GoBGP

- Open Source BGP implementation
  - https://github.com/osrg/gobgp

- Written in Go

- Main Target Applications
  1. High performance Route Server for IX
  2. Integration with data analysis systems
  3. BGPd for white box switches
Motivation

Why another BGP implementation?

SDN Era has begun

We need SDN-Native BGP implementation!
GoBGP Overview

What SDN-Native means

1. High Performance
   • Existing OSS BGPd are mainly single-threaded
   • GoBGP can exploit multicore
   • Aimed to be run on modern/commodity hardwares
GoBGP Overview

What SDN-Native means

1. High Performance
2. API-first Architecture
   • Existing BGPd are mainly CLI-first
   • Automation/Integration using “expect” is painful…
   • GoBGP uses gRPC
   • 10 languages binding
   • Integration with your software is smooth
GoBGP Overview

GRPC
A high performance, open source, general RPC framework that puts mobile and HTTP/2 first.

Source code on GitHub: grpc, grpc-java, grpc-go.

Powerful IDL
Define your service using Protocol Buffers, a powerful binary serialization toolset and language.

```java
message HelloRequest {
  string greeting = 1;
}

message HelloResponse {
  string reply = 1;
}
```

Libraries in ten languages
Automatically generate idiomatic client and server stubs for your service in a variety of languages.
gRPC has libraries in: C, C++, Java, Go, Node.js, Python, Ruby, Objective-C, PHP and C#.

HTTP/2
Building on the HTTP/2 standard brings many capabilities such as bidirectional streaming, flow control, header compression, multiplexing requests over a single TCP connection and more.

These features save battery life and data usage on mobile devices while speeding up services and web applications running in the cloud.
GoBGP Overview

What SDN-Native means

1. High Performance
2. API-first Architecture
3. Vendor-Neutral Configuration Model
   • Existing BGPd configuration varies
   • GoBGP uses OpenConfig
     • YANG model for BGP
     • draft-ietf-idr-bgp-model-00
     • Cisco’s support just announced
GoBGP Overview

What SDN-Native means

1. High Performance
   • Go

2. API-first Architecture
   • gRPC

3. Vendor-Neutral Configuration Model
   • OpenConfig
GoBGP Basics

- Comes with two binary
  - gobgpd : bgp daemon
  - gobgp : cli tool (uses gRPC underneath)
GoBGP Basics – show neighbors

```
/home/vagrant% docker exec g1 gobgp neighbor
Peer        AS  Up/Down State  #Advertised Received Accepted
172.17.0.3  65001 00:02:11 Establ 3 1 1
172.17.0.4  65002 00:01:31 Establ 3 3 3
172.17.0.5  65003 00:01:31 Establ 2 2 2
172.17.0.6  65004 00:00:05 Active 0 0 0
```

