

# SeRViTR: A Framework for Trust and Policy Management for a Secure Internet and Its Proof-of-Concept Implementation

Xuan Liu<sup>1</sup>, Akira Wada<sup>2</sup>, Tianyi Xing<sup>3</sup>, Parikshit Juluri<sup>1</sup>, Yasuhiro Sato<sup>4</sup>,  
Shingo Ata<sup>2</sup>, Dijiang Huang<sup>3</sup>, Deep Medhi<sup>1</sup>

<sup>1</sup>University of Missouri–Kansas City, USA, <sup>2</sup>Osaka City University, Japan, <sup>3</sup>Arizona State University, USA,

<sup>4</sup>Japan Coast Guard Academy, Japan

**Abstract**—A secure network is considered to be an important goal of the Future Internet; one way this can be embodied is by having flexible and robust routing functionalities with intrinsic security mechanisms. 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. Based on these potential needs, a flexible, scalable, and robust routing framework that enables fine-grained flow control under fixed or dynamic policies called the *Virtual Trusted Routing and Provisioning Domain (VTRouPD)* [11] has been recently proposed. In this paper, we present a framework called the *Secure and Resilient Virtual Trust Routing (SeRViTR)* framework, which is a proof-of-concept model of VTRouPD at the implementation level. SeRViTR has particular entities that are designed for policy management and trust management between different VTRouPDs to enable a secure Internet. We define the roles of each entity within the SeRViTR framework as well as the messages exchanged between them. We also discuss how policy management and trust negotiation can be achieved. Moreover, we present validation on the functional implementation of several SeRViTR components to illustrate how to create virtual domains and change of trust levels between virtual domains.

**Index Terms**—Secure routing, virtualization, policy and trust management.

## I. INTRODUCTION

One of the factors that has led to the success of the current Internet is its flexible and robust routing functionality. However, with the rapid growth of the Internet, many issues have risen that cannot be solved by a plain routing capability. One of the major problems is in regard to network *trustworthiness*. Currently, the Internet lacks trustworthiness at the network level as a fundamental function. There are often ad hoc approaches to thwart attack traffic injected by malicious users. It is also difficult to protect the leakage of confidential information to untrusted users. As a result, even though the Internet has become a critical infrastructure for society, users remain concerned about the safety of the Internet. End users are still required to have a deeper knowledge regarding the security for protecting their own communication.

In general, the current Internet has a limited capability of *trustworthiness*. For example, a secure tunnel (e.g., IPSec [13] [14]) is established to exchange the routing information, or access control lists 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 solution. From a routing standpoint, the current Internet provides a simple and network-centric packet forwarding function by only referring 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 is required to be more flexible, and have fine-grained flow controls based on a policy, while addressing trustworthiness.

Based on the above-mentioned background, a flexible, scalable, and robust routing framework that enables fine-grained flow control under fixed or dynamic policies called Virtual Trusted Routing and Provisioning Domain (VTRouPD) [11] has been recently proposed. A VTRouPD is constructed by a collection of networking resources including routers and switches based on virtualization techniques; e.g., constructing virtual managed domains through tunneling and VLAN technologies. Within one or spanning multiple VTRouPDs, we can further create user-centric virtual routing domains that are denoted as  $\mu$ VTRouPDs. To realize VTRouPDs and  $\mu$ VTRouPDs, there are many technical issues to be solved. First, to support flexible traffic control according to various service or user requirements, diversification of routing functionality is mandatory instead of the unified routing strategy (i.e., shortest path forwarding) used in the current Internet. For this, the network should be virtualized and sliced according to control policies in place that change dynamically. Delivery of critical traffic should be especially secured by using an isolated slice from other traffic and controlled independently. Second, the integrated routing framework should support both user- or service-centric traffic controls, and provide secured communication with differentiated security requirements.

In this paper, we present a framework called the *Secure and Resilient Virtual Trust Routing (SeRViTR)* framework, which is a proof-of-concept model of VTRouPD at the implementation level. We first discuss the *trustworthiness*, which SeRViTR focuses on, and then we describe our design and implementation model of SeRViTR, by enumerating key

components.

The paper is organized as follows: we first describe related works in Section II, then clarify the routing with *trustworthiness* in Section III. Our design model of SeRViTR with descriptions on detailed operations of key components is presented in Section IV and Section V. We next discuss the implementation issues in Section VI. Finally, we conclude this paper with future research topics in Section VII.

