OpenBGPD Illustrated

Masakazu Asama @ EBUG

OpenBGPD とは？

- OpenBGPD は Open Source(BSD License) な Border Gateway Protocol Version 4(BGP-4) の実装のひとつ
- OpenBSD Project の一部として Henning Brauer と Claudio Jeker を中心に開発されている
- FreeBSD では ports/net/openbgpd で利用可能
- 他の BSD にも簡単に移植可能と思われるがそれ以外の OS に移植する為には Routing Socket の操作に関する部分を書き直す必要があるため今のところは利用できない(と思われる)

BGP-4 とは？

- BGP-4 はインターネットの中核をなすルーティングプロトコル
- インターネットを自立システム(AS: おおよそ ISP のことを指していると考えて良い)の集合と考え AS 間におけるルーティング情報の交換をするためのプロトコル
- もともとは IPv4 ネットワークのことのみを考えたプロトコルだったが Multiprotocol Extensions for BGP-4(RFC4760) により他のプロトコルもサポート可能となり IPv6 も Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing(RFC2545) によりサポートされている

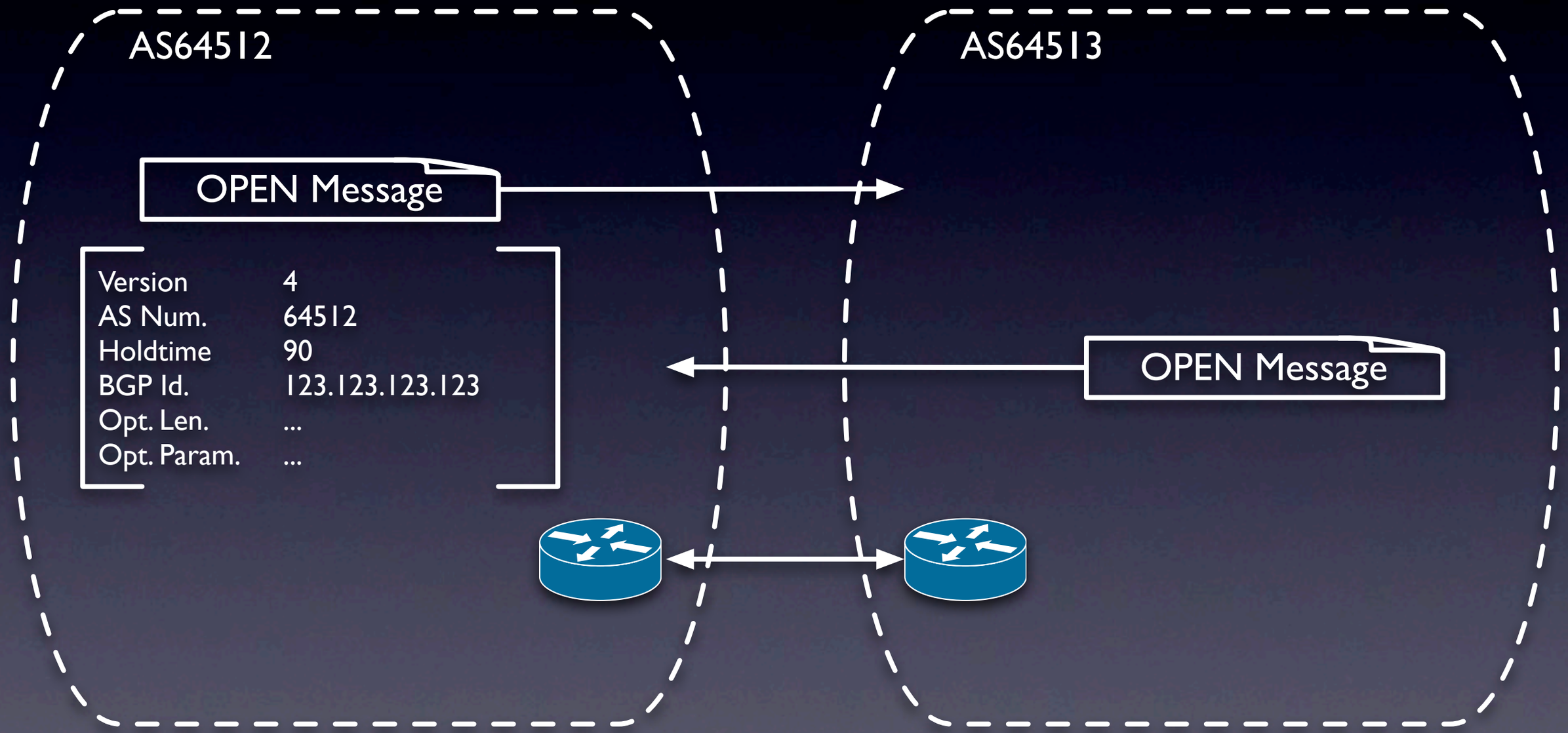
BGP-4 の基本動作(0)

