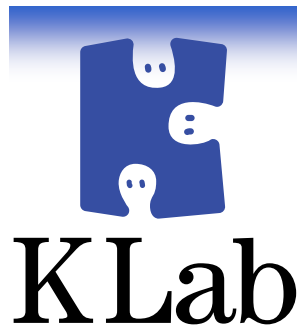
DSASのあれこれ



2007年3月23日

KLab 株式会社

KLab

自己紹介

KLab

時間がもったいないので

KLab

省略しようかな・・・・

というのもあれなので簡単に・・・

KLab

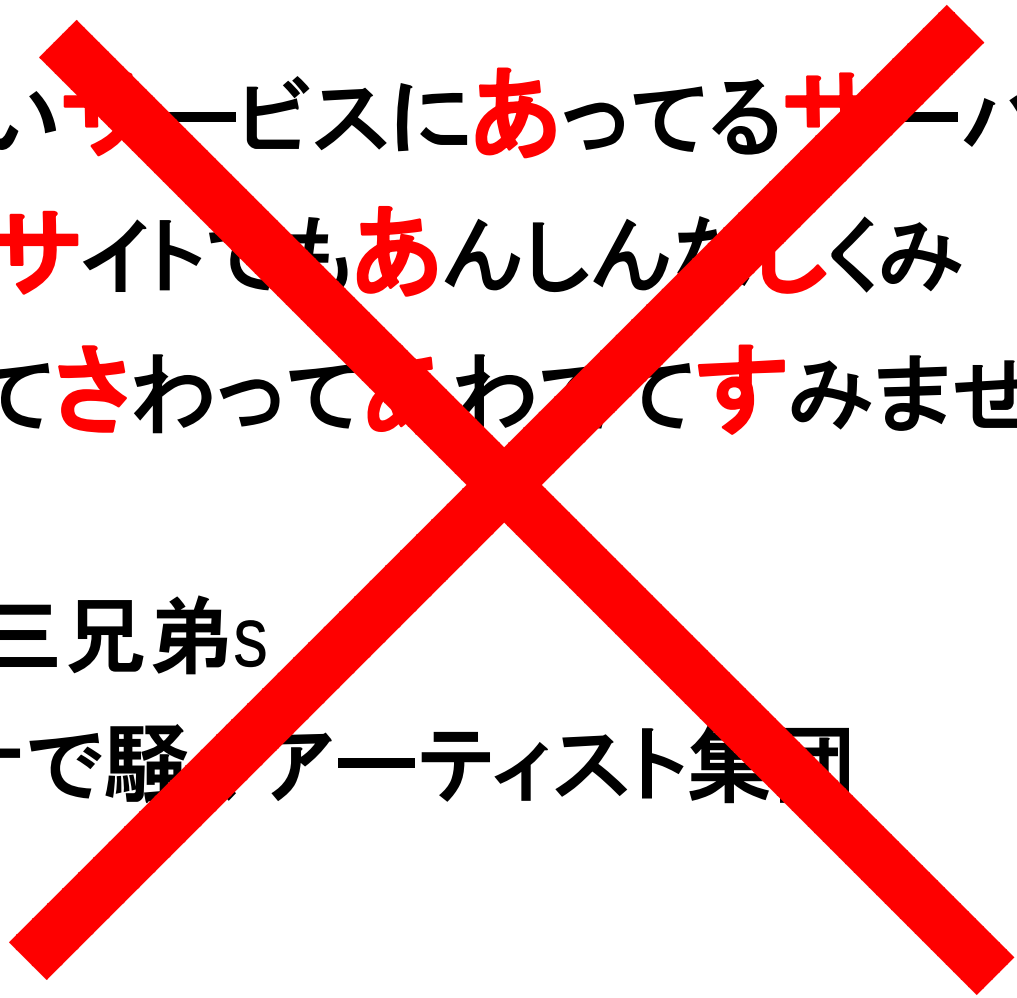
氏名：安井　真伸(やすい まさのぶ)
年齢：　３４歳
出身：　北海道
好きな仕事：　障害対応（？）
嫌いな仕事：　ルーチンワーク（？）

KLab株式会社の「Kラボラトリー」という部門に所属し、いろいろやったりあれこれしたりしています。主にL2,L3まわりのネットワーク構成をごにょごにょするのが好きです。

KLab

早速ですがDSASってなに？

DSAS とは ...



- でっかいサービスにあってるサーバ
- どんなサイトでもあんしんなしくみ
- だまってさわってあわててすみません

- だんご三兄弟s
- ドナドナで騒ぐアーティスト集団

DSAS とは ...

KLab

- Dynamic Server Assign Systemの略
- KLab独自のサーバインフラの呼び名
- いろいろやったりあれこれした成果の集大成
- 特徴
  - オープンソースベースのシステム
  - 単一故障点のないシステム
  - アクセスの増加に柔軟に対応可能
  - 管理者が楽に運用できるような工夫が満載

DSASの設計思想

DSASのコンセプト

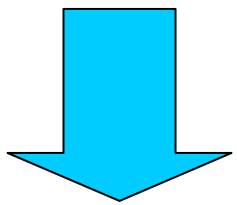
KLab

システム管理者にも十分な睡眠を！

そのためにはどうすればいいか・・・

KLab

- KLabには24時間体制で人間がシステムを監視できる体制も余裕もありません
- ならば、壊れても緊急対応を要しないシステムを組む必要があるでしょう
- そのためには何をすればよいのでしょうか

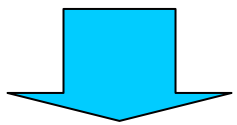


かたっぱしから冗長構成にしてみよう！

冗長構成に対する懸念

KLab

- 冗長構成にすると本当に無停止になるの？
- 中途半端に停止して結局全停したりしない？
- 下手に二重化するとループしたりとかしない？
- 構成が複雑になってトラブルの原因にならん？



冗長化を考えずに運用し続けるのと

これらの問題に立ち向かうのと

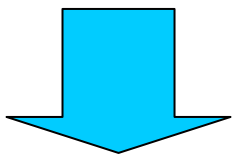
どちらの方が面白そうですか？

何事もチャレンジあるのみ！

KLab

- 問題が発生したら解決すればよいです
- 障害は復旧させればよいです
- 頭をひねって設計するのも楽しいです
- 運用を楽にする仕組みを作るのも楽しいです

考え方ひとつで「懸念」は「楽しみ」に変わります

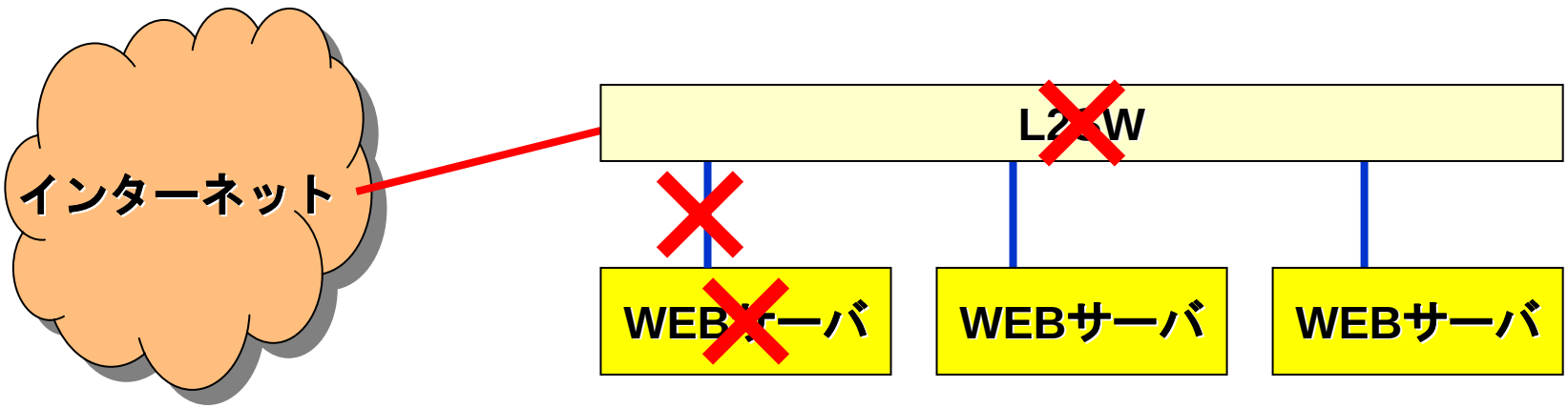


DSASはこのような考えのもとに創られています

KLab技術者の発想

かたっぱしから冗長構成にしてみよう！

- NICを冗長化してみよう！
- L2SWを冗長化してみよう！
- WEBサーバを冗長化してみよう！



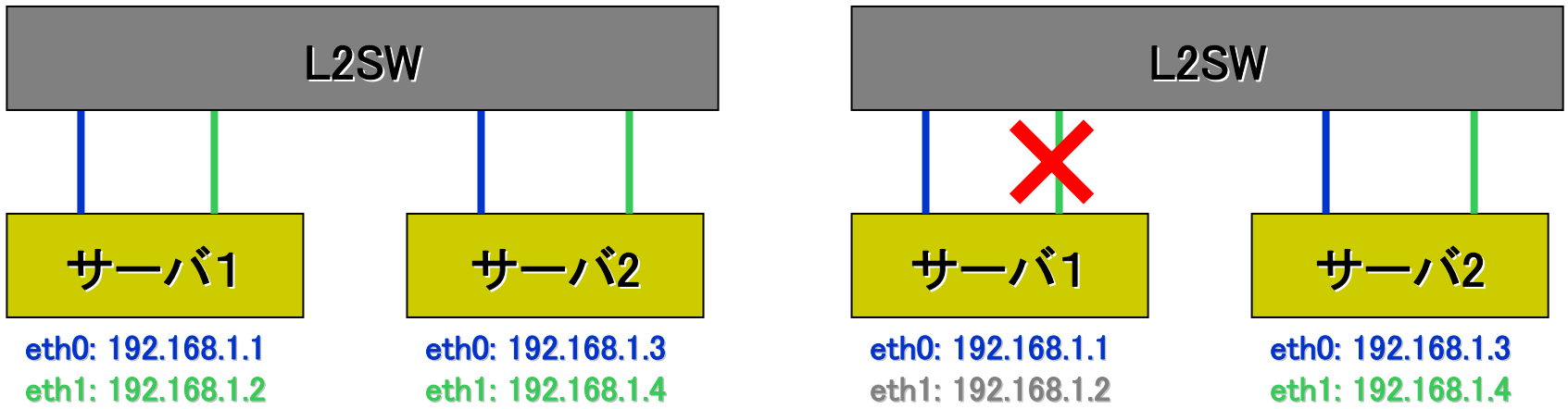
NICを冗長化してみよう

KLab

- 最近のサーバは標準で二つ以上のLANポートを搭載しているものが大半を占めています
- これを有効活用しない手はありません
- LANケーブルが抜けても誰も困らないようにしたいです
  - 片方のケーブルが刺さってさえいれば通信できる
  - どちらのケーブルが抜けても同じアドレスで通信できる
  - こんな動作が理想

NICを冗長化してみよう

各ポートに個別のアドレスを振ったらどうなるでしょう

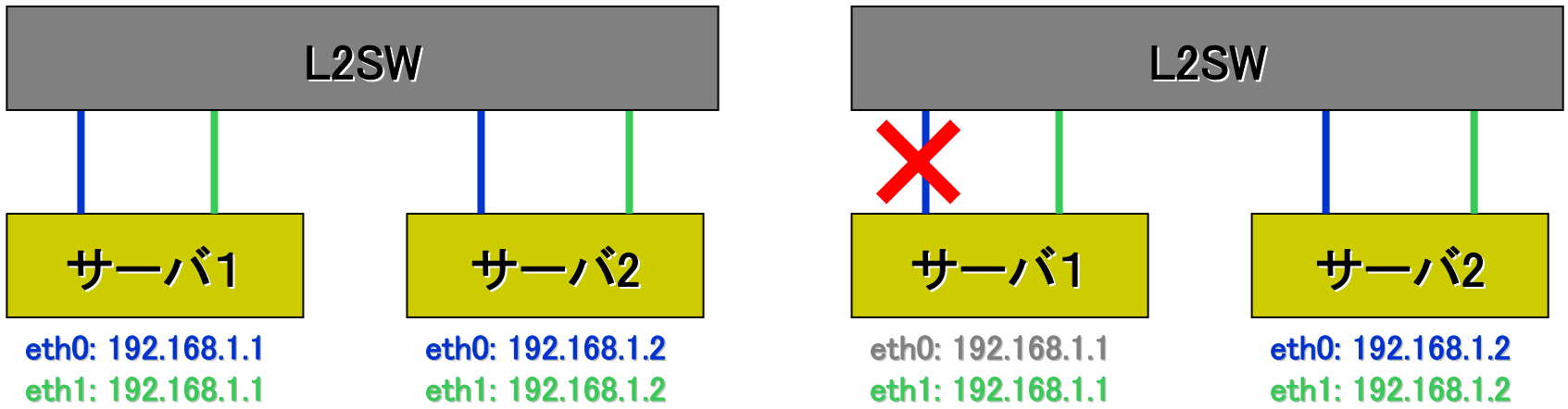


- とりあえずIPアドレスの管理が大変です（単純に数が倍）
- LANケーブルを抜くとそのIPアドレスと通信できなくなります
- これでは冗長構成とはいえません

NICを冗長化してみよう

KLab

各ポートに同じアドレスを振ったらどうなるでしょう

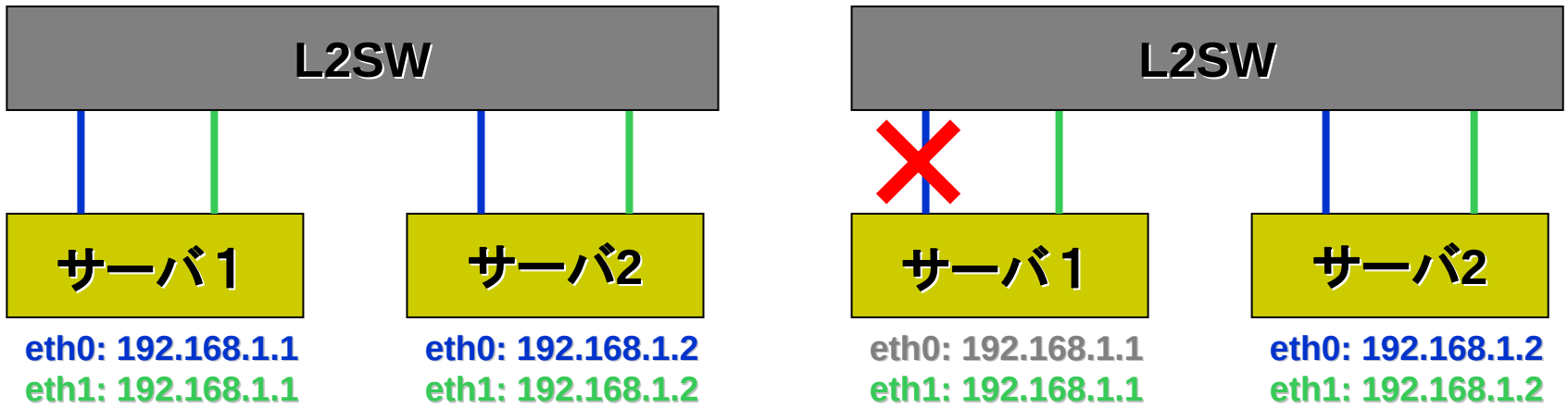


- どちらのポートで通信するかは経路テーブルに依存します
- リンクダウンしても経路テーブルを書き換えてくれません

NICを冗長化してみよう

KLab

各ポートに同じアドレスを振ったらどうなるでしょう



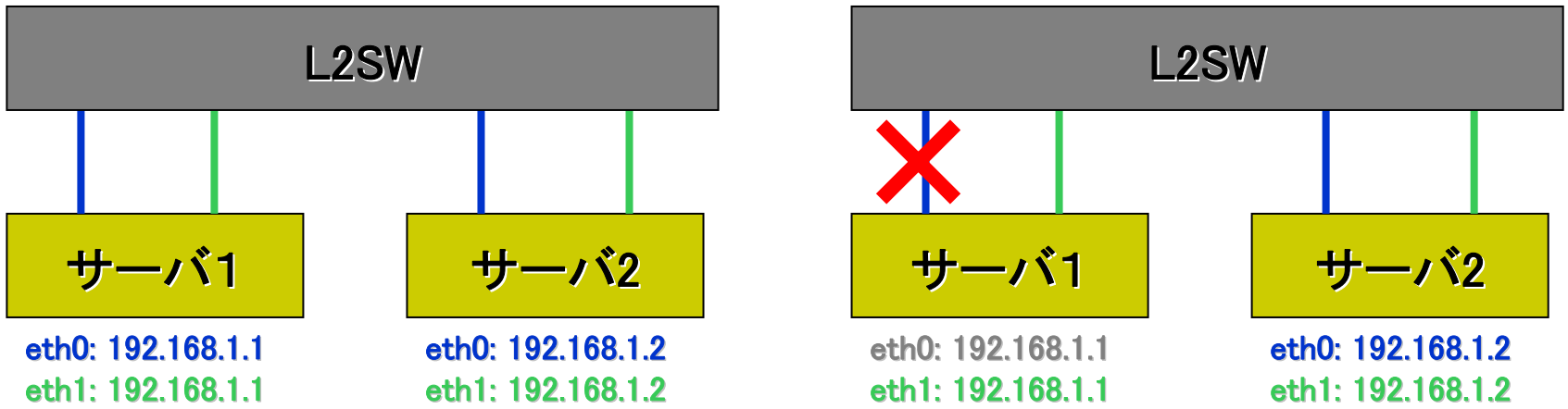

```
サーバ1:# route -n
Kernel IP routing table
Destination  Gateway         Genmask         Flags Metric Ref Use Iface
192.168.1.0  0.0.0.0         255.255.255.0   U     0      0   0   eth0
192.168.1.0  0.0.0.0         255.255.255.0   U     0      0   0   eth1
0.0.0.0      192.168.1.254   0.0.0.0         UG    0      0   0   eth0
```

