Intent-Based Networking Another Wave of Hype?

Ivan Pepelnjak (ip@ipSpace.net)
Network Architect

ipSpace.net AG

Who is Ivan Pepelnjak (@ioshints)

Past

- Kernel programmer, network OS and web developer
- Sysadmin, database admin, network engineer, CCIE
- Trainer, course developer, curriculum architect
- Team lead, CTO, business owner

Present

- Network architect, consultant, blogger, webinar and book author

Focus

- SDN and network automation
- Large-scale data centers, clouds and network virtualization
- Scalable application design
- Core IP routing/MPLS, IPv6, VPN

CISCO CCIE EMERITUS

vmware vEXPERT

Intent-Based What?

> Intent-based networking signifies a paradigm shift for our industry and a completely new era of networking

> Intent-based networking (IBN) is a form of network administration that incorporates artificial intelligence (AI), network orchestration and machine learning (ML) to automate administrative tasks across a network.

Intent-Based What?

Intent

A **clearly formulated** or planned **intention** (thank you, M-W)

Intention

A determination to act in a certain way

What one intends to do or bring about (recursive: see recursive)

How is it different from *declarative programming*?

Declarative Programming

> In computer science, declarative programming is a programming paradigm — a style of building the structure and elements of computer programs — that expresses the logic of a computation without describing its control flow.

To paraphrase:

- We tell the system **what** we want it to do (or in **what state** we want it to be)
- This is sometimes called **policy** because it sounds better
- We don't tell the system **how** to do it

Is This What We're Talking About?

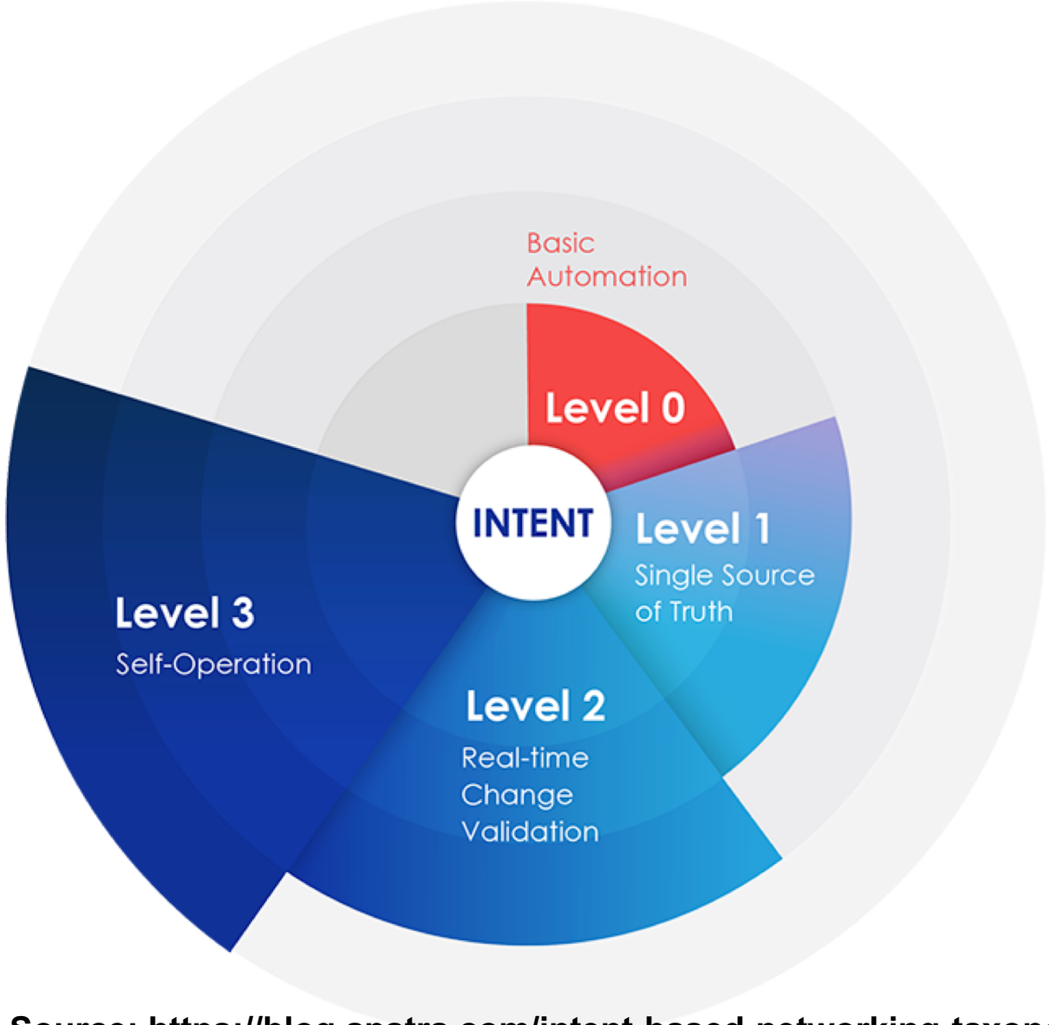
```
router ospf 1
 router-id 192.168.0.2
!
interface Loopback0
 ip address 192.168.0.2 255.255.255.255
 ip ospf 1 area 0
```

- Does it express our intent? Yes
- Do we tell the box how to do things? No
- Is this what $vendor marketers are talking about? Hmm…
- Is this what they will deliver? If only we'd be so lucky…

Is the whole thing just an orchestration system with an abstraction layer?

