OSPFの理論と実践

リンクステート型経路制御の原理・原則

2024/04/24

小原泰弘

© 2024 APRESIA Systems, Ltd. All Rights Reserved.

目次

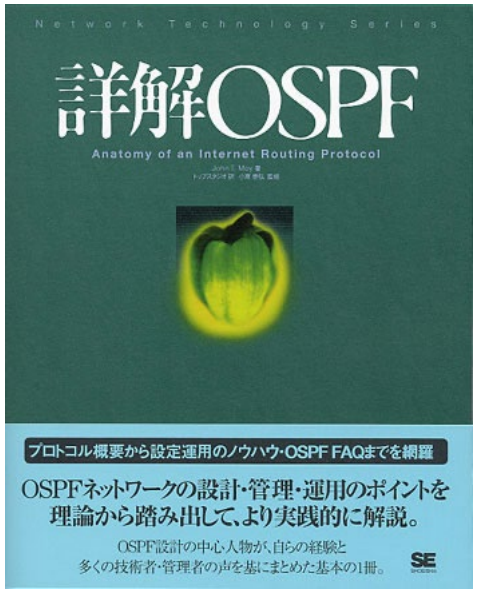
講義概要

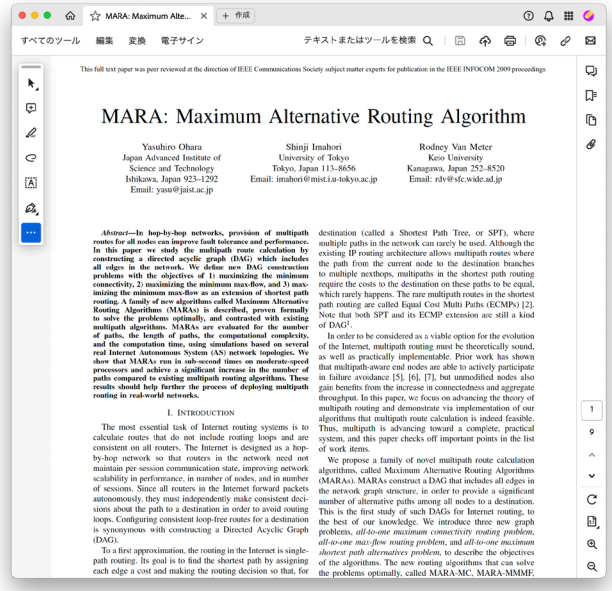
- 対象者
  - OSPF超初心者・文系向け
  - OSPF中級者向け（デモ等）
- 目的
  - OSPF・IS-ISの動作原理を理解する
  - 仕様がスムーズに読めるようになる
  - **仕様を読まなくても済む**（RFC2328: 244p、RFC5340: 94p）
  - 開発・運用に役立つ（LSDBが読めるようになる）
  - 漫画（ポンチ絵）で分かる…「漫画で分かるOSPF」
- 内容
  - OSPFの概念：LSDB、Network-LSA
  - 最短経路計算の理論：最適性の原理、単調性・等張性
  - ダイクストラ計算の概要、計算量、ヒープソートによるプライオリティキューの候補リスト、スケーラビリティ
  - ダイクストラ計算参照コードの説明、デモ
  - OSPF運用の実際
  - OSPF v.s. IS-IS

講師紹介

- 詳解OSPF　監訳者
- GNU Zebra ospf6d 作者
- ダイクストラ計算拡張してシミュレーションして論文出版した経験あり（IEEE INFOCOM '09）

- 博士（政策・メディア）慶應SFC
- UCSC　ポスドク（ストレージ研究）




MARA: Maximum Alternative Routing Algorithm

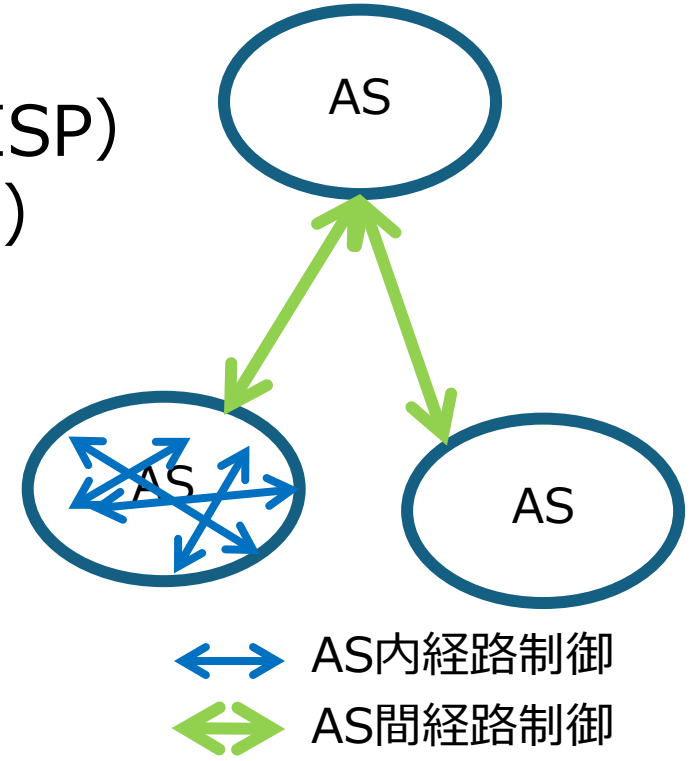
Yasuhiro Ohara
Japan Advanced Institute of Science and Technology
Ishikawa, Japan 923-1292
Email: yasu@jaist.ac.jp

Shinji Imahori
University of Tokyo
Tokyo, Japan 113-8656
Email: imahori@mist.i.u-tokyo.ac.jp

Rodney Van Meter
Keio University
Kanagawa, Japan 252-8520
Email: rdv@sfc.wide.ad.jp

OSPF概要

- OSPF：Open Shortest Path First
  「（仕様が）公開されている SPF 計算するやつ」の意味
- L3経路制御：オンリンクは繋がる。１ホップ以遠を計算する。
- リンクステート型経路制御プロトコル
- AS：自律システム（Autonomous System）（≒ISP）
- IGP：AS内経路制御（Interior Gateway Protocol）
- OSPFv2 (IPv4) / OSPFv3 (IPv6)
- 近隣探索（Neighbor Discovery）
  - Hello サブプロトコル
- LSDB：リンクステートデータベース
- SPF計算：最短経路計算、ダイクストラ計算



