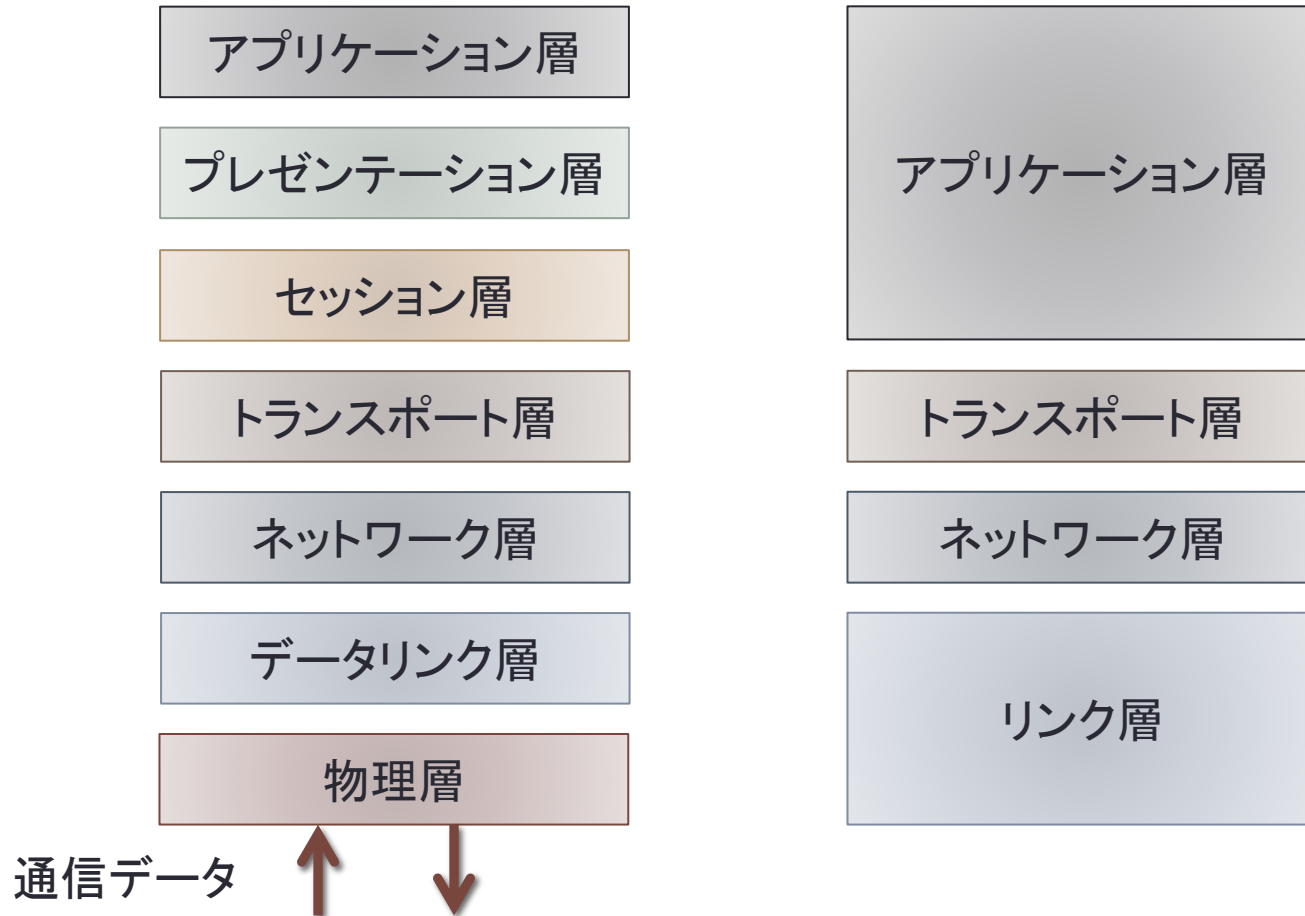


情報通信ネットワーク 第5回

理工学部情報科学科

松澤 智史

本日は……ネットワーク層(続き)



ネットワーク層の役割

- セグメントデータの packets によるカプセル化とアンカプセル化
- 論理アドレスを利用した異なるネットワーク上のノードとの packets 交換
- ルーティングプロトコルを利用した通信経路の決定

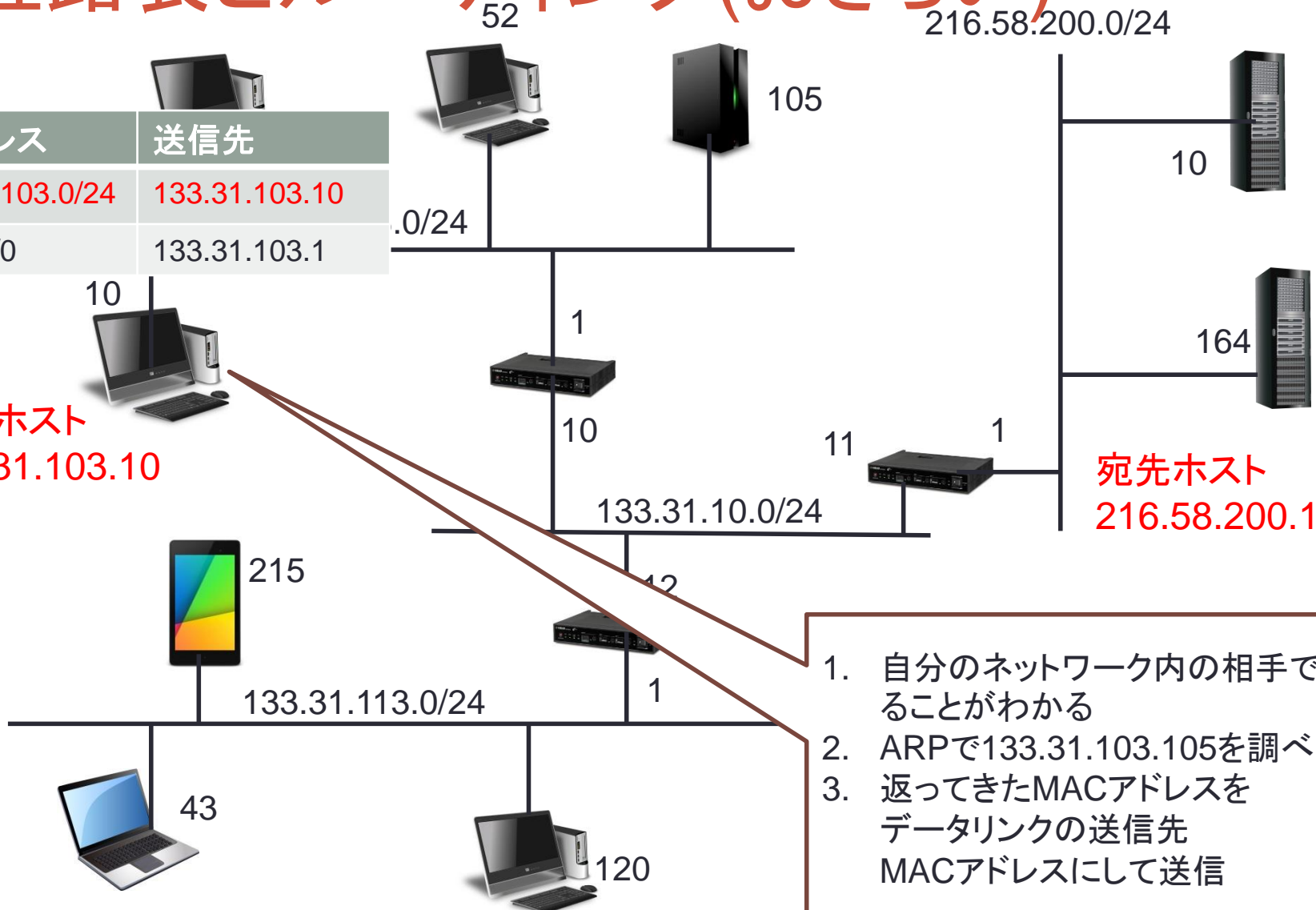
教科書より

経路表とルーティング(おさらい)

IPアドレス	送信先
133.31.103.0/24	133.31.103.10
0.0.0.0/0	133.31.103.1

送信ホスト
133.31.103.10

宛先ホスト
216.58.200.164

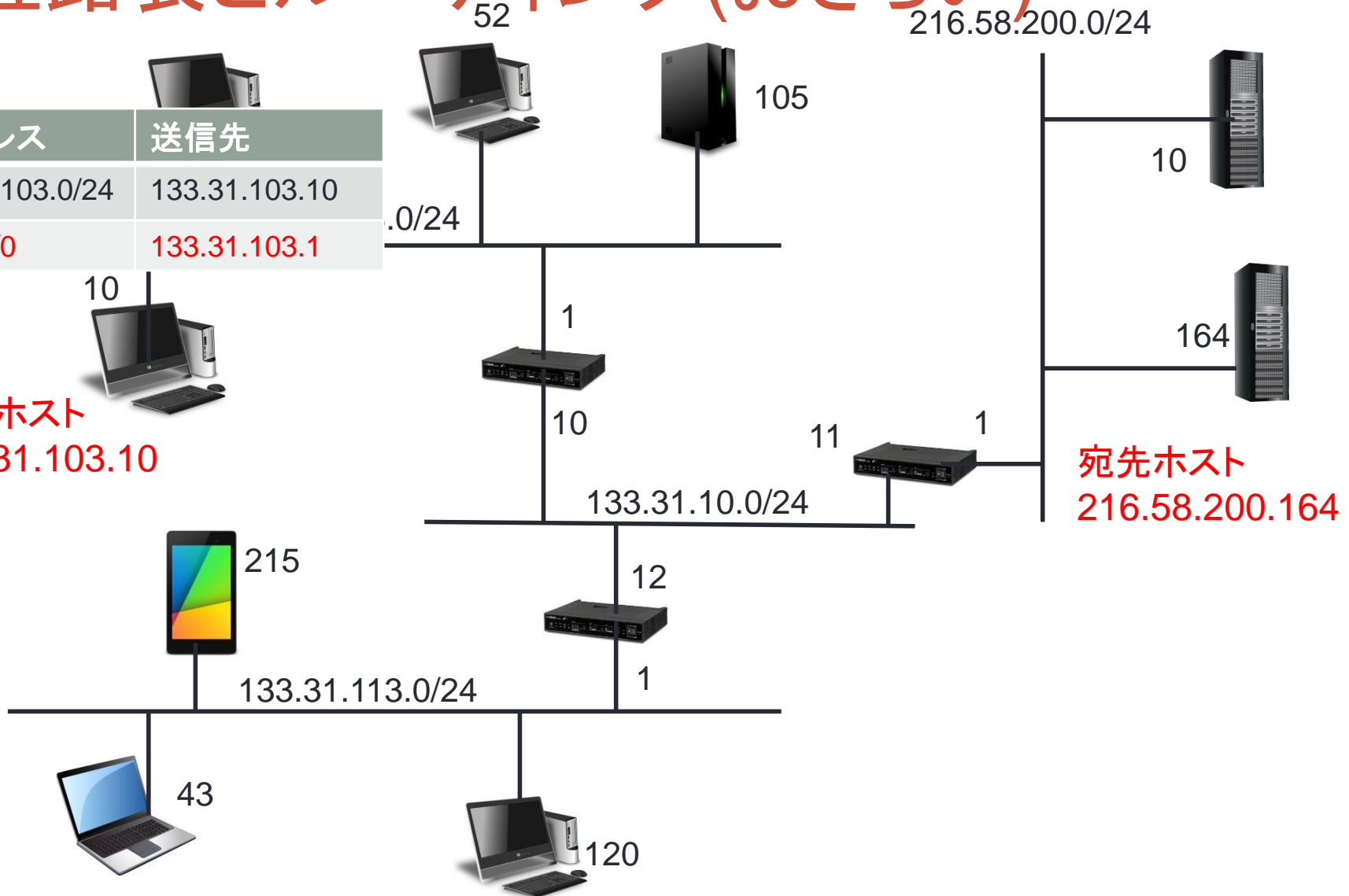


1. 自分のネットワーク内の相手であることがわかる
2. ARPで133.31.103.105を調べる
3. 返ってきたMACアドレスをデータリンクの送信先MACアドレスにして送信

経路表とルーティング(おさらい)

IPアドレス	送信先
133.31.103.0/24	133.31.103.10
0.0.0.0/0	133.31.103.1

送信ホスト
133.31.103.10

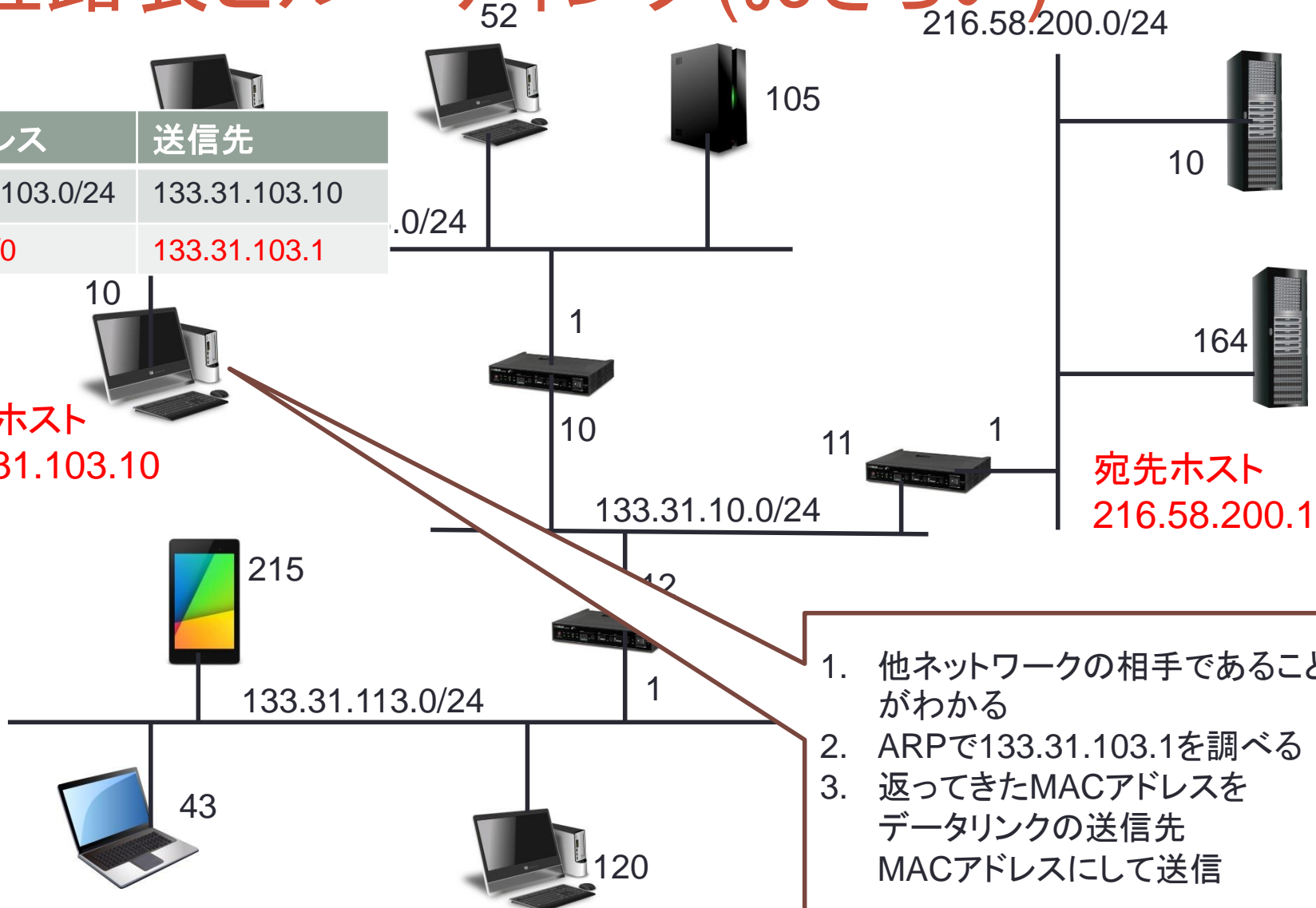


経路表とルーティング(おさらい)

IPアドレス	送信先
133.31.103.0/24	133.31.103.10
0.0.0.0/0	133.31.103.1

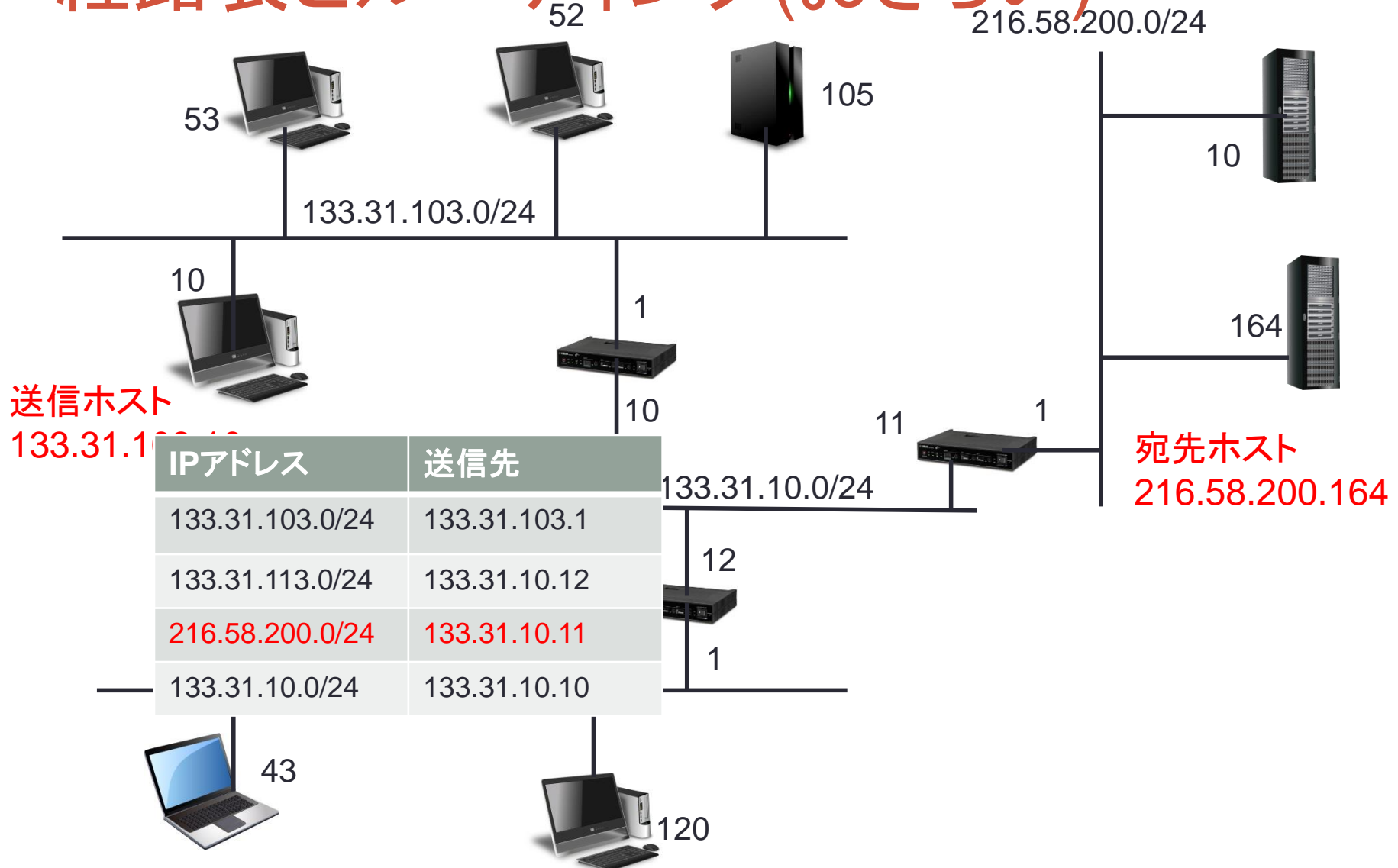
送信ホスト
133.31.103.10

宛先ホスト
216.58.200.164

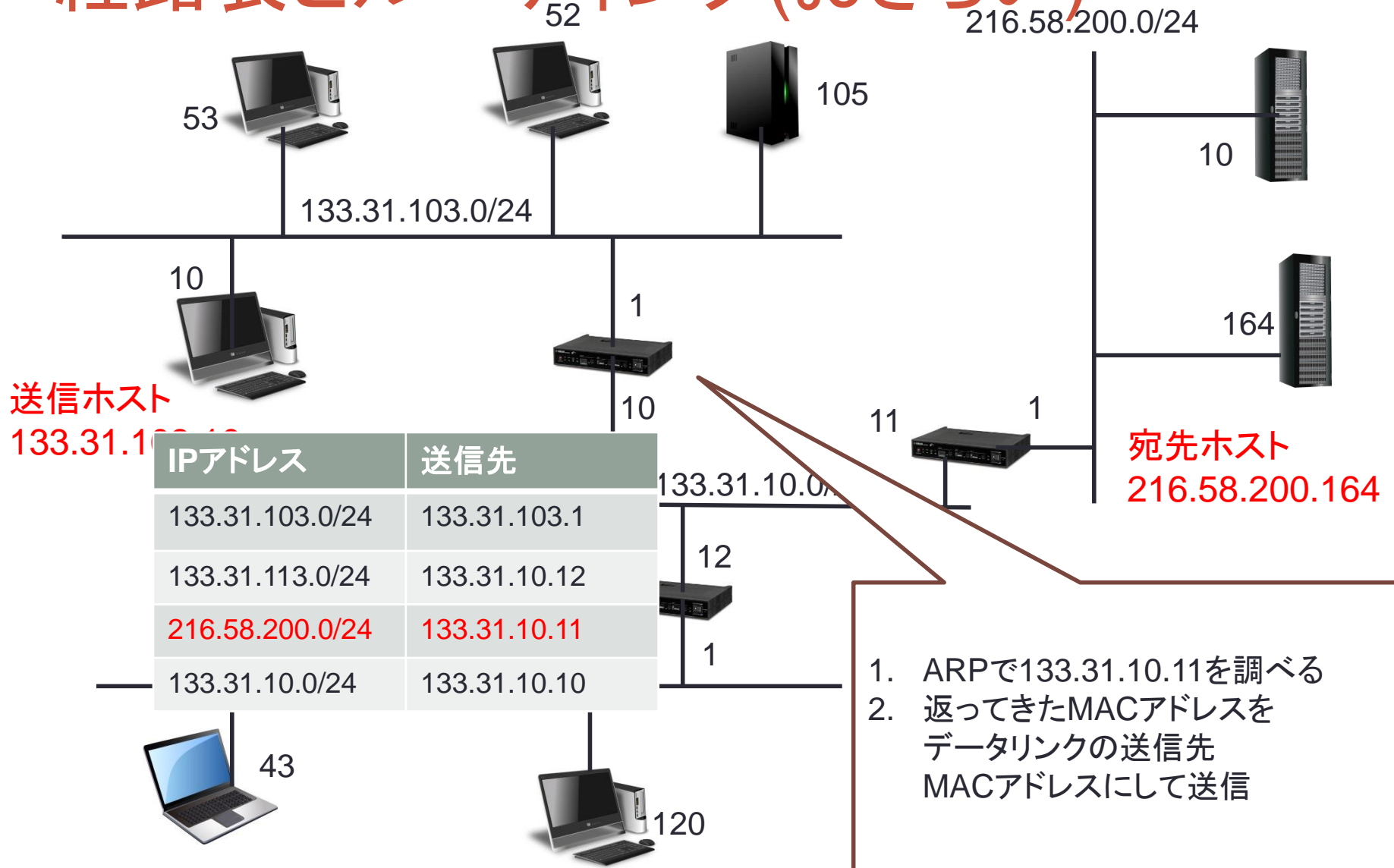


1. 他ネットワークの相手であることがわかる
2. ARPで133.31.103.1を調べる
3. 返ってきたMACアドレスをデータリンクの送信先MACアドレスにして送信

