Networkless:
Current Status and Future Approaches

2018-08-28

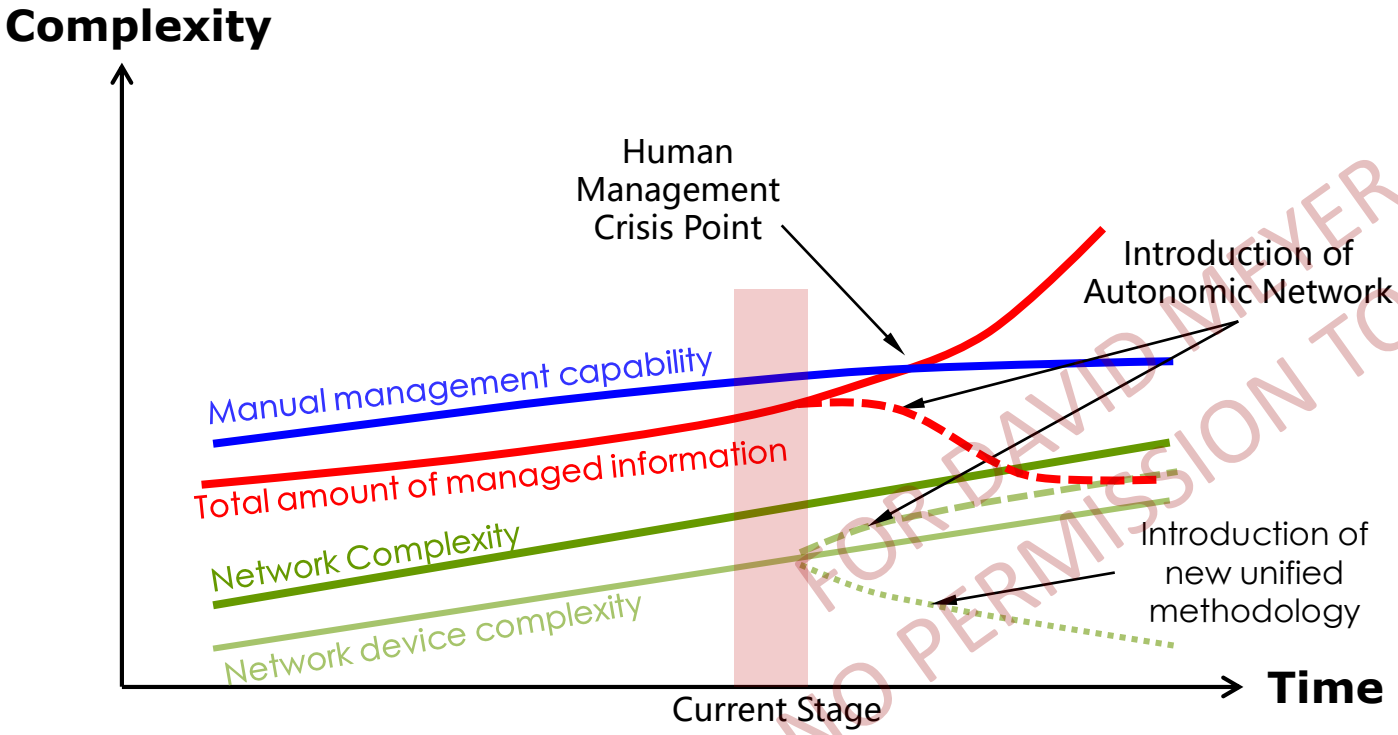
Contents

- **Background: Human cost issue**
- **IETF Standardization Work (Anima)**
- **Ongoing Research Projects**
- **A Roadmap to the Destination (Networkless)**

Contents

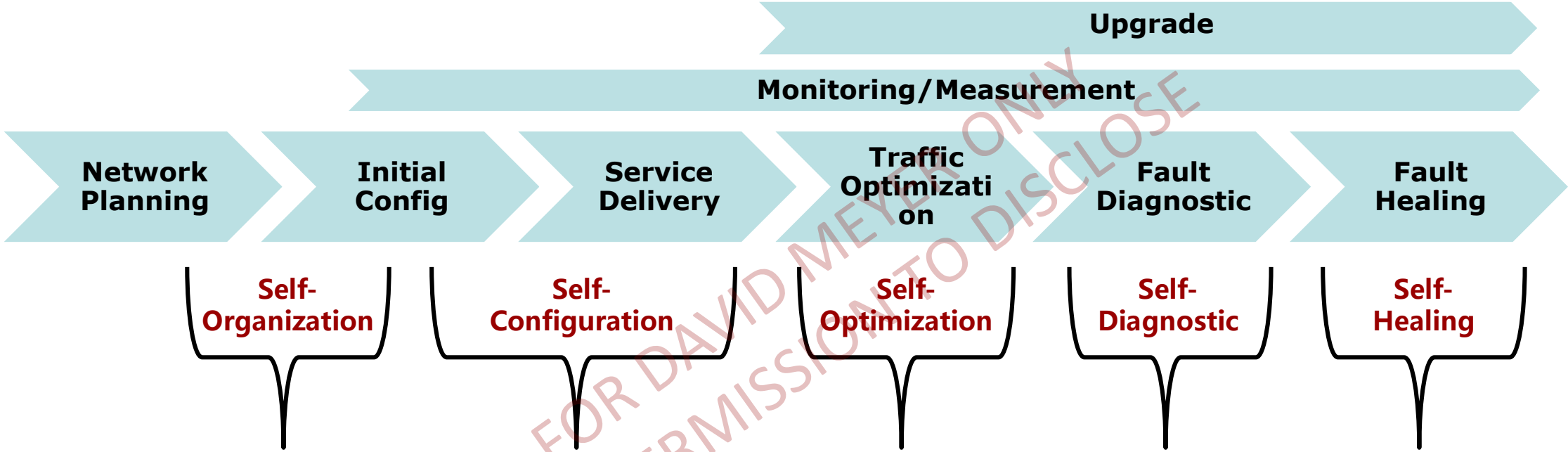
- **Background: Human cost issue**
- **IETF Standardization Work (Anima)**
- **Ongoing Research Projects**
- **A Roadmap to the Destination (Networkless)**

Background： Human Cost Issue is Becoming more and more Serious



- Human-based management cannot handle the more and more complex network.
- Introducing autonomous OAM into network could simplify the human management, reduce the human error and the cost of network maintenance.
- Autonomous OAM also requires network devices/system become more intelligent and complex.

Main Aspects of Network OAM



Contents

- **Background: Human cost issue**
- **IETF Standardization Work (Anima)**
- **Ongoing Research Projects**
- **A Roadmap to the Destination (Networkless)**

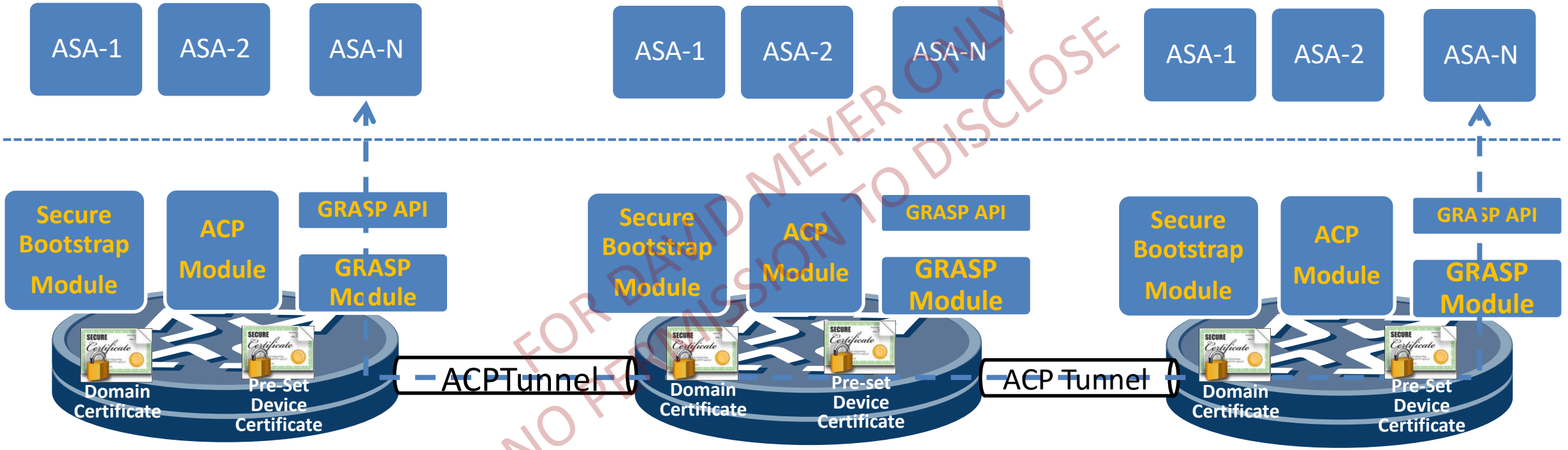
Ongoing IETF work

- **IETF Anima Working Group**
  - **A**utonomic **N**etworking **I**ntegrated **M**odel **A**pproach
    - Formed in late 2014, led by Huawei and Cisco
  - "Integrated Model Approach" indicates that Anima is not a "Clean Slate"; rather, it could be integrated into current networks (e.g. co-exist with NMS/SDN).
  - According to current charter, Anima aims at developing some "re-useable components", which means some common technologies that could be used among different scenarios.