経路制御プロトコルの分類

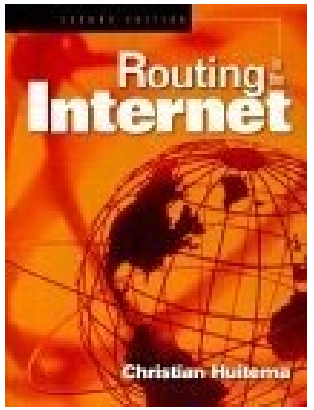
- AS間経路制御プロトコル：EGP（Exterior Gateway Protocol）
  - BGP（Border Gateway Protocol）
- AS内経路制御プロトコル：IGP（Interior Gateway Protocol）
  - OSPF、IS-IS、RIP(ng)、(E)IGRP、、、（MANET系）

- リンクステート型経路制御プロトコル
  - OSPF、IS-IS
- ディスタンスベクタ型経路制御プロトコル
  - RIP(ng)、(E)IGRP
- パスベクタ型経路制御プロトコル
  - BGP

- OSPF：リンクステート型IGP
- BGP：パスベクタ型EGP

歴史的背景

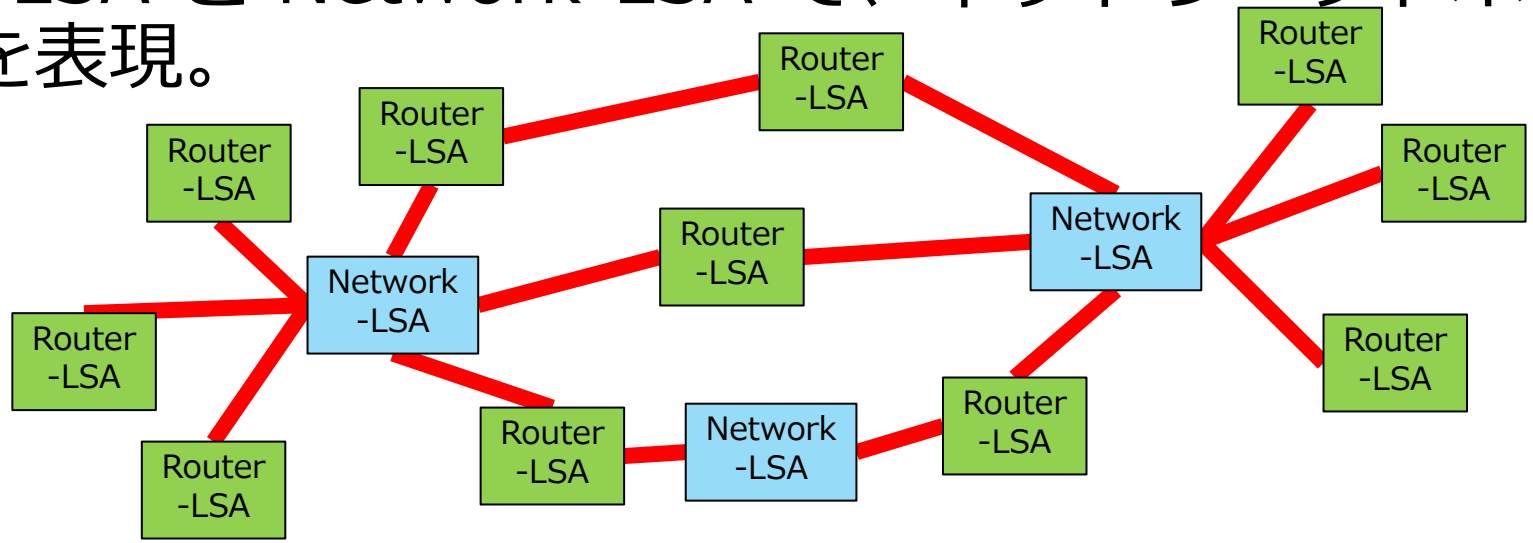
- ディスタンスベクタ型は伝言ゲーム方式。フィルタ簡単
- リンクステート型は「カード共有・各自で計算」方式。フィルタ困難
- 歴史：
  - 簡単なのでディスタンスベクタ型で始めた（RIP）
  - タイミング問題で経路ループする問題で苦しんだ
    - バウンシングエフェクト、スプリットホライズン、カウンティングトゥインフィニティ
  - 方向と数値コストだけで計算するからループする
    - →リンクステート：リンクの状態を共有して計算する
    - →パスベクタ：経由ノードリスト（パス）で計算する
- OSPF：
  - 面で自動計算、楽だが細かい修正効かない。最適コスト計算はNP完全
  - 何でもいいから経路計算して、というときに使う
- BGP：
  - 経路ごとにパスの詳細をポリシ含めて計算。
  - 安全のために自動は無し。ほとんど手動で詳細に通るパスを確認する



Routing in the Internet、
Prentice Hall、2000

LSAとLSDB

- Link State Advertisement（LSA）
- Link State DataBase（LSDB）
- LS type, Link State ID, Advertising Router で識別されるカード。
- Router-LSA と Network-LSA で、ネットワークトポロジ（グラフ）を表現。