AS64512

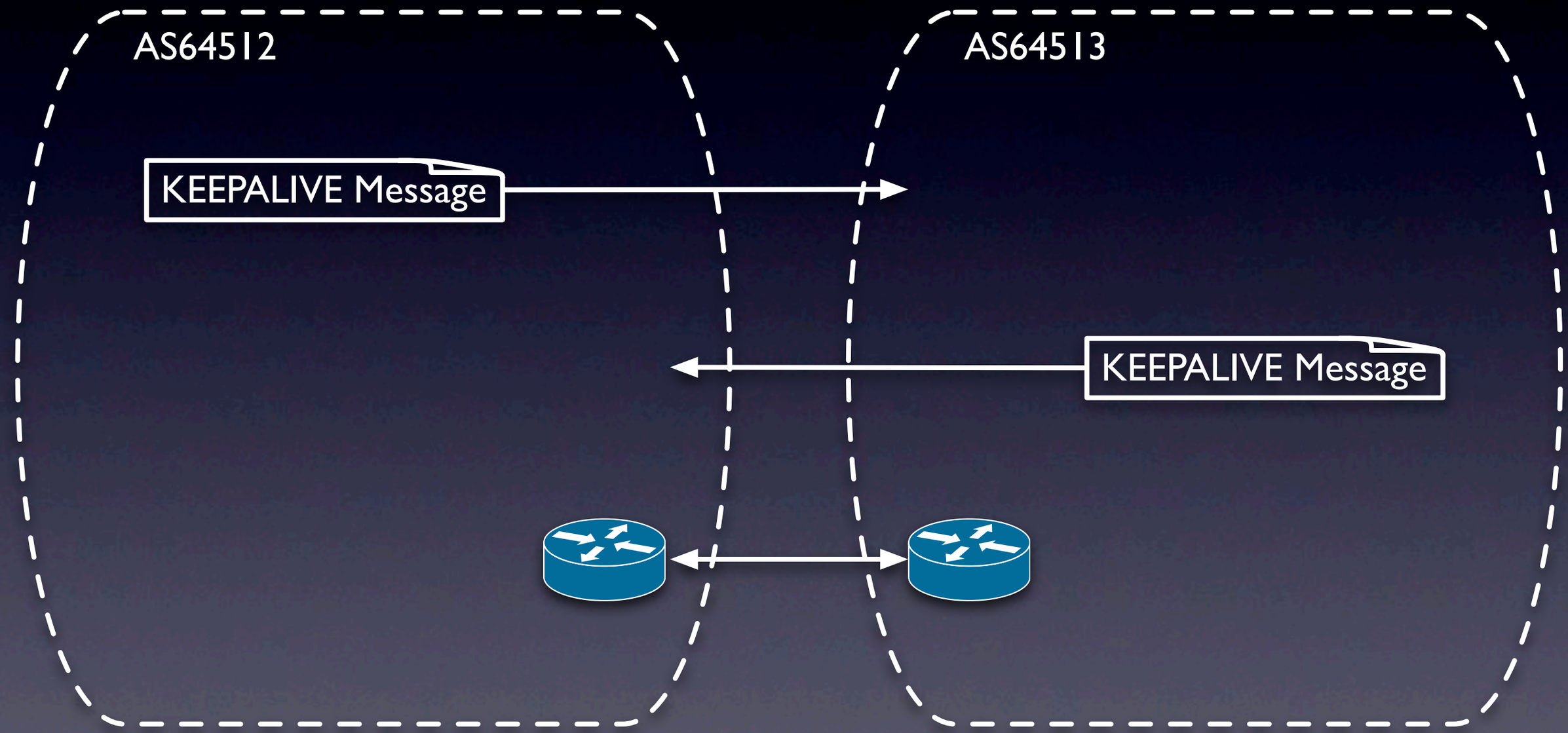
AS64513

片方の BGP-4 ルーターから
もう片方の BGP-4 ルーターに対して
TCP/179 コネクションを確立

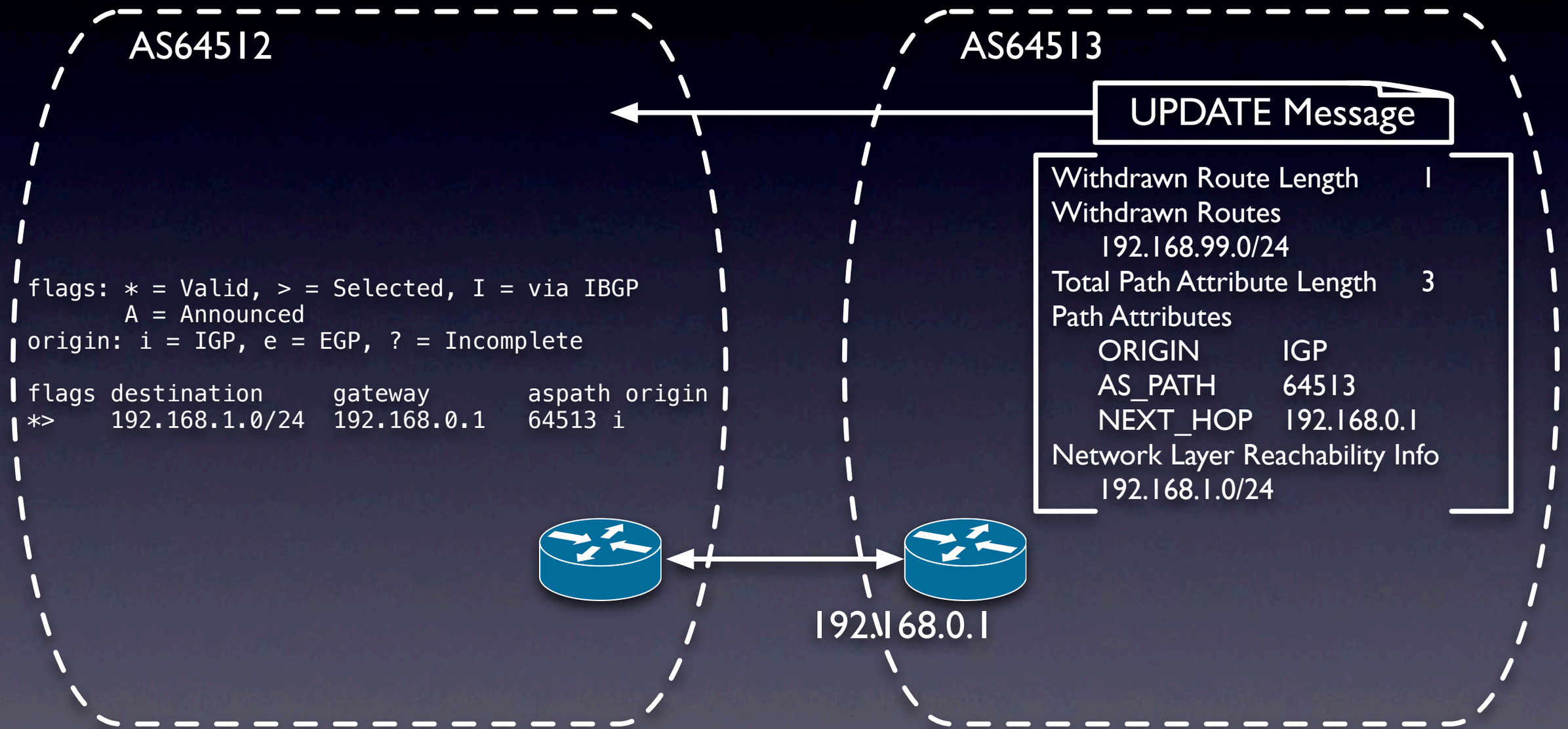
BGP-4 の基本動作(1)


AS64512
AS64513
OPEN Message
Version 4
AS Num. 64512
Holdtime 90
BGP Id. 123.123.123.123
Opt. Len. ...
Opt. Param. ...
OPEN Message

BGP-4 の基本動作(2)



BGP-4 の基本動作(3)


AS64512
AS64513
UPDATE Message
Withdrawn Route Length 1
Withdrawn Routes
192.168.99.0/24
Total Path Attribute Length 3
Path Attributes
ORIGIN IGP
AS_PATH 64513
NEXT_HOP 192.168.0.1
Network Layer Reachability Info
192.168.1.0/24
flags: * = Valid, > = Selected, I = via IBGP
A = Announced
origin: i = IGP, e = EGP, ? = Incomplete
flags destination gateway aspath origin
*> 192.168.1.0/24 192.168.0.1 64513 i
192.168.0.1

BGP-4 の基本動作(4)

AS64512

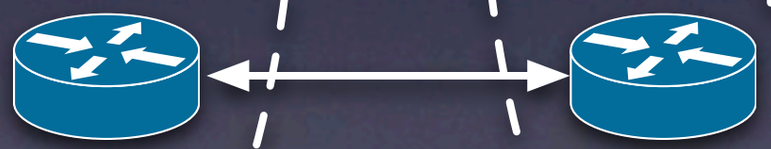
AS64513

UPDATE Message

Withdrawn Route Length	0
Withdrawn Routes	
Total Path Attribute Length	3
Path Attributes	
ORIGIN	IGP
AS_PATH	64513 64514
NEXT_HOP	192.168.0.1
Network Layer Reachability Info	
192.168.2.0/24	
192.168.3.0/24	