※ eth0のLANケーブルが抜けたら通信できなくなります

NICを冗長化してみよう

KLab

各ポートに同じアドレスを振ったらどうなるでしょう



- どちらのポートで通信するかは経路テーブルに依存します
- リンクダウンしても経路テーブルを書き換えてくれません

- そもそも arpテーブルの不整合で通信できなくなる事もあります
- これは冗長構成じゃないどころか、全くお話になりません

Bondingモジュールの紹介

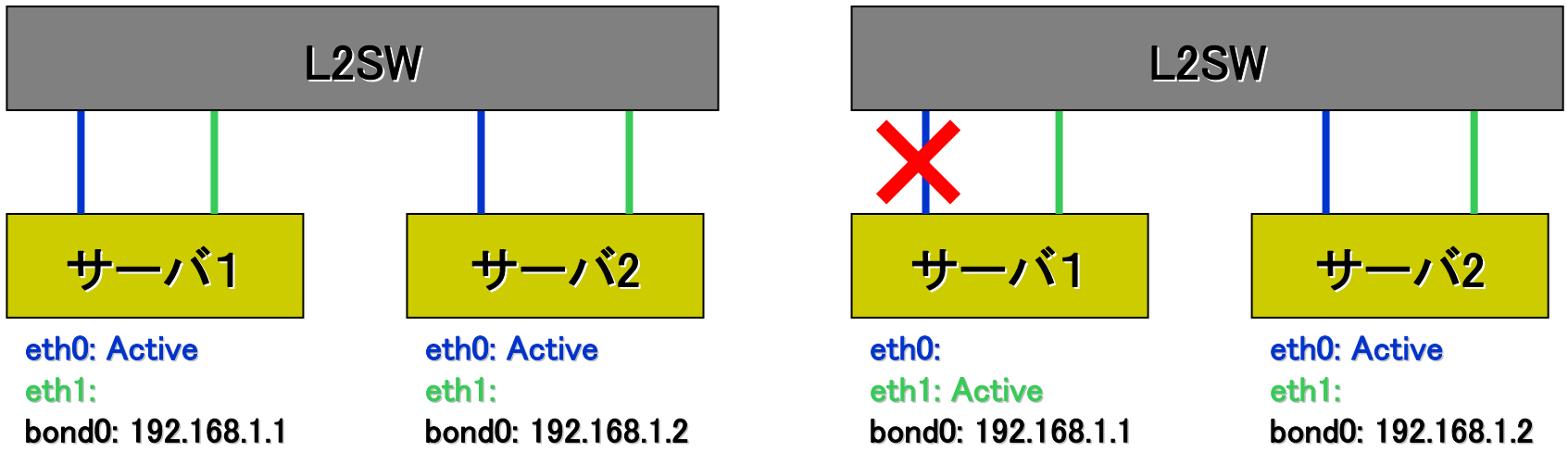
KLab

- Linuxには複数のNICを結合するBondingというモジュールがあります。
- このモジュールは仮想的なネットワークデバイスを生成し、複数の物理デバイス（eth0とかeth1とか）を様々な方式で結合します。
  - balance-rr
  - active-backup　← DSASではこれを使っています
  - balance-xor
  - broadcast
  - 802.3ad
  - balance-tlb
  - balance-alb

  ※これらの詳細はカーネル付属文書の networking/bonding.txt を参照してください
- Bondingの利点は、「実際に利用可能な物理デバイスを使って通信できる」という点です。Bondingモジュールは定期的に物理デバイスのリンクステータスを監視し、リンクダウンを検出すると、そのデバイスの利用を停止します。

NICを冗長化してみよう

Bondingモジュールを使ってみます



- eth0やeth1にはIPアドレスを割り当てません
- bond0という仮想デバイスにIPアドレスを割り当てます
- 実際の送受信はeth0もしくはeth1を利用します
- eth0もしくはeth1のどちらかが繋がってさえいればよいです
- これはよさげです

Bondingの使い方

KLab

- Bondingドライバは Linux カーネルに含まれています
- 物理デバイスの割当には ifenslave コマンドを使います
- 使い方はこんなに簡単です！
  - /etc/modprobe.conf にモジュールオプションを記述

```
alias   bond0 bonding
options bond0 mode=active-backup primary=eth0 miimon=100
```

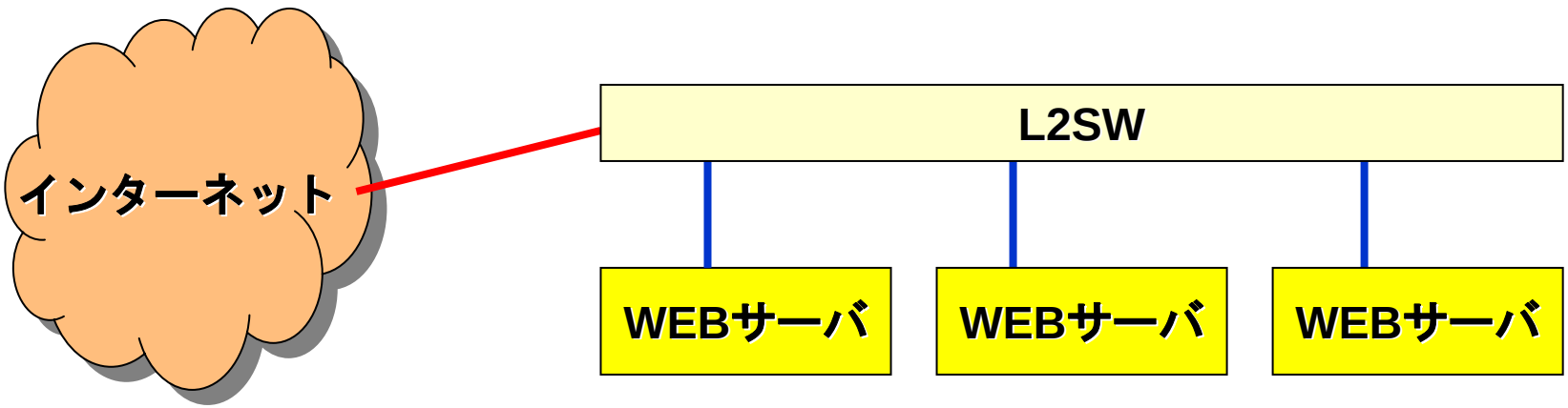
  - # modprobe bond0
  - # ifenslave -f bond0 eth0 eth1
  - ifconfig などで bond0 に IPアドレスを設定する
  - これで bond0 が使えるようになります

KLab技術者の発想

KLab

かたっぱしから冗長構成にしてみよう！

- NICを冗長化してみよう！
- L2SWを冗長化してみよう！
- WEBサーバを冗長化してみよう！



L2SWを冗長化してみよう

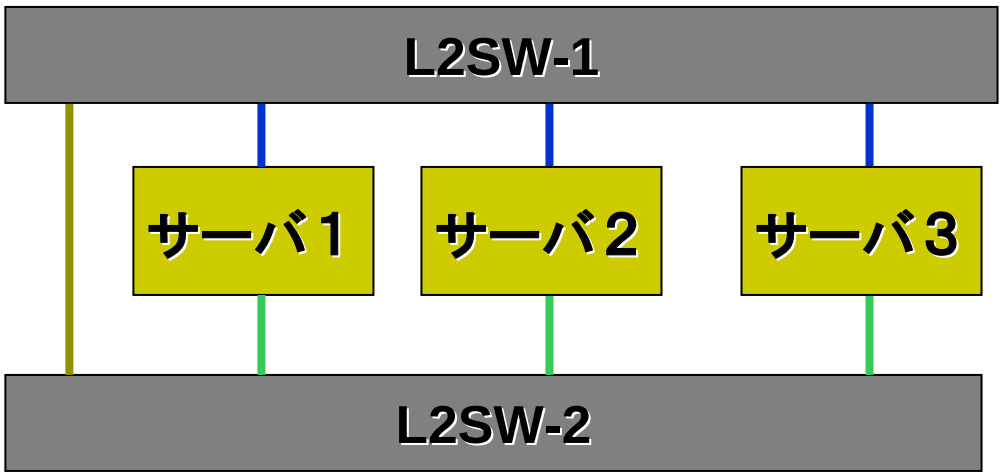
KLab

- かたちあるものは必ず壊れます
- ファームの更新などで再起動したい時だってあります
- そんな時でもできるだけサービスを停止したくはありません
- NICの冗長化ができたことだしL2SWも冗長構成にしてみましょう

L2SWを冗長化してみよう

KLab

- この構成の問題点
  - サーバ2の青色の線を切ると他のサーバと通信できなくなります
  - L2SW-1とL2SW-2を接続すればこの問題は回避できます
- サーバの台数が増えてポートが足りなくなったらどうしましょう
- L2SWを増設することになりますが、その接続はどのようにしましょう

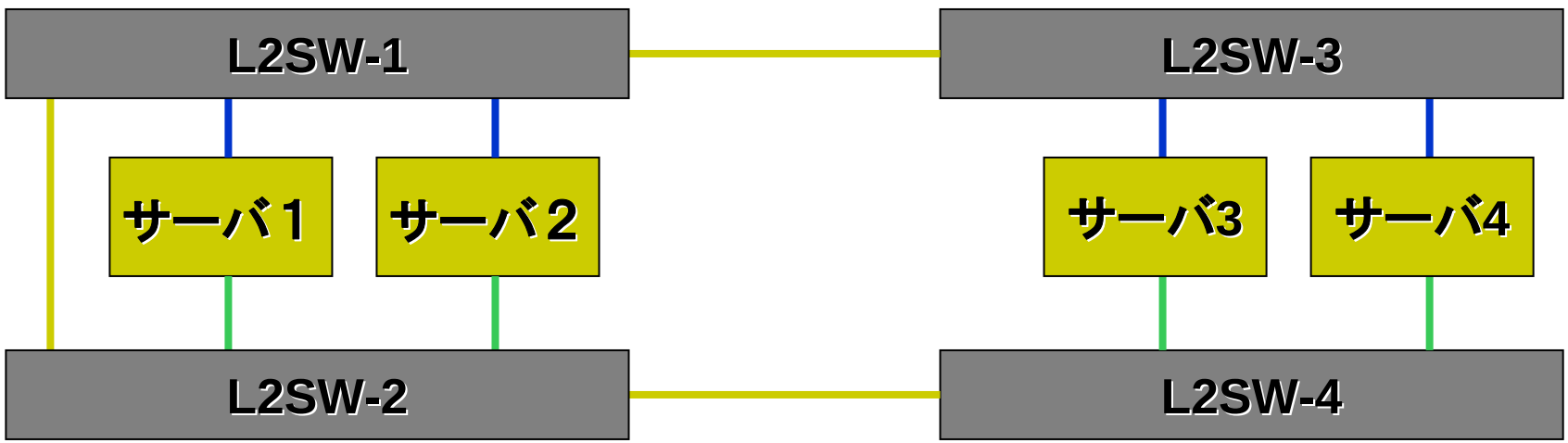


青色はbondingのActive
緑色はbondingのBackup

L2SWを冗長化してみよう

KLab

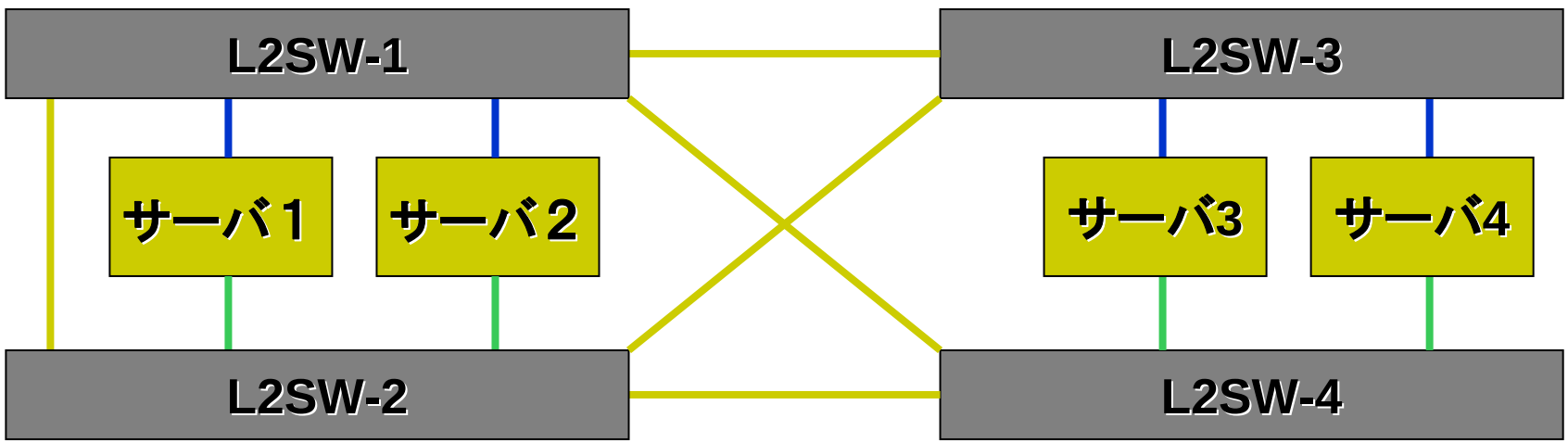
- この構成の問題点
  - L2SW-1 がダウンするとサーバ1、2とサーバ3、4が通信できなくなります
  - そこで、L2SW-2とL2SW-3を接続し、L2SW-1とL2SW-4も接続してみます



L2SWを冗長化してみよう

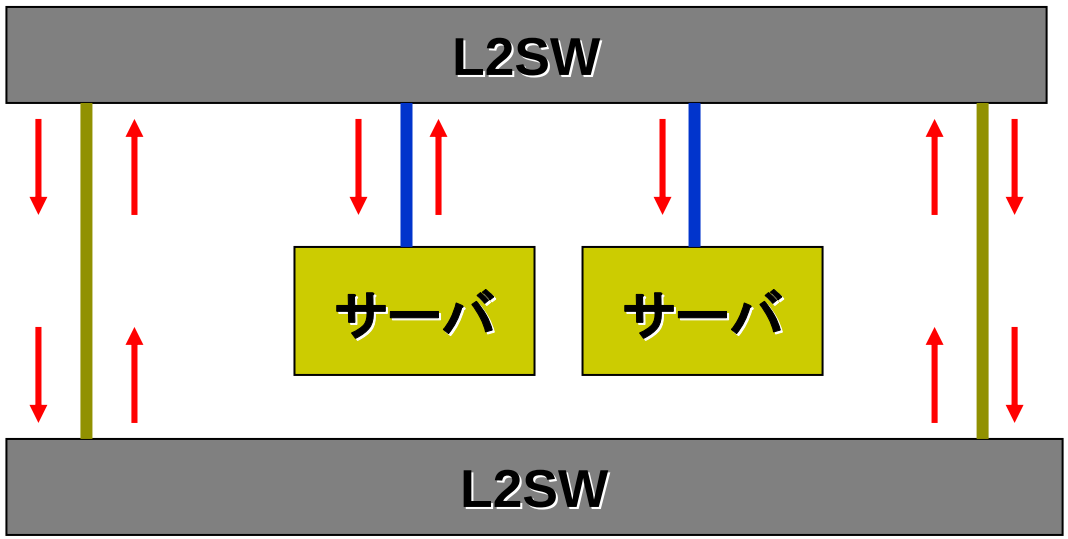
KLab

- この構成の問題点
  - 線を繋いだ瞬間にブロードキャストストームが発生します
  - 全く通信できなくなります



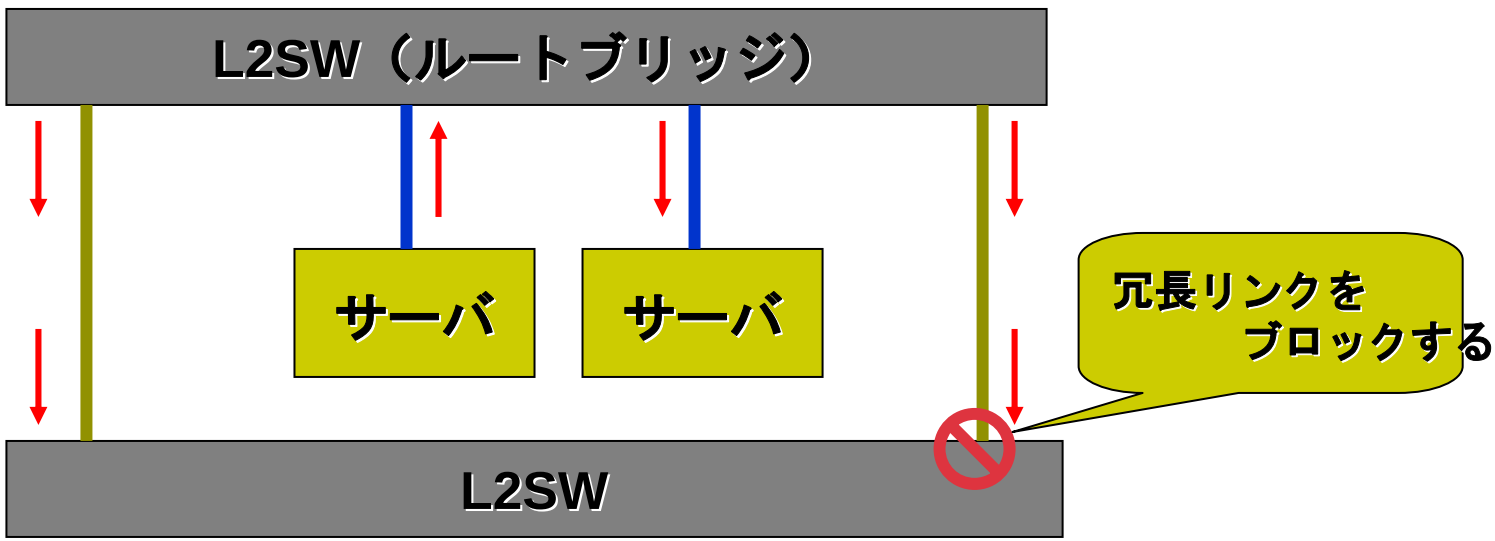
ブロードキャストストームとは

- L2SW(ブリッジ)で冗長リンク(ループ)を形成し、ブロードキャストを送出すると、同じ経路を無限に転送され続けることになります
- こうなると、一瞬でブロードキャストが帯域を使い果たしてしまい、通常の通信が全くできなくなってしまいます
- この状態を「ブロードキャストストーム」と呼びます



