学生実験
2日目
経路制御

IPネットワークアーキテクチャ

江崎研究室

インターネットにおける経路数

全ての経路情報 a.k.a. フルルート
約500,000

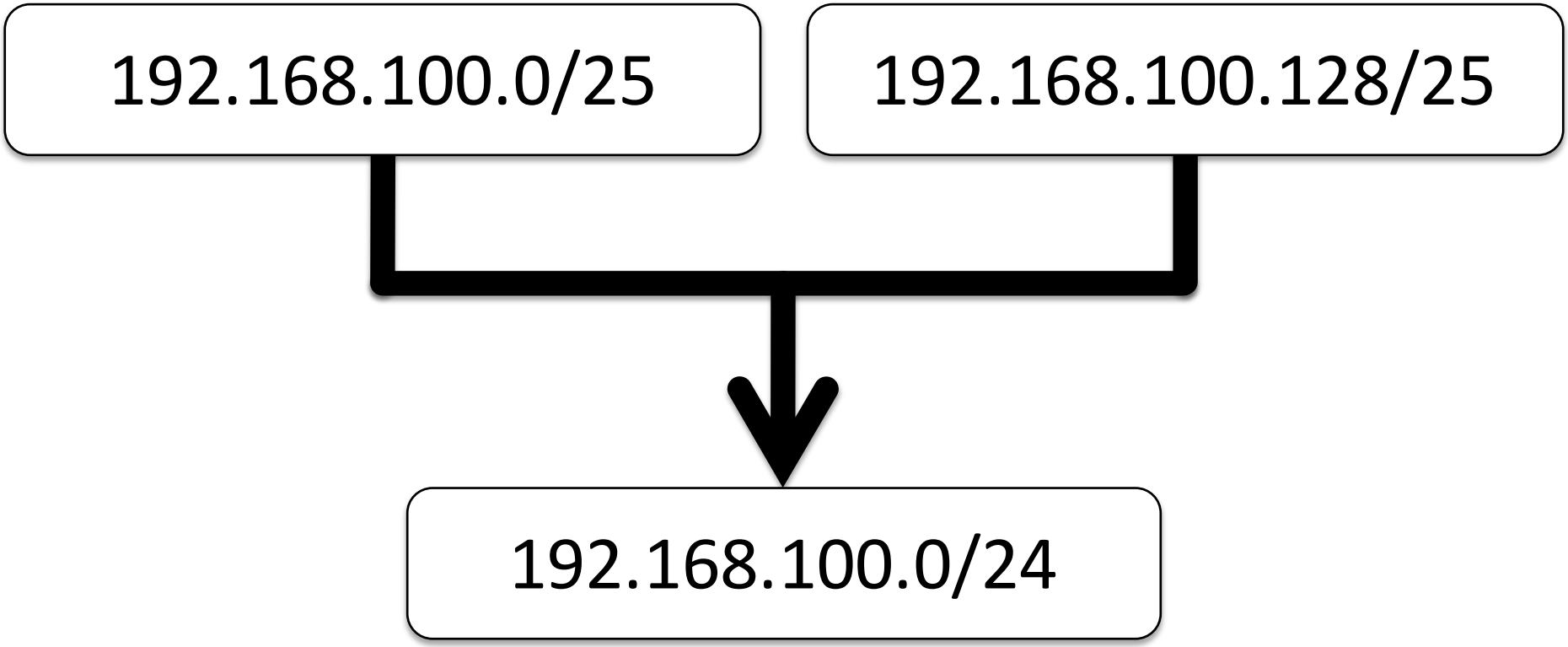
http://bgp.potaroo.net/bgprpts/rva-index.html

