POSTED ON NOV 19, 2015 TO DATA CENTER ENGINEERING, NETWORKING & TRAFFIC

Open networking advances with Wedge and FBOSS

By Jasmeet Bagga Zhiping Yao

Adaptability and customization are not typically what comes to mind when people think about network switches. Hardware is often proprietary, and if you're buying a vendor switch, you don't control the frequency or speed of new features or bug fixes. These constraints are inconvenient at best, particularly for large production environments. In the past two years, the Open Compute Networking Project has made big strides in disaggregating networking hardware and software and creating a vibrant open networking ecosystem, enabling faster innovation, and creating better efficiency in data center deployments.

For our part, last year we introduced our top-of-rack (TOR) network switch, Wedge, along with FBOSS, Linux-based software for controlling switches. Thousands of Wedges have now been deployed throughout our data centers, and we're pleased to announce that the Wedge design was recently accepted by OCP. Wedge is generally available for ordering by anyone in the networking community, and its designs are open for all to review and customize.

Wedge: Fully open

We created the most open design package to date for Wedge by defining and engineering the hardware of the switch, and then contributing everything needed for anyone to rebuild, re-engineer, or manufacture the switch. To get to this point, we underwent a full engineering effort, from compiling detailed specifications to developing the mechanics and board design. The team went through several design iteration processes to validate the design quality, including thermal design and mechanical testing, and qualified the Wedge design to fit into our data center.

We have submitted all the details needed for any contract manufacturer (CM) to produce the switch. We submitted a detailed Bill-of-Material, the schematic, the Cadence Allegro CAD files, the Gerber files, and Native Mechanic design files. Based on this information, it is possible to essentially manufacture this switch with any CM or ODM. Anybody can take the design, modify it, and resubmit it to OCP. This fully open approach enables the community of OCP contributors to benefit jointly from a state-of-the-art design while focusing on iterative improvement.

Getting ready to scale: “Operationalizing” Wedge and FBOSS

Unlike with traditional closed-hardware switches, with Wedge anyone can modify or replace any of the components in our design to better meet their needs. In the past year, we have scaled our Wedge and FBOSS deployment in production at Facebook from a few switches to several thousand. To get to that point, we had to make continual adjustments and improvements — all of which contribute to the reliability of Wedge and FBOSS for anyone who is interested in using them.

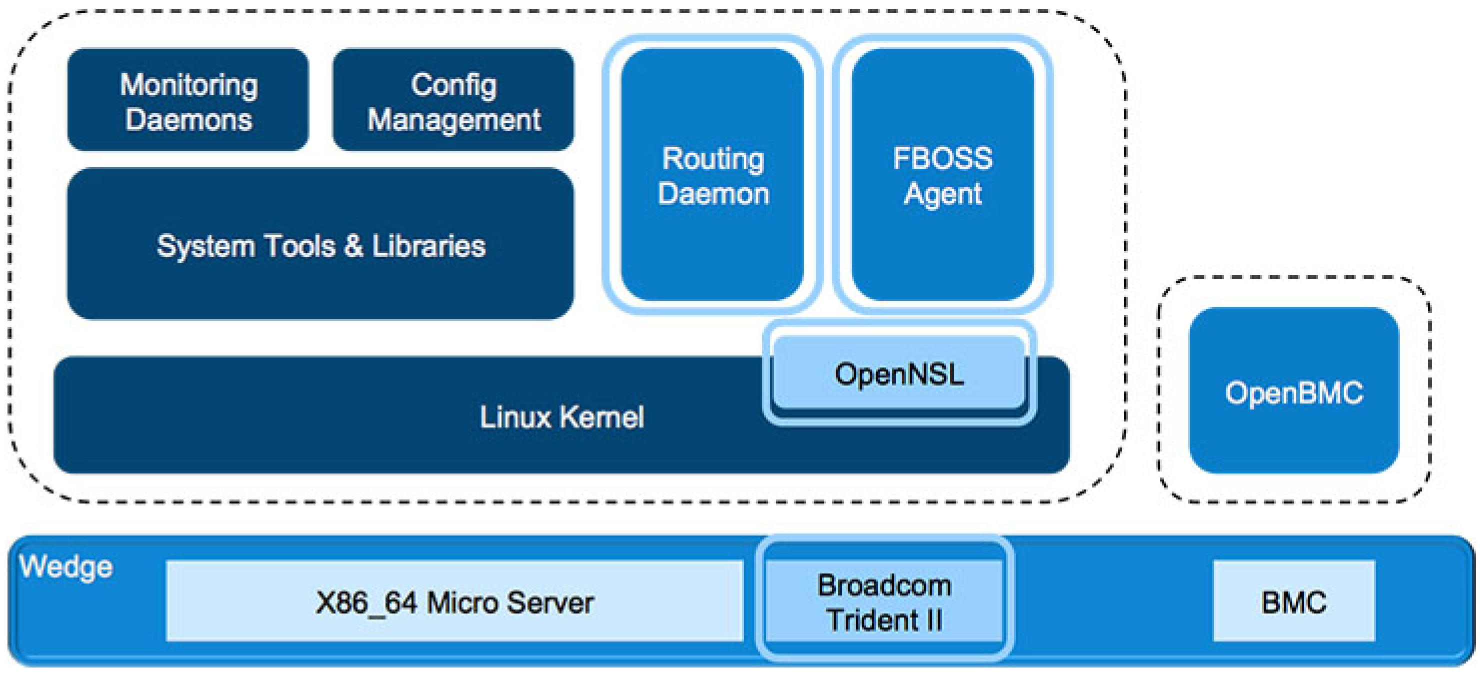
Customizing software to increase flexibility and reliability

