

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: April 5, 2013

S. Amante
Level 3 Communications, Inc.
J. Medved
Cisco Systems, Inc.
T. Nadeau
Juniper Networks
October 2, 2012

Topology API Use Cases
draft-amante-irs-topology-use-cases-00

Abstract

This document describes use cases for gathering routing, forwarding and policy information, (hereafter referred to as topology information), about the network and reflecting changes to the topology back into the network and related systems. It describes several applications that need to view or change the topology of the underlying physical or logical network. This document further demonstrates a need for a "Topology Manager" and related functions that collects topology data from network elements and other data sources, coalesces the collected data into a coherent view of the overall network topology, and normalizes the network topology view for use by clients -- namely, applications that consume or want to change topology information.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 5, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Statistics Collection	5
1.2. Inventory Collection	5
1.3. Requirements Language	6
2. Terminology	6
3. Orchestration, Collection & Presentation Framework	7
3.1. Overview	7
3.2. Topology Manager	8
3.3. Policy Manager	10
3.4. Orchestration Manager	11
4. Use Cases	12
4.1. Virtualized Views of the Network	12
4.1.1. Capacity Planning and Traffic Engineering	12
4.1.2. Services Provisioning	15
4.1.3. Rapid IP Renumbering, AS Migration	15
4.1.4. Troubleshooting & Monitoring	17
4.2. Path Computation Element (PCE)	17
4.3. ALTO Server	18
5. Acknowledgements	19
6. IANA Considerations	19
7. Security Considerations	19
8. References	20
8.1. Normative References	20
8.2. Informative References	20
Authors' Addresses	20

1. Introduction

In today's networks, a variety of applications, such as Traffic Engineering, Capacity Planning, Security Auditing or Services Provisioning (for example, Virtual Private Networks), have a common need to acquire and consume network topology information. Unfortunately, all of these applications are (typically) vertically integrated: each uses its own proprietary normalized view of the network and proprietary data collectors, interpreters and adapters, which speak a variety of protocols, (SNMP, CLI, SQL, etc.) directly to network elements and to back-office systems. While some of the topological information can be distributed using routing protocols, unfortunately it is not desirable for some of these applications to understand or participate in routing protocols.

This approach is incredibly inefficient for several reasons. First, developers must write duplicate 'network discovery' functions, which then become challenging to maintain over time, particularly as/when new equipment are first introduced to the network. Second, since there is no common "vocabulary" to describe various components in the network, such as physical links, logical links, or IP prefixes, each application has its own data model. To solve this, some solutions have distributed this information in the normalized form of routing distribution. However, this information still does not contain "inactive" topological information, thus not containing information considered to be part of a network's inventory.

These limitations lead to applications being unable to easily exchange information with each other. For example, applications cannot share changes with each other that are (to be) applied to the physical and/or logical network, such as installation of new physical links, or deployment of security ACL's. Each application must frequently poll network elements and other data sources to ensure that it has a consistent representation of the network so that it can carry out its particular domain-specific tasks. In other cases, applications that cannot speak routing protocols must use proprietary CLI or other management interfaces which represent the topological information in non-standard formats or worse, semantic models.

Overall, the software architecture described above at best results in incredibly inefficient use of both software developer resources and network resources, and at worst, it results in some applications simply not having access to this information.

Figure 1 is an illustration of how individual applications collect data from the underlying network. Applications retrieve inventory, network topology, state and statistics information by communicating directly with underlying Network Elements as well as with

intermediary proxies of the information. In addition, applications transmit changes required of a Network Element's configuration and/or state directly to individual Network Elements, (most commonly using CLI or Netconf). It is important to note that the "data models" or semantics of this information contained within Network Elements are largely proprietary with respect to most configuration and state information, hence why a proprietary CLI is often the only choice to reflect changes in a NE's configuration or state. This remains the case even when standards-based mechanisms such as Netconf are used which provide a standard syntax model, but still often lack due to the proprietary semantics associated with the internal representation of the information.

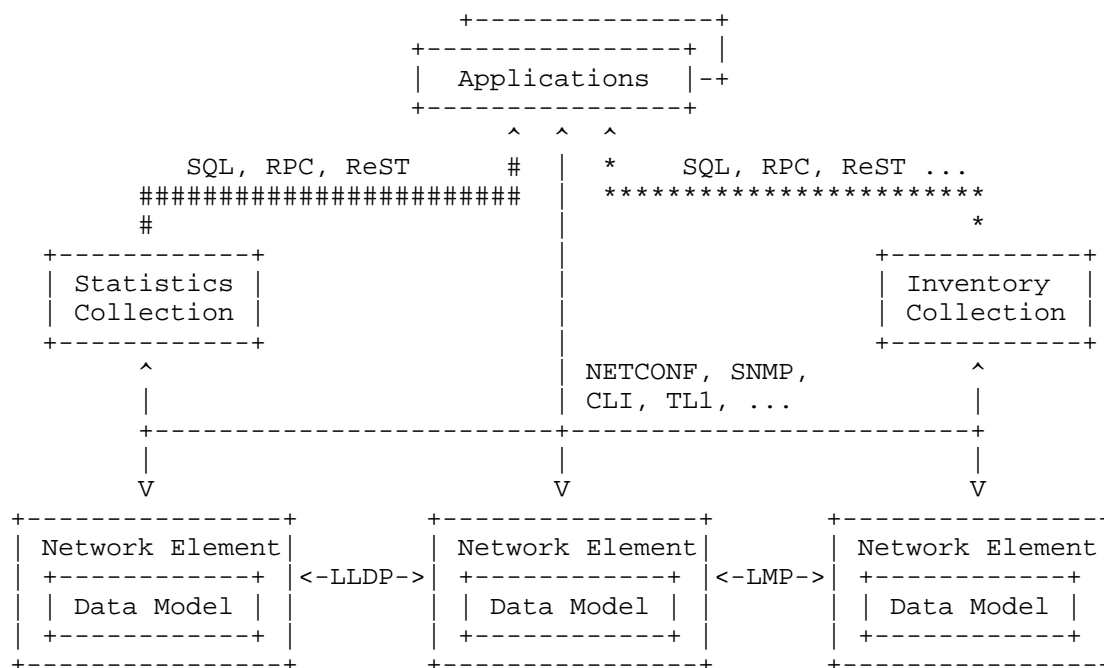


Figure 1: Applications getting topology data

Figure 1 shows how current management interfaces such as NETCONF, SNMP, CLI, etc. are used to transmit or receive information to/from various Network Elements. The figure also shows that protocols such as LLDP and LMP participate in topology discovery, specifically to discover adjacent network elements.

The following sections describe the "Statistics Collection" and "Inventory Collection" functions.

1.1. Statistics Collection

In Figure 1, "Statistics Collection" is a dedicated infrastructure that collects statistics from Network Elements. It periodically polls Network Elements (for example, every 5-minutes) for octets transferred per interface, per LSP, etc. Collected statistics are stored and collated, (for example, to provide hourly, daily, weekly 95th-percentile figures), within the statistics data warehouse. Applications typically query the statistics data warehouse rather than poll Network Elements directly to get the appropriate set of link utilization figures for their analysis.

1.2. Inventory Collection

"Inventory Collection" is a network function responsible for collecting network element component and state (i.e.: interface up/down, SFP/XFP optics inserted into physical port, etc.) information directly from network elements, as well as storing inventory information about physical network assets that are not retrievable from network elements, (hereafter referred to as a inventory asset database). Inventory Collection from network elements commonly use SNMP and CLI to acquire inventory information. The information housed in the Inventory Manager is retrieved by applications using a variety of protocols: SQL, RPC, etc. Inventory information, retrieved from Network Elements, is updated in the Inventory Collection system on a periodic basis to reflect changes in the physical and/or logical network assets. The polling interval to retrieve updated information is varied depending on scaling constraints of the Inventory Collection systems and expected intervals at which changes to the physical and/or logical assets are expected to occur.

Examples of changes in network inventory that need be learned by the Inventory Collection function are as follows:

- o Discovery of new Network Elements. These elements may or may not be actively used in the network (i.e.: provisioned but not yet activated).
- o Insertion or removal of line cards or other modules, i.e.: optics modules, during service or equipment provisioning.
- o Changes made to a specific Network Element through a management interface by a field technician.
- o Indication of an NE's physical location and associated cable run list, at the time of installation.

- o Insertion or removal of cables that result in dynamic discovery of a new or lost adjacent neighbor, etc.

1.3. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119]

2. Terminology

The following briefly defines some of the terminology used within this document.

Inventory Manager: Describes a function of collecting network element inventory and state information directly from network elements, and potentially associated offline inventory databases, via standards-based data models. Components contained in this super set might be visible or invisible to a specific network layer, i.e.: a physical link is visible within the IGP, however the Layer-2 switch through which the physical link traverses is unknown to the Layer-3 IGP. .

Policy Manager: Describes a function of attaching metadata to network components/attributes. Such metadata is likely to include security, routing, L2 VLAN ID, IP numbering, etc. policies that enable the Topology Manager to: a) assemble a normalized view of the network for clients to access; b) allow clients (or, upper-layer applications) read-only vs. read-write access to various network layers and/or network components, etc. The Policy Manager function may be a sub-component of the Topology Manager or it may be a standalone. This will be determined as the work with IRS evolves.

Topology Manager: Network components (inventory, etc.) are retrieved from the Inventory Manager and synthesized with information from the Policy Manager into cohesive, normalized views of network layers. The Topology Manager exposes normalized views of the network via standards-based data models to Clients, or higher-layer applications, to act upon in a read-only and/or read-write fashion. The Topology Manager may also push information back into the Inventory Manager and/or Network Elements to execute changes to the network's behavior, configuration or state.

Orchestration Manager: Describes a function of stitching together resources (i.e.: compute, storage) and/or services with the network or vice-versa. The Orchestration Manager relies on the capabilities provided by the other "Managers" listed above in order to realize a complete service.

Normalized Topology Data Model: A data model that is constructed and represented using an open, standards-based model that is consistent between implementations.

Data Model Abstraction: The notion that one is able to represent the same set of elements in a data model at different levels of "focus" in order to limit the amount of information exchanged in order to convey this information.

Multi-Layer Topology: Topology is commonly referred to using the OSI protocol layering model. For example, Layer 3 represents routed topologies that typically use IPv4 or IPv6 addresses. It is envisioned that, eventually, multiple layers of the network may be represented in a single, normalized view of the network to certain applications, (i.e.: Capacity Planning, Traffic Engineering, etc.)

Network Element (NE): refers to a network device that typically is addressable (but not always), and hosts. It is sometimes referred to as Nodes.

Links: Every NE contains at least 1 link. These are used to connect the NE to other NEs in the network. Links may be in a variety of states including up, down, administratively down, internally testing, or dormant. Links are often synonymous with network ports on NEs.

3. Orchestration, Collection & Presentation Framework

3.1. Overview

Section 1 demonstrates the need for a network function that would provide a common, standard-based topology view to applications. Such topology collection/management/presentation function would be a part wider framework, that would also include policy management and orchestration. The framework is shown in Figure 2.

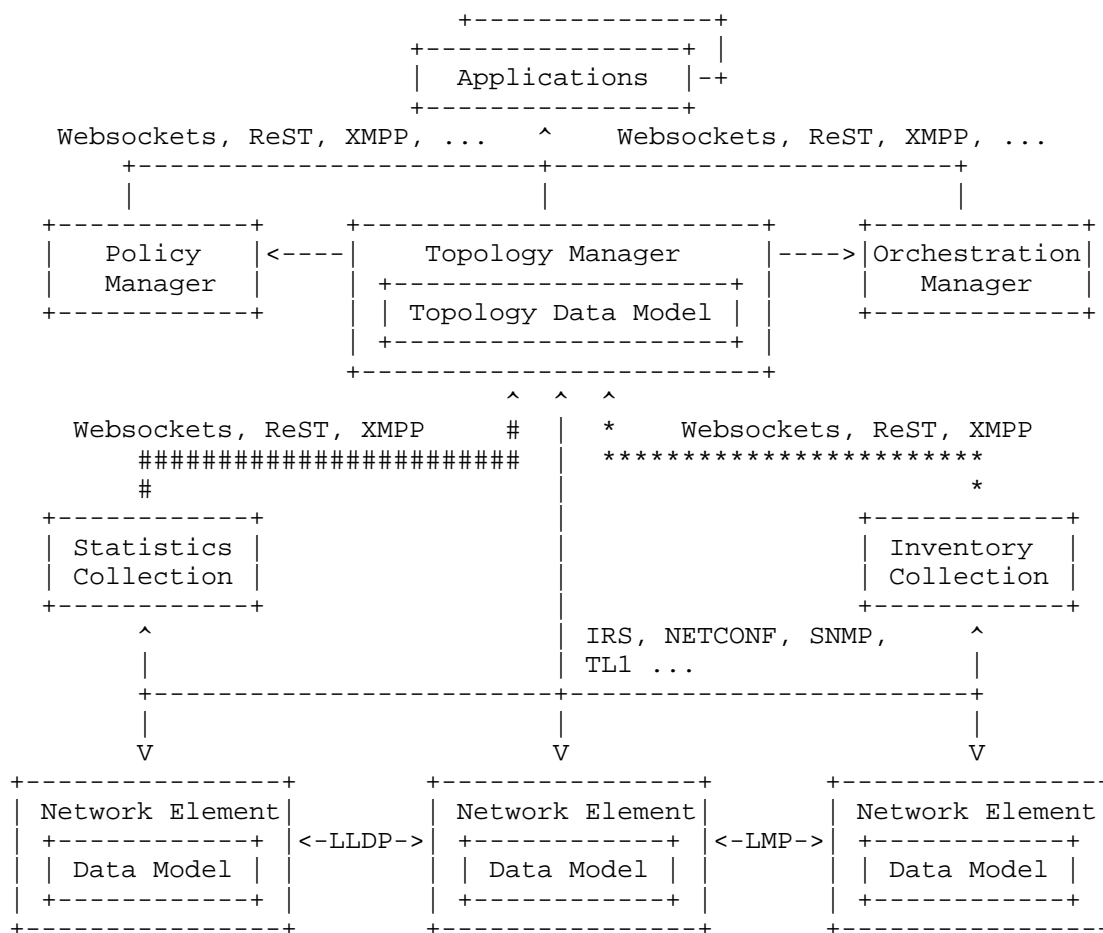


Figure 2: Topology Manager

The following sections describe in detail the Topology Manager, Policy Manager and Orchestration Manager functions.

3.2. Topology Manager

The Topology Manager is responsible for retrieving topological information from the network via a variety of sources. The first most obvious source is the "live" IGP or an equivalent mechanism. "Live" IGP provides information about links that are components of the active topology, in other words links that are present in the Link State Database (LSDB) and are eligible for forwarding. The second source of topology information is the Inventory Collection system, which provides information for network components not visible

within the IGP's LSDB, (i.e.: links or nodes, or properties of those links or nodes, at lower layers of the network).

The Topology Manager would synthesize retrieved information into cohesive, abstracted views of the network using a standards-based, normalized topology data model. The Topology Manager can then expose these data models to Clients, or higher-layer applications using a northbound interface, which would be a protocol/API commonly used by higher-layer applications to retrieve and update information. Examples of such protocols are ReST, Websockets, or XMPP. Topology Manager's clients would be able to act upon the information in a read-only and/or read-write fashion, (based on policies defined within the Policy Manager).

Clients may request changes to the network topology by publishing changes within data models and sending those to the Topology Manager. The Topology Manager internally validates the requested changes against various constraints and, if the changes are permitted, the Topology Manager updates associated Managers (Policy or Inventory Managers), communicates those changes to the individual network elements and, finally, verifies that those configurations were properly received and executed by the network elements.

It is envisioned that the Topology Manager will ultimately contain topology information for multiple layers of the network: Transport, Ethernet and IP/MPLS as well as multiple (IGP) areas and/or multiple Autonomous Systems (ASes). This allows the Topology Manager to stitch together a holistic view of several layers of the network, which is an important requirement, particularly for upper-layer Traffic Engineering, Capacity Planning and Provisioning Clients (applications) used to design, augment and optimize IP/MPLS networks that require knowledge of underlying Shared Risk Link Groups (SRLG) within the Transport and/or Ethernet layers of the network.

The Topology Manager must have the ability to discover and communicate with network elements who are not only active and visible within the Link State Database (LSDB) of an active IGP, but also network elements who are active, but invisible to a LSDB (e.g.: L2 Ethernet switches, ROADM's, etc.) that are part of the underlying Transport network. This requirement will influence the choice of protocols needed by the Topology Manager to communicate to/from network elements at the various network layers.

It is also important to recognize that the Topology Manager will be gleaning not only (relatively) static inventory information from the Inventory Manager, i.e.: what linecards, interface types, etc. are actively inserted into network elements, but also dynamic inventory information, as well. With respect to the latter, network elements

are expected to rely on various Link Layer Discovery Protocols (i.e.: LLDP, LMP, etc.) that will aid in automatically identifying an adjacent node, port, etc. at the far-side of a link. This information is then pushed to or pulled by the Topology Manager in order for it to have an accurate representation of the physical topology of the network.

3.3. Policy Manager

The Policy Manager is the function used to enforce and program policies applicable to network component/attribute data. Policy enforcement is a network-wide function that can be consumed by various network elements and services including the Inventory Manager, Topology Manager or other network elements. Such policies are likely to encompass the following.

- o Logical Identifier Numbering Policies
 - * Correlation of IP prefix to link based on type of link (P-P, P-PE, PE-CE, etc.)
 - * Correlation of IP Prefix to IGP Area
 - * Layer-2 VLAN ID assignments, etc.
- o Routing Configuration Policies
 - * OSPF Area or IS-IS Net-ID to Node (Type) Correlation
 - * BGP routing policies, i.e.: nodes designated for injection of aggregate routes, max-prefix policies, AFI/SAFI to node correlation, etc.
- o Security Policies
 - * Access Control Lists
 - * Rate-limiting
- o Network Component/Attribute Data Access Policies. Client's (upper-layer application) read-only or read-write access to Network Components/Attributes contained in the "Inventory Manager" as well as Policies contained within the "Policy Manager" itself.

The Policy Manager function may be a sub-component of the Topology or Orchestration Manager or it may be a standalone. This will be determined as the work with IRS evolves.

3.4. Orchestration Manager

The Orchestration Manager provides the ability to stitch together resources (i.e.: compute, storage) and/or services with the network or vice-versa. Examples of 'generic' services may include the following:

- o Application-specific Load Balancing
- o Application-specific Network (Bandwidth) Optimization
- o Application or End-User specific Class-of-Service
- o Application or End-User specific Network Access Control

The above services could then enable coupling of resources with the network to realize the following:

- o Network Optimization: Creation and Migration of Virtual Machines (VM's) so they are adjacent to storage in the same DataCenter.
- o Network Access Control: Coupling of available (generic) compute nodes within the appropriate point of the data-path to perform firewall, NAT, etc. functions on data traffic.

The Orchestration Manager is expected to exchange data models with the Topology Manager, Policy Manager and Inventory Manager functions. In addition, the Orchestration Manager is expected to support publish and subscribe capabilities to those functions, as well as to Clients, to enable scalability with respect to event notifications.

The Orchestration Manager may receive requests from Clients (applications) for immediate access to specific network resources. However, Clients may request to schedule future appointments to reserve appropriate network resources when, for example, a special event is scheduled to start and end.

Finally, the Orchestration Manager should have the flexibility to determine what network layer(s) may be able to satisfy a given Client's request, based on constraints received from the Client as well as those constraints learned from the Policy and/or Topology Manager functions. This could allow the Orchestration Manager to, for example, satisfy a given service request for a given Client using the optical network (via OTN service) if there is insufficient IP/MPLS capacity at the specific moment the Client's request is received.

The operational model is shown in the following figure.

TBD.

Figure 3: Overall Reference Model

4. Use Cases

4.1. Virtualized Views of the Network

4.1.1. Capacity Planning and Traffic Engineering

When performing Traffic Engineering and/or Capacity Planning of an IP/MPLS network, it is important to account for SRLG's that exist within the underlying physical, optical and Ethernet networks. Currently, it's quite common to create and/or take "snapshots", at infrequent intervals, that comprise the inventory data of the underlying physical and optical layer networks. This inventory data then needs to be massaged or normalized to conform to the data import requirements of sometimes separate Traffic Engineering and/or Capacity Planning tools. This process is error-prone and inefficient, particularly as the underlying network inventory information changes due to introduction of, for example, new network element makes or models, linecards, capabilities, etc. at the optical and/or Ethernet layers of the underlying network.

This is inefficient with respect to the time and expense consumed by software developer, Capacity Planning and Traffic Engineering resources to normalize and sanity check underlying network inventory information, before it can be consumed by IP/MPLS Capacity Planning and Traffic Engineering applications. Due to this inefficiency, the underlying physical network inventory information, (containing SRLG and corresponding critical network asset information), used by the IP/MPLS Capacity Planning and TE applications is not updated frequently, thus exposing the network to, at minimum, inefficient utilization and, at worst, critical impairments.

An Inventory Manager function is required that will, first, extract inventory information from network elements -- and potentially associated offline inventory databases to acquire physical cross-connects and other information that is not available directly from network elements -- at the physical, optical, Ethernet and IP/MPLS layers of the network via standards-based data models. Data models and associated vocabulary will be required to represent not only components inside or directly connected to network elements, but also to represent components of a physical layer path (i.e.: cross-connect panels, etc.) The aforementioned inventory will comprise the complete set of inactive and active network components.

A Statistics Collection Function is also required. As stated above, it will collect utilization statistics from Network Elements, archive and aggregate them in a statistics data warehouse. Summaries of these figures then need to be exposed in normalized data models to the Topology Manager so it can easily acquire historical link and LSP utilization figures that can be used to, for example, build trended utilization models to forecast expected changes to the physical and/or logical network components to accommodate network growth.

The Topology Manager function may then augment the Inventory Manager information by communicating directly with Network Elements to reveal the IGP-based view of the active topology of the network. This will allow the Topology Manager to include dynamic information from the IGP, such as Available Bandwidth, Reserved Bandwidth, etc. Traffic Engineering (TE) attributes associated with links, contained with the Traffic Engineering Database (TED) on Network Elements.

It is important to recognize that extracting topology information from the network solely via an IGP, (such as IS-IS TE or OSPF TE), is inadequate for this use case. First, IGP's only expose the active components (e.g. vertices of the SPF tree) of the IP network; unfortunately, they are not aware of "hidden" or inactive interfaces within IP/MPLS network elements, (e.g.: unused linecards or unused ports), or components that reside at a lower layer than IP/MPLS, e.g. Ethernet switches, Optical transport systems, etc. This occurs frequently during the course of maintenance, augment and optimization activities on the network. Second, IGP's only convey SRLG information that have been first applied within the router's configurations, either manually or programatically. As mentioned previously, this SRLG information in the IP/MPLS network is subject to being infrequently updated and, as a result, may inadequately account for critical, underlying network fate sharing properties that are necessary to properly design resilient circuits and/or paths through the network.

In this use case, the Inventory Manager will need to be capable of using a variety of existing protocols such as: NETCONF, CLI, SNMP, TL1, etc. depending on the capabilities of the network elements. The Topology Manager will need to be capable of communicating via an IGP from a (set of) Network Elements. It is important to consider that to acquire topology information from Network Elements will require read-only access to the IGP. However, the end result of the computations performed by the Capacity Planning Client may require changes to various IGP attributes, (e.g.: IGP metrics, TE link-colors, etc.) These may be applied directly by devising a new capability to either: a) inject information into the IGP that overrides the same information injected by the originating Network Element; or, b) allowing the Topology and/or Inventory Manager the

ability to write changes to the Network Element's configuration in order to have it adjust the appropriate IGP attribute(s) and re-flood them throughout the IGP. It would be desirable to have a single mechanism (data model or protocol) that allows the Topology Manager to read and write IGP attributes.

Once the Topology Manager function has assembled a normalized view of the topology and synthesized associated metadata with each component of the topology (link type, link properties, statistics, intra-layer relationships, etc.), it can then expose this information via its northbound API to Clients. In this use case that means Capacity Planning and Traffic Engineering applications, which are not required to know innate details of individual network elements, but do require generalized information about the node and links that comprise the network, e.g.: links used to interconnect nodes, SRLG information (from the underlying network), utilization rates of each link over some period of time, etc. In this case, it is important that any Client that understands both the web services API and the normalized data model can communicate with the Topology Manager in order to understand the network topology information that was provided by network elements from potentially different vendors, all of which likely represent that topology information internally using different models. If the Client had gone directly to the network elements themselves, it would have to translate and then normalize these different representations for itself. However, in this case, the Topology Manager has done that for it.

When this information is consumed by the Traffic Engineering application, it may run a variety of CSPF algorithms the result of which is likely a list of RSVP LSP's that need to be (re-)established, or torn down, in the network to globally optimize the packing efficiency of physical links throughout the network. The end result of the Traffic Engineering application is "pushing" out to the Topology Manager, via a standard data model to be defined here, a list of RSVP LSP's and their associated characteristics, (i.e.: head and tail-end LSR's, bandwidth, priority, preemption, etc.). The Topology Manager then would consume this information and carry out those instructions by speaking directly to network elements, perhaps via PCEP Extensions for Stateful PCE [I-D.ietf-pce-stateful-pce], which in turn initiates RSVP signaling through the network to establish the LSP's.

After this information is consumed by the Capacity Planning application, it may run a variety of algorithms the result of which is a list of new inventory that is required to be purchased (or, redeployed) as well as associated work orders for field technicians to augment the network for expected growth. It would be ideal if this information was also "pushed" back into the Topology and, in

turn, Inventory Manager as "inactive" links and/or nodes, so that as new equipment is installed it can be automatically correlated with original design and work order packages associated with that augment.

4.1.2. Services Provisioning

Beyond Capacity Planning and Traffic Engineering applications, having a normalized view of just the IP/MPLS layer of the network is still very important for other mission critical applications such as Security Auditing and IP/MPLS Services Provisioning, (e.g.: L2VPN, L3VPN, etc.). With respect to the latter, these types of applications should not need a detailed understanding of, for example, SRLG information, assuming that the underlying MPLS Tunnel LSP's are known to account for the resiliency requirements of all services that ride over them. Nonetheless, for both types of applications it is critical that they have a common and up-to-date normalized view of the IP/MPLS network in order to easily instantiate new services at the appropriate places in the network, in the case of VPN services, or validate that ACL's are configured properly to protect associated routing, signaling and management protocols on the network, with respect to Security Auditing.

For this use case, what is most commonly needed by a VPN Service Provisioning application is as follows. First, Service PE's need to be identified in all markets/cities where the customer has identified they want service. Next, does their exist one, or more, Services PE's in each city with connectivity to the access network(s), e.g.: SONET/TDM, used to deliver the PE-CE tail circuits to the Service's PE. Finally, does the Services PE have available capacity on both the PE-CE access interface and its uplinks to terminate the tail circuit? If this were to be generalized, this would be considered an Resource Selection function. Namely, the VPN Provisioning application would iteratively query the Topology Manager to narrow down the scope of resources to the set of Services PE's with the appropriate uplink bandwidth and access circuit capability plus capacity to realize the requested VPN service. Once the VPN Provisioning application has a candidate list of resources it then requests the Topology Manager to go about configuring the Services PE's and associated access circuits to realize the customer's VPN service.

4.1.3. Rapid IP Renumbering, AS Migration

A variety of reasons exist for the "rapid renumbering" of IPv4/IPv6 prefixes and ASN's in an IP/MPLS network. Perhaps the most common reason is as a result of mergers, acquisitions or divestitures of companies, organizations or divisions.

Inside the network of an Enterprise or Service Provider, there

already exist protocols such as DHCP or SLAAC to support rapid renumbering of hosts, (i.e.: servers, laptops, tablets, etc.). These are outside the scope of this document. However, there still exists a critical need to quickly renumber network infrastructure, namely: router interfaces, management interfaces, etc. in order to: a) avoid overlapping RFC 1918 addresses in previously separate domains; b) allow for (better) aggregation of IP prefixes within areas/domains of an IGP; c) allow for more efficient utilization of globally unique IPv4 addresses, which are in limited supply; d) realize business synergies of combining two different AS'es into one, etc.

