テレコムキャリア網を前提に考えていたSDNのあれこれ

堀場 勝広

Agenda

テレコムキャリア網で「広義のSDN」を使いたかった理由

広義のSDNを構成する3要素（ID、**Policy、Topology**）

広義のSDNにおける「宣言型のネットワーク運用」の関係性

広義のSDNにおいて宣言する「Policyの一貫性」の重要性

Policyの一貫性を担保するための「理論：Promise Theory」

Policyの一貫性を担保するための「実践：Group Based Policy」

2009年ごろ、はじめてOpenFlowを見て思ってたこと

「コレを運用するの怖くない？」

コントローラとスイッチの間が切れたらどうなるの？

- 切れること自体が問題ではなく、その間にトポロジが変わった場合の問題？

OAMとか生存確認ってどうするの？

- NMSがあるシステムはIn-band OAMを持ってるが、まさかEthterOAMとか？

それらがあるとして、ルーティングのアルゴリズムで保証されてることはどう担保するの？

- 理論的にはループフリーであること、一定時間で代替経路に収束すること等
- 自分で一から書くってわけじゃないよね？（まず「ルータとは何か」から紐解くこと[1]）

[1] TremaでOpenFlowプログラミング ＃12 ルータ (前編)
https://yasuhito.github.io/trema-book/#router_part1

テレコムキャリア網の特徴

（Enterprise/DCネットワークと比較して）

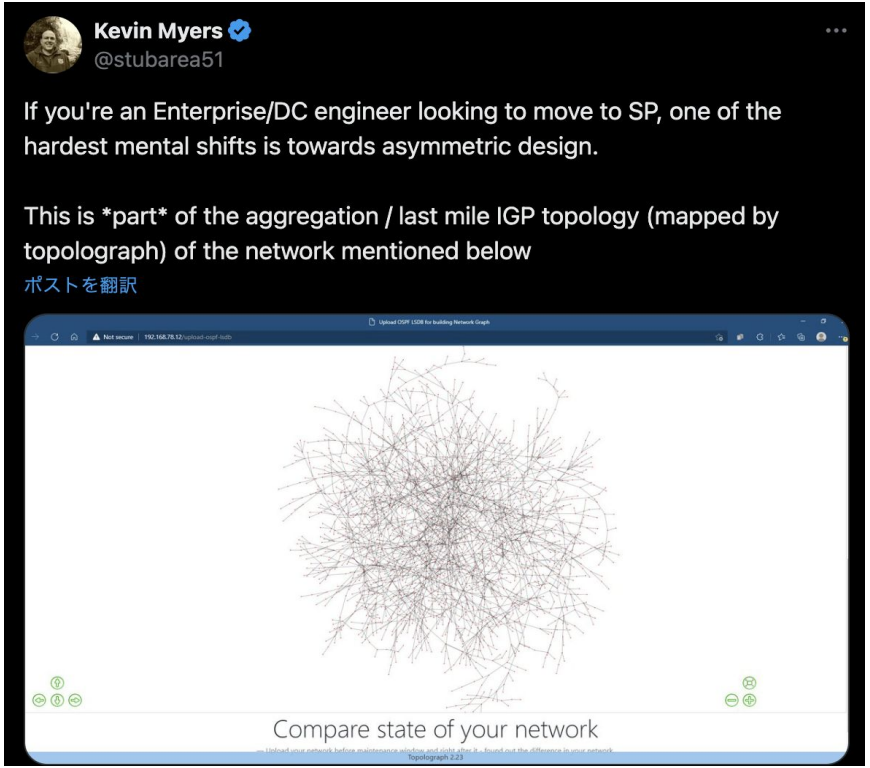
- 物理的・地理的に広く分散した拠点
- 複雑なトポロジ（非対称な形）
- 尋常ではないノード数（数千~万）
- Out-of-Band接続を持ちにくい
- 駆けつけメンテナンスが超大変

最も困るのは物理的に繋がってのに通信断

人間の頭でパッと代替経路が想像しにくい

→多少の非最適経路があっても到達性が命

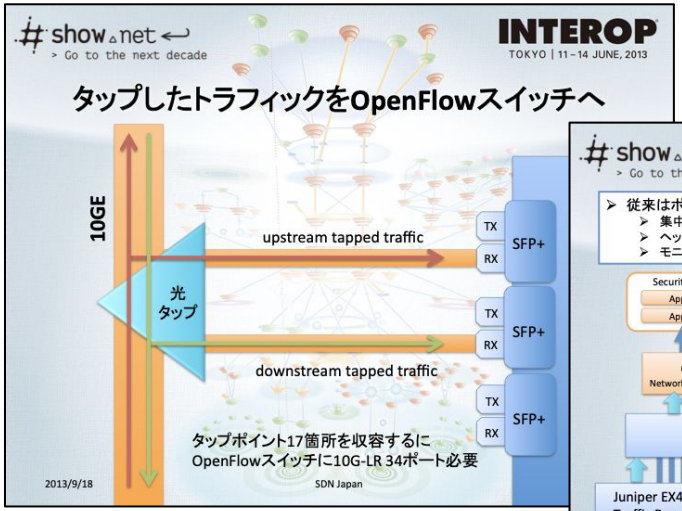
→自律分散協調型の「広義な SDN」が適当

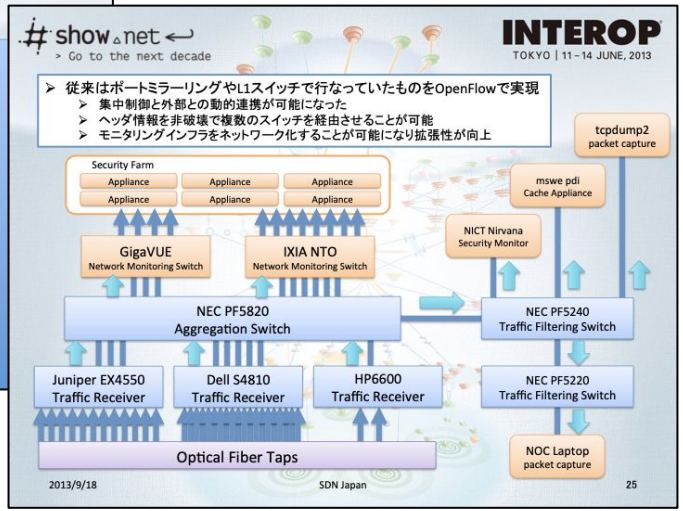


https://x.com/stubarea51/status/1441017036977680384

今なら分かるOpenFlowの面白さ

回線交換的なドメインを選べば便利なものである



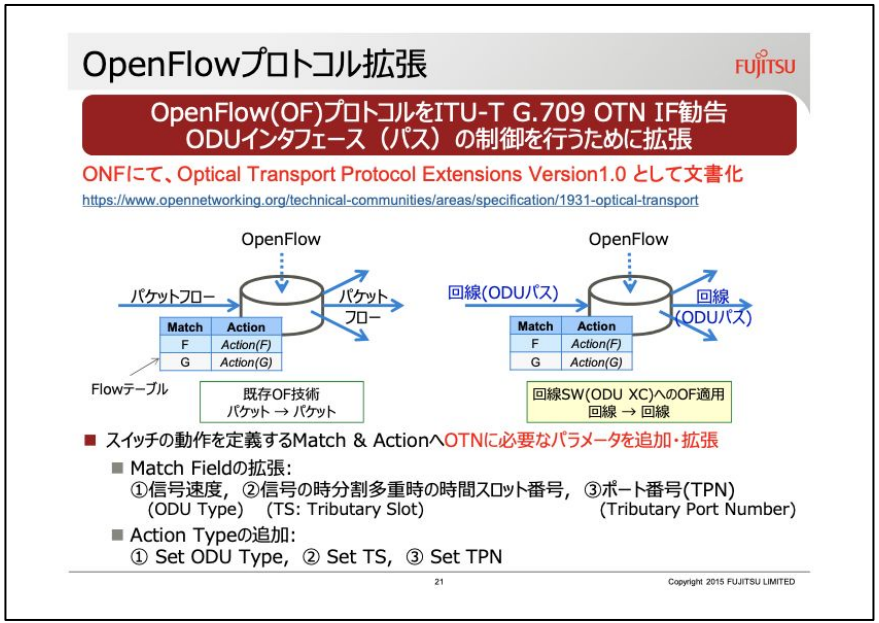


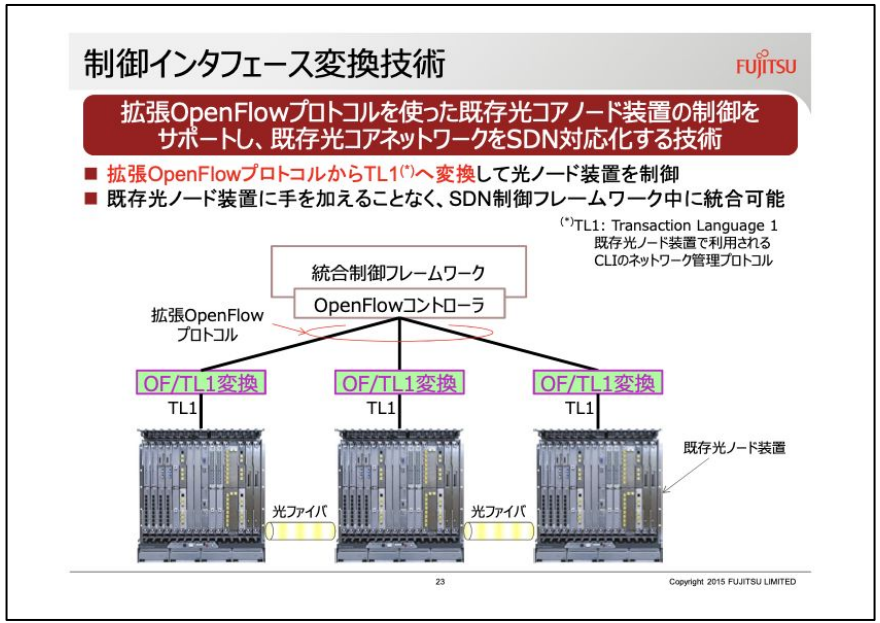
https://www.onic.jp/archives/2013/slide/0918_09noc-2.pdf

https://www.arista.com/assets/data/pdf/Whitepapers/Broadcast_Transition_from_SDI_to_Ethernet_jp.pdf

今なら分かるOpenFlowの面白さ（Cont'd）

光伝送の人にはドンピシャな技術だった？（TL-1の置き換えのようにも見えた





https://o3project.github.io/ja/event/20151021/04_光トランスポートネットワーク向け SDN技術の紹介 .pdf

偉大な先人の教え(1)

ONIC 2019 Day2

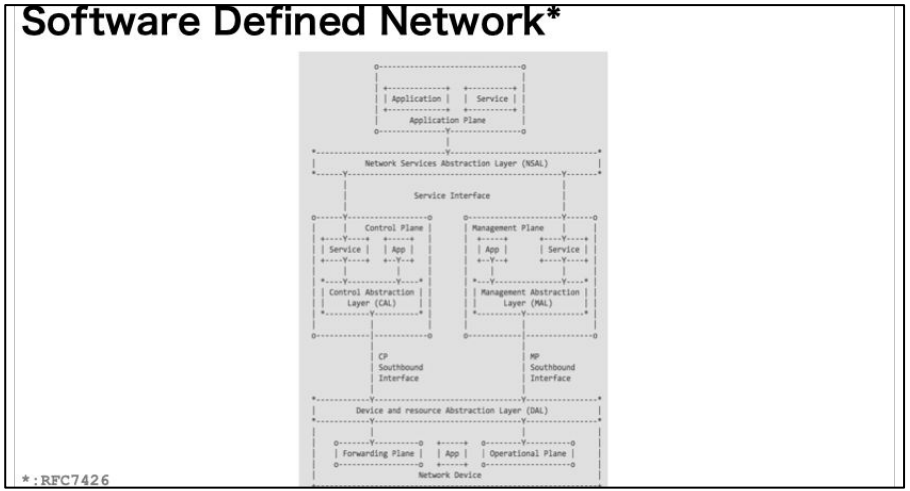
Software Defined Networkへの道のり(SoftBank 松嶋さん)

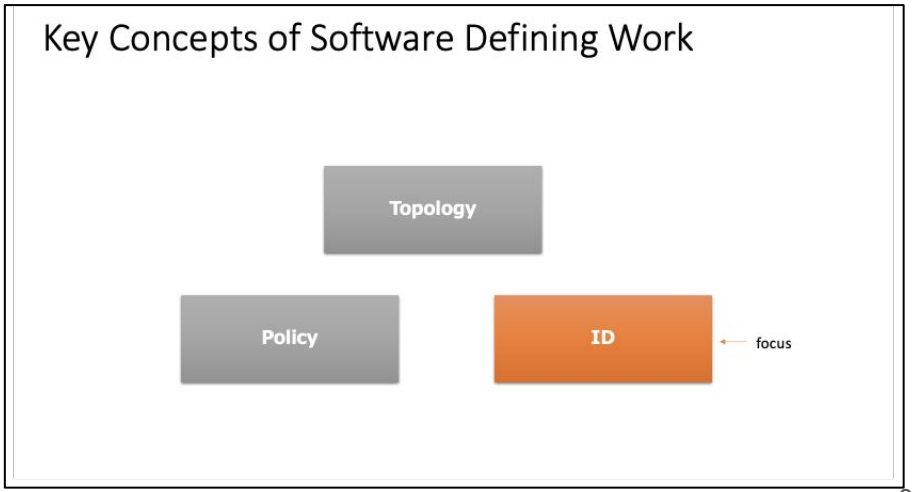
https://www.onic.jp/archives/2019/slide/onic2019_matsushima.pdf