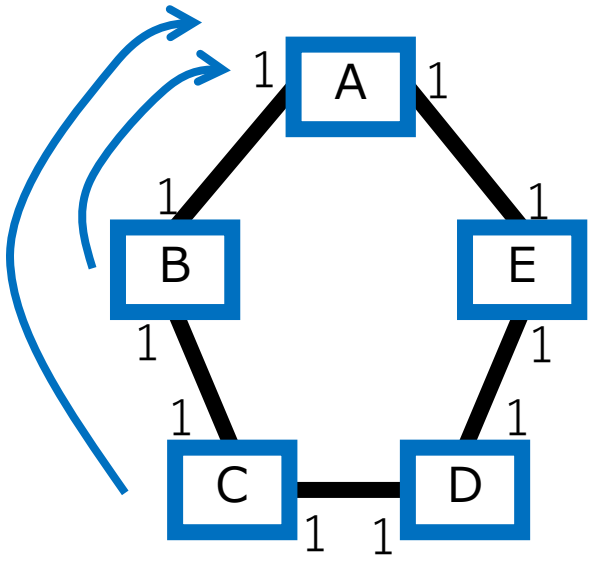
LSDBの同期と経路の整合性（無矛盾）

- 同じ入力に、同じアルゴリズム（手順）をかけるので、結果が同じになる。
- 同じLSDBに、同じダイクストラ計算をかけるので、経路に矛盾が無い。
- 同じLSDBじゃないと経路ループの危険性。→ LSDBの同期が大事。
- 同じダイクストラ計算じゃないと経路ループの危険性。→ フィルタができない。
- OSPF 隣接関係（adjacency）で LSDB 同期する（Fullになる）。
- LSDB 同期したルータとしか経路の整合性が保証できない。
- Full になった隣接関係しか LSA に載せない。
- 計算途中（transient state）で経路ループするのは仕方がない。