## II. RELATED WORK

Network virtualization research has been active in recent years. A recent survey [10] states network virtualization may occur at four different layers: the physical layer, link layer, network layer, and application layer. In particular, Planet-Lab [3] is an application-layer virtualization while VINI [5] and VNET [6] are network and link-layer virtualizations, respectively. In [9], network virtualization in GpENI [16] was presented that uses the VINI framework. With virtualization techniques, various new services arise such as network management and resource management. In [8], a programmable hardware platform to construct virtual data planes that focus on hardware implementation was presented. In [17], the authors presented VROOM (Virtual ROuters On the Move), which is a new network-management primitive that avoids unnecessary changes to the logical topology by allowing (virtual) routers to freely move from one physical node to another. The work presented in [12] focuses on the accountability in a virtualized hosting environment. [15] introduces a policy-based resource management function into a virtual network environment based on a two-phase resource distributive model. Being a virtual laboratory, the Global Environment for Network Innovations (GENI)[1] is an open and large-scale experimental environment for researchers to collaborate and explore future networks. GENI enables the network to be sliceable so that users can share resources and achieve isolations. Although there are several projects joining GENI such as OpenFlow[2] and ProtoGENI[4], there is not much work on policy or trust management in such network virtualization environments, especially one that focuses on secure routing.

Our approach is different than the above. Our goal is to 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 an an intrinsic property. We provide a fine-grained virtualization for user-centric networking services by allowing to set a secure policy. Furthermore, we also design a mechanism to dynamically negotiate trust levels between domains.

## III. ROUTING MODEL WITH INTEGRATED TRUSTWORTHINESS

*Trustworthiness* is a fundamental criterion of routing in SeRViTR. We start with our perspective on trustworthiness that is used in the context of SeRViTR. Trustworthiness needed to be realized for a trustable network in the Future Internet may

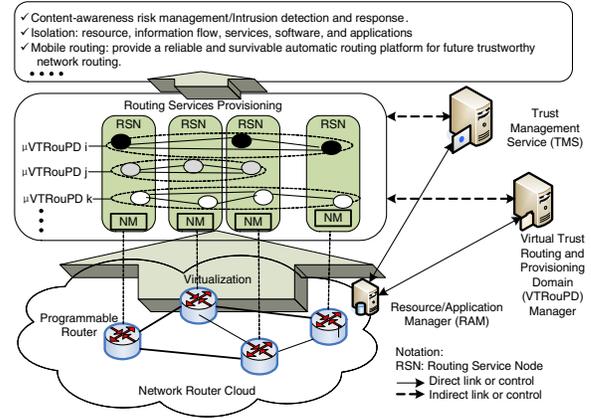


Fig. 1. SeRViTR: The Basic Concept

vary based on different viewpoints from end users, to network operators, or to service providers, as follows.

- 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 the situation when 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, etc.) must be confidential, secured, and protected.
- From the viewpoint of service providers, *trustworthiness of services* is the situation that the service is protected, secured, and exclusive of anonymous users. Not only users, but also 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 each other, managed by an integrated security policy, and a secure routing protocol along with flexible traffic engineering. Our aim with SeRViTR is to show a viable proof-of-concept model to achieve these goals.

## IV. SeRViTR: OVERVIEW, MODELS AND SPECIFICATION

### A. Using SeRViTR to Realize VTRouPD

In [11], VTRouPD was proposed as a new conceptual model supporting the following features: (a) multiple routing domains can be constructed based on services under different trust levels, (b) trust levels are pre-calculated at the network level in order to handle security between routing domains, (c) the

TABLE I  
VTRouPD COMPONENTS WITH ROLE DESCRIPTIONS

VTRouPD Component	Role	Description
VTRouPD	Managed Domain	Physical Topology and Virtual Domain used as the scope of Policy which Administrator inputs
$\mu$ VTRouPD	Virtual Domain (intra)	The sub-domain which consists of virtual routers under a particular routing policy in a single VTRouPD
	Virtual Domain (spanning)	The sub-domain spanning multiple VTRouPDs consists of virtual routers under a particular policy, i.e., if $\mu$ is a $\mu$ VTRouPD, and $V_i$ and $V_j$ and two VTRouPDs that $\mu$ spans, then $\mu \subseteq V_i \cup V_j, \mu \not\subseteq V_i, \mu \not\subseteq V_j$
Trust Management Service (TMS)	Authentication Server	Generating and distributing authentication information to Policy Manager
	Policy Manager	Invoking $\mu$ VTRouPD creation/deletion by providing policy management
	Trust Level Regulator	Trust level negotiation between multiple VTRouPDs
VTRouPD Manager	Domain Controller	Creating $\mu$ VTRouPDs based on the policy
Routing Service Node	Virtual Router	Physical routers where virtual router instances can be created on physical machines.
Resource Application Manager (RAM)	Resource management	

entry points have the capability to authorize and/or control traffic when necessary.

