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

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

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

Abstract

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

Status of this Memo

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

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

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

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

Table of Contents

1. Introduction

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

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

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

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

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

1.1. Functional Overview

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

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

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

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

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

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

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

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

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

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

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

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

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

1.2. Example Use-Cases

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

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

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

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

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

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

distribution tree that provides redundancy.

2. Programmatic Interfaces

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

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

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

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

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

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

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

3. Common Interface Considerations

3.1. Capabilities

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

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

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

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

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

3.2. Identity, Authorization, Authentication, and Security

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

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

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

3.3. Speed and Frequency of State Installation

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

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

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

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

3.4. Lifetime of IRS-Installed Routing System State

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

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

There are four basic models to be supported.

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

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

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

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

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

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

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

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

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

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

4. Bidirectional Interfaces to the Routing System

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

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

interoperability between different vendor implementations.

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

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

4.1. Static Routing

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

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

4.1.1. Routing Information Base Interface

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

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

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

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

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

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

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

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

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

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

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

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

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

4.1.2. Label Forwarding Information Base Interface

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

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

4.1.3. Multicast Routing Information Base Interface

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

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

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

4.2. Beyond Destination-based Routing

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

4.2.1. Policy-Based Routing Interface

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

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

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

4.2.2. QoS State

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

4.3. Protocol Interactions

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

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

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

4.3.1. IGP Interfaces

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

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

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

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

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

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

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

4.3.2. BGP Interface

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

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

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

4.3.3. PIM and mLDP Interfaces

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

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

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

4.4. Triggered Sessions and Signaling

4.4.1. OAM-related Sessions Interface

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

4.4.2. Dynamic Session Creation

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

4.4.3. Triggered Signaling

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

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

5. Interfaces for Learned Information from the Routing System

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

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

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

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

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

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

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

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

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

5.1. Efforts to Obtain Topological Data

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

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

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

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

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

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

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

5.2. Measurements

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

5.3. Events

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

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

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

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

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

6. Manageability Considerations

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

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

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

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

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

7. IANA Considerations

This document includes no request to IANA.

8. Security Considerations

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

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

More specifics on the security requirements requires further investigation.

9. Acknowledgements

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

10. Informative References

[I-D.gredler-idr-ls-distribution]
Gredler, H., Medved, J., Previdi, S., and A. Farrel, "North-Bound Distribution of Link-State and TE Information using BGP", draft-gredler-idr-ls-distribution-02 (work in progress), July 2012.

[I-D.ietf-isis-genapp]
Ginsberg, L., Previdi, S., and M. Shand, "Advertising Generic Information in IS-IS", draft-ietf-isis-genapp-04 (work in progress), November 2010.

[I-D.ietf-pce-stateful-pce]
Crabbe, E., Medved, J., Varga, R., and I. Minei, "PCEP Extensions for Stateful PCE", draft-ietf-pce-stateful-pce-01 (work in progress), July 2012.

[RFC3137] Retana, A., Nguyen, L., White, R., Zinin, A., and D. McPherson, "OSPF Stub Router Advertisement", RFC 3137, June 2001.

[RFC5250] Berger, L., Bryskin, I., Zinin, A., and R. Coltun, "The OSPF Opaque LSA Option", RFC 5250, July 2008.

[RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek, "Architecture for IP Flow Information Export", RFC 5470, March 2009.

[RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules", RFC 5575, August 2009.

[RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, October 2009.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

Authors' Addresses

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

Email: akatlas@juniper.net

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

Email: tnadeau@juniper.net

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

Email: wardd@cisco.com