経路ループの仕組み

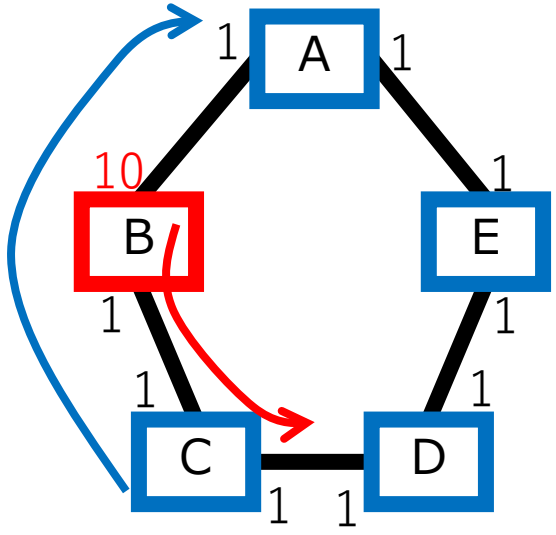
- I/Fコストは出力時（outgoing）にかかる

平常時

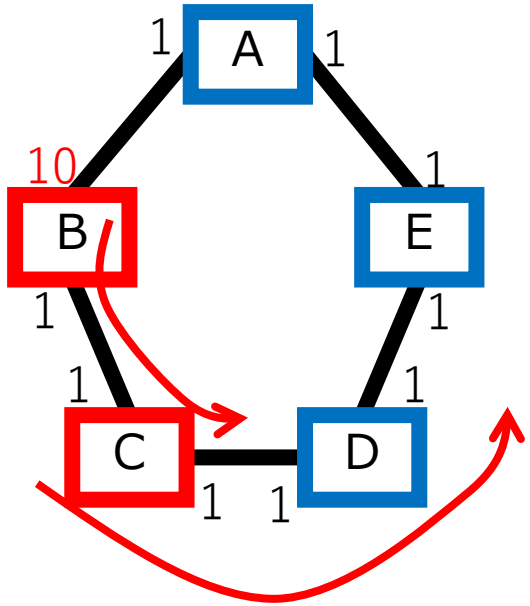


B、Cにおける
Aへの経路

コスト変更直後：
Bのみ計算完了
経路ループ発生

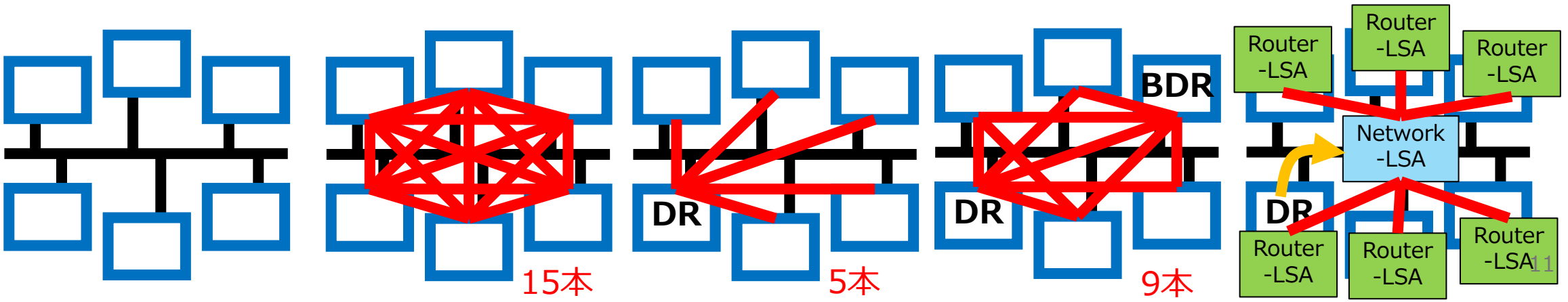


コスト変更直後：
Cも計算完了
経路ループ解消

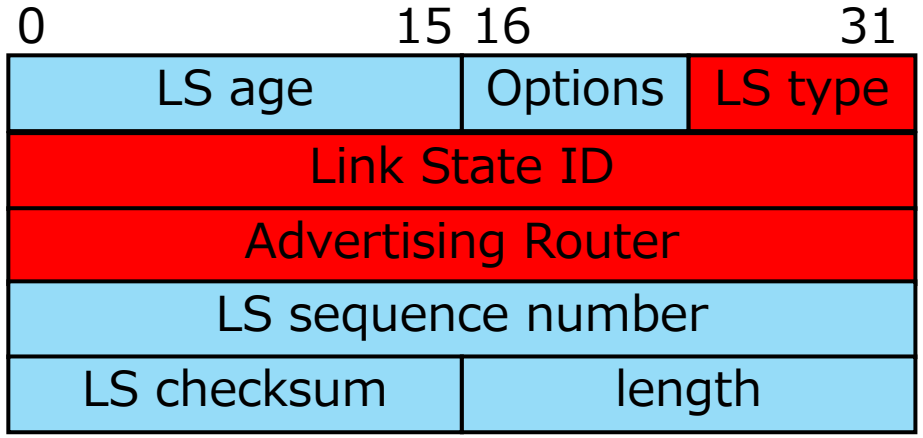


Network-LSA、代表ルータ

- イーサセグメントの代表ルータ、LSA
  - Ethernet（イーサネット）がものすごく流行った。ブロードキャストセグメントに分類される。
  - イーサセグメントには数十、数百のルータが繋がってもおかしくない。
  - この時、LSDB同期をルータ組み合わせ同士で実施するのは非効率。O(N^2)（ex.イーサセグメント上のルータが30台であれば、 30 x 29 / 2 := 450 本のneighbor LSDB同期）
  - イーサセグメントには代表ルータ（Designated Router: DR）を置き、イーサセグメントを代表（代理）させる。他のルータは代表ルータとだけLSDB同期する。代表ルータと同期していれば、間接的に他ルータとも同期している。
  - 代表ルータがダウンした時のために、バックアップ代表ルータ（Backup DR: BDR）も決めておく。DR/BDR両方ダウン（二重故障）しない限り、このイーサセグメント上のLSDB同期を初めからやり直さなくても良い。
- DR/BDRは、イーサセグメントごと。OSPFインターフェイスの状態（ステート）。



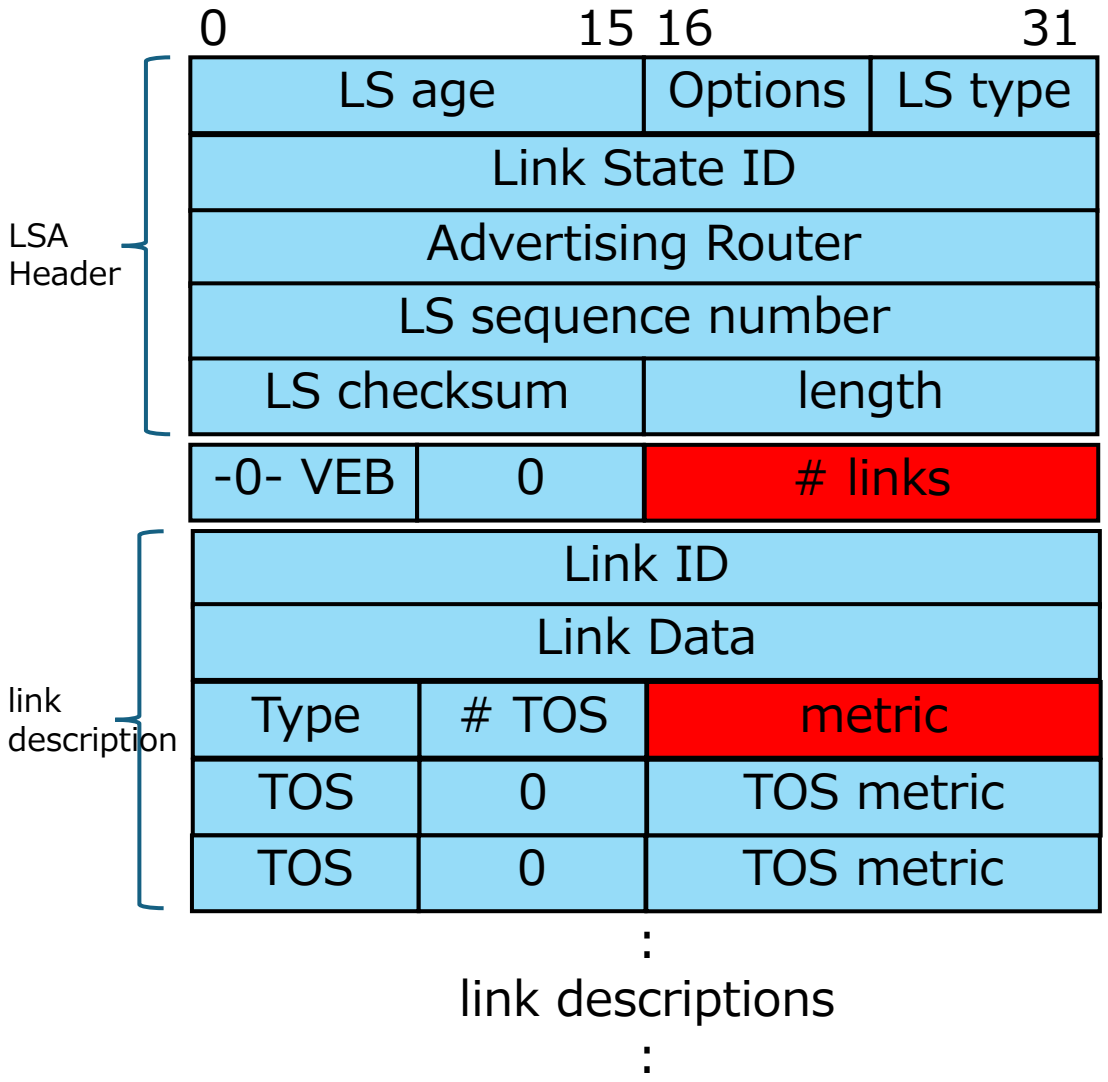
LSA共通部分: LSA ヘッダ



- LS age:
  - LSAの生存時間。古くなったらリフレッシュ（再作成）する。
- Options:
  - オプションフラグ。
- length:
  - LSAの長さ（バイト）。

