超ざっくり

SRv6入門

浅間正和@有限会社銀座堂

ENOG55(2019/02/22)

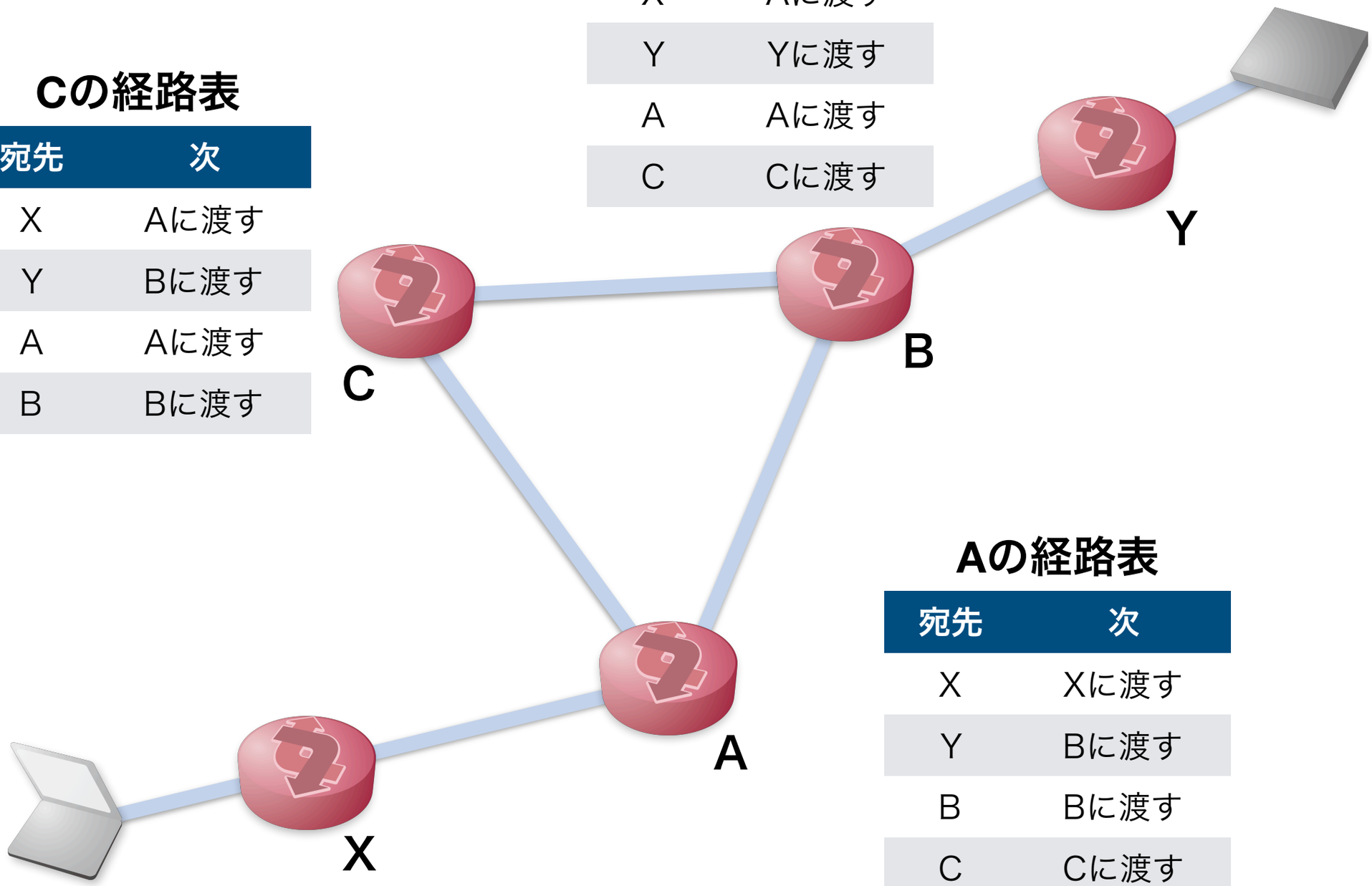
こんなときどうする？

Bの経路表

宛先	次
X	Aに渡す
Y	Yに渡す
A	Aに渡す
C	Cに渡す

Cの経路表

宛先	次
X	Aに渡す
Y	Bに渡す
A	Aに渡す
B	Bに渡す



Aの経路表

宛先	次
X	Xに渡す
Y	Bに渡す
B	Bに渡す
C	Cに渡す

こんなときどうする？

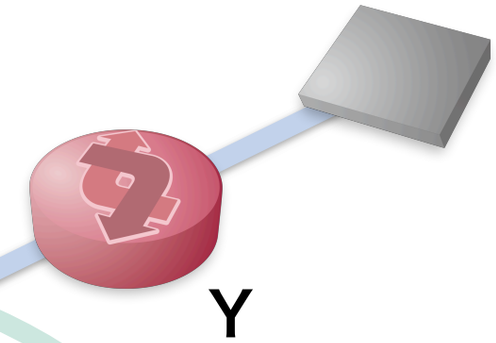
Bの経路表

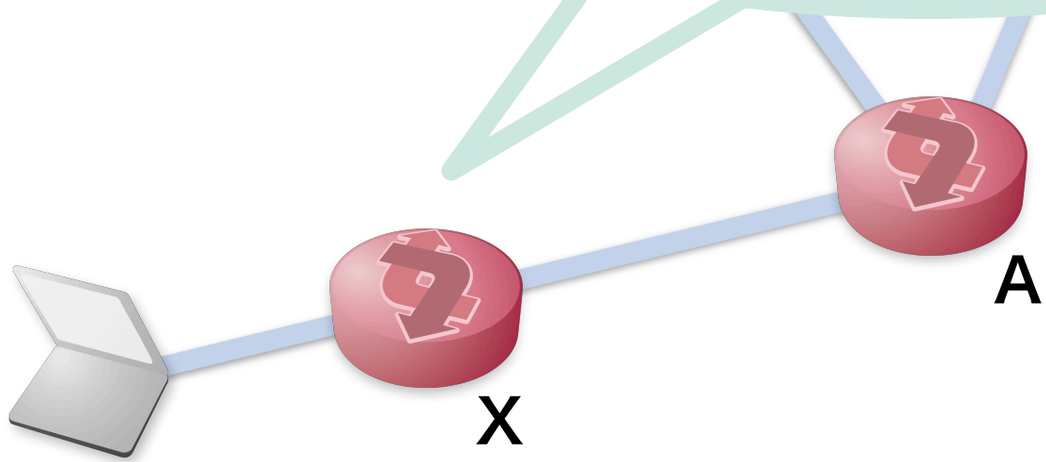
宛先	次
X	Aに渡す
Y	Yに渡す
A	Aに渡す

Cの経路表

宛先	次
X	Aに渡す
Y	Bに渡
A	Aに
B	Bに

サーバと通信するとき
電話は低遅延の経路
ウェブは広帯域の経路
を通るようにしたいなあ





Aの経路表

宛先	次
X	Xに渡す
Y	Bに渡す
B	Bに渡す
C	Cに渡す

Multi-Topology(MT)？

Bの広帯域経路表

宛先	次
X	
Y	
A	
C	

NEW

Bの低遅延経路表

宛先	次
X	Aに渡す
Y	Yに渡す
A	Aに渡す
C	Cに渡す

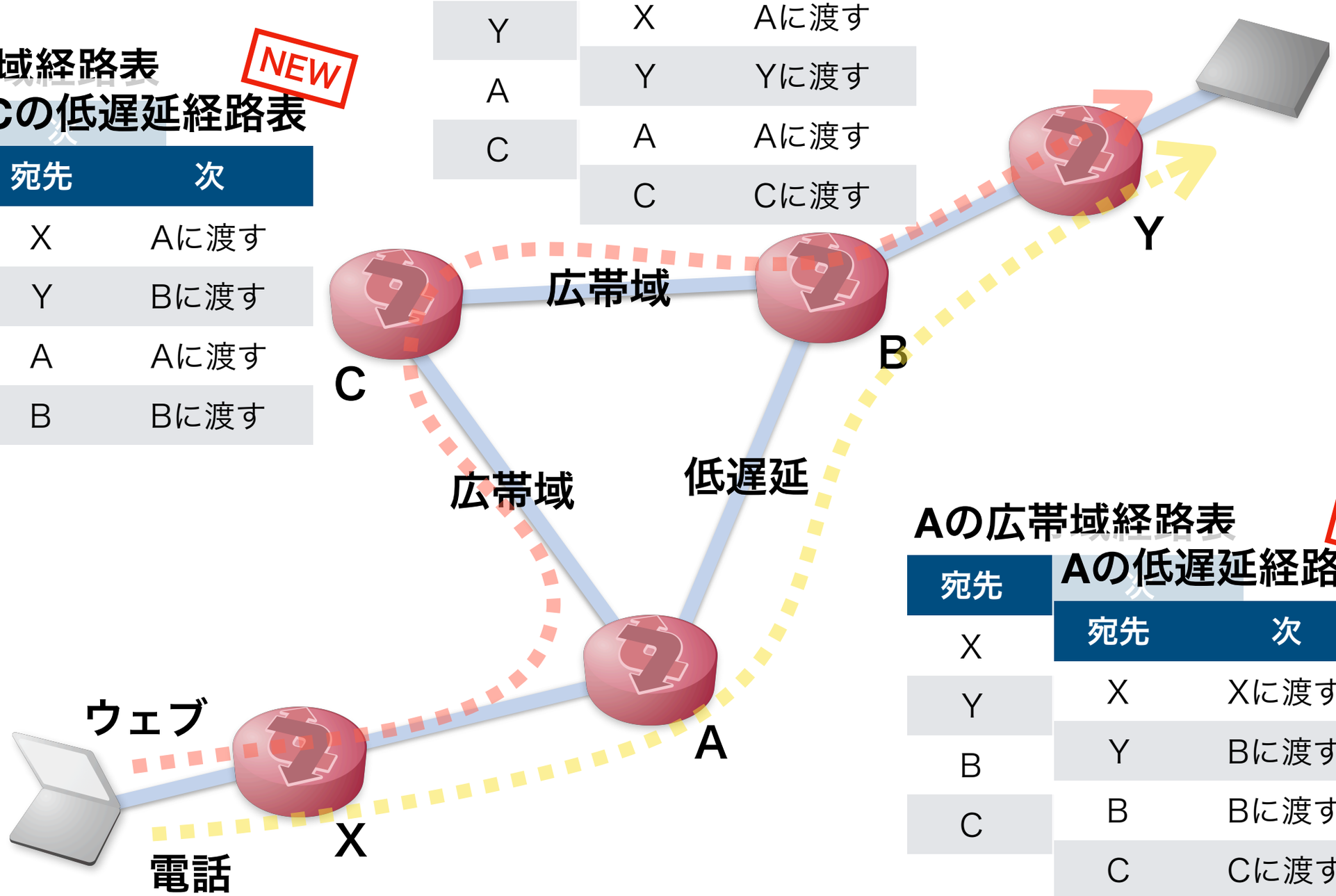
Cの広帯域経路表

宛先	次
X	
Y	
A	
B	

NEW

Cの低遅延経路表

宛先	次
X	Aに渡す
Y	Bに渡す
A	Aに渡す
B	Bに渡す



Aの広帯域経路表

宛先	次
X	
Y	
B	
C	