RSTP(IEEE802.1w)の紹介

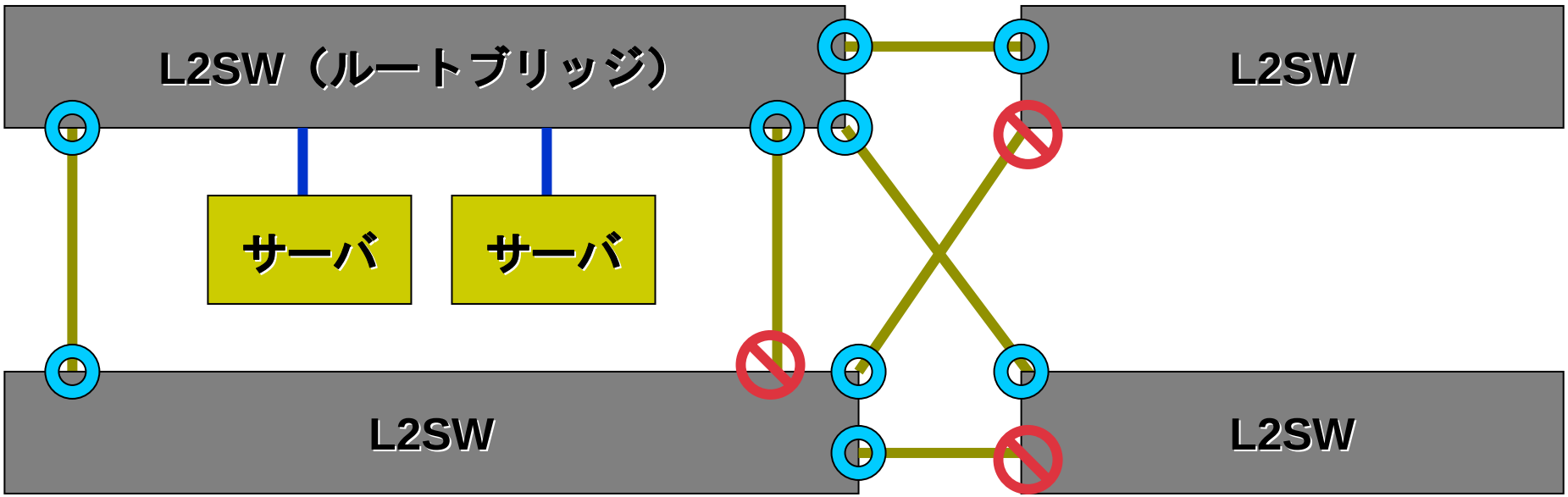
- RSTPはネットワークのループを検知して、ブロードキャストストームが発生しないように特定のポートをブロック(通信できないように)する仕組みです
- 機器がダウンしたりケーブルが外れたりしてネットワークトポロジに変更があると、ブロックを解除したり別なポートをブロックしたりすることで、常に正常な通信ができるように調整してくれます
- RSTPの必須設定項目は「プライオリティ」だけです(超簡単！)
- プライオリティの一番小さい機器が「ルートブリッジ」となります



RSTP(IEEE802.1w)の紹介

KLab

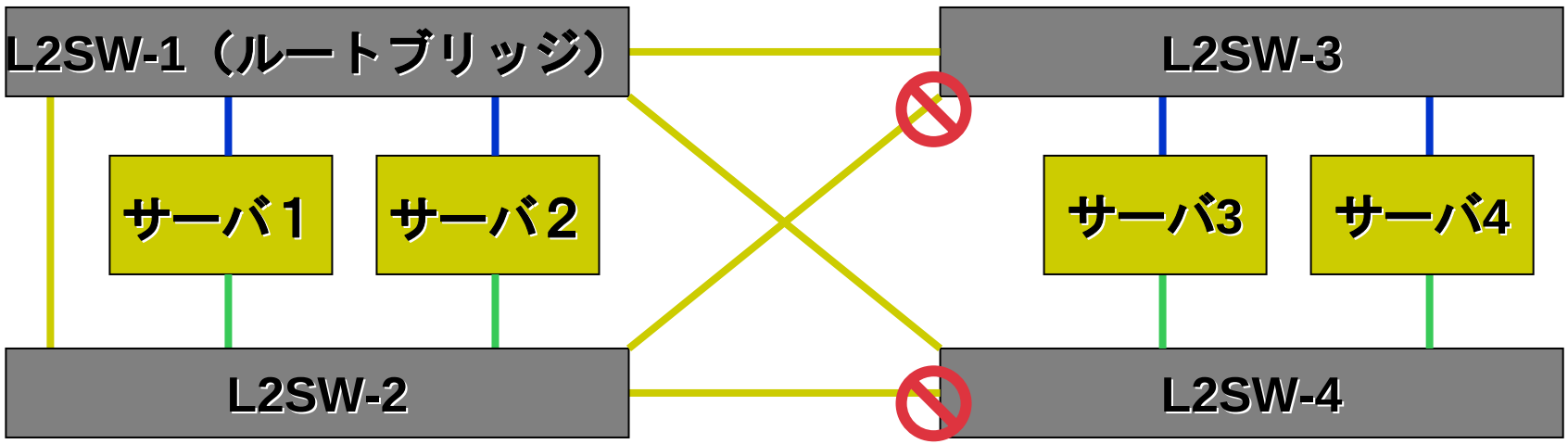
- RSTPの動作
  - ルートブリッジのポートはブロックしません
  - ルートブリッジと直接接続されているポートをフォワーディングにします
  - 同じ機器に対して複数の線が接続されている場合は、片方をブロッキングにします
  - ルートブリッジと直接接続されていないセグメントでは
  - ルートブリッジに近いポートをフォワーディングにします
  - 残ったポートをブロッキングにします



L2SWを冗長化してみよう

KLab

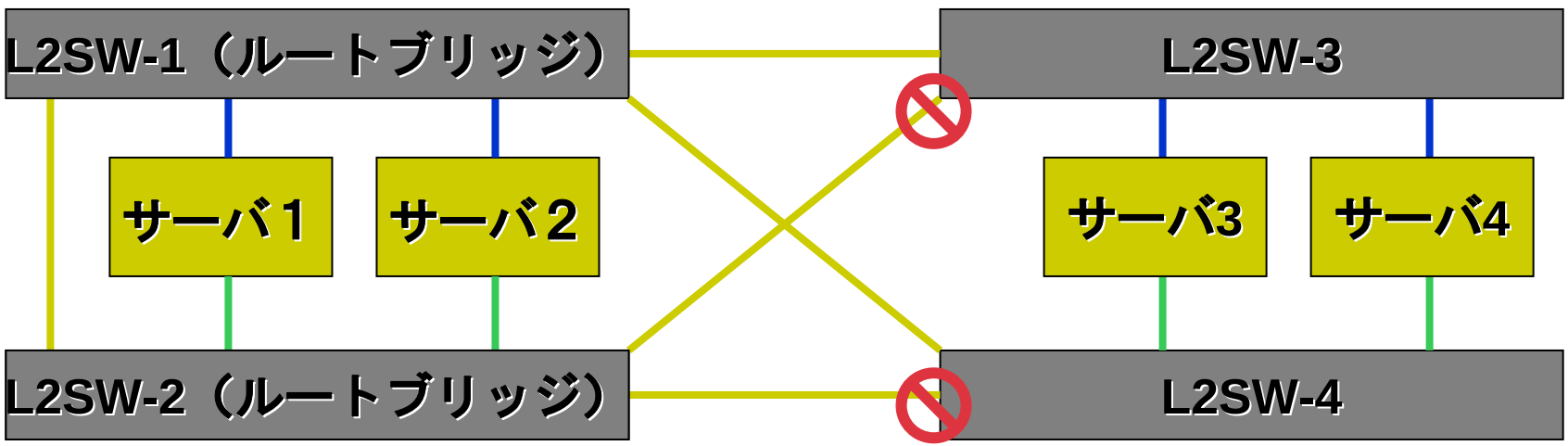
- 全L2SWのRSTPを有効にします
  - ルートブリッジはL2SW-1とします
  - 下図のポートがブロッキングされ、ブロードキャストストームは発生しません
- どこかの線が切れるとどうなるか
  - 瞬時にブロッキングポートがフォワーディング状態になります
- どこかのL2SWがダウンするとどうなるか
  - 瞬時にブロッキングポートがフォワーディング状態になります



L2SWを冗長化してみよう

KLab

- 全L2SWのRSTPを有効にします
  - ルートブリッジはL2SW-1とします
  - 下図のポートがブロッキングされ、ブロードキャストストームは発生しません
- どこかの線が切れるとどうなるか
  - 瞬時にブロッキングポートがフォワーディング状態になります
- どこかのL2SWがダウンするとどうなるか
  - 瞬時にブロッキングポートがフォワーディング状態になります

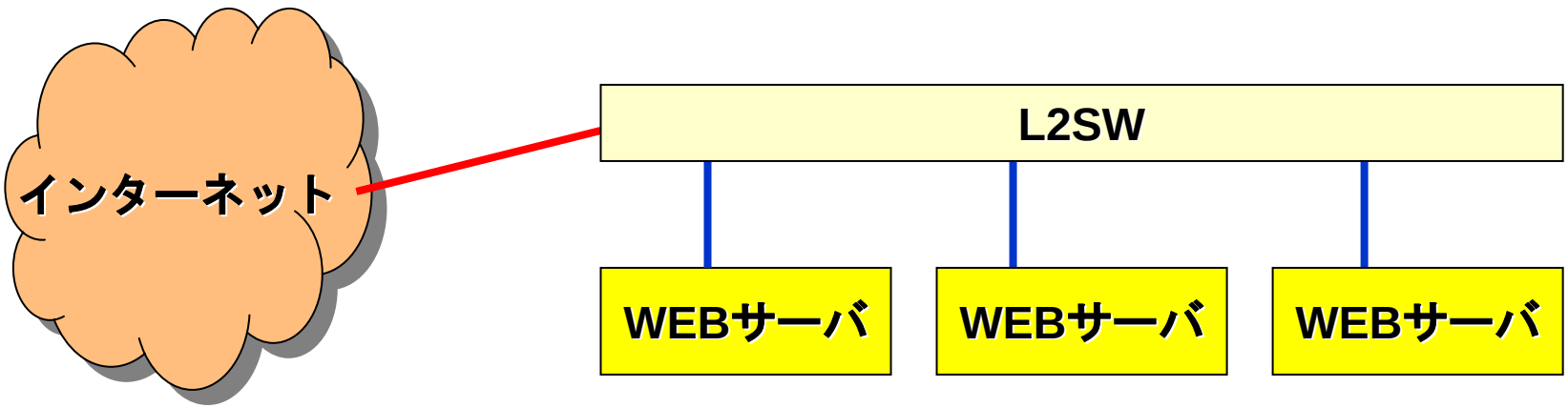


KLab技術者の発想

KLab

かたっぱしから冗長構成にしてみよう！

- NICを冗長化してみよう！
- L2SWを冗長化してみよう！
- WEBサーバを冗長化してみよう！

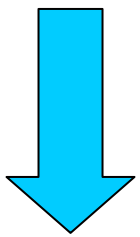


WEBサーバを冗長化してみよう

KLab

- WEBサーバの冗長化はロードバランサを導入するのが楽です
- ロードバランサを導入すると
  - 何台ものWEBサーバに負荷分散することができます
  - 停止したWEBサーバは自動的に負荷分散から外せます

- でも買うと結構高いっす・・・
- 検証用と称して買うのは厳しいです
- 開発環境への導入なんて夢ですかねえ



LVSでLinuxをロードバランサにできます！

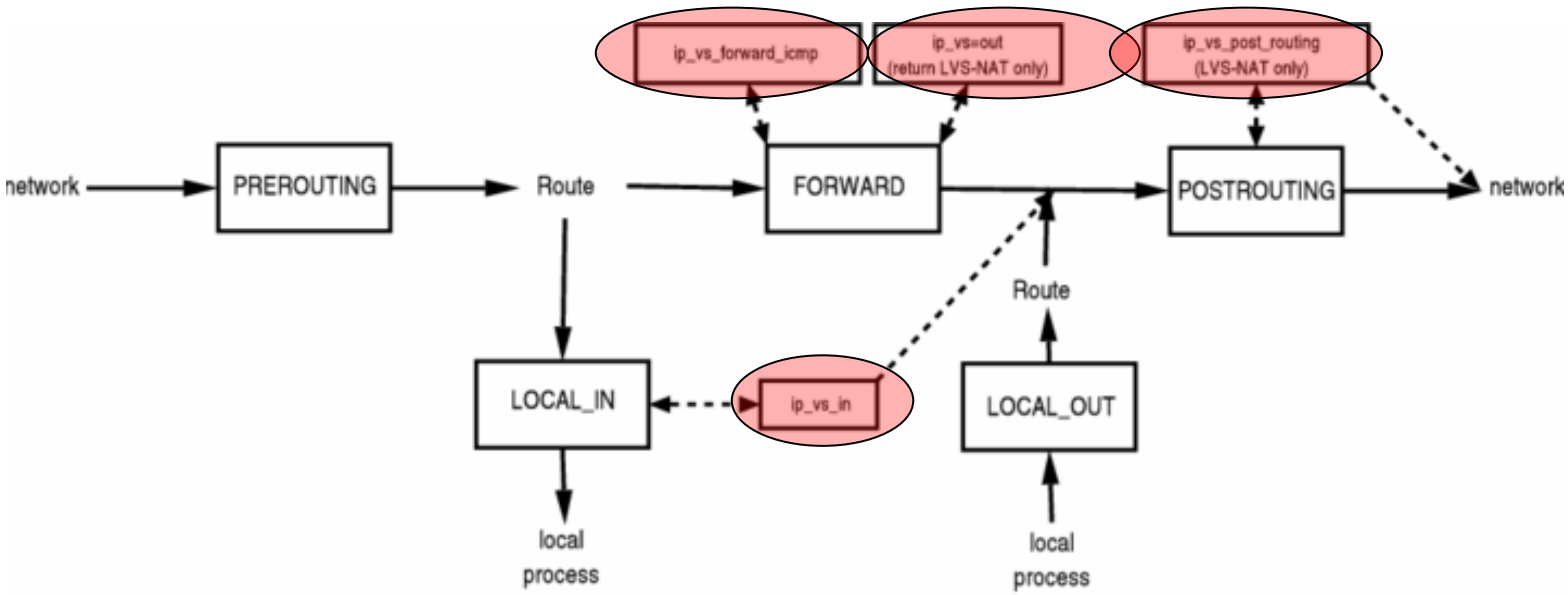
LVSとは

KLab

- LVS – Linux Virtual Server Project
  - Linuxで、高性能かつ高可用性をもったサーバシステムを作ろうというプロジェクト。
- IPVS – IP Virtual Server
  - LVSプロジェクトの成果物のひとつ
  - LinuxをL4ロードバランサにできる

