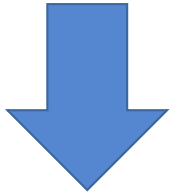
Software-Defined Networking におけるコントローラとスイッチの割り当てに関する検討

東京都市大学　大学院
佃　優作　塩本 公平　森田 達也　林 經正

背景

IoTの発展やICTの多様化により柔軟で拡張性のあるネットワークが求められている.



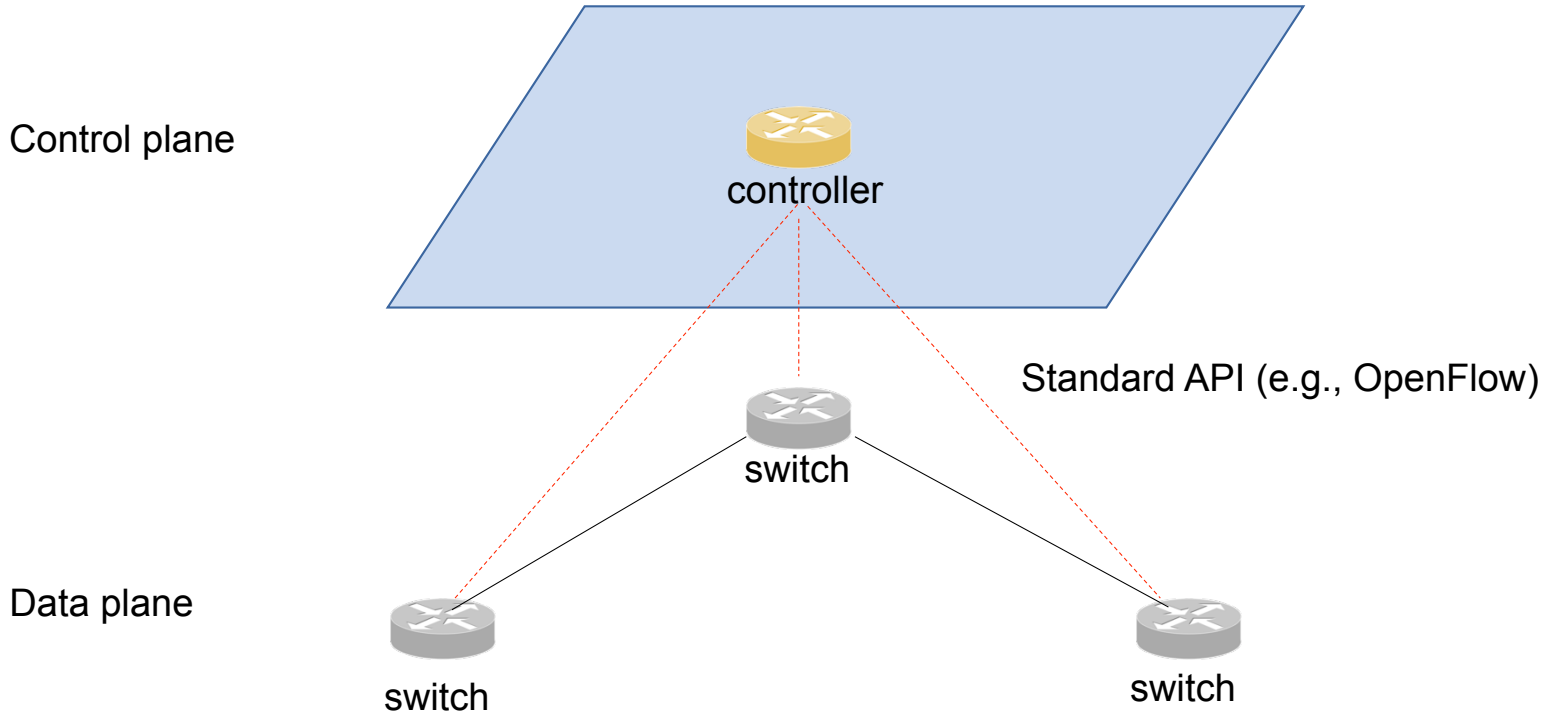
そのようなネットワークを実現するためにSDN(Software-Defined Networking)が注目されている.