（広義の）SDNを構成する3大要素

IETFの定義する（広義の）SDNではControl PlaneとManagement Planeが併設

- そこで交換される重要な要素はTopology、Policy、IDの3つである
- このプレゼンテーションでは「ID」に注目

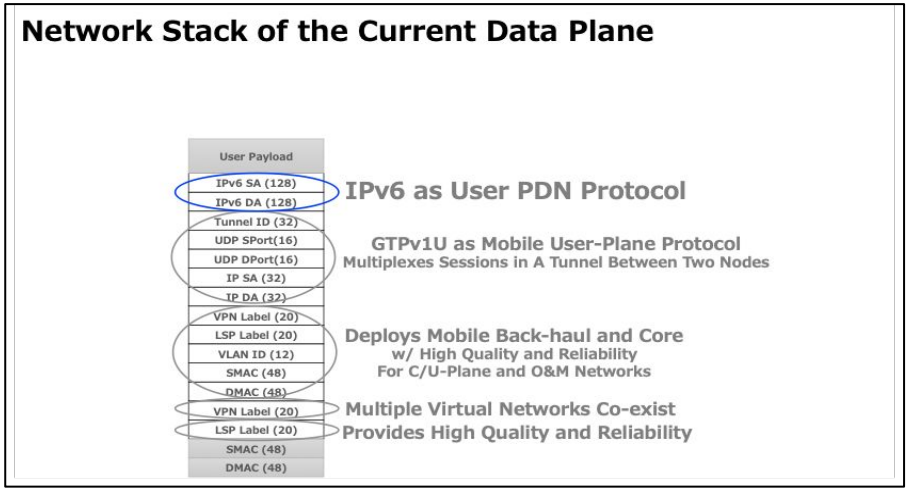


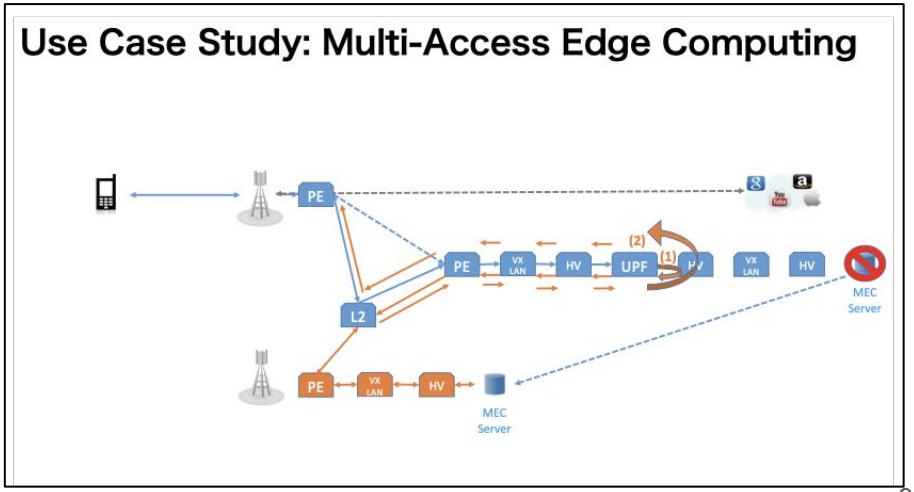


IDが乱立してると何が起こるのか

テレコムキャリア網が長く抱える悩みの源泉

- NNIにおける多段オーバーレイネットワークの最適化という地獄への入り口
- UNI接続における回線の縫い合わせを静的に管理するエクセルの無限地獄

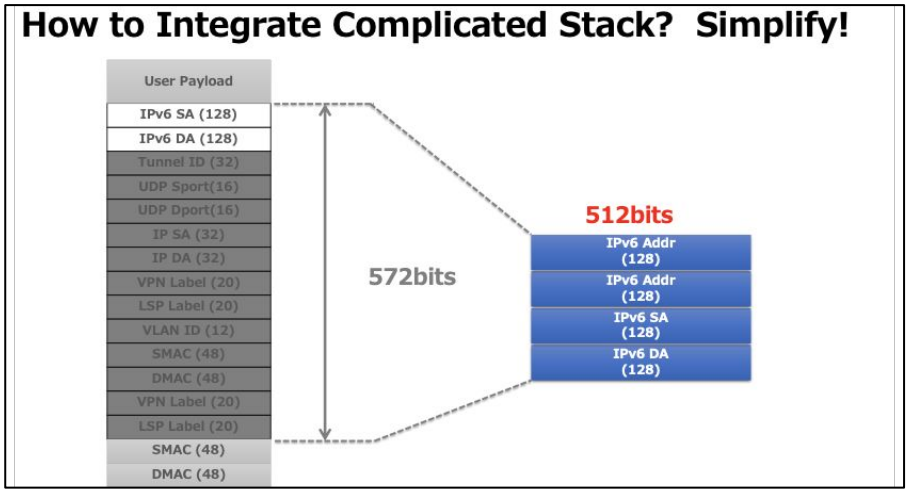


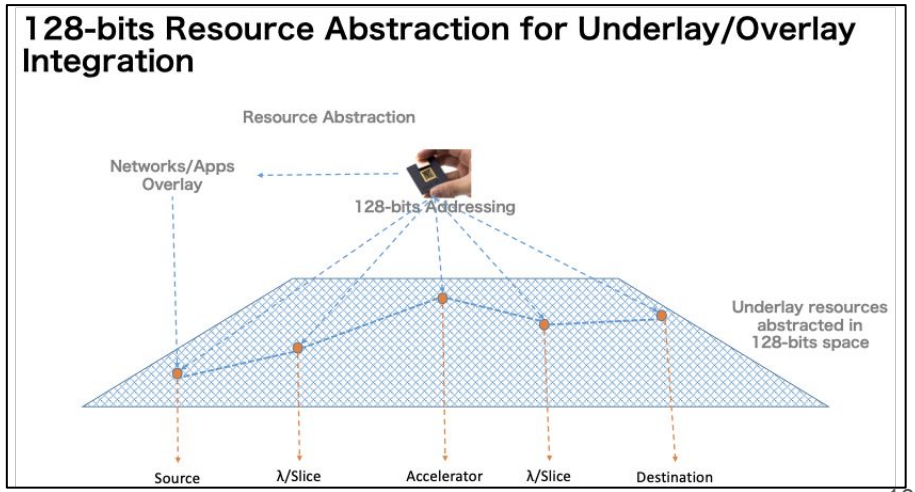


IDを統一するということの重要性

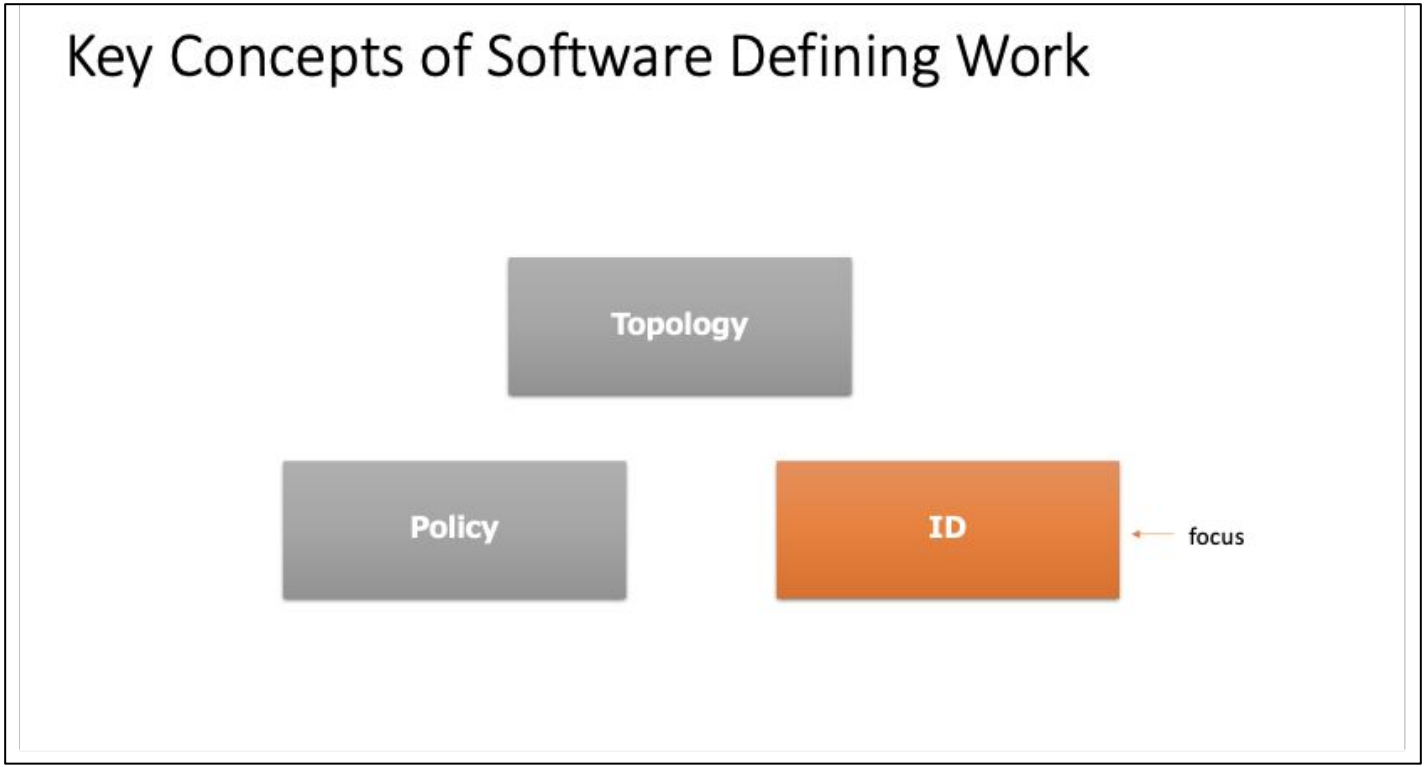
一枚のID空間の中でドメイン内のパス（経路の集合）、アクションを表現しきる

- SRv6とNetwork Programmingに至った気持ちがよく分かる話
- Ingress-Egress(Locator)間トラフィック制御、EgressにおけるD-Plane(ID)処理

