

Internet Engineering Task Force
I2RS working group
Internet Draft
Category: Informational

N. Bitar
Verizon
G. Heron
L. Fang
Microsoft
R. Krishnan
Brocade Communications
N. Leymann
Deutsche Telekom
H. Shah
Ciena
S. Chakrabarti
W. Haddad
Ericsson

Expires: August 2014

February 14, 2014

Interface to the Routing System (I2RS) for Service Chaining:
Use Cases and Requirements

draft-bitar-i2rs-service-chaining-01

Abstract

Service chaining is the concept of applying an ordered set of services to a packet or a flow. Services in the chain may include network services such as load-balancing, firewalling, intrusion prevention, and routing among others. Criteria for applying a service chain to a packet or flow can be based on packet/flow attributes that span the OSI layers (e.g., physical port, Ethernet MAC header information, IP header information, transport, and application layer information). This document describes use cases and I2RS (Information to the Routing System) requirements for the discovery and maintenance of services topology and resources. It also describes use cases and I2RS requirements for controlling the forwarding of a packet/flow along a service chain based on packet/flow attributes.

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/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 August 14, 2014.

Copyright Notice

Copyright (c) 2014 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.

Conventions used in this document

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].

Table of Contents

1. Introduction.....	4
2. Abbreviations and Definitions.....	5
2.1. Abbreviations.....	5
2.2. Definitions.....	5
3. Service Chaining Use Cases and Requirements.....	5
3.1. Services topology.....	5
3.2. Monitoring Information.....	8
3.3. Traffic Redirection, Forwarding and Service Chaining.....	9
4. Service Chaining via BGP-based Redirection.....	12
5. Operational Considerations.....	13
6. IANA Considerations.....	13
7. Security Considerations.....	13
8. Acknowledgements.....	13
9. References.....	13
9.1. Normative References.....	13
9.2. Informative References.....	14
Authors' Addresses.....	14

1. Introduction

Several networking scenarios involve applying a set of services to a packet or flow. For instance, when a host in a protected zone initiates a session to a server outside the zone, the session may be directed to a chain of a Wide Area Network (WAN) application acceleration service, a network address and port translation (NAPT) service, and a firewall. On the server side, another set of services may also be applied. Such a sequence of services applied to a packet or flow is referred to as a service chain. Services in the chain may include deep packet inspection (DPI), load-balancing, firewalling, intrusion prevention, and routing among others.

Criteria for applying a service chain to a packet or flow can be based on packet/flow attributes that span the OSI layers. Such attributes may include the physical/virtual port on which the packet arrives, Ethernet MAC header information (e.g., VLAN ID), IP header information (e.g., source IP address), transport header information (e.g., TCP destination port number), and application layer information among others.

The transition from one service to the next in a service chain may be conditioned on the output of the current service, or may be non-conditional (pre-determined). A new mechanism, to be defined, may also enrich the packet transition in a service chain by passing service-specific information and/or information pertaining to preceding services in the chain along with the packet being processed. This type of mechanism and its influence are outside the scope of this document. In addition, this version of the document addresses the simple use case of pre-determined service chains applied to non-dropped packets with no additional information from preceding services. The service path for a packet/flow may be established via a management plane or routing, and may be enforced in the data plane via different mechanisms, as discussed in this document.

Services in a chain can be co-located on one system and/or physically separated across systems. In either case, a service may be running in its own virtualized system space or natively on the hosting system.

This document describes use cases and I2RS [i2rs-prob] requirements for the discovery and maintenance of services topology and resources. It also describes use cases and I2RS requirements for controlling the forwarding of a packet/flow along a service chain based on packet/flow attributes.

2. Abbreviations and Definitions

2.1. Abbreviations

2.2. Definitions

3. Service Chaining Use Cases and Requirements

A service chain is an ordered set of services applied to a packet or flow. It is often the case that when a flow in a bidirectional session is assigned to a service chain, the reverse flow of the same session is required to traverse the same chain in the reverse order. Assigning a flow to a service chain is often defined at an abstract level. Mapping a service chain to a network requires knowledge of the available services, their locations and available resources so that services are properly engineered on the services infrastructure. This section describes requirements and applicability for such information, and for directing traffic through a service chain.

3.1. Services topology

In order to establish a service chain that applies to a packet/flow, it is important to have a topology of the service nodes. A service node can be a service running natively within a system (e.g., a service card or a service engine in a router), a virtual machine (VM) hosted on a server, a VM hosted on a service engine within a system (e.g., a service card in a router), or a dedicated standalone service hardware appliance. In addition, a service node may be dedicated to a customer (e.g., an IPVPN customer), globally shared across customers or a customer set of VPNs, or available to be assigned in whole or in part to a customer or a set customer VPNs. A customer and tenant are used synonymously in this document. How a service node is created is outside the scope of this document. Resources on a service node that are not assigned to a customer context (e.g., VRF) will be logically referred to as a non-assigned service node with free available resources. A service node that can be shared in a global context will be referred to as a global service node. It should be noted, that once a service node is bound to a context, then it is only available for a virtual network (VN) associated with that context.

Different service node types may have information specific to the service(s) they provide. A service node information model needs to describe information common (generic) to all service node types and extensible to be sub-classed so that the service

Internet-Draft I2RS for Service Chaining February 2014
specific information can be represented. The common information
is:

- . Service node address: A service node must have a unique address in a service topology. A service node identifier address can be:
 - o An IP address when feasible. Such a service node can be a VM, a services engine within a system, or a hardware appliance.
 - o The tuple (service node IP address, hosting system IP address). This applies when there is need to identify the system hosting the service node or when the service node IP address is only reachable within the hosting system.
 - o The tuple (hosting system IP-address, system internal identifier for the service engine). This applies when the service engine is not IP addressable and is within a system. A potential system internal identifier for a service engine may be (system_slot_number.subslot_number.engine_number).
- . For each service node, the following information is required:
 - o Supported service type (e.g., NAT, FW). A node may support multiple service types.
 - o Number of virtual contexts (tenants) that can be supported. This parameter will indicate the maximum number of contexts that can be created on the service node.
 - o Number of virtual contexts (e.g., VRFs) available.
 - o Supported context type (e.g., VRF).
 - o Customer ID if the service node is dedicated to a customer. This indicates who can use this service node.
 - o List of supported (customer ID, virtual contexts). Note that one context per customer is a degenerate case. This will be the global context for a given customer on a service node.

For each service node, virtual context and service type, the following information may be specified, depending on the service resource requirement. That is, some of the information listed here may not be relevant for some services.

- o Service bandwidth capacity
- o Supported Packet rate (packets per second)
- o Supported Bandwidth (e.g., in kbps)
- o IP Forwarding Information Base size per address family
- o Routing Information Base size
- o MAC Forwarding database size
- o Number of 64-bit statistics counters for policy-based accounting
- o Number of supported Access lists (ACLs) per type (e.g., number of bits per ACL, and ACL type if applicable)
- o Number of supported flows for services that require it (e.g., Firewall, NAT, stateful load-balancing, Deep Packet Inspection (DPI)) per flow type (i.e., fields identifying a flow) or flow identification key size. For systems that allow flexible memory usage across flow types and/or key sizes, it is sufficient to track available memory allocated for flows.

In addition to the services topology, it is important to have a view of the Virtual Network (VN) topology (VNT) and access points to which a services topology applies. The topology of such a VN could be relatively static, but it may also be dynamic, especially in a cloud environment where compute, storage, applications and associated networks may be created and removed over a short time scale. The description of a VN topology encompassing the access points is important in order to enable installation of policies for service chaining at the right access points, instantiate the services if needed, and perform the necessary monitoring as described in later sections. VN topology information requirements are described in [i2rs-topology-reqts], but they need to be augmented with the following information:

- . Access ports (systems and ports) per VN. A port may be physical or logical on a physical port.
- . Addresses reachable on an access port.

3.2. Monitoring Information

Service chaining requires the ability to monitor the state of each service node, including liveness and resource utilization. If a service node failure is detected, an action may be taken to create another service node and steer traffic to it. If a service node is hitting a resource utilization threshold, traffic may be directed to other service nodes, and/or additional service nodes may be created.

The following is a set of parameters that needs be monitored per service node per virtual context, and per service type as applicable. It should be noted that some services may not require all the parameters listed here to be monitored.

- . Bandwidth utilization (e.g., in kbps)
- . Packet rate utilization (packets per second)
- . Bandwidth utilization per CoS (e.g., in kbps)
- . Packet rate utilization per Cos
- . Memory utilization and available memory
- . RIB utilization per address family
- . FIB utilization per address family
- . Flow resource utilization per flow type
- . CPU utilization as applicable
- . Available storage

The following is a set of parameters that needs to be monitored globally per physical system (e.g., host server) providing services or hosting service nodes. Note that some parameters may not be needed for some services:

- . Bandwidth utilization (e.g., in kbps)

- . Packet rate utilization (packets per second)
- . Bandwidth utilization per Class of Service (CoS)
- . Packet rate per CoS (packets per second)
- . Memory utilization and available memory
- . RIB utilization and available RIB memory if applicable per address family
- . FIB utilization and available FIB entries if applicable per address family
- . Flow resource utilization per flow type if applicable
- . CPU utilization if applicable
- . Power utilization
- . Available storage

Such information needs to be maintained on the distributed system hosting a service node, and/or service node as applicable. In addition, a mechanism to monitor the liveness of a service node must be available. For some use cases, liveness and resource utilization information needs to be accessible to a management/control plane that provides for creation of service nodes and orchestration of service chains. Some of this information may also be maintained in the management/orchestration system and validated with the distributed system where the services are instantiated. For some other use cases, a service node and/or hosting system may need to be programmed to update a management system with that information periodically or when a configured high watermark or low watermark is reached for a parameter. Thus, the interface to the service nodes and/or hosting systems must provide a mechanism that enables a management/control system to pull resource utilization information from these nodes and systems, and for these nodes and system to send updates on resource utilization to a designated system.

3.3. Traffic Redirection, Forwarding and Service Chaining

In a service chain, it is important to be able to direct traffic from one service node to another. Some solutions may provide this capability via dynamic routing, data-plane based

policy-based routing, source based routing or a combination. Traffic redirection to a service chain requires the ability to program the routing system with a classification rule that identifies a packet/flow and an associated action that directs the corresponding packet(s) to the first node in the service chain. The focus in this section is on a hop-by-hop policy-based routing (PBR) and source based service routing. At the redirection point, classification rules MUST support the following information that encompasses Layer1-7 information, any of which may be wild-carded or left unspecified for a particular case:

- . Port
- . VLAN/VLAN stack
- . MAC source address
- . MAC destination address
- . Host/subnet Source IP address
- . Host/subnet Destination IP address
- . IP version
- . IP protocol
- . Source port/port-range
- . Destination port/port-range
- . Optionally, application-layer information such as key words in a URI, content type or user agent

As a result of the classification, an action will need to be specified to direct the matching packet to a service node, or to perform other action(s). The following actions MUST be supported:

- . Forward to a specified Outgoing port (physical or logical):
 - o VLAN ID
 - o IP/GRE tunnel

- o RSVP-TE tunnel
 - o Pseudowire (PW)
 - o Other types of tunneling protocols
- . Steer the packet to a VRF
- . Mirror packet:
 - o To an IP destination
 - o To a port
 - o over a VLAN
 - o over an IP/GRE tunnel
 - o over an RSVP-TE tunnel
 - o over a Pseudowire
 - o over other types of tunneling
- . Route. This could be the default behavior at the tail end of a chain or the result of no match.
- . Route the packet to a specific system that is multiple IP hops away (Layer 3 policy based routing). The destination system IP address must be specified along with the tunneling type. The action must result in encapsulating the packet to the destination. At the destination, a policy must be installed to apply a service in a specific context to the arriving packet, or direct the traffic to a local service node.
- . Insert a source route header in the transmitted packet that identifies the nodes along the service path. The service route may be composed of IPv4 routes, IPv6 routes and/or a stack of MPLS labels. The source route may capitalize on existing mechanism or new mechanisms that are outside the scope of this document. At the destination, a policy must be installed to apply a service in a specific context to the arriving packet, or direct the traffic to a local service node.

- . Insert a source route+service header that identifies the service path and the service type to be applied at each node. This will require the definition of a new data plane header that carries such information.

The number of classification rules and associated actions, as well as the rate of programmability/removal of these rules will be highly application dependent. When the service chain is based on static policy (e.g., applied to a port, a source subnet, a VN), these rules will be programmed on a system at the rate of provisioning. When the attributes of the policies are relatively static (e.g., applied to a fixed port in fixed wireline access), the rate of provisioning on the forwarding system could be low, on the order of few hundred per day. When the attributes are more dynamic, such as in a mobile environment on a system handling a large number of users, that rate could be much higher. In a cloud environment where tenant systems may be spun up and removed on a relatively short time scale this rate could be on the order of few hundreds to thousands a minute at a DC GW for instance. In all cases, if the state is not kept in a persistent storage on the forwarding system(s), system reboot actions will trigger the need for a high provisioning rate, on the order at few thousands per second. When policies are triggered by data-plane, the rate of policy provisioning will be on the order of flow rates and removal will be dependent on the flow duration. These rates will be highly dependent on the applications as well, but at a system that is handling a large number of flows, the protocol used in provisioning must be very efficient to handle a very large number of flows.

4. Service Chaining via BGP-based Redirection

BGP-based steering of a traffic flow to a first service point may be required in certain cases. In this case, a router hosting a service node or connected to a service node will advertise a flow specification that causes a system that receives the advertisement to redirect a packet or mirror a copy of the packet that matches the flow specification to the advertising route [BGP-flowspec]. When the advertising router supports the i2rs BGP service, an I2RS interface to the router can provision the router with the appropriate BGP policy as well as install on that router a forwarding policy that directs the packet when received to the appropriate service node. Such BGP advertisements can be chained to effect the chaining of multiple services.

5. Operational Considerations

6. IANA Considerations

There is IANA action required by this document.

7. Security Considerations

Service chaining imposes several security issues that must be addressed. First, the control system that installs policies on the routing system must be trusted by that system. An untrusted control system may install policies that hijack traffic, cause denial of service, or mirror traffic to an untrusted entity for eavesdropping. Thus the communication channel between a control system and routing system must be authenticated, and may be encrypted. In addition, when services are being offered to multiple VPN customers with overlapping IP addresses, it is important that the customer privacy is maintained when applying a service chain to a customer packet/flow. Thus, the ability to identify the context in which a service needs to be applied is important. In addition, policies must be installed in the appropriate context. Finally, congesting a service node can result in packet drops that may effectively result in a denial of service. Thus, obtaining information about the performance of a service node is important to detect overload conditions and take corrective action.

8. Acknowledgements

The authors thank David McDysan and Alia Atlas for their comments.

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.

[i2rs-prob] Atlas, A., Nadeau, T., and Ward, D., "Interface to the Routing System Problem Statement", draft-ietf-i2rs-problem-statement-00, August 2013. Work in progress.

[i2rs-topology-reqts] Medved, J., et al., "Topology API Requirements", draft-medved-i2rs-topology-requirements, February 2013. Work in progress.

[BGP-flowspec] Uttaro, J., et al., "BGP Flow-Spec Extended Community for Traffic Redirect to IP Next Hop", draft-simpson-idr-flowspec-redirect-02, November 2012. Work in Progress.

9.2. Informative References

Authors' Addresses

Nabil Bitar
Verizon
60 Sylvan Rd.
Waltham, MA 02145
EMail: nabil.n.bitar@verizon.com

Giles Heron
Cisco Systems
EMail: giheron@cisco.com

Luyuan Fang
Microsoft
EMail: luyuanf@gmail.com

Ram Krishnan
Brocade Communications
San Jose, CA 95134
EMail: ramk@brocade.com

Nicolai Leymann
Deutsche Telekom
Winterfeldtstrasse 21-27
10781 Berlin
Germany
EMail: n.leymann@telekom.de

Himanshu Shah
Ciena
EMail: hshah@ciena.com

Samita Chakrabarti
Ericsson
EMail: samita.chakrabarti@ericsson.com

Wassim Haddad

Internet-Draft

I2RS for Service Chaining

February 2014

Ericsson

EMail: wassim.haddad@ericsson.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 16, 2014

A. Atlas
Juniper Networks
J. Halpern
Ericsson
S. Hares
Hickory Hill Consulting
D. Ward
Cisco Systems
T. Nadeau
Brocade
February 12, 2014

An Architecture for the Interface to the Routing System
draft-ietf-i2rs-architecture-02

Abstract

This document describes an architecture for a standard, programmatic interface for state transfer in and out of the Internet's routing system. It describes the basic architecture, the components, and their interfaces with particular focus on those to be standardized as part of I2RS.

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 August 16, 2014.

Copyright Notice

Copyright (c) 2014 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. Drivers for the I2RS Architecture	4
1.2. Architectural Overview	4
2. Terminology	8
3. Key Architectural Properties	10
3.1. Simplicity	10
3.2. Extensibility	10
3.3. Model-Driven Programmatic Interfaces	11
4. Security Considerations	11
4.1. Identity and Authentication	12
4.2. Authorization	13
5. Network Applications and I2RS Client	13
5.1. Example Network Application: Topology Manager	14
6. I2RS Agent Role and Functionality	14
6.1. Relationship to its Routing Element	15
6.2. I2RS State Storage	15
6.2.1. I2RS Agent Failure	15
6.2.2. Starting and Ending	16
6.2.3. Reversion	16
6.3. Interactions with Local Config	17
6.4. Routing Components and Associated I2RS Services	17
6.4.1. Routing and Label Information Bases	18
6.4.2. IGPs, BGP and Multicast Protocols	19
6.4.3. MPLS	19
6.4.4. Policy and QoS Mechanisms	20
6.4.5. Information Modeling, Device Variation, and Information Relationships	20
6.4.5.1. Managing Variation: Object Classes/Types and Inheritance	20
6.4.5.1.1. Managing Variation: Optionality	21
6.4.5.1.2. Managing Variation: Templating	21
6.4.5.1.3. Object Relationships	22
7. I2RS Client Agent Interface	23
7.1. One Control and Data Exchange Protocol	23
7.2. Communication Channels	23
7.3. Capability Negotiation	23
7.4. Identity and Security Role	24
7.4.1. Client Redundancy	24

7.5. Connectivity	24
7.6. Notifications	25
7.7. Information collection	26
7.8. Multi-Headed Control	26
7.9. Transactions	27
8. Manageability Considerations	27
9. IANA Considerations	28
10. Acknowledgements	28
11. Informative References	28
Authors' Addresses	28

1. Introduction

Routers that form the Internet's routing infrastructure maintain state at various layers of detail and function. For example, a typical router maintains a Routing Information Base (RIB), and implements routing protocols such as OSPF, ISIS, and BGP to exchange protocol state and other information about the state of the network with other routers.

