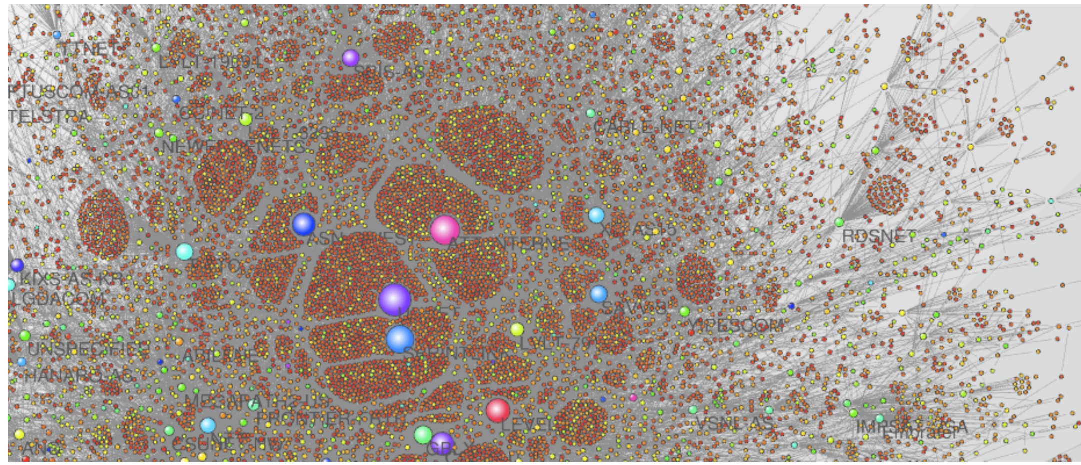
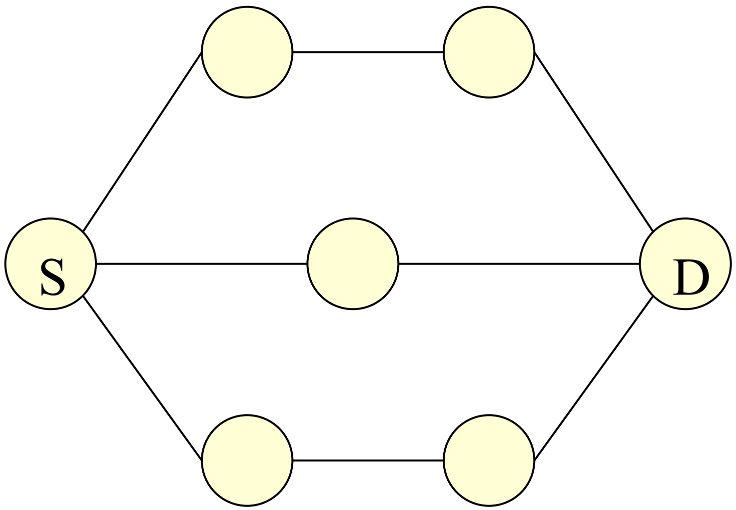
情報ネットワーク論I
経路制御（1）



門林 雄基
奈良先端科学技術大学院大学

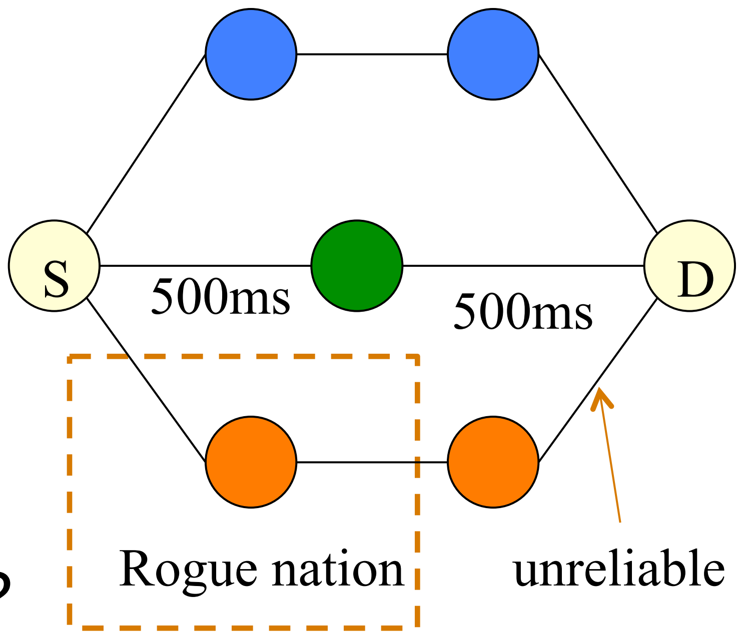
経路制御問題

- 送信元から宛先までどうやって辿り着くか?
- どの経路が最善か?
  - ホップ数
  - 遅延
  - 帯域
  - ポリシー制約、コスト...
- 誰が決めるか?
  - ルータ?
  - 送信元?
- どうやって送信の失敗を調べるか?
- オーバーヘッドはどの程度発生するか?



経路制御問題

- 送信元から宛先までどうやって辿り着くか?
- どの経路が最善か?
  - ホップ数
  - 遅延
  - 帯域
  - ポリシー制約、コスト…
- 誰が決めるか?
  - ルータ?
  - 送信元?
- どうやって送信の失敗を調べるか?
- オーバーヘッドはどの程度発生するか?

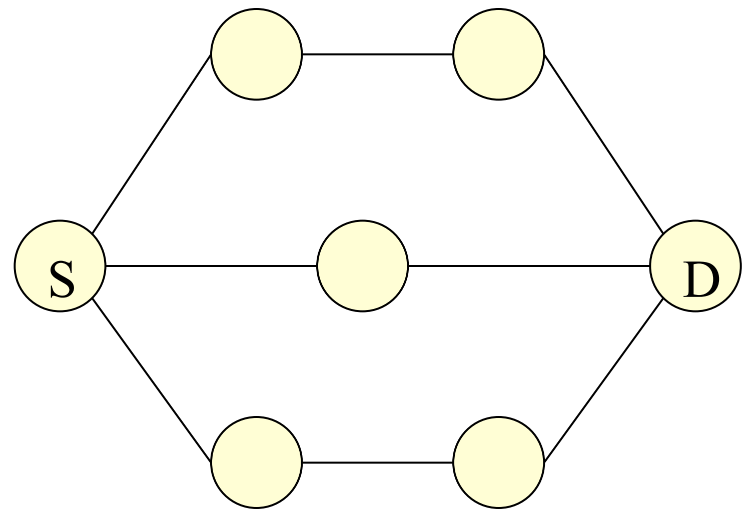


経路制御問題の解空間

- ネットワークの表現:
  - グラフ,
  - 行列
- 情報収集:
  - ネットワークじゅうで
  - いくつかのルータで
  - 近傍だけで
- 経路計算:
  - 全てのルータで
  - いくつかのルータで

… 解の実例: 経路制御システム

経路制御システムの類型



- Static routing
  - 経路をあらかじめ決定
- Dynamic routing
  - ネットワークの状況を動的に反映

- Source-based routing
  - 送信元のノードが送信先までの経路を計算する
- Hop-by-hop routing
  - 全てのノードでパケットの転送先を計算する

本講義の焦点: Dynamic routing, hop-by-hop routing

- Static routing
  - 経路をあらかじめ決定
- Dynamic routing
  - ネットワークの状況を動的に反映

- Source-based routing
  - 送信元のノードが送信先までの経路を計算する
- Hop-by-hop routing
  - 全てのノードでパケットの転送先を計算する