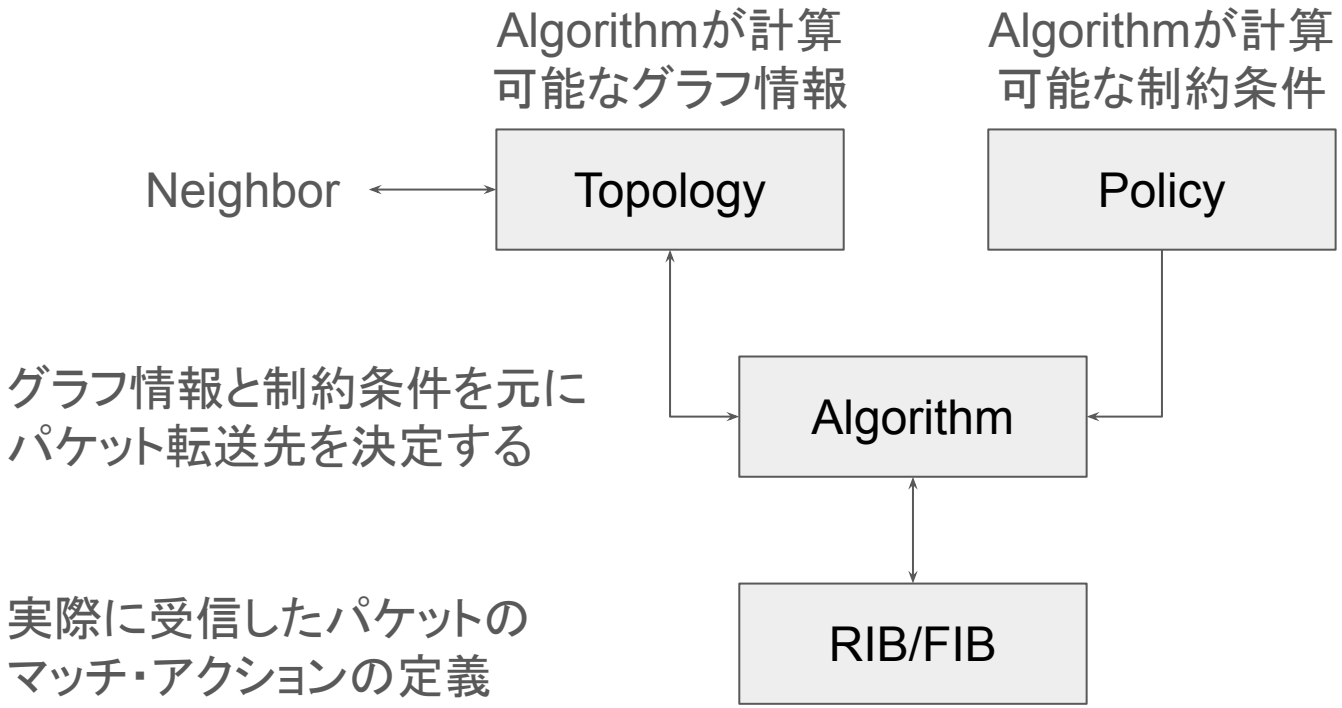




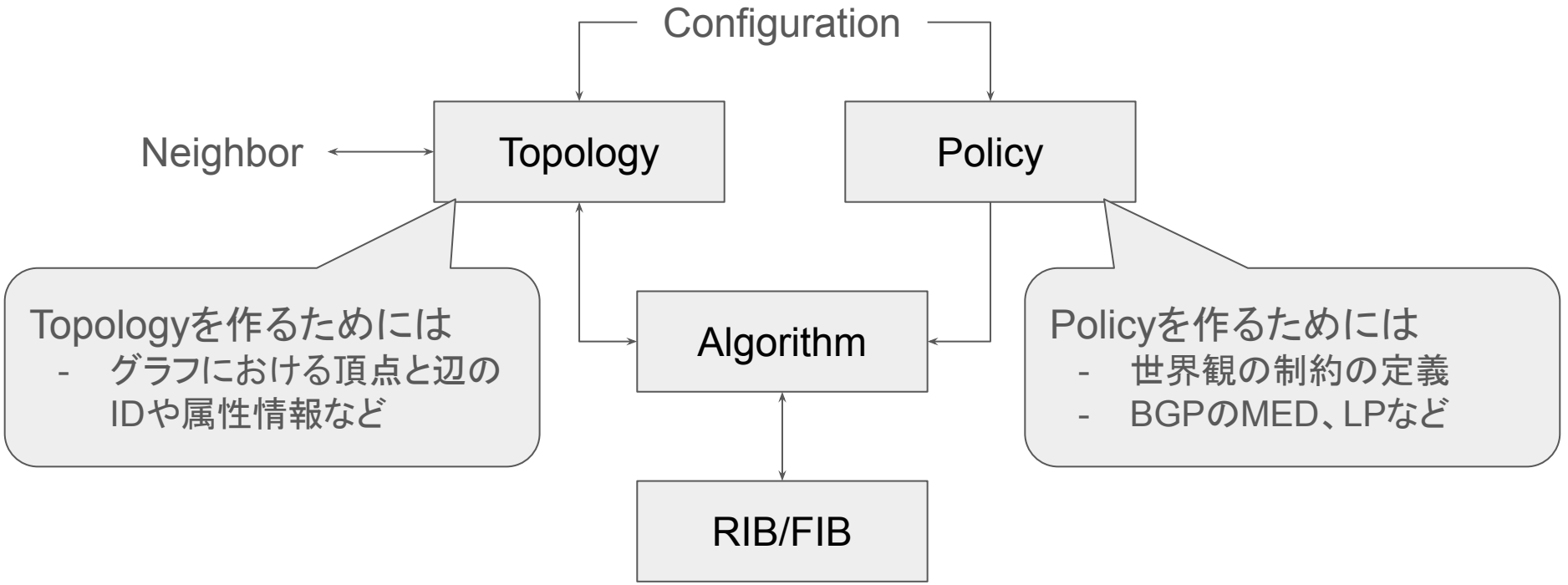
さて、残りの2つ（ToplogyとPolicy）について



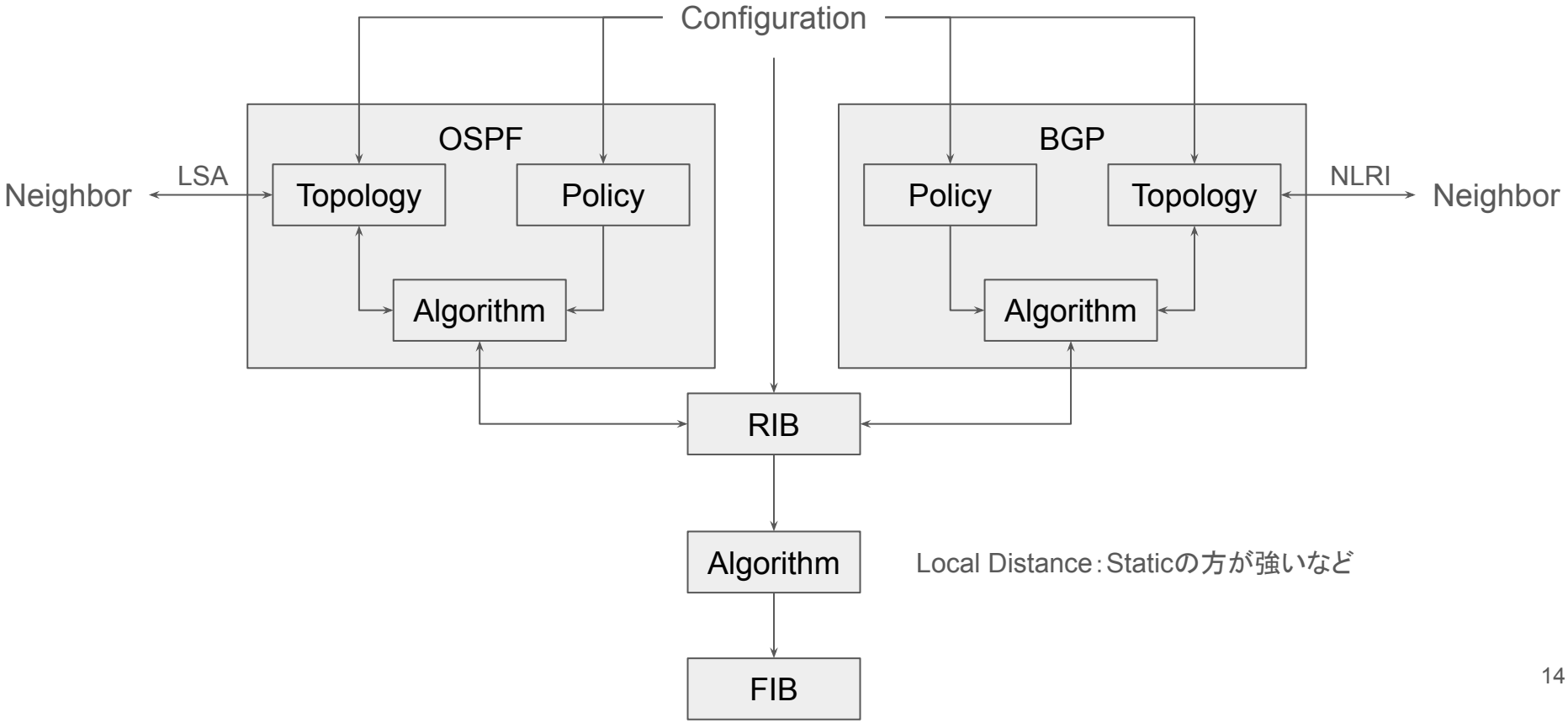
広義のSDNにおける自律ノードの中で起こっていること



広義のSDNにおける静的なConfigurationの役割



（余談）実際にはRouting Protocolの世界はもう少し複雑



偉大な先人の教え(2)

JANOG 47 Day2

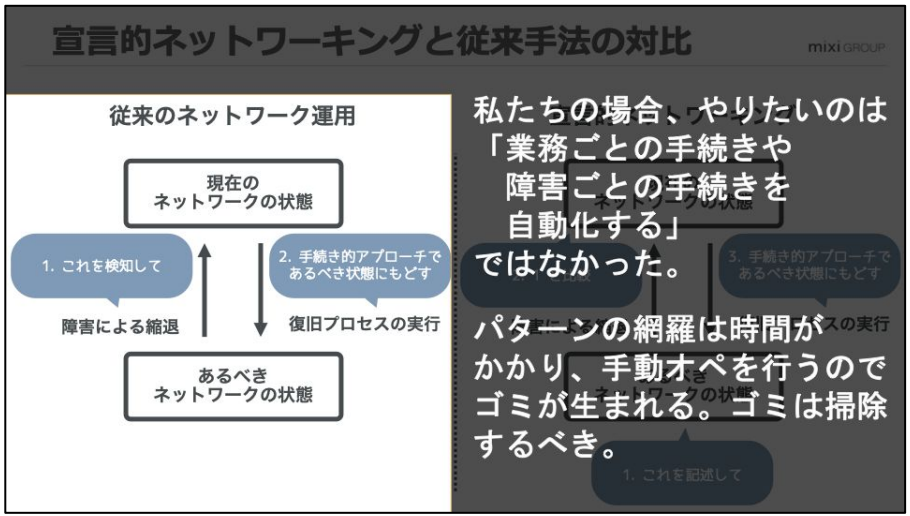
走りながら作るネットワークテンプレートシステム(Mixi 小島さん)

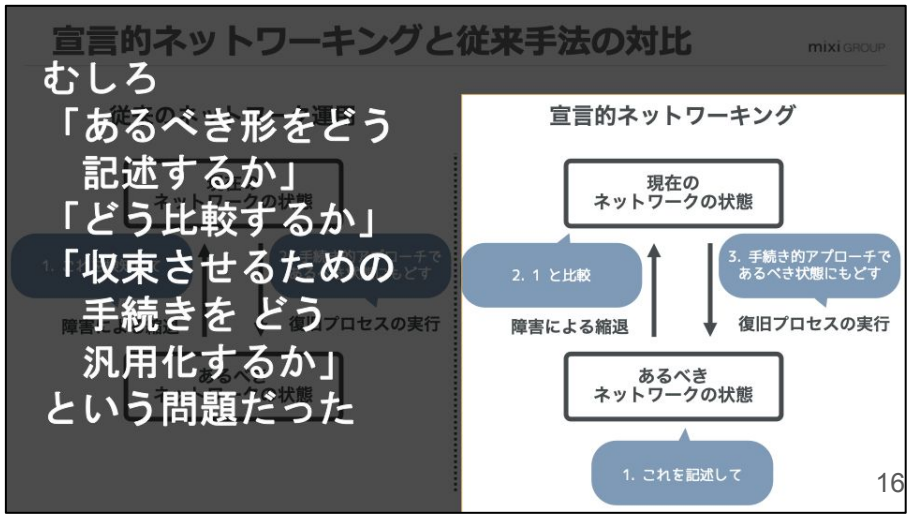
https://www.janog.gr.jp/meeting/janog47/template/