The set of IPv4 and IPv6 prefixes that have been configured on point-to-point, LAN, Loopback, Tunnel, Management, PE-CE and other interfaces would be gathered from all network elements by the Inventory Manager function. Similarly, the set of ASN's that have been configured on individual NE's, as the global BGP Autonomous System Number, and the PE-CE interfaces is also acquired from the Inventory Manager. Afterward, an "inventory" report of the total number, based on type, of IPv4/IPv6 prefixes could be quickly assembled to understand how much address space is required to accommodate the existing network, but also future growth plans. Next, a new IP prefix and ASN would be assigned to the overall network. An operator may then decide to manually carve up the IP prefix into sub-prefixes that are assigned to various functions or interface types in the network, i.e.: all Loopback interface addresses are assigned from a specific GUA IPv4/IPv6 prefix. Other rules may be crafted by the operator so that, for example, GUA IPv4/IPv6 prefixes for interfaces within each IGP area are assigned out of contiguous address space so that they may be (easily) summarized within the IGP configuration. Finally, the set of ASN's, IP prefixes, rules and/or policies governing how their are to be assigned are encoded in a data model/schema and sent to a Topology Manager (TM). The Topology Manager is then responsible for communicating changes to the Inventory Manager and/or Network Elements in a proper sequence, or order of operations, so as to not lose network connectivity from the Topology Manager to the network elements.

This function could be extended further whereby the Orchestration Manager would be used in order to automatically create a list of IP addresses and their associated DNS names, which would then be "pushed" to Authoritative DNS servers so that interface names would get updated in DNS automatically. In addition, the Orchestration Manager function could notify a "Infrastructure Security" application that IP prefixes on the network has changed so that it then updates ACL's used to, for example, protect IP/MPLS routing and signaling protocols used on the network.

4.1.4. Troubleshooting & Monitoring

Once the Topology Manager has a normalized view of several layers of the network, it's then possible to more easily expose a richer set of data to network operators when performing diagnosis, troubleshooting and repairs on the network. Specifically, there is a need to (rapidly) assemble a current, accurate and comprehensive network diagram of a L2VPN or L3VPN service for a particular customer when either: a) attempting to diagnose a service fault/error; or, b) attempting to augment the customer's existing service. Information that may be assembled into a comprehensive picture could include physical and logical components related specifically to that customer's service, i.e.: VLAN's or channels used by the PE-CE access circuits, CoS policies, historical PE-CE circuit utilization, etc. The Topology Manager would assemble this information, on behalf of each of the network elements and other data sources in and associated with the network, and could present this information in a vendor-independent data model to applications to be displayed allowing the operator (or, potentially, the customer through a SP's Web portal) to visualize the information.

4.2. Path Computation Element (PCE)

As described in [RFC4655] a PCE can be used to compute MPLS-TE paths within a "domain" (such as an IGP area) or across multiple domains (such as a multi-area AS, or multiple ASes).

- o Within a single area, the PCE offers enhanced computational power that may not be available on individual routers, sophisticated policy control and algorithms, and coordination of computation across the whole area.
- o If a router wants to compute a MPLS-TE path across IGP areas its own TED lacks visibility of the complete topology. That means that the router cannot determine the end-to-end path, and cannot even select the right exit router (Area Border Router - ABR) for an optimal path. This is an issue for large-scale networks that need to segment their core networks into distinct areas, but which still want to take advantage of MPLS-TE.

The PCE presents a computation server that may have visibility into more than one IGP area or AS, or may cooperate with other PCEs to perform distributed path computation. The PCE needs access to the topology and the Traffic Engineering Database (TED) for the area(s) it serves, but [RFC4655] does not describe how this is achieved. Many implementations make the PCE a passive participant in the IGP so that it can learn the latest state of the network, but this may be sub-optimal when the network is subject to a high degree of churn, or

when the PCE is responsible for multiple areas.

The following figure shows how a PCE can get its TED information using a Topology Server.

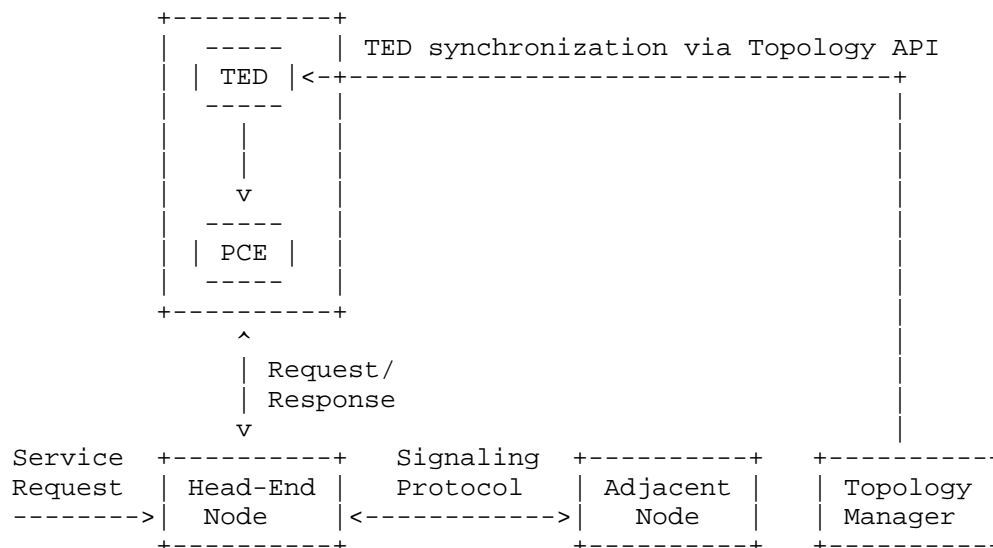


Figure 4: Topology use case: Path Computation Element

4.3. ALTO Server

An ALTO Server [RFC5693] is an entity that generates an abstracted network topology and provides it to network-aware applications over a web service based API. Example applications are p2p clients or trackers, or CDNs. The abstracted network topology comes in the form of two maps: a Network Map that specifies allocation of prefixes to PIDs, and a Cost Map that specifies the cost between PIDs listed in the Network Map. For more details, see [I-D.ietf-alto-protocol].

ALTO abstract network topologies can be auto-generated from the physical topology of the underlying network. The generation would typically be based on policies and rules set by the operator. Both prefix and TE data are required: prefix data is required to generate ALTO Network Maps, TE (topology) data is required to generate ALTO Cost Maps. Prefix data is carried and originated in BGP, TE data is originated and carried in an IGP. The mechanism defined in this document provides a single interface through which an ALTO Server can retrieve all the necessary prefix and network topology data from the underlying network. Note an ALTO Server can use other mechanisms to get network data, for example, peering with multiple IGP and BGP

Speakers.

The following figure shows how an ALTO Server can get network topology information from the underlying network using the Topology API.

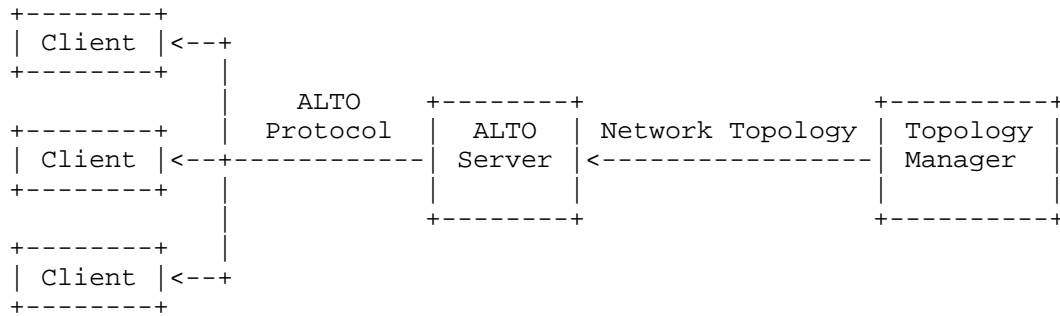


Figure 5: Topology use case: ALTO Server

5. Acknowledgements

The authors wish to thank Alia Atlas, Dave Ward, Hannes Gredler, Stefano Previdi for their valuable contributions and feedback to this draft.

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

At the moment, the Use Cases covered in this document apply specifically to a single Service Provider or Enterprise network. Therefore, network administrations should take appropriate precautions to ensure appropriate access controls exist so that only internal applications and end-users have physical or logical access to the Topology Manager. This should be similar to precautions that are already taken by Network Administrators to secure their existing Network Management, OSS and BSS systems.

As this work evolves, it will be important to determine the appropriate granularity of access controls in terms of what individuals or groups may have read and/or write access to various types of information contained with the Topology Manager. It would

be ideal, if these access control mechanisms were centralized within the Topology Manager itself.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

8.2. Informative References

- [I-D.atlas-irs-problem-statement]
Atlas, A., Nadeau, T., and D. Ward, "Interface to the Routing System Problem Statement", draft-atlas-irs-problem-statement-00 (work in progress), July 2012.
- [I-D.ietf-alto-protocol]
Alimi, R., Penno, R., and Y. Yang, "ALTO Protocol", draft-ietf-alto-protocol-13 (work in progress), September 2012.
- [I-D.ietf-pce-stateful-pce]
Crabbe, E., Medved, J., Varga, R., and I. Minei, "PCEP Extensions for Stateful PCE", draft-ietf-pce-stateful-pce-01 (work in progress), July 2012.
- [I-D.ward-irs-framework]
Atlas, A., Nadeau, T., and D. Ward, "Interface to the Routing System Framework", draft-ward-irs-framework-00 (work in progress), July 2012.
- [RFC4655] Farrel, A., Vasseur, J., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, August 2006.
- [RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, October 2009.

Authors' Addresses

Shane Amante
Level 3 Communications, Inc.
1025 Eldorado Blvd
Broomfield, CO 80021
USA

Email: shane@level3.net

Jan Medved
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134
USA

Email: jmedved@cisco.com

Thomas D. Nadeau
Juniper Networks
1194 N. Mathilda Ave.
Sunnyvale, CA 94089
USA

Email: tnadeau@juniper.net

Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 15, 2013

A. Atlas
Juniper Networks
S. Hares
Huawei Technologies
J. Halpern
Ericsson
September 11, 2012

A Policy Framework for the Interface to the Routing System
draft-atlas-irs-policy-framework-00

Abstract

A key aspect of the Interface to the Routing System (IRS) is what mechanisms it includes to carry policy information and to enable policy control. This applies both in the protocol itself and in the sub-interfaces associated with the different components of the routing system. Similarly, the data-models associated with the sub-interfaces must be capable of expressing the appropriate granularity for access and authorization-related policy. This document describes the policy framework for IRS.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 15, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. General IRS Policy	4
3.1. Policy between commissioner and agent	6
3.1.1. Identity	6
3.1.2. Security Role	6
3.1.3. Security Model	7
3.1.4. Scope and Influence	7
3.1.5. Resources	8
3.1.6. Connectivity	9
3.1.7. Priority	9
3.1.8. Precedence	10
3.2. Policy between Agent and Local System	12
3.2.1. Local Configuration	13
3.2.2. Removal of IRS-installed State	14
3.2.3. On Reboot	14
4. Policy in a Sub-Interface	15
4.1. Resource Reservation and Three-Phase Commit	15
4.2. Defining IRS Behavior Based on Implicit and Explicit Policy	15
4.2.1. Example of Implicit Policy	16
4.2.2. Passing Explicit Policy	17
4.2.2.1. Explicit policy on Data Forwarding, Resources, and Policy passing	17
4.2.2.2. Example of Explicit Policy	17
5. Acknowledgements	18
6. IANA Considerations	18
7. Security Considerations	18
8. Informative References	18
Authors' Addresses	18

1. Introduction

The Interface to the Routing System (IRS) provides read and write access to the information and state that enable the routing components of routing elements. The IRS is introduced and described in [I-D.atlas-irs-problem-statement] and [I-D.ward-irs-framework].

Policy helps provide filters and control on the level of access to information and state that is enabled by individual protocol interactions. A clear view of the policy features desirable at the IRS is important to shape the architecture and requirements for the protocols and sub-interfaces of the IRS. Policy can be explicitly defined or implicitly assumed in a system, and can be enforced by that system's rules and behavior. Since IRS provides sub-interfaces to routing sub-systems that already have policy defined (implicitly or explicitly), it is important to consider the existing policy mechanisms and how an IRS sub-interface should interact with them.

IRS policy has four different aspects that need to be considered.

1. Policy related to the IRS protocol interactions between different systems.
2. Policy related to the interaction between the IRS Agent and the local system to which the IRS Agent is providing an interface.
3. Sub-interface policy to support scope and influence restrictions and to preserve necessary policy associated with the related routing sub-system.
4. Policy that can be installed or read via a sub-interface's data-model that is associated with the related routing sub-system.

2. Terminology

The following memorable terminology is used in this document.

agent or IRS Agent: An IRS Agent provides the supported IRS sub-interfaces to the local system's routing sub-systems. The IRS Agent understands the IRS protocol and can be contacted by commissioners.

commissioner: A commissioner speaks the IRS protocol to communicate with IRS Agents and uses the IRS sub-interfaces to accomplish a task as instructed by the commissioner's local application. A commissioner can be seen as the part of an application that supports IRS and could be a software library.

scope: The set of information which the particular IRS entity (agent or commissioner) is authorized to read. This access includes the permission to see the existence of data and the ability to retrieve the value of that data. In the context of an interaction between a commissioner and an agent, the effective scope is restricted to the intersection of the scopes of the two entities.

influence: The set of field values which the particular IRS entity (agent or commissioner) is authorized to install. This access can restrict what fields can be modified or created, and what specific value sets and ranges can be installed. In the context of an interaction between a commissioner and an agent, the effective influence is restricted to the intersection of the influences of the two entities.

resources: A resource is an IRS-specific use of memory, storage, or execution that a commissioner may consume due to its IRS operations. The amount of each such resource that a commissioner may consume in the context of a particular agent can be constrained. Examples of such resources could include: the number of installed operations, number of operations that haven't reached their start-time, etc. These are not protocol-specific resources or network-specific resources.

role or security role: A security role specifies the scope, influence, resources, precedence values, etc. that a commissioner or agent has.

identity: A commissioner is associated with exactly one specific identity. State installed by a particular identity is owned by that identity; state ownership can not be transferred. It is possible for multiple communication channels to use the same identity; in that case, the assumption is that the associated commissioner is coordinating such communication. Similarly, an agent is associated with a specific identity.

3. General IRS Policy

IRS needs its own implicit and explicit policy. This section articulates some of the those key concepts and policy decisions. The IRS policy applies to interactions between the agent and commissioners and between the agent and the local system.

The agent's externally perceivable behavior and associated policy is a key aspect of IRS that must be described. The commissioner's behavior and functionality is specifically out-of-scope except where

it needs to be described with respect to the agent's behavior and the IRS protocol.

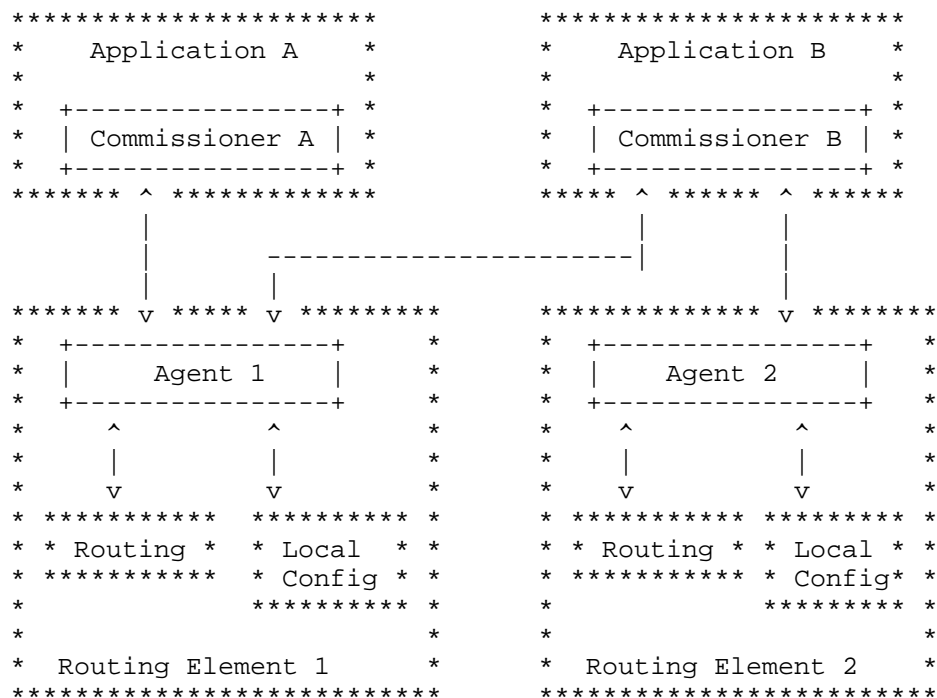


Figure 1: Architecture of commissioners and agents

As can be seen in Figure 1, a commissioner can communicate with multiple agents. The application associated with a commissioner may have multiple tasks it is accomplishing (separate functions, short-term versus longer-term, etc) and each such task may involve a set of agents which may or may not differ.

As can also be seen in Figure 1, an IRS Agent may communicate with multiple commissioners. Each commissioner may send the agent a variety of install operations. The set of install operations received by an agent may overlap and conflict. No simple protocol or policy mechanisms by an agent can completely avoid indirect interactions between different install operations. The functional partitioning between the different commissioners must be done to avoid undesirable indirect interactions.

3.1. Policy between commissioner and agent

Multiple commissioners can communicate with the same agent. The agent must have policies to manage the resulting complexity. Implicit policy includes the assumptions about communication between the commissioner and agent. Explicit policy includes mechanisms to arbitrate between different commissioners, between operations of the same commissioner, and to manage state owned by an commissioner inside the agent.

3.1.1. Identity

By definition, a commissioner is associated with exactly one identity. An agent will store data that is owned by a particular commissioner, based upon that commissioner's identity. Since a commissioner can communicate via multiple transport channels and no channel needs to be active for the agent to have associated state, the commissioner's identity is used to identify the ownership of the data stored by the agent.

Similarly, by definition, an agent is associated with exactly one identity. An commissioner may also store local state associated with a particular agent. The agent's identity can be used to identify ownership of the data stored by the commissioner.

The details of what constitutes an identity can be dependent upon the specifics of the IRS protocol and selected security mechanisms. However, there are some critical considerations for identity that do impose constraints.

An identity is not tied to a single communication channel. A commissioner may use multiple IP addresses; an identity should not be tied to a specific IP address. If the commissioner or agent is associated with a system that may be mobile, that should be considered in its identification. Finally, the syntax and semantics for identifiers used for a commissioner and for an agent may be different.

3.1.2. Security Role

In the context of an agent, each commissioner will have a security role. The commissioner's identity and associated security role will have to be verified via an acceptable security mechanism. A variety of such mechanisms are anticipated to meet different security and operational objectives. Example mechanisms might include a role assertion from the commissioner to the agent that the agent can cryptographically verify or having the agent to use an already trusted protocol to verify the commissioner's security role and

identity.

An agent must know the scope, influence, and resources associated with each particular security role. This information may vary across different agents even in the same network or it may be consistent across different agents in the same network. The latter can be enforced by having a commissioner that is authorized to influence the meta-data model of security roles on the relevant set of agents.

A security role also defines what precedence values (See Section 3.1.8) a commissioner can use.

3.1.3. Security Model

As described above, roles identify the scope, influence, and resources allowed to an IRS Commissioner. The policy model therefore needs to include these roles. The question of the bindings of identities to roles, and the selection of identities are protocol specific matters outside the scope of this document.

The policy model for roles needs to address two dimensions. It needs to create the roles themselves. This should allow for use of techniques like inheritance, presumably with some rules on how role definitions can augment or restrict the inherited definitions.

The security model also needs to define, by reference to the policy model itself, the scope and influence of the role. The question of defining the resources of a role is for further study. The role definition needs to indicate what types and instances of data can be observed and what information about those instances entities with that role can observe. The security model also needs to define which data items can be modified, and what modifications (ranges, specified values, or other assertions that must be met) are permitted.

3.1.4. Scope and Influence

Scope and influence are specified as part of a security role. A security role may be defined and managed in an external repository, centralized within an administration. The security role definitions must be accessible to an agent.

In the context of an interaction between a commissioner and an agent, the effective scope or influence is restricted to the intersection of the scopes or influence of the two entities.

What information a particular commissioner is authorized to read is known as the commissioner's scope. A scope includes the ability to see that particular data exists and to read the same data. The scope

can have its constraints specified in terms of specific portions of data models.

Similarly, what information a commissioner can install may be constrained. This is known as its influence. The influence is specific in both the parts of the data models and in the set and range of data that can be installed. For example, a commissioner might be able to write static routes in the RIB data-model for prefixes in 10.0/16.

While the commissioner's behavior and functionality is specifically out-of-scope, it is useful to describe the same scope and influence concepts for an agent operating in the context of a commissioner.

An agent's scope is the set of data that the agent can read or have access to. An agent would generally learn such data because the commissioner has sent that data to the agent in an operation.

An agent's influence is the set and range of data that the agent is allowed to provide to the commissioner and that will be accepted by the commissioner. For instance, commissioner B may accept next-hop change notifications for prefix 10.0/16 from agent 1 but not from agent 2.

3.1.5. Resources

When a commissioner sends operations to an agent, those operations can consume resources. Therefore, it is important that the agent have policy to limit the resources available to a particular commissioner. This is based on the commissioner's identity and security role. Such resource policy specifications need to be provided in a data-model that can be modified by appropriately authorized commissioners or local configuration.

Examples of such resource constraints include:

- Number of installed operations owned,

- Number of operations that haven't reached their start-time, and

- Number of event notifications registered for.

As discussed in Section 3.1.7, a commissioner can specify priorities for the operations it sends.

If compute resources are considered, it is not the intent to try and determine the computation associated with particular operations. Instead, the constraint could be on amount of compute-time given to a

commissioner every pre-defined period. This can provide a mechanism for fair sharing of compute resources between commissioners.

3.1.6. Connectivity

A commissioner does not need to maintain an active communication channel with an agent. Therefore, an agent may need to open a communication channel to the commissioner to communicate previously requested information. The lack of an active communication channel does not imply that the associated commissioner is non-functional. When communication is required, the agent or commissioner can open a new communication channel.

State held by an agent that is owned by a commissioner should not be removed or cleaned up when a commissioner is no longer communicating - even if the agent cannot successfully open a new communication channel to the commissioner.

3.1.7. Priority

The motivating example for priority is when a single commissioner is sending operations to accomplish multiple tasks. For example, one task might be long-term and another task might deal with unexpected requests that are more important. In this case, the commissioner may wish to provide a hint to the relevant agents as to which operations should be done first.

Communication from a commissioner can come across multiple channels, so simply specifying that operations be done in order is not sufficient. Additionally, all operations may not be immediately carried out, due to varying start-times or other constraints. With these factors and this motivating example, it is useful to introduce the concept of prioritization for operations sent from the same commissioner.

By introducing the concept of priority for operations, a commissioner can accomplish multiple uncorrelated tasks that affect the same agent with the specified prioritization.

A default priority can be specified for each particular communication channel. In addition, an IRS operation can specify a priority to use instead. Priorities between operations from different commissioners need not be compared.

The priority can be used by an agent to determine which operation from a commissioner to execute next.

3.1.8. Precedence

A mechanism is needed for the agent to determine what state to install when there are overlapping install operations. An install operation may overlap with locally-installed configuration state or with a previous install operation that was requested by a commissioner. The mechanism to resolve this is termed "precedence". No simple mechanism can fully handle indirect interactions; considering such interactions is out-of-scope. Indirect interactions must be considered when different commissioners are given their tasks.

A critical aspect of precedence-based decisions is that preference is only given based on arrival time of the install operation when multiple commissioners use the same precedence value.

Each install operation has a precedence associated with it. This precedence may come from the default associated with the commissioner, with the specific communication channel, or with the specific operation. The range of possible precedence values that can be used is known based on the commissioner's security role. The determination of the precedence associated with any operation is a policy decision at the agent, but may utilize any or all of the information described above.

When an install operation is executed, the agent first determines if there is overlapping existing IRS-installed state. If not, the agent must determine if it overlaps existing local-configuration state. Local-configuration state will also have a precedence associated with it so that the agent can make an appropriate decision.

A commissioner can specify whether an install operation should be store-if-not-best. This allows a commissioner to determine what happens when an install operation doesn't win the precedence comparison. If store-if-not-best is specified, then the install operation succeeds and the associated installed state is stored but not actively installed by the agent. If store-if-not-best is not specified, then the install operation will fail.

The store-if-not-best flag is stored with the installed operation's precedence. If the agent determines that an installed operation must be preempted, then the agent consults the store-if-not-best flag. If store-if-not-best is specified, then the agent stores the preempted operation and does not notify the associated commissioner. If store-if-not-best is not specified, then the agent notifies the associated commissioner of the preemption and removes the previously installed state.

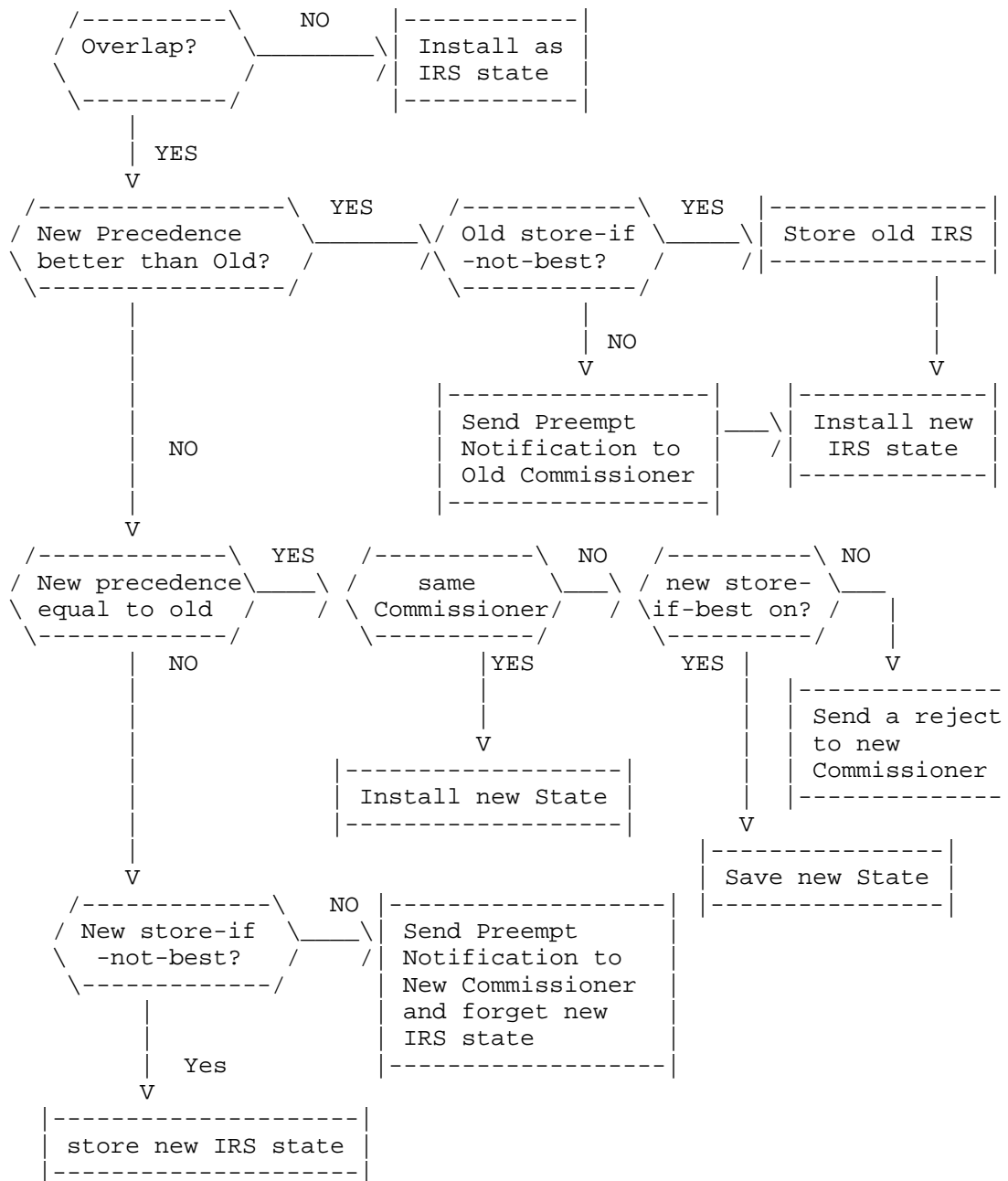


Figure 2: Precedence Decision-Making

If the overlapping new operation has a precedence that is better than

the existing state, then the agent should preempt the existing state and act according to the existing state's store-if-not-best flag. If that store-if-not-best flag is set, the agent will store the old state and install the new state. If the store-if-not-best flag is clear, the agent will send a preemption notification to the Old IRS Commissioner, install the new IRS state, and forget the old.

