OpenvSwitch で作る透過 L3DSR

Akira KAMIO

whoami

神尾　皓 (Akira KAMIO)

- 株式会社サイバーエージェント
  - CyberAgent group Infrastructure Unit (CIU)
  - Cycloud Div, Compute Team
- 2020/06 中途入社
  - 入社以来、一貫してIaaS基盤の開発/運用業務に従事
  - Qemu/KVM や高集約コンピュートノードの最適化に興味がある

Contents

CyberAgent

1

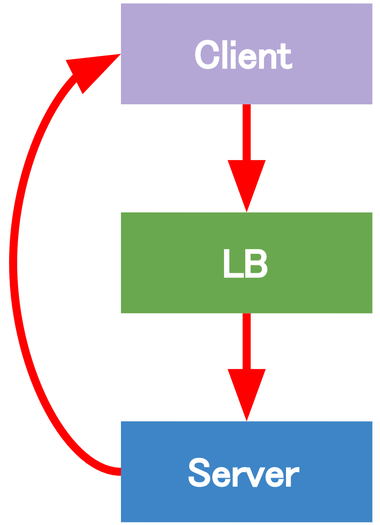
L3DSR とは

CyberAgent

L3DSR について

- Inline 構成
  - リクエストを LB で NAT して、レスポンスも LB を経由する

- L3DSR 構成
  - リクエストを LB で NAT せず real server へ送信
  - レスポンスは real server から直接返す
  - L3 を跨いだ NW 構成が可能 -> 柔軟性が高い



L3DSR の実装方法

- DSCP (TOS)
  - リクエストパケットに DSCP bit を付与して real server へ転送
- IPIP
  - リクエストパケットを IPIP でカプセルして real server へ転送
- GRE
  - リクエストパケットを GRE でカプセルして real server へ転送

L3DSR の課題

- サーバ側に L3DSR の設定が必要
  - DSCP -> iptable で特定の DSCP bit が立っているときに曲げる
  - IPIP -> IPIP トンネル I/F を作成
  - GRE -> GRE トンネル I/F を作成

L3DSR の課題

- サーバ側に L3DSR の設定が必要
  - DSCP -> iptable で特定の DSCP bit が立っているときに曲げる
  - IPIP -> IPIP トンネル I/F を作成
  - GRE -> GRE トンネル I/F を作成

サーバに特別な設定を入れずに　L3DSR できたら嬉しい！

2

透過 L3DSR の設計

透過 L3DSR の要件

・サーバに特別な設定を入れなくて良いようにする

＝＞ハイパーバイザ上で

・リクエスト：VIP 宛の dsr IP を Server IP に変換

・レスポンス：Client 宛のパケットの src IP を VIP に変換

透過 L3DSR の要件

・サーバに特別な設定を入れなくて良いようにする

＝＞ハイパーバイザ上で

・リクエスト：VIP 宛の dsr IP を Server IP に変換

・レスポンス：Client 宛のパケットの src IP を VIP に変換

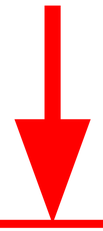
透過 L3DSR の要件

・サーバに特別な設定を入れなくて良いようにする

＝＞ハイパーバイザ上で

・リクエスト：VIP 宛の dsr IP を Server IP に変換

・レスポンス：Client 宛のパケットの src IP を VIP に変換



L3DSR 宛の変換したパケットをすべて覚えている必要がある

3

Open vSwitch 実装

CyberAgent

第1稿: DSCP パターン

・まずは DSCP パターンについて検討した

CLI -> LB
SRC IP: CLI IP
DST IP: VIP

LB -> OVS
SRC IP: CLI IP
DST IP: VM IP
TOS: 32

OVS -> VM
SRC IP: CLI IP
DST IP: VM IP

VM -> OVS
SRC IP: VM IP
DST IP: CLI IP

OVS -> CLI
SRC IP: VIP
DST IP: CLI IP

・あくまで色付けとしての利用のため既存の NW に影響しないように設定する必要がある
  ・IP Precedence 部をフルビットにしない
    ・IP Precedence 部フルビットはネットワーク機器によっては制御パケットとして認識されてしまう
  ・そのため DSCP は 1-47（48 から IP Precedence 部がフルビット）を色付け用として用いる

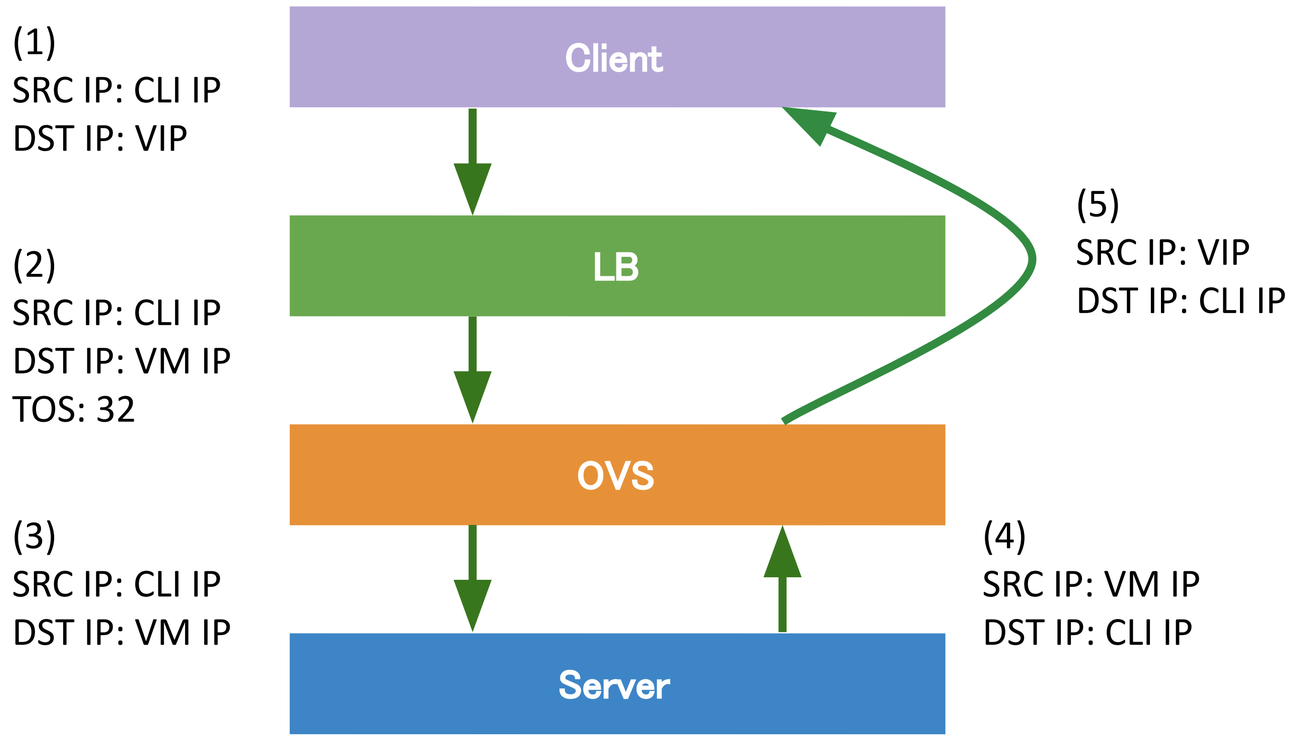
例：
CLI IP: 124.0.124.100
VIP: 203.80.24.55
VM IP: 10.200.100.88
TOS: 32

TOS の値が 32 のときに VIP 宛の通信を reg0 に記録
ovs-ofctl add-flow br0 "table=0, priority=100, ip, nw_dst=10.200.100.88
actions=load:NXM_NX_IP_TOS[]->NXM_NX_REG0[], goto_table:1"

reg0 が 32（TOS の値）のときに SRC IP を VIP に書き換える
ovs-ofctl add-flow br0 "table=1, priority=110, ip, nw_src=10.200.100.88, reg0=0x20
actions=mod_nw_src:203.80.24.55, normal"

