KDDI R&D LABS

OpenFlow Controllerの冗長化に関する検討

株式会社KDDI研究所
統合コアネットワーク制御グループ
黒木 圭介 / 林 通秋

■概要

1. 背景
2. 課題と目的
3. 利用する機能
4. 同一サイトにおける冗長化について
5. サイト間における冗長化について
6. 今後の課題
7. 考察とまとめ

KDDI R&D LABS

■概要

1. 背景

2. 課題と目的

3. 利用する機能

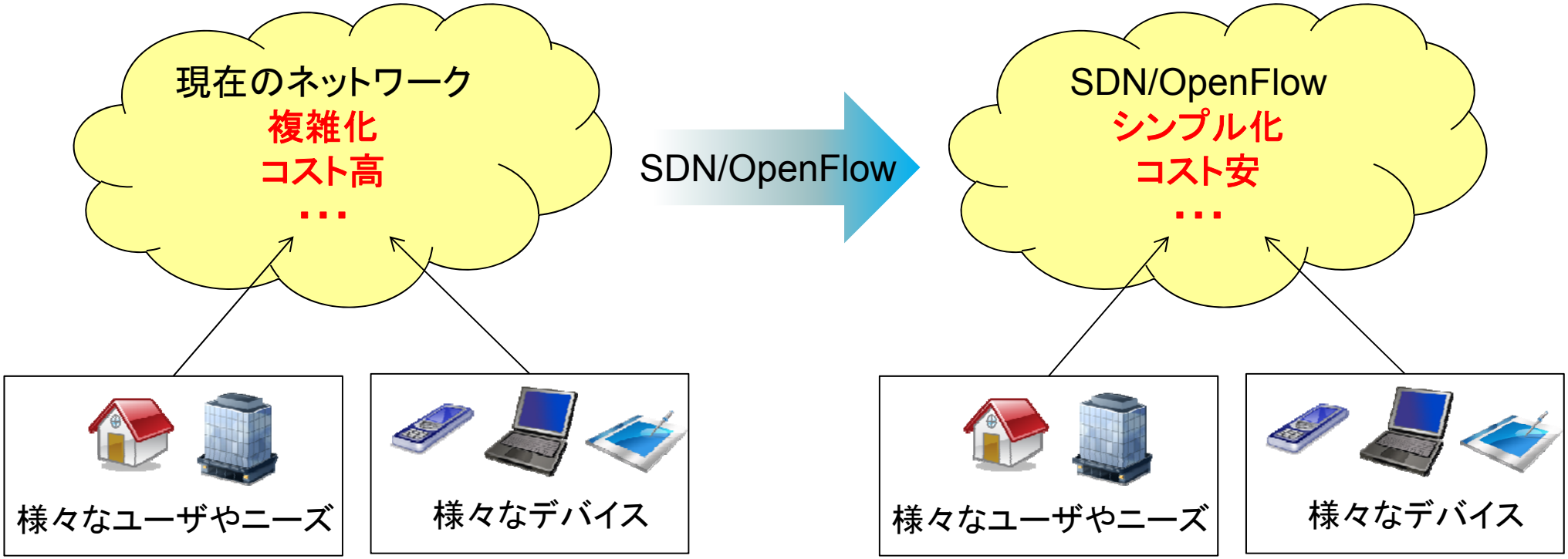
4. 同一サイトにおける冗長化について

5. サイト間における冗長化について

6. 今後の課題

7. 考察とまとめ

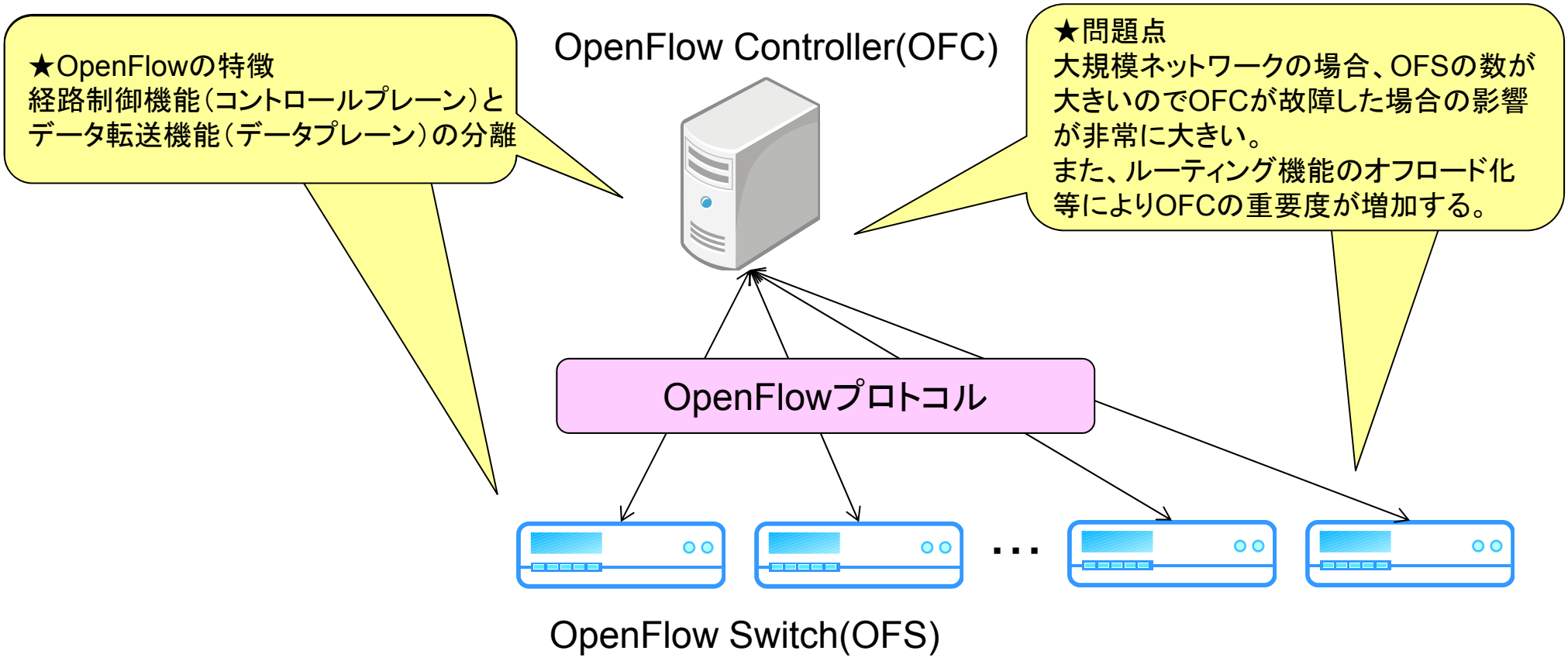
■背景
インターネットにおいて様々なニーズが存在する昨今、ネットワークはより複雑に、そしてよりコスト高になってきており、その解決策の1つとしてSDN/OpenFlowの研究開発が盛んに行われている。