Routers know how to convert all of this information into the forwarding operations that are installed in the forwarding plane. The forwarding plane and the specified forwarding operations then contain active state information that describes the expected and observed operational behavior of the router and which is also needed by the network applications. Network-oriented applications require easy access to this information to learn the network topology, to verify that programmed state is installed in the forwarding plane, to measure the behavior of various flows, routes or forwarding entries, as well as to understand the configured and active states of the router.

This document sets out an architecture for a common, standards-based interface to this information. This Interface to the Routing System (I2RS) facilitates control and observation of the routing-related state (for example, a Routing Element RIB manager's state), as well as enabling network-oriented applications to be built on top of today's routed networks. The I2RS is a programmatic asynchronous interface for transferring state into and out of the Internet's routing system. This I2RS architecture 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 I2RS are clear data models that define the semantics of the information that can be written and read. The I2RS provides a framework for registering for and requesting the appropriate information for each particular application. The I2RS

provides a way for applications to customize network behavior while leveraging the existing routing system as desired.

Although the I2RS architecture is general enough to support information and data models for a variety of data, the I2RS, and therefore this document, are specifically focused on an interface for routing data.

1.1. Drivers for the I2RS Architecture

There are four key drivers that shape the I2RS architecture. First is the need for an interface that is programmatic, asynchronous, and offers fast, interactive access. Second is the access to structured information and state that is frequently not directly configurable or modeled in existing implementations or configuration protocols. Third is the ability to subscribe to structured, filterable event notifications from the router. Fourth, the operation of I2RS is to be data-model driven to facilitate extensibility and provide standard data-models to be used by network applications.

I2RS is described as an asynchronous programmatic interface, the key properties of which are described in Section 5 of [I-D.ietf-i2rs-problem-statement].

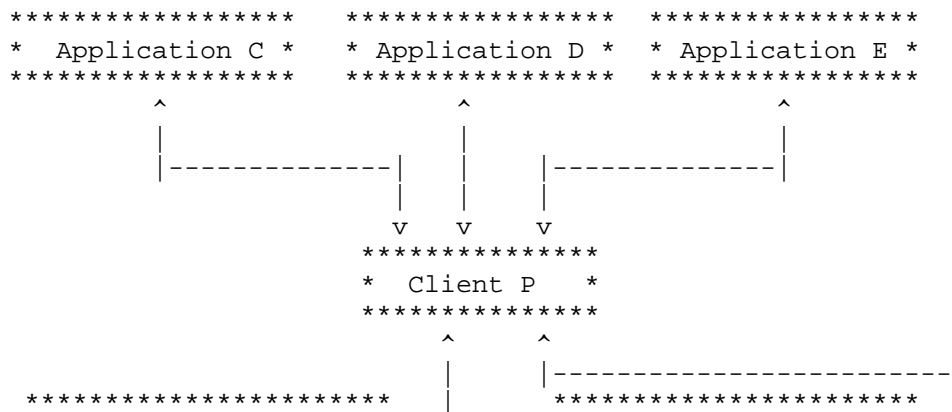
The I2RS facilitates obtaining information from the router. The I2RS provides the ability to not only read specific information, but also to subscribe to targeted information streams and filtered and thresholded events.

Such an interface also facilitates the injection of ephemeral state into the routing system. A non-routing protocol or application could inject state into a routing element via the state-insertion functionality of the I2RS and that state could then be distributed in a routing or signaling protocol and/or be used locally (e.g. to program the co-located forwarding plane). I2RS will only permit modification of state that would be safe, conceptually, to modify via local configuration; no direct manipulation of protocol-internal dynamically determined data is envisioned.

1.2. Architectural Overview

Figure 1 shows the basic architecture for I2RS between applications using I2RS, their associated I2RS Clients, and I2RS Agents. Applications access I2RS services through I2RS clients. A single client can provide access to one or more applications. In the figure, Clients A and B provide access to a single application, while Client P provides access to multiple applications.

The scope of I2RS is to define the interactions between the I2RS agent and the I2RS client and the associated proper behavior of the I2RS agent and I2RS client.



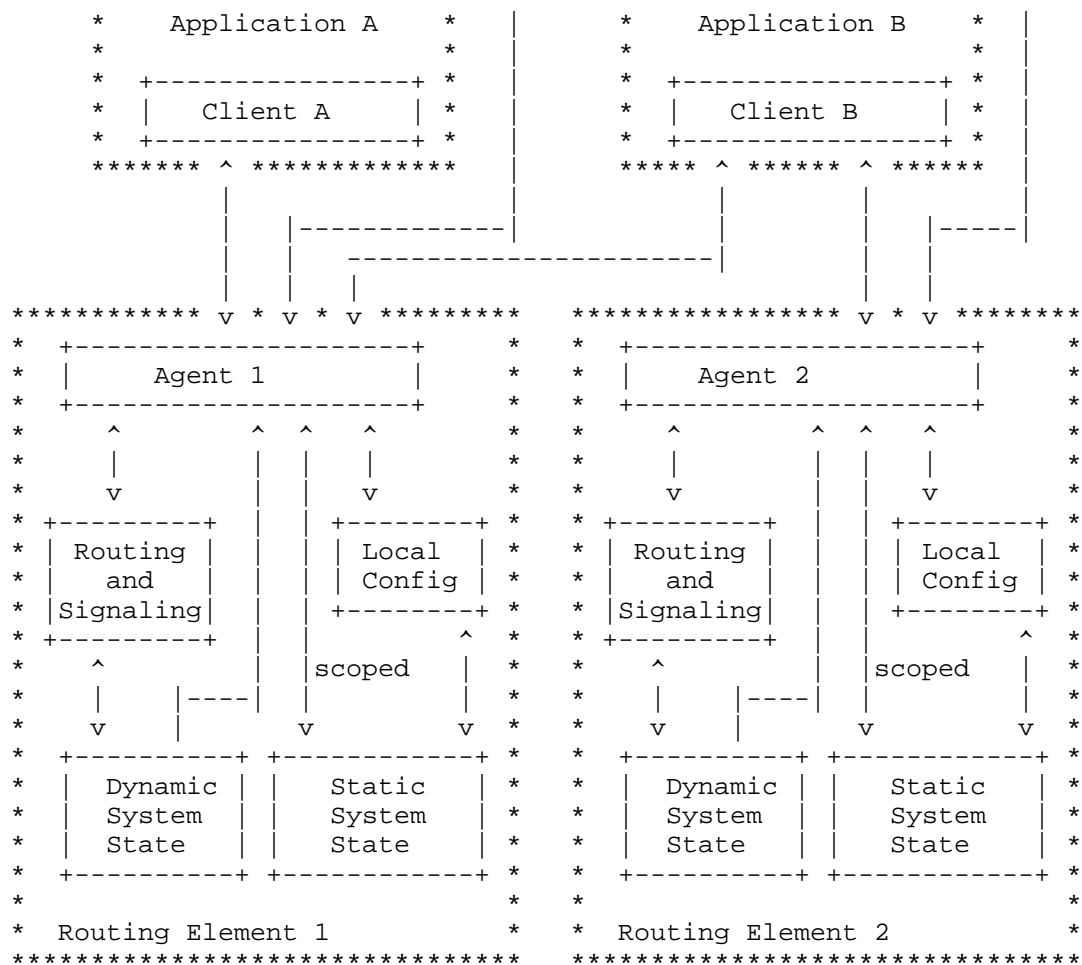


Figure 1: Architecture of I2RS clients and agents

Routing Element: A Routing Element implements some subset of the routing system. It does not need to have a forwarding plane associated with it. Examples of Routing Elements can include:

- * A router with a forwarding plane and RIB Manager that runs ISIS, OSPF, BGP, PIM, etc.
- * A server that runs BGP as a Route Reflector
- * An LSR that implements RSVP-TE, OSPF-TE, and PCEP and has a forwarding plane and associated RIB Manager.

- * A server that runs ISIS, OSPF, BGP and uses ForCES to control a remote forwarding plane.

A Routing Element may be locally managed, whether via CLI, SNMP, or NETCONF.

Routing and Signaling: This block represents that portion of the Routing Element that implements part of the Internet routing system. It includes not merely standardized protocols (i.e. IS-IS, OSPF, BGP, PIM, RSVP-TE, LDP, etc.), but also the RIB Manager layer.

Local Config: A Routing Element will provide the ability to configure and manage it. The Local Config may be provided via a combination of CLI, NETCONF, SNMP, etc. The black box behavior for interactions between the state that I2RS installs into the routing element and the Local Config must be defined.

Dynamic System State: An I2RS agent needs access to state on a routing element beyond what is contained in the routing subsystem. Such state may include various counters, statistics, and local events. This is the subset of operational state that is needed by network applications based on I2RS that is not contained in the routing and signaling information. How this information is provided to the I2RS agent is out of scope, but the standardized information and data models for what is exposed are part of I2RS.

Static System State: An I2RS agent needs access to static state on a routing element beyond what is contained in the routing subsystem. An example of such state is specifying queueing behavior for an interface or traffic. How the I2RS agent modifies or obtains this information is out of scope, but the standardized information and data models for what is exposed are part of I2RS.

I2RS Agent: See the definition in Section 2.

Application: A network application that needs to observe the network or manipulate the network to achieve its service requirements.

I2RS Client: See the definition in Section 2.

As can be seen in Figure 1, an I2RS client can communicate with multiple I2RS agents. An I2RS client may connect to one or more I2RS agents based upon its needs. Similarly, an I2RS agent may communicate with multiple I2RS clients - whether to respond to their requests, to send notifications, etc. Timely notifications are

critical so that several simultaneously operating applications have up-to-date information on the state of the network.

As can also be seen in Figure 1, an I2RS Agent may communicate with multiple clients. Each client may send the agent a variety of write operations. In order to keep the protocol simple, the current view is that two clients should not be attempting to write (modify) the same piece of information. Such collisions may happen, but are considered error cases that should be resolved by the network applications and management systems.

In contrast, although multiple I2RS clients may need to supply data into the same list (e.g. a prefix or filter list), this is not considered an error and must be correctly handled. The nuances so that writers do not normally collide should be handled in the information models.

The architectural goal for the I2RS is that such errors should produce predictable behaviors, and be reportable to interested clients. The details of the associated policy is discussed in Section 7.8. The same policy mechanism (simple priority per I2RS client) applies to interactions between the I2RS agent and the CLI/SNMP/NETCONF as described in Section 6.3.

In addition it must be noted that there may be indirect interactions between write operations. A trivial example of this is when two different but overlapping prefixes are written with different forwarding behavior. Detection and avoidance of such interactions is outside the scope of the I2RS work and is left to agent design and implementation.

2. Terminology

The following terminology is used in this document.

agent or I2RS Agent: An I2RS agent provides the supported I2RS services from the local system's routing sub-systems by interacting with the routing element to provide specified behavior. The I2RS agent understands the I2RS protocol and can be contacted by I2RS clients.

client or I2RS Client: A client implements the I2RS protocol, uses it to communicate with I2RS Agents, and uses the I2RS services to accomplish a task. It interacts with other elements of the policy, provisioning, and configuration system by means outside of the scope of the I2RS effort. It interacts with the I2RS agents to collect information from the routing and forwarding system. Based on the information and the policy oriented interactions, the

I2RS client may also interact with I2RS agents to modify the state of the routing system the client interacts with to achieve operational goals. An I2RS client can be seen as the part of an application that uses and supports I2RS and could be a software library.

service or I2RS Service: For the purposes of I2RS, a service refers to a set of related state access functions together with the policies that control their usage. The expectation is that a service will be represented by a data-model. For instance, 'RIB service' could be an example of a service that gives access to state held in a device's RIB.

read scope: The set of information which the I2RS client is authorized to read. The read scope specifies the access restrictions to both see the existence of data and read the value of that data.

notification scope: The set of events and associated information that the I2RS Client can request be pushed by the I2RS Agent. I2RS Clients have the ability to register for specific events and information streams, but must be constrained by the access restrictions associated with their notification scope.

write scope: The set of field values which the I2RS client is authorized to write (i.e. add, modify or delete). This access can restrict what data can be modified or created, and what specific value sets and ranges can be installed.

scope: When unspecified as either read scope, write scope, or notification scope, the term scope applies to the read scope, write scope, and notification scope.

resources: A resource is an I2RS-specific use of memory, storage, or execution that a client may consume due to its I2RS operations. The amount of each such resource that a client may consume in the context of a particular agent may be constrained based upon the client's security role. An example of such a resource could include the number of notifications registered for. These are not protocol-specific resources or network-specific resources.

role or security role: A security role specifies the scope, resources, priorities, etc. that a client or agent has.

identity: A client is associated with exactly one specific identity. State can be attributed to a particular identity. It is possible for multiple communication channels to use the same

identity; in that case, the assumption is that the associated client is coordinating such communication.

secondary identity: An I2RS Client may supply a secondary opaque identity that is not interpreted by the I2RS Agent. An example use is when the I2RS Client is a go-between for multiple applications and it is necessary to track which application has requested a particular operation.

3. Key Architectural Properties

3.1. Simplicity

There have been many efforts over the years to improve the access to the information available to the routing and forwarding system. Making such information visible and usable to network management and applications has many well-understood benefits. There are two related challenges in doing so. First, the quantity and diversity of information potentially available is very large. Second, the variation both in the structure of the data and in the kinds of operations required tends to introduce protocol complexity.

Having noted that, it is also critical to the utility of I2RS that it be easily deployable and robust. Complexity in the protocol hinders implementation, robustness, and deployability. Also, data models complexity may complicate extensibility.

Thus, one of the key aims for I2RS is to keep the protocol and modeling architecture simple. So for each architectural component or aspect, we ask ourselves "do we need this complexity, or is the behavior merely nice to have?" Protocol parsimony is clearly a goal.

3.2. Extensibility

Naturally, extensibility of the protocol and data model is very important. In particular, given the necessary scope limitations of the initial work, it is critical that the initial design include strong support for extensibility.

The scope of the I2RS work is being restricted in the interests of achieving a deliverable and deployable result. The I2RS Working Group is modeling only a subset of the data of interest. It is clearly desirable for the data models defined in the I2RS to be useful in more general settings. It should be easy to integrate data models from the I2RS with other data. Other work should be able to easily extend it to represent additional aspects of the network elements or network systems. This reinforces the criticality of

designing the data models to be highly extensible, preferably in a regular and simple fashion.

The I2RS Working Group is defining operations for the I2RS protocol. It would be optimistic to assume that more and different ones may not be needed when the scope of I2RS increases. Thus, it is important to consider extensibility not only of the underlying services' data models, but also of the primitives and protocol operations.

3.3. Model-Driven Programmatic Interfaces

A critical component of I2RS is the standard information and data models with their associated semantics. While many components of the routing system are standardized, associated data models for them are not yet available. Instead, each router uses different information, different mechanisms, and different CLI which makes a standard interface for use by applications extremely cumbersome to develop and maintain. Well-known data modeling languages exist and may be used for defining the data models for I2RS.

There are several key benefits for I2RS in using model-driven architecture and protocol(s). First, it allows for transferring data-models whose content is not explicitly implemented or understood. Second, tools can automate checking and manipulating data; this is particularly valuable for both extensibility and for the ability to easily manipulate and check proprietary data-models.

The different services provided by I2RS can correspond to separate data-models. An I2RS agent may indicate which data-models are supported.

4. Security Considerations

This I2RS architecture describes interfaces that clearly require serious consideration of security. First, here is a brief description of the assumed security environment for I2RS. The I2RS Agent associated with a Routing Element is a trusted part of that Routing Element. For example, it may be part of a vendor-distributed signed software image for the entire Routing Element or it may be trusted signed application that an operator has installed. The I2RS Agent is assumed to have a separate authentication and authorization channel by which it can validate both the identity and permissions associated with an I2RS Client. To support numerous and speedy interactions between the I2RS Agent and I2RS Client, it is assumed that the I2RS Agent can also cache that particular I2RS Clients are trusted and their associated authorized scope. This implies that either in a pull model, the permission information may be old until the I2RS Agent rerequests it, or in a push model, that the

authentication and authorization channel can notify the I2RS Agent of changes.

An I2RS Client is not automatically trustworthy. It has identity information and applications using that I2RS Client should be aware of the scope limitations of that I2RS Client. If the I2RS Client is acting as a broker for multiple applications, managing the security, authentication and authorization for that communication is out of scope; nothing prevents I2RS and a separate authentication and authorization channel from being used. Regardless of mechanism, an I2RS Client that is acting as a broker is responsible for determining that applications using it are trusted and permitted to make the particular requests.

Different levels of integrity, confidentiality, and replay protection are relevant for different aspects of I2RS. The primary communication channel that is used for client authentication and then used by the client to write data requires integrity, privacy and replay protection. Appropriate selection of a default required transport protocol is the preferred way of meeting these requirements.

Other communications via I2RS will not require integrity, confidentiality, and replay protection. For instance, if an I2RS Client subscribes to an information stream of prefix announcements from OSPF, those may require integrity but probably not confidentiality or replay protection. Similarly, an information stream of interface statistics may not even require guaranteed delivery. In Section 7.2, more reasoning for multiple communication channels is provided. From the security perspective, it is critical to realize that an I2RS Agent may open a new communication channel based upon information provided by an I2RS Client; to avoid an indirect attack, such a request must be done in the context of an authenticated and authorized client whose communications cannot have been altered.

4.1. Identity and Authentication

As discussed above, all control exchanges between the I2RS client and agent should be authenticated and integrity protected (such that the contents cannot be changed without detection). Further, manipulation of the system must be accurately attributable. In an ideal architecture, even information collection and notification should be protected; this may be subject to engineering tradeoffs during the design.

I2RS clients may be operating on behalf of other applications. While those applications' identities are not needed for authentication or

authorization, each application should have a unique opaque identifier that can be provided by the I2RS client to the I2RS agent for purposes of tracking attribution of operations to support functionality such as accounting and troubleshooting.