Two Anima Groupsets: ANI & ASA

ASA（Autonomic Service Agent）：

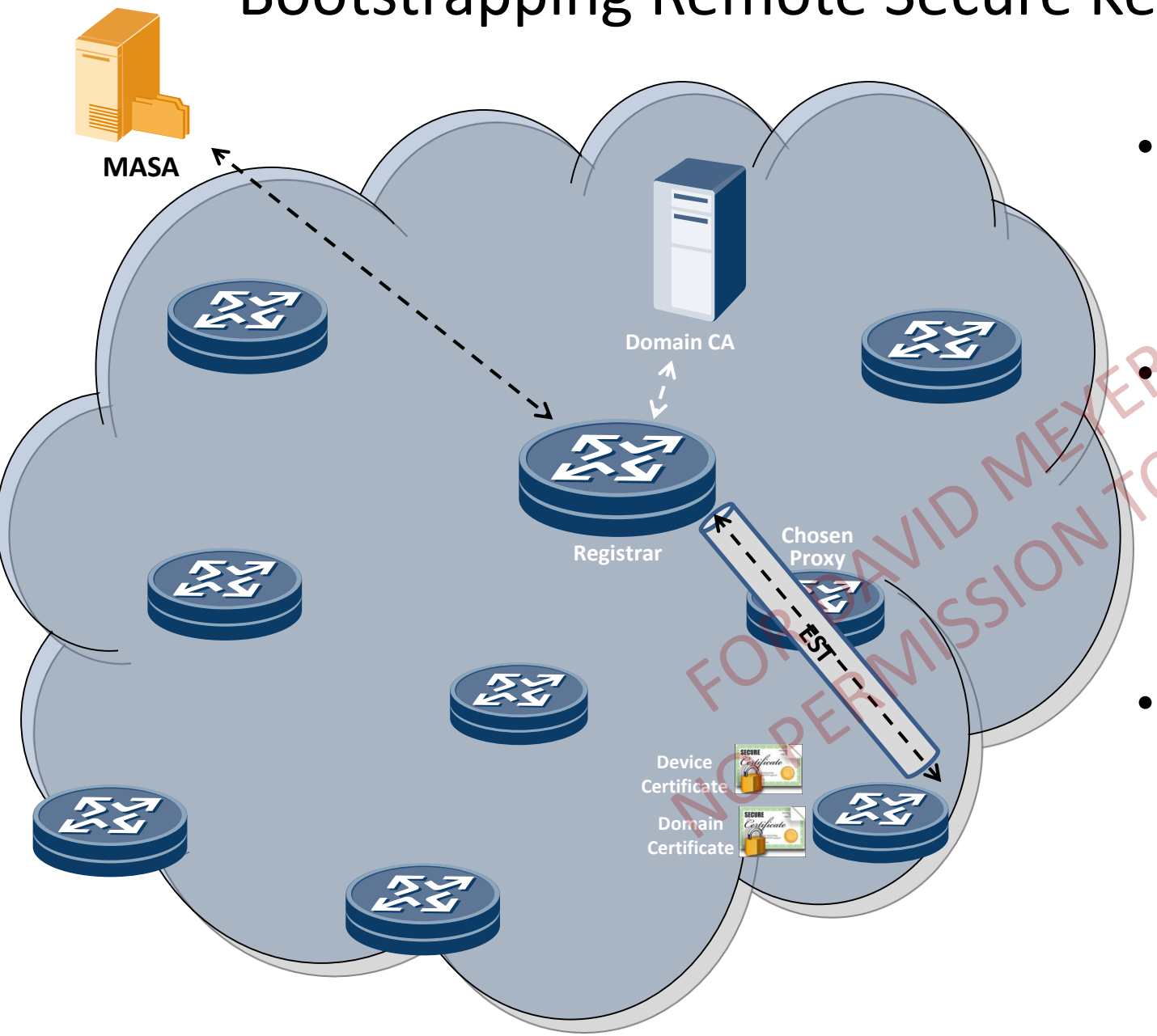
- Could be considered as “Apps” within network devices.
- ASAs interact with each other to fufill specific managemnt tasks such as parameter configuration, service delivery etc



ANI (Autonomic Network Infrastructure), including 3 “re-useable components”：

- Secure Bootstrap： access authentication of new devices; assigning Autonomic Domain certificates to new devices
- ACP： hop-by-hop encrypt IP tunnels between nodes in an Autonomic Domain, to form a stable VPN dedicated for management channel.
- GRASP： 1) signaling protocol for interaction between ASAs to fulfill specific management tasks; running in ACP to gain security protection.
  2) signaling protocol during Bootstrap discovery and ACP formation.

Bootstrapping Remote Secure Key Infrastructures (BRSKI)



- Enrollment to the Registrar
  - The new device enrollments itself to the Registrar mainly by authentication of the Device Certificate.
  - They use EST protocol for certificate authentication.(EST: Enrollment over Secure Transport, RFC7030)
- MASA Service (Optional)
  - A Manufacturer Authorized Signing Authority (MASA) service on the global Internet. The MASA provides a repository for audit log information concerning privacy protected bootstrapping events.
  - In short, MASA is for more reliable authentication for the new device, to prove maliciously re-use of the device certificate.
- Issuing the Domain Certificate
  - After authenticating the new device, the registrar requests a Domain Certificate for the new device from the Domain CA.
  - Then, the new device generates a ULA prefix based on the Domain Name attribute in the Domain Certificate, and assigns itself a ULA address to communicate with other entities in the domain.
  - It uses the keys in the Domain Certificate for any further encrypt communication.

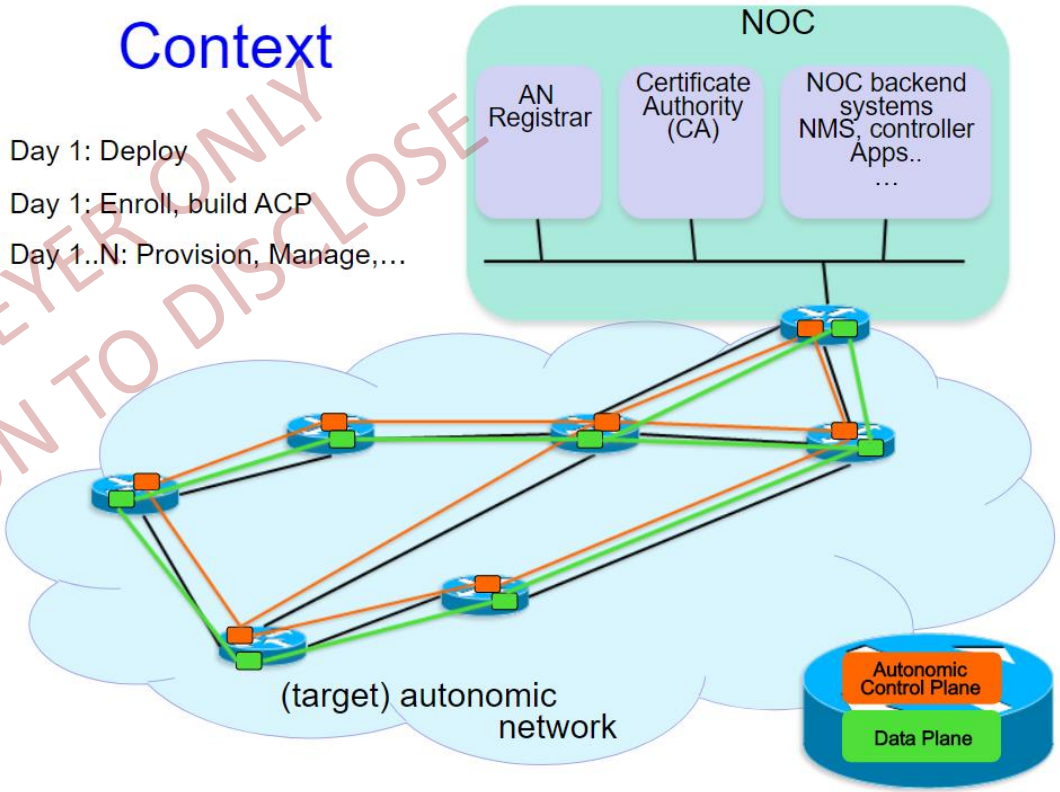
ACP： Autonomic Control Plane

Main usage of ACP

- **NE Plug-N-Play**： NE automatically enable routing without any configuration
- **Reliable Management Channel**: NMS/Controller can always connect to NEs through ACP, even if the normal data plane was broken

