



GoBGP – BGP daemon in the Open Networking Era

2016/10/21

NTTソフトウェアイノベーションセンタ (SIC)

石田渉

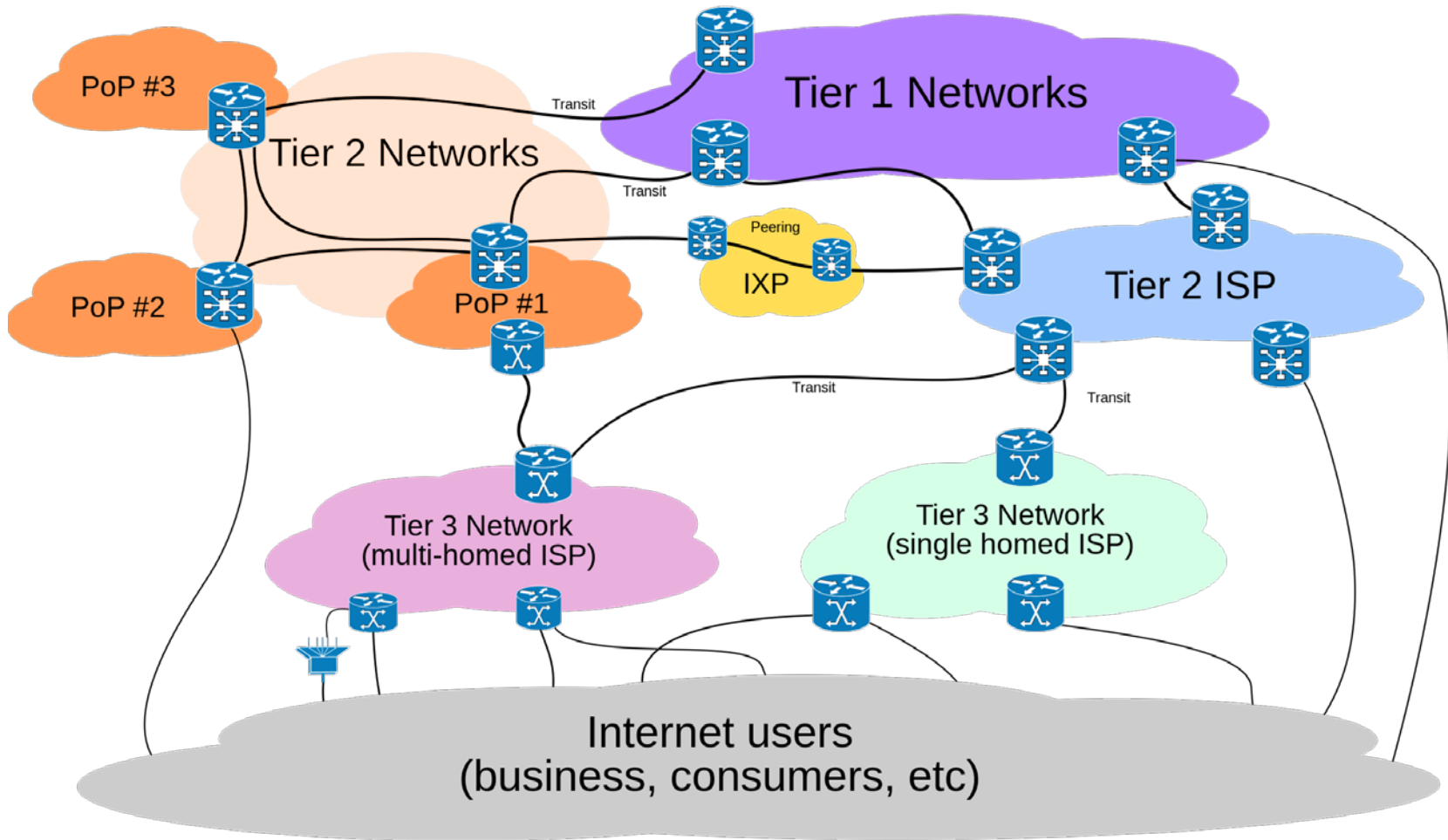
- NTT発のオープンソースBGP実装
 - <http://github.com/osrg/gobgp>