IPVSとは

- Netfilterと連携して動く
- kernel module
  - カーネル空間で動くので速い



Linux Kernel Netfilter Hooks and LVS

Horms <horms@verge.net.au>, v0.1.9-1, October 2003

IPVSの設定

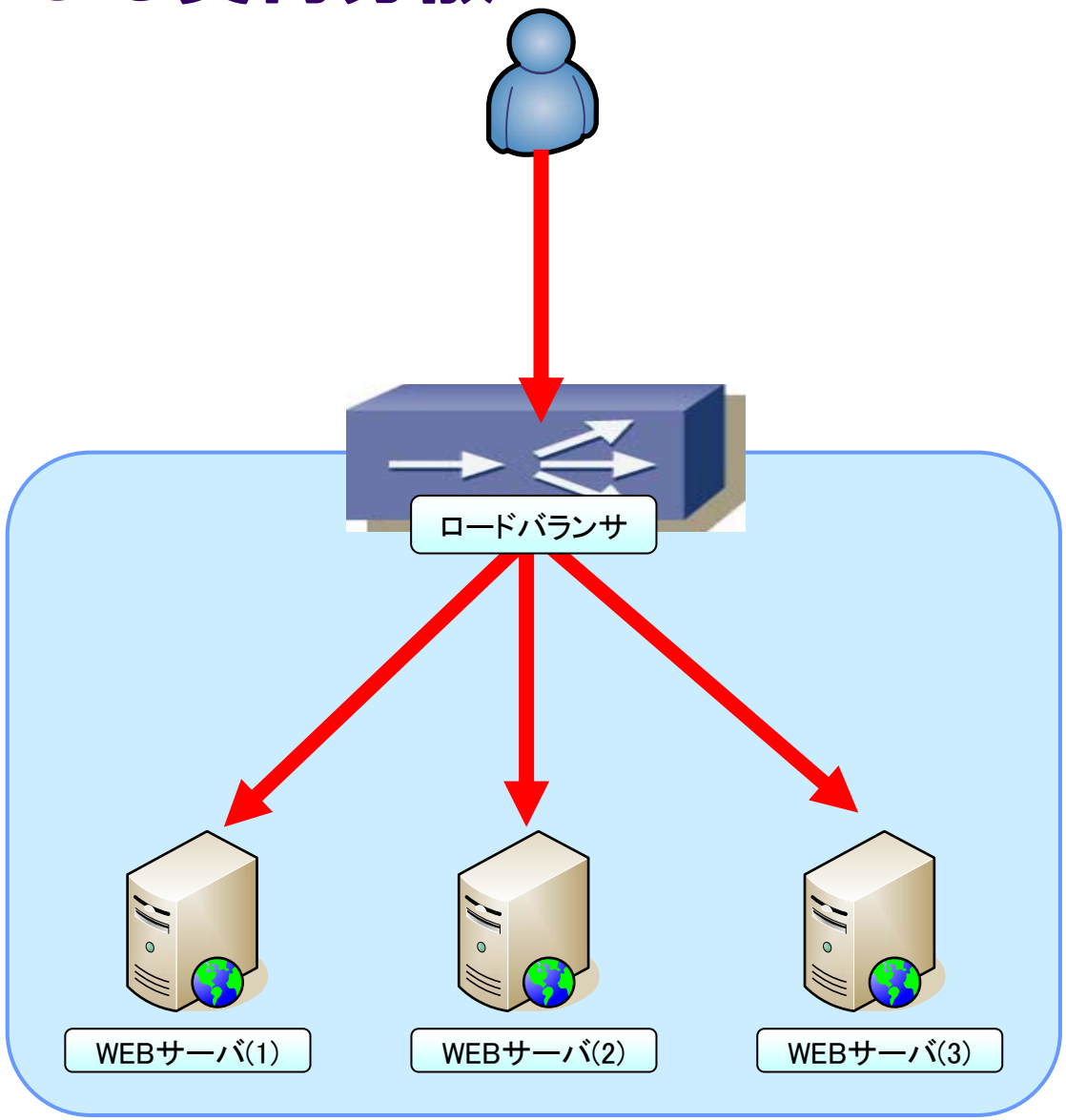
KLab

- IPVSはipvsadmコマンドで制御できます
- 例えばwww.example.org:80宛てを ← VIP
  - 192.168.31.101
  - 192.168.31.102 ⇔ WEBサーバ
  - 192.168.31.103
- に分散するには

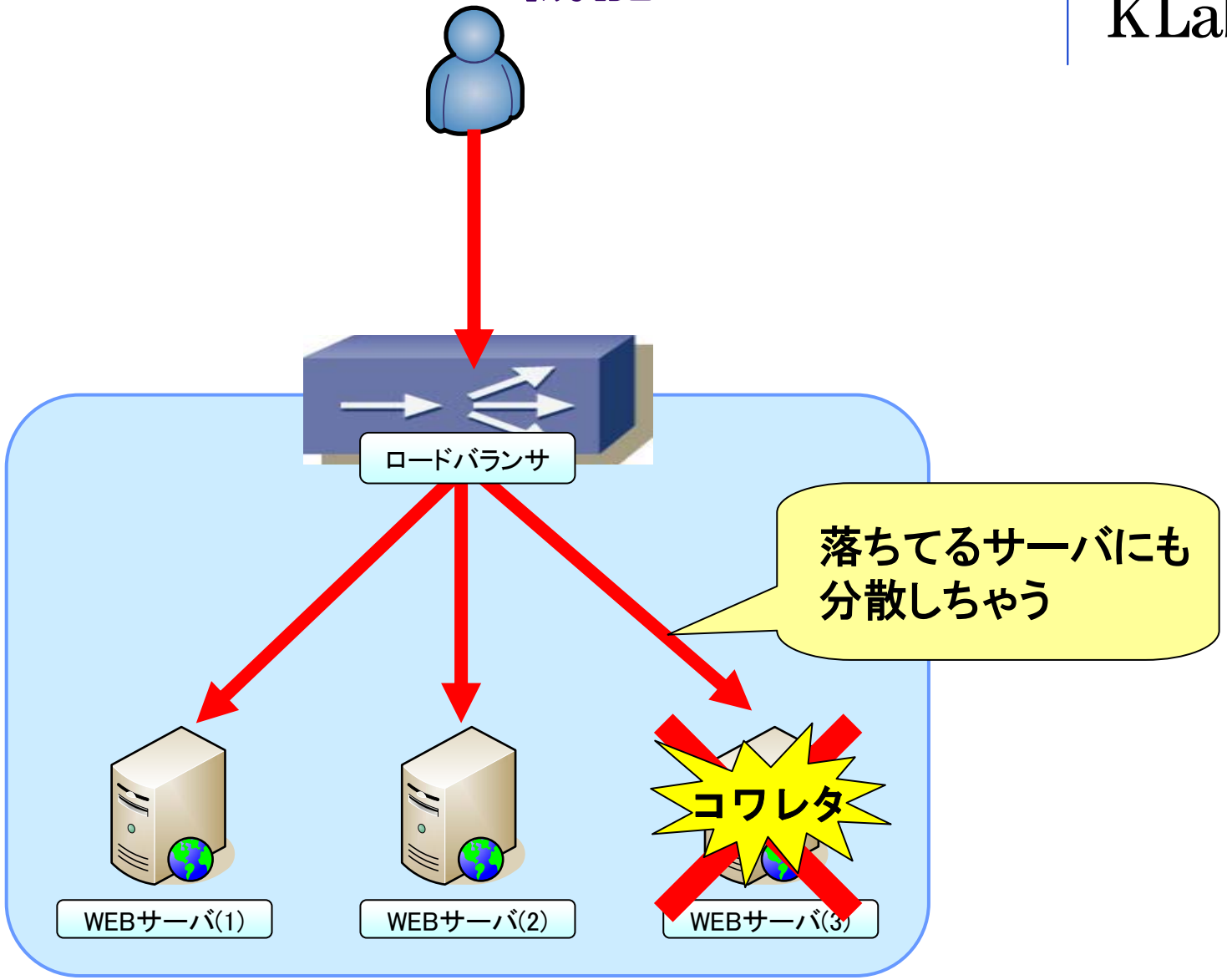
```
# ipvsadm -A -t www.example.org:80 -s lc
# ipvsadm -a -t www.example.org:80 -r 192.168.31.101 –m
# ipvsadm -a -t www.example.org:80 -r 192.168.31.102 -m
# ipvsadm -a -t www.example.org:80 -r 192.168.31.103 -m
```

- これだけでできちゃいます

IPVSによる負荷分散



IPVSにはヘルスチェック機能がない

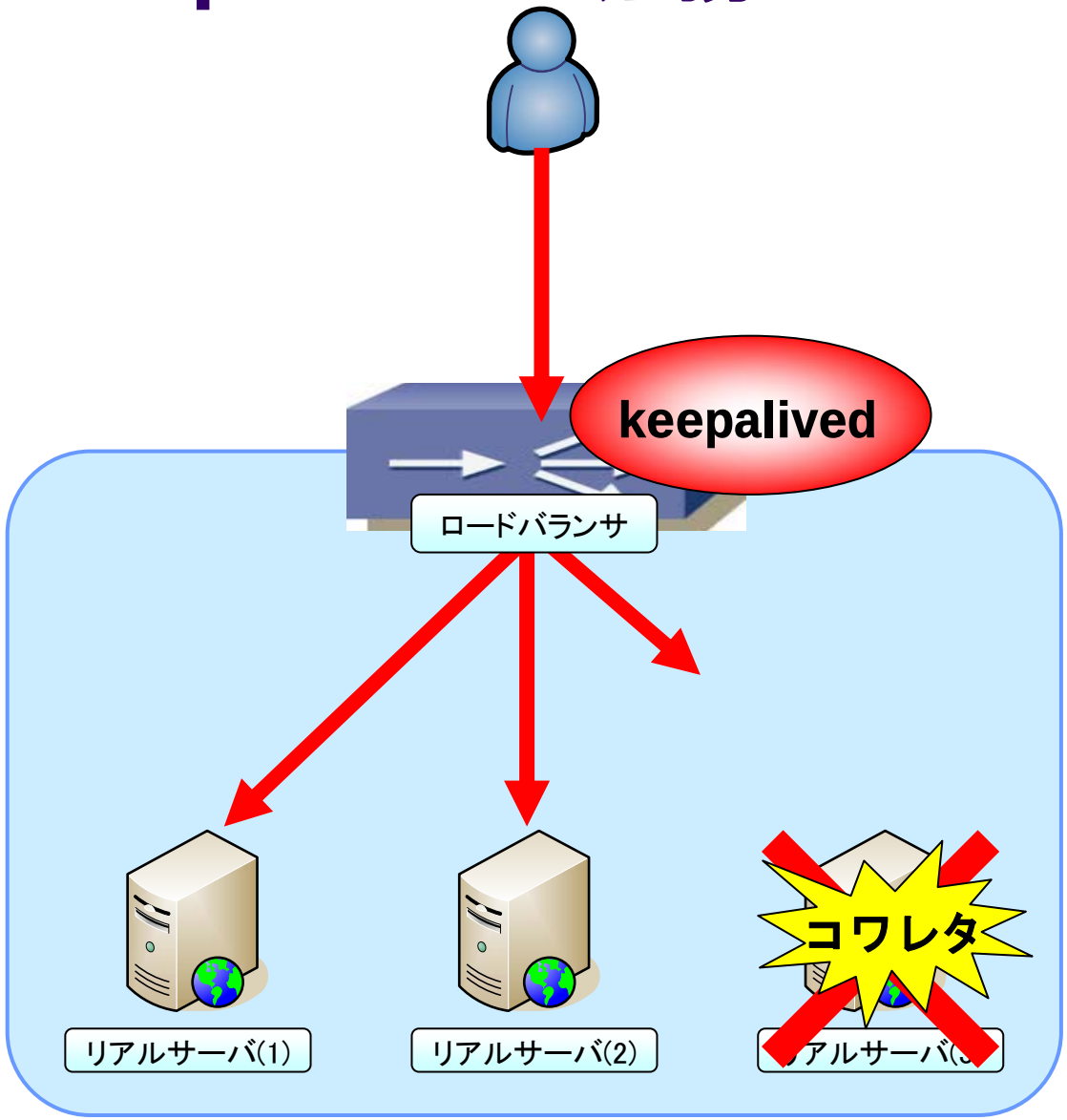


keepalivedの機能（CHECK）

KLab

- IPVSと連携できるヘルスチェック機能
  - 定期的にリアルサーバにヘルスチェックします
    - HTTP_GET　(HTTPでアクセスしてみる)
    - SSL_GET　(HTTPSでアクセスしてみる)
    - TCP_CHECK　(TCPセッションを張ってみる)
    - SMTP_CHECK　(HELO って呼んでみる)
    - MISC_CHECK　(任意のコマンドを実行する)
  - 期待した応答が得られなければ
  - IPVS的に分散から外すことができる

