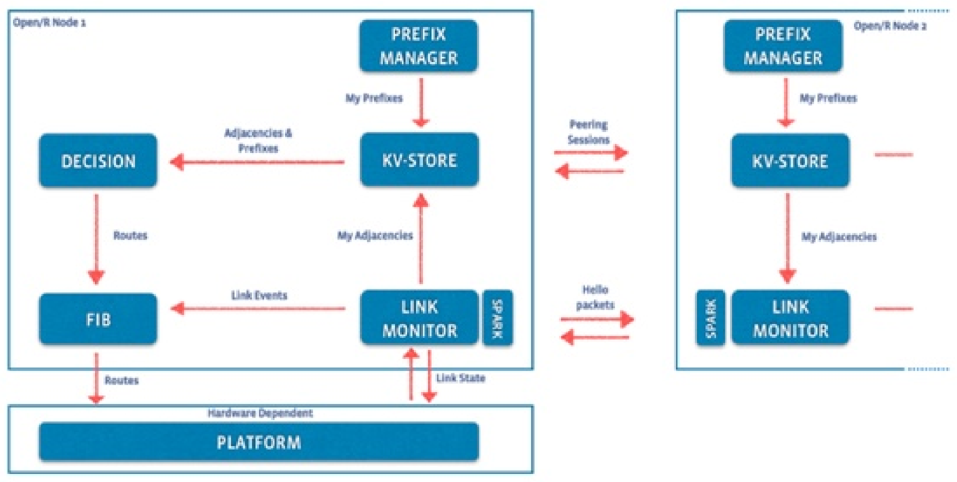
POSTED ON NOV 15, 2017 TO CONNECTIVITY, NETWORKING & TRAFFIC

Open/R: Open routing for modern networks



By Saif Hasan Petr Lapukhov Anuj Madan Omar Baldonado

- We are open-sourcing Open/R, an extensible network routing platform that enables rapid innovation in network functions and applications.
- Open/R is being used in Facebook's backbone and data center networks.
- The platform supports different network topologies (such as WANs, data center fabrics, and wireless meshes) as well as multiple underlying hardware and software systems (FBOSS, Arista EOS, Juniper JunOS, Linux routing, etc.).
- Open/R provides a platform to disseminate state across the network and allows new applications to be built on top of it. It also provides northbound interfaces to allow integration with external controllers.
- Open/R supports features like automatic IP prefix allocation, RTT-based cost metrics, graceful restart, fast convergence, and drain/undrain operations.
- We have been working with external partners and operators to support and use Open/R, and we invite more operators, ISPs, vendors, systems integrators, and researchers to leverage Open/R as a platform for implementing new network routing ideas and applications.

Open-sourcing Open/R

As more people come online and consume richer content, the complexity of the networks underlying the flow of information also grows. While traditional routing protocols have worked well over the past 30 years, it can be challenging and time-consuming to quickly push extensions or entirely new protocols into networking devices. New advances in routing often require carefully extending an existing protocol or designing new overlay extensions. Further, most protocols were initially designed based on constrained hardware and software environment assumptions from decades ago. To continue delivering rich, real-time, and highly engaging user experiences over networks, it's important to accelerate innovation in the routing domain.

Today, we are open-sourcing Open/R, a routing platform that enables rapid innovation in network functions. Open/R was originally built for the Terragraph wireless backhaul network, but we quickly saw its potential use for Facebook's backbone, where it provides more efficient and precise control of a global fiber network. Now, we are even starting to roll it out into our data center fabrics, running inside FBOSS and on our Open Compute Project networking hardware like Wedge 100. Open/R is a single platform that spans a wide variety of network domains and designs, and we are looking to apply it in even more use cases.