背景：
Software-Defined Networkingとは？

SDNとはソフトウェアでネットワークを定義すること.

ネットワーク機器の制御機能とデータ転送機能を分離させることで，複数のネットワーク機器をまとめて管理することができる.

背景：
Software-Defined Networkingとは？



背景：
なぜSDNなのか？

従来のネットワークでは，ネットワーク機器によって独自のソフトウェアが定義されている.

→ネットワークの構築や変更をする際，多くの機器の設定が必要

SDNでは，ネットワーク機器をコントローラでまとめて管理している.

→ネットワークの構築や変更も柔軟に行える

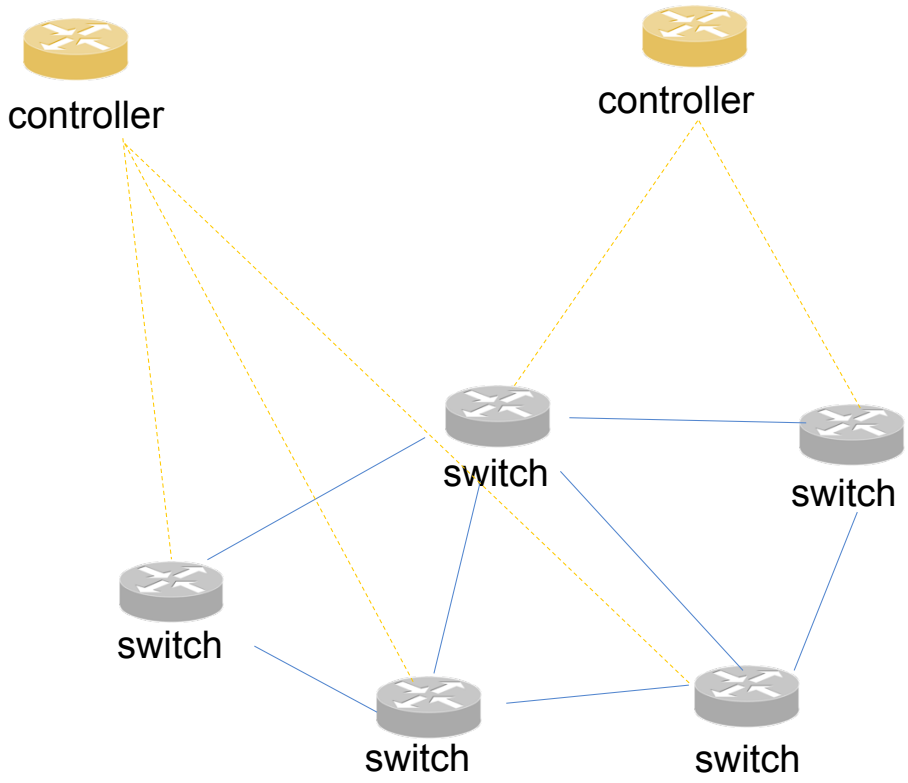
背景：
コントローラ配置問題とは

一つのコントローラで全てのネットワーク機器を集中制御することは不可能である.

物理的な複数のコントローラで集中制御する必要がある.

背景：
コントローラ配置問題とは

コントローラとスイッチ
対応関係を位置付ける



背景：
コントローラ配置問題とは

複数のコントローラを使用する場合，以下のことが重要になる.

