Taking Network Management into the 21st Century.

Rob Shakir (rjs@jive.com)
IPArch/NetDevOps - Jive Communications, Inc.

Paris, France - January 2016

HELLO!

Rob Shakir
IP Architect / “NetDevOps” @ JIVE

#1: STARTUP

catalyst2

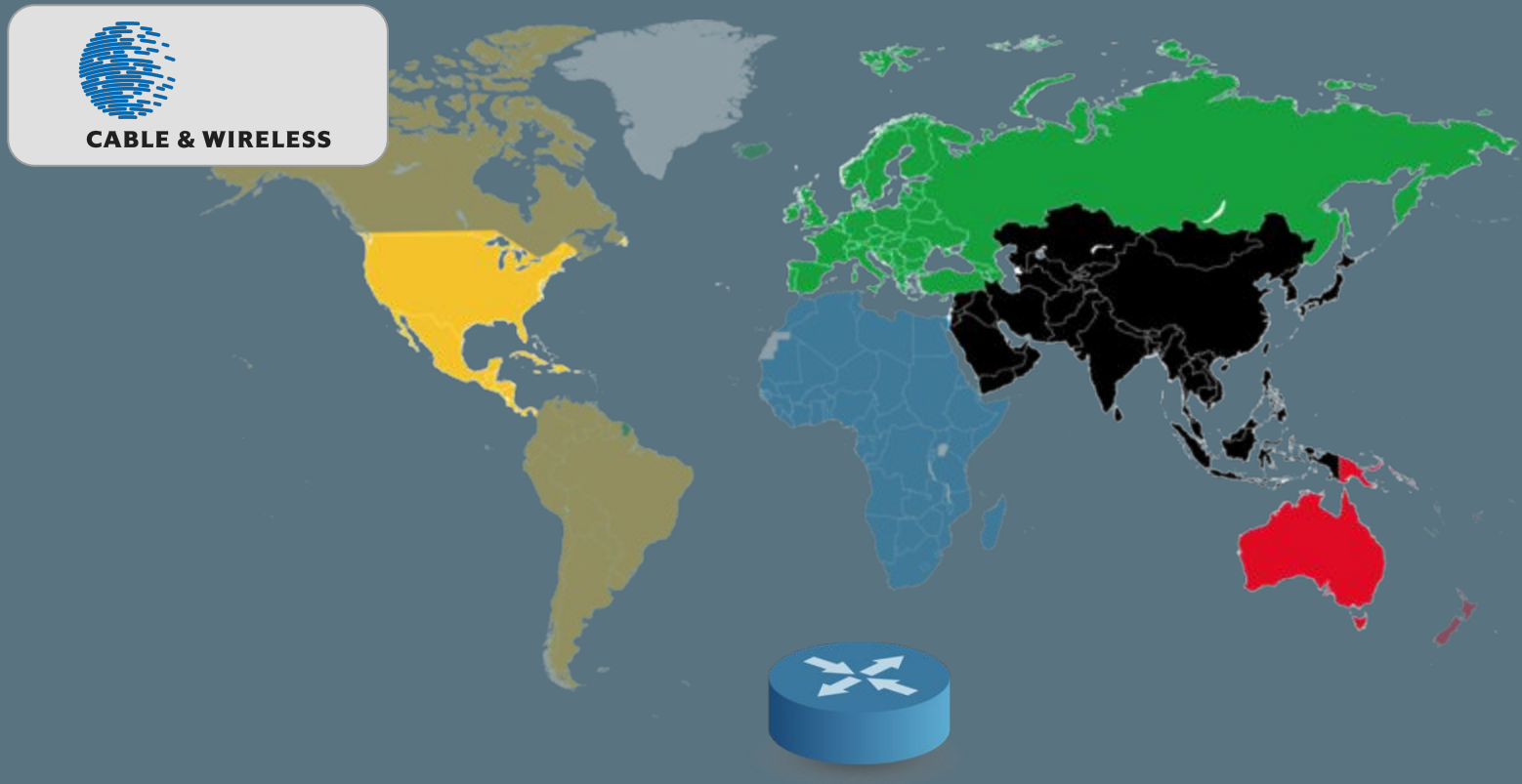
Servers, network, management infrastructure, billing...

#2: NSP

pipex

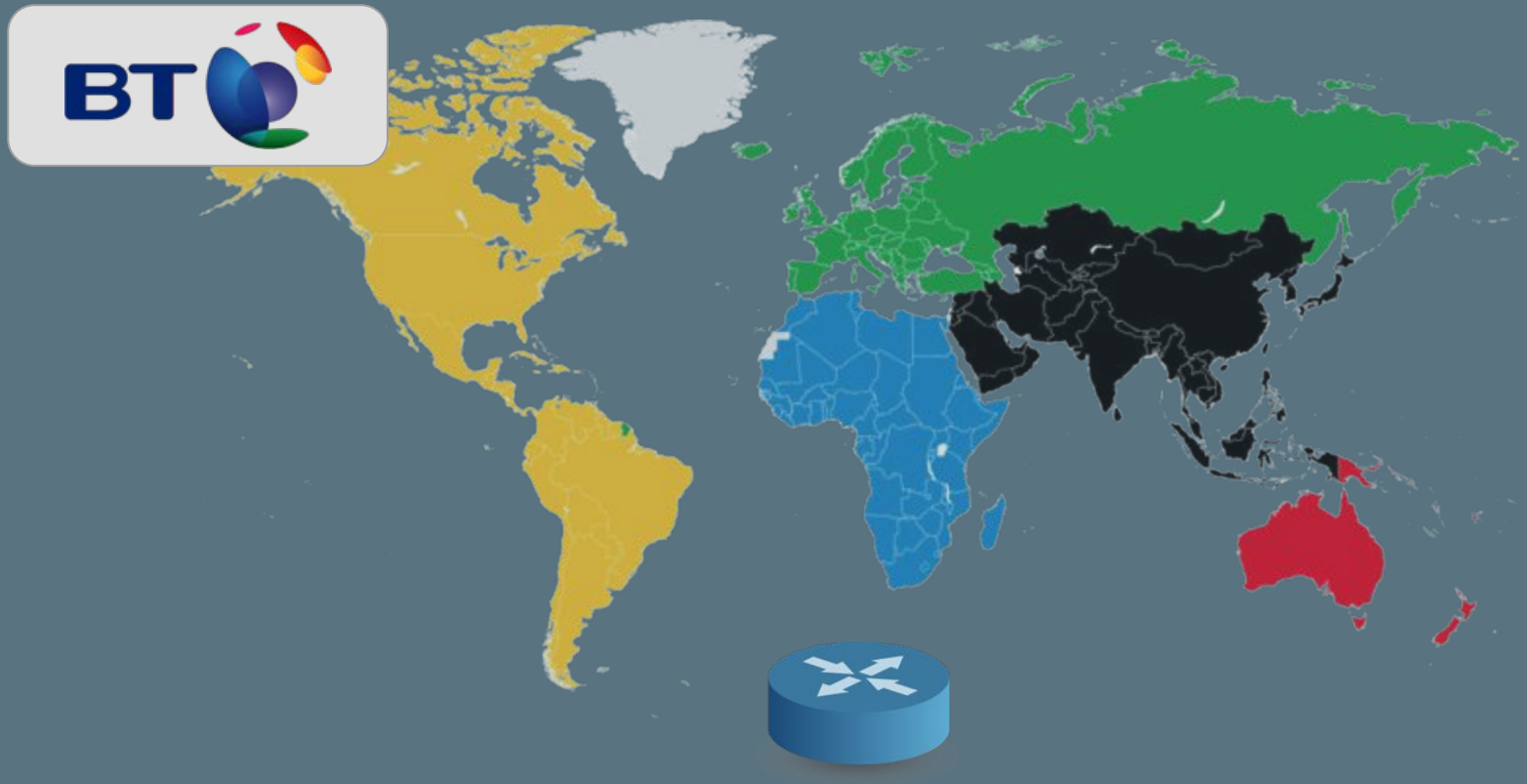
Network infrastructure, and management tooling.

#3: GLOBAL TELCO



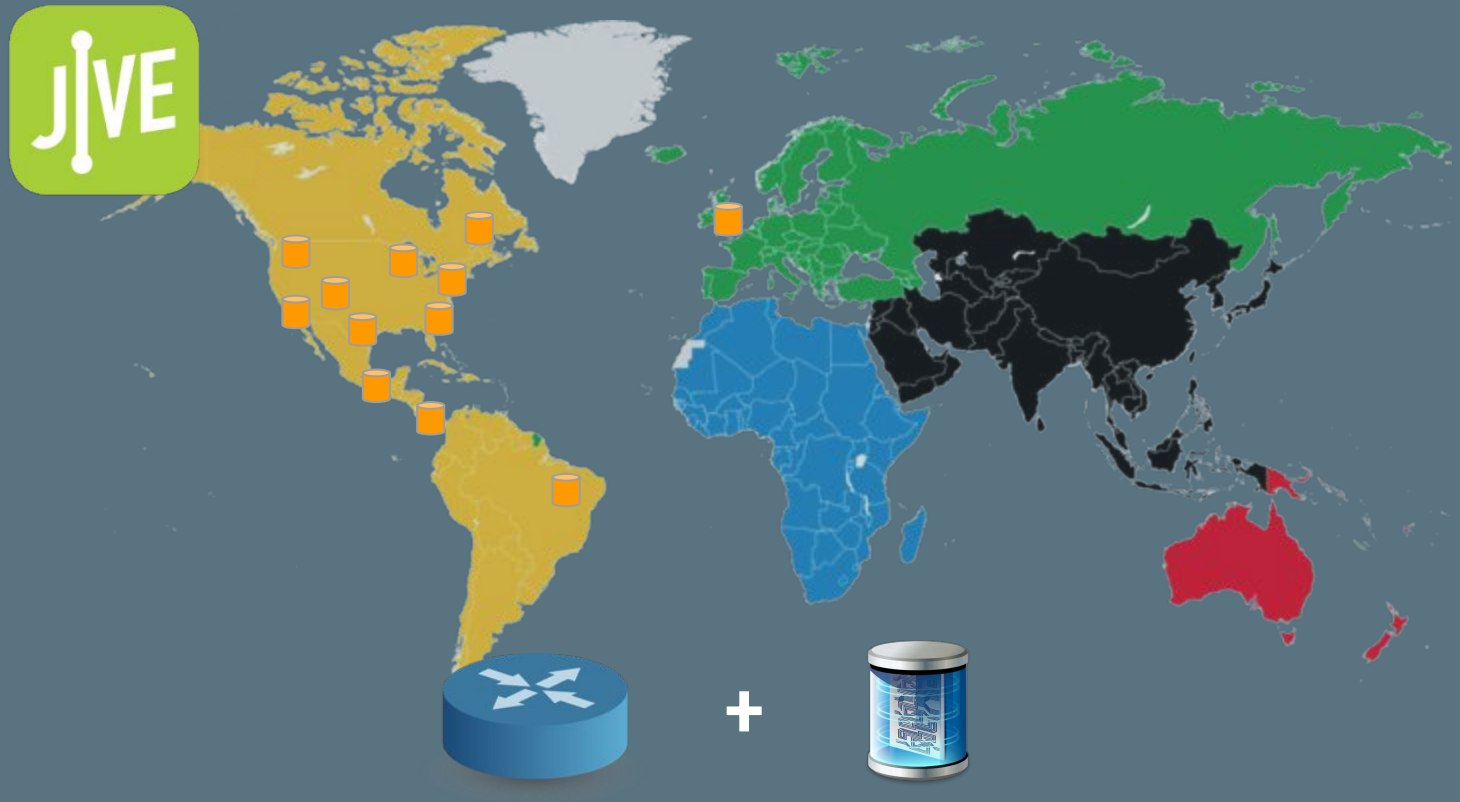
Network infrastructure - but lots of it!

#4: INCUMBENT TELCO



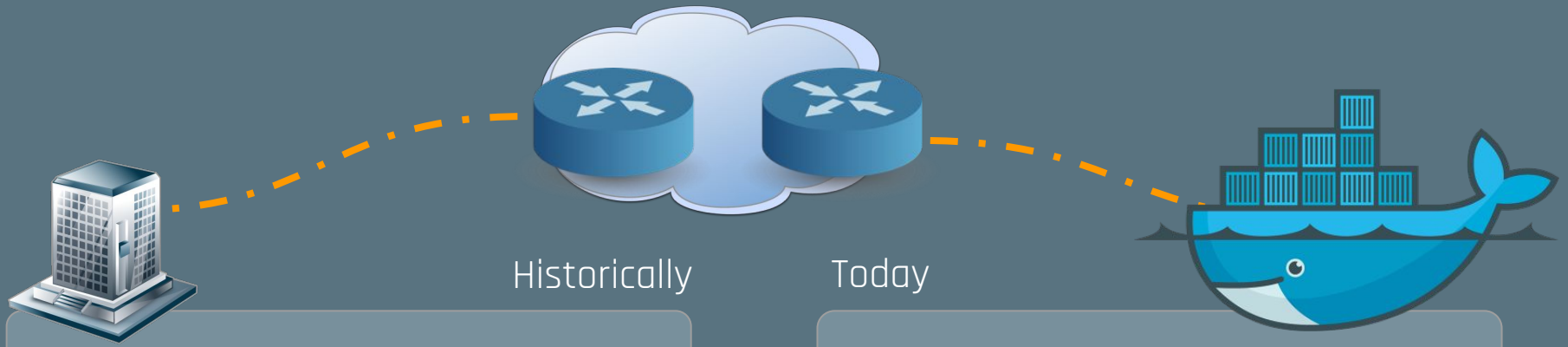
A huge array of different infrastructure and technologies.

CURRENTLY: CLOUD UC PROVIDER



The ‘full network stack’

HOT TOPIC: MANAGING THE NETWORK



Historically

- 30-90+ days to plan changes - physical build required.
- Change very infrequent - requires on-site changes or reprovisioning.

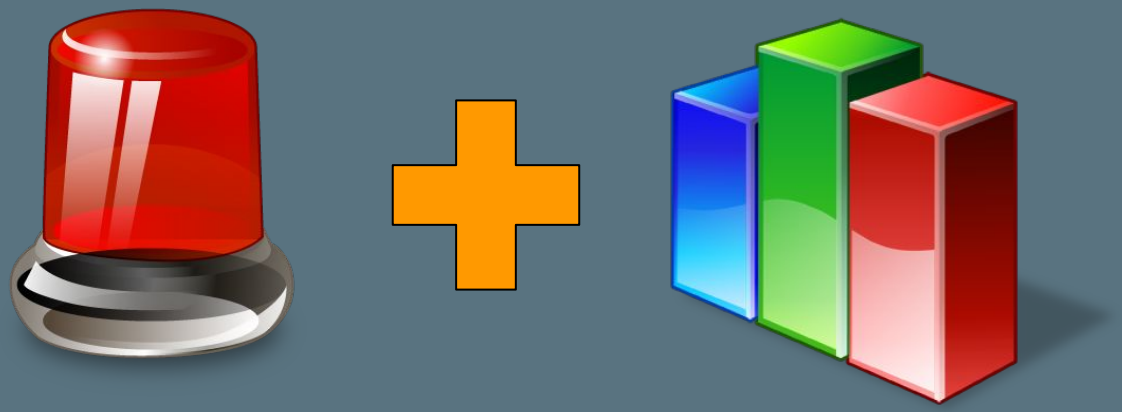
Today

- Zero time to plan changes - self-service infrastructure.
- Change is very frequent - can be reactive (to failures or load characteristics)

MANAGING A NETWORK SERVICE: #1: PROVISIONING

- Typical process thought of when considering 'management'
- Taking details of a service - and instantiating configuration on devices
- Requires **read/write** interfaces - to validate current config and add new

MANAGING A NETWORK SERVICE: #2: MONITORING

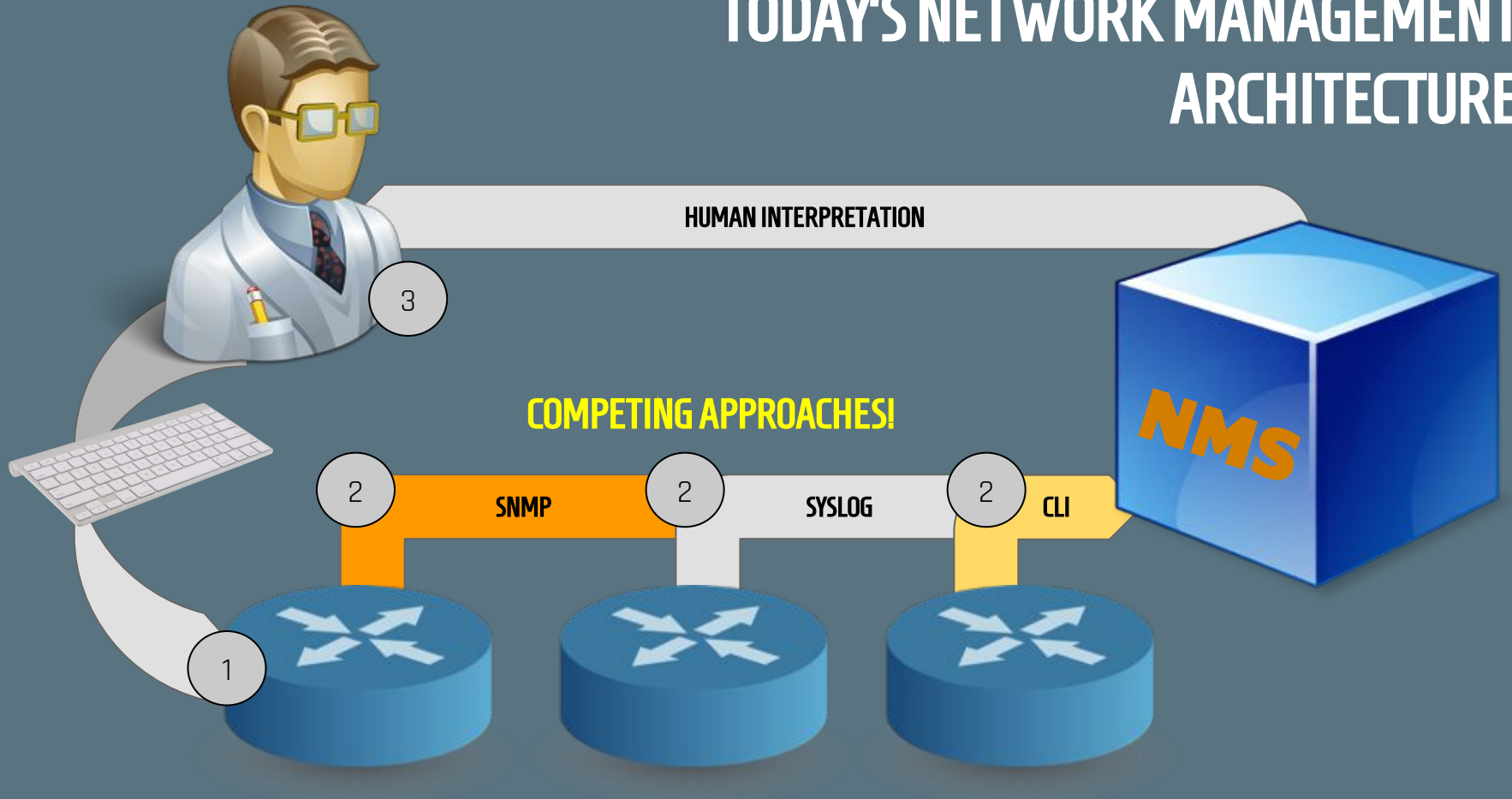


- Retrieval of events and metrics that are associated with a configured entity
- Alarming - determining that an issue (e.g., failure) has occurred
- Metrics - used to bill, analyse quality of experience and capacity planning.

MANAGING A NETWORK SERVICE: #3: RESPONDING TO EVENTS

- Reacting to events that occur within the network - and determining remedial measures
- Key element of the management of a network - actually fixes problems!

TODAY'S NETWORK MANAGEMENT ARCHITECTURE



FOCUSING DOWN ON SNMP + IETF MIBS

Data organised into tables - each with indexes

```
rjs@mgmt:~$ snmpwalk -v2c -c COMMUNITY ROUTER PATH
iso.3.6.1.2.1.31.1.1.1.1.1 = STRING: "Gi0/0/0"
iso.3.6.1.2.1.31.1.1.1.1.2 = STRING: "Gi0/0/1"
iso.3.6.1.2.1.31.1.1.1.1.3 = STRING: "Gi0/0/2"
iso.3.6.1.2.1.31.1.1.1.1.4 = STRING: "Gi0/0/3"
iso.3.6.1.2.1.31.1.1.1.1.5 = STRING: "Gi0"
iso.3.6.1.2.1.31.1.1.1.1.6 = STRING: "Vo0"
iso.3.6.1.2.1.31.1.1.1.1.7 = STRING: "Nu0"
iso.3.6.1.2.1.31.1.1.1.1.8 = STRING: "Lo0"
iso.3.6.1.2.1.31.1.1.1.1.9 = STRING: "Lo1"
iso.3.6.1.2.1.31.1.1.1.1.10 = STRING: "Gi0/0/3.4"
iso.3.6.1.2.1.31.1.1.1.1.11 = STRING: "Gi0/0/3.10"
```

Queries over UDP with get and bulk get operations

Typed data definitions

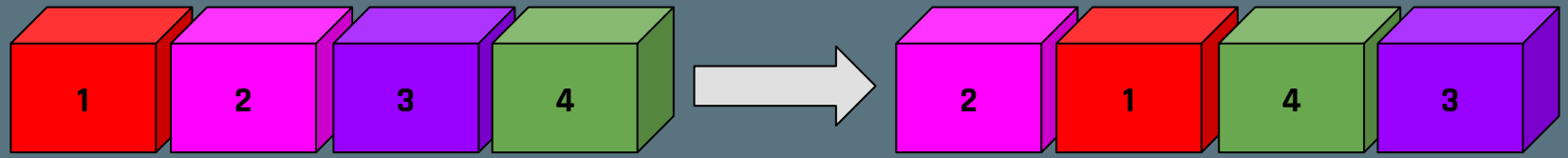
Schema defined in ASN.1

```
ifIndex OBJECT-TYPE
    SYNTAX      InterfaceIndex
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "A unique value, greater than zero, for each interface.  It
            is recommended that values are assigned contiguously
            starting from 1.  The value for each interface sub-layer
            must remain constant at least from one re-initializ
            the entity's network management system
            initialization."
    ::= { ifEntry 1 }
```

Access (e.g., RO, RW) defined per element

Strict table structure defined by indexes

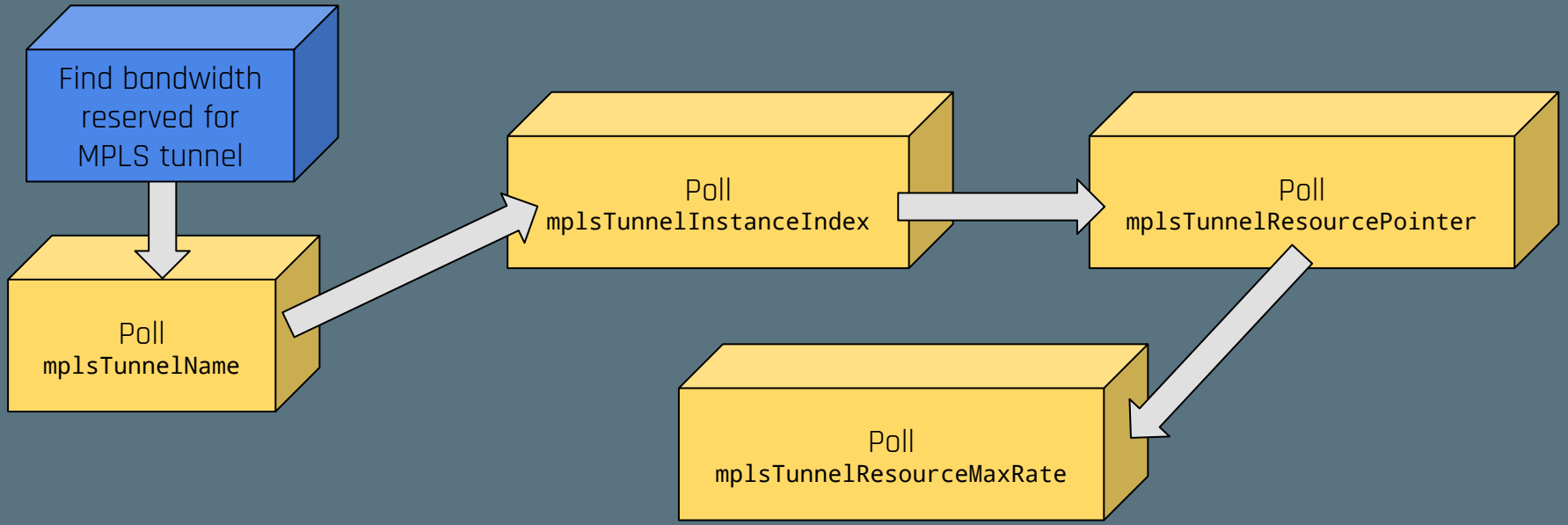
PROTOCOL SUCCESS: STRUCTURE ISSUES FOR DEVICES



Strict data structure defined - re-ordering can be computationally expensive.
Polling a device 'wrong' <u>can</u> result in a loss of other functions (e.g., routing)!

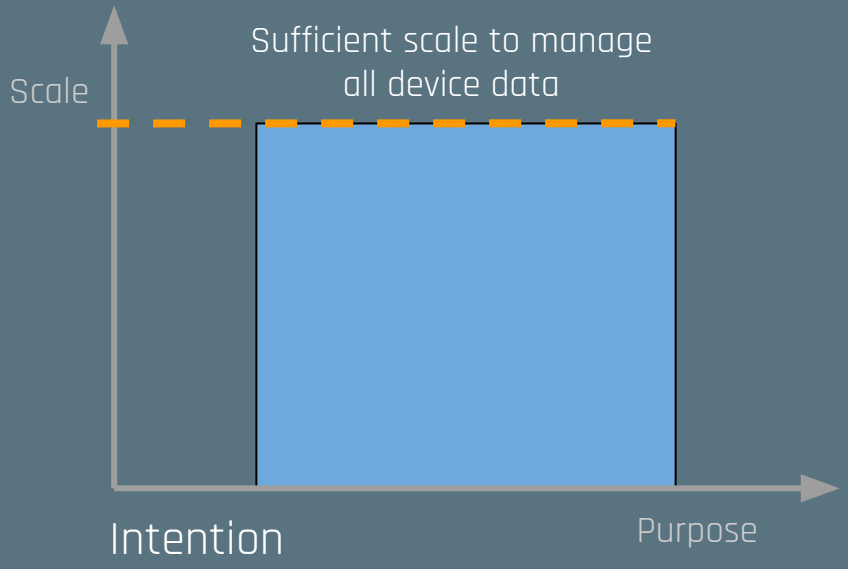
- Gets worse with random access - bulk get vs. get enforced.
- Could have been predicted? Internal systems databases known prior to MIB development?

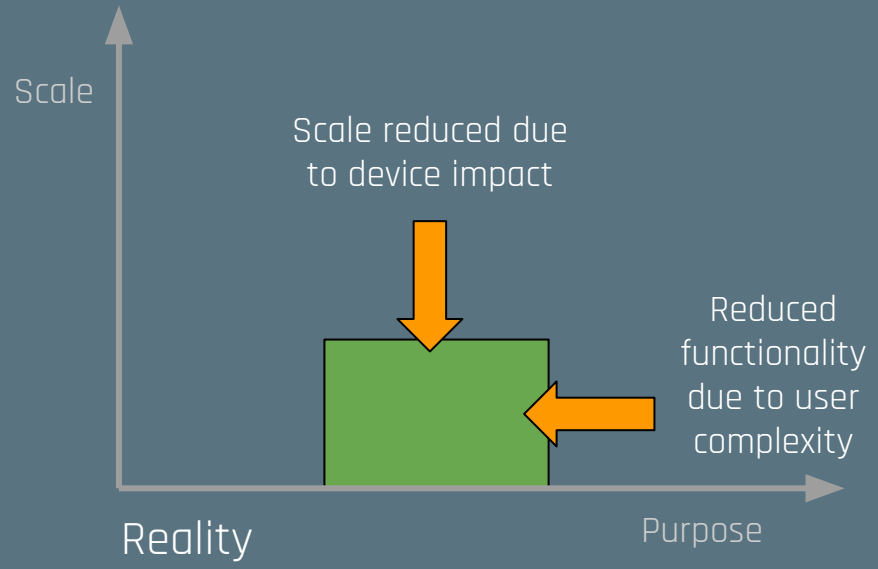
PROTOCOL SUCCESS: STRUCTURE ISSUES FOR OPERATORS



- Data structure not intuitive, and not optimised for operational use cases.
- Competes with `'sh mpls traffic-eng tunnel Tun10 | i Bandwidth Requested'`

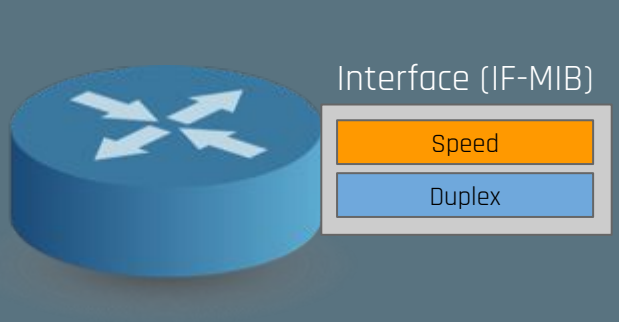
LESSON I:
KNOW YOUR USERS

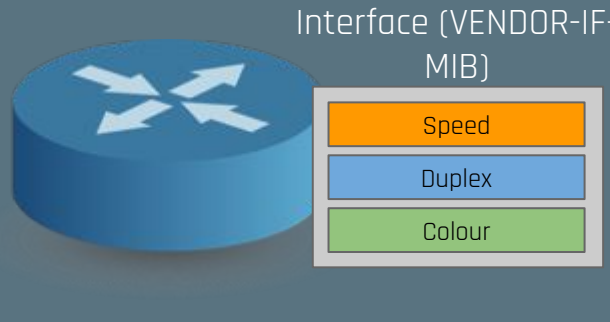




- Two sets of users: device implementors, and operators.
- Neither use case well catered for - drives use of non-standard MIBs or other interfaces.

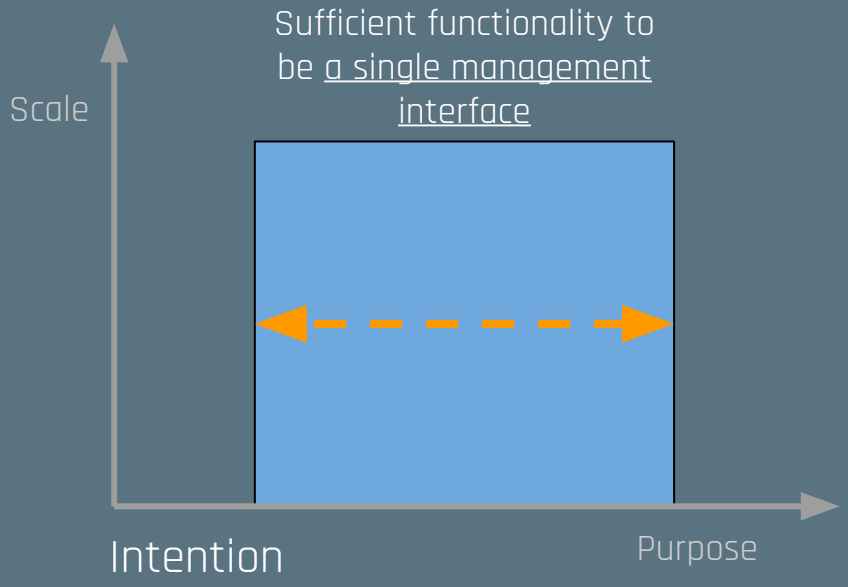
PROTOCOL SUCCESS: THINKING ABOUT USER FLEXIBILITY

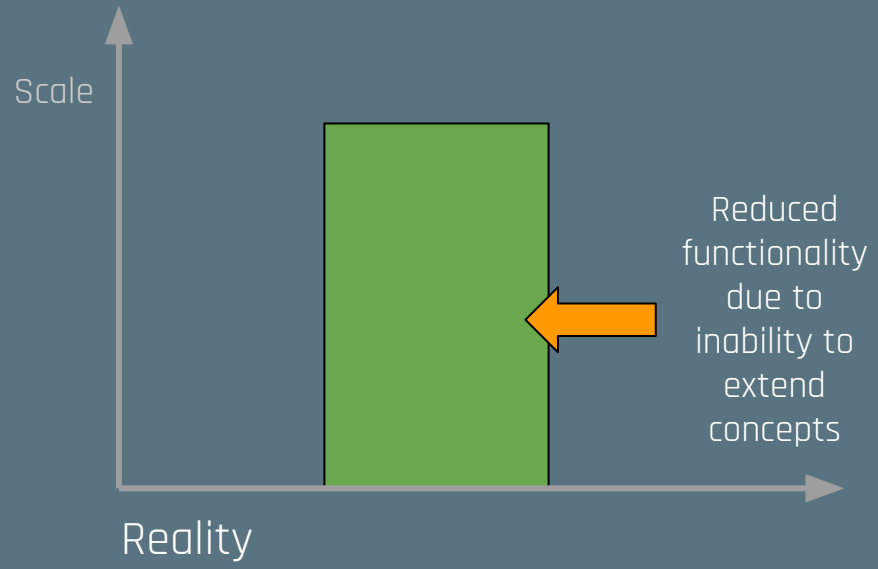




- Structured tables, without ability to add proprietary (or pre-standard) features.
- How should user extend these?
  - Specific 'extension' MIBs?
  - Fork MIBs?
- Both negatively impact usability - split per-vendor, or split logical constructs.

LESSON II: EXTENSIBILITY IS KEY





- Features are always going to evolve - extensibility is a MUST!
- Where alternate interfaces allow more usable interaction - SNMP lost.

HOW DOES SNMP'S RFC 5218 RESULT SHEET LOOK?

Meeting a real need

Improves network manageability

Incremental deployability

Co-exists with other management

Open code availability

Open source daemons and pollers

Open maintenance

Open IETF standards process

Good technical design

Lack of flexibility causes scale issues

Extensibility

Requires opting-out of the standards

