Control Plane) is for management and control traffic (i.e., the traffic of the service request, downloading applications from our repository and so on). On the other hand, the data network is for data traffic among different VMs, or different terminals via VMs. From Fig. 10, there are 4 networks in each cluster. The incoming and outgoing traffic switches isolate the traffic going out of or coming into the G-PLaNE domain. With this design, we can easily control the privilege of resources accessing the Internet, which enhances the security of the resource network environment.

The communications between VMs go through the data network. The data network switch is a managed switch with VLAN support that enables different VMs being in different virtual domains. Additionally, the G-PLaNE management network connects the internal NetFlow and sFlow monitoring systems to dynamically monitor the network performance by an administrator. Not only is VM-to-VM communication in one physical cluster considered, but also those VMs located at different clusters are considered. For this reason, the OpenFlow switch [34] has been introduced to establish the inter-domain data link. To increase efficiency and security, each G-PLaNE server is installed with Open vSwitch [5] with which the traffic between two VMs in the same physical server does not need to go through the physical data network switch so that it is exposed in public. The detail of this dual switch design is further explained in Section 7.2.

7.1.3. Network Programmability

Both the OpenFlow switch (OFS) and Open vSwitch (OVS) are OpenFlow-based switches. In OpenFlow architecture, a controller executes all control tasks of the switches and also those used for deploying new networking frameworks, such as new routing protocols or optimized cross layer packet-switching algorithms. With these features, a programmable network is established to provide network programmability for Cloud providers. It is feasible to develop a tenant-based policy or protocol to control both internal OVS and external OFS in a virtual networking environment. There are several OFS controllers available following the OpenFlow standard, such as Onix [30], SNAC [13], and NOX [25]. OVS, as well as the controllers, can be easily deployed on VMs since they are software-based.
With the dynamic resource provisioning mechanism supported, users are able to request a dedicated private virtual OpenFlow network upon G-PLaNE and develop their own network topology and control mechanism. A virtual network is created from several templates including the software based OpenFlow switch and different controllers pre-installed in the VM. A user can easily turn the claimed general virtual network into an OpenFlow-based programmable virtual network by enabling some pre-installed functions. Besides this OpenFlow based switch/control model, there are also all-in-one routing suites; for example, the Quagga Routing Suite [8] can be deployed into the virtual network upon which users can develop their research and experiments.

7.2. Virtual Network Construction

We chose a geo-distributed architecture to support resource provisioning over multiple clusters. The resource network can be created by different configurations due to different requirements: 1) a single physical server, 2) multiple servers within one cluster (servers in the same cluster are connected through a Layer-2 physical switch), and 3) multiple servers belonging to different clusters.

7.2.1. Intra-Cluster Network Creation

Intra-Cluster means there is always a native Layer-2 connection among all resources within the same cluster. To create a virtual network within the same cluster, VLAN technology is deployed. As we previously mentioned, it is inefficient to forward packets through the managed switch from one VM to another one in the same physical resource provisioning server. Therefore, each XenServer has an internal Open vSwitch enabled to handle traffic inside the physical server as shown in Fig. 11.

Open vSwitch is designed to enable massive network automation through programmatic extensions, while still supporting standard management interfaces and protocols (e.g., NetFlow, sFlow, RSPAN, ERSpan, CLI, LACP, 802.1ag). Open vSwitch can operate as a software-based switch running within the hypervisor (Xen Dom 0) in which many security control functions can be implemented. With Open vSwitch enabled, a packet sent from one VM to another one within the same physical server does not need to be exposed out of the physical box.

When a virtual network is created within the same cluster but across different physical servers, a packet sent from one VM to another one on a different server should go through the physically managed switch by enabling trunk ports. The virtual network containing multiple VMs in different physical servers is simply created by assigning the same VLAN ID so that it is virtually isolated from other resources.