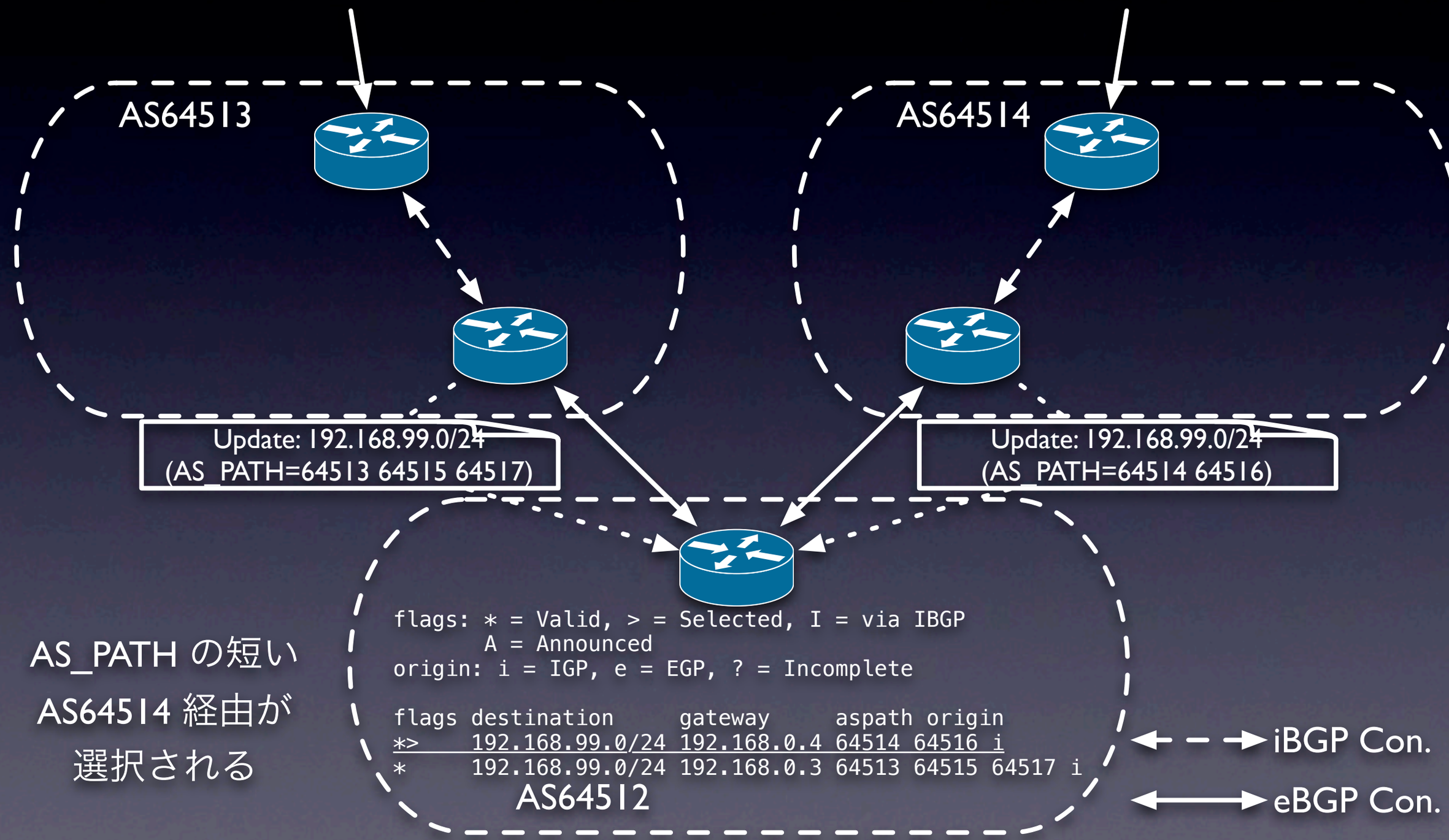
```
flags: * = Valid, > = Selected, I = via IBGP
       A = Announced
origin: i = IGP, e = EGP, ? = Incomplete

flags destination     gateway       aspath origin
*>    192.168.1.0/24  192.168.0.1   64513 i
*>    192.168.2.0/24  192.168.0.1   64513 64514 i
*>    192.168.3.0/24  192.168.0.1   64513 64514 i
```



192.168.0.1

BGP-4 の基本動作(5)


AS64513
AS64514
Update: 192.168.99.0/24
(AS_PATH=64513 64515 64517)
Update: 192.168.99.0/24
(AS_PATH=64514 64516)
flags: * = Valid, > = Selected, I = via IBGP
A = Announced
origin: i = IGP, e = EGP, ? = Incomplete
flags destination gateway aspath origin
*> 192.168.99.0/24 192.168.0.4 64514 64516 i
* 192.168.99.0/24 192.168.0.3 64513 64515 64517 i
AS64512
AS_PATH の短い
AS64514 経由が
選択される
iBGP Con.
eBGP Con.

UPDATE for IPv4

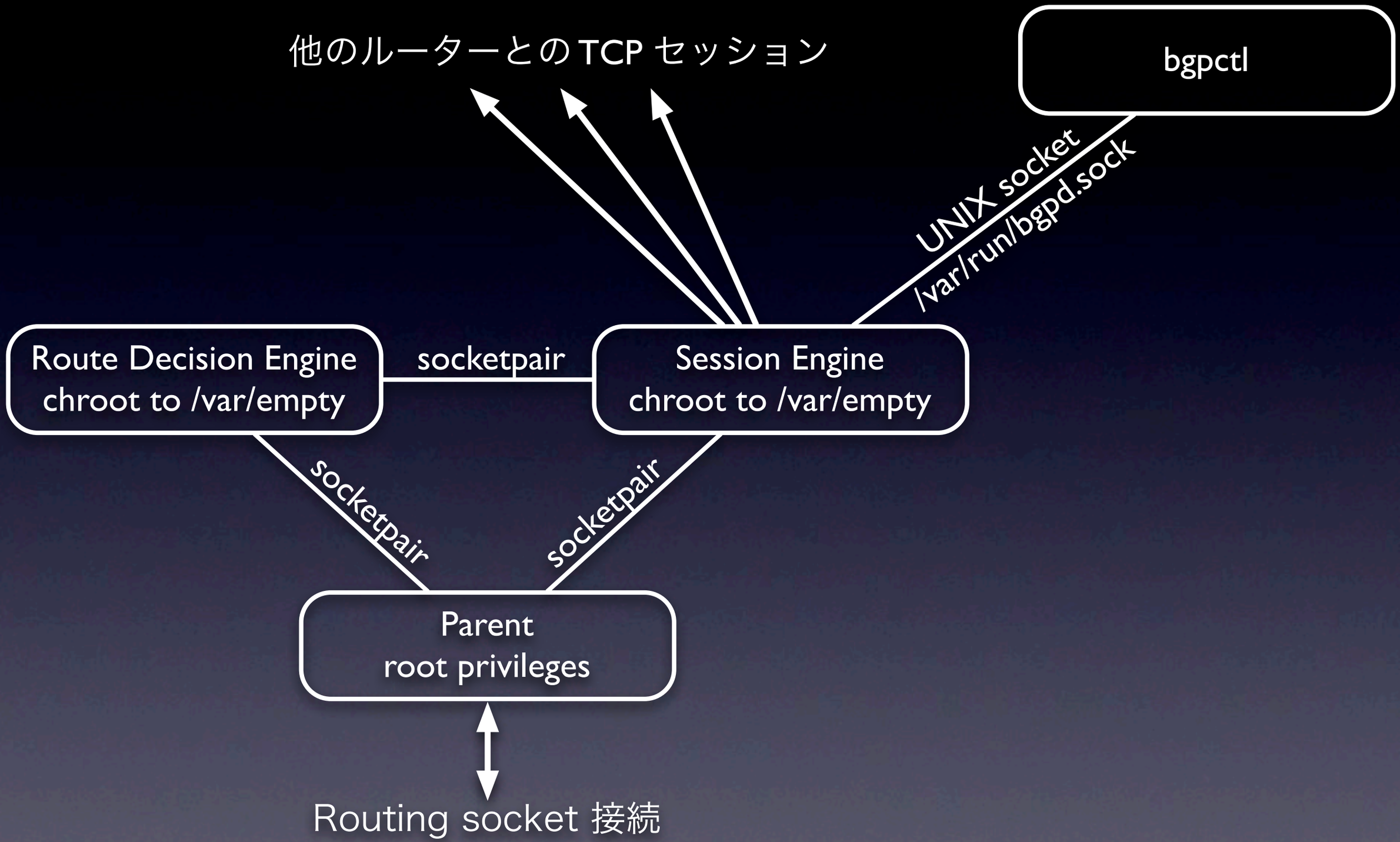
- Withdrawn Route Length 2
- Withdrawn Routes
  - 192.168.98.0/24
  - 192.168.99.0/24
- Total Path Attribute Length 3
- Path Attributes
  - ORIGIN IGP
  - AS_PATH 64513
  - NEXT_HOP 192.168.0.1
- Network Layer Reachability Info
  - 192.168.1.0/24
  - 192.168.2.0/24

UPDATE for IPv6

- Withdrawn Route Length 0
- Withdrawn Routes
- Total Path Attribute Length 4
- Path Attributes
  - ORIGIN IGP
  - AS_PATH 64513
  - MP_REACH_NLRI
    - Address family IPv6
    - Subsequent a.f. Unicast
    - Next hop 2001:db8::1
    - NLRI
      - 2001:db8:0:1::/64
      - 2001:db8:0:2::/64
  - MP_UNREACH_NLRI
    - Address family IPv6
    - Subsequent a.f. Unicast
    - Withdrawn routes
      - 2001:db8:0:98::/64
      - 2001:db8:0:99::/64
- Network Layer Reachability Info

OpenBGPD の特徴(1)

- 機能毎に 3 つのプロセスを稼働させる
  - Parent: 以下の 2 つのプロセスの親プロセス. 唯一 root 権限で chroot もされずに稼働するプロセス. 設定ファイルの再読み込みや Routing socket との通信等を行う.
  - Session Engine: 他の BGP ルーターとのセッションを管理するプロセス. bgpctl コマンドからの Unix socket 接続も受け付ける.
  - Route Decision Engine: 他のルーターから受け取った情報と自分自身の情報をもとにルーティングテーブルを計算する.
- Route Decision と Session を分離することでルーティング計算の最中であっても確実にセッションを保持することが可能となる


