Internet Engineering Task Force (IETF)
Request for Comments: 7920
Category: Informational
ISSN: 2070-1721

A. Atlas, Ed.
Juniper Networks
T. Nadeau, Ed.
Brocade
D. Ward
Cisco Systems

June 2016

Problem Statement for the Interface to the Routing System

Abstract

Traditionally, routing systems have implemented routing and signaling (e.g., MPLS) to control traffic forwarding in a network. Route computation has been controlled by relatively static policies that define link cost, route cost, or import and export routing policies. Requirements have emerged to more dynamically manage and program routing systems due to the advent of highly dynamic data-center networking, on-demand WAN services, dynamic policy-driven traffic steering and service chaining, the need for real-time security threat responsiveness via traffic control, and a paradigm of separating policy-based decision-making from the router itself. These requirements should allow controlling routing information and traffic paths and extracting network topology information, traffic statistics, and other network analytics from routing systems.

This document proposes meeting this need via an Interface to the Routing System (I2RS).

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Traditionally, routing systems have implemented routing and signaling (e.g., MPLS) to control traffic forwarding in a network. Route computation has been controlled by relatively static policies that define link cost, route cost, or import and export routing policies. The advent of highly dynamic data-center networking, on-demand WAN services, dynamic policy-driven traffic steering and service chaining, the need for real-time security threat responsiveness via traffic control, and a paradigm of separating policy-based decision-making from the router itself has created the need to more dynamically manage and program routing systems in order to control routing information and traffic paths and to extract network topology information, traffic statistics, and other network analytics from routing systems.

As modern networks continue to grow in scale and complexity and desired policy has become more complex and dynamic, there is a need to support rapid control and analytics. The scale of modern networks and data centers and the associated operational expense drives the need to automate even the simplest operations. The ability to quickly interact via more complex operations to support dynamic policy is even more critical.

In order to enable network applications to have access to and control over information in the different vendors' routing systems, a publicly documented interface is required. The interface needs to support real-time, asynchronous interactions using efficient data models and encodings that are based on and extend those previously defined. Furthermore, the interface must be tailored to provide a solid base on which a variety of use cases can be supported.

To support the requirements of orchestration software and automated network applications to dynamically modify the network, there is a need to learn topology, network analytics, and existing state from the network as well as to create or modify routing information and network paths. A feedback loop is needed so that changes made can be verifiable and so that these applications can learn and react to network changes.

Proprietary solutions to partially support the requirements outlined above have been developed to handle specific situations and needs. Standardizing an interface to the routing system will make it easier to integrate use of it into a network. Because there are proprietary partial solutions already, the standardization of a common interface should be feasible.

It should be noted that during the course of this document, the term "applications" is used. This is meant to refer to an executable program of some sort that has access to a network, such as an IP or MPLS network, via a routing system.

2. I2RS Model and Problem Area for the IETF

Managing a network of systems running a variety of routing protocols and/or providing one or more additional services (e.g., forwarding, classification and policing, firewalling) involves interactions between multiple components within these systems. Some of these systems or system components may be virtualized, co-located within the same physical system, or distributed. In all cases, it is desirable to enable network applications to manage and control the services provided by many, if not all, of these components, subject to authenticated and authorized access and policies.

A data-model-driven interface to the routing system is needed. This will allow expansion of what information can be read and controlled and allow for future flexibility. At least one accompanying protocol with clearly defined operations is needed; the suitable protocol(s) can be identified and expanded to support the requirements of an Interface to the Routing System (I2RS). These solutions must be designed to facilitate rapid, isolated, secure, and dynamic changes to a device's routing system. These would facilitate wide-scale deployment of interoperable applications and routing systems.

The I2RS model and problem area for IETF work is illustrated in Figure 1. This document uses terminology defined in [RFC7921]. The I2RS agent is associated with a routing element, which may or may not be co-located with a data plane. The I2RS client could be integrated in a network application or controlled and used by one or more separate network applications. For instance, an I2RS client could be provided by a network controller or a network orchestration system that provides a non-I2RS interface to network applications and an I2RS interface to I2RS agents on the systems being managed. The scope of the data models used by I2RS extends across the entire routing system and the selected protocol(s) for I2RS.