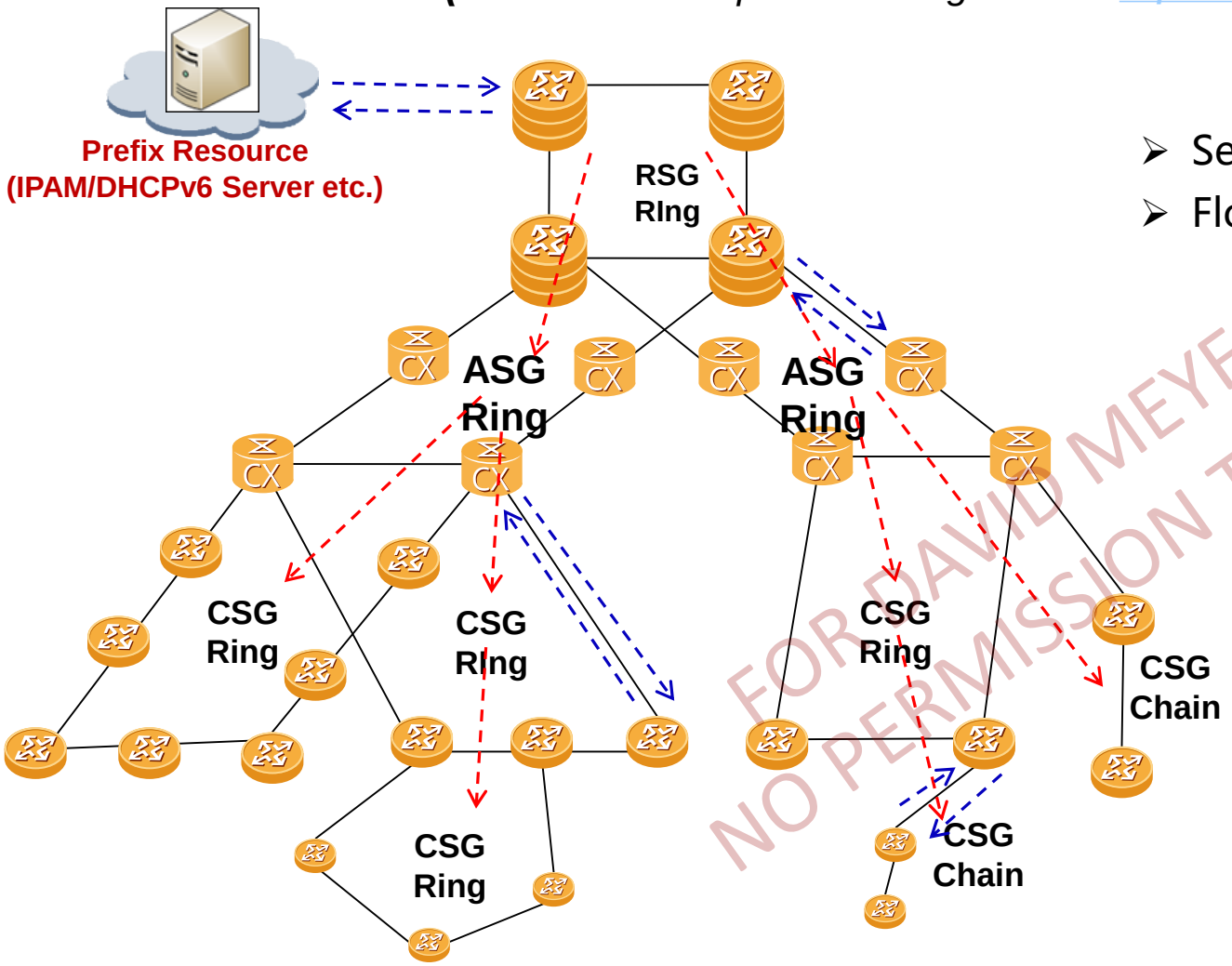
(RFC8368 describes the technical details of how NMS connects to ACP： https://datatracker.ietf.org/doc/rfc8368/)

- **Control Channel between NEs**: allowing NEs to directly deal with some management problems through horizontal interactions



ASA Example-1: Prefix Management in large-scale network

(draft-ietf-anima-prefix-management，https://tools.ietf.org/html/draft-ietf-anima-prefix-management-07)



- Set up the prefix allocation policy according to NE role
- Flood the policy to all NEs

GRASP Flooding Messages

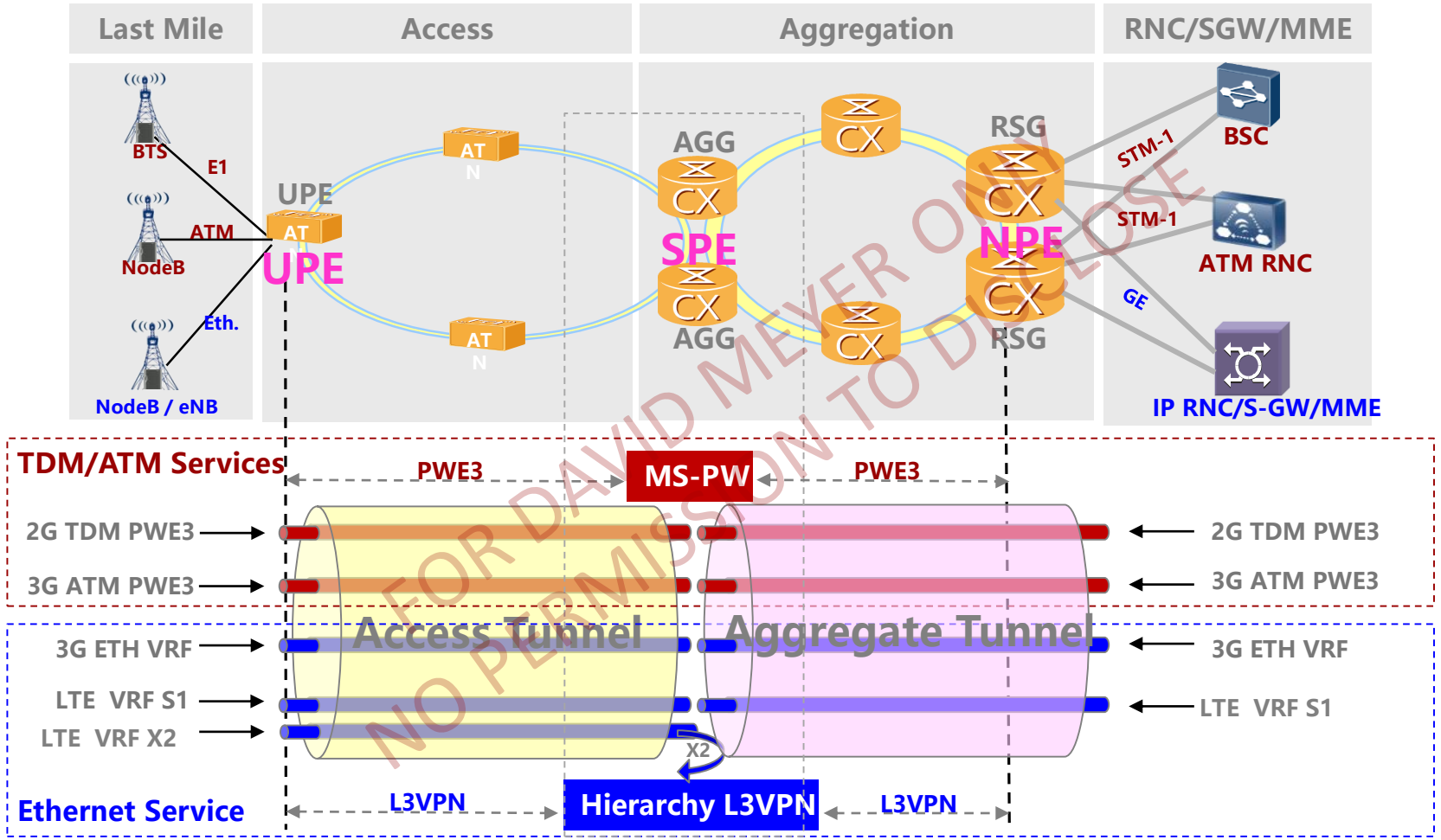
["role", "RSG"],["prefix_length", 34],
["role", "ASG"],["prefix_length", 44],
["role", "CSG"],["prefix_length", 56],

- NEs discover prefix resource and request prefixes by themselves

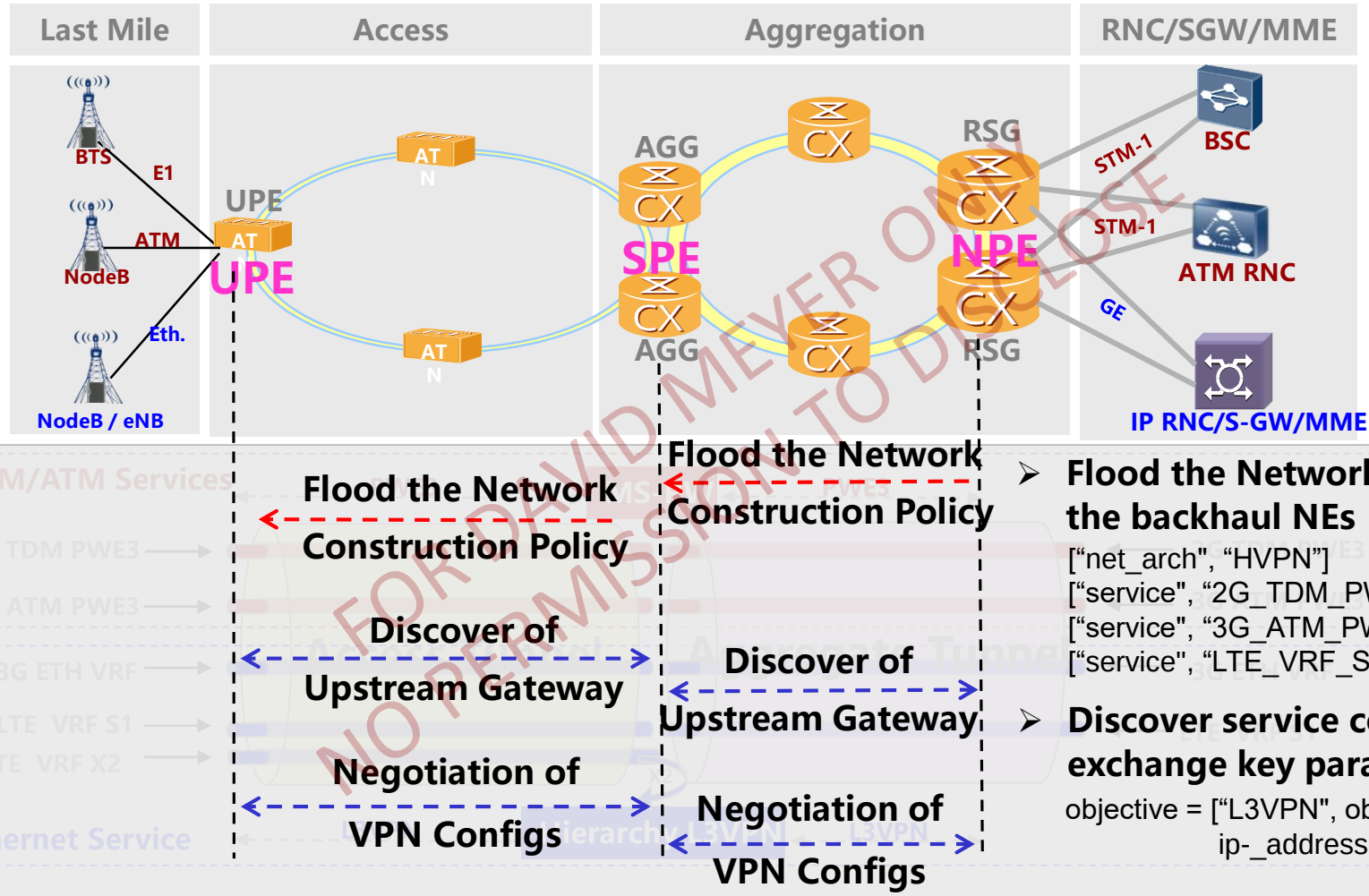
GRASP Discovery/Negotiation Messages

objective = ["PrefixManager", objective-flags, loop-count, PD-support, length, ?prefix]