他のルーターとのTCP セッション
bgpctl
UNIX socket
/var/run/bgpd.sock
Route Decision Engine
chroot to /var/empty
socketpair
Session Engine
chroot to /var/empty
socketpair
socketpair
Parent
root privileges
Routing socket 接続

OpenBGPD の特徴(2)

- いわゆる CLI は用意されていない
- かわりに設定ファイル /etc/bgpd.conf で設定を行う
- 設定を変更したいときは /etc/bgpd.conf を書き換え後コマンドラインから bgpctl reload を実行する
- 設定の読み込みと反映は atomically に行われる
- 通常のコマンド bgpctl を用いて
  - 他のルーターとの接続状態を確認
  - 他のルーターとの接続を切断
  - 他のルーターから収集した情報を表示

  といった処理を行えるためシェルスクリプトによる自動化等を容易に実現することが可能となる

Example of bgpd.conf

```
AS 64512
router-id 192.168.0.1
network 192.168.0.1/24
network 2001:db8::/64
group "peering AS64513" {
    remote-as 64513
    neighbor 192.168.0.2 {
        descr   "AS64513 IPv4"
        local-address 192.168.0.1
        announce IPv4 unicast
        announce IPv6 none
    }
    neighbor 2001:db8::2 {
        descr   "AS64513 IPv6"
        local-address 2001:db8::1
        announce IPv4 none
        announce IPv6 unicast
    }
}
group "peering AS64514" {
    remote-as 64514
    neighbor 192.168.0.3 {
        descr   "AS64514 IPv4"
        local-address 192.168.0.1
        announce IPv4 unicast
        announce IPv6 none
    }
}
deny from any inet
allow from any inet prefixlen 8 - 24
deny from any prefix 0::/0 prefixlen <= 128
allow from any prefix 0::/0 prefixlen <= 48
```

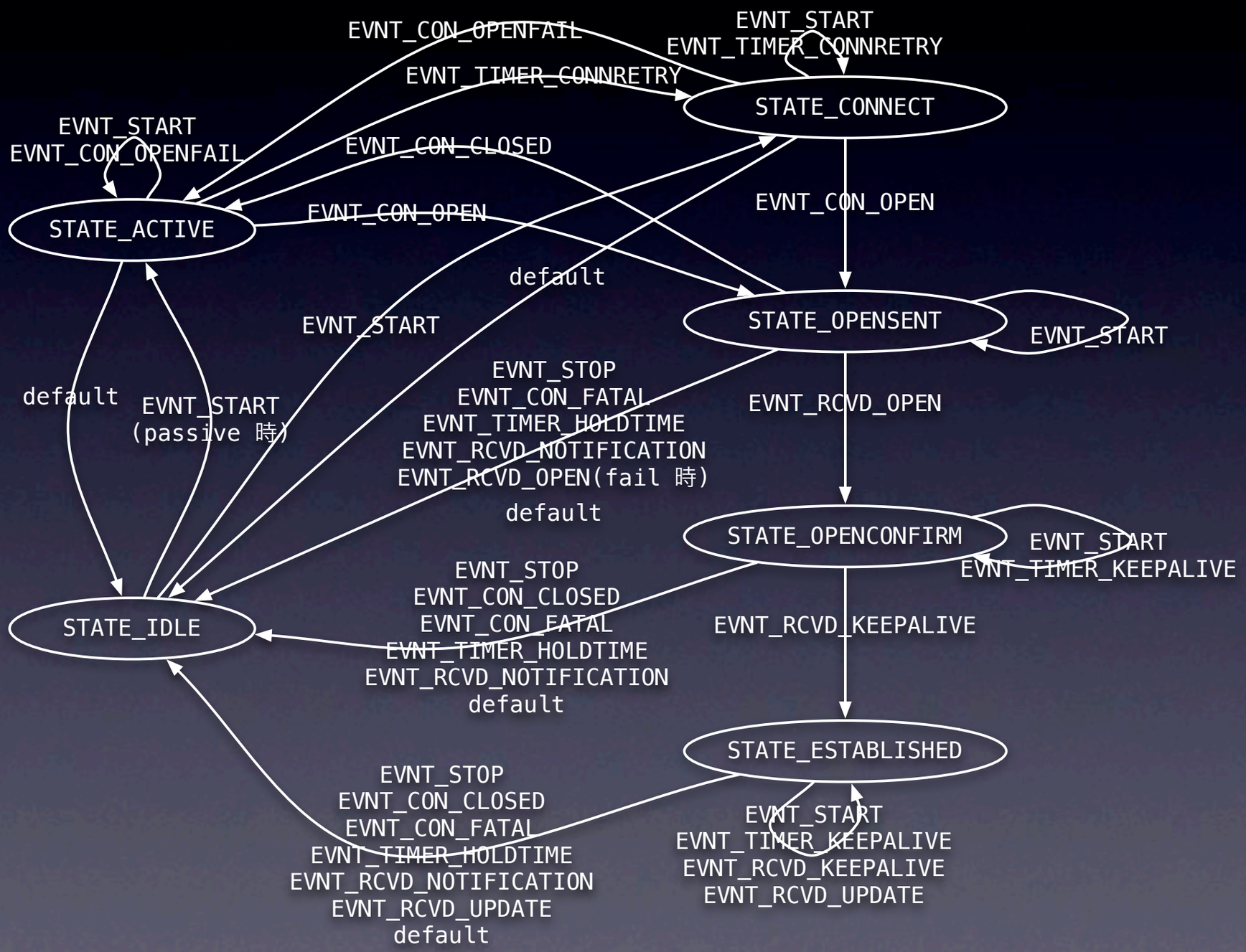
Parent

- Configuration ファイルの parse 処理
  - pfctl と同じく yacc で構文解析を行う
  - 解析内容に基づき bgpd_config や peer, network, filter_rule といった構造体に内容を格納する
- Route Decision Engine からの要求による処理
  - ルーティング情報の変更/削除の要求があったとき
  - Next Hop/pftable の追加/削除依頼があったとき
- Session Engine からの要求による処理
  - Configuration の再読み込み要求があったとき
  - show nexthop/show interface 等の show 系コマンドがあったとき
  - 等々...

Session Engine

- bgpctl からの接続を待つ UNIX domain socket の準備
- bgpctl から接続があったときには Parent や Route Decision Engine から適宜情報を収集し結果を表示したり, Parent や Route Decision Engine に対して変更要求をだしたりする
- 各 peer(neighbor) 毎に FSM(Finite State Machine: 後述) を初期化
- 各 peer(neighbor) を Established 状態まで持っていく
- peer(neighbor) から受け取った UPDATE message を Route Decision Engine に渡したり, 逆に経路が増えたり減ったりした際には UPDATE message を peer(neighbor) に渡したりする

Finite State Machine



```
void
bgp_fsm(struct peer *peer, enum session_events event)
{
    switch (peer->state) {
    case STATE_NONE:
        /* nothing */
        break;
    case STATE_IDLE:
        switch (event) {
        case EVNT_START:
            ...
            break;
        default:
            /* ignore */
            break;
        }
        break;
    case STATE_CONNECT:
        switch (event) {
        case EVNT_START:
            /* ignore */
            break;
        case EVNT_CON_OPEN:
            session_tcp_established(peer);
            session_open(peer);
            ...
            change_state(peer, STATE_OPENSENT, event);
            break;
        ...
        }
        break;
    ...
    }
}
```

```
void
change_state(struct peer *peer, enum session_state state, enum session_events event)
{
    ...
    switch (state) {
    case STATE_IDLE:
        ...
        timer_stop(peer, Timer_IdleHoldReset);
        session_close_connection(peer);
        ...
        if (peer->state == STATE_ESTABLISHED)
            session_down(peer);
        ...
        break;
    case STATE_CONNECT:
        break;
    case STATE_ACTIVE:
        break;
    case STATE_OPENSENT:
        break;
    case STATE_OPENCONFIRM:
        break;
    case STATE_ESTABLISHED:
        timer_set(peer, Timer_IdleHoldReset, peer->IdleHoldTime);
        ...
        session_up(peer);
        break;
    default:      /* something seriously fucked */
        break;
    }
    ...
    peer->prev_state = peer->state;
    peer->state = state;
}
```

Route Decision Engine

- Session Engine から受け取った UPDATE message を蓄積し RIB(Routing Information Base) を構築する
- 構築した RIB からある prefix への最適な path を計算し Parent に対してルーティングテーブルを更新するよう依頼する

II

Route Decision Process

- Session Engine を通じて他の peer(neighbor) に UPDATE message を送る

Routing Information Base

- RFC4271 には以下の 3 つの RIB について記載されている
  - Adj-RIBs-In: peer(neighbor) から受信した経路がそのまま格納される.
  - Loc-RIB: Adj-RIBs-In に対してローカルポリシーを適用した結果が格納される.
  - Adj-RIBs-Out: peer(neighbor) へ送信する経路が格納される.