We developed FBOSS to handle a weekly cadence of new features and bug fixes. This pace is typical for many Facebook software services. To achieve this, we had to make sure we could update thousands of switches seamlessly, without any traffic loss that could cause outages:

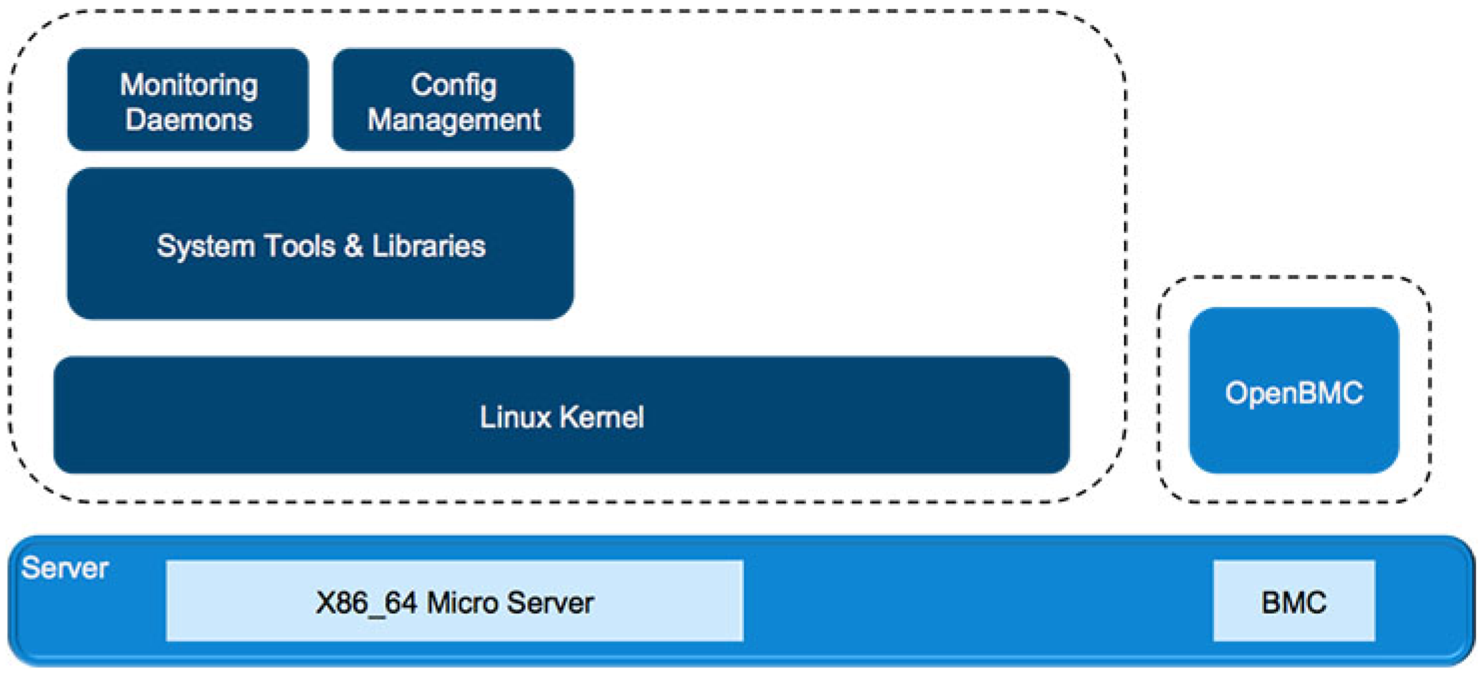
- **Detailed monitoring.** We custom-built fbossdeploy, an internal tool that conducts weekly FBOSS code updates, by basing it on a similar tool we built for our software load balancer, **Proxygen**. We monitor for problems as code is slowly pushed to an initial set of switches — known as canaries — before starting a wider software update. We learned that monitoring only before and after a push is not enough. Some failures don’t show up immediately: For example, if there were bugs when restarting an FBOSS agent, they might not show up until the FBOSS agent synced its state to hardware. Also, not all failures are caught by checking rack reachability. Some of our Facebook services rely on announcing **virtual IPs** (VIP) as part of their operation. The VIP is announced via BGP peering between the server and rack switch. If there were a bug in our routing daemon whereby on restart it suddenly no longer announced the VIP, we wouldn’t be able to catch it just by checking server reachability. All this pointed to a need for doing more in-depth service-level monitoring during the rack switch software upgrade process.
- **Nonstop forwarding.** Warm Boot, a feature provided by the Broadcom ASIC, ensures that traffic forwarding continues to work while the FBOSS agent is restarted. Once it comes back online, data structures are re-created exactly as they were before. We didn’t get to this outcome right away, and in fact, our detailed monitoring showed brief error spikes when we first did weekly updates. After we studied these short outages rigorously, we achieved reliable lossless forwarding in all configurations.

Lessons from treating a switch like a server

Many network operators we’ve spoken with share our goal of managing our networking devices and servers in the same way. With that goal in mind, we designed both Wedge and FBOSS to closely resemble server hardware and server software, which has enabled us to deploy, monitor, and control these systems alongside our servers and storage.