参考:世界のインターネット人口=約18億人
IPアドレス32bit=約43億通り

想像より多いですか？少ないですか？

ヒント）ひとつのネットワーク＝ひとつの経路

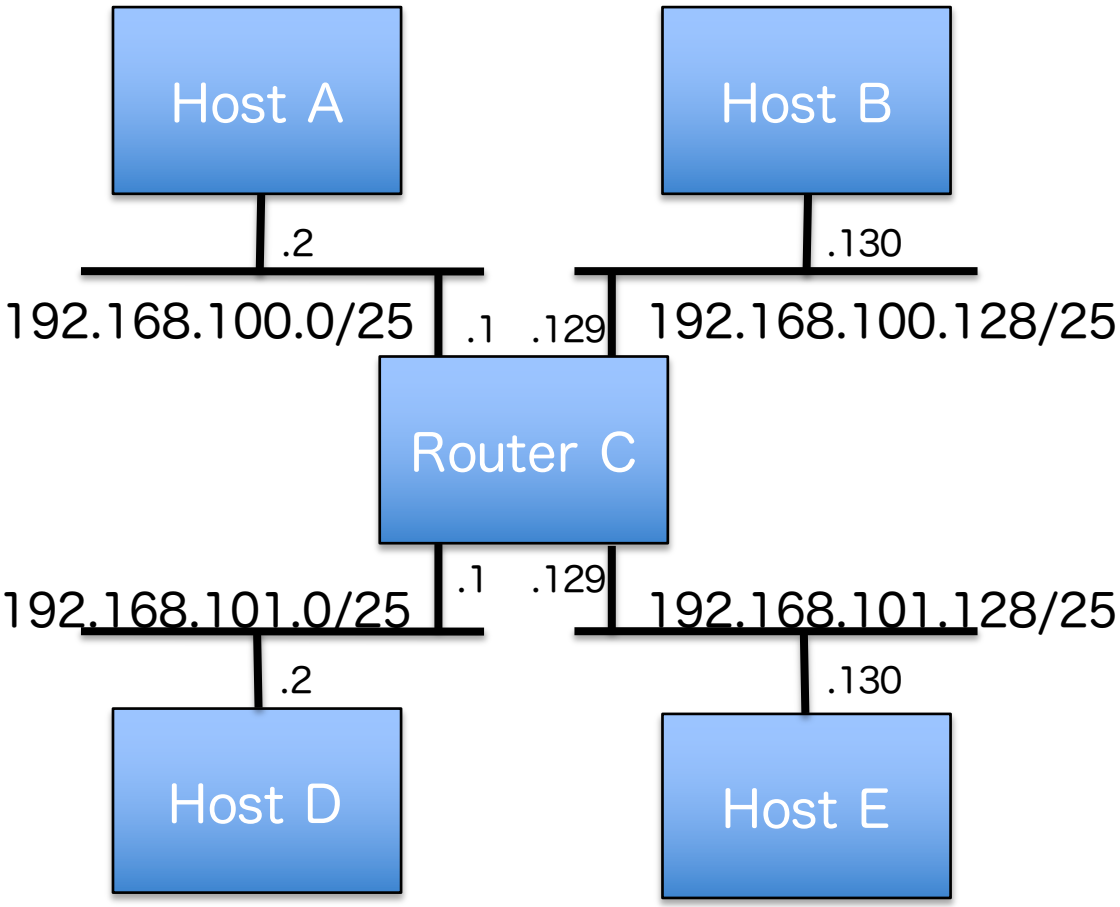
経路情報の集約



同じ方向であれば経路情報を集約できる

課題(1) 経路の集約

最後の課題で作成したネットワーク構成等において、アドレス割り当てを工夫し、最低一つのルータで経路集約できるようにしなさい。



ルータにおける経路選択

経路表の例）

```
$ ip route
10.100.0.0/16  via 192.168.0.2  dev eth0
10.100.10.0/24  via 192.168.1.2  dev eth1
192.168.0.0/24  dev eth0 proto kernel  scope link  src 192.168.0.1
192.168.1.0/24  dev eth1 proto kernel  scope link  src 192.168.1.1
```