IPVSとkeepalivedの連携

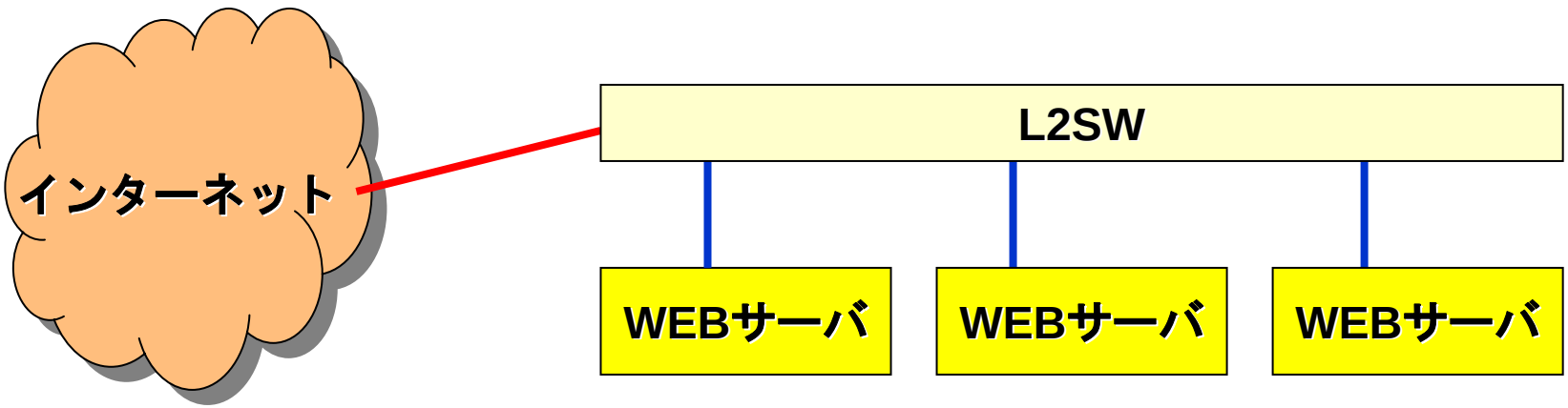


KLab技術者の発想

KLab

かたっぱしから冗長構成にしてみよう！

- NICを冗長化してみよう！
- L2SWを冗長化してみよう！
- WEBサーバを冗長化してみよう！



そういえば、ロードバランサも
冗長化しないといけませんね

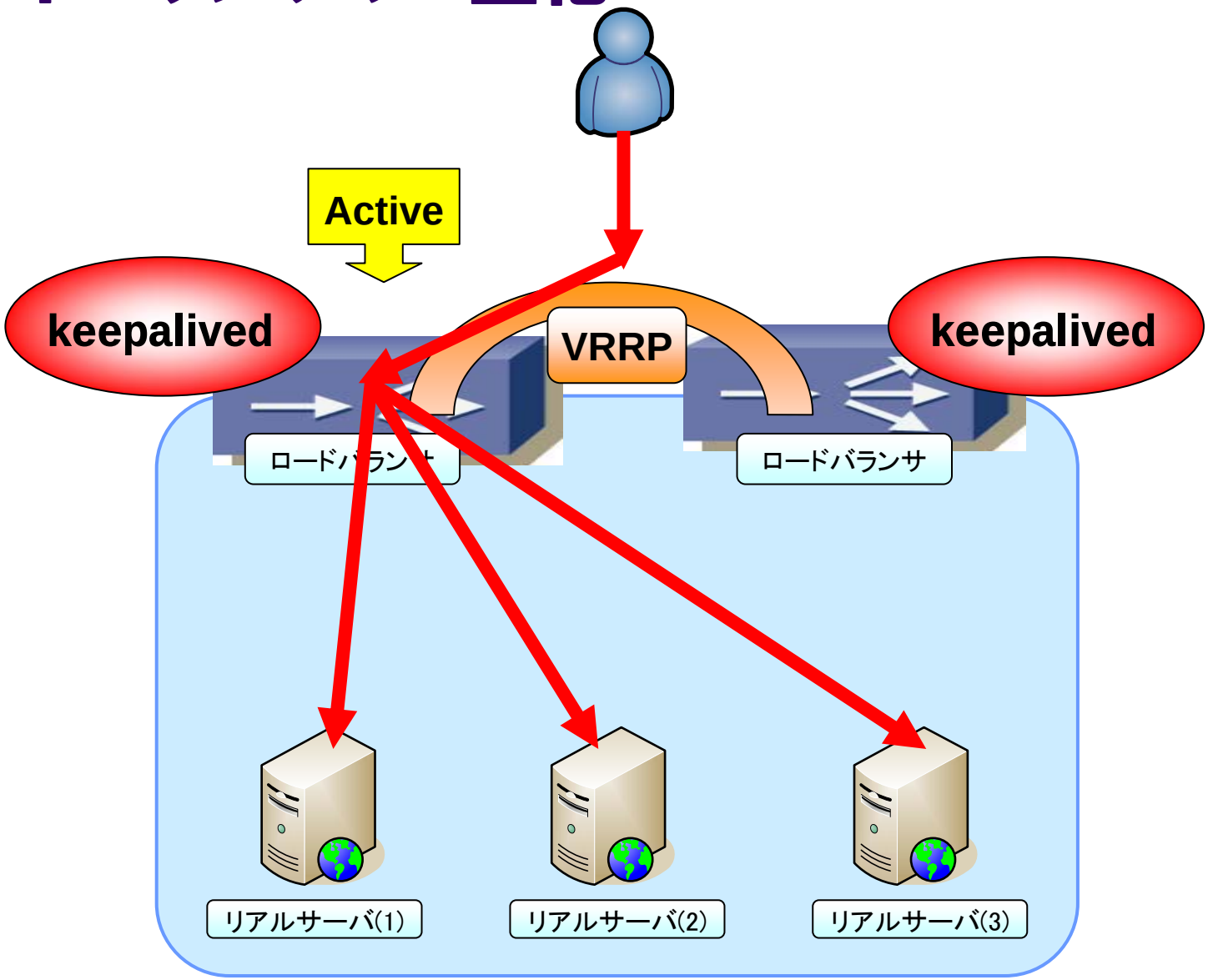
keepalivedの機能（VRRP）

KLab

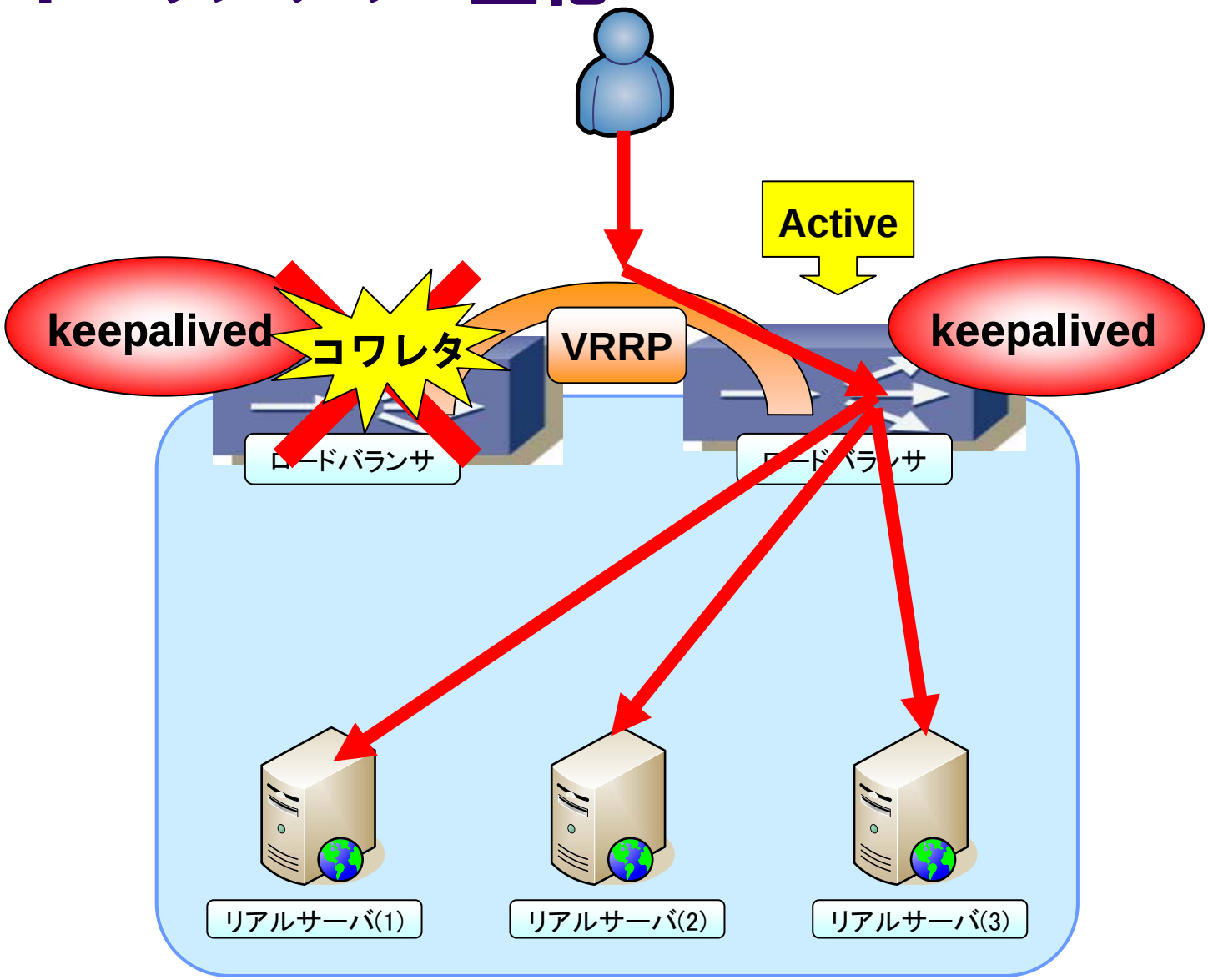
- VRRP - Virtual Router Redundancy Protocol
  - ルータ(ロードバランサ)の
    - 二重化
    - フェイルオーバ
  - を実現するためのものです
  - マルチキャストで死活確認します

- もう1台ロードバランサをいれるだけで冗長化できそうです

ロードバランサの二重化



ロードバランサの二重化



KLab

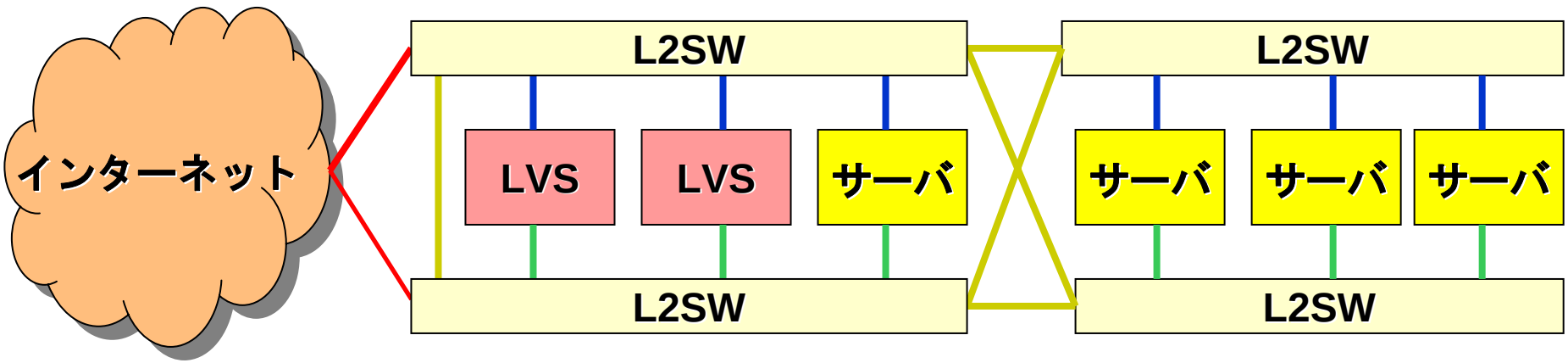
これで単一故障点のない
システムになりました(＾＾)/

ここまでの話をまとめると

こんな構成になりました！

KLab

- インターネット回線は2本契約(Active/Backup)
- ロードバランサは冗長構成(VRRP)
- WEBサーバも冗長構成(IPVS)
- L2SWも冗長構成（RSTP）
- NICも冗長構成(bonding)