手続き型から宣言型のネットワーク運用へ

運用の自動化、Closed Loopを実現したいと思った時

手続き（手順）を自動化よりも**あるべき形の記述 + 収束に向けた手続きの汎用化**



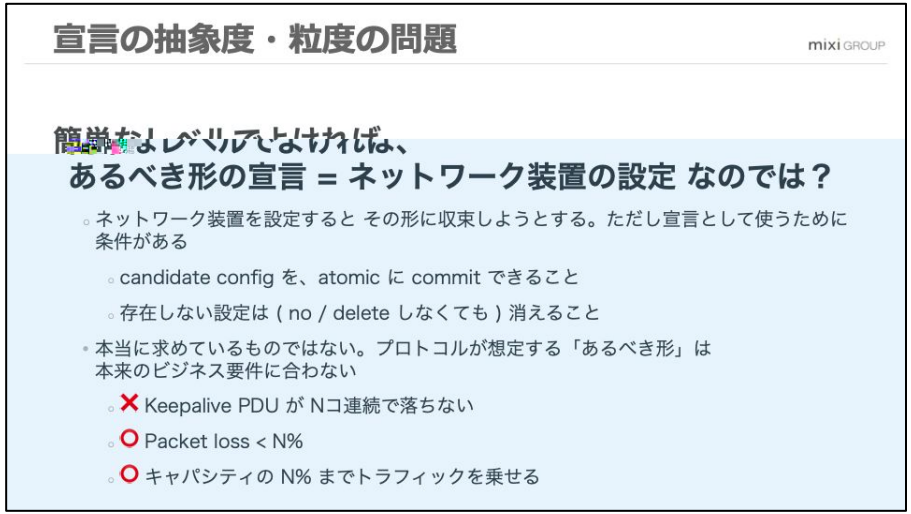


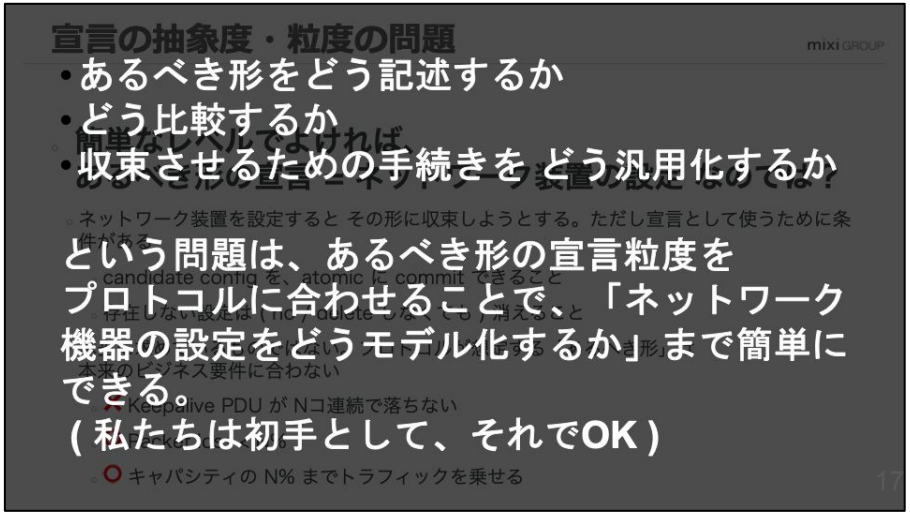
宣言型のネットワーク運用とは？

（広義のSDNの場合）「あるべき形の宣言 = 装置の設定」と考えて良いかも

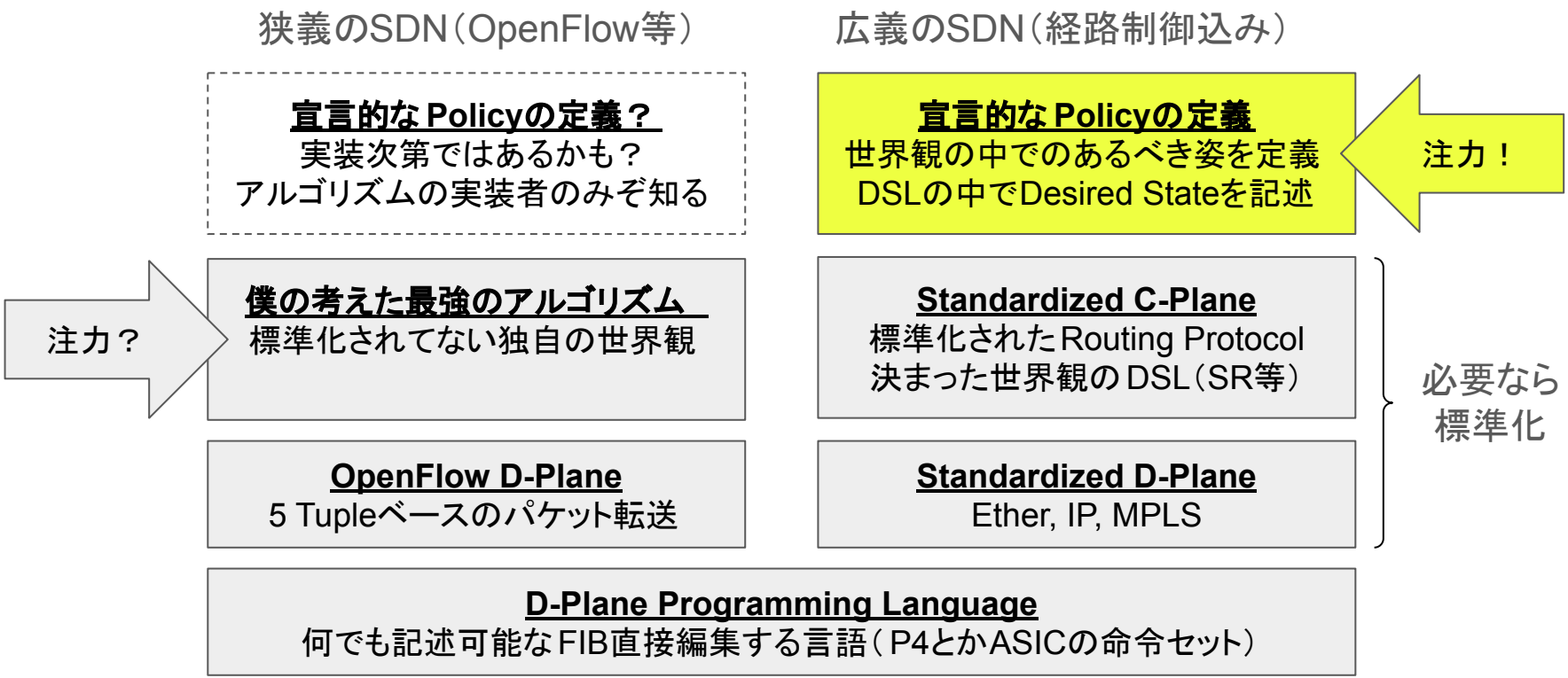
装置にプロトコルの設定をすると、その形に収束しようとする

ただし・・・規定されたプロトコル内で表現できることしかできない



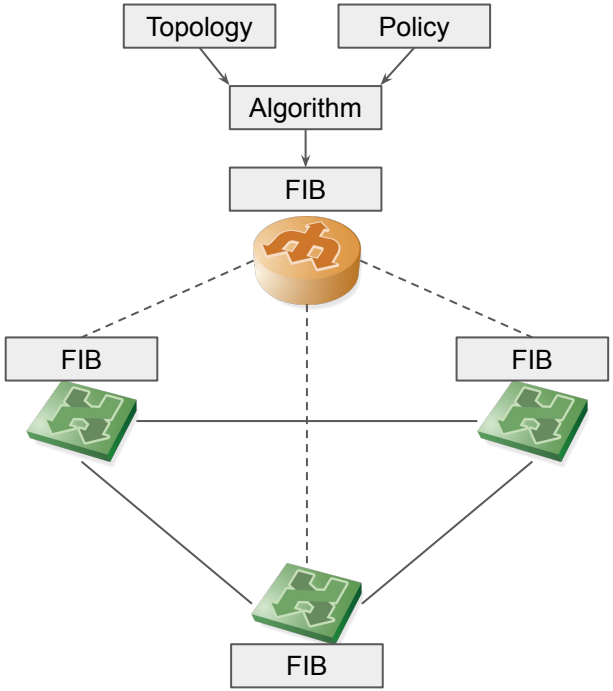


広義のSDNで注力すべき場所は「宣言的なPolicyの定義」

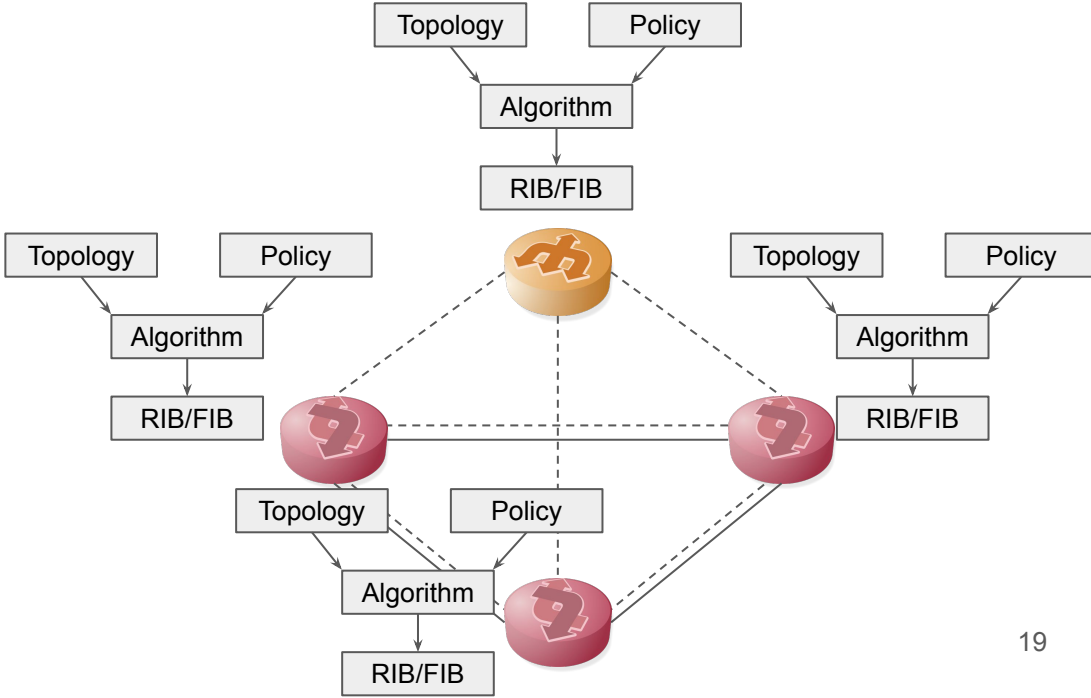


狭義のSDNと広義のSDNにおけるPolicyの取り扱い

狭義のSDN（中央集権システム）
代表者がTopologyとPolicyから全員のFIBを生成
ノードは配布されたFIBを適用するのみ



広義のSDN（自律分散＋コントローラ）
自律ノードは共通の Algorithmを持ちTopologyを交換
個々が持つPolicyを掛け合わせてRIB/FIBを生成



自律分散システムでは「Policyの一貫性」が超重要！

狭義のSDN　コントローラがTopologyの変化とAlgorithmを一元管理

コントローラとノード群のTopologyの一貫性が保てないとFIBに不整合

広義のSDN　自律ノード間でTopologyの変化とAlgorithmを共有

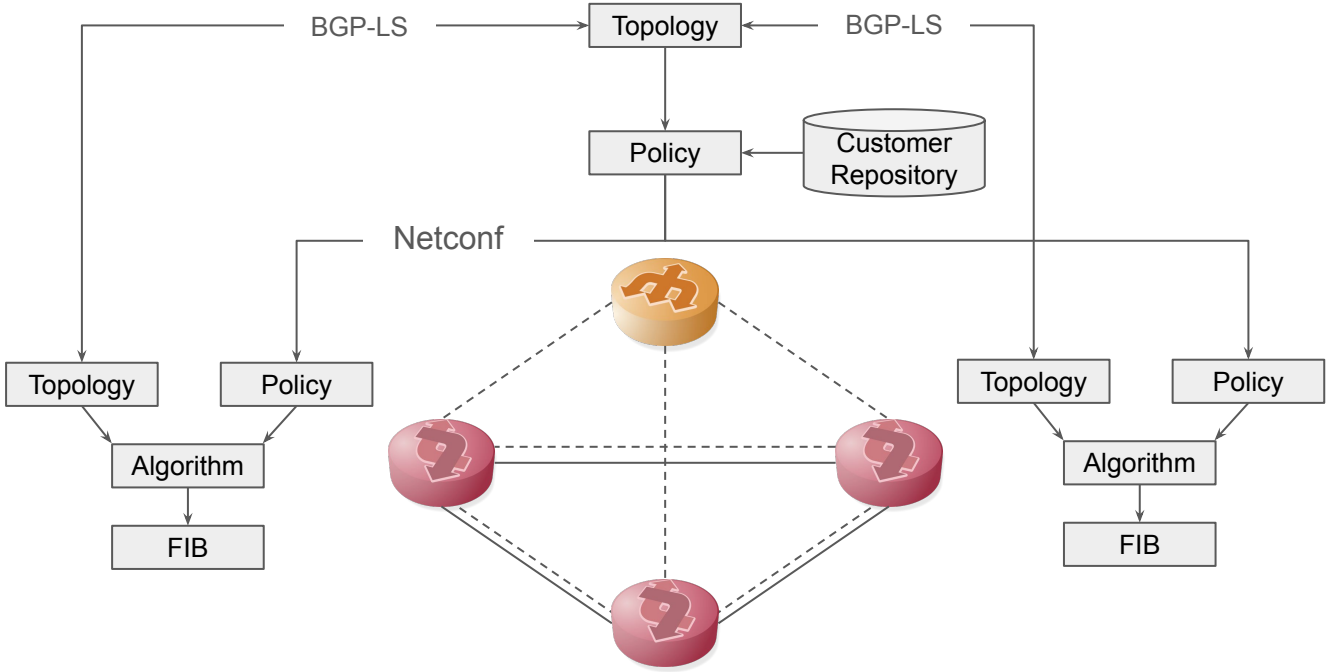
自律ノード間でPolicyの一貫性が保てないとFIBに不整合

極端な話コントローラは一貫性のあるPolicyを配り終えたら不要

	狭義のSDN（中央集権システム）	広義のSDN（自律分散＋コントローラ）
得意	コントローラにおける正しい Topologyを前提としたFIBの一貫性	自律ノード間で交換される Topologyと Algorithmの一貫性
苦手	（障害発生時の）コントローラとノード群におけるTopologyの一貫性	自律ノード間での Policyの一貫性（に紐づくFIBの整合性）