Question

10.100.10.10 宛ての
パケットが来た場合
どちらの経路が選択されるか？

Answer

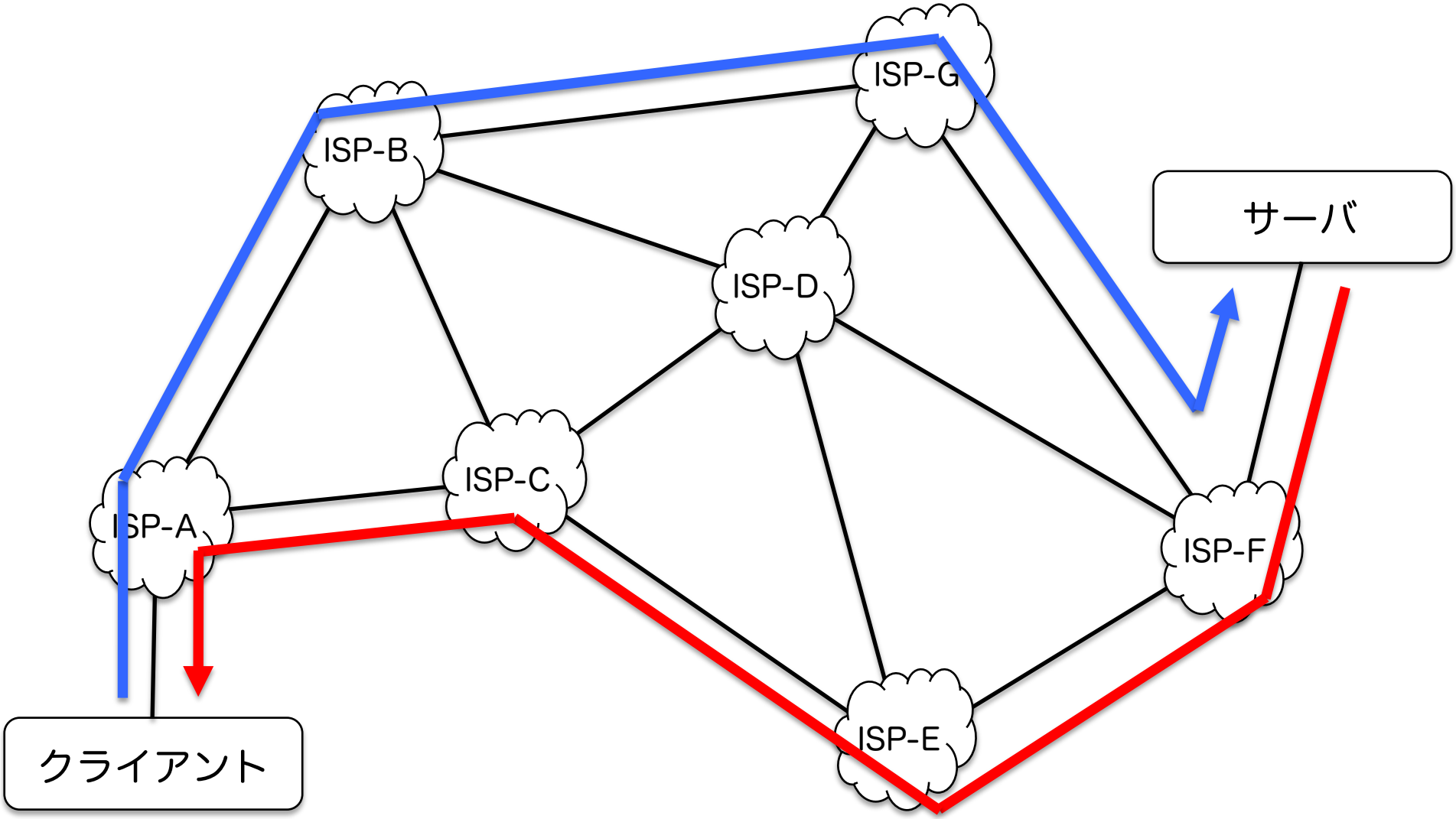
10.100.10.0/24 が選択される
ネットマスクが長い経路がより正確

最長一致（ロンゲストマッチ）

課題(2) 最長一致の確認

課題(1)と同じネットワーク構成で、アドレスアサインを工夫し、最低一つのルータで複数の経路項目にマッチする場合を作成し、pingとtracerouteを用いて、最長一致が実現されることを確認しなさい。