If the overlapping existing state has the same precedence and the same commissioner associated, then the agent completes the install operation; otherwise, the agent must reject or store the install operation, based on the store-if-not-best flag.

If the new overlapping operation has a precedence that is worse than the existing state, then the agent must reject or store the install operation, based on the state of the new store-if-not-best flag. If the store-if-not-best flag is set, then then the agent will store the new IRS state. If the store-if-not-best flag is clear, then the the IRS agent will send a preempt notification to the new commissioner and forget the new IRS state.

This decision process is illustrated in Figure 2.

When an uninstall operation (e.g. remove) is done, the stored state with the next best precedence should be selected and installed.

A consequence of the precedence policy mechanism is that a commissioner must be able to handle its installed operations being preempted at any time, either explicitly or simply by having the active state changed. Such preemption can be minimized by appropriate separation of tasks, with their associated install operations, between the local systems of the commissioners and by knowledgeable local system configuration.

3.2. Policy between Agent and Local System

It is critical to understand and clearly specify how IRS interacts with local configuration. The key questions are:

1. What happens when Local Configuration overlaps with IRS installed state?
2. What happens when IRS installed state is removed?
3. How is state recreated when a local system reboots?

A consequence of using IRS is that the local system's state may not be synchronized with the local configuration. Since this is a change in understood behavior, any discrepancies should be clearly visible

to the operator with an associated explanation.

Logically, the local configuration is essentially modeled as a local commissioner, with its own precedence, identity, and security role and immediate permanent install operations. The key differences are both that all relevant local configuration state need not be cached by the agent and that reboot imposes the need to process local configuration state before any other IRS-installed state.

3.2.1. Local Configuration

The local system's local configuration may have overlapping influence with that of one or more commissioners using an agent. Therefore, explicit and implicit policy interactions must be specified. The mechanism that IRS provides for deciding between overlapping install operations is "precedence". This same mechanism can be used to decide between local configuration and an IRS operation. Local configuration can specify the precedence value to be used for the local system.

A precedence value that causes the desired behavior can be specified as follows. (MAX is the highest precedence given to a commissioner. MIN is the lowest precedence given to a commissioner.)

MAX+1 Precedence: If the local configuration has a precedence higher than that given to any commissioner, then state from the local configuration will always be installed. If any IRS-installed state is therefore preempted, the agent will notify the associated commissioner.

MIN-1 Precedence: If the local configuration has a precedence lower than that given to any commissioner, then IRS-installed state will always override local configuration. That this preemption has occurred should be reflected in how the local system displays its state.

Other Precedence: The local configuration can have higher precedence than that given to some commissioners, lower precedence than that given to other commissioners, and equal precedence to that given to other commissioners. Then some local configuration state may be preempted by IRS-installed state while some IRS-installed state can be preempted by local configuration.

Local-configuration wins all precedence ties.

Just as an agent must check to determine if an install operation overlaps with existing installed state, the process of committing local configuration must check to see if there is overlapping IRS-

installed state.

What the process of committing local configuration is can vary by local system. Well known examples are when a return is sent to the CLI and when an explicit commit command is specified. How the proper checks for interaction between the agent and local configuration are done is a local system matter.

Similarly, when an agent checks to see if an install operation overlaps with existing installed state, the agent must determine if it overlaps with existing local configuration.

If the precedence associated with local configuration is changed, then it is retroactive. All local configuration state stored by the agent must be updated with the new precedence value and installation decisions made for overlapping data. This change could be very disruptive.

3.2.2. Removal of IRS-installed State

When a piece of local configuration is removed, the local system goes back to the appropriate system default. However, when an operation removes some IRS-installed state is removed, the correct behavior is not to just go back to the system default. Instead, any stored state must be considered - whether that comes from local configuration or stored IRS install operations that didn't have the highest precedence. If there is any stored state, then the highest precedence of the options is selected and installed. That existing overlapping state might come from the local -configuration.

If IRS's implicit policy were to just go to the system default, then the local configuration and the local system state would not be synchronized and there would be no remaining IRS-state to explain the discrepancy. Since IRS state can also be stored and not installed, the same mechanism can be used for stored IRS install operations and for local configuration.

3.2.3. On Reboot

When the local system reboots, only persistent IRS-installed state is preserved by the agent. The implicit policy for IRS is that the local configuration is read and installed first. After the local system has its local configuration installed, the persistent IRS install operations are executed to bring the system to the persistent state.

4. Policy in a Sub-Interface

It is critical to consider how policy influences a sub-interface when defining the sub-interface and its associated data-model(s). There are several different aspects to consider.

How are scope and influence policy specified in the data model? What granularity levels are necessary for the particular sub-interface?

How does the implicit policy in the associated routing sub-system effect what IRS can be allowed to influence?

Are the implicit policies of the associated routing sub-system captured in the semantic content of the information model, data model, and description?

What explicit policy communicated in the associated routing sub-system needs to be included in the data-model? What indirection and abstractions are needed?

4.1. Resource Reservation and Three-Phase Commit

Some agents and sub-interfaces may offer the ability to reserve resources required by operations before the operation start time. There are two aspects to how to support this.

First, if the agent can do time-aware resource reservation, then an install operation can specify "reserve-only" to prompt an acknowledgement or failure as to the ability of the agent to confirm the reservation. Then the commissioner can either send an operation to commit the reservation, which causes the associated install operation, or to remove the reservation. A "reserve-only" operation will have its reservation expire at the end of its associated life-time.

Second, part of a sub-interface's data-model may be to request a reservation with a known start-time and duration. An example might be reserving a specific bandwidth on a path for an LSP between two devices. It is important to consider whether a particular sub-interface should offer a time-based reservation service as part of its data-model.

4.2. Defining IRS Behavior Based on Implicit and Explicit Policy

The semantics in a data-model must respect and describe the implicit policy of the associated routing sub-system. This doesn't imply that the data-model components should instantiate it or allow reading or

writing.

Policy Routing systems must deal with the verification, reading and installing of routes from sources such as EGP, IGP, and static routes. Policy routing may also control forwarding and the monitoring of data forwarding; and data resources. The explicit policy examples are given for the routing framework. It is assumed the reader can extend this framework to the data forwarding and data resource arena.

4.2.1. Example of Implicit Policy

The ISIS protocol specification uses implicit policy to set constraints on level 1 peers. Due to this fact, many ISIS implementations only let one level 1 ISIS peer associate with one Level 2 peer domain.

This policy is not encoded in any local configuration directly, but is rather included as an implicit policy. When local configuration policy is checked (prior to a configuration commit), this local policy is checked. If the configuration input from a CLI is in error, the input will be rejected, and the CLI will warn the user. Similarly programmatic interfaces for the local configuration cause the implicit policy to be checked.

IRS data models guide the commissioner in an interoperable interaction with the reading and installation of data at a particular agent. The IRS data models must contain both the implicit policy and the explicit policy. Although an agent may not report the IRS implicit policy in the protocol, the commissioner must know of the existence of the implicit policy.

This knowledge allows the commissioner to know the implicit policy interactions on different systems in a heterogeneous network. For example, assume a situation where a commissioner is talking to two agents - one on system A and one on system B. The routing process on system A has has different implicit rules for the ISIS Level 1 peer to Level 2 peer connection than the routing process on system B. Routing process A is built to allow one level 1 ISIS peer associated with 2 ISIS Level 2 peers. Routing process B upholds the standard implicit policy that 1 level ISIS peer can only be associated with 1 ISIS Level 2 peer. The commissioner setting up the ISIS peering in a network containing system A and system B must know that System A will allow a level 1 peer to connect to 2 ISIS Level 2 peers. When the commissioner's scope allows it to read data from system A and system B, it should not flag the difference in ISIS level 1 peer connections as a problem. Instead the commissioner will need to determine if the use of the different configurations can cause a network problem.

4.2.2. Passing Explicit Policy

Routing systems' explicit policy controls protocols, associates/deassociates interfaces, route verification policy, route forwarding policy, route aggregation policy, and route deaggregation policy. All of this policy can be found in the detailed configuration specification of a routing process. However, even via CLI, it is rarely possible to configure all the possible options. Other configuration mechanisms do not have public models for all the private router configuration. The developers of a routing system often have a complete policy model either in formal modeling languages or informal language.

Explicit policy contained in an IRS data model is the detailed configuration model at the deepest level that an agent can access. This detailed configuration model may come from IETF Standards and/or the vendor specific configurations. The public data models must specify a vendor specific tree where the individual configuration is plugged into.

4.2.2.1. Explicit policy on Data Forwarding, Resources, and Policy passing

Forwarding policy has to do with the data flow may also be controlled by an agent. If so, the explicit policy must be placed in a data model along with the implicit policy.

Lastly, protocols have begun to pass explicit policy about passing policy. Examples of this type of policy are BGP ORFs, BGP Flowspecs, and ISIS policy passing. Commissioners must know the implicit policy and explicit policy this policy impacts, and the precedence between these policy. Due to the extensive use of BGP ORFs and the growing use in BGP Flowspecs policy, early data models for BGP should describe the implicit policy, explicit policy, policy precedence for the BGP ORFs and BGP FlowSpecs, and how they interacts with other BGP, route policy and preferences. This information should be placed inside an IRS Data Model for an Agent supporting these features.

These explicit models for BGP policy are not trivial, but these models exist today. Frequently, IRS data models may be simply a casting of existing implicit policy and explicit policy into a common standard form so that programmatic interfaces may interact with a routing element.

4.2.2.2. Example of Explicit Policy

There are two clear explicit policy pieces for ISIS. First is the peer level. Second is the policy of the external routes to be

redistributed into and out of ISIS.

5. Acknowledgements

The authors would like to thank Ross Callon, Adrian Farrel, David Meyer, David Ward, Rex Fernando, Russ White, Bruno Risjman, and Thomas Nadeau for their suggestions and review.

6. IANA Considerations

This document includes no request to IANA.

7. Security Considerations

This is empty boilerplate for now.

8. Informative References

[I-D.atlas-irs-problem-statement]
Atlas, A., Nadeau, T., and D. Ward, "Interface to the
Routing System Problem Statement",
draft-atlas-irs-problem-statement-00 (work in progress),
July 2012.

[I-D.ward-irs-framework]
Atlas, A., Nadeau, T., and D. Ward, "Interface to the
Routing System Framework", draft-ward-irs-framework-00
(work in progress), July 2012.

Authors' Addresses

Alia Atlas
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

Email: akatlas@juniper.net

Susan Hares
Huawei Technologies

Email: shares@ndzh.com

Joel Halpern
Ericsson

Email: Joel.Halpern@ericsson.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 31, 2013

A. Atlas, Ed.
T. Nadeau
Juniper Networks
D. Ward
Cisco Systems
July 30, 2012

Interface to the Routing System Problem Statement
draft-atlas-irs-problem-statement-00

Abstract

As modern networks grow in scale and complexity, the need for rapid and dynamic control increases. With scale, the need to automate even the simplest operations is important, but even more critical is the ability to quickly interact with more complex operations such as policy-based controls.

In order to enable applications to have access to and control over information in the Internet's routing system, we need a publically documented interface specification. The interface needs to support real-time, transaction-based interactions using efficient data models and encodings. Furthermore, the interface must support a variety of use cases including those where verified control feed-back loops are needed.

This document expands upon these statements of requirements to provide a problem statement for an interface to the Internet routing system.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 31, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. IRS Model and Problem Area for The IETF	3
3. Standard Data-Models of Routing State for Installation	5
4. Learning Router Information	5
5. Desired Aspects of a Protocol for IRS	6
6. Existing Management Interfaces	7
7. Acknowledgements	8
8. IANA Considerations	8
9. Security Considerations	8
10. Informative References	9
Appendix A. Gaps and Concerns for SNMP	9
Appendix B. Gaps and Concerns with NetConf	10
Authors' Addresses	11

1. Introduction

As modern networks grow in scale and complexity, the need for rapid and dynamic control increases. With scale, the need to automate even the simplest operations is important, but even more critical is the ability to quickly interact with more complex operations such as policy-based controls.

With complexity comes the need for more sophisticated automated applications and orchestration software that can process large quantities of data, run complex algorithms, and adjust the routing state as required in order to support the applications, their calculations and their policies. Changes made to the routing state of a network by external applications must be verifiable by those applications to ensure that the correct state has been installed in the right places.

Mechanisms to support the requirements outlined above have been developed piecemeal as proprietary solutions to specific situations and needs. A standard protocol, clear operations that an application can initiate with that protocol, and data-models to support such actions would facilitate wide-scale deployment of interoperable applications and routing systems. That a protocol designed to facilitate rapid, isolated, secure, and dynamic routing changes is needed motivates the creation of an Interface to The Routing System (IRS).

2. IRS Model and Problem Area for The IETF

Managing a network of deployed devices running a variety of routing protocols involves interactions among multiple different functions and components that exist within the network. Some of these components are virtual while some are physical; all should be made available to be managed and manipulated by applications, given that appropriate access, authentication, and policy hurdles have been crossed. The management of only some of these components requires standardization, as others have already been standardized. The IRS model is intended to incorporate existing mechanisms where appropriate, and to build extensions and new protocols where needed. The IRS model and problem area proposed for IETF work is illustrated in Figure 1.

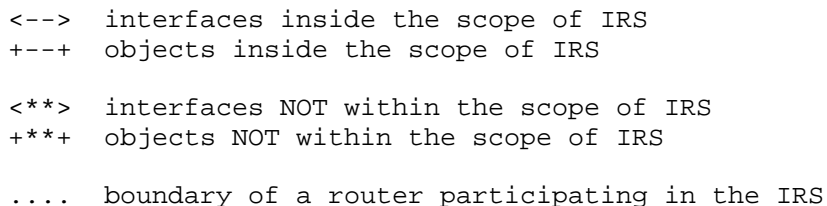


Figure 1: IRS model and Problem Area

A critical aspect of IRS is defining a suitable protocol or protocols to carry messages between the IRS Clients and the IRS Agent, and defining the encapsulation of data within those messages. This should provide a clear transfer syntax that is straightforward for applications to use (e.g., a Web Services design paradigm), and should provide the key features specified in Section 5.

The second critical aspect is semantic-aware data-models for information in the routing system and in a topology database. The data-models should be separable across different features of the managed components, versioned, and combine to provide a network data-model.

3. Standard Data-Models of Routing State for Installation

There is a need to be able to precisely control routing and signaling state based upon policy or external measures. This can range from simple static routes to policy-based routing to static multicast replication and routing state. This means that the data model employed needs to handle indirection as well as different types of tunneling and encapsulation. The relevant MIB modules (for example [RFC4292]) lack the necessary generality and flexibility. In addition, by having IRS focus initially on interfaces to the RIB layer (e.g. RIB, LFIB, multicast RIB, policy-based routing), the ability to use routing indirection allows flexibility and functionality that can't be as easily obtained at the forwarding layer.

Efforts to provide this level of control have focused on standardizing data models that describe the forwarding plane (e.g. ForCES [RFC3746]). IRS recognizes that the routing system and a router's OS provide useful mechanisms that applications could usefully harness to accomplish application-level goals.

In addition to interfaces to the RIB layer, there is a need to configure the various routing and signaling protocols with differing dynamic state based upon application-level policy decisions. The range desired is not available via MIBs at the present time.

4. Learning Router Information

A router has information that applications may require so that they can understand the network, verify that programmed state is installed in the forwarding plane, measure the behavior of various flows, and

understand the existing configuration and state of the router. IRS provides a framework for applications to register for asynchronous notifications and for them to make specific requests for information.

Although there are efforts to extend the topological information available, even the best of these (e.g., BGP-LS [I-D.gredler-idr-ls-distribution]) still provides only the current active state as seen at the IGP layer and above. Detailed topological state that provides more information than the current functional status is needed by applications; only the active paths or links are known versus those desired or unknown to the routing topology.

For applications to have a feedback loop that includes awareness of the relevant traffic, an application must be able to request the measurement and timely, scalable reporting of data. While a mechanism such as IPFIX [RFC5470] may be the facilitator for delivering the data, the need for an application to be able to dynamically request that measurements be taken and data delivered is critical.

There are a wide range of events that applications could use for either verification of router state before other network state is changed (e.g. that a route has been installed), to act upon changes to relevant routes by others, or upon router events (e.g. link up/down). While a few of these (e.g. link up/down) may be available via MIB Notifications today, the full range is not - nor is there the ability to set up the router to trigger different actions upon an event's occurrence.

5. Desired Aspects of a Protocol for IRS

This section describes required aspects of a protocol that could support IRS. Whether such a protocol is built upon extending existing mechanisms or requires a new mechanism requires further investigation.

The key aspects needed in an interface to the routing system are:

Multiple Simultaneous Asynchronous Operations: A single application should be able to send multiple operations to IRS without needing to wait for each to complete before sending the next.

Configuration Not Re-Processed: When an IRS operation is processed, it does not require that any of the configuration be processed. I.e., the desired behavior is orthogonal to the static configuration.

Duplex: Communications can be established by either the router or the application. Similarly, events, acknowledgements, failures, operations, etc. can be sent at any time by both the router and the application. The IRS is not a pure pull-model where only the application queries to pull responses.

High-Throughput: At a minimum, the IRS Agent and associated router should be able to handle hundreds of simple operations per second.

Responsive: It should be possible to complete simple operations within a sub-second time-scale.

Multi-Channel: It should be possible for information to be communicated via the interface from different components in the router without requiring going through a single channel. For example, for scaling, some exported data or events may be better sent directly from the forwarding plane, while other interactions may come from the control-plane. Thus a single TCP session would not be a good match.

Timing of State Installation and Expiration: The ability to have state installed with different lifetimes and different start-times is very valuable. In particular, the ability of an IRS client to request that a pre-sent operation be started based upon a dynamic event would provide a powerful functionality.

To extract information in a scalable fashion that is more easily used by applications, the ability to specify filtering constructs in an operation requesting data or requesting an asynchronous notification is very valuable.

6. Existing Management Interfaces

This section discusses the combination of the abstract data models, their representation in a data language, and the transfer protocol commonly used with them as a single entity. While other combinations are possible, the combinations described are those that have significant deployment.

There are three basic ways that routers are managed. The most popular is the command line interface (CLI), which allows both configuration and learning of device state. This is a proprietary interface resembling a UNIX shell that allows for very customized control and observation of a device, and, specifically of interest in this case, its routing system. Some form of this interface exists on almost every device (virtual or otherwise). Processing of information returned to the CLI (called "screen scraping") is a

burdensome activity because the data is normally formatted for use by a human operator, and because the layout of the data can vary from device to device, and between different software versions. Despite its ubiquity, this interface has never been standardized and is unlikely to ever be standardized. IRS does not involve CLI standardization.

The second most popular interface for interrogation of a device's state, statistics, and configuration is The Simple Network Management Protocol (SNMP) and a set of relevant standards-based and proprietary Management Information Base (MIB) modules. SNMP has a strong history of being used by network managers to gather statistical and state information about devices, including their routing systems. However, SNMP is very rarely used to configure a device or any of its systems for reasons that vary depending upon the network operator. Some example reasons include complexity, the lack of desired configuration semantics (e.g., configuration "roll-back", "sandboxing" or configuration versioning), and the difficulty of using the semantics (or lack thereof) as defined in the MIB modules to configure device features. Therefore, SNMP is not considered as a candidate solution for the problems motivating IRS.

Finally, the IETF's Network Configuration (or NetConf) protocol has made many strides at overcoming most of the limitations around configuration that were just described. However, the lack of standard data models have hampered the adoption of NetConf.

7. Acknowledgements

The authors would like to thank Ken Gray for their suggestions and review.

8. IANA Considerations

This document includes no request to IANA.

9. Security Considerations

Security is a key aspect of any protocol that allows state installation and extracting of detailed router state. More investigation remains to fully define the security requirements, such as authorization and authentication levels.

10. Informative References

- [I-D.gredler-idr-ls-distribution]
Gredler, H., Medved, J., Previdi, S., and A. Farrel,
"North-Bound Distribution of Link-State and TE Information
using BGP", draft-gredler-idr-ls-distribution-02 (work in
progress), July 2012.
- [RFC3746] Yang, L., Dantu, R., Anderson, T., and R. Gopal,
"Forwarding and Control Element Separation (ForCES)
Framework", RFC 3746, April 2004.
- [RFC4292] Haberman, B., "IP Forwarding Table MIB", RFC 4292,
April 2006.
- [RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek,
"Architecture for IP Flow Information Export", RFC 5470,
March 2009.

Appendix A. Gaps and Concerns for SNMP

Though SNMP can allow state to be written, the overhead of the required infrastructure is quite high. Clients and servers that wish to use SNMP must build in and understand a large number of MIB modules, including many proprietary modules. Even when ignoring the overhead in building the SNMP processing and handling functions into an application, these properties lend themselves well to read-only operations. A critical lack in MIB modules for read-write (i.e., for configuration) operations is that there is no semantic understanding of the objects defined in the modules encoded in those modules. Any required semantic knowledge must be specifically hand-coded into applications or ignored. Further, many MIB modules do not allow the writing of state, and this limits coverage; owing to the cumbersome nature, there has not been interest in increasing coverage.

An additional deficiency in using SNMP MIB modules either for reading or writing comes in the inherent co-mingling of semantics and syntax in the form of indexing requirements. SNMP MIB modules contain tables that also define an index format. This format is then translated - often mapped onto - a device's actual implementation. Such a mapping means an implementation either matches the module's indexing during searches or not; if not, then an implementation is slowed down when it searches for objects. Even in not-so-extreme cases, such slow performance can result in the SNMP manager's request timing-out owing to the delay of the SNMP agent's response.

For example, if a search of N*M objects is stipulated as (N, M) in

the standard MIB module, but the implementation happens to choose to index its tables internally as (M, N), this will result in search times of $O(N^2)$. When N or M become large, as they do in routing tables, this results in wasted processing cycles for the device, and either extremely long wait times for queries, or simply a lack of answers to queries. It is a clear requirement for the IRS to not suffer from this issue.

In addition to these difficulties, SNMP matches up to the key needed aspects as follows:

Multiple Simultaneous Asynchronous Operations: Supported, but difficult for configuration.

Configuration Not Re-Processed: Supported

Duplex: The manager must initiate communications with the SNMP agent on the router. With the limited exception of Notifications and InformRequests defined in a MIB module, this is a pull model.

High-Throughput: Possible

Responsive: Possible

Multi-Channel: Possible

The key gaps identified for SNMP are:

- a. Infrastructure Overhead
- b. Lack of Semantic Information in the Data-Model
- c. Required Indexing, from mingling of semantics and syntax, badly impacting performance or driving implementation decisions.
- d. Limited interest and use for configuration
- e. Communication model isn't naturally duplex.

Appendix B. Gaps and Concerns with NetConf

While NetConf has solved many of the deficiencies present in SNMP in terms of configuration, it still does not satisfy a number of requirements needed to manage today's routing information. First, the lack of standard data models have hampered the adoption of NetConf; a significant amount of per-vendor customization is still needed. The transport mechanisms that are currently defined (e.g.,

SOAP/BEEP) for NetConf are not those commonly used by modern applications (e.g., ReST or JSON).

NetConf primarily facilitates configuration rather than reading of state or handling asynchronous events.

NetConf matches up to the key needed aspects as follows:

Multiple Simultaneous Asynchronous Operations: Not Possible

Configuration Not Re-Processed: Not Possible

Duplex: Not Possible - strict pull model.

High-Throughput: Unlikely - Can depend on configuration size

Responsive: Unlikely - Can depend on configuration size

Multi-Channel: Not Possible

Authors' Addresses

Alia Atlas (editor)
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

Email: akatlas@juniper.net

Thomas Nadeau
Juniper Networks
1194 N. Mathilda Ave.
Sunnyvale, CA 94089
USA

Email: tnadeau@juniper.net

Dave Ward
Cisco Systems
Tasman Drive
San Jose, CA 95134
USA

Email: wardd@cisco.com

Network Working Group
Internet Draft
Intended status: Informational

Dimitri Papadimitriou
Martin Vigoureux
Alcatel-Lucent

Expires: April 14, 2013

Wouter Tavernier
Didier Colle
UGent
October 15 2012

IRS Architectural Framework
draft-dimitri-irs-arch-frame-00.txt

Abstract

This document details the possible architectural and design choices in order to determine i) the spatial location of the so-called "IRS interface" and the functional entities it directly and/or indirectly involves, ii) the number of levels of redirection between routing modules and application (and associated identifier space), and iii) the various constraints that can be anticipated when dealing with the coupling of functional planes including the prevention of oscillations between two or more routing system states, coupling effects (leading to amplification chains) and, concurrency (leading to antagonistic action-reaction).

Disclaimer: this document doesn't intend to promote any specific architecture; its purpose is to understand (some of) the architectural challenges when designing interaction between routing system and applicative levels/systems.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>

This Internet-Draft will expire on April 14 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction.....	3
Conventions used in this document.....	4
2. Terminology, Notation and Acronyms.....	5
2.1. Terminology and Notation.....	5
2.2. Acronyms.....	6
3. Use Cases.....	7
4. Architectures.....	9
4.1. Dual architectures.....	9
4.1.1. Boundary on external interface.....	9
4.1.2. Boundary on internal interface.....	13
4.2. Star architecture.....	15
4.2.1. Co-located Dispatch Function.....	15
4.2.2. Non Co-located Dispatch Function.....	16
4.3. Meshed architecture.....	18
5. Comparative Analysis.....	19
6. Security Considerations.....	19
7. IANA Considerations.....	19
8. Conclusions.....	19
9. References.....	20
9.1. Normative References.....	20
9.2. Informative References.....	20
10. Acknowledgments.....	20

1. Introduction

Nowadays, applicative layers don't only rely on the shared infrastructure for information communication purposes but also for information storage (e.g., content delivery/caching, large data sets) and processing (e.g., grid computing, virtual machines). The expansion of the functional role of the shared infrastructure has in turn induced the rise of intermediate systems/application level hubs under the control / supervision of providers (usually ranging from application providers to mediators).

With the current Internet model where the routing system is a closed system, applications have to run their own measurement end-to-end to determine forwarding path characteristics through traffic monitoring but also have no means by which some network path can be enforced (by network path we refer here to the path as computed on-line by routing algorithms or those made available by off-line means). The best example is the triangular inequality violation where the shortest path between two end points (shortcut) may not be the best bandwidth x delay path between these two points. Indeed, as there is no deployed mean (e.g., source routing) by which applicative layers can tell the network which path IP datagrams shall follow, measurements don't allow any subsequent decision tuning on which network path to follow to reach a given destination. The alternative is to drive the routing table entries by applicative-triggers exchanged by means of an API (referred to as IRS since one end of this interface terminates in the "routing system") assuming applications share with the routing system a common understanding of distance and resource usage metrics. In other terms, compared to the decoupling between traffic engineering decisions concerning network path and applicative layers needs, such model would allow cooperation in the routing decision process.

In this context, the actual objectives of the present document are the following:

- i) Enumerate possible architectural and design choices to determine the spatial location of the so-called "IRS interface" and the functional entities it directly and/or indirectly involves. In turn, this enables determining internal vs. external interfaces (those that MAY be standardized and those that MUST be standardized, and those that require specific architectural focus when dealing with security aspects).

- ii) Determine the number of levels of redirection between routing modules and application (and the associated identifier space).

iii) Identify pro's and con's of possible architectures depending on use cases (bottom-up) that in general will combine one or more of these elements: distance/path, topology ((abstract) nodes, (abstract) links), location/mobility (end-point, data, etc.); note that the inclusion of traffic properties leads de facto to the introduction of monitoring points.

iv) Anticipate the constraints (from base distributed control and decision theory) when dealing with the coupling of functional planes and some architectural invariants identified (top-down). These includes the prevention of i) oscillations between two or more routing system states, ii) coupling effects (leading to amplification chains) and, iii) concurrency (leading to antagonistic action-reaction).

v) Even if the initial architectural scope would be limited to single autonomous system as starting point, propose a generic enough description to enable inter-domain use cases in the future.