KLab

めでたしめでたし

KLab

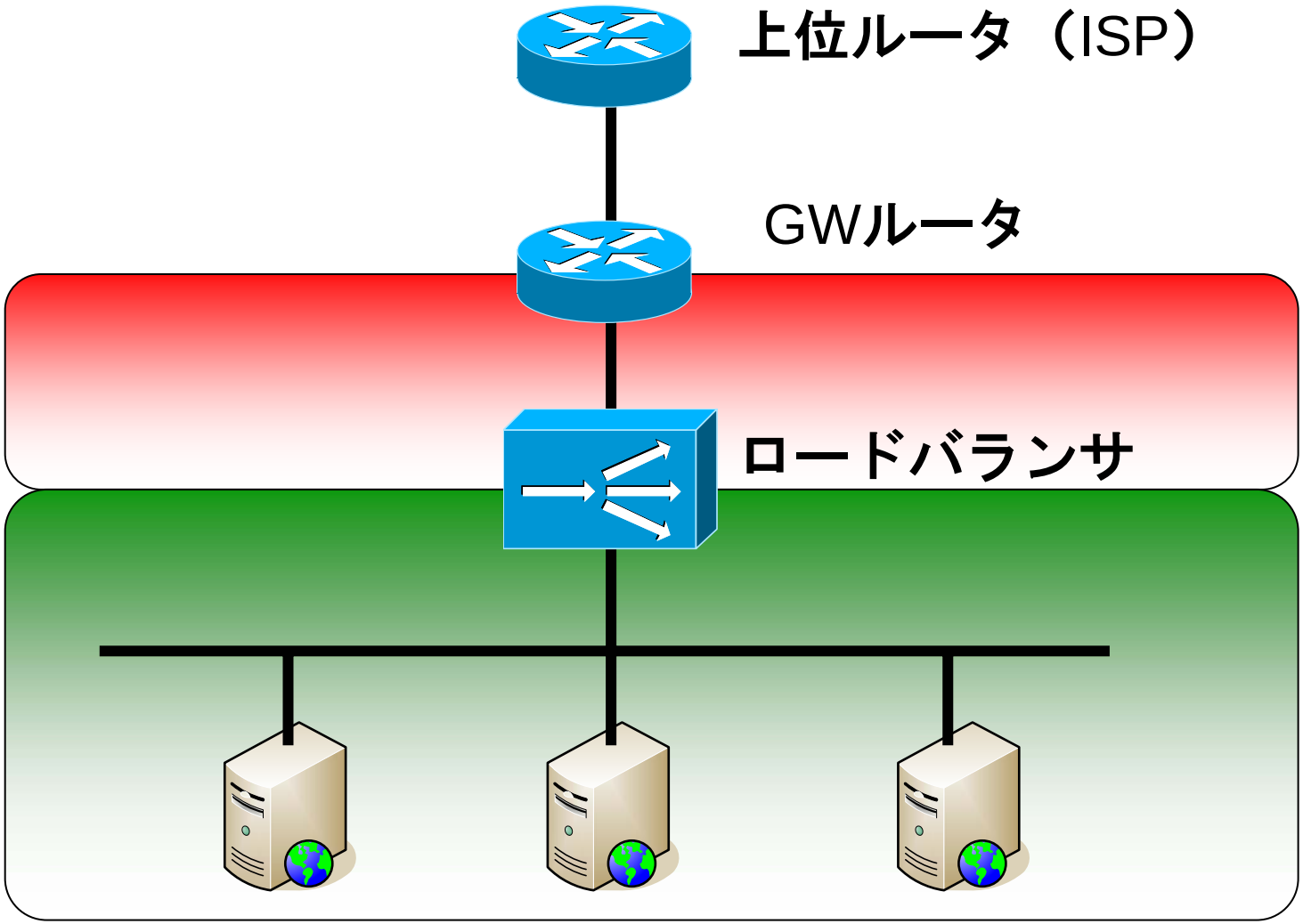
ところで・・・

ロードバランサを導入する場合

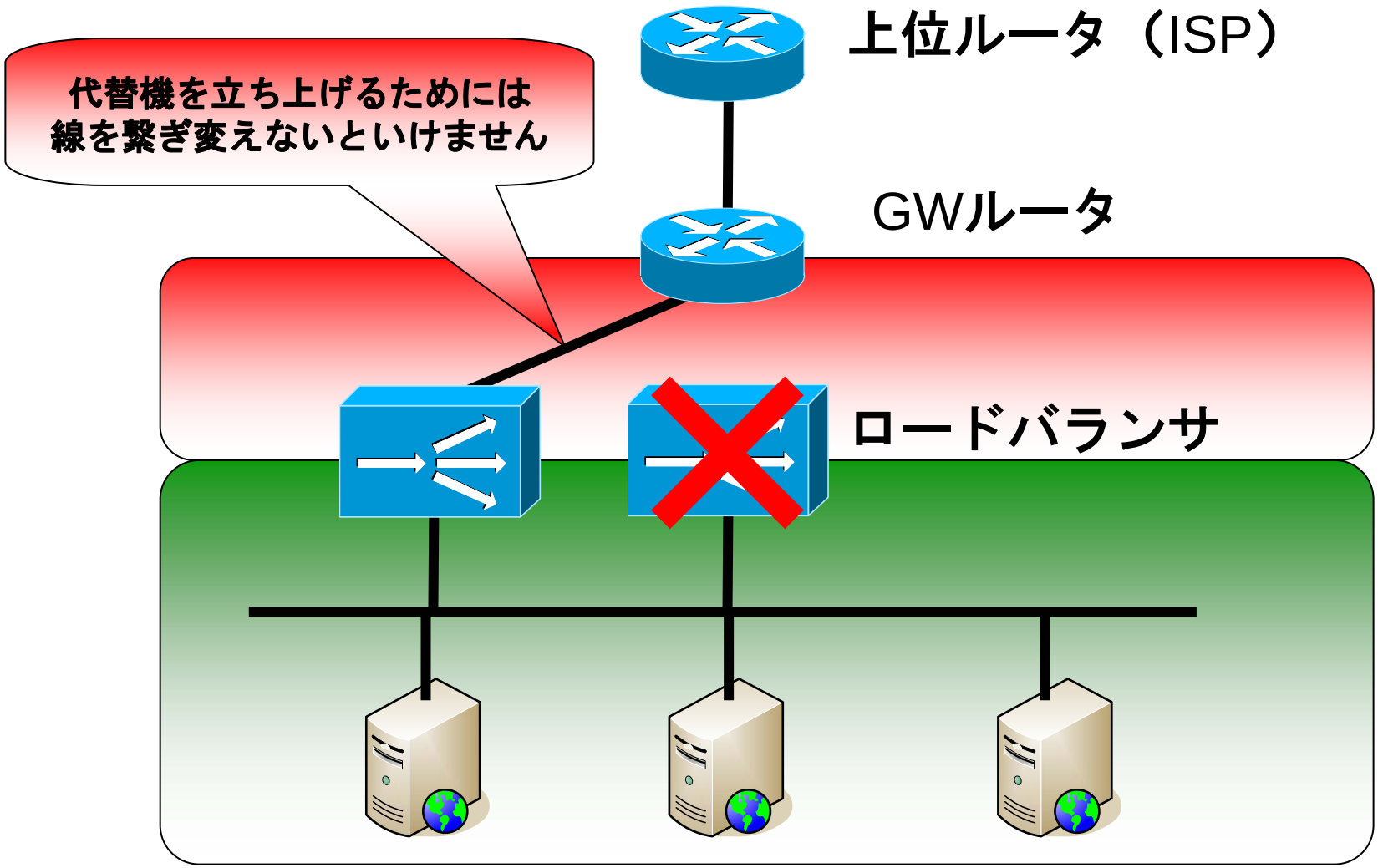
どんな構成にしますか？

ネットワーク境界にロードバランサを設置

KLab



故障時の対応が若干面倒



なにが問題か

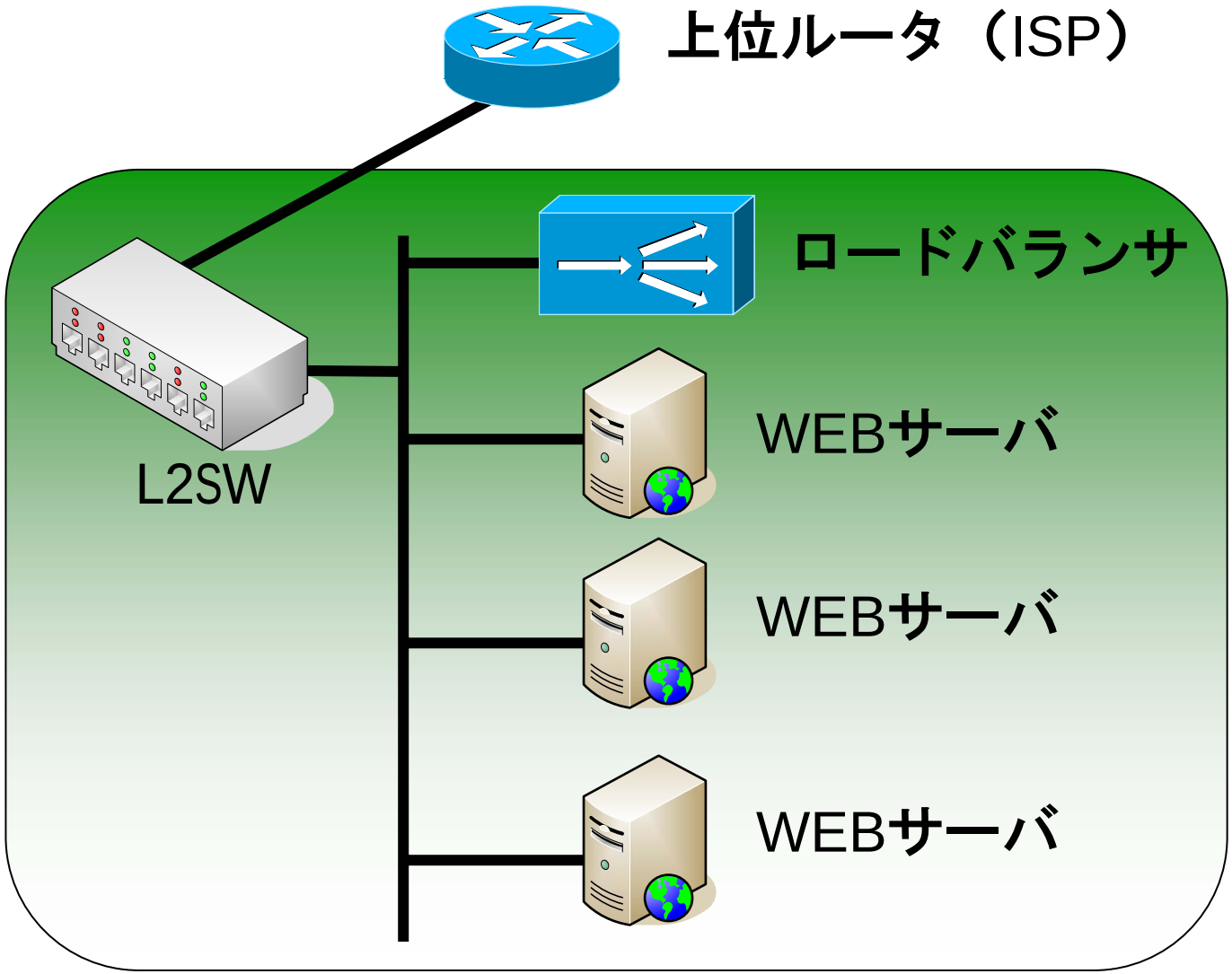
KLab

- 超重要なロードバランサが壊れたときに
  現地に行って代替機の設置作業などが必要

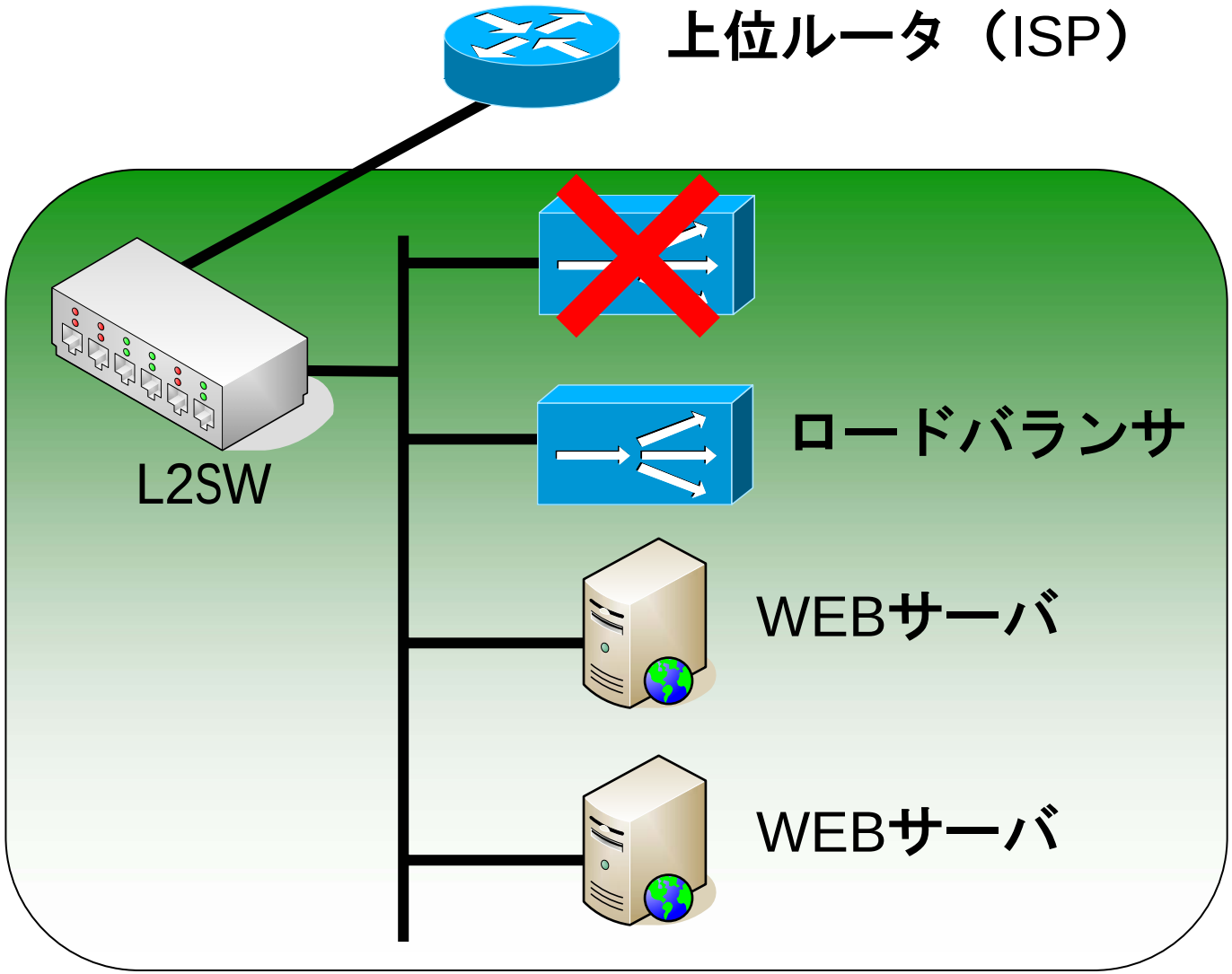
- LANケーブルの抜き間違いによる事故も誘発
  しかねません　→　全停止の危険性

- なにはともあれ面倒かもです
- 精神衛生上あまりよろしくありません

DSASではこうしています



DSASではこうしています



なにが嬉しいか

KLab

- ロードバランサは Linux + IPVS + keepalived です
- つまり、どのマシンでもロードバランサにできます
- ケーブルを繋ぎかえる必要がないので
  - 現地に行く必要がありません
  - 短時間で復旧することができます
- 壊れたマシンは後でゆっくりどうにかすればいいです

KLab

この構成ステキすぎです！

KLab

と、思った方、手を挙げてください！

KLab

じつは・・・・・

KLab

これには大きな落とし穴があります！

KLab

ロードバランサの外側と内側で
ブロードキャストドメインが一緒なんです

なにが問題か

KLab

- L2SWと上位ルータを繋ぐ線はインターネット接続用です
- ランニングコストがかかっていて、帯域も限られています
- それなのにこの構成では
  - 全サーバからのARPリクエストが流れ出てしまいます
  - 全サーバからのマルチキャストも流れ出てしまいます
  - 帯域を無駄に消費してしまいます

- 逆に、上位ルータからのブロードキャストも内部に流れ込んでしまいます
- もしかすると上位ルータから内部のサーバに直接アクセスできちゃうかもしれません

KLab

これどうしましょう

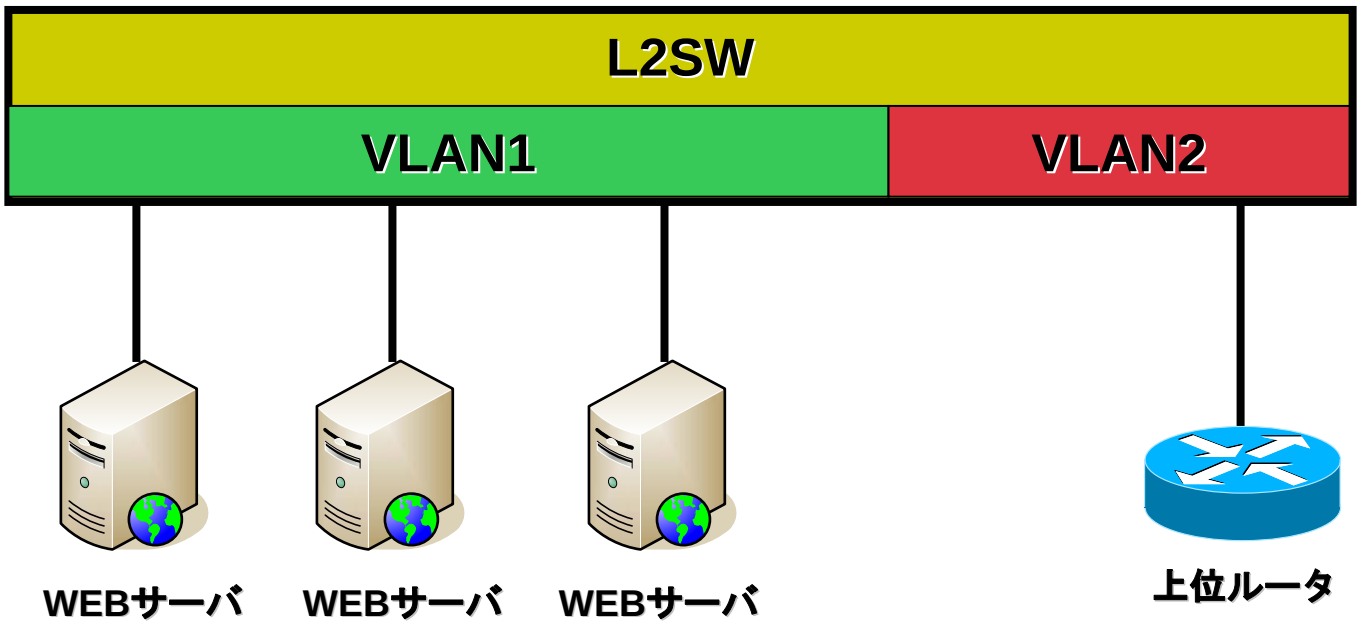
KLab

やっぱVLAN切るのがベターすかね

VLANとは

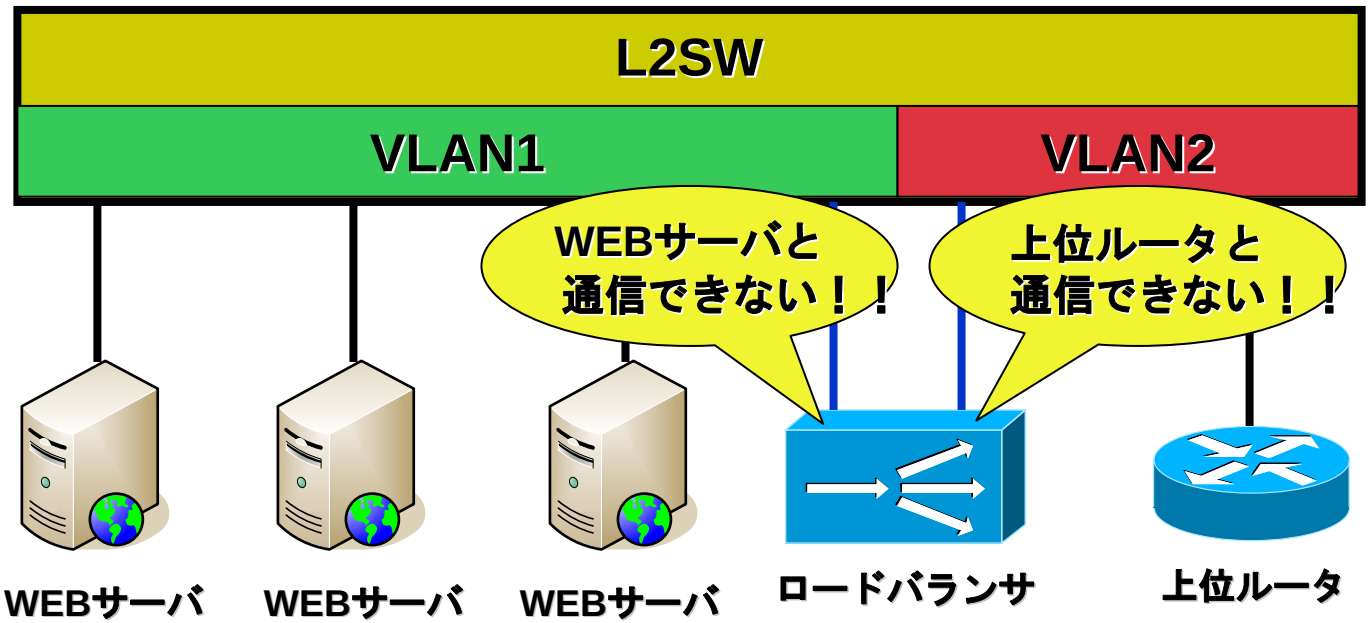
KLab

- L2SWでブロードキャストドメインを分割する仕組みです
- 以下のようにVLANを切ると、VLAN1に接続された機器とVLAN2に接続された機器同士は直接通信できなくなります。



ロードバランサはどこに繋ごう

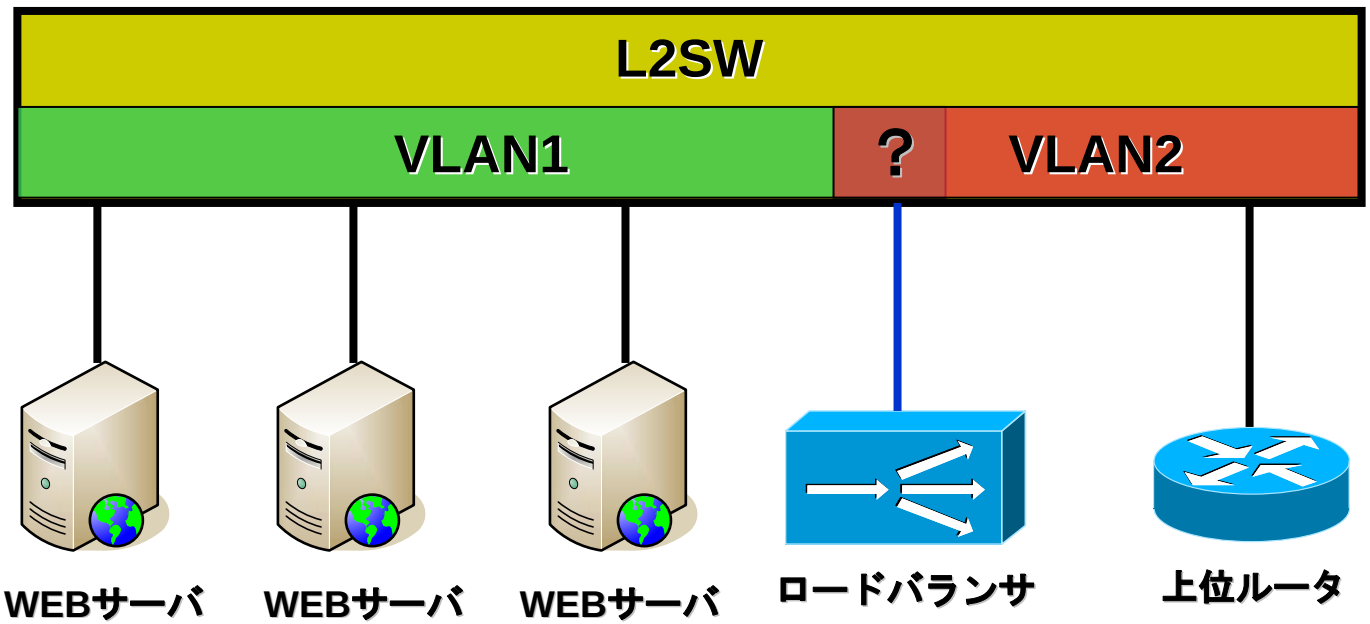
- VLANを切る事でブロードキャストドメインを分ける事ができました
- では、ロードバランサはどちらのVLANに繋げばよいのでしょうか
- VLAN1に繋ぐと上位ルータと通信ができなくなります
- VLAN2に繋ぐとWEBサーバと通信ができなくなります