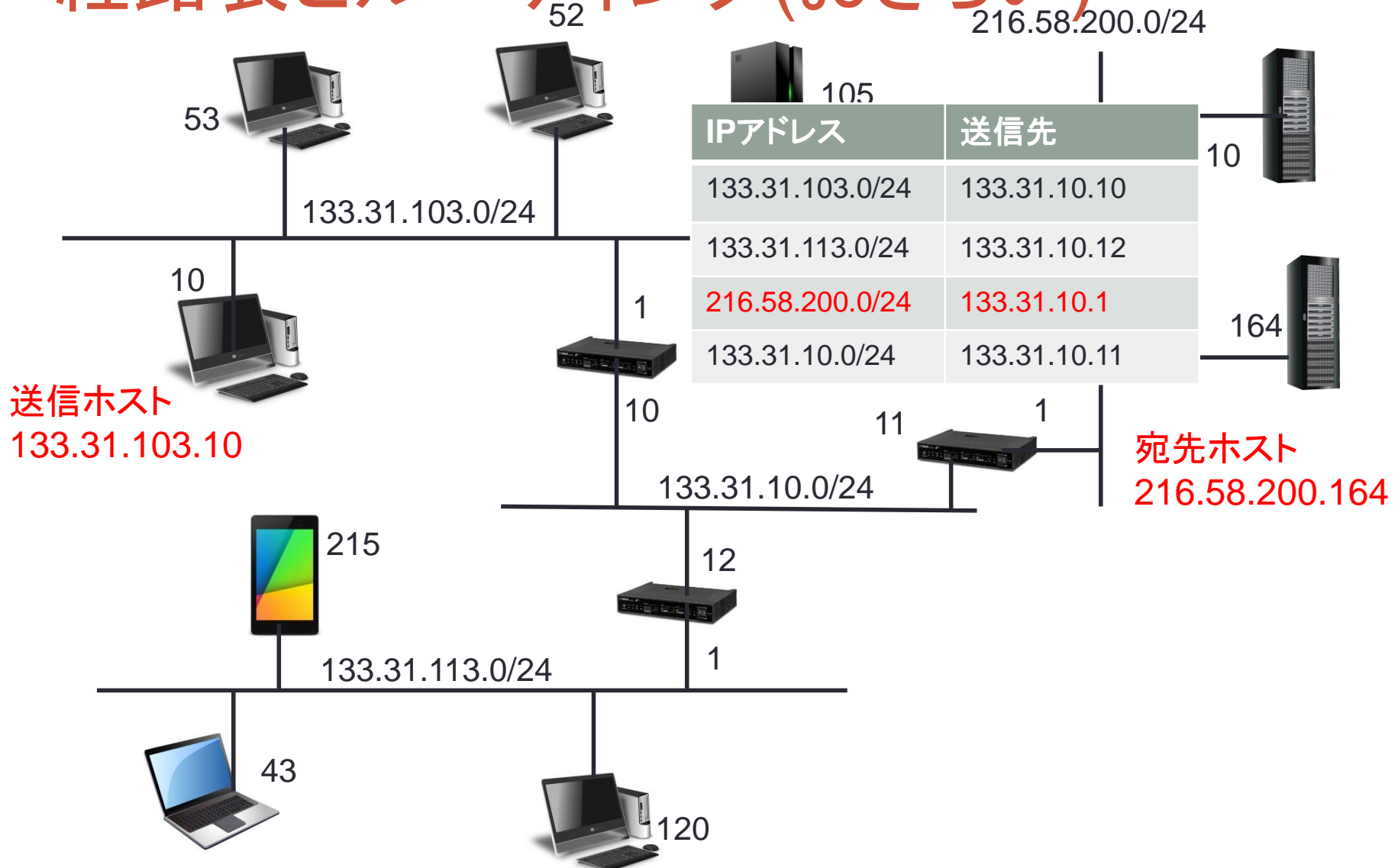
経路表とルーティング(おさらい)



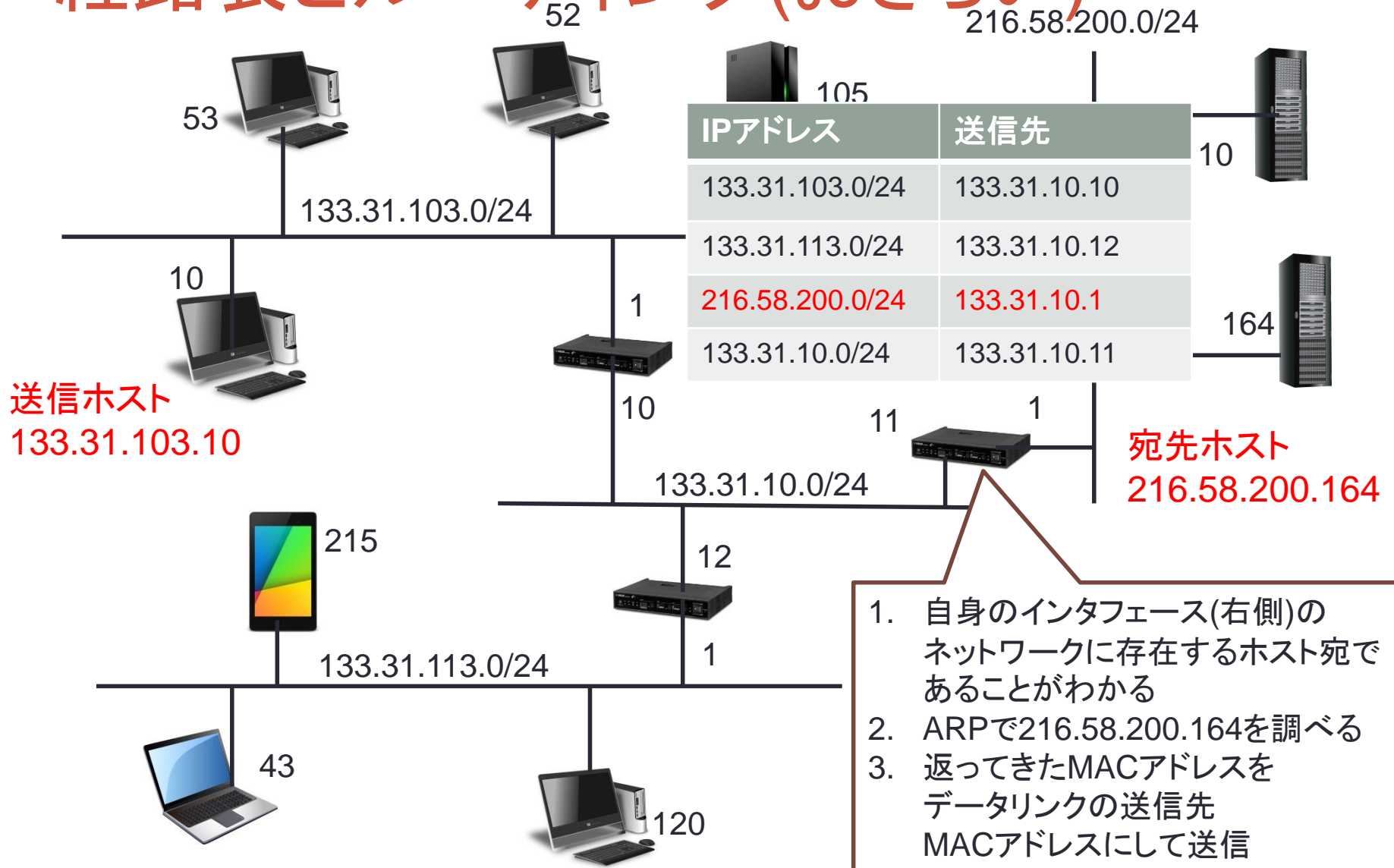
経路表とルーティング(おさらい)



経路表とルーティング(おさらい)



経路表とルーティング(おさらい)



経路表とルーティングまとめ

- 各ルータは経路表(ルーティングテーブル)に従い次の転送先を決定する
- 適切な経路表を持つ必要がある

静的ルーティングと動的ルーティング

- 静的ルーティング
 - 管理者が設定する
 - 小さなネットワーク向き
- 動的ルーティング
 - ルーティングプロトコルを用いてルータ同士が情報交換して経路表を作成する

スタティックルーティング

routeコマンドで設定する

- Linux (unix系OS)の場合の例:

```
# route add -net 192.56.76.0 netmask 255.255.255.0 dev eth0
```

```
# route add default gw 155.155.155.1
```

- Windowsの場合の例:

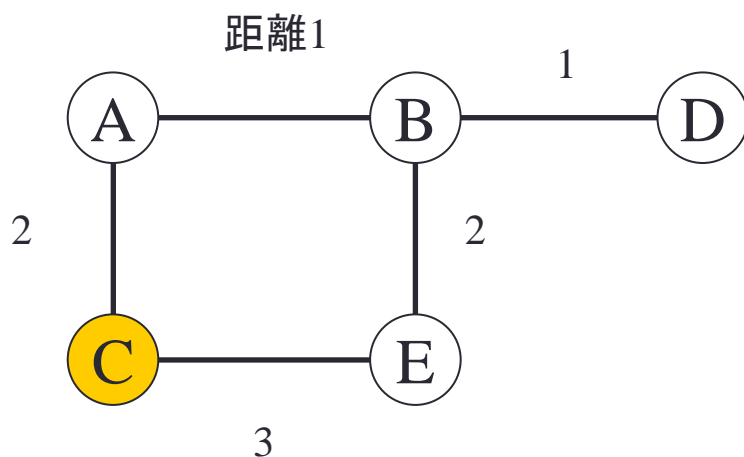
```
> route ADD 157.0.0.0 MASK 255.0.0.0 157.55.80.1 IF2
```

ルーティングプロトコル

- 経路情報をルータ同士で交換する
- アルゴリズムの違いによって複数の種類がある
 - RIP (Routing Information Protocol)
 - OSPF (Open Shortest Path First)
 - BGP (Border Gateway Protocol) ... AS間通信などに利用
- 宛先ネットワークまでの距離を表す「メトリック」という値が割り出され、メトリックの小さい経路を最適な経路とする

RIP (Routing Information Protocol)

- ほとんどのルータに搭載されているプロトコル
- ルータが数台程度の時に利用される
- 現在のバージョンは2. (1はCIDR・手動集約経路未対応)
- ディスタンスベクタ(Distance Vector)方式で制御する

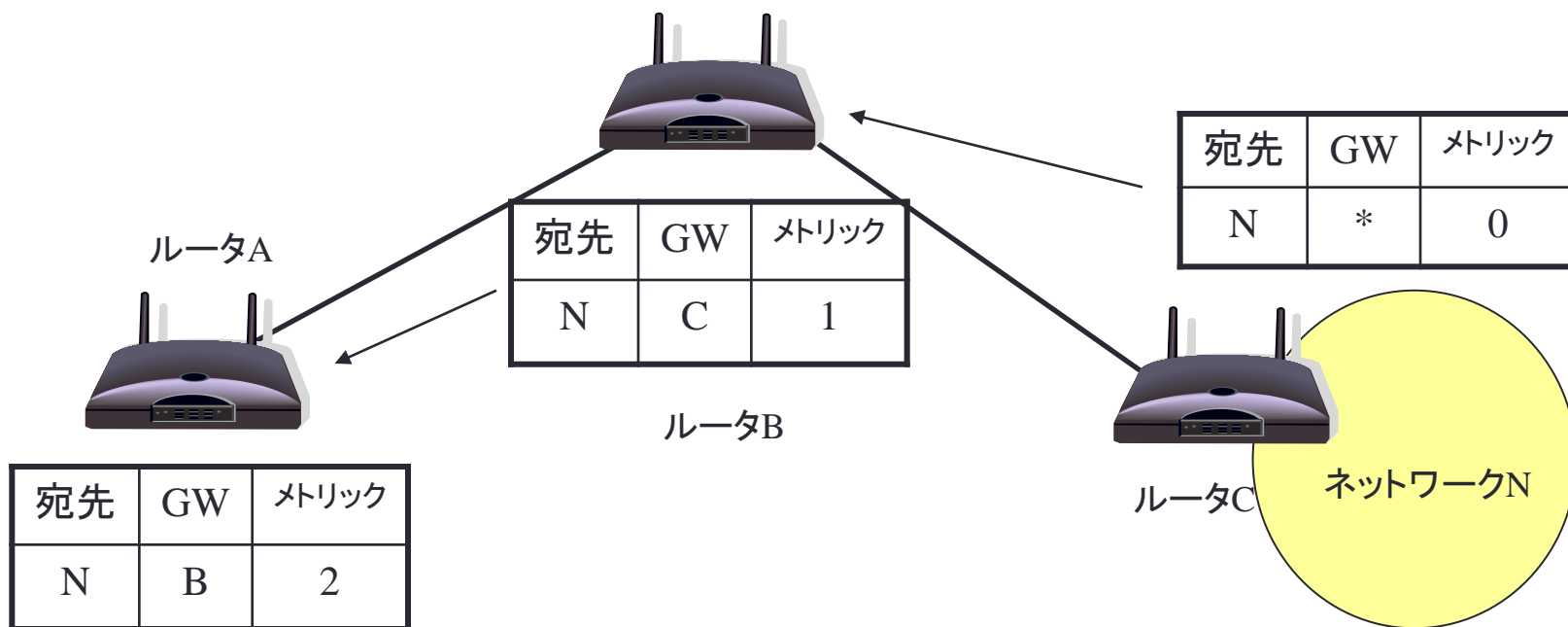


宛先	方向	距離
A	A	2
B	A	3
C	-	0
D	A	4
E	E	3

Cのテーブル

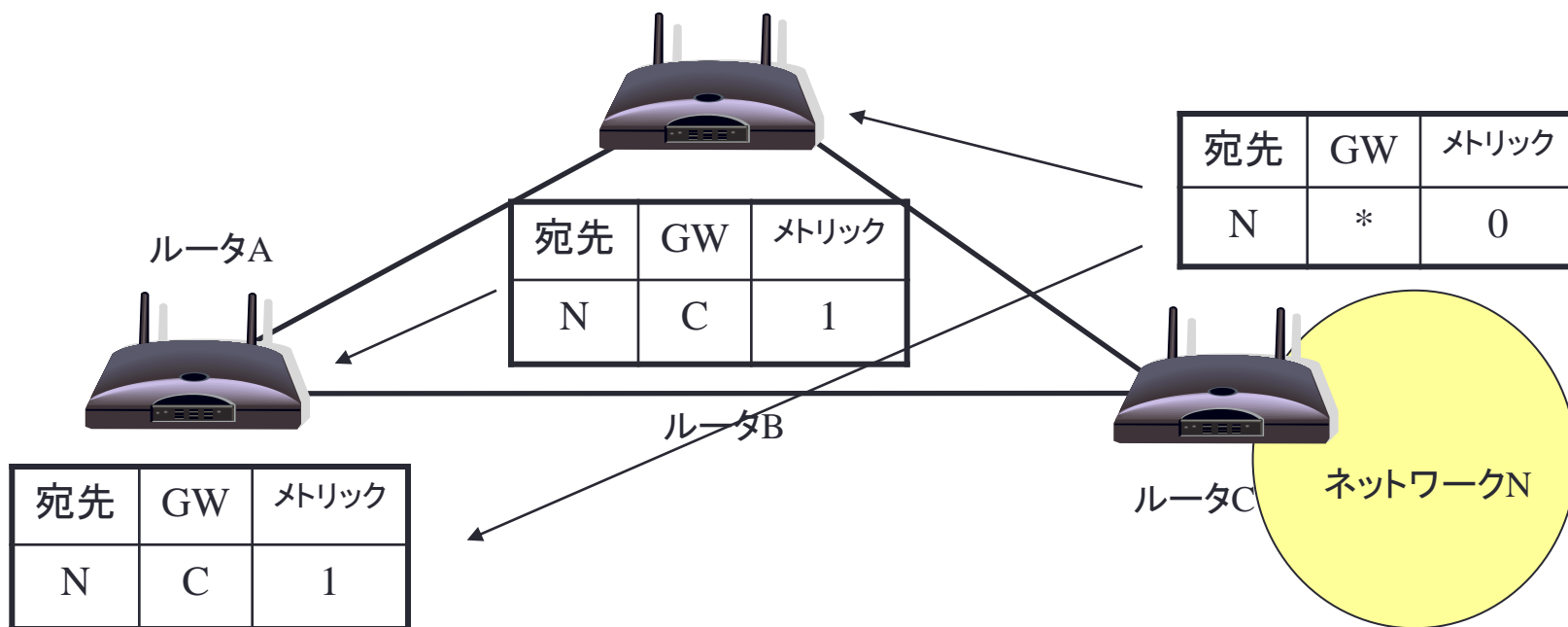
RIP (Routing Information Protocol)

- 経路交換の手順
 - IPアドレスとメトリックを隣接ルータと30秒ごとに交換(RIP advertisement)する
 - 経路情報を受け取ったルータはメトリックを1足して追加する
 - 同じ宛先からの情報を受け取った場合はメトリックの小さい方を採用する



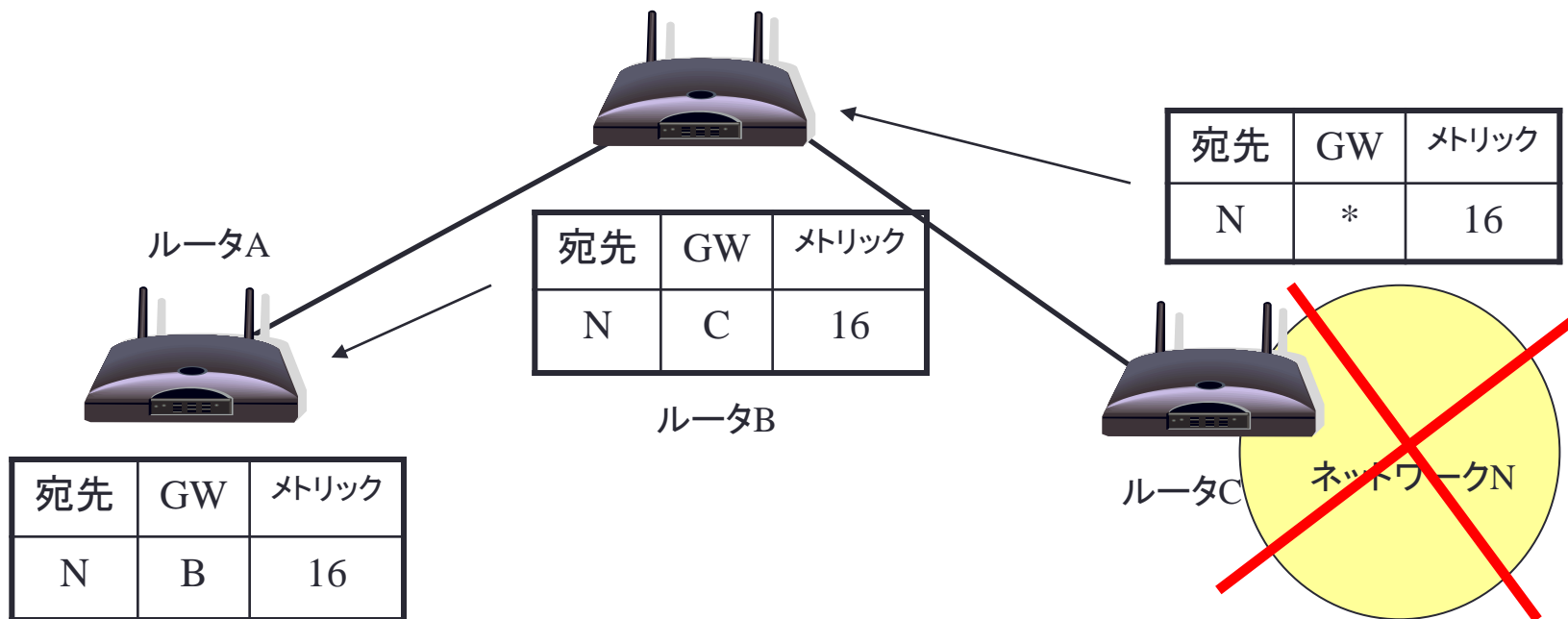
RIP (Routing Information Protocol)