GoBGP

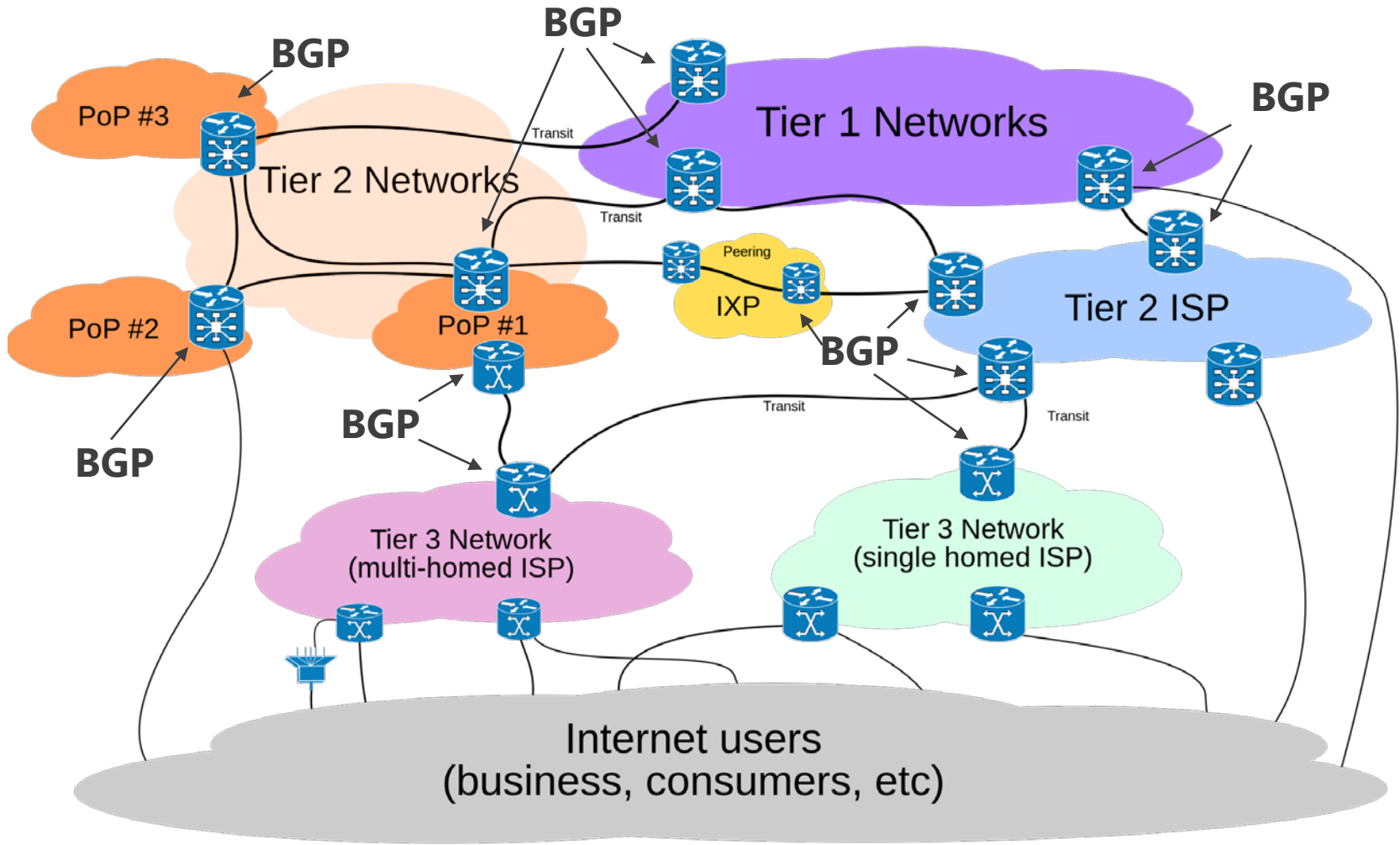
- **Border Gateway Protocol**
- **インターネットを支えるルーティングプロトコルとして世界中で利用されている**

BGP?



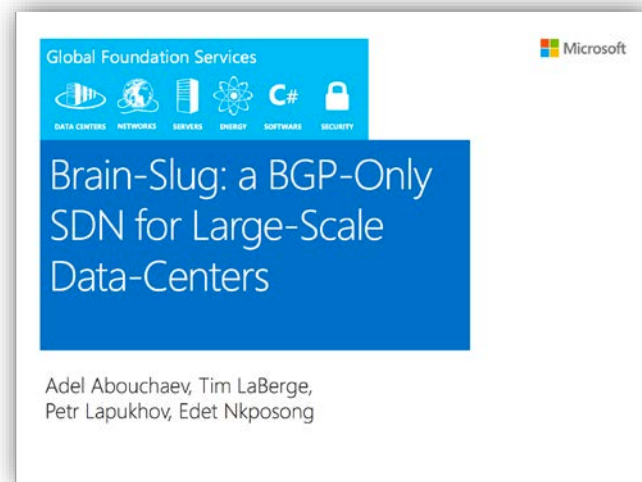
<https://en.wikipedia.org/wiki/Internet>

BGP!