Facebook has taken a software-centric approach to the future of our network, and over the past several years we have been sharing our learnings with the network community. For instance, we have shared code from Facebook's open switching system (FBOSS); details about our network design, management, and monitoring tools such as

Robotron, FCR, and fbflow; and designs of our intelligent traffic control systems such as Express Backbone and Edge Fabric. Now, by releasing the modern, easily extensible, and well-tested Open/R platform, we intend to further push the state-of-the-art in the networking community.

In this spirit, we are excited to announce that we have been working with several external partners to bring Open/R to production, both inside the Facebook network and externally. Open/R now runs on Arista switching platforms, integrating with the open source EOS SDK, as well on the Juniper QFX and PTX routing platforms using gRPC-based APIs. WiLine Networks, a wireless ISP, has worked with Tieto to deploy Open/R into production in its commercial network. We invite other vendors, operators, ISPs, systems integrators, and researchers to leverage Open/R as a platform for implementing new routing domain ideas to make the next generation of networks more efficient, more intelligent, and easier to operate.

Applicability across different network domains

We were quite happy to discover that Open/R is flexible and extensible enough to handle many types of network technologies and topologies. It all started with Terragraph, which presented a challenge to traditional network protocols: thousands of wireless nodes communicating outdoors with weather, foliage, and other obstacles constantly interfering with the connectivity. A large Layer 2 mesh would not be as efficient and resilient, while traditional Layer 3 protocols, designed mostly for basic connectivity and reachability, are poorly suited to highly variable and dynamic outdoor conditions. “Control it all centrally” seemed like an easy design solution, but impractical given the lack of out-of-band management and large size of the wireless mesh. Thus, we designed Open/R to be the distributed networking platform on top of the Terragraph network that could cope with its large scale and rapid changes, and allow for fast extensibility.

When we initially shared that we were developing Open/R, several members of the networking community reached out to learn more, especially in the wireless space. Wireless network designers are all facing the same challenge: how to disseminate information about the underlying network quickly and frequently enough, given how fast wireless networks can change. WiLine Networks had started trying to modify existing protocols as the basis for its platform, but when representatives learned about Open/R, they and Tieto engaged with Facebook to see if Open/R could meet their needs. Their particular network design had some new, interesting challenges for Open/R, but together we were able to extend Open/R to accommodate the new requirements. Open/R is now part of their production network. In a similar vein, other providers of next-generation, millimeter-wave networks saw the direct applicability of Open/R, and we have started discussions with members of the Telecom Infra Project (TIP).

Internally, we saw that Open/R could address some of the challenges in the Facebook backbone network. While the backbone is an entirely different domain from a metro wireless mesh (the backbone has thousands of miles of fiber that span continents, cross oceans, and circle the globe), from a networking perspective it faces similar problems. Our new Express Backbone required control of traffic and routing across many widely different paths and rapid response to failures. Instead of using a traditional routing protocol such as IS-IS, we decided to introduce Open/R into the Express Backbone as the distributed control plane for the network, where it would work together with a centralized controller. Here, Open/R provides basic connectivity in a network, fast reactions to network events, and a “distributed information bus” to export new information to the controller and agents on the network nodes.

Most recently, we turned our attention to the Facebook data center fabrics — again, a domain very different from Terragraph and the global backbone. By design, the fabric is highly uniform and densely meshed, with many possible paths to account for and leverage. For many years, Facebook has been operating these large-scale fabrics solely with Border Gateway Protocol (BGP). While BGP brings its strengths, especially with respect to policy enforcement and scale, we saw opportunities to improve and simplify the design by having Open/R and BGP work together. Open/R in this part of the network is running on top of FBOSS and on our own OCP networking hardware such as Wedge 100.

Routing in a modern hardware and software environment

Many traditional routing protocols were designed in the past, with a strong focus on optimizing for hardware-limited embedded systems such as CPUs and RAM. In addition, protocols were designed as purpose-built solutions to solve the particular problem of routing for connectivity, rather than as a flexible software platform to build new applications in the network.