経路制御とは: 経路制御の機能

- End-to-end の到達性の確保
- 最短経路の自動発見
- トラフィックの分散

- 障害が発生したリンクの迂回
- ネットワーク障害の分離
- 管理ポリシの反映

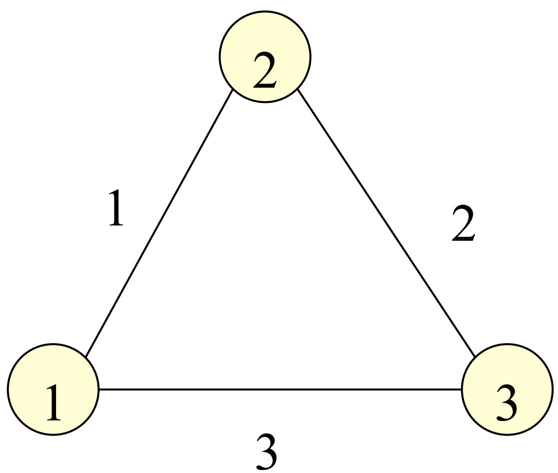
経路制御システムの特徴

- ネットワークの表現
  - ネットワークトポロジ
  - リンク間の接続属性
- 情報交換
  - 通信のオーバーヘッド
  - 伝搬速度
- 計算アルゴリズム
  - 計算にかかるオーバーヘッド
  - 経路の収束速度

- 経路制御システム: プロトコル + 情報 + アルゴリズム

ネットワークトポロジ

- トポロジ: 弾性変形によって変わることのない幾何学的な配置



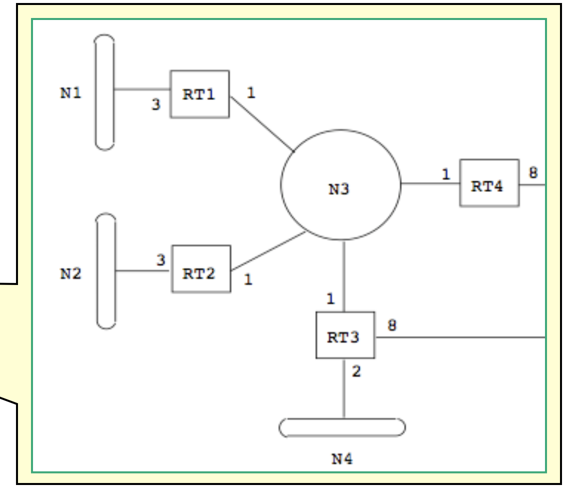
グラフによる表現

$$
\begin{array}{c c}
 & \begin{array}{ccc} 1 & 2 & 3 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} \infty & 1 & 3 \\ 1 & \infty & 2 \\ 3 & 2 & \infty \end{pmatrix}
\end{array}
$$

行列による表現

経路制御とは: 経路制御の構造

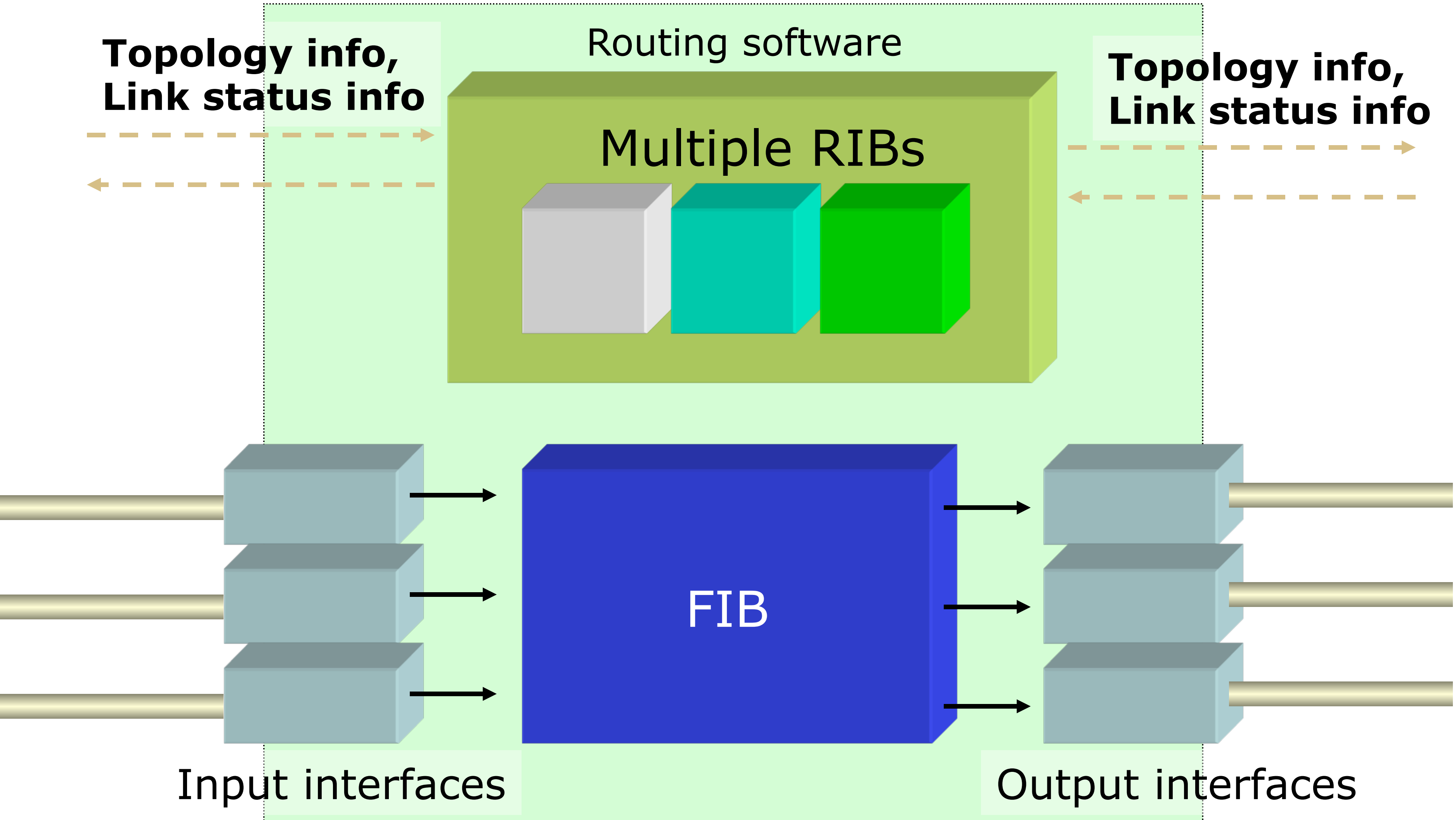
- 経路制御プロトコル
  - 隣接ルータを発見
  - トポロジ情報を交換
  - リンクの情報を交換



→ 経路情報 (RIB: Routing Information Base) を計算

- 複数の経路制御プロトコル → 複数のRIB
- 複数のRIBから単一のFIBへ
  (FIB: Forwarding Information Base)

Gateway Model



経路制御アルゴリズムの種類

- 距離ベクトル型経路制御
- リンク状態型経路制御
- パスベクトル型経路制御

- これらのアルゴリズムの違い:
  - トポロジ表現
  - 伝搬範囲 / 頻度 / リンク状態のタイミング
  - 最短経路の計算アルゴリズム
- トレードオフ:
  - 規模拡張性
  - 収束時間
  - アルゴリズムの単純さ

Questions?