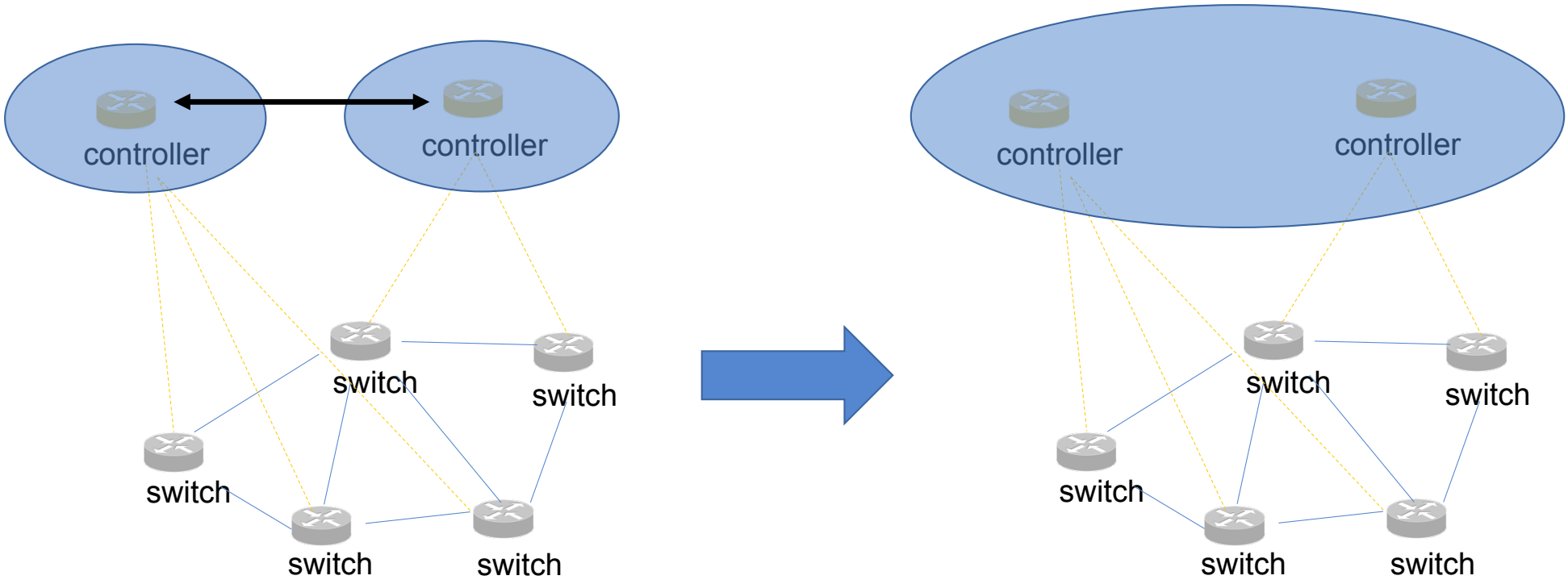
・どこにコントローラを配置するか
・どのスイッチをコントローラに制御させるか

これらを考慮することはコントローラ配置問題として扱われる.