■課題と目的

OpenFlowの技術を大規模ネットワークへ適用する場合に、OpenFlow Controllerの冗長性の確保をどのように実現していくべきかを考えなくてはいけない。

- 1台のOpenFlow Controllerで数千台のOpenFlow Switchを制御するのはありえない
- 重要度が大きいOpenFlow Controllerの冗長性を確保するべき



3. 利用する機能(Multiple Controllers)の動作

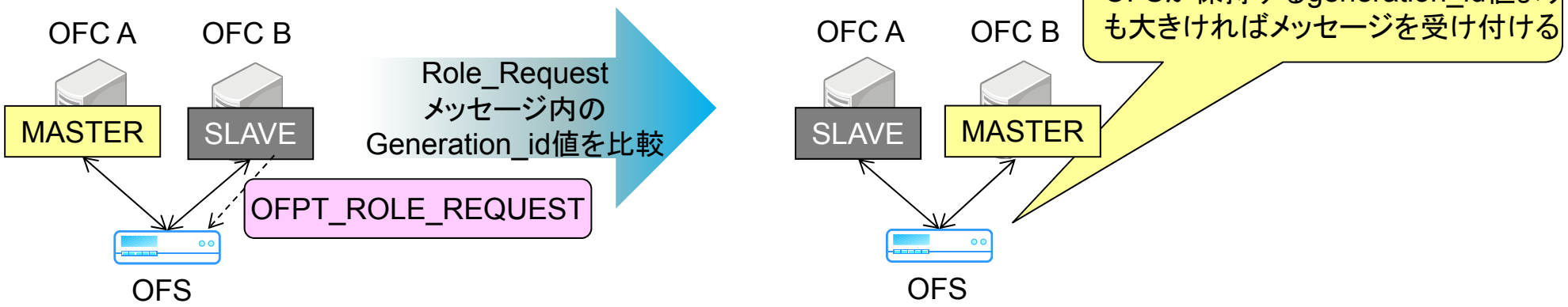
■利用する機能

本研究における冗長化にはOpenFlow Switch Specification Version1.2より規定されている
「Multiple Controllers」機能を利用する。

■Multiple Controllersにて規定されているOpenFlow Controllerの役割の違い

役割	特徴
EQUAL	・OFSに対してフルアクセス権を持っている ・Packet-IN等の非同期メッセージを全て受け取る
SLAVE	・OFSに対してread-onlyのアクセス権を持つ ・Packet-IN等の非同期メッセージを受け取らない
MASTER	・EQUALと同様の権限を持つが、OFSはMASTERを一つ しか持てない