距離ベクトル型経路制御
Distance vector routing

距離ベクトル型経路制御

DN p.397

Bellman−Ford algorithm

- 1. 自身への距離ベクトルを0に、他の宛先への距離ベクトルを∞に
- 2. 自身がもつ全ての距離ベクトルを隣接する全てのルータへ転送
- 3. 隣接するルータから到着した距離ベクトルと、自身から隣接ルータまでの距離を使用して、他の宛先への距離が最小となるような距離ベクトルを求める

- *Initial condition*
  - $b_i^{(0)} \leftarrow \infty \quad (i \neq D)$
  - $b_D^{(0)} \leftarrow 0$
- *Repetition*
  - $b_i^{(m)} \leftarrow \min_{j \in V_i}\{b_j^{(m-1)} + d_{ij}\}$
- *Terminal condition*
  - $b_i^{(m)} = b_i^{(m-1)}$
- ここで、D: 宛先ルータ、$b_i^{(m)}$: m回の反復演算によって求めたルータi, D の距離、V_i: ルータiと隣接するルータの集合、d_{ij}: ルータi, j を結ぶリンクの距離

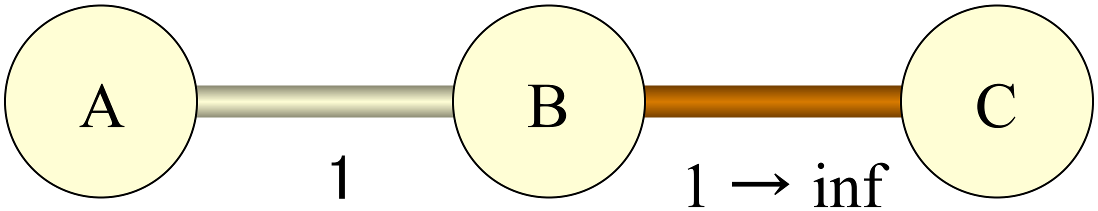
距離ベクトル型の問題点

- ルータ数Nに対し、
  - 計算量 $O(N^3)$
  - トラフィック $O(N^2)$

- 定期的に距離ベクトルを送るので、収束が遅い
  - ・ 新たな経路が発生した場合や、
  - ・ 既存の経路がなくなった場合

- 収束が遅い → 一貫性のない過渡的状態
  - Counting to infinity problem

無限カウント問題 (Counting to infinity)



Suppose link B-C went down.

B thinks: (C, inf)

A says: (C, 2)

B thinks: (C, 3) via A

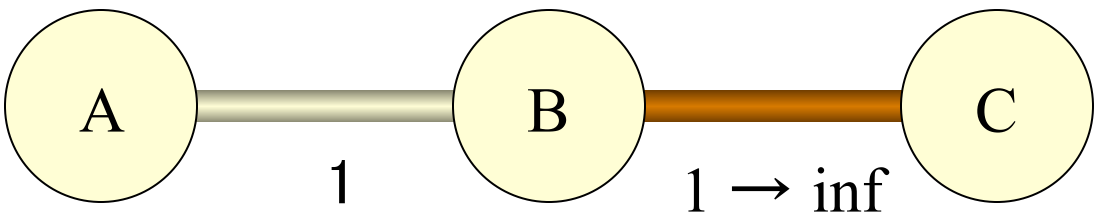
B says: (C, 3)

A thinks: (C, 4) via B

…

Split horizon

- 無限カウント問題に対処するための手法
- 学習した経路の送出元に対しては、その学習した経路の内容を通知しない



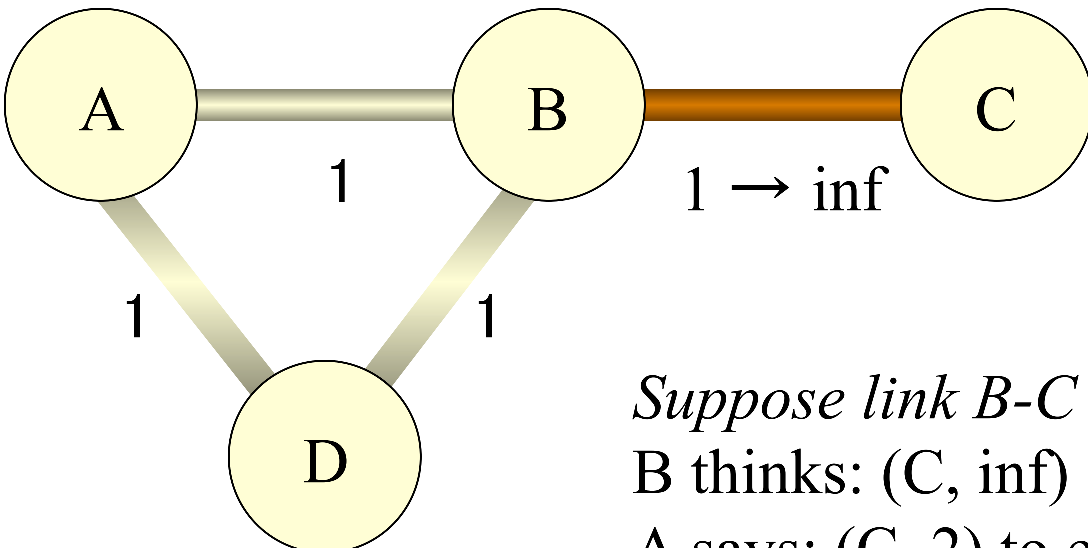
Suppose link B-C went down.

B thinks: (C, inf)

A says: (C, 2) to everyone except B

…

Limitation of split horizon



Suppose link B-C went down.

B thinks: (C, inf)

A says: (C, 2) to everyone except B

D thinks: (C, 3) via A

D says: (C, 3)

B thinks: (C, 4) via D

…

RIP: Routing Information Protocol

RFC 2453, RFC 2080

- 距離ベクトル型の経路制御プロトコル
- RIP−2 (IPv4), RIPng (IPv6)
- 比較的小規模なネットワークで使われる
  - 実装、運用が容易

Questions?

リンク状態型経路制御
Link-state routing

リンク状態型経路制御

- ルータ、リンクの情報(リンク状態情報)を収集
  - ・ ルータをnodeとし、リンクをarcとする有向グラフ
- リンク状態データベース (LSDB) を作成
  - ・ リンク状態情報をあつめたネットワークの地図

- LSDBをもとに、
  Dijkstra algorithmを用いて最短経路を計算
  - 計算量 O(N^2)
    - ・ データ構造を工夫することでさらに改善可能

Dijkstra algorithm

- *Initially,*
  - $d_s \leftarrow 0$
  - $d_j \leftarrow c_{sj}$ (for every $j \in V - \{s\}$)
  - $P \leftarrow \{s\}$
- *Find the next closest node:*
  - $d_i \leftarrow \min_{j \in V-P} \{d_j\}$
  - $P \leftarrow P \cup \{i\}$
- *Update labels:*
  - $d_k \leftarrow \min_{k \in V-P} \{d_k, d_i + c_{ik}\}$
- *Terminal condition:*
  - $P = V$
- ここで V: 全ルータ, s: 起点ルータ, c_{ik}: i, kのリンクコスト、d_j: sからjへの最小コスト、P: 最小コスト確定済みルータ