NEW

Aの低遅延経路表

宛先	次
X	Xに渡す
Y	Bに渡す
B	Bに渡す
C	Cに渡す

MPLS？

CのIP経路表

宛先	次
X	
Y	
A	
B	

NEW

CのMPLS経路表

宛先	次
L1	Aに渡す
L2	Bに渡す
L3	Aに渡す
L4	Bに渡す

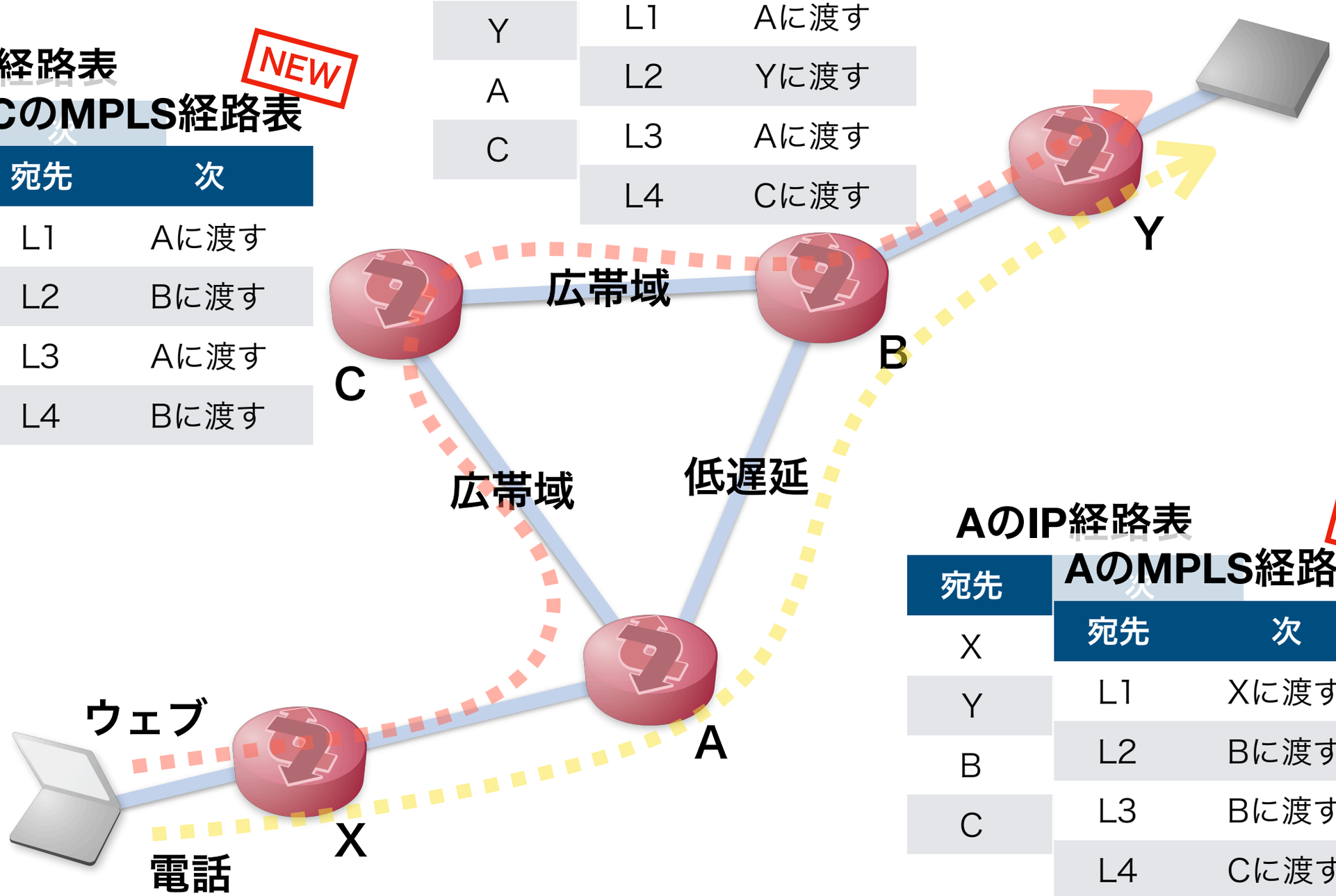
BのIP経路表

宛先	次
X	
Y	
A	
C	

NEW

BのMPLS経路表

宛先	次
L1	Aに渡す
L2	Yに渡す
L3	Aに渡す
L4	Cに渡す



AのIP経路表

宛先	次
X	
Y	
B	
C	

NEW

AのMPLS経路表

宛先	次
L1	Xに渡す
L2	Bに渡す
L3	Bに渡す
L4	Cに渡す

こんなときどうする？再び

BのIP経路表

宛先	次
X	
Y	
A	
C	

BのMPLS経路表

宛先	次
L1	Aに渡す
L2	Yに渡す
L3	Aに渡す

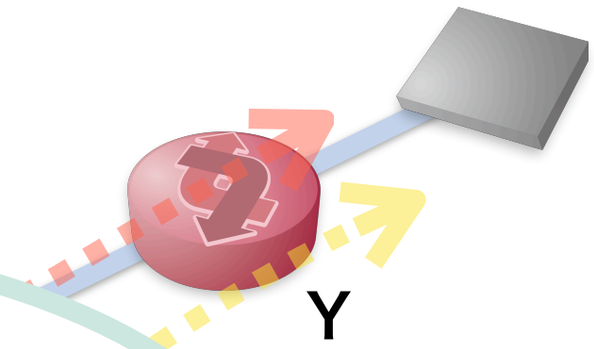
CのIP経路表

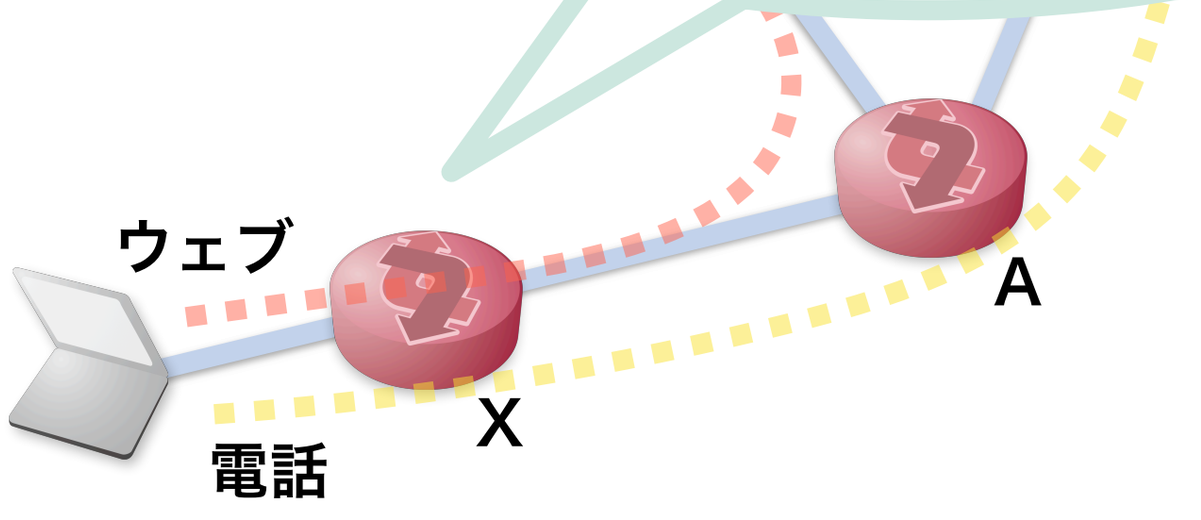
宛先	次
X	
Y	
A	
B	

CのMPLS経路表

宛先	次
L1	Aに渡す
L2	Bに渡す
L3	Aに渡す
L4	Bに渡す

サーバと通信するとき
特定のクライアントからのときは
ファイアウォール
を通るようにしたいなあ





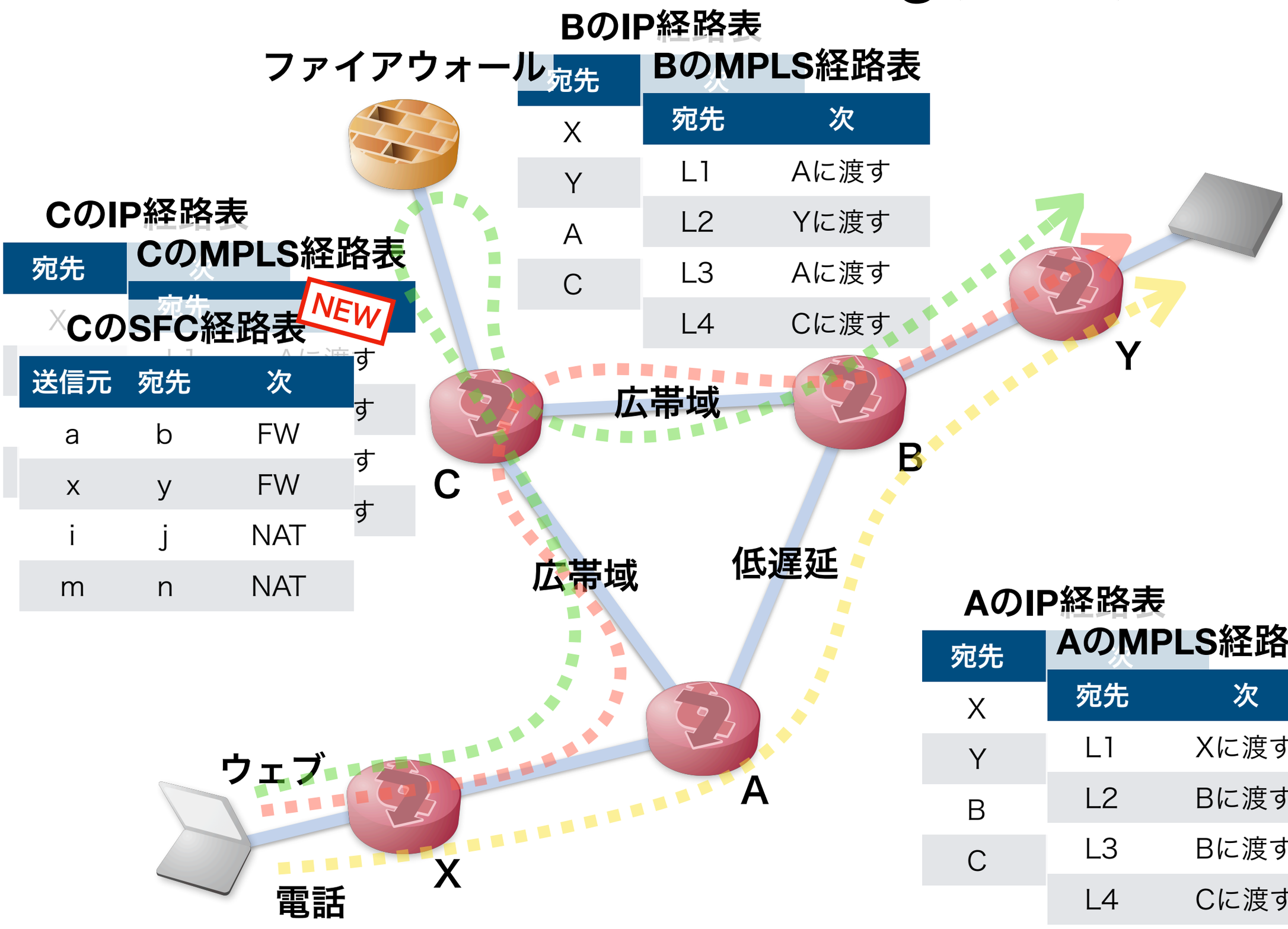
AのIP経路表

宛先	次
X	
Y	
B	
C	

AのMPLS経路表

宛先	次
L1	Xに渡す