<https://en.wikipedia.org/wiki/Internet>

- インターネット以外のユースケース
 - VPNキャリアバックボーン
 - データセンタネットワーク
 - DDoSプロテクション (FlowSpec)
 - KVS (draft-lapukhov-bgp-opaque-signaling)



<https://www.nanog.org/sites/default/files/wed.general.brainslug.lapukhov.20.pdf>

f

なぜ新しいBGP実装？



コンピューティング

メインフレーム



IAサーバ + Linux

ネットワーキング

垂直統合された
ブラックボックス



Open Networking

Open Networking時代に適合したBGP実装が
必要

1. モダンハードウェアへの適合

- マルチコア、潤沢なメモリリソース
- コモディティ化するネットワークハードウェア
 - ホワイトボックススイッチ

2. 他ソフトウェアとの連携の容易さ

- do one thing well

3. ベンダニュートラルな設定項目

- 他の同種ソフトウェアとの同時運用・切り替えの容易さ

1. モダンハードウェアへの適合

- マルチコアを利用するプログラムが記述しやすいGo言語で実装
- ホワイトボックススイッチ上でも動作

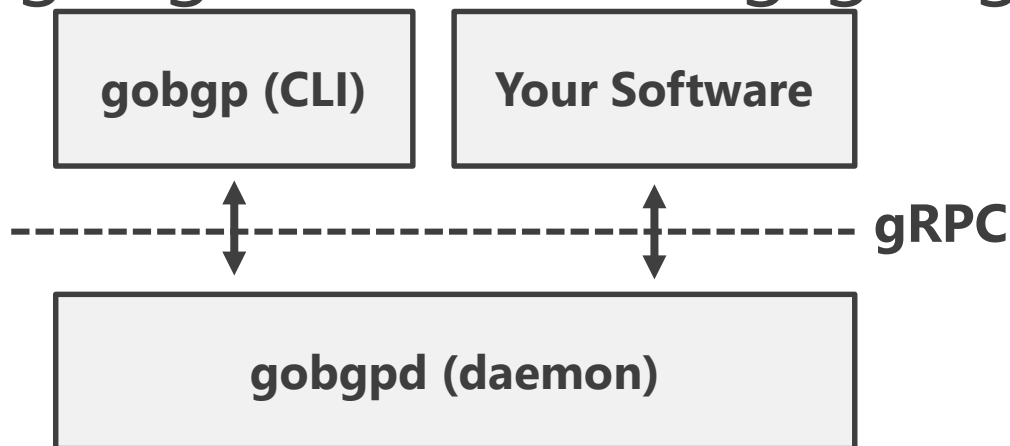
2. 他ソフトウェアとの連携の容易さ

- BGP機能に集中
 - FIB管理, MGMTシステムは外出し
- gRPC APIを通じたBGPの設定, 情報取得が可能
 - FIB管理, MGMTシステム(CLI含む)もAPIを利用して実装

3. ベンダニュートラルな設定項目

- コンフィギュレーションモデルはOpenConfig準拠
- OpenConfig : Google, Microsoftらが策定するオープンなコンフィギュレーションモデル(YANGモデル)

- **gobgpd(daemon)とgobgp(CLI)から構成**
 - **インストール方法**
 - go get
github.com/osrg/gobgp/gobgpd
 - go get github.com/osrg/gobgp/gobgp



GoBGP CLI : BGP neighborの表示



```
1. _zsh_tmux_plugin_run (vagrant)
ubuntu@ubuntu-xenial ~ $ docker exec g1 gobgp neighbor
Peer          AS   Up/Down State      |#Advertised Received Accepted
172.17.0.3    65001 00:00:56 Establ     |          2           1           1
172.17.0.4    65002 00:01:13 Establ     |          2           1           1
172.17.0.5    65003 00:01:10 Establ     |          2           1           1
ubuntu@ubuntu-xenial ~ $
```

[0] 1:~* "ubuntu-xenial" 01:19 21-Oct-16

GoBGP CLI : BGP neighborの表示



```
1. _zsh_tmux_plugin_run (vagrant)
172.17.0.5 65003 00:01:10 Establ | 2 1 1
ubuntu@ubuntu-xenial ~ $ docker exec g1 gobgp neighbor 172.17.0.3
BGP neighbor is 172.17.0.3, remote AS 65001
  BGP version 4, remote router ID 192.168.0.2
  BGP state = BGP_FSM_ESTABLISHED, up for 00:01:10
  BGP OutQ = 0, Flops = 0
  Hold time is 90, keepalive interval is 30 seconds
  Configured hold time is 90, keepalive interval is 30 seconds
  Neighbor capabilities:
    multiprotocol:
      ipv4-unicast:  advertised and received
      route-refresh:  advertised and received
      4-octet-as:  advertised and received
      cisco-route-refresh:  received
  Message statistics:
      Sent      Rcvd
  Opens:           2          1
  Notifications:  0          0
  Updates:         2          1
  Keepalives:      3          4
  Route Refresh:   0          0
  Discarded:       0          0
  Total:           7          6
  Route statistics:
    Advertised:    2
    Received:      1
    Accepted:      1
ubuntu@ubuntu-xenial ~ $
```

GoBGP CLI : 経路表示



```
1. _zsh_tmux_plugin_run (vagrant)
ubuntu@ubuntu-xenial ~ $ docker exec g1 gobgp global rib
  Network      Next Hop      AS_PATH      Age      Attrs
*> 10.0.1.0/24  172.17.0.3    65001        00:01:38  [{Origin: i} {Med: 0}]
*> 10.0.2.0/24  172.17.0.4    65002        00:01:55  [{Origin: i} {Med: 0}]
*> 10.0.3.0/24  172.17.0.5    65003        00:01:52  [{Origin: i} {Med: 0}]
ubuntu@ubuntu-xenial ~ $
```