第1稿: DSCP パターン

・まずは DSCP パターンについて検討した



第1稿: DSCP パターン

・まずは DSCP パターンについて検討した

CLI -> LB
SRC IP: CLI IP
DST IP: VIP

LB -> OVS
SRC IP: CLI IP
DST IP: VM IP
TOS: 32

OVS -> VM
SRC IP: CLI IP
DST IP: VM IP

VM -> OVS
SRC IP: VM IP
DST IP: CLI IP

OVS -> CLI
SRC IP: VIP
DST IP: CLI IP

・あくまで色付けとしての利用のため既存の NW に影響しないように設定する必要がある
  ・IP Precedence 部をフルビットにしない
    ・IP Precedence 部フルビットはネットワーク機器によっては制御パケットとして認識されてしまう
  ・そのため DSCP は 1-47（48 から IP Precedence 部がフルビット）を色付け用として用いる

例：
CLI IP: 124.0.124.100
VIP: 203.80.24.55
VM IP: 10.200.100.88
TOS: 32

TOS の値が 32 のときに VIP 宛の通信を reg0 に記録
ovs-ofctl add-flow br0 "table=0, priority=100, ip, nw_dst=10.200.100.88
actions=load:NXM_NX_IP_TOS[]->NXM_NX_REG0[], goto_table:1"

reg0 が 32（TOS の値）のときに SRC IP を VIP に書き換える
ovs-ofctl add-flow br0 "table=1, priority=110, ip, nw_src=10.200.100.88, reg0=0x20
actions=mod_nw_src:203.80.24.55, normal"

第1稿: DSCP パターン

ボツ

・まずは DSCP パターンについて検討した

CLI -> LB
SRC IP: CLI IP
DST IP: VIP

LB -> OVS
SRC IP: CLI IP
DST IP: VM IP
TOS: 32

OVS -> VM
SRC IP: CLI IP
DST IP: VM IP

VM -> OVS
SRC IP: VM IP
DST IP: CLI IP

OVS -> CLI
SRC IP: VIP
DST IP: CLI IP

・あくまで色付けとしての利用のため既存の NW に影響しないように設定する必要がある
  ・IP Precedence 部をフルビットにしない
    ・IP Precedence 部フルビットはネットワーク機器によっては制御パケットとして認識されてしまう
  ・そのため DSCP は 1-47（48 から IP Precedence 部がフルビット）を色付け用として用いる

例：
CLI IP: 124.0.124.100
VIP: 203.80.24.55
VM IP: 10.200.100.88
TOS: 32

TOS の値が 32 のときに VIP 宛の通信を reg0 に記録
ovs-ofctl add-flow br0 "table=0, priority=100, ip, nw_dst=10.200.100.88
actions=load:NXM_NX_IP_TOS[]->NXM_NX_REG0[], goto_table:1"

reg0 が 32（TOS の値）のときに SRC IP を VIP に書き換える
ovs-ofctl add-flow br0 "table=1, priority=110, ip, nw_src=10.200.100.88, reg0=0x20
actions=mod_nw_src:203.80.24.55, normal"

第1稿: DSCP パターン

ボツ

・まずは DSCP パターンについて検討した

CLI -> LB
SRC IP: CLI IP
DST IP: VIP

LB -> OVS
SRC IP: CLI IP
DST IP: VM IP
TOS: 32

OVS -> VM
SRC IP: CLI IP

SRC IP: VIP
DST IP: CLI IP

・あくまで色付けとしての利用のため既存の NW に影響しないように設定する必要がある
  ・IP Precedence 部をフルビットにしない
    ・IP Precedence 部フルビットはネットワーク機器によっては制御パケットとして認識されてしまう
  ・そのため DSCP は 1-47（48 から IP Precedence 部がフルビット）を色付け用として用いる

例 :
CLI IP: 124.0.124.100
VIP: 203.80.24.55
VM IP: 10.200.100.88
TOS: 32

DSCP bit と VIP の紐づけを管理するエージェントが必要
OVS の flow rule だけで実現する方法があるはず …！

actions=mod_nw_src:203.80.24.55, normal"