- 経路交換の手順
 - IPアドレスとメトリックを隣接ルータと30秒ごとに交換(RIP advertisement)する
 - 経路情報を受け取ったルータはメトリックを1足して追加する
 - 同じ宛先からの情報を受け取った場合はメトリックの小さい方を採用する



RIP (Routing Information Protocol)

- 経路がなくなった時の動作
 - ルータ間が切れた場合はタイムアウト(180秒)によってメトリックを16にして通知する
 - 直接繋がっているネットワークが切断された場合はメトリックを16にして、即通知する(Triggered Update)



RIPの問題点(ループ)

宛先	GW	メトリック
N	B	2



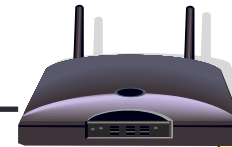
ルータA

宛先	GW	メトリック
N	C	1

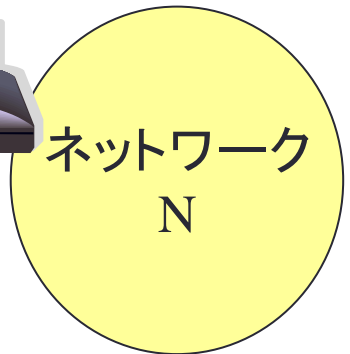


ルータB

宛先	GW	メトリック
N	*	0



ルータC



RIPの問題点(ループ)

宛先	GW	メトリック
N	B	2



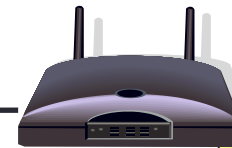
ルータA

宛先	GW	メトリック
N	C	1



ルータB

宛先	GW	メトリック
N	*	0



ルータC



リンクダウン発生

ネットワーク
N

RIPの問題点(ループ)

宛先	GW	メトリック
N	B	2



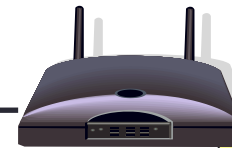
ルータA

宛先	GW	メトリック
N	C	1



ルータB

宛先	GW	メトリック
N	*	0



ルータC

ネットワーク
N

ネットワークNの情報を通知

RIPの問題点(ループ)

宛先	GW	メトリック
N	B	2



ルータA

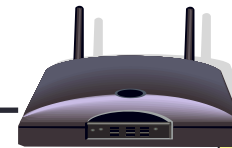
宛先	GW	メトリック
N	A	3



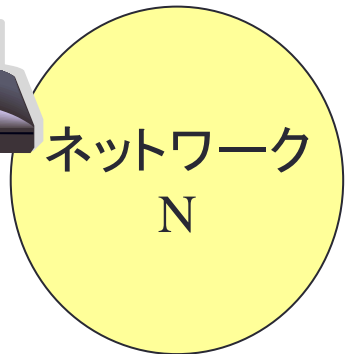
ルータB



宛先	GW	メトリック
N	*	0



ルータC

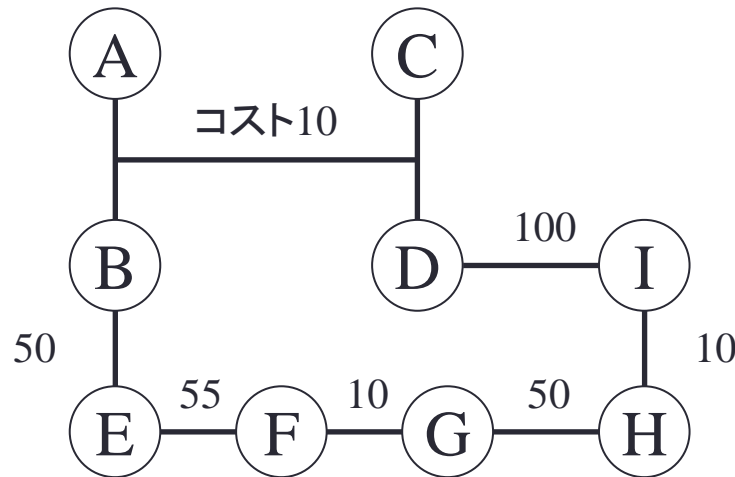


RIPのループ防止策

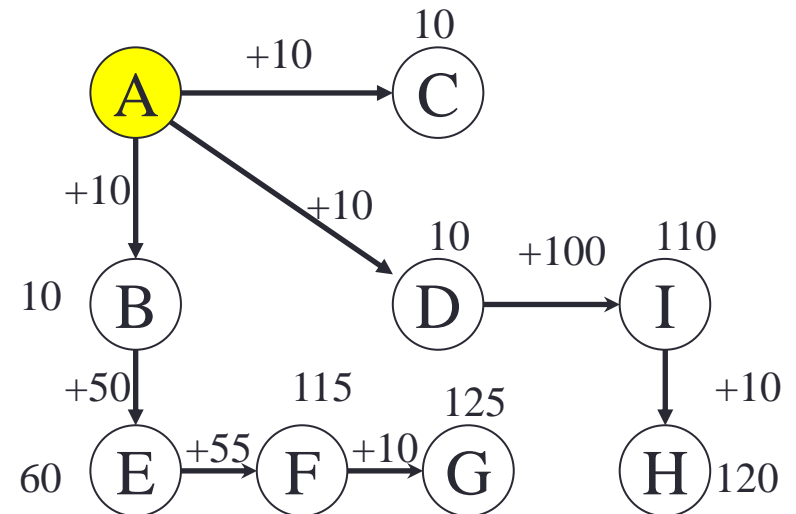
- スプリットホライズン
 - 経路情報の送信元となるルータには、その経路情報を送らない
 - ただし3点以上のループには対応できない
- イベント・トリガー・アップデート
 - ネットワークの変更内容を、直ちに他のルーターに伝える

OSPF (Open Shortest Path First)

- 大規模なネットワークに対応したルーティングプロトコル
 - 収束が早い
 - 回線帯域を考慮した経路制御が可能
- 複雑なため安価なルータには実装されていないケースもある
- リンクステート(Link State)方式を用いて経路を計算する



ネットワークの構成とコスト

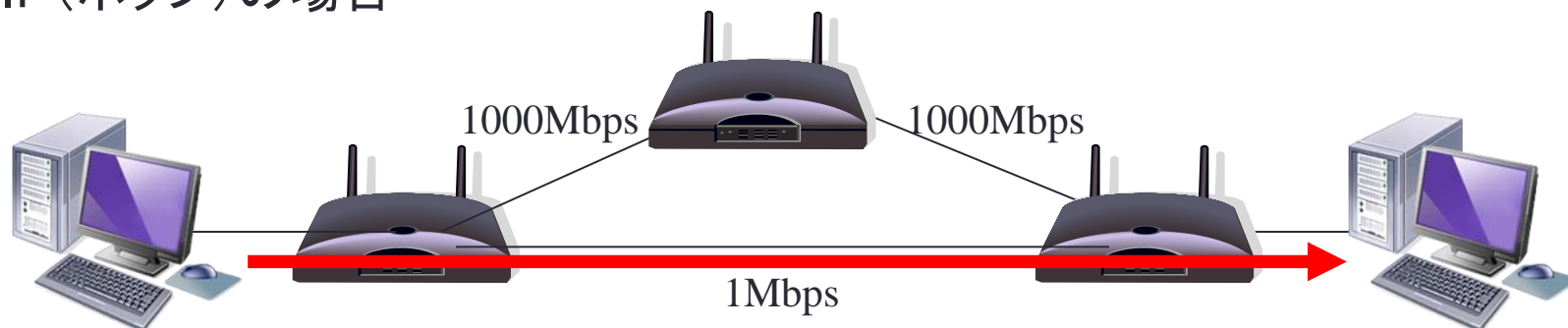


ルータAを頂点とした最短パスツリー

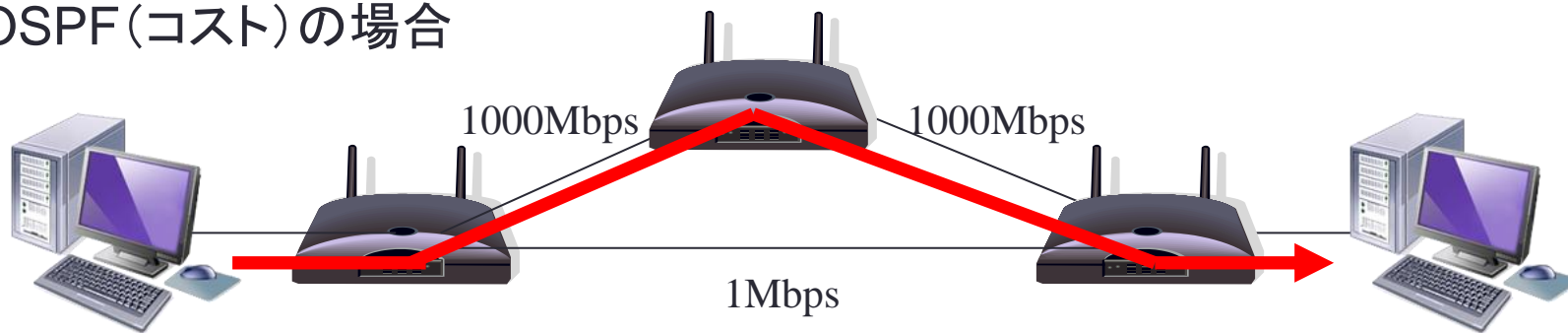
OSPF (Open Shortest Path First)

コストとホップの違い

RIP (ホップ) の場合



OSPF (コスト) の場合



※ Ciscoルータはデフォルトでコストを $100 \div \text{Mbps}$ としている

OSPF (Open Shortest Path First)

- データベース
 - 各ルータはLSDB(Link State DataBase)というOSPF用のDBを持つ
- ルータを識別する方法
 - 32bitのルータIDを用いる. (明示的に設定しなければIPアドレスのいずれかが選択される)
- OSPFルータ同士の通信
 - OSPF自身が再送制御などのトランスポート層プロトコルの機能を持っている
 - 経路広告にマルチキャストを利用している
 - ルータ間で主従関係を構成する

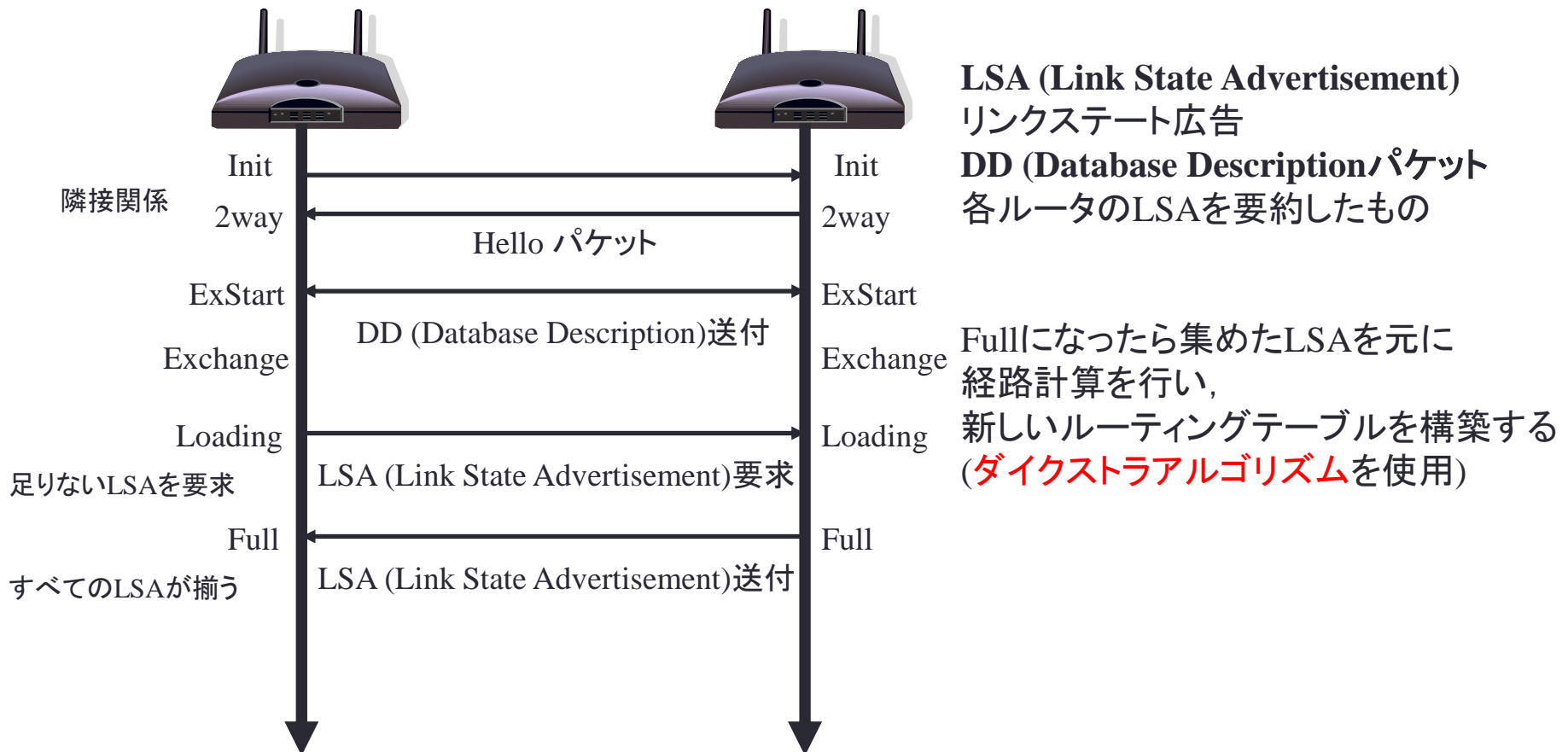
OSPF (Open Shortest Path First)

OSPFパケットタイプ

- Hello
 - 隣接関係の確立と維持
- DD
 - トポロジデータベースの内容を説明. 隣接関係の確立時に使用
- LSA
 - 2つのルータ間のリンク状態をまとめた特殊なパケット. 後述
 - LSU
 - LSRへの応答. LSA情報を含む
 - LSR
 - 隣接ルータにトポロジデータベースの一部を要求
 - LSAck
 - 確認応答

OSPF (Open Shortest Path First)

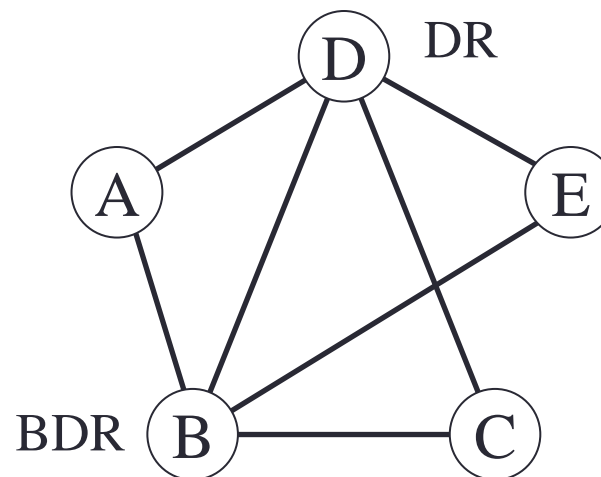
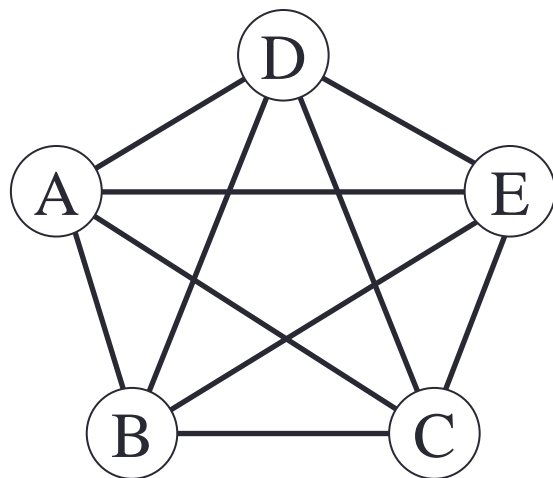
OSPFの動作のフロー



OSPF (Open Shortest Path First)

- 隣接関係の確立

- 効率的な関係を結ぶために、OSPFではDR (Designated Router)とBDR (Backup DR)を設けている
- 各ルータはDRとBDRに強い隣接関係を結ぶ
- DR/BDRの選択には管理者の設定したプライオリティの値を参照する
- このネットワーク内のLSA交換はすべてDRと行われる

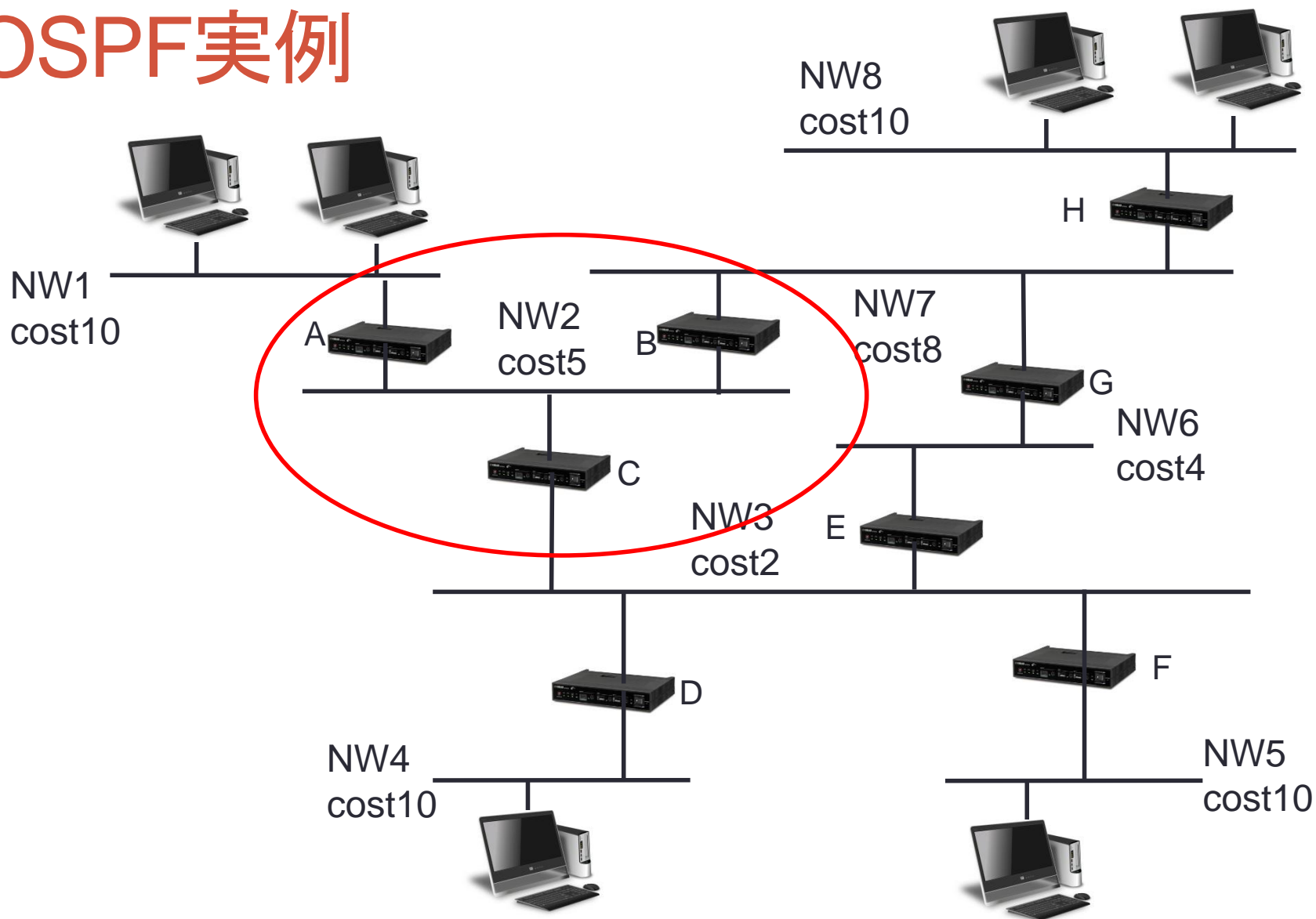


OSPF (Open Shortest Path First)