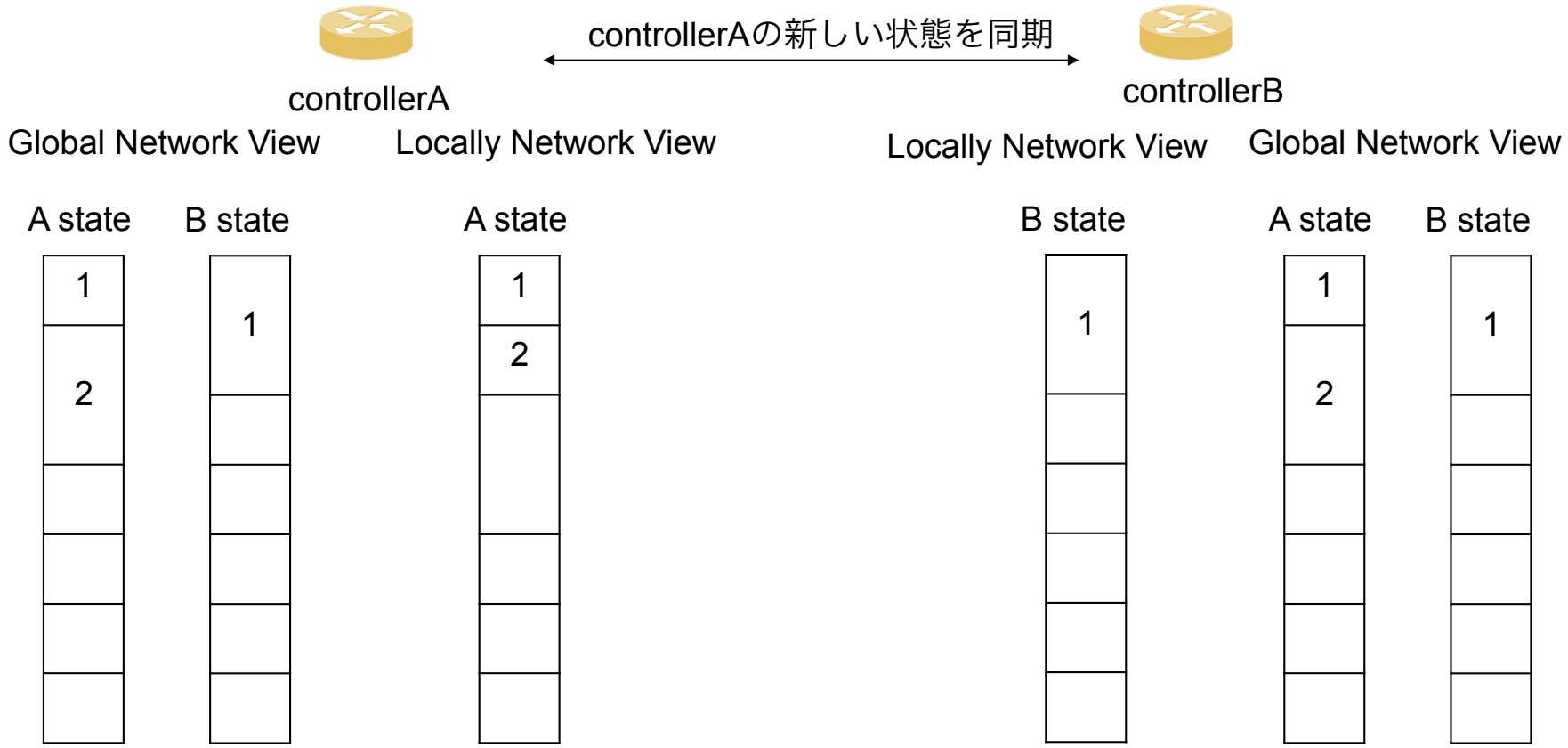
解きたい課題

複数のコントローラを使用する場合，各コントローラ同士は自身の制御するスイッチの情報を交換して同期しなければならない.