Today, CPU and RAM limitations are no longer a major bottleneck; at the same time, the application requirements and use cases that networks need to satisfy are rapidly evolving. These changes have shifted the focus of network engineers from optimizing protocols to building new applications quickly, which can be challenging to do on top of existing protocols. The Open/R platform enables the rapid innovation required to support modern network designs while reducing operational complexity.

We designed Open/R to run on multiple hardware platforms. All hardware- and system-dependent pieces have been abstracted out via RPC interfaces (for example, route programming and link/address discovery). This has allowed us to run the same Open/R implementation on different kinds of hardware devices, with different ASICs or even different CPU families. This flexibility provides a layer of homogeneity on top of heterogeneous network devices, making the network easier to control and manage.

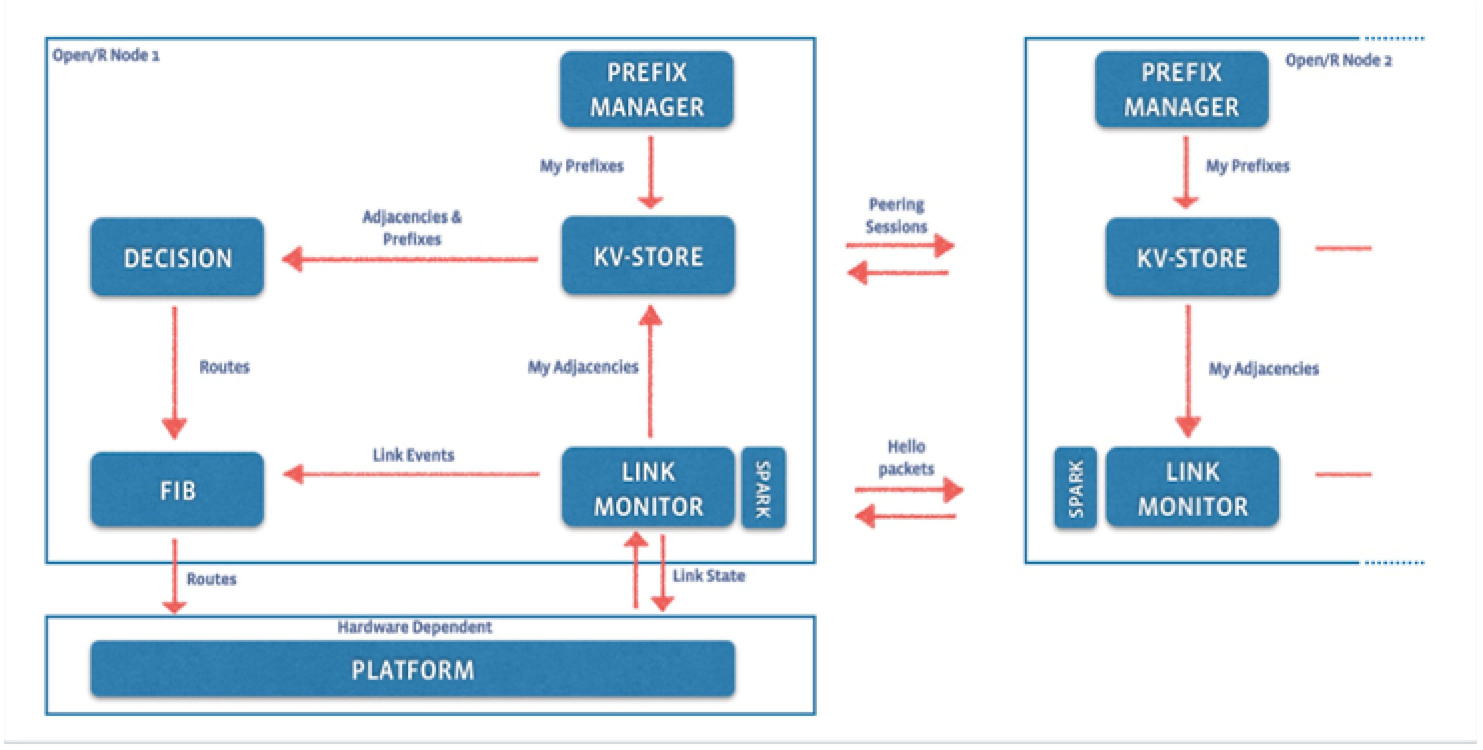
Designing for rapid iteration made it necessary to include robust testing procedures that provide confidence and clarity around every code change and new feature release. We strive to have 100 percent code coverage to validate every line of code logic and catch errors as early as possible. However, unit tests alone are clearly not sufficient. For every change in the code, we test Open/R's actual behavior in a large emulated network topology of thousands of nodes, running regression tests and checking the whole set of features to ensure backward compatibility. To do this, we built a framework to spin up large virtual topologies using lightweight containers as nodes across multiple physical machines. We will also be open-sourcing this emulation framework along with the Open/R platform.