4.2. Authorization

All operations using I2RS, both observation and manipulation, should be subject to appropriate authorization controls. Such authorization is based on the identity and assigned role of the I2RS client performing the operations and the I2RS agent in the network element.

I2RS Agents, in performing information collection and manipulation, will be acting on behalf of the I2RS clients. As such, each operation authorization will be based on the lower of the two permissions of the agent itself and of the authenticated client. The mechanism by which this authorization is applied within the device is outside of the scope of I2RS.

The appropriate or necessary level of granularity for scope can depend upon the particular I2RS Service and the implementation's granularity. An approach to a similar access control problem is defined in the NetConf Access Control Model[RFC6536]; it allows arbitrary access to be specified for a data node instance identifier while defining meaningful manipulable defaults. The ability to specify one or more groups or roles that a particular I2RS Client belongs and then define access controls in terms of those groups or roles is expected. When a client is authenticated, its group or role membership should be provided to the I2RS Agent. The set of access control rules that an I2RS Agent uses would need to be either provided via Local Config, exposed as an I2RS Service for manipulation by authorized clients, or via some other method.

5. Network Applications and I2RS Client

I2RS is expected to be used by network-oriented applications in different architectures. While the interface between a network-oriented application and the I2RS client is outside the scope of I2RS, considering the different architectures is important to sufficiently specify I2RS.

In the simplest architecture, a network-oriented application has an I2RS client as a library or driver for communication with routing elements.

In the broker architecture, multiple network-oriented applications communicate in an unspecified fashion to a broker application that contains an I2RS Client. That broker application requires additional

functionality for authentication and authorization of the network-oriented applications; such functionality is out of scope for I2RS but similar considerations to those described in Section 4.2 do apply. As discussed in Section 4.1, the broker I2RS Client should determine distinct opaque identifiers for each network-oriented application that is using it. The the broker I2RS Client can pass along the appropriate value as a secondary identifier which can be used for tracking attribution of operations.

In the third architecture, a routing element or network-oriented application that uses an I2RS Client to access services on a different routing element may also contain an I2RS agent to provide services to other network-oriented applications. However, where the needed information and data models for those services differs from that of a conventional routing element, those models are, at least initially, out of scope for I2RS. Below is an example of such a network application

5.1. Example Network Application: Topology Manager

A Topology Manager includes an I2RS client that uses the I2RS data models and protocol to collect information about the state of the network by communicating directly with one or more I2RS agents. From these I2RS agents, the Topology Manager collects routing configuration and operational data, such as interface and label-switched path (LSP) information. In addition, the Topology Manager may collect link-state data in several ways - either via I2RS models, by peering with BGP-LS[I-D.ietf-idr-ls-distribution] or listening into the IGP.

The set of functionality and collected information that is the Topology Manager may be embedded as a component of a larger application, such as a path computation application. As a stand-alone application, the Topology Manager could be useful to other network applications by providing a coherent picture of the network state accessible via another interface. That interface might use the same I2RS protocol and could provide a topology service using extensions to the I2RS data models.

6. I2RS Agent Role and Functionality

The I2RS Agent is part of a routing element. As such, it has relationships with that routing element as a whole, and with various components of that routing element.

6.1. Relationship to its Routing Element

A Routing Element may be implemented with a wide variety of different architectures: an integrated router, a split architecture, distributed architecture, etc. The architecture does not need to affect the general I2RS agent behavior.

For scalability and generality, the I2RS agent may be responsible for collecting and delivering large amounts of data from various parts of the routing element. Those parts may or may not actually be part of a single physical device. Thus, for scalability and robustness, it is important that the architecture allow for a distributed set of reporting components providing collected data from the I2RS agent back to the relevant I2RS clients. As currently envisioned, a given I2RS agent would have only one locus per I2RS service for manipulation of routing element state.

6.2. I2RS State Storage

State modification requests are sent to the I2RS agent in a routing element by I2RS clients. The I2RS agent is responsible for applying these changes to the system, subject to the authorization discussed above. The I2RS agent will retain knowledge of the changes it has applied, and the client on whose behalf it applied the changes. The I2RS agent will also store active subscriptions. These sets of data form the I2RS data store. This data is retained by the agent until the state is removed by the client, overridden by some other operation such as CLI, or the device reboots. Meaningful logging of the application and removal of changes is recommended. I2RS applied changes to the routing element state will not be retained across routing element reboot. The I2RS data store is not preserved across routing element reboots; thus the I2RS agent will not attempt to reapply such changes after a reboot.

6.2.1. I2RS Agent Failure

If it is possible for an I2RS Agent to fail independently of the associated routing element, the behavior for any associated ephemeral I2RS state needs to be clearly described. The I2RS state should be preserved until the associated routing element has itself rebooted or until the I2RS state is explicitly torn down. This is desirable since the I2RS Client has no way of learning that an I2RS Agent has unexpectedly failed until that I2RS Agent has restarted; in the interval between failure and recovery, the I2RS Client will be assuming that its ephemeral state remains. If failure of the I2RS agent causes the ephemeral I2RS state to be removed, then this should be indicated via a capability.

There are two different failure types that are possible and each has different behavior.

Unexpected failure: In this case, the I2RS Agent has unexpectedly crashed and thus cannot notify its clients of anything. If an I2RS Agent can crash separately from its associated routing element, then that I2RS Agent must cache each known I2RS Client. When an I2RS Agent starts, it notifies each saved I2RS Client that the I2RS Agent is up and includes an agent-boot-count that indicates how many times the I2RS Agent has restarted since the associated routing element restarted. The agent-boot-count allows an I2RS Client to determine if the I2RS Agent has restarted; if so, the I2RS Client may need to resubscribe to notifications and information streams. The I2RS Agent should also indicate whether the I2RS ephemeral state was preserved in the Routing Element.

Graceful failure: In this case, the I2RS Agent can do specific limited work as part of the process of being disabled. First, the I2RS Agent can optionally notify all its clients that their state is being torn down; if no such notification is sent, then that ephemeral state is not torn down. Second, the I2RS Agent must notify all its cached clients that the agent is going down.

6.2.2. Starting and Ending

When an I2RS client applies changes via the I2RS protocol, those changes are applied and left until removed or the routing element reboots. The network application may make decisions about what to request via I2RS based upon a variety of conditions that imply different start times and stop times. That complexity is managed by the network application and is not handled by I2RS.

6.2.3. Reversion

An I2RS Agent may decide that some state should no longer be applied. An I2RS Client may instruct an Agent to remove state it has applied. In all such cases, the state will revert to what it would have been without the I2RS; that state is generally whatever was specified via the CLI, NETCONF, SNMP, etc. I2RS Agents will not store multiple alternative states, nor try to determine which one among such a plurality it should fall back to. Thus, the model followed is not like the RIB, where multiple routes are stored at different preferences.

An I2RS Client may register for notifications, subject to its notification scope, regarding state modification or removal by a particular I2RS Client.

6.3. Interactions with Local Config

Changes may originate from either Local Config or from I2RS. The modifications and data stored by I2RS are separate from the local device configuration, but conflicts between the two must be resolved in a deterministic manner that respects operator-applied policy. That policy can determine whether Local Config overrides a particular I2RS client's request or vice versa. To achieve this end, either by default Local Config always wins or, optionally, a routing element may permit a priority to be configured on the device for the Local Config mechanism. The policy mechanism in the later case is comparing the I2RS client's priority with that priority assigned to the Local Config.

When the Local Config always wins, some communication between that subsystem and the I2RS Agent is still necessary. That communication contains the details of each specific device configuration change that the I2RS Agent is permitted to modify. In addition, when the system determines, that a client's I2RS state is preempted, the I2RS agent must notify the affected I2RS agents; how the system determines this is implementation-dependent.

It is critical that policy based upon the source is used because the resolution cannot be time-based. Simply allowing the most recent state to prevail could cause race conditions where the final state is not repeatably deterministic.

6.4. Routing Components and Associated I2RS Services

For simplicity, each logical protocol or set of functionality that can be compactly described in a separable information and data model is considered as a separate I2RS Service. A routing element need not implement all routing components described nor provide the associated I2RS services. When a full implementation is not mandatory, an I2RS Service should include a capability model so that implementations can indicate which parts of the service are supported. Each I2RS Service requires an information model that describes at least the following: data that can be read, data that can be written, notifications that can be subscribed to, and the capability model mentioned above.

The initial services included in the I2RS architecture are as follows.

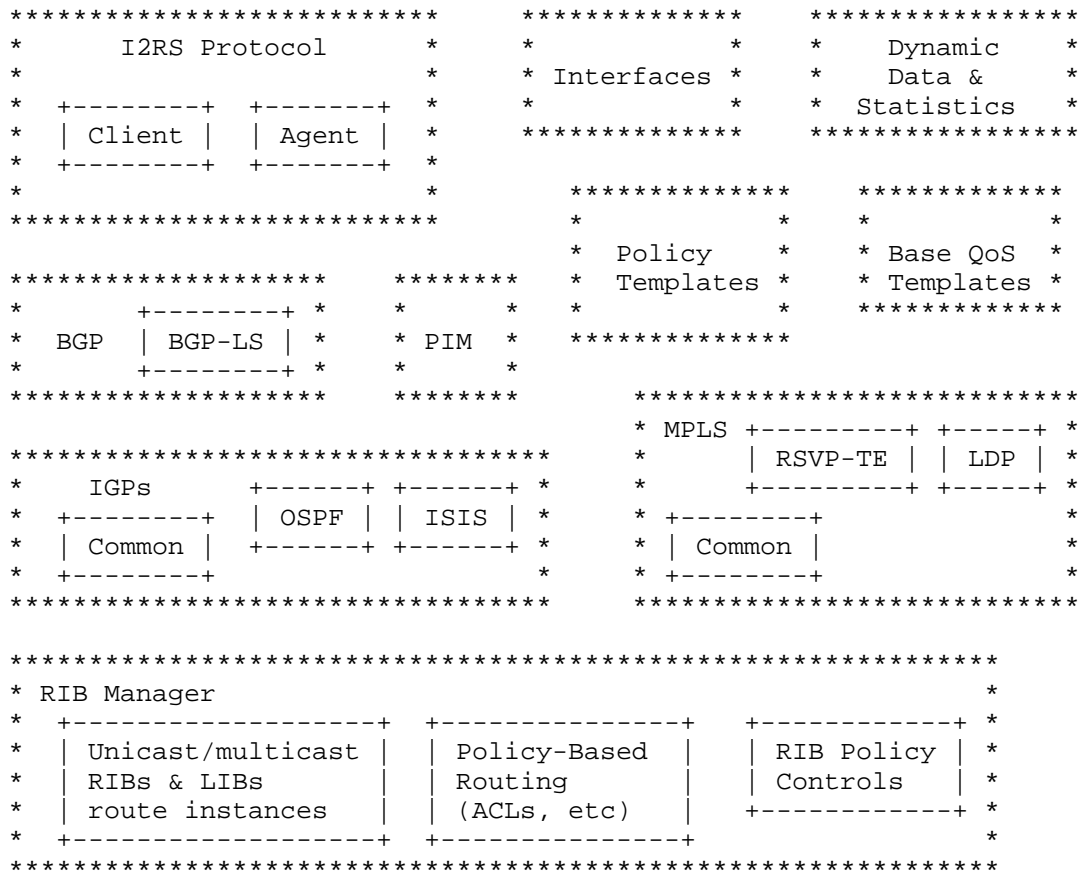


Figure 2: Anticipated I2RS Services

There are relationships between different I2RS Services - whether those be the need for the RIB to refer to specific interfaces, the desire to refer to common complex types (e.g. links, nodes, IP addresses), or the ability to refer to implementation-specific functionality (e.g. pre-defined templates to be applied to interfaces or for QoS behaviors that traffic is direct into). Section 6.4.5 discussing information modeling constructs and the range of relationship types that are applicable.

6.4.1. Routing and Label Information Bases

Routing elements may maintain one or more Information Bases. Examples include Routing Information Bases such as IPv4/IPv6 Unicast or IPv4/IPv6 Multicast. Another such example includes the MPLS Label Information Bases, per-platform- or per-interface." This

functionality, exposed via an I2RS Service, must interact smoothly with the same mechanisms that the routing element already uses to handle RIB input from multiple sources, so as to safely change the system state. Conceptually, this can be handled by having the I2RS Agent communicate with a RIB Manager as a separate routing source.

The point-to-multipoint state added to the RIB does not need to match to well-known multicast protocol installed state. The I2RS Agent can create arbitrary replication state in the RIB, subject to the advertised capabilities of the routing element.

6.4.2. IGPs, BGP and Multicast Protocols

A separate I2RS Service can expose each routing protocol on the device. Such I2RS services may include a number of different kinds of operations:

- o reading the various internal RIB(s) of the routing protocol is often helpful for understanding the state of the network. Directly writing to these protocol-specific RIBs or databases is out of scope for I2RS.
- o reading the various pieces of policy information the particular protocol instance is using to drive its operations.
- o writing policy information such as interface attributes that are specific to the routing protocol or BGP policy that may indirectly manipulate attributes of routes carried in BGP.
- o writing routes or prefixes to be advertised via the protocol.
- o joining/removing interfaces from the multicast trees
- o subscribing to an information stream of route changes
- o receiving notifications about peers coming up or going down

For example, the interaction with OSPF might include modifying the local routing element's link metrics, announcing a locally-attached prefix, or reading some of the OSPF link-state database. However, direct modification of the link-state database MUST NOT be allowed in order to preserve network state consistency.

6.4.3. MPLS

I2RS Services will be needed to expose the protocols that create transport LSPs (e.g. LDP and RSVP-TE) as well as protocols (e.g. BGP, LDP) that provide MPLS-based services (e.g. pseudowires, L3VPNs,

L2VPNs, etc). This should include all local information about LSPs originating in, transiting, or terminating in this Routing Element.

6.4.4. Policy and QoS Mechanisms

Many network elements have separate policy and QoS mechanisms, including knobs which affect local path computation and queue control capabilities. These capabilities vary widely across implementations, and I2RS cannot model the full range of information collection or manipulation of these attributes. A core set does need to be included in the I2RS information models and supported in the expected interfaces between the I2RS Agent and the network element, in order to provide basic capabilities and the hooks for future extensibility.

By taking advantage of extensibility and sub-classing, information models can specify use of a basic model that can be replaced by a more detailed model.

6.4.5. Information Modeling, Device Variation, and Information Relationships

I2RS depends heavily on information models of the relevant aspects of the Routing Elements to be manipulated. These models drive the data models and protocol operations for I2RS. It is important that these informational models deal well with a wide variety of actual implementations of Routing Elements, as seen between different products and different vendors. There are three ways that I2RS information models can address these variations: class or type inheritance, optional features, and templating.

6.4.5.1. Managing Variation: Object Classes/Types and Inheritance

Information modeled by I2RS from a Routing Element can be described in terms of classes or types or object. Different valid inheritance definitions can apply. What is appropriate for I2RS to use is not determined in this architecture; for simplicity, class and subclass will be used as the example terminology. This I2RS architecture does require the ability to address variation in Routing Elements by allowing information models to define parent or base classes and subclasses.

The base or parent class defines the common aspects that all Routing Elements are expected to support. Individual subclasses can represent variations and additional capabilities. When applicable, there may be several levels of refinement. The I2RS protocol can then provide mechanisms to allow an I2RS client to determine which classes a given I2RS Agent has available. Clients which only want basic capabilities can operate purely in terms of base or parent

classes, while a client needing more details or features can work with the supported sub-class(es).

As part of I2RS information modeling, clear rules should be specified for how the parent class and subclass can relate; for example, what changes a subclass can make to its parent? The description of such rules should be done so that it can apply across data modeling tools until the I2RS data modeling language is selected.

6.4.5.1.1. Managing Variation: Optionality

I2RS Information Models must be clear about what aspects are optional. For instance, must an instance of a class always contain a particular data field X? If so, must the client provide a value for X when creating the object or is there a well-defined default value? From the Routing Element perspective, in the above example, is support of X required so that values for X can be accepted and processed? If not, how does the I2RS client determine whether the I2RS agent can accept and apply values for X?

Optional behavior can also be extended to the ranges of values a given piece of information can take, the length of strings, the existence of particular events, and other aspects of information. The information model needs to be clear about what is required of the clients, what is required of agents, and what is permitted to each one.

6.4.5.1.2. Managing Variation: Templating

A template is a collection of information to address a problem; it cuts across the notions of class and object instances. A template provides a set of defined values for a set of information fields and can specify a set of values that must be provided to complete the template. Further, a flexible template scheme may that some of the defined values can be over-written.

For instance, assigning traffic to a particular service class might be done by specifying a template Queueing with a parameter to indicate Gold, Silver, or Best Effort. The details of how that is carried out are not modeled. This does assume that the necessary templates are made available on the Routing Element via some mechanism other than I2RS. The idea is that by providing suitable templates for tasks that need to be accomplished, with templates implemented differently for different kinds of Routing Elements, the client can easily interact with the Routing Element without concern for the variations which are handled by values included in the template.

If implementation variation can be exposed in other ways, templates may not be needed. However, templates themselves could be objects referenced in the protocol messages, with Routing Elements being configured with the proper templates to complete the operation. This is a topic for further discussion.

6.4.5.1.3. Object Relationships

Objects (in a Routing Element or otherwise) do not exist in isolation. They are related to each other. One of the important things a class definition does is represent the relationships between instances of different classes. These relationships can be very simple, or quite complicated. The following lists the information relationships that the information models need to support.
[[Editors' note: All of these are for discussion, and it is expected that the list may be changed during WG discussion.]]

6.4.5.1.3.1. Initialization

The simplest relationship is that one object instances is initialized by copying another. For example, one may have an object instance that represents the default setup for a tunnel, and all new tunnels have fields copied from there if they are not set as part of establishment. This is closely related to the templates discussed above, but not identical. Since the relationship is only momentary it is often not formally represented in modeling, but only captured in the semantic description of the default object.

6.4.5.1.3.2. Correlation Identification

Often, it suffices to indicate in one object that it is related to a second object, without having a strong binding between the two. So an Identifier is used to represent the relationship. This can be used to allow for late binding, or a weak binding that does not even need to exist. A policy name in an object might indicate that if a policy by that name exists, it is to be applied under some circumstance. In modeling this is often represented by the type of the value.