ロードバランサはどこに繋ごう

KLab

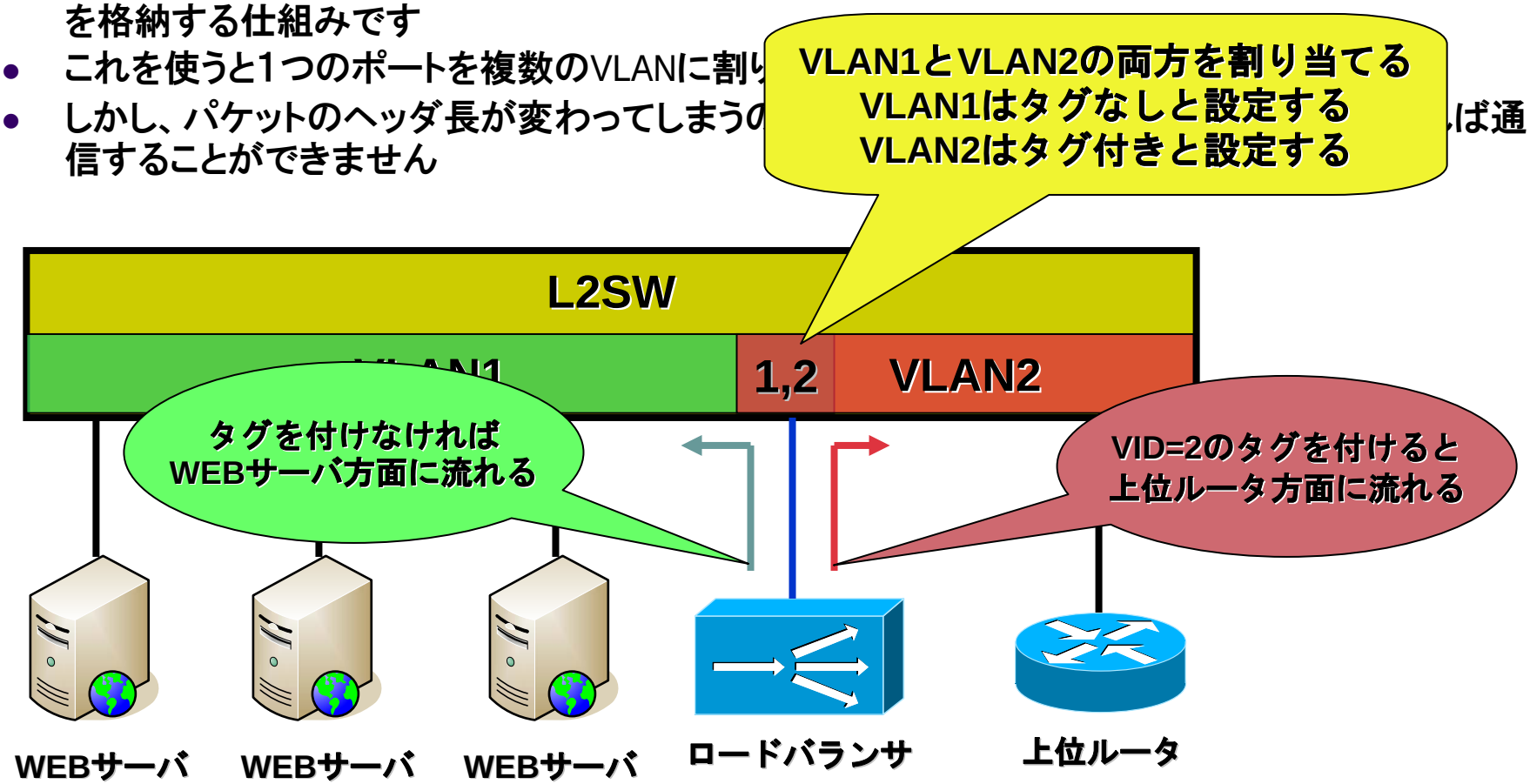
- ロードバランサは両方のVLANと通信できる必要があります
- ロードバランサ用のポートはVLAN1とVLAN2に割り当てなければいけません
- そのためには、ロードバランサが送出するパケットがどちらのVLANに向けたものなのかを識別できる仕組みが必要です



こんな時にはタグVLAN(IEEE802.1Q)

KLab

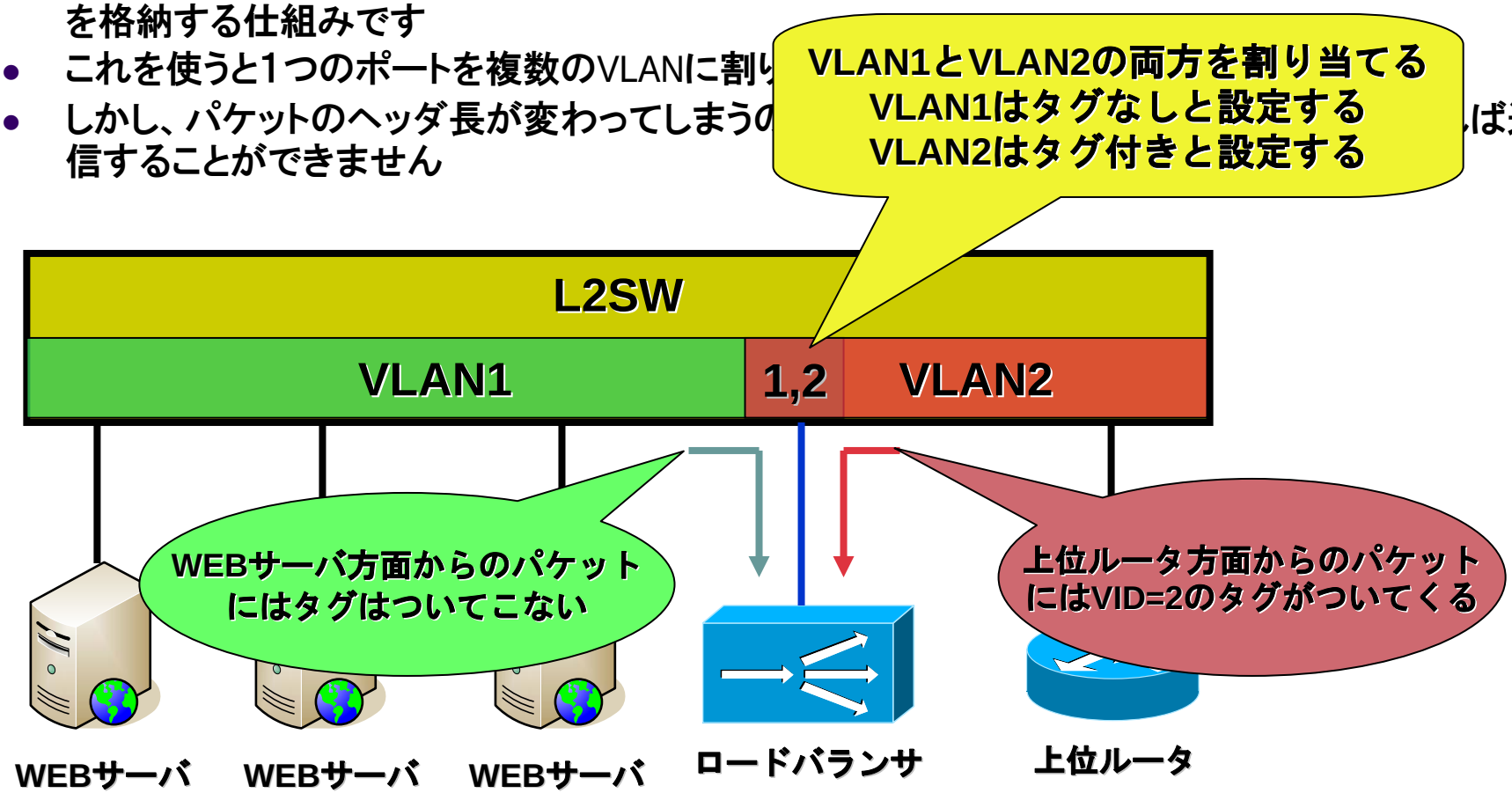
- タグVLANとは、パケットのイーサフレームに2オクテットのデータフィールドを追加して、そこにVIDを格納する仕組みです
- これを使うと1つのポートを複数のVLANに割り
- しかし、パケットのヘッダ長が変わってしまうの ば通信することができません



こんな時にはタグVLAN(IEEE802.1Q)

KLab

- タグVLANとは、パケットのイーサフレームに2オクテットのデータフィールドを追加して、そこにVIDを格納する仕組みです
- これを使うと1つのポートを複数のVLANに割り
- しかし、パケットのヘッダ長が変わってしまうの ば通信することができません



LinuxでもタグVLANが使えます

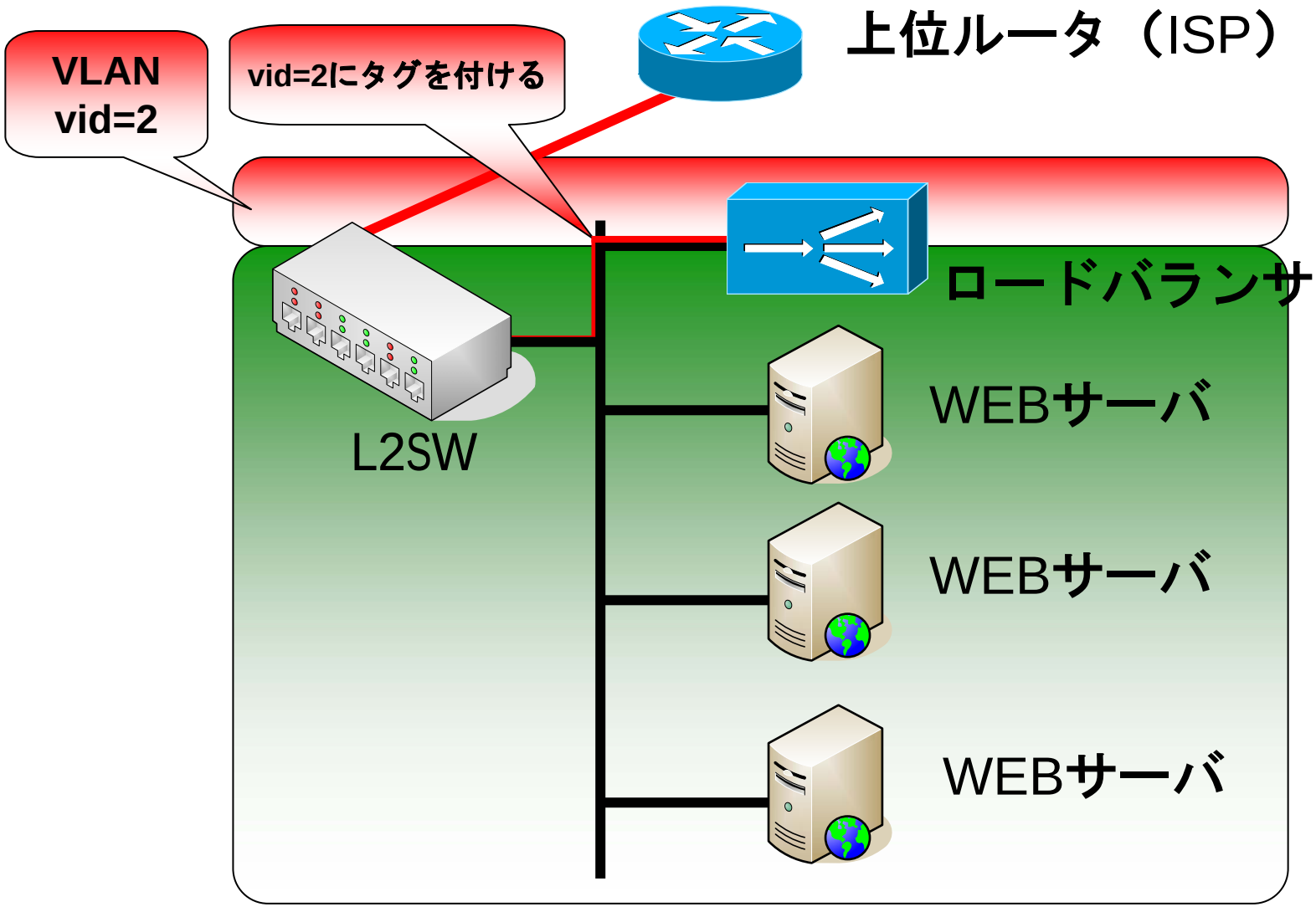
KLab

- LinuxのタグVLAN機能

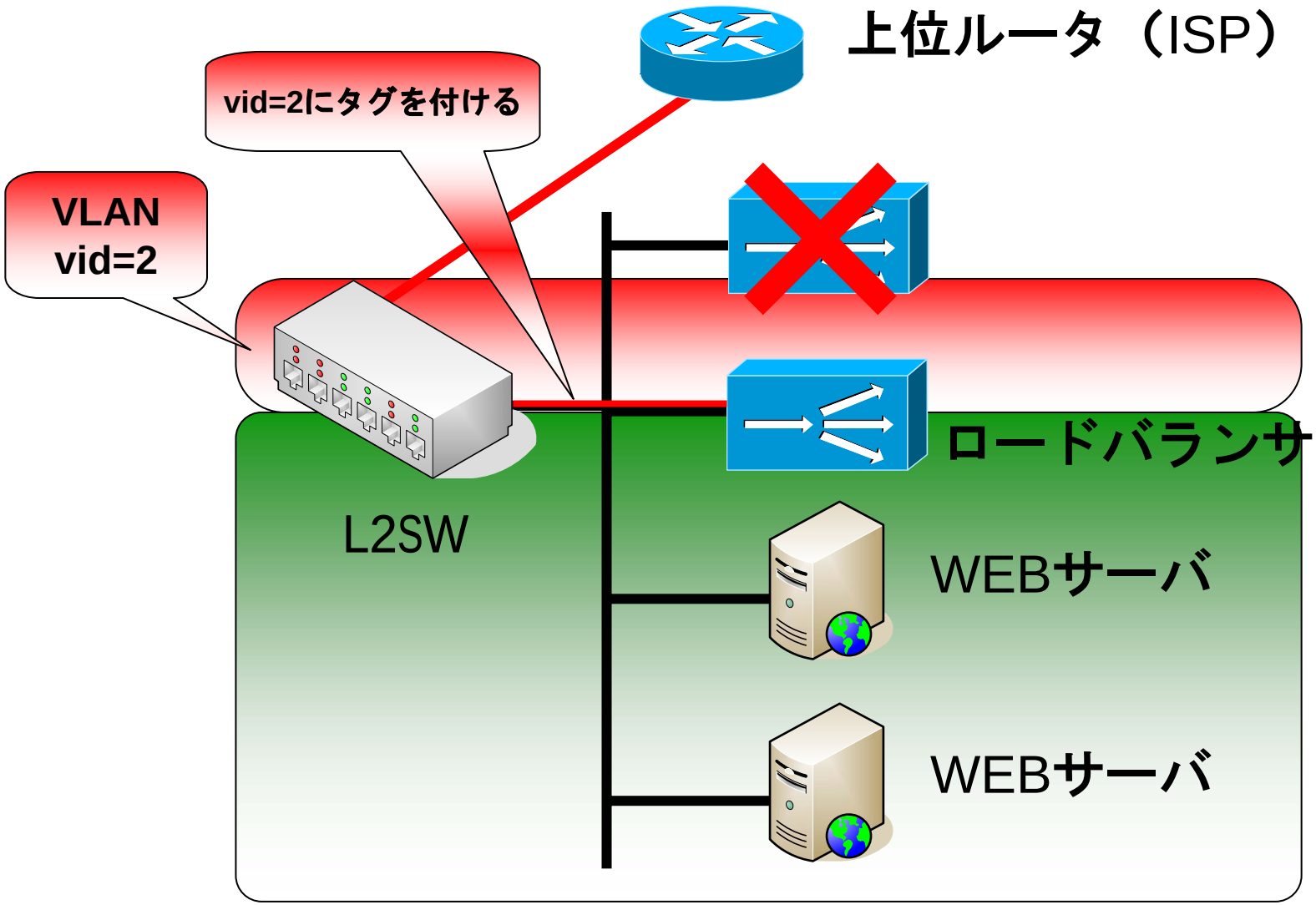
```
# vconfig add bond0 2
# ifconfig bond0.2 上位ルータと通信するためのグローバルアドレス
```

  - vconfigコマンドで bond0.2 というデバイスを作成します
  - bond0.2には上位ルータと通信するためのアドレスを割り当てます
  - bond0にはWEBサーバと通信するためのアドレスを割り当てます
  - 上位ルータに向かうパケットにはVLANタグがつきます
  - 上位ルータから到達するパケットにもVLANタグがついてきます
  - WEBサーバに向かうパケットにはVLANタグはつきません
  - WEBサーバから到達するパケットにもVLANタグはついてきません

タグVLANでセグメント分割



故障時の対応



LVSのマシンを交換するときは

KLab

- L2SWのタグVLANの設定を変更
- 暇そうなWEBサーバをロードバランサにする
- 作業は全てリモートから
  - 作業時間を短縮できる
  - 事故防止にもつながる

