Carrier Networks for 'Cloud-Natives'

Why is everything so hard?

Times are challenging for carriers. IP networks are complex to run. Costs are increasing. And it's hard to add, test and automate new services.

But it's not like that for the big cloud companies - these *'cloud-natives'* have found new ways to operate at web-scale. For example, AWS can add as much new infrastructure in one day as it had in its entire operation when it was a $7B revenue business. One of their engineers can operate 10,000 servers. Or they can drop in a new microservice whenever they want one – like simply adding Facebook Chat for 2 billion existing users.

Of course, SDN promised to solve all these problems for the network. But it's proving hard to migrate existing networks and harder still to make SDN robust enough at carrier-scale.

So, where does RtBrick come into all this?

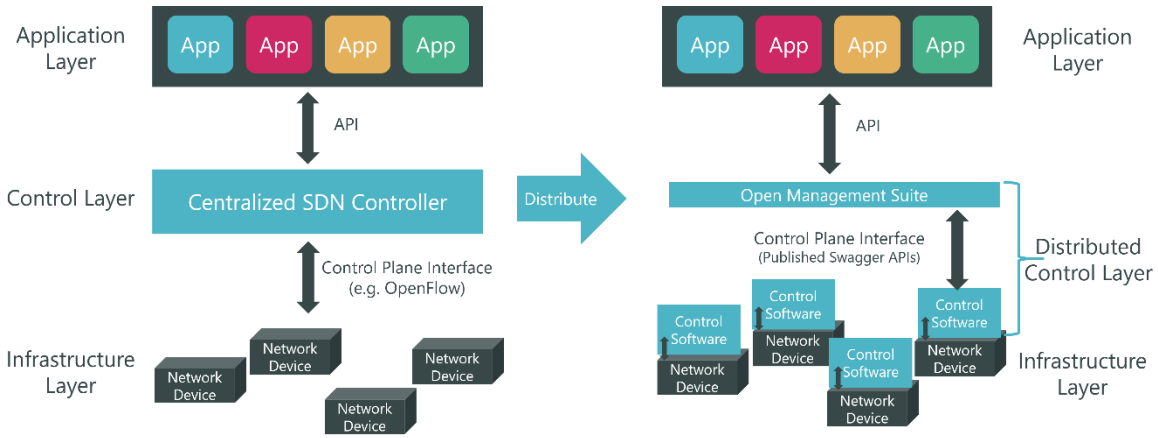
We thought, why can't we build and run *networks* like cloud-natives. So, we designed carrier routing software that is more robust, more controllable and costs far less than a conventional network. It allows you to build a *distributed* Software Defined Network, that delivers at carrier-scale.

> "The focus is on having a more open, more flexible solution. The whole story is about disaggregation of hardware and software. RtBrick brings a piece of software that sits on the switches. They have built software that can program Broadcom chips, so they do the full BNG function."
>
> Robert Soukup, Deutsch Telekom's senior program manager for Access 4.0

Why distribute your SDN?

Distributing the SDN control layer is key. Firstly, it achieves all SDN's objectives:

- Agile – it's still easy to add and test new services
- Highly programmable – every network component is exposed to higher level systems
- Lower cost – because it uses low-cost bare-metal hardware and operations can be automated



Centralized SDN | Distributed SDN

But in addition, it's also:

- Lower risk – a highly distributed control system is more robust and can contain any 'blast radius'
- More scalable – unrestricted by the I/O limits of a single controller
- Easier to operate and migrate - network elements can work side-by-side with legacy routers, and can even be controlled by a classical SDN system

How do we make this work?