Typically, from a system's perspective, traffic management and network resource management components are necessary for monitoring and managing a network. There are three components involved to achieve this purpose, as presented in Fig. 1: *Trust Management Service (TMS)*, *VTRouPD Manager*, and *Resource and Application Manager (RAM)*. Every physical router that belongs to a VTRouPD has complete routing information of its own VTRouPD, and we name it the *Routing Service Node (RSN)*. In each VTRouPD, one or multiple router instances, or virtual routers, are loaded at every RSN. The *VTRouPD Manager* is in charge of sub-domain creation/deletion/updating, while TMS is the Trust Authority (TA) for the system. TMS manages the secure routing from two aspects. On one hand, it handles the cryptographic key and parameter distribution and revocation. On the other hand, it also provides identity search, federation services for multiple administrative domains, policy checking and enforcement functions. Hence, identity and federation services for physical nodes belonging to multiple administrative domains are supported.

The conceptual VTRouPD model in [11] was proposed from the perspective of a high-level design. In order to realize the concept described in the VTRouPD model, we have designed a comprehensive architecture called Secure and Resilient Virtual Trust Routing (SeRViTR) that crystallizes the components defined in the VTRouPD model. In this work, we design functional components at the implementation level to realize the roles of TMS and VTRouPD Manager. We also define the exchanging messages for the communication between these functional components. Table I presents the VTRouPD components and their roles in the SeRViTR framework, and Table II summarizes the terminologies for the key fields defined in the message types. These notations will be used through the rest of the paper. In the following subsections, we illustrate the message types and SeRViTR components successively.

TABLE II  
TERMS USED IN THE MESSAGE TYPES

Term	Description
VirtualDomainID	The identifier for identifying the Virtual Domain
RSNID	The identifier for identifying the Routing Service Node
FlowID	The identifier for identifying flow
ActionID	The identifier for kinds of processes
Trust	Trustworthiness of flow
TrustID	The identifier for identifying Trust
OutboundDomainID	The identifier used for communication between different VTRouPDs

### B. SeRViTR Message Types

When proposing VTRouPD, it was identified that the Trust Management Service (TMS) should be responsible for two tasks. In this paper, we further elaborate the TMS by assigning it three different roles: *Authentication Server*, *Policy Manager*, and *Trust Level Regulator*, which are discussed in Section IV. In Fig. 2, the *Policy Manager* communicates with every other individual SeRViTR component. In order to implement policy management, we introduce 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.

1) *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. 3(a). It has a field to indicate action (Create (C) / Modify (M) / Delete (D) / Reply (R)) to be taken by the *Virtual Domain* along with the list of *Routing Service Node*'s identifiers and the *VirtualDomainID*. Any resource to be assigned to the *Virtual Domain* is also indicated.

2) *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. 3(b). It carries the *TrustID* and

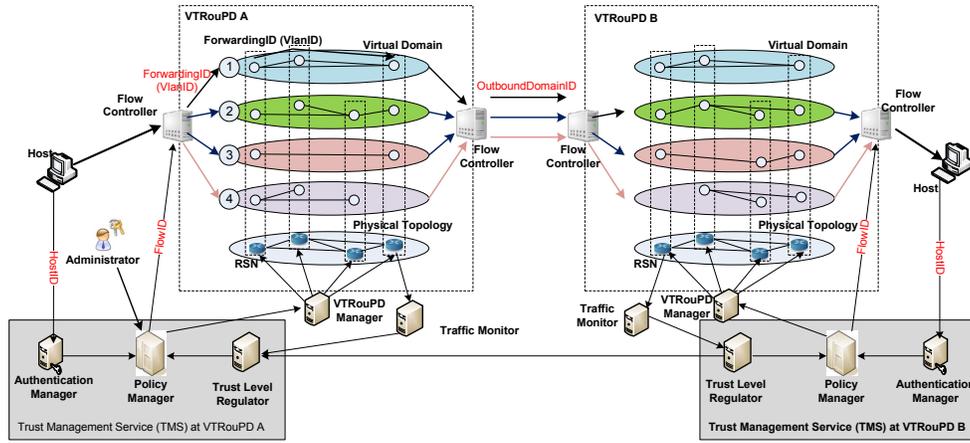


Fig. 2. SeRViTR: Architectural Overview

*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[7], in which we define a 256-bit field for *FlowID*. The information content of *FlowID* is shown in Table III.