L2	Bに渡す
L3	Bに渡す
L4	Cに渡す

Service Function Chaining(SFC)？



CのIP経路表

宛先	次
X	
Y	
A	
B	

CのMPLS経路表

宛先	次

NEW

CのSFC経路表

送信元	宛先	次
a	b	FW
x	y	FW
i	j	NAT
m	n	NAT

BのIP経路表

宛先	次
X	
Y	
A	
C	

BのMPLS経路表

宛先	次
L1	Aに渡す
L2	Yに渡す
L3	Aに渡す
L4	Cに渡す

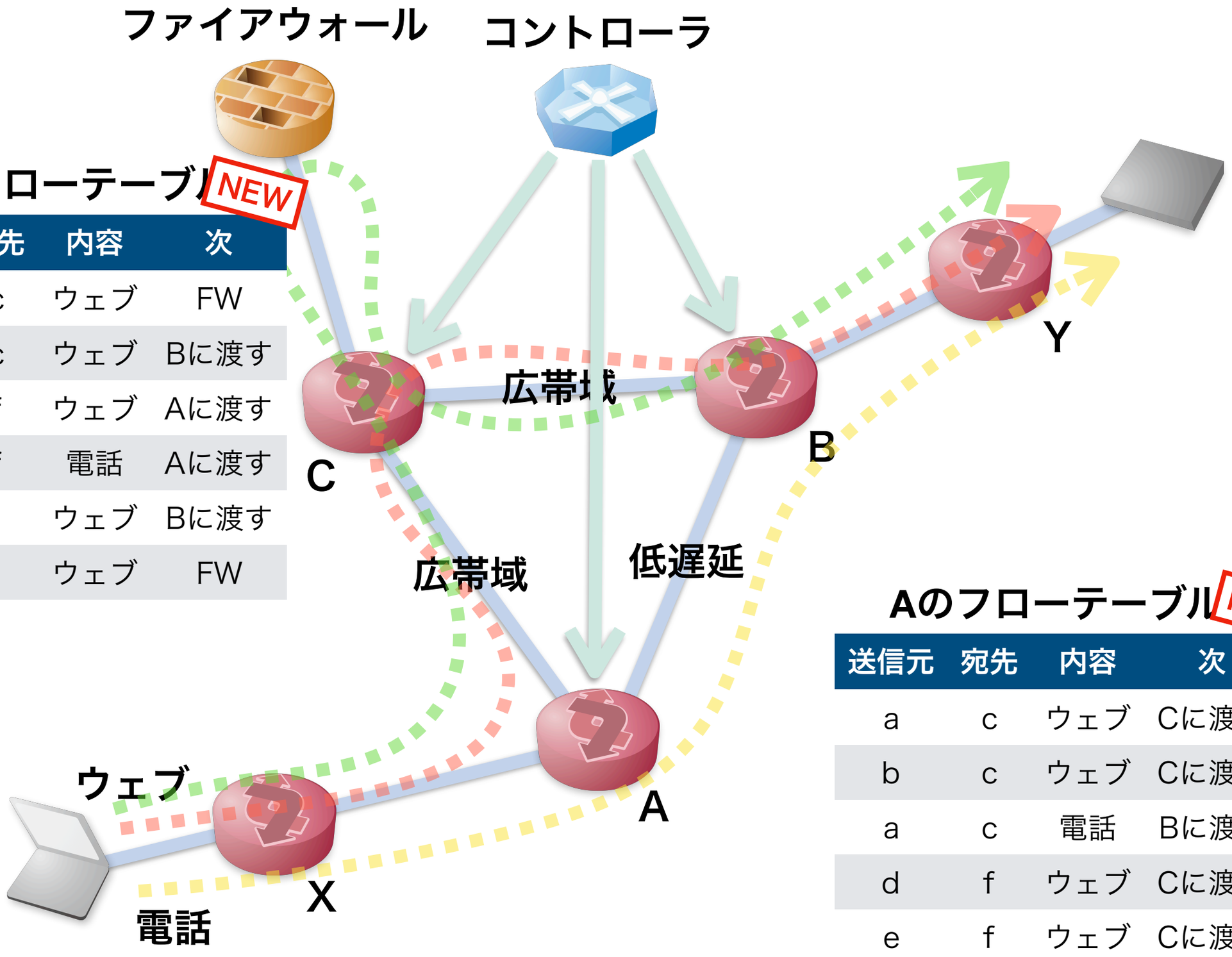
AのIP経路表

宛先	次
X	
Y	
B	
C	

AのMPLS経路表

宛先	次
L1	Xに渡す
L2	Bに渡す
L3	Bに渡す
L4	Cに渡す

OpenFlow？



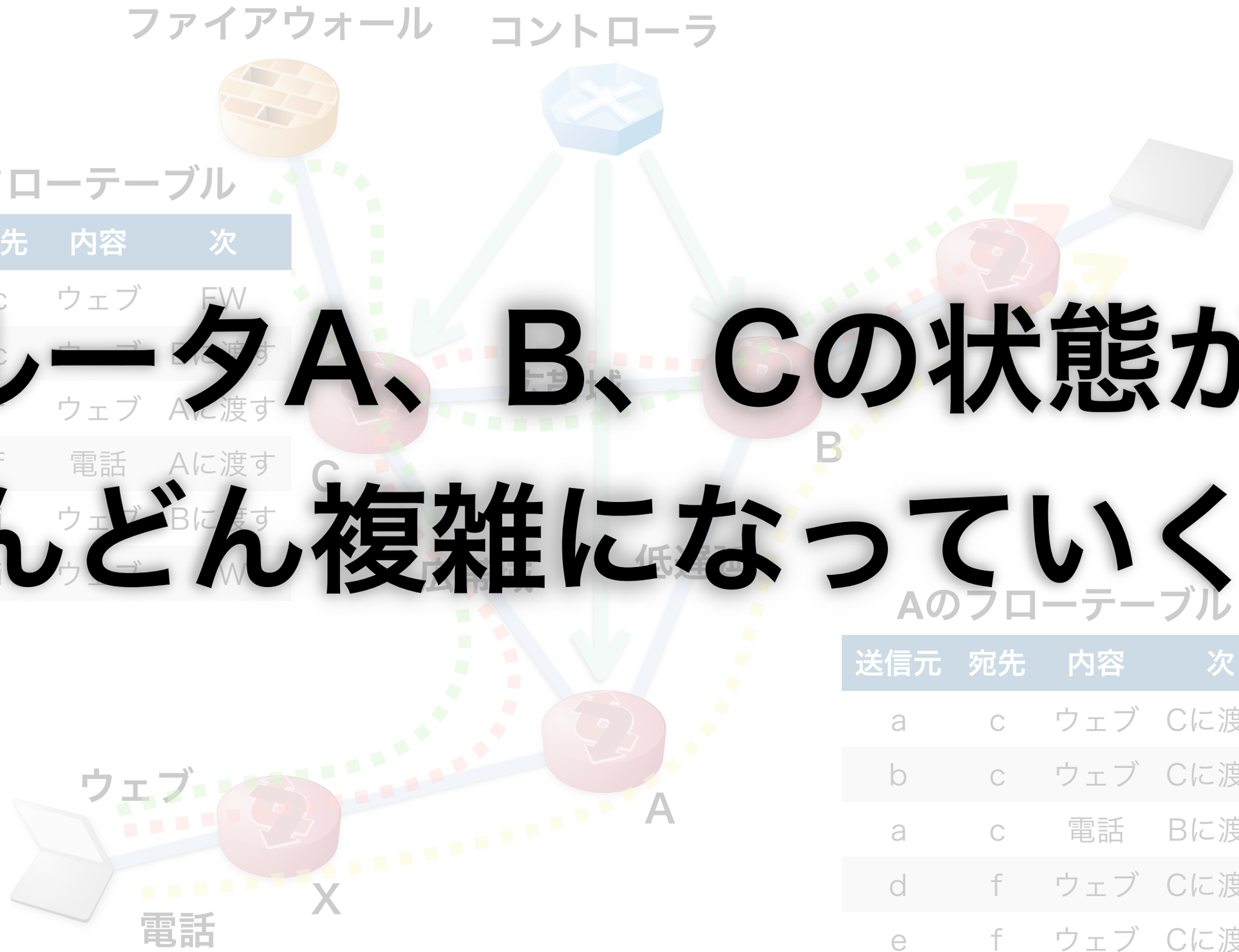
Cのフローテーブル NEW

送信元	宛先	内容	次
a	c	ウェブ	FW
b	c	ウェブ	Bに渡す
d	f	ウェブ	Aに渡す
e	f	電話	Aに渡す
g	i	ウェブ	Bに渡す
h	i	ウェブ	FW

Aのフローテーブル NEW

送信元	宛先	内容	次
a	c	ウェブ	Cに渡す
b	c	ウェブ	Cに渡す
a	c	電話	Bに渡す
d	f	ウェブ	Cに渡す
e	f	ウェブ	Cに渡す
d	f	電話	Bに渡す

OpenFlow？



Cのフローテーブル

送信元	宛先	内容	次
a	c	ウェブ	FW
b	c	ウェブ	Bに渡す
d	f	ウェブ	Aに渡す
e	f	電話	Aに渡す
d	f	ウェブ	Bに渡す
b	i	ウェブ	W

