The field of network architecture faces two core problems: complexity and flexibility. Networks must be more flexible to match how quickly business moves, and they must become less complex to provide improved performance at reduced cost. One approach to these problems is software-defined networks (SDNs), a loose collection of protocols built around the idea of centralizing some or all of the control plane and policies a network needs to run. The IETF’s Interface to the Routing System (I2RS) protocol tracks and controls the dynamic state in routers and switches passing this traffic. Here, we examine the historical context that led to I2RS’s development, present a use case for the I2RS protocol, describe the protocol framework, and highlight future work on I2RS in the IETF.

I2RS’s Historical Context
Mobile data traffic and cloud services are changing the profile of Internet data traffic. Mobile data traffic grew 70 percent in 2012 to 10.26 exabytes (Ebytes) per month, with 51 percent of the traffic being video data. This mobile traffic is expected to grow to 240 Ebytes per year by 2017. Traffic from users to data centers in 2012 was 438 Ebytes, and traffic between data centers was 173 Ebytes, figures projected to grow to 1,160 and 468 Ebytes, respectively, by 2017. To put this in perspective, from 1987 to 2012, the amount of total Internet traffic was approximately 150 Ebytes. Yet even current traffic levels are dwarfed by intra-data-center traffic, which Cisco estimated at 1,138 Ebytes in 2012, with 5,021 Ebytes projected for 2017. As traffic levels increase — and network costs scale with traffic requirements — network operators are looking for ways to reduce the costs of carrying the data generated from homes, mobile access points, and businesses. One method operators are examining is reducing network device costs by moving to less expensive “white-box” equipment, and separating the data plane from the control plane (thus virtualizing the control plane). Operators are also looking at SDNs to provide centralized policy management, and possibly even a totally centralized control plane, to offer faster service provisioning with less complex management.

Software-defined networking (SDN) is an industry term that implies flexible software control of network dataflows. Rather than entering into the debate surrounding SDN’s definition, we use the term to designate one type of control software that virtualizes control plane functions in a distributed, centralized, or combined distributed/centralized fashion. We present use cases to clarify how network operators can utilize the SDN approach to the control plane/data plane split. The forwarding plane can be controlled by OpenFlow Switch (OFS), I2RS dynamic static changes, traditional methods (the Simple Network Management Protocol, the Network Configuration Protocol [NETCONF]), or proprietary Command Line Interface, or
any combination of these protocols. Google has deployed SDN with OFS on its internal network. Operators have experimented with aligning application flows for data center interconnect (DCI) or user-to-server flows with optical paths using SDN and OFS or Multiprotocol Label Switching (MPLS). Operators are also looking to virtualize edge functions in set-top boxes (vSTB) or customer-premises routers (vCPE) using the SDN approach. Operators have begun specifying requirements for a specific set of such network function virtualization (NFV) in ETSI (www.etsi.org). Some NFV proponents suggest that most control plane functions should be applications running on server hardware in data centers.

I2RS aims to provide the type of flexibility operators expect from SDNs in parallel to (or in a hybrid mode with) the network-layer control plane. The I2RS framework draft states that I2RS’s design will provide applications with access to the dynamic information that routers have about topology, events, traffic, and status; and let applications augment routing control processes based on this information. I2RS development comes as one response to lessons learned from three SDN deployments, including Google’s with OFS, early experiments with SDN controlling flows directly over optical networks, and SDN deployment in virtual edge devices, such as vSTB and vCPE routers.

Lessons from Google’s SDN/OFS Deployment
In April 2012, Google announced the transition of its internal “G” network connecting its data centers from traditional protocols to an SDN-based network with OFS-controlled switches. Google uses open routing stacks with the Border Gateway Protocol (BGP), Intermediate System to Intermediate System (ISIS), or Open Shortest Path First to collect information from switches or routers to send to OFS controllers running the centralized control applications (see Figure 1). The OFS controllers then interface with the OFS-controlled switches, each with nonblocking 10-Gigabit Ethernet ports. After spending two years (from 2010 to 2012) developing this capability and four months deploying the network, Google hopes to see a 20 to 30 percent cost reduction. Google’s data centers and data center interconnects use cheap switches with sufficient numbers of machines to route around any failures. After successfully completing this deployment, Google turned toward its Internet-facing routers, which require 99.999 percent reliability, and asked whether SDN could reduce costs on these routers. Could a protocol be invented to leverage control plane software’s existing high reliability and provide a dynamic interface to the knowledge and control of the routing functions? Google’s questions led it, equipment vendors, and carriers to investigate a
new interface to the routing system (I2RS).