非対称な経路



課題(3) 経路の確認

インターネットには研究や運用管理に有用なpingやtracerouteを実行できるサイトが存在する。”looking glass traceroute”をウェブ検索のキーワードとし、そのようなサイトを探しなさい。また、自分のPCとそのサイト間、および複数のサイト間でtracerouteを実行し、経路の対称性の有無を確認しなさい。

経路制御技術

静的経路制御

メリット	• シンプル • ネットワークの状況に左右されない安定性
デメリット	• 冗長経路を上手に使えない • 規模が大きい場合、設定でミスが発生する可能性あり

動的経路制御

メリット	• ネットワークの状況に応じて柔軟に動作 • 設定項目がネットワークの規模に比例しない
デメリット	• 状況によってはループが発生する可能性あり • プロトコルの理解が複雑（かもしれない）

「到達性がある」ことの意味

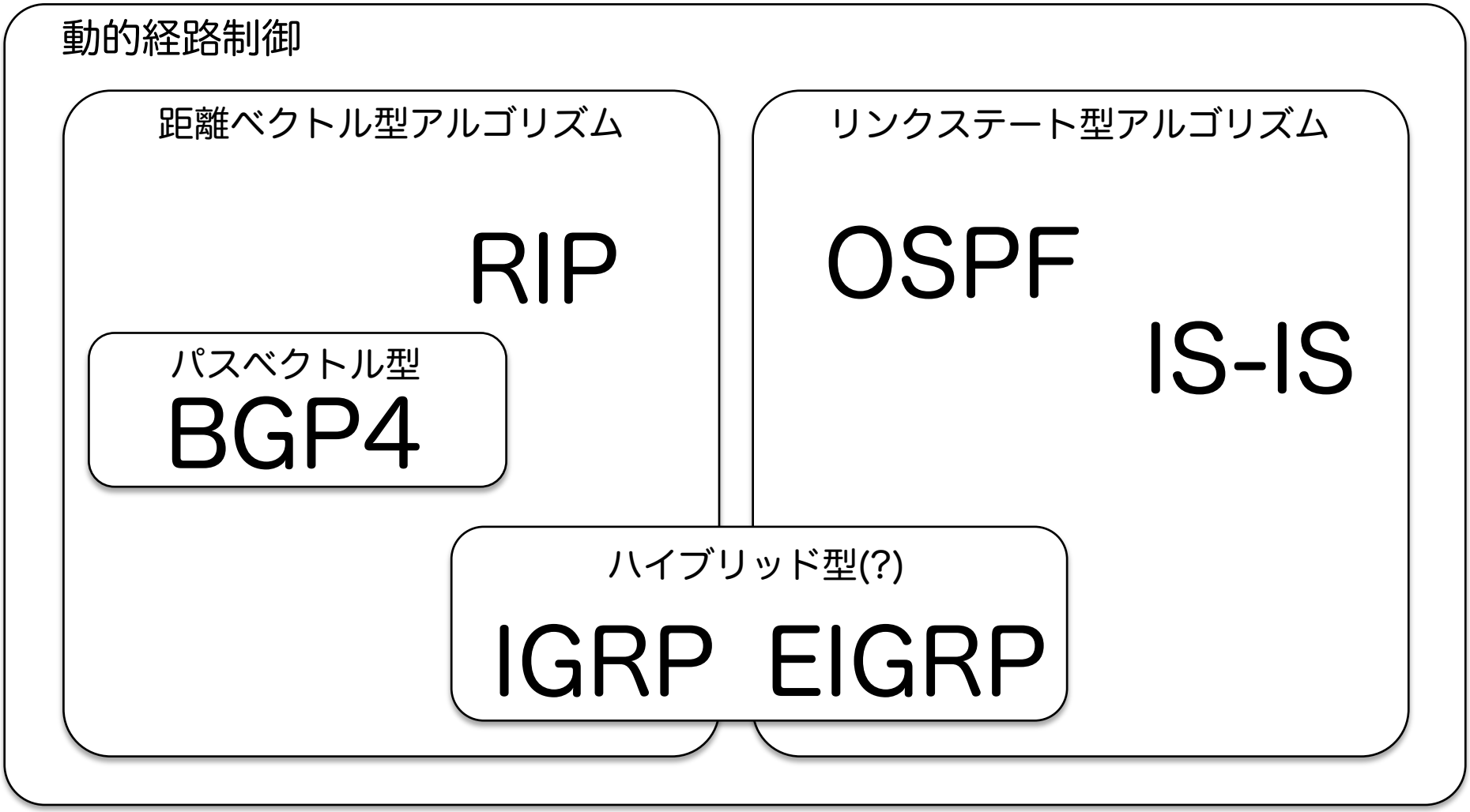
到達性がある＝経路情報がある　　到達性がない＝経路情報がない

インターネット津々浦々まで経路が存在することが重要

経路の広告

動的経路制御方式によるネットワークの存在の伝搬

経路制御分類学(1)


動的経路制御
距離ベクトル型アルゴリズム
RIP
パスベクトル型
BGP4
ハイブリッド型(?)
IGRP
EIGRP
リンクステート型アルゴリズム
OSPF
IS-IS

各アルゴリズムの概説

- 距離ベクトル型アルゴリズム
  - 友達の友達は友達
  - 自分の知っている友達リストを隣の友達と交換
  - 宛先のネットワークを距離と方向で表す
    - 距離＝ホップ数
    - 方向＝Next Hop Router
- リンクステート型アルゴリズム
  - 自分の周りの友達情報を友達全員に伝える
  - 自分を中心とした友達のネットワークを計算
  - 目的とする友達に到達する最短パスを利用する

経路制御分類学(2)

動的経路制御

IGP (Interior Gateway Protocol)

OSPF

RIP

IS-IS

IGRP

EIGRP

EGP (Exterior Gateway Protocol)