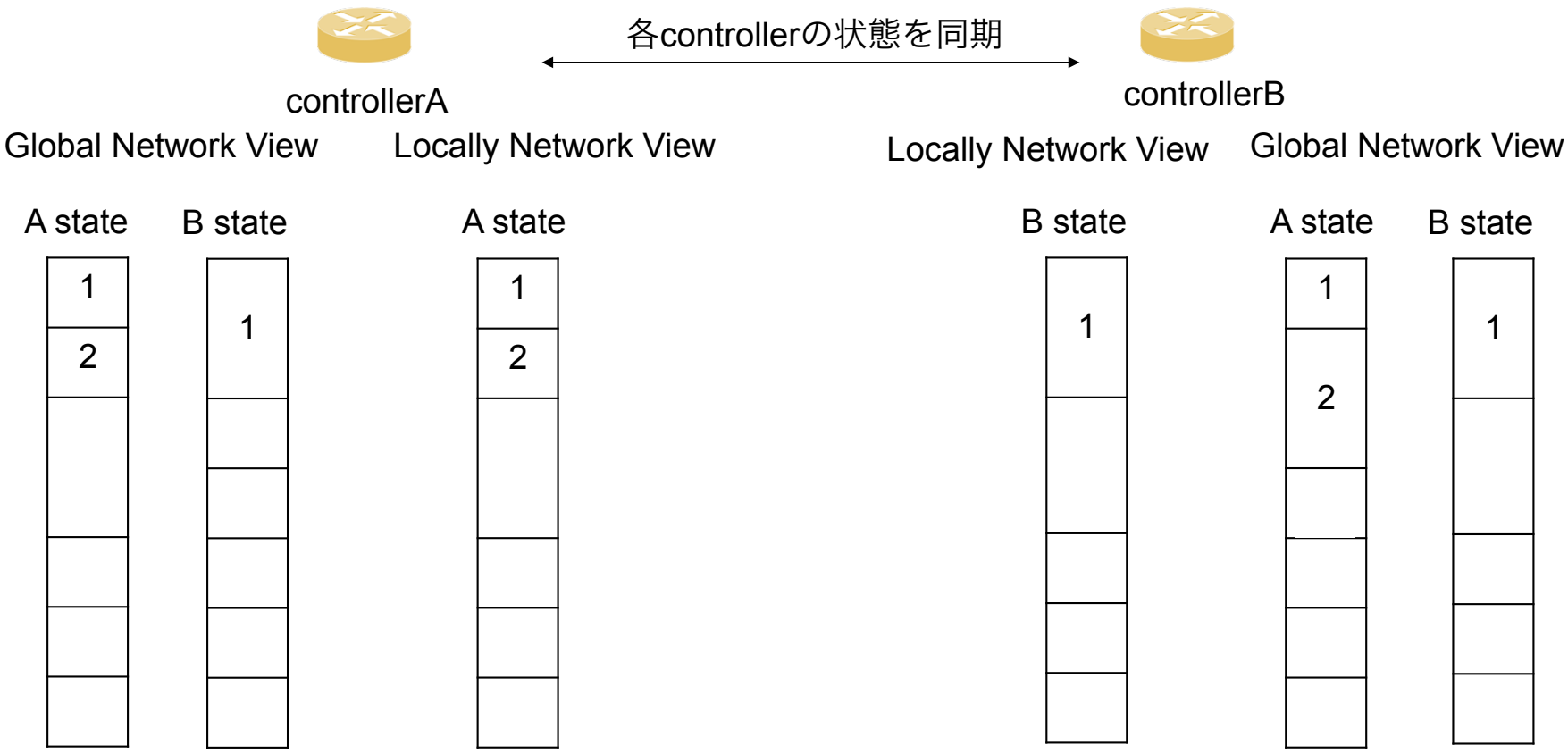
解きたい課題


controller
controller
switch
switch
switch
switch
switch
controller
controller
switch
switch
switch
switch
switch