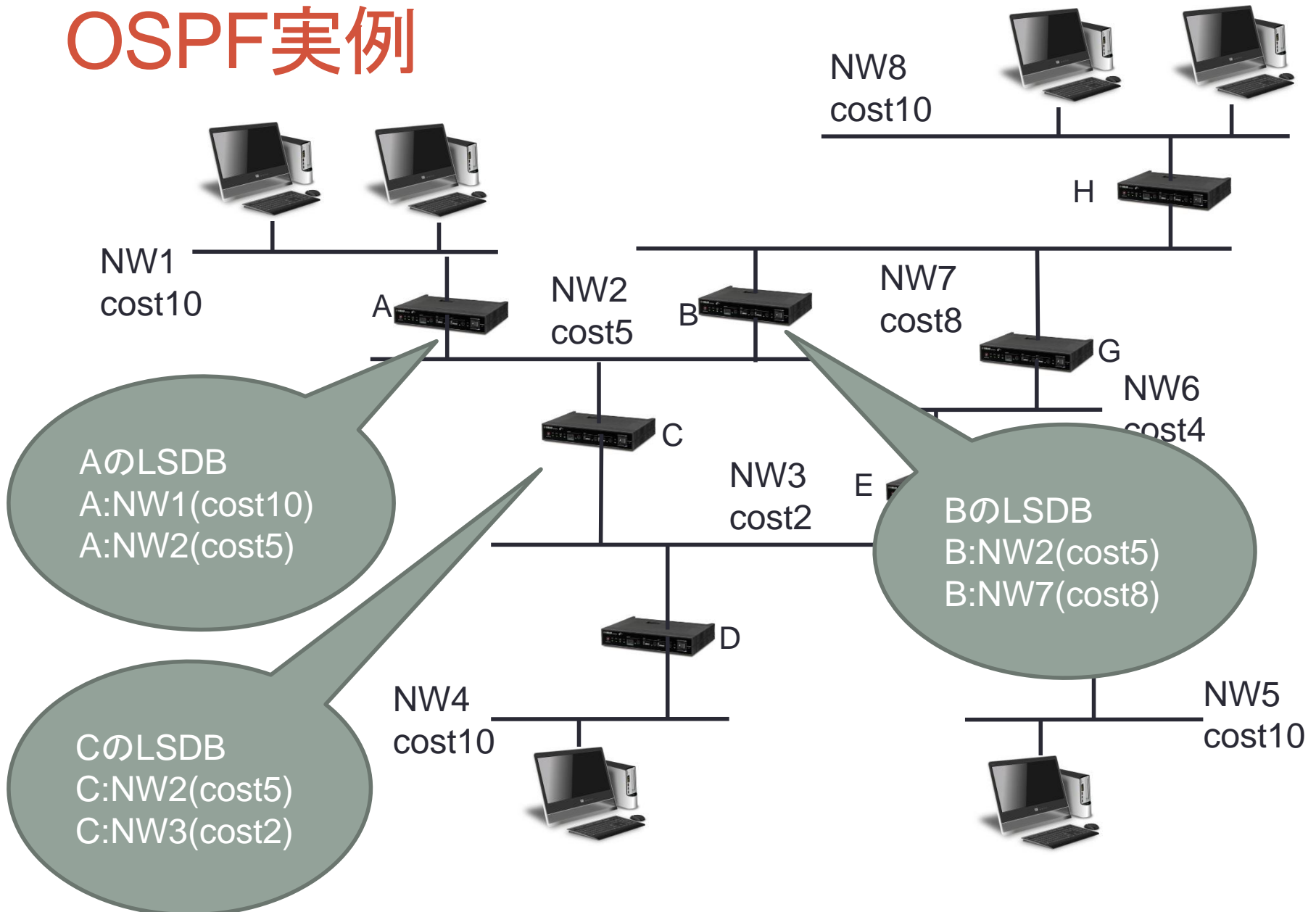
ネットワーク構成が変化した場合

- Helloパケットを利用するケース
 - Helloパケットは10秒に1度流れるが40秒間途絶えた場合、ルータが停止したと判断し、そのルータのLSAを削除する
- 直接トポロジ変化を検知したケース
 - LSDBからリンクダウンしたネットワークのLSAを削除し、即座にDRとBDRに通知する.
 - DRは配下のルータに対してトポロジ変更があった旨の通知を送信する