  ただし上記の RIB を必ず実装しなければならないわけではない
- OpenBGPD では Adj-RIBs-In と Loc-RIB の 2 つと Configuration に応じた RIB が用意される

UPDATE 受信処理(1)

- `rde_update_dispatch(...)`
  - ❖ 受信した UPDATE message につき 1 回実行される.
    `rde_attr_parse(...)` により attributes を parse し
    `rde_aspath` 構造体を構築する.Withdraw なら
    `rde_update_withdraw(...)` を, NLRI なら
    `rde_update_update(...)` をそれぞれ実行する.
- `rde_update_update(...)`
  - ❖ UPDATE message 中に含まれる prefix 1 つにつき 1 回実行される.
    RIB の数だけ `path_update(...)` を呼び出す.

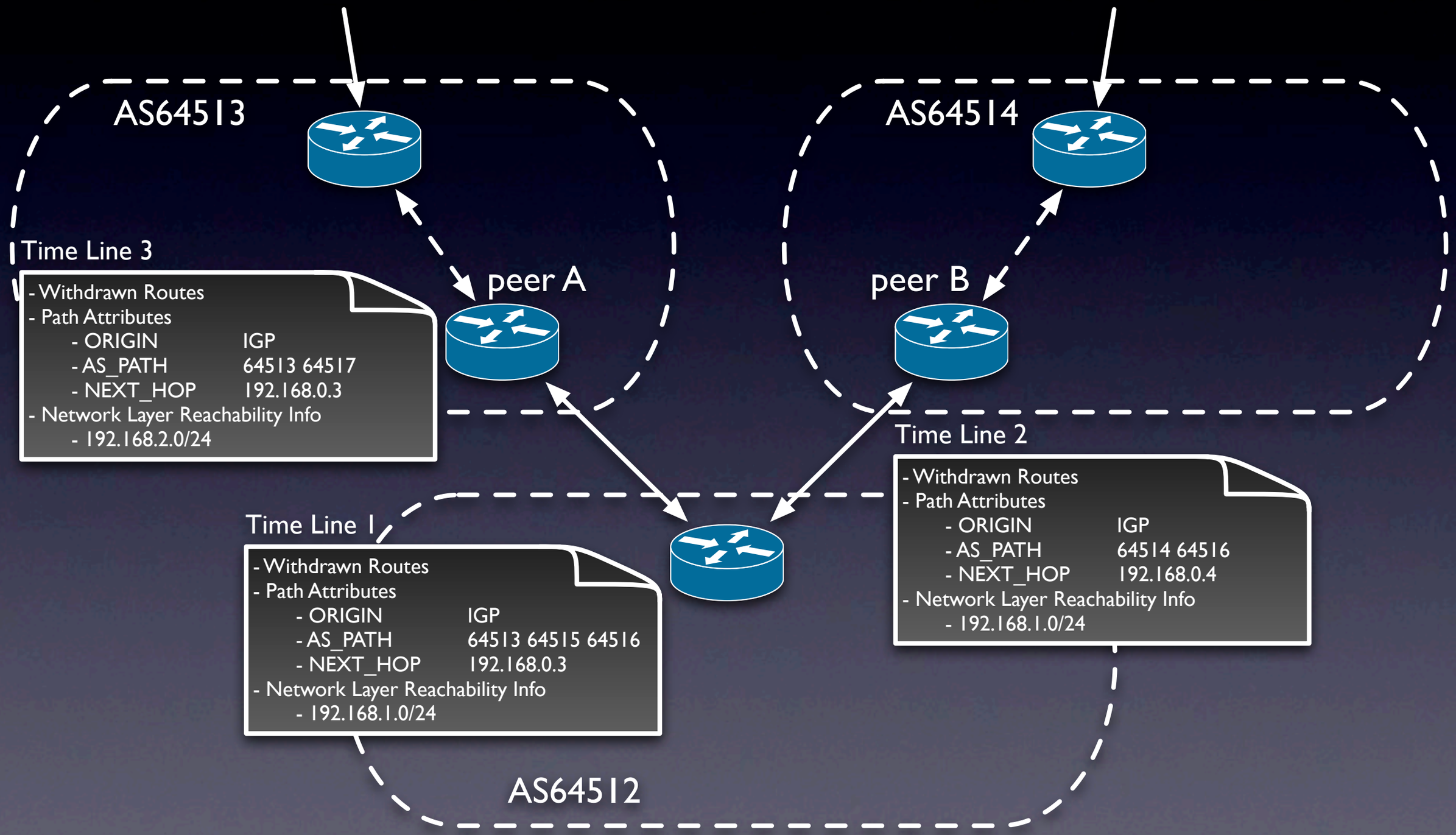
UPDATE 受信処理(2)

- path_update(...)
  - ❖ すでに該当する prefix 構造体が存在し更新の必要がある場合は prefix_move(...) を実行する.prefix 構造体が存在しない場合は prefix_add(...) を実行する.prefix 構造体が存在し更新の必要がない場合は lastchange 変数を更新するだけ.
- prefix_add(...)
  - ❖ 該当する prefix 構造体が存在しない場合新しい prefix 構造体を用意し prefix_link(...) を実行する.存在し更新の必要がある場合は prefix_move(...) を実行する.ない場合は lastchange 変数を更新するだけ.

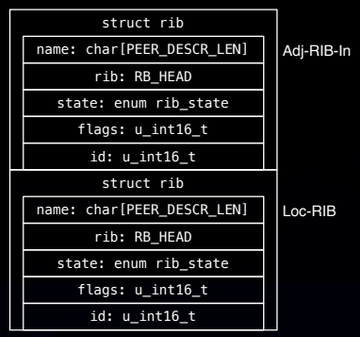
UPDATE 受信処理(3)

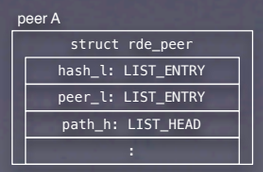
- prefix_link(...)
  - ❖ prefix 構造体と rib_entry 構造体, rde_aspath 構造体を接続し prefix_evaluate(...) を実行する.
- prefix_evaluate(...)
  - ❖ rib_entry の prefix_h(prefix 構造体のリスト) にベストな prefix 構造体が先頭にくるよう挿入し active がその prefix を向くようにする.
- prefix_cmp(...)
  - ❖ 2 つの prefix のうちどちらがベストかを判定する. NEXT_HOP の到達性, LOCAL_PREF 属性値, AS_PATH 属性の長さ, ORIGIN 属性値, MED 属性値, ... 等々を比較し決定する.