■MASTER選出の動作



但し、Controller間の通信は特に規定されておらず、これらの機能をどのように利用すれば冗長化が実現できるかを考える必要がある

KDDI R&D LABS

■概要

1. 背景

2. 課題と目的

3. 利用する機能

4. 同一サイトにおける冗長化について

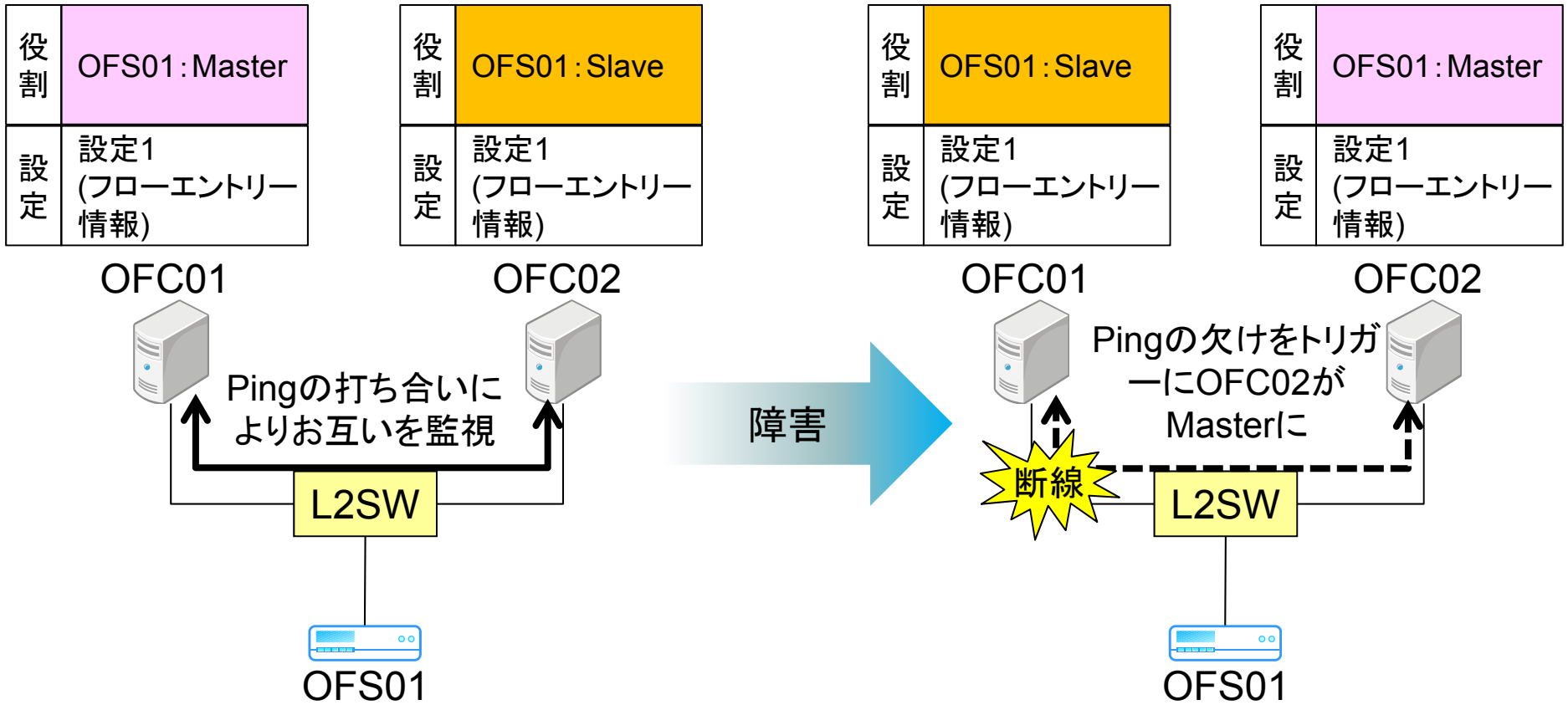
5. サイト間における冗長化について

6. 今後の課題

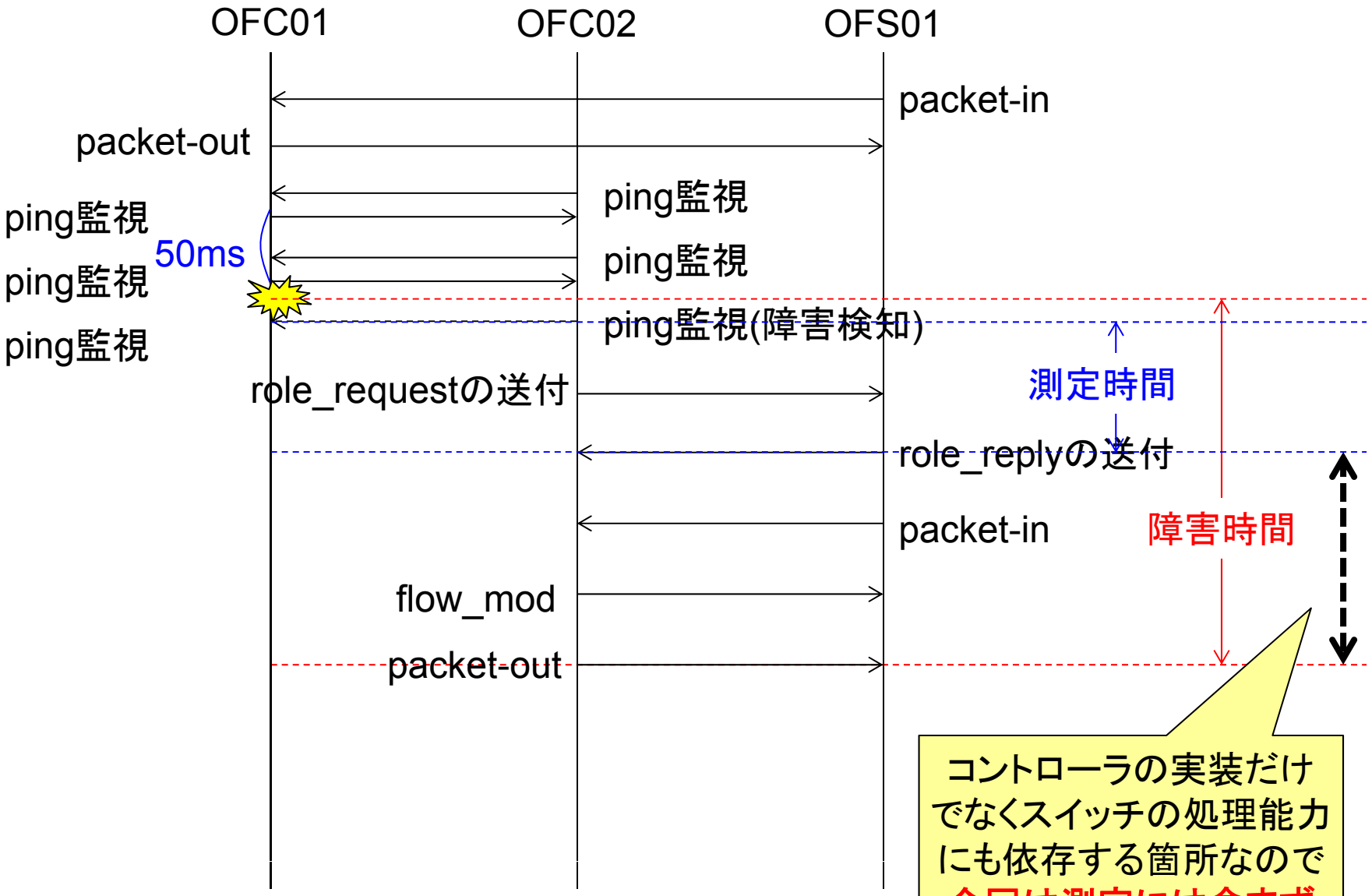
7. 考察とまとめ

4. 同一サイトでの冗長化について　動作概要

■同一サイトでの冗長化
OpenFlow Controller(OFC)同士がPingにて死活監視を行い、SlaveがMasterの障害を検知した場合、自身の役割をMasterに変更する。
■特徴
冗長構成を実現するOFCは両者とも同一のフローエントリー情報を保持している。