As depicted in Figure 1, the I2RS client and I2RS agent in a routing system are objects with in the I2RS scope. The selected protocol(s) for I2RS extend between the I2RS client and I2RS agent. All other objects and interfaces in Figure 1 are outside the I2RS scope for standardization.

```
        +***************+   +***************+   +***************+
        *  Application  *   *  Application  *   *  Application  *
        +***************+   +***************+   +***************+
        |  I2RS Client  |           ^                  ^
        +---------------+           *                  *
                 ^                  *   ****************
                 |                  *   *
                 |                  v   v
                 |           +---------------+         +-------------+
                 |           |  I2RS Client  |<------->| Other I2RS  |
                 |           +---------------+         | Agents      |
                 |                   ^                 +-------------+
                 |________________   |
                                  |  |  <== I2RS Protocol
                                  |  |
       ...........................|..|..................................
       .                          v  v                                 .
       . +*************+     +---------------+      +****************+ .
       . *  Policy     *     |               |      *   Routing  &   * .
       . * Database    *<***>|  I2RS Agent   |<****>*   Signaling    * .
       . +*************+     |               |      *   Protocols    * .
       .                     +---------------+      +****************+ .
       .                        ^   ^     ^                  ^         .
       . +*************+        *   *     *                  *         .
       . *  Topology   *        *   *     *                  *         .
       . *  Database   *<*******+   *     *                  v         .
       . +*************+            *     *         +****************+ .
       .                            *     +********>*  RIB Manager   * .
       .                            *               +****************+ .
       .                            *                        ^         .
       .                            v                        *         .
       .                 +*******************+               *         .
       .                 * Subscription &    *               *         .
       .                 * Configuration     *               v         .
       .                 * Templates for     *      +****************+ .
       .                 * Measurements,     *      *  FIB Manager   * .
       .                 * Events, QoS, etc. *      *  & Data Plane  * .
       .                 +*******************+      +****************+ .
       .................................................................
```

```
     <-->  interfaces inside the scope of I2RS Protocol

     +--+  objects inside the scope of I2RS-defined behavior

     <**>  interfaces NOT within the scope of I2RS Protocol

     +**+  objects NOT within the scope of I2RS-defined behavior

     <==   used to point to the interface where the I2RS Protocol
           would be used

     ....  boundary of a router supporting I2RS
```

Figure 1: I2RS Model and Problem Area

The protocol(s) used to carry messages between I2RS clients and I2RS agents should provide the key features specified in Section 5.

I2RS will use a set of meaningful data models for information in the routing system and in a topology database. Each 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. As shown in Figure 1, I2RS needs to interact with several logical components of the routing element: policy database, topology database, subscription and configuration for dynamic measurements/events, routing and signaling protocols, and its Routing Information Base (RIB) manager. This interaction is both for writing (e.g., to policy databases or RIB manager) as well as for reading (e.g., dynamic measurement or topology database). An application should be able to combine data from individual routing elements to provide network-wide data model(s).

The data models should translate into a concise transfer syntax, sent via the I2RS protocol, that is straightforward for applications to use (e.g., a web services design paradigm). The information transfer should use existing transport protocols to provide the reliability, security, and timeliness appropriate for the particular data.

3. Standard Data Models of Routing State for Installation

As described in Section 1, there is a need to be able to precisely control routing and signaling state based upon policy or external measures. One set of data models that I2RS should focus on is for interacting with the RIB layer (e.g., RIB, Label Information Base (LIB), multicast RIB, policy-based routing) to provide flexibility and routing abstractions. As an example, the desired routing and signaling state might 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.

Efforts to provide this level of control have focused on standardizing data models that describe the forwarding plane (e.g., Forwarding and Control Element Separation (ForCES) [RFC3746]). I2RS recognizes that the routing system and a router's OS provide useful mechanisms that applications could usefully harness to accomplish application-level goals. Using routing indirection, recursion, and common routing abstractions (e.g., tunnels, Label Switched Paths (LSPs), etc.) provides significant flexibility and functionality over collapsing the state to individual routes in the Forwarding Information Base (FIB) that need to be individually modified when a change occurs.

