Meet re6stnet, Nexedi's open source IPv6 network in production in the cloud - since 2012.

June 23, 2015, written by Klaus Wölfel and Sven Franck

Today Mark Shuttleworth announced a new workaround for the IPv4 address space shortage by claiming that *"IPv6 is nowehre[sic] to be seen on the clouds"* (source).

Introducing re6stnet

This blog post is about re6stnet (repo, spec, on pypi), our open source network, which solves the address space shortage problem by providing IPv6 on top of existing IPv4. Re6stnet is in production at Nexedi and clients since 2012 and is also used in our open source cloud-stack SlapOS. We did not write much about re6stnet until now, but thought this might be a good opportunity to explain what it is and how it works.

Wikipedia on IPv4 address exhaustion

"IPv4 address exhaustion is the depletion of the pool of unallocated Internet Protocol Version 4 (IPv4) addresses, which has been anticipated since the late 1980s. This depletion is the reason for the development and deployment of its successor protocol, IPv6." The problem is particularly pressing in the context of cloud computing, where hundreds of services can be instantiated in a short time and want to be accessed individually from the internet. While different short- and midterm mitigation efforts exist, the only long term-solution to IPv4 is the deployment of IPv6 (source).

This is the reason our open source cloud-stack SlapOS was designed for IPv6 from the ground up. Since IPv6 was and still is not available everywhere, we had to find a solution to provide reliable IPv6 to machines running SlapOS.

Welcome re6stnet!

Re6stnet is an IPv6-based low latency overlay network. It creates a resilient, scalable IPv6 network "on top" of an existing ipv4 network by creating tunnels (connections) on demand and then routing targeted traffic through these tunnels. We use re6stnet to give IPv6 adresses to machines where only IPv4 is available and more importantly, to guarantee connectedness between computers which have existing route connections to mitigate possible failures of the direct route. So one could say we basically use it to establish our own routing system on the top of DNS routing.

How is routing done?

re6stnet routing technology shares similarities with peer-to-peer (P2P) technologies such as Skype, PPStream or bittorrent. We initialize a mesh of direct routes between selected edges in a network and leverage the mesh to create indirect routes between all edges. Thanks to the babel (RFC 6126) distancevector routing protocol, the best route is always selected to interconnect edges.

This way re6stnet allows to create very large, stable and private networks as base architecture for our applications and implementations (side note: to ensure resiliency we put part of our test (and live) system servers in hard to reach places and countries).

re6stnet network structure

A re6stnet network consists of at least one server (re6st-registry) and many nodes (re6stnet). The server is only used to deliver certificates for secure authentification in establishing tunnels, and to bootstrap new nodes. Re6stnet can detect and take into account nodes present on the local network and guarantees - that if there is an existing route between two machines - traffic will be correctly routed between these two machines. Even if the registry node is down, the probability that the network isn't connected is very low for big enough networks (more than a hundred nodes).

Benefits from using re6stnet

Well, most importantly we have stable IPv6 everywhere - including on IPv4 legacy networks. Besides that we use it to optimize routing for low latency or high throughput based on application need and to aggregate bandwidth from multiple ISPs for both uplink and downlink among others.

To conclude and come back to where we started, there is IPv6 in the cloud. It's been there for a while. It's open-source and it works quite well. If you are interested, check out the repo or get in touch.