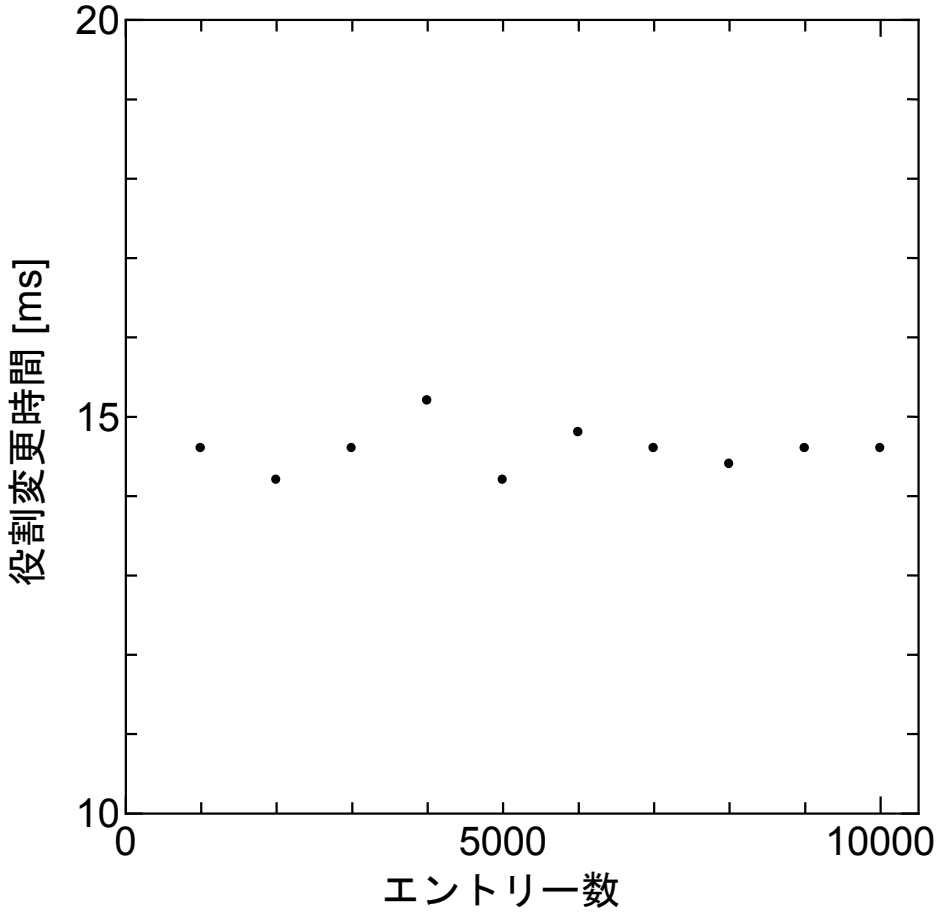


4. 同一サイトでの冗長化について シーケンス図



4. 同一サイトでの冗長化について　実験結果

■実験結果
下のグラフはエントリー数を増加させた場合の、障害検知から役割変更までにかかる時間を測定したものである。
ほぼエントリー数には依らず**約15msで役割の変更**を行える。



KDDI R&D LABS

■概要

1. 背景

2. 課題と目的

3. 利用する機能

4. 同一サイトにおける冗長化について

5. サイト間における冗長化について

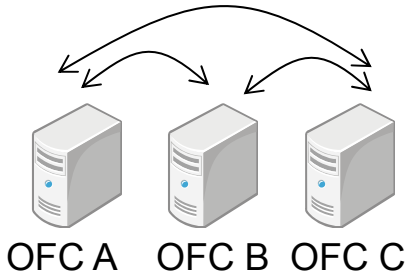
6. 今後の課題

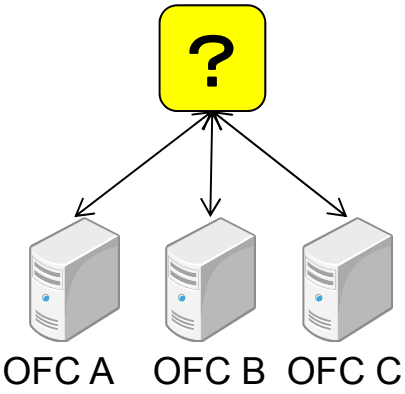
7. 考察とまとめ

5. サイト間におけるOpenFlow Controller冗長化の手法比較

■OpenFlow Controller冗長化の手法比較
例えば、冗長化の手法として、以下の2案を比較してみる。
データの引継の柔軟性を加味し、第3者で制御する手法で実現できないかを考えてみる。

こちらで実験





手法	Controller間で通信させる場合	第3者で制御する場合
Controllerの負荷	△（1:Nの通信）	○（1:1の通信）
データ引継の柔軟性	△（引継先の決定権をだれが持つか？）	○
コスト	○	△

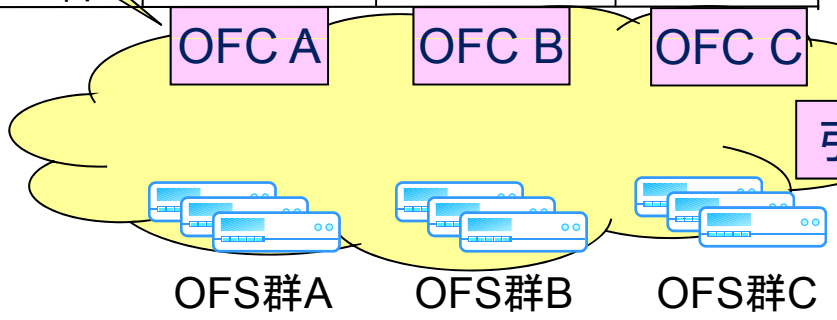
5. サイト間における冗長化手法のコンセプト

■提案するOpenFlow Controller冗長化の特徴
本研究で考案する冗長化の特徴はController間の設定引継ぎと役割管理を外出しにし、第3者がOpenFlow Controllerの役割制御を行うことである。
■引継管理を外出しにするメリット
・コントローラ間の監視による負荷を軽減
・障害時の影響度の最小化