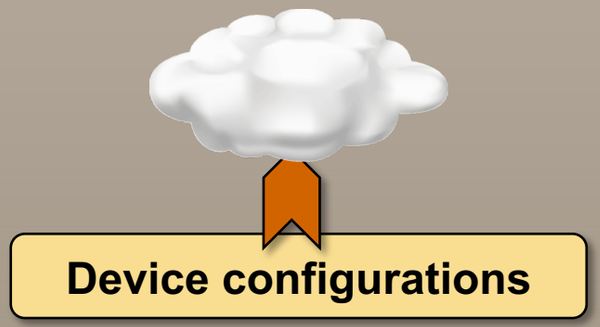
Many Layers of Intent

Intent-Based Networking Taxonomy



Source: https://blog.apstra.com/intent-based-networking-taxonomy

Device Configurations



Device Configuration = Intent

```
router ospf 1
 router-id 192.168.0.2
!
interface Loopback0
 ip address 192.168.0.2 255.255.255.255
 ip ospf 1 area 0
```

Device configurations are a classical example of declarative programming

- You tell the device what to do, not how to do it
- Sometimes it's hard to express your logic in declarative terms
- Welcome to nerd knobs and 12-step BGP route selection process

Device Configuration = Intent

```
router ospf 1
 router-id 192.168.0.2
!
interface Loopback0
 ip address 192.168.0.2 255.255.255.255
 ip ospf 1 area 0
```

- Every device configuration is really a data structure describing your intent
- In most cases masked by configuration syntax
- Very obvious when using XML in Junos or IETF/OpenConfig YANG data models
- Full of variables and values we don't care about (but devices do)

Abstract the Data Model

Everything should be made as simple as possible, but no simpler

- Structure the data into objects instead of using too many scalar variables
- Remove redundancies from the data model
- Use simple terms in data models and business logic to fill in the blanks

The supreme goal of all theory is to make the **irreducible basic elements as simple and as few as possible** without having to **surrender the adequate representation** of a single datum of experience.

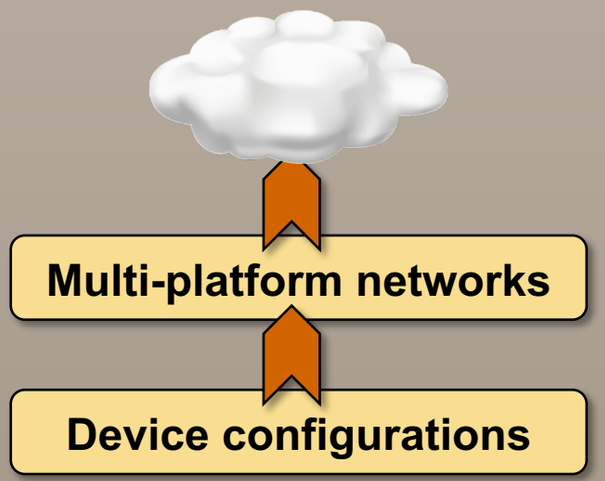
(Albert Einstein's original quote)

Sample Abstracted Device Data Model

```
hostname: E1
ospf:
  process_id: 1
  router_id: 192.168.0.2
  interfaces:
    "Loopback 0":
       area: 0
       cost: 10
```

- Describes the intended device configuration in more abstract terms
- Used to generate target device configuration
- Syntactic fluff is removed from the data model
- **Use configuration templates to transform data models into usable device configurations**

Multi-Platform Networks



Each Platform Has a Intent Syntax

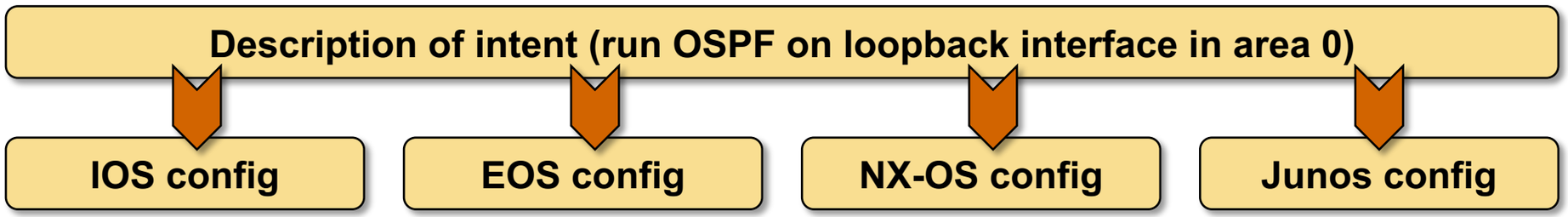
```
router ospf 1
 router-id 192.168.0.2
!
interface Loopback0
 ip address 192.168.0.2 255.255.255.255
 ip ospf 1 area 0
```

Expressing your intent becomes interesting when dealing with multiple platforms (even from the same vendor)

- It's like trying to speak English, French and Mandarin at the same time
- Why can't everyone just agree on a single language?

Using a Unified Data Model

```
hostname: E1
ospf:
  process_id: 1
  router_id: 192.168.0.2
  interfaces:
    "Loopback 0":
       area: 0
       cost: 10
```

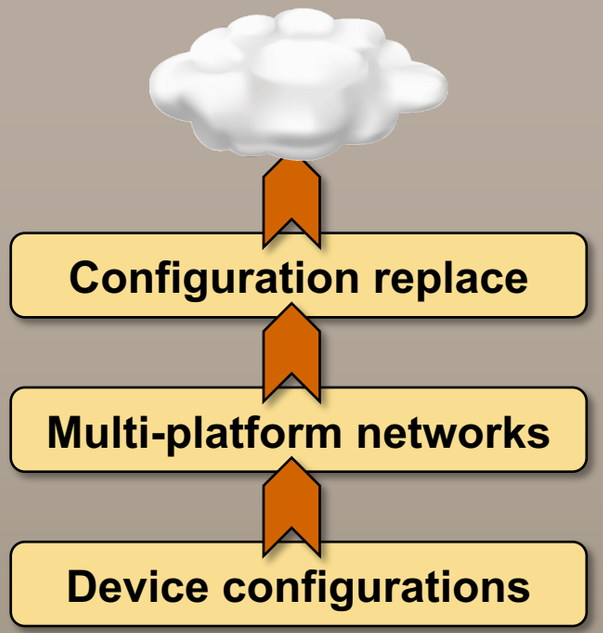


Reinventing the Wheel

```
hostname: E1
ospf:
  process_id: 1
  router_id: 192.168.0.2
  interfaces:
    "Loopback 0":
       area: 0
       cost: 10
```