3) *Routing Table Update Message*: A *Routing Table Update Message* is communicated between the *Policy Manager* and *Routing Service Nodes* for routing table updates. The packet format for a *Routing Table Update message* is shown in Fig. 3(c). It carries the *ActionID* along with an input/output port and Forwarding ID for input and output packets.

4) *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. 3(d). It has a field to indicate any priority for an *OutboundDomainID*, along with the *Outbound-DomainID* and Forwarding ID for input and output packets.

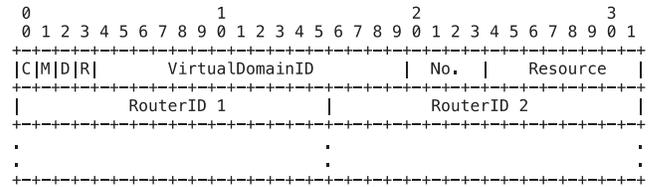
TABLE III  
FLOW ID: INFORMATION CONTENT

Input Port (8)	Ethernet Frame Type (16)
VLAN ID (16)	VLAN Priority (8)
IP ToS (8)	Protocol Number (8)
Source IP Address (32)	Destination IP Address (32)
Source MAC Address (48)	Destination MAC Address (48)
Source Port Number (16)	Destination Port Number (16)

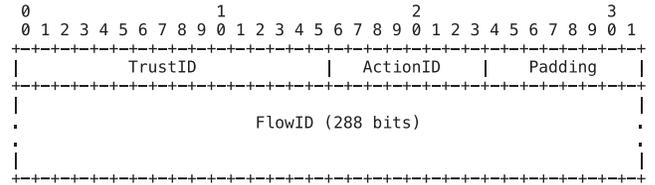
### C. SeRViTR Components

Table I shows the equivalent relationship between *VTRouPD* and *SeRViTR* components. In this part, we will illustrate the role of each *SeRViTR* components, and the interactive relation between each other.

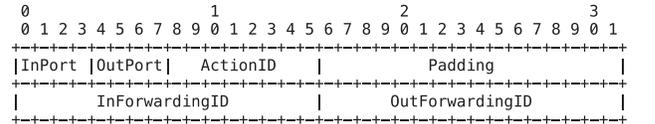
1) *VTRouPD*: A *VTRouPD* is the domain with administrative control. Within a *VTRouPD*, there can be multiple *Virtual Domains*, each being identified by a *VirtualDomainID*.



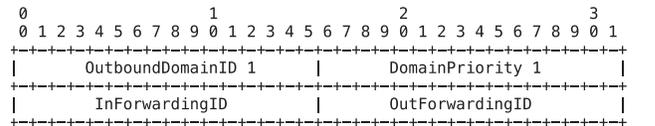
(a) Virtual Domain Management message



(b) Flow Table Update message



(c) Routing Table update message



(d) Outbound Domain ID Notification

Fig. 3. Message Type and Corresponding Packet Format

2)  $\mu$ *VTRouPD*: A  $\mu$ *VTRouPD* is a user-centric virtual routing domain that spans *VTRouPDs*. A  $\mu$ *VTRouPD* can operate at the flow-level, where the flow is recognized by assigning security policies. A *VTRouPD* may contain multiple  $\mu$ *VTRouPDs*. If there is only one *VTRouPD* between the source and the destination, then a  $\mu$ *VTRouPD* is also a *Virtual*

Domain, or sub-domain, inside the VTRouPD. However, If there are multiple VTRouPDs between the source and the destination, a  $\mu$ VTRouPD represents the whole virtual routing domain that spans different VTRouPDs. In other words, the  $\mu$ VTRouPD, in this case, spans multiple *Virtual Domains*.