■ポイント①
地域毎若しくはサイト毎にOpenFlow Controllerが存在することを想定

■ポイント②
地域毎若しくはサイト毎にそのOpenFlow Switchに対するOpenFlow Controllerの役割を分担する

	コントローラの役割		
OFS群A	Master	Slave	Slave
OFS群B	Slave	Master	Slave
OFS群C	Slave	Slave	Master



■ポイント③
引継管理サーバは各コントローラ内のフローエントリー情報や役割を管理し、コントローラの1つが故障した際に引継を制御する

5. サイト間での冗長化について　動作概要 1/3

■サイト間での冗長化
引継管理サーバがOFCの監視を行い、例えばOFC01の障害を検知した場合、本当にOFC01が故障しているかを検証し(障害判定フェーズ)、障害と判断した場合にOFC01の設定情報をOFC02に引継がせ、役割をSlaveからMasterに変更させる(引継フェーズ)。
■特徴
冗長構成を実現するOFCは別のフローエントリー情報を保持している。

役割	OFS01：Master
	OFS02：Slave
設定	設定1 (フローエントリー情報)

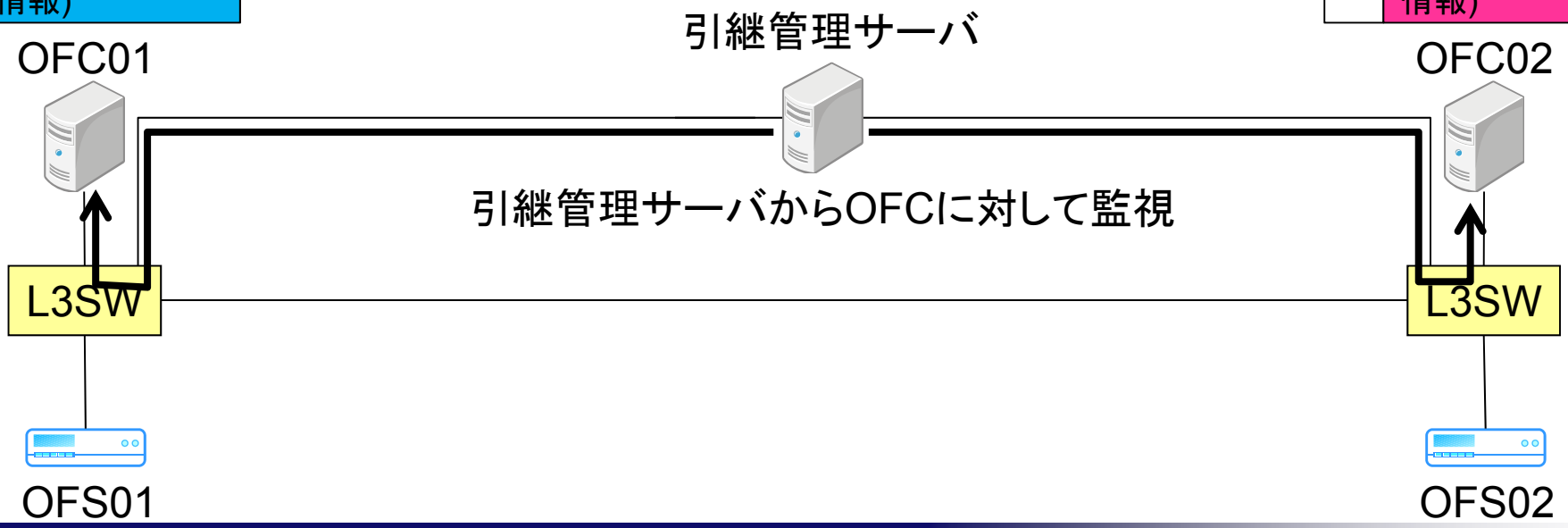
OFC01

引継管理システムが保持する情報

Host名	設定情報	OFS01に対する役割	OFS02に対する役割
OFC01	設定1	Master	Slave
OFC02	設定2	Slave	Master

役割	OFS01：Slave
	OFS02：Master
設定	設定2 (フローエントリー情報)

OFC02



5. サイト間での冗長化について　動作概要 2/3

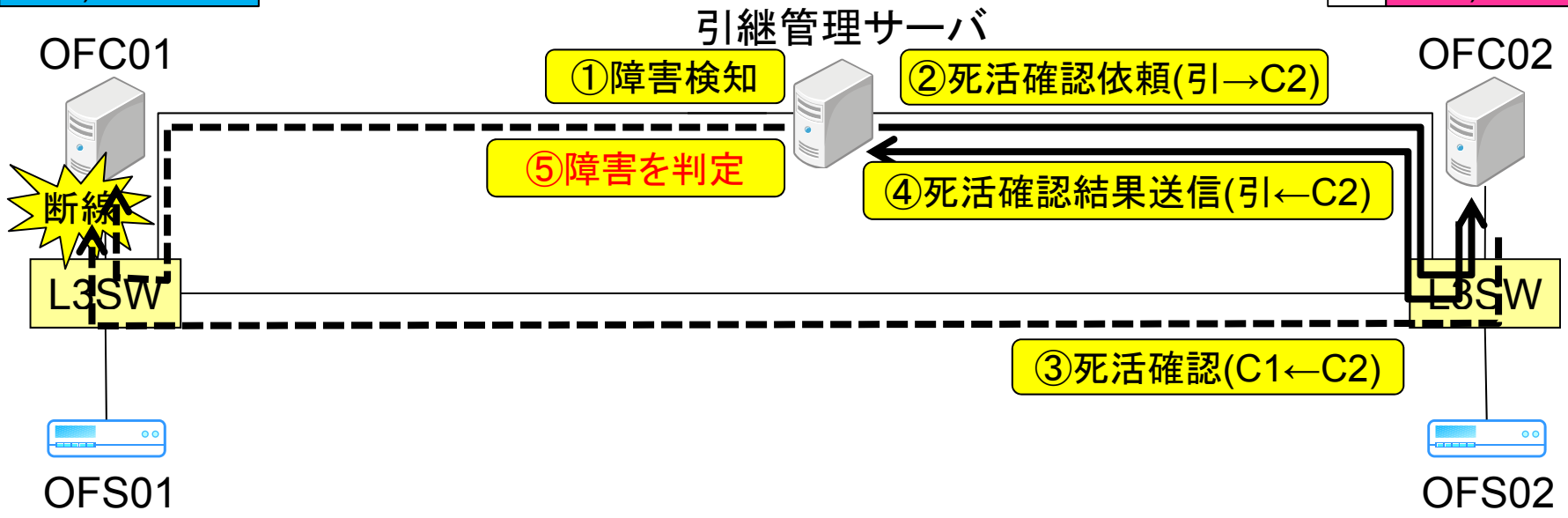
■サイト間での冗長化
障害判定フェーズ
　無駄な引継を阻止するために障害判定を行う