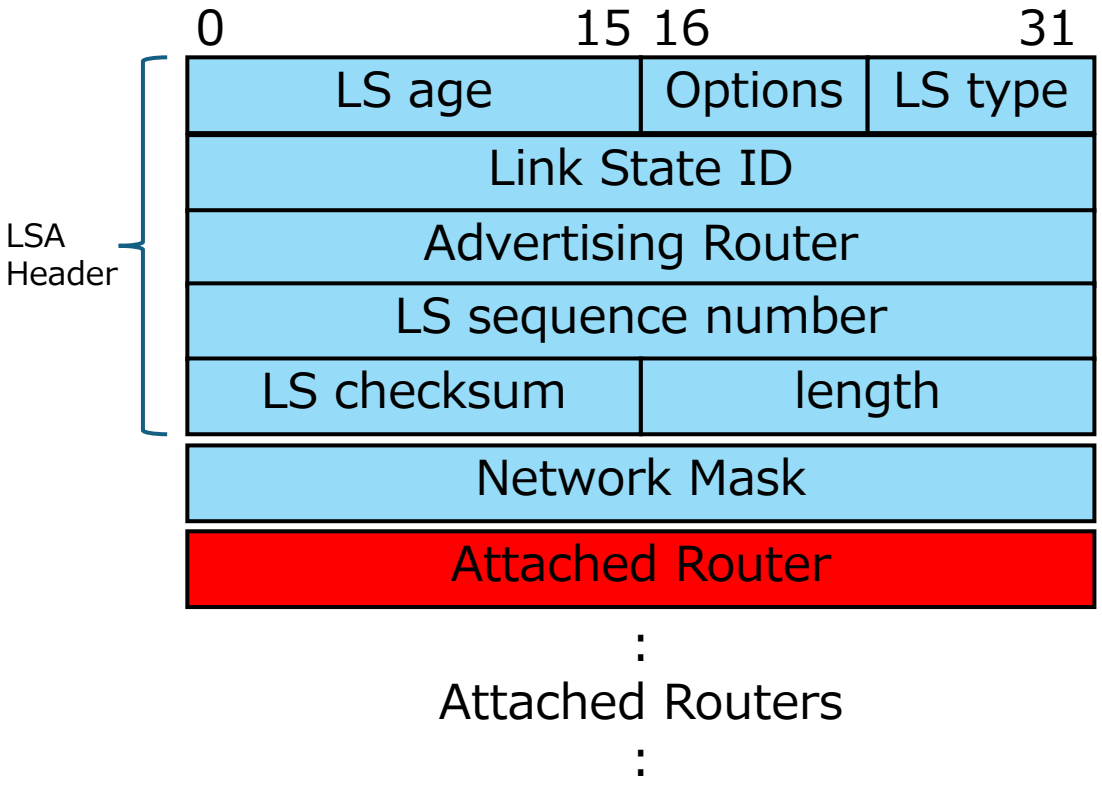
- LS type:
  - LSAのタイプ。Router-LSA(1)やNetwork-LSA(2)など。
- Link State ID:
  - Router-LSAの場合はRouter-ID。Network-LSAの場合はネットワークIPアドレス。
- Advertising Router:
  - LSA作成元ルータのRouter-ID。
- LS sequence number:
  - LSAのインスタンスを表す。リフレッシュ（再作成）されると増える16進数。
- LS checksum:
  - チェックサム。内容が壊れてないかチェックできる。

Router-LSA



- # links:
  - link descriptionの数
- link description:
  - Type: name: Link-ID
    - 1: P-to-P: nei's router-ID
    - 2: transit: Network-LSA's Link state ID
    - 3: stub: IP network address
    - 4: virtual: nei's router-ID
  - Link Data: network address mask か、MIB-II ifIndex か、IP interface address。
  - #TOS: 後続のTOSフィールドの数。
  - metric: link の TOS-0 メトリック。

Network-LSA



- Network Mask:
  - セグメントのnetwork address mask。
- Attached Router:
  - セグメントに存在するルータのRouter-ID。

SPF計算：ダイクストラ計算（イメージ）

- ネットワークを丸（ルータ）と線（リンク）でつなぐ。
- それを全部外して、バラバラにして袋に入れる。（外した部分はマークつけておいて、後で戻したり確認できるようにしておく）
- 袋から自分自身の丸を一個取り出す。（その丸への経路を確定する）
  1. 取り出した丸Aから、繋がってる線を全部調べる。
  2. 線の先の丸Bを袋から取り出して一個一個調べる。
  3. 丸A経由の丸Bのコストが元々の丸Bに書いてあるコストより低ければ、丸Bのコストを上書き更新する。コストが高ければ、書き換えずに丸Bを袋に戻す。
  4. 袋ん中の丸の中で一番コストが低いものを取り出す。（その丸への経路を確定する）
  5. 袋から丸が無くなれば、計算終了。そうでなければ、1.に戻る（繰り返し）。
- ネットワークが全部繋がってれば、最終的に袋から全部の丸が無くなる（全ての丸への経路が計算できる）