- You're not the only one thinking about data models
- IETF and OpenConfig working group are doing the same
- Use their models if they make sense
- They will never cover all your needs

Replace Configuration



Simple Changes Result in Complex Operations

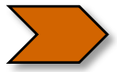
```
hostname: E1
ospf:
  process_id: 2
  router_id: 192.168.0.2
  interfaces:
    "Loopback 1":
       area: 0
       cost: 10
```

We want to slightly modify our intent

- Rename OSPF process ID
- Rename loopback interface

How Do You Get From Here to There?

```
router ospf 1
 router-id 192.168.0.2
!
interface Loopback0
 ip address 192.168.0.2…
 ip ospf 1 area 0
```



```
router ospf 2
 router-id 192.168.0.2
!
interface Loopback1
 ip address 192.168.0.2…
 ip ospf 2 area 0
```

Making that change on most network devices is excruciating

- Requires careful choreography
- Easy to get it wrong
- **reload in 5** is your only friend

The Right and Wrong Ways of Doing Things

Changing configuration of a Linux daemon:

- Edit desired configuration (your intent) – on device, off device, with an orchestration system…
- Tell the daemon to reload configuration
- In many cases a hitless operation

Changing network device configuration:

- Compare current and desired configuration (sometimes manually)
- Execute a number of steps that will bring current configuration as close to desired configuration as possible (have fun with **service-policy**)

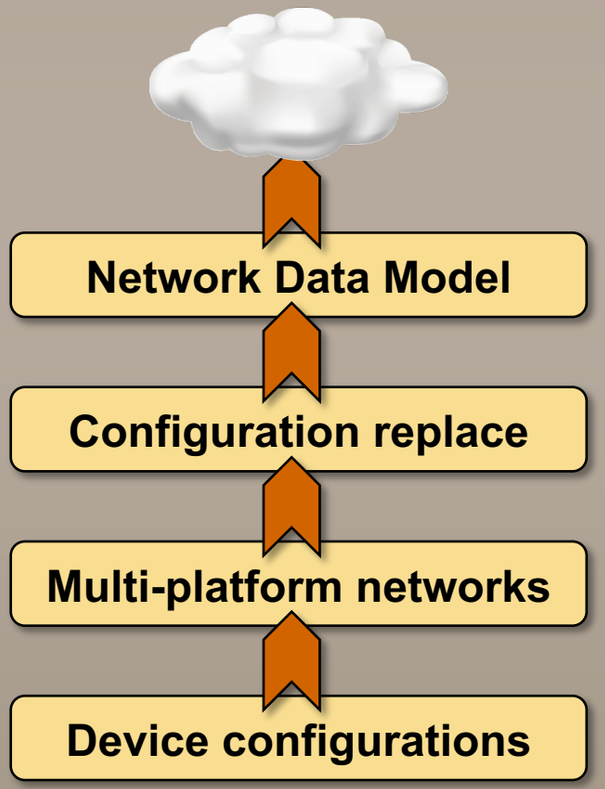
Network devices were never designed as all-or-nothing systems

What Should You Ask For?

- Programmable interface (API)
- Structured operational data (in JSON or XML format)
- Device configuration in structured (JSON/XML) format
- **Atomic configuration changes**
- Configuration rollback
- **Configuration replace**
- **Contextual configuration diff**
- Support for industry-standard models (IETF and OpenConfig)
- Feature parity (API to CLI)

More @ http://blog.ipspace.net/2016/10/network-automation-rfp-requirements.html

Network-Wide Data Model



Describe the Desired Network-Wide State

```
---
fabric:
  - {left: E1, left_ip: 10.0.0.21, left_port: GigabitEthernet0/2,
     right: E2, right_ip: 10.0.0.22, right_port: GigabitEthernet0/2,
     cost: 5 }
  - {left: E1, left_ip: 10.0.0.13, left_port: GigabitEthernet0/1,
     right: PE1, right_ip: 10.0.0.14, right_port: GigabitEthernet0/1,
     cost: 10 }
```