It is anticipated that the time dimension will limit applicability of such information exchange to the control part of the routing system (i.e. not the forwarding component). Indeed, the closer to the forwarding component the smaller the time scale to ensure proper functionality. The full Shortest Path Tree (SPT) computation takes of the order of 10microsec per IGP destination prefix, optimizations can be obtained using incremental Shortest Path First (iSPF) algorithms. Updating the central node Routing Table (RT) ranges in the same order of magnitude per destination prefix. The consecutive time the routing engine CPU is spending to update the RT entries or their distribution to the Forwarding Table (FT) is determined by the quantum of the swapping process. The quantum time can typically be configured between an order of 10 to 100ms. If we would consider an upper level components and interactions not directly associated to the routing system, it would logically operate in the order of the order 100ms or more generally of the order of seconds and above. The routing system is thus expected to remain the entity in charge of short-term adaptations to network "topological" or "reachability" changes.

Conventions used in this document

In examples, "C:" and "S:" indicate lines sent by the client and server respectively.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

In this document, syntax specification uses the augmented Backus-Naur Form (BNF) as described in RFC-2234 [RFC2234].

2. Terminology, Notation and Acronyms

2.1. Terminology and Notation

- Agent: in general terms equipping a given function (in present context an exchange function) with memory property leads to the notion of object. Providing these objects with the capacity to adapt and to modify their behavior with respect to changing/variable conditions leads to the notion of adaptive agent.
- Application: for the sake of clarity, application refers here to computer software (organized collections of computer data and instructions) performing tasks such as data processing and/or storage, running on top of TCP/IP and accessible directly (or indirectly) to the its end-users on terminals/hosts/servers to accomplish these specific tasks.

Note: in practice, it is expected that the IRS will be initially used in the context of "network applications", i.e., programs receiving access to the routing modules and routing table in order to associate specific routing protocol decision/execution and routing table entries to specific needs as explained in Section 3.

- APP_i: agent associated to application i (where index i = 1,...,L) that enables information exchanges with the dispatch function d.
- Boundary: determines/indicates the logical intersecting edge/area between the routing system and the application system. The existence as well as to the proper operation and management of this edge/area is key in developing and maintaining coherence across the intersecting systems.
- Co-located: refers to the placement of entities in close physical/logical proximity so that remotely/distantly they appear as occupying a single position/location.
- Dispatch function (D and d): generic term describing the functional block redirecting information. Redirection of information is defined either between dispatch functions or between dispatch function and agents.

Additionally the following functionality may be associated to this functional block: syntax and semantic information processing, aggregation/abstraction of information, orchestration, and scheduling.

Equipping this function with (at least some of) these additional functionality is also part of the architectural discussion and specification.

- External interface: interface that is addressable by objects/components outside of the objects/components they interconnect/put in relation and which usually belong to different entities; each object/component defining an entry point to the entity to which they individually belong thus resulting in higher security requirements.
- Internal interface: interface that is not addressable by objects/components outside of the objects/components it interconnects/puts in relation or the single entity that comprises them. The objects/components interacting through an internal interface are co-located.
- M_m: traffic monitoring module/point m (where index m = 1,..P). Monitoring modules capture traffic and interface either with the IRS or with other routing modules via the dispatch function.
- RTR_j (Router j, where index j = 1,..,M): entity hosting multiple routing modules (RM).
- RM_k (Routing module k, where index k = 1,..,N): functional entity such as Interior Gateway Protocols (IGP), Exterior Gateway Protocol (EGP), etc. including the node global Routing Table (database structure and controller) as well as shared routing path computation functions (which for RSVP-TE is referred to as Path Computation Element or PCE). An agent indexed in the same way is associated to each routing module interacting with a dispatch function.

2.2. Acronyms

EGP	Exterior Gateway Protocol
FT	Forwarding Table
IGP	Interior Gateway Protocol
IRS	Interface to the RS
RM	Routing Module
RS	Routing System
RT	Routing Table

RTR Router

3. Use Cases

The general purpose of the IRS is to enable the augmentation or the enhancement of the IGP (and even the EGP) based routing system with i) functionality it is not able to perform when operating on a pure IGP prefix basis with or without additional state information associated to their link (e.g. spatial traffic engineering information), ii) functionality involving time varying information at a higher rate than what IGP (and even EGP) can sustain by design, and/or iii) functionality it has not been initially designed for but that strongly influence the working of the routing system (this category includes anomaly detection, intrusion detection, etc.). Cases belonging to the sets i) and ii) would increase the memory cost and adaptation cost if the IGP would be directly involved in the corresponding routing information exchange process.

From this perspective, use cases can be further classified as follows:

- Isolation of source (some of which may be included in IGP prefixes and/or prefixes included in the routing table) and/or destination addresses subset of the prefixes stored by the routing table (some of which being derived from the IGP routing information base).
- Tuning of the routing entries parameters associated to destination prefix(es) (with as particular case, host-based prefixes) being subset of IGP destination prefixes from which routing table entries are derived.
- Sequencing/ordering of the computation and activation of routing table entries (where the sequence if left to the IGP alone would for instance delay convergence).

Note that the cases outlined here below are not claimed to be genuine but expectedly illustrative of the possible usage of the "IRS interface". The proposed classes are expected to be generic enough in order to "unify" its design.

Isolation refers to cases dealing with distributed intrusion detection and distributed traffic anomaly detection (e.g., dDoS) where the objective is to detect and identify the intrusion/anomaly and possible prevent this intrusion/anomaly to further propagate. The latter typically involves determining through which router the corresponding traffic enters and possibly leaves the routing domain as well as determining the best router from where and to which the

incriminated traffic should be deflected. The term distributed means that several routers monitors collect traffic traces/records from the routing domain and the IRS is used as interface to a network application module capable of processing traffic records taken out of various monitors. Techniques enabling detection of never-seen-before patterns are now available (see e.g., [ECODE]) that limit the amount of configuration and maintenance required on each node (as long as the information captured on multiple monitors can be cross-processed). In case of isolation, the propagation of the information by means of the IGP would increase the number of entries to be stored at each router while only some of them require the corresponding information.

Tuning refers to use cases dealing with traffic engineering path computation and decision on a finer granularity than the one enforced by the destination prefixes distributed by means of the IGP. Parameter setting includes the possibility to associate a given incoming traffic flow to a specific routing entry (instead of using the "default" entry as determined by the destination address). Such parameters include in particular the bandwidth x delay product that can be computed for some of the (edge-to-edge) segments crossing the routing domain. The IRS can also be used to tune the parameter of the active monitors used to compute the available and residual capacity together with the edge-to-edge delay. On the other hand, passive monitors would not need to be activated on edge routers but placed so as to monitor specific spatial patterns of traffic. Moreover, instead of running at a default rate, the rate of packet / flow capture could be individually adapted depending on the applicative needs and the topological location of the router following that pattern.

Sequencing typically involves cases related to distributed decisions that need to be taken more rapidly and/or executions that need to be performed more rapidly than the convergence time IGP would impose (with all associated detrimental effects, e.g., routing loops, loss-of-traffic, etc.). This class includes fast re-routing (activation of a loop-free alternate routing/forwarding entry) but also configuration and maintenance operations involving for instance link metric changes, activation/deactivation of interfaces, etc.

As stated above, enabling several of these use cases would increase the memory and adaptation cost if the IGP would be directly involved in the corresponding routing information exchange process. Instead by assuming that i) not all additional entries are needed during the same period of time and ii) the number of active entries \ll total number of routing entries, one should be able to ensure higher "scaling" (or equivalently lower cost of scale) compared to the default situation IGP imposes (resulting from flooding of routing information).

4. Architectures

The below section/sub-sections detail the architectures as far as our understanding goes in providing an interaction channel between the routing system (and its individual routers) and the applicative level (and its numerous applications).

4.1. Dual architectures

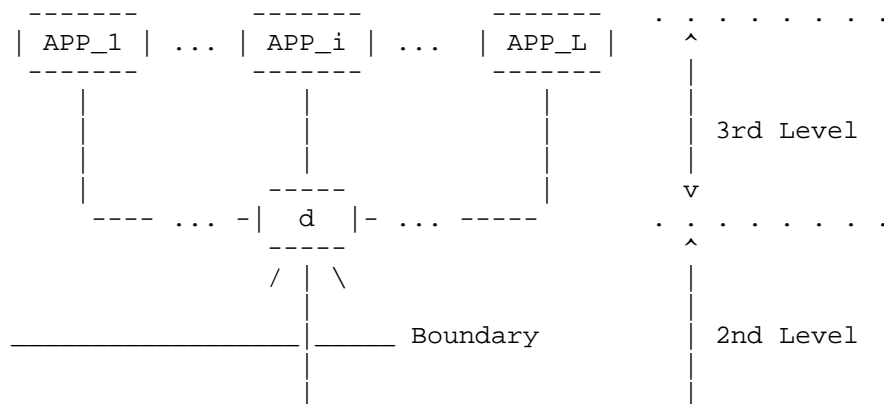
These architectures are characterized by three levels (or tiers) of redirection and a logical boundary between the routing system and the application system defined by the interface between their respective dispatch function.

4.1.1. Boundary on external interface

This architecture is similar to the one exposed in the [RTG_Area] presentation, where each routing entity or router (RTR) owns its individual dispatch function D and each APP agent is associated to a dispatch function d, distinct from the function D. This architecture involves three levels of redirection as depicted in Fig.1a. The logical boundary between the routing system and the application system is determined by the external interface between the dispatch function d and D (see below).

From this figure, the following relationships can be derived:

- Relationship APP to d: n:m (particular case: 1:1)
- Relationship d to D: m:n (particular case: 1:1)
- Relationship D to RM (same RTR): 1:n (particular case: 1:1)



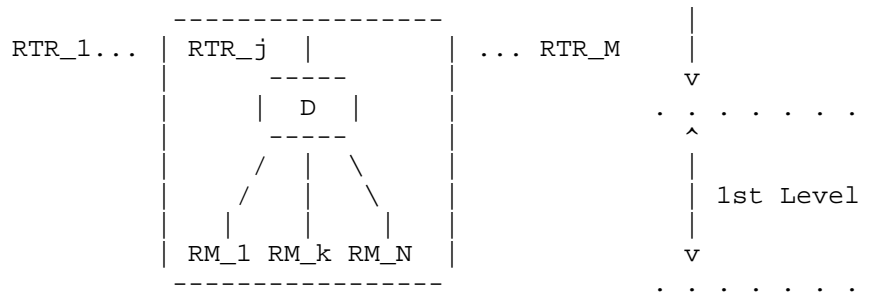


Fig.1a: Boundary on external interface d-D (co-located dispatch function D)

From Fig.1a, one can identify the following:

* Dual dispatch function:

- . One dispatch function D is associated to each RTR entity.
This dispatch function D is co-located with the RTR to which it is associated.
- . One dispatch function d is associated to a set of one or more APP agents.
This dispatch function d is not co-located with the APP agents to which it is associated.

* RTR level:

- . Includes co-located dispatch function D
- . Agents running on each routing module RM_k communicate with the dispatch function D via an internal interface (the associated routing modules may be known to the local dispatch function D by means of a registration process).

* Interfaces:

- . External interface between APP agents and d
- . External interface between dispatch functions d and D
- . Internal interface between D and RM_k agent of RTR_j
- . In terms of IRS:
 - The IRS would include the interface defined between the dispatch functions D and d (which is an external interface)
 - The question which remains is: does the IRS span the interfaces defined between i) RM_k and the dispatch function D and/or ii) APP_i and the dispatch function d. If not then the dispatch function will have to provide the syntax and semantic functionality to process messages receives from various agents.

Note that one can extend this architecture and assume that the dispatch function D is not co-located to its associated RTR as depicted in Fig.1b. This architecture also involves three levels of redirection as depicted in Fig.1b. In that case, the following relationships can be derived:

- Relationship APP to d: n:m (particular case: 1:1)
- Relationship d to D: m:n (particular case: 1:1)
- Relationship D to RTR: m:n (particular case: 1:1), meaning that a given dispatch function D can be shared among multiple RTRs.

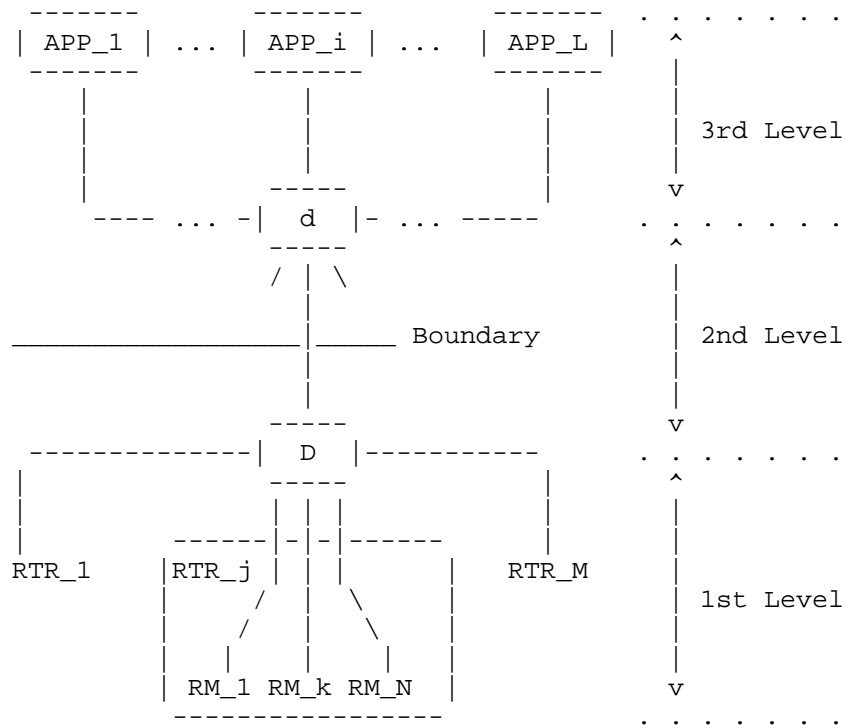


Fig.1b: Boundary on external interface d-D (non co-located dispatch function D)

From Fig.1b, one can identify the following:

- * Dual dispatch function:

- . One dispatch function D is associated to a set of one or more RTR.
This dispatch function D is not co-located with either of the RTR to which it is associated.
 - . One dispatch function d is associated to a set of one or more APP
This dispatch function d is not co-located with either of the APP to which it is associated.
 - . The dispatch functions d and D are themselves not co-located, i.e., exchanges are performed by means of an external interface.
- * RTR level:
- . No co-located dispatch function D
 - . Agents running on each routing module RM_k communicate with the dispatch function D via an external interface (the associated routing module may be known to the local function D by means of a registration process).
- * Interfaces:
- . External interface between APP agents and d
 - . External interface between dispatch functions d and D
 - . External interface between D and RM_k agents of RTR_j
- . In terms of IRS:
- The IRS would include the interface defined between the dispatch function D and d (which is an external interface)
 - Here too, the question which remains is: does the IRS span the interfaces defined between i) RM_k and the dispatch function D and/or ii) APP_i and the dispatch function d. If not then the dispatch function will have to provide the syntax and semantic functionality to process messages receives from various agents.

Remark: this architecture mandates/imposes to standardize all interfaces involved in the exchange process between the routing system and the applicative level.

4.1.2. Boundary on internal interface

This architecture involves three levels of redirection as depicted in Fig.1c. The main differences with the architecture depicted in Section 4.1.1/Fig.1a are i) non co-located dispatch function D (with its associated RTR) and ii) an internal interface between the dispatch function D and d (instead of an external interface). The main differences with the architecture depicted in Section 4.1.1/Fig.1b is the internal interface between dispatch functions D and d (instead of an external interface). The logical boundary between the routing system and the application system is determined by the internal interface between the dispatch function d and D.

The following relationships can be derived:

- Relationship APP to d: n:m (particular case: 1:1)
- Relationship d to D: m:n (particular case: 1:1)
- Relationship D to RTR: m:n (particular case: 1:1)

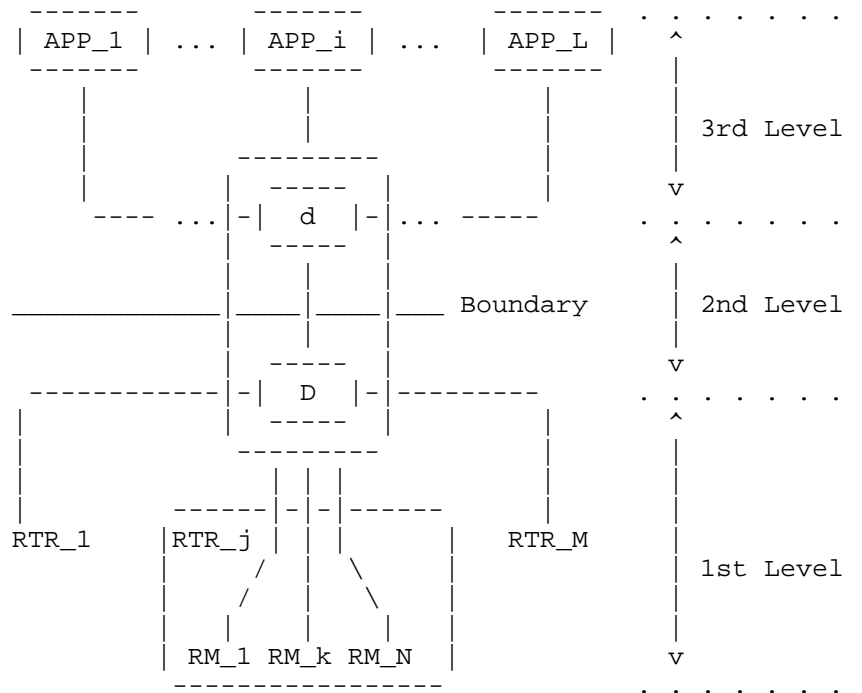


Fig.1c: Boundary on internal interface d-D (non co-located dispatch function D)

From Fig.1c, one can identify the following:

- * Dual dispatch function:
 - . One dispatch function D is associated to a set of one or more RTR.
 - . One dispatch function d is associated to a set of one or more APP.
 - . The dispatch functions d and D are themselves co-located, i.e., exchanges are performed by means of an internal interface.
- * RTR level:
 - . No co-located dispatch function D
 - . Agents running on each routing module RM_k communicate with the dispatch function D via external interface (associated routing module may be known to the local function D by means of a registration process).
- * Interfaces:
 - . External interface between APP agents and d
 - . Internal interface between dispatch functions d and D
 - . External interface between D and RM_k agents of RTR_j
- . In terms of IRS:
 - The IRS would include the interface defined between the dispatch function D and d (which is an internal interface)
 - Here too, the question which remains is: does the IRS span the interfaces defined between i) RM_k and the dispatch function D and/or ii) APP_i and the dispatch function d. If not then the dispatch function will have to provide the syntax and semantic functionality to process messages receives from various agents.

Remark: this architecture doesn't require standardizing the interface between dispatch functions (only procedures performed by the dispatch functions would need specification).

4.2. Star architecture

These architectures are characterized by i) two levels of redirection, and ii) common dispatch function (dD) between APP and RTR. The boundary in these architectures can be seen as determined by the intersecting/common function dD. When this function is supplied with memory property, it is referred to as a boundary object.

4.2.1. Co-located Dispatch Function

In this case, the following relationships can be derived:

- Relationship APP to dD: n:m (particular case: 1:1)
- Relationship dD to RM (same RTR): 1:n (particular case: 1:1)

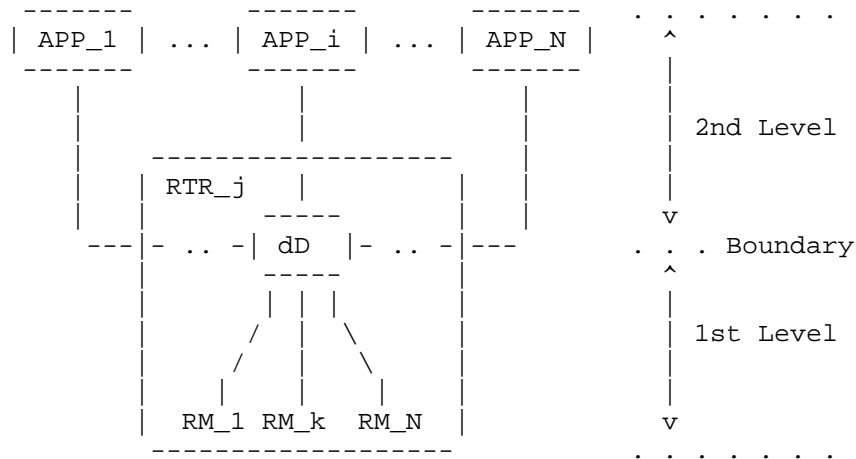


Fig.2a: Star architecture - Co-located common dispatch function

From Fig.2a, one can identify:

- * Common dispatch function:
 - . A dispatch function dD is associated to each RTR. The same dispatch function dD is associated to a set of one or more APP. One can thus refer the function dD to as a common dispatch function
 - . The dispatch function dD associated to the RTR_j is co-located with the RTR_j
- * RTR level:
 - . Includes co-located dispatch function dD

. Agents running on each routing module RM_k communicate with the dispatch function dD via internal interface (associated routing module may be known to the local function D by means of a registration process).

* Interfaces:

. External interface between APP agents and dispatch function dD
 . Internal interface between dispatch function dD and RM_k agents of RTR_j

. In terms of IRS:

- In this case, the IRS would span the interfaces defined between APP_i and the dispatch function dD. The question
- The question which remains is: does the IRS span the interfaces defined between RM_k and the dispatch function D. If not then the dispatch function will have to provide the syntax and semantic functionality to process messages receives from various agents.

4.2.2. Non Co-located Dispatch Function

In this case, the following relationships can be derived:

- Relationship APP to dD: n:m (particular case: 1:1)
- Relationship dD to RTR: m:n (particular case: 1:1)

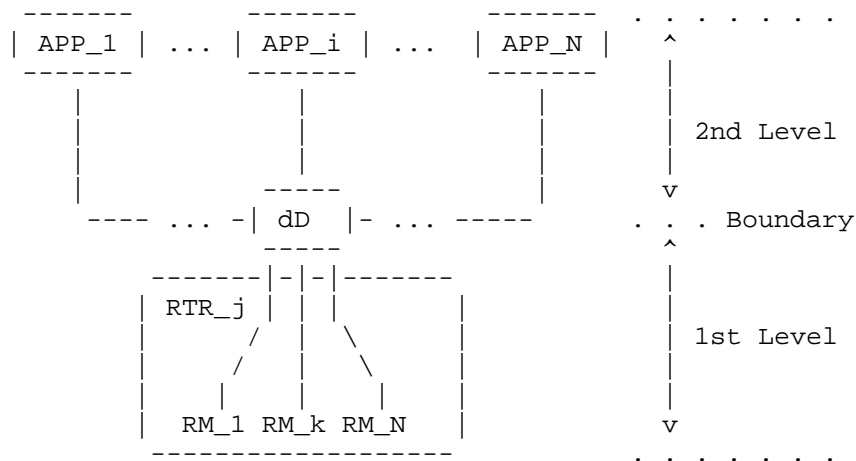


Fig.2b: Star Architecture - Non co-located common dispatch function

From Fig.2b, one can identify:

- * Common dispatch function:
 - . One common dispatch function dD is associated to a set of one or more RTR and to a set of one or more APP.
 - . This dispatch function is neither co-located with the router (RTR) (or its routing modules) nor with the APP agents. In this case, one can refer to a commonly shared dispatch function dD.
- * RTR level:
 - . No co-located dispatch function dD
 - . Agents running on each routing module RM_k communicate with the dispatch function dD via external interface (associated routing module agents may be known to the local function D by means of a registration process).
- * Interfaces:
 - . External interface between APP agents and dispatch function dD
 - . External interface between dispatch function dD and RM_k agents of RTR_j
- . In terms of IRS:
 - In this case the IRS would span i) the interfaces defined between APP_i and the dispatch function dD and ii) the interfaces defined between RM_k and the dispatch function dD.

4.3. Meshed architecture

Alternatively, one may consider that each agent executes its own dispatch function and the relationship between APP_i and RM_k agents is n:m. In this architecture, the boundary is thus determined by the set of APP agent to RM agent relationships.

This architecture however implies that i) APP and RM agents are knowledgeable of each other, and ii) the individual routers must provide means to ensure consistency and coherence of decisions (by their routing modules) but also to prevent concurrency between decisions (leading to antagonistic action-reaction). For this purpose, a decision control function (hD), performing decision verification, consistency-check, etc. is to be considered that is co-located with individual RTR (see Fig.3). The interface between each routing module RM_k and the function hD (indicated with asterisks in Fig.3) falls beyond the scope of IRS.

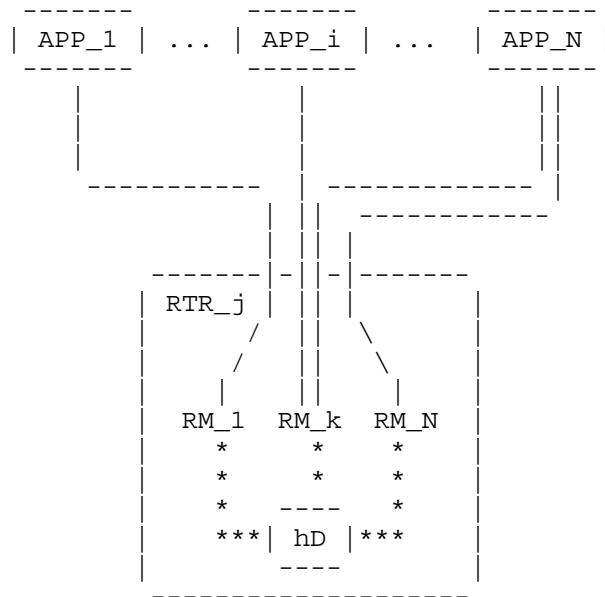


Fig.3: Meshed architecture

In a sense, the architectures presented in Section 3.1 and 3.2 perform inbound decision control whereas the architecture depicted in Fig.3 performs outbound decision control. Inbound (to RM) means that the function D (in the Fig.1 and Fig.2) processes the incoming information to ensure that the proper decision left subsequently to

individual RM but the function D is not part of the decision control process (verification, consistency-check, etc.). This process is performed by the RM themselves. On the other hand, outbound (to RM) means that the function hD (in Fig.3) performs verification, consistency-check, etc. of the (individual) decisions taken by the RMs.

5. Comparative Analysis

TBD.

6. Security Considerations

There are multiple levels at which security considerations shall be embedded and from the start as part of the architecture. Indeed, application and routing systems may (and often will) be under the control of different parties/administrative unit and communication between them (using the various external interfaces outlined in this document) may be established across the Internet.

- Accessibility and communication: prevent denial of system/service and overloads, enforce authentication of individual communicating entities and prevent unauthorized access/masquerades (authorization).

- Information exchange: enforce information integrity and validity (verification process) as well as prevent information/data spoofing; information confidentiality may be critical for certain environments, not necessarily for others (e.g., NREN).

- Semantic processing: assuming that routing modules would receive information to derive decisions, individual information can be valid but its interpretation and resulting decisions may lead to executions weakening the routing system.

7. IANA Considerations

None

8. Conclusions

TBD

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2234] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

9.2. Informative References

- [RTG_Area] Atlas, A., et al. "Interface to the Internet Routing System (IRS)", Routing Area Meeting, Vancouver (BC), Canada, July 2012.
- [ECODE] Project website: <http://www.ecode-project.eu>
- [OFELIA] Project website: <http://www.fp7-ofelia.eu>

10. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

Copyright (c) 2012 IETF Trust and the persons identified as authors of the code. All rights reserved.