[0] 1:~* "ubuntu-xenial" 01:19 21-Oct-16

GoBGP CLI : 経路注入



```
1. _zsh_tmux_plugin_run (vagrant)
ubuntu@ubuntu-xenial ~ $ docker exec g1 gobgp global rib add 10.10.10.0/24
ubuntu@ubuntu-xenial ~ $ docker exec g1 gobgp global rib add key hello value world -a opaque
ubuntu@ubuntu-xenial ~ $ █

[0] 1:~* "ubuntu-xenial" 01:21 21-Oct-16
```

GoBGP CLI : 経路のリアルタイムモニタリング



```
1. vagrant ssh (vagrant)
/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 CLI : -j オプション



```
1. vagrant ssh (vagrant)
/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 : gRPC API



```
1. vagrant ssh (vagrant)
1 import gobgp_pb2
2 import sys
3
4 _TIMEOUT_SECONDS = 10
5
6
7 def run(gobgpd_addr, neighbor_addr):
8     with gobgp_pb2.early_adopter_create_GobgpApi_stub(gobgpd_addr, 8080) as stub:
9         peer = stub.GetNeighbor(gobgp_pb2.Arguments(rf=4, name=neighbor_addr), _TIMEOUT_SECONDS)
10        print("BGP neighbor is %s, remote AS %d" % (peer.conf.neighbor_address, peer.conf.peer_as))
11        print(" BGP version 4, remote router ID %s" % (peer.conf.id))
12        print(" BGP state = %s, up for %s" % (peer.info.bgp_state, peer.timers.state.uptime))
13        print(" BGP OutQ = %d, Flops = %d" % (peer.info.out_q, peer.info.flops))
14        print(" Hold time is %d, keepalive interval is %d seconds" % (peer.timers.state.negotiated_hold_time,
15                                                                    peer.timers.state.keepalive_interval))
16        print(" Configured hold time is %d, keepalive interval is %d seconds" % (peer.timers.config.hold_time,
17                                                                    peer.timers.config.keepalive_interval))
18
19 if __name__ == '__main__':
20     gobgp = sys.argv[1]
21     neighbor = sys.argv[2]
22     run(gobgp, neighbor)
~
~
NORMAL get_neighbor.py unix | utf-8 | python 4% 1:1
[0] 1:zsh* "trusty" 01:05 19-Nov-15
```

- BGP neighborを取得するPythonスニペット

GoBGP : gRPC API

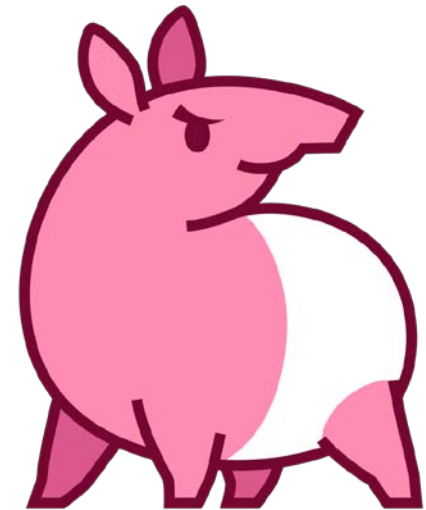


```
1. vagrant ssh (vagrant)
/home/vagrant/.go/src/github.com/osrg/gobgp/tools/grpc/python% python get_neighbor.py 172.17.0.2 172.17.0.6
BGP neighbor is 172.17.0.6, remote AS 65004
  BGP version 4, remote router ID 192.168.0.5
  BGP state = BGP_FSM_IDLE, up for 3865
  BGP OutQ = 0, Flops = 0
  Hold time is 0, keepalive interval is 0 seconds
  Configured hold time is 90, keepalive interval is 30 seconds
/home/vagrant/.go/src/github.com/osrg/gobgp/tools/grpc/python% [master]
```

[0] 1:zsh* "trusty" 01:56 19-Nov-15

- expect不要

- JPNAP IXルートサーバ
- Calico (コンテナネットワークング)
- ホワイトボックススイッチ/Kubernetes
- BGPMon, FlowSpec経路インジェクタ...