第2稿: IPIP パターン

- IPIP であれば VIP の紐づけを管理しなくて良い…！

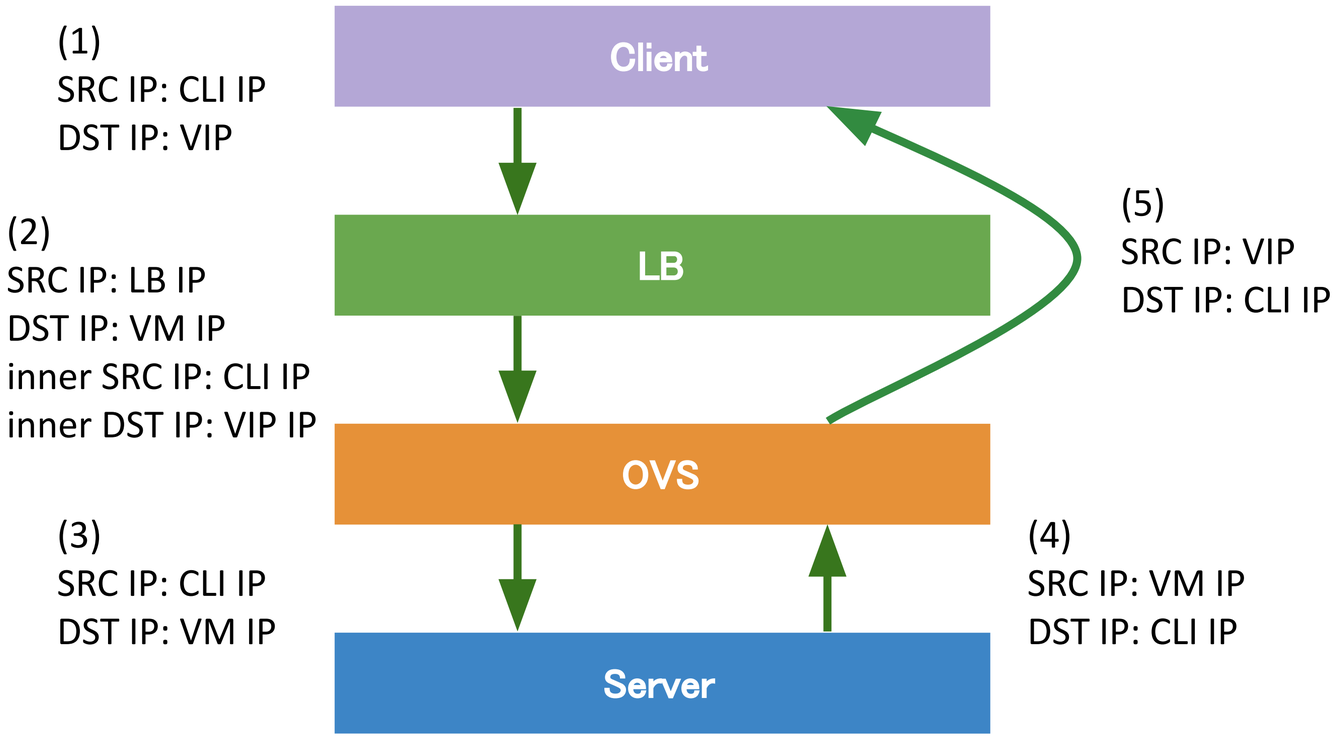
CLI -> LB
SRC IP: CLI IP
DST IP: VIP

LB -> OVS
SRC IP: VIP IP
DST IP: VM IP
inner SRC IP: CLI IP
inner DST IP: VIP IP

OVS -> VM
SRC IP: CLI IP
DST IP: VM IP

VM -> OVS
SRC IP: VM IP
DST IP: CLI IP

OVS -> CLI
SRC IP: VIP
DST IP: CLI IP



第2稿: IPIP パターン

- IPIP であれば VIP の紐づけを管理しなくて良い…！

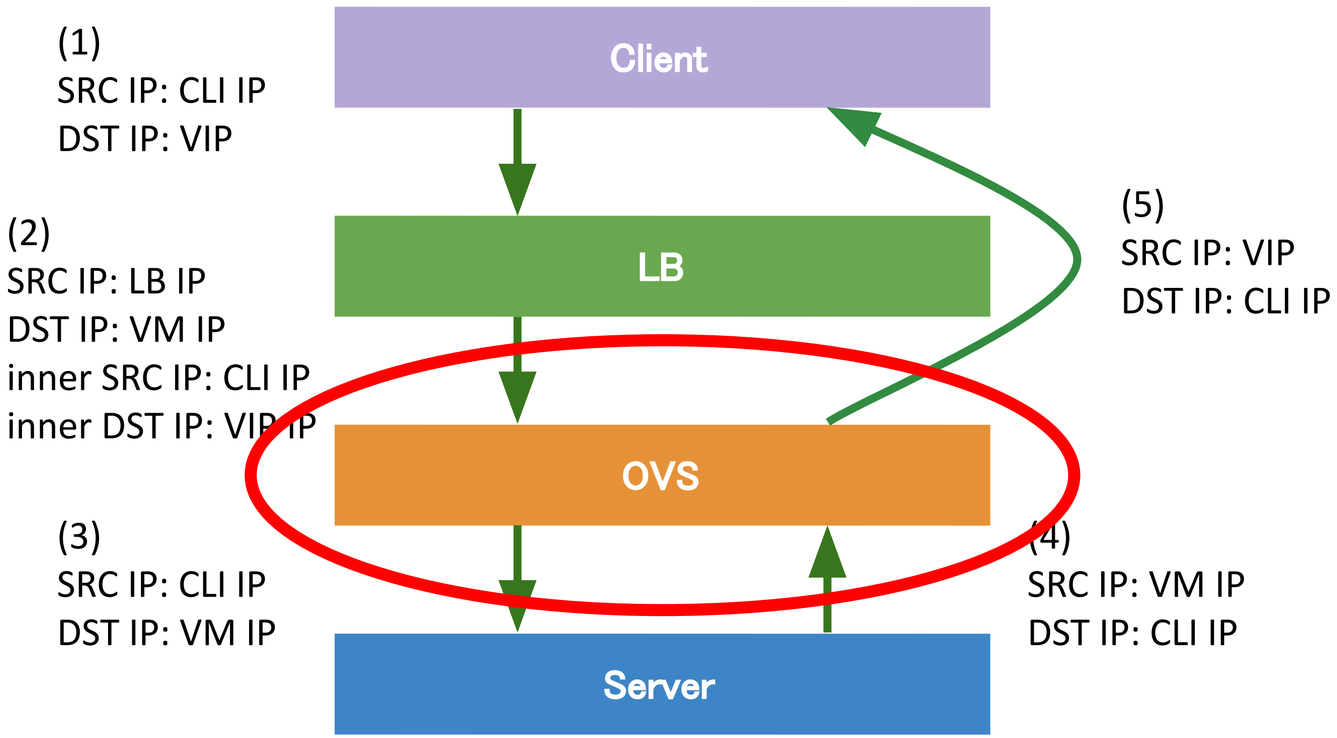
CLI -> LB
SRC IP: CLI IP
DST IP: VIP

LB -> OVS
SRC IP: VIP IP
DST IP: VM IP
inner SRC IP: CLI IP
inner DST IP: VIP IP

OVS -> VM
SRC IP: CLI IP
DST IP: VM IP

VM -> OVS
SRC IP: VM IP
DST IP: CLI IP

OVS -> CLI
SRC IP: VIP
DST IP: CLI IP



第2稿: IPIP パターン

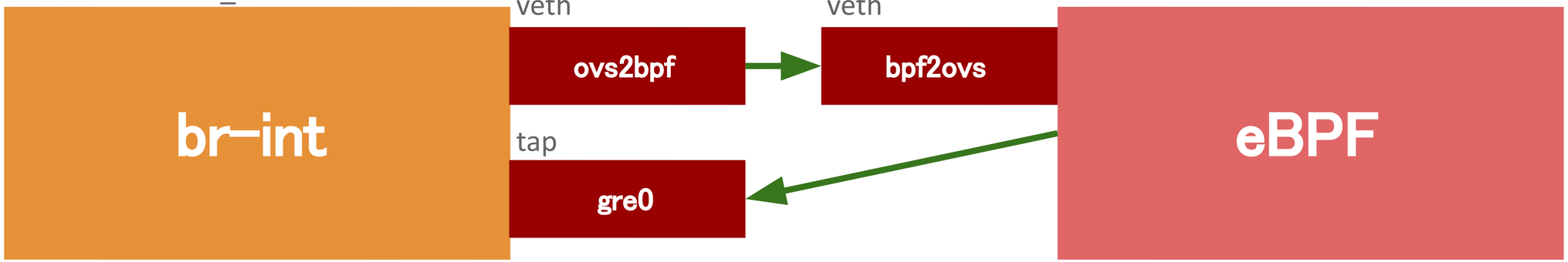
・IPIP であれば VIP の紐づけを管理しなくて良い…！

OVS

OVS に入ってくるパケット
SRC IP: LB_IP
DST IP: VM_IP
inner SRC IP: CLI_IP
inner DST IP: VIP_IP

(1) ip_proto=4, src=LB_IP

(2) eBPF で IPIP パケットを GRE パケットに変換、再送



(3) conntrack, ct_mark=0x04 を記録tun_src->reg0 (CLI_IP), tun_dst->reg1 (VIP_IP)
SRC: CLI_IP, DST: VM_IP のパケットを action NORMAL に託す

(4) 返りのパケットは SRC: VM_IP, DST: CLI_IP
conntrack に載っている && ct_mark=0x04 のとき SRC IP を reg0 (VIP IP) に書き換え

第2稿: IPIP パターン

・IPIP であれば VIP の紐づけを管理しなくて良い…！

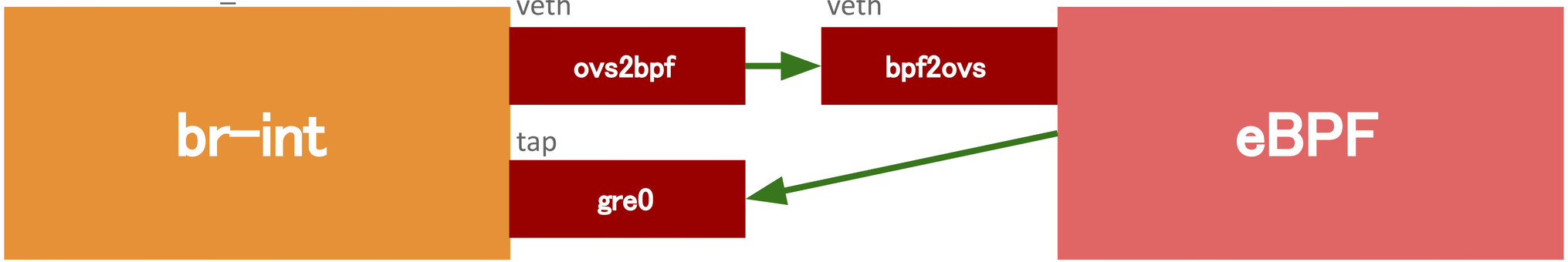
OVS

OVS に入ってくるパケット
SRC IP: LB_IP
DST IP: VM_IP
inner SRC IP: CLI_IP
inner DST IP: VIP_IP

OVS の flow rule だけで完結して上手くいきそう …！

(1) ip_proto=4, src=LB_IP

(2) eBPF で IPIP パケットを GRE パケットに変換、再送



(3) conntrack, ct_mark=0x04 を記録tun_src->reg0 (CLI_IP), tun_dst->reg1 (VIP_IP)
SRC: CLI_IP, DST: VM_IP のパケットを action NORMAL に託す

(4) 返りのパケットは SRC: VM_IP, DST: CLI_IP
conntrack に載っている && ct_mark=0x04 のとき SRC IP を reg0 (VIP IP) に書き換え

第2稿: IPIP ーン

ボツ

・IPIP であれば VIP の紐づけを管理しなくて良い…！

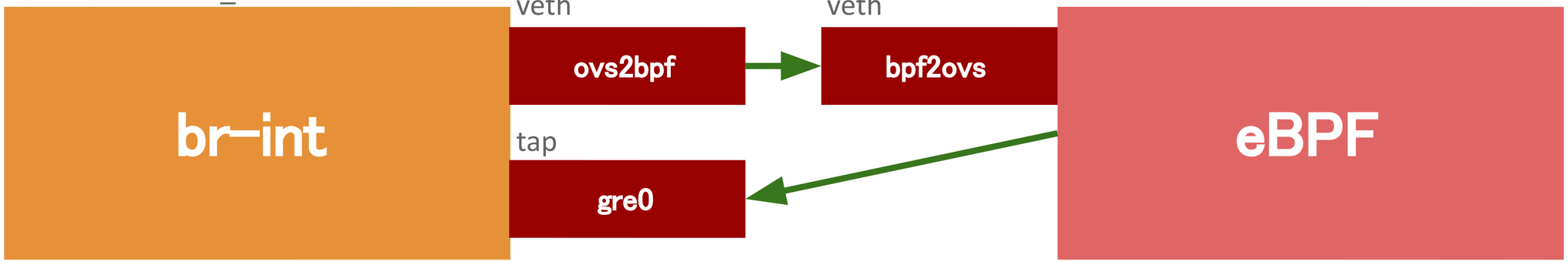
OVS

OVS に入ってくるパケット
SRC IP: LB_IP
DST IP: VM_IP
inner SRC IP: CLI_IP
inner DST IP: VIP_IP