リンク状態型の利点と欠点

- 時間計算量:　距離ベクトル型に比べ少ない
- トラフィック:リンク数＋ルータ数に比例して増大
- 領域計算量:　リンク数＋ルータ数に比例して増大

- 収束時間:　　短くなければならない
- 無限カウント問題は起きない
- リンクコストの柔軟な設定が可能

OSPF: Open Shortest Path First

RFC 2328, 5340

- リンク状態型の経路制御プロトコル
- OSPFv2 (IPv4), OSPFv3 (IPv6)

- 機能
  - 近接ルータの認識
  - リンク状態情報の交換とLSDBの作成
  - 最短経路木の計算
  - + 代表ルータ、バックアップ代表ルータ
  - + エリアによる階層化
  - + EGPとの連携

OSPF Hello

- 同一リンク上の、近接ルータの発見
  - 224.0.0.5, ff02::5 (AllSPFRouters) へHelloパケット送信
- Helloパケット中に近接ルータのリストを明記
  - 互いに双方向通信可能であることを確認
- (代表ルータ、バックアップ代表ルータの選出)
- 定期的に送信し、リンクの切断を検出

隣接関係の形成

- 近接(neighbor) → 隣接(adjacent)
- 隣接関係にないと経路情報を交換しない

- 2台のルータが隣接関係になるまで:
  - Hello → 近接
  - 互いのLSDBを同期
  - 同期完了 → 隣接関係

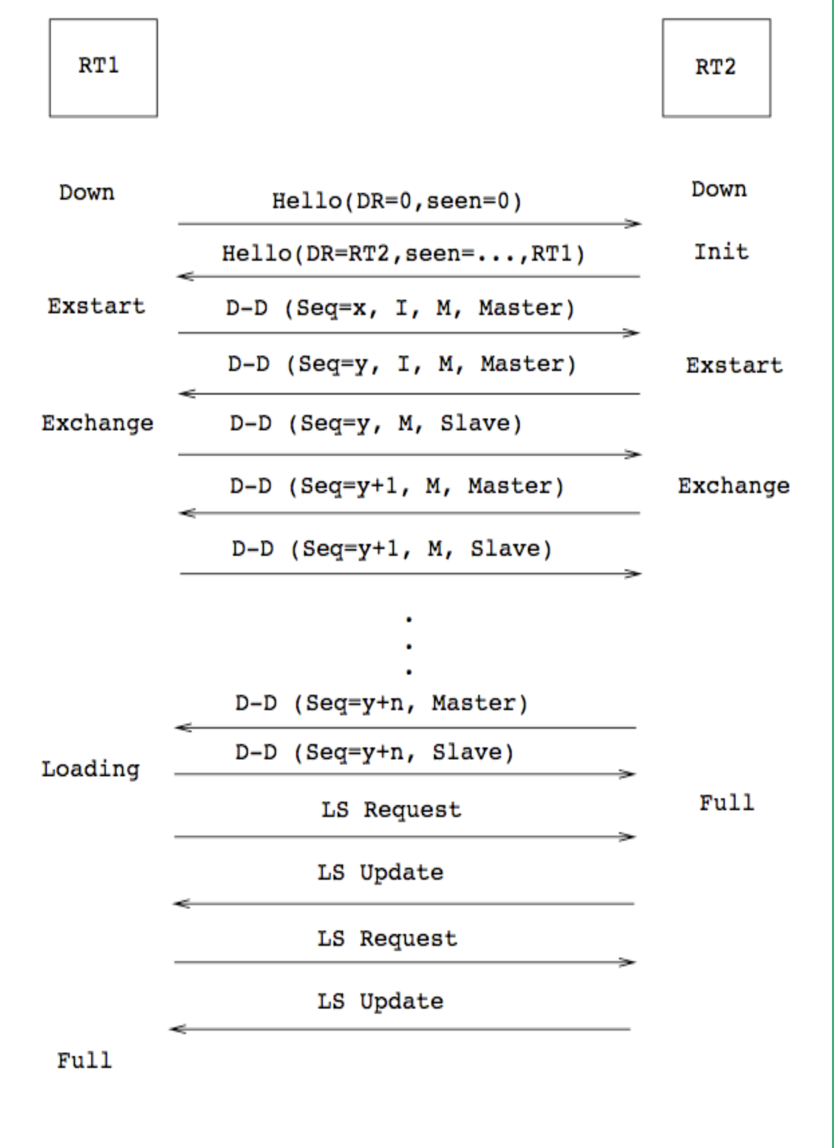


Figure 14: An adjacency bring-up example

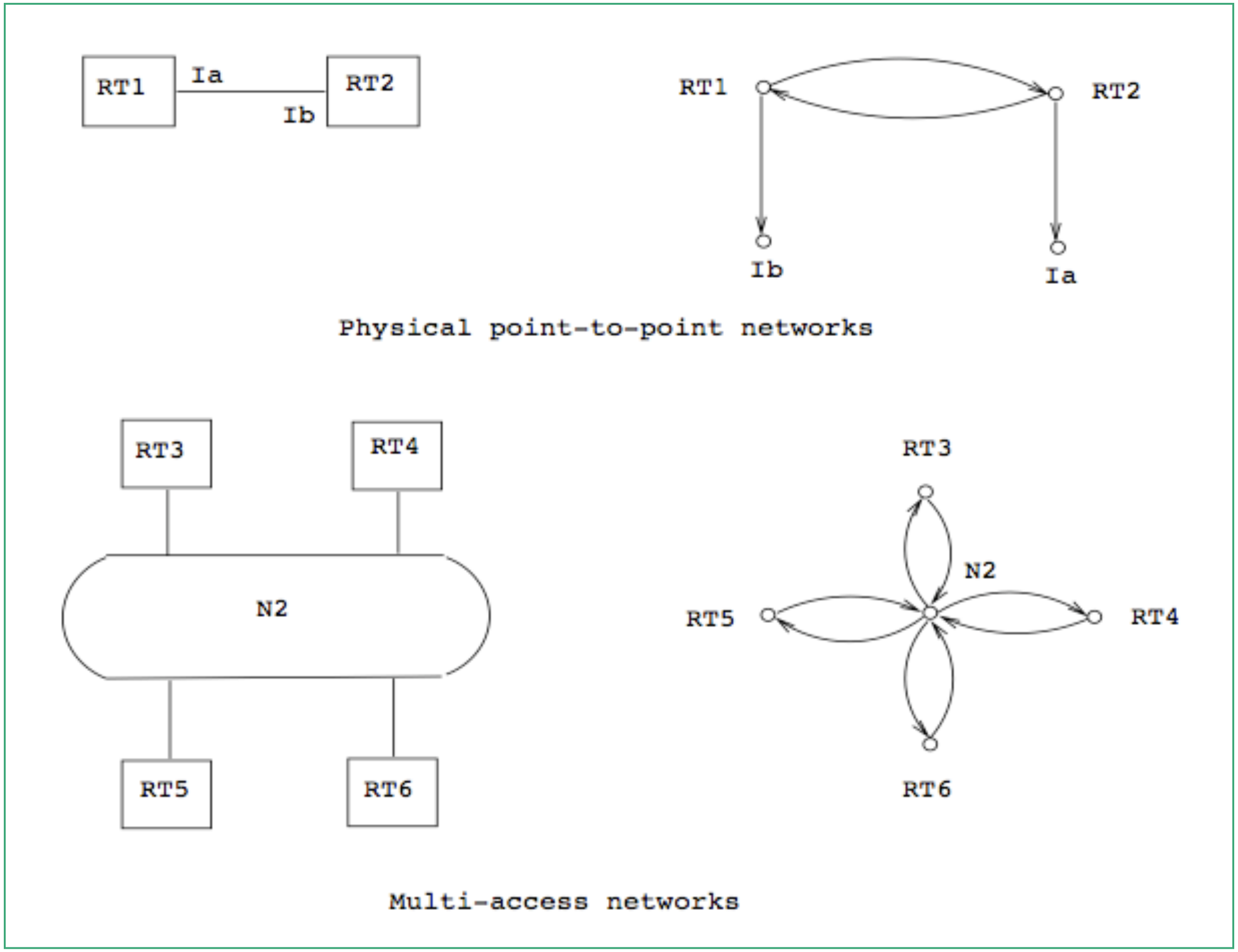
Questions?