6.4.5.1.3.3. Object References

Sometimes the relationship between objects is stronger. A valid ARP entry has to point to the active interface over which it was derived. This is the classic meaning of an object reference in programming. It can be used for relationships like containment or dependence. This is usually represented by an explicit modeling link.

6.4.5.1.3.4. Active Reference

There is an even stronger form of coupling between objects if changes in one of the two objects are always to be reflected in the state of the other. For example, if a Tunnel has an MTU, and link MTU changes need to immediately propagate to the Tunnel MTU, then the tunnel is actively coupled to the link interface. This kind of active state coupling implies some sort of internal bookkeeping to ensure consistency, often conceptualized as a subscription model across objects.

7. I2RS Client Agent Interface

7.1. One Control and Data Exchange Protocol

This I2RS Architecture presumes that there is one I2RS protocol for control and data exchange. This helps meet the goal of simplicity and thereby enhances deployability. Whether such a protocol is built upon extending existing mechanisms or requires a new mechanism is under active investigation. That protocol may use several underlying transports (TCP, SCTP, DCCP), with suitable authentication and integrity protection mechanisms. These different transports can support different types of communication (e.g. control, reading, notifications, and information collection) and different sets of data. Whatever transport is used for the data exchange, it must also support suitable congestion control mechanisms.

7.2. Communication Channels

Multiple communication channels and multiple types of communication channels are required. There may be a range of requirements (e.g. confidentiality, reliability), and to support the scaling there may need to be channels originating from multiple sub-components of a routing element and/or to multiple parts of an I2RS client. All such communication channels will use the same higher level protocol. Use of additional channels for communication will be coordinated between the I2RS client and the I2RS agent.

7.3. Capability Negotiation

The support for different protocol capabilities and I2RS Services will vary across I2RS Clients and Routing Elements supporting I2RS Agents. Since each I2RS Service is required to include a capability model (see Section 6.4), negotiation at the protocol level can be restricted to protocol specifics and which I2RS Services are supported.

Capability negotiation (such as which transports are supported beyond the minimum required to implement) will clearly be necessary. It is important that such negotiations be kept simple and robust, as such mechanisms are often a source of difficulty in implementation and deployment.

The protocol capability negotiation can be segmented into the basic version negotiation (required to ensure basic communication), and the more complex capability exchange which can take place within the base protocol mechanisms. In particular, the more complex protocol and mechanism negotiation can be addressed by defining information models for both the I2RS Agent and the I2RS Client. These information models can describe the various capability options. This can then represent and be used to communicate important information about the agent, and the capabilities thereof.

7.4. Identity and Security Role

Each I2RS Client will have a unique identity; it can also have secondary identities to be used for troubleshooting. A secondary identity is merely a unique, opaque identifier that may be helpful in troubleshooting. Via authentication and authorization mechanisms based on the primary unique identity, the I2RS Client will have a specific scope for reading data, for writing data, and limitations on the resources that can be consumed. The scopes need to specify both the data and the value ranges.

7.4.1. Client Redundancy

I2RS must support client redundancy. At the simplest, this can be handled by having a primary and a backup network application that both use the same client identity and can successfully authenticate as such. Since I2RS does not require a continuous transport connection and supports multiple transport sessions, this can provide some basic redundancy. However, it does not address concerns for troubleshooting and accountability about knowing which network application is actually active. At a minimum, basic transport information about each connection and time can be logged with the identity.

7.5. Connectivity

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

communication is required, the agent or client can open a new communication channel.

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

For many applications, it may be desirable to clean up state if a network application dies before removing the state it has created. Typically, this is dealt with in terms of network application redundancy. If stronger mechanisms are desired, mechanisms outside of I2RS may allow a supervisory network application to monitor I2RS clients, and based on policy known to the supervisor clean up state if applications die. More complex mechanism instantiated in the I2RS agent would add complications to the I2RS protocol and are thus left for future work.

Some examples of such a mechanism include the following. In one option, the client could request state clean-up if a particular transport session is terminated. The second is to allow state expiration, expressed as a policy associated with the I2RS client's role. The state expiration could occur after there has been no successful communication channel to or from the I2RS client for the policy-specified duration.

7.6. Notifications

As with any policy system interacting with the network, the I2RS Client needs to be able to receive notifications of changes in network state. Notifications here refers to changes which are unanticipated, represent events outside the control of the systems (such as interface failures on controlled devices), or are sufficiently sparse as to be anomalous in some fashion. A notification may also be due to a regular event.

Such events may be of interest to multiple I2RS Clients controlling data handled by an I2RS Agent, and to multiple other I2RS clients which are collecting information without exerting control. The architecture therefore requires that it be practical for I2RS Clients to register for a range of notifications, and for the I2RS Agents to send notifications to a number of Clients. The I2RS Client should be able to filter the specific notifications that will be received; the specific types of events and filtering operations can vary by information model and need to be specified as part of the information model.

The I2RS information model needs to include representation of these events. As discussed earlier, the capability information in the model will allow I2RS clients to understand which events a given I2RS Agent is capable of generating.

For performance and scaling by the I2RS client and general information privacy, an I2RS Client needs to be able to register for just the events it is interested in. It is also possible that I2RS might provide a stream of notifications via a publish/subscribe mechanism that is not amenable to having the I2RS agent do the filtering.

7.7. Information collection

One of the other important aspects of the I2RS is that it is intended to simplify collecting information about the state of network elements. This includes both getting a snapshot of a large amount of data about the current state of the network element, and subscribing to a feed of the ongoing changes to the set of data or a subset thereof. This is considered architecturally separate from notifications due to the differences in information rate and total volume.

7.8. Multi-Headed Control

As was described earlier, an I2RS Agent interacts with multiple I2RS Clients who are actively controlling the network element. From an architecture and design perspective, the assumption is that by means outside of this system the data to be manipulated within the network element is appropriately partitioned so that any given piece of information is only being manipulated by a single I2RS Client.

Nonetheless, unexpected interactions happen and two (or more) I2RS clients may attempt to manipulate the same piece of data. This is considered an error case. This architecture does not attempt to determine what the right state of data should be when such a collision happens. Rather, the architecture mandates that there be decidable means by which I2RS Agents handle the collisions. The mechanism for this is to have a simple priority associated with each I2RS clients, and the highest priority change remains in effect. In the case of priority ties, the first client whose attribution is associated with the data will keep control.

In order for this approach to multi-headed control to be useful for I2RS Clients, it is important that it be possible for an I2RS Client to register for changes to any changes made by I2RS to data that it may care about. This is included in the I2RS event mechanisms. This also needs to apply to changes made by CLI/NETCONF/SNMP within the

write-scope of the I2RS Agent, as the same priority mechanism (even if it is "CLI always wins") applies there. The I2RS client may then respond to the situation as it sees fit.

7.9. Transactions

In the interest of simplicity, the I2RS architecture does not include multi-message atomicity and rollback mechanisms. Rather, it includes a small range of error handling for a set of operations included in a single message. An I2RS Client may indicate one of the following three error handling for a given message with multiple operations which it sends to an I2RS Agent:

Perform all or none: This traditional SNMP semantic indicates that other I2RS agent will keep enough state when handling a single message to roll back the operations within that message. Either all the operations will succeed, or none of them will be applied and an error message will report the single failure which caused them not to be applied. This is useful when there are, for example, mutual dependencies across operations in the message.

Perform until error: In this case, the operations in the message are applied in the specified order. When an error occurs, no further operations are applied, and an error is returned indicating the failure. This is useful if there are dependencies among the operations and they can be topologically sorted.

Perform all storing errors: In this case, the I2RS Agent will attempt to perform all the operations in the message, and will return error indications for each one that fails. This is useful when there is no dependency across the operation, or where the client would prefer to sort out the effect of errors on its own.

In the interest of robustness and clarity of protocol state, the protocol will include an explicit reply to modification or write operations even when they fully succeed.

8. Manageability Considerations

Manageability plays a key aspect in I2RS. Some initial examples include:

Resource Limitations: Using I2RS, 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 important.

Configuration Interactions: The interaction of state installed via the I2RS and via a router's configuration needs to be clearly defined. As described in this architecture, a simple priority that is configured is used to provide sufficient policy flexibility.

9. IANA Considerations

This document includes no request to IANA.

10. Acknowledgements

Significant portions of this draft came from draft-ward-i2rs-framework-00 and draft-atlas-i2rs-policy-framework-00.

The authors would like to thank Nitin Bahadur, Shane Amante, Ed Crabbe, Ken Gray, Carlos Pignataro, Wes George, Ron Bonica, Joe Clarke, Juergen Schoenwalder, Jamal Hadi Salim, Scott Brim, and Thomas Narten for their suggestions and review.

11. Informative References

- [I-D.ietf-i2rs-problem-statement]
Atlas, A., Nadeau, T., and D. Ward, "Interface to the Routing System Problem Statement", draft-ietf-i2rs-problem-statement-00 (work in progress), August 2013.
- [I-D.ietf-idr-ls-distribution]
Gredler, H., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and TE Information using BGP", draft-ietf-idr-ls-distribution-04 (work in progress), November 2013.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

Authors' Addresses

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

Email: akatlas@juniper.net

Joel Halpern
Ericsson

Email: Joel.Halpern@ericsson.com

Susan Hares
Hickory Hill Consulting

Email: shares@ndzh.com

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

Email: wardd@cisco.com

Thomas D. Nadeau
Brocade

Email: tnadeau@lucidvision.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: February 17, 2014

A. Atlas, Ed.
T. Nadeau, Ed.
Juniper Networks
D. Ward
Cisco Systems
August 16, 2013

Interface to the Routing System Problem Statement
draft-ietf-i2rs-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 network applications to have access to and control over information in the Internet's routing system, we need a publicly documented interface specification. The interface needs to support real-time, asynchronous interactions using data models and encodings that are efficient and potentially different from those available today. Furthermore, the interface must be tailored to support a variety of use cases.

This document expands upon these statements of requirements to provide a detailed problem statement for an Interface to the Routing System (I2RS).

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 February 17, 2014.

Copyright Notice

Copyright (c) 2013 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	2
2. I2RS 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 I2RS	6
6. Acknowledgements	8
7. IANA Considerations	8
8. Security Considerations	8
9. Informative References	8
Appendix A. Existing Management Interfaces	8
Authors' Addresses	9

1. Introduction

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

With complexity comes the need for more sophisticated automated network 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 network applications, their computations 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 correct places.

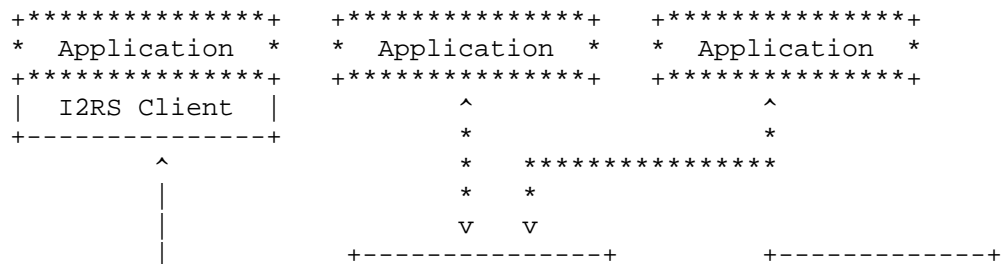
In the past, mechanisms to support the requirements outlined above have been developed piecemeal as proprietary solutions to specific

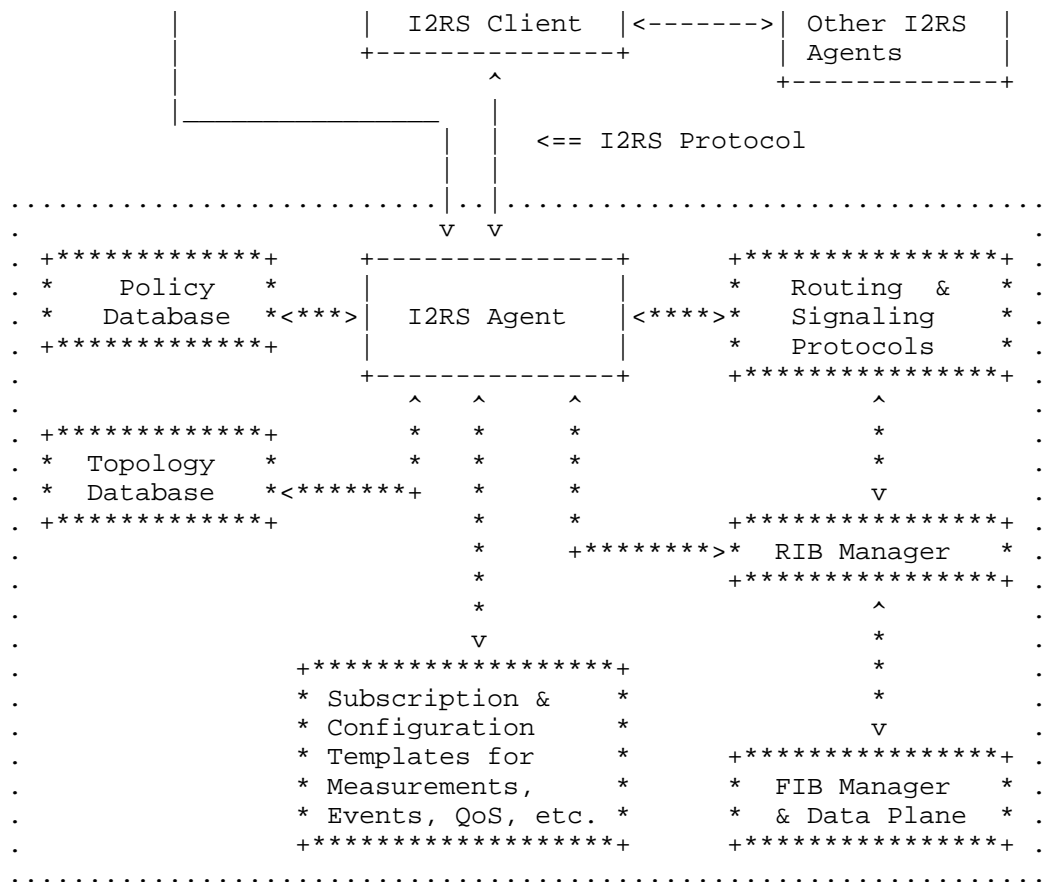
situations and needs. Many routing elements have an external interface to interact with routing - but since these vary between vendors, it is difficult to integrate use of those interfaces into a network. The existence of such proprietary interfaces demonstrates both that the need for such an interface is understood and that technology solutions are understood. What is needed are technological solutions with clearly defined operations that an application can initiate, and data-models to support such actions. These would facilitate wide-scale deployment of interoperable applications and routing systems. These solutions must be designed to facilitate rapid, isolated, secure, and dynamic changes to a device's routing system. In order to address these needs, the creation of an Interface to the Routing System (I2RS) is needed.

It should be noted that during the course of this document, we will discuss and use the term "applications". This is meant to refer to an executable program of some sort that has access to a network, such as an IP network.

2. I2RS Model and Problem Area for The IETF

Managing a network of production devices running a variety of routing protocols involves interactions between multiple components within a device. Some of these components are virtual while some are physical; it may be desirable for many, or even all of these components to 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 require standardization, as others have already been standardized. The I2RS model is intended to incorporate existing mechanisms where appropriate, and to build extensions and new protocols where needed. The I2RS model and problem area for IETF work is illustrated in Figure 1. The I2RS Agent is associated with a routing element, which may or may not be co-located with a data-plane. The I2RS Client is used and controlled by one or more network applications; they may be co-located or the I2RS Client might be part of a separate application, such as an orchestrator or controller.





<--> interfaces inside the scope of I2RS
 +--+ objects inside the scope of I2RS

<*> interfaces NOT within the scope of I2RS
 +**+ objects NOT within the scope of I2RS

.... boundary of a router participating in the I2RS

Figure 1: I2RS model and Problem Area

A critical aspect of I2RS is defining a suitable protocol or protocols to carry messages between the I2RS Clients and the I2RS Agent, and defining the data-models for use with those I2RS protocol(s). The data models should translate into 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 information should use existing transport protocols to provide the reliability, security, and timeliness appropriate for the particular data.

The second critical aspect are semantic-aware data-models for information in the routing system and in a topology database. The data-model should describe the meaning and relationships of the modeled items. The data-models should be separable across different features of the managed components, versioned, and extendable. An application should be able to combine data from individual routing elements to provide network-wide data-model(s).

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, to usefully model next-hops, the data model employed needs to handle next-hop indirection and recursion (e.g. a prefix X is routed like prefix Y) 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 I2RS focus initially on interfaces to the RIB layer (e.g. RIB, LIB, 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]). I2RS posits 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. I2RS provides a framework so that applications can register for asynchronous notifications and can 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 provide 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 potentially available (e.g. administratively down) or unknown (e.g. to peers or customers) 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 standardized ability to set up the router to trigger different actions upon an event's occurrence so that a rapid reaction can be accomplished.

5. Desired Aspects of a Protocol for I2RS

This section describes required aspects of a protocol that could support I2RS. 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 via I2RS without being required to wait for each to complete before sending the next.

Very Fine Granularity of Data Locking for Writing: When an I2RS operation is processed, it is required that the data locked for writing is very granular (e.g. a particular prefix and route) rather than extremely coarse, as is done for writing configuration. This should improve the number of concurrent I2RS operations that are feasible and reduce blocking delays.

Multi-Headed Control: Multiple applications may communicate to the same I2RS agent in a minimally coordinated fashion. It is necessary that the I2RS agent can handle multiple requests in a well-known policy-based fashion. Data written can be owned by different I2RS clients.

Duplex: Communications can be established by either the I2RS client (i.e.: that resides within the application or is used by it to communicate with the I2RS agent), or the I2RS agent. Similarly, events, acknowledgements, failures, operations, etc. can be sent at any time by both the router and the application. The I2RS is not a pure pull-model where only the application queries to pull responses.

High-Throughput: At a minimum, the I2RS Agent and associated router should be able to handle a considerable number of operations per second above what basic Netconf or a proprietary CLI can.

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.

Scalable, Filterable Information Access: 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.

Secure Control: Any ability to manipulate routing state must be subject to authentication and authorization. Such communications must also have its integrity protected.

Extensible and Interoperability: Both the I2RS protocol and models must be extensible and interoperate between different versions of protocols and models.