インターネットマルチフィード株式会社



Press Release

2016年 9月 30日

日本電信電話株式会社

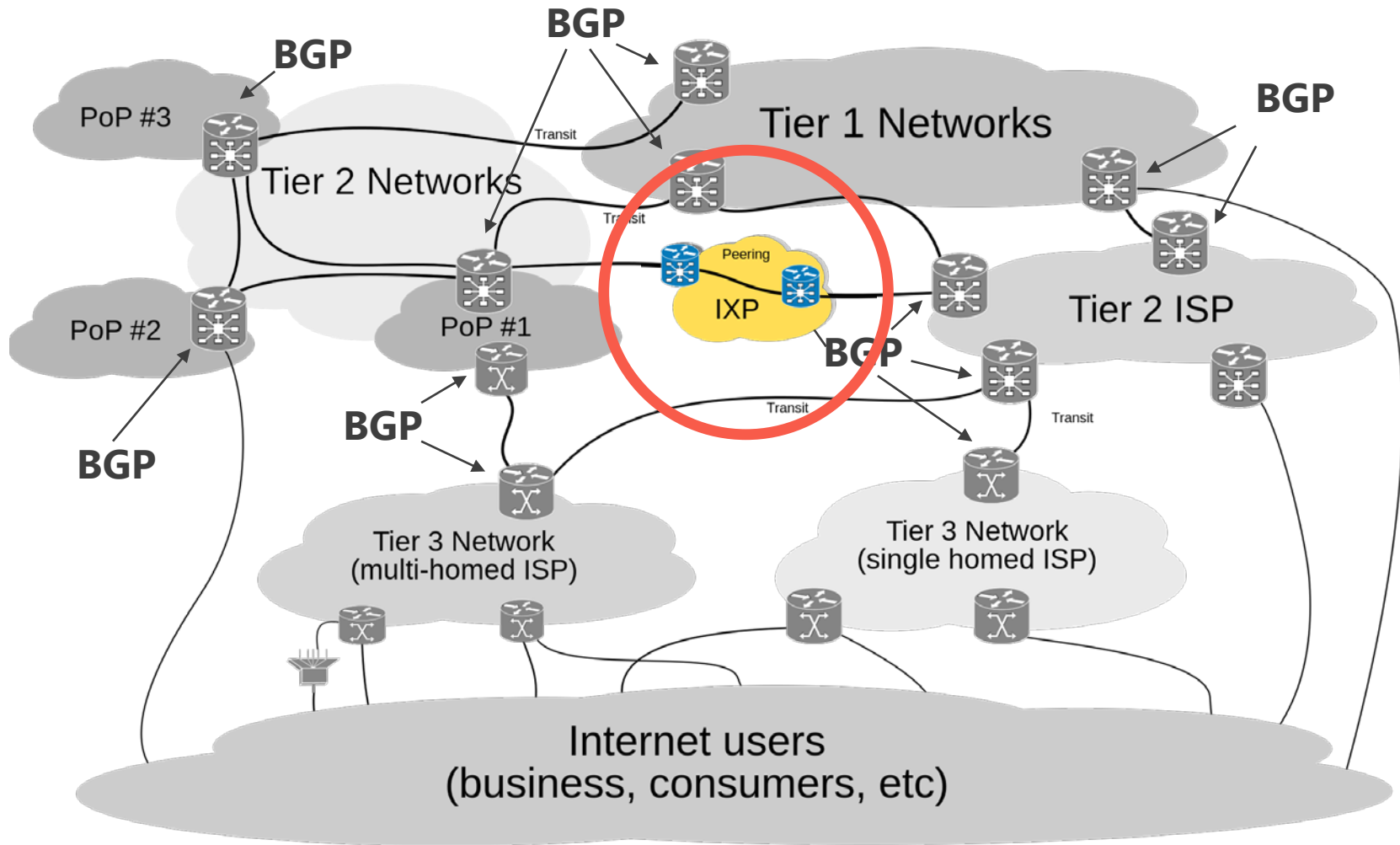
インターネットマルチフィード株式会社

NTT発のオープンソースソフトウェアGoBGPをインターネットマルチフィード社の JPNAPサービスに導入、運用の自動化を促進し、大幅な効率化を実現 ～リードタイムが、新規契約で10分の1、設定変更が30分の1に～

日本電信電話株式会社（本社：東京都千代田区、代表取締役社長：鷓浦 博夫、以下：NTT）とインターネットマルチフィード株式会社（本社：東京都千代田区、代表取締役社長：鈴木 幸一、以下：MF社）は、NTTがOSS(オープンソースソフトウェア)として開発するインターネットの経路制御機能を提供する「GoBGP」をMF社が提供するインターネット・エクスチェンジ(IX)^{(*)1}サービスであるJPNAPへの適用に向けて連携を行い、商用導入を実現いたしました。GoBGPの自動化機能を活用し、JPNAPのRouteFEEDサービス^{(*)2}の運用の自動化を実現することで、RouteFEEDサービスの新規契約におけるリードタイムを10分の1、既存のお客さまからの設定変更オーダーのリードタイムを30分の1に短縮しました。運用自動化により、従来の手動の設定変更でのヒューマンエラーによるトラブルを防ぐことができ、当該運用稼働も10分の1程度に削減することが出来ます。GoBGPのIX事業者向けの商用導入はJPNAPが世界初となります。なお、この成果は、2016年10月にスリランカで開催されるアジアパシフィック地域のIX事業者の会議(APIX^{(*)3})で報告いたします。