BGP4

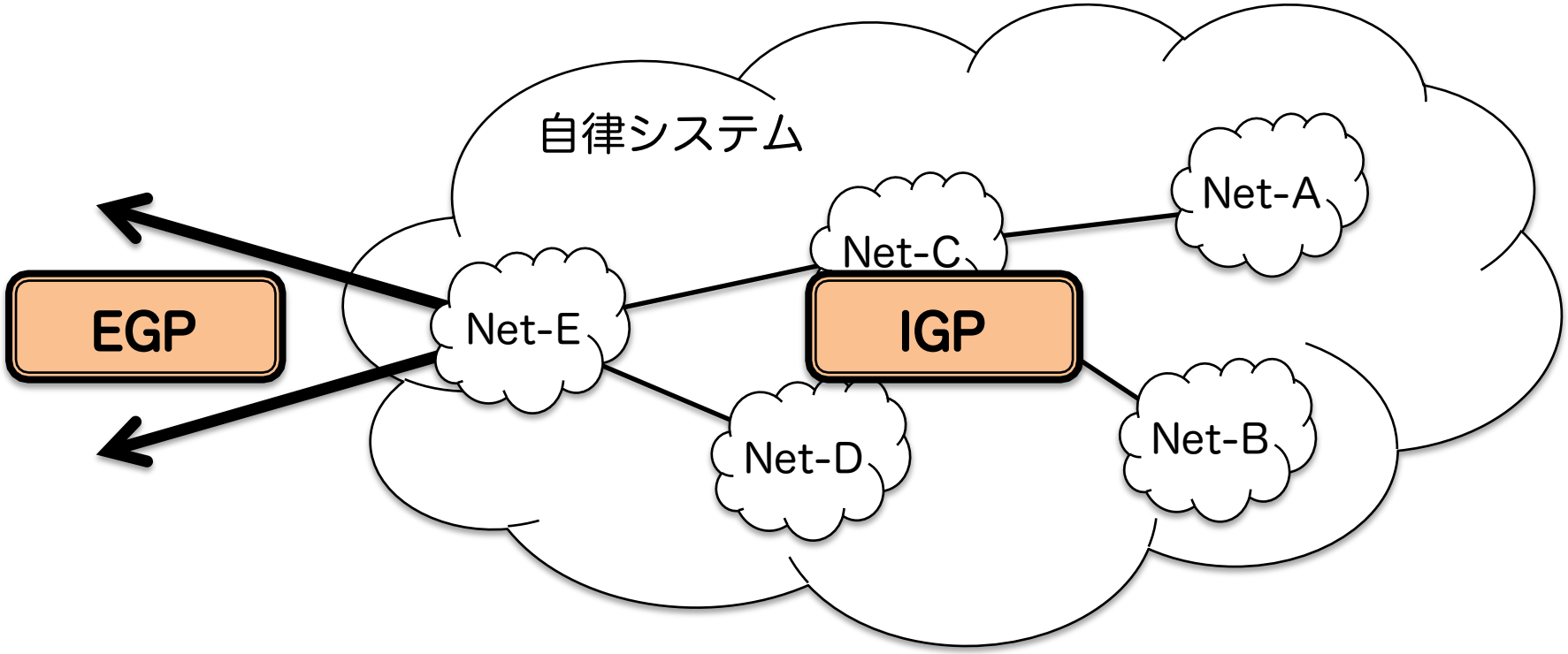
EGP

IGPとEGP

- IGP (Interior Gateway Protocol)
  - 自律システム内で経路情報を交換する
- EGP (Exterior gateway protocol)
  - 自律システム間で経路情報を交換する

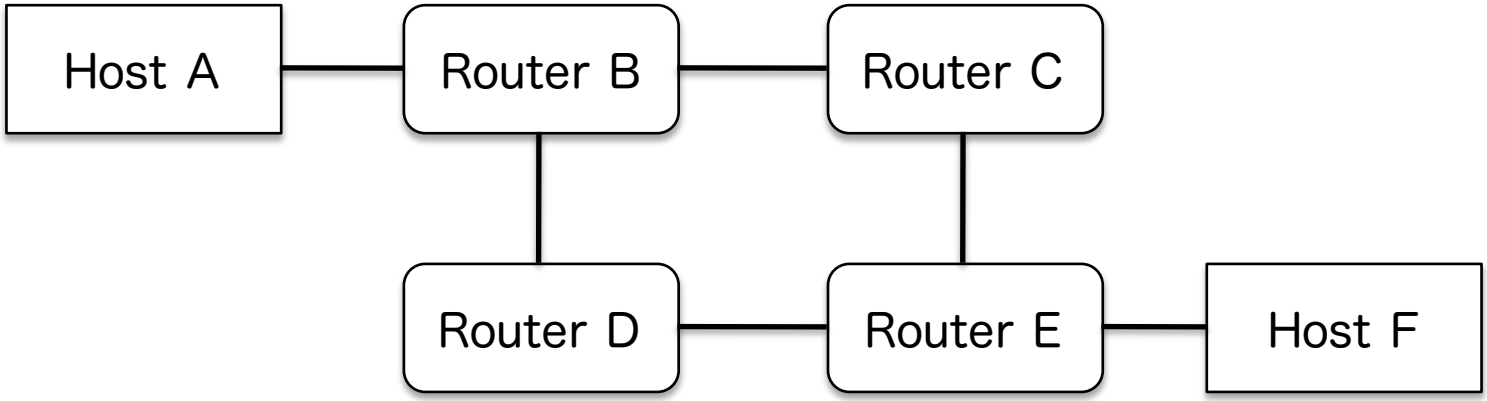
自律システム

- ネットワーク運用を自律して行う
- 独自に経路情報を選択できるネットワーク
- 通常、規模の大きなISP



課題(4) 静的経路制御を利用した設定

1. 下記のネットワークを構築
2. 静的経路制御でループするような設定を導入
3. どのような動作をするか想像してみる
4. ping/tracerouteを利用して確認
5. tcpdumpを用いて流れるパケットの様子を確認



