ExaBGP Introduction and Real Life Use Cases

Trex Workshop 2016

nebula

Quick intro

- Me
  - Anton Aksola (aakso@Twitter,IRCNet,Github)
  - Network Architect / Software Developer @Nebula Oy/AS29422
  - Working with OpenStack and related cloudy things
  - Doing software development around OpenStack (patches and features)
  - Previously worked as a network engineer for about 10 years

  - Not affiliated with Exa Networks

nebula

ExaBGP

- ExaBGP is a highly flexible BGP speaker that allows you to control BGP announcements programmatically
- It can also receive BGP Updates from peers and feed those to your application in parsed form
- Takes care of all the details running BGP State Machine, keepalives and protocol encoding
- Supports IPv4/IPv6, L2VPN, L3VPN, FlowSpec etc.
- You can concentrate on your application and enjoy technologies such as anycasting

nebula

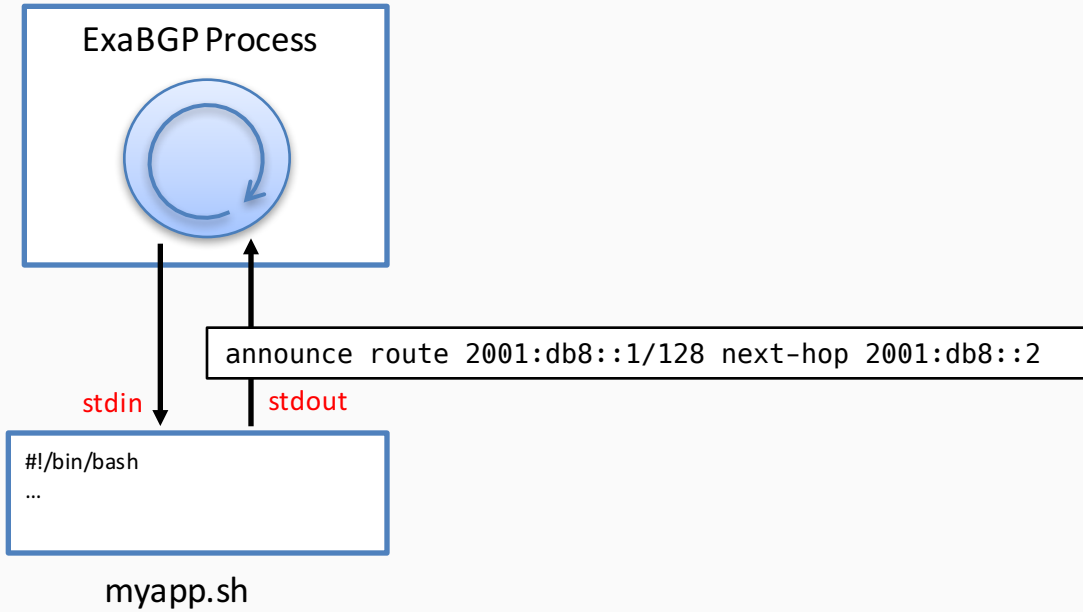
ExaBGP

- ExaBGP should not be compared to Quagga or BIRD as ExaBGP **does not** interact with the operating system to apply any announcements to the routing table
- Currently it doesn't support any Route Server or Route Reflector features
- Focus is clearly in delivering easy BGP interface for applications and scripts to consume

nebula

Basic Example

- Integration is done by configuring your application in ExaBGP's configuration file
- ExaBGP starts your app as a subprocess
- Unix Pipes are used to do IO
- You can announce/withdraw routes, add/remove neighbors, ask state etc.