Open/R design updates

As we've previously described, Open/R leverages existing open source codebases for two core functions. First, the encoding and decoding of control plane data structures is done using Thrift. Second, the logic to exchange data between network nodes is implemented using ZeroMQ. This design choice helped us eliminate a lot of technical overhead and develop Open/R quickly, concentrating on higher-level features rather than on the low-level plumbing. In addition, using existing technologies allowed us to create tools and libraries in languages other than C++ to interact with Open/R. An example is the Breeze CLI tool, which is written in Python and provides ways to interact with the different Open/R modules.

Since our initial post describing Open/R's architecture, several enhancements and new features have been added. One major architectural improvement is the introduction of the Platform module. It abstracts out system-dependent parts and exposes APIs to feed low-level information to other modules or to program the specific underlying hardware. For vanilla Linux nodes with software routing, it provides default handlers to address forwarding information base (FIB) programming and interface state change notifications for Open/R. For systems that are capable of doing packet forwarding directly in hardware (for example, our Terragraph nodes or Express Backbone routers), the module implements FIB handlers that interact with the specific hardware SDKs. For example, in our Wedge 100 platform, the FBOSS agent directly implements Open/R's FIB programming interface.

The following diagram summarizes Open/R's high-level architecture:



- KV-STORE: functions as a replicated key-value store that enables distributed communications and state replication.
- Spark: performs neighbor discovery on interfaces using Link-Local Multicast, and reports neighbor activities.
- LinkMonitor: monitors system interfaces via Platform, manages Spark sessions on those, and advertises discovered neighbors in KV-STORE.
- PrefixManager: performs automatic prefix suballocation for ad hoc configuration.
- Decision: computes routing information based on the topological information learned via KV-STORE.
- FIB: serves as a proxy for programming computed routes via Platform, maintains forwarding state (computed routes).
- Platform: implements route programming and interface discovery logic for the target hardware platform.

Emulation and CLI

Open/R testing relies on the ability to quickly set up large virtual networks without the need for actual hardware. To facilitate this, we built a generic emulation library — a lightweight wrapper over `systemd-nspawn` — and an emulation tool that starts thousands of Linux containers and forms a mesh of tunnels among them with a single command. To emulate packet loss, delay, or jitter, the tool supports the creation of different network environments, such as WAN or wireless networks, by configuring traffic conditioners. This makes it easy to test Open/R's behavior under different scenarios, and helps ramp up developers new to Open/R, providing a safe way to experiment.

The platform comes with the Breeze CLI, a command-line interface tool for network engineers to interact with the mesh of Open/R instances. It allows the inspection of running state information, such as adjacencies, nodes, prefixes, routes, counters, and latest convergence stats, as well as performing control operations such as draining and undraining nodes, setting custom metrics on links, and announcing and withdrawing prefixes. For example, a useful feature is the ability to perform real-time inspection of the platform's replicated state database (the KV-STORE), which allows engineers to inspect any activity in the whole network (nodes coming up, adjacencies tearing down, or metric values changing) just by connecting to a single node.