7.2.2. Inter-Cluster Network Creation

To enable provisioning of a virtual network across clusters in G-PLaNE, we establish Layer-2 GRE tunnels among three sites. After a Layer-2 tunnel is established, VLAN can function well upon a Layer-2 tunnel since it is a 2.5 layer technology, strictly speaking. Although there are some options to establish the Layer-2 tunnel, we chose the OpenFlow solution since it is user-centric and can be easily extended due to its programmability. OpenFlow is an open standard that enables researchers to run experimental protocols. In a classical router or switch, the fast packet forwarding (data path) and the high level routing decisions (control path) occur on the same device. An OpenFlow switch separates these two functions. The data path portion still resides on the switch, while high-level routing decisions are moved to a separate controller, typically a standard server. The OpenFlow switch and controller communicate via the OpenFlow protocol, which defines messages, such as packet-received, send-packet-out, modify-forwarding-table, and get-stats. The data path of an OpenFlow switch presents a clean flow table abstraction; each flow table entry contains a set of packet fields to match, and an action (such as send-out-port, modify-field, or drop).

7.3. Deploying SeRViTR on the G-PLaNE

SeRViTR is deployed in a virtual networking environment that is supported by the G-PLaNE. The µVTRouPD is a vital constituent part in the SeRViTR architecture and it requires isolation as well as scalability when constructing virtual networks. With the G-PLaNE system data network switch, which is VLAN supported, VMs can be grouped into different Virtual Domains by tagging VLAN IDs, and the ones which have been used can be queried through the database of the G-PLaNE system. Fig. 11 shows a high level Virtual Domain creation by grouping VMs into distinct VLANs. Particularly, consider Cluster A at ASU in Fig. 12 where one XenServer is reserved from the resource pool for creating SeRViTR functional managers. Regarding the virtual routers, we may also choose to reserve XenServers from the resource pool and customize an arbitrary number of VMs as dedicated virtual routers by deploying a routing suite (i.e., OpenFlow switch, Quagga, and so on) on it. As a second option, we can directly use a physical OpenFlow Switch upon a dedicated OpenFlow enabled switch or NetFPGA and use VLAN tags to achieve isolation between Virtual Domains.

We first deployed a single VTRouPD connecting three sites, ASU, UMKC, and OCU on top of the G-PLaNE. OCU manages the control units, namely, Policy Manager and Domain Controller, while ASU and UMKC deploy one Flow Controller and one Routing Service Node at each site, respectively.
The G-PLaNE allows the clusters to be scalable. Therefore, we were able to construct multiple VTRouPDs over geodistributed locations. For example, take the SeRViTR clusters at ASU and UMKC presented in Fig. 12; here, two VTRouPDs were created. Within the cluster at each site, the G-PLaNE resource pool contains physical XenServers where VMs are created. Recall that all VMs can be created and deployed as any form of functional entity; thus, we can customize VMs as dedicated SeRViTR functional components as well as virtual routers. Particularly, VTRouPD Managers, and Policy Managers are implemented on VMs created on one XenServer from the resource pool. In our deployment, the Flow Controllers and the Routing Service Nodes are implemented on physical OpenFlow Switches upon NetFPGA.

To establish a geo-distributed multi-domain infrastructure for SeRViTR, we established a Layer-2 GRE tunnel using G-PLaNE, so that any two sites have either a direct or an indirect Layer-2 connection. An OpenFlow Switch was deployed to establish the tunnel. It uses the OpenFlow protocol and also supports various controllers that speak OpenFlow protocol. In this case, a Layer-2 GRE tunnel was chosen so that any Layer-2 above technology, i.e., VLAN, is enabled upon this Layer-2 tunnel. The real entity being tunneled is the virtual bridge that can be attached to any VIF (Virtual Interface) or PIF (Physical Interface). This means that both the physical XenServer or the virtual machine are actually in this tunnel. Using this flexible mechanism to establish the tunnel also guarantees the easiness of future extensions by using OpenFlow protocol. We also tested the delay among all sites over the GRE tunnel. The average latency from UMKC to OCU (Japan) is 219 ms, the average latency from UMKC to ASU is 59ms, and the average latency from OCU to ASU is 278 ms.

8. Discussion and Future Work

So far, we presented SeRViTR, starting from its stop, the framework to the international testbed that we have deployed as a proof-of-concept. We are currently exploring additional features to improve SeRViTR. In this section, we briefly discuss our ongoing work on the extended policy and trust management functionalities at the Policy Manager, because policy and trust management are the core features of the SeRViTR framework.