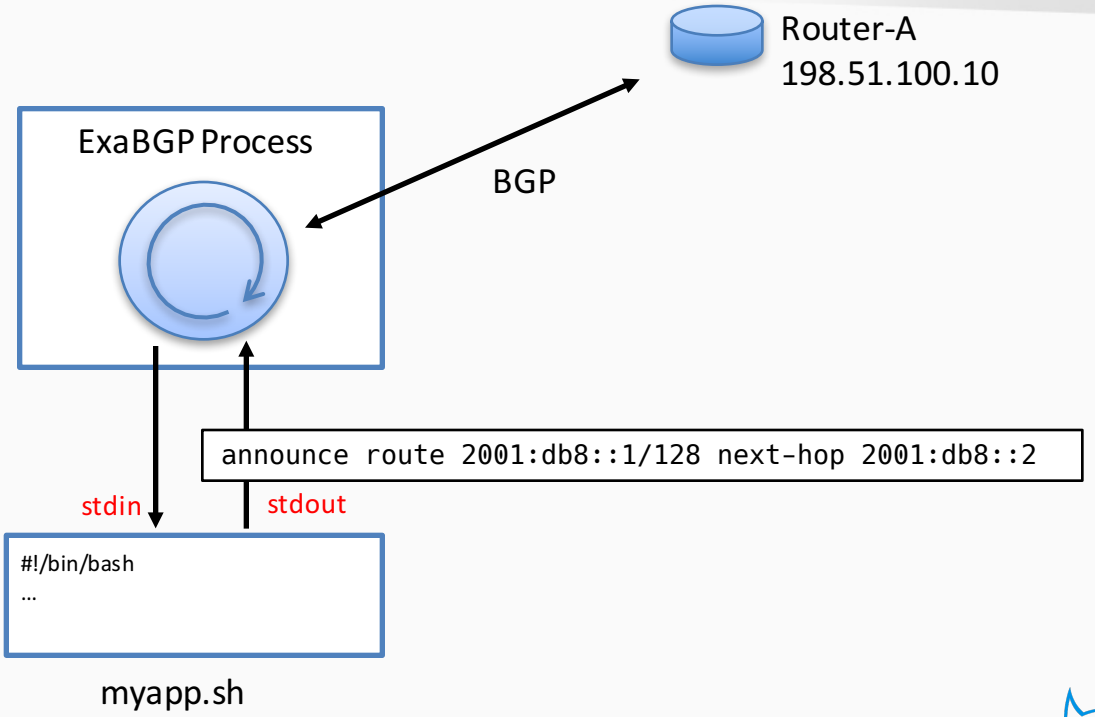


nebula

Basic Example

```
neighbor 198.51.100.10 {
      router-id 1.1.1.1;
      local-address 198.51.100.1;
      local-as 65510;
      peer-as 65511;
      family {
            ipv4 unicast;
            ipv6 unicast;
      }
      process myapp {
            run myapp.sh;
      }
}
```

ExaBGP configuration



Use case – Anycasted Service

- The most obvious use case is to use ExaBGP to announce reachability information for our app
- The app would be served from an IP address that is being announced potentially by multiple ExaBGP instances in different servers
- We would put that IP address to our loopback interface
- We would monitor the application from the script that ExaBGP starts
- The script would announce/withdraw the address according to the state of the app

nebula

Use case – Anycasted Service

Helper Script – monitor.sh

```
#!/bin/bash

IP=$1; shift
NEXTHOP=$1; shift
CMD=$@

state="DOWN"
while true; do
     sleep 2
     if $($CMD 2>/dev/null 1>/dev/null); then
          if [ "$state" == "DOWN" ]; then
               echo "announce route $IP next-hop $NEXTHOP"
               state="UP"
          fi
     else
          if [ "$state" == "UP" ]; then
               echo "withdraw route $IP next-hop $NEXTHOP"
               state="DOWN"
          fi
     fi
done
```

nebula

Use case – Anycasted Service

Configuration – monitoring a web server using curl

```
neighbor 198.51.100.10 {
     router-id 1.1.1.1;
     local-address 198.51.100.1;
     local-as 65510;
     peer-as 65511;
     family {
          ipv4 unicast;
          ipv6 unicast;
     }
     process monitor-myapp {
          run monitor.sh 192.0.2.1/32 198.51.100.1 curl –s –f http://localhost;
     }
}
```

nebula

Use case – Anycasted Service

- The script I showed is just for illustration
- For real stuff you should use the healthcheck module that comes with ExaBGP
- It provides more configurable behavior and more features
  - med adjustment instead of withdrawal
  - hysteresis mitigation with rise/fall values
  - loopback address population
  - hooks that execute on state change (for alerting and reporting)
  - logging