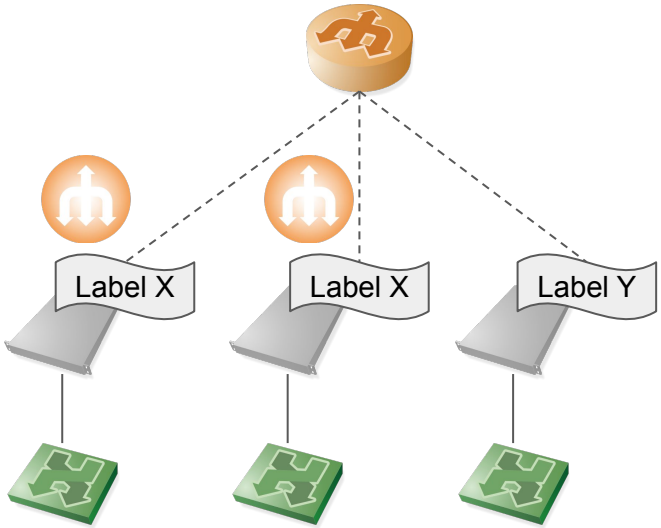
中央集権型のPolicy「一貫性」と自律分散型の経路制御

Topologyの交換とFIBの生成は自律ノードが共通アルゴリズムで実施する
一貫したPolicyを配り終えてしまえば後は自律ノードが何とかするはず



（閑話休題）Kubernetesの宣言型が普通に動く理由？

Topology = Workerであり、Podをホストする先をLabelで決める
ネットワーク的に他者のPolicyとの一貫性を考慮しなくて良い



Orchestrator

Pod

Worker

ToR

Policy(Intent)の依存関係の範囲

～ようやく本題～
Policyの一貫性を担保するには！？

そもそもPolicyって何でしょう？

語源：　ギリシャ語「πόλις（ポリス、都市国家）」

ラテン語「politia（政体、政治、公共秩序）」

英語「policy」（政策、方針、施策）

本質：　意思決定や行動のルールやガイドライン

目的：　皆がガイドラインに従って自律的に行動することで目標を達成する

用法：　ネットワーク、セキュリティ、プライバシー

Policyの構成要素

何を条件にして、どんな行動を取れば、全体系の目的が達成できるのだろうか？

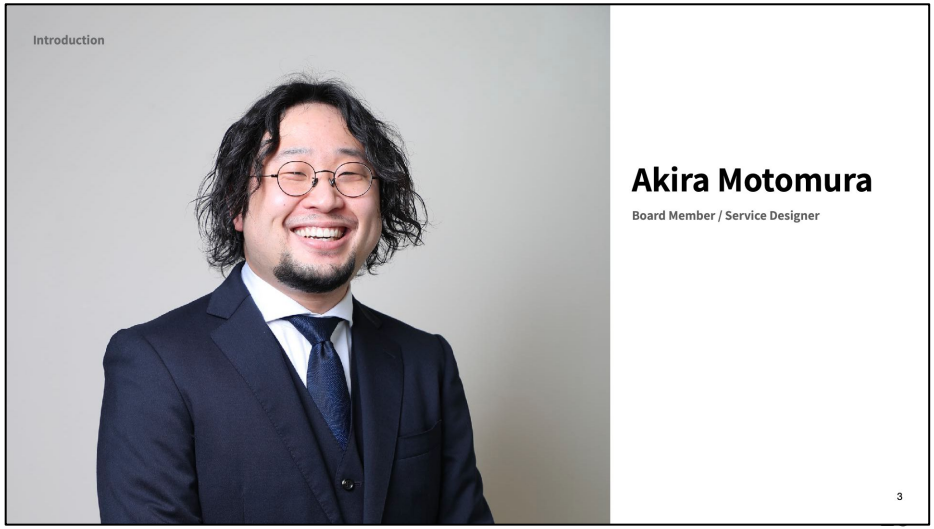
目的	**Objective**	**達成したいゴールは何か？**
条件	**Condition**	**Policyが適用される条件**
行動	**Action**	**条件が満たされた時に取る行動**
優先順位	Priority	複数のPolicyが同時に適用された場合の優先順位
適用範囲	Scope	Policyが適用される対象の明確化
柔軟性	Flexibility	状況変化への対応（条件が変わると目的達成のための行動も変わる）
規律	Compliance	目的達成に向けて起こす行動に対する制約
観測	Observation	起こした行動が目的を達成しているかの評価

偉大な先人の教え（3）

DEMATSIGNTERS’22 Tokyo

Design for trust: Applying Promise Theory to collaborate effectively with your remote teams（YUMEMI 本村さん）

https://speakerdeck.com/akiramotomura/designing-for-trust-applying-promise-theory-to-collaborate-effectively-with-your-remote-teams



Promise Theory

Promise Theory by Mark Burgess in 2004

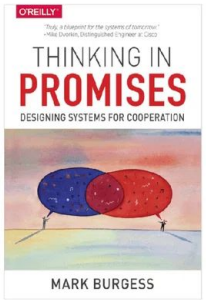
- 自律エージェント間での協力的な行動を記述する情報のモデリング言語
- コロナ禍の協調作業に応用してる話だが・・・さておき理論の説明を引用

Promise Theory

A modeling language of information to describe and discuss cooperative behavior among different agents or actors, proposed by Mark Burgess, an independent theorist and practitioner in the field of information science, in 2004.

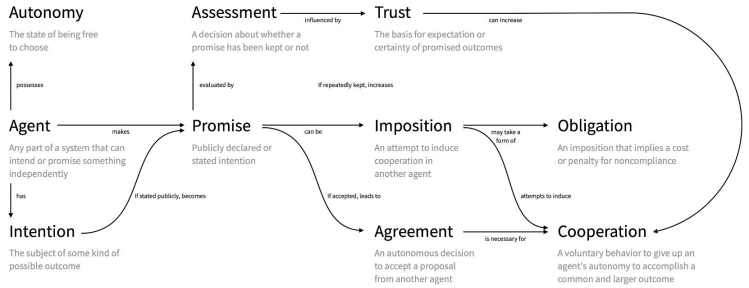
It has the capability to visualize, analyze, and solve any problems and bottle-necks of how people communicate and collaborate with each other in a formalized way.

Also, it offers a completely new way to understand the word around us.



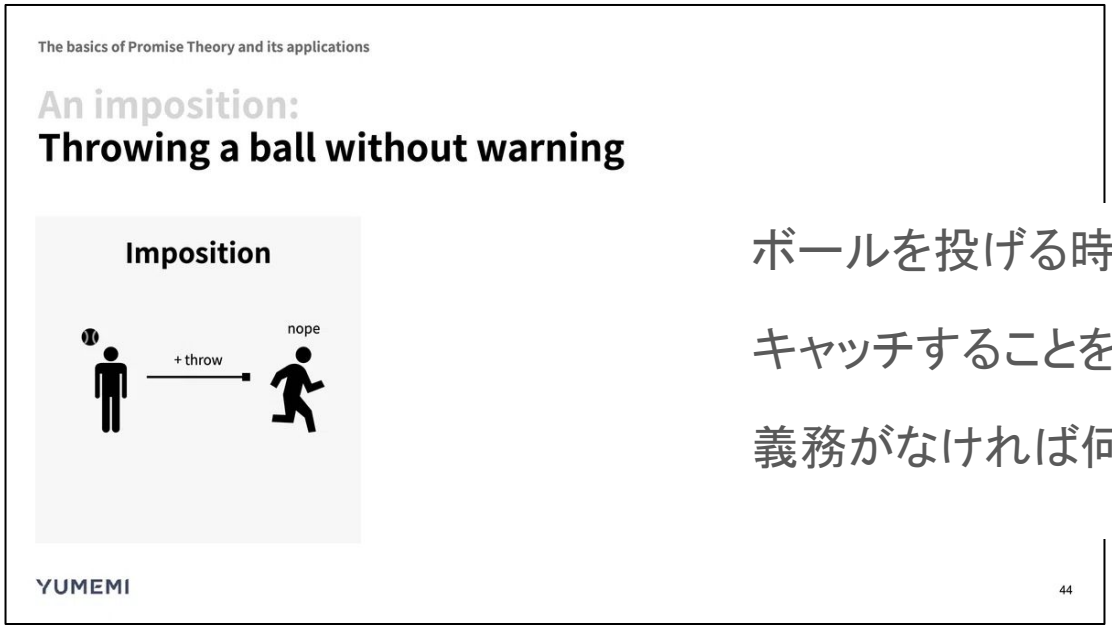
https://www.amazon.com/gp/product/B01092PYG8/ref=dbs_a_def_rwt_bibl_vppi_i1