KLab

これでロードバランサが故障しても
簡単に復旧できるようになりました

KLab

めでたしめでたし

KLab

そういえば・・・・

DSASのコンセプトってなんでしたっけ？

DSASのコンセプト

KLab

システム管理者にも十分な睡眠を！

でしたね・・・

余談ですが

KLab

以前、ある勉強会にて

KLab

「MySQLのマスタは SPOFじゃないんですか？」

と、某???社のかっこいいおにいさんに
スルドイ突っ込みを頂いた事があります

そのときは、

「自動だろうと手動でやろうと、
フェイルオーバには違いないので、
SPOFではありません」

なんて・・・・・

KLab

強気な態度に出てしまいました(_ _)m

KLab

しかし今回は・・・

KLab

「MySQLのマスタがこけると
夜中でも起こされるんじゃないですか？」

KLab

と言われると・・・

KLab

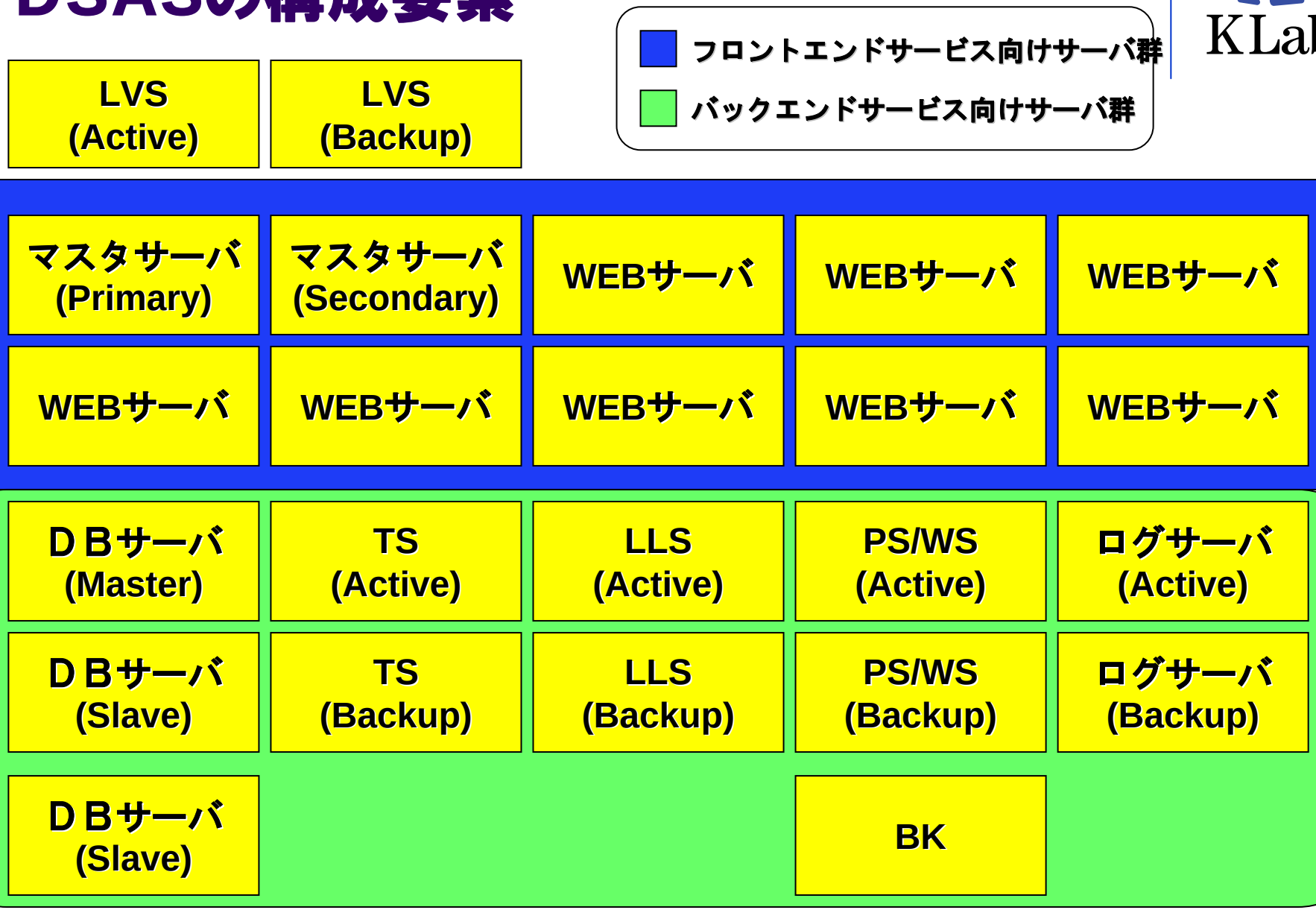
返す言葉もございません ｍ(＿＿)ｍ

KLab

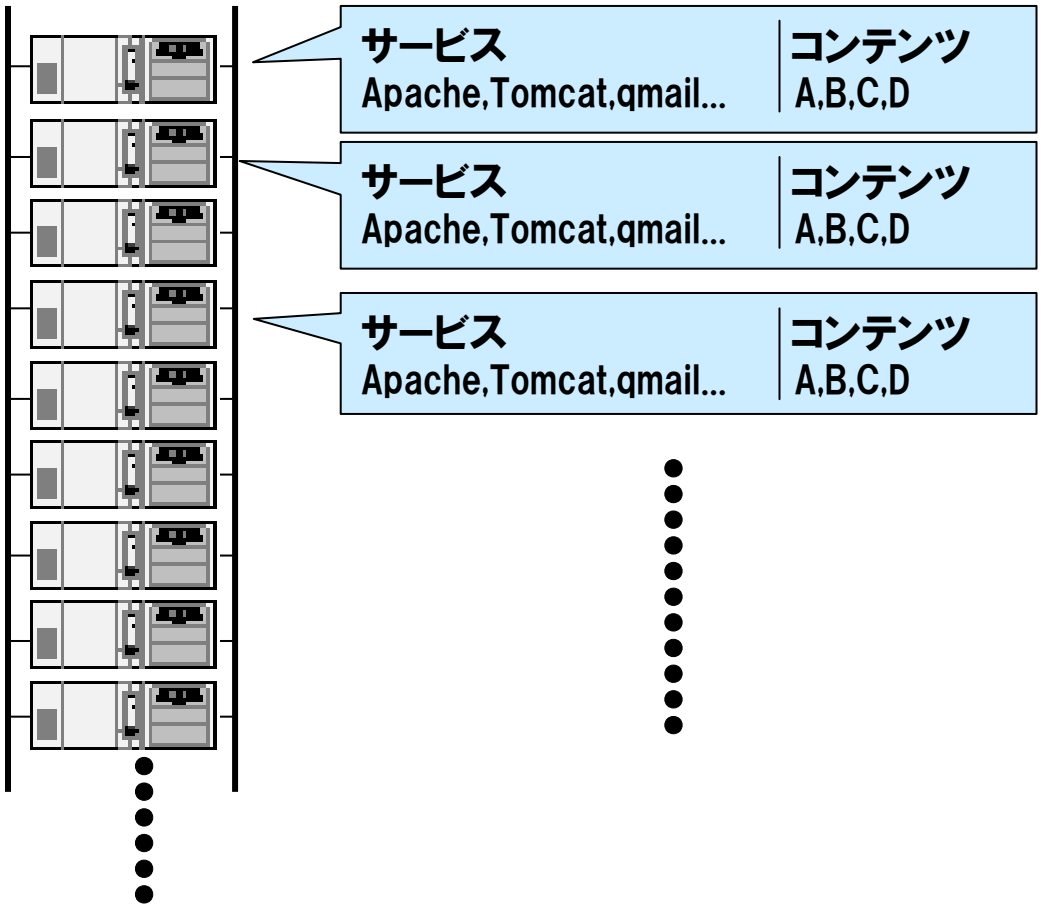
ということで、
この質問は自粛していただきますよう、
よろしくお願いいたします。

DSASの構成

DSASの構成要素



どのサーバもディスクの中身が同じ



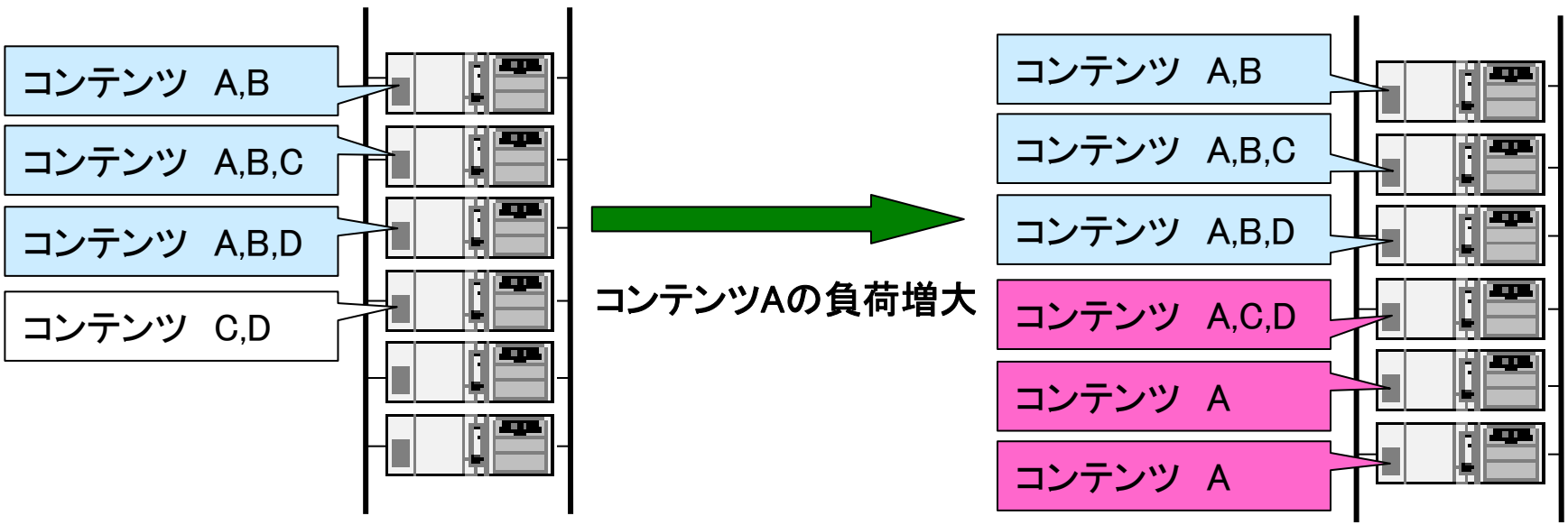
すべてのハードディスクの内容が同じ

どのサーバも全てのコンテンツ、
全てのサービスを提供可能

割当を変えるだけで
各サーバの役割が早変わり

アクセス増減に柔軟な対応が可能

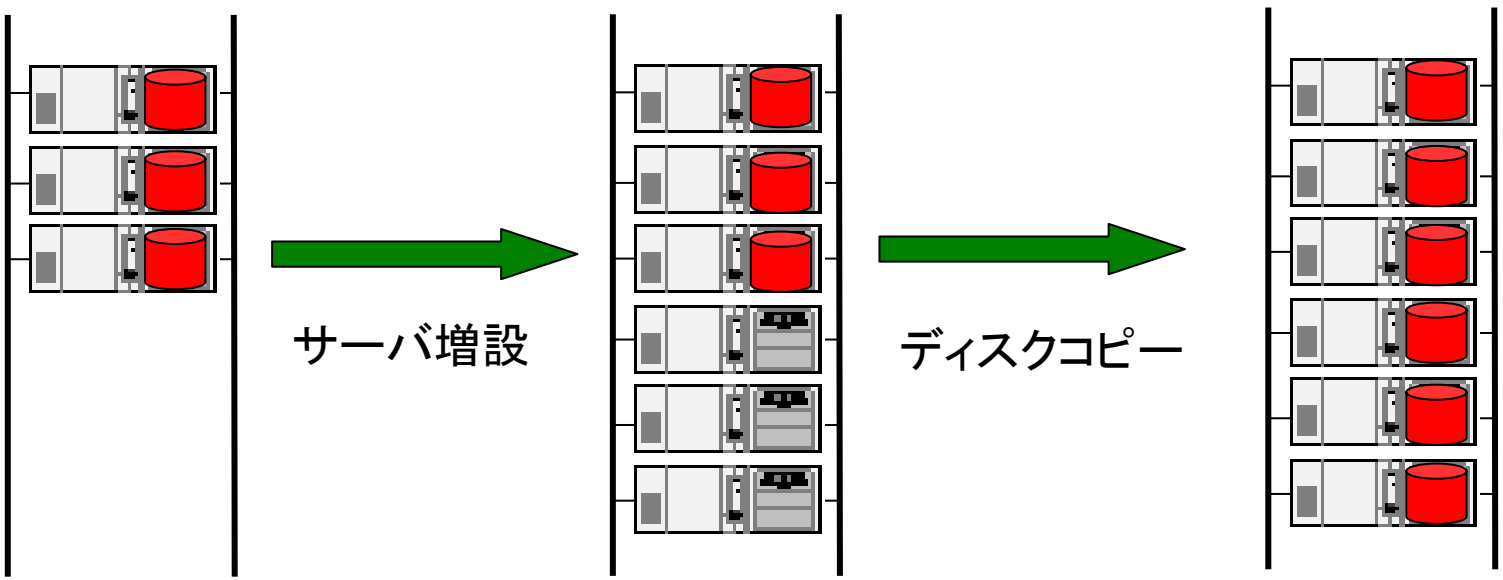
- ディスクの中身が同じなので、サーバ割り当てを操作するだけで増強完了
  - CMやメルマガなどの急なアクセス増加にも即時に対応可能
- アクセスが少ないコンテンツはサーバの相乗りも可能



故障機の復旧やサーバの増設が容易

KLab

- ディスクをコピーするだけでサーバ増設が完了
- サーバが故障した場合も迅速な復旧が可能
- PCサーバを利用しているのでパーツの手配も容易



DSASの構成要素

フロントエンドサービス向けサーバ群

KLab

- マスタサーバ
  - その名の通り、全サーバのマスタとなるサーバです
  - サイトを更新する場合は、まずマスタサーバにファイルを転送した後、専用のスクリプトを使って全サーバに展開します
  - Apacheなどをバージョンアップするときも、マスタサーバで実施してから全サーバに展開します
  - 万一、マスタサーバが壊れても、WEBサーバをマスタサーバにすることができます
  - また、なぜかdhcpやtftpサーバだったりもします
    - ネットブートするマシンはここからファイルを取得します

フロントエンドサービス向けサーバ群

KLab

- WEBサーバ
  - エンドユーザ向けのサービスを提供するサーバです
  - ロードバランサはWEBサーバに対して負荷分散します
  - 全サーバがどのサービスでも提供する事ができますが、
  - 割り当てられたサービスだけを提供しています
  - 「サービスの割り当て」なんて面倒な事をせずに、
  - 「全サーバで全サービスを提供して、全サーバに負荷分散」
  - でいいんじゃないの？って話もありますが・・・・・
    - ほとんどのサービスでTomcatをつかっています
    - サービス毎にクラスローダが動きます
    - 全サービスを読み込ませるとメモリがいくらあっても足りません
  - 必要なサービスだけをロードできる仕組みにしています

サーバ個別の設定はどうするの？

KLab

- ディスクの中身がみんな同じって言ったって
  - ホスト名がみんな同じだと訳わからなくなりますね
  - IPアドレスもみんな同じだと困った事になりますね
  - すべてのサーバが同じ機種とも限らないですよね
- マシン毎の設定ファイルはどうしても必要
  - いろんなファイルに設定が分散されてるのはいや！
  - /etc/hostname とか /etc/network/* とかいろいろ

/etc/dsas.confの登場

KLab

- マシンの固有情報はすべてここに集約
  - ホスト名
  - IPアドレス
  - ロードするカーネルモジュール
  - マウントするブロックデバイス
  - とかとか・・・・

/etc/dsas.confの内容

KLab

```
┬┌
#----- BaseConfigration -----
SYSTEMNAME=w101

#-----  NetworkSetting  -----
IPADDRESS0=192.168.1.1/22
GATEWAY[0]=192.168.1.254

#----- HardWare Setting -----
  HWTYPE=S6014HT
 MODULES=("bond0 arp_ip_target=$LVSADDRESS,$LLSADDRESS")
BTDEVICE=/dev/sda1
SWDEVICE=/dev/sda2
DBDEVICE=
FSDEVICE=
```

/etc/dsas.confの利用

KLab

- KLabオリジナルの起動スクリプトがいろいろ入ってます
  - /etc/dsas.confを参照して
    - ホスト名を設定したり
    - IPアドレスを設定したり
    - モジュールをロードしたり
    - ブロックデバイスをマウントしたり
    - その他もろもろやってます

DSASの構成要素

KLab

残念ながら時間がきてしまいました

KLab

まあ、なんだかんだいっても

DSASで利用している機能の大半は

KLab

- ずっとずっと昔から世にある仕組みだったり
- 誰かが知恵を絞って作り上げた仕様だったり
- 世界中の方々が手間と時間をかけて作ってくださったものだったりするわけで・・・・・

OSSベースでシステムを組むためには

KLab

- いっぱいある実装の中から必要なものを選択できること
- 足りない部分は自分で工夫してどうにかできること
- 問題が起きても人のせいにしないこと

これらが肝になりますよね、、、

「知恵」と「勇気」と「優しさ」が必要ですよね(><)

KLab

ということで

KLab

世界中のオープンソーステクノロジーに貢献されている方々に感謝！

KLab

本日、参加してくださった皆様に感謝！

今日はここまでにしたいとおもいます

KLab

ご清聴ありがとうございました

KLab

では質疑応答タイムにしたいと思います