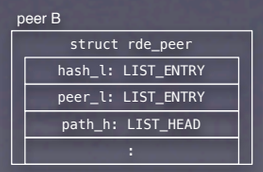
Example Scenario



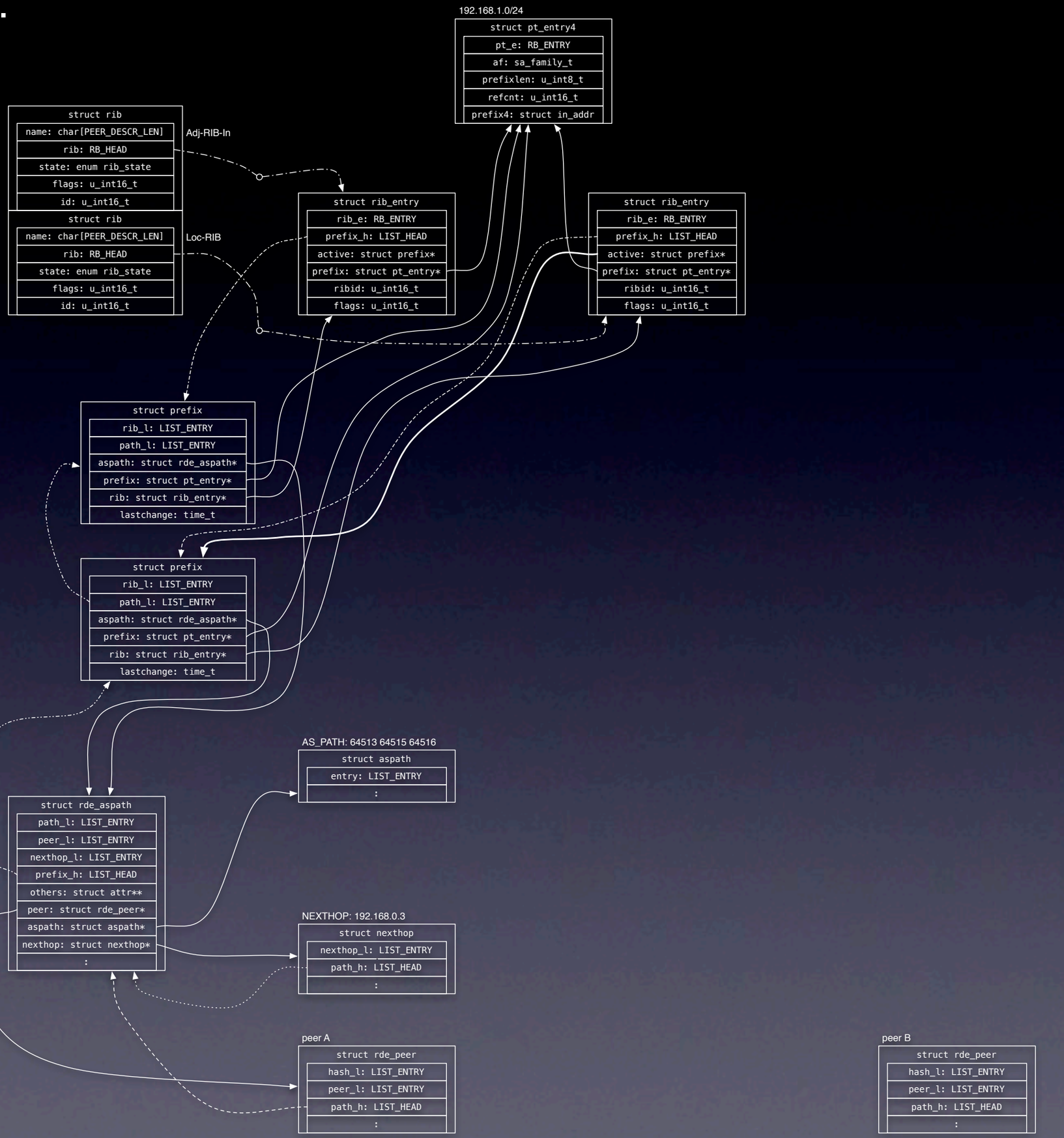
Step 0.


struct rib
name: char[PEER_DESCR_LEN]
rib: RB_HEAD
state: enum rib_state
flags: u_int16_t
id: u_int16_t
Adj-RIB-In
struct rib
name: char[PEER_DESCR_LEN]
rib: RB_HEAD
state: enum rib_state
flags: u_int16_t
id: u_int16_t
Loc-RIB

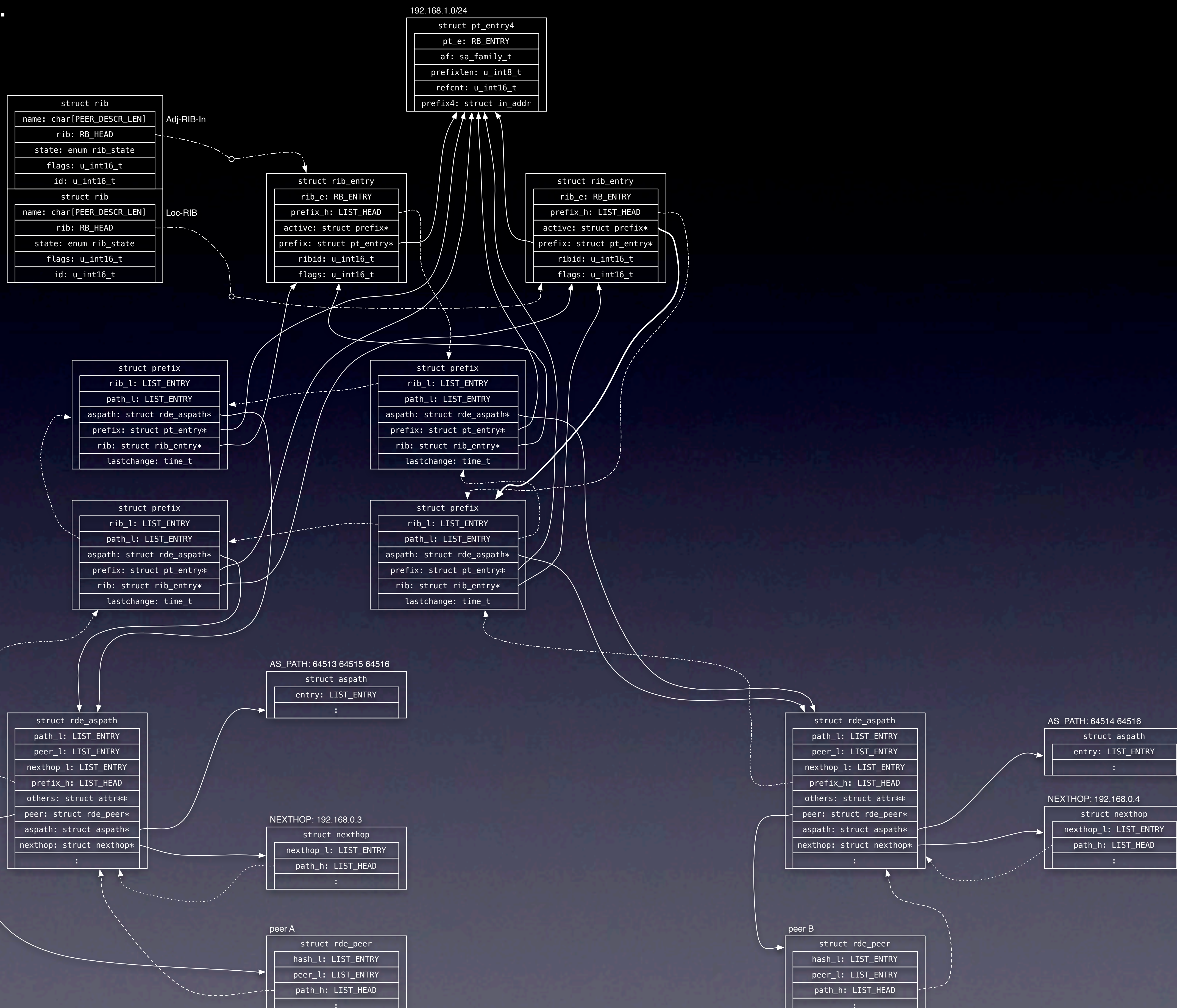

peer A
struct rde_peer
hash_l: LIST_ENTRY
peer_l: LIST_ENTRY
path_h: LIST_HEAD
:


peer B
struct rde_peer
hash_l: LIST_ENTRY
peer_l: LIST_ENTRY
path_h: LIST_HEAD
:

Step 1.