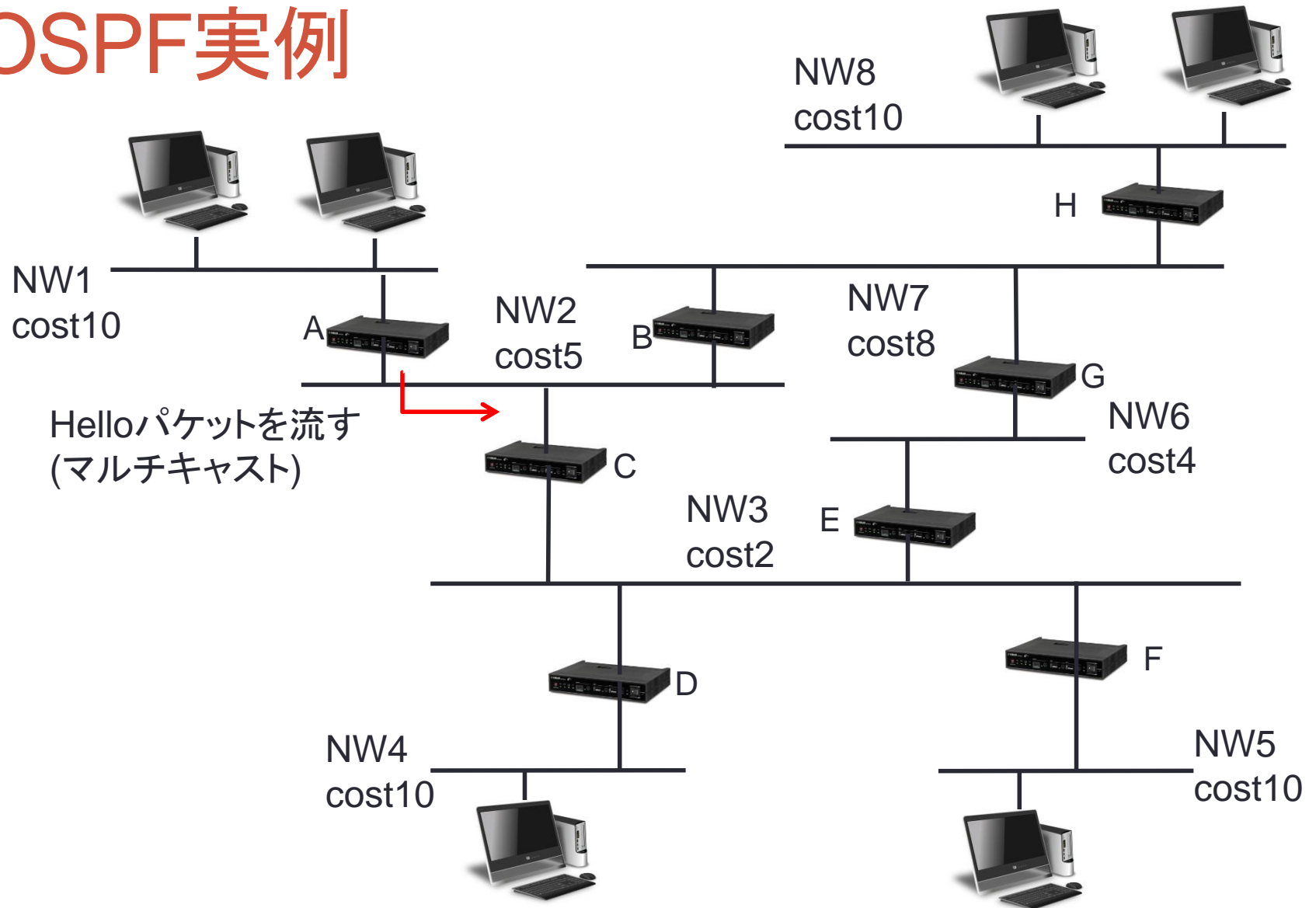
OSPF实例



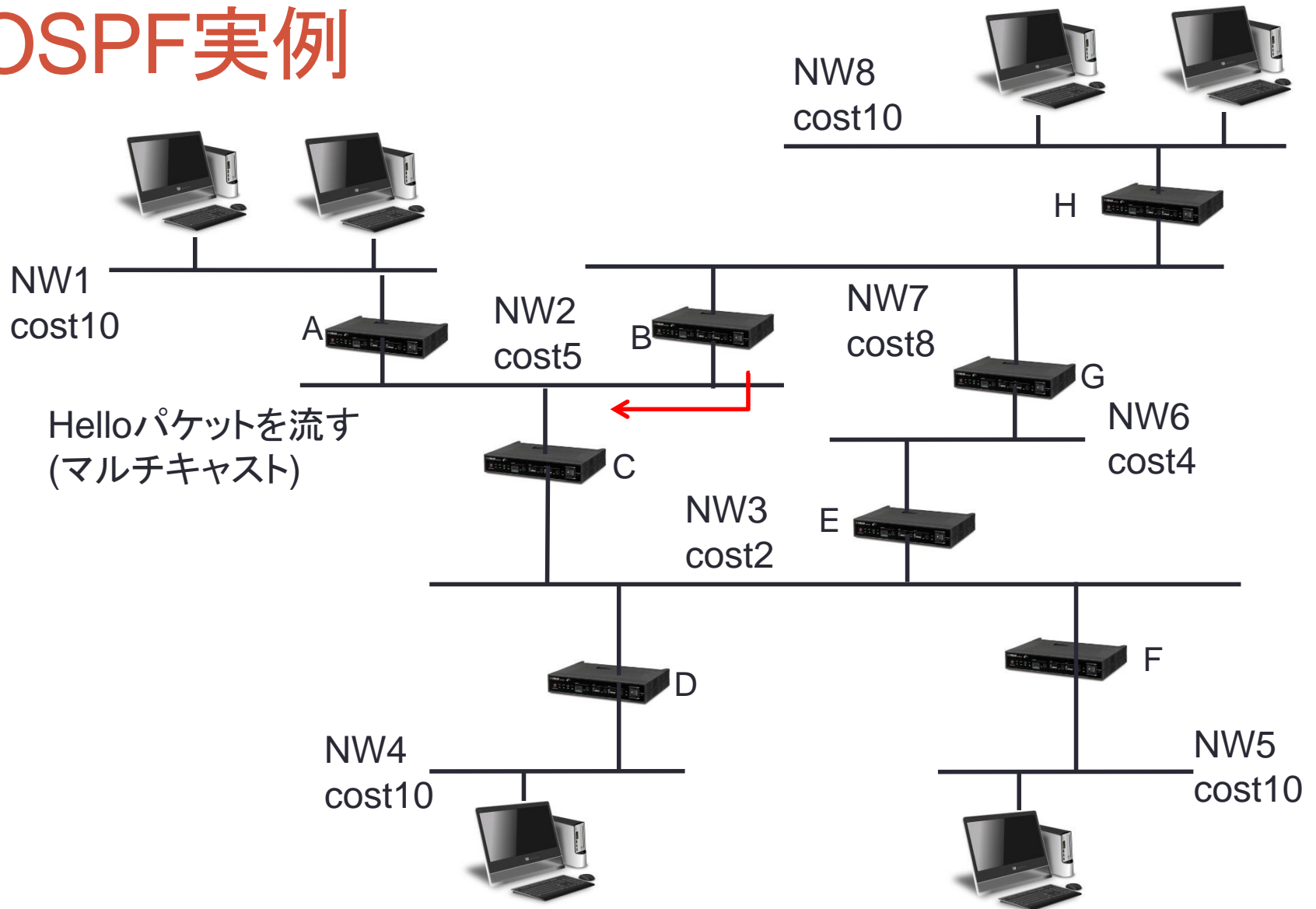
OSPF実例



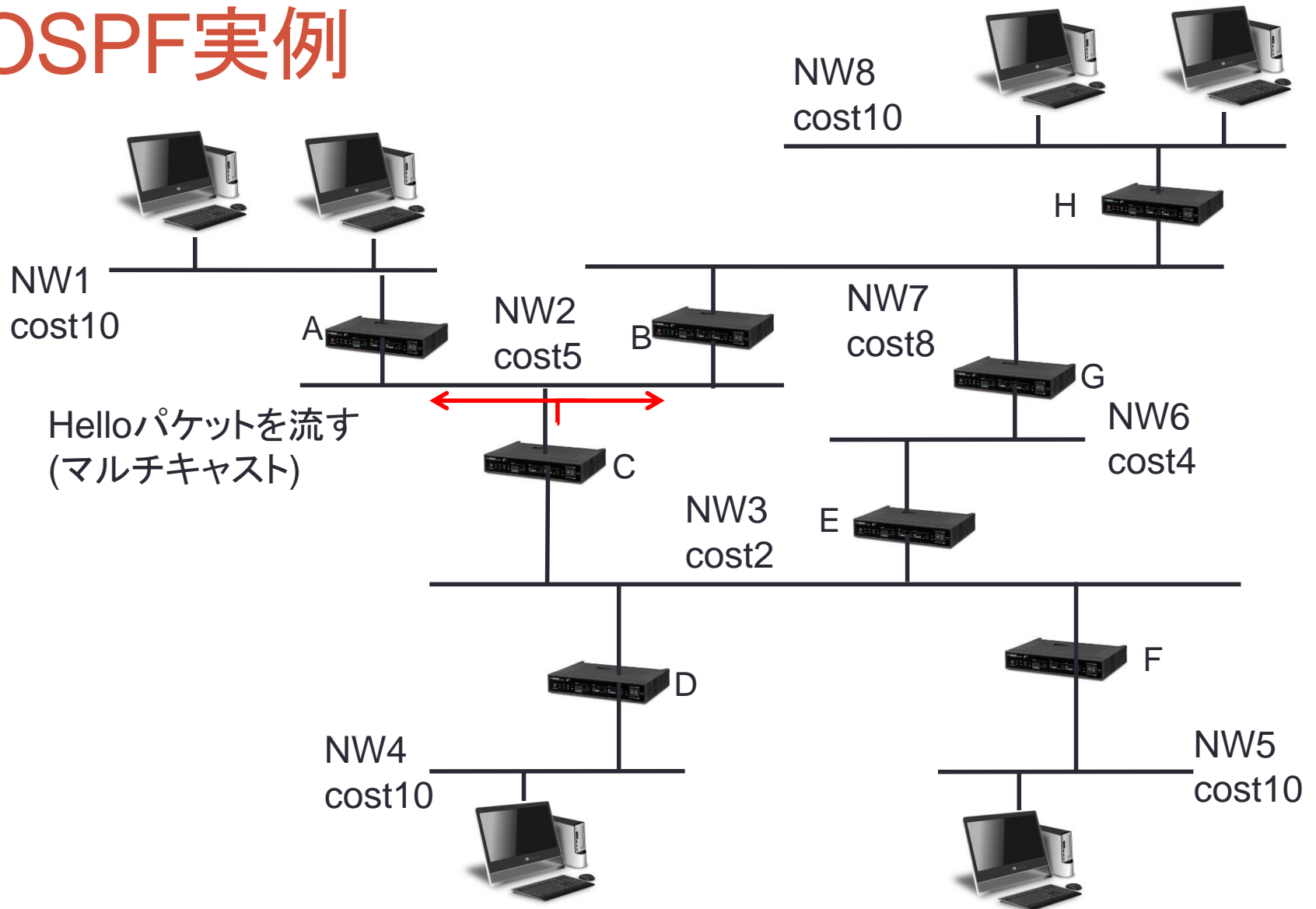
OSPF実例



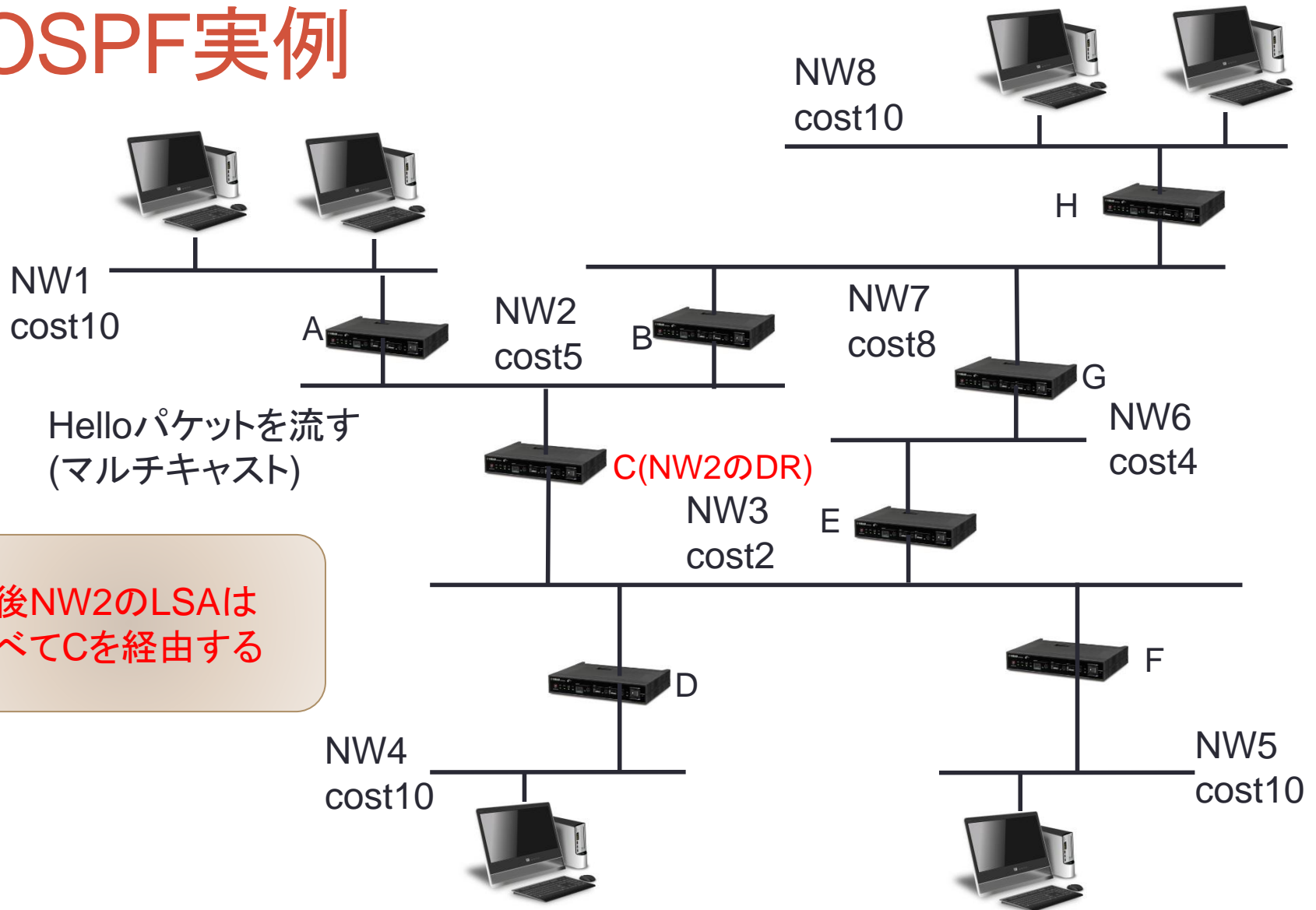
OSPF実例



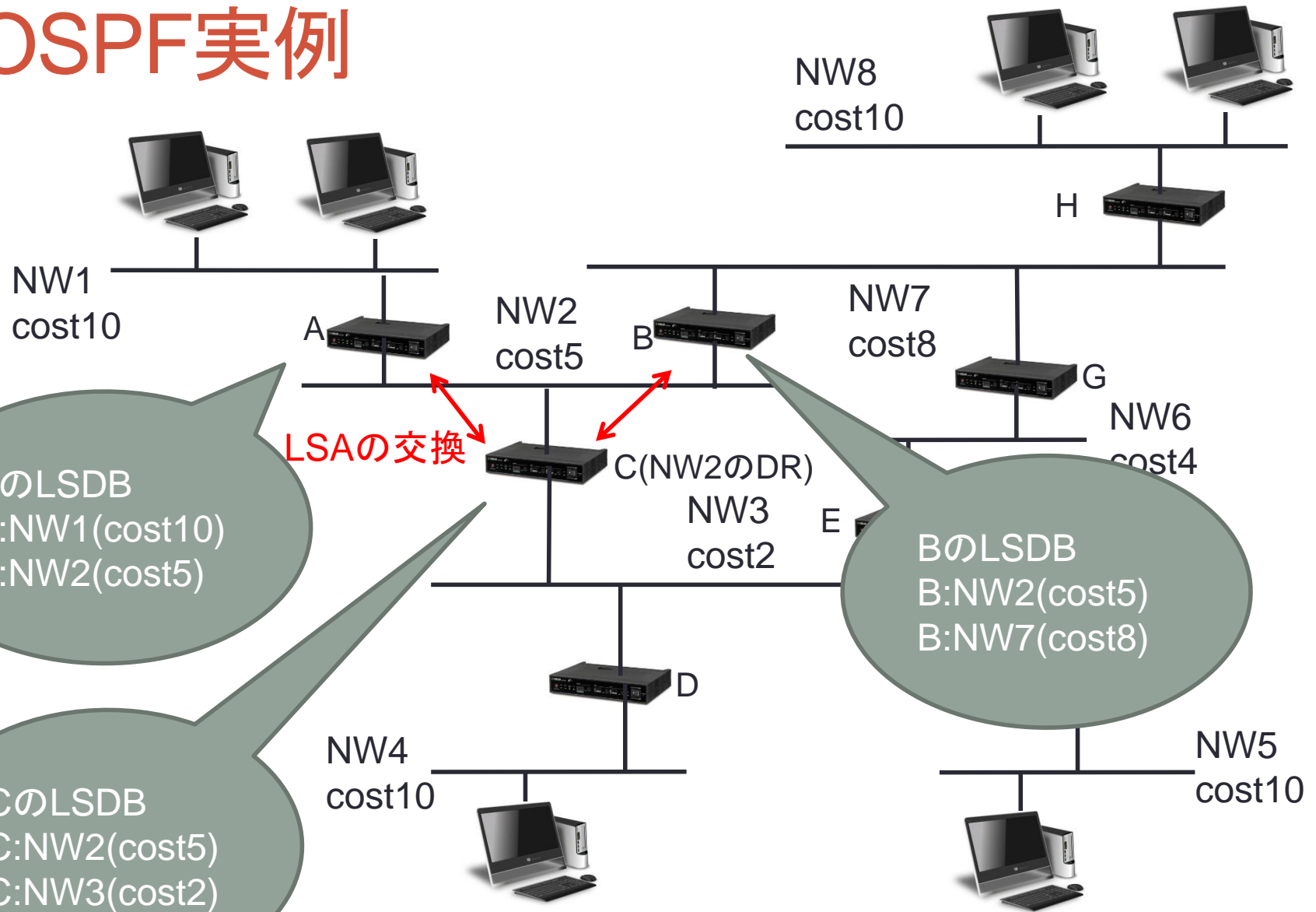
OSPF実例



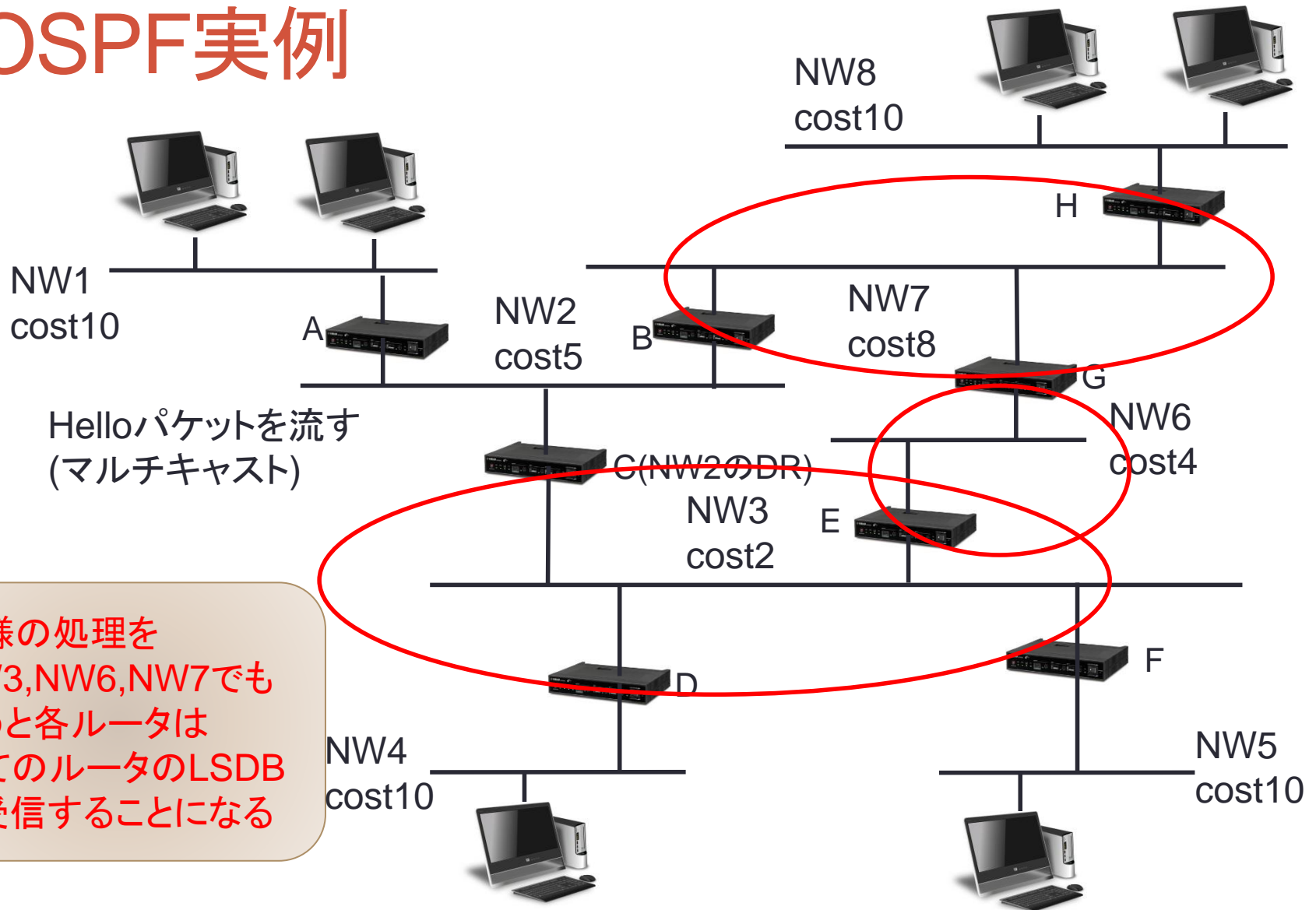
OSPF実例



OSPF実例



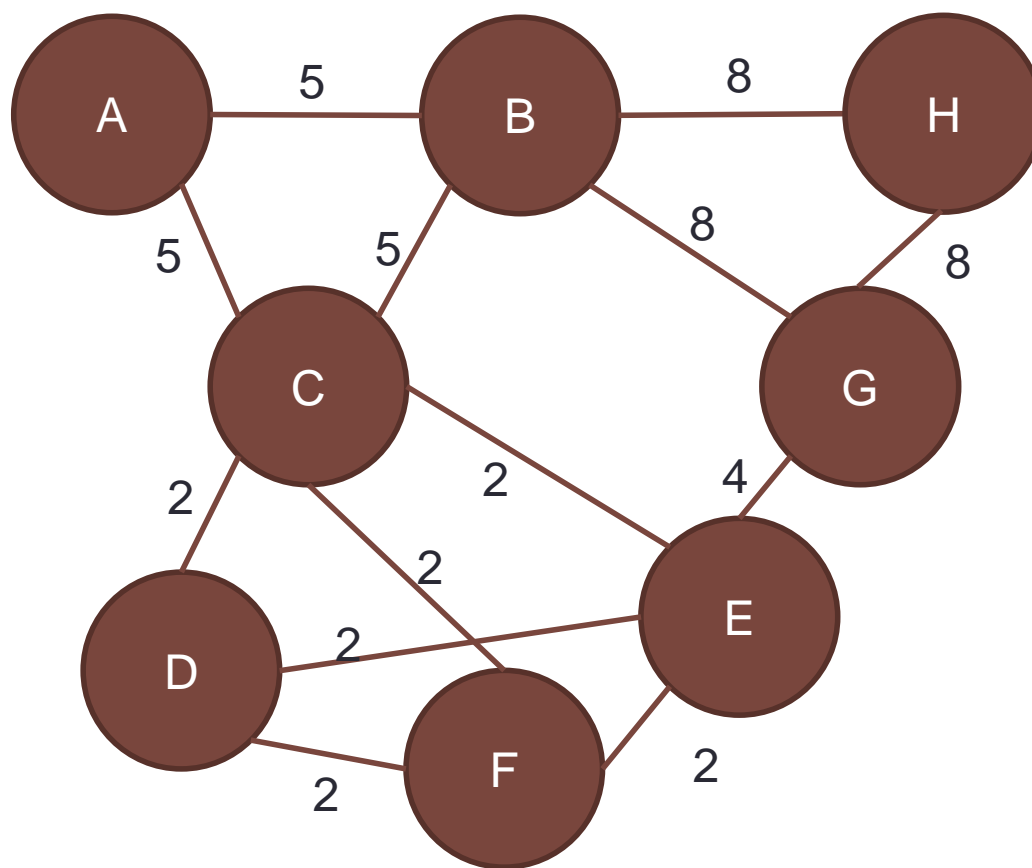
OSPF実例



同様の処理を
NW3,NW6,NW7でも
行くと各ルータは
全てのルータのLSDB
を受信することになる

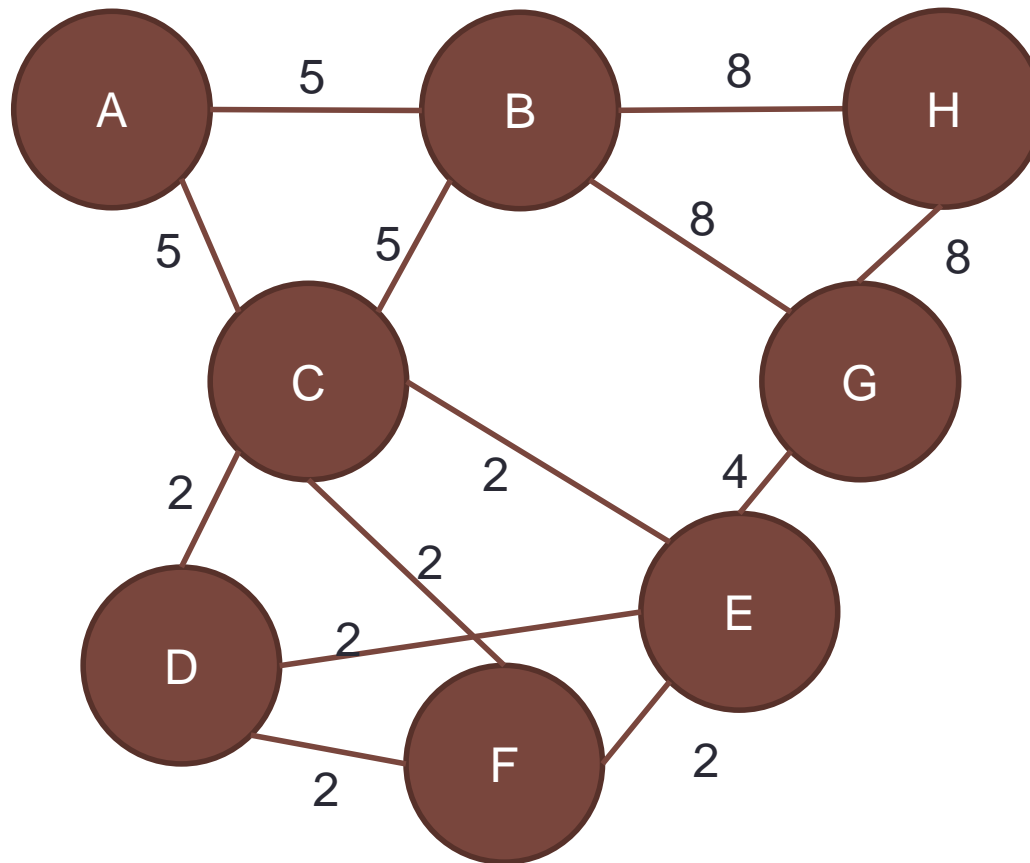
OSPF実例

各ルータは以下のようなグラフを作ることができる



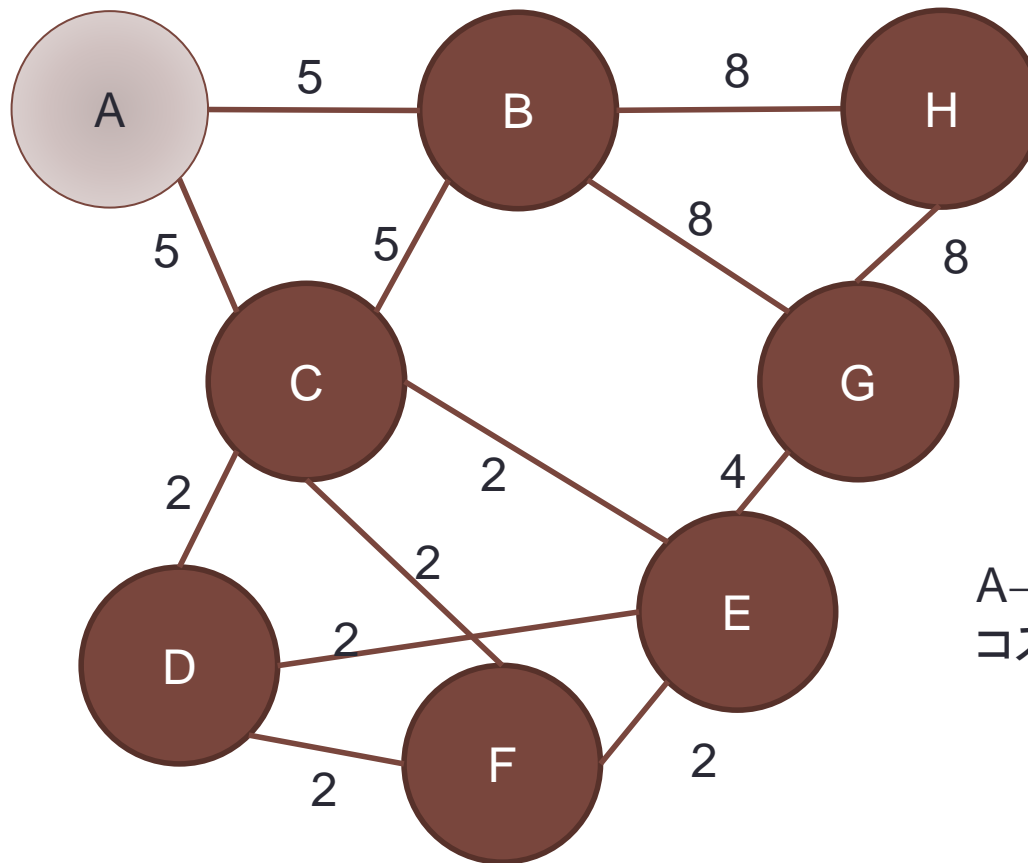
OSPF実例

例：Aの経路表作成(ダイクストラ法)



OSPF実例

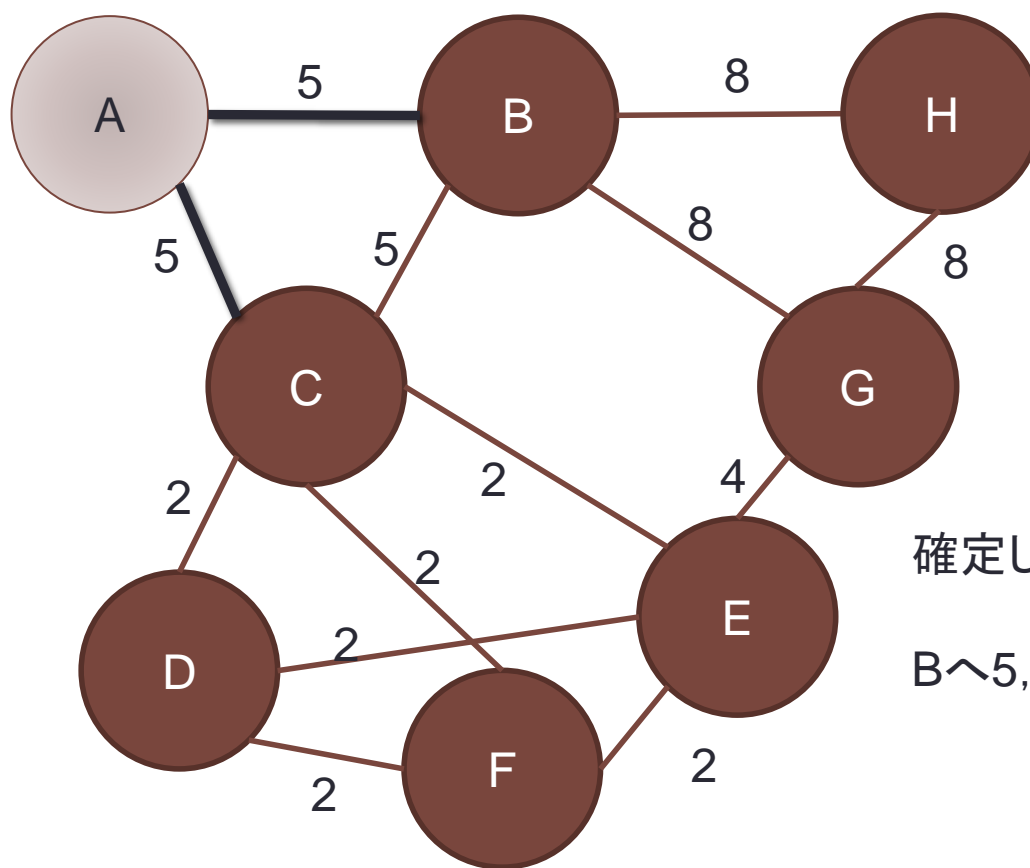
例：Aの経路表作成(ダイクストラ法)



A→Aは
コスト0なので最短確定

OSPF実例

例：Aの経路表作成(ダイクストラ法)

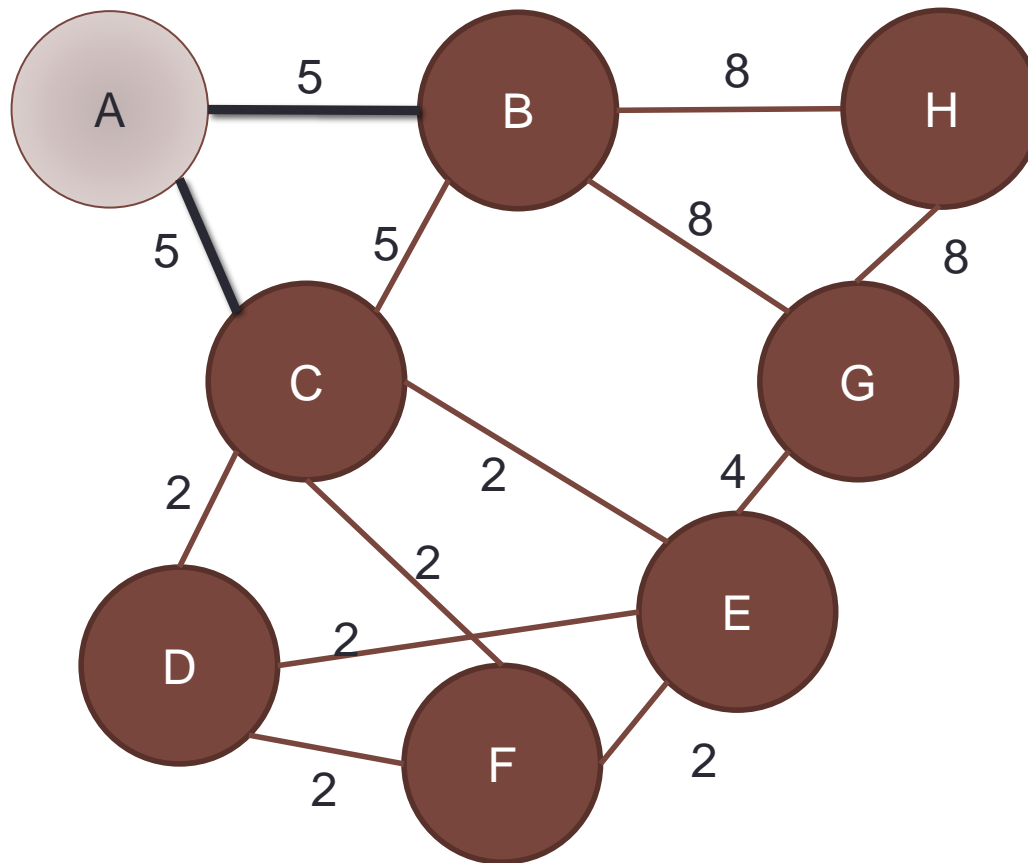


確定したAからのリンクを見る

Bへ5, Cへ5が発見できる

OSPF実例

例：Aの経路表作成(ダイクストラ法)

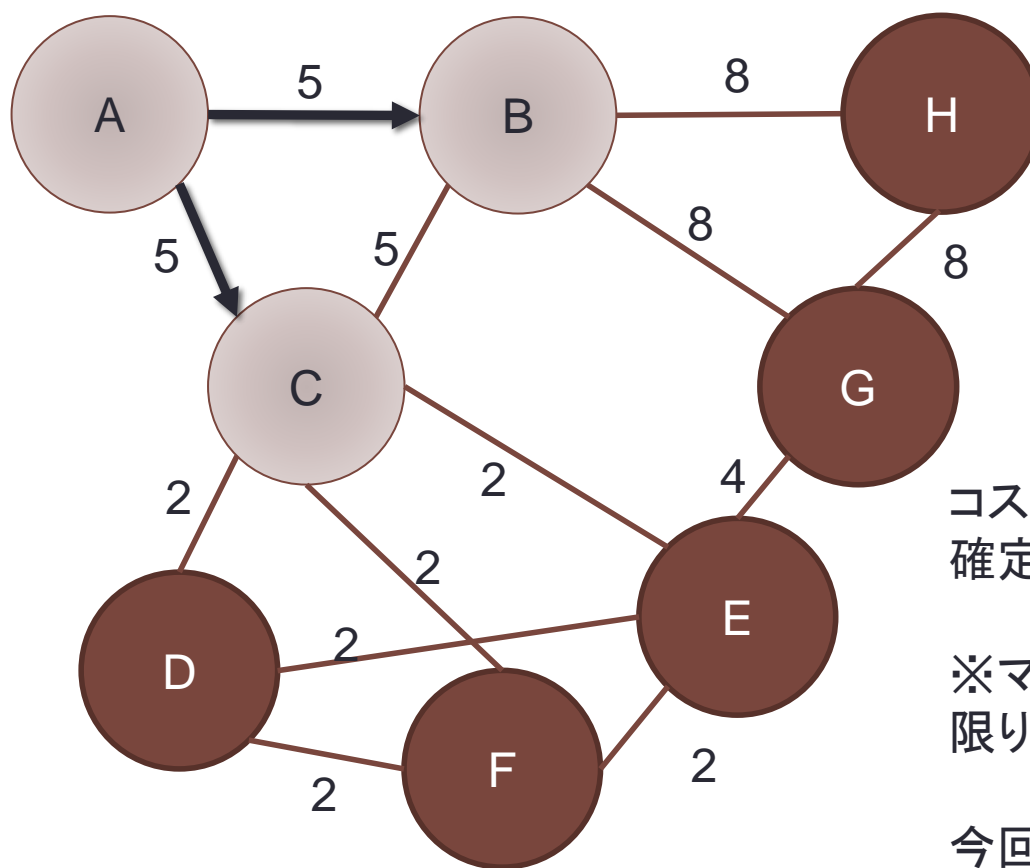


Bへの経路
A→B cost5

Cへの経路
A→C cost5
が候補となる

OSPF実例

例：Aの経路表作成(ダイクストラ法)



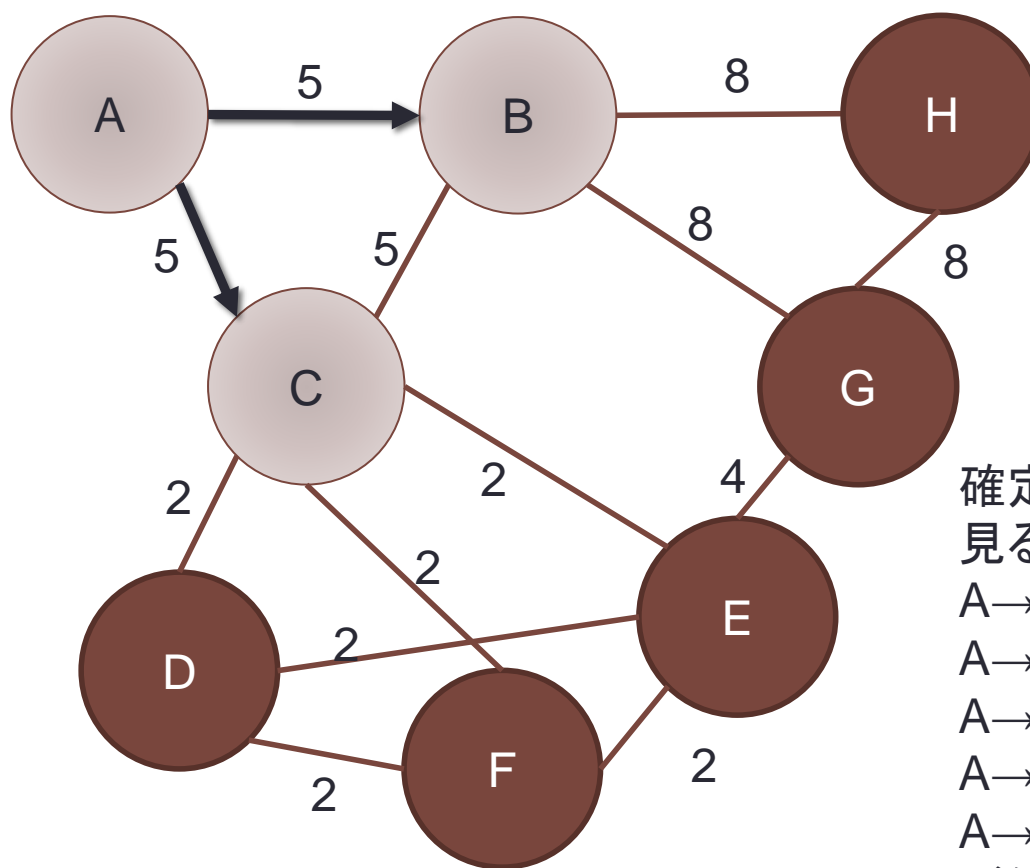
コストが小さい方を
確定させる

※マイナスのコストがない
限り確定できる

今回の場合は両方確定

OSPF実例

例：Aの経路表作成(ダイクストラ法)



確定したBとCからのリンクを見ると

A→B→H(cost13)

A→B→G(cost13)

A→C→D(cost7)

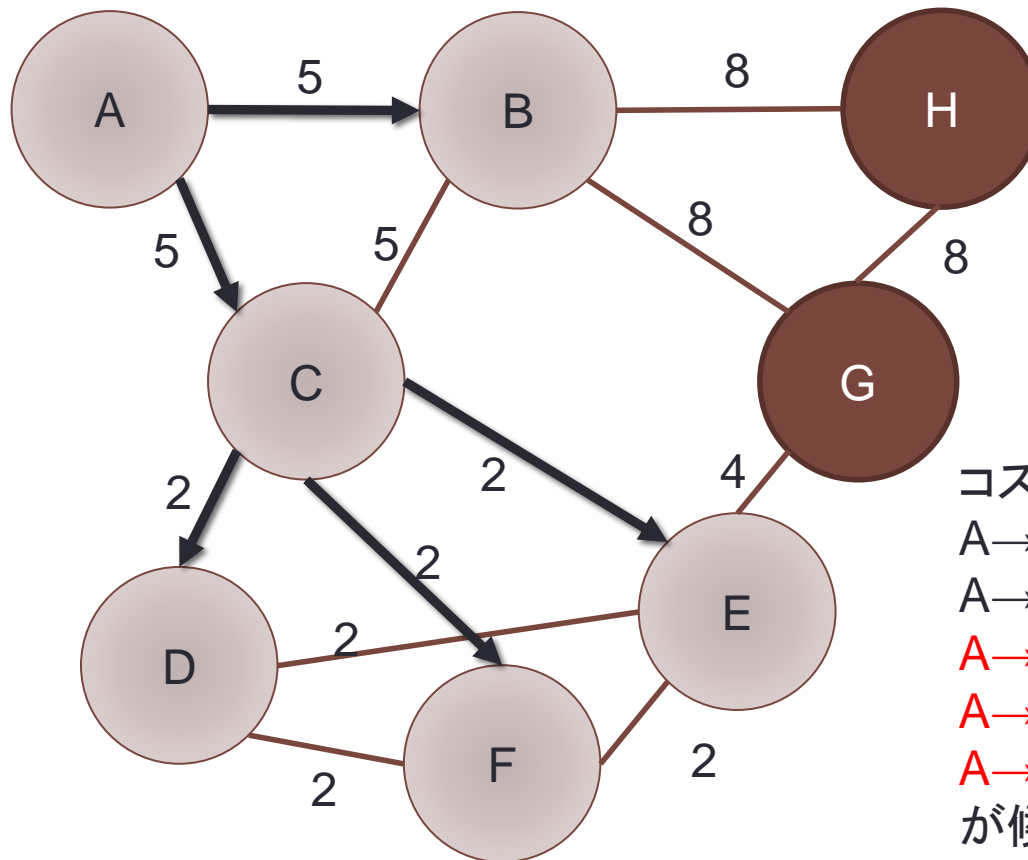
A→C→E(cost7)

A→C→F(cost7)

が候補となる

OSPF実例

例：Aの経路表作成(ダイクストラ法)



コストの小さいものを確定

A→B→H(cost13)

A→B→G(cost13)

A→C→D(cost7)

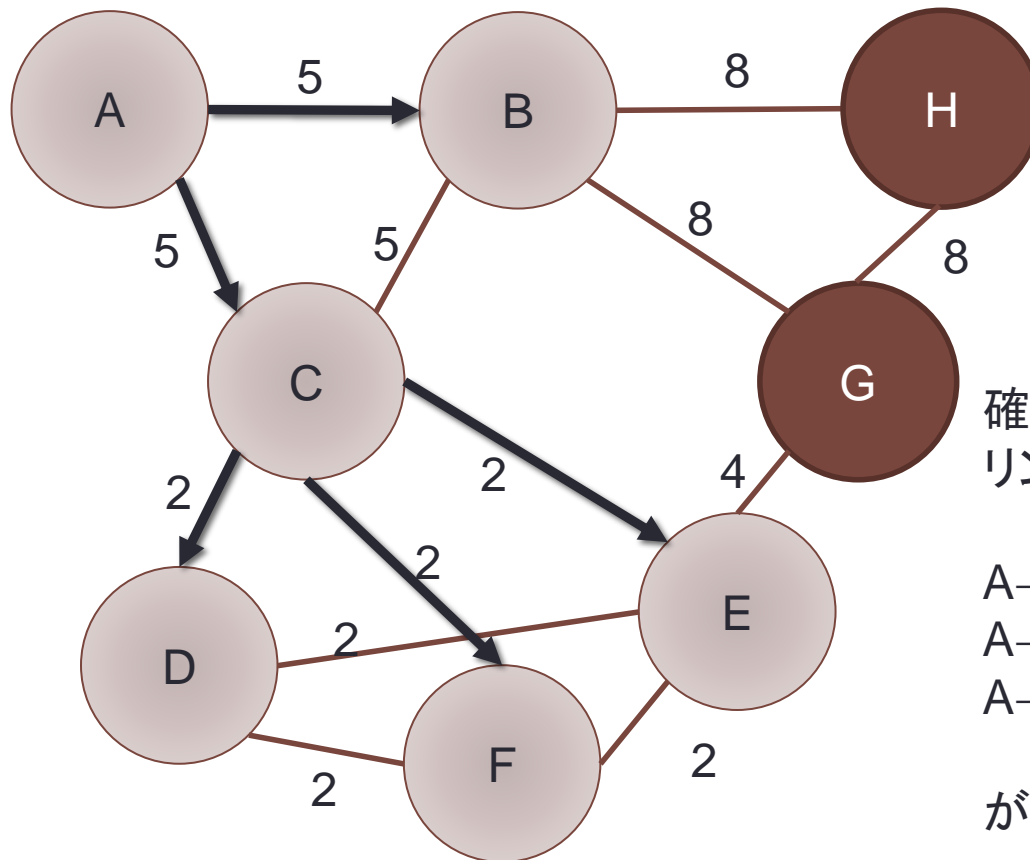
A→C→E(cost7)

A→C→F(cost7)

が候補となる

OSPF実例

例：Aの経路表作成(ダイクストラ法)



確定したD,E,Fからの
リンクをみる

A→B→H(cost13)

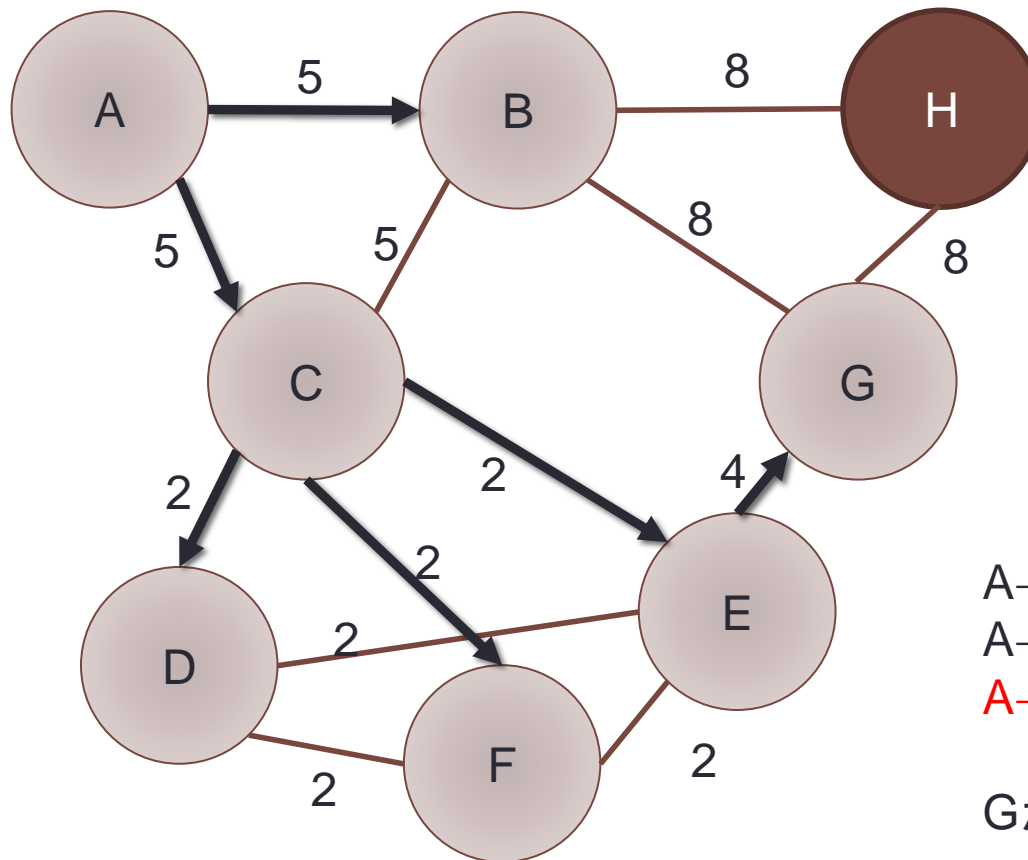
A→B→G(cost13)

A→C→E→G(cost11)

が候補となる

OSPF実例

例：Aの経路表作成(ダイクストラ法)



A→B→H(cost13)

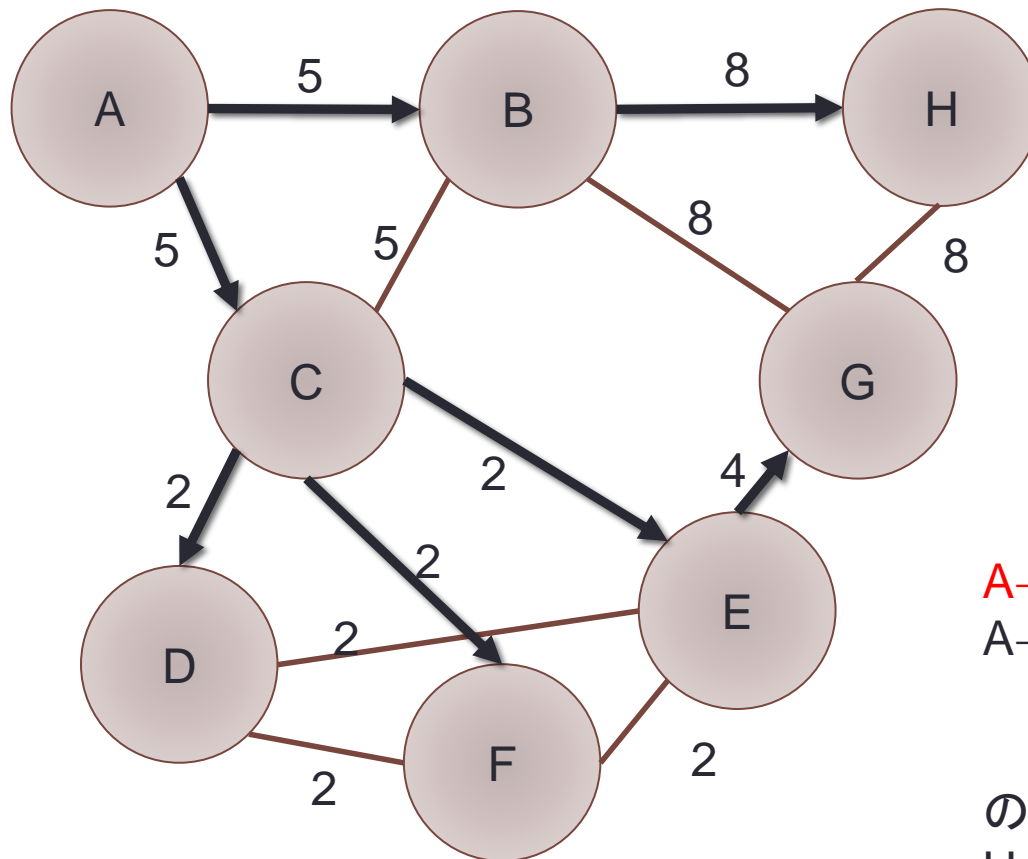
A→B→G(cost13)

A→C→E→G(cost11)

Gが確定する

OSPF実例

例：Aの経路表作成(ダイクストラ法)



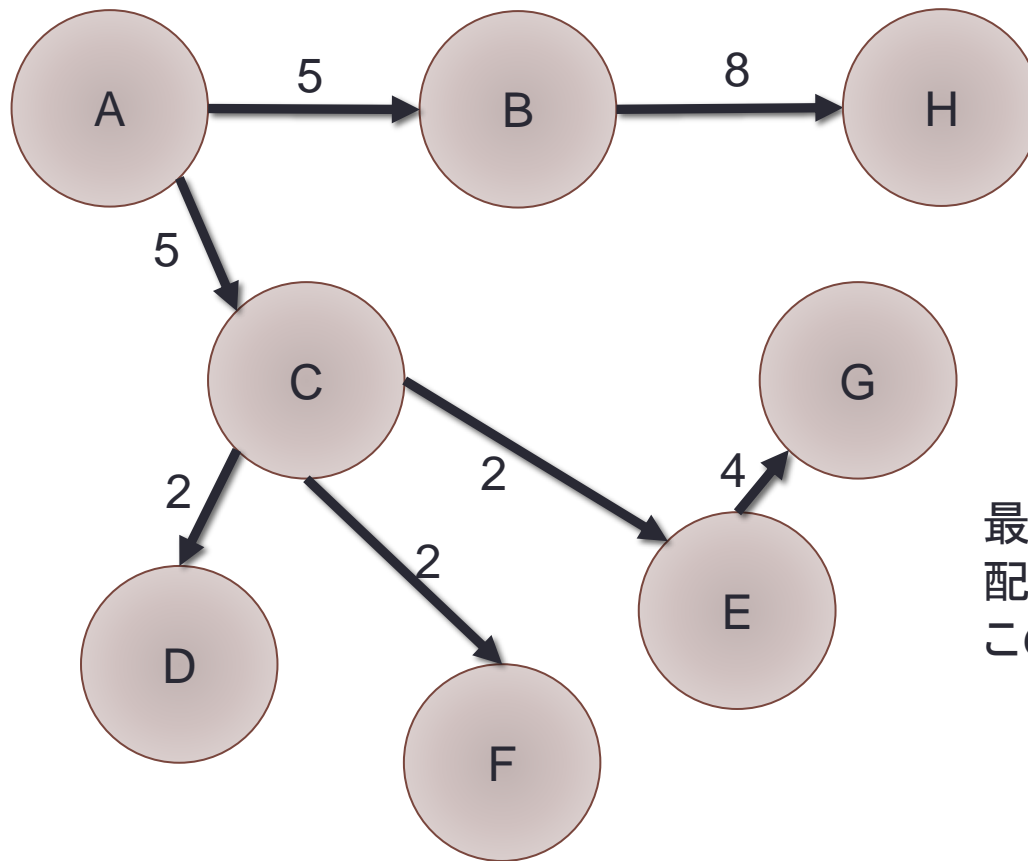
$A \rightarrow B \rightarrow H$ (cost 13)

$A \rightarrow C \rightarrow E \rightarrow G \rightarrow H$ (cost 19)

の2種類が残って
Hが確定する

OSPF実例

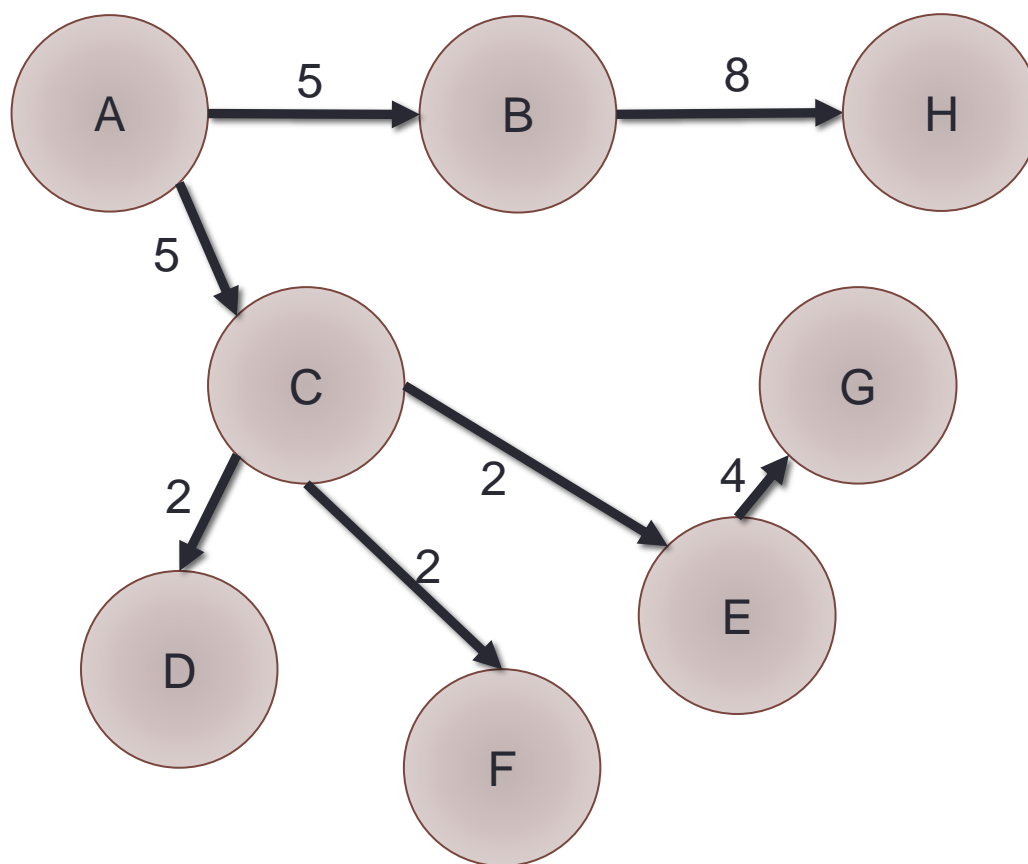
例：Aの経路表作成(ダイクストラ法)



最終的にAからの
配送経路は
このように確定する

OSPF実例

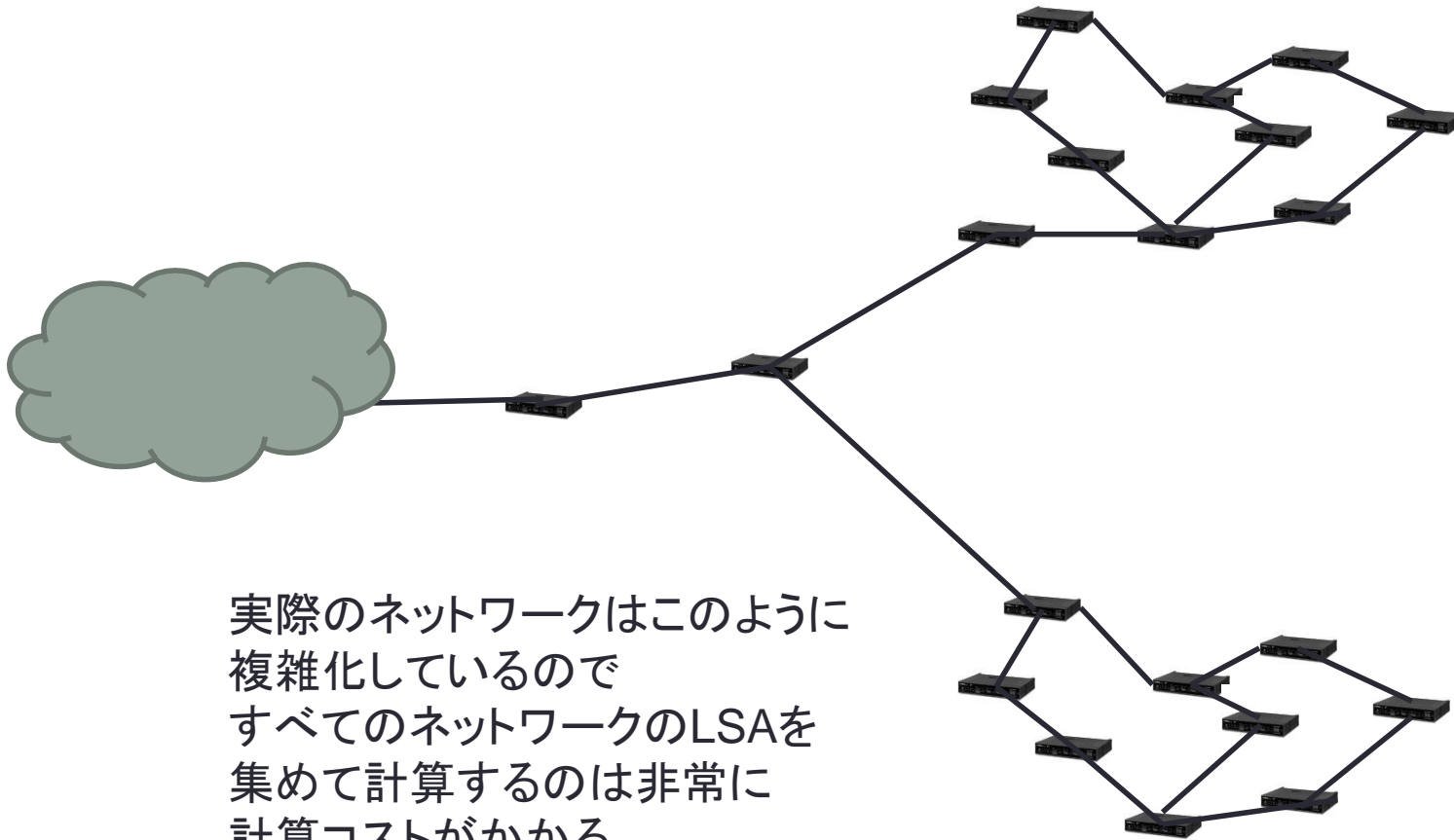
例：Aの経路表作成(ダイクストラ法)



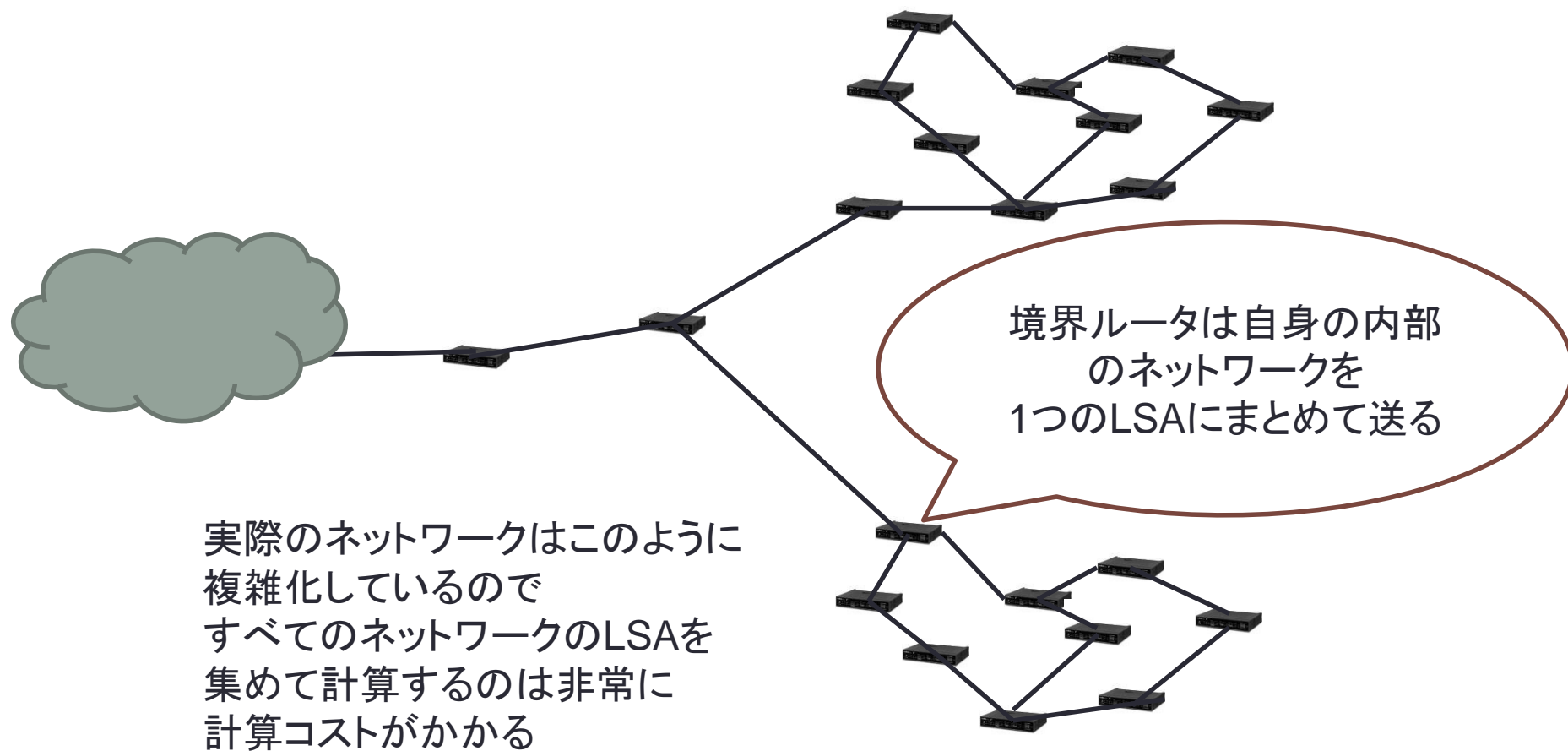
Aの経路表

宛先	次Hop
B	B
C	C
D	C
E	C
F	C
G	C
H	B

OSPF経路集約



OSPF経路集約



OSPFとRIPの違い

- RIP

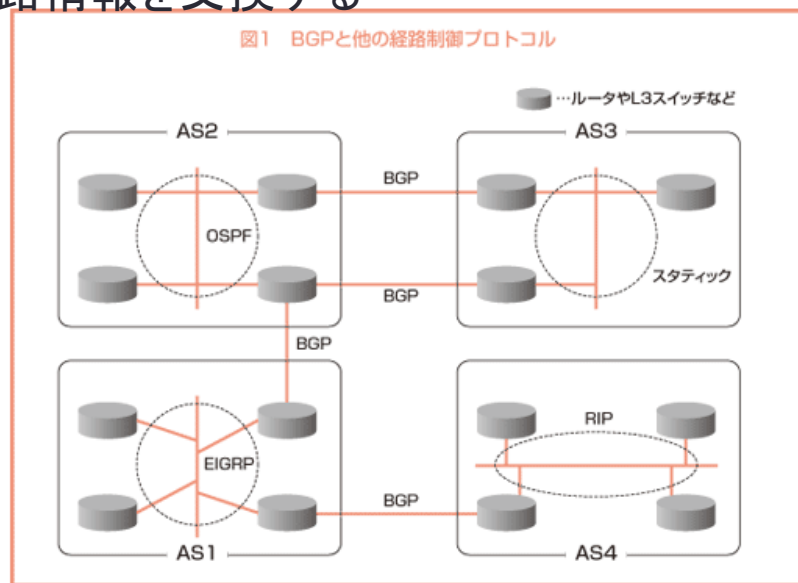
- ホップ数をメトリックとする
- 隣接ルータから得られた目的までの距離を元に最短経路を決定する
- 大規模ネットワークには向かない

- OSPF

- リンクのコストをメトリックとする
- LSAによって各ルータ, リンクの情報を得る
- 各ルータが自分で経路を計算する → RIPのようなループが起きない
- 経路集約が可能

BGP

- ルーティングプロトコル
 - 内部で使用するIGP(Interior Gateway Protocol)
 - RIP, OSPF
 - 外部で使用するEGP(Exterior Gateway Protocol)
 - AS(Autonomous System)間で経路情報を交換する
 - BGP



AS一覧

<https://www.nic.ad.jp/ja/ip/as-numbers.txt>

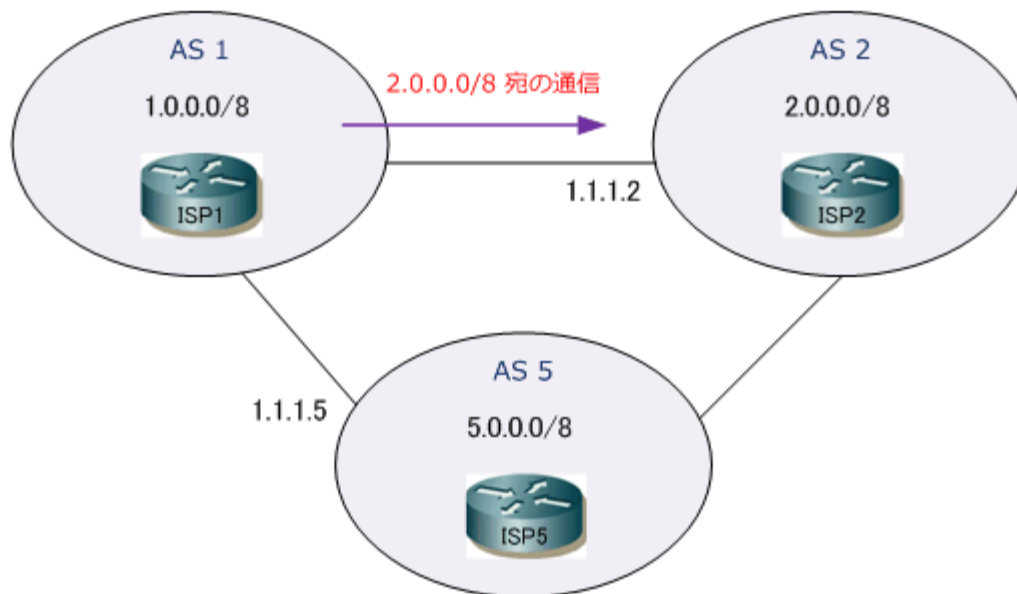
BGP

パスベクタ型ルーティングプロトコル

パスベクタ型ルーティングプロトコル

1.1.1.2 をネクストホップとするASパス [AS_PATH : AS2] ← ベストパスとなる最短経路

1.1.1.5 をネクストホップとするASパス [AS_PATH : AS5 AS2]



IPのルーティング以外の機能

まずはIPヘッダを見てみよう

0	4	8	16	19	31
バージョン Version ip_v	ヘッダ長 Header Length ip_hl	サービスタイプ Type Of Service ip_tos	パケット長 total length ip_len		
識別子 Identification ip_id			フラグ Flag	フラグメントオフセット Fragment Offset ip_off	
生存時間 Time to Live ip_ttl		プロトコル番号 protocol ip_p	ヘッダチェックサム Header Checksum ip_sum		
始点 IP アドレス Source IP Address ip_src					
終点 IP アドレス Destination IP Address ip_dst					

ヘッダ長

IPヘッダの長さ/4の値

サービスタイプ

サービス品質の定義(ほとんど使用されない)

識別子

IPフラグメンテーションにおいて利用される
IPパケットを識別するための番号

フラグとフラグメントオフセット

フラグメンテーションにおいて利用される、
特別なフラグ情報とオフセット数値(後述)

生存時間

IPパケットの寿命(可能ホップ数)

プロトコル番号

上位プロトコルの種類を示す番号を格納

各フィールドの詳細の前に・・・

Wiresharkというパケットモニタソフトウェアを導入してみよう

配布元 <http://www.wireshark.org/>



NEWS

Get Acquainted ▾

Get Help ▾

Develop ▾

Our Sponsor

SharkFest

Download Wireshark

The current stable release of Wireshark is 2.0.3. It supersedes all previous releases.

Stable Release (2.0.3)

- Windows Installer (64-bit)
- Windows Installer (32-bit)
- Windows PortableApps® (32-bit)
- OS X 10.6 and later Intel 64-bit .dmg
- OS X 10.6 and later Intel 32-bit .dmg
- Source Code

Old Stable Release (1.12.11)

Go Beyond with Riverbed Technology

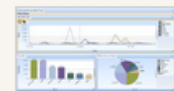
Riverbed is Wireshark's primary sponsor and provides our funding. They also make great products.

I have a lot of traffic...

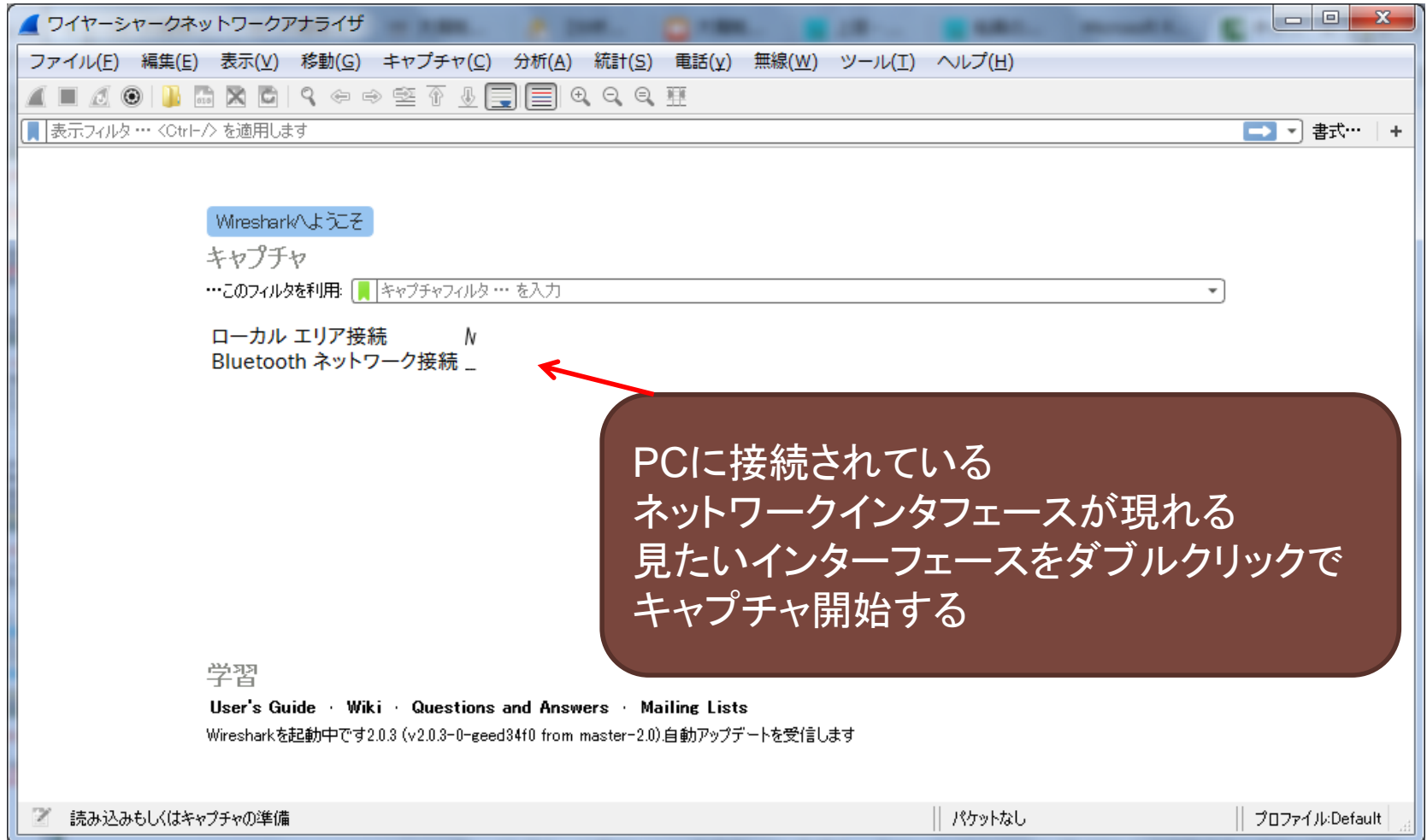
ANSWER: SteelCentral™ Packet Analyzer PE

\$29.95/yr

- Visually rich, powerful LAN analyzer
- Quickly access very large pcap files
- Professional, customizable reports
- Advanced triggers and alerts
- Fully integrated with Wireshark and AirPcap™



wireshark



HTTP(web用のプロトコル)を取る

フィルタをhttpにしてみる

No.	Time	Source	Destination	Protocol	Length	Info
2	0.617839	fe80::796a:5e60:2...	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
3	1.700172	fe80::bcbb:e121:9...	fe80::796a:5e60:2...	SSDP	456	HTTP/1.1 200 OK
4	1.964247	fe80::bcbb:e121:9...	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
11	2.971824	fe80::796a:5e60:2...	fe80::bcbb:e121:9...	SSDP	456	HTTP/1.1 200 OK
12	3.617809	fe80::796a:5e60:2...	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
37	4.668783	192.168.0.151	49.212.235.154	HTTP	749	GET /wireshark.html HTTP/1.1
54	4.685404	49.212.235.154	192.168.0.151	HTTP	234	HTTP/1.1 304 Not Modified

パケット単位で1つずつ表示される

Frame 37: 749 bytes on wire (5992 bits), 749 bytes captured (5992 bits) on interface 0
Ethernet II, Src: WistronI_c2:36:e1 (f0:de:f1:c2:36:e1), Dst: Century_7a:0b:49 (00:80:6d:7a:0b:49)
Internet Protocol Version 4, Src: 192.168.0.151, Dst: 49.212.235.154
Transmission Control Protocol, Src Port: 53781 (53781), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 695
Hypertext Transfer Protocol

上で選んだパケットの各層のプロトコルと主な内容が表示される

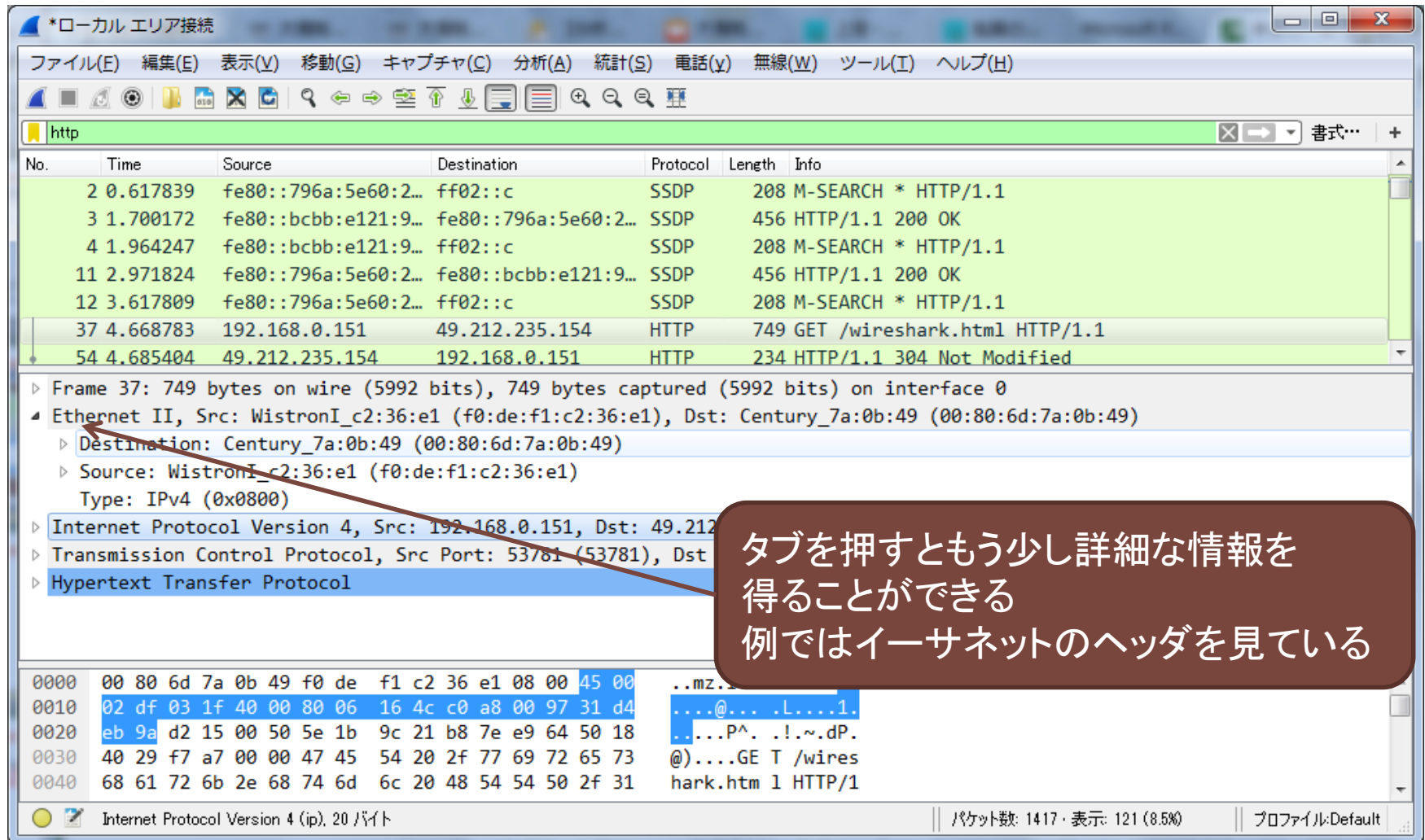
0000 00 80 6d 7a 0b 49 f0 de f1 c2 36 e1 08 00 45 00 ..mz.I...6...E.
0010 02 df 03 1f 40 00 80 06 16 4c c0 a8 00 97 31 d4@...L...1.
0020 eb 9a d2 15 00 50 5e 1b 9c 21 b8 7e e9 64 50 18P^.!~.dP.
0030 40 29 f7 a7 00 00 47 45 54 20 2f 77 69 72 65 73 @)....GE T /wires
0040 68 hark.htm 1 HTTP/1

上で選んだプロトコルの該当箇所が青色で表示される

パケットの中身が16進数とASCII文字列で表示される

パケット数: 1417 · 表示: 121 (8.5%) プロファイル: Default

HTTP(web用のプロトコル)を取る



The screenshot shows the Wireshark network protocol analyzer interface. The packet list on the left shows several SSDP and HTTP packets. Packet 37 is selected, showing details for Ethernet II, Internet Protocol Version 4, Transmission Control Protocol, and Hypertext Transfer Protocol. A callout box points to the 'Destination' field in the Ethernet II section, explaining that clicking on a protocol layer provides more detailed information.

タブを押すともう少し詳細な情報を
得ることができる
例ではイーサネットのヘッダを見ている

No.	Time	Source	Destination	Protocol	Length	Info
2	0.617839	fe80::796a:5e60:2...	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
3	1.700172	fe80::bcbb:e121:9...	fe80::796a:5e60:2...	SSDP	456	HTTP/1.1 200 OK
4	1.964247	fe80::bcbb:e121:9...	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
11	2.971824	fe80::796a:5e60:2...	fe80::bcbb:e121:9...	SSDP	456	HTTP/1.1 200 OK
12	3.617809	fe80::796a:5e60:2...	ff02::c	SSDP	208	M-SEARCH * HTTP/1.1
37	4.668783	192.168.0.151	49.212.235.154	HTTP	749	GET /wireshark.html HTTP/1.1
54	4.685404	49.212.235.154	192.168.0.151	HTTP	234	HTTP/1.1 304 Not Modified

Frame 37: 749 bytes on wire (5992 bits), 749 bytes captured (5992 bits) on interface 0
Ethernet II, Src: WistronI_c2:36:e1 (f0:de:f1:c2:36:e1), Dst: Century_7a:0b:49 (00:80:6d:7a:0b:49)
Destination: Century_7a:0b:49 (00:80:6d:7a:0b:49)
Source: WistronI_c2:36:e1 (f0:de:f1:c2:36:e1)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.0.151, Dst: 49.212.235.154
Transmission Control Protocol, Src Port: 53781 (53781), Dst Port: 80 (80)
Hypertext Transfer Protocol

0000 00 80 6d 7a 0b 49 f0 de f1 c2 36 e1 08 00 45 00 ..mz..
0010 02 df 03 1f 40 00 80 06 16 4c c0 a8 00 97 31 d4@... .L....1.
0020 eb 9a d2 15 00 5e 1b 9c 21 b8 7e e9 64 50 18P^ .!.~.dP.
0030 40 29 f7 a7 00 00 47 45 54 20 2f 77 69 72 65 73 @)....GE T /wires
0040 68 61 72 6b 2e 68 74 6d 6c 20 48 54 54 50 2f 31 hark.htm l HTTP/1

Internet Protocol Version 4 (ip), 20 バイト | パケット数: 1417 · 表示: 121 (8.5%) | プロファイル: Default

IPのルーティング以外の機能

さて、本題に戻ってIPヘッダ

0	4	8	16	19	31
バージョン Version ip_v	ヘッダ長 Header Length ip_hl	サービスタイプ Type Of Service ip_tos	パケット長 total length ip_len		
識別子 Identification ip_id			フラグ Flag	フラグメントオフセット Fragment Offset ip_off	
生存時間 Time to Live ip_ttl		プロトコル番号 protocol ip_p	ヘッダチェックサム Header Checksum ip_sum		
始点 IP アドレス Source IP Address ip_src					
終点 IP アドレス Destination IP Address ip_dst					

ヘッダ長

IPヘッダの長さ/4の値

サービスタイプ

サービス品質の定義(ほとんど使用されない)

識別子

IPフラグメンテーションにおいて利用される
IPパケットを識別するための番号

フラグとフラグメントオフセット

フラグメンテーションにおいて利用される、
特別なフラグ情報とオフセット数値(後述)

生存時間

IPパケットの寿命(可能ホップ数)

プロトコル番号

上位プロトコルの種類を示す番号を格納

IPのルーティング以外の機能

さて、本題に戻ってIPヘッダ

0	4	8	16	19	31
バージョン Version ip_v	ヘッダ長 Header Length ip_hl	サービスタイプ Type Of Service ip_tos	パケット長 total length ip_len		
識別子 Identification ip_id			フラグ Flag	フラグメントオフセット Fragment Offset ip_off	
生存時間 Time to Live ip_ttl	プロトコル番号 protocol ip_p		ヘッダチェックサム Header Checksum ip_sum		
始点 IP アドレス Source IP Address ip_src					
終点 IP アドレス Destination IP Address ip_dst					

ヘッダ長

IPヘッダの長さ/4の値

サービスタイプ

サービス品質の定義(ほとんど使用されない)

識別子

IPフラグメンテーションにおいて利用される
IPパケットを識別するための番号

フラグとフラグメントオフセット

フラグメンテーションにおいて利用される、
特別なフラグ情報とオフセット数値(後述)

生存時間

IPパケットの寿命(可能ホップ数)

プロトコル番号

上位プロトコルの種類を示す番号を格納

IPのルーティング以外の機能

さて、本題に戻ってIPヘッダ

0	4	8	16	19	31
バージョン Version ip_v	ヘッダ長 Header Length ip_hl	サービスタイプ Type Of Service ip_tos	パケット長 total length ip_len		
識別子 Identification ip_id			フラグ Flag	フラグメントオフセット Fragment Offset ip_off	
生存時間 Time to Live ip_ttl	プロトコル番号 protocol ip_p		ヘッダチェックサム Header Checksum ip_sum		
始点IPアドレス Source IP Address ip_src					
終点IPアドレス Destination IP Address ip_dst					

ヘッダ長

IPヘッダの長さ/4の値

サービスタイプ

サービス品質の定義(ほとんど使用されない)

識別子

IPフラグメンテーションにおいて利用される
IPパケットを識別するための番号

フラグとフラグメントオフセット

フラグメンテーションにおいて利用される、
特別なフラグ情報とオフセット数値(後述)

生存時間

IPパケットの寿命(可能ホップ数)

プロトコル番号

上位プロトコルの種類を示す番号を格納

IPのルーティング以外の機能

さて、本題に戻ってIPヘッダ

0	4	8	16	19	31
バージョン Version ip_v	ヘッダ長 Header Length ip_hl	サービスタイプ Type Of Service ip_tos	パケット長 total length ip_len		
識別子 Identification ip_id			フラグ Flag	フラグメントオフセット Fragment Offset ip_off	
生存時間 Time to Live ip_ttl		プロトコル番号 protocol ip_p	ヘッダチェックサム Header Checksum ip_sum		
始点 IP アドレス Source IP Address ip_src					
終点 IP アドレス Destination IP Address ip_dst					

ヘッダ長

IPヘッダの長さ/4の値

サービスタイプ

サービス品質の定義(ほとんど使用されない)

識別子

IPフラグメンテーションにおいて利用される
IPパケットを識別するための番号

フラグとフラグメントオフセット

フラグメンテーションにおいて利用される、
特別なフラグ情報とオフセット数値(後述)

生存時間

IPパケットの寿命(可能ホップ数)

プロトコル番号

上位プロトコルの種類を示す番号を格納

識別子、フラグ、フラグメントオフセット

- IPの長さは $2^{16}=65535$ オクテットまで(パケット長が16ビット)
- データリンクのフレームの長さはそれ以下のことが多い
 - イーサネットは1500オクテットまで



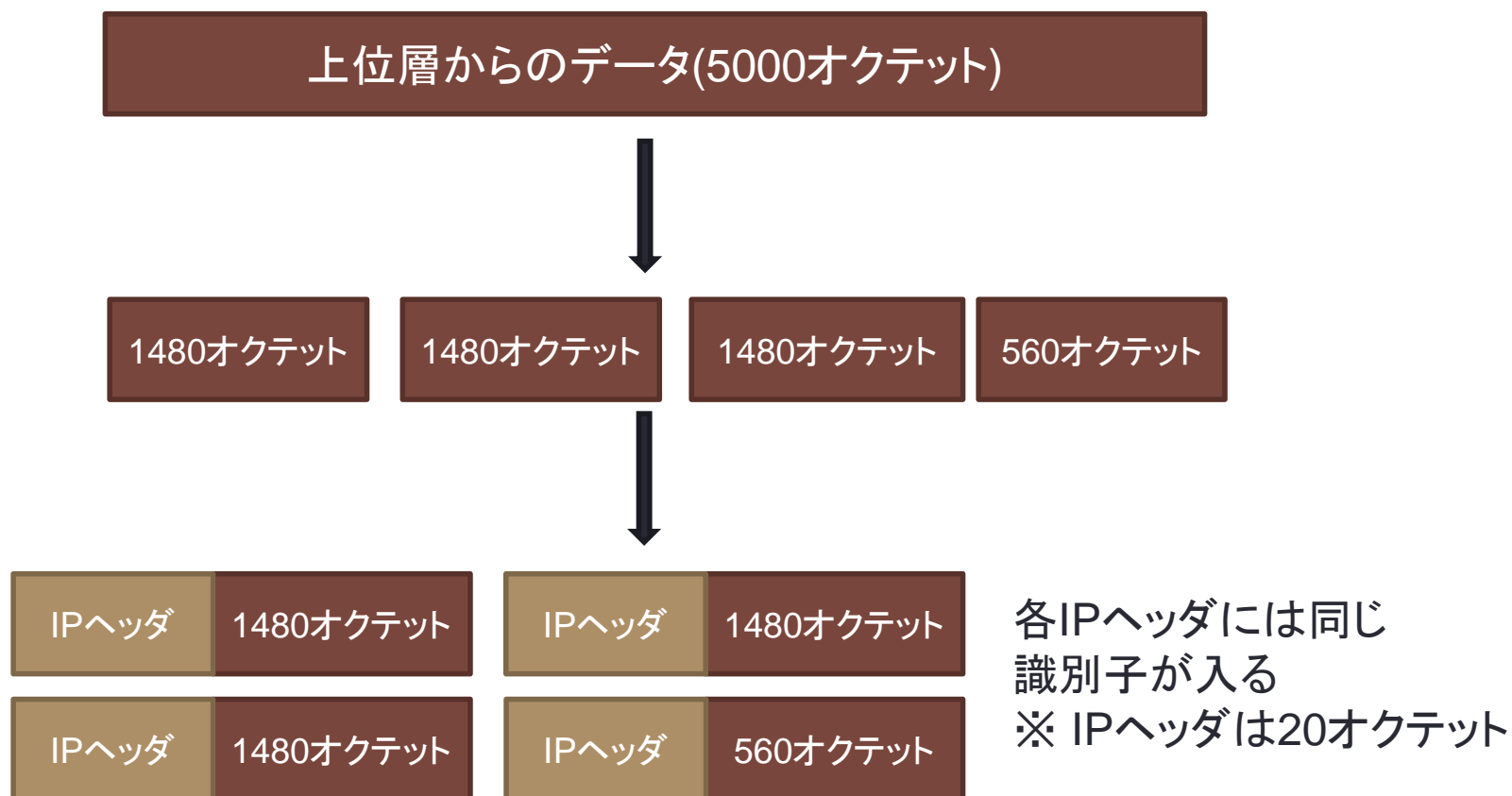
- データをIP層で分割して複数のイーサネットフレームにする必要がある



- 識別子、フラグ、フラグメントオフセットはIPパケットを送信側で分割して受信側で復元する機能を提供する

識別子

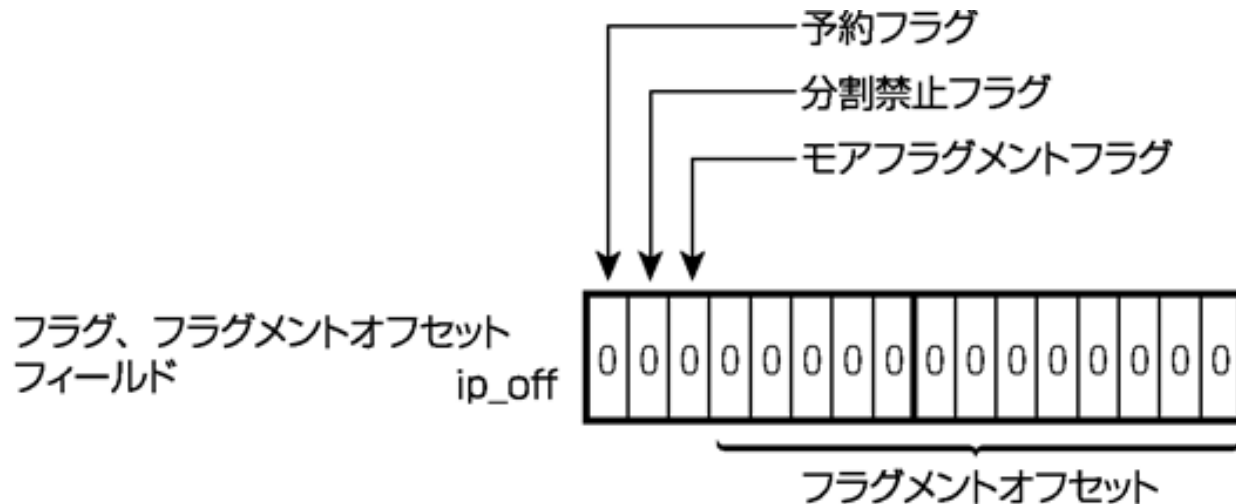
- 16ビットの数値
- 分割データのヘッダに同じ識別子を記入する



フラグとフラグメントオフセット

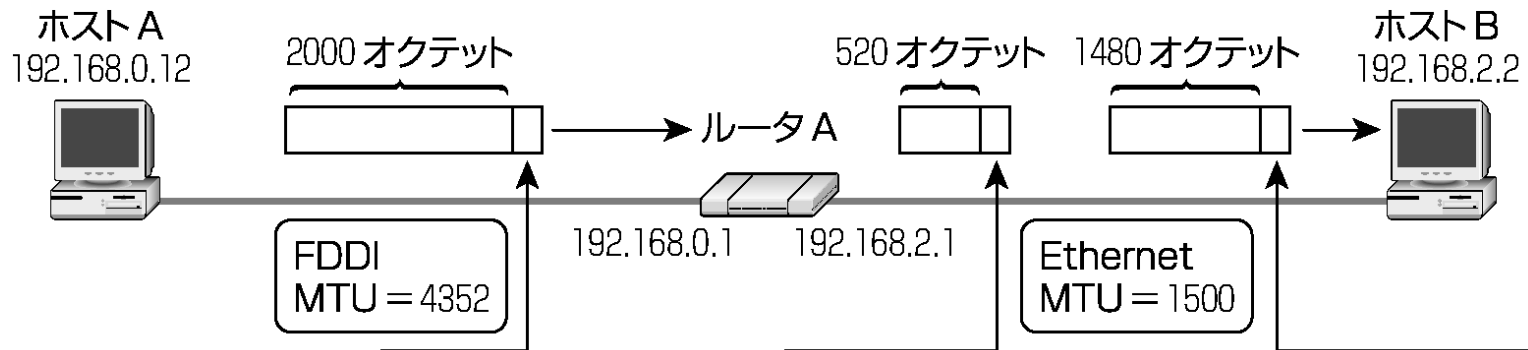
先頭の3ビットがフラグ

残りの13ビットがオフセットとして使われる



- 予約フラグは使われない
- 分割禁止はこのIPパケットは分割してはいけないことを示す
- モアフラグメントは, このIPパケットの後にもデータがあることを示す
- オフセットは分割データのどこまでのデータかを示す(単位8オクテット)

IPフラグメント(分割)の例



IP ヘッダ

V:4	HL:5	TOS:	TL:	2020	
ID:	12345		0	0	FO: 0
TTL:	P:	CKSUM:			
SRC IP:		192.168.0.12			
DST IP:		192.168.2.2			

IP ヘッダ

V:4	HL:5	TOS:	TL:	1500	
ID:	12345	0	0	M	FO: 0
TTL:	P:	CKSUM:			
SRC IP:	192.168.0.12				
DST IP:	192.168.2.2				

IP ヘッダ

V:4	HL:5	TOS:	TL:540			
ID:12345			0	0	0	FO:185
TTL:		P:	CKSUM:			
SRC IP:			192.168.0.12			
DST IP:			192.168.2.2			

MTU(Maximum Transmission Unit)
データリンクごとの最大フレーム長

FO
フラグメントオフセットは8オクテット単位

IPのルーティング以外の機能

0	4	8	16	19	31
バージョン Version ip_v	ヘッダ長 Header Length ip_hl	サービスタイプ Type Of Service ip_tos	パケット長 total length ip_len		
識別子 Identification ip_id			フラグ Flag	フラグメントオフセット Fragment Offset ip_off	
生存時間 Time to Live ip_ttl		プロトコル番号 protocol ip_p	ヘッダチェックサム Header Checksum ip_sum		
始点 IP アドレス Source IP Address ip_src					
終点 IP アドレス Destination IP Address ip_dst					

ヘッダ長

IPヘッダの長さ/4の値

サービスタイプ

サービス品質の定義(ほとんど使用されない)

識別子

IPフラグメンテーションにおいて利用される
IPパケットを識別するための番号

フラグとフラグメントオフセット

フラグメンテーションにおいて利用される、
特別なフラグ情報とオフセット数値(後述)

生存時間

IPパケットの寿命(可能ホップ数)

プロトコル番号

上位プロトコルの種類を示す番号を格納

IPのルーティング以外の機能

0	4	8	16	19	3
バージョン Version ip_v	ヘッダ長 Header Length ip_hl	サービスタイプ Type Of Service ip_tos	パケット長 total length ip_len		
識別子 Identification ip_id			フラグ Flag	フラグメントオフセット Fragment Offset ip_off	
生存時間 Time to Live ip_ttl	プロトコル番号 protocol ip_p		←	ヘッダチェックサム Header Checksum ip_sum	
始点 IP アドレス Source IP Address ip_src					
終点 IP アドレス Destination IP Address ip_dst					

TCP:6
UDP:17
など

ヘッダ長

IPヘッダの長さ/4の値

サービスタイプ

サービス品質の定義(ほとんど使用されない)

識別子

IPフラグメンテーションにおいて利用される
IPパケットを識別するための番号

フラグとフラグメントオフセット

フラグメンテーションにおいて利用される、
特別なフラグ情報とオフセット数値(後述)

生存時間

IPパケットの寿命(可能ホップ数)

プロトコル番号

上位プロトコルの種類を示す番号を格納

IPのルーティング以外の機能

0	4	8	16	19	31
バージョン Version ip_v	ヘッダ長 Header Length ip_hl	サービスタイプ Type Of Service ip_tos	パケット長 total length ip_len		
識別子 Identification ip_id			フラグ Flag	フラグメントオフセット Fragment Offset ip_off	
生存時間 Time to Live ip_ttl	プロトコル番号 protocol ip_p		ヘッダチェックサム Header Checksum ip_sum		
始点 IP アドレス Source IP Address ip_src					
終点 IP アドレス Destination IP Address ip_dst					

ヘッダチェックサム

ヘッダの誤りを検出する機構である

チェックサムの計算方法については上位層でも
同じようなものがあるので、そちらで解説する

今回のまとめ

- ルーティング
 - 静的ルーティングと動的ルーティング
 - RIPはホップ, OSPFはコストで動的計算を行う
 - RIPは隣接ルータの情報を元に, OSPFはルータ自身が経路計算をする
- IPヘッダ
 - 送信アドレスと受信アドレスがある(前回までの話)
 - IPはデータの分割と復元をサポートしている
 - 分割と復元には識別子, フラグ, フラグメントを用いる

質問あればどうぞ

次回はネットワーク層(つづき)！

～ネットワーク層のもう1つのプロトコルと
次世代IPについて～