A concept map of Promise Theory



Imposition（押し付け）

自律エージェント間では、手続きを相手に強要したとしても、
何かしら義務（コストやペナルティ）を課さねば成立しない



ボールを投げる時に相手が

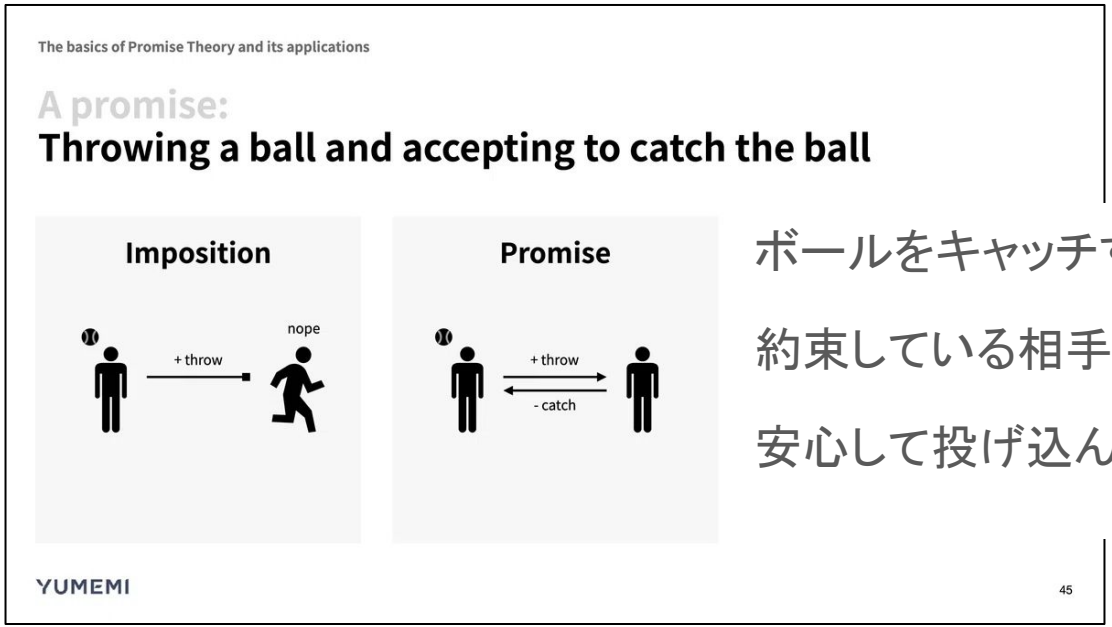
キャッチすることを強要しても

義務がなければ何もしない

Promise（約束）

自律エージェントが個々に「約束」を宣言しておけば、

「約束」しているエージェントには「要求」を投げ込んでも成立する



ボールをキャッチすることを

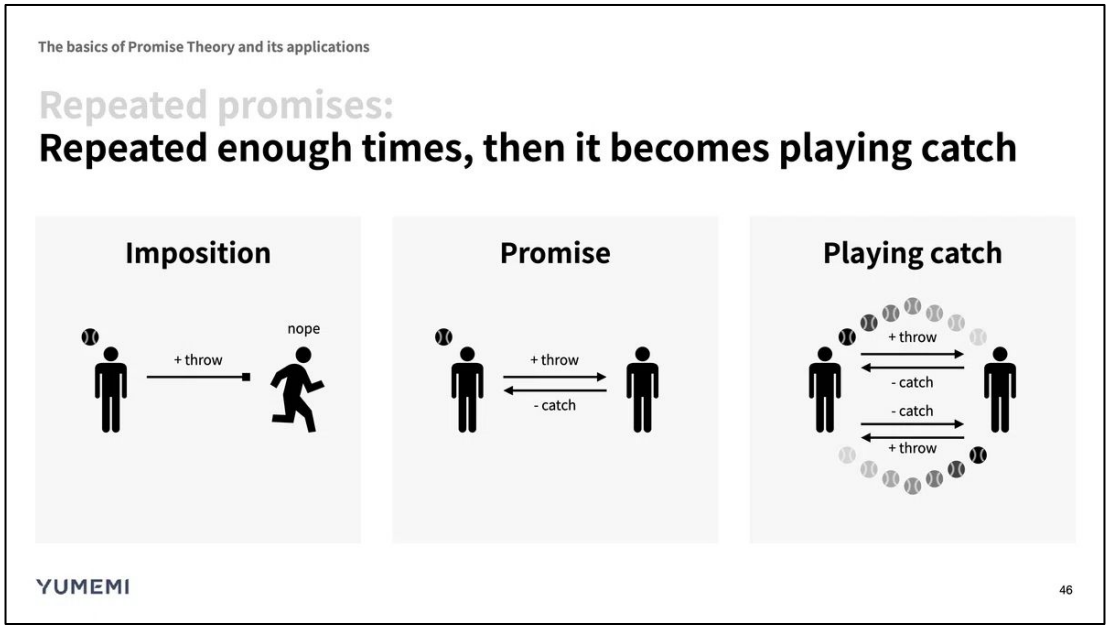
約束している相手がいるならば

安心して投げ込んでも成立する

Cooperation（協調）

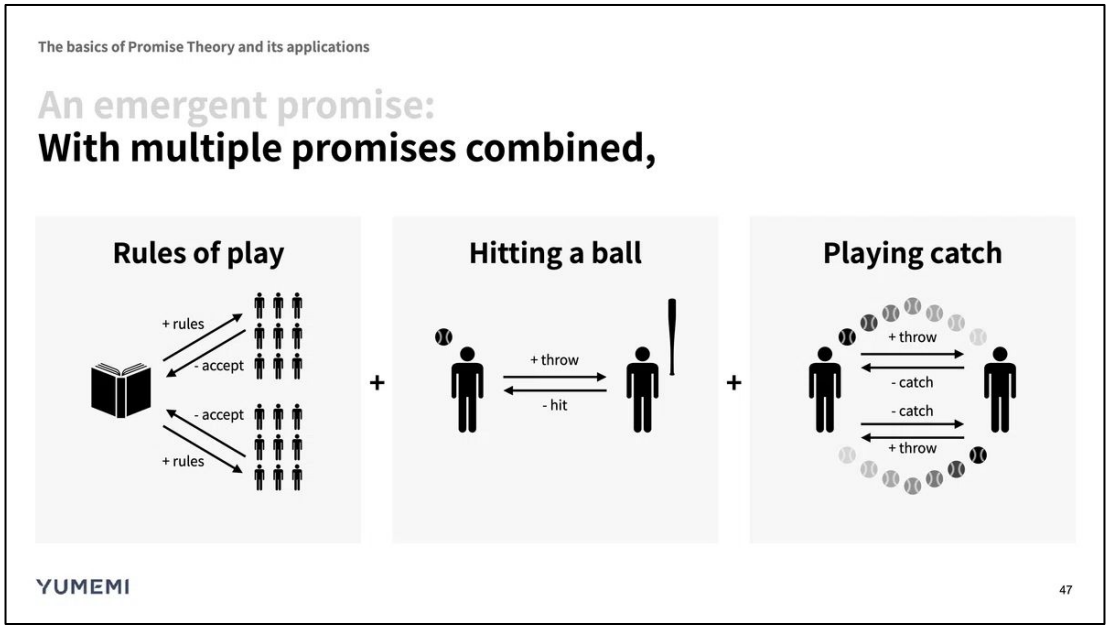
複数の約束と要求が一致することで成り立つ相互作用

一定の合意事項としての役割を果たす（キャッチボールするなど）



約束の連鎖による複雑な協調

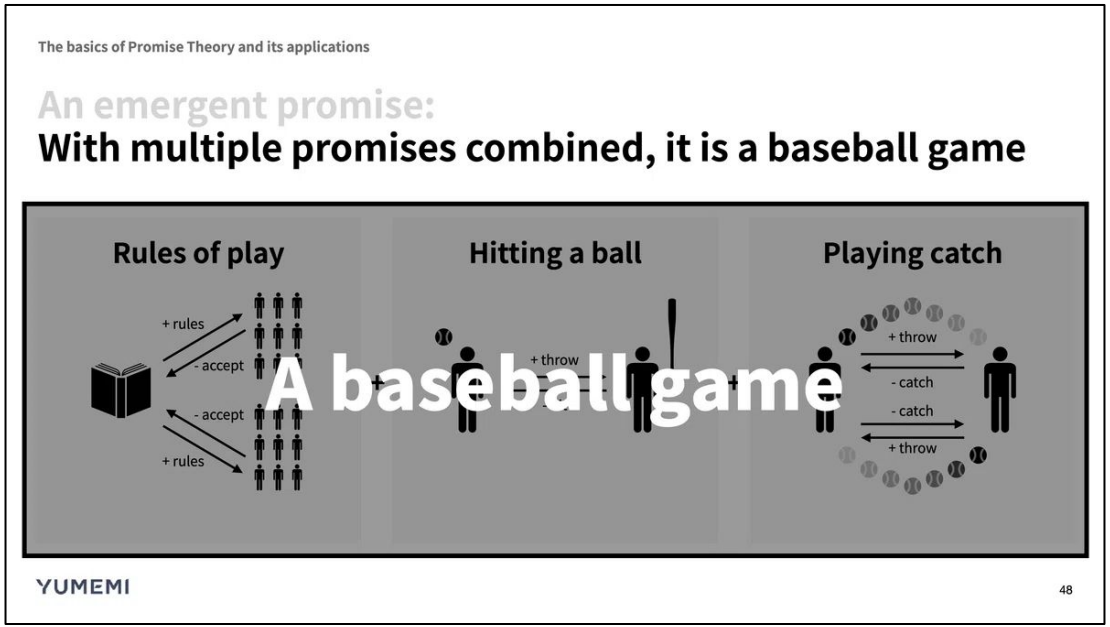
要求と約束のペアが積み重なると、複雑だが一貫性のあるルールが実現する



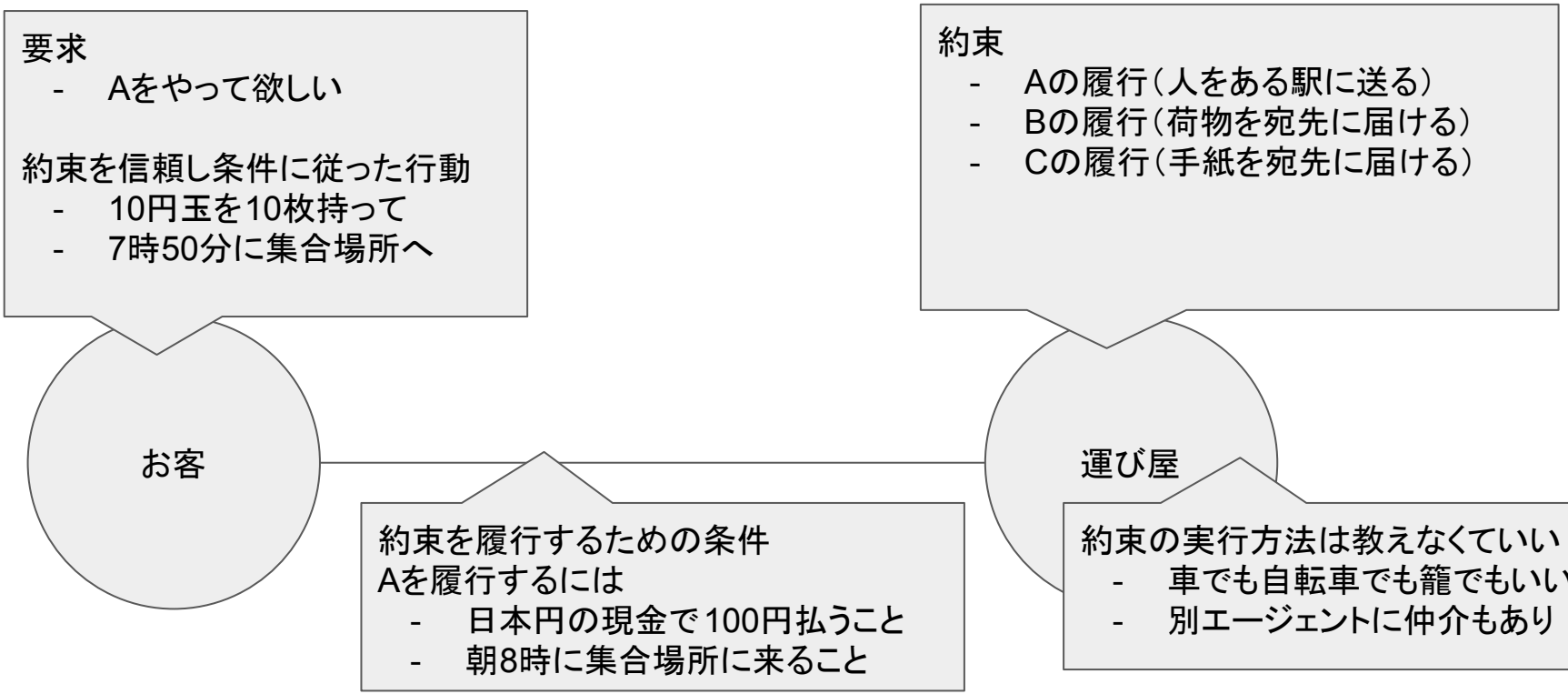
目標の達成 ＝ 約束の連鎖

要求と約束のペアが積み重なると、複雑だが一貫性のあるルールが実現する

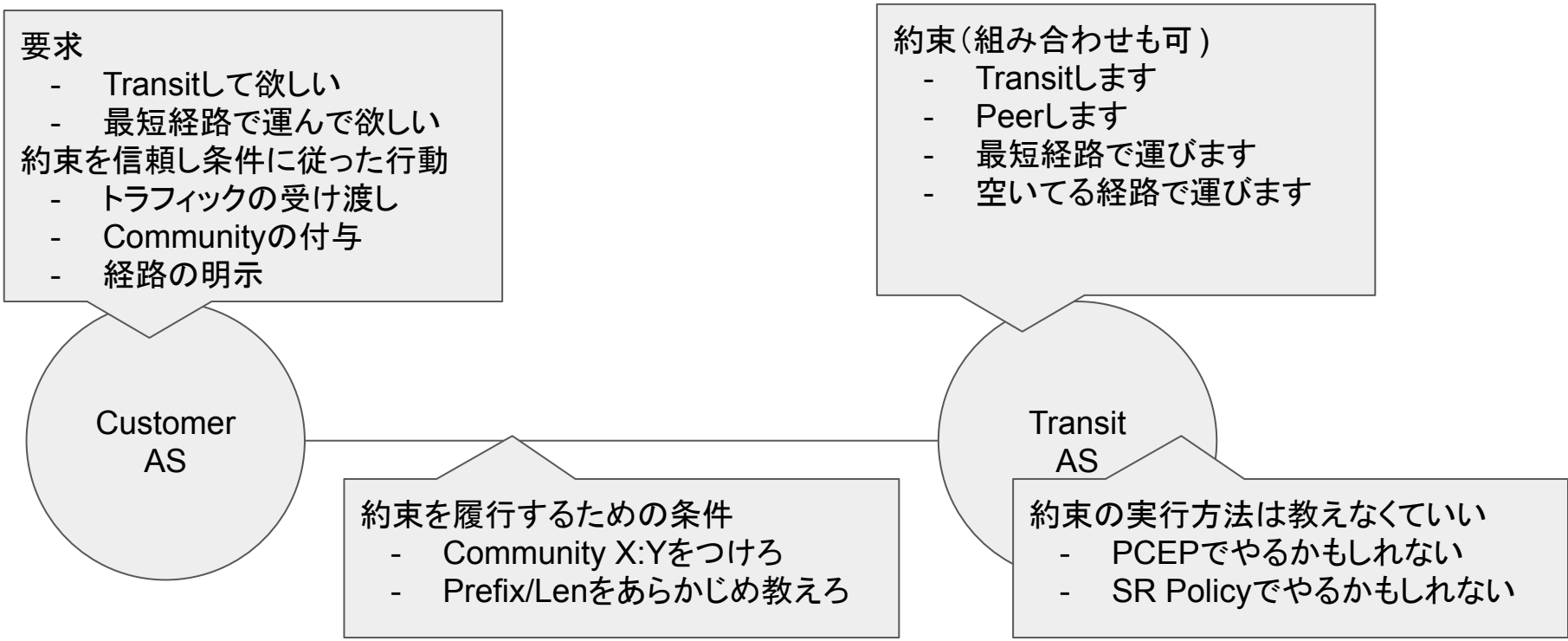
キャッチボール、ピッチャーとバッター、その他ルールが連鎖して野球が成立



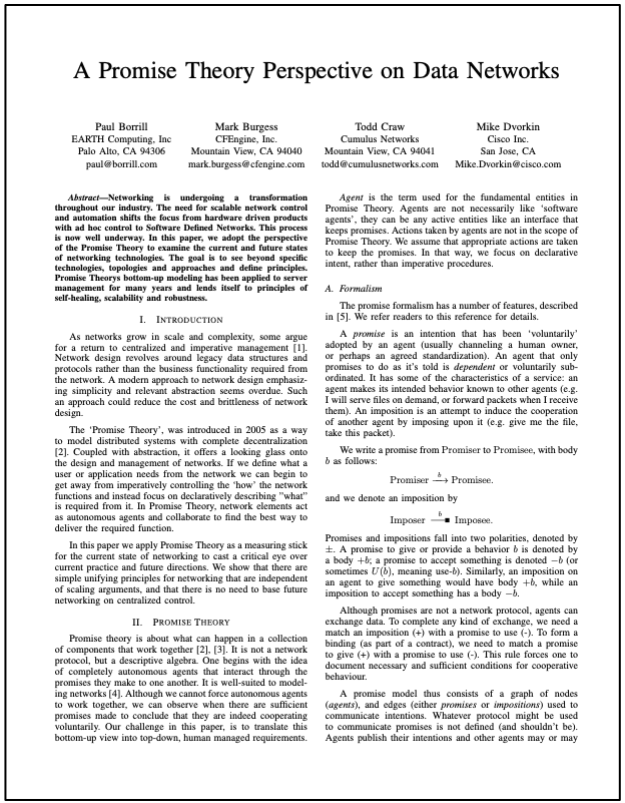
Promise Theoryで大事なこと



Promise TheoryっぽくeBGPを考えてみる



Promise Theoryとネットワーキング



A Promise Theory Perspective on Data Networks

Paul Borrill
EARTH Computing, Inc
Palo Alto, CA 94306
paul@borrill.com

Mark Burgess
CFEngine, Inc.
Mountain View, CA 94040
mark.burgess@cfengine.com

Todd Craw
Cumulus Networks
Mountain View, CA 94041
todd@cumulusnetworks.com

Mike Dvorkin
Cisco Inc.
San Jose, CA
Mike.Dvorkin@cisco.com

***Abstract*—Networking is undergoing a transformation throughout our industry. The need for scalable network control and automation shifts the focus from hardware driven products with ad hoc control to Software Defined Networks. This process is now well underway. In this paper, we adopt the perspective of the Promise Theory to examine the current and future states of networking technologies. The goal is to see beyond specific technologies, topologies and approaches and define principles. Promise Theorys bottom-up modeling has been applied to server management for many years and lends itself to principles of self-healing, scalability and robustness.**

I. Introduction

As networks grow in scale and complexity, some argue for a return to centralized and imperative management [1]. Network design revolves around legacy data structures and protocols rather than the business functionality required from the network. A modern approach to network design emphasizing simplicity and relevant abstraction seems overdue. Such an approach could reduce the cost and brittleness of network design.