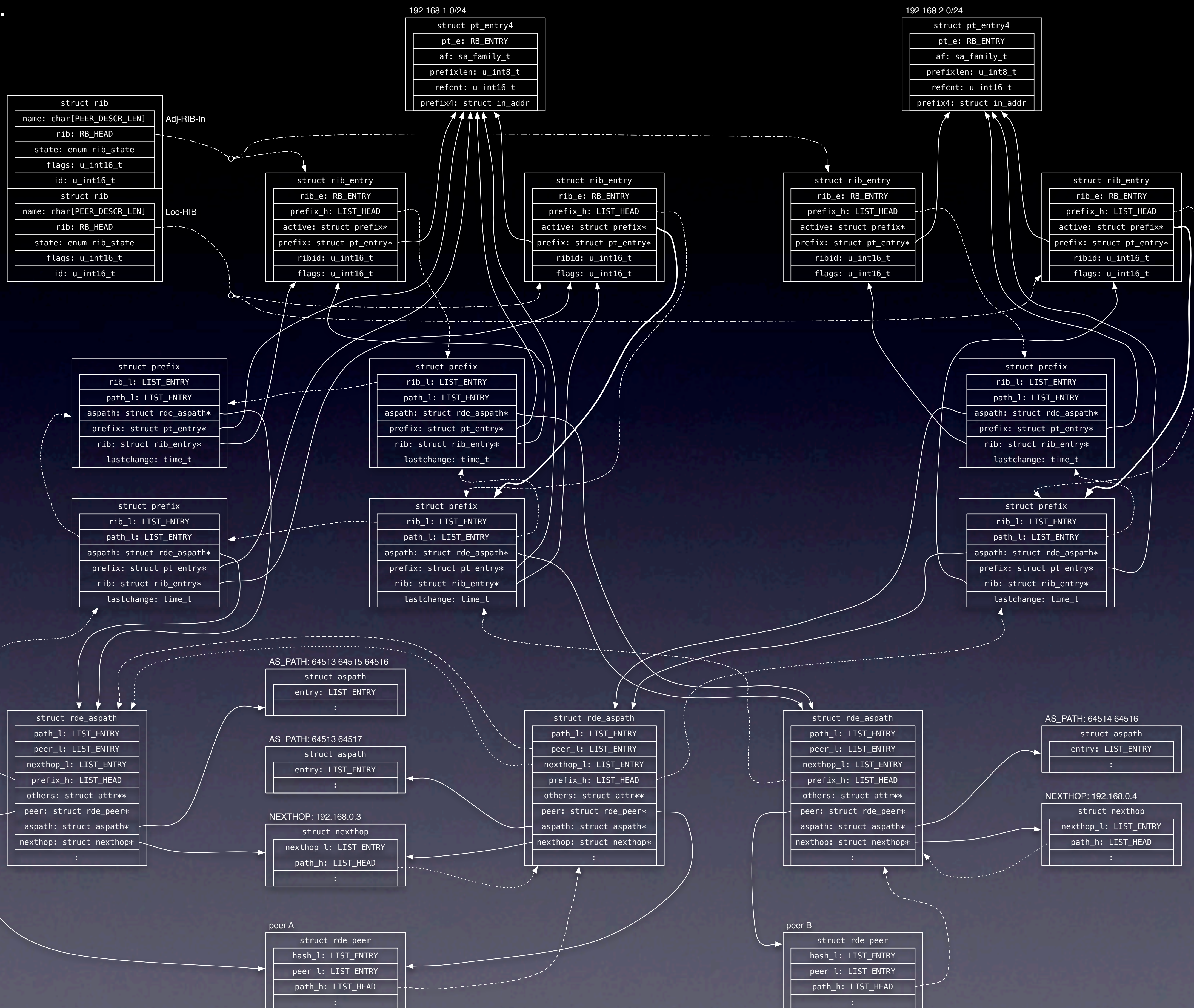

192.168.1.0/24
struct pt_entry4
pt_e: RB_ENTRY
af: sa_family_t
prefixlen: u_int8_t
refcnt: u_int16_t
prefix4: struct in_addr
struct rib
name: char[PEER_DESCR_LEN]
rib: RB_HEAD
state: enum rib_state
flags: u_int16_t
id: u_int16_t
Adj-RIB-In
struct rib
name: char[PEER_DESCR_LEN]
rib: RB_HEAD
state: enum rib_state
flags: u_int16_t
id: u_int16_t
Loc-RIB
struct rib_entry
rib_e: RB_ENTRY
prefix_h: LIST_HEAD
active: struct prefix*
prefix: struct pt_entry*
ribid: u_int16_t
flags: u_int16_t
struct rib_entry
rib_e: RB_ENTRY
prefix_h: LIST_HEAD
active: struct prefix*
prefix: struct pt_entry*
ribid: u_int16_t
flags: u_int16_t
struct prefix
rib_l: LIST_ENTRY
path_l: LIST_ENTRY
aspath: struct rde_aspath*
prefix: struct pt_entry*
rib: struct rib_entry*
lastchange: time_t
struct prefix
rib_l: LIST_ENTRY
path_l: LIST_ENTRY
aspath: struct rde_aspath*
prefix: struct pt_entry*
rib: struct rib_entry*
lastchange: time_t
AS_PATH: 64513 64515 64516
struct aspath
entry: LIST_ENTRY
:
struct rde_aspath
path_l: LIST_ENTRY
peer_l: LIST_ENTRY
nexthop_l: LIST_ENTRY
prefix_h: LIST_HEAD
others: struct attr**
peer: struct rde_peer*
aspath: struct aspath*
nexthop: struct nexthop*
:
NEXTHOP: 192.168.0.3
struct nexthop
nexthop_l: LIST_ENTRY
path_h: LIST_HEAD
:
peer A
struct rde_peer
hash_l: LIST_ENTRY
peer_l: LIST_ENTRY
path_h: LIST_HEAD
:
peer B
struct rde_peer
hash_l: LIST_ENTRY
peer_l: LIST_ENTRY
path_h: LIST_HEAD
:

Step 2.