Strongly Consistent

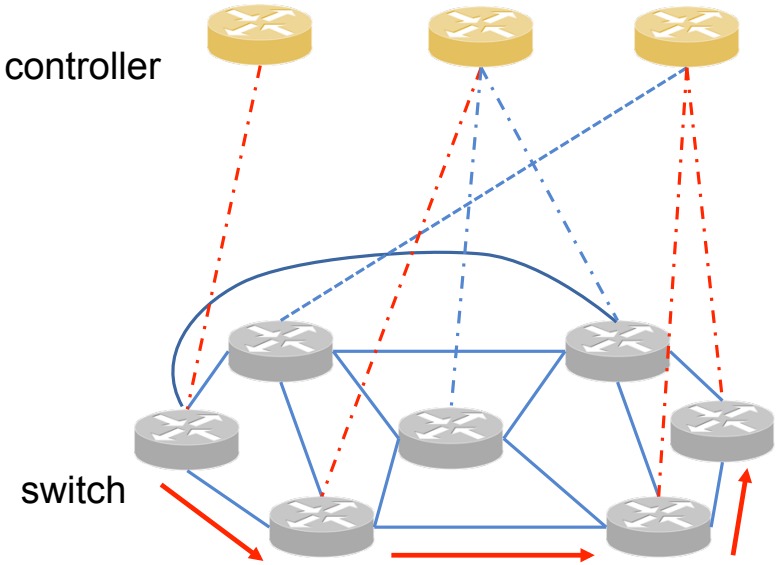


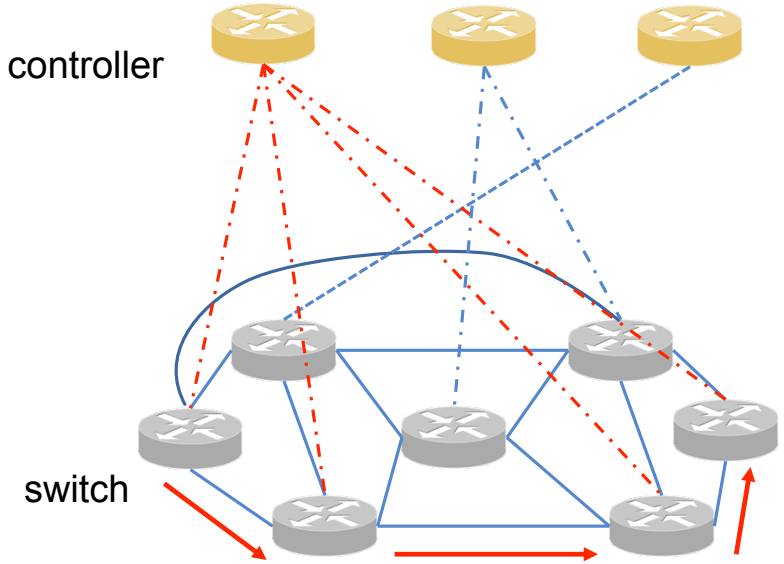
Eventually Consistent



アイデアの方向性

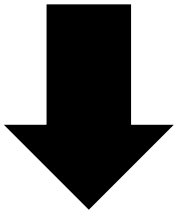
できる限り同期が少なくするコントローラの割り当てを行う.
→スイッチ同士のトラフィック量に基づいた割り当て





スイッチのクラスタリング

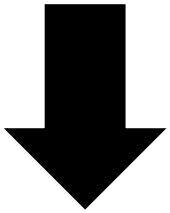
トラフィック量の多いスイッチ群をコントローラに割り当てるためには，各スイッチをクラスタリングしなけらばならない.



Infinite Relational Modelを用いる

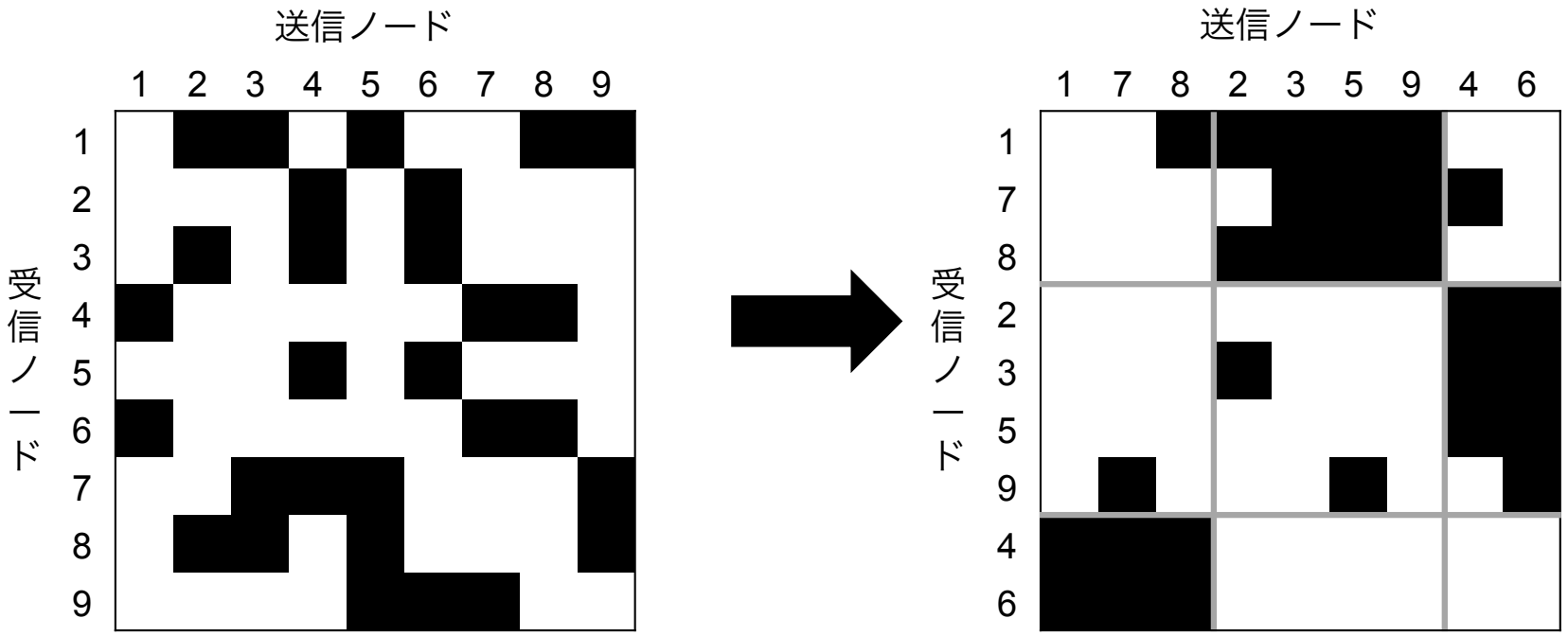
Infinite Relational Model

IRMは複数の要素を関係性があるグループとないグループにクラスタリングを行うことができる.