Part of the input that led to this document has been derived from the outcomes of the [ECODE] project (Grant No.223936) that is funded by the Framework Program 7 (FP7) of the European Commission and from the [OFELIA] FP7 project; both projects are part of the Future Internet Research and Experimentation Initiative (FIRE) of the European Commission.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- o Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- o Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- o Neither the name of Internet Society, IETF or IETF Trust, nor the names of specific contributors, may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Authors' Addresses

Dimitri Papadimitriou
Alcatel-Lucent Bell N.V.
Dep.: Bell Labs Benelux
Copernicuslaan 50
2018 Antwerpen
Belgium
Email: dimitri.papadimitriou@alcatel-lucent.com
Tel: +32 3 2408491

Martin Vigoureux
Alcatel-Lucent Bell Labs France
Dep.: Bell Labs France
Route de Villejust
Nozay 91620
France
Email: martin.vigoureux@alcatel-lucent.com

Didier Colle
Ghent University
Dep.: INTEC
Gaston Crommenlaan 8 bus 201
9050 Ledeberg - Gent
Belgium
Email: didier.colle@intec.ugent.be
Tel: +32 9 3314900

Wouter Tavernier
Ghent University
Dep.: INTEC
Gaston Crommenlaan 8 bus 201
9050 Ledeberg - Gent
Belgium
Email: wouter.tavernier@intec.ugent.be
Tel: +32 9 3314900

Routing Area Working Group
Internet-Draft
Intended status: Informational
Expires: April 18, 2013

S. Hares
Huawei Technologies (USA)
M. Chen
Huawei
October 15, 2012

Use Cases for Virtual Connections on Demand (VCoD) and Virtual Network
on Demand using Interface to Routing System
draft-hares-use-case-vn-vc-00

Abstract

Software Defined Networks (SDN) provide a way to virtualize and abstract the network and present the virtual or abstract resources to the third-party applications running in software. The application can utilize a programmable interface to receiving these virtual or abstract resources in a form that allows monitoring or manipulation of resources. Various programmatic interfaces have been proposed to interface directly to the forwarding plane (OpenFlow, ForCES), or do device configuration (NETCONF). ALTO has proposed a informational interface to the application. Only the programmatic Interface to the Routing System (IRS) provides an interface directly to the routing system to utilize all aspects of the routing system as a system.

The IRS system interacts with the control plane processes to monitor best paths to any destination and to change the routing information base (RIB) or MPLS label information Base (LIB) which feeds the forwarding tables the information needed to actually switch traffic at a local level.

This document outlines how SDN networks can use the IRS interface to implement an automated set of network services for Virtual Connection on Demand (VCoD) and Virtual Network on Demand (VNoD)

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference

material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 18, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
2. Background	4
3. Virtual Circuit on Demand	6
4. Virtual Network on Demand (VNoD)	8
5. IANA Considerations	9
6. Security Considerations	10
7. Normative References	10
Authors' Addresses	11

1. Introduction

The Interface to the Routing System Framework [IRS] describes a mechanism where the distributed control plane can be augmented by an outside control plane through an open, accessible interface, including the Routing Information Base (RIB) and the label interface (LIB) individual devices. IRS provides a "halfway point" between completely replacing the traditional distributed control planes and directly configure devices via off-board processes.

This draft proposes a set of use cases to use IRS mechanisms to implement a Software Defined Network (SDN) with virtual connections and virtual networks as automated services. This document focuses on how IRS would support two automated network services: Virtual Connection on Demand (VCoD) and Virtual Network on Demand (VNoD). The SDN service provides the basic connection and a guidance ("self-help") functionality.

This paper contains a background section, a use case for IRS in VCoD, and a use case for IRS in VNoD.

SDN is a new adventure for the Internet space. Each new adventure in the Internet space requires lots of use cases so that the IETF may determine the critical protocols.

2. Background

Applications and network layer flows have run independently since the Internet started in the late 1980s. Provisioning of network services and big flows has been done by service providers statically or with proprietary. Recently, new server and host technologies have increase application data traffic flows across the network. With the advent of data center providers and cloud services, applications life cycles have shortened to weeks rather than years. The need for fast automated provisioning of virtual network connections or quick provisioning of virtual private networks has increased.

Software Defined Networks have have three areas of challenge to provide such quick network services: a) how to control the network flows, b) interfaces to networks, and c) how do calculate where these network flows go.

Network flows can be controlled at the forwarding device level or the network control plane level. Various programmable interfaces have been proposed to provide control over individual forwarding devices. OpenFlow, for instance, provides a mechanism to replace the dynamic control plane processes on individual forwarding devices throughout a

network with off box processes that interact with the forwarding tables on each device. Another example is NETCONF, which provides a fast and flexible mechanism to interact with device configuration and policy.

The tradeoff with the device level approach to control flows has to do with benefits and challenges of having control systems off-board. The benefit of off-board control systems is that the calculation unit can be centralized. The challenge of the off-board control system has a technical challenge and a deployment challenge. The technical challenge is that off-board control systems may encounter time-delays and communication failure. The deployment issues concerns utilizing new protocols for this communication which may also have issues in deployment. The promised benefits of off-board devices are reduction in operational costs, improving scaling, control, and visibility. OpenFlow, for instance, provides a mechanism to replace the dynamic control plane processes on individual forwarding devices throughout a network with off box processes that interact with the forwarding tables on each device. Another example is NETCONF, which provides a fast and flexible mechanism to interact with device configuration and policy.

The Interface to Routing System (IRS) interface provides an interface to all aspects of the routing system as a system. This interface allows the SDN approach to utilize the existing control plane software without changing it. The IRS system interacts with the control plane processes to monitor best paths to any destination and to interact with the routing information base (RIB) or MPLS label information base (LIB) which feeds the forwarding tables the information needed to actually switch traffic at a local level.

Let us turn to the next challenge, the interface to the applications.

Many academic efforts (e.g. Internet) have examined the benefits in allowing applications to obtain more network information when making decisions on how connect webs of interfaces. Recently, the IETF ALTO protocol has been charted to provide resource information for peer-to-peer applications. Expansions to ALTO's application interface have been proposed to pass information regarding bandwidth and network topologies. This ALTO work may apply to some large flow Virtual Connections or Virtual Private networks need. However, these ALTO use cases do not necessarily consider the on-demand issues or IRS. This document presents these use cases.

This document describes a set of use cases which describe how automated creation of Virtual Connection on Demand (VCoD) and Virtual Networks On Demand (VNoD) based in SDN logic can be accomplished by using an interface to the routing system (IRS).

There are several types of network services that can be considered as network services over which virtual connections or virtual networks can be created. These network services include: optical, Ethernet (VLAN and SPB), Internet Protocol (IP), Multiprotocol Label Switching (MPLS). Each of these networks can provide traffic engineered paths, policy control (e.g. Access control Lists (ACLs)), security services, or some form of virtual LAN services (VLAN, VxAN, L2/L3 VPN). The examples in this document focus on the transport and VPN related services that can be abstracted into Virtual Connection (VC) and Virtual Network (VN).

These abstract services (VC or VN) are logical services that can be mapped to specific services. For example, a flow can be mapped to a flow such as OpenFlow might provision through a set of networks. Or a Flow might be mapped to a TE-LSP. These logical services provide a uniform abstract service model that allows applications to configure VC or VN services independent of the actual network technology implementing it.

The use cases below leverage the SDN architecture and model and the IRS Framework to implement Virtual Circuit on Demand (VCoD) and Virtual Network on Demand (VNoD).

Please note that this draft builds on the premise that SDN solutions can augment rather than replace traditional distributed control planes. Each use case is presented in its own section.

3. Virtual Circuit on Demand

The Virtual circuit on demand (VCoD) first needs to discover where the IRS commissioners acting as controllers are. After selecting the IRS commissioner which will control the VCoD circuit, the application sends a requests to create, delete, modify or query circuits. At this point, the IRS Controller takes these requests and performs the appropriate operations. The discovery protocol and these communications are outside the IRS protocol and framework. The protocol could be ALTO that informs application which IRS commissioner can support VCoD service.

Once the IRS Commissioner is chartered with the task of setting up virtual circuits, the IRS Commission will communicate with the IRS Agents in the nodes (routing/switching/optical) to set-up these virtual circuits. In the example topology below, IRS Commissioner 1 has received a request to set up a Virtual circuit from edge 1 to edge 2. The IRS commissioner works with the IRS Agent1 on node 1, the IRS Agent 2 on node 2, the IRS Agent 3 on node 3, and the IRS Agent 4 on node 4 to set up the virtual circuit. IRS Commissioner 1

is a VCoD capable IRS commissioner with logic to aid set-up, monitoring, changing, and decommissioning of this circuit. IRS Agents 1-4 contain the necessary logic to translate the IRS Commissioner's commands to create the virtual circuit's link on their interface.

The IRS framework defines the portion of this system that goes from the VCoD-capable IRS commissioner to/from the VCoD-capable IRS Agents. The IRS Commissioner can request information from the IRS Agent such as topology or interface statics or available circuits, and influence how the IRS Agents create the circuits. The topology information passed between the IRS commission and Agent would include for this application possible virtual connections to a destination and the available bandwidth on that circuit. The interface statistics exchanged could involve historical or instant statistics on exit point performance, jitter, delay. The available of circuits could involve any time-based availability for on-demand future usage.

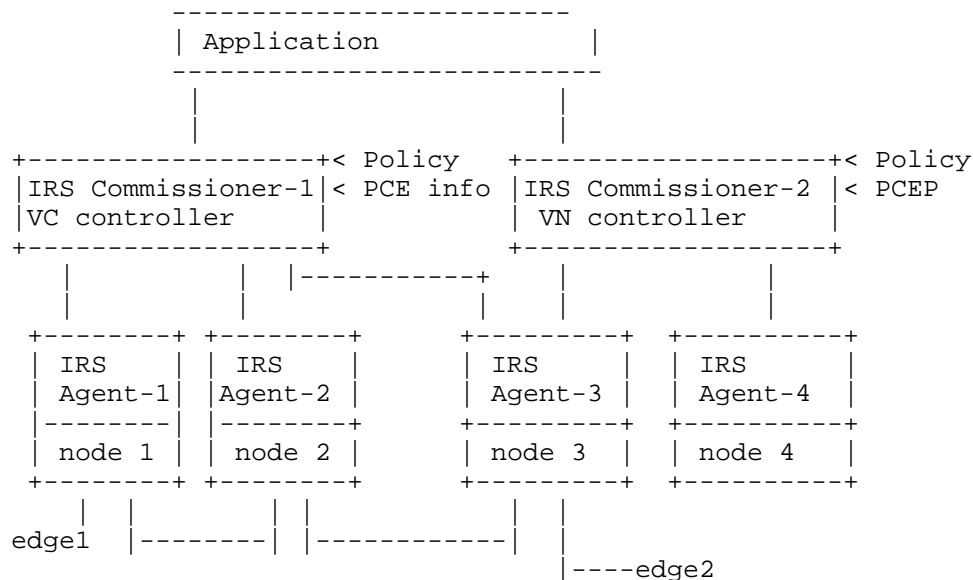
Past solutions in this area have included uses of device configuration across multiple nodes (SNMP or NETCONF based) with proprietary services combined with topology queries. The lack of a coordinated responses to routing topology queries has created problems in quickly obtaining and configuring changes for Virtual Circuits. New algorithms services in routing/switch such as Fast-Reroute of RSVP or IGPs have aided the automatic re-establishment of some circuits, but the complexity of some of these algorithms increases cost within the network elements. It's often difficult to justify the added complexity in the database and algorithms of routing protocols to solve what is considered a point case.

The following things need to be supported for this application:

- o IRS Agents should provide the ability to read the virtual connection topology database for the technology supported. For optical, these are the optical connections and what node they connect to. For MPLS, this is virtual circuit available, and what nodes they connect to. For IP technologies, this could include the GRE tunnels and what interface it connects to. For Ethernet circuits this should involve circuit type (e.g, point-to-point (p2p) or point-to-multipoint (p2mp)) and what nodes it can reach.
- o IRS Agent should provide the ability to influence the configuration of a virtual circuit in a node.
- o IRS Agent should provide monitor and provide statistics on the virtual connection to the IRS Commissioner. The IRS commissioner can then determine if the connection falls below a quality level the application has requested. If the IRS Commissioner does

determine the circuit is below the required quality, it could create another circuit. The IRS Commission may choose to create the second virtual circuit, transfer flows, and then break the first circuit.

Example Topology for Virtual Circuit on Demand (VCoD).



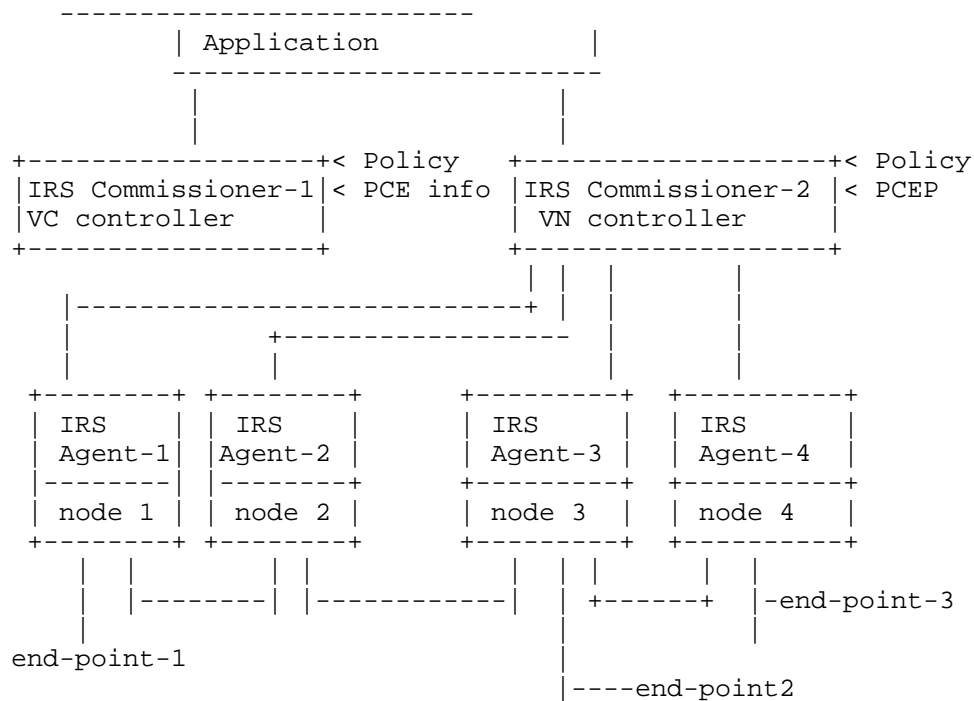
While the set-up of these virtual circuits is possible with current technology, the lack of the IRS-like framework makes VCoD network complex. With this support, VCoD may be able to reduce complexity on the individual nodes.

4. Virtual Network on Demand (VNoD)

Virtual Networks on Demand (VNoD) are simply extensions to the Virtual Connections on Demand concept. The IRS Commissioner is tasked to create a Virtual network instead of a single connection.

The example sequence would be that the application discovers IRS Commission-2 who is a VNoD via a protocol outside the IRS framework (e.g. ALTO). The IRS Commissioner-2 works with the IRS AGENTS 1-4 to set-up a virtual network. This involves the following:

- o gathering potential topology information (in order to create the network,
- o set-up the virtual network (via influencing configurations on node),
- o monitoring changes in topology (in order to potential failovers,
- o influencing changes to virtual network via configurations, and
- o removing the virtual network after the demand has expired.



This topology shares some configuration needs with the central membership computation for MPLS VPNs from (draft-white-irs-use-cases) but the mechanisms are not specific to MPLS VPNs.

5. IANA Considerations

This document includes no request to IANA.

6. Security Considerations

This document has no security issues as just contains use cases.

7. Normative References

[I-D.amante-irs-topology-use-cases]

Amante, S., Medved, J., and T. Nadeau, "Topology API Use Cases", draft-amante-irs-topology-use-cases-00 (work in progress), October 2012.

[I-D.atlas-irs-policy-framework]

Atlas, A., Hares, S., and J. Halpern, "A Policy Framework for the Interface to the Routing System", draft-atlas-irs-policy-framework-00 (work in progress), September 2012.

[I-D.atlas-irs-problem-statement]

Atlas, A., Nadeau, T., and D. Ward, "Interface to the Routing System Problem Statement", draft-atlas-irs-problem-statement-00 (work in progress), July 2012.

[I-D.bernstein-alto-large-bandwidth-cases]

Bernstein, G. and Y. Lee, "Use Cases for High Bandwidth Query and Control of Core Networks", draft-bernstein-alto-large-bandwidth-cases-02 (work in progress), July 2012.

[I-D.medved-irs-topology-requirements]

Medved, J., Gredler, H., Previdi, S., and S. Amante, "Topology API Requirements", draft-medved-irs-topology-requirements-00 (work in progress), October 2012.

[I-D.rfernando-irs-framework-requirement]

Fernando, R., Medved, J., Ward, D., Atlas, A., and B. Rijsman, "IRS Framework Requirements", draft-rfernando-irs-framework-requirement-00 (work in progress), October 2012.

[I-D.ward-irs-framework]

Atlas, A., Nadeau, T., and D. Ward, "Interface to the Routing System Framework", draft-ward-irs-framework-00 (work in progress), July 2012.

[I-D.white-irs-use-case]

White, R., Hares, S., and R. Fernando, "Use Cases for an Interface to the Routing System",
draft-white-irs-use-case-00 (work in progress),
September 2012.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

Authors' Addresses

Susan Hares
Huawei Technologies (USA)
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: Susan.Hares@huawei.com

Mach Chen
Huawei
No. 3 Xinxu Road, Shangdo-di
Hai-Dan District, Beijing 100085
USA

Email: mach@huawei.com

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: April 12, 2013

J. Medved
Cisco Systems, Inc.
H. Gredler
Juniper Networks
S. Previdi
Cisco Systems, Inc.
S. Amante
Level 3 Communications, Inc.
October 9, 2012

Topology API Requirements
draft-medved-irs-topology-requirements-00

Abstract

This document describes requirements for collecting topology data from network elements and other sources, creating a coherent view of the network topology from the collected data, and presenting the topology view to various applications that need to: a) understand the topology; and, b) interact/change the topology of the physical or logical network. for reflecting changes to the topology back in network elements..

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 12, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Requirements Language	3
2. Operational Model	3
3. Topology Manager Requirements	5
3.1. General Requirements	5
3.2. Data Model Requirements	6
3.2.1. Layer 2 Data Model Requirements	8
3.2.2. IP/MPLS Layer Data Model Requirements	9
3.3. Northbound (Client) API Requirements	10
3.4. Southbound (Network & Device) API Requirements	11
4. Network Element Requirements	12
5. Acknowledgements	13
6. IANA Considerations	13
7. Security Considerations	13
8. Normative References	13
Appendix A. Additional Stuff	13
Authors' Addresses	13

1. Introduction

[I-D.amante-irs-topology-use-cases] demonstrates the need for a network function that would provide a common, standard-based topology view to applications. This function is called 'Topology Manager' in this document. which specifies requirements for the Topology Manager. Topology Manager requirements fall into one of the following categories:

General: describes general requirements for the Topology Manager

Data Model: describes requirements for the network topology data model & view

Northbound (Client) API: describes requirements for Topology Manager's communication with clients

Southbound (Network & Device) API: describes requirements for Topology Manager's communication with Network Elements and other topology data sources

Network Elements: describes requirements for the Network Element's data model, capabilities, etc. required to support collection of network topology data

The rest of this document is organized as follows. First, the Topology Manager's operational model is described in Section 2. Topology Manager requirements are presented in Section 3. Finally, Network Element requirements are presented in Section 4..

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Operational Model

As defined in [I-D.amante-irs-topology-use-cases], the Topology Manager performs the following tasks:

- o Collection of topology data from multiple sources: Network Elements, routing protocols, inventory collection, statistics collection, etc, as discussed in [I-D.amante-irs-topology-use-cases]. Note that some of the data sources, such as statistics or inventory, will be used not only by the Topology Manager, but by other applications as well. Note

also that topology data sources may reside in multiple IGP areas or multiple ASes, and/or in multiple network layers.

- o Creation of global topology view / common data model from the collected data; the topology view can span multiple network layers as well as multiple routing and switching domains. The global view includes all network elements and resources existing in the infrastructure, whether they are currently actively used or not. An example consists of reconstructing the global view of the network including router or switch ports that are available but not in use.
- o Presentation of the network view / model to clients (applications). The Topology Manager can create multiple application-specific views from its common global topology database.

The operational model is shown in Figure 1.

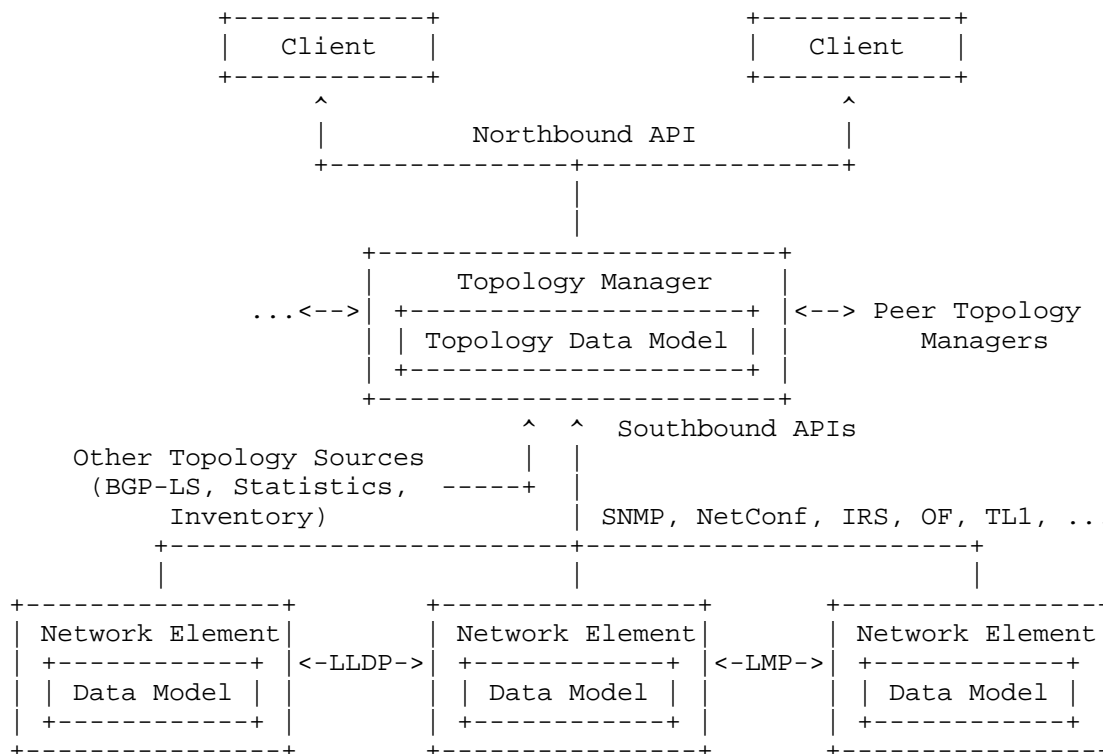


Figure 1: Topology Manager operational model

Topology information from network elements is relayed into the Topology Manager Function via its Southbound API, as shown in Figure 1. Sources of topology information may be Network Elements at different layers of the network, such as appliances, routers, L2 switches, optical transponders, optical switches. or monitoring, provisioning and network analytics tools, such as Statistics Collection or an Inventory subsystem.

The Topology Manager Function reconstructs the network/global view (that can be on a per client or per application base) and distributes it to its clients through northbound APIs. Examples of possible northbound APIs are ReST, XMPP or Websockets. The Topology Manager Function can be instantiated in a standalone server, be a part of a comprehensive orchestration / data collection / presentation framework (see [I-D.amante-irs-topology-use-cases]), or embedded in a routing element. A client can be an application or a function in an upper layer framework, such as a policy function.

Depending on the data it collects, a Topology Manager may not have visibility into the entire network. In order too create a global topology, the Topology Manager may get complementary partial topologie views from other Topology Managers via a Peer Topology Manager API.

3. Topology Manager Requirements

3.1. General Requirements

The general requirements are as follows:

- TMF.GEN-1: IRS MUST define a common vocabulary that describes attributes associated with topological components of a network. This is more commonly referred to as a "normalized" topology.
- TMF.GEN-2: IRS MUST define a data model that describes the topological components represented by the Topology Manager service. This data model will be written using a common vocabulary, that allows one to assemble the components of a network topology so that one can, for example, describe a physical link and all of its associated physical layer attributes such as: media type, bandwidth, MTU, etc.

- TMF.GEN-3: The Topology Manager has a Northbound (Client) API and multiple Southbound APIs for collecting topology data from networks. Southbound APIs can be, for example, SNMP, NETCONF, CLI, IRS, OpenFlow, or routing protocols (IGPs/BGP).
- TMF.GEN-4: The Topology Manager has an East-West (Peer Manager) API to exchange topology information with peer Topology Managers. Topologies exchanged with peer Topology Managers can be real or abstract. Note that the East-West API can be the same as the Northbound API.
- TMF.GEN-5: The Topology Manager MUST be able to collect and process topology data from hundreds of thousands of sources (network elements, etc.). Being able to collect data from large number of sources is especially important in very large optical and transport network.
- TMF.GEN-6: The Topology Manager SHOULD be able to provide multiple application-specific views to different clients.
- TMF.GEN-7: The Topology Manager MUST allow for bi-directional flow of topology data between clients and the network: clients MUST be able to get network topology information and to inject policies and/or state related to network topology.
- TMF.GEN-8: The transport protocol on all Topology Manager APIs MUST support incremental updates between Topology Managers or between a Topology Manager and a client.

3.2. Data Model Requirements

The requirements for the topology data model are as follows:

- TMF.DM-1: The topology data model MUST be able to describe topology and characteristics of different network layers:
- * Optical DWDM (for future study)
 - * Optical OTN (for future study)
 - * L2 (Aggregated links, L2 topologies)
 - * IP/MPLS,

- * VPNs

- * Services, such as cloud services, or CDNs

- TMF.DM-2: The topology data model MUST support multiple administrative domains.
- TMF.DM-3: The Topology Manager MUST provide data normalization, i.e. various types of topology information from different network elements at different network layers and in different admin domains is processed into a single, common format for clients' consumption.
- TMF.DM-4: The topology data model MUST be able to convey enough information so that a client can correlate topologies in different layers and from multiple administrative domains.
- TMF.DM-5: The topology data model MUST support multi-layer grouping of elements as a means of coalescing different nodes and links into "network layers". For example, links with IPv4 addresses might represent Layer 3 of the network topology while links with Ethernet MAC addresses might represent Layer 2. Furthermore, virtual private networks can be represented using this grouping mechanism.
- TMF.DM-6: Association between components of different layers is needed as a means of indicating relationships (i.e.: dependency, stacking, etc...) between components where layers are associated. For example, a layer 2 port might have several IPv4 or IPv6 interfaces that utilize it. These would be represented as associations to and from these components.
- TMF.DM-7: Both active and inactive topologies MUST be represented in the topology database. Inactive topology is not exhaustively seen in IGP routing protocols. It must be retrieved by querying inventory in network elements, for example new line cards inserted in a chassis, new chassis, ports in the down state, etc.
- TMF.DM-8: The topology data model MUST be hierarchical and MUST support summarization of sub-topologies. Topology summarization and creation of abstract topologies can be provided by the Topology Manager or the Orchestration Function

- TMF.DM-9: The topology data model MUST be able to describe abstract topologies. Abstract topologies can contain real and abstract nodes and real and abstract links. An abstract topology MAY be used by a provider to describe characteristics of a transit network (bandwidth, delay, protection, etc.)
- TMF.DM-10: The topology data model MUST support dynamic data, such as link and node utilizations (perhaps as optional attributes).
- TMF.DM-11: The topology data model MUST allow a user (operator) to determine the path between two nodes. The path should be traceable at all network layers that participate in the delivery of packets between two nodes. For example, for IP traffic exchanged between 2 routers, the user should be able to identify all L2 switches and optical switches that carry the traffic between the routers.
- TMF.DM-12: The topology data model MUST support multiple BGP Autonomous Systems and multiple IGP areas. Support for multiple administrative domains is for further study.
- TMF.DM-13: The topology data model MUST be human-friendly, i.e. not SNMP MIBs, but something much more analogous to YANG models.
- TMF.DM-14: The data model SHOULD support topology abstraction, allowing clients that consume topology information in a constrained manner. For example, a client wishing to view only interfaces and nodes present in a sub-graph of the Layer 3 topology should be able to specify an interest in this subset of information rather than having to read out and parse through the entire set of links and nodes.