ルータA、B、Cの状態が

どんどん複雑になっていく…

Aのフローテーブル

送信元	宛先	内容	次
a	c	ウェブ	Cに渡す
b	c	ウェブ	Cに渡す
a	c	電話	Bに渡す
d	f	ウェブ	Cに渡す
e	f	ウェブ	Cに渡す
d	f	電話	Bに渡す

エンドツーエンド原理

- **エンドツーエンド原理(End-to-End Principle)** は、コンピュータネットワークの古典的設計原理であり、1981年に Jerome H. Saltzer、David P. Reed、David Dana Clark らの論文 "End-to-end arguments in system design" で初めてその概念が提唱された。
  通信プロトコルの操作は可能な限り通信システムの終端で行い、また制御対象のリソースになるべく近いところで行うべきである
  というもの。

出典: フリー百科事典『ウィキペディア(Wikipedia)』

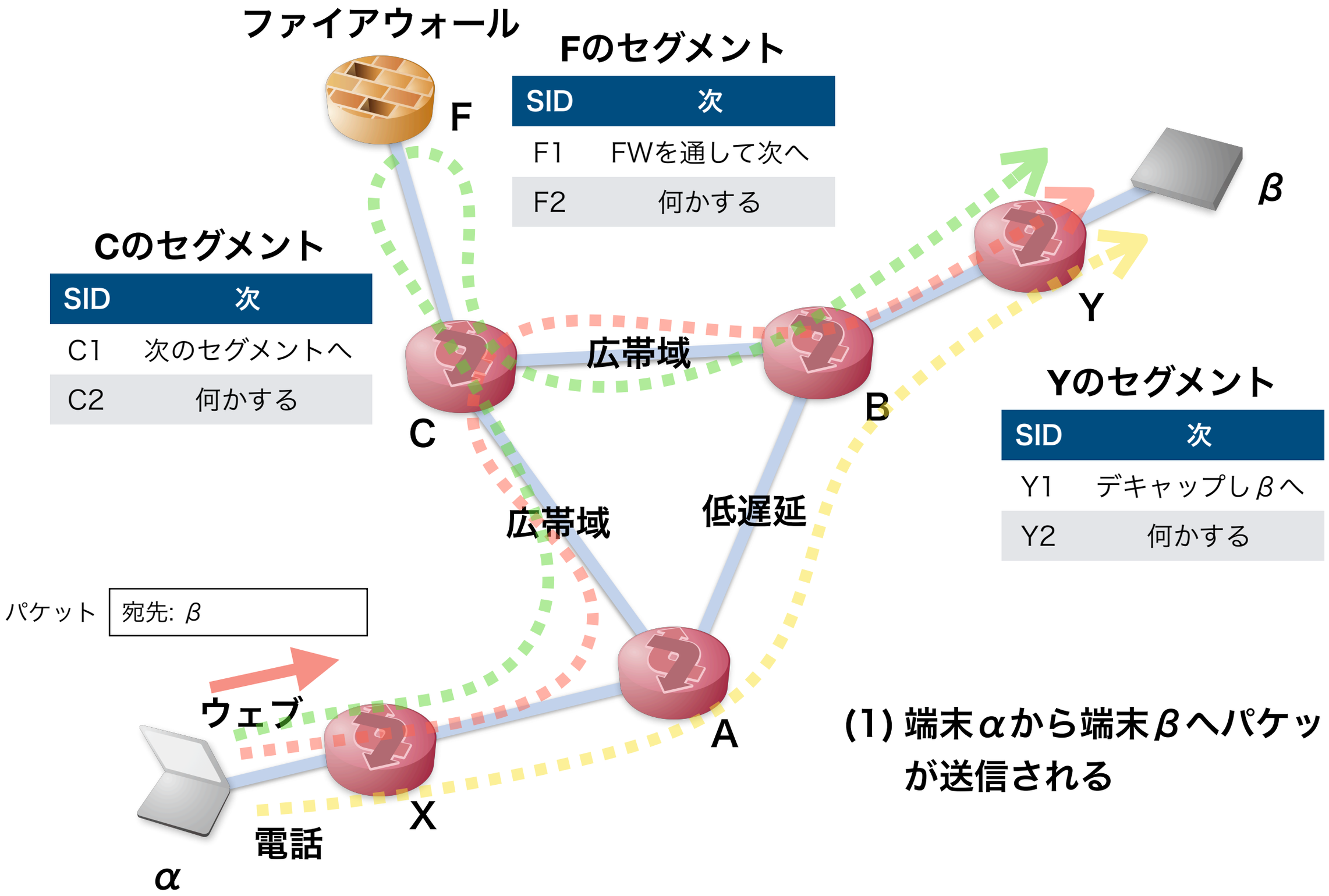
エンドツーエンド原理

- **エンドツーエンド原理(End-to-End Principle)** は、コンピュータネットワークの古典的設計原理であり、1981年に Jerome H. Saltzer、David P. Reed、David Dana Clark らの論文 "End-to-end arguments in system design" で初めてその概念が提唱された。
  通信プロトコルの操作は可能な限り通信システムの終端で行い、また制御対象のリソースになるべく近いところで行うべきである
  というもの。