ASA Example-2: VPN Service Configurations in Mobile Backhaul Networks

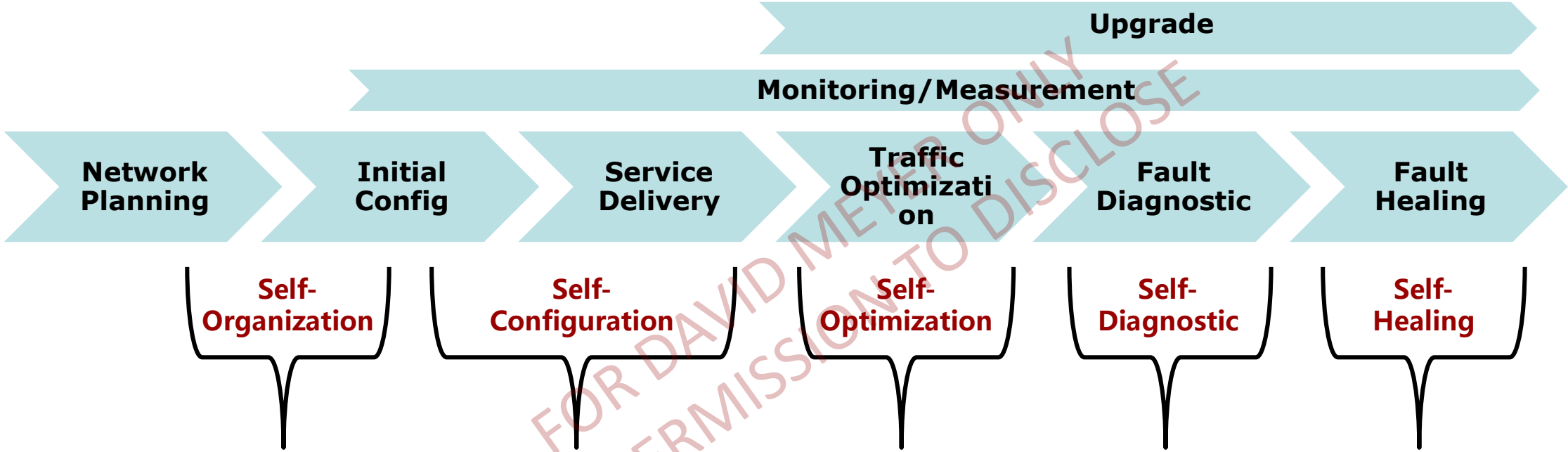


ASA Example-2: VPN Service Configurations in Mobile Backhaul Networks



- **Flood the Network Construction Policy to all the backhaul NEs**
  ["net_arch", "HVPN"]
  ["service", "2G_TDM_PWE3"]
  ["service", "3G_ATM_PWE3"]
  ["service", "LTE_VRF_S1/X2"]
- **Discover service config counterpart and exchange key parameters**
  objective = ["L3VPN", objective-flags, loop-count, role, ip-_address, VLAN_ID, VRF_ID]

Main Aspects of Network OAM



- **Separate domains**
  - *Routing domains*
  - *Administrative domains*
- **Identify key NEs**
  - *Gateways/PE etc.*
- **NE bootstrap**
  - *Join in a domain **Anima BRSKI***
  - *Initial connectivity **Anima ACP***

- **Underlay config**
  - *Addressing/Naming **Anima Prefix Management ASA***
  - *Routing etc.*
- **Service config *Anima Service Config ASA***
  - *L2VPN/L3VPN*
  - *Tunnel, PW etc.*

- **Load balancing**
  - *balance the traffic in different links in the network*
- **Congestion control**
- **Flow path selection**

- **Fault locating**
  - *Mis-configurations*
  - *Hardware failure*
  - *Software failure*
- **Fault Prediction**
  - *Hardware failure*

- **Failure isolation**
- **Failure healing**

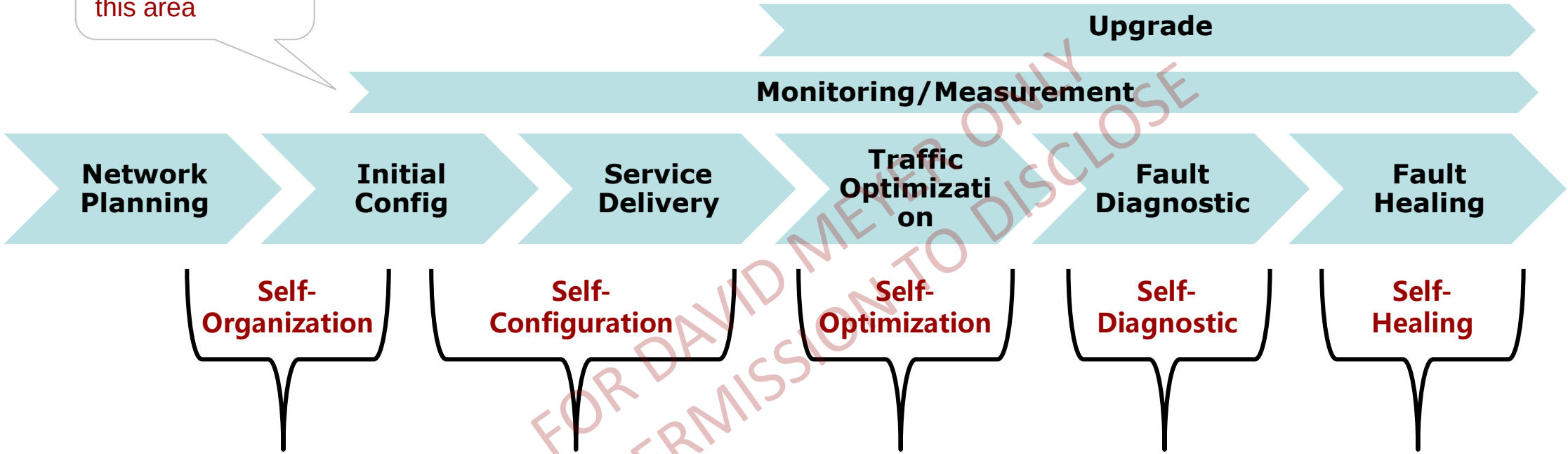
In theory, Anima ASA can almost cover every aspect of “Self-”

Contents

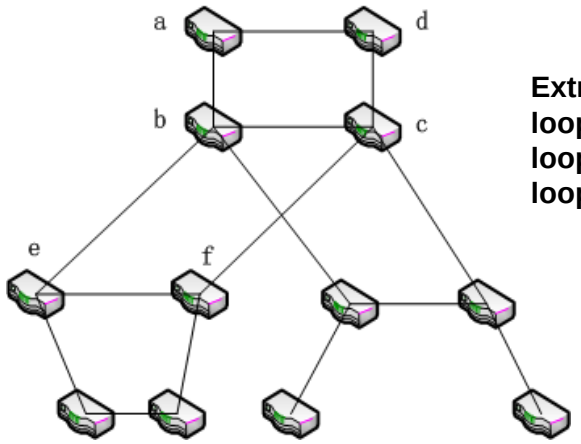
- **Background: Human cost issue**
- **IETF Standardization Work (Anima)**
- **Ongoing Research Projects**
- **A Roadmap to the Destination (Networkless)**