3.2.1. Layer 2 Data Model Requirements

The requirements for the L2 topology data model are as follows:

- TMF.DM.L2-1: Inventory, (link) status and counter information MUST be retrieved from L2 Network Elements via NETCONF and SNMP.
- TMF.DM.L2-2: L2 Network Elements MUST be able to provide data obtained from L2 topology discovery protocols.

3.2.2. IP/MPLS Layer Data Model Requirements

The requirements for the IP/MPLS topology data model are as follows:

- TMF.DM.IP-1: The data model of the IP/MPLS layer MUST support both link topology and prefixes.
- TMF.DM.IP-2: Link topology may be retrieved from an IGP, BGP-LS, or by getting node neighbor information directly from network elements via a management protocol such as SNMP, NETCONF or CLI.
- TMF.DM.IP-3: Links in the IP/MPLS data model can include, among others, one or more of the following link attributes:
- * Local and Remote anchor node IDs (Router ID, AS#, Area ID, MT Topology ID)
 - * Metrics
 - * Admin Group
 - * Max link bandwidth
 - * Unreserved / utilized bandwidth
 - * Link Protection type
 - * MPLS Protocol mask
 - * IS-IS DIS
 - * OSPF DR/BDR
 - * Link prefix
 - * Link characteristics (BW, delay, error rate)
 - * Link description
 - * Link-specific timers, (Hello & Holddown)
- TMF.DM.IP-4: Nodes in the IP/MPLS data model MAY include one or more of the following node attributes:
- * Node Type: simple node / aggregate node

- * Intra area router
- * Inter area router (ISIS L1L2 or OSPF ABR)
- * AS boundary router
- * MPLS-VPN Edge router (PE)
- * Tunnel head-end/tail-end
- * Node ID:: Node ID (Router ID, AS#, Area ID, MT Topology ID)
- * Flags: Overload, Attached, External, ABR
- * Node capabilities as defined in RFC5073
- * BGP: aggregate IP prefix + mask (with associated tie-down route information to inject the aggregate route into BGP)
- * BGP policy associated with a given iBGP/eBGP neighbor; policy matching criteria can be, among others, remote-AS, local-AS, Local-AS tweaks to manipulate AS_PATH recv'd or transmitted, timers (Hello, HoldTime), AFI/SAFI, route-map/policy-statements applied/active (including associated prefix-lists, AS_PATH regexp filters, etc. and resulting manipulation of BGP path attributes, e.g.: changing LOCAL_PREF, MED, BGP community, etc.)
- * BGP statistics collection: number of IP paths/prefixes learned per neighbor

3.3. Northbound (Client) API Requirements

The requirements for the Topology Manager's northbound (client) interface are as follows:

- TMF.NB-1: The transport protocol between the Topology Manager and its clients SHOULD have efficient encoding so that large topologies can be transferred with reasonable performance.

- TMF.NB-2: The transport protocol MUST support flow-control in case a client cannot digest updates fast enough from its server.
- TMF.NB-3: The transport protocol MUST support push of topology updates from the Topology Manager to clients.
- TMF.NB-4: The protocol MUST implement a mechanism through which a client can efficiently determine the latest information especially when it receives multiple copies of the same topology from multiple Topology Managers. An example is the IGP Sequence Number that is used on IGP routing updates.
- TMF.NB-5: The Northbound API MUST support publishing of events to a dynamic list of clients/applications. This is helpful for the Northbound API to signal events as they occur in the network, e.g.: insertion or removal of linecards, etc.
- TMF.NB-6: The Northbound API SHOULD support subscription to events from a dynamic list of clients/applications. This will allow the Topology System to react dynamically when, for example, new requests for services are asked to be created.
- TMF.NB-7: The Northbound API SHOULD allow clients to subscribe to a rich assortment of attributes and/or data models so that they are automatically notified of changes within the network, (as a result of a node failure, card insertion, etc.), as well as changes made by other upper-layer applications to the network, (for example, addition/subtraction of physical links to the network during network augmentation or optimization, etc.)
- TMF.NB-8: The Northbound API MUST provide a means for non-routers to communicate with the model. Until now, clients that could gather network topology and inventory information were generally limited to those that could speak routing protocols.

3.4. Southbound (Network & Device) API Requirements

The requirements for the Topology Manager's southbound (network element) interface are as follows:

- TMF.SB-1: The transport protocol between the Topology Manager and the topology data sources (Network Elements, etc.) SHOULD have efficient encoding so that large data models can be transferred with reasonable performance.
- TMF.SB-2: The transport protocol MUST support incremental updates from a Network Element to the Topology Manager.
- TMF.SB-3: The transport protocol MUST support push of topology updates from a Network Element to the Topology Manager.

4. Network Element Requirements

The requirements for the topology data model are as follows:

- NE-1: Each network element SHOULD contain an inventory database, which SHOULD be a definitive source of information with respect to the physical HW and logical, locally significant identifiers, e.g.: VLANs, deployed in the system. The Topology Manager collects inventory data from network elements and creates an authoritative view of physical hardware deployed in the network.
- NE-2: The inventory DB SHOULD be augmented with the physical properties associated with the ports/interfaces that are directly connected to the device, (e.g.: BW, media type, etc.).
- NE-3: The inventory DB MAY be augmented through the IRS framework pushing information into the network element to, for example, associate information the installer may have knowledge of, but the network element may not be able to learn on its own, e.g.: it's physical location (address), rack/bay information, etc.
- NE-4: Relevant network elements should automatically and dynamically acquire information associated with their immediately adjacent neighbors using protocols such as LLDP, LMP, etc. A network element should further augment it's physical port inventory DB with this information so that the system can report on whom it's directly connected.

5. Acknowledgements

The authors would like to thank Alia Atlas, Chris Metz, Dave Ward, and Tom Nadeau for their comments and input.

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

tbd.

8. Normative References

[I-D.amante-irs-topology-use-cases]

Amante, S., Medved, J., and T. Nadeau, "Topology API Use Cases", draft-amante-irs-topology-use-cases-00 (work in progress), October 2012.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

Appendix A. Additional Stuff

This becomes an Appendix.

Authors' Addresses

Jan Medved
Cisco Systems, Inc.
170, West Tasman Drive
San Jose, CA 95134
USA

Email: jmedved@cisco.com

Hannes Gredler
Juniper Networks
USA

Email: hannes@juniper.net

Stefano Previdi
Cisco Systems, Inc.
170, West Tasman Drive
San Jose, CA 95134
USA

Email: sprevidi@cisco.com

Shane Amante
Level 3 Communications, Inc.
1025 Eldorado Blvd
Broomfield, CO 80021
USA

Email: shane@level3.net

INTERNET-DRAFT
Intended Status: Informational
Expires: April 11, 2013

R. Fernando
J. Medved
D. Ward
Cisco

A. Atlas
B. Rijsman
Juniper Networks
October 11, 2012

IRS Framework Requirements
draft-rfernando-irs-framework-requirement-00

Abstract

The Interface to Routing System (IRS) allows an application to programmatically query and modify the state of the network. This document defines the requirements for IRS with appropriate reasoning where required.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2012 IETF Trust and the persons identified as the

document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1	Introduction	3
1.1	Terminology	3
2.	IRS Overview	4
3.	IRS Framework Terminology	4
4.	IRS Framework Design Objectives	7
5.	IRS Framework Requirements	9
5.1	General Assumptions	9
5.2	Transport Requirements	10
5.3	Identity Requirements	11
5.4	Message Encoding Requirements	12
5.5	Message Exchange Pattern Requirements	13
5.6	API Method Requirements	15
5.7	Service and SDM Requirements	16
5.7	Security Requirements	18
5.8	Performance and Scale Requirements	19
5.9	Availability Requirements	20
5.10	Application Programmability Requirements	20
5.11	Operational Requirements	21
6	Security Considerations	21
7	Acknowledgements	22
8.	References	22
8.1	Normative References	22
	Authors' Addresses	22

1 Introduction

Routers, switches and network appliances that form today's network infrastructure maintain state at various layers of detail and function. For example, each router has a Routing Information Base (RIB), and the routing protocols (OSPF, ISIS, BGP, etc.) each maintain protocol state and information about the state of the network.

IRS [IRS-FRMWK] defines a standard interface through well defined APIs to access this information. The information collected by an application could be used to influence the routing system in conjunction with user defined policies in a feedback loop.

IRS enables this feedback loop so that applications can not only collect information but also use them to influence the network. The goal is to facilitate control and diagnosis of the routing infrastructure, as well as enable sophisticated applications to be built on top of today's network infrastructure.

Over time applications would evolve and with it their requirements too. IRS MUST be extensible so that future requirements can be easily factored in. IRS should be modular and extensible. It should be simple to understand and friendly to application developers.

This document describes some of these requirements in detail taking into consideration the use cases described in [2]. Particular attention is paid to API and the application consumption model so that it is developer friendly.

This document's scope is purely to collect and document requirements for the IRS framework. This could serve three purposes:

- a. To help the stakeholders (equipment vendors, application programmers or interested IETF participants), to arrive at a common understanding of the important elements of IRS.
- b. To provide requirements to the designers of IRS framework on the different aspects of the framework that needs consideration in the design process.
- c. To allow the stakeholders to evaluate technology choices that are suitable for IRS, to identify gaps in them and to help evolve them to suite IRS's needs.

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",

"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. IRS Overview

IRS provides a standard interface for applications to read and write state in a network device. Since the application and the network device could reside in different physical nodes, IRS could be viewed as a distributed client-server system.

IRS can also be viewed as a "framework" that helps reduce the "start up" cost in developing network based applications. A framework codifies a set of principles, patterns and software artifacts that allow application developers to quickly develop new applications.

Instead of designing each application from scratch, the IRS framework provides a set of infrastructure that abstracts the application independent mechanisms. This approach enhances software agility, reusability and portability.

This document aims at making sure that the requirements of the IRS framework are well articulated by describing its high level objectives, the concepts and components involved, how they are related and what their requirements are.

3. IRS Framework Terminology

Before we delve into the details of the IRS framework, it might help to establish some basic terminology.

Service: For the purposes of IRS, a service refers to a set of related state access functions together with the policies that control its usage. For instance, 'RIB service' could be an example of a service that gives access to state held in a device's RIB.

Server: Is a system that implements one or more services so that other client systems can call them through well defined interfaces. A server can export multiple services. A server is typically a network device.

Client: Is a system that calls a service implemented by a server through the well defined interface. A client can make use of multiple services from many servers. A client is typically a network application.

Participants: The server and client are collectively called the

participants of a service.

Transport: Is any mode of communication on an end-to-end basis between the server and client that allows them to exchange data. In principle, the transport hides the topology and other network properties from the participants of a service.

Messages: Messages are logical chunks of data that are exchanged between service participants.

Message Exchange Pattern: Is a categorization of different ways in which messages could be exchanged between service participants. MEPs specify the sequence, order, direction and cardinality of messages exchanged. Request-response and asynchronous notifications are examples of MEPs. MEPs are also sometimes referred to as the session protocol.

Message Data Model: The schema representing the structure of messages being exchanged between the service participants. The MDMs can specify certain constraints such as the data type, length, format and allowed values of fields in messages.

Message Encoding: The "wire" representation of messages exchanged between service participants.

API Method: Is an application level procedure or a function that is invoked by the client to query or modify the state held in the server.

Service Scope: Is the functional scope of a service. The service scope is established during the service definition phase.

Service Data Model: The schema representing the conceptual structure of the state held in the server for a given service. The SDMs can specify certain constraints such as the data type, length, format and allowed values for fields representing the state. They also describe the relationship between the state.

Modeling Language: Is a language that defines schema for Message Data Models and Service Data Models.

Namespaces: Allows a method for uniquely identifying and scoping of schemas declared for messages and services. Namespace is an important consideration when defining services and messages.

Service State or State: Is the general data held by the server for a given service.

State Element: A programmable state present in the server. State Element could vary in granularity.

State Identifier: A unique identity for the state element. The identifier is derived from the SDM and uses the same naming convention as the SDM. State Identifier can be viewed as the 'key' for the state.

State Value or 'value': This is a value that is assigned to a particular state identifier (key). The state is referred using the State Identifier or 'key' in operations that sets or transfers the value of the state.

State Owner: Identity of the client that was the source of a state held in the server.

State lifetime: The duration up to which the state is maintained in the server.

Datastore: This is the physical mechanism used to store a service's state.

Capabilities: Capabilities represents the functionality supported by a server including the services supported and exported to clients.

Authentication: Mechanism that allows a server to recognize the identity of a client.

Authorization: Determination of what an authenticated client is allowed to do.

Confidentiality: Specifies if data remains confidential while in transit between service participants.

Policy: For the purposes of this document, a policy is an explicit user configurable modification to the default behavior of the system. The enforcement could be conditional; they could become effective only when certain conditions are met.

As can be seen, there are many aspects to be considered in designing the IRS framework. The next section describes the broad objectives of the framework and breaks down the concerns so that each's requirements can be individually examined.

4. IRS Framework Design Objectives

The goal is to provide a framework with the infrastructural components needed to develop intelligent applications that control the network. These are some of the core guiding principles and objectives that should be kept in mind when designing that framework.

- a. Requirements Driven: The design of the framework should be pragmatic and requirements driven. Having adequate provisions to meet the needs of current applications yet making key aspects extensible to meet future needs should be the goal.
- b. Simple to Program: The success of any architectural framework depends on the how simple it is to understand and implement against. When presented with multiple choices to perform a function, choosing one of them instead of supporting all of them might lead to simpler design. In doing so, the design should consider the most important requirements and the most common deployment scenarios.
- c. Standards Based: The need for a standards-based approach to network programmability has been recognized by many standardization groups including IETF. All aspects of IRS should be open standards based. However, IRS should specify mechanisms to extend it in vendor specific manner. The aspects of IRS that could be extended should be identified in this document and should be supported by an implementation.
- d. Design for Scale and Performance: The design should meet current and future performance and scale needs. It goes without saying that scale and performance should be key criteria for making design choices. There are well understood design patterns that allow us to compose a scalable, high performing system.
- e. Extensible: IRS will be deployed in environments whose requirements evolve over time. Hence the system should be designed with provisions that will allow significant enhancements to be added to meet specified future goals and requirements. An extensible and future-proof design will drive better adoption as it is a promise against future technology churn.
- f. Promote Reuse: Reuse in this context refers to using existing tools, technologies and mechanisms instead of inventing them from scratch. It also refers to reusing a network device's current set of capabilities that applications could harness without reinventing them from scratch.

g. Promote Portability: Portability refers to the ease with which software written for one device or environment can be moved to work seamlessly with another device or environment to achieve similar functionality. A fundamental requirement for IRS is to achieve predictive and consistent behavior when applications are migrated from one platform or environment to another.

h. Security: IRS could be deployed in environments where it might be subjected to threats and denial-of-service attacks that might cause intentional damage to the functioning of a network. This could be in the form of loss of service, degradation of performance, loss of confidentiality, etc. Therefore, the security aspects should be carefully thought through when designing IRS.

i. Separation of concerns: The components of the system should be decoupled from each other as much as possible to achieve clear separation of concerns. This modularity would allow for interchangeable design and implementation choices that address the individual components requirements.

j. Robustness: Robustness is the ability of a system to operate in the face of failures and errors. It is also its ability to correctly and predictably recover from such errors and to settle to a known state. Since applications that use the IRS framework are remote and would be controlling the entire network, ensuring fault tolerance is an important consideration.

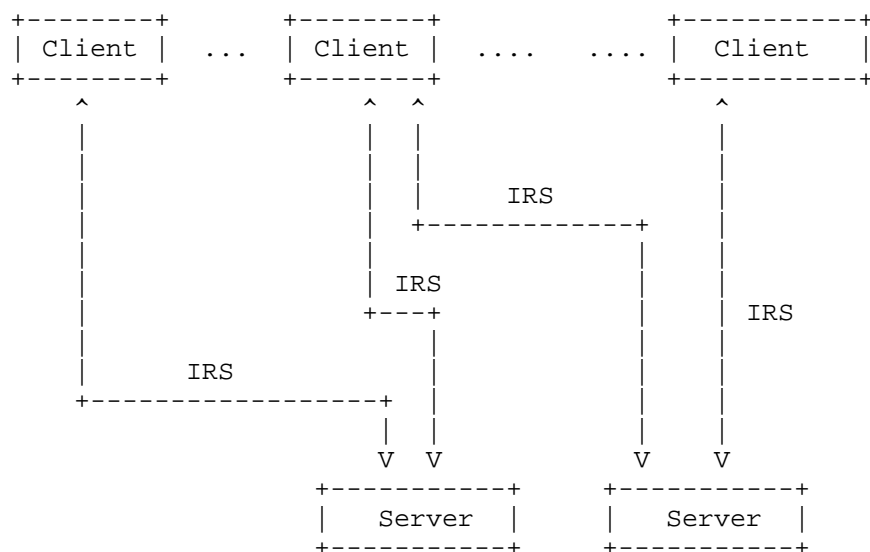
Most of these requirements cut across all the components of the system and hence should be kept in mind while designing each component and the system as a whole.

5. IRS Framework Requirements

This section is divided into multiple sub-sections, each dealing with a specific consideration of IRS framework design. As we list the requirements under each subsection, we'll annotate each requirement with what high level objectives they meet. A reason for creating the requirement is additionally provided where appropriate.

5.1 General Assumptions

This section captures the general, high level assumptions of the IRS framework. Since the design choices for the IRS framework are many, some simplifying assumptions could make the framework requirements more tangible and useful.



G.1 Programmatic access to the state held in a network device is provided to an application by exposing a set of API's from the device to the application. Due to this characteristic, IRS is a client-server protocol/framework. IRS must provide mechanisms for the client to discover services that a server provides.

G.2 The client can use the API's provided by the server to programmatically add, modify, delete and query state held in the server. Additionally clients can register for certain events and be notified when those events occur.

G.3 The client and the server communicate using a simple transport connection. The client initiates the transport connection to the server. The server does not know the number and timing of the connections from its clients.

G.4 A service provides access to the state held in the server structured according to the SDM of that service. A service allows a client the ability to manipulate the service state.

G.5 The IRS MUST define a data model to describe the SDMs supported in the server and MUST define a data modeling language to formally describe that data model. IRS MUST specify the mapping from the service data model to the message data model and subsequently to the client API.

5.2 Transport Requirements

The transport layer provides connectivity between the client and the server. This section details the transport requirements.

T.1 There should exist a default transport connection between the client and the server for communication. This control connection is point-to-point and should provide in-order and reliable delivery of data in both directions. The simplest IRS setup will only have a single transport session between the participants.

T.2 Depending on the data being exchanged, there could be additional transport connections between the client and server defined in future. The characteristics of these additional transport connections will be dictated by the requirements that create them.

T.3 The transport connection between the client and server should have mechanisms to support authentication, authorization and optionally provide confidentiality of data exchanged between the client and the server. See 'Security Requirements' for more details.

T.4 A client could connect to multiple servers. Similarly, a server could accept connections from multiple clients.

T.5 The exact technology used for the transport layer should be replaceable. There should be a single mandatory transport that should be supported by all participants. This requirement will ensure that there is always an interoperable transport mechanism between any client and any server.

T.6 Clients and servers by default communicate using a point-to-point transport connection.

T.7 Point-to-multipoint transport are mainly used to scale the system by avoiding ingress replication when the same message has to be delivered to multiple receivers. P2MP transport would work hand-in-hand with a P2MP MEP. The subject of P2MP transport and P2MP MEP is for future work.

T.8 Once the transport connection is up, it is desirable to keep it up and use it to perform multiple operations. This requirement ensures that the system scales by amortizing the session setup cost across multiple operations. Session down events do not have an impact on the state maintained by the server.

T.9 After the transport connection comes up, the participants exchange capabilities and other session parameters before exchanging service related messages.

T.10 Messages pertaining to multiple services could be exchanged over a single transport connection.

T.11 The "default" transport connection between the client and server is purely for control plane message exchanges. Data plane packets are not expected to be sent over this "default" connection. When required, data plane 'punt' and 'inject' packets between participants could be designed as a service in itself that sets up a 'punt-inject-transport' that processes the right characteristics.

T.12 For operational reasons, there MUST be a need to identify a transport connection failure. To satisfy this requirement, transport level keep-alives could be used. If the underlying transport connection does not provide a keep-alive mechanism, it should be provided at the IRS protocol level. For example, if TCP is used as a transport, TCP keep-alives could be used to detect transport session failures.

5.3 Identity Requirements

IRS could be used in a multi-domain distributed environment. Therefore a fool-proof way to ascertain the identity of clients is of utmost importance. Identity provides authenticated access to clients to state held by the server.

I.1 Each client should have a unique identity that can be verified by the server. The authentication could be direct or through an identity broker.

I.2 The server should use the client's identity to track state

provided by the client. State ownership enables the multiple clients to edit their shared state. This is useful during client death or disconnection when state owned by one client might be delegated to another client that shares the same identity.

I.3 The client's identity should be independent of the location or the network address of the physical node in which it is hosted. This allows the client to move between physical nodes. It also allows a standby client to take over when the primary fails and allows shared state editing by multiple clients as discussed in I.2.

I.4 A client that reboots or reconnects after a disconnection MUST have the same identity if it wishes to continue to operate on the state that it previously injected.

I.5 A clients ability to operate on a state held by the server is expressed at the granularity of a service. A service could be read-only or read-write by a client possessing a particular identity.

I.6 A policy on the server could dictate the services that could be exposed to clients. Upon identity verification, the authorized services are exported to the client by capability announcement.

I.7 A client can edit (write, delete) only the state that was injected by it or other clients with the same shared identity. Therefore, two conditions must be met for a client to edit a state through a session. First, the client should receive capability from the server that it has 'edit' permissions for the service in question, and, secondly, the state that it edits should be its own state.

I.8 When there is a single client and it dies, operational provisions should be made to garbage collect its state by a client that shares the original clients identity.

I.9 The server retains the client's identity till all of its state is purged from the server.

5.4 Message Encoding Requirements

Clients and servers communicate by exchanging messages between them. Message encoding is the process of converting information content in a message to a form that can be transferred between them.

ME.1 Every message between the client and the server is encoded in a

transport independent frame format.

ME.2 Each message is serialized on the senders side and de-serialized on the receivers side. The technology used for encoding and decoding messages could be negotiated between the client and the server.

ME.3 A mandatory default encoding standard should be specified and implemented by all IRS participants. This ensures that there is an interoperable default encoding mechanism between any client and any server.

ME.4 The mandatory encoding technology chosen should be well supported by a developer community and should be standards based. Availability of tools and language bindings should be one of the criteria in selecting the mandatory encoding technology.

ME.5 If multiple message encoding is supported in the framework, the encoding used for the current session should be configured using a policy on the server side and negotiated using capabilities. Note that currently there is no requirement to support multiple encoding schemes.

ME.6 The message encoding standard should be language and platform neutral. It should provide tools to express fields in messages platform independent IDL based language.

ME.7 The encoding/decoding mechanism should be fast and efficient. It should allow for operation on legacy equipment.

ME.8 The encoding scheme should allow for optional fields and backward compatibility. It should be independent of the transport and the message exchange pattern used.

ME.9 Human readability of messages exchanged on the wire might be a goal but it is secondary to efficiency needs.

5.5 Message Exchange Pattern Requirements

Message exchange patterns form the basis for all service level activities. MEPs create a pattern of message exchanges that any task can be mapped to whether initiated by a client or the server. This section provides the requirements for MEPS.

MEP.1 IRS defines three types of messages between the client and the server. First, capabilities need to be exchanged on session

establishment. Second, API commands send down from client to server to add, delete, modify and query state. And third, asynchronous notifications from server to client when interesting state changes occur.

MEP.2 The above message exchanges can be satisfied by two message exchange patterns. Capabilities and asynchronous notifications can be satisfied by one-way unsolicited fire and forget message. API commands can be satisfied using a request-response message exchange. The base IRS framework should thus support at least these two MEPs.

MEP.3 For a request-response MEP, the server should acknowledge every request message from the client with a response message.

MEP.4 The response message in a request-response MEP should indicate that the server has received the message, done some basic sanity checking on its contents and has accepted the message. The arrival of a response does not mean all post processing of the message has completed.

MEP.5 The response message should indicate an error and carry error information if there was a failure to process the request. The error code should be accompanied by a descriptive reason for the failure.

MEP.6 Error codes should indicate to the client which layer generated that error (transport, message parsing, schema validation, application level failure, etc). IRS framework should specify a standard set of error codes.

MEP.7 The request-response messages should be asynchronous. That is, the client should not stop-and-wait for one message to be acknowledged before it transmits the next request.

MEP.8 To satisfy MEP.5, there needs to be a mechanism such as a message-id, carried in the response that helps the sender correlate the response message to its original request.

MEP.9 The response messages need not arrive in the order in which the request was transmitted.

MEP.10 The request message should carry an application cookie that should be returned back to it in the corresponding response.

MEP.11 Besides the request-response MEP, there is a need for a fire and forget MEP. Asynchronous notifications from the server to the client could be carried using this MEP. Fire and forget MEPs can be used in both client-to-server and server-to-client directions.

MEP.12 The fire-and-forget MEP does not carry a message-id but it should carry a cookie that can be set by the sender and processed by the receiver. The cookie could help the receiver of the message to use the message for its intended purpose.

5.6 API Method Requirements

API methods specify the exact operation that one participant intends to perform. This section outlines the requirements for API methods.

A.1 The IRS framework should provide for a simple set of API methods, invoked from the client to the server. These methods should allow to add, modify, query and delete of state that the server maintains.

A.2 The IRS framework should provide for two methods, subscribe and unsubscribe, that the client can use to express its interest in specific state changes in the server.

A.3 The API methods discussed in A.1 and A.2 should be transported in a request-response MEP from the client to the server.

A.4 The API framework should provide for a single notify method from the server to the client when interested state changes occur. The notification method should be transported in a fire-and-forget MEP from the server to the client.

A.5 The framework should define a set of base API methods for manipulating state. These should be generic and should not service specific.

A.6 All API methods that affect the state in the server should be idempotent. That is, the final state on the server should be independent of the number of times a state change method with the same parameters was invoked by the client.

A.7 All API methods should support a batched mode for efficiency purposes. In this mode multiple state entries are transmitted in a single message with a single operation such as add, delete, etc. For methods described in A.1 and A.2 which elicit a response, the failure mechanism that is specific to a subset of state in the batch should be devised. Notify method should also support a batched mode.

A.8 Since the API methods are primarily oriented towards state transfer between the client and server, there should be a identifier (or a key) to uniquely identify the state being addressed.

A.9 API methods that refer to value of a particular state should carry the state identifier (key) as well as the its value. For instance, during a state add operation, both the identifier (key) and the value should be passed down from the client to the server.

A.10 Besides the basic API methods that are common to all services, a server could support proprietary methods or service specific methods. The framework should devise a mechanism to express these methods and their semantics through a modelling language or otherwise. The ability to support additional API methods should be conveyed to the client through the capability message.

A.11 Transactions allow a set of operations to be completed atom(all or nothing) and that the end result is consistent. This might be a requirement for some network applications and the framework designers should keep this requirement in mind during the design phase.

5.7 Service and SDM Requirements

S.1 Each service is associated with a service data model that defines the type and structure of the state pertaining to that service. IRS should provide mechanisms to manage the state held in the server in accordance to the SDM.

S.2 The data model should have the ability to express one-to-one, one-to-many and hierarchical relationships between entities.

S.3 The base IRS API methods should allow a client to add, modify, query and delete state information.

S.4 Neither the transport or the MEP should have any bearing on the structure of the state being transferred. Each service module in the server would be responsible for interpreting the structure of the state being transferred corresponding to the SDM.

S.5 A client, after proper identification, could operate on multiple 'services' that are exported to it. A client could have read-only or read-write access to a service. This is expressed by exchanging capability information with the client.

S.6 The arrangement and structure of state (SDM) should be expressed in a network friendly data modelling language.

S.7 Service data model once defined should be able to be extended. Service data models should be able to express mandatory and optional elements. It should also have the ability to express exceptions

for unsupported elements in the model. These are requirements for the modelling language.

S.8 For every service that it wishes to expose to a client, the server should send capabilities that indicate the service data model, any exceptions to it and the optional features of the data model that it supports.

S.9 A service data model could be dependent on another SDM and should have the ability to refer to state elements in another service data model.

S.10 A state element expressed in a data model could be writeable by a client or purely readable. Readable state elements are populated and managed by the server and clients don't have the ability to write their value. Routing next-hops added by a client is an example of read-write state. Statistics associated with that next-hop is an example of read-only state. The modelling language should have the ability to express this constraint.