6. Acknowledgements

The authors would like to thank Ken Gray, Ed Crabbe, Nic Leymann, Carlos Pignataro, and Kwang-koog Lee for their suggestions and review.

7. IANA Considerations

This document includes no request to IANA.

8. 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.

9. 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. Existing Management Interfaces

This section discusses as a single entity the combination of the abstract data models, their representation in a data language, and the transfer protocol commonly used with them. While other combinations of these existing standard technologies are possible, the ways 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. I2RS 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 I2RS.

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. Naturally, I2RS may help define needed information and data models. Additional extensions to handle multi-headed control may need to be added to NetConf and/or appropriate data models.

Authors' Addresses

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

Email: akatlas@juniper.net

Thomas D. Nadeau (editor)
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
Expires: August 18, 2014

N. Bahadur, Ed.
Bracket Computing
R. Folkes, Ed.
Juniper Networks, Inc.
S. Kini, Ed.
Ericsson
J. Medved
Cisco
February 14, 2014

Routing Information Base Info Model
draft-ietf-i2rs-rib-info-model-02

Abstract

Routing and routing functions in enterprise and carrier networks are typically performed by network devices (routers and switches) using a routing information base (RIB). Protocols and configuration push data into the RIB and the RIB manager installs state into the hardware; for packet forwarding. This draft specifies an information model for the RIB to enable defining a standardized data model. Such a data model can be used to define an interface to the RIB from an entity that may even be external to the network device. This interface can be used to support new use-cases being defined by the IETF I2RS WG.

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 August 18, 2014.

Copyright Notice

Copyright (c) 2014 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. Conventions used in this document	5
2. RIB data	5
2.1. RIB definition	6
2.2. Routing instance	6
2.3. Route	7
2.4. Nexthop	9
2.4.1. Nexthop types	11
2.4.2. Nexthop list attributes	12
2.4.3. Nexthop content	13
2.4.4. Special nexthops	14
3. Reading from the RIB	14
4. Writing to the RIB	14
5. Notifications	15
6. RIB grammar	15
7. Using the RIB grammar	18
7.1. Using route preference	18
7.2. Using different nexthops types	19
7.2.1. Tunnel nexthops	19
7.2.2. Replication lists	19
7.2.3. Weighted lists	19
7.2.4. Protection lists	20
7.2.5. Nexthop chains	20
7.2.6. Lists of lists	21
7.3. Performing multicast	21
8. RIB operations at scale	22
8.1. RIB reads	22
8.2. RIB writes	22
8.3. RIB events and notifications	22
9. Security Considerations	22
10. IANA Considerations	23

11. Acknowledgements	23
12. References	23
12.1. Normative References	23
12.2. Informative References	23
Authors' Addresses	24

1. Introduction

Routing and routing functions in enterprise and carrier networks are traditionally performed in network devices. Traditionally routers run routing protocols and the routing protocols (along with static config) populate the Routing information base (RIB) of the router. The RIB is managed by the RIB manager and the RIB manager provides a north-bound interface to its clients i.e. the routing protocols to insert routes into the RIB. The RIB manager consults the RIB and decides how to program the forwarding information base (FIB) of the hardware by interfacing with the FIB manager. The relationship between these entities is shown in Figure 1.

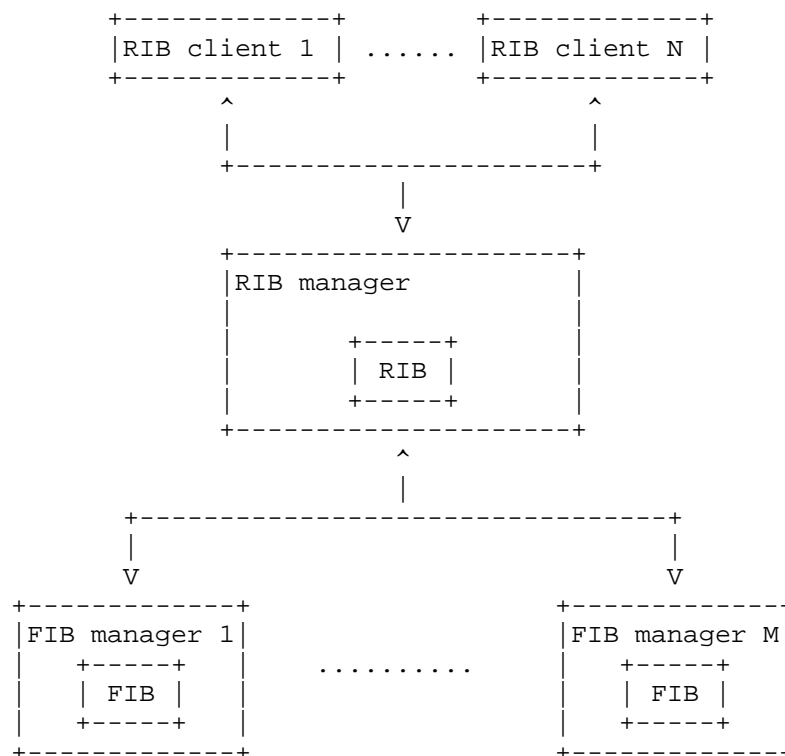


Figure 1: RIB manager, RIB clients and FIB managers

Routing protocols are inherently distributed in nature and each router makes an independent decision based on the routing data received from its peers. With the advent of newer deployment paradigms and the need for specialized applications, there is an emerging need to guide the router's routing function [I-D.ietf-i2rs-problem-statement]. Traditional network-device protocol-based RIB population suffices for most use cases where distributed network control is used. However there are use cases which the network operators currently address by configuring static routes, policies and RIB import/export rules on the routers. There is also a growing list of use cases [I-D.white-i2rs-use-case], [I-D.hares-i2rs-use-case-vn-vc] in which a network operator might want to program the RIB based on data unrelated to just routing (within that network's domain). Programming the RIB could be based on other information such as routing data in the adjacent domain or the load on storage and compute in the given domain. Or it could simply be a programmatic way of creating on-demand dynamic overlays (e.g. GRE tunnels) between compute hosts (without requiring the hosts to run traditional routing protocols). If there was a standardized publicly documented programmatic interface to a RIB, it would enable further networking applications that address a variety of use-cases [I-D.ietf-i2rs-problem-statement].

A programmatic interface to the RIB involves 2 types of operations - reading from the RIB and writing (adding/modifying/deleting) to the RIB. [I-D.white-i2rs-use-case] lists various use-cases which require read and/or write manipulation of the RIB.

In order to understand what is in a router's RIB, methods like per-protocol SNMP MIBs and show output screen scraping are used. These methods are not scalable, since they are client pull mechanisms and not proactive push (from the router) mechanisms. Screen scraping is error prone (since the output format can change) and is vendor dependent. Building a RIB from per-protocol MIBs is error prone since the MIB data represent protocol data and not the exact information that went into the RIB. Thus, just getting read-only RIB information from a router is a hard task.

Adding content to the RIB from an external entity can be done today using static configuration mechanisms provided by router vendors. However the mix of what can be modified in the RIB varies from vendor to vendor and the method of configuring it is also vendor dependent. This makes it hard for an external entity to program a multi-vendor network in a consistent and vendor-independent way.

The purpose of this draft is to specify an information model for the RIB. Using the information model, one can build a detailed data

model for the RIB. That data model could then be used by an external entity to program a network device.

The rest of this document is organized as follows. Section 2 goes into the details of what constitutes and can be programmed in a RIB. Guidelines for reading and writing the RIB are provided in Section 3 and Section 4 respectively. Section 5 provides a high-level view of the events and notifications going from a network device to an external entity, to update the external entity on asynchronous events. The RIB grammar is specified in Section 6. Examples of using the RIB grammar are shown in Section 7. Section 8 covers considerations for performing RIB operations at scale.

1.1. Conventions used in this document

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 [RFC2119].

2. RIB data

This section describes the details of a RIB. It makes forward references to objects in the RIB grammar (Section 6). A high-level description of the RIB contents is as shown below.

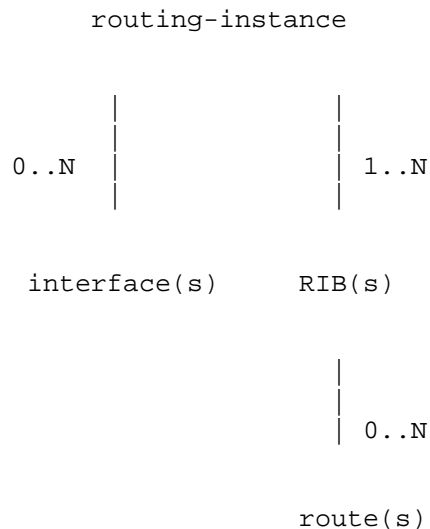


Figure 2: RIB model

2.1. RIB definition

A RIB is an entity that contains routes. A RIB is identified by its name and a RIB is contained within a routing instance (Section 2.2). The name **MUST** be unique within a routing instance. All routes in a given RIB **MUST** be of the same type (e.g. IPv4). Each RIB **MUST** belong to a routing instance.

A routing instance can have multiple RIBs. A routing instance can even have two or more RIBs with the same type of routes (e.g. IPv6). A typical case where this can be used is for multi-topology routing ([RFC4915], [RFC5120]).

Each RIB can be optionally associated with a `ENABLE_IP_RPF_CHECK` attribute that enables Reverse path forwarding (RPF) checks on all IP routes in that RIB. Reverse path forwarding (RPF) check is used to prevent spoofing and limit malicious traffic. For IP packets, the IP source address is looked up and the rpf interface(s) associated with the route for that IP source address is found. If the incoming IP packet's interface matches one of the rpf interface(s), then the IP packet is forwarded based on its IP destination address; otherwise, the IP packet is discarded.

2.2. Routing instance

A routing instance, in the context of the RIB information model, is a collection of RIBs, interfaces, and routing parameters. A routing instance creates a logical slice of the router and allows different logical slices; across a set of routers; to communicate with each other. Layer 3 Virtual Private Networks (VPN), Layer 2 VPNs (L2VPN) and Virtual Private Lan Service (VPLS) can be modeled as routing instances. Note that modeling a Layer 2 VPN using a routing instance only models the Layer-3 (RIB) aspect and does not model any layer-2 information (like ARP) that might be associated with the L2VPN.

The set of interfaces indicates which interfaces are associated with this routing instance. The RIBs specify how incoming traffic is to be forwarded. And the routing parameters control the information in the RIBs. The intersection set of interfaces of 2 routing instances **MUST** be the null set. In other words, an interface **MUST NOT** be present in 2 routing instances. Thus a routing instance describes the routing information and parameters across a set of interfaces.

A routing instance **MUST** contain the following mandatory fields.

- o `INSTANCE_NAME`: A routing instance is identified by its name, `INSTANCE_NAME`. This **MUST** be unique across all routing instances in a given network device.

- o rib-list: This is the list of RIBs associated with this routing instance. Each routing instance can have multiple RIBs to represent routes of different types. For example, one would put IPv4 routes in one RIB and MPLS routes in another RIB.

A routing instance MAY contain the following optional fields.

- o interface-list: This represents the list of interfaces associated with this routing instance. The interface list helps constrain the boundaries of packet forwarding. Packets coming on these interfaces are directly associated with the given routing instance. The interface list contains a list of identifiers, with each identifier uniquely identifying an interface.
- o ROUTER_ID: The router-id field identifies the network device in control plane interactions with other network devices. This field is to be used if one wants to virtualize a physical router into multiple virtual routers. Each virtual router MUST have a unique router-id. ROUTER_ID MUST be unique across all network devices in a given domain.

2.3. Route

A route is essentially a match condition and an action following the match. The match condition specifies the kind of route (IPv4, MPLS, etc.) and the set of fields to match on. Figure 3 represents the overall contents of a route.

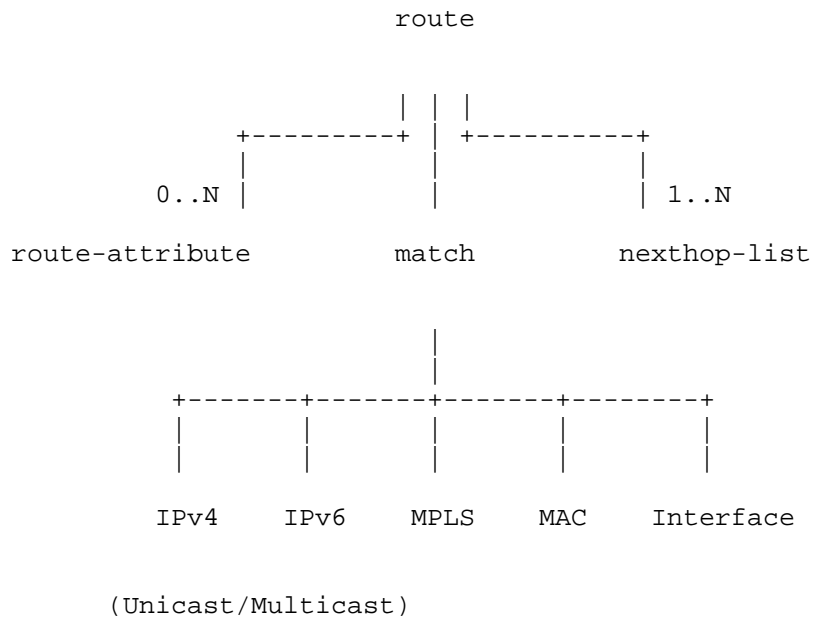


Figure 3: Route model

This document specifies the following match types:

- o IPv4: Match on destination IP address in the IPv4 header
- o IPv6: Match on destination IP address in the IPv6 header
- o MPLS: Match on a MPLS label at the top of the MPLS label stack
- o MAC: Match on MAC destination addresses in the ethernet header
- o Interface: Match on incoming interface of the packet
- o IP multicast: Match on (S, G) or (*, G), where S and G are IP prefixes

Each route MUST have associated with it the following mandatory route attributes.

- o ROUTE_PREFERENCE: This is a numerical value that allows for comparing routes from different protocols. Static configuration is also considered a protocol for the purpose of this field. It is also known as administrative-distance. The lower the value,

the higher the preference. For example there can be an OSPF route for 192.0.2.1/32 with a preference of 5. If a controller programs a route for 192.0.2.1/32 with a preference of 2, then the controller's route will be preferred by the RIB manager. Preference should be used to dictate behavior. For more examples of preference, see Section 7.1.

Each route can have associated with it one or more optional route attributes.

- o route-vendor-attributes: Vendors can specify vendor-specific attributes using this. The details of this attribute is outside the scope of this document.

2.4. Nexthop

A nexthop represents an object resulting from a route lookup. For example, if a route lookup results in sending the packet out a given interface, then the nexthop represents that interface.

Nexthops can be fully resolved nexthops or unresolved nexthop. A resolved nexthop has adequate information to send the outgoing packet to the destination by forwarding it on an interface to a directly connected neighbor. For example, a nexthop to a point-to-point interface or a nexthop to an IP address on an Ethernet interface has the nexthop resolved. An unresolved nexthop is something that requires the RIB manager to determine the final resolved nexthop. For example, a nexthop could be an IP address. The RIB manager would resolve how to reach that IP address, e.g. is the IP address reachable by regular IP forwarding or by a MPLS tunnel or by both. If the RIB manager cannot resolve the nexthop, then the nexthop remains in an unresolved state and is NOT a candidate for installation in the FIB. Future RIB events can cause an unresolved nexthop to get resolved (like that IP address being advertised by an IGP neighbor). Conversely resolved nexthops can also become unresolved (e.g. in case of a tunnel going down) and hence would no longer be candidates to be installed in the FIB.

When at least one of a route's nexthops is resolved, then the route can be used to forward packets. Such a route is considered eligible to be installed in the FIB and is henceforth referred to as a FIB-eligible route. Conversely, when all the nexthops of a route are unresolved that route can no longer be used to forward packets. Such a route is considered ineligible to be installed in the FIB and is henceforth referred to as a FIB-ineligible route. The RIB information model allows an external entity to program routes whose nexthops may be unresolved initially. Whenever an unresolved nexthop

gets resolved, the RIB manager will send a notification of the same (see Section 5).

The overall structure and usage of a nexthop is as shown in the figure below.

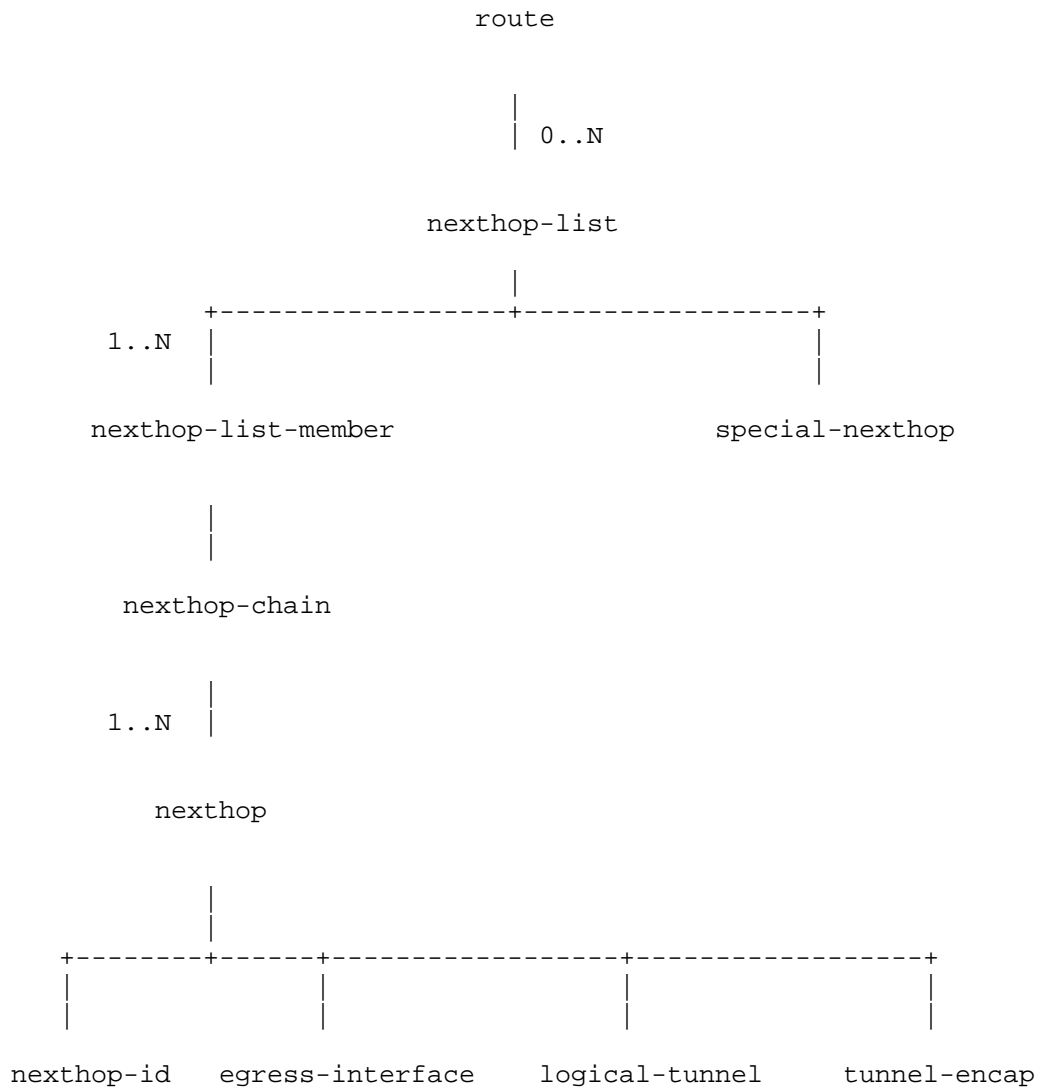


Figure 4: Nexthop model

Nexthops can be identified by an identifier to create a level of indirection. The identifier is set by the RIB manager and returned

to the external entity on request. The RIB data-model SHOULD support a way to optionally receive a nexthop identifier for a given nexthop. For example, one can create a nexthop that points to a BGP peer. The returned nexthop identifier can then be used for programming routes to point to the same nexthop. Given that the RIB manager has created an indirection for that BGP peer using the nexthop identifier, if the transport path to the BGP peer changes, that change in path will be seamless to the external entity and all routes that point to that BGP peer will automatically start going over the new transport path. Nexthop indirection using identifiers could be applied to not just unicast nexthops, but even to nexthops that contain chains and nested nexthops (Section 2.4.1).