OSPFにおけるトポロジの表現

- LSA (Link State Advertisement)
  - Type 1: Router LSA
  - Type 2: Network LSA
  - Type 3: Summary LSA (network)
  - Type 4: Summary LSA (AS boundary)
  - Type 5: AS External LSA

- LSA共通ヘッダ
  - 存続期間、シーケンス番号が設けられている
  - LSAの新旧を判断することが可能

OSPFにおけるトポロジの表現(2)

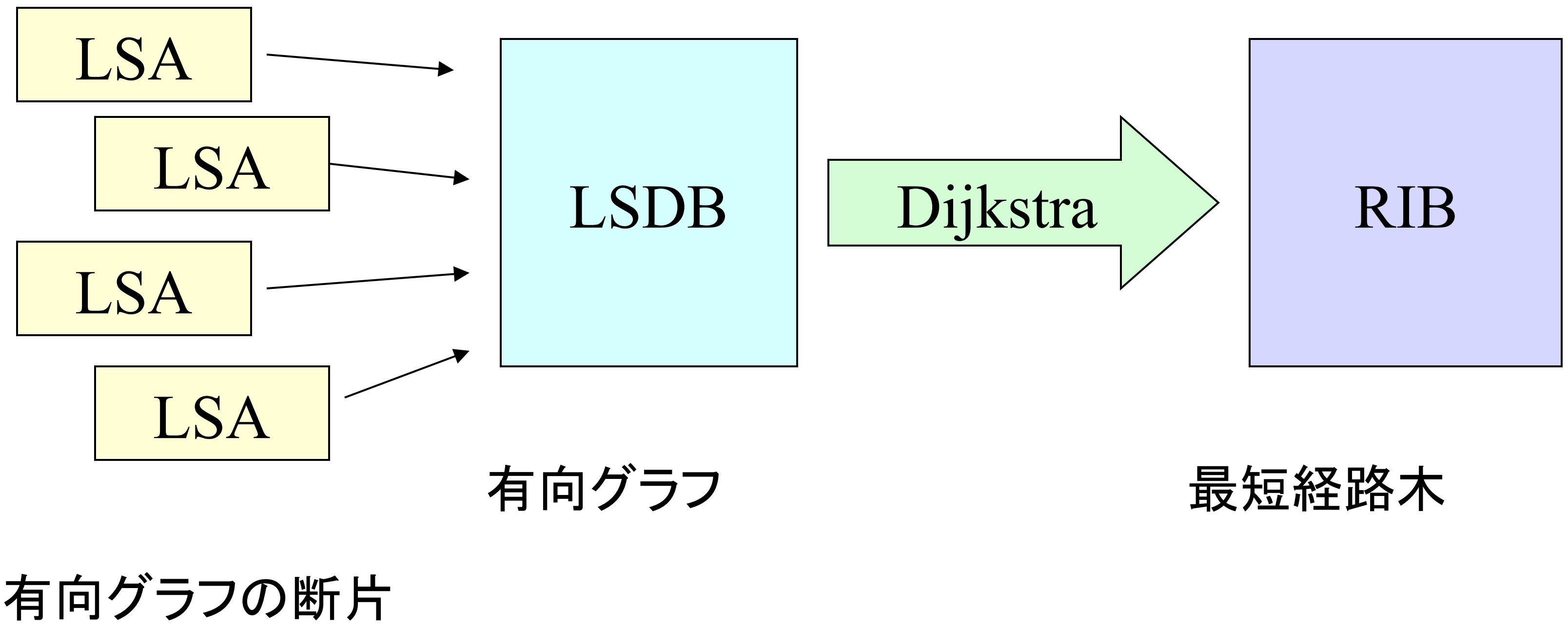


リンク状態の通知 (Link State Update)

- Flooding
  - 更新されたLSAを受け取ったとき、
    受け取ったインターフェース以外に送出
- LSDBを更新、最短経路木の再計算

Questions?

LSAからRIBへ



LSAからRIBへ(2)

- 以下３枚のスライド:
  - OSPFネットワークの例

  - 導出された有向グラフ
    - 個々のLSAがあらわす部分グラフ
  - RT6を頂点とする最短経路木
    - Dijkstra’s algorithmを用いて生成