Note that policy-based management is not new to computer networking research. A common real-world application of policy-based management is the firewall rules, for security authorization. With firewall rules, packets are able to be filtered out based on the source or destination IP addresses and port numbers. Role-Based Access Control (RBAC) [39] Specification is a policy model proposed to define permission according to the roles instead of individual people [16]. The Attribute-Based Access Control (ABAC) [48] proposed by DETERlab adds the attribute concept to the RBAC framework, and its goal is to provide a novel federation framework instead of using the Public Key Infrastructure. In [16], the authors also described a number of security and trust specifications for access control models and various languages that are used for access control policy specifications including Logic-Based Language and Role-based Security Languages. A security policy can also be described in the form of an XML specification, such as using XACML.

SeRViTR is designed to provide a user-centric secure routing domain, so users’ requests on the network resources and trust management should be taken into consideration. Therefore, when designing the new features, we added new input information known as Accounting Status and AccountID. These two terms are used to identify how much a user will pay to request network resources (i.e., bandwidth) for his/her µVTRouPD. On the other hand, Policy Manager will also make a judgement on the behavior of this user’s µVTRouPD to determine whether it...
is safe to keep at the same trust level. Thus, we define Behavior Pattern as a type of flow, application, or alert.

Based on the discussion above, we plan to extend the current protocols in SeRViTR to support the added functions to the Policy Manager. We divide the role of Policy Manager into three parts: Host and Flow Management, Behavior Management, and Resource Management.

- At the input boundary, with Host and Flow Management, the Policy Manager will receive the HOSTID and Accounting Information and calculate the Trust and Bandwidth request for the corresponding flow. At the output boundary, the Policy Manager will send an Flow Table Update Message to the Flow Controller to update the Flow Table. Meanwhile, it will also send an Outbound DomainID Notification to the Trust Level Regulator.

- Behavior Management is an important function since there is a need to periodically update trust level between VTRouPDs. The Policy Manager will receive Behavior Notification and Trust Level Change Notification and make the judgement on the behavior pattern. Here, the behavior pattern refers to the type of flow, application or alert. According to the judgement, the Policy Management will make a decision whether there is a need to adjust the trust level between two VTRouPDs. Together with the Outbound DomainID Notification, it will send the Trust Level Change Notification to the Trust Level Regulator.

- Resource Management refers to making requests to assign the bandwidth for a μVTRouPD. This function is being combined with our current implementation with μVTRouPD creation.

9. Conclusion

In this paper, we presented the design of the SeRViTR framework for the future Internet according to the idea of the VTRouPD from our earlier work. We introduced the role and responsibility of VTRouPD components in the SeRViTR architecture. Specifically, we illustrated the VTRouPD Trust Management Service that is able to setup trust levels for different virtual domains within a VTRouPD, and we negotiated the trustworthiness levels of the flows between VTRouPDs. Moreover, we designed a geo-distributed resource provisioning system called G-PLaNE to support virtual network creation between international research platforms. The G-PLaNE system is discussed in terms of system components, network architectures, and so on. Virtual network creation, as a major service provided by G-PLaNE, is explained from two perspectives, an intra-domain virtual network and an inter-domain virtual network. We deployed SeRViTR architecture on the G-PLaNE system to achieve Layer-2 tunneling between the US and Japan sites, and we validated the virtual domain creation according to the trust levels. From our basic result of implementation, we have been able to create flow-level μVTRouPDs under different trust levels and migrate the flow to the μVTRouPD under a lower trust level given that the anomaly traffic is detected in the flow. In our future work, we will consummate the functionalities of the Policy Manager, by extending the current SeRViTR protocols, to better support user-centric secure routing domains.

Acknowledgments

This work has been supported by a US-Japan collaborative project grant with US NSF Grants CNS-1029562 and CNS-1029546, and Japan NICT International Collaborative Research Grant. It is also partially supported by Office of Naval Research’s (ONR) Young Investigator Program (YIP) grant and an HP IRP grant.

References

of 6th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TridentCom), pages 428–441, Berlin, Germany, May 2010.


Appendix

SeRViTR Message Types

The Trust Management Service (TMS) takes three different roles: Authentication Server, Policy Manager, and Trust Level Regulator, which are discussed in Section 4.4. In Fig. 2, the Policy Manager communicates with every other individual SeRViTR component. In order to implement policy management, we created four message types with their packet formats. They are the main information exchanged between the Policy Manager and other SeRViTR components when setting up routing policies or negotiating trust levels between VTRouPDs.