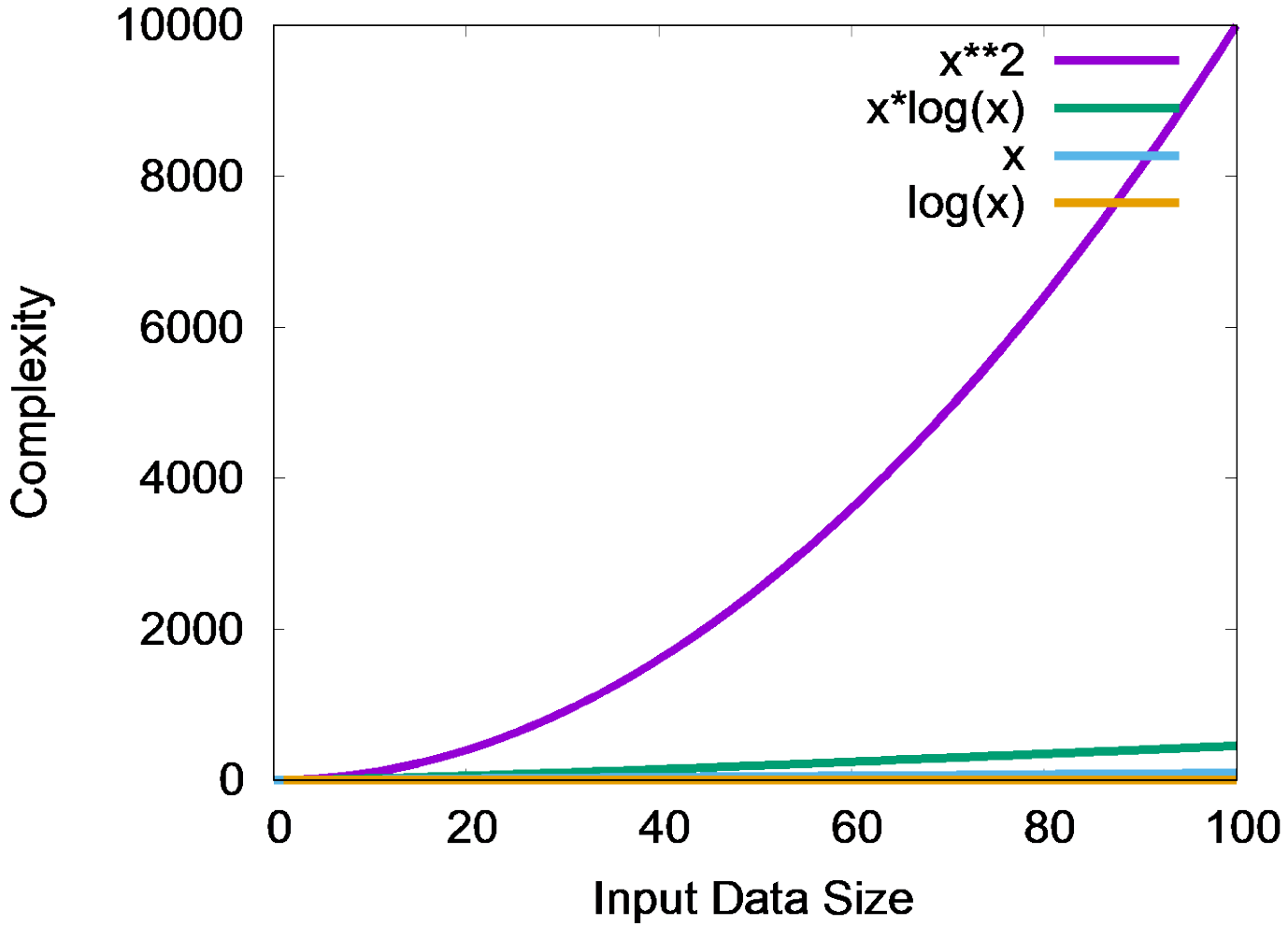
SPF計算：ダイクストラ計算（理論の独自解釈）

- 「現時点で最も短いパスは、その始点終点間の最短パスである。」
  - パスは、リンクを足して伸長していくしかない。今後見つかるパスは、すべて現在の候補パスを伸ばしたものである。それぞれ伸長して、３位が２位を逆転することはあるが、伸長しない１位を逆転することはできない。どのような今後のパスも、現在の１位を超えることはできない。
  - 背理法で簡単に証明可能。「後からもっと短いパスが見つかるか？」
- 最適性の原理：「最短パスの部分パスも最短パスである」
- メトリック（コスト）の単調性（monotonicity）が収束条件、等張性（isotonicity）が最適条件（BGP代数の論文[*]を参照）。
  - monotonicity: パスが伸長された時にメトリックが減少しない。
  - isotonicity: 2つのパスに同じリンクを足してもメトリックの大小関係が変わらない。
  - [*]: "Network routing with path vector protocols: theory and applications", SIGCOMM '03, https://dl.acm.org/doi/10.1145/863955.863963
- コストは正の整数。負のコストはヤバい。（最短パスが定義できない）
- コスト0もちょっとヤバい。（無限にECMPが増える）

計算量

- 疑似コード（手続き）を書き出して、インストラクション数やループ回数などを数える。

アルゴリズムイントロダクション 、近代科学社、2013



SPF計算：ダイクストラ計算（疑似コード）

dijkstra_calc (root ∈ G):

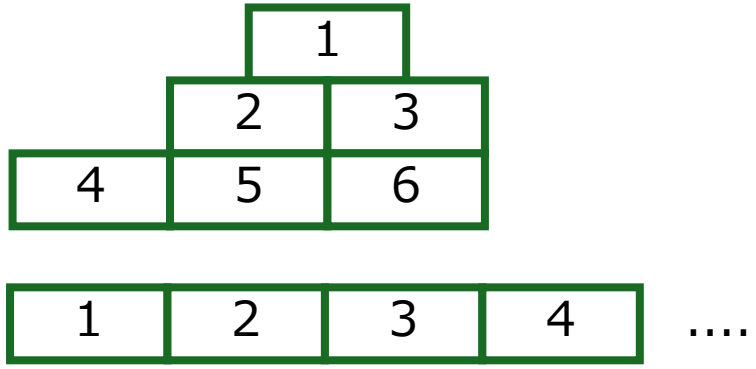
```
candidate_list = pqueue_create ();
pqueue_enqueue (root, candidate_list);

while (candidate_list->size)
 {
   c = pqueue_dequeue (candidate_list);
   spf_tree_install (c);
   /* Call the just added vertex "v" */
   v = c;
   for (link = v->olinks->head; link;
        link = link->next)
    {
      w = link->dst;
```

```
      if (w->cost < v->cost + link->cost)
        continue;

      if (w->cost > v->cost + link->cost)
        w->cost = v->cost + link->cost;
      else if (w->cost == v->cost + link->cost)
        ecmp_add (w, v, link);

      pqueue_enqueue (w, candidate_list);
    }
 }

pqueue_delete (candidate_list);
```