OVS の flow rule だけで完結して上手くいきそう …！

(1) ip_proto=4, src=LB_IP

(2) eBPF で IPIP パケットを GRE パケットに変換、再送



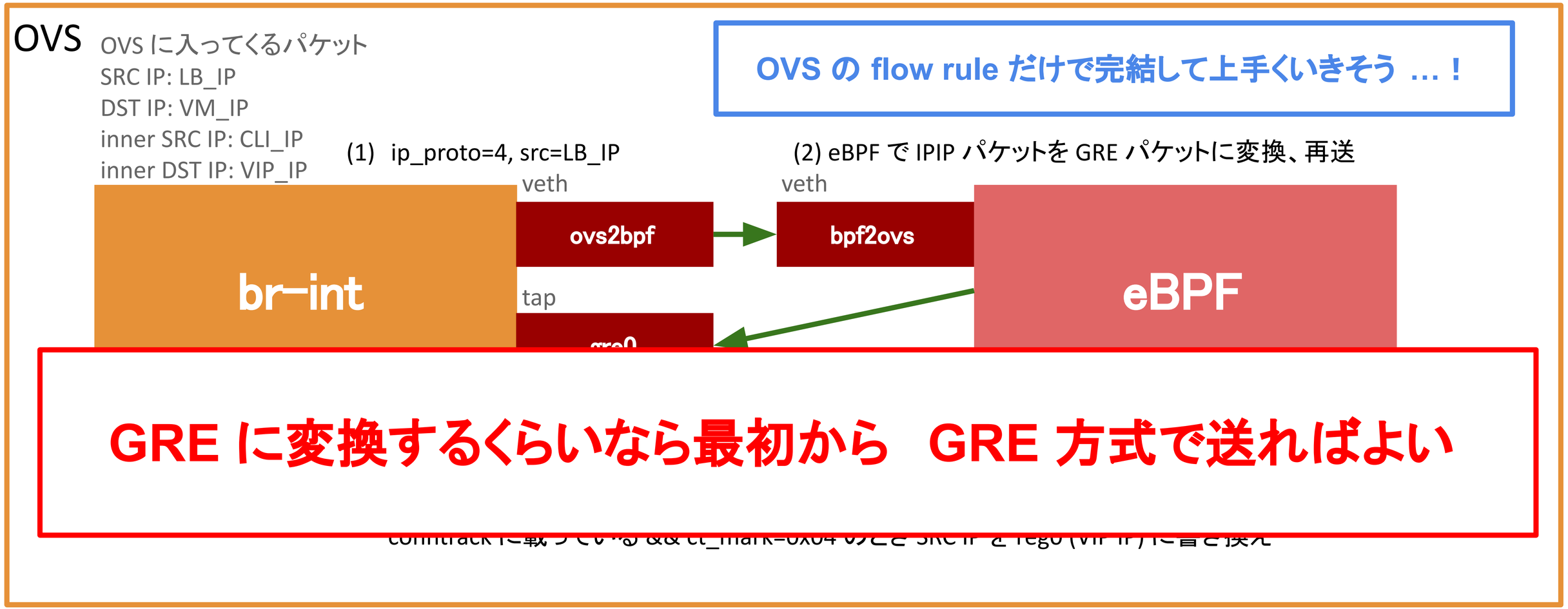
(3) conntrack, ct_mark=0x04 を記録tun_src->reg0 (CLI_IP), tun_dst->reg1 (VIP_IP)
SRC: CLI_IP, DST: VM_IP のパケットを action NORMAL に託す

(4) 返りのパケットは SRC: VM_IP, DST: CLI_IP
conntrack に載っている && ct_mark=0x04 のとき SRC IP を reg0 (VIP IP) に書き換え