注意点）
ループを構成し、動作させるには、
Reverse Path Filterを無効にしなければならない

ルータ動作時のシステム設定

- IPフォワーディング
  - 他ホストから受け取ったパケットを別の経路へ再送信するための設定
  - サーバでないPCをルータ動作することは稀なためデフォルトはオフ
- Reverse Path Fileter
  - あるパケットに対する返信がそのパケットの入ってきたインターフェースに向かわない場合、そのパケットを破棄するための設定
  - (複数インターフェースをもつ場合)出したパケットに対する応答は同じインターフェースに戻るべきという思想
  - 各インターフェース毎に設定が必要
  - スプーフィング等を防ぐためデフォルトはオン

これらの設定がデフォルトのままだと正しく動作しない

- sysctlコマンドにより設定

```
# sysctl -w net.ipv4.ip_forward=1
# sysctl -w net.ipv4.conf.all.rp_filter=0 など
```

- network-manager起動時に再設定される (毎回設定が必要)
- 毎度の手動操作はトラブルを誘引するので、実験webページTIPSの「シェルスクリプト」で紹介しているスクリプトを利用しまとめて設定すること

課題(5) 動的経路制御を利用した設定

1. RIPの設定
   - Quaggaの利用
2. RouterC/Dを停止させ経路の切替を確認
3. tcpdumpを用いて流れるパケットを観測
4. Quaggaのデバッグモードを利用し動作を確認

Quaggaとは?

PCで動作する経路制御プログラム
Zebraをベースに開発
CiscoのCLIに類似した設定方法
Cf. Vyatta (vee-AH-tah)なんてのもある

Quagga 入門 (1)

インストール：

```
# apt-get  install  quagga
```

設定ファイルの導入、確認（必要なものだけ、今回の場合）：

```
# cp /usr/share/doc/quagga/examples/zebra.conf.sample  /etc/quagga/zebra.conf
# cp /usr/share/doc/quagga/examples/ripd.conf.sample  /etc/quagga/ripd.conf
# cp /usr/share/doc/quagga/examples/vtysh.conf.sample  /etc/quagga/vtysh.conf
```

起動デーモンの指定:
/etc/quagga/daemonsを編集 （zebraとripdが起動するよう設定）

Quaggaの実行：

```
# /etc/init.d/quagga  start
```

次のスライドに続く

Quagga 入門 (2)