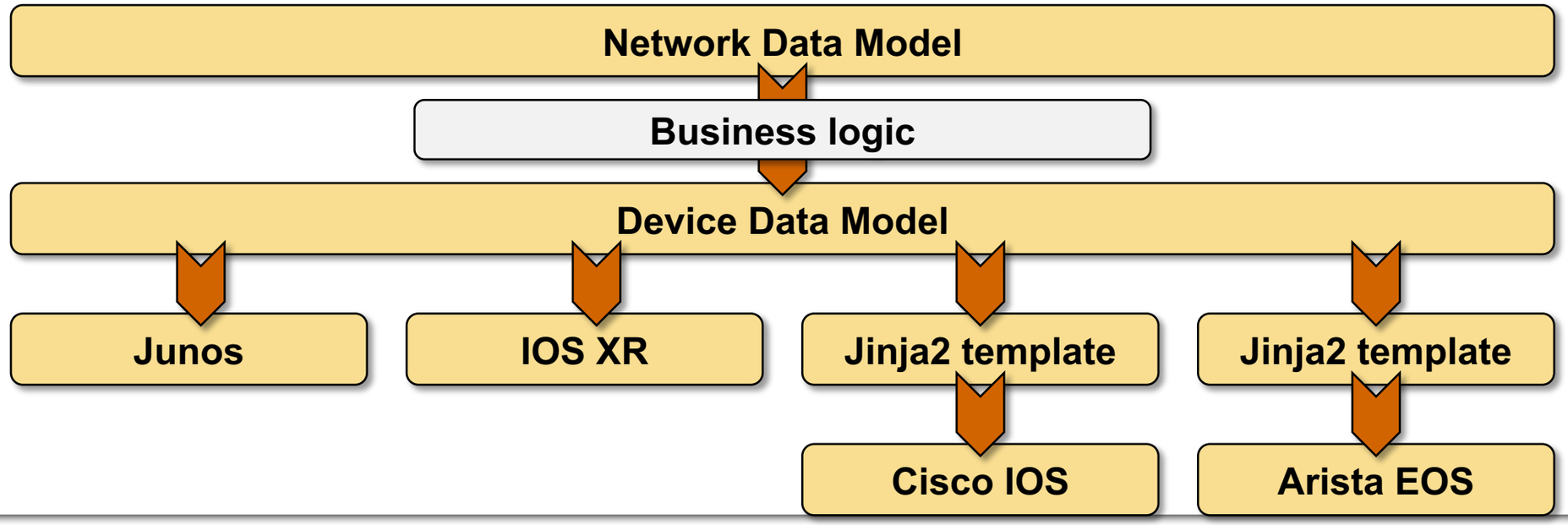
Describe the desired state of the network, not individual devices

- Nodes and links
- Core and edge links
- Subnets instead of IP addresses

More @ https://www.ipspace.net/kb/DataModels/

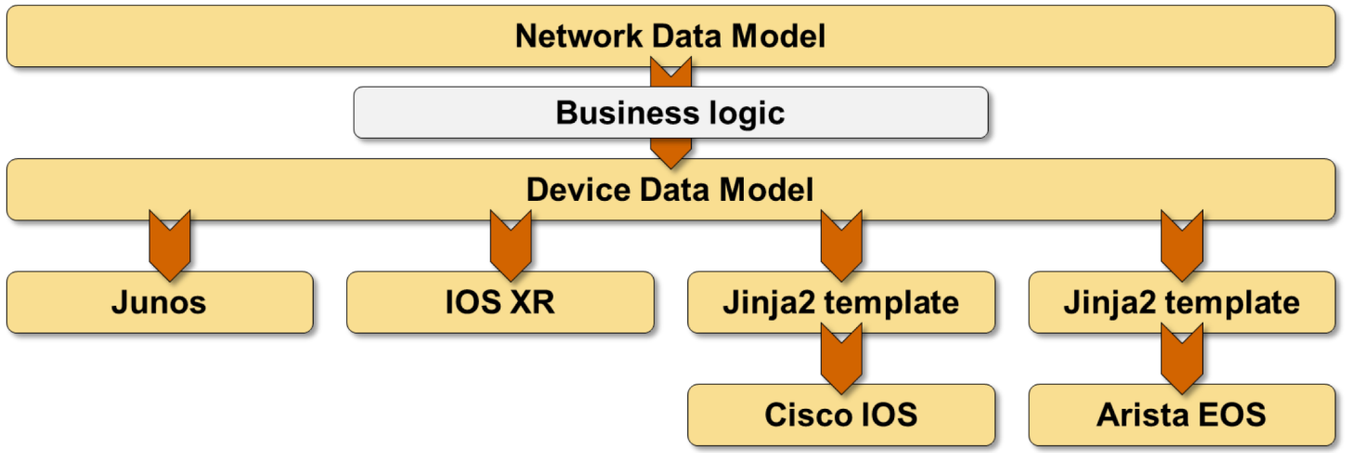
Using the Network-Wide Data Model

```
---
fabric:
  - {left: E1, left_ip: 10.0.0.21, left_port: GigabitEthernet0/2,
     right: E2, right_ip: 10.0.0.22, right_port: GigabitEthernet0/2,
     cost: 5 }
  - {left: E1, left_ip: 10.0.0.13, left_port: GigabitEthernet0/1,
     right: PE1, right_ip: 10.0.0.14, right_port: GigabitEthernet0/1,
     cost: 10 }
```



Typical Business Logic

- Automatically allocate IPv4 subnets to core links
- Compute IPv4 interface addresses
- Select routing protocol to use on core links
- Set routing protocol parameters
- Create a list of BGP neighbors

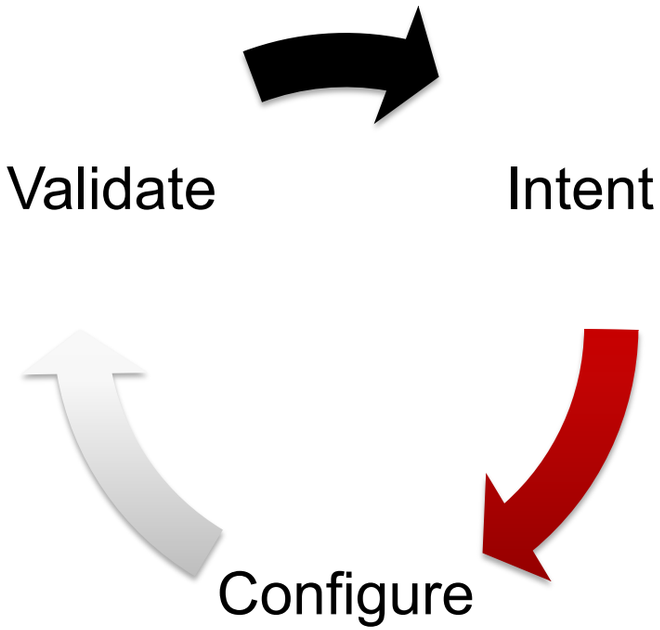


Ultimate Simplification of the Data Model

```
---
num_spines: 2
num_leaves: 4
hosts_per_leaf: 1
spine_to_leaf_ports: "swp[1-4]"
leaf_to_spine_ports: "swp51 swp52"
leaf_to_server_ports: "swp[1-2]"
```