Open/R and centralized control

The distributed aspect is important for network reliability. Still, intelligent decisions are optimally done when they are offloaded to a centralized controller, such as for traffic engineering purposes. For that, Open/R provides APIs allowing remote agents to learn the link state or subscribe to database updates, such as notifications of a link capacity change. For example, this could be used to compute label-switched paths and program them on the network from a central location. This functionality allows the combination of "local" (in-network) event-handling with centralized logic, as we've implemented successfully in our Express Backbone project.

Summary of features

While Open/R implements many features similar to those found in traditional protocols like IS-IS and OSPF, it also introduces a few unique features of its own.

Here's a quick summary:

- IPv6 first, leveraging IPv6 link-local addresses to achieve zero-touch configuration. No special network configuration is required.
- Support for native IPv4 routing when needed.
- Ad hoc network prefix allocation and IP configuration for nodes in the network from a larger aggregate prefix.
- A graceful restart that enables live software updates without disrupting traffic forwarding.
- Support for draining and undraining for nodes and links.
- Dynamic link RTT metrics computed and smoothed out from active probes.
- The ability to set custom metric values, statically or dynamically.
- Fast network convergence with smart back-off timers for link or node failures.
- Continuous health checking of the network through live reachability probing.
- An API for integration with centralized controllers.
- A Python library to interact with all the main Open/R processes.
- The ability to extend the platform to disseminate all sorts of additional information, and even to introduce enhancements or variations to the path computation logic.

Conclusion

While traditional routing protocols have been instrumental to the progress of technology in the past few decades, we are approaching the point where networks need to evolve even faster. Open/R is an open platform that makes it easy to rapidly test and deploy new ideas at scale, making our networks more efficient, quicker to deploy, and easier to manage.

We strongly encourage network operators, researchers, vendors, engineers, and the overall networking community to use Open/R to implement their ideas and build modern networks that are more open and can evolve faster than ever.

And finally, a huge thanks goes to the many teams that worked with the core development group in order to get Open/R to this important milestone. This includes the Terragraph team, the Express Backbone team, Data Center Network Engineering, the FBOSS team, the Facebook Infrastructure Partnerships team, and our external partners.

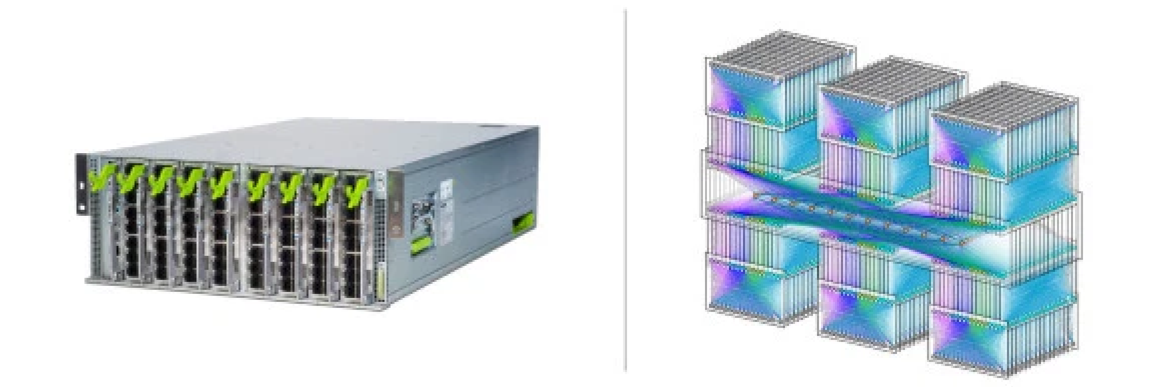
TAGS: OPEN SOURCE

Like Share Be the first of your friends to like this.