GoBGP Basics – show neighbor

```
/home/vagrant% docker exec g1 gobgp neighbor 172.17.0.4
BGP neighbor is 172.17.0.4, remote AS 65002
   BGP version 4, remote router ID 192.168.0.3
   BGP state = BGP_FSM_ESTABLISHED, up for 00:02:00
   BGP OutQ = 0, Flops = 0
Hold time is 0, keepalive interval is 30 seconds
Configured hold time is 90, keepalive interval is 30 seconds
Neighbor capabilities:
   BGP_CAP_MULTIPROTOCOL:
      RF_IPV4_UC: advertised and received
   BGP_CAP_ROUTE_REFRESH: advertised and received
   BGP_CAP_FOUR_OCTET_AS_NUMBER: advertised and received
   BGP_CAP_ROUTE_REFRESH_CISCO: received
Message statistics:
    Sent  Rcvd
   Opens: 2   2
   Notifications: 0   2
   Updates: 7   4
   Keepalives: 5   7
   Route Refresh: 0   0
```
GoBGP Basics – show global rib

```
/home/vagrant% docker exec g1 gobgp global rib
<table>
<thead>
<tr>
<th>Network</th>
<th>Next Hop</th>
<th>AS_PATH</th>
<th>Age</th>
<th>Attrs</th>
</tr>
</thead>
<tbody>
<tr>
<td>* 10.0.1.0/24</td>
<td>172.17.0.3</td>
<td>65001</td>
<td>00:03:16</td>
<td>[{Origin: i} {Med: 0}]</td>
</tr>
<tr>
<td>* 10.0.2.0/24</td>
<td>172.17.0.4</td>
<td>65002</td>
<td>00:02:36</td>
<td>[{Origin: i} {Med: 200}]</td>
</tr>
<tr>
<td>* 10.0.3.0/24</td>
<td>172.17.0.5</td>
<td>65003</td>
<td>00:02:36</td>
<td>[{Origin: i} {Med: 100}]</td>
</tr>
<tr>
<td>* 10.0.3.0/24</td>
<td>172.17.0.4</td>
<td>65002 65005 65003</td>
<td>00:02:36</td>
<td>[{Origin: i} {Med: 200}]</td>
</tr>
<tr>
<td>* 10.0.6.0/24</td>
<td>172.17.0.5</td>
<td>65003 65005</td>
<td>00:02:36</td>
<td>[{Origin: i} {Med: 100}]</td>
</tr>
<tr>
<td>* 10.0.6.0/24</td>
<td>172.17.0.4</td>
<td>65002 65005</td>
<td>00:02:36</td>
<td>[{Origin: i} {Med: 200}]</td>
</tr>
</tbody>
</table>
/home/vagrant%  
```
GoBGP Basics – monitor new best

```
/home/vagrant% docker exec g1 gobgp monitor global rib
[ROUTE] 10.10.0.0/24 via 172.17.0.3 aspath [65001] attrs [{Origin: i} {Med: 0}]
[ROUTE] 10.20.0.0/24 via 172.17.0.3 aspath [65001] attrs [{Origin: i} {Med: 0}]
[ROUTE] 10.40.0.0/24 via 172.17.0.3 aspath [65001] attrs [{Origin: i} {Med: 0}]
[ROUTE] 10.30.0.0/24 via 172.17.0.3 aspath [65001] attrs [{Origin: i} {Med: 0}]
[DELRROUTE] 10.10.0.0/24 via 172.17.0.3 aspath [65001] attrs [{Origin: i} {Med: 0}]
[DELRROUTE] 10.20.0.0/24 via 172.17.0.3 aspath [65001] attrs [{Origin: i} {Med: 0}]
```
GoBGP Basics - json option

```
/home/vagrant% docker exec g1 gobgp neighbor 172.17.0.3 -j
{"conf":{"remote_ip":"172.17.0.3","id":"192.168.0.2","remote_as":65001,"remote_cap":[{"code":1,"value":65537},{"code":128},{"code":2},{"code":65,"value":65001}],"local_cap":[{"code":2},{"code":1,"value":65537},{"code":65,"value":65000}],"info":{"messages":{"received":{"UPDATE":12,"OPEN":1,"KEEPALIVE":25,"TOTAL":38},"sent":{"UPDATE":13,"OPEN":1,"KEEPALIVE":24,"TOTAL":38}},"bgp_state":"BGP_FSM_ESTABLISHED","admin_state":"ADMIN_STATE_UP","received":3,"accepted":3,"advertized":3},"timers":{"config":{"hold_time":90,"keepalive_interval":30},"state":{"uptime":704,"downtime":704}}}
/home/vagrant%
```
GoBGP Basics – use from python

```python
import gobgp_pb2
import sys

_TIMEOUT SECONDS = 10

def run(gobgp_addr, neighbor_addr):
    with gobgp_pb2.early_adoption_create_GobgpApi_stub(gobgp_addr, 8080) as stub:
        peer = stub.GetNeighbor(gobgp_pb2.Arguments(rf=4, name=neighbor_addr), _TIMEOUT SECONDS)
        print("BGP neighbor is %s, remote AS %d" % (peer.conf.neighbor_address, peer.conf.peer_as))
        print(" BGP version 4, remote router ID %s" % (peer.conf.id))
        print(" BGP state = %s, up for %s" % (peer.info.bgp_state, peer.timers.state.uptime))
        print(" BGP OutQ = %d, Flops = %d" % (peer.info.out_q, peer.info.flops))
        print(" Hold time is %d, keepalive interval is %d seconds" % (peer.timers.state.negotiated_hold_time, peer.timers.state.keepalive_interval))
        print(" Configured hold time is %d, keepalive interval is %d seconds" % (peer.timers.config.hold_time, peer.timers.config.keepalive_interval))

if __name__ == '_main_':
    gobgp = sys.argv[1]
    neighbor = sys.argv[2]
    run(gobgp, neighbor)
```

• This snippet results in…
GoBGP Basics – use from python

• This! No more “expect”
Main Target Application

1. High performance Router Server for IX

2. Integration with data analysis systems

3. BGPd for whitebox switches
Route Server for IX

GoBGP as a route server
- supports multiple RIBs
- flexible policy enforcement points

A → Adj-RIB-In for A → Adj-RIB-Out for A → A
B → Adj-RIB-In for B → Adj-RIB-Out for B → B
C → Adj-RIB-In for C → Adj-RIB-Out for C → C

○ : In Policy  ● : Import Policy  ● : Export Policy
Supported Policy Condition/Action

- **Condition**
  - Prefix, Source neighbor
  - AS Path (contents, length)
  - Community, Extended Community
  - RPKI validation result

- **Action**
  - Permit/Deny
  - Add/Replace/Remove (extended) community, med, aspath
  - Arithmetic operation of med
Main Target Application

1. High performance Router Server for IX

2. Integration with data analysis systems

3. BGPd for whitebox switches
Integration with data analysis systems

GoBGP as a BGP sub-system for data analysis

e.g.) BGPmon (Colorado State Univ.), FastNetMon

![Diagram showing integration with data analysis systems]

- **A**
- **B**
- **C**

BGP Peering

GoBGP

gRPC

\(\lambda\)

stream processing

BigData Analysis

Copyright©2015 NTT Corp. All Rights Reserved.
Main Target Application

1. High performance Router Server for IX
2. Integration with data analysis systems
3. BGPd for whitebox switches
BGPd for whitebox switches

- BIG wave of open-networking
  - Network commoditization
  - Expansion of the use of whitebox switches

- GoBGP can be run on top of whitebox switches
  - Ported on Cumulus and Open Network Linux
BGPd for whitebox switches

- FIB modification via zapi and netlink
  - zapi: api for zebra and quagga routing daemons
  - IPv4/IPv6 unicast FIB modification is done via zapi
  - Other FIB modification is done via netlink
    - e.g.) zapi doesn’t support mac fdb modification
BGPd for whitebox switches

- Usecase: EVPN+VXLAN
  - mac address exchange occurs in BGP
- Interoperability with Cisco/Juniper! @Interop Tokyo 2015
Other features

- Full route MRT injection less than 1min
  - For testing your new gear
- Route monitoring (MRT dump)
  - BMP is also on the roadmap
- Route Reflector
  - Addpath is also on the roadmap
- RPKI validation
- Flowspec
- VPN support (L2VPN(EVPN), L3VPN, VRF, RTC)
Summary

Please try it out!

Your comment, feedback, patch and star on github is very welcome ;)