Virtual Domain Management Message

The Virtual Domain Management Message is exchanged between the Policy Manager and VTRouPD Manager to manage Virtual Domains. The packet format for the Virtual Domain Management message is shown in Fig. 13(a). It has a field to indicate the action (Create (C) / Modify (M) / Delete (D) / Reply (R)) to be taken by the Virtual Domain along with the list of the Routing Service Node’s identifiers and the VirtualDomainID. Any resource to be assigned to the Virtual Domain is also indicated.

Table 4: Terms Used in the Message Types

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VirtualDomainID</td>
<td>The identifier for identifying the Virtual Domain</td>
</tr>
<tr>
<td>RSNID</td>
<td>The identifier for identifying the Routing Service Node</td>
</tr>
<tr>
<td>FlowID</td>
<td>The identifier for identifying flow</td>
</tr>
<tr>
<td>ActionID</td>
<td>The identifier for kinds of processes</td>
</tr>
<tr>
<td>Trust</td>
<td>Trustworthiness of flow</td>
</tr>
<tr>
<td>TrustID</td>
<td>The identifier for identifying Trust</td>
</tr>
<tr>
<td>OutboundDomainID</td>
<td>The identifier used for communication between different VTRouPDs</td>
</tr>
</tbody>
</table>
(a) Virtual Domain Management Message

<table>
<thead>
<tr>
<th>C</th>
<th>M</th>
<th>D</th>
<th>R</th>
<th>VirtualDomainID</th>
<th>No.</th>
<th>ResourceID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RouterID 1</td>
<td></td>
<td>RouterID 2</td>
</tr>
</tbody>
</table>

(b) Flow Table Update Message

<table>
<thead>
<tr>
<th>InPort</th>
<th>OutPort</th>
<th>ActionID</th>
<th>Padding</th>
</tr>
</thead>
<tbody>
<tr>
<td>InForwardingID</td>
<td>OutForwardingID</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(c) Routing Table update Message

<table>
<thead>
<tr>
<th>OutboundDomainID</th>
<th>DomainPriority</th>
</tr>
</thead>
<tbody>
<tr>
<td>InForwardingID</td>
<td>OutForwardingID</td>
</tr>
</tbody>
</table>

(d) Outbound Domain ID Notification

<table>
<thead>
<tr>
<th>DomainPriority</th>
<th>OutboundDomainID</th>
<th>ForwardingID</th>
</tr>
</thead>
<tbody>
<tr>
<td>InForwardingID</td>
<td>OutForwardingID</td>
<td></td>
</tr>
</tbody>
</table>

Figure 13: Message Type and Corresponding Packet Format

Table 5: Flow ID: Information Content

<table>
<thead>
<tr>
<th>Input Port (8)</th>
<th>Ethernet Frame Type (16)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer-2 ID (16)</td>
<td>Layer-2 Priority (8)</td>
</tr>
<tr>
<td>IP ToS (8)</td>
<td>Protocol Number (8)</td>
</tr>
<tr>
<td>Source IP Address (32)</td>
<td>Destination IP Address (32)</td>
</tr>
<tr>
<td>Source MAC Address (48)</td>
<td>Destination MAC Address (48)</td>
</tr>
<tr>
<td>Source Port Number (16)</td>
<td>Destination Port Number (16)</td>
</tr>
</tbody>
</table>

Flow Table Update Message

A Flow Table Update Message is for communication between the Policy Manager and Flow Controller to update the flow table at the Flow Controller. The packet format for the Flow Table Update message is shown in Fig. 13(b). It carries the TrustID and ActionID, along with the FlowID. In particular, the FlowID is the identifier of the input flow and the ActionID specifies the action that can be the Attach/Strap/Modify ForwardingID or Encrypt/Decrypt flows. Here, in order to identify packets associated with a flow, marked by FlowID, we use OpenFlow Switch Specification v1.1 [14], in which we define a 256-bit field for FlowID. The information content of FlowID is shown in Table 5.

Routing Table Update Message

A Routing Table Update Message is communicated between the VTRouPD Manager and Routing Service Nodes for routing table updates. The packet format for a Routing Table Update message is shown in Fig. 13(c). It carries the ActionID along with an input/output port and Forwarding ID for input and output packets.