Main Aspects of Network OAM

Also covered some work in this area



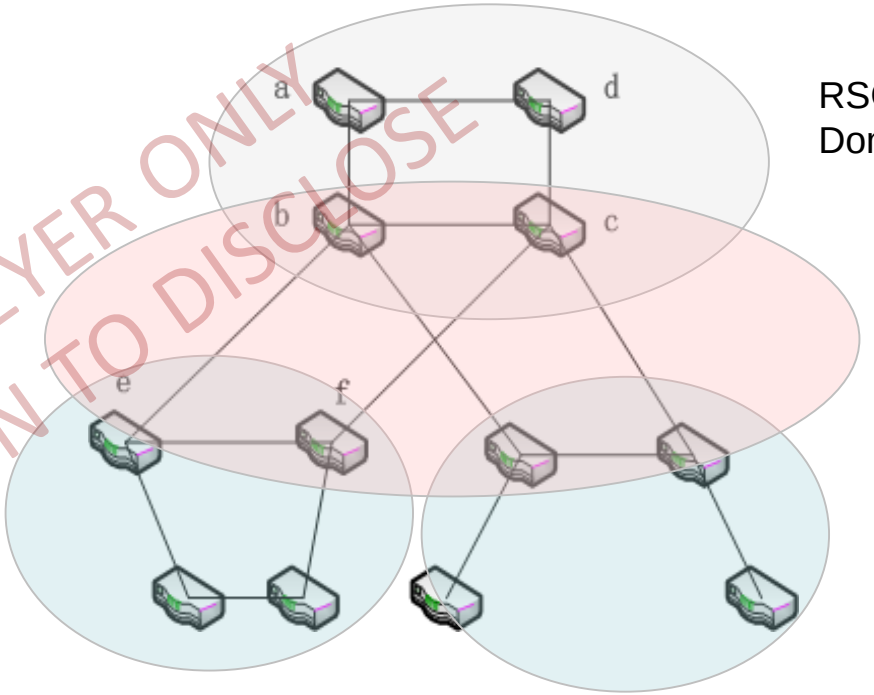
Self-Organization: domain recognition



Extract meta-structure:
loop1：a→b→c→d→a
loop2：b→e→c→f→b
loop3：a→b→e→f→c→d→a

RSGnum	AGGnum	ASGnum	CSGnum	haveRSG	haveAGG	haveASG	haveCSG
1	2	0	0	yes	yes	no	no
1	2	0	0	yes	yes	no	no
0	2	3	0	no	yes	yes	no
0	1	1	1	no	yes	yes	yes
0	1	2	2	no	yes	yes	yes
0	1	5	2	no	yes	yes	yes
0	1	5	2	no	yes	yes	yes

Using Cluster(XMEANS) to sort out different types of meta-structure



RSG Domain

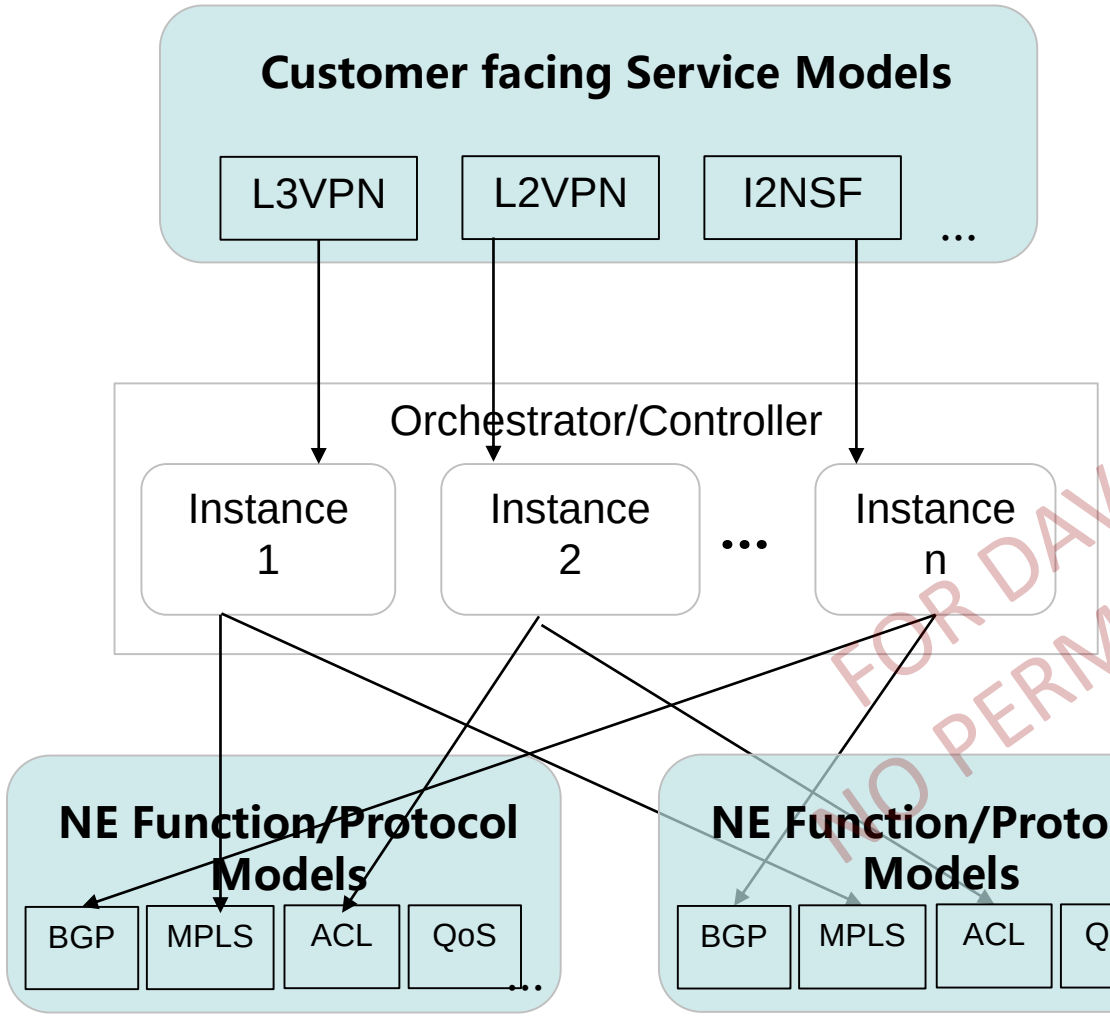
ASG Domain

CSG Domain

RSG/ASG/CSG domains are highly relevant to routing domains and service configuration policies

Self-configuration: Autonomous Service Delivery

Current Approaches



Issue 1: too much details in current SM, which implies:

- Cost a lot of human labor
- The more details, the harder to achieve a unified model

Issue 2: controller is hard to scale

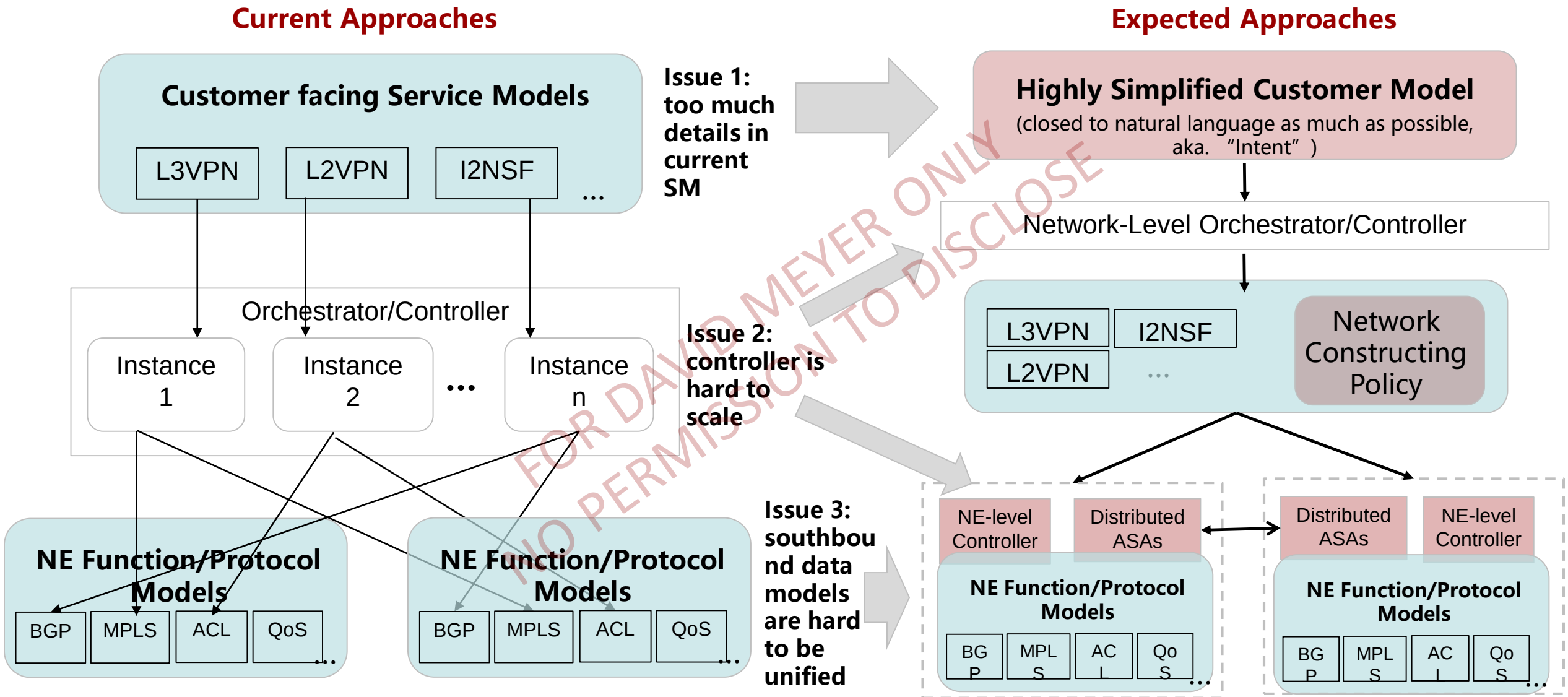
- Binding to specific service and underlay models; need to develop new instance when service/underlay varies
- Need to compile each single model in each NE

```
module: ietf-l3vpn-svc
   +--rw l3vpn-svc
      +--rw vpn-profiles
      |  +--rw valid-provider-identifiers
      |     +--rw cloud-identifier* [id] {cloud-access}?
      |     |  +--rw id    string
      |     +--rw encryption-profile-identifier* [id]
      |     |  +--rw id    string
...
+--rw routing-protocols
                |  +--rw routing-protocol* [type]
                |     +--rw type      identityref
                |     +--rw ospf {rtg-ospf}?
                |     |  +--rw address-family*   address-family
                |     |  +--rw area-address      yang:dotted-quad
                |     |  +--rw metric?           uint16
                |     |  +--rw sham-links {rtg-ospf-sham-link}?
                |     |     +--rw sham-link* [target-site]
                |     |        +--rw target-site    svc-id
                |     |        +--rw metric?        uint16
                |     +--rw bgp {rtg-bgp}?
                |     |  +--rw autonomous-system    uint32
                |     |  +--rw address-family*      address-family
...(hundreds of YANG Data Model Elements)
```