Switch System Overview



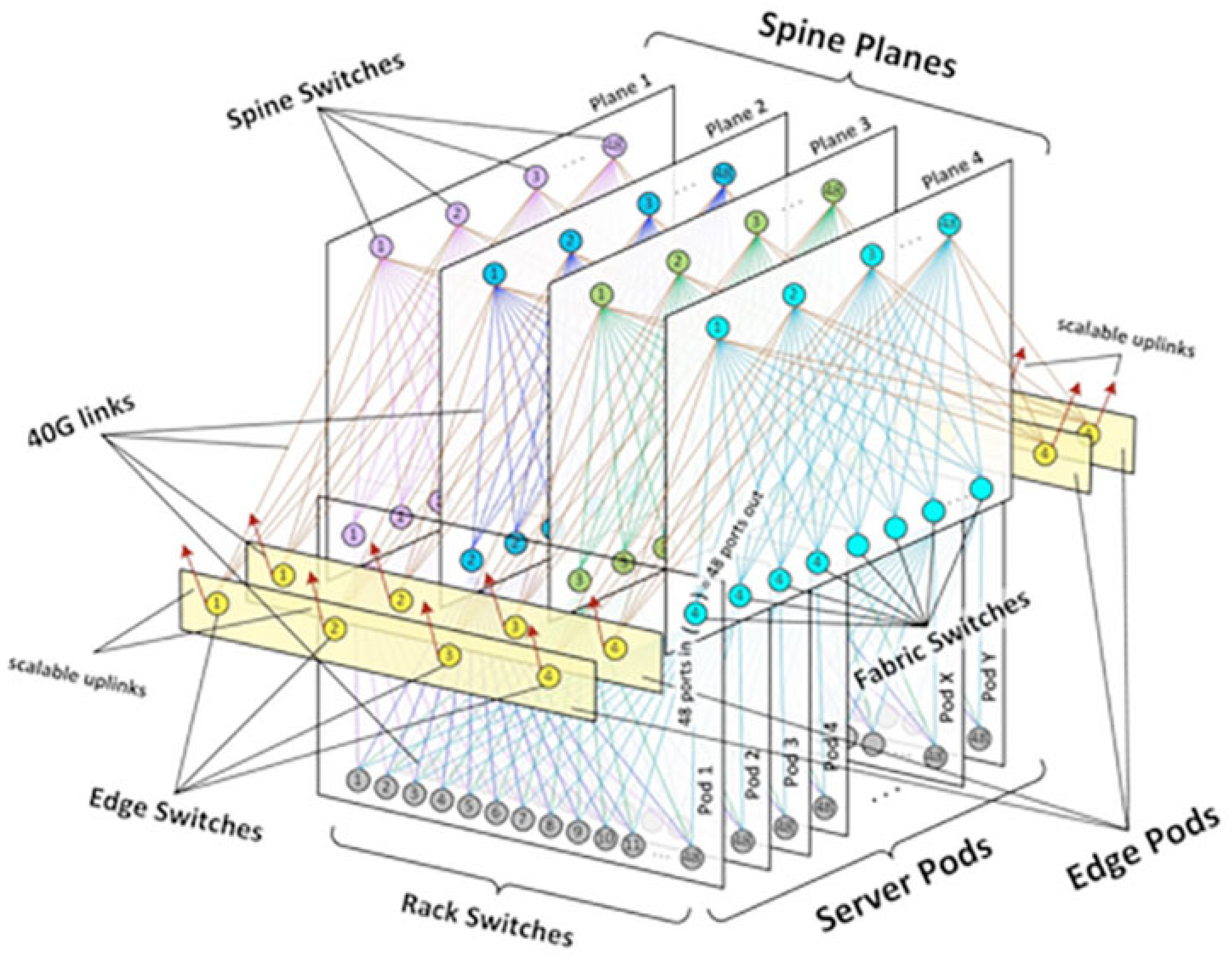
Server System Overview

While this has been a huge win for us, we also learned some good lessons early on in the process:

- **Keeping Wedge at the core.** One issue we encountered centered around our use of common service infrastructure to manage processes on the switch. This service infrastructure is updated and rolled out frequently, like all Facebook services. Those rollouts are done in a controlled manner, meaning they get pushed to a very low percentage of servers and then gradually to the rest of the fleet. What we discovered was that not all of those teams updated their rollout process to include Wedge switches before pushing to all hosts on Facebook. Eventually, a Wedge-specific bug crept into the process management service, causing it to start crashing on Wedges. From our work addressing the bug, we learned that Wedge switches need to be part of the early rollout of all common service infrastructure that FBOSS uses.
- **Circular dependency.** The aforementioned outage also brought to light a circular dependency between some of the services that FBOSS uses and the switch itself. These services on startup would try to communicate with the master nodes in their region, but if all switches in the region were down and the FBOSS switch needed this service to run either the FBOSS agent or the routing daemon, we would be deadlocked. Now we make sure that there are no circular dependencies with any of the services that FBOSS depends on.

Protecting the CPU

An often-seen failure mode in switches occurs when the CPU is overloaded to the point where it falls behind on its control plane duties, like sending routing protocol messages or programming the ASIC, which is when traffic outages can appear. In our case, all our new data centers are based on our next-generation fabric design, which is a high-density Clos network deployed to meet our unprecedented growth challenges when building infrastructure at scale. At the heart of this design is the three-tier architecture shown below.