- Specify the number of switches in a fabric
- Specify connectivity (port ranges)
- The business logic creates all fabric configuration data

Use Network Data Model to Validate Network State




```
---
fabric:
  - {left: E1, left_port: GigabitEthernet0/2,
     right: E2, right_port: GigabitEthernet0/2,
     cost: 5 }
  - {left: E1, left_port: GigabitEthernet0/1,
     right: PE1, right_port: GigabitEthernet0/1,
     cost: 10 }
```

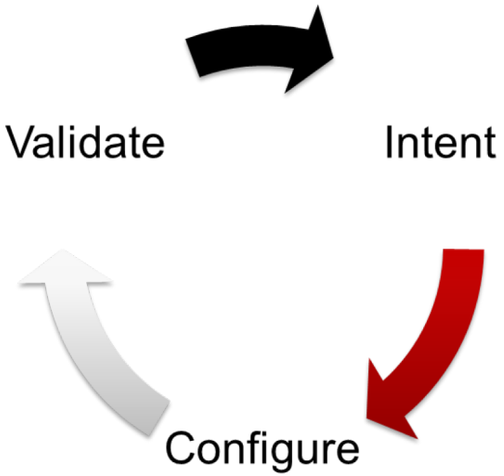
This is what Apstra AOS is doing

Sample Validations

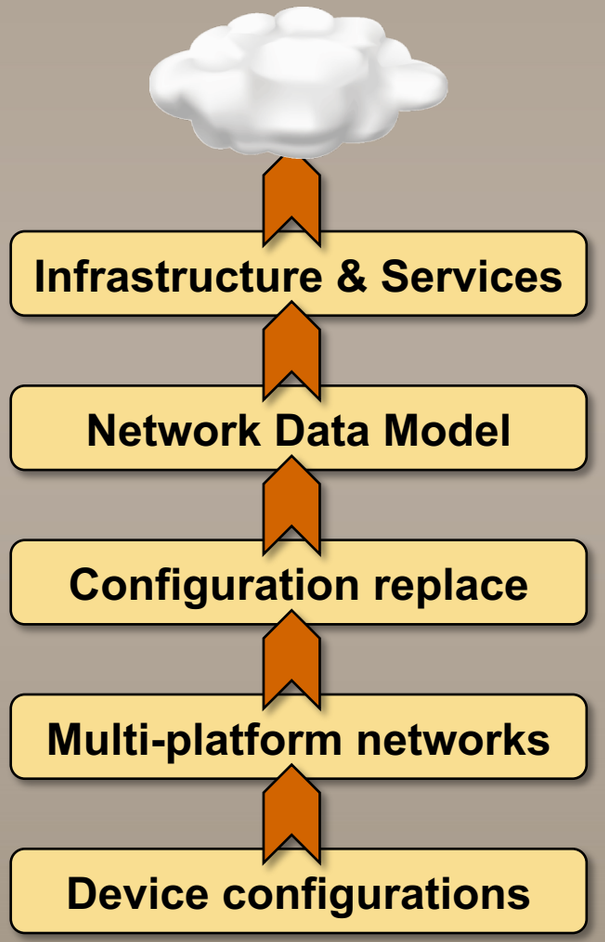
- Are all core interfaces operational?
- Is the fabric wired correctly (based on LLDP neighbors)
- Are OSPF or BGP adjacencies established?
- Are the correct routes propagated across the network?
- Are the end-to-end connectivity tests successful?

You could do all these tests in a traditional network

- Reverse-engineer network topology from device configurations
- Snapshot a state and make it the baseline
- **Use a network data model is easier, cleaner, and more predictable**



Infrastructure and Services Data Models



Infrastructure and Services Data Model

Split the data model into two or more data models:

Infrastructure data model(s) define stable transport network

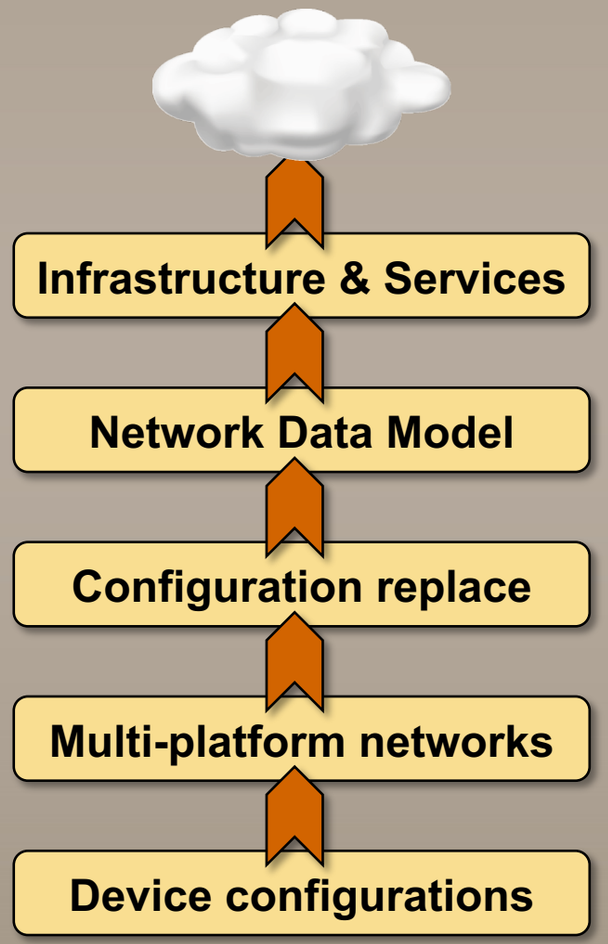
- Reflects the network topology
- Changes are rare
- Rarely needs self-service or instantaneous deployment