Outbound Domain ID Notification

The Outbound Domain ID Notification is exchanged between the Policy Manager and Trust Level Regulator for trust negotiation among VTRouPDs. The packet format for Outbound Domain ID Notification is shown in Fig. 13(d). It has a field to indicate any priority for an OutboundDomainID, along with the OutboundDomainID and ForwardingID for input and output packets.

Author Biographies

Shingo Ata received M.E. and Ph.D. degrees in Informatics and Mathematical Science from Osaka University, Japan in 1998 and 2000, respectively. From 2003 to 2006, he was a Lecturer in the Graduate School of Engineering at Osaka City University, and an Associate Professor from 2006 to 2013. Currently, he is a Professor at the Graduate School of Engineering at Osaka City University. His research works include networking architecture, design of communication protocols, and performance modeling on communication networks.

Dijiang Huang received his B.S. degree from Beijing University of Posts and Telecommunications, China 1995. He received his M.S., and Ph.D. degrees from the University of Missouri–Kansas City, in 2001 and 2004, respectively. He is currently an associate professor of Computer Science in the School of Computing Informatics and Decision System Engineering at Arizona State University. His current research interests are computer networking, security, and privacy. He is an associate edi-
Xuan Liu is a Ph.D. student at the University of Missouri–Kansas City. She received her B.S. in Communication Engineering from China University of Geosciences (CUG) in June 2007 and her M.S. in Computer Science from the University of Missouri–Kansas City in December 2010. Her research interests include network virtualization, information centric networking, computer networking modeling and optimization.

Akira Wada received his B.E. in Information and Communication Engineering and his M.E. in Physical Electronics and Informatics from Osaka City University, Osaka, Japan, in 2011 and 2013, respectively. He is currently a Ph.D. student at the Graduate School of Engineering, Osaka City University, Japan. His research interests are in network security, software-defined networks and the future Internet. He is a student member of the IEEE.

Chun-Jen Chung received his M.S. degree in computer science from New York University. He is working toward the Ph.D. degree in the School of Computing Informatics and Decision Systems Engineering (CIDSE) at Arizona State University. Prior to that, he worked as a software developer at Microsoft and Oracle for several years. His current research interests include computer and network security, cloud system security, security in the software defined networking, and trusted computing in mobile devices and cloud computing.
Yasuhiro Sato received his B.E., M.E., and Ph.D. degrees in Information and Communication Engineering from Osaka City University, Osaka, Japan, in 2004, 2006, and 2009, respectively. From 2009 to 2013, he was a Lecturer in the Faculty of Maritime Safety Technology at Japan Coast Guard Academy. Since 2013, he has been an Associate Professor in the Faculty of Maritime Safety Technology at Japan Coast Guard Academy. His research work is in the area of modeling and evaluation of network performance. He is a member of IEEE.

Deep Medhi is a Curators’ Professor in the Department of Computer Science & Electrical Engineering at the University of Missouri-Kansas City, USA, and a honorary professor in the Department of Computer Science & Engineering at the Indian Institute of Technology–Guwahati, India. He received B.Sc. in Mathematics from Cotton College, Gauhati University, India, M.Sc. in Mathematics from the University of Delhi, India, and his Ph.D. in Computer Sciences from the University of Wisconsin-Madison, USA. Prior to joining UMKC in 1989, he was a member of the technical staff at AT&T Bell Laboratories. He served as an invited visiting professor at the Technical University of Denmark, a visiting research fellow at Lund Institute of Technology, Sweden, and State University of Campinas, Brazil. As a Fulbright Senior Specialist, he was a visitor at Bilkent University, Turkey, and Kurukshetra University, India. He is the Editor-in-Chief of Springer’s Journal of Network and Systems Management, and is on the editorial board of IEEE/ACM Transactions on Networking, IEEE Transactions on Network and Service Management, and IEEE Communications Surveys & Tutorials. He has published over 125 papers, and is co-author of the books, Routing, Flow, and Capacity Design in Communication and Computer Networks (2004) and Network Routing: Algorithms, Protocols, and Architectures (2007), both published by Morgan Kaufmann Publishers, an imprint of Elsevier Science. His research interests are multi-layer networking, network virtualization, data center optimization, and network routing, design, and survivability. His research has been funded by NSF, DARPA, and industries.