The 'Promise Theory', was introduced in 2005 as a way to model distributed systems with complete decentralization [2]. Coupled with abstraction, it offers a looking glass onto the design and management of networks. If we define what a user or application needs from the network we can begin to get away from imperatively controlling the 'how' the network functions and instead focus on declaratively describing "what" is required from it. In Promise Theory, network elements act as autonomous agents and collaborate to find the best way to deliver the required function.

In this paper we apply Promise Theory as a measuring stick for the current state of networking to cast a critical eye over current practice and future directions. We show that there are simple unifying principles for networking that are independent of scaling arguments, and that there is no need to base future networking on centralized control.

II. Promise Theory

Promise theory is about what can happen in a collection of components that work together [2], [3]. It is not a network protocol, but a descriptive algebra. One begins with the idea of completely autonomous agents that interact through the promises they make to one another. It is well-suited to modeling networks [4]. Although we cannot force autonomous agents to work together, we can observe when there are sufficient promises made to conclude that they are indeed cooperating voluntarily. Our challenge in this paper, is to translate this bottom-up view into top-down, human managed requirements.

Agent is the term used for the fundamental entities in Promise Theory. Agents are not necessarily like 'software agents', they can be any active entities like an interface that keeps promises. Actions taken by agents are not in the scope of Promise Theory. We assume that appropriate actions are taken to keep the promises. In that way, we focus on declarative intent, rather than imperative procedures.

A. Formalism

The promise formalism has a number of features, described in [5]. We refer readers to this reference for details.

A *promise* is an intention that has been 'voluntarily' adopted by an agent (usually channeling a human owner, or perhaps an agreed standardization). An agent that only promises to do as it's told is *dependent* or voluntarily subordinated. It has some of the characteristics of a service: an agent makes its intended behavior known to other agents (e.g. I will serve files on demand, or forward packets when I receive them). An imposition is an attempt to induce the cooperation of another agent by imposing upon it (e.g. give me the file, take this packet).

We write a promise from Promiser to Promisee, with body b as follows:

$$\text{Promiser} \xrightarrow{b} \text{Promisee}.$$

and we denote an imposition by

$$\text{Imposer} \xrightarrow{b} \blacksquare \text{ Imposee}.$$

Promises and impositions fall into two polarities, denoted by $\pm$. A promise to give or provide a behavior b is denoted by a body $+b$, a promise to accept something is denoted $-b$ (or sometimes $U(b)$, meaning use-b). Similarly, an imposition on an agent to give something would have body $+b$, while an imposition to accept something has a body $-b$.

Although promises are not a network protocol, agents can exchange data. To complete any kind of exchange, we need a match an imposition (+) with a promise to use (-). To form a binding (as part of a contract), we need to match a promise to give (+) with a promise to use (-). This rule forces one to document necessary and sufficient conditions for cooperative behaviour.

A promise model thus consists of a graph of nodes (*agents*), and edges (either *promises* or *impositions*) used to communicate intentions. Whatever protocol might be used to communicate promises is not defined (and shouldn't be). Agents publish their intentions and other agents may or may

A Promise Theory Perspective on Data Networks
https://arxiv.org/pdf/1405.2627

ネットワークの世界にPromise Theoryの応用を検討

- エージェント = 自律ノード、ソフトウェア
- やれることを約束する
  - HTTPのトラフィックを受け取り処理する
  - IPのトラフィックを受け取り宛先に応じて転送する

ネットワークに適用するには課題の検討・解決

- トポロジとのマッピング方法
- スケーラビリティ
- 様々な表現（サービス志向、プロキシなど）

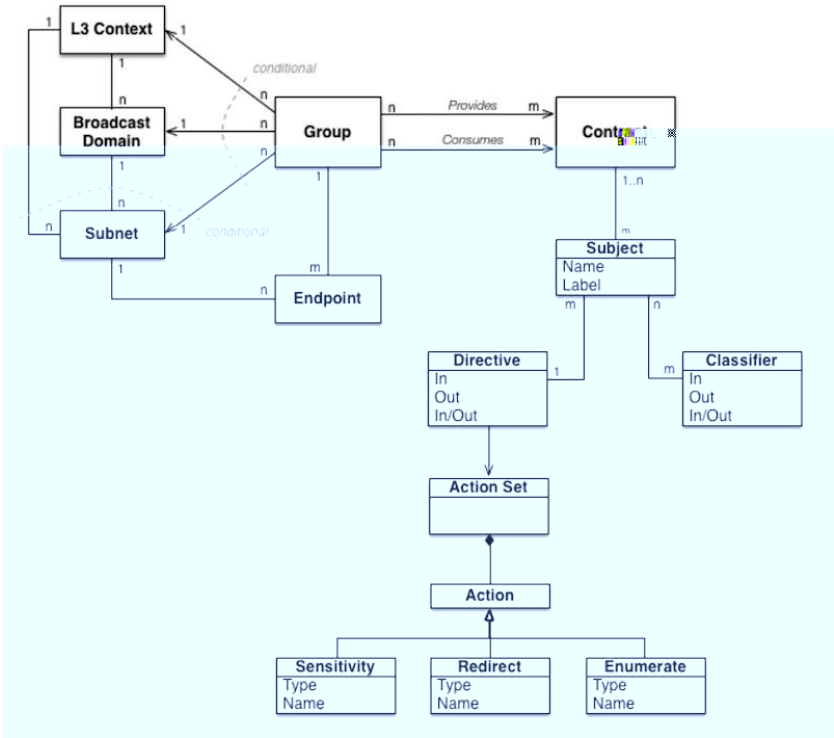
Group Based Policy（GBP）

Promise Theoryを元にネットワークの
ポリシーを記述するフレームワーク

様々な実装で活用されている

- Cisco ACI
- Juniper Apstra
- OpenStack
- etc…

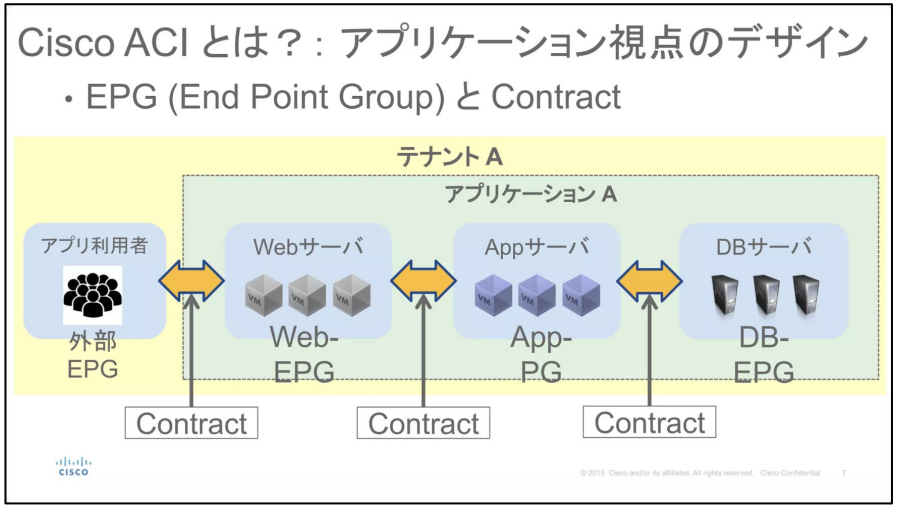
が、実際には実装技術に非依存に
ポリシーが表現できる汎用的な仕組み

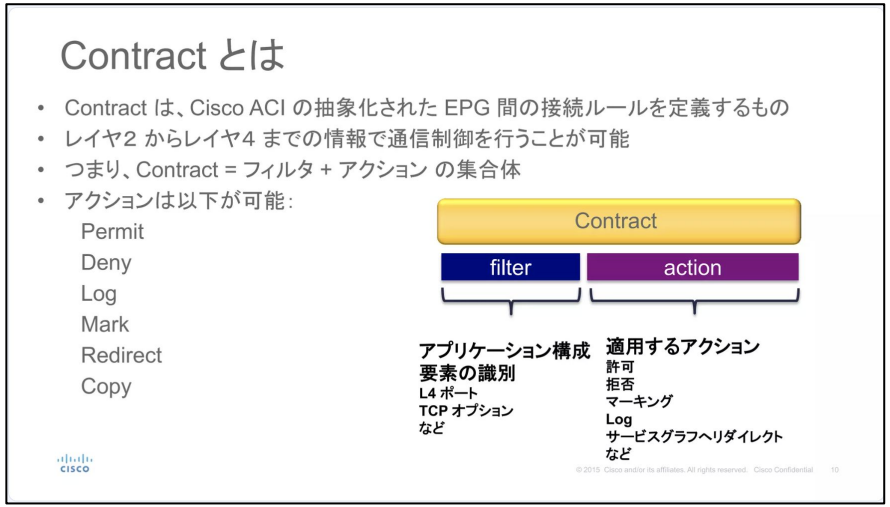


Group Based Policyの代表例（Cisco ACI）

Web / App / DBの三層構造のフィルタを上手に表現している
例えばAWSのVPCにおけるACL設定の一貫性を担保するのに利用できる

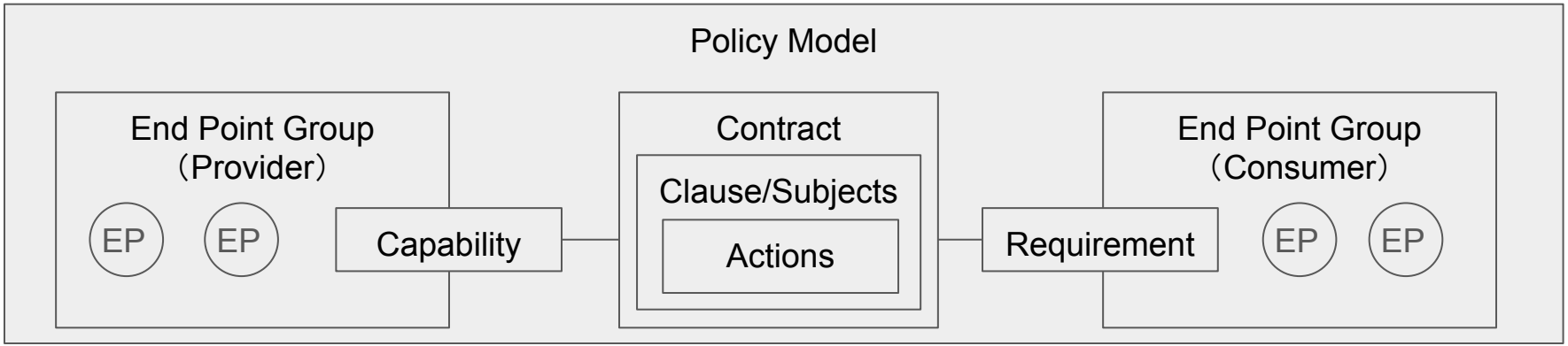
ただGBP自体は「セキュリティポリシー」を超えて汎用的にポリシー記述できる





【Cisco Data Center Forum 2015】Cisco ACI 活用例
https://www.slideshare.net/slideshow/ltdata-cisco-aci/47326117