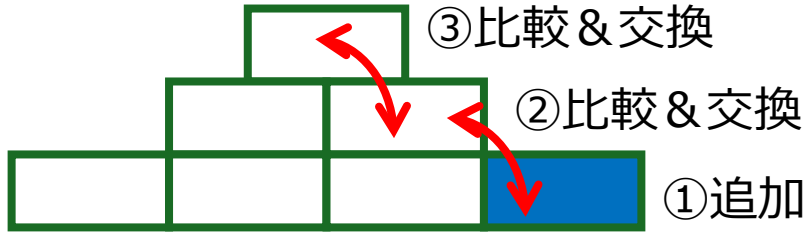
候補リスト実装と計算量

- プライオリティキュー（ヒープソートによる）
- 木の配列表現
- O(log n)（木の段の高さ）

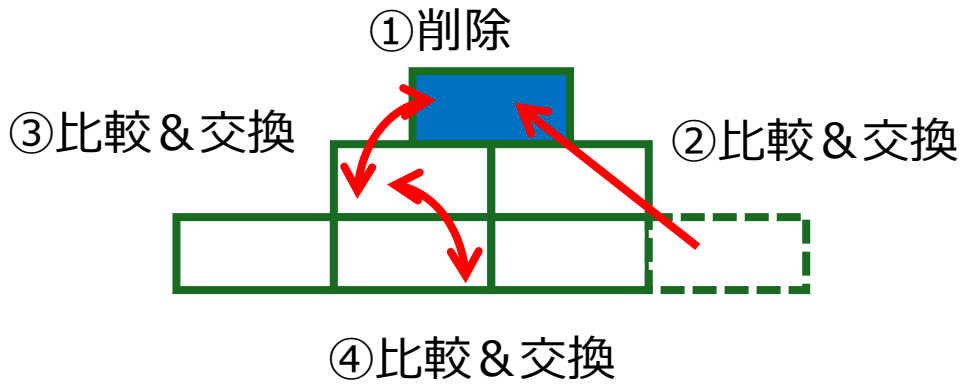
木の配列表現

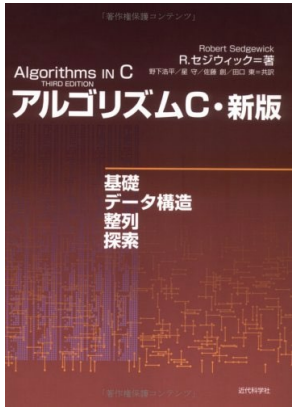


プライオリティキューへの追加



プライオリティキューからの削除と再整列





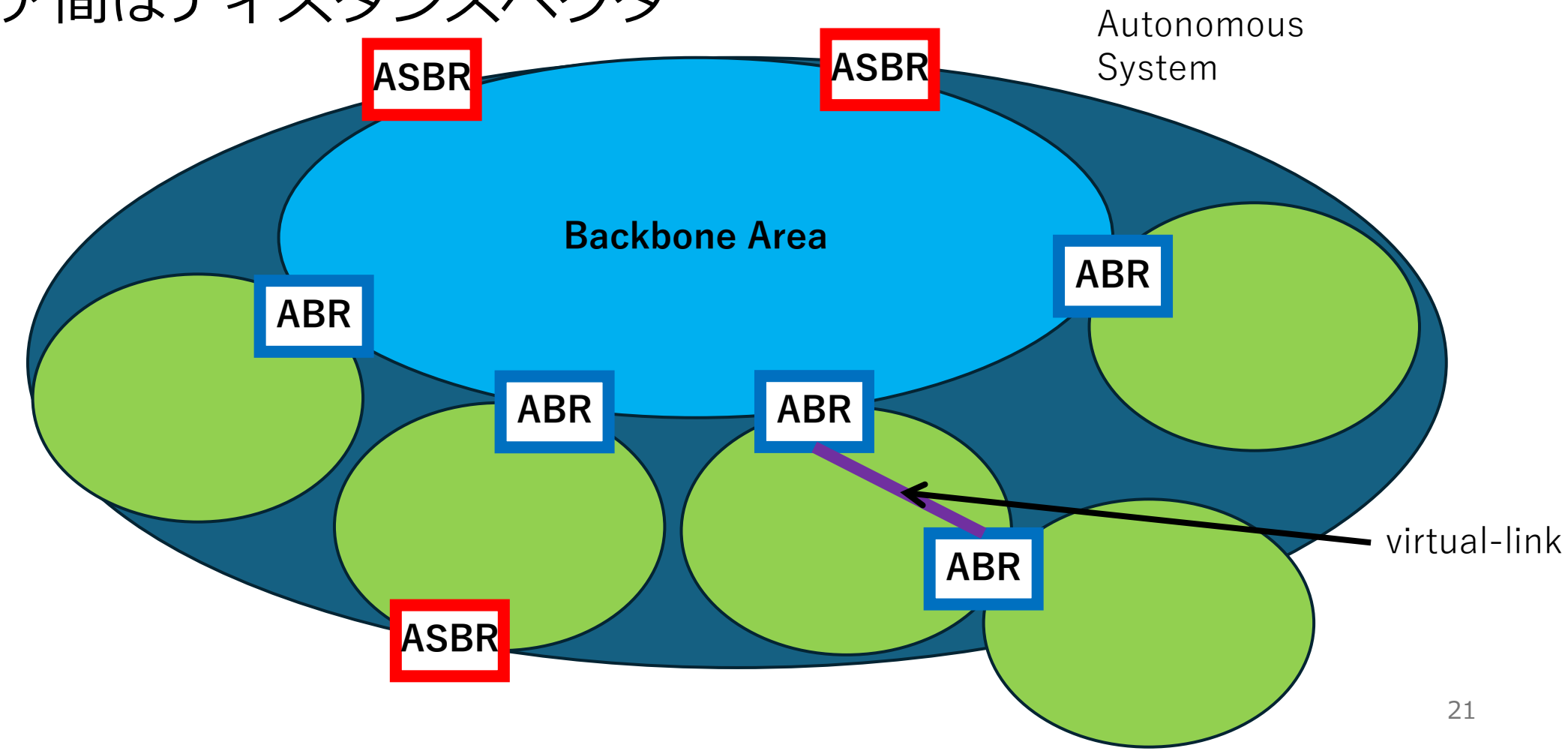
セジウィック：アルゴリズムC、近代科学社、2018

OSPF運用の実際

- TwoWay, Full 以外は全てエラー
- neighborが張れない：HelloInterval、RtrDeadInterval ミスマッチ
- ExChangeで止まってる：MTUミスマッチ
- 経路やLSAがパタパタする（超高速）：Router-ID コンフリクト
- 経路やLSAがパタパタする（中程度）：回線の up/down
- LS Sequence見ていると分かる（普通は30分に 1 進む）
- ドメイン内のルータ数は？：Router-LSAの数と同等
- スケーラビリティ：実装によってルータ50台から危険と言われる（が、実装による）