第2稿: IPIP パターン

ボツ

・IPIP であれば VIP の紐づけを管理しなくて良い…！



第3稿: GRE パターン

・パケットの流れは IPIP とほぼ変わらない（IPIP -> GRE になる）

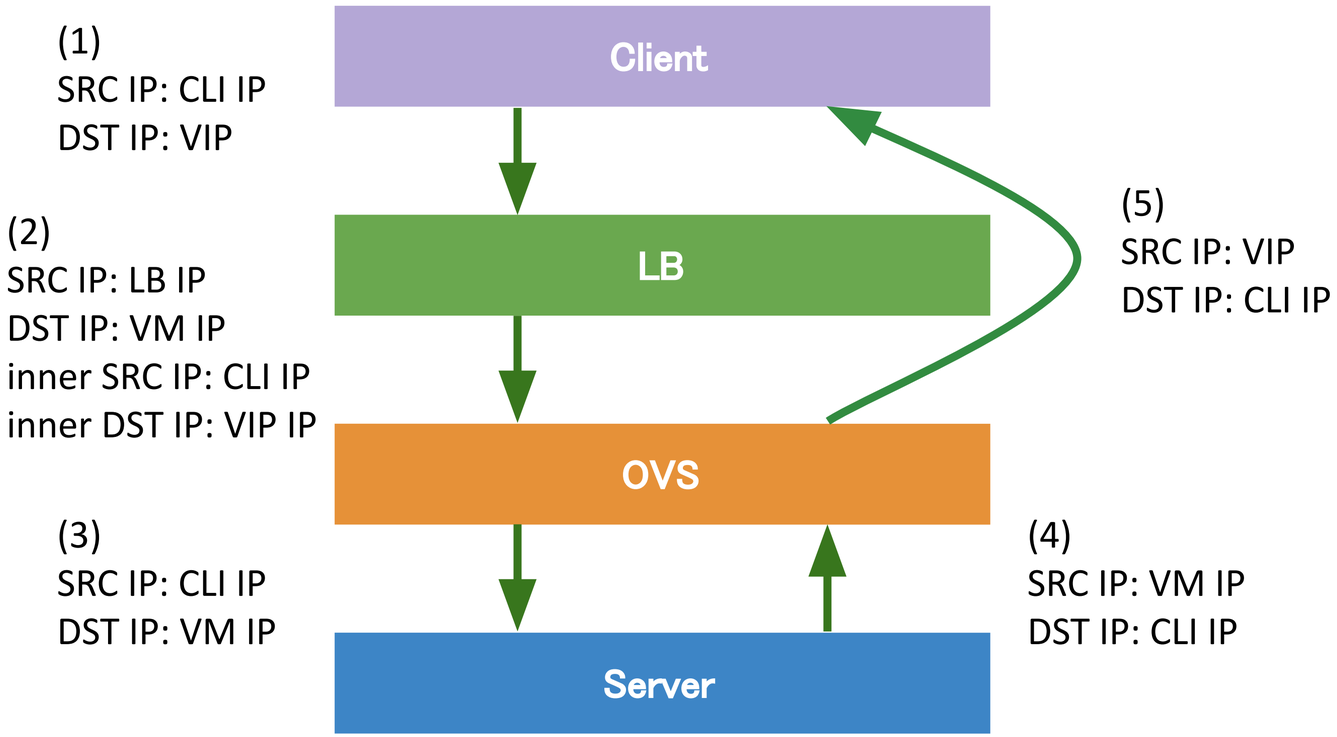
CLI -> LB
SRC IP: CLI IP
DST IP: VIP

LB -> OVS
SRC IP: VIP IP
DST IP: VM IP
inner SRC IP: CLI IP
inner DST IP: VIP IP

OVS -> VM
SRC IP: CLI IP
DST IP: VM IP

VM -> OVS
SRC IP: VM IP
DST IP: CLI IP

OVS -> CLI
SRC IP: VIP
DST IP: CLI IP



第3稿: GRE パターン

・パケットの流れは IPIP とほぼ変わらない（IPIP -> GRE になる）

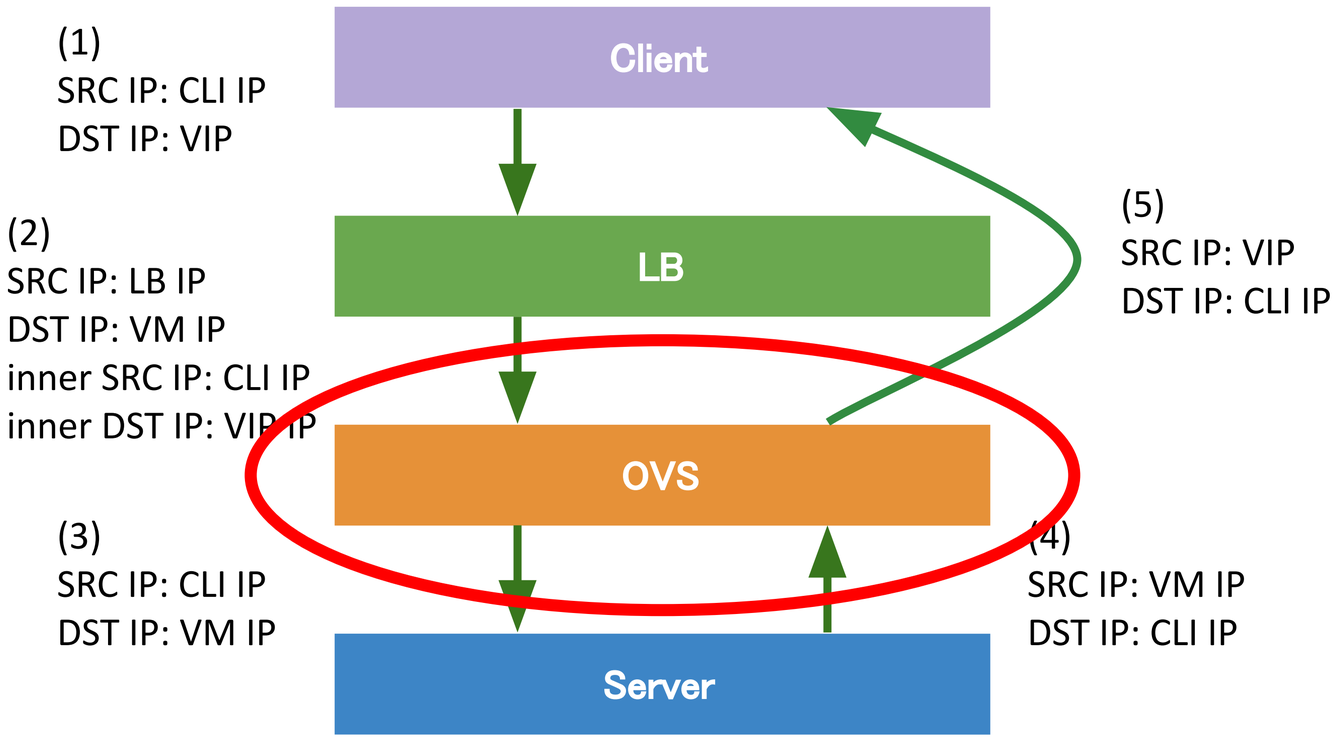
CLI -> LB
SRC IP: CLI IP
DST IP: VIP

LB -> OVS
SRC IP: VIP IP
DST IP: VM IP
inner SRC IP: CLI IP
inner DST IP: VIP IP

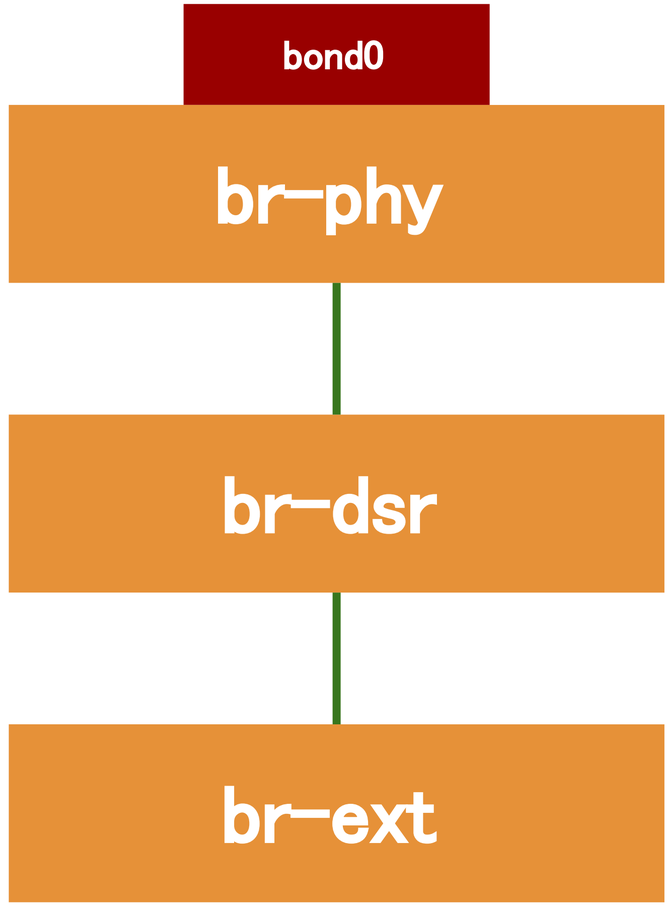
OVS -> VM
SRC IP: CLI IP
DST IP: VM IP

VM -> OVS
SRC IP: VM IP
DST IP: CLI IP

OVS -> CLI
SRC IP: VIP
DST IP: CLI IP



第3稿: GRE パターン



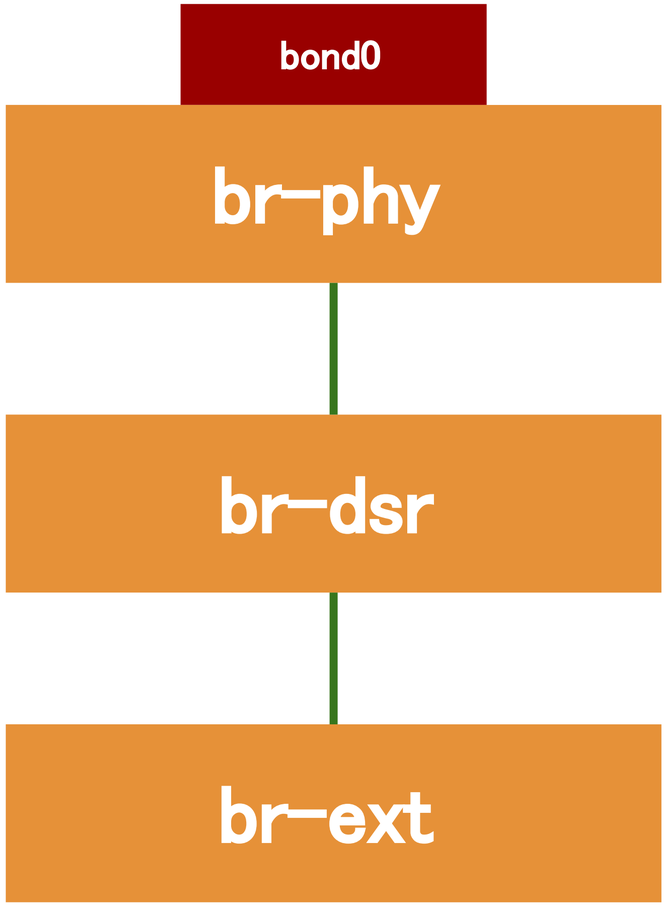
(1) GRE かつ LB IP の場合 decap アクションで GRE decap
ip_proto=47, src=LB_IP, actions=decap

(2) conntrack zone=47, VIP を CT_MARK に記録
move:NXM_OF_IP_DST[]->NXM_NX_CT_LABEL[0..31]

(3) conntrack zone=47 のものの IP SRC を CT_MARK に記録していた VIP に書き換え
move:NXM_NX_CT_LABEL[0..31]->NXM_OF_IP_SRC[]

第3稿: GRE パターン

OVS



机上では上手く行きそう …！

(1) GRE かつ LB IP の場合 decap アクションで GRE decap
ip_proto=47, src=LB_IP, actions=decap