出典: フリー百科事典『ウィキペディア(Wikipedia)』

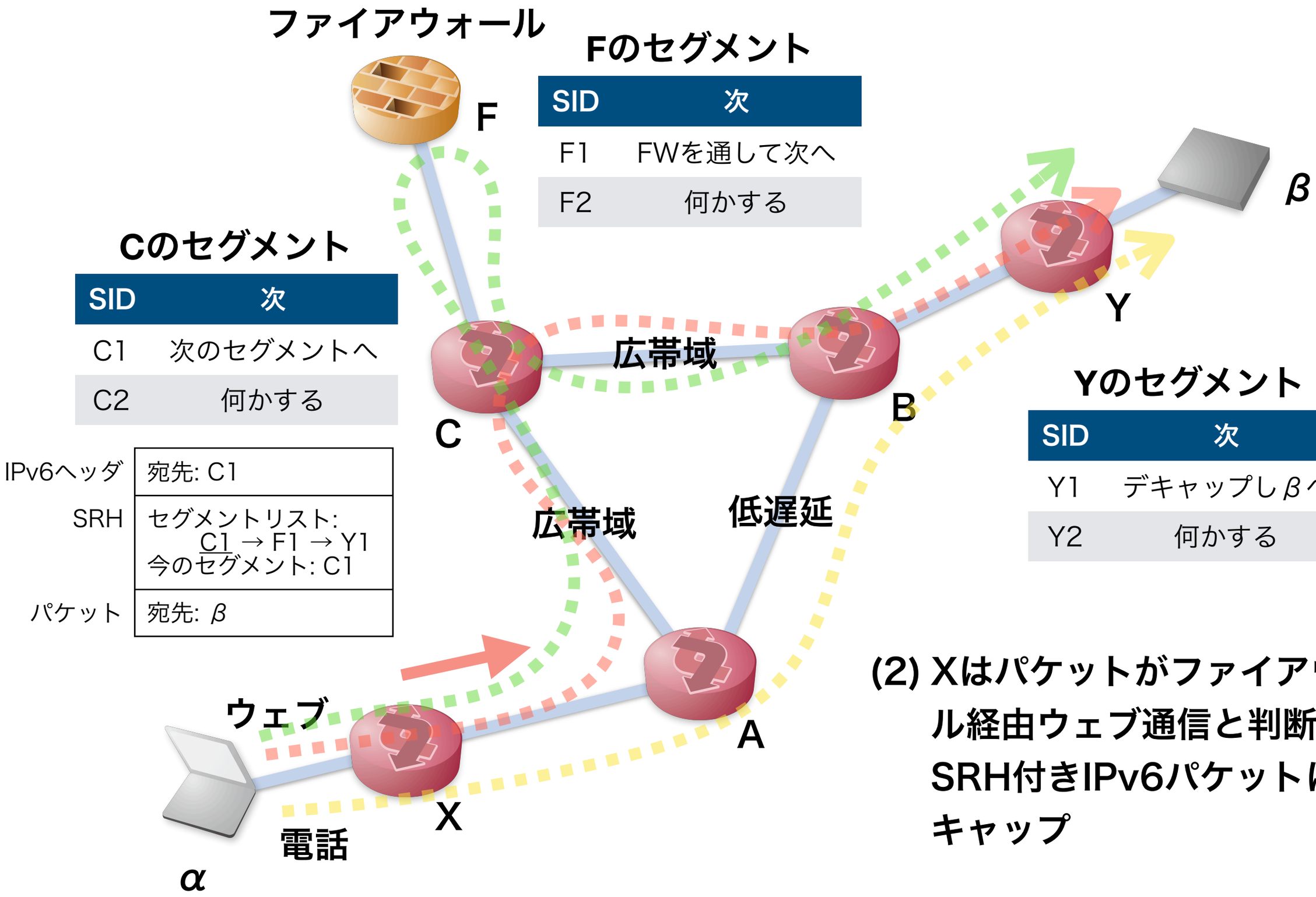
中間ノードはなるたけ
シンプルにしたいじゃん？

SRv6(ファイアウォール経由ウェブ通信)



(1) 端末αから端末βへパケットが送信される

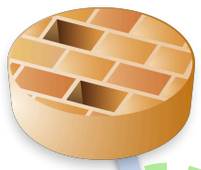
SRv6(ファイアウォール経由ウェブ通信)



(2) Xはパケットがファイアウォール経由ウェブ通信と判断してSRH付きIPv6パケットにエンキャップ

SRv6(ファイアウォール経由ウェブ通信)