OSPFネットワークの例

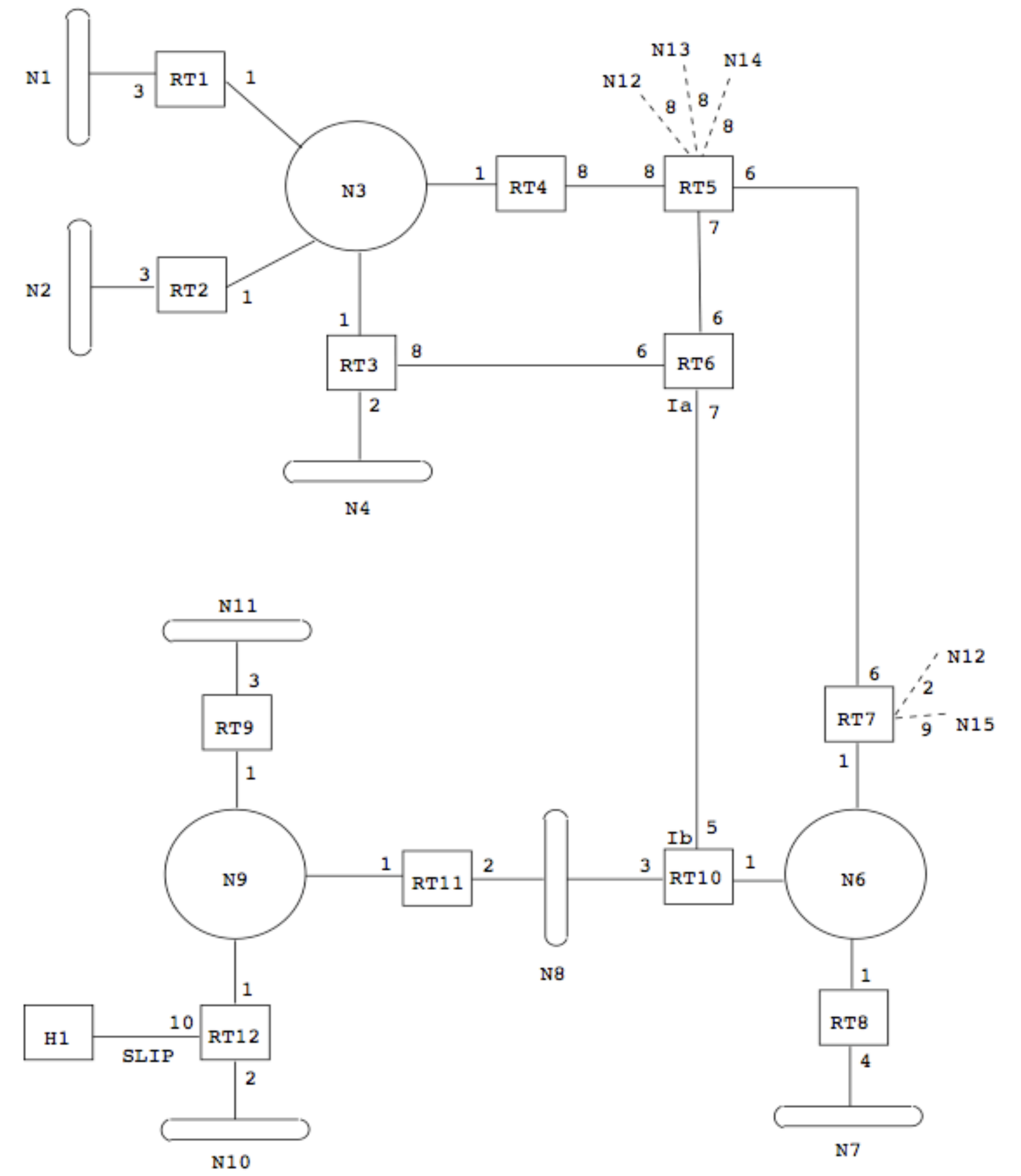


Figure 2: A sample Autonomous System

導出された有向グラフ

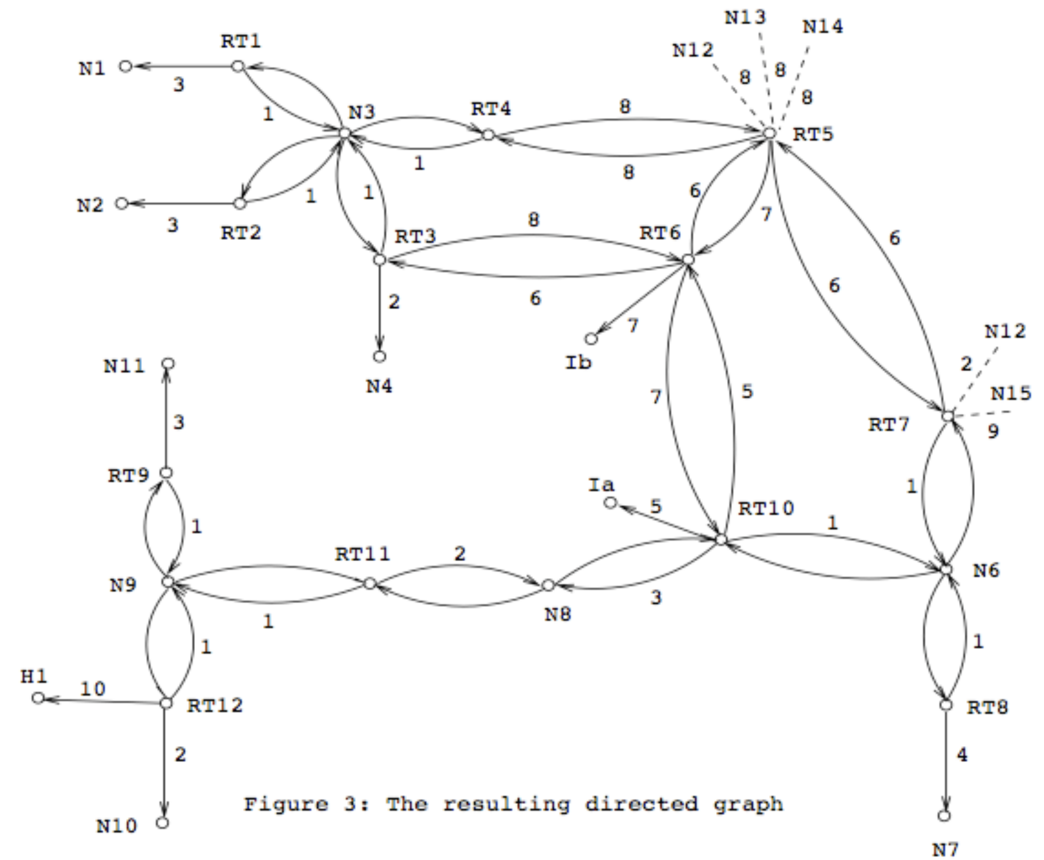


Figure 3: The resulting directed graph

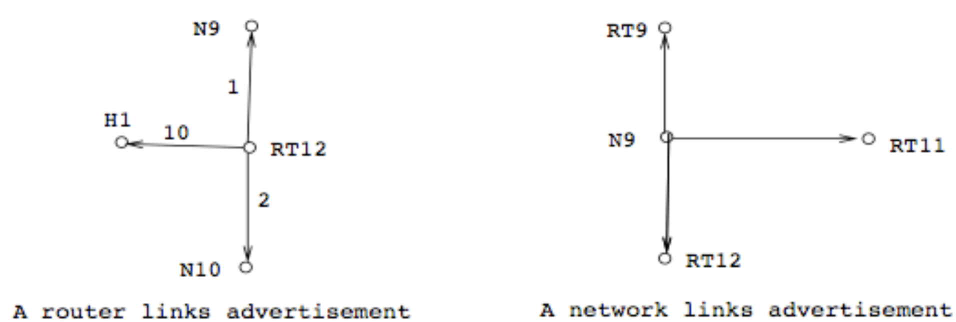


Figure 4: Individual link state components

RT6を頂点とする最短経路木

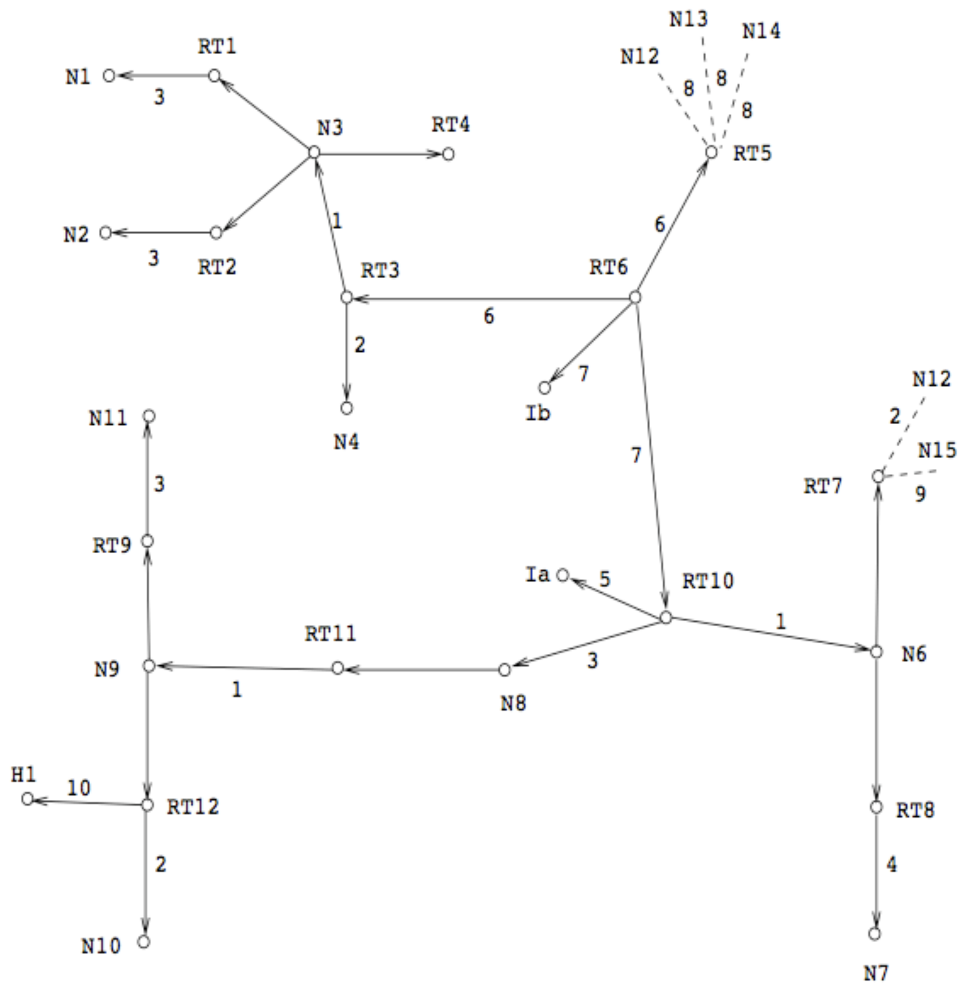


Figure 5: The SPF tree for router RT6

Destination	*Next Hop*	*Distance*
N1	RT3	10
N2	RT3	10
N3	RT3	7
N4	RT3	8
Ib	*	7
Ia	RT10	12
N6	RT10	8
N7	RT10	12
N8	RT10	10
N9	RT10	11
N10	RT10	13
N11	RT10	14
H1	RT10	21
RT5	RT5	6
RT7	RT10	8

Table 2: The portion of Router RT6's routing table listing local destinations.

OSPF area

- 分割統治法による解決
  - 計算量
  - For intra-area link changes, the Dijkstra's Algorithm used in OSPF has a time complexity of O(s*logs), where s is the size of the Area.
    (X. Xiao et al., “Reducing Routing Table Computation Cost in OSPF”, INET’99)
  - 通信量　(障害の隠蔽)