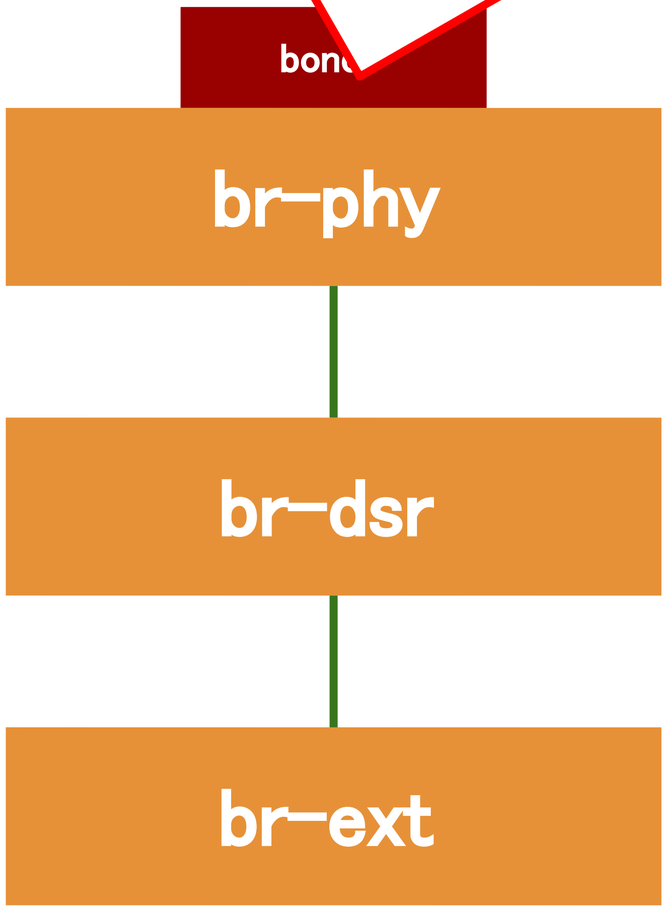
(2) conntrack zone=47, VIP を CT_MARK に記録
move:NXM_OF_IP_DST[]->NXM_NX_CT_LABEL[0..31]

(3) conntrack zone=47 のものの IP SRC を CT_MARK に記録していた VIP に書き換え
move:NXM_NX_CT_LABEL[0..31]->NXM_OF_IP_SRC[]

第3稿: GRE ーン

ボツ

OVS



机上では上手く行きそう …！

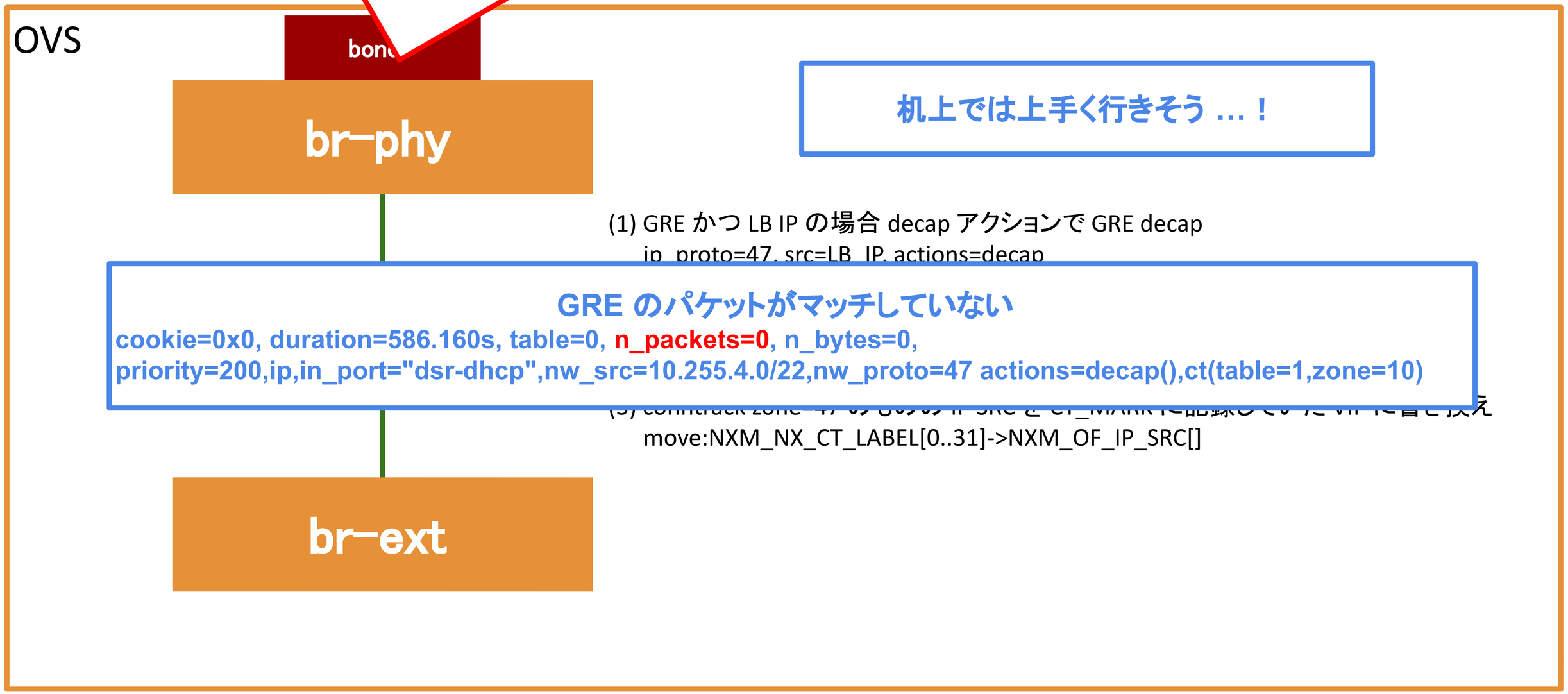
(1) GRE かつ LB IP の場合 decap アクションで GRE decap
ip_proto=47, src=LB_IP, actions=decap

(2) conntrack zone=47, VIP を CT_MARK に記録
move:NXM_OF_IP_DST[]->NXM_NX_CT_LABEL[0..31]

(3) conntrack zone=47 のものの IP SRC を CT_MARK に記録していた VIP に書き換え
move:NXM_NX_CT_LABEL[0..31]->NXM_OF_IP_SRC[]