<http://www.mfeed.co.jp/press/2016/20160930.html>

JPNAP IXルートサーバ

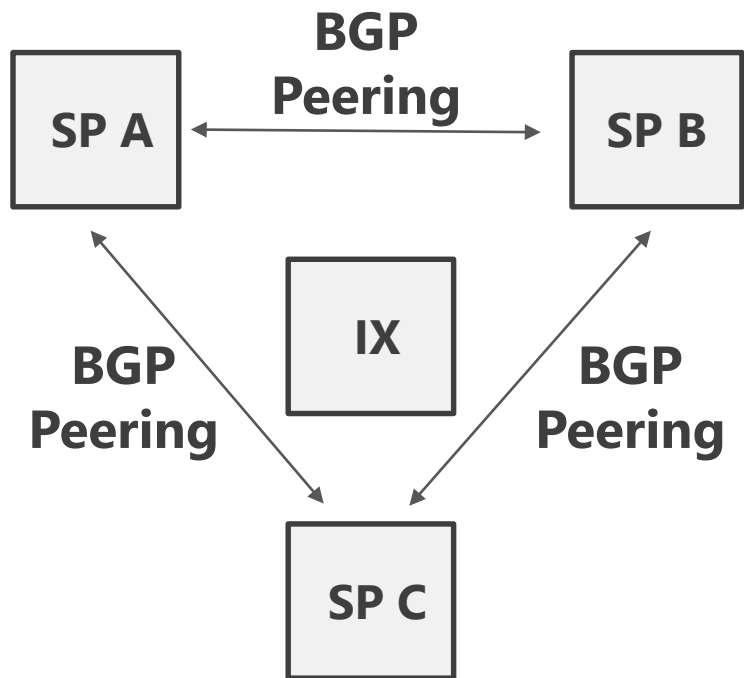


<https://en.wikipedia.org/wiki/Internet>

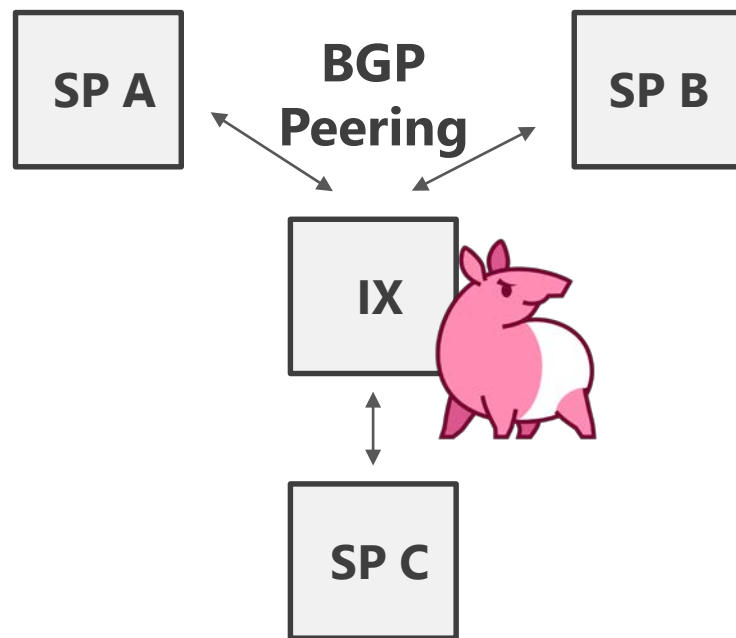
ルートサーバ



ルートサーバ無



ルートサーバ有



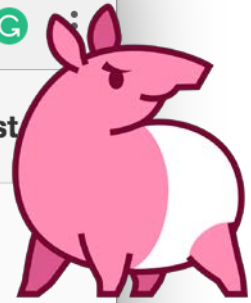
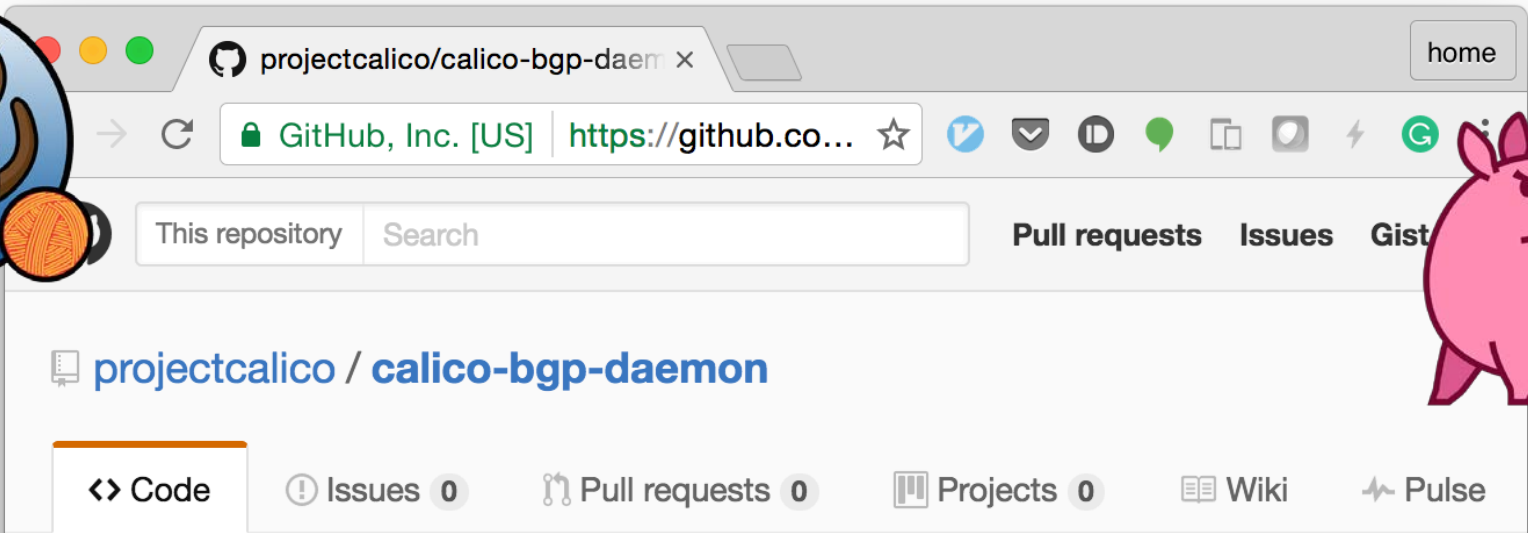
- IXの利用者のBGP運用負荷を低減
 - IXルートサーバのみ接続すればよい
- ルートサーバにはピア数に対するスケーラビリティが必要
 - 一方データプレーンはいらない

Calico (コンテナネットワーキング)