nebula

Use case – Anycasted Service

Configuration – using healthcheck module

```
neighbor 198.51.100.10 {
     router-id 1.1.1.1;
     local-address 198.51.100.1;
     local-as 65510;
     peer-as 65511;
     family {
          ipv4 unicast;
          ipv6 unicast;
     }
     process monitor-myapp {
          run python –m exabgp healthcheck --cmd ”curl –s –f http://localhost”
--ip 192.0.2.1 --next-hop 198.51.100.1;
     }
}
```

nebula

Our experiences

- Our company has been using ExaBGP and healthcheck module for several years in multi-datacenter deployments
- ExaBGP seems to be very stable and we haven't had any real issues with it
- The fact that it has no other dependencies than Python simplifies the deployment
- It by default doesn't run or require a TCP listener (you can enable it though)
- Recently we had a new project that also required anycasting so ExaBGP was again used

nebula

Case study - NELB

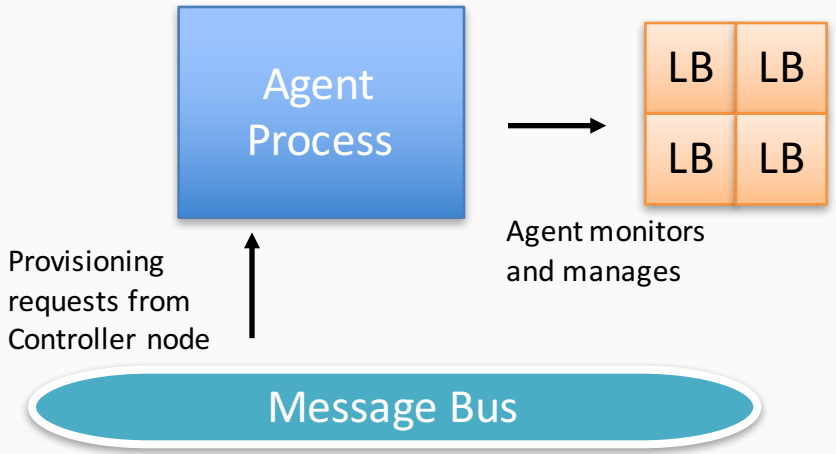
- Nebula Elastic Load Balancer (Load Balancer as a Service)
- Distributed system with public API, controller and agent nodes
- Dynamically allocated IPv4 and IPv6 addresses need to be announced to PE routers
- Load Balancers can come and go so routes need be quickly announced and withdrawed
- Previous state must be recovered in case of a failure

nebula

Case study - NELB

Agent Node

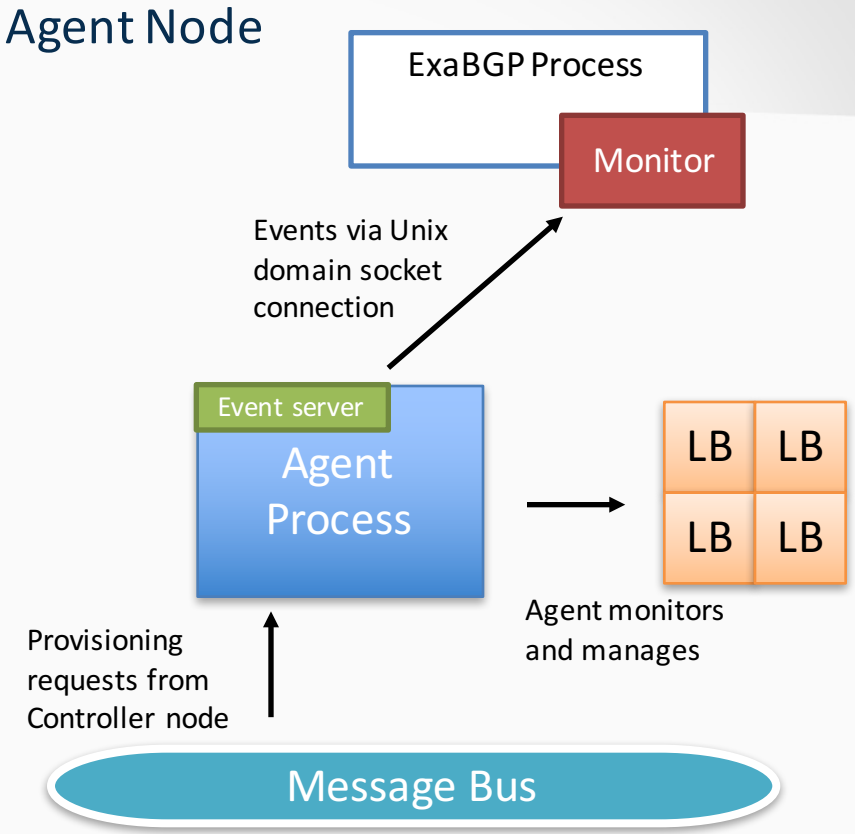
- Each load balancer has minimum of 3 IP addresses
- Agent monitors and locally repairs load balancers if they crash
- Provisioning requests come from the Controller which has full view of the whole system
- How to integrate ExaBGP into this system?