**Lessons from SDN and Optical**

Academics, service providers, and equipment providers met at the IEEE Image and Signal Processing and Analysis (ISPA) and IP+ Optical Network (IPOP) 2012 conferences to discuss early results on the network efficiency gains that an SDN approach could have when application traffic flows are aligned with optical paths. Work in the cooperative data processing models field has shown that aligning data processing, storage, and networking between data centers could theoretically drive total resource utilization from 60 to 98–99 percent. Operators and vendor experiments reported at ISPA and IPOP also showed 20 to 30 percent efficiency gains using experimental SDN controllers and OFS to align application traffic to optical paths. Because optical equipment such as Google’s external network devices requires high reliability, the operators who participated in this experiment have also been investigating using I2RS to control traffic at the optical layer.

**Lessons from Edge Networks**

In 2012, carriers also investigated virtualizing edge boxes (such as vSTB, CPE, and mobile backhaul). SDN-controlled trials on vSTB, home gateways, and CPEs convinced carriers and equipment providers that the services provided by edge devices would benefit from an SDN-style split in control and forwarding functions. Basic network services such as AAA, firewalls, DNS, load balancing, and security could be virtualized. In addition, new services such as traffic cleaning from spam traffic could be deployed in minutes rather than days with the SDN approach. One component of ETSI’s work on NFV is the refinement of requirements based on known use cases for edge devices.

**A Perfect Storm of Need**

By summer 2012, this perfect storm of requirements for a new interface into the current routing systems from disparate data sources — such as data centers, cloud computing, mobile operators, carriers, vendors, and academics — swept into standards bodies, including the IETF. The initial framework documents for the I2RS work arrived at the 84th IETF meeting in late July 2012. By October 2012, 12 documents describing the problem, framework, use cases, policy issues, and architecture were ready for review at the 85th IETF meeting, and an IETF working group was chartered in early 2013. This group will first collect requirements for the I2RS interface based on use cases and compare them against existing protocols. This gap analysis has encouraged the experts working on I2RS to aim at extending an existing protocol (or protocols) to meet these needs quickly. We next review one of these use cases and show how it fits into the framework I2RS is creating.

**Building I2RS on Use Cases**

Building new technologies such as I2RS is a process awash with “teachable moments” for the engineers involved. But technologies succeed because they’re used in the real world — so the engineer must always return to how the technology is applicable in real life. What’s the primary point of network design and management today? In (perhaps) abstract terms, it’s the dispersion of policy. As an example, consider the dispersion of policy in the routed control plane as regards security (see Figure 2).

An attacker sends a stream of traffic from someplace behind router G toward the attacked server. Network administrators can react to this attack in numerous ways:

- The administrator can configure filters at routers C and D to block the traffic.
- The customer can call the provider and ask that filters be configured at routers E and F. To save internal bandwidth, the provider might take the time to trace the flow back to the entry point into its network, and configure filters at router G.
- The customer might pass the traffic destined to the attacked server through a “scrubbing service,” either internal or external (as shown in the figure). This could require a complex set of changes to the DNS and routing systems.
Each approach illustrates how policy can be dispersed across a wide array of devices in response to a specific situation that’s most likely temporary.