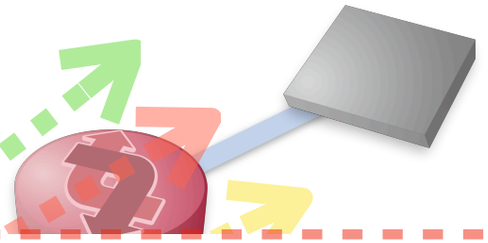
ファイアウォール



F

Fのセグメント

SID	次
F1	FWを通して次へ
F2	何かする



β

Cのセグメント

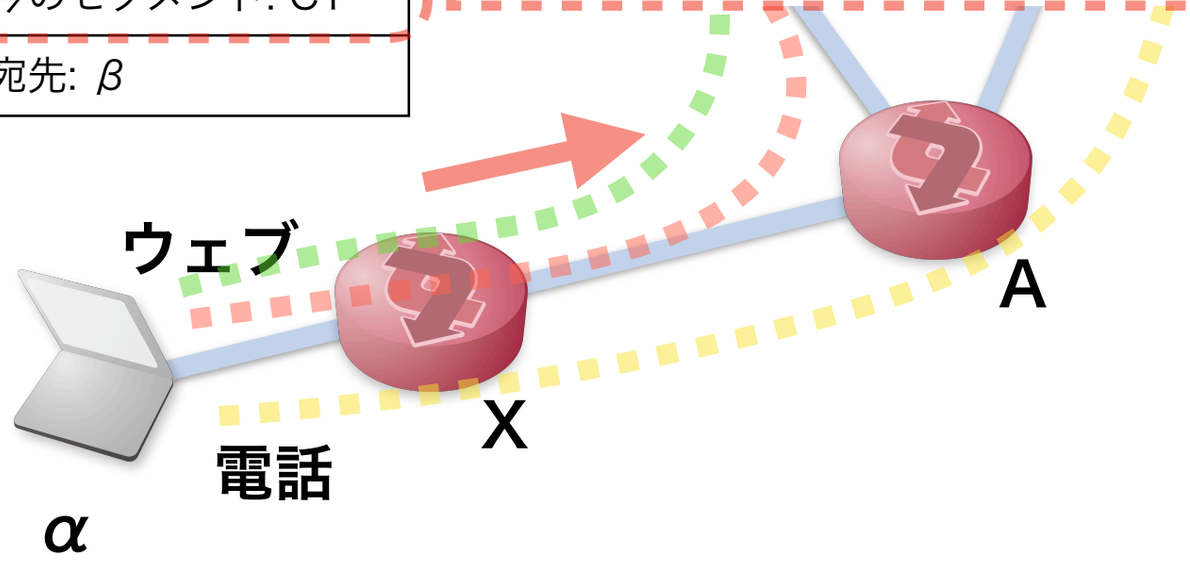
SID	次
C1	次のセグメントへ
C2	何かする

IPv6ヘッダ	宛先: C1
SRH	セグメントリスト: C1 → F1 → Y1 今のセグメント: C1
パケット	宛先: β

SRH ... Segment Routing Header の略。IPv6拡張ヘッダであるルーティングヘッダのひとつ。

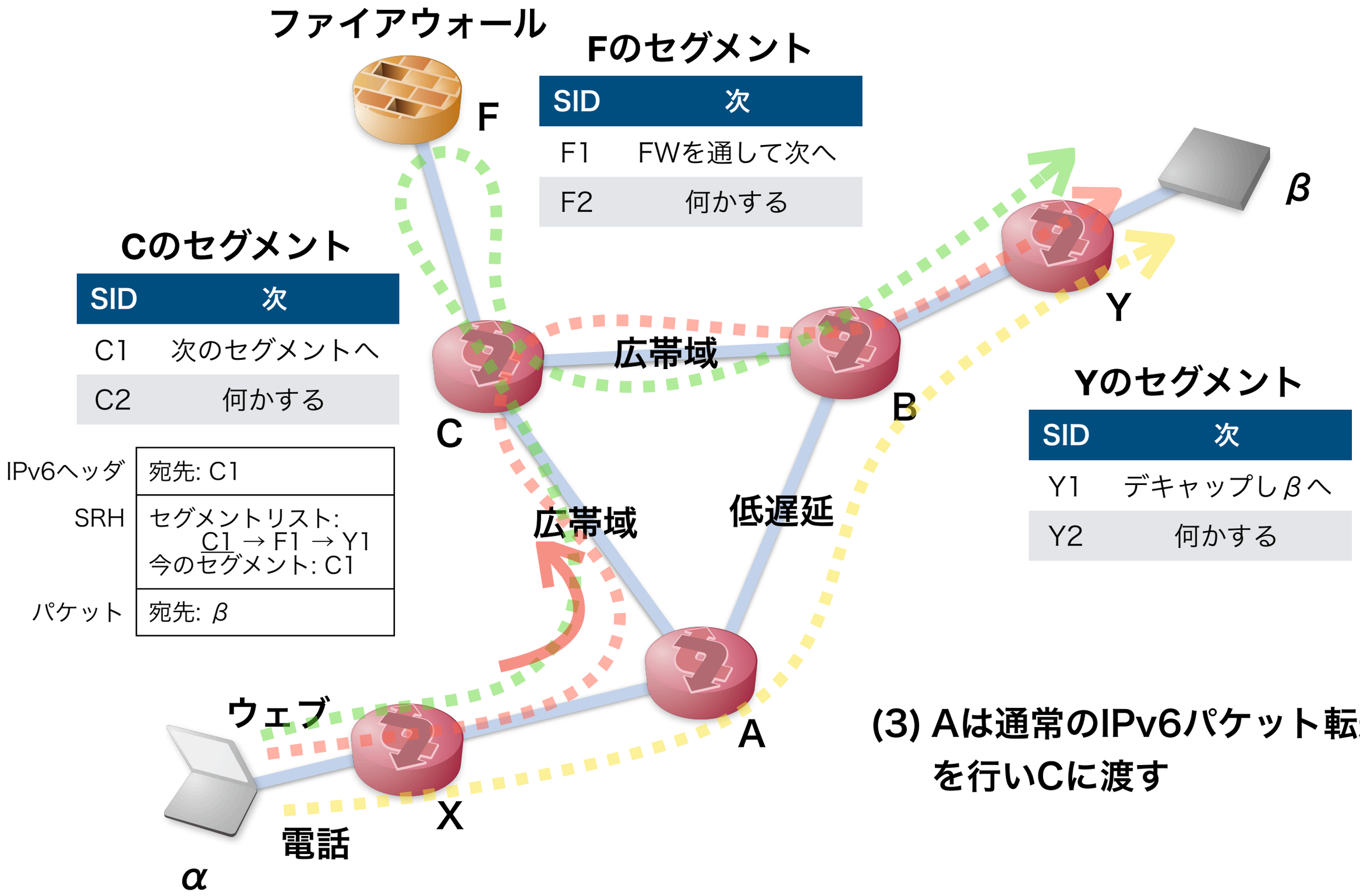
セグメントリスト ... 実行したいセグメントのリスト。

セグメント ... 命令みたいなもの。(だいたい)IPv6宛先アドレスになる。識別子は SID とも呼ばれる。

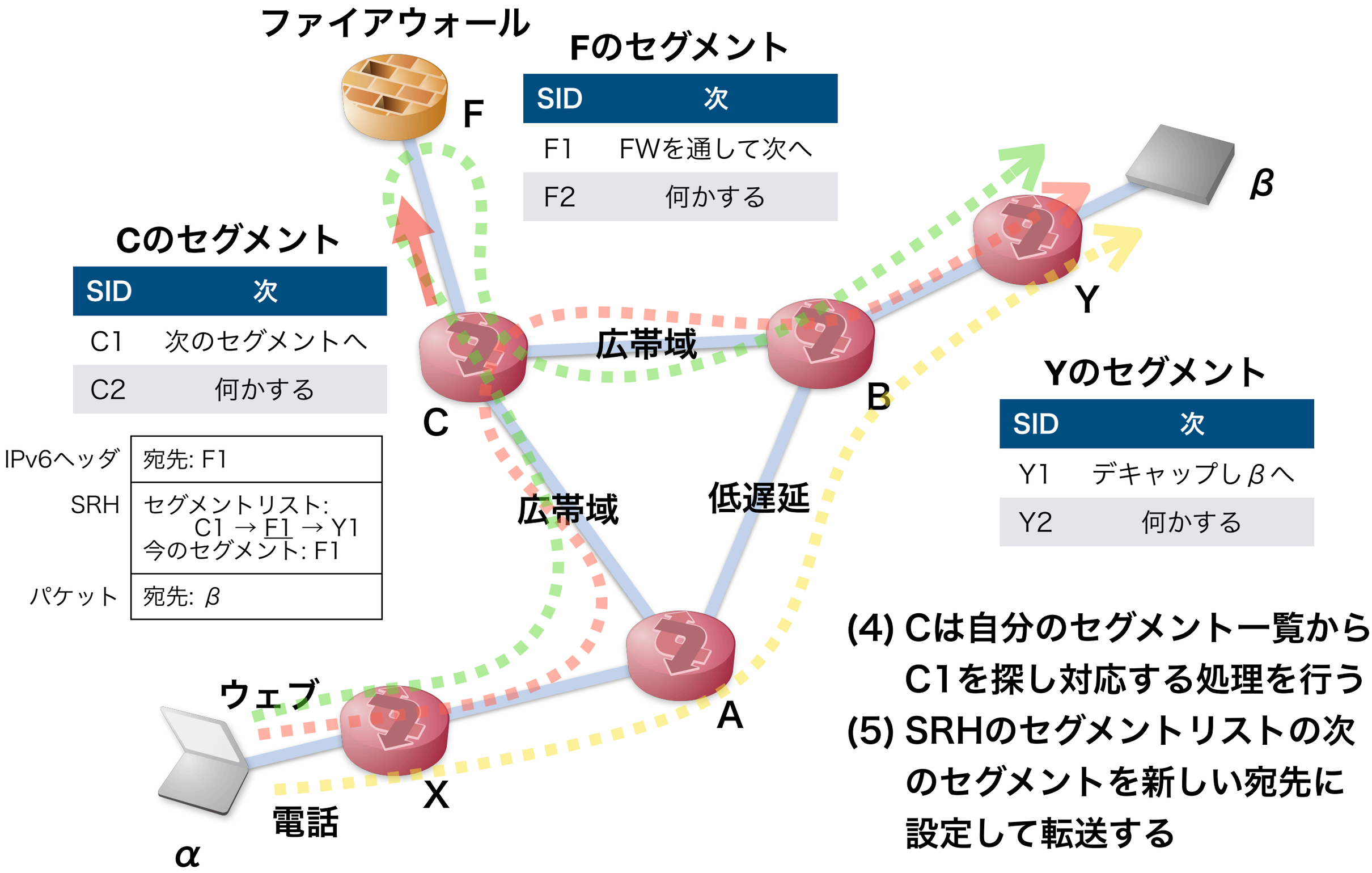


(2) Xはパケットがファイアウォール経由ウェブ通信と判断してSRH付きIPv6パケットにエンキャップ

SRv6(ファイアウォール経由ウェブ通信)



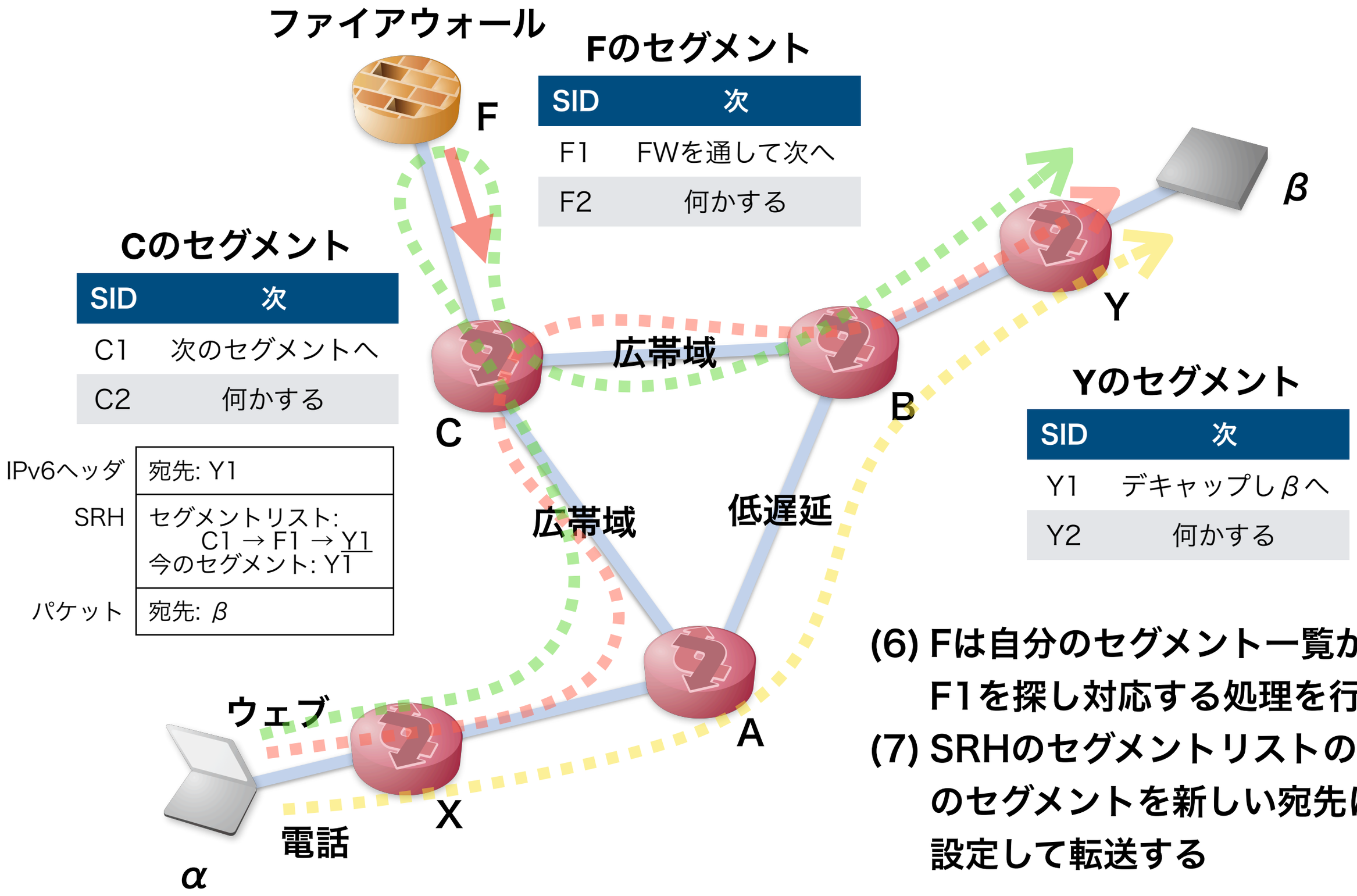
SRv6(ファイアウォール経由ウェブ通信)