2.4.1. Nexthop types

This document specifies a very generic, extensible and recursive grammar for nexthops. Nexthops can be

- o Unicast nexthops - pointing to an interface
- o Tunnel nexthops - pointing to a tunnel
- o Replication lists - list of nexthops to which to replicate a packet
- o Weighted lists - for load-balancing
- o Protection lists - for primary/backup paths
- o Nexthop chains - for chaining headers, e.g. MPLS label over a GRE header
- o Lists of lists - recursive application of the above
- o Indirect nexthops - pointing to a nexthop identifier
- o Special nexthops - for performing specific well-defined functions

It is expected that all network devices will have a limit on how many levels of lookup can be performed and not all hardware will be able to support all kinds of nexthops. RIB capability negotiation becomes very important for this reason and a RIB data-model MUST specify a way for an external entity to learn about the network device's capabilities. Examples of when and how to use various kinds of nexthops are shown in Section 7.2.

Tunnel nexthops allow an external entity to program static tunnel headers. There can be cases where the remote tunnel end-point does

not support dynamic signaling (e.g. no LDP support on a host) and in those cases the external entity might want to program the tunnel header on both ends of the tunnel. The tunnel nexthop is kept generic with specifications provided for some commonly used tunnels. It is expected that the data-model will model these tunnel types with complete accuracy.

Nexthop chains can be used to specify multiple headers over a packet, before a packet is forwarded. One simple example is that of MPLS over GRE, wherein the packet has an inner MPLS header followed by a GRE header followed by an IP header. The outermost IP header is decided by the network device whereas the MPLS header and GRE header are specified by the controller. Not every network device will be able to support all kinds of nexthop chains and an arbitrary number of header chained together. The RIB data-model SHOULD provide a way to expose nexthop chaining capability supported by a given network device.

2.4.2. Nexthop list attributes

For nexthops that are of the form of a list(s), attributes can be associated with each member of the list to indicate the role of an individual member of the list. Two kinds of attributes are specified:

- o **PROTECTION_PREFERENCE:** This provides a primary/backup like preference. The preference is an integer value that should be set to 1 (primary) or 2 (backup). Only when all the primary nexthops fail is the traffic re-routed through the backup nexthops. This attribute must be specified for all the members of a list or none of them.
- o **LOAD_BALANCE_WEIGHT:** This is used for load-balancing. Each list member MUST be assigned a weight between 1 and 99. The weight determines the proportion of traffic to be sent over a nexthop used for forwarding as a ratio of the weight of this nexthop divided by the weights of all the nexthops of this route that are used for forwarding. To perform equal load-balancing, one MAY specify a weight of "0" for all the member nexthops. The value "0" is reserved for equal load-balancing and if applied, MUST be applied to all member nexthops.

A nexthop list MAY contain elements that have both **PROTECTION_PREFERENCE** and **LOAD_BALANCE_WEIGHT** set. When both are set, it means under normal operation the network device should load balance the traffic over all FIB-eligible nexthops of the current protection preference.

2.4.3. Nexthop content

At the lowest level, a nexthop can be one of:

- o identifier: This is an identifier returned by the network device representing another nexthop or another nexthop chain.
- o EGRESS_INTERFACE: This represents a physical, logical or virtual interface on the network device. Address resolution must not be required on this interface. This interface may belong to any routing instance.
- o IP address: A route lookup on this IP address is done to determine the egress interface. Address resolution may be required depending on the interface.
 - * An optional RIB name can also be specified to indicate the RIB in which the IP address is to be looked up. One can use the RIB name field to direct the packet from one domain into another domain. By default the RIB will be the same as the one that route belongs to.
- o EGRESS_INTERFACE and IP address: This can be used in cases e.g. where the IP address is a link-local address.
- o EGRESS_INTERFACE and MAC address: The egress interface must be an ethernet interface. Address resolution is not required for this nexthop.
- o tunnel encap: This can be an encap representing an IP tunnel or MPLS tunnel or others as defined in this document. An optional egress interface can be specified to indicate which interface to send the packet out on. The egress interface is useful when the network device contains Ethernet interfaces and one needs to perform address resolution for the IP packet.
- o logical tunnel: This can be a MPLS LSP or a GRE tunnel (or others as defined in this document), that is represented by a unique identifier (E.g. name).
- o RIB_NAME: A nexthop pointing to a RIB indicates that the route lookup needs to continue in the specified RIB. This is a way to perform chained lookups.

2.4.4. Special nexthops

This document specifies certain special nexthops. The purpose of each of them is explained below:

- o DISCARD: This indicates that the network device should drop the packet and increment a drop counter.
- o DISCARD_WITH_ERROR: This indicates that the network device should drop the packet, increment a drop counter and send back an appropriate error message (like ICMP error).
- o RECEIVE: This indicates that that the traffic is destined for the network device. For example, protocol packets or OAM packets. All locally destined traffic SHOULD be throttled to avoid a denial of service attack on the router's control plane. An optional rate-limiter can be specified to indicate how to throttle traffic destined for the control plane. The description of the rate-limiter is outside the scope of this document.

3. Reading from the RIB

A RIB data-model MUST allow an external entity to read entries, for RIBs created by that entity. The network device administrator MAY allow reading of other RIBs by an external entity through access lists on the network device. The details of access lists are outside the scope of this document.

The data-model MUST support a full read of the RIB and subsequent incremental reads of changes to the RIB. An external agent SHOULD be able to request a full read at any time in the lifecycle of the connection. When sending data to an external entity, the RIB manager SHOULD try to send all dependencies of an object prior to sending that object.

4. Writing to the RIB

A RIB data-model MUST allow an external entity to write entries, for RIBs created by that entity. The network device administrator MAY allow writes to other RIBs by an external entity through access lists on the network device. The details of access lists are outside the scope of this document.

When writing an object to a RIB, the external entity SHOULD try to write all dependencies of the object prior to sending that object. The data-model MUST support requesting identifiers for nexthops and collecting the identifiers back in the response.

Route programming in the RIB MUST result in a return code that contains the following attributes:

- o Installed - Yes/No (Indicates whether the route got installed in the FIB)
- o Active - Yes/No (Indicates whether a route is fully resolved and is a candidate for selection)
- o Reason - E.g. Not authorized

The data-model MUST specify which objects are modify-able objects. A modify-able object is one whose contents can be changed without having to change objects that depend on it and without affecting any data forwarding. To change a non-modifiable object, one will need to create a new object and delete the old one. For example, routes that use a nexthop that is identified by a nexthop-identifier should be unaffected when the contents of that nexthop changes.

5. Notifications

Asynchronous notifications are sent by the network device's RIB manager to an external entity when some event occurs on the network device. A RIB data-model MUST support sending asynchronous notifications. A brief list of suggested notifications is as below:

- o Route change notification, with return code as specified in Section 4
- o Nexthop resolution status (resolved/unresolved) notification

6. RIB grammar

This section specifies the RIB information model in Routing Backus-Naur Form [RFC5511].

```
<routing-instance> ::= <INSTANCE_NAME>
                        [<interface-list>] <rib-list>
                        [<ROUTER_ID>]
```

```
<interface-list> ::= (<INTERFACE_IDENTIFIER> ...)
```

```
<rib-list> ::= (<rib> ...)
<rib> ::= <RIB_NAME> <rib-family>
          [<route> ... ]
          [ENABLE_IP_RPF_CHECK]
```

```

<rib-family> ::= <IPV4_RIB_FAMILY> | <IPV6_RIB_FAMILY> |
                  <MPLS_RIB_FAMILY> | <IEEE_MAC_RIB_FAMILY>

<route> ::= <match> <nexthop-list>
            [<route-attributes>]
            [<route-vendor-attributes>]

<match> ::= <ipv4-route> | <ipv6-route> | <mpls-route> |
            <mac-route> | <interface-route>

<ipv4-route> ::= <destination-ipv4-address> | <source-ipv4-address> |
                  (<destination-ipv4-address> <source-ipv4-address>)
<destination-ipv4-address> ::= <ipv4-prefix>
<source-ipv4-address> ::= <ipv4-prefix>
<ipv4-prefix> ::= <IPV4_ADDRESS> <IPV4_PREFIX_LENGTH>

<ipv6-route> ::= <destination-ipv6-address> | <source-ipv6-address> |
                  (<destination-ipv6-address> <source-ipv6-address>)
<destination-ipv6-address> ::= <ipv6-prefix>
<source-ipv6-address> ::= <ipv6-prefix>
<ipv6-prefix> ::= <IPV6_ADDRESS> <IPV6_PREFIX_LENGTH>

<mpls-route> ::= <MPLS> <MPLS_LABEL>
<mac-route> ::= <IEEE_MAC> ( <MAC_ADDRESS> )
<interface-route> ::= <INTERFACE> <INTERFACE_IDENTIFIER>

<multicast-source-ipv4-address> ::= <IPV4_ADDRESS>
                                     <IPV4_PREFIX_LENGTH>
<multicast-source-ipv6-address> ::= <IPV6_ADDRESS>
                                     <IPV6_PREFIX_LENGTH>

<route-attributes> ::= <ROUTE_PREFERENCE> [<LOCAL_ONLY>]
                       [<address-family-route-attributes>]

<address-family-route-attributes> ::= <ip-route-attributes> |
                                       <mpls-route-attributes> |
                                       <ethernet-route-attributes>

<ip-route-attributes> ::= <>
<mpls-route-attributes> ::= <>

```

```

<ethernet-route-attributes> ::= <>
<route-vendor-attributes> ::= <>

<nexthop-list> ::= <special-nexthop> |
                    ((<nexthop-list-member>) |
                     ([<nexthop-list-member> ... ] <nexthop-list> ))

<nexthop-list-member> ::= (<nexthop-chain> |
                           <nexthop-chain-identifier> )
                           [<nexthop-list-member-attributes>]
<nexthop-list-member-attributes> ::= [<PROTECTION_PREFERENCE>]
                                     [<LOAD_BALANCE_WEIGHT>]

<nexthop-chain> ::= (<nexthop> ...)
<nexthop-chain-identifier> ::= <NEXTHOP_NAME> | <NEXTHOP_ID>
<nexthop> ::= (<nexthop-identifier> | <EGRESS_INTERFACE> |
               <ipv4-address> | <ipv6-address> |
               (<EGRESS_INTERFACE> (<ipv4-address> | <ipv6-address>)
                [<RIB_NAME>]) |
               (<EGRESS_INTERFACE> <IEEE_MAC_ADDRESS>) |
               (<tunnel-encap> [<EGRESS_INTERFACE>]) |
               <logical-tunnel> |
               <RIB_NAME>)

<nexthop-identifier> ::= <NEXTHOP_NAME> | <NEXTHOP_ID>
<nexthop-address> ::= (<IPv4> <ipv4-address>) |
                      (<IPv6> <ipv6-address>) |
                      (<IEEE_MAC> <IEEE_MAC_ADDRESS>)
<special-nexthop> ::= <DISCARD> | <DISCARD_WITH_ERROR> |
                     (<RECEIVE> [<COS_VALUE>] [<rate-limiter>])
<rate-limiter> ::= <>

<logical-tunnel> ::= <tunnel-type> <TUNNEL_NAME>
<tunnel-type> ::= <IP> | <MPLS> | <GRE> | <VxLAN> | <NVGRE>

<tunnel-encap> ::= (<IPv4> <ipv4-header>) |
                   (<IPv6> <ipv6-header>) |
                   (<MPLS> <mpls-header>) |
                   (<GRE> <gre-header>) |
                   (<VXLAN> <vxlan-header>) |
                   (<NVGRE> <nvgre-header>)

```

```

<ipv4-header> ::= <SOURCE_IPv4_ADDRESS> <DESTINATION_IPv4_ADDRESS>
                  <PROTOCOL> [<TTL>] [<DSCP>]

<ipv6-header> ::= <SOURCE_IPv6_ADDRESS> <DESTINATION_IPv6_ADDRESS>
                  <NEXT_HEADER> [<TRAFFIC_CLASS>]
                  [<FLOW_LABEL>] [<HOP_LIMIT>]

<mpls-header> ::= (<mpls-label-operation> ...)
<mpls-label-operation> ::= (<MPLS_PUSH> <MPLS_LABEL> [<S_BIT>]
                           [<TOS_VALUE>] [<TTL_VALUE>]) |
                           (<MPLS_POP> [<TTL_ACTION>])

<gre-header> ::= <GRE_IP_DESTINATION> <GRE_PROTOCOL_TYPE> [<GRE_KEY>]
<vxlan-header> ::= (<ipv4-header> | <ipv6-header>)
                  [<VXLAN_IDENTIFIER>]
<nvgre-header> ::= (<ipv4-header> | <ipv6-header>)
                  <VIRTUAL_SUBNET_ID>
                  [<FLOW_ID>]

```

Figure 5: RIB rBNF grammar

7. Using the RIB grammar

The RIB grammar is very generic and covers a variety of features. This section provides examples on using objects in the RIB grammar and examples to program certain use cases.

7.1. Using route preference

Using route preference a client can pre-install alternate paths in the network. For example, if OSPF has a route preference of 10, then another client can install a route with route preference of 20 to the same destination. The OSPF route will get precedence and will get installed in the FIB. When the OSPF route is withdrawn, the alternate path will get installed in the FIB.

Route preference can also be used to prevent denial of service attacks by installing routes with the best preference, which either drops the offending traffic or routes it to some monitoring/analysis station. Since the routes are installed with the best preference, they will supersede any route installed by any other protocol.

7.2. Using different nexthops types

The RIB grammar allows one to create a variety of nexthops. This section describes uses for certain types of nexthops.

7.2.1. Tunnel nexthops

A tunnel nexthop points to a tunnel of some kind. Traffic that goes over the tunnel gets encapsulated with the tunnel encap. Tunnel nexthops are useful for abstracting out details of the network, by having the traffic seamlessly route between network edges.

7.2.2. Replication lists

One can create a replication list for replication traffic to multiple destinations. The destinations, in turn, could be complex nexthops in themselves - at a level supported by the network device. Point to multipoint and broadcast are examples that involve replication.

A replication list (at the simplest level) can be represented as:

```
<nexthop-list> ::= <nexthop> [ <nexthop> ... ]
```

The above can be derived from the grammar as follows:

```
<nexthop-list> ::= <nexthop-list-member> [<nexthop-list-member> ...]  
<nexthop-list> ::= <nexthop-chain> [<nexthop-chain> ...]  
<nexthop-list> ::= <nexthop> [ <nexthop> ... ]
```

7.2.3. Weighted lists

A weighted list is used to load-balance traffic among a set of nexthops. From a modeling perspective, a weighted list is very similar to a replication list, with the difference that each member nexthop MUST have a LOAD_BALANCE_WEIGHT associated with it.

A weighted list (at the simplest level) can be represented as:

```
<nexthop-list> ::= (<nexthop> <LOAD_BALANCE_WEIGHT>)
                  [(<nexthop> <LOAD_BALANCE_WEIGHT>)... ]
```

The above can be derived from the grammar as follows:

```
<nexthop-list> ::= <nexthop-list-member> [<nexthop-list-member> ...]
<nexthop-list> ::= (<nexthop-chain> <nexthop-list-member-attributes>)
                  [(<nexthop-chain>
                   <nexthop-list-member-attributes>) ...]
<nexthop-list> ::= (<nexthop-chain> <LOAD_BALANCE_WEIGHT>)
                  [(<nexthop-chain> <LOAD_BALANCE_WEIGHT>) ... ]
<nexthop-list> ::= (<nexthop> <LOAD_BALANCE_WEIGHT>)
                  [(<nexthop> <LOAD_BALANCE_WEIGHT>)... ]
```

7.2.4. Protection lists

Protection lists are similar to weighted lists. A protection list specifies a set of primary nexthops and a set of backup nexthops. The <PROTECTION_PREFERENCE> attribute indicates which nexthop is primary and which is backup.