3) *Policy Manager*: A *Policy Manager* is associated with each VTRouPD, and the role of the *Policy Manager* is many-fold:

- It invokes policies that are assigned by the *Administrator*
- It is in charge of policy management. We use XACML (eXtensible Access Control Markup Language) to describe the rules, and each rule is identified by  $\langle \text{FlowID}, \text{Trust} \rangle$ . In order to ensure security, we use a unique 16-bit integer to identify the *Trust* based on the policy
- It announces the request of creation or deletion of *Virtual Domains* to the VTRouPD Manager. When a new *TrustID* is obtained, the *Policy Manager* will generate the request and send it to the VTRouPD Manager through the *Virtual Domain Management Message* (Fig 3(a))
- It sends a *Flow Table Update Message* (Fig 3(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 *Policy Manager* maintains three tables, the Rule-set Table, the Trust-level Table, and the Virtual Domain Table. The operation of the *Policy Manager* is depicted in Fig 4

4) *Trust Level Regulator*: 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*.

5) *Authentication Server*: 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.

6) *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 *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 *Virtual-DomainID*, and inserts it into the *Virtual Domain Management Table*. Similarly, in order to delete a *Virtual Domain*, the *VTRouPD Manager* takes the entry off 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*

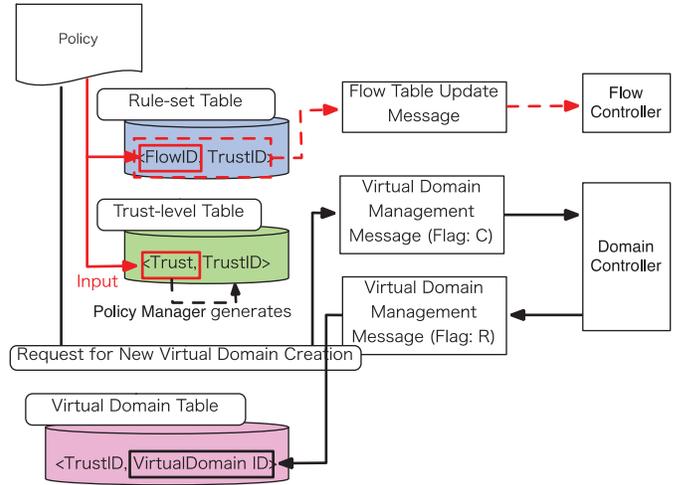


Fig. 4. Operations of Policy Manager

*Manager* sends the *Routing Table Update Message*(Fig 3(c)) to the *Routing Service Node* to update its routing table. Particularly, the *ForwardingID* is set as the *VlanID* in our implementation. For clarity, we use the *VlanID* in the rest of this paper.

7) *Routing Service Node*: A *Routing Service Node* 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 *VlanID* denoted in the routing table.

8) *Flow Controller*: The *Flow Controller* is the new component we added in SeRviTR, and it is placed at the edge of the VTRouPD. A *Flow Controller* forwards flows to appropriate *Virtual Domains* based on a given policy. For any outgoing packet, the *VlanID* is removed from a data packet; while for any incoming packet, a *VlanID* 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*.

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

## V. POLICY AND TRUST MANAGEMENT IN SeRviTR

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

### A. Challenges for Policy and Trust Management

To establish VTRouPDs, we require network routers to be programmable, i.e., we should be able to create multiple virtual routers (VRs) on the same physical router and each VR is responsible for a particular *Virtual Domain*. However, there is

a certain roadblock since the router's hardware configuration restricts the number of *Virtual Domains* on each router. Secondly, 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 independently managed in different *VTRouPDs*. Hence, different administrative domains can have different compositions of *VTRouPDs*. To federate the trust management among the *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 as the initial trust level. The trust level can be measured using metrics such as, percentage of good traffic transited, reliability of the routing system, trust levels of ingress and egress neighboring domains, etc.

### B. Trust Management Service Sequence Diagram

Considering the challenges above, we propose our methods. In the following, we present sequence diagrams for a number of situations of *VTRouPD* Trust Management Services.

1) *Policy Setting*: Policy setting involves a series of steps, including creating *Virtual Domains*, and updating routing tables and flow tables. The sequence diagram for policy setting is shown in Fig. 5. As we can see from this figure, the *Administrator* inputs a policy, and on receiving this, 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 will generate and send 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 Nodes*. 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. At last, the *Policy Manager* will notify the *Administrator* that the policy setting is completed.