Early on during our testing, we ran into several scenarios that showed just how important it is to protect the CPU and the control plane. We have now configured hard limits on the amount of traffic that gets sent to CPU from the switch ASIC. Further, we prioritize control plane traffic to the switch to ensure that the control plane of the fabric is up as much as possible. Here is just one of the debugging scenarios we went through as we scaled out.

The route aggregation in the fabric is designed such that fabric switches announce an aggregate for the rack switches underneath. This is essential for keeping our routing table sizes in check. The rack switch has uplinks to four fabric switches, and if a link between a rack switch and a fabric switch goes down, the fabric switch continues to get traffic for this rack subnet because of the aggregate route, causing that traffic to get blackholed. To work around this problem, we arrange rack switches in chunks called backup groups. Each rack switch in a backup group backs up its peers by announcing the peers' subnet to fabric switches with a different BGP community, and thus provides alternate pathing in case of link failure.

This works well with one link down, but if an entire rack switch goes down, the backup peers soon notice the withdrawn subnet and withdraw the backup route as well. However, before all backup peers withdraw from this route, there is an extremely transient routing loop that continues until the packet TTL expires (or maximum hop count is hit for IPv6 packets), at which point the packet is sent to the switch CPU. At our data rates, even a brief burst like this is too much for the CPU, and soon enough we are no longer even able to keep up BGP peering sessions, exacerbating the situation.

Now that the CPU has been protected, any situation that tries to send too much traffic to the switch CPU — including this one — is no longer a problem.

Looking ahead

As exciting as the Wedge and FBOSS journey has been, we still have a lot of work ahead of us. Eventually, our goal is to use Wedge for every top-of-rack switch throughout our data centers.

We've been working with a number of vendors and operators to help them start using Wedge and to share our experience with deployment of OCP switches in order to create a bigger open source ecosystem in networking. For example, Big Switch Networks now provides an Open Network Linux image that includes everything you need to run FBOSS on a Wedge and start programming it: http://opennetlinux.org/wedge.

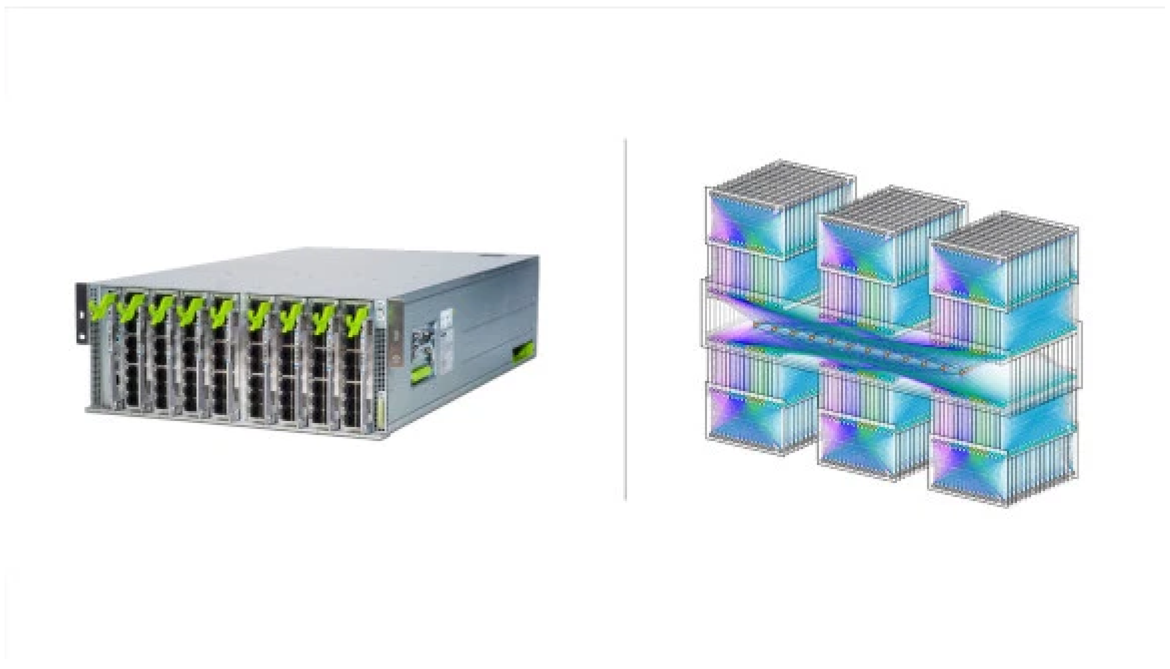
We also don't plan to stop with the rack switch. Work is already underway in scaling our software to operate at higher speeds and handle higher complexity. We are working on Wedge 100, a 32x100G switch that we are looking forward to sharing with the networking community soon. This is in addition to our work adapting Wedge to a much bigger aggregation switch called 6-pack, which uses Wedge as its foundation and stacks 12 of these Wedges in a modular and nonblocking aggregation switch. FBOSS as a software stack runs across our growing platform of network switches: Wedge, 6-pack, and now Wedge 100.