In addition to interfaces to control the RIB layer, there is a need to dynamically configure policies and parameter values for the various routing and signaling protocols based upon application-level policy decisions.

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, measure the behavior of various flows, and understand the existing configuration and state of the router. I2RS should provide 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 [RFC7752]) still only provide the current active state as seen at the IGP and BGP layers. Detailed topological state that provides more information than the current functional status (e.g., active paths and links) is needed by applications. Examples of missing information include paths or links that are 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 IP Flow Information Export (IPFIX) [RFC5470] may be the facilitator for delivering the data, providing the ability for an application to dynamically request that measurements be taken and data delivered is important.

There is a wide range of events that applications could use to support verification of router state before other network state is changed (e.g., that a route has been installed) and to allow timely action in response to changes of relevant routes by others or to 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 (e.g., route installed, route changed, primary LSP changed, etc.)

5. Aspects to be Considered for an I2RS Protocol

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 independent atomic 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 be 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 at different times; data may even be overwritten by a different I2RS client. The details of how this should be handled are described in [RFC7921].

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 (for example, 10,000 per second to handle many individual subscriber routes changing simultaneously).

Low Latency: Within a sub-second timescale, it should be possible to complete simple operations (e.g., reading or writing a single prefix route).

Multiple Channels: 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. One channel, with authorization and authentication, may be considered primary; only an authorized client can then request that information be delivered on a different channel. Writes from a client are only expected on channels that provide authorization and authentication.

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 and Access: Any ability to manipulate routing state must be subject to authentication and authorization. Sensitive routing information also may need to be provided via secure access back to the I2RS client. Such communications must be integrity protected. Most communications will also require confidentiality.

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

6. Security Considerations

Security is a key aspect of any protocol that allows state installation and extracting of detailed router state. The need for secure control and access is mentioned in Section 5. More architectural security considerations are discussed in [RFC7921]. Briefly, the I2RS agent is assumed to have a separate authentication and authorization channel by which it can validate both the identity and the permissions associated with an I2RS client. Mutual authentication between the I2RS agent and I2RS client is required. Different levels of integrity, confidentiality, and replay protection are relevant for different aspects of I2RS.

7. References

7.1. Normative References

[RFC7921] Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", RFC 7921, DOI 10.17487/RFC7921, June 2016, <http://www.rfc-editor.org/info/rfc7921>.

7.2. Informative References

[RFC3746] Yang, L., Dantu, R., Anderson, T., and R. Gopal, "Forwarding and Control Element Separation (ForCES) Framework", RFC 3746, DOI 10.17487/RFC3746, April 2004, <http://www.rfc-editor.org/info/rfc3746>.

[RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek, "Architecture for IP Flow Information Export", RFC 5470, DOI 10.17487/RFC5470, March 2009, <http://www.rfc-editor.org/info/rfc5470>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <http://www.rfc-editor.org/info/rfc7752>.

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 ones 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. CLI standardization is not considered as a candidate solution for the problems motivating I2RS.

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 rollback, 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 Protocol (NETCONF) [RFC6241] has made many strides at overcoming most of the limitations around configuration that were just described. However, as a new technology and with the initial lack of standard data models, the adoption of NETCONF has been slow. As needed, I2RS will identify and define information and data models to support I2RS applications. Additional extensions to handle multi-headed control may need to be added to NETCONF and/or appropriate data models.

Acknowledgements

The authors would like to thank Ken Gray, Ed Crabbe, Nic Leymann, Carlos Pignataro, Kwang-koog Lee, Linda Dunbar, Sue Hares, Russ Housley, Eric Grey, Qin Wu, Stephen Kent, Nabil Bitar, Deborah Brungard, and Sarah Banks for their suggestions and review.

Authors' Addresses

Alia Atlas (editor)
Juniper Networks

Email: akatlas@juniper.net

Thomas D. Nadeau (editor)
Brocade

Email: tnadeau@lucidvision.com

Dave Ward
Cisco Systems

Email: wardd@cisco.com