観測データにトラフィック量を用いることで，通信頻度が多いスイッチのグループを作成する.

Infinite Relational Model



予備実験の内容と結果・考察

スイッチ数が8つ，コントローラ数が２つでトラフィックの重みを以下のようにする.

	#0	#1	#2	#3	#4	#5	#6	#7
#0	0	5	5	5	1	1	1	1
#1	5	0	5	5	1	1	1	1
#2	5	5	0	5	1	1	1	1
#3	5	5	5	0	1	1	1	1
#4	1	1	1	1	0	5	5	5
#5	1	1	1	1	5	0	5	5
#6	1	1	1	1	5	5	0	5
#7	1	1	1	1	5	5	5	0

このときの同期が必要な確率を，ランダムにコントローラを配置した場合と，トラフィックに基づいて配置した場合のそれぞれで算出.

予備実験の内容と結果・考察

$$\sum_{j\in P} y_j = \rho$$

$$\sum_{j\in P} x_{ij} = 1 \qquad \forall i \in S$$

$$x_{ij} \le y_j \qquad \forall i \in S$$

$$Z \ge \sum_{k\in P, k\neq j} t_{jk} \qquad \forall j \in P$$

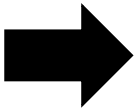
$$y_j,\ x_{ij} \in \{0, 1\} \qquad \forall i \in S \qquad \forall j \in P$$

予備実験の内容と結果・考察

同期確率$=\sum_{k\in P, k\neq j} t_{j} / \sum_{k\in P} t_{jk}$ $\forall j\in P$

予備実験の内容と結果・考察

	#0	#1	#2	#3	#4	#5	#6	#7
#0	0	5	5	5	1	1	1	1
#1	5	0	5	5	1	1	1	1
#2	5	5	0	5	1	1	1	1
#3	5	5	5	0	1	1	1	1
#4	1	1	1	1	0	5	5	5
#5	1	1	1	1	5	0	5	5
#6	1	1	1	1	5	5	0	5
#7	1	1	1	1	5	5	5	0



	#0	#1	#2	#3	#4	#5	#6	#7
#0	0	1	1	1	0	0	0	0
#1	1	0	1	1	0	0	0	0
#2	1	1	0	1	0	0	0	0
#3	1	1	1	0	0	0	0	0
#4	0	0	0	0	0	1	1	1
#5	0	0	0	0	1	0	1	1
#6	0	0	0	0	1	1	0	1
#7	0	0	0	0	1	1	1	0

予備実験の内容と結果・考察

	#6	#7	#4	#5	#0	#1	#3	#2
#6	0	1	1	1	0	0	0	0
#7	1	0	1	1	0	0	0	0
#4	1	1	0	1	0	0	0	0
#5	1	1	1	0	0	0	0	0
#0	0	0	0	0	0	1	1	1
#1	0	0	0	0	1	0	1	1
#3	0	0	0	0	1	1	0	1
#2	0	0	0	0	1	1	1	0

予備実験の内容と結果・考察

ランダム配置

	コントローラ0	コントローラ2
接続スイッチ	0,1,7,5	2,3,4,6
同期確率	0.61	0.69

提案配置

	コントローラ0	コントローラ4
接続スイッチ	0,1,2,3	4,5,6,7
同期確率	0.23	0.23

おわりに

トラフィック量の多いスイッチ群をコントローラに管理させることで，同期数を削減できることが確認できた.

今後は実環境に近づけるため，トラフィック生成モデルやスイッチ間の経路を考慮し，仮想的なネットワークでの実験を行っていく予定である.