S.11 Query and notification API should be able to carry both read-only as well as read-write state.

S.12 Besides specifying a SDM, a service should also specify the interesting state changes that clients can subscribe to for notifications.

S.13 A client which is authenticated to access a service (either read-only or read-write) can subscribe to state change events.

S.14 A subscribe method should optionally have a filter associated. This increases the efficiency by filtering out events that the client is not interested in. The notification filter should have the ability to express state identifiers and wildcards for values.

S.15 The base API operations should be generic and allow a client to operate on multiple services with the same set of methods. Each service dictates its one schema or SDM.

S.16 IRS protocol should allow a server to export standard services as well as vendor proprietary services. A namespace scheme should be devised to recognize standard and proprietary services.

S.17 The server should indicate to the client the availability of infrastructure to manage the state that it maintains. This includes but not limited to the availability of persistent store, the availability of timer to clean up state after a specified

timeout, the ability to clean up state on the occurrence of an event, etc. Equipped with this information, the client is responsible for the lifetime of the state.

S.18 Each state should have a set of meta data associated with it. This includes the state's owner, the state's lifetime attributes, a creation and modification timestamp, etc. This information would aid in the debugging of the system. An authenticated client that is exposed to a service should also have access to the meta data associated with that service's state.

5.7 Security Requirements

Security requirements should be thought through up front to avoid expensive rework to the framework. Adding security requirements once the system is designed could be an expensive and painful process. This section calls out some security concerns to be kept in mind while designing the framework.

SEC.1 Every client should be authenticated and associated with an identity. A secure mechanism to uniquely identify a client such as certificates should be adopted.

SEC.2 Every client should have an authorized role whereby only certain state can be accessed and only certain operations can be performed by that client. To keep the model simple and applications portable, authorization should be at a per service level and not on individual state element level.

SEC.3 The framework should provide for information confidentiality and information integrity as options.

SEC.4 Every state maintained by the server should be tagged with the client's identity as well as meta-data to indicate last access and last modifications time-stamps. This ensures accountability and helps auditing the system.

SEC.5 The framework designers are strongly encouraged to provide mechanisms to "hook" into third-party security infrastructure to achieve these security goals whenever possible. This keeps applications programmers free of security concerns and yet provides a flexible, configurable and well integrated security model.

5.8 Performance and Scale Requirements

Performance requirements are usually weaved in with the functional requirements of a system. They feature in every decision made to fulfill the systems requirements. Performance and scale are a complex function of many things. Hence performance requirements cannot be precisely quantified by a single number. This section lays out some common sense guidelines that should be kept in mind while designing the system from a scale and performance standpoint.

PS.1 The request-response MEP should be asynchronous. This ensures that a system is not stuck waiting for a response and makes the entire system more responsive and increases concurrency between operations.

PS.2 When applicable, messages should carry application level cookies that enable an application to quickly lookup the context necessary to process a message. The management of the cookie is the applications responsibility.

PS.3 The framework should allow for bulk operations which amortizes the communication and messaging costs.

PS.4 Provide for a binary encoding option for messages between the participants.

PS.5 Provide for a non-encrypted transport between the service participants.

PS.6 Provide for message prioritization.

PS.7 Multiple operations could be completed with one transport session.

PS.8 Keep the server as stateless with respect to the number and location of each client.

PS.9 For notifications, support filtered subscription.

PS.10 If a client requires to re-synchronize state with the server, device a mechanism to do this efficiently without transferring all the state between them.

PS.11 Allow clients that perform infrequent operations to disconnect their transport connection without cleaning up their state.

PS.12 Create the basic necessary mechanisms in the framework and build everything else as a service if possible.

5.9 Availability Requirements

The ability of the system to withstand operational failures and function in a predictable manner is called availability. A few guidelines that are important are,

A.1 Provide a 'superuser' identity that is capable of changing security policies, clearing state and perform other actions that override client initiated actions in the system.

A.2 Handle session disconnection and client deaths gracefully. These should have the least impact on the system.

A.3 Log client connections and disconnections and provide this as a well known service to authenticated users.

A.4 Notify clients of message processing and other errors through error codes in messages.

A.5 Have a mechanism to gracefully terminate the session between the client and the server.

A.6 Provide a mechanism for authenticated clients to query the load attributes of the system, both instantaneous and running average. Provide this as a service.

5.10 Application Programmability Requirements

The framework should pay particular attention the the requirements of application programmers. A well written framework should improve the productivity of programmers and shorten the time to make an application. This section has some issues to consider when devising the framework from an applications standpoint.

AP.1 A client programming framework should allow applications writers to focus on the app functionality rather than mechanisms required to communicate with the server.

AP.2 The application once written to certain requirements should be portable to other identical environments. The framework should not have fine grained data access controls as this would lead to a poorly written application with portability issues.

AP.3 The framework should be devised in a manner that it is possible to automate code generation and constraint checking in popular programming languages. Generated code can then be used readily by

application programmers instead of dealing with the nitty-gritties of the system.

AP.4 Define a common repository for SDMs from which clients can obtain the SDMs they are interested in and automatically generate most of the boilerplate code.

AP.5 Provisions should be made for debugging & troubleshooting tools that includes message trace, call traces, access to relevant server traces and logs, packet decode tools to trace & decode messages on the wire, consistency checkers of state inserted into a server.

AP.6 The toolset should have a general portion (for common functions, such as session management) and SDM-specific portions (for example, a flag to control generation of debug code in code generated for a particular SDM).

AP.7 The framework should define SDMs and MDMS in a language neutral format so as to enable code generation in multiple programming languages.

5.11 Operational Requirements

O.1 There is a need to identify operational performance parameters of the system and provide mechanisms to retrieve them from a running system.

O.2 Provide a way to upgrade a service independently of the other services. This modularity allows uninterrupted operation of the all but one service which is being upgraded.

O.3 Provide a detailed workflow for bringing about a new service. This workflow will start with the need to introduce a new service and address the following: How SDMs defined? Where are they standardized? How are new entities (MEPs, transport, encoding) introduced? What are the tools and workflow involved to develop and operationalize a service. The intent is to introduce a level of understanding about stakeholders responsibilities.

O.4 Provide mechanisms and methodologies to test a new service before deployment.

6 Security Considerations

See "Security Requirements", section 5.7 above.

7 Acknowledgements

Thanks to the following people for reviewing and providing feedback:
Alexander Clemm, John McDowell.

8. References

8.1 Normative References

[IRS-FRMWK] A. Atlas, T. Nadeau, D. Ward, "Interface to the Routing
System Framework", draft-ward-irs-framework-00

Authors' Addresses

Rex Fernando, Ed.
170 W Tasman Dr,
San Jose, CA 95134

EMail: rex@cisco.com

Jan Medved
Cisco Systems
170 W Tasman Dr,
San Jose, CA 95134

Email: jmedved@cisco.com

David Ward
Cisco Systems
170 W Tasman Dr,
San Jose, CA 95134

Email: wardd@cisco.com

Alia Atlas
Juniper Networks
10 Technology park Drive
Westford, MA 01886

Email: akatlas@juniper.net

Bruno Rijsman
Juniper Networks
10 Technology Park Drive
Westford, MA 01886

Email: brijsman@juniper.net

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 31, 2013

A. Atlas, Ed.
T. Nadeau
Juniper Networks
D. Ward
Cisco Systems
July 30, 2012

Interface to the Routing System Framework
draft-ward-irs-framework-00

Abstract

This document describes a framework for a standard, programmatic interface for full-duplex, streaming state transfer in and out of the Internet's routing system. It lists the information that might be exchanged over the interface, and describes the uses of an interface to the Internet routing system.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 31, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Functional Overview	3
1.2. Example Use-Cases	5
2. Programmatic Interfaces	6
3. Common Interface Considerations	7
3.1. Capabilities	7
3.2. Identity, Authorization, Authentication, and Security	8
3.3. Speed and Frequency of State Installation	8
3.4. Lifetime of IRS-Installed Routing System State	9
3.5. Start-Time of IRS-Installed Routing System State	10
4. Bidirectional Interfaces to the Routing System	10
4.1. Static Routing	11
4.1.1. Routing Information Base Interface	11
4.1.2. Label Forwarding Information Base Interface	12
4.1.3. Multicast Routing Information Base Interface	13
4.2. Beyond Destination-based Routing	13
4.2.1. Policy-Based Routing Interface	13
4.2.2. QoS State	14
4.3. Protocol Interactions	14
4.3.1. IGP Interfaces	14
4.3.2. BGP Interface	15
4.3.3. PIM and mLDP Interfaces	15
4.4. Triggered Sessions and Signaling	16
4.4.1. OAM-related Sessions Interface	16
4.4.2. Dynamic Session Creation	16
4.4.3. Triggered Signaling	16
5. Interfaces for Learned Information from the Routing System	16
5.1. Efforts to Obtain Topological Data	17
5.2. Measurements	18
5.3. Events	18
6. Manageability Considerations	19
7. IANA Considerations	19
8. Security Considerations	19
9. Acknowledgements	20
10. Informative References	20
Authors' Addresses	21

1. Introduction

Routers that form the Internet's routing infrastructure maintain state at various layers of detail and function. For example, each router has a Routing Information Base (RIB), and the routing protocols (OSPF, ISIS, BGP, etc.) each maintain protocol state and information about the state of the network.

A router also has information that may be required for applications to understand the network, verify that programmed state is installed in the forwarding plane, measure the behavior of various flows, and understand the existing configuration and state of the router. Furthermore, routers are configured or implemented with procedural or policy-based instructions for how to convert all of this information into the forwarding operations that are installed in the forwarding plane, and this is also state information that describes the behaviour of the router.

This document sets out a framework for a common, standard interface to allow access to all of this information. This Interface to the Routing System (IRS) would facilitate control and diagnosis of the routing infrastructure, as well as enabling sophisticated applications to be built on top of today's routed networks. The IRS is a programmatic, streaming interface for transferring state into and out of the Internet's routing system, and recognizes that the routing system and a router's OS provide useful mechanisms that applications could harness to accomplish application-level goals.

Fundamental to the IRS is a clear data model that defines the semantics of the information that can be written and read. The IRS provides a framework for registering for and requesting the appropriate information for each particular application. The IRS provides a way for applications to customize network behaviour while leveraging the existing routing system.

The IRS, and therefore this document, is specifically focused on an interface for routing and forwarding data.

1.1. Functional Overview

There are three key aspects to the IRS. First, the interface is a programmatic streaming interface meaning that it is asynchronous and offers fast, interactive access. Second, the IRS gives access to information and state that is not usually configurable or modeled in existing implementations or configuration protocols. Third, the IRS gives applications the ability to learn additional, structured, filterable information and events from the router.

IRS is described as a streaming programmatic interface; the key properties that are intended are:

Multiple Simultaneous Asynchronous Operations: A single application should be able to send multiple operations to IRS without needing to wait for each to complete before sending the next.

Configuration Not Re-Processed: When an IRS operation is processed, it does not require that any of the configuration be processed. I.e. the desired behavior with regard to static configuration is the same as learning a new BGP route - completely orthogonal.

Duplex: Communications can be established by either the router or the application. Similarly, events, acknowledgements, failures, operations, etc. can be sent at any time by both the router and the application. This is not a pure pull-model where only the application queries to pull responses.

High-Throughput: At a minimum, the IRS should be able to handle hundreds of operations per second.

Responsive: It should be possible to complete simple operations within a sub-second time-scale.

Multi-Channel: It should be possible for information to be communicated via the interface from different components in the router without requiring going through a single channel. For example, for scaling, some exported data or events may be better sent directly from the forwarding plane, while other interactions may come from the control-plane. Thus a single TCP session per application would not be a good match.

Such an interface facilitates the specification of non-permanent state into the routing system as well as the extraction of that information and additional dynamic information from the routing system. A non-routing protocol or application could inject state into a networking node's OS via the state-insertion aspects of the interface, that could then be distributed in a routing or signaling protocol.

Where existing mechanisms can provide part of the desired functionality, the coverage and gaps are briefly discussed in this document.

The existing mechanisms, such as SNMP and NetConf, that allow state to be written and read do not meet all of the above key properties needed for IRS. The overhead of infrastructure is also quite high and many MIBs do not, in definition or practice, allow writing of

state. There is also very limited capability to add new application-specific state to be distributed via the routing system. Conversely, NetConf is challenging for reading state from a router.

ForCES is another method for writing state into a router, but its focus is on the forwarding plane. By focusing on the forwarding plane, it requires that the forwarding plane be modeled and programmable and ignores the existence and intelligence of the router OS and routing system. ForCES provides a lower-level interface than IRS is intended to address.

1.2. Example Use-Cases

A few brief examples of ways an application could use the IRS are presented here. These are intended to give a sense of what could be done rather than to be primary and detailed motivational use-cases.

Route Control via Indirection: By enabling an application to install routes in the RIB, it is possible that when, for example, BGP resolves its IGP next-hop via the RIB, that could be to an application-installed route. In general, when a route is redistributed from one protocol to another, this is done via the RIB and such a route could have been installed via the IRS interface.

Policy-Based Routing of Unknown Traffic: A static route, installed into the RIB, could direct otherwise unrecognized traffic towards an application, through whatever appropriate tunnel was required, for further handling. Such a static route could be programmed with indirection, so that its outgoing path is whatever is used by another particular route (e.g. to a particular server).

Services with Fixed Hours: If an application were to provide services only during fixed time-periods, the application could install both a specific route on the local router in the RIB and advertise the associated prefix as being attached to the local router via the IGP. If the application knew the fixed hours, the state so installed could be time-based and automatically removed at approximately the correct time.

Traffic Mirroring: The interface to the multicast RIB could be used to mirror a particular traffic flow to both its original destination and a data collector.

Static Multicast Trees: An application could set up static (or partially static) multicast flows via entries in the multicast RIB without requiring an associated multicast protocol. This could be useful in networks with a fixed topology and well-planned

distribution tree that provides redundancy.

2. Programmatic Interfaces

A number of management interfaces exist today that allow for the indirect programming of the routing system. These include proprietary CLI, Netconf, and SNMP. However, none of these mechanisms allows for the direct programming of the routing system. Such streaming interfaces are needed to support dynamic time-based applications.

These interfaces should cater to how applications typically interact with other applications and network services rather than forcing them to use older mechanisms that are more complex to understand and implement, as well as operate.

The most critical component of the IRS is developing standard data models with their associated semantics. While many routing protocols are standardized, associated data models for IRS are not yet available. Instead, each router uses different information, mechanisms, and CLI which makes a standard interface for use by applications extremely cumbersome to develop and maintain. Well-known data modeling languages, such as YANG [RFC6020], exist and might be used for defining the necessary data models; more investigation into alternatives is required. It is understood that some portion (hopefully a small subset) will remain as proprietary extensions; the data models must support future extensions and proprietary extensions.

Since the IRS will need to support remote access between applications running on a host or server and routers in the network, at least one standard mechanism must be identified and defined to provide the transfer syntax, as defined by a protocol, used to communicate between the application and the routing system. Common functionality that IRS needs to support includes acknowledgements, dependencies, request-reserve-commit.

Appropriate candidate protocols must be identified that reduce the effort required by applications and, preferably, are familiar to application developers. Ideally, this should not require that applications understand and implement existing routing protocols to interact with IRS. These interfaces should instead be based on light-weight, rapidly deployable approaches; technology approaches must be evaluated but examples could include ReSTful web services, JSON, XMPP, and XML. These interfaces should possess self-describing attributes (e.g. a web services interface) so that applications can quickly query and learn about the active capabilities of a device.

It may be desirable to also define the local syntax (e.g. programming language APIs) that applications running local to a router can use.

Since evolution is anticipated in IRS over time, it is important that versioning and backwards compatibility are basic supported functionality. Similarly, common consistent error-handling and acknowledgement mechanisms are required that do not severely limit the scalability and responsiveness of these interfaces.

3. Common Interface Considerations

3.1. Capabilities

Capability negotiation is a critical requirement because different implementations and software versions will have different abilities. Similarly, applications may have different capabilities for receiving exported information.

The IRS will have multiple interfaces, each with their own set of capabilities. Such capabilities may include the particular data model and what operations can be performed at what scale.

The capabilities negotiated may be filtered based upon different information, such as the application's authorization, application's capabilities, and the desired granularity for abstraction which the application understands. Different types of authorization may require the router to advertise different capabilities and restrictions.

The capability negotiation may take place at different levels of detail based upon the application and the specific functions in the IRS that the application is negotiating. The router and application must use the IRS to agree upon the proper level of abstraction for the interaction. For example, when an application describes a route between two topological items, these items may vary in detail from a network domain's name at a high level, or down to the port forwarding specifics of a particular device.

The data-model and capabilities available for an element may depend upon whether the element is physical or virtual; the virtual/physical distinction does not matter to IRS. Similarly, the location of the element may influence how an application converses with the associated router.

3.2. Identity, Authorization, Authentication, and Security

Applications that wish to manipulate or interrogate the state of the routing system must be appropriately authorized. This means that at least one means of determining the unique identity of an application and its associated access privileges must be available; this implies that the identity and associated access privileges must be verifiable from the router being programmed.

Furthermore, being able to associate a state and the modifications to a state with a specific application would aid in troubleshooting and auditing of the routing system. By associating identity and authorization with installed state, other applications with appropriate authority can clean up state abandoned by failed applications, if necessary.

Security of communication between the application and the router is also critical and must be considered in the design of the mechanisms to support these programmatic interfaces.

3.3. Speed and Frequency of State Installation

A programmatic interface does not by itself imply the frequency of state updates nor the speed at which the state installation is required. These are critical aspects of an interface and govern what an application can use the interface for. The difference between sub-second responsiveness to millions of updates and a day delay per update is, obviously, drastic. The key attributes of the programmatic interface are described in Section 1 and include that the interface must be asynchronous.

For each interface in IRS, it will be necessary to specify expected scaling, responsiveness, and performance so that applications can understand the uses to which the IRS can be used.

IRS must support asynchronous streaming real-time interactions between the applications and router. IRS must assume that there are many unrelated applications that may be simultaneously using IRS. This implies that applications must be able to subscribe to change events that notify them about changes done to state by other applications or configuration.

Furthermore, IRS should construct interfaces that cater to different scaling and frequency of update parameters. For example, slow, but detailed queries of the system, or fast yet higher level (less detailed) queries or modifications.

3.4. Lifetime of IRS-Installed Routing System State

In routers today, the lifetime of different routing state depends upon how that state was learned and committed. If the state is configuration state, then it is ephemeral when just in the running configuration or persistent when written to the startup configuration. If the state is learned via a routing protocol or SNMP, it is ephemeral, lasting only until the router reboots or the state is withdrawn.

Unlike previous injection mechanisms that implied the state lifetime, IRS requires that multiple models be supported for the lifetime of state it installs. This is because the lifetime or persistence of state of the routing system can vary based on the application that programmed it, policies or security authorization of the application.

There are four basic models to be supported.

Ephemeral: State installed by the application remains on the router in its active memory until such time as it is either removed by a routing or signaling protocol, removed by a configuration initiated by an application, or the router reboots. In the case of the latter, past state is forgotten when the router reboots.

Persistent: State installed by the application remains on the router across reboots or restarts of the system. It can be dynamically removed or manipulated by an application, by configuration, or by the routing system itself. This state does not appear in the router's configuration; it is processed after all the configuration upon a reboot.

Time-Based: When state is installed by the application, it has an expiration time specified. When that time has passed, the state is removed from the router. It can also be dynamically removed or manipulated by an application, by configuration or the routing system itself. State that hasn't expired will remain on a router through reboots.

Time-Based Ephemeral: When state is installed by the application, it has an expiration time specified. When that time has passed, the state is removed from the router. It can also be dynamically removed or manipulated by an application, by configuration, by the routing system itself, or by the router rebooting. Past state is forgotten after the router reboots.

3.5. Start-Time of IRS-Installed Routing System State

To provide flexibility, pre-programming, and handle dependencies, it is necessary to have multiple models of when a operation is to be handled. There are the following basic models to be supported.

Immediate: When the operation is received, it should be acted upon as quickly as reasonable (e.g. queued with other outstanding requests if necessary).

Time-Based: An application may provide an operation that is to be initiated at a particular time. When the specified time is reached, the operation should be acted upon as quickly as reasonable. Implementations may, of course, strive to improve the time-accuracy at which the operation is initiated.

Triggered: The operation should be initiated when the specified triggering event has happened. A triggering event could be the successful or failed completion of another operation. A triggering event could be a system event, such as an interface up or down, or another event such as a particular route changing its next-hops.

Because it is possible to request operations in models other than "Immediate" and some of the start-times will be at an unknown future point (e.g. "Triggered"), it is not feasible to guarantee that the resources required by an operation will always be available without reserving them from the time the operation is received. While that type of resource reservation should be possible, applications must also be able to handle an operation failing or being preempted due to resources or due to a higher priority or better authorized application taking ownership of the associated state or resource.

4. Bidirectional Interfaces to the Routing System

IRS is a bidirectional programmatic interface that allows both routing and non-routing applications to install, remove, read, and otherwise manipulate the state of the routing system.

Just as the Internet routing system is not a single protocol or implementation layer, neither does it make sense for the IRS to be at a single layer or reside within a single protocol. For each protocol or layer, there are different data models, abstractions and interface syntaxes and semantics required. However, with this in mind, it is ideal that a minimal set of mechanism(s) to define, transfer and manipulate this state will be specified with as few optional characteristics as possible. This will foster better

interoperability between different vendor implementations.

Since IRS is focused on the routing system, the layers of interest start with the RIB and continue up through the IGPs, BGP, RSVP-TE, LDP, etc. The intent is neither to provide interfaces to the forwarding plane nor to provide interfaces to application layers.

It is critical that these interfaces provide the ability to learn state, filtered by request, as well as install state. IRS assumes that there will be multiple applications using IRS and therefore the ability to read state is necessary to fully know the router's state. In general, if an interface allows the setting of state, the ability to read and modify that state is also necessary.

4.1. Static Routing

The ability to specify static routes exists via CLI and MIBs but these mechanisms do not provide a streaming programmatic interface. IRS solves this problem by proposing interfaces to the RIB, LFIB, and Multicast RIBs.

By installing static routes into the RIB layer, IRS is able to utilize the existing router OS and its mechanisms for distributing the selected routes into the FIB and LIB. This avoids the need to model or standardize the forwarding plane.

4.1.1. Routing Information Base Interface

The RIB is populated with routes and next-hops as supplied by configuration, management, or routing protocols. A route has a preference based upon the specific source from which the route was derived. Static routes, specified via CLI, can be installed with an appropriate preference. The FIB is populated by selecting from the RIB based on policy and tie-breaking criteria.

The IRS interface should allow dynamic reading and writing of routes into the RIB. There are several important attributes associated with doing so, as follows:

Preference Value: This allows decisions between conflicting routes, whether IRS-installed or otherwise. IRS-installed routes can each be installed with a different preference value.

Route Table Context: There can be different route table contexts in the RIB. Examples include multiple protocols (e.g. IPv4, IPv6), multiple topologies, different uses, and multiple networks (e.g. VRF tables for VPNs). Appropriate application-level abstractions are required to describe the desired route table context.

Route or Traffic Identification The specific IP prefix or even interface must be specified.

Outgoing Path and Encapsulation: It is necessary to specify the outgoing path and associated encapsulation. This may be done directly or indirectly. This is one of the more complex aspects with the following considerations.

Primary Next-Hops: To support multi-path forwarding, multiple primary next-hops can be specified and the traffic flows split among them.

Indirection: Instead of specifying particular primary next-hops, it is critical to be able to provide the ability for indirection, such as is used between BGP routes and IGP routes. Thus, the outgoing path might be specified via indirection to be the same as another route's.

Encapsulation: Associated with each primary next-hop can be details on the type of encapsulation for the packet. Such encapsulation could be MPLS, GRE, etc. as supported by the router.

Protection: For fast-reroute protection, each primary next-hop may have one or more alternate next-hops specified. Those are to be used when the primary next-hop fails.

DSCP: For QoS, the desired DSCP to be used for the outgoing traffic can be specified.

It is useful for an application to be able to read out the RIB state associated with particular traffic and be able to learn both the preferred route and its source as well as other candidates with lower preference.

Although there is no standardized model or specification of a RIB, it may be possible to build an interoperable bi-directional interface without one.

4.1.2. Label Forwarding Information Base Interface

The LFIB has a similar role to the RIB for MPLS labeled packets. Each entry has slightly different information to accommodate MPLS forwarding and semantics. Although static MPLS can be used to configure specific state into the LFIB, there is no bidirectional programmatic interface to program, modify, or read the associated state.

Each entry in the LFIB requires a MPLS label context (e.g. platform, per-interface, or other context), incoming label, label operation, and next-hops with associated encapsulation, label operation, and so on. Via the IRS LFIB interface, an application could supply the information for an entry using either a pre-allocated MPLS label or a newly allocated MPLS label that is returned to the application.

4.1.3. Multicast Routing Information Base Interface

There is no bidirectional programmatic interface to add, modify, remove or read state from the multicast RIB. This IRS interface would add those capabilities.

Multicast forwarding state can be set up by a variety of protocols. As with the unicast RIB, an application may wish to install a new route for multicast. The state to add might be the full multicast route information - including the incoming interface, the particular multicast traffic (e.g. (source, group) or MPLS label), and the outgoing interfaces and associated encapsulations to replicate the traffic too.

The multicast state added need not match to well-known protocol installed state. For instance, traffic received on an specified set, or all, interfaces that is destined to a particular prefix from all sources or a particular prefix could be subject to the specified replication.

4.2. Beyond Destination-based Routing

Routing decisions and traffic treatment is not merely expressible via destination-based routing or even (S, G) routing, such as in multicast. Capturing these aspects into appropriate interfaces for the IRS provides the ability for applications to control them as well.

4.2.1. Policy-Based Routing Interface

A common feature of routers is the ability to specify policy-based routing (PBR) rules for accepting, dropping, or differently forwarding particular traffic. This is a very useful functionality for an application to be able to rapidly add and remove state into. Such state would indicate the particular traffic to be affected and its subsequent behavior (e.g. drop, accept, forward on specified outgoing path and encapsulation, QoS, DSCP marking, policing, etc.). Such state is made more complex by the potential importance of ordering among the PBR rules.

While PBR rules can be specified via CLI, this mechanism is not a

streaming programmatic interface nor is there generally the ability to specify particular time-based lifetimes for each rule.

4.2.2. QoS State

While per-hop behaviors are defined as well as standard DSCP meanings, the details of QoS configuration are not standardized and can be highly variable depending upon platform. It is NOT a goal of this work to standardize QoS configurations. Instead, a data object model can define push/pull configurations. More investigation is needed to better describe the details.

4.3. Protocol Interactions

Providing IRS interfaces to the various routing protocols allows applications to specify policy, local topology changes, and availability to influence the routing protocols in a way that the detailed addition or modification of routes in the RIB does not.

The decision to distribute the routing state via a routing or signaling protocol depends upon the protocol-layer at which this state is injected into the routing system. It may also depend upon which routing domain or domains this information is injected as well.

In addition it is necessary to have the ability to pull state regarding various protocols from the router, a mechanism to register for asynchronous events, and the means to obtain those asynchronous events. An example of such state might be peer up/down.

4.3.1. IGP Interfaces

The lack of a streaming programmatic interface to the IGPs limits the ability of applications to influence and modify the desired behavior of the IGP.

An application may need to indicate that a router is overloaded (via ISIS or the method described in [RFC3137]) because that router does not yet have sufficient state synchronized or installed into it. When critical state is provided not merely by routers but also from applications via the IRS, a synchronization mechanism can be needed.

The ability for an application to modify the local topology can be part of this interface. One possibility is to allow modification of local interface metrics to generally influence selected routes. A more extensive interface might include the ability to create a OSPF or ISIS adjacency across a specified interface (virtual or real) with the appropriate associated encapsulation.

The ability to attach a prefix to the local router would provide a straightforward method for an application to program a single router and have the proper routes computed and installed by all other routers in the relevant domains. Additional aspects to the prefix attachment, such as the metric with which to attach the prefix and fast-reroute characteristics, would be part of the interface.