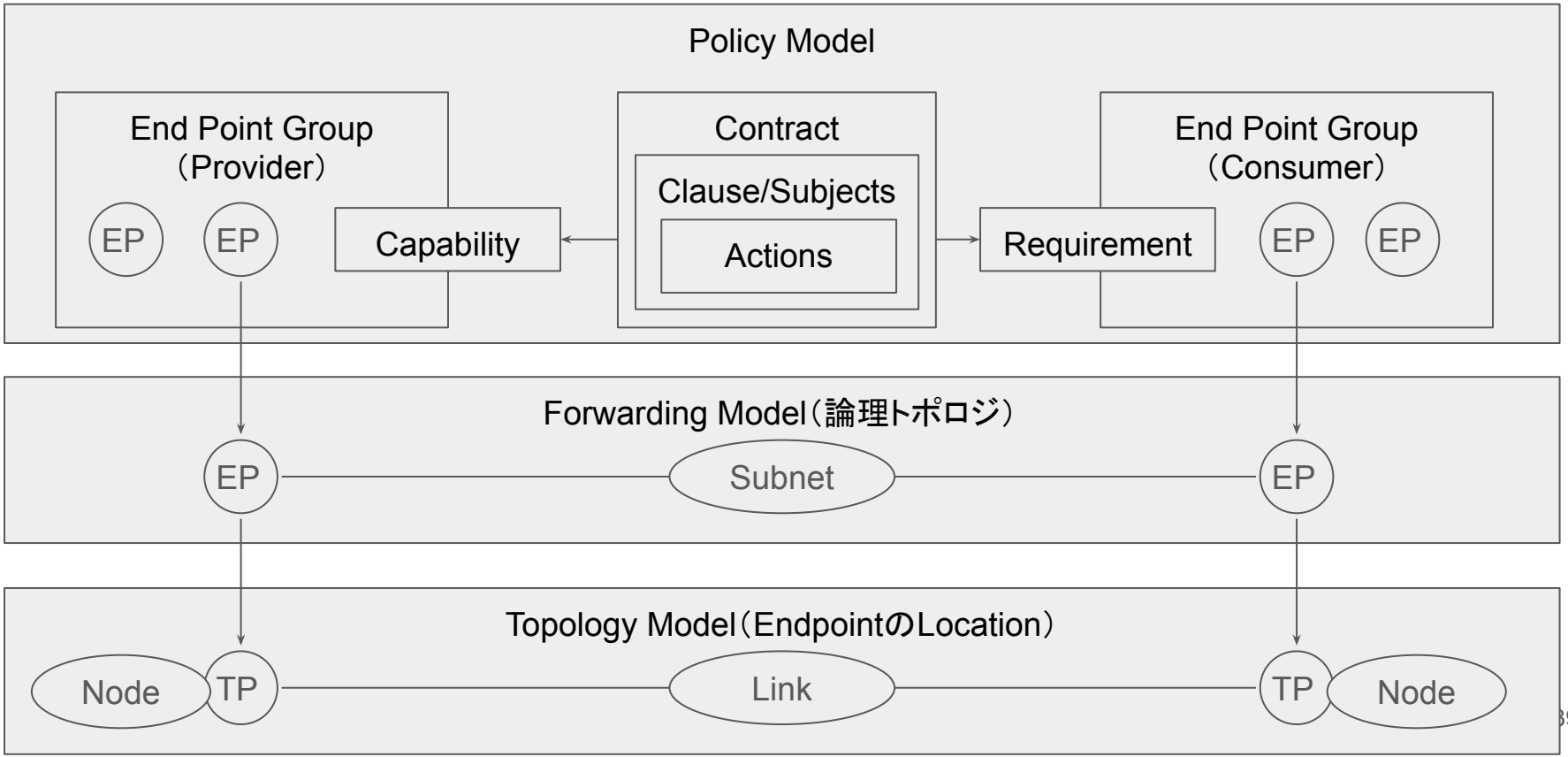
GBPにおけるInformation Model（Policy）



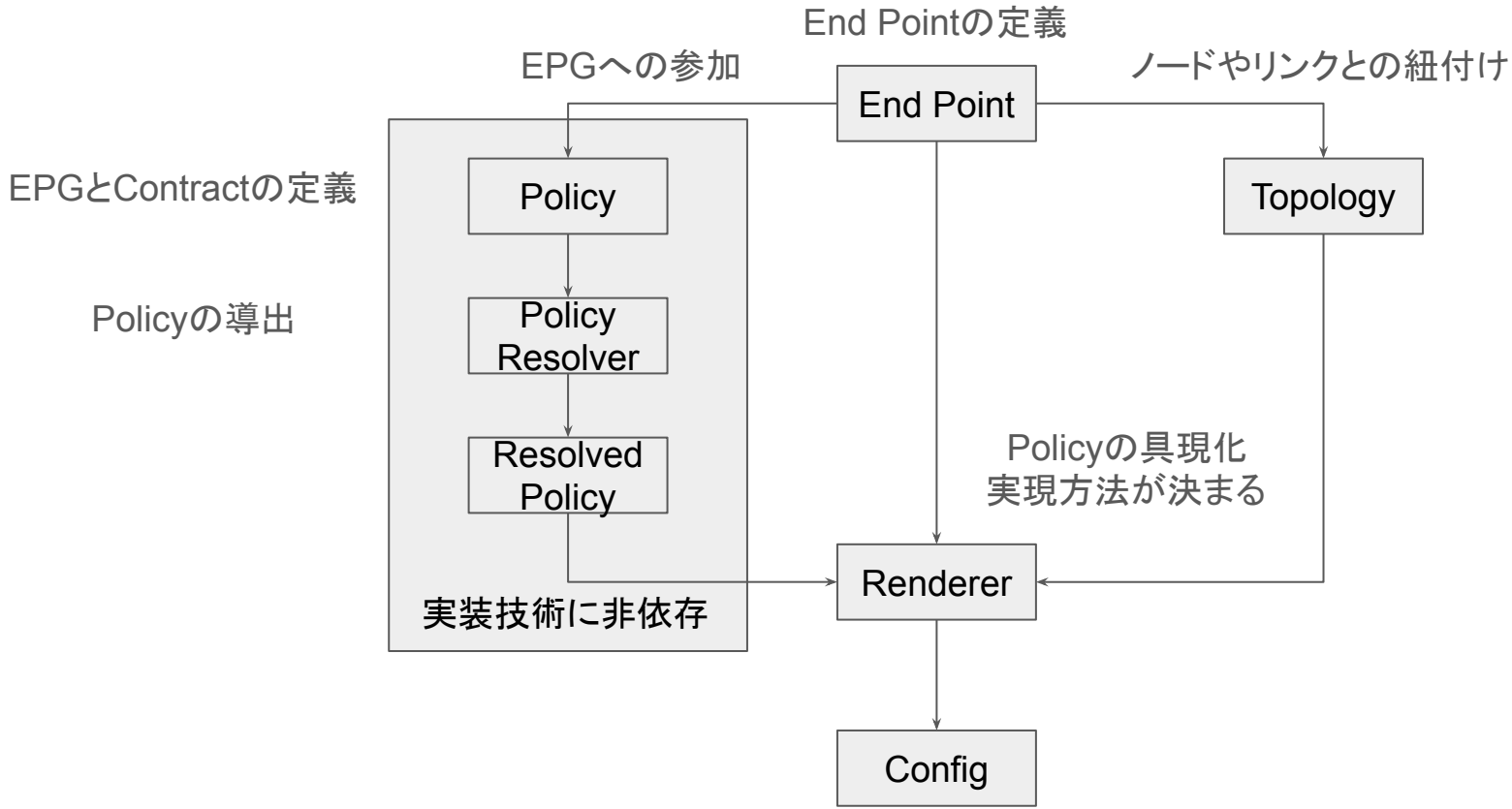
End Point（EP）	対象ネットワークにおけるエンドポイントの単位
End Point Group（EPG）	Agentだと思って良い、EPをグルーピングする
Capability / Requirement	Agentの約束と要求、EPGに設定する
Clause / Subject / Action	約束と要求の条件と履行する具体的な行動の束（契約）

これらを設定すると、一貫性のあるPolicyが組み上がる（Policy Resolution）
だがしかし・・・これだけだとEPが属するノード（設定先ルータ）等が未解決

GBPにおけるInformation Model（Forwarding&Topology）



Group Based PolicyにおけるConfigの生成（Rendering）



ユースケースに応じた拡張性

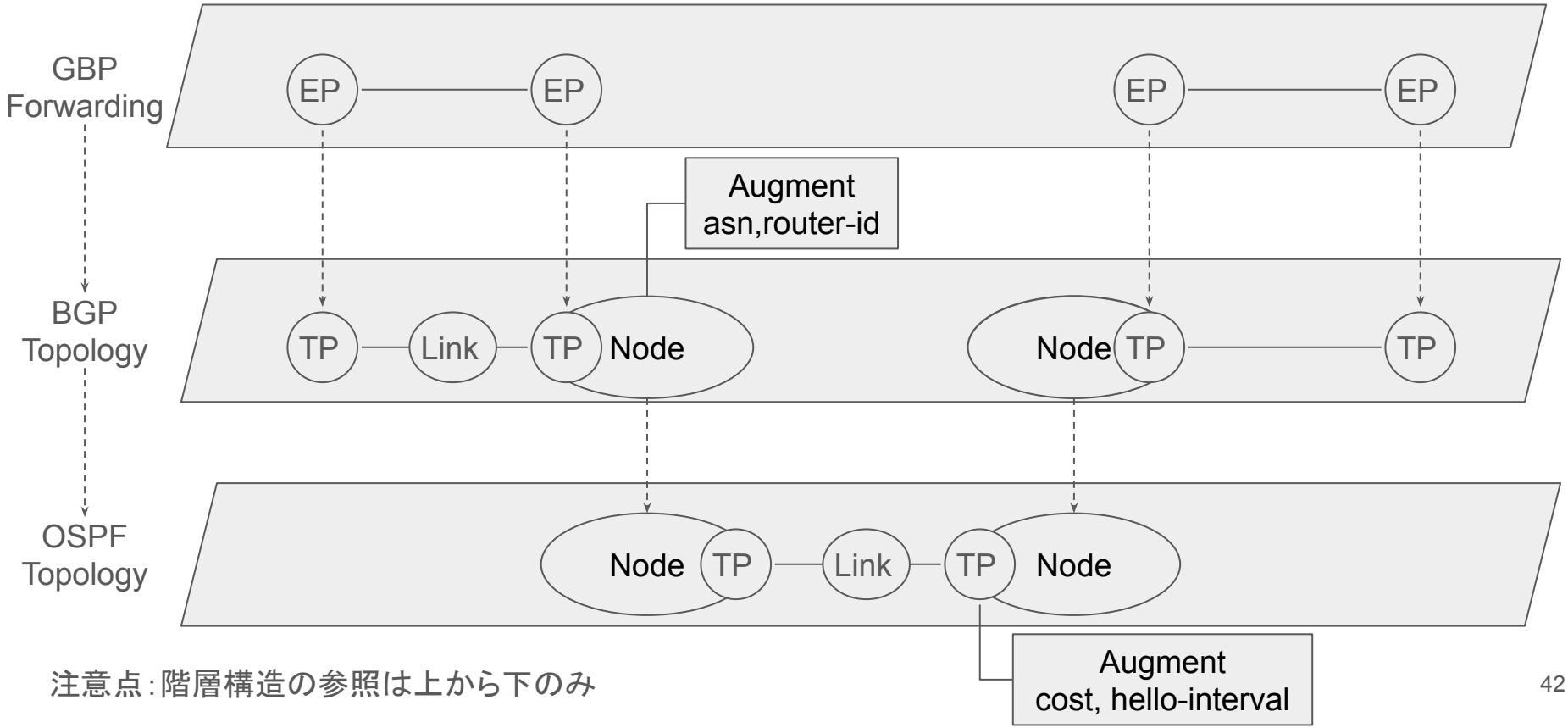
YANGのAugmentが全てを解決する

Renderer: Resolved Policyの解釈方法を自由に設計すれば良い
一貫性あるPolicyはConfigして初めて意味を持つ
When I use a word, it means what I choose it to mean – neither more nor less.

Endpoint: L2とL3の情報しか持っていない
ただし、Endpoint-Locationを書ける（instance-identifier）

Topology: Endpoint-Locationの参照先
ietf-network-topology（Node, Link, Termination-Point）の定義を基に
YANGをAugmentすることで拡張可能
例) ospf-topologyならTPにcostやhello-intervalを加える

Topologyの階層構造の表現とAugment



注意点：階層構造の参照は上から下のみ

Promise Theory/GBPの参考文献

A promise theory perspective on data networks
https://arxiv.org/pdf/1405.2627

Intent-based networking for the enterprise: a modern network architecture
https://dl.acm.org/doi/pdf/10.1145/3538513

Promise Theory—What Is It?
https://dl.acm.org/doi/fullHtml/10.5555/2666018.2666021

Promise Theory and Applications
https://www.youtube.com/@PromiseTheoryandApplications

Virtual Network Configuration with Group-based Policy using OpenDaylight
https://www.slideshare.net/slideshow/virtual-network-configuration-with-groupbased-policy-using-opendaylight/56162781

Group Based Policy: Open Source Policy in OpenDaylight and OpenStack Neutron
https://www.slideshare.net/slideshow/group-based-policy-open-source-policy-in-opendaylight-and-openstack-neutron/38330932

Cisco ACI and OpenStack - Group-based Policy (GBP) & OpFlex
https://www.slideshare.net/slideshow/cisco-aci-and-openstack-groupbased-policy-gbp-opflex/51093210

まとめ

テレコムキャリア網には「広義の」SDNが適当

広義のSDNにおいては「Policyの一貫性」が肝である

経路制御プロトコル下でPolicyの一貫性を保つには「Promise Theory」が有効

Promise Theoryを実現する「Group Based Policy」はSDNの重要なピース