Thanks to the entire Wedge and FBOSS teams for all their hard work getting us to this scale so quickly. We're looking forward to sharing even more with the community.

TAGS: HARDWARE INFRA OPEN COMPUTE

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

Related Posts

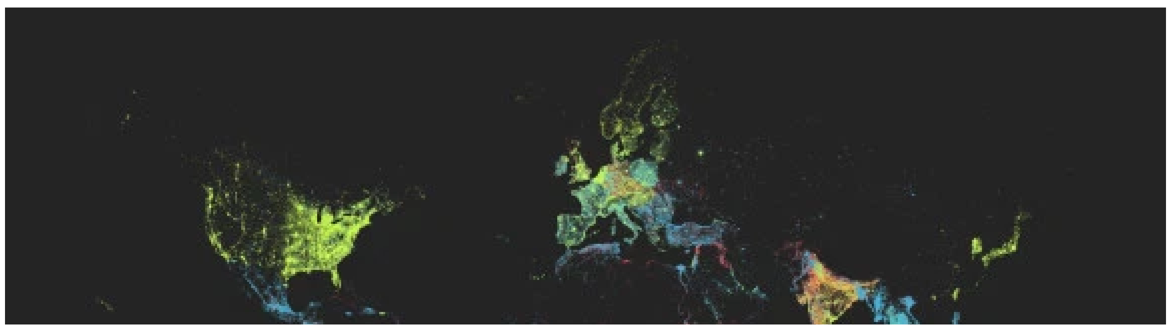


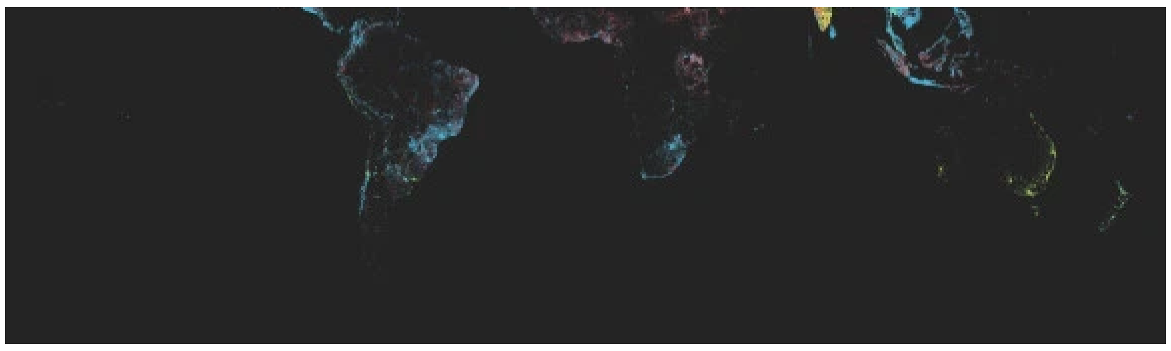
Mar 14, 2019

Reinventing Facebook's data center network

Sep 04, 2018

FBOSS: Building switch software at scale and in the open





Dec 12, 2017

2017 Year in review: Better global networks

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

Like

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

Facebook Developers

Like

RSS

Subscribe

AboutCareersPrivacyCookiesTermsHelp