第3稿: GRE ーン

ボツ



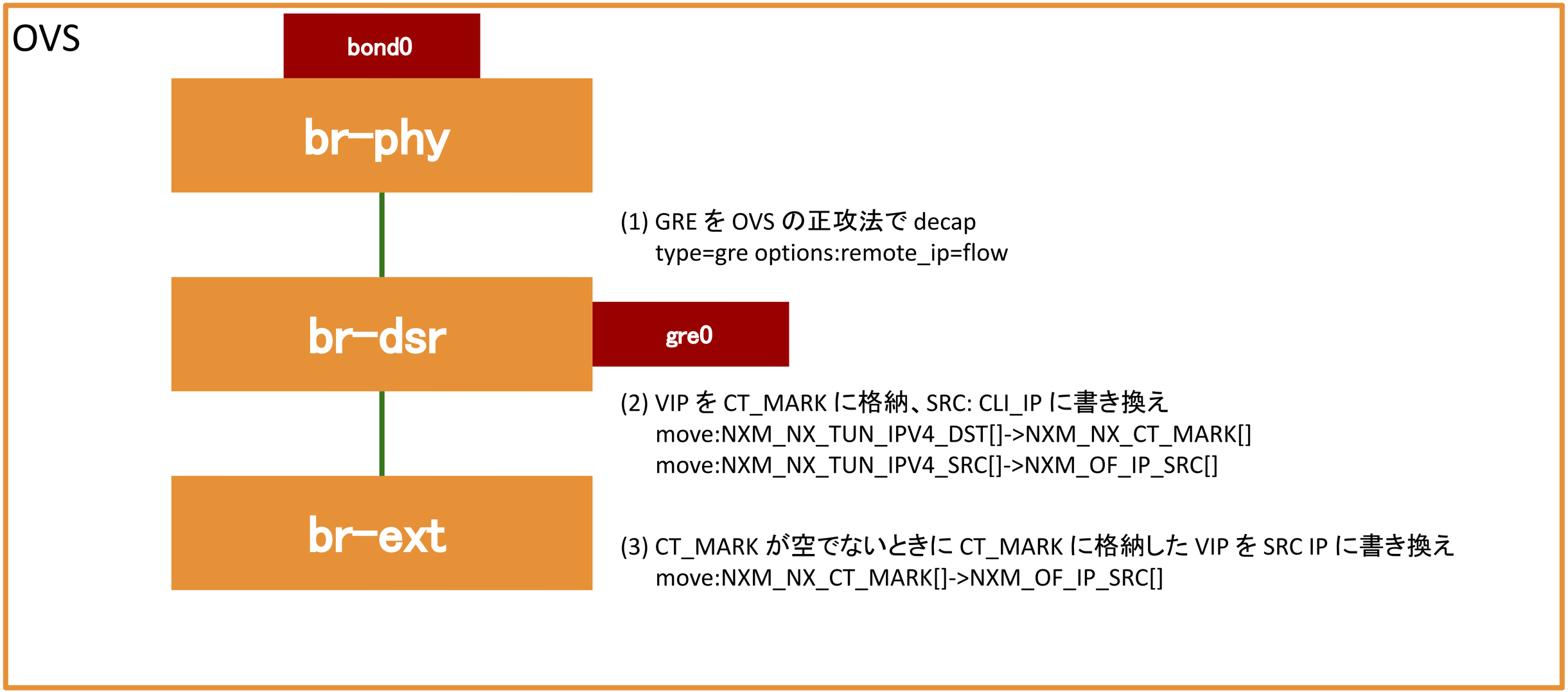
机上では上手く行きそう …！

第3稿: GRE ーン

ボツ


OVS
bond
br-phy
机上では上手く行きそう …！
(1) GRE かつ LB IP の場合 decap アクションで GRE decap
ip, proto=47, src=LB_IP, actions=decap
GRE のパケットがマッチしていない
cookie=0x0, duration=586.160s, table=0, n_packets=0, n_bytes=0,
priority=200,ip,in_port="dsr-dhcp",nw_src=10.255.4.0/22,nw_proto=47 actions=decap(),ct(table=1,zone=10)
OpenvSwitch で GRE は decap アクションで decap できない

第N稿: GRE パターン

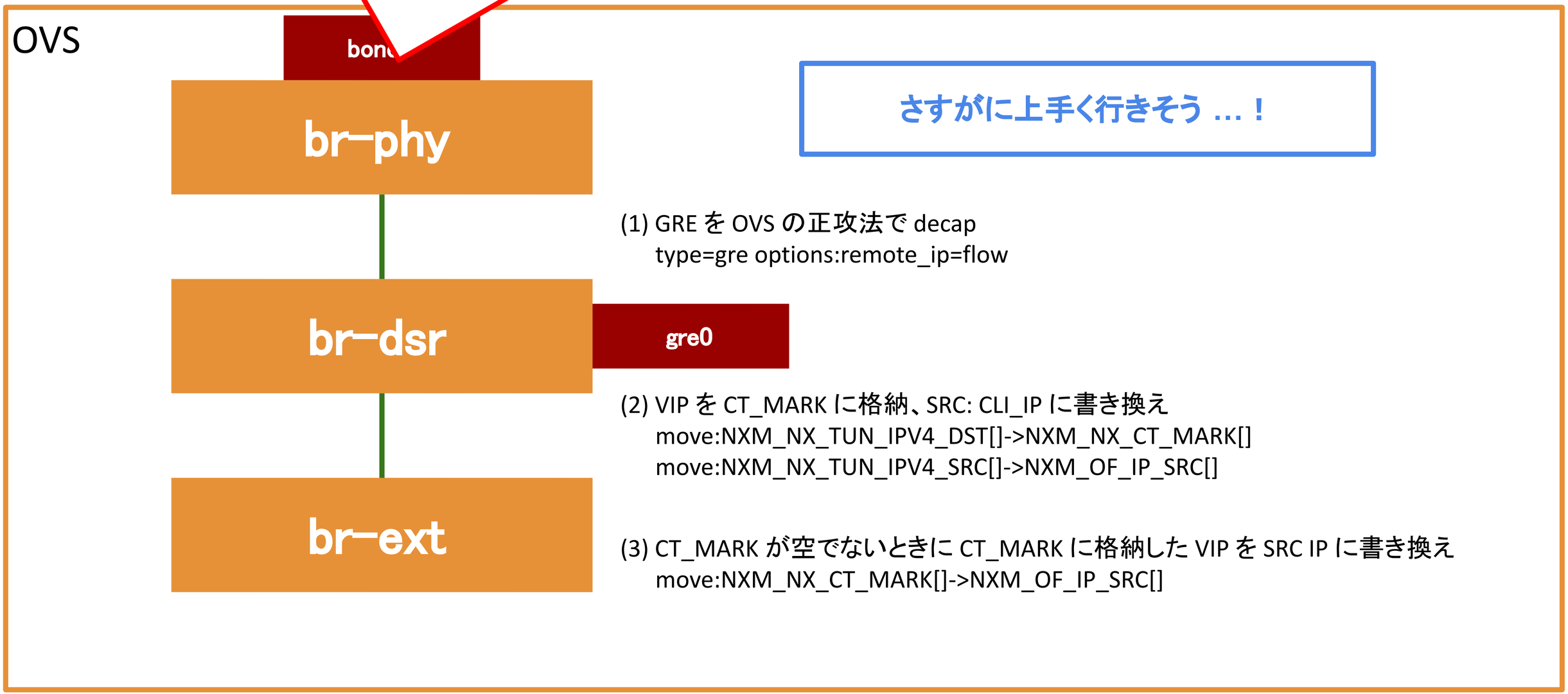

OVS
bond0
br-phy
br-dsr
gre0
br-ext
(1) GRE を OVS の正攻法で decap
type=gre options:remote_ip=flow
(2) VIP を CT_MARK に格納、SRC: CLI_IP に書き換え
move:NXM_NX_TUN_IPV4_DST[]->NXM_NX_CT_MARK[]
move:NXM_NX_TUN_IPV4_SRC[]->NXM_OF_IP_SRC[]
(3) CT_MARK が空でないときに CT_MARK に格納した VIP を SRC IP に書き換え
move:NXM_NX_CT_MARK[]->NXM_OF_IP_SRC[]

第N稿: GRE パターン


OVS
bond0
br-phy
br-dsr
gre0
br-ext
さすがに上手く行きそう …！
(1) GRE を OVS の正攻法で decap
type=gre options:remote_ip=flow
(2) VIP を CT_MARK に格納、SRC: CLI_IP に書き換え
move:NXM_NX_TUN_IPV4_DST[]->NXM_NX_CT_MARK[]
move:NXM_NX_TUN_IPV4_SRC[]->NXM_OF_IP_SRC[]
(3) CT_MARK が空でないときに CT_MARK に格納した VIP を SRC IP に書き換え
move:NXM_NX_CT_MARK[]->NXM_OF_IP_SRC[]