- 以下２枚のスライド
  - OSPF area の構成例
  - 分割された有向グラフ

OSPF area の構成例

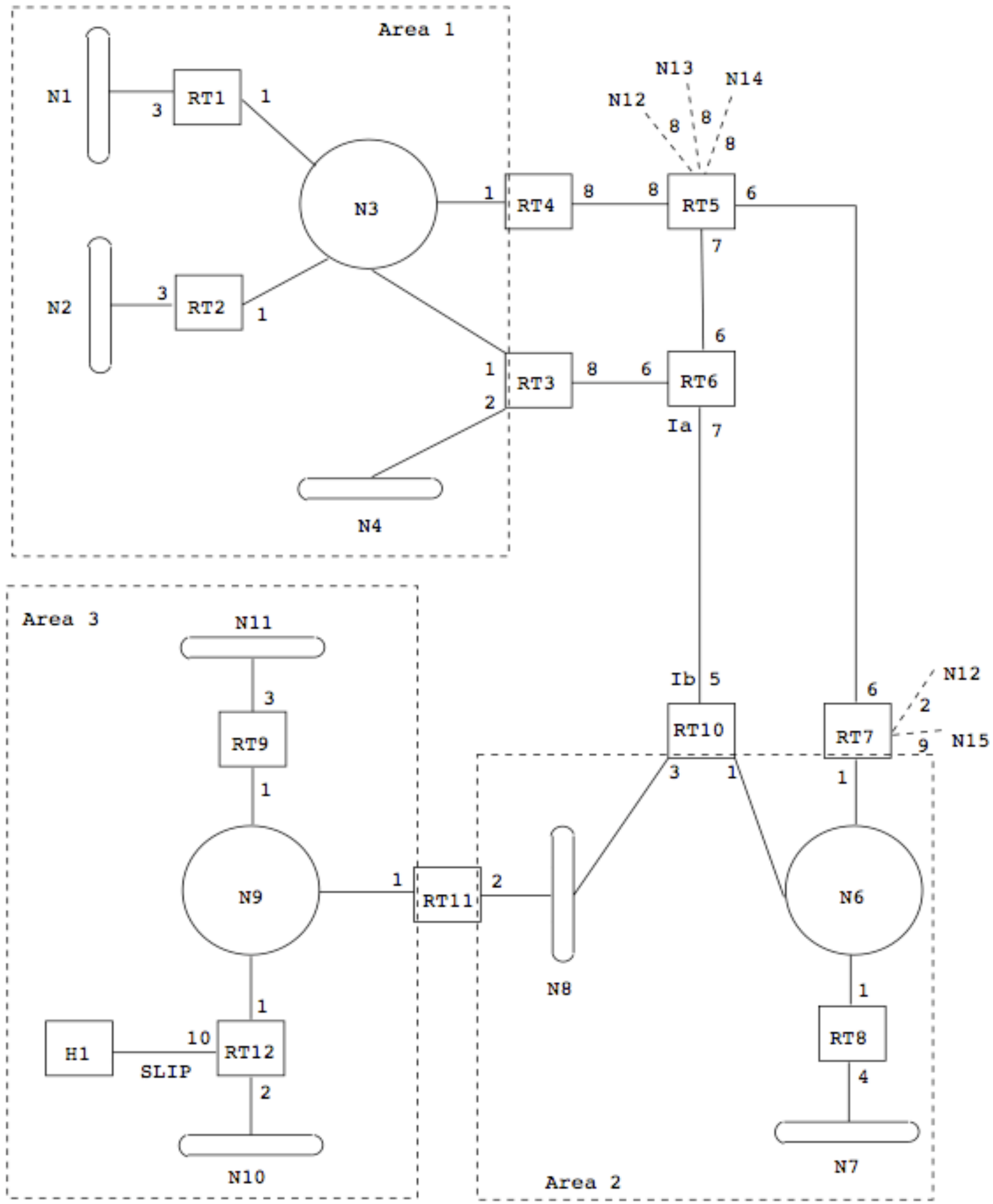


Figure 6: A sample OSPF area configuration

分割された有向グラフ

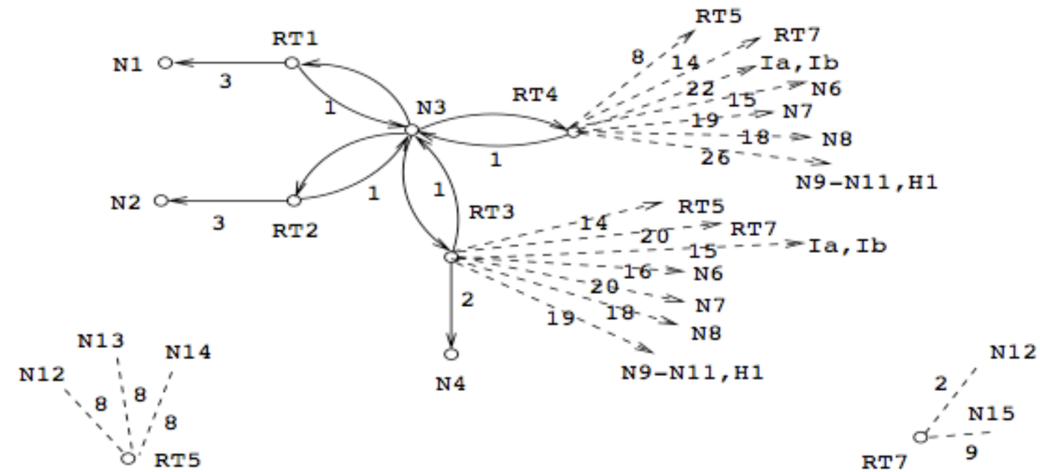


Figure 7: Area 1's Database

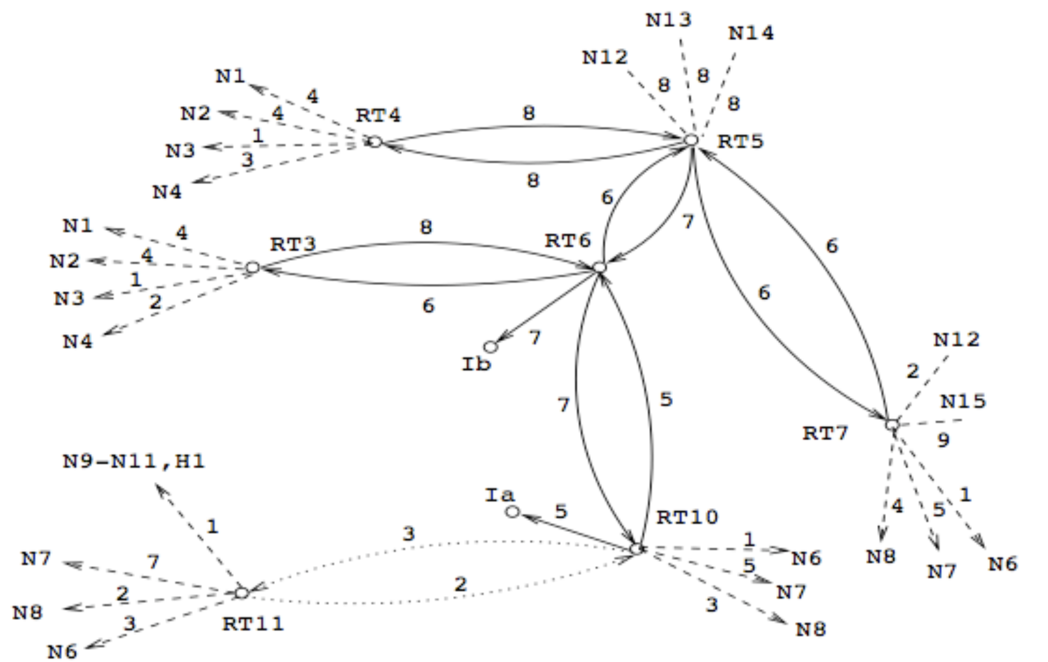