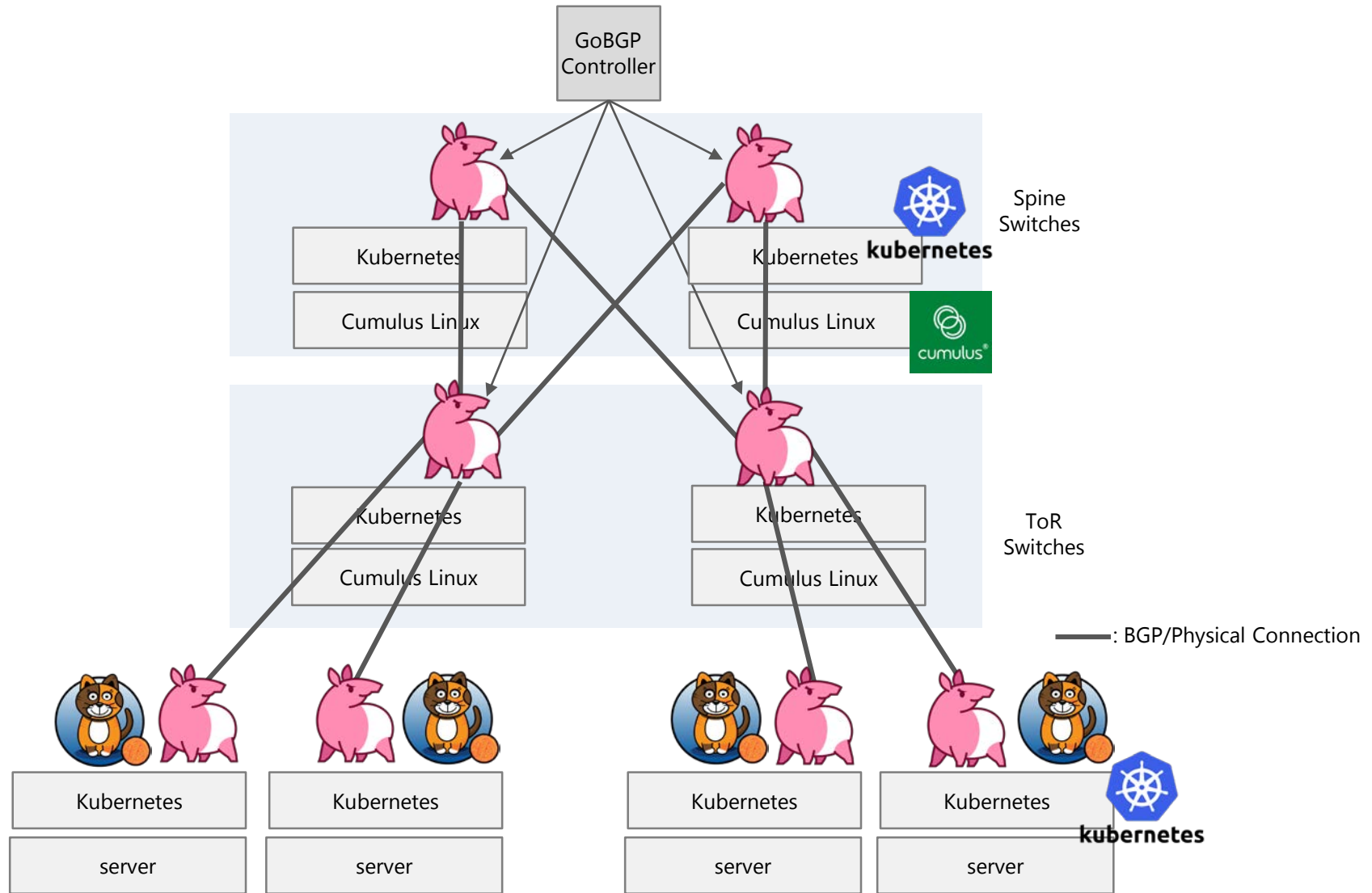


- **Project Calico**

- 様々なコンテナオーケストレータ(Kubernetes, Docker, Mesos)に対応したネットワーキングフレームワーク
 - マルチホスト, マルチテナント環境でのファイアウォール, isolationの提供
 - スケーラビリティのためBGPを利用
 - BGPデーモンとしてGoBGPが組み込まれている
 - <https://github.com/projectcalico/calico-bgp-daemon>



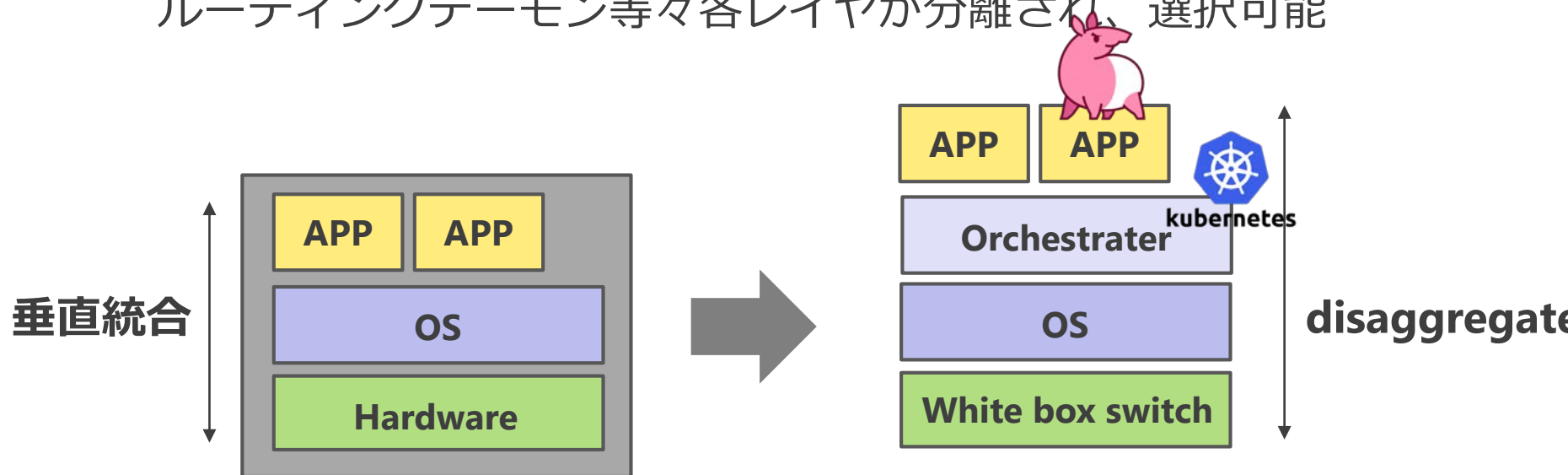
ホワイトボックススイッチ/Kubernetes



ホワイトボックススイッチ/Kubernetes



- NTT SICのラボ・生活ネットワークに導入
 - ホワイトボックススイッチ + Cumulus Linux + Kubernetes + GoBGP
 - BGPルータを含む様々なサービスをコンテナとしてデプロイ
 - デプロイ・アップグレード・死活監視はKubernetesの仕組みを利用
 - C-D分離にとどまらず、ハードウェア, OS, オーケストレータ, ルーティングデーモン等々各レイヤが分離され、選択可能



ホワイトボックススイッチ/Kubernetes



世の中を変えるオープンテクノロジーとアイデアの集結

Okinawa Open Days 2016

@Okinawa-Jichikaikan
DEC. 5-8, 2016

<https://www.facebook.com/okinawaopenlabs>

The poster features a blue and white pixelated background with a map of Japan. It includes the event title "Okinawa Open Days 2016" in large orange letters, the location "@Okinawa-Jichikaikan" and dates "DEC. 5-8, 2016" in pink, and a Facebook link at the bottom. There are also several colorful flower icons scattered across the design.

詳細はOkinawaOpenDays2016
で！

- **GoBGP : Open Networking時代に適合したBGP実装として鋭意開発中**
 - **ぜひ試してみてください**
 - **コメント、PR歓迎です**
 - <https://github.com/osrg/gobgp>
 - <https://gobgp.slack.com>