Services data model(s) are customer-focused

- Assume a stable transport network
- Define services delivered on top of the transport network
- Continuously changes
- Self-service and on-demand deployment is highly desirable
- Might need transactional consistency

This is what Cisco NSO is doing

Automated Remediation



Automated Network Remediation

Holy Grail: Networks that fix themselves or adapt to changes

A few examples:

- Identify links with degraded performance ➔ reroute traffic
- Identify router problems (memory leaks) ➔ drain the traffic, reload the device
- ToR switch failure ➔ migrate the virtual machines

Getting there:

- Don’t expect a vendor-supplied miracle
- Someone will have to do extensive customization
- Try to use small, reusable components

We had this for ages: see also Routing Protocols

Closing the Loop: Automating Event Management

Detect changes in the network

- Syslog, telemetry…
- Continuous validation

Correlate telemetry data and changes to detect significant events

- Hard problem
- Don’t expect machine learning miracles any time soon

Decide which events are worth acting upon

Define what to do when an actionable event happens

- The really hard part (see also: Byzantine Generals Problem)
- Sometimes a manual intervention is the only way to go

You Need Data… Lots of Data

Detecting short bursts is even harder

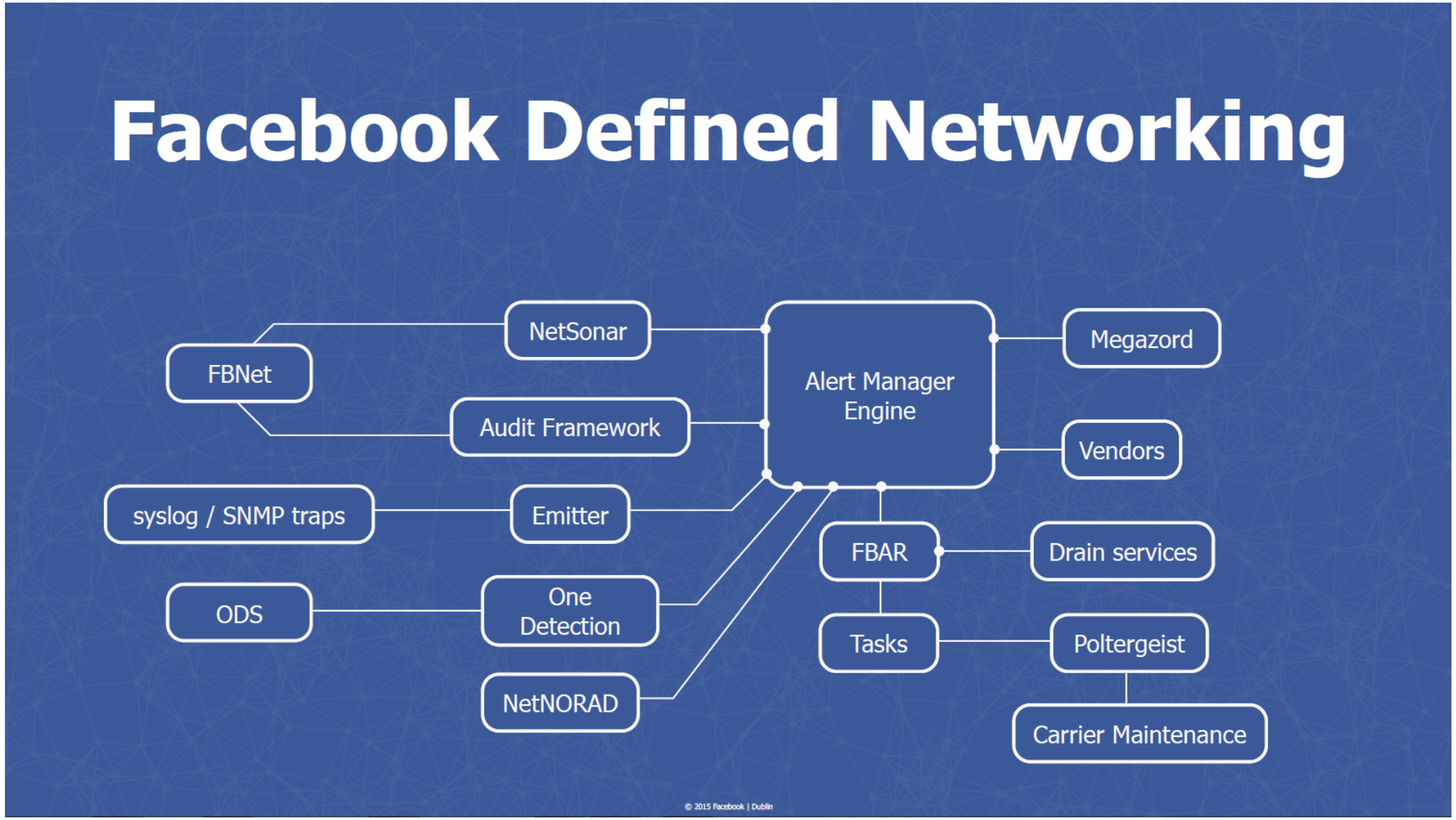
- 5-minute averages are not good enough, you need at least 95%
- Short polling intervals (few seconds) make little sense
- Streaming telemetry is the only way to go