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

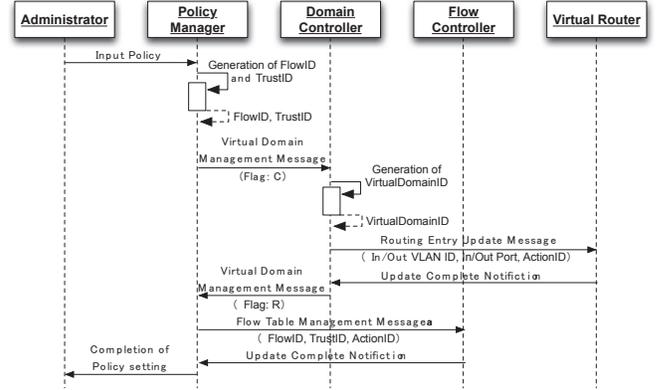


Fig. 5. Policy Setting

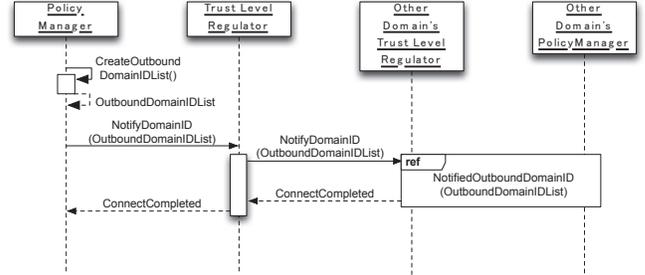


Fig. 6. Outbound Trust Level Notification

is shown in Fig. 6. 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.

3) *Trust Level Change Notification*: Periodically, there would be 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. 7. 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.

## VI. SERVITR EXPERIMENTAL ENVIRONMENT AND FUNCTIONAL IMPLEMENTATION

Our implementation environment involves three sites, which are Osaka City University in Japan, Arizona State University and the University of Missouri - Kansas City in the United States. The experimental environment consists of three *VTRouPDs*, one located at each site, respectively. A *Trust*

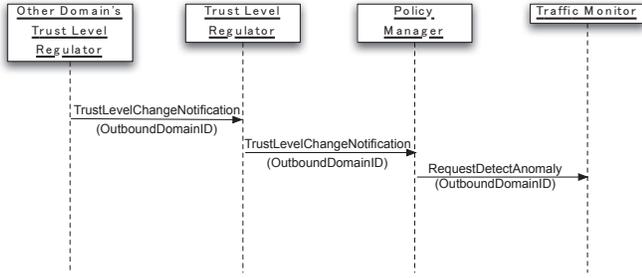


Fig. 7. Trust Level Change Notification

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 one hand, we are building tunnels among the three sites, and we have built GRE tunnels through Open vSwitches and passed basic connection testing. On the other hand, we have implemented the *VTRouPD Manager*, the *Flow Controller* and the *Routing Service Nodes* in a two-*VTRouPD* environment within one site.

#### A. Geo-distributed Multi-Domain Infrastructure

To establish a geo-distributed multi-domain infrastructure, we have established a layer-2 GRE tunnel so that any two sites have either a direct or an indirect layer-2 connection. An Open vSwitch, which is another intelligent mode besides the native linux bridge mode in Xen[18] virtualization, is 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 has been 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 since the tunnel is established by using an Open vSwitch under the OpenFlow protocol. Additional descriptions can be found at [19].

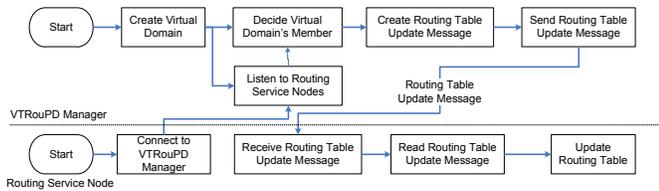


Fig. 8. Communication between the *VTRouPD Manager* and the *Routing Service Node*

#### B. Experimental Environment

Our first experiment is to create virtual domains, where the flows are correctly differentiated by the trust level. Since we are yet to implement the *Policy Manager* and *Trust Level Regulator*, the *VTRouPD Manager* creates *Virtual Domains* and assigns the *FlowID* and the *TrustID*.

Our experimental environment 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++. 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 different *VlanIDs* 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 system can communicate with each other.