役割	OFS01：Master
	OFS02：Slave
設定	設定1 (フローエントリー情報)

引継管理システムが保持する情報

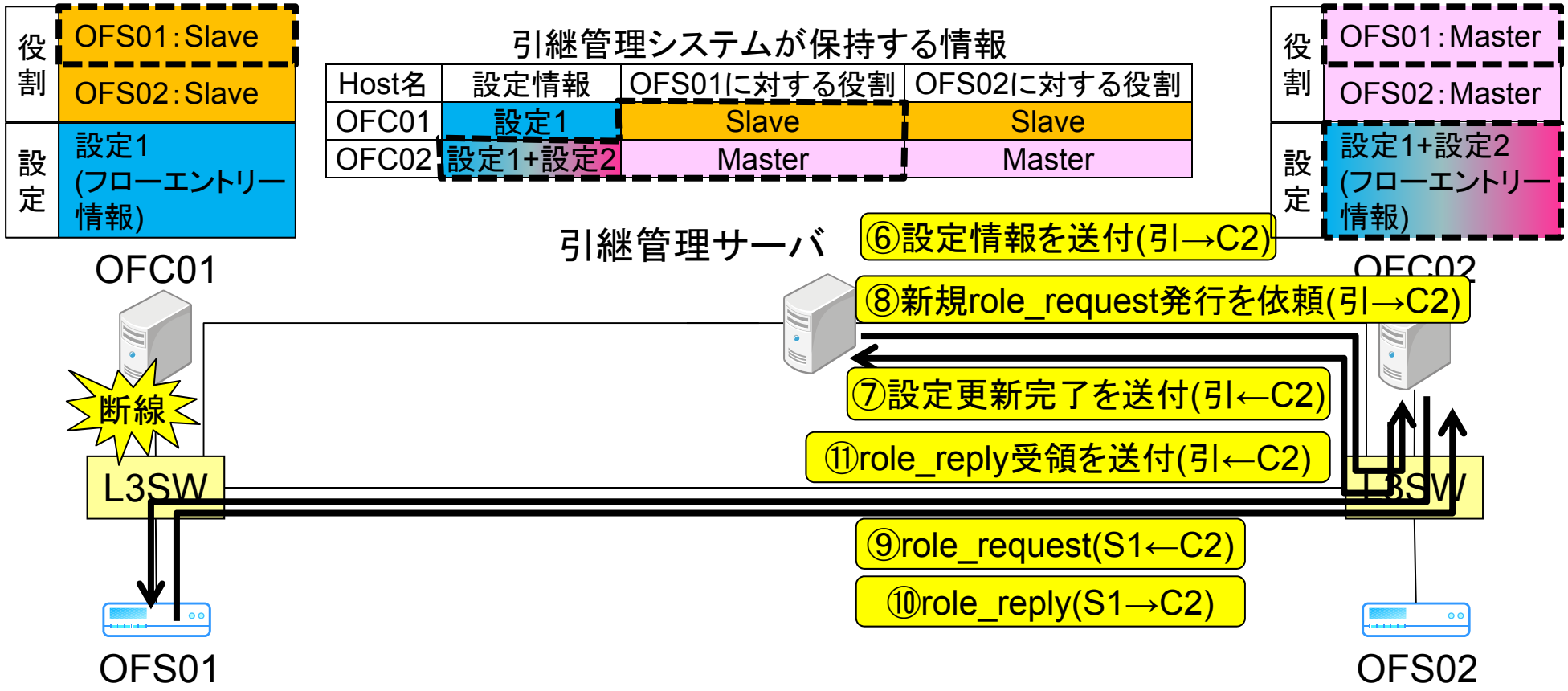
Host名	設定情報	OFS01に対する役割	OFS02に対する役割
OFC01	設定1	Master	Slave
OFC02	設定2	Slave	Master

役割	OFS01：Slave
	OFS02：Master
設定	設定2 (フローエントリー情報)