You're looking for an unknown needle in a haystack

Build the tools (and responses to events) as you identify the challenges

- You need an extensible tool
- Use a modular architecture
- Use lessons learned from failures to grow the toolkit

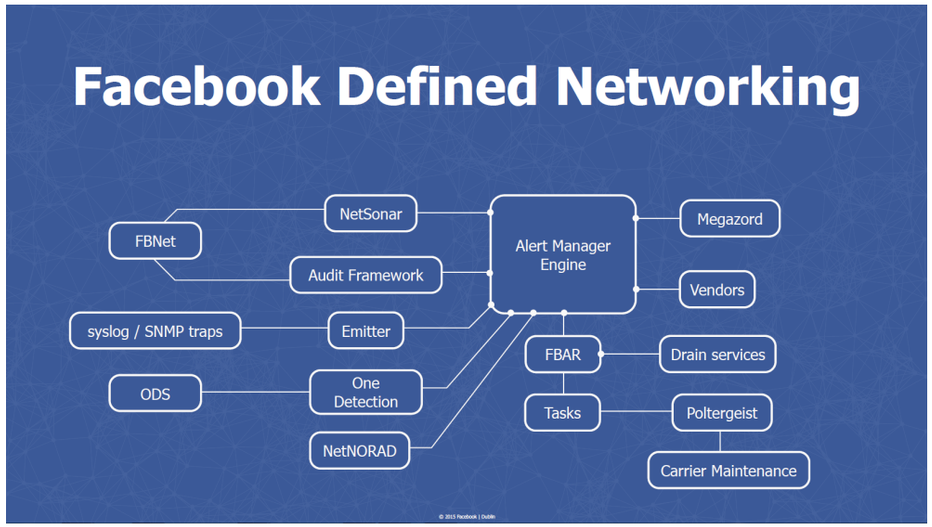
What Works: Facebook-Defined Networking



Source: How Facebook Learned to Stop Worrying and Love the Network (Jose Leitao, David Rothera, RIPE 71)

Lessons Learned

- Start small
- Build components as you need them (the Unix way)
- Integrate components into bigger systems
- Add auto-remediation logic as needed
- It ain’t done till you tested it in production (several times)



Marketing Promises

> […] a vision to create an intuitive system that anticipates actions, stops security threats in their tracks, and continues to evolve and learn. It will help businesses to unlock new opportunities and solve previously unsolvable challenges in an era of increasing connectivity and distributed technology.

> […] an intuitive system that constantly learns, adapts, automates and protects, to optimize network operations and defend against today's evolving threat landscape.

Back to Reality

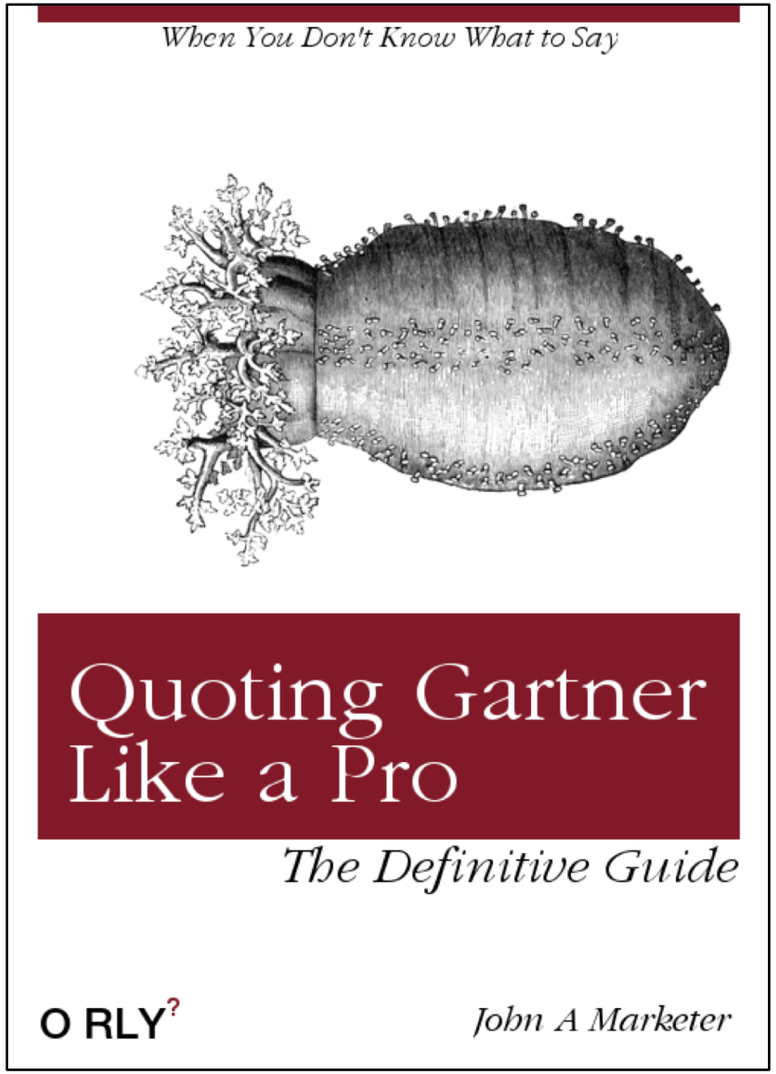
Quoting Gartner Like a Pro

Translation and Validation– The system takes a higher-level business policy (*what*) as input from end users and converts it to the necessary network configuration (*how*). The system then generates and validates the resulting design and configuration for correctness.

Automated Implementation – The system can configure the appropriate network changes (*how*) across existing network infrastructure. This is typically done via network automation and/or network orchestration.

Awareness of Network State – The system ingests real-time network status for systems under its administrative control, and is protocol- and transport-agnostic.