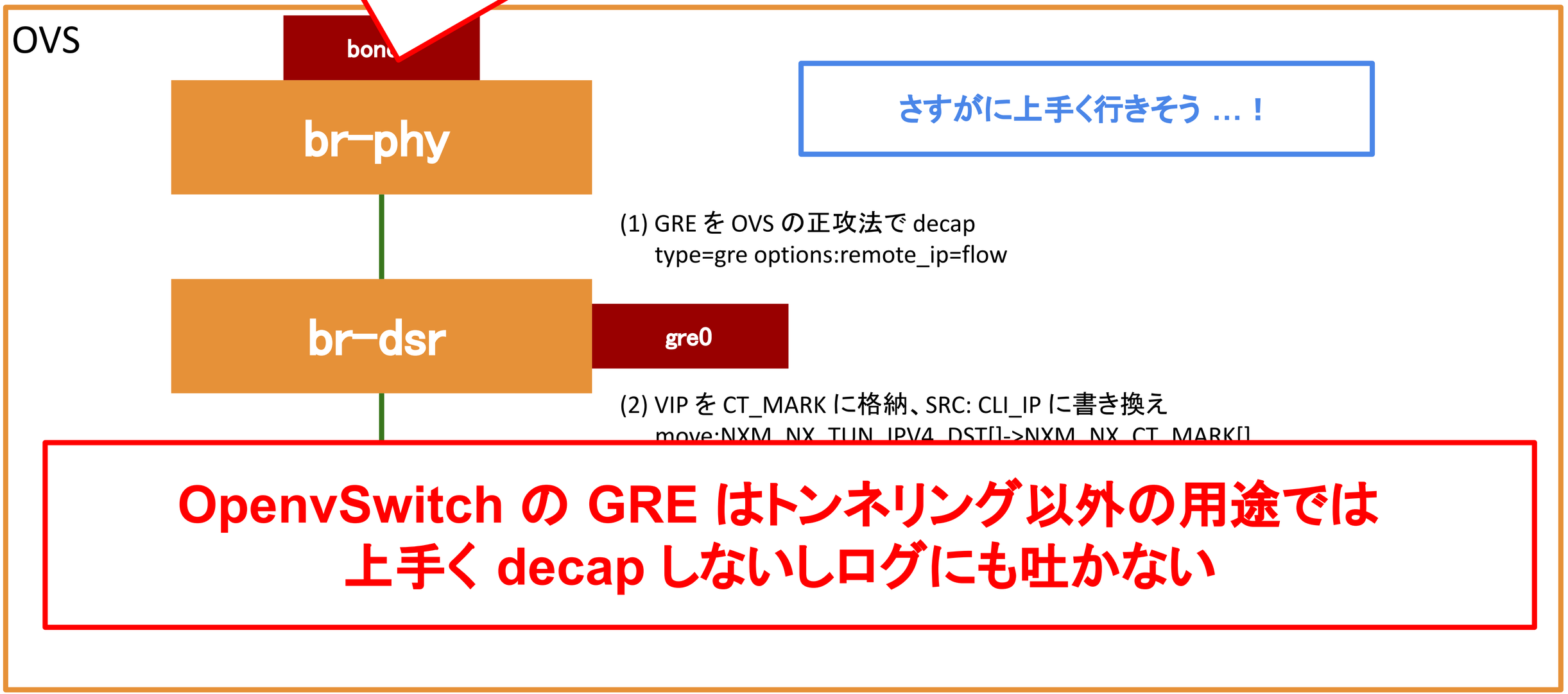
第N稿: GRE ーン

ボツ


OVS
bond
br-phy
br-dsr
gre0
br-ext
さすがに上手く行きそう …！
(1) GRE を OVS の正攻法で decap
type=gre options:remote_ip=flow
(2) VIP を CT_MARK に格納、SRC: CLI_IP に書き換え
move:NXM_NX_TUN_IPV4_DST[]->NXM_NX_CT_MARK[]
move:NXM_NX_TUN_IPV4_SRC[]->NXM_OF_IP_SRC[]
(3) CT_MARK が空でないときに CT_MARK に格納した VIP を SRC IP に書き換え
move:NXM_NX_CT_MARK[]->NXM_OF_IP_SRC[]

第N稿: GRE ーン

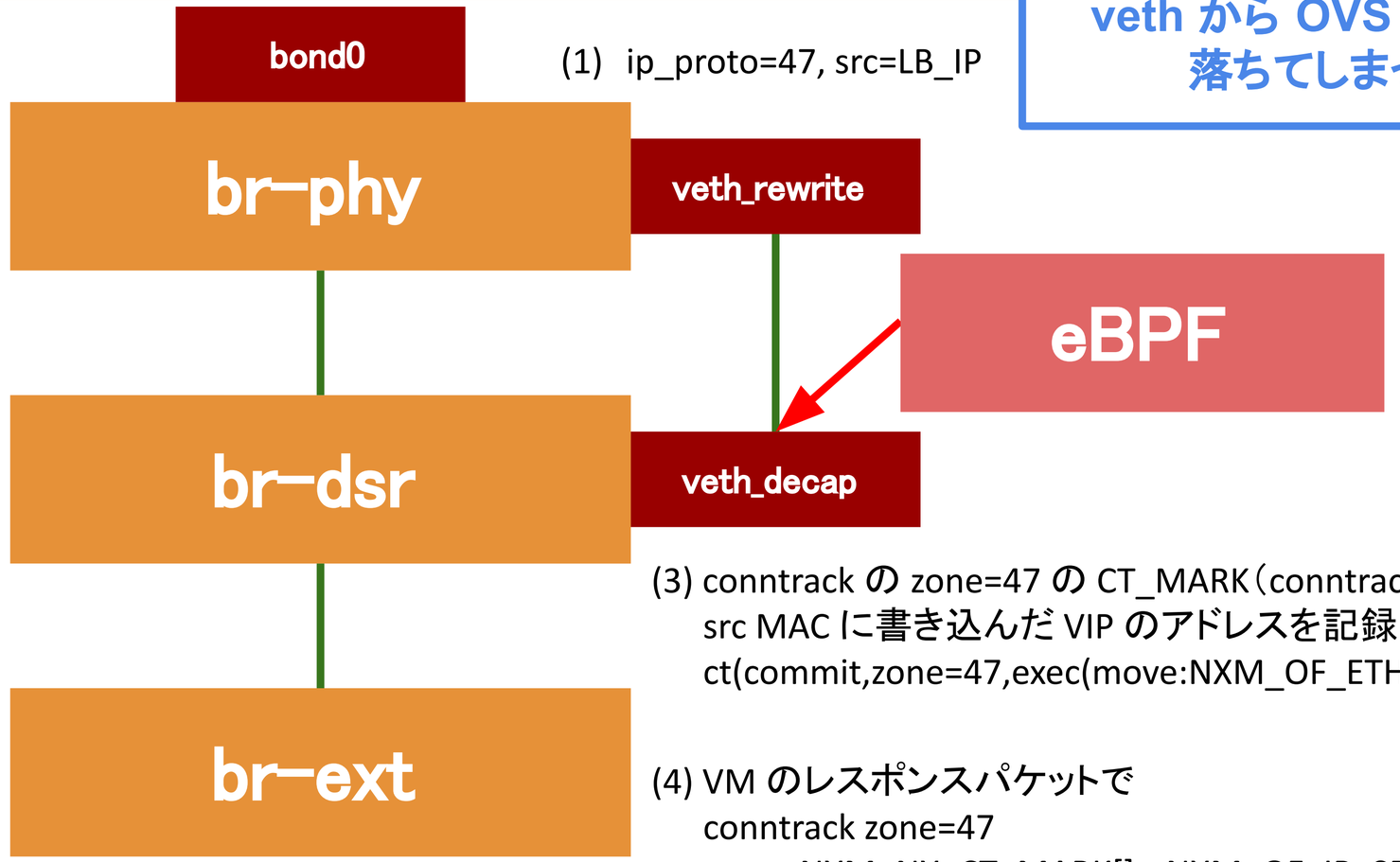
ボツ


OVS
bond
br-phy
br-dsr
gre0
さすがに上手く行きそう …！
(1) GRE を OVS の正攻法で decap
type=gre options:remote_ip=flow
(2) VIP を CT_MARK に格納、SRC: CLI_IP に書き換え
move:NXM_NX_TUN_IPV4_DST[]->NXM_NX_CT_MARK[]
OpenvSwitch の GRE はトンネリング以外の用途では
上手く decap しないしログにも吐かない

その後数十パターン試行錯誤…

決定稿: GRE パターン

本来は pkt_mark 領域に記録したかったが
veth から OVS に入ってくるところでその情報が
落ちてしまっており、利用できなかった



(1) ip_proto=47, src=LB_IP

(2) GRE と Outer IP ヘッダを decap
src MAC の 6 byte のうち 4 byte に
VIP のアドレスを格納

(3) conntrack の zone=47 の CT_MARK（conntrack mark）の 32 bit に
src MAC に書き込んだ VIP のアドレスを記録
ct(commit,zone=47,exec(move:NXM_OF_ETH_SRC[0..31]->NXM_NX_CT_MARK[]))

(4) VM のレスポンスパケットで
conntrack zone=47
move:NXM_NX_CT_MARK[]->NXM_OF_IP_SRC[]
を行い、SRC: VIP_IP, DST: CLI_IP のパケットに書き換える

性能評価

- Apache Bench で評価
  - 1127.08 req/sec
  - 平均　　43 ms
  - 99%ile　157 ms

- OVS
  - w/ dpdk
  - bridge type: netdev

※この測定後、すぐに環境を壊すことになったので
負荷を上げた試験やさらに深い試験は実施できず…
負荷はかなり余裕があったので、10倍以上は食えるように見えた
（PMD thread や CPU usage）

大切な教訓

- OpenvSwitch の GRE 処理はクセが強い
  - 今回は制御しきれず eBPF に逃がした
- OpenvSwitch で複雑なパケット処理を行いたい場合は

  flow rule だけで無理せず eBPF に逃がすことを検討するべき
- 複雑なパケット処理を行う場合、**TCP ヘッダまでであれば**

  eBPF は高速に処理することができる

4

まとめ

CyberAgent

まとめ

- 透過 L3DSR を OpenvSwitch の flow rule + eBPF で実現
  - 設定を同期する agent などは不要
- パフォーマンスも 1Krps 程度までは問題ないことを確認
  - この10倍程度は少なくとも処理できそう
- 今後 L3DSR を使うことがあれば積極的に採用したい

EOP