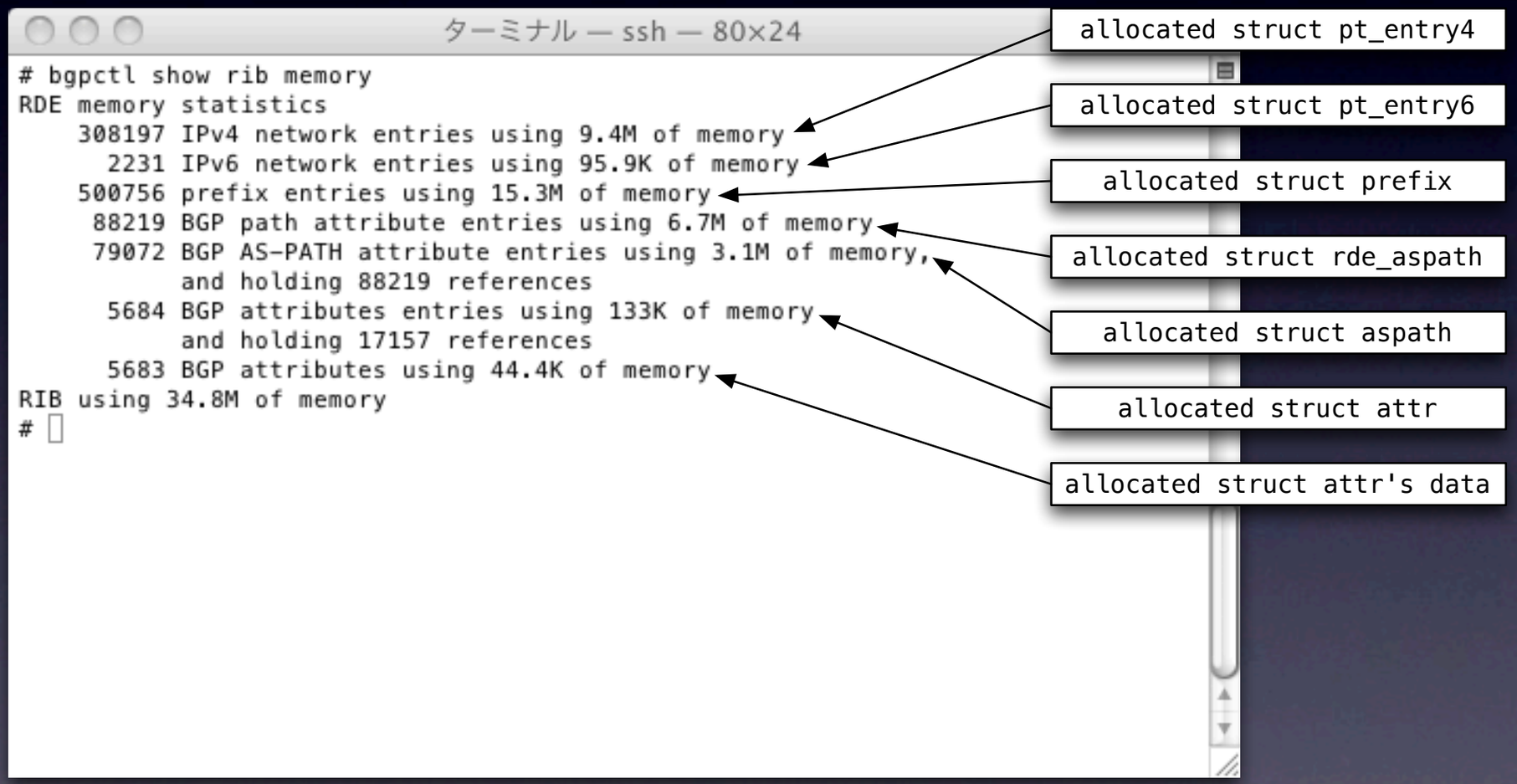
192.168.1.0/24
struct pt_entry4
pt_e: RB_ENTRY
af: sa_family_t
prefixlen: u_int8_t
refcnt: u_int16_t
prefix4: struct in_addr
struct rib
name: char[PEER_DESCR_LEN]
rib: RB_HEAD
state: enum rib_state
flags: u_int16_t
id: u_int16_t
Adj-RIB-In
struct rib
name: char[PEER_DESCR_LEN]
rib: RB_HEAD
state: enum rib_state
flags: u_int16_t
id: u_int16_t
Loc-RIB
struct rib_entry
rib_e: RB_ENTRY
prefix_h: LIST_HEAD
active: struct prefix*
prefix: struct pt_entry*
ribid: u_int16_t
flags: u_int16_t
struct rib_entry
rib_e: RB_ENTRY
prefix_h: LIST_HEAD
active: struct prefix*
prefix: struct pt_entry*
ribid: u_int16_t
flags: u_int16_t
struct prefix
rib_l: LIST_ENTRY
path_l: LIST_ENTRY
aspath: struct rde_aspath*
prefix: struct pt_entry*
rib: struct rib_entry*
lastchange: time_t
struct prefix
rib_l: LIST_ENTRY
path_l: LIST_ENTRY
aspath: struct rde_aspath*
prefix: struct pt_entry*
rib: struct rib_entry*
lastchange: time_t
struct prefix
rib_l: LIST_ENTRY
path_l: LIST_ENTRY
aspath: struct rde_aspath*
prefix: struct pt_entry*
rib: struct rib_entry*
lastchange: time_t
struct prefix
rib_l: LIST_ENTRY
path_l: LIST_ENTRY
aspath: struct rde_aspath*
prefix: struct pt_entry*
rib: struct rib_entry*
lastchange: time_t
AS_PATH: 64513 64515 64516
struct aspath
entry: LIST_ENTRY
:
struct rde_aspath
path_l: LIST_ENTRY
peer_l: LIST_ENTRY
nexthop_l: LIST_ENTRY
prefix_h: LIST_HEAD
others: struct attr**
peer: struct rde_peer*
aspath: struct aspath*
nexthop: struct nexthop*
:
NEXTHOP: 192.168.0.3
struct nexthop
nexthop_l: LIST_ENTRY
path_h: LIST_HEAD
:
peer A
struct rde_peer
hash_l: LIST_ENTRY
peer_l: LIST_ENTRY
path_h: LIST_HEAD
:
struct rde_aspath
path_l: LIST_ENTRY
peer_l: LIST_ENTRY
nexthop_l: LIST_ENTRY
prefix_h: LIST_HEAD
others: struct attr**
peer: struct rde_peer*
aspath: struct aspath*
nexthop: struct nexthop*
:
AS_PATH: 64514 64516
struct aspath
entry: LIST_ENTRY
:
NEXTHOP: 192.168.0.4
struct nexthop
nexthop_l: LIST_ENTRY
path_h: LIST_HEAD
:
peer B
struct rde_peer
hash_l: LIST_ENTRY
peer_l: LIST_ENTRY
path_h: LIST_HEAD
:

Step 3.


192.168.1.0/24
struct pt_entry4
pt_e: RB_ENTRY
af: sa_family_t
prefixlen: u_int8_t
refcnt: u_int16_t
prefix4: struct in_addr
192.168.2.0/24
struct pt_entry4
pt_e: RB_ENTRY
af: sa_family_t
prefixlen: u_int8_t
refcnt: u_int16_t
prefix4: struct in_addr
struct rib
name: char[PEER_DESCR_LEN]
rib: RB_HEAD
state: enum rib_state
flags: u_int16_t
id: u_int16_t
Adj-RIB-In
struct rib
name: char[PEER_DESCR_LEN]
rib: RB_HEAD
state: enum rib_state
flags: u_int16_t
id: u_int16_t
Loc-RIB
struct rib_entry
rib_e: RB_ENTRY
prefix_h: LIST_HEAD
active: struct prefix*
prefix: struct pt_entry*
ribid: u_int16_t
flags: u_int16_t
struct prefix
rib_l: LIST_ENTRY
path_l: LIST_ENTRY
aspath: struct rde_aspath*
prefix: struct pt_entry*
rib: struct rib_entry*
lastchange: time_t
struct rde_aspath
path_l: LIST_ENTRY
peer_l: LIST_ENTRY
nexthop_l: LIST_ENTRY
prefix_h: LIST_HEAD
others: struct attr**
peer: struct rde_peer*
aspath: struct aspath*
nexthop: struct nexthop*
:
AS_PATH: 64513 64515 64516
struct aspath
entry: LIST_ENTRY
:
AS_PATH: 64513 64517
struct aspath
entry: LIST_ENTRY
:
AS_PATH: 64514 64516
struct aspath
entry: LIST_ENTRY
:
NEXTHOP: 192.168.0.3
struct nexthop
nexthop_l: LIST_ENTRY
path_h: LIST_HEAD
:
NEXTHOP: 192.168.0.4
struct nexthop
nexthop_l: LIST_ENTRY
path_h: LIST_HEAD
:
peer A
struct rde_peer
hash_l: LIST_ENTRY
peer_l: LIST_ENTRY
path_h: LIST_HEAD
:
peer B
struct rde_peer
hash_l: LIST_ENTRY
peer_l: LIST_ENTRY
path_h: LIST_HEAD
:

Best path lookup for 192.168.1.0/24

RDE memory statistics



参考情報

- OpenBGPD - bringing full views to OpenBSD since 2004 by Claudio Jeker
  - http://2009.asiabsdcon.org/papers/abc2009-P2A-paper.pdf
  - ❖ AsiaBSDCon 2009 Tokyo, 12-15 March, 2009 の発表資料
- The Design and Implementation of OpenBGPd by André Oppermann and Claudio Jeker
  - http://www.networx.ch/The%20Design%20and%20Implementation%20of%20OpenBGPD.pdf
  - ❖ SWINOG-9 Berne, 29. September 2004 の発表資料
- A secure BGP implementation by Henning Brauer
  - http://2004.eurobsdcon.org/uploads/media/EBSD04_slides_44.pdf
  - ❖ EuroBSDCon 2004 in Karlsruhe, 2004 の発表資料
- RFC4271: A Border Gateway Protocol 4 (BGP-4)
  - http://tools.ietf.org/html/4271
- RFC4760 Multiprotocol Extensions for BGP-4
  - http://tools.ietf.org/html/4760
- RFC2545 Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing
  - http://tools.ietf.org/html/2545