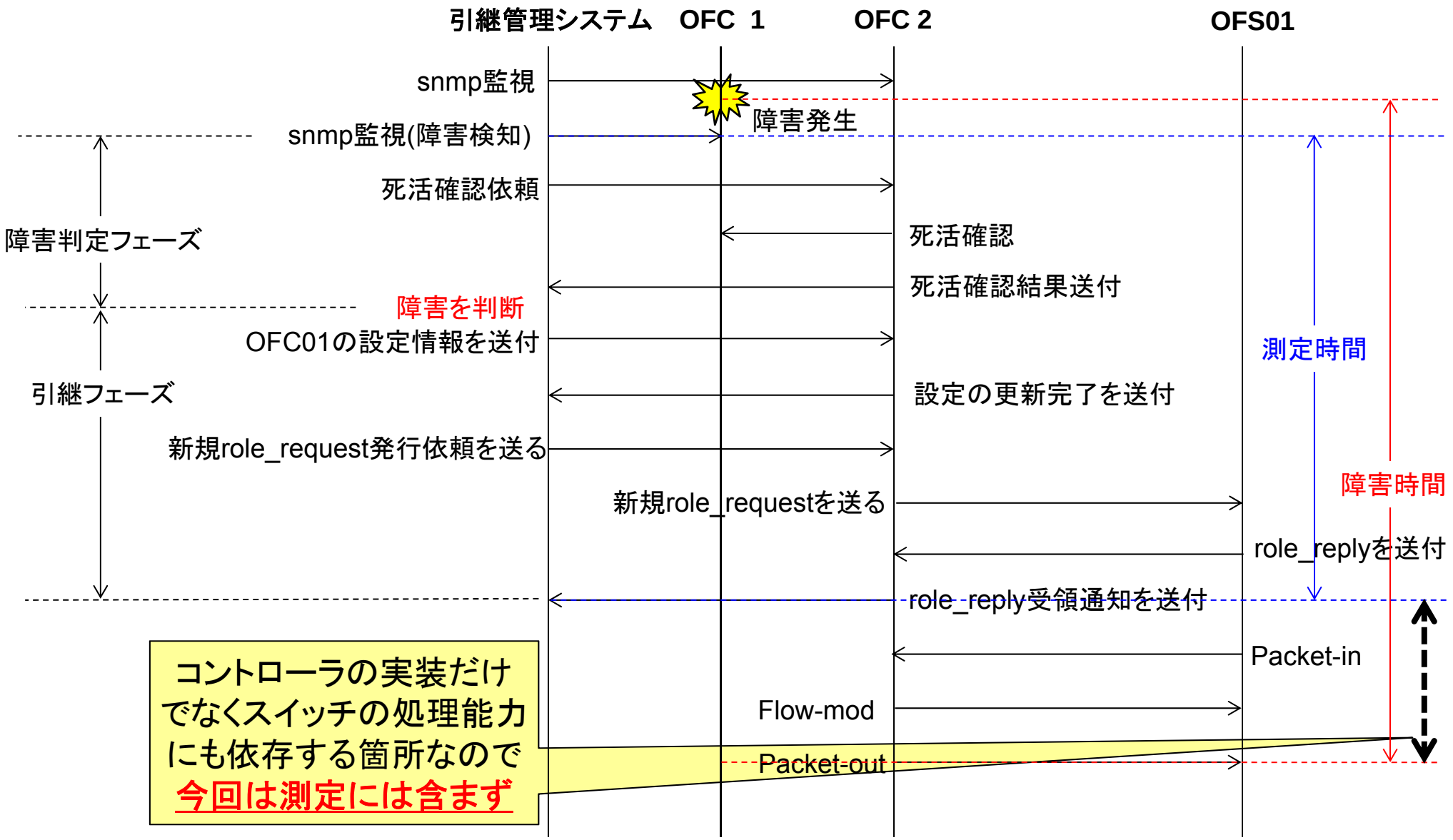


5. サイト間での冗長化について　動作概要 3/3

■サイト間での冗長化
引継フェーズ
　実際に引継先コントローラの情報を更新する



5. サイト間冗長におけるシーケンス図



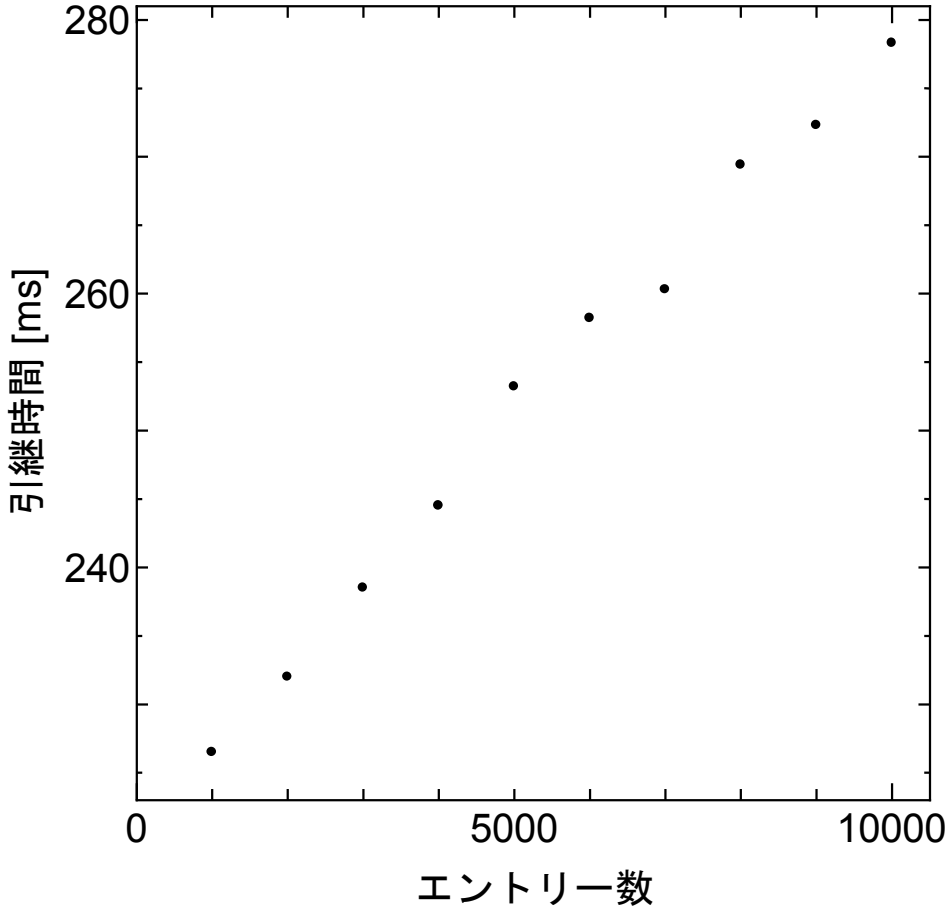
5. サイト間の冗長化について　実験結果

■実験結果

下のグラフはエントリー数を増加させた場合の、障害検知から役割変更までにかかる時間を測定したものである。

エントリー数の増加に対してほぼ線形に増加している。**10000エントリーで約278ms**である。

※現行技術におけるサイレント障害(≒コントロールプレーンの故障)の最短検知時間は約300ms~450ms



KDDI R&D LABS

■概要

1. 背景

2. 課題と目的

3. 利用する機能

4. 同一サイトにおける冗長化について

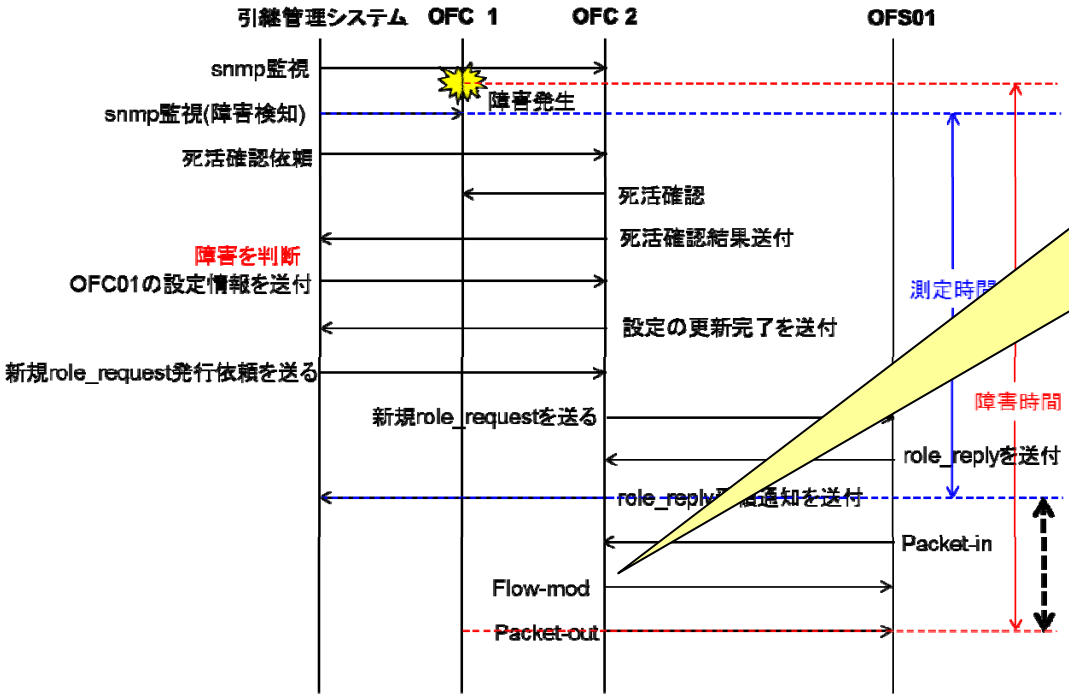
5. サイト間における冗長化について

6. 今後の課題

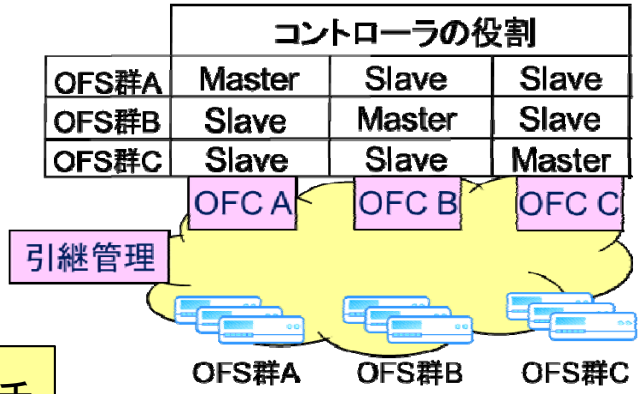
7. 考察とまとめ

■今後の課題
・断時間の測定を行う場合にスイッチの負荷をなるべく小さくするようなflow_modの送付が必要

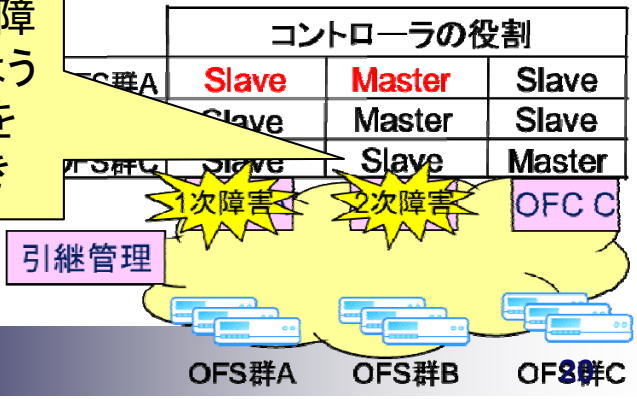
・複数台のコントローラで冗長化を図る場合
引継先のコントローラが管理スイッチが増えることによる高負荷状態を回避しなくてはならない
- コントローラのCPU / 管理スイッチ台数　等の管理



大量のflow_modを送付することでSWの負荷が高騰する可能性がある



管理スイッチが増えることによる2次障害を防ぐような仕組みを考えるべき



7. 考察とまとめ

■まとめ

・OpenFlow Specification1.2より規定されている「multiple controller」の機能を用いる事により OpenFlow Controllerの冗長が可能である。

・同一サイト内でMaster/Slave(Active/Standby)での冗長を行った場合、コントローラが保持するエントリー数に依らず約15ms程度で役割の変更が可能であり、OpenFlowネットワークにおいても高可用性なネットワークが構築できる可能性を示した。

・引継を管理する第3者サーバを設置することで、サイト間の冗長を可能にすることができる。

・サイト間の冗長においては設定情報を引継ぐという動作が発生するため、エントリー数の増加により引継時間が増加する。現状の実装では10000エントリーで約278msであり現行技術の検知時間と同等レベルを維持できる。

・今後の課題としてflow_modの送付の仕方をどのように設計すべきか、またコントローラの引継における2次障害を防ぐ工夫を施す必要があると考える。