Beyond such pure routing information, the need for an application to be able to install state to be flooded via an IGP has already been recognized. [I-D.ietf-isis-genapp] specifies a mechanism for flooding generalized application information via ISIS, but does not describe how an application can generate or consume this information. Similarly, [RFC5250] specifies Opaque LSAs for OSPF to provide for application-specific information to be flooded. An IRS interface and associated data object model would provide such a mechanism.

Additional investigation will identify other state that applications may wish to install.

From the IGP, applications via IRS can extract significant topological information about the routers, links, and associated attributes.

4.3.2. BGP Interface

BGP carries significant policy and per-application specific information as well as internet routes. A significant interface into BGP is expected, with different data object models for different applications. For example, the IRS interface to BGP could provide the ability to specify the policy on which paths BGP chooses to advertise. Additionally, the ability to specify information with an application-specified AFI/SAFI could provide substantial flexibility and control.

An existing example of application information carried in BGP is BGP Flowspec [RFC5575] which can be used to provide traffic filtering and aid in handling denial-of-service attacks.

The ability to extract information from BGP is also quite critical. A useful example of this is the information available from BGP via [I-D.gredler-idr-ls-distribution], which allows link-state topology information to be carried in BGP.

4.3.3. PIM and mLDP Interfaces

For PIM and mLDP, there are at least two types of state that an application might wish to install. First, an application might add an interface to join a particular multicast group. Second, an

application might provide an upstream route for traffic to be received from - rather than having PIM or mLDP need to consult the unicast RIB.

Additional investigation will identify other state that applications may wish to install.

4.4. Triggered Sessions and Signaling

4.4.1. OAM-related Sessions Interface

An application may need to trigger new OAM sessions (e.g. BFD, VCCP, etc.) using an appropriate template. For example, there may be applications that need to create a new tunnel, verify its functionality via new triggered OAM sessions, and then bring it into service if that OAM indicates successful functionality. More investigation is needed to better describe the details.

4.4.2. Dynamic Session Creation

An application may wish to trigger a peering relationship for a protocol. For instance, a targeted LDP session may be required to exchange state installed locally with a remote router. More investigation is needed to better describe the different cases and details.

4.4.3. Triggered Signaling

To easily create dynamic state throughout the network, an application may need to trigger signaling via protocols such as RSVP-TE. An example of such an application can be a Stateful Path Computation Element (PCE)[I-D.ietf-pce-stateful-pce], which has control of various LSPs that need to be signaled.

More investigation is needed to better describe the different cases and details.

5. Interfaces for Learned Information from the Routing System

Just as applications need to inject state into the routing system to meet various application-specific and policy-based requirements, it is critical that applications be able to also extract necessary state from the routing system.

A part of each of these interfaces is the ability to specify the generation of the desired information (e.g., collecting specific per-flow measurements) and the ability to specify appropriate filters to

indicate the specifics and abstraction level of the information to be provided

The types of information to extract can be generally grouped into the following different categories.

Topological: The need to understand the network topology, at a suitable abstraction layer, is critical to applications. Connectivity is not sufficient - the associated costs, bandwidths, latencies, etc. are all important aspects of the network topology that strongly influence the decision-making and behavior of applications.

Measurements: Applications require measurements of traffic and network behavior in order to have a more meaningful feedback control loop. Such information may be per-interface, per-flow, per-firewall rule, per-queue, etc.

Events: There are a variety of asynchronous events that an application may require or use as triggering conditions for starting other operations. An obvious example is interface state events.

Configuration: For some aspects, it may be necessary for applications to be able to learn about the routing configuration on a box. This is partially available via various MIBs and NetConf. What additional information needs to be exported and the appropriate mechanisms needs further examination.

The need to extract information from the network is not new; there is on-going work in the IETF in this area. This framework describes those efforts in the context of the above categories and starts the discussion of the aspects still required.

5.1. Efforts to Obtain Topological Data

Topological data can be defined and presented at different layers (e.g. Layer-2, Layer-3) and with different characteristics exposed or hidden (e.g. physical or virtual, SRLGs, bandwidth, latency, etc.). It can also have different states, such as configured but unavailable, configurable, active, broken, administratively disabled, etc.

To solve the problem of only being able to obtain topological data via listening to the IGP in each area, BGP-LS [I-D.gredler-idr-ls-distribution] defines extensions to BGP so that link-state topology information can be carried in BGP and a single BGP listener in the AS can therefore learn and distribute the entire

AS's current link-state topology. BGP-LS solves the problem of distributing topological information throughout the network. While IRS may expand the information to be distributed, IRS addresses the API aspect of BGP-LS and not the network-wide distribution.

At another level, ALTO [RFC5693] provides topological information at a higher abstraction layer, which can be based upon network policy, and with application-relevant services located in it. The mechanism for ALTO obtaining the topology can vary and policy can apply to what is provided or abstracted.

Neither of these fully meet the need to obtain detailed, layered topological state that provides more information than the current functional status. While there are currently no sufficiently complete standards, the need for such functionality can be deduced by the number of proprietary systems that have been developed to obtain and manage topology; even Element Management Systems start with the need for learning and manipulating the topology. Similarly, orchestration layers for applications start with the need to manage topology and the associated database.

Detailed topology includes aspects such as physical nodes, physical links, virtual links, port to interface mapping, etc. The details should include the operational and administrative state as well as relevant parameters ranging from link bandwidth to SRLG membership. Layering is critical to provide the topology at the level of abstraction where it can be easily used by the application.

A key aspect of this interface is the ability to easily rate-limit, filter and specify the desired information to be extracted. This will help in allowing the interface to scale when queries are done.

5.2. Measurements

IPFIX [RFC5470] provides a way to measure and export per-traffic flow statistics. Applications that need to collect information about particular flows thus have a clear need to be able to install state to configure IPFIX to measure and export the relevant flows to the appropriate collectors.

5.3. Events

A programmatic interface for application to subscribe to asynchronous events is necessary. In addition to the interface state events already mentioned, an application may wish to subscribe to certain OAM-triggered events that aren't otherwise exported.

A RIB-based event could be reporting when the next-hops associated

with a route have changed. Other events could be used to verify that forwarding state has been programmed. For example, an application could request an event whenever a particular route in the RIB has its forwarding plane installation completed.

When an application registers for events, the application may request to get only the first such event, all such events, or all events until a certain time.

The full set of such events, that are not specifically related to other interfaces, needs to be investigated and defined.

6. Manageability Considerations

Manageability plays a key aspect in IRS. Some initial examples include:

Data Authorization Levels: The data-models used for IRS need the ability to indicate the required authorization level for installing or reading a particular subset of data. This allows control of what interactions each application can have.

Identity Authorization Levels: Associated with an application's identity should be an identity authorization level that is in a heirarchy so that higher authorized applications can manage and remove the state and resources used by other applications. The top of such a heirarchy would be the router configuration itself.

Resource Limitations: Using IRS, applications can consume resources, whether those be operations in a time-frame, entries in the RIB, stored operations to be triggered, etc. The ability to set resource limits based upon authorization is critical.

Configuration Interactions: The interaction of state installed via the IRS and via a router's configuration needs to be clearly defined.

7. IANA Considerations

This document includes no request to IANA.

8. Security Considerations

This framework describes interfaces that clearly require serious consideration of security. The ability to identify, authenticate and

authorize applications that wish to install state is necessary and briefly described in Section 3.2. Security of communications from the applications is also required.

More specifics on the security requirements requires further investigation.

9. Acknowledgements

The authors would like to thank Ken Gray, Adrian Farrel, Bruno Rijsman, Rex Fernando, Jan Medved, John Scudder, and Hannes Gredler for their suggestions and review.

10. Informative References

- [I-D.gredler-idr-ls-distribution]
Gredler, H., Medved, J., Previdi, S., and A. Farrel,
"North-Bound Distribution of Link-State and TE Information
using BGP", draft-gredler-idr-ls-distribution-02 (work in
progress), July 2012.
- [I-D.ietf-isis-genapp]
Ginsberg, L., Previdi, S., and M. Shand, "Advertising
Generic Information in IS-IS", draft-ietf-isis-genapp-04
(work in progress), November 2010.
- [I-D.ietf-pce-stateful-pce]
Crabbe, E., Medved, J., Varga, R., and I. Minei, "PCEP
Extensions for Stateful PCE",
draft-ietf-pce-stateful-pce-01 (work in progress),
July 2012.
- [RFC3137] Retana, A., Nguyen, L., White, R., Zinin, A., and D.
McPherson, "OSPF Stub Router Advertisement", RFC 3137,
June 2001.
- [RFC5250] Berger, L., Bryskin, I., Zinin, A., and R. Coltun, "The
OSPF Opaque LSA Option", RFC 5250, July 2008.
- [RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek,
"Architecture for IP Flow Information Export", RFC 5470,
March 2009.
- [RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J.,
and D. McPherson, "Dissemination of Flow Specification
Rules", RFC 5575, August 2009.

- [RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, October 2009.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

Authors' Addresses

Alia Atlas (editor)
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

Email: akatlas@juniper.net

Thomas Nadeau
Juniper Networks
1194 N. Mathilda Ave.
Sunnyvale, CA 94089
USA

Email: tnadeau@juniper.net

Dave Ward
Cisco Systems
Tasman Drive
San Jose, CA 95134
USA

Email: wardd@cisco.com

Routing Area Working Group
Internet-Draft
Intended status: Informational
Expires: March 23, 2013

R. White
Verisign
S. Hares
Huawei Technologies (USA)
R. Fernando
Cisco Systems
September 19, 2012

Use Cases for an Interface to the Routing System
draft-white-irs-use-case-00

Abstract

Programmatic interfaces to provide control over individual forwarding devices in a network promise to reduce operational costs while improving scaling, control, and visibility into the operation of large scale networks. To this end, several programmatic interfaces have been proposed. OpenFlow, for instance, provides a mechanism to replace the dynamic control plane processes on individual forwarding devices throughout a network with off box processes that interact with the forwarding tables on each device. Another example is NETCONF, which provides a fast and flexible mechanism to interact with device configuration and policy.

There is, however, no proposal which provides an interface to all aspects of the routing system as a system. Such a system would not interact with the forwarding system on individual devices, but rather with the control plane processes already used to discover the best path to any given destination through the network, as well as interact with the routing information base (RIB), which feeds the forwarding table the information needed to actually switch traffic at a local level.

This document describes a set of use cases such a system could fulfill. It is designed to provide underlying support for the framework, policy, and other drafts describing the Interface to the Routing System (IRS).

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 23, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
2. Optimized Exit Control	4
3. Distributed Reaction to Network Based Attacks	7
4. Remote Service Routing	8
5. Within Data Center Routing	10
6. Temporary Overlays between Data Centers	12
7. Central membership computation for MPLS based VPNs	13
8. Normative References	14
Authors' Addresses	15

1. Introduction

The Interface to the Routing System Framework [IRS] describes a mechanism where the distributed control plane can be augmented by an outside control plane through an open, accessible interface, including the Routing Information Base (RIB), in individual devices. This represents a "halfway point" between completely replacing the traditional distributed control plane and directly configuring devices to distribute policy or modifications to routing through off-board processes. This draft proposes a set of use cases that explain where the work described in [IRS] will be useful. The goal is to inform not only the community's understanding of where IRS fits in the larger scheme of SDN proposals, but also to inform the requirements, framework, and specification of IRS to provide the best fit for the purposes which make the most sense for this type of programmatic interface.

Towards this end the authors have searched for a number of different use cases representing not only complex modifications of the control plane, including interaction with applications and network conditions, but also simpler use cases. The array of use cases presented here should provide the reader with a solid understanding of the power of an SDN solution that will augment, rather than replace, traditional distributed control planes.

Each use case is presented in its own section.

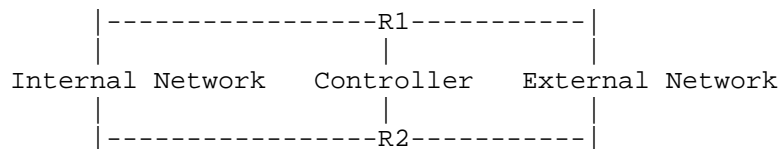
2. Optimized Exit Control

At edges where traffic exits along two or more possible paths, it is often desirable to choose a path based on more information the dynamic control plane provides. For instance, a network operator may want to take into account factors such as:

- o Cost per unit of data sent, including time of day variations, surcharges over a specific amount of data transmitted, and surcharges for transmitting data to specific types of destinations.
- o Urgency of data traffic or flow.
- o Exit point performance, including historical jitter, delay, and available bandwidth, possibly on a per destination basis.
- o Availability of a specific destination through a given link at the per destination basis (more specific than the routing protocol provides).

A number of possible solutions have been proposed or deployed in the past. For instance, the necessary metrics could be added to [BGP], or any other routing protocol, to provide the necessary information, and fine-tuned algorithms could be developed and deployed. Massive changes to well known and understood distributed control plane protocols to resolve a single use case, however, are not likely to be productive for the community as a whole. It's often difficult to justify the added complexity in the database and algorithms of routing protocols to solve what is considered a point case.

Another alternative has been the development of specific appliances designed to monitor the information necessary to provide an optimal edge decision, and then to use some automated configuration mechanism to transmit the decision to the edge routers. An example is illustrated in the figure below.



The controller in this network must:

- o Discover the topology of the network from R1 and R2.
- o Compare the current traffic flow information to policies set administratively by the network operator.
- o Monitor the flow of traffic from the perspective of R1 and R2.
- o Inject forwarding information to directly impact the traffic flow at the edge devices, or modify the policy of the existing distributed (dynamic) control plane already running in the network.

Many of these steps is challenging for currently available solutions.

To discover the topology at the edge routers, the controllers can either participate in the control plane, or walk the local routing table using a network management protocol. Neither of these options are optimal in this case because the controlling process cannot interact dynamically with the local topology information in near real time through such mechanisms.

Injecting forwarding information directly into the RIB on the individual devices in this network is possible today through the configuration of static routes through some external mechanism, such

as SNMP, NETCONF, or by direct external interaction with the devices' CLI. None of these options are attractive because:

- o They modify the actual configuration of the device (unlike a dynamic routing process).
- o They are too persistent (routes installed through static configuration persist across device reboots).
- o The controller cannot interact with the routing table in parallel with other routing processes. For instance, when a routing process attempts to install a new route in the routing table, there is often a callback or other notification to the other routing processes running on the same device; this notification provides important information the controller can take into account in its view of the current state of the routing table, and the state of the device's routing table. Interface level events also often trigger notifications from the RIB to local routing processes; these notifications would be invaluable for the controller to modify injected routing state in reaction to network topology events.
- o Routes installed through the an off box controller through the CLI or XML interface are difficult to redistribute into other protocols to draw traffic to a specific exit point, and it can be difficult to fine tune how these injected routes interact with routes learned through other routing processes.

IRS can resolve these issues by providing an open interface to the local RIB on each device, allowing the controller to interact with the RIB just as a local routing process would. This would allow the controlling process to see the topology information in the RIB dynamically, receiving near real time updates for route removals, installs, and other events, and without relying on static configuration to inject forwarding information each device can use.

Summary of IRS Capabilities and Interactions:

- o IRS should provide the ability to read the local RIB of each forwarding device, including the destination prefix (NLRI), a table identifier (if the forwarding device has multiple forwarding instances), the metric of each installed route, a route preference, and an identifier indicating the installing process.
- o The ability to monitor the available routes installed in the RIB of each forwarding device, including near real time notification of route installation and removal. This information must include the destination prefix (NLRI), a table identifier (if the

forwarding device has multiple forwarding instances), the metric of the installed route, and an identifier indicating the installing process.

- o The ability to install destination based routes in the local RIB of each forwarding device. This must include the ability to supply the destination prefix (NLRI), a table identifier (if the forwarding device has multiple forwarding instances), a route preference, a route metric, a next hop, an outbound interface, and a route process identifier.
- o The ability to interact with various policies configured on the forwarding devices, in order to inform the policies implemented by the dynamic routing processes. This interaction SHOULD be through existing configuration mechanisms, such as NETCONF, and SHOULD be recorded in the configuration of the local device so operators are aware of the full policy implemented in the network from the running configuration.
- o The ability to interact with traffic flow and other network traffic level measurement protocols and systems, in order to determine path performance, top talkers, and other information required to make an informed path decision based on locally configured policy.

3. Distributed Reaction to Network Based Attacks

Quickly modifying the control plane to reroute traffic for one destination while leaving a standard configuration in place (filters, metrics, and other policy mechanisms) is a challenge --but this is precisely the challenge of a network engineer attempting to deal with a network incursion. The ability to redirect specific flows of information or specific classes of traffic into, through, and back out of traffic analyzers on the fly is crucial in these situations. The following network diagram provides an illustration of the problem.

```

Valid Source---\  /--R2-----\
                  R1                      R3---Valid Destination
Attack Source--/  \--Monitoring Device-----/

```

Modifying the cost of the link between R1 and R2 to draw the attack traffic through the monitoring device in the distributed control plane will, of necessity, also draw the valid traffic through the monitoring device. Drawing valid traffic through a monitoring device introduces delay, jitter, and other quality of service issues, as well as posing a problem for the monitoring device itself in terms of

traffic load and management.

An IRS controller could stand between the detection of the attack and the control plane to facilitate the rapid modification of control and forwarding planes to either block the traffic or redirect it to analysis devices connected to the network.

Summary of IRS Capabilities and Interactions:

- o The ability to monitor the available routes installed in the RIB of each forwarding device, including near real time notification of route installation and removal. This information must include the destination prefix (NLRI), a table identifier (if the forwarding device has multiple forwarding instances), the metric of the installed route, and an identifier indicating the installing process.
- o The ability to install source and destination based routes in the local RIB of each forwarding device. This must include the ability to supply the destination prefix (NLRI), the source prefix (NLRI), a table identifier (if the forwarding device has multiple forwarding instances), a route preference, a route metric, a next hop, an outbound interface, and a route process identifier.
- o The ability to install a route to a null destination, effectively filtering traffic to this destination.
- o The ability to interact with various policies configured on the forwarding devices, in order to inform the policies implemented by the dynamic routing processes. This interaction SHOULD be through existing configuration mechanisms, such as NETCONF, and SHOULD be recorded in the configuration of the local device so operators are aware of the full policy implemented in the network from the running configuration.
- o The ability to interact with traffic flow and other network traffic level measurement protocols and systems, in order to determine path performance, top talkers, and other information required to make an informed path decision based on locally configured policy.

4. Remote Service Routing

In hub and spoke overlay networks, there is always an issue with balancing between the information held in the spoke routing table, optimal routing through the network underlying the overlay, and mobility. Most solutions in this space use some form of centralized

route server that acts as a directory of all reachable destinations and next hops, a protocol by which spoke devices and this route server communicate, and caches at the remote sites.

An IRS solution would use the same elements, but with a different control plane. Remote sites would register (or advertise through some standard routing protocol, such as BGP), the reachable destinations at each site, along with the address of the router (or other device) used to reach that destination. These would, as always, be stored in a route server (or several redundant route servers) at a central location.

When a remote site sends a set of packets to the central location that are eventually destined to some other remote site, the central location can forward this traffic, but at the same time simply directly insert the correct routing information into the remote site's routing table. If the location of the destination changes, the route server can directly modify the routing information at the remote site as needed.

An interesting aspect of this solution is that no new and specialized protocols are needed between the remote sites and the centralized route server(s). Normal routing protocols can be used to notify the centralized route server(s) of modifications in reachability information, and the route server(s) can respond as needed, based on local algorithms optimized for a particular application or network. For instance, short lived flows might be allowed to simply pass through the hub site with no reaction, while longer lived flows might warrant a specific route to be installed in the remote router. Algorithms can also be developed that would optimize traffic flow through the overlay, and also to remove routing entries from remote devices when they are no longer needed based on far greater intelligence than simple non-use for some period of time.

Summary of IRS Capabilities and Interactions:

- o The ability to read the local RIB of each forwarding device, including the destination prefix (NLRI), a table identifier (if the forwarding device has multiple forwarding instances), the metric of each installed route, a route preference, and an identifier indicating the installing process.
- o The ability to monitor the available routes installed in the RIB of each forwarding device, including near real time notification of route installation and removal. This information must include the destination prefix (NLRI), a table identifier (if the forwarding device has multiple forwarding instances), the metric of the installed route, and an identifier indicating the

installing process.

- o The ability to install destination based routes in the local RIB of each forwarding device. This must include the ability to supply the destination prefix (NLRI), a table identifier (if the forwarding device has multiple forwarding instances), a route preference, a route metric, a next hop, an outbound interface, and a route process identifier.

5. Within Data Center Routing

Data Centers have evolved into massive topologies with thousands of server racks and millions of hosts. Data Centers use BGP with ECMP, ISIS (with multiple LAGs), or other protocols to tie the data center together. Data centers are currently designed around a three or four tier structure with: server, top-of-rack switches, aggregation switches, and router interfacing the data center to the Internet. Microsoft's usage of BGP in the data center, described in [Lapukh-BGP], examines many of these elements of data center design.

One key element of these Data Center routing infrastructures is the ability to quickly read topology information and excute configuration from a centralized location. Key to this environment is the tight feedback loop between learning about topology changes or loading changes, and instantiating new routing policy. Without IRS, may Data Centers are using extra physical topologies or logical topologies to work around the features.

For example, Microsoft's network uses BGP because the topology state could be read from BGP impementations in a consistent fashion. Microsoft might have chosen a different routing protocol (such as ISIS) if the routing protocol state had been easier to obtain. Microsoft chose BGP for the data center because routers had a good BGP interface with topology information.

An IRS solution would use the same in the elements, but with a different control plane. The IRS enable control plane could provide the Data Center 4 tier infrastructure the quick access to topology and data flow information needed for traffic flow optimization. Changes to the Data Center infrastructure done via the IRS could have a tight feedback loop.

Again, this solution would reduce the need for new and specialized protocols while giving the Data Center the control it desire. The IRS routing interface could be extended to virtual routers.

Summary of IRS Capabilities and Interactions:

- o The ability to read the local RIB of each forwarding device, including the destination prefix (NLRI), a table identifier (if the forwarding device has multiple forwarding instances), the metric of each installed route, a route preference, and an identifier indicating the installing process.
- o The ability to monitor the available routes installed in the RIB of each forwarding device, including near real time notification of route installation and removal. This information must include the destination prefix (NLRI), a table identifier (if the forwarding device has multiple forwarding instances), the metric of the installed route, and an identifier indicating the installing process.
- o The ability to install destination based routes in the local RIB of each forwarding device. This must include the ability to supply the destination prefix (NLRI), a table identifier (if the forwarding device has multiple forwarding instances), a route preference, a route metric, a next hop, an outbound interface, and a route process identifier.
- o The ability to read the tables of other local protocol processes running on the device. This reading action SHOULD be supported through an import/export interface which can present the information in a consistent manner across all protocol implementations, rather than using a protocol specific model for each type of available process.
- o The ability to inject information directly into the local tables of other protocol processes running on the forwarding device. This injection SHOULD be supported through an import/export interface which can inject routing information in a consistent manner across all protocol implementations, rather than using a protocol specific model for each type of available process.
- o The ability to interact with various policies configured on the forwarding devices, in order to inform the policies implemented by the dynamic routing processes. This interaction SHOULD be through existing configuration mechanisms, such as NETCONF, and SHOULD be recorded in the configuration of the local device so operators are aware of the full policy implemented in the network from the running configuration.
- o The ability to interact with traffic flow and other network traffic level measurement protocols and systems, in order to determine path performance, top talkers, and other information required to make an informed path decision based on locally configured policy.

6. Temporary Overlays between Data Centers

Data Centers within one organization may operate as one single entity even though the Data Centers are geographically distributed fashion. Applications are load balanced within Data Centers and between data centers to take advantage of cost economics in power, storage, and server availability for compute resources. Applications are also transfer to alternate data centers in case of failures within a data center. To reduce time during failure, Data Centers often replicate user storage between two or more data centers. During the transfer of stored information prior to a Data Center to Data Center move, the Data Center controllers need to dynamically acquire a large amount of inter-data center bandwidth through an overlay network, often during off hours.

IRS could provide the connection between the overlay network configuration, local policies, and the control plane to dynamically bring a large bandwidth inter-data center overlay or channel into use, and then to remove it from use when the data transfer is completed.

Similarly, during a fail-over, a control process within data centers interacts with a group host process and the network to seamless move the processing to another data center. During the fail-over case, additional process state may need to be moved as well to restart the system. The difference between these data-to-data center moves is immediate and urgent need to move systems. If an application (such as medical or banking services) pays to have this type of fail-over, it is likely the service will pay for preemption on network bandwidth. IRS can allow the Data Center network and the Network connecting the data center to preempt other best-effort traffic to send this priority data flow. After the high priority data flow has finished, networks can return to their previous condition

Summary of IRS Capabilities and Interactions:

- o The ability to read the local RIB of each forwarding device, including the destination prefix (NLRI), a table identifier (if the forwarding device has multiple forwarding instances), the metric of each installed route, a route preference, and an identifier indicating the installing process.
- o The ability to monitor the available routes installed in the RIB of each forwarding device, including near real time notification of route installation and removal. This information must include the destination prefix (NLRI), a table identifier (if the forwarding device has multiple forwarding instances), the metric of the installed route, and an identifier indicating the

installing process.

- o The ability to install destination based routes in the local RIB of each forwarding device. This must include the ability to supply the destination prefix (NLRI), a table identifier (if the forwarding device has multiple forwarding instances), a route preference, a route metric, a next hop, an outbound interface, and a route process identifier.
- o The ability to interact with various policies configured on the forwarding devices, in order to inform the policies implemented by the dynamic routing processes. This interaction SHOULD be through existing configuration mechanisms, such as NETCONF, and SHOULD be recorded in the configuration of the local device so operators are aware of the full policy implemented in the network from the running configuration.
- o The ability to interact with policies and configurations on the forwarding devices using time based processing, either through timed auto-rollback or some other mechanism. This interaction SHOULD be through existing configuration mechanisms, such as NETCONF, and SHOULD be recorded in the configuration of the local device so operators are aware of the full policy implemented in the network from the running configuration.
- o The ability to interact with traffic flow and other network traffic level measurement protocols and systems, in order to determine path performance, top talkers, and other information required to make an informed path decision based on locally configured policy.

7. Central membership computation for MPLS based VPNs

MPLS based VPNs use route target extended communities to express membership information. Every PE router holds incoming BGP NLRI and processes them to determine membership and then import the NLRI into the appropriate MPLS/VPN routing tables. This consumes resources, both memory and compute on each of the PE devices.

An alternative approach is to monitor routing updates on every PE from the attached CEs and then compute membership in a central manner. Once computed the routes are pushed to the VPN RIBs of the participating PEs.

This centralization of membership control has a few advantages.

- o The membership mechanism (route-targets) need not be configured in each of the PEs and can be expressed once centrally.
- o No resources in the PEs need to be spent to categorize routes into the VRF tables that they belong and to filter out unwanted state.
- o Doing it centrally means the availability of almost unlimited compute capacity to compute membership and hence can be done in a scaleable manner.
- o More sophisticated routing policies and filters can be applied during the central import/export process than can be expressed and performed using the traditional route target mechanism.
- o Routes can be selectively pushed only to the participating PE's further reducing the memory load on the individual routers in the network. This further obviates for a distributed mechanisms such as rt constraints to reduce unnecessary path state in the routers.

Note that centrally computation of membership can be applied to other scenarios as well such as VPLS, MVPNs, MAC VPNs etc. Depending on the scenario, what gets monitored from the CE might vary. Central computation will especially help VPLS where multi-homing and load balancing using distributed techniques has particularly been a challenge.

Also note that one of the biggest promises of central route computation is simplification and reduction of computation and memory load on all devices in the network. This use case is just one example that illustrates these benefits of central computation very well.

Summary of IRS Capabilities and Interactions:

- o The ability to read the loc-RIB-In BGP table that gets all the routes that the CE has provided to a PE router.
- o The ability to install destination based routes in the local RIB of the PE devices. This must include the ability to supply the destination prefix (NLRI), a table identifier, a route preference, a route metric, a next-hop tunnel through which traffic would be carried

8. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

Authors' Addresses

Russ White
Verisign
12061 Bluemont Way
Reston, VA 20190
USA

Email: riwhite@verisign.com

Susan Hares
Huawei Technologies (USA)
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: Susan.Hares@huawei.com

Rex E. Fernando
Cisco Systems
170 W Tasman Dr
San Jose, CA 95134
USA

Email: rex@cisco.com