Quaggaへのアクセス：
(1) $ telnet localhost ripd

daemon名か
ポート番号で指定

daemon	port
zebra	2601
ripd	2602
ospfd	2604
bgpd	2605
ospf6d	2606

(2) # vtysh
　直接、プログラムにも接続可能

RIPルータの設定：
隣接するネットワーク及び
インターフェースを設定

RIPの設定例）

ここにRIPを動作させる
ネットワーク情報が追加される

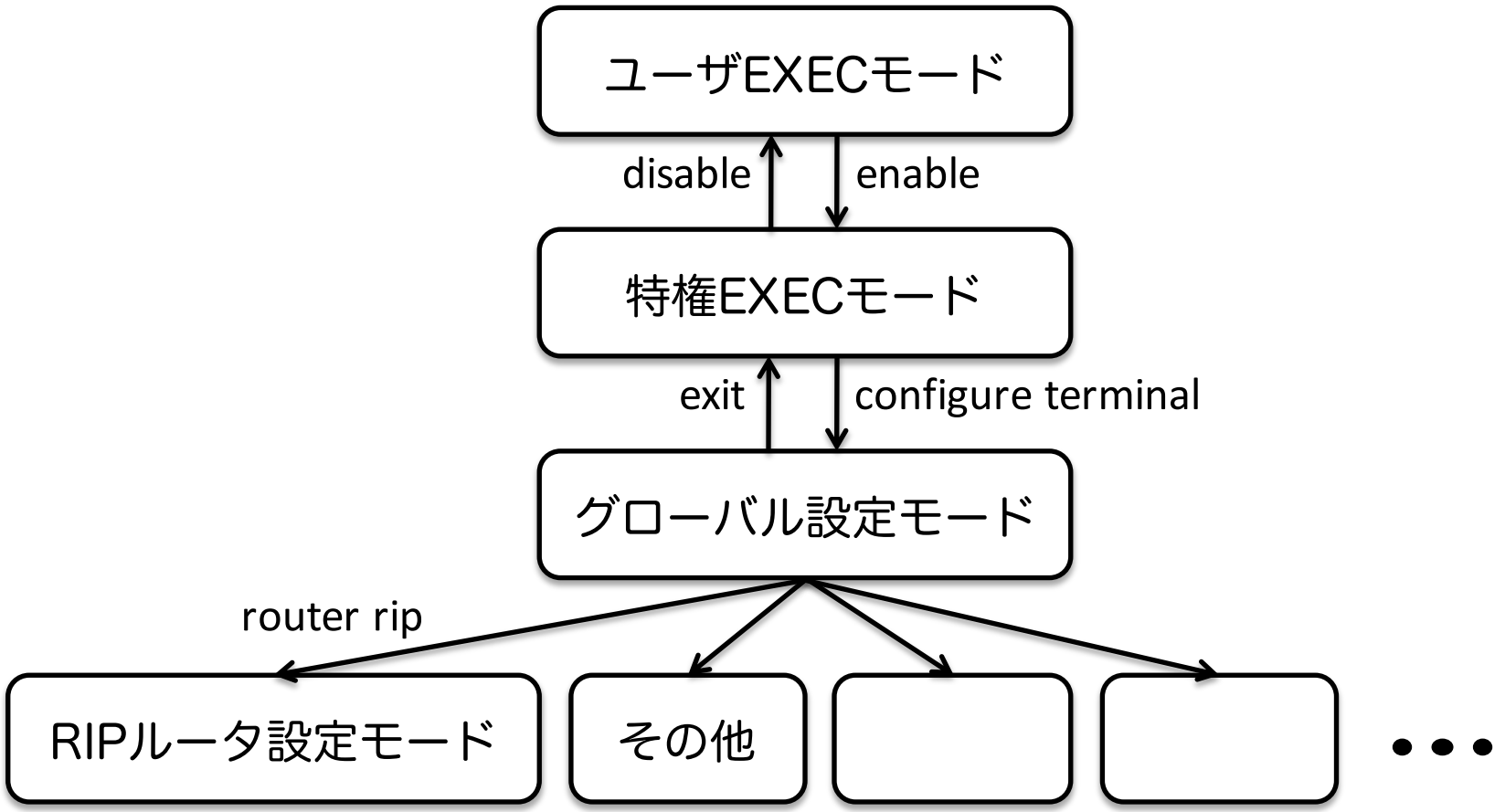
```
router rip
  network a.b.c.d/n
  network eth?
```

詳しくは実験webページのTIPSの資料を参照

プロンプト

- ユーザEXECモード
  Router>
- 特権EXECモード
  Router#
- グローバル設定モード
  Router(config)#
- 個別設定モード
  Router(config-router)# など

モード（状態）の移動


ユーザEXECモード
disable
enable
特権EXECモード
exit
configure terminal
グローバル設定モード
router rip
RIPルータ設定モード
その他

コマンドライン編集

入力	説明
ctrl-a	行頭にカーソルを移動
esc-b	１単語分、カーソルを左に移動
ctrl-b	１文字分、カーソルを左に移動
ctrl-e	行末にカーソルを移動
ctrl-f	１文字分、カーソルを右に移動
esc-f	１単語分、カーソルを右に移動
ctrl-k	カーソルの位置から後ろを削除
ctrl-d	カーソルの位置の文字を削除

ヒストリ機能

入力	説明
ctrl-p	一つ前のコマンドに戻る
ctrl-n	前に戻したコマンドから新しい方に移る
show history	実行したコマンドを表示

以前に入力したコマンドを繰り返し入力するときに使用
少しずつ変更がある似たようなコマンドの入力の時に便利

ヘルプ機構・補完機構

- ？
  - 入力可能なコマンドを表示
- TABキー
  - コマンド入力を補完
- ヘルプと補完は密接な関係がある
- コマンド名をすべて入力する必要はない

設定ファイルの確認

- 動作中の設定を確認する
  `show running-config`
- 起動時に参照される設定を確認する
  `show startup-config`