Issue 3: southbound data models are hard to be unified

- Vendor varies to each other
- Operator varies to each other
- A long-term puzzle from the SNMP era

Self-configuration: Autonomous Service Delivery



Self-configuration: Autonomous Service Delivery

Expected Approaches

Highly Simplified Customer Model
(closed to natural language as much as possible, aka. "Intent")

Network-Level Orchestrator/Controller

L3VPN | I2NSF | L2VPN

Network Constructing Policy

NE-level Controller | Distributed ASAs

NE Function/Protocol Models

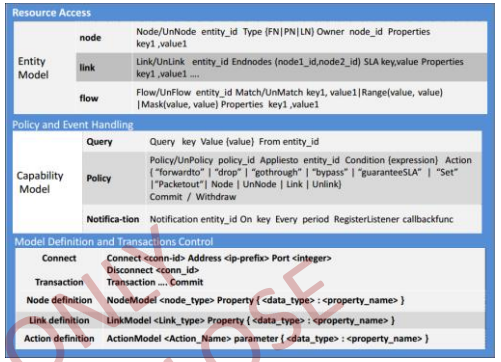
BGP | MPLS | ACL | QoS | ...

Distributed ASAs | NE-level Controller

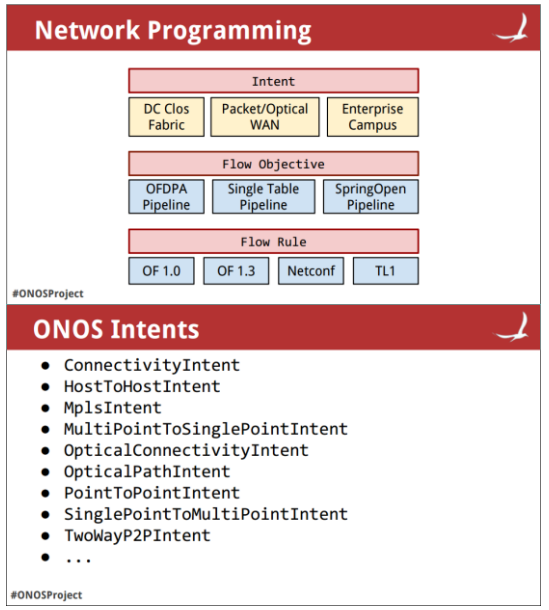
NE Function/Protocol Models

BGP | MPLS | ACL | QoS | ...

- AI-based Intent interpreting; infer service model and constructing policy.
- Potential tool: Knowdledge Graph; Machine Inference



Examples of Intent: NEMO & ONOS



Network domain division policy
- Based on specific topology (e.g. a Ring, a sub-tree)
- Based on a specific number of neighbors

Network Architect policy
- Network layer (e.g. access/aggregation/core; spine/leaf)
- Protocol selection

NE roles
- Role-1 (e.g. CSG), Role-2 (ASG), Role-3 (RSG)

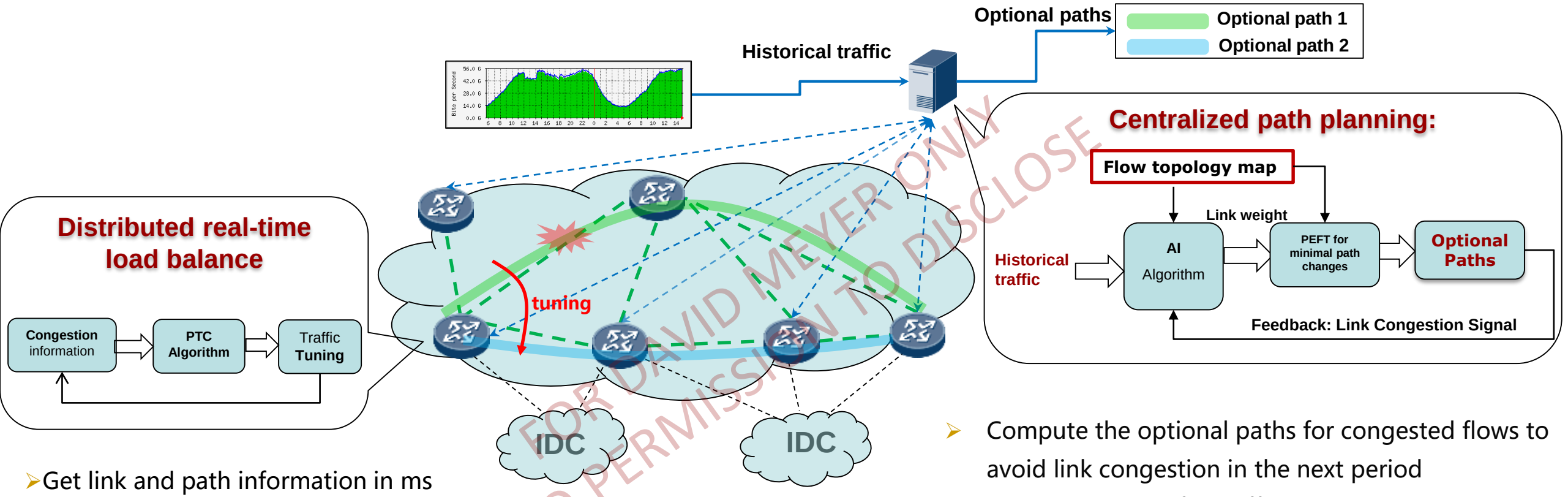
Unified Network/Computation/Storage Resource Management | Gateway/Controller Election — Anima General ASA

Event Queue | Distributed Storage | Multi-Agent Conflict Handle

Discovery | Synchronization | Negotiation | Flooding | Pub-Sub — Anima ANI and Beyond

Distributed Platform

Self-Optimization: Network-wide Load Balancing

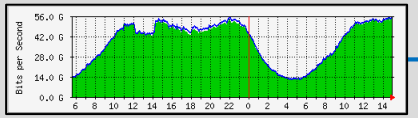


- Get link and path information in ms
- Use PTC algorithm to achieve traffic tuning on optional paths

- Compute the optional paths for congested flows to avoid link congestion in the next period
- Use AI algorithm for traffic prediction
- Use PEFT algorithm to ensure path adjustment with minimal changes

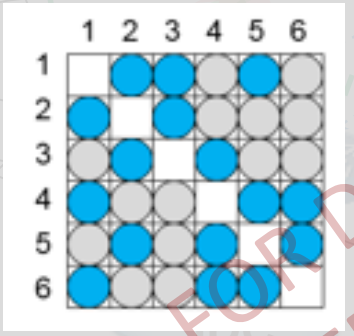
Self-Optimization: network-wide load balancing

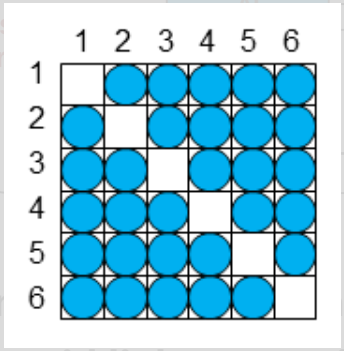
Historical traffic



A sub-problem: how to more effectively get the traffic matrix?

Using part of the traffic matrix information to "fill up" the whole traffic matrix

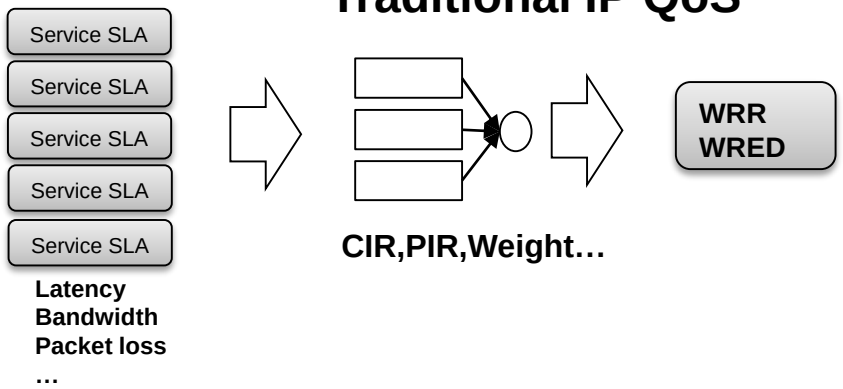




- "Fill up" algorithms: encoder-decoder, GAN
- Goal: using 30% TM info to fill up 90% TM
- Key challenge: how to select the 30% nodes