A protection list can be represented as:

```
<nexthop-list> ::= (<nexthop> <PROTECTION_PREFERENCE>)
                  [(<nexthop> <PROTECTION_PREFERENCE>)... ]
```

A protection list can also be a weighted list. In other words, traffic can be load-balanced among the primary nexthops of a protection list. In such a case, the list will look like:

```
<nexthop-list> ::= (<nexthop> <PROTECTION_PREFERENCE>
                  <LOAD_BALANCE_WEIGHT>)
                  [(<nexthop> <PROTECTION_PREFERENCE>
                   <LOAD_BALANCE_WEIGHT>)... ]
```

7.2.5. Nexthop chains

A nexthop chain is a nexthop that puts one or more headers on an outgoing packet. One example is a Pseudowire - which is MPLS over some transport (MPLS or GRE for instance). Another example is VxLAN

over IP. A nexthop chain allows an external entity to break up the programming of the nexthop into independent pieces - one per encapsulation.

A simple example of MPLS over GRE can be represented as:

```
<nexthop-list> ::= (<MPLS> <mpls-header>) (<GRE> <gre-header>)
```

The above can be derived from the grammar as follows:

```
<nexthop-list> ::= <nexthop-list-member> [<nexthop-list-member> ...]  
<nexthop-list> ::= <nexthop-chain>  
<nexthop-list> ::= <nexthop> [ <nexthop> ... ]  
<nexthop-list> ::= <tunnel-encap> (<nexthop> [ <nexthop> ...])  
<nexthop-list> ::= <tunnel-encap> (<tunnel-encap>)  
<nexthop-list> ::= (<MPLS> <mpls-header>) (<GRE> <gre-header>)
```

7.2.6. Lists of lists

Lists of lists is a complex construct. One example of usage of such a construct is to replicate traffic to multiple destinations, with high availability. In other words, for each destination you have a primary and backup nexthop (replication list) to ensure there is no traffic drop in case of a failure. So the outer list is a protection list and the inner lists are replication lists of primary/backup nexthops.

7.3. Performing multicast

IP multicast involves matching a packet on (S, G) or (*, G), where both S (source) and G (group) are IP prefixes. Following the match, the packet is replicated to one or more recipients. How the recipients subscribe to the multicast group is outside the scope of this document.

In PIM-based multicast, the packets are IP forwarded on an IP multicast tree. The downstream nodes on each point in the multicast tree is one or more IP addresses. These can be represented as a replication list (Section 7.2.2).

In MPLS-based multicast, the packets are forwarded on a point to multipoint (P2MP) label-switched path (LSP). The nexthop for a P2MP LSP can be represented in the nexthop grammar as a <logical-tunnel> (P2MP LSP identifier) or a replication list (Section 7.2.2) of <tunnel-encap>, with each tunnel encap representing a single mpls downstream nexthop.

8. RIB operations at scale

This section discusses the scale requirements for a RIB data-model. The RIB data-model should be able to handle large scale of operations, to enable deployment of RIB applications in large networks.

8.1. RIB reads

Bulking (grouping of multiple objects in a single message) MUST be supported when a network device sends RIB data to an external entity. Similarly the data model MUST enable a RIB client to request data in bulk from a network device.

8.2. RIB writes

Bulking (grouping of multiple write operations in a single message) MUST be supported when an external entity wants to write to the RIB. The response from the network device MUST include a return-code for each write operation in the bulk message.

8.3. RIB events and notifications

There can be cases where a single network event results in multiple events and/or notifications from the network device to an external entity. On the other hand, due to timing of multiple things happening at the same time, a network device might have to send multiple events and/or notifications to an external entity. The network device originated event/notification message MUST support bulking of multiple events and notifications in a single message.

9. Security Considerations

All interactions between a RIB manager and an external entity MUST be authenticated and authorized. The RIB manager MUST protect itself against a denial of service attack by a rogue external entity, by throttling request processing. A RIB manager MUST enforce limits on how much data can be programmed by an external entity and return error when such a limit is reached.

The RIB manager MUST expose a data-model that it implements. An external agent MUST send requests to the RIB manager that comply with the supported data-model. The data-model MUST specify the behavior of the RIB manager on handling of unsupported data requests.

10. IANA Considerations

This document does not generate any considerations for IANA.

11. Acknowledgements

The authors would like to thank the working group co-chairs and reviewers on their comments and suggestions on this draft. The following people contributed to the design of the RIB model as part of the I2RS Interim meeting in April 2013 - Wes George, Chris Liljenstolpe, Jeff Tantsura, Sriganesh Kini, Susan Hares, Fabian Schneider and Nitin Bahadur.

12. References

12.1. Normative References

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

12.2. Informative References

- [I-D.hares-i2rs-use-case-vn-vc]
Hares, S., "Use Cases for Virtual Connections on Demand (VCoD) and Virtual Network on Demand using Interface to Routing System", draft-hares-i2rs-use-case-vn-vc-00 (work in progress), February 2013.
- [I-D.ietf-i2rs-problem-statement]
Atlas, A., Nadeau, T., and D. Ward, "Interface to the Routing System Problem Statement", draft-ietf-i2rs-problem-statement-00 (work in progress), August 2013.
- [I-D.white-i2rs-use-case]
White, R., Hares, S., and A. Retana, "Protocol Independent Use Cases for an Interface to the Routing System", draft-white-i2rs-use-case-01 (work in progress), August 2013.
- [RFC4915] Psenak, P., Mirtorabi, S., Roy, A., Nguyen, L., and P. Pillay-Esnault, "Multi-Topology (MT) Routing in OSPF", RFC 4915, June 2007.
- [RFC5120] Przygienda, T., Shen, N., and N. Sheth, "M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-ISs)", RFC 5120, February 2008.

[RFC5511] Farrel, A., "Routing Backus-Naur Form (RBNF): A Syntax
Used to Form Encoding Rules in Various Routing Protocol
Specifications", RFC 5511, April 2009.

Authors' Addresses

Nitin Bahadur (editor)
Bracket Computing
320 Soquel Way
Sunnyvale, CA 94085
US

Email: nitin_bahadur@yahoo.com

Ron Folkes (editor)
Juniper Networks, Inc.
1194 N. Mathilda Avenue
Sunnyvale, CA 94089
US

Phone: +1 408 745 2000
Email: ronf@juniper.net
URI: www.juniper.net

Sriganesh Kini (editor)
Ericsson

Email: sriganesh.kini@ericsson.com

Jan Medved
Cisco

Email: jmedved@cisco.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 16, 2014

K. Patel
R. Fernando
Cisco Systems
H. Gredler
Juniper Networks
S. Amante
Level 3 Communications, Inc.
R. White
Ericsson
S. Hares
Hickory Hill Consulting
February 12, 2014

Use Cases for an Interface to BGP Protocol
draft-keyupate-i2rs-bgp-usecases-01.txt

Abstract

A network routing protocol like BGP is typically configured and analyzed through some form of Command Line Interface (CLI) or NETCONF. These interactions to control BGP and diagnose its operation encompass: configuration of protocol parameters, display of protocol data, setting of certain protocol state and debugging of the protocol.

Interface to the Routing System's (I2RS) Programmatic interfaces, as defined in [draft-ietf-i2rs-architecture], provides an alternate way to control and diagnose the operation of the BGP protocol. I2RS may be used for the configuration, manipulation, analyzing or collecting the protocol data. This document describes set of use cases for which I2RS can be used for BGP protocol. It is intended to provide a base for the solution draft describing a set of interfaces to the BGP protocol.

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 August 16, 2014.

Copyright Notice

Copyright (c) 2014 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.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
1.1. Requirements Language	3
2. BGP Protocol Operation	4
2.1. BGP Error Handling for Internal BGP Sessions	4
3. BGP Route Manipulation	4
3.1. Customized Best Path Selection Criteria	5
3.2. Flowspec Routes	5
3.3. Route Filter Routes for Legacy Routers	5
3.4. Optimized Exit Control	6
4. BGP Events	6
4.1. Notification of Routing Events	7
4.2. Tracing Dropped BGP Routes	8
4.3. BGP Protocol Statistics	8
5. Central membership computation for MPLS based VPNs	9

6. Marking Overlapping Traffic Engineering Routes for Removal	10
7. Security Considerations	11
8. Acknowledgements	11
9. References	11
9.1. Normative References	11
9.2. Informative References	12
Appendix A. BGP Configuration	13
A.1. BGP Protocol Configuration	14
A.2. BGP Policy Configuration	14
Authors' Addresses	16

1. Introduction

Typically, a network routing protocol like BGP is configured and results of its operation are analyzed through some form of Command Line Interface (CLI) or NETCONF. These interactions to control BGP and diagnose its operation encompass: configuration of protocol parameters, display of protocol data, setting of certain protocol state and debugging of the protocol.

The I2RS Framework document [I-D.ietf-i2rs-architecture] describes a mechanism to control network protocols like BGP using a set of programmatic interfaces. These programmatic interfaces allow one to control the BGP protocol by analyzing its operational state and routing protocol data, plus manipulating BGP's configuration to achieve various goals. The I2RS is not intended to replace any existing configuration mechanisms, (i.e.: Command Line Interface or NETCONF). Instead, I2RS is intended to augment those existing mechanisms by defining a standardized set of programmatic interfaces to enable easier configuration, interrogation and analysis of the BGP protocol.

This document describes set of use cases for which I2RS's programmatic interfaces can be used to control and analyze the operation of BGP. The use cases described in this document cover the following aspects of BGP: protocol parameter configuration, protocol route manipulation and tracking of protocol events. The goal is to inform the community's understanding of where the I2RS BGP extensions fit within the overall I2RS architecture. It is intended to provide a basis for the solutions draft describing the set of Interfaces to the BGP protocol.

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. BGP Protocol Operation

It is increasingly common for services facilitated via BGP to be subject to severe, widespread disruptions (outages), primarily due to the destructive teardown of BGP sessions as a result of receiving malformed BGP attributes. The document Operational Requirements for Enhanced Error Handling Behaviour in BGP-4

[I-D.ietf-grow-ops-reqs-for-bgp-error-handling] outlines requirements to try to minimize the scope of the impact attributed to such errors. Unfortunately, more fine-grained BGP error handling solutions, which would result in little to no impact on the operation of BGP protocol, remain elusive.

2.1. BGP Error Handling for Internal BGP Sessions

It is possible that I2RS could enable enhanced error handling techniques for Internal BGP sessions. At a minimum, I2RS-capable BGP routers could signal an event such as "Malformed Attribute Received" toward an I2RS controller(s). I2RS controller(s) may already have a real-time view of BGP routes, and corresponding BGP attributes, or may dynamically interrogate BGP routers in the network to identify the present propagation scope of the BGP route(s) that are affected. Finally, the I2RS controller(s) could then signal back to BGP routers to apply a filter that would block propagation of the BGP attribute or BGP route, as necessary, in order to temporarily aid in consistency of BGP routing information across the entire network until a permanent fix can be developed and deployed within BGP routers.

I2RS would enable the global visibility and global control over the operational state of BGP, within a given Autonomous System, that is necessary to facilitate the learning of, rapid response to and more fine-grained isolation/scoping of BGP protocol events that currently cause a destructive tear-down of BGP sessions that lead to widespread disruptions of services.

3. BGP Route Manipulation

Multiprotocol BGP [RFC4760] provides support to carry routing information for different BGP address families. Route manipulation is heavily done across these different address families for different reasons. BGP IPv4 and IPv6 address families use BGP Communities [RFC1997] and other IBGP and EBGp attributes to manipulate BGP routes for Traffic Engineering purpose. BGP VPN address families use Extended Communities [RFC4360] to filter unwanted BGP routes. BGP Flowspec address family [RFC5575] is used to install Flow based filters to filter unwanted data traffic. The following sub-sections

describe the use of IRS towards BGP Route Manipulation for different BGP address families.

3.1. Customized Best Path Selection Criteria

The BGP customized Bestpath facilitates custom bestpath computations within a BGP speaking network. It is usually used within an IBGP network. Customized bestpaths use special extended communities known as cost communities. Cost communities carry enough information; Point of Insertion (POI) and the cost value to signal where in BGP bestpath the customize checks need to be done. Both, the traffic engineering as well as backdoor (SHAM) links use customized bestpath computation.

With I2RS, it would be possible for an I2RS controller to push routes with custom cost communities on the BGP routers for Traffic Engineering purpose. I2RS controller now can act as a central entity keeping track of all Traffic engineering data that get applied to BGP routes within an IBGP network.

3.2. Flowspec Routes

The BGP flowspec address family is used to disseminate the traffic flow specification to the BGP Autonomous System Border Routers (ASBRs) and Provider Edge (PE) routers. Both, the BGP ASBRs and the PEs would translate the received BGP traffic flow specification into an Access Control List (ACL) and install it in router's forwarding path. Using such ACLs routers can now classify, shape, rate limit, filter, or redirect traffic flows.

With I2RS, it would be possible for an I2RS controller to push traffic flow specifications to the BGP ASBRs and the PE routers. I2RS controller can act as a central entity tracking all the traffic flow specifications that are installed within an IBGP network. I2RS controller could also prioritize and control the announcement of traffic flow specifications according to various ASRBs and PE router's capacity. BGP ASBRs and PE routers MAY forward traffic flow specifications received from EBGp speakers to I2RS Agents. This would allow I2RS agents to centrally manage and track any externally received traffic flow specifications.

3.3. Route Filter Routes for Legacy Routers

The BGP Route Filter address family is used to disseminate the Route Target filter information between VPN BGP speakers. This information is then used to build a route distribution graph that helps in limiting the propagation of VPN NLRI within a VPN network. However, it requires that all the BGP VPN routers are upgraded to support this

functionality. Otherwise, the graph information is incomplete when a VPN network consists of legacy routers that participates in VPN but does not implement the BGP route filter address family.

With I2RS, it would be possible for an I2RS controller to push router filter information to BGP RR routers on behalf of all legacy routers that participates in VPN but does not support or implement the BGP route filter address family. I2RS controller can act as a central entity tracking all the configured Route Filters for legacy routers and push them on appropriate RRs who in turn would push it to ASBRs and PE routers. In this way, I2RS agents help build an optimal route distribution graph that would assist in filtering of VPN NLRIs in a VPN network.

3.4. Optimized Exit Control

Optimized Exit Control is used to provide route optimization and load distribution for multiple network connections between networks. Network operators can monitor IP traffic flows and then could define policies and rules based on traffic class performance, link bandwidth monetary costs, link load distribution, traffic types, link failures, etc.

With I2RS, it would be possible for an I2RS controller to manipulate BGP routes and its parameters that influence BGP bestpath decisions. I2RS controller could act as a central entity that would monitor and manipulate BGP routes based on central network based policies. Such routes would then be injected by a I2RS controller into the network so as to get the load distribution for multiple network connections.

4. BGP Events

Given the extremely large number of BGP Routes in networks, it is critical to have scalable mechanisms that can be used to monitor for events affecting routing state and, consequently, reachability. In addition, similar tools are needed in order to monitor BGP protocol statistics, which help operators and developers better understand scalability of software and hardware that BGP utilizes.

I2RS could provide a publish-subscribe capability to applications to:

- o request monitoring of BGP routes and related events; and,
- o subscribe to the I2RS controller to receive events related to BGP routes or other protocol-related events of interest.

4.1. Notification of Routing Events

There are certain IP prefixes, for example those that are arbitrarily classified by a given network operator as "high visibility" by its end-users, for which immediate notification of changes in their state are extremely useful to know about. Upon notification of such events, a Network Operations Center (NOC) could respond to customer inquiries in a more timely fashion; alternatively, the NOC may decide to perform Traffic Engineering to restore service, etc.

Currently, the only way to learn of such events is for a BGP monitoring system to establish a BGP session with a multitude of BGP routers in an AS. Then, the BGP monitoring system needs to look through all BGP UPDATE's in order to identify those events that are of interest to it. Note, this doesn't account for the fact that there are several applications that might be simultaneously interested in learning of events to a given IP prefix nor the fact that some applications may want to dynamically insert or remove "IP prefixes of interest", depending on the needs of their constituent applications.

With I2RS, it is conceivable that applications could tell an I2RS controller, through a North-Bound API, their "IP prefixes" (or, AS_PATH's, BGP communities, etc.) that are of interest. For example, a NOC application may be interested in changes to high visibility content or service-provider Web sites; alternatively, a security application may be interested in events associated with a different set of IP prefixes. The I2RS controller would then consolidate the list of IP prefixes, and associated characteristics, to be monitored and program BGP routers in an AS to observe this subset of routes for changes. Some examples of changes in routing state might include:

- o an IP prefix being announced or withdrawn
- o an IP prefix being suppressed, due to route flap dampening
- o an alternative best-path being chosen for a given IP prefix

When the requisite events for a BGP Route are observed by a BGP router, it would notify I2RS agents.

The I2RS agents would have a publish/subscribe mechanism whereby various sets of applications may subscribe to events of interest. The I2RS controller would then publish these events so applications would immediately receive them and take the appropriate domain-specific action necessary.

4.2. Tracing Dropped BGP Routes

It is extremely useful to operators to be able to rapidly identify instances where a BGP route is not being propagated within an Autonomous System. At a minimum, this could result in sub-optimal performance when attempting to reach such destinations.

There are two instances when this scenario will occur. First, when a Service Provider is using "Soft Reconfiguration Inbound", it allows their ASBR routers to receive a copy of a BGP route, but show that route was not permitted into the Adj-RIB-In most likely as a result of the inbound BGP policy not permitting that IP prefix. Thus, this BGP route is not even eligible for BGP Path Selection. The second instance is where the BGP route is permitted by the inbound BGP policy into the Adj-RIB-In, but due to BGP Path Selection (i.e.: lower LOCAL_PREF, longer AS_PATH length, etc.) was not chosen as the best path and, subsequently, this particular BGP route is not forwarded on to other internal BGP speakers in the AS. In both instances, the BGP route is only visible within the ASBR on which that BGP route was first learned. Needless to say, in large Service Provider networks with a numerous interconnects to a single customer it can be very time-consuming to discover where such a BGP route is learned before ultimately determining why the route was blocked or not preferred.

With I2RS, it would be possible for an I2RS controller to rapidly gather information from across a large set of BGP routers in the network to determine at what ASBR's the BGP route is being learned. Next, the I2RS controller could interrogate those routers BGP policies to determine the root cause of why the route was either not learned or not preferred in BGP. Finally, if necessary, the I2RS controller(s) could amend BGP policies and push them out to BGP routers to permit the BGP route or make it a preferred route according to the BGP path selection algorithm.

4.3. BGP Protocol Statistics

There are a variety of statistics related to the operation of BGP that are invaluable to network operators. These statistics generally help operators, and developers, understand the present state and future scalability of BGP.