By interacting directly with the local layer-3 forwarding engine and routing protocols, the I2RS controller provides a way to more quickly and easily disperse and manage the policy required to deal with the attack. If B is a detection system (such as an intrusion-detection system) that can also interact with I2RS agents located in the node, then the policies required for each possible response become much easier to disperse and manage:

- The detection system could inject a routing-table-based filter at routers C and D to block any traffic matching the attack’s flow label. This filter could be removed later, when the detection system no longer detects incoming traffic matching this flow label.
- The detection system could create a policy that attaches a policy to the BGP routes being advertised toward the provider (or using FlowSpec), asking the provider to block the traffic someplace in its network.
- If the right security protocols are in place, the detection system might even interact directly with the routers in the provider’s network, routing the affected traffic through a scrubbing service — and removing the modifications when the attack traffic is no longer sensed.

Security is, of course, only one of the many use cases driving I2RS development — but it’s also one of the most compelling.

I2RS Framework Documents

The use case we present illustrates the framework concepts that the I2RS documents specify — in this example, the user has a problem that the SDN/I2RS framework can solve by dispersing policy within the network and to the network edges quickly and dynamically. The I2RS problem state draft describes the requirement that network applications (such as the security application) should be able to access information on the events, network topologies, traffic information, and policies within a node to augment existing routing policy as regards flows or application traffic at specific times. To provide this information, the I2RS work will:

- create data models at the Routing Information Base layer (for example, Unicast RIB, Multicast RIB, or the Label Forwarding Information Base), router and MPLS protocol layer, state and event filters (at all layers), and policy;
- design a framework for integrating external data into routing; and design an interface to configure device- and network-level interfaces and protocols.

The I2RS framework draft suggests a structure of one or more I2RS agents residing in a routing process that might attach to a single I2RS client performing a function (see Figure 3). An example of the single I2RS client is a security-monitoring application. Multiple I2RS clients can attach to a single I2RS agent as client A and client B do in Figure 3. Client A can be the security intrusion-detection client, while client B reconfigures the pathways.

The multiheaded control that the I2RS clients (such as clients A and B) might have for a single I2RS agent or multiple agents requires that the I2RS protocol have a framework in which the user can specify the right policy for agent and client interactions. The I2RS policy framework specifies mechanisms on how to determine:

- a client’s data ownership or configuration;
- I2RS arbitration if multiple controllers attempt to enforce policy;
- hand-offs from one client to another when an agent is no longer connected;
- garbage collection of ephemeral state after all clients go away; and
- time qualities (start, persistence across reboot, and duration/state expiration).

Figure 3. Interface to the Routing System (I2RS) framework. This framework includes both a data model and potential protocols to support the use cases outlined in the various documents presented to the I2RS working group at the IETF.
The IETF community prizes running code and rough consensus. The perfect storm of interest in I2RS that we’ve described has inspired the community to experiment with extensions to existing protocols such as NETCONF/YANG and Forwarding and Control Element Separation (ForCES) to see whether they can be utilized to solve these necessary use cases. The working group is developing many documents as needed to describe the experiments and gaps among current protocols. The working group’s final output will include:

- an architecture (framework, security, and policy),
- tightly scoped use cases that the APIs will support,
- abstract data models that align with these use cases,
- requirements for I2RS protocols and encoding languages, and
- an analysis of other IETF protocols and encoding languages against the I2RS requirements.

The IETF I2RS working group is scheduling extra time to meet between IETFs to hasten firming up use cases and the discussion of key technical issues. The perfect storm of need is pushing the I2RS working group quickly to the shore of completion.

References

Susan Hares is VP Technology and Strategy at ADARA. Her research interests include network complexity, interdomain routing, and software-defined networks. Hares has an MA in organizational leadership from Regents University, a BS in computer engineering from the University of Michigan, and is a candidate in the Regent University Organizational Leadership PhD program. She’s active in the IETF, the cochair of the Interdomain Routing working group, a member of the IESG, and active in IEEE and OIF. Contact her at shares@adaranetworks.com.

Russ White is a principal research engineer at Verisign. His research interests include network complexity, design, and architecture; mobility; control plane security; the intersection between business and technology; and other areas of network control planes. White has an MSIT in network design from Capella University, and is currently working on an MAC in pursuit of a PhD. He’s on the IETF Routing Area Directorate, a member of the IEEE Computer Society, and holds a number of professional certifications in network design. Contact him at russw@riw.us.