Self-Optimization: Autonomous SLA Guarantee

Traditional IP QoS

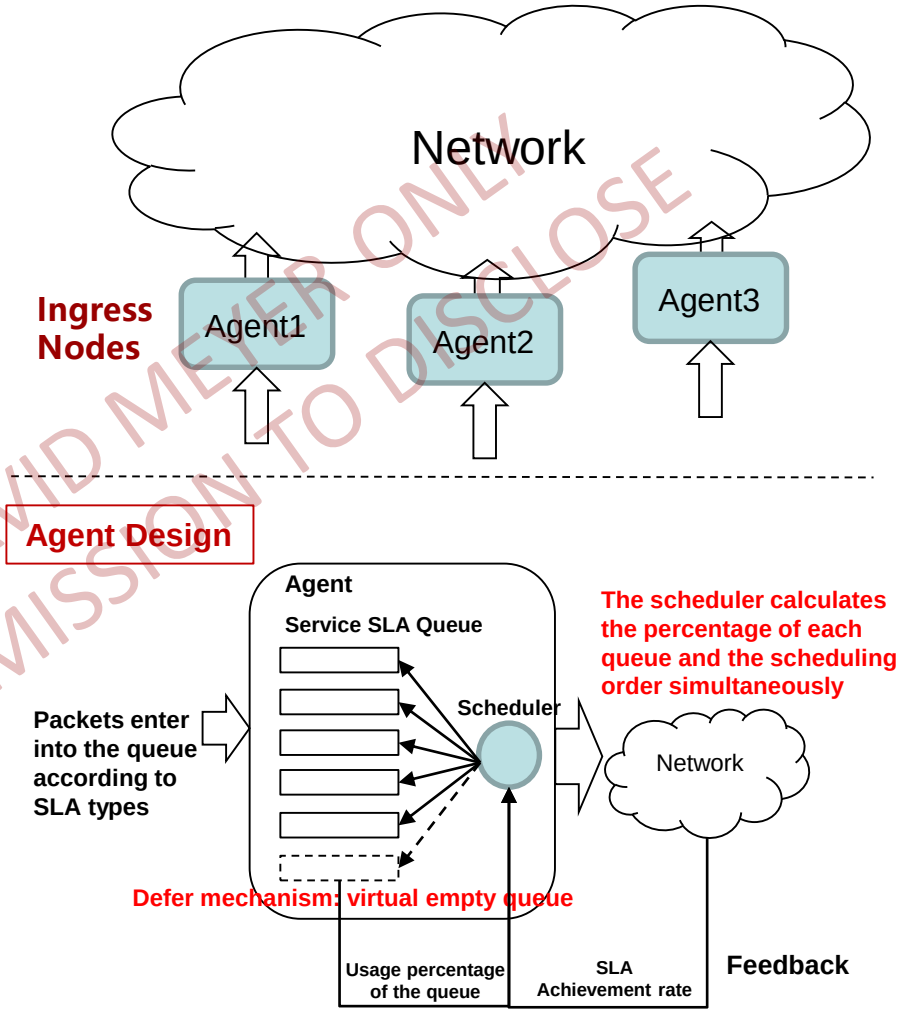


Design Goal:
Coarse granularity bandwidth operation to achieve DiffServ QoS model（cannot satisfy FCT in scenarios as DC etc.）

Principle:
Static IP QOS configuration based on admin’ s experience (cannot make real-adjustment according to network status)

Mechanisms:
WFQ for small scale scheduling DWRR for large scale scheduling etc.

Self-Scheduling



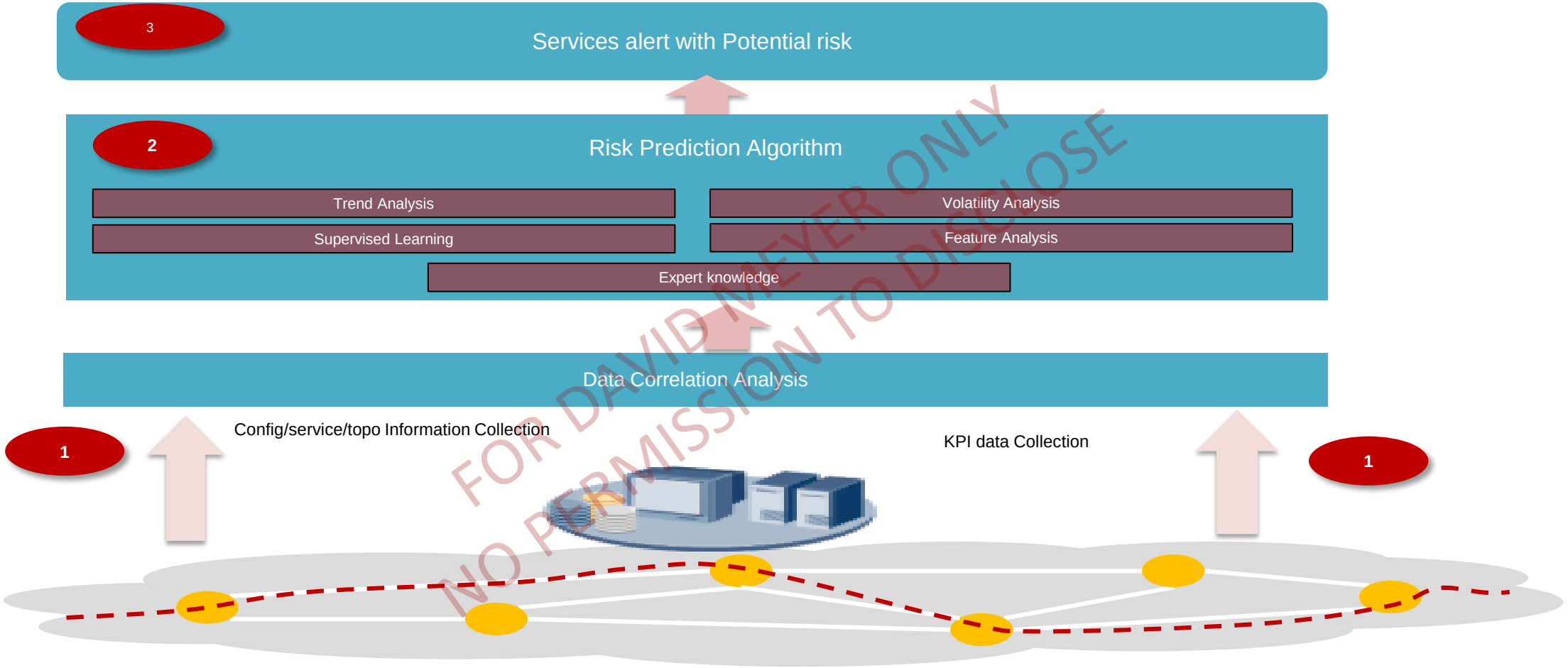
Designing Goal:
directly fulfill SLA

Principle:
real-time control loop to make packet scheduling adjustment according to SLA requirements and the real-time network feedback

Mechanism:
Neural Network based reinforcement learning to enhance WFQ

Self-Diagnostic/Healing

(Note: the work in this slide is owned by another department in Huawei)

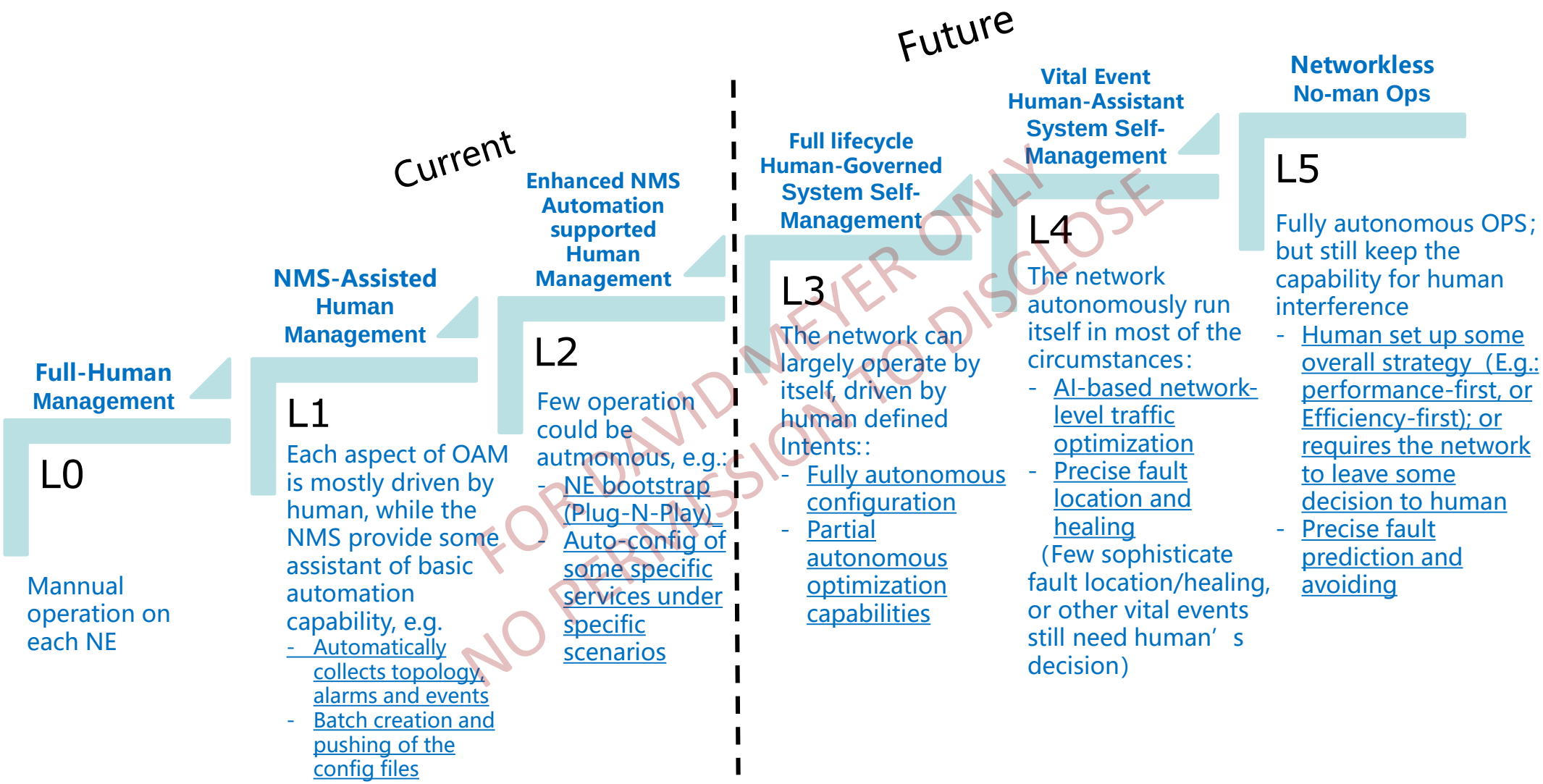


- over 60% of service failure caused by degraded optical performance can only be handled passively (OTN Network)
- AI-based scheme will predict network failures by learning from historical data pattern and adoption of advanced network model, with an accuracy as high as 80%.