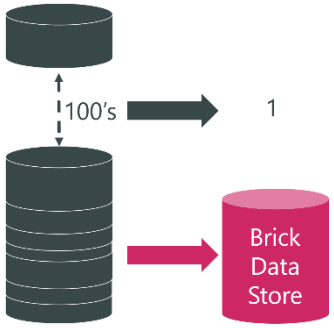
We've done three fundamental, new things that have changed everything:

1. We've used one 'state database' in each network element (separated from run-code)
2. We've created modular code – built from simple blocks running microservices
3. We've used a 'cloud-native' approach design principles, with code running in Linux containers and using Web 2.0 tools

One 'state database' – a massive simplification

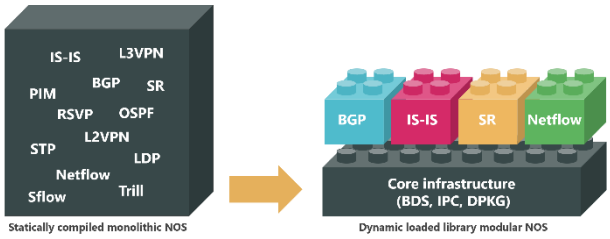
In the past, limitations in silicon performance meant that routing code had to optimize for CPU and memory utilization. This led to hundreds of different state databases in each router, each optimized for a different job. But we've replaced all of these with a single database in each router, operating independently from run-code. This has brought a huge gain in simplicity.

- ~1000 times fewer database interactions to manage
- It's more robust - everything is versioned and preserved in one place, enabling fast restart and re-sync of state
- We can scale by using hundreds of processor cores - this is how the cloud-natives scale (Amazon uses 128 cores in a server). Even mobile phones can do that, yet traditional routers can't!
- It's easier to program, with every element exposed to higher level systems via a single interface
- And this simplicity makes it possible to automate the testing of every feature to reduce the risk of bugs



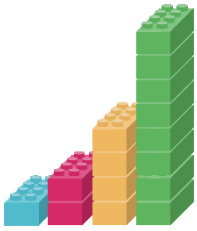
Modular code – deploy *only* what you need, test *everything* you use

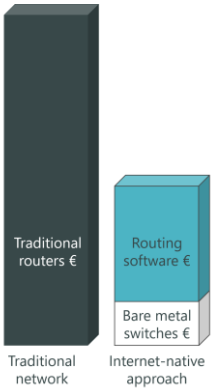
We've taken an approach to our software that is native to virtual IT environments. You get your own unique code-version, built from discrete blocks and delivered as a Linux container. Your code is no longer bloated with features that you don't use. This allows us to upgrade or add each feature independently, in sub-milliseconds, with no service disruption. As a result, route-processing, updates and restart is 20x faster than JUNOS or IOS-XR. And you can test individual microservices, rather than having to test the whole system as a 'black box'.



From Carrier-Grade to Web-Scale – the cloud-native approach

Rather than try to build an unattainable flawless system, we've taken a web-scale approach and focussed on self-healing systems. Our composable code and single state database makes it easier to contain risk. For example, we can isolate different routing universes (such as public and enterprise networks), watchdogs can detect issues and redeploy software if needed, and it's inherently simpler to control, as we've replaced a 'full-mesh' of logical relationships between the control system and many network element databases with single links.





Resulting in a simpler, safer, lower-cost operating environment

You can manage your whole network from a single browser window. And you get automation straight out-of-the-box. Simply power-on your bare metal switches and they will self-register, download the correct software image, discover their topology and all microservices self-start. Automated software version management reduces risk of human error, and new microservices can be updated and tested independently of other operating software. The system also includes tools to simplify repair, such as 'clone a switch' for zero-touch replacement of broken hardware. And there's in-depth alarm and status management, with thresholds and abnormalities automatically flagged: from data rates to fan speeds. You can glue this into your existing OSS using REST and RPC interfaces, such as gNMI (Google Network Management Interface). You can expect the capital and operational costs to be significantly lower than traditional systems.

Transform your carrier into a 'cloud-native'

So now you can have everything SDN promised, and more…

- You can migrate to a simpler, safer network
- You can operate your network at web-scale
- And you can operate at cloud cost-levels

Why don't you get in touch to find out more about RtBrick?

RtBrick is a privately held company, with staff in India, Germany, Austria, Belgium, Netherlands, UK and the USA. Investors include Deutsch Telekom Capital Partners and Swisscom Ventures.