Next, we illustrate how the *VTRouPD Manager* communicates with a *Routing Service Node* based on the policy setting mechanism to get the routing table updated, shown in Fig.8. Upon detecting the connection from the *Routing Service Nodes*, the *VTRouPD Manager* will determine the members for the *Virtual Domains* it has created, and afterward sends the *Routing Table Update Message* to the *Routing Service Nodes* to get the routing table updated.

#### C. Experimental Scenario: Multiple Domains

We have setup a connection between two end-hosts belonging to two different *VTRouPDs* as shown in Fig. 9. In this scenario, our goal is to validate the functionalities of the *Flow Controller*, the *VTRouPD Manager* and the *Routing Service Node*. With each *VTRouPD* setup, there are two *Flow Controllers*, one each at ingress and egress points; a *VTRouPD Manager* is connected to three *Routing Service Nodes*. 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 to be at the highest trust level. When the host, PC1, at *VTRouPD A* sends out both types of packets to the ingress *Flow Controller* (FC1) with the pre-determined policy rule  $\langle \text{FlowID}, \text{Trust} \rangle$ , the FC1 updates its flow table, attaches *VlanIDs* to SSH and HTTP flows respectively, and sends the *VlanID* to the *Virtual Domain* according to the *TrustID*. Meanwhile, the *Routing Service Nodes* update their routing tables. When both flows arrive at the egress *Flow Controller* (FC2), FC2 strips the previous *VlanID*, and attaches a new *VlanID* to each flow and sends this new *VlanID* to the ingress *Flow Controller* (FC3) at *VTRouPD B*. Finally, when flows arrive at FC4, the egress *Flow Controller*, FC4, strips the *VlanIDs* and forwards the flow to the destination PC2.

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*. We change the *TrustID* manually, and in turn, the policy rule is updated at FC3, which then attaches a different *VlanID* to the SSH flow and sends it to a different *Virtual Domain* that is under a lower trust level. The degraded SSH flow is represented as a dotted arrow in Fig.9.

## VII. SUMMARY AND FUTURE WORK

In this paper, we propose an overview of the design of SeRvITR framework for the Future Internet according to

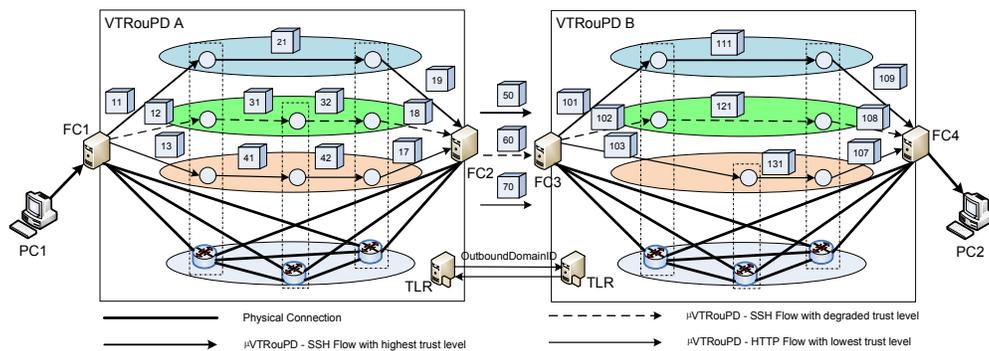


Fig. 9. Multiple-Domain Environment

the idea of the VTRouPD from our earlier work. We introduce the role and responsibility of VTRouPD components in the SerViTR architecture. Specifically, we illustrate the VTRouPD Trust Management Service that is able to setup trust levels for different virtual domains within a VTRouPD, and negotiate the trustworthiness levels of the flows between VTRouPDs. From our basic result of implementation, we have been able to create flow-level  $\mu$ VTRouPDs under different trust levels, and migrate the flow to the  $\mu$ VTRouPD under a lower trust level given that the anomaly traffic is detected in the flow.

In the future, we plan to implement the rest of the components in SerViTR architecture. In particular, we will have the Policy Manager, Trust Level Regulator, and Traffic Monitor to fully provide Trust Management Service. We shall declare policies and corresponding metrics to evaluate whether a flow is good or not. Our experiments in this paper were only implemented locally at one site, and we plan to scale it to the other two institutions through globally layer-2 VLAN techniques, constructing at least one VTRouPD at each institution.