Contents

- Background: Human cost issue
- IETF Standardization Work (Anima)
- Ongoing Research Projects
- A Roadmap to the Destination (Networkless)

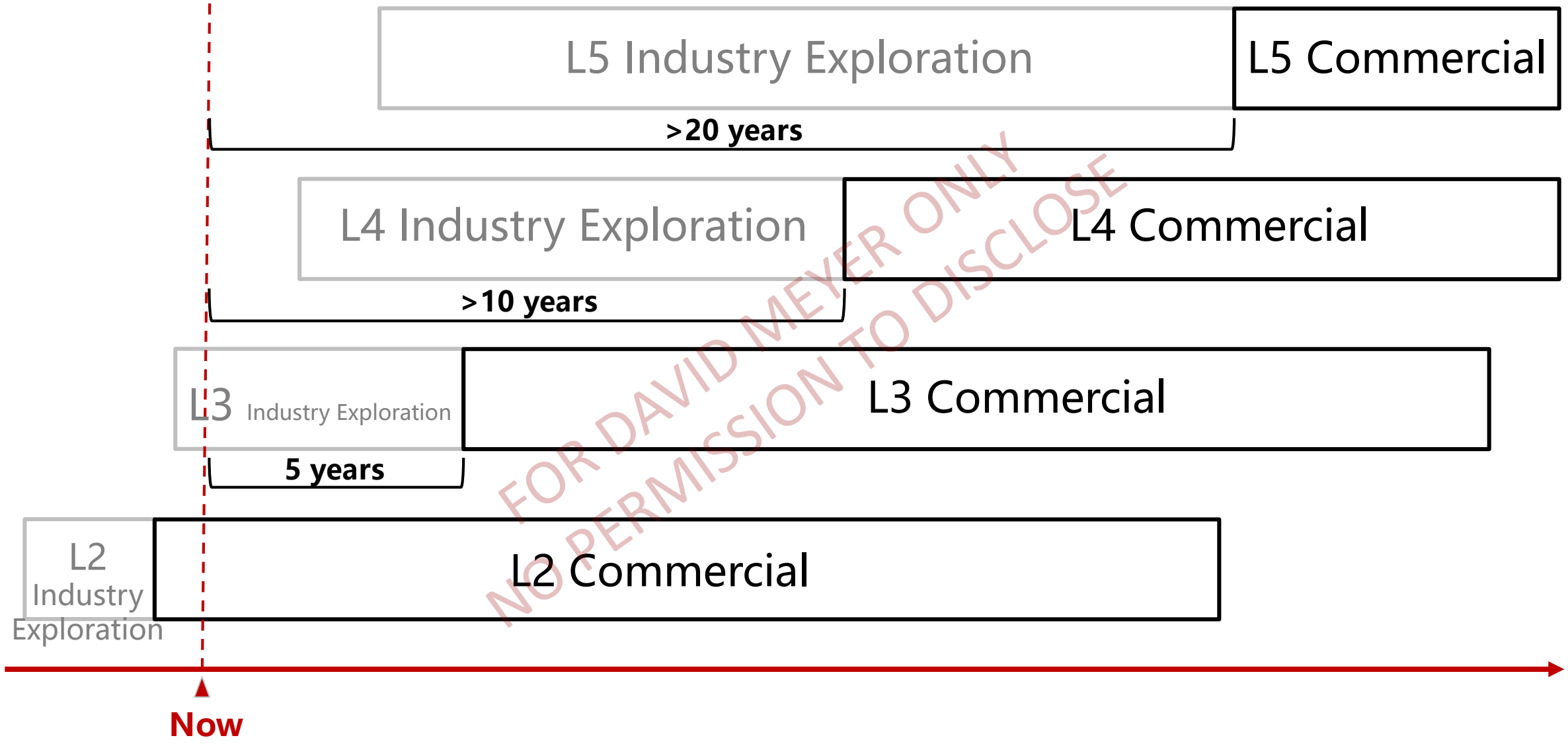
Autonomous Network Levels



“Self-” Capabilities in each Level

	Self-Org (NE bootstrap)	Self-Config	Self-org (domain/NE roles)	Self-optim	Self-diag	Self-healing
L5	System	System	System	System	System	System
L4	System	System	System	System	Human-assisted System	Human-assisted System
L3	System	System	System-assisted Human	System-assisted Human	System-assisted Human	System-assisted Human
L2	System	System-assisted Human	Human	Human	Human	Human
L1	System-assisted Human	System-assisted Human	Human	Human	Human	Human
L0	Human	Human	Human	Human	Human	Human

L2-L5 Timeslot (Note: some single point techniques might be ahead of the overall level)



L1-L5 Technologies Layout

	Self-Organization	Self-Configuration	Self-Optimization	Self-Diagnostic	Self-Healing
L5	**Self-Construction of Network Topologies** • for wireless network or overlay virtual networks	N/A	**Autonomous Optimization** • The network generates optimization policies by itself, and keep the performance at the best level; • Meanwhile, achieve balance between performance and cost.	**Precise Fault Prediction**	**Fault Avoiding** • According to the prediction, avoid the fault by backup, adjust traffic etc.
L4	**Network Architecture and NE roles Self-identification** • identify topology characteristics and divide network layers; • identify roles such as access/aggregation/ core gateway etc.	**Networking Policies Inference** • System infers network architecture and solutions and compiles detailed NE configs. • All detailed configs are hosted by software. • More and more machine-native configs rather than human interfaces.	**Comprehensive SLA/QoS Self-Optimization** • The network autonomically optimize delay, bandwidth etc. according to admin or App's requirements; • The network autonomically achieve measurement according to the optimization goal.		
L3	**Network Areas Self-Division and Key NEs election** • IGP Area self-division; controller election etc.	**NE Configs Auto-Compiling** • Admins design network architecture and solutions, the network autonomically compile detailed NE configs.		**Precise Fault Location** • Precise alarms to report the exact fault events. • Precise location to reveal the real root cause.	**Programmable Healing** • Admin can set specific healing policies based on a set of general and abstracted rules of dealing with fault.
L2	**NE Plug-N-Play** • NEs automatically get connected with the NMS, current solutions includes DCN, Anima ACP、ZeroTouch etc.	**Specific Scenarios/Protocols Auto-config** • User/Admin config service model; system compiles NE configs according to fixed rules	**Auto Traffic Load Balance** • Controller dynamically adjust paths to achieve balanced traffic load, according to specific algorithms; • NE can achieve port-based load balance locally	**Automatic Data Analysis** • Software collects data around the whole network, and use data mining/machine learning and decision tree to aggregate alarms and analyze the cause.	**Protocol-based Healing** • Fixed healing functions built into NEs, such as BFD、 FRR etc.
L1	N/A	**NE Configs Auto-delivery** • Admins design detailed configs of each NE, NMS automatically delivers the configs.	**Static Traffic Engineering**	**NMS-assisted manual diagnostic**	**NMS-assisted manual healing**

Key Supporting Capabilities

	Intent Expression/Interpreting	Operation Interface	Decision	Sensing/Analysis
L5	**Natural Language alike Intent** • NL style service description • System interprets it into networking policies	**Machine-native Autonomous API** • The machines would autonomously construct the content of the APIs to fulfill the need of collaboration between modules.	**Real AI**	**Network Event Prediction** **Traffic Trend Prediction**
L4	**Logic Expression Intent** • Describe networking build up policies/solutions • System interprets it into NE behaviors	**Network-level Declarative API** • User/Admin oriented declarative API, to make the network be called as a service.	**Machine Learning** **Machine Inference** • General control loops, driven by specific Intents (e.g. Intent provides the Reward definition of the reinforcement learning) • Config/optimization/diagnostic/healing policies inference	**Network Modeling** **Pattern Recognition** • Comprehensive modeling for complex network problems; • Pattern recognition to identify current network status
L3	**NE Intent** • Describe the NE-level policies such as config policies, config goal, optimization goal in a certain way that the NE can directly interpret it.	**NE-level Declarative API** • Controller oriented NE-level declarative API		**Real-time holographic data** • Network Digital Twin • NE deeply sense local traffic and fault etc.
L2		**NE-level Imperative API** • Controller oriented NE-level API containing detailed configurations. (E.g. Openflow、Netconf/YANG)	**Programmable Control Loops** • Algorithms (in Controller) for specific functions and scenarios (might embedded some Machine Learning capabilities.)	**Data Analysis** • Telemetry, Network Visualization, and logs analysis etc.
L1		**NE CLI**	**Fixed Control Loops** • Fixed process, such as IGP, DHCP, Anima BRSKI/ACP etc.	**Statistic, Probing** • Ping/Trace etc.

Thank you