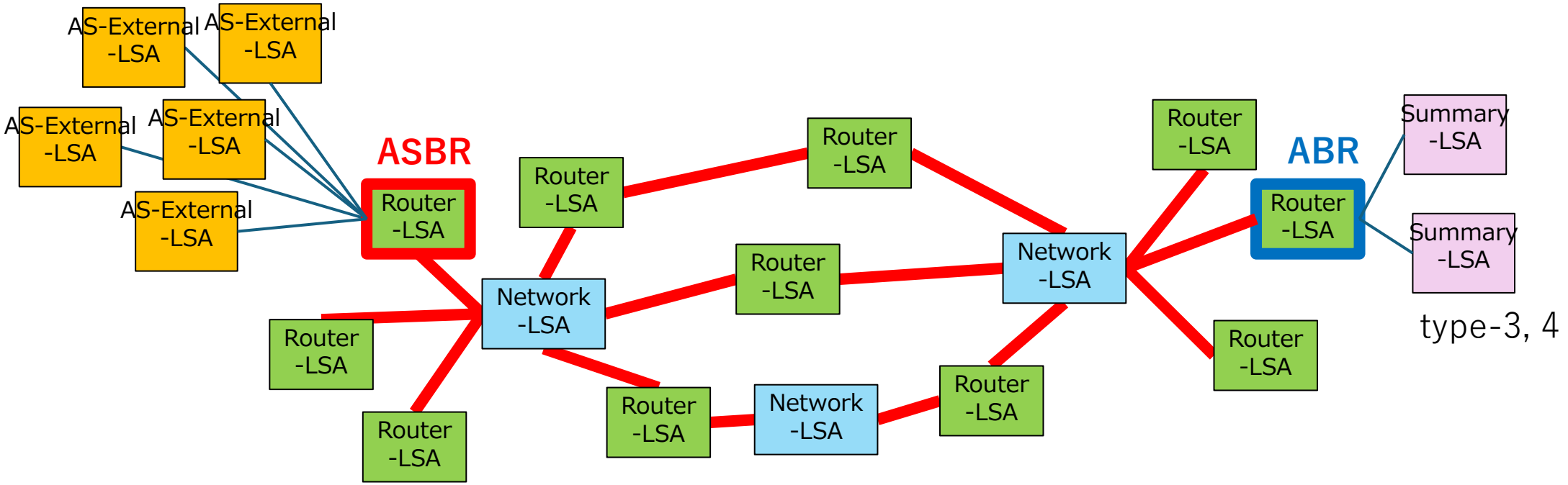
OSPFエリア

- エリア間はディスタンスベクタ



プレフィックス経路、AS間経路、外部経路

- トポロジ計算の後、ラベル的に貼ってあるプレフィックスへの経路を計算
- IPv6も同様



その他

- QoS metric
- スタブエリア
- Virtual-link
- Not-so-stubby-Area: NSSA
- graceful restart
- TE
- BFD
- SRv6

OSPF v.s. IS-IS

- 細かい v.s. 大雑把
- OSPF LSA + LSUpdate = IS-IS LSP
- counting up age (3600) v.s. counting down
- IPv4/v6 別々 v.s. 一緒
- エリアの違い（平面 v.s. 二階建て）（ノードが境界 v.s. リンクが境界）
- 新技術サポート遅い v.s. 早い

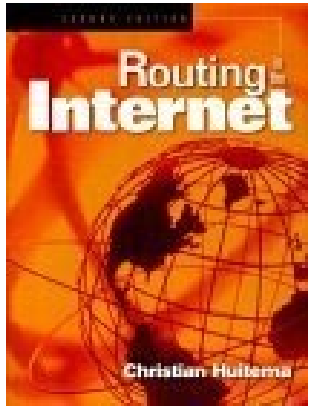
ダイクストラ計算コードデモ

- ダイクストラ計算自体はかなり早い
- 数千ノード、一万リンクとか可能
- OSPFでもIS-ISでも同じ
- 実装による

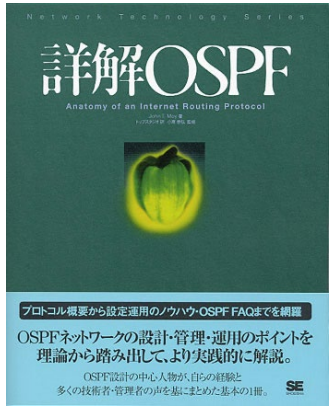
OSPF運用状態・LSDB表示例

- show ip ospf neighbor
- show ip ospf database

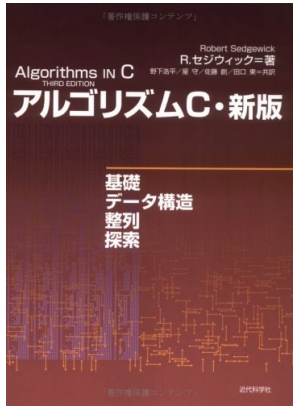
参考文献



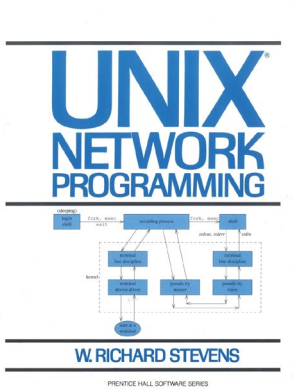
Routing in the Internet、Prentice Hall、2000



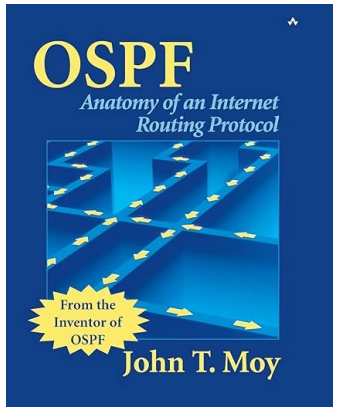
詳解OSPF: Anatomy of an Internet Routing Protocol、翔泳社、2003



セジウィック：アルゴリズムC、近代科学社、2018



UNIX Network Programming、Prentice Hall、1990



OSPF: Anatomy of an Internet Routing Protocol、Addison-Wesley Professional、1998

アルゴリズムイントロダクション 、近代科学社、2013