Related Posts

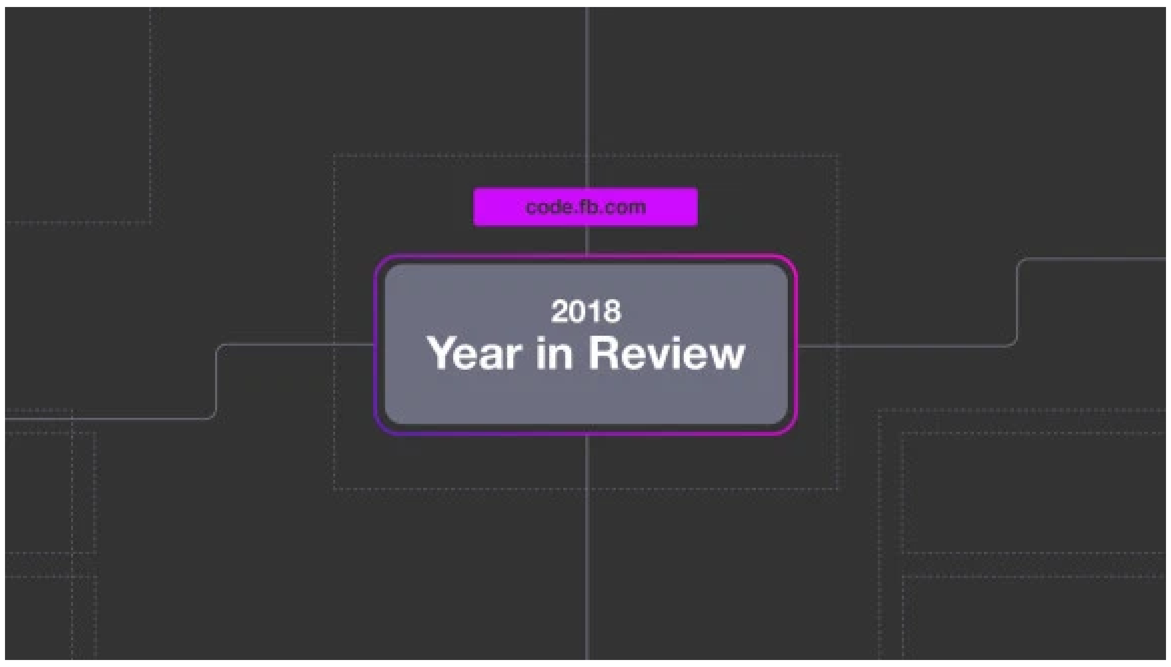
Dec 12, 2017

2017 Year in review: Better global networks



Mar 14, 2019

Reinventing Facebook's data center network



Jan 02, 2019

Open source year in review

Related Positions

Data Center Systems Thermal Engineer

MENLO PARK, US

Workforce Planning Program Manager, Mission Control

MENLO PARK, US

Architect, Data Center Design

DUBLIN, IRELAND

Enterprise Support Tech

DUBLIN, IRELAND

Data Center Facilities Engineering, Electrical Engineer

WASHINGTON, US

See All Jobs

Join Our Engineering Community

Available Positions

Data Center Systems Thermal Engineer
MENLO PARK, US
Workforce Planning Program Manager, Mission Control
MENLO PARK, US
Architect, Data Center Design
DUBLIN, IRELAND
Enterprise Support Tech
DUBLIN, IRELAND
Data Center Facilities Engineering, Electrical Engineer
WASHINGTON, US

See All Jobs

Stay Connected

Facebook Engineering
Like

@fbOpenSource
Follow

facebook research

Facebook Research
Like

Facebook Developers
Like

RSS
Subscribe

Open Source

Facebook believes in building community through open source technology. Explore our latest projects in Artificial Intelligence, Data Infrastructure, Development Tools, Front End, Languages, Platforms, Security, Virtual Reality, and more.

ANDROID
iOS
WEB
BACKEND
HARDWARE

Learn More

AboutCareersPrivacyCookiesTermsHelp