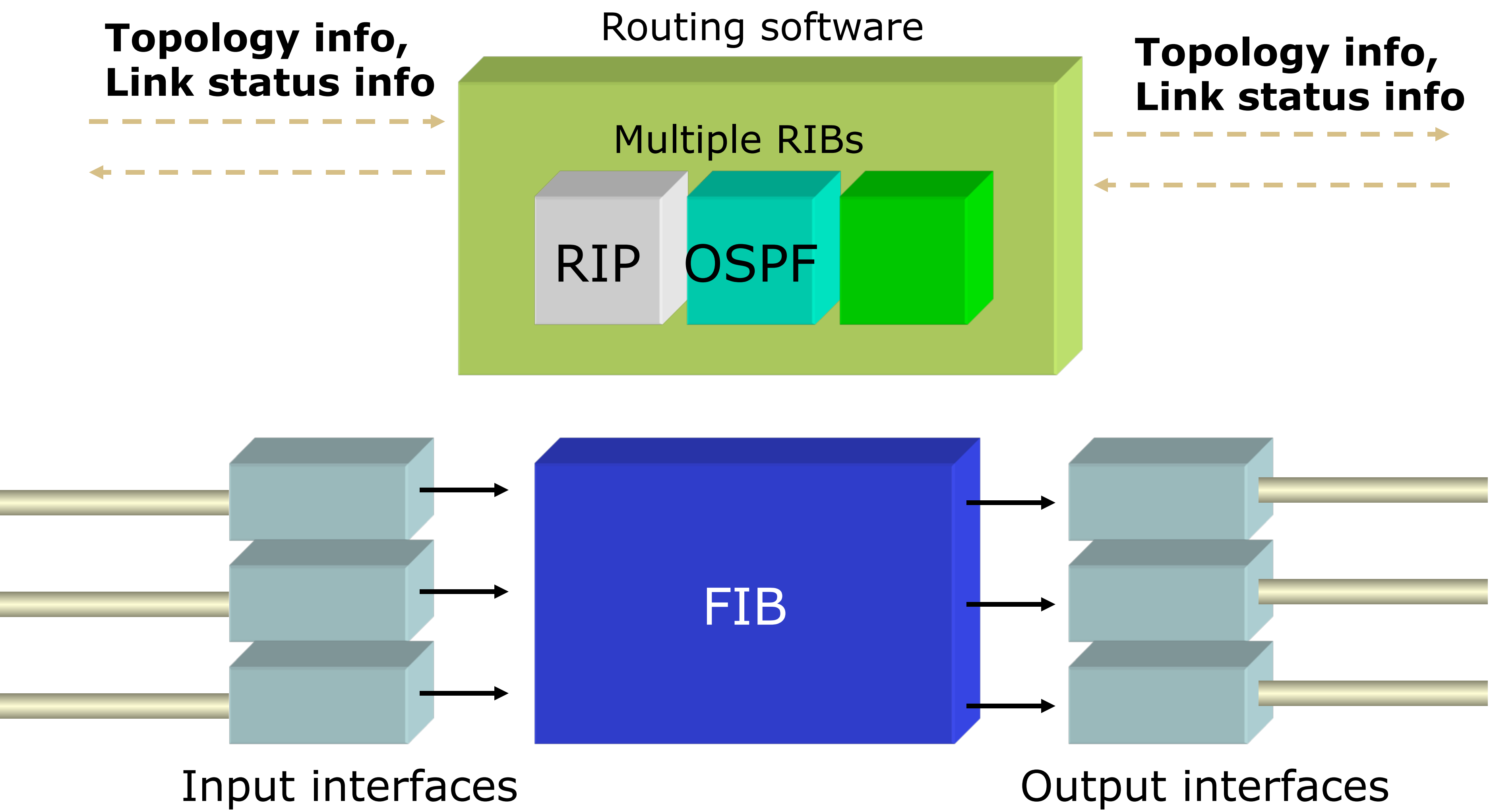


Figure 8: The backbone database

Gateway Model Revisited



Questions?

まとめ

- 経路制御システム: 問題領域と特徴
- 距離ベクトル型経路制御
  - Bellman-Fordアルゴリズム
  - RIPプロトコル
- リンク状態型経路制御
  - Dijkstraアルゴリズム
  - OSPFプロトコル
- 次回
  - IGPとEGP、階層型経路制御
  - パスベクトル型経路制御
  - 経路制御ポリシの実現