(4) Cは自分のセグメント一覧から C1を探し対応する処理を行う

(5) SRHのセグメントリストの次のセグメントを新しい宛先に設定して転送する

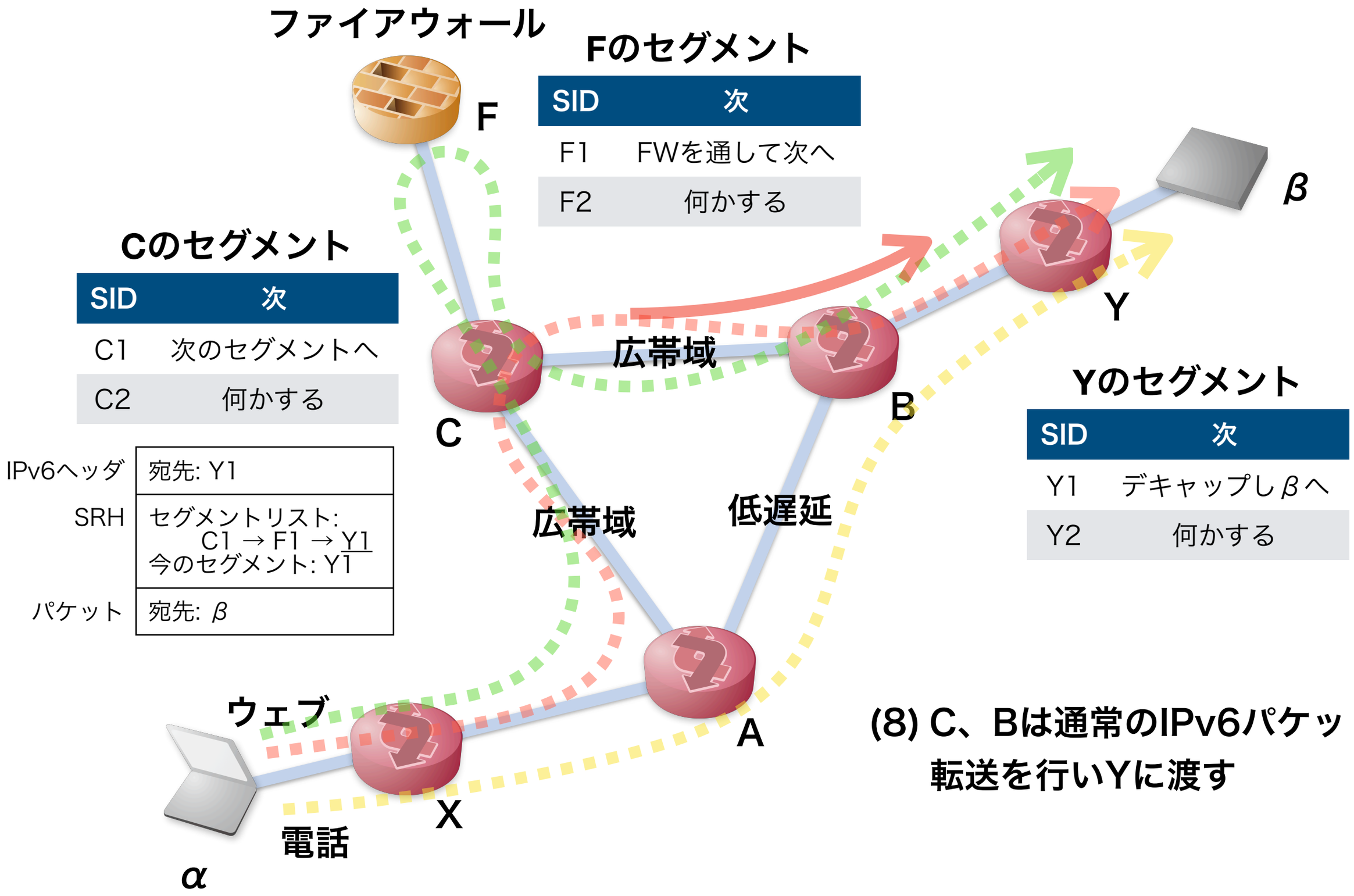
SRv6(ファイアウォール経由ウェブ通信)



(6) Fは自分のセグメント一覧から F1を探し対応する処理を行う

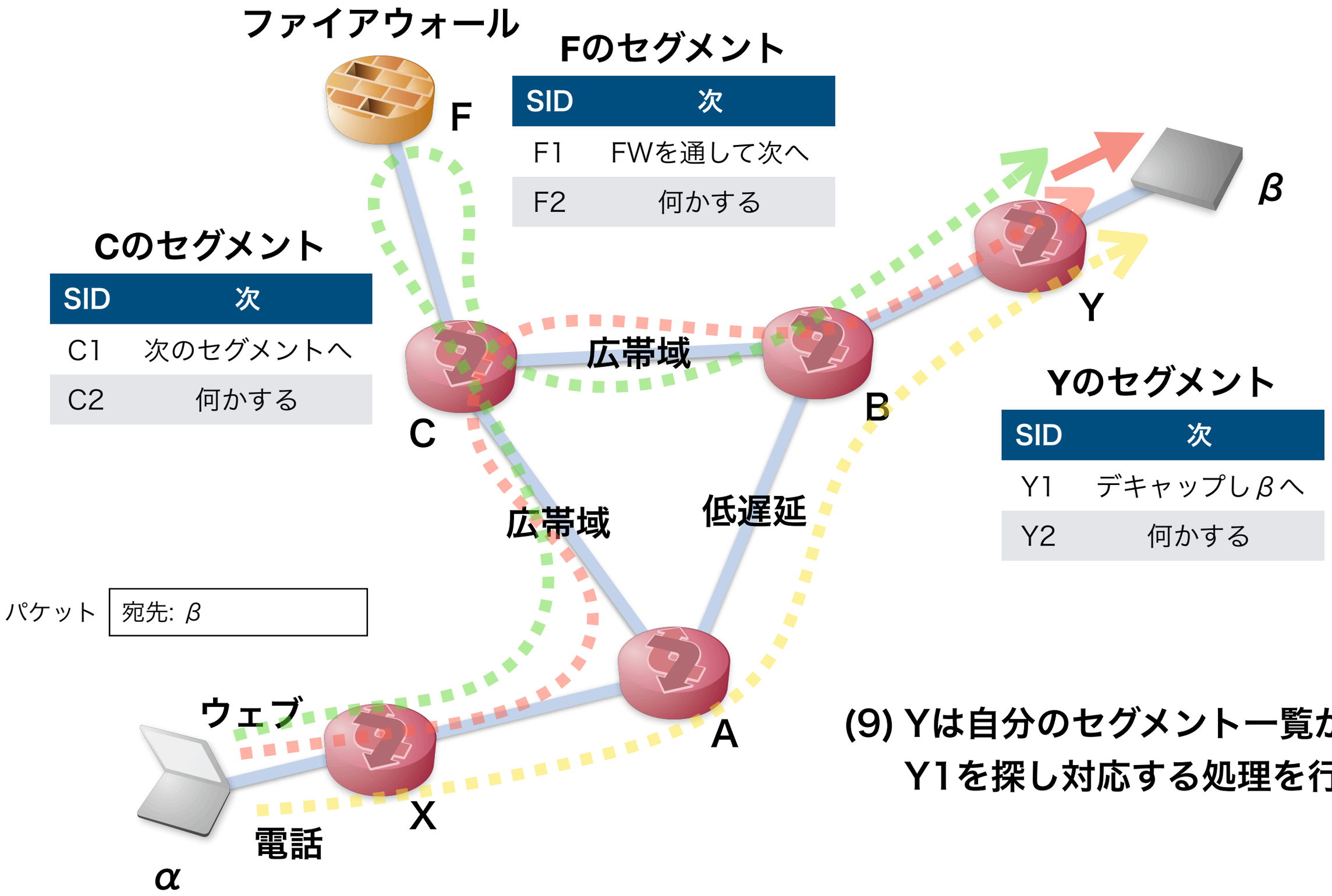
(7) SRHのセグメントリストの次のセグメントを新しい宛先に設定して転送する

SRv6(ファイアウォール経由ウェブ通信)



(8) C、Bは通常のIPv6パケット転送を行いYに渡す

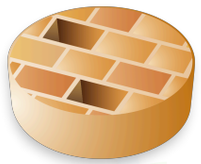
SRv6(ファイアウォール経由ウェブ通信)



(9) Yは自分のセグメント一覧から
Y1を探し対応する処理を行う

SRv6(ファイアウォール経由ウェブ通信)