One statistic that is invaluable to operators is the current number of BGP routes learned through an eBGP session. Operators then apply a command against each eBGP session to limit the maximum number of BGP routes that may be learned through that eBGP session before a warning message is triggered and/or the eBGP session is torn down completely. This configuration capability is often referred to as a

"max-prefix limit". This command must be routinely audited and, if necessary, adjusted in order to not trigger a false warning or teardown due to the natural organic growth in BGP routes learned from a given BGP neighbor.

I2RS agents could provide an invaluable capability to help audit and re-program the "max-prefix limit" on a periodic basis, which is generally once per day. Specifically, the first task would be for an I2RS controller to validate that there is a "max-prefix limit" applied to every eBGP session. (If there is not, that should either trigger a red alarm to the NOC to manually fix this condition or for the I2RS controller to automatically apply a "max-prefix limit" that would alleviate this hazardous condition). Assuming there is a "max-prefix limit" already in place, the I2RS controller would simultaneously retrieve, from each BGP router, the current number of BGP routes learned through a BGP session and value used for the "max-prefix limit" on that same BGP session. These two values could then be handed off to an application that determines if adjustments in the "max-prefix limit" value are required for each BGP session. The application would then notify the I2RS controller of the subset of eBGP sessions and their associated change in "max-prefix limit" value, whereby the I2RS controller would then adjust the BGP protocol configuration on each requisite BGP router in the network. Finally, it should be noted that the above is just one method whereby "max-prefix limit" values are adjusted. It's similarly possible that the BGP routers may, through the I2RS, pull the "max-prefix limit" values for each eBGP neighbor they have on-board on a periodic basis and validate their accuracy.

The above is just one use case related to BGP protocol statistics. There are wealth of other BGP protocol statistics or state information that would be invaluable to have programmatic visibility into that operators do not have today.

5. 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 and others. 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 I2RS 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

6. Marking Overlapping Traffic Engineering Routes for Removal

It is often the case that routes are advertised not to provide reachability (in the strict sense), but rather to provide optimal reachability, or to engineer the path traffic takes to a particular

destination. While this can improve the efficiency of a network's operation, it can also increase the amount of state carried in the control plane beyond the point where the additional state has any real effect on traffic flow. Removing Overlapping Routes [I-D.white-grow-overlapping-routes] provides a mechanism designed to remove these traffic engineering routes once they are beyond the point of actually impacting traffic flows in the network.

Summary of I2RS Capabilities and Interactions:

- o The ability to read the loc-RIB-in BGP table to discover overlapping routes, and determine which may be safely marked for removal.
- o The ability to modify filtering rules and initiate a re-computation of the local BGP table through those policies to cause specific routes to be marked for removal at the outbound eBGP edge.

7. Security Considerations

The BGP use cases described in this document assumes use of I2RS programmatic interfaces described in the I2RS framework mentioned in [I-D.ietf-i2rs-architecture]. This document does not change the underlying security issues inherent in the existing in [I-D.ietf-i2rs-architecture].

8. Acknowledgements

The authors would like to thank Ed Crabbe, Joel Halpern, Wes George, Carlos Pignataro, Jon Mitchell and Bill Atwood for their comments and suggestions.

9. References

9.1. Normative References

- [I-D.ietf-i2rs-architecture]
Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture-00 (work in progress), August 2013.
- [RFC1997] Chandrasekeran, R., Traina, P., and T. Li, "BGP Communities Attribute", RFC 1997, August 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.
- [RFC3392] Chandra, R. and J. Scudder, "Capabilities Advertisement with BGP-4", RFC 3392, November 2002.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.
- [RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.
- [RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, February 2006.
- [RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, January 2007.

9.2. Informative References

- [I-D.ietf-grow-ops-reqs-for-bgp-error-handling]
Shakir, R., "Operational Requirements for Enhanced Error Handling Behaviour in BGP-4", draft-ietf-grow-ops-reqs-for-bgp-error-handling-05 (work in progress), July 2012.
- [I-D.ietf-i2rs-architecture]
Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture-00 (work in progress), August 2013.
- [I-D.mcpherson-irr-routing-policy-considerations]
McPherson, D., Amante, S., Osterweil, E., and L. Blunk, "IRR & Routing Policy Configuration Considerations", draft-mcpherson-irr-routing-policy-considerations-01 (work in progress), September 2012.
- [I-D.white-grow-overlapping-routes]
White, R., Retana, A., and S. Hares, "Filtering of Overlapping Routes", draft-white-grow-overlapping-routes-01 (work in progress), February 2013.
- [RFC2622] Alaettinoglu, C., Villamizar, C., Gerich, E., Kessens, D., Meyer, D., Bates, T., Karrenberg, D., and M. Terpstra, "Routing Policy Specification Language (RPSL)", RFC 2622, June 1999.

- [RFC2858] Bates, T., Rekhter, Y., Chandra, R., and D. Katz,
"Multiprotocol Extensions for BGP-4", RFC 2858, June 2000.
- [RFC5156] Blanchet, M., "Special-Use IPv6 Addresses", RFC 5156,
April 2008.
- [RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J.,
and D. McPherson, "Dissemination of Flow Specification
Rules", RFC 5575, August 2009.
- [RFC5735] Cotton, M. and L. Vegoda, "Special Use IPv4 Addresses",
RFC 5735, January 2010.

Appendix A. BGP Configuration

The configuration of BGP is arduous to establish and maintain, particularly on networks whose services have a requirement for complex routing policies. This need is magnified by the need to routinely perform changes to large numbers of BGP routers to, for example: add or remove customer's BGP sessions, announce or withdraw (customer) IP prefixes in BGP, modify BGP policies to effect changes in Traffic Engineering, audit BGP routers to ensure they have consistent and appropriate BGP policies, and others.

There are three categories of BGP configuration:

1. Local BGP routing protocol configuration: local Autonomous System Number (ASN), BGP path selection properties of the router, injection of (aggregate) routes into BGP, etc.
2. Local BGP policies: policies designed to filter and/or manipulate BGP attributes associated with BGP routes learned through BGP sessions. These policies typically live in the global configuration of a BGP router, but are applied on a per-BGP neighbor basis (or, group of BGP neighbors); and,
3. BGP neighbor sessions: remote ASN, remote IP address, address families, BGP policies to applied to routes, max-prefix limits, etc.

The sum total of BGP configuration on a BGP router is typically the largest quantify of configuration on Service Provider's BGP routers, by a fairly large margin. When that is combined with the large set of routine configuration changes, mentioned above, it should be fairly clear that systematic reading, configuration and control of BGP routers through a mechanism like I2RS would greatly benefit all operators of BGP routers.

While it may not be possible to provide programmatic APIs for esoteric vendor-specific policy configuration, it is possible to provide such API's for BGP protocol specific configuration and the more commonly used BGP routing policies.

A.1. BGP Protocol Configuration

Ability to enable and disable new address families within a BGP protocol for a network of BGP speaking routers is a challenge. The challenge is mainly in keeping track of BGP speaker's feature capabilities and then configuration of new address families on a multiple BGP speakers within a given network. With the necessary information, I2RS agents allow a network operator to push configuration information for enabling and disabling of new address families on a partial or entire set of BGP speakers within a given network. This would assist in building BGP overlay networks as needed.

For VPN address families, the main challenge lies in the complex VPN configuration required to setup the control plane for Customer VPNs. The configuration involves creating a Virtual Routing and Forwarding instance (VRF), a Route Distinguisher (RD) that ensures each customer prefixes remains unique across VPNs, and Route Targets (RT) that help ensure that the Customer prefixes are segregated appropriately so that they do not cross the VPN boundaries. I2RS would allow a network operator to push such configuration from a central location where a global VPN provisioning information could be stored. This helps avoid manual configuration of a VPN on multiple routers. Instead the configuration is controlled and pushed through a central I2RS controller using a programmatic set of APIs on targeted set of BGP speakers.

Use of I2RS agents to announce protocol configuration information would simplify and automate configuration of BGP protocol in IBGP deployments where the protocol based policies are seldom used. To facilitate such a centralized configuration model, BGP speakers could be extended to use programmatic APIs to announce their feature capabilities as part of protocol initialization to the centralize I2RS agents. This would assist I2RS agents to auto-discover BGP protocol capabilities of various BGP speakers in a given network. I2RS agents in turn would use the information towards enabling/disabling of BGP specific features on BGP speakers.

A.2. BGP Policy Configuration

Filtering of BGP routes is strongly recommended to control the announcements of BGP prefixes across the internet. Most providers

make extensive use of BGP prefix filtering policies at the edge of their networks. The reasons for filtering BGP prefixes are:

- o Avoid Unwanted Route Announcements. Filter prefixes that MUST not be routed [RFC5735], [RFC5156]. Filter prefixes that are not allocated by Internet Routing Registries.
- o Facilitate Route Summarization. Filter prefixes beyond certain agreed prefix mask length between providers. Route Summarization helps control BGP RIB and FIB table size.
- o Defensive Security. Filter prefixes from Stub customer ASes that are not owned by the customers. Filter customer prefixes announced by other providers. This helps avoid prefix hijacking.

A set of standards-based schemas to enable configuration of Local BGP policies and BGP neighbor sessions was realized through the Routing Policy Specification Language (RPSL) [RFC2622]. The RPSL defined a standards-based schemas, or 'objects' as it called them, that defined:

- o binding of IP prefixes to (one or more) Origin AS, (route objects);
- o collections of routes (route-set objects);
- o collections of Autonomous Systems (as-set objects); and,
- o routing policy of an Autonomous System to/from its adjacent neighbor AS'es, (aut-num objects)

Each ASN is responsible for creation, modification and deletion of its RPSL objects in an Internet Routing Registry (IRR). IRR's are typically operated by Regional Internet Registries (RIR's) and a few dozen larger ISP's and independent organizations. The IRR's provide a well-known location for all organizations attached to the Internet to retrieve or update RPSL objects.

While still widely and actively used by Internet Service Providers, the prevailing belief is that the data contained in the IRR's is inaccurate, primarily due to a lack of deployed authorization method with respect to the creation or modification of RPSL objects. It should be noted that this criticism is not directed at the previously defined RPSL schemas, but rather at the data contained in RPSL schemas by end-users of the IRR system. Please refer to the IRR And Routing Policy Configuration Considerations [I-D.mcpherson-irr-routing-policy-considerations] document for a more thorough discussion of the history and present state of the IRR's.

Currently, RPSL schemas are exchanged between non-routing systems (servers) used within the IRR system. In addition, open-source and proprietary applications create or modify RPSL schemas, as necessary, to signal the announcement (or, withdrawal) of an IP prefix from an ASN or the creation (or, teardown) of a neighbor relationship between two adjacent ASN's. Most importantly, these RPSL schemas are consumed by similar applications to automatically build routing policies, (i.e.: lists of IP prefixes, corresponding Origin ASN's and /or AS_PATH's), that then get translated to device-specific syntax (i.e.: CLI) before being pushed into individual BGP routers to effect routing policy on the network. It is common for Internet Service Providers to perform updates to these routing policies across their entire network on a daily basis.

With I2RS it would be desirable to change the last step in the above process so that BGP policies derived from RPSL schemas, and other information sources, are translated into standards-based schemas that are then pushed, or pulled, into individual BGP routers. More generally, I2RS agents could use API's to gather information required to build various types of BGP routing policies plus the corresponding set of Autonomous System Border Routers (ASBR's) where such policies need to be applied in the network and, finally, making those changes to individual network elements so those BGP policies take effect in the network. In doing so, a network operator now has a centralized way of building and making these policies take effect across the network in a coordinated manner.

Authors' Addresses

Keyur Patel
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95134
USA

Email: keyupate@cisco.com

Rex Fernando
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95134
USA

Email: rex@cisco.com

Hannes Gredler
Juniper Networks
1194 N. Mathilda Ave
Sunnyvale, CA 94089
USA

Email: hannes@juniper.net

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

Email: shane@level3.net

Russ White
Ericsson

Email: russw@riw.us

Susan Hares
Hickory Hill Consulting

Email: shares@ndzh.com

i2rs
Internet-Draft
Intended status: Informational
Expires: August 18, 2014

R. White
IETF
S. Hares
Hickory Hill
A. Retana
Cisco Systems, Inc.
February 14, 2014

Protocol Independent Use Cases for an Interface to the Routing System
draft-white-i2rs-use-case-02

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 (I2RS).

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 August 18, 2014.

Copyright Notice

Copyright (c) 2014 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	2
2. Distributed Reaction to Network Based Attacks	3
3. Remote Service Routing	5
4. Within Data Center Routing	7
5. Temporary Overlays between Data Centers	9
6. References	10
6.1. Normative References	10
6.2. Informative References	11
Authors' Addresses	11

1. Introduction

The Architecture for the Interface to the Routing System [I-D.ietf-i2rs-architecture] allows for 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. The RIB Info Model [I-D.ietf-i2rs-rib-info-model] specifies the information elements accessible by the I2RS system in the RIB.

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 utilizing the RIB information model will be useful. The goal is to inform not only the community's understanding of where I2RS fits in the larger scheme of SDN proposals, but also to inform the requirements, framework, and specification of I2RS 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.

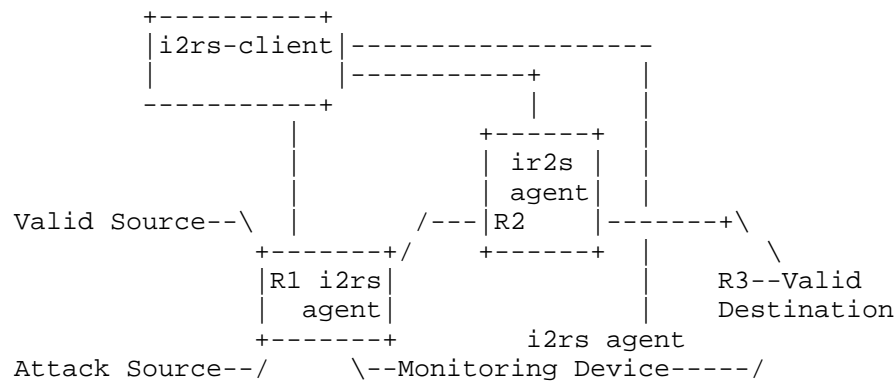
2. Distributed Reaction to Network Based Attacks

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 I2RS 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 I2RS 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. The information pulled from the RIB must include the destination prefix (NLRI), the table identifier (if the forwarding device has multiple forwarding instances), the metric of the installed route, and the identifier for 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.

Comparison of I2RS Capabilities versus the I2RS RIB

The RIB Info Model [I-D.ietf-i2rs-rib-info-model] specifies the routes as: Routing-instance, RIB, route where route has attributes, family attributes (IPv4, Ipv6, MPLS, MAC, interface), and next-hop list. The RIB info model does not keep information on the FIB the route was installed in, the metric of the installed route, or the identifier of the installing process.

The RIB Info Model does not provide a specific indication that the default (zero length prefix) route can be installed, but this can be implied from the different match lengths.

The ability to interact with various policies via NETCONF has not be specified directly. Indications that this should occur in the must respond with a return code that indicates the route is installed in FIB, but it does not save the FIB table identifier or the installing process.

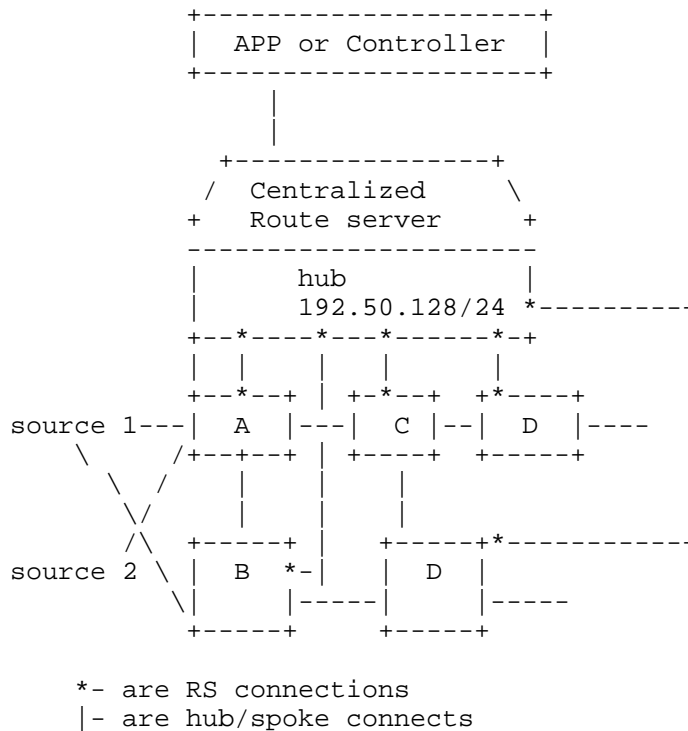
The ability to interact with traffic flow and other network traffic level measurement protocols and systems is not included in any I2RS information model.

3. 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 I2RS 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.

4. 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. [I-D.lapukhov-bgp-routing-large-dc] examines many of these elements of data center design.

One element of these Data Center routing infrastructures is the ability to quickly read topology information and execute 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 I2RS, many Data Centers are using extra physical topologies or logical topologies to work around the features.

An I2RS solution would use the same elements, but with a different control plane. The I2RS enabled 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 I2RS 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 I2RS 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.

5. Temporary Overlays between Data Centers

Data Centers within one organization may operate as one single entity even though they may be geographically distributed. 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.

I2RS 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. I2RS 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.

6. References

6.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3746] Yang, L., Dantu, R., Anderson, T., and R. Gopal, "Forwarding and Control Element Separation (ForCES) Framework", RFC 3746, April 2004.

6.2. 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-i2rs-architecture]
Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture-01 (work in progress), February 2014.
- [I-D.ietf-i2rs-rib-info-model]
Bahadur, N., Folkes, R., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-ietf-i2rs-rib-info-model-01 (work in progress), October 2013.
- [I-D.lapukhov-bgp-routing-large-dc]
Lapukhov, P., Premji, A., and J. Mitchell, "Use of BGP for routing in large-scale data centers", draft-lapukhov-bgp-routing-large-dc-06 (work in progress), August 2013.

Authors' Addresses

Russ White
IETF

Email: russw@riw.us

Susan Hares
Hickory Hill

Email: shares@ndzh.com

Alvaro Retana
Cisco Systems, Inc.
7025 Kit Creek Road
Research Triangle Park, NC 27617
USA

Email: aretana@cisco.com