Assurance and Dynamic Optimization/Remediation– The system continuously validates (in real time) that the original business intent of the system is being met, and can take corrective actions (such as blocking traffic, modifying network capacity or notifying) when desired intent is not met.



Questions from the Field

- What they really mean by Intent Based Networking?
- How do they describe the Intent to the system?
- How do they map Business Intent & Technology intent and describe it to system?
- Does the system translate the intent into configurations/policies on it's own or they use some sort of policy language for admin to be used to describe Business and Technical intents to system ?
- How does intent get documented and updated over time?
- How do you describe intent for things like Backup path etc. ?
- How do you modify Intent over time and what would be the touch points?
- How is their Intent Based Networking different from policy based network (Remember Promise Theory that ACI Works on) ?
- Does the intent description part only takes care of configuration side of things or go further with Network Design as part of another abstraction layer ?

Source: Deepak Arora, http://deepakarora1984.blogspot.com/2017/11/few-questions-you-should-ask-your-fav.html

Questions from the Field (2)

- Are there any dry run capabilities into the system ?
- What sort of checks does the system use to verify described intent and return feedback to admin?
- How do they maintain the consistency of the intent throughout the life-cycle - Plan, Design, Deploy, Verify, Operate & Optimize?
- How do they create visualization of Intent to present to management and architects?
- Which Data model they follow to describe intent and push/change configurations?
- How do they deal with mix of legacy networks & Cloud Networks in hybrid environment?
- How do they handle leaky abstraction & grey failures?
- How do they deal with scale?
- Do they support APIs to describe intent and to work with other systems as part of larger ecosystem?

Source: Deepak Arora, http://deepakarora1984.blogspot.com/2017/11/few-questions-you-should-ask-your-fav.html

Questions from the Field (3)

- How do they control AI and ML since those systems can create their own algorithms and break the closed loop over time? (remember what happened with Facebook's AI attempt)
- How do they track changes in networks created by AI/ML dynamically?
- How does AI/ML understand if it's actual pattern change in network during attack or just traffic pattern change driven by an special event such as heavy load on billing systems during financial year end for example?
- Do they use graph approach (building Network Graph such as Link State Protocol Does) or a relation approach (describing links with properties) across network components to describe intent?

Source: Deepak Arora, http://deepakarora1984.blogspot.com/2017/11/few-questions-you-should-ask-your-fav.html

What Could Possibly Go Wrong?

> *(6) It is easier to move a problem around (for example, by moving the problem to a different part of the overall network architecture) than it is to solve it.*
> *(6a) (corollary). It is always possible to add another level of indirection.*

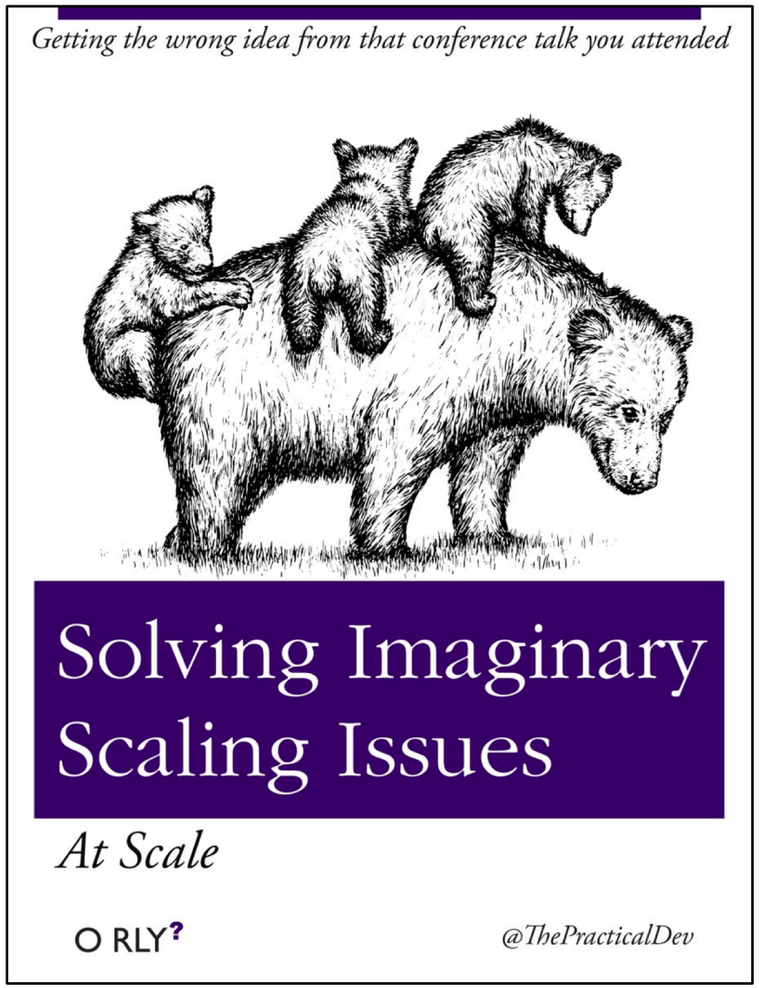
RFC 1925

> *All non-trivial abstractions, to some degree, are leaky (exposing details that should remain hidden)*

Joel Spolsky (law of leaky abstractions)

You're replacing

- Device vendor lock-in with orchestration vendor lock-in
- Your network architects with vendor programmers



Or you can invest into premium people and build your own

Questions?

Web:	ipSpace.net
Blog:	blog.ipSpace.net
Email:	ip@ipSpace.net
Twitter:	@ioshints

Data center:	ipSpace.net/NextGenDC
Automation:	ipSpace.net/NetAutSol
Webinars:	ipSpace.net/Webinars
Consulting:	ipSpace.net/Consulting