The policy management system has the flexibility of defining the security policy for each domain and can be used as an experimental tool. For example, GENI provides collaborative and exploratory environments for academia, industry and the public, which is a good platform to which the policy management system can be extended. We plan to make this system public for experimenters so that users can claim, configure resources, define their own policy, and evaluate the policy mechanism in the real system. For this, GENI provides good lower level infrastructure support for various platforms. Thus, our next step is to explore how to make this policy management system an aggregate in the GENI platform.

#### ACKNOWLEDGEMENT

This work is supported by US NSF grants CNS-1029562 and CNS-1029546, Office of Naval Research's (ONR) Young Investigator Program (YIP), an HP IRP grant, and Japan NICT International Collaborative Research Grant.

#### REFERENCES

[1] "GENI: Global Environment for Network Innovations," <http://www.geni.net/>.  
 [2] "OpenFlow," <http://www.openflow.org/>.

[3] "PlanetLab," <http://www.planet-lab.org/>.  
 [4] "ProtoGENI," <http://www.protogeni.net>.  
 [5] "VINI: Virtual Network Infrastructure," <http://vini-veritas.net/>.  
 [6] "Virtuoso: Resource Management and Prediction for Distributed Computing using Virtual Machines," [http://virtuoso.cs.northwestern.edu](http://virtuoso.cs.northwestern.edu/).  
 [7] "Open Flow Switch Specification," <http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf>, February 2011.  
 [8] M. B. Anwer and N. Feamster, "Building a fast, virtualized data plane with programmable hardware," in *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, ser. VISA '09. New York, NY, USA: ACM, 2009, pp. 1–8. <http://doi.acm.org/10.1145/1592648.1592650>  
 [9] R. Cherukuri, X. Liu, A. Bavier, J. Sterbenz, and D. Medhi, "Network virtualization in GpENI: Framework, implementation and integration experience," in *Proc. of 3rd IEEE/IFIP International Workshop on Management of the Future Internet (ManFI'2011)*, Dublin, Ireland, May 2011, pp. 1212–1219.  
 [10] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Comput. Netw.*, vol. 54, pp. 862–876, April 2010. <http://dx.doi.org/10.1016/j.comnet.2009.10.017>  
 [11] D. Huang, S. Ata, and D. Medhi, "Establishing Secure Virtual Trust Routing And Provisioning Domains For Future Internet," in *Proceedings of IEEE Globecom, The Next Generation Networking Symposium*, 2010.  
 [12] E. Keller, R. B. Lee, and J. Rexford, "Accountability in hosted virtual networks," in *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, ser. VISA '09. New York, NY, USA: ACM, 2009, pp. 29–36. <http://doi.acm.org/10.1145/1592648.1592654>  
 [13] S. Kent and R. Atkinson, "Ip authentication header," IETF RFC 2402, Tech. Rep., 1998.  
 [14] S. Kent and R. Atkinson, "Ip encapsulating security payload (esp)," IETF RFC 2406, Tech. Rep., 1998.  
 [15] T. Miyamura, S. Kamamura, and K. Shiimoto, "Policy-based resource management in virtual network environment," in *Network and Service Management (CNSM)*, 2010, pp. 282 – 285.  
 [16] J. Sterbenz, D. Medhi, B. Ramamurthy, C. Scoglio, D. Hutchison, B. Plattner, T. Anjali, A. Scott, C. Buffington, G. Monaco, D. Gruenbacher, R. McMullen, J. Rohrer, J. Sherrell, P. Angu, R. Cherukuri, H. Qian, and N. Tare, "The Great Plains Environment for Network Innovation (GpENI): A programmable testbed for future Internet architecture research," in *Proc. of 6th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TridentCom)*, Berlin, Germany, May 2010, pp. 428–441.  
 [17] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford, "Virtual routers on the move: Live router migration as a network-management primitive," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 231–242, August 2008. <http://doi.acm.org/10.1145/1402946.1402985>  
 [18] xen.org, "Xen Hypervisor," <http://www.xen.org/>.  
 [19] T. Xing, X. Liu, C.-J. Chung, A. Wada, S. Ata, D. Huang, and D. Medhi, "Constructing virtual networking environment in a Geo-distributed Programmable Layer-2 Networking Environment (G-PLaNE)," in *IEEE 5th International Workshop on the Network of the Future (FutureNet-V)*, Ottawa, Canada, June 2012.