nebula

Case study - NELB

- We decided to run ExaBGP on each Agent node
- Agent process has the local state so we need another process ExaBGP can start (the monitor)
- Agent and ExaBGP should be loosely coupled so we needed an event system
- The monitor process translates events to ExaBGP commands



nebula

Case study - NELB

Agent Node

```
/build/bin$ ./monitor --monitor.outputformat=exabgp --monitor.exabgp.ipv4nh=198.51.100.1 --
monitor.exabgp.ipv6nh=2001:db8::2 --monitor.socket=unix:///run/nelb_events.sock
DEBU[0000] client starting          component=client pkg=agent/monitor
DEBU[0000] connected                action=connect component=client pkg=agent/monitor
DEBU[0000] start worker             component=client pkg=agent/monitor worker=connection
DEBU[0000] start worker             component=client pkg=agent/monitor worker=output_exabgp
announce route 192.168.0.1/32 next-hop 198.51.100.1
announce route 192.168.0.0/32 next-hop 198.51.100.1
announce route fc08::b0d:185f:5e9f:6261/128 next-hop 2001:db8::2
withdraw route 192.168.0.1/32 next-hop 198.51.100.1
withdraw route 192.168.0.0/32 next-hop 198.51.100.1
withdraw route fc08::b0d:185f:5e9f:6261/128 next-hop 2001:db8::2
announce route 192.168.0.3/32 next-hop 198.51.100.1
announce route 192.168.0.2/32 next-hop 198.51.100.1
announce route fc08::d275:c141:3fd4:1801/128 next-hop 2001:db8::2
announce route 192.168.0.5/32 next-hop 198.51.100.1
announce route 192.168.0.4/32 next-hop 198.51.100.1
announce route fc08::3561:fa53:47e0:1fbc/128 next-hop 2001:db8::2
```

Initial replay of state on connect

Load balancer was deleted

Two more load balancers were provisioned

nebula

More use cases

No production experience on these but worth looking into

- Remotely Triggered Blackholing (RTBH)
  - BGP commander: https://github.com/crazed/bgpcommander
  - Integrates ExaBGP to etcd to allow you to inject arbitrary announcements via etcd
  - You could also inject Flow routes to do fine grained traffic control
- More advanced DoS detection with RTBH
  - Fastnetmon: https://github.com/pavel-odintsov/fastnetmon
  - Integrated solution that can detect attacks from sflow/pcap sources and announce blackhole routes via ExaBGP

nebula

Alternatives

- While ExaBGP is great for simple applications sometimes the interfacing via subprocesses seems somewhat weird
- One interesting recent alternative is GoBGP
  - It uses gRPC for integration
  - They have recently released a guide how to embed GoBGP into your Go application
  - RR, Route Server and Policy support
- RYU BGP Speaker library
  - Clean Python API to embed BGP into your Python application
  - Features seem to be quite limited compared to ExaBGP
- BaGPipe BGP
  - Orange's ExaBGP fork that is more focused in delivering IP-VPN/EVPN for VMs. It has dataplane integration with Open vSwitch

nebula

Thank you

ExaBGP: github.com/Exa-Networks/exabgp

My contact details:

aakso@iki.fi / anton.aksola@nebula.fi

Github.com/aakso

linkedin.com/in/aakso

nebula