ファイアウォール



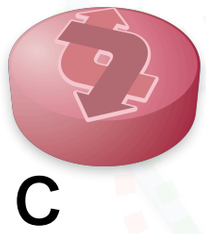
F

Fのセグメント

SID	次
F1	FWを通して次へ
F2	何かする

Cのセグメント

SID	次
C1	次のセグメントへ
C2	何かする



C

B

Yのセグメント

SID	次
Y1	デキャップしβへ
Y2	何かする

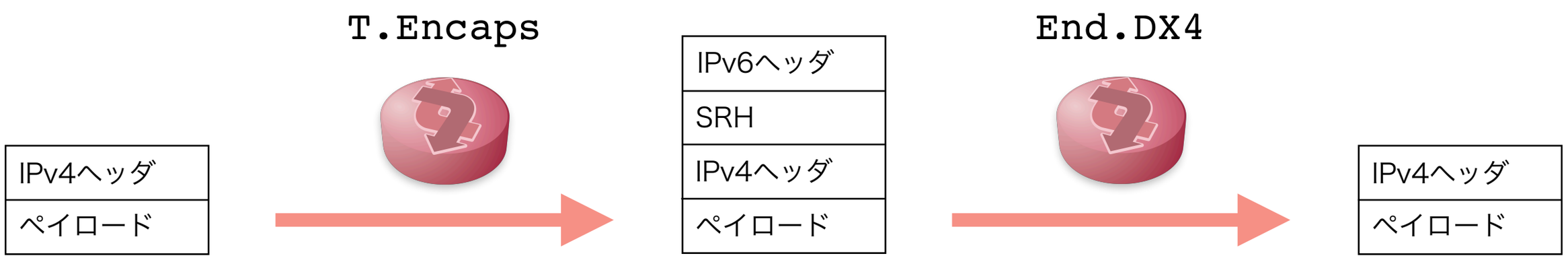
A

中間ノードはそれぞれの到達性と
自分自身のセグメントだけ気にすれば良い

SRv6(ファイアウォール経由ウェブ通信)

終端ノードは通信の属性から
どのセグメントを通すかを判断し
セグメントリストを構築する

セグメントいろいろ


IPv4ヘッダ
ペイロード
T.Encaps
IPv6ヘッダ
SRH
IPv4ヘッダ
ペイロード
End.DX4
IPv4ヘッダ
ペイロード
IPv6ヘッダ
ペイロード
T.Encaps
IPv6ヘッダ
SRH
IPv6ヘッダ
ペイロード
End.DX6
IPv6ヘッダ
ペイロード
L2ヘッダ
ペイロード
T.Encaps.L2
IPv6ヘッダ
SRH
L2ヘッダ
ペイロード
End.DX2
L2ヘッダ
ペイロード

セグメントいろいろ


T.Insert
End.X(PSP)
IPv6ヘッダ
ペイロード
IPv6ヘッダ
SRH
ペイロード
IPv6ヘッダ
ペイロード
T.Encaps
End.DT6
IPv6ヘッダ
ペイロード
IPv6ヘッダ
SRH
IPv6ヘッダ
ペイロード
IPv6ヘッダ
ペイロード
T.Encaps
End.B6.Insert
IPv6ヘッダ
ペイロード
IPv6ヘッダ
SRH
IPv6ヘッダ
ペイロード
IPv6ヘッダ
SRH
SRH
IPv6ヘッダ
ペイロード

セグメントいろいろ

	意味
End.*X*	クロスコネクト あらかじめ指定されたネクストホップへ転送
End.*T*	テーブルルックアップ あらかじめ指定されたテーブルからネクストホップを探索し転送
End.*D*	デキャップ デキャップする(IPv6ヘッダとSRv6ヘッダを取り除く)
End.*2*	L2 中身はL2フレーム
End.*4*	IPv4 中身はIPv4パケット
End.*6*	IPv6 中身はIPv6パケット
End.*B*	バインディングSID バインディングSIDを展開(SIDからセグメントリストを展開)

SRv6(ファイアウォール経由ウェブ通信)

Y

X

終端ノードは通信の属性から
どのセグメントを通すかを判断し
セグメントリストを構築する

SRv6(ファイアウォール経由ウェブ通信)

終端ノードはどうやって
セグメントの存在を知るの？？

終端ノードはどうやって
必要なセグメントを判断するの？

終端ノードは通信の属性から
どのセグメントが必要かを判断し
セグメントリストを構築する

イラスト: (C)いらすとや

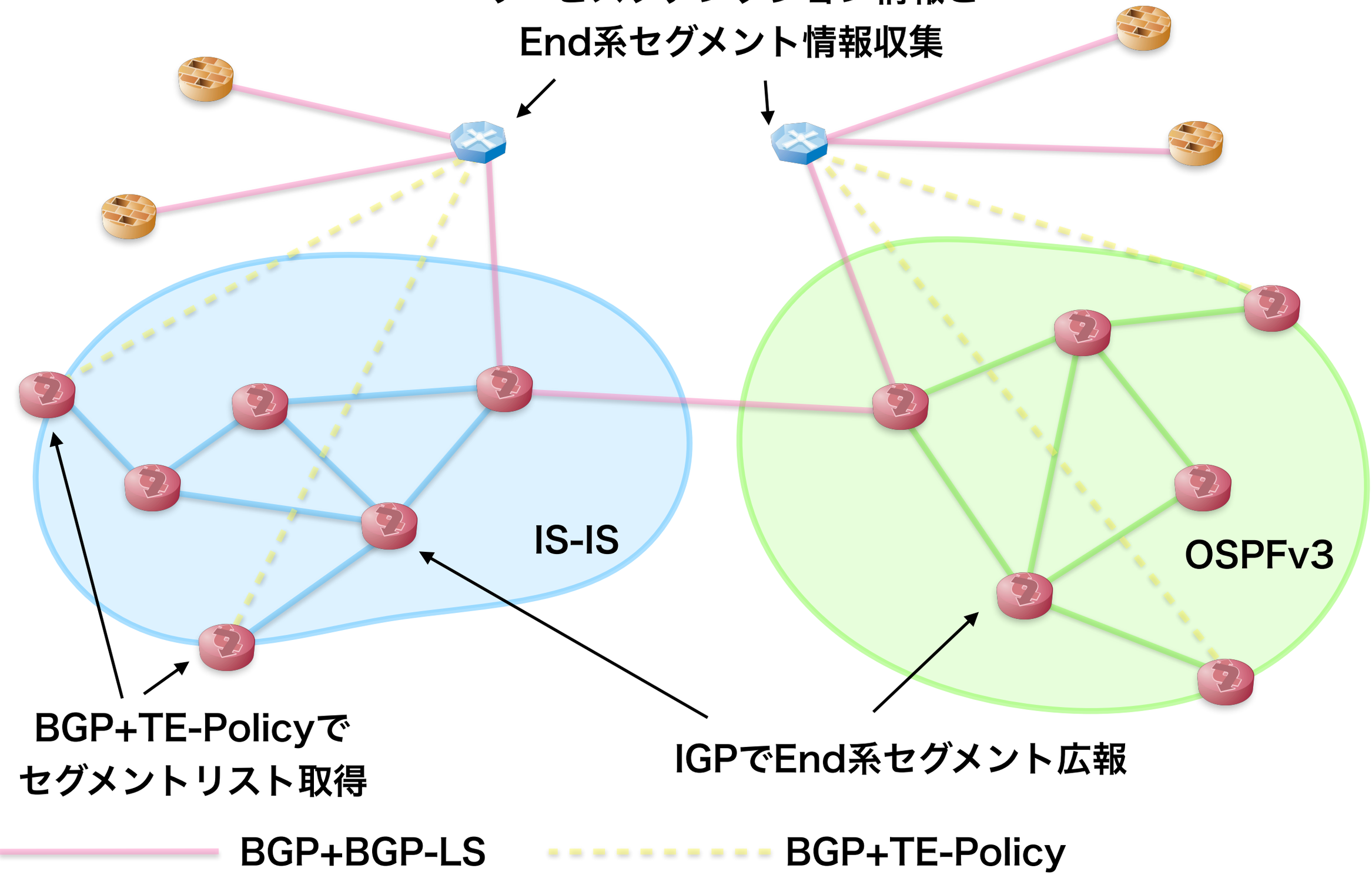
End.X

SRH[SL]...河野さん

参考情報

- draft-bashandy-isis-srv6-extensions
  - IS-ISでSRv6セグメントを配る拡張
- draft-li-ospf-ospfv3-srv6-extensions
  - OSPFv3でSRv6セグメントを配る拡張
- draft-dawra-idr-bgpls-srv6-ext
  - IS-IS/OSPFv3で収集したSRv6セグメントをBGP-LSで配る拡張
- draft-dawra-idr-bgp-sr-service-chaining
  - BGP-LSでサービスファンクションの情報を配る拡張
- draft-previdi-idr-segment-routing-te-policy
  - BGPでトラフィックエンジニアリングのポリシーを配る拡張
- draft-ietf-lsr-flex-algo
  - IS-IS/OSPFv3で用いるパス計算のアルゴリズムを定義する拡張
- draft-dawra-idr-srv6-vpn
  - SRv6でVPNサービスを提供する方法

こんな感じ？


BGP-LSでトポロジ情報と
サービスファンクション情報と
End系セグメント情報収集
IS-IS
OSPFv3
BGP+TE-Policyで
セグメントリスト取得
IGPでEnd系セグメント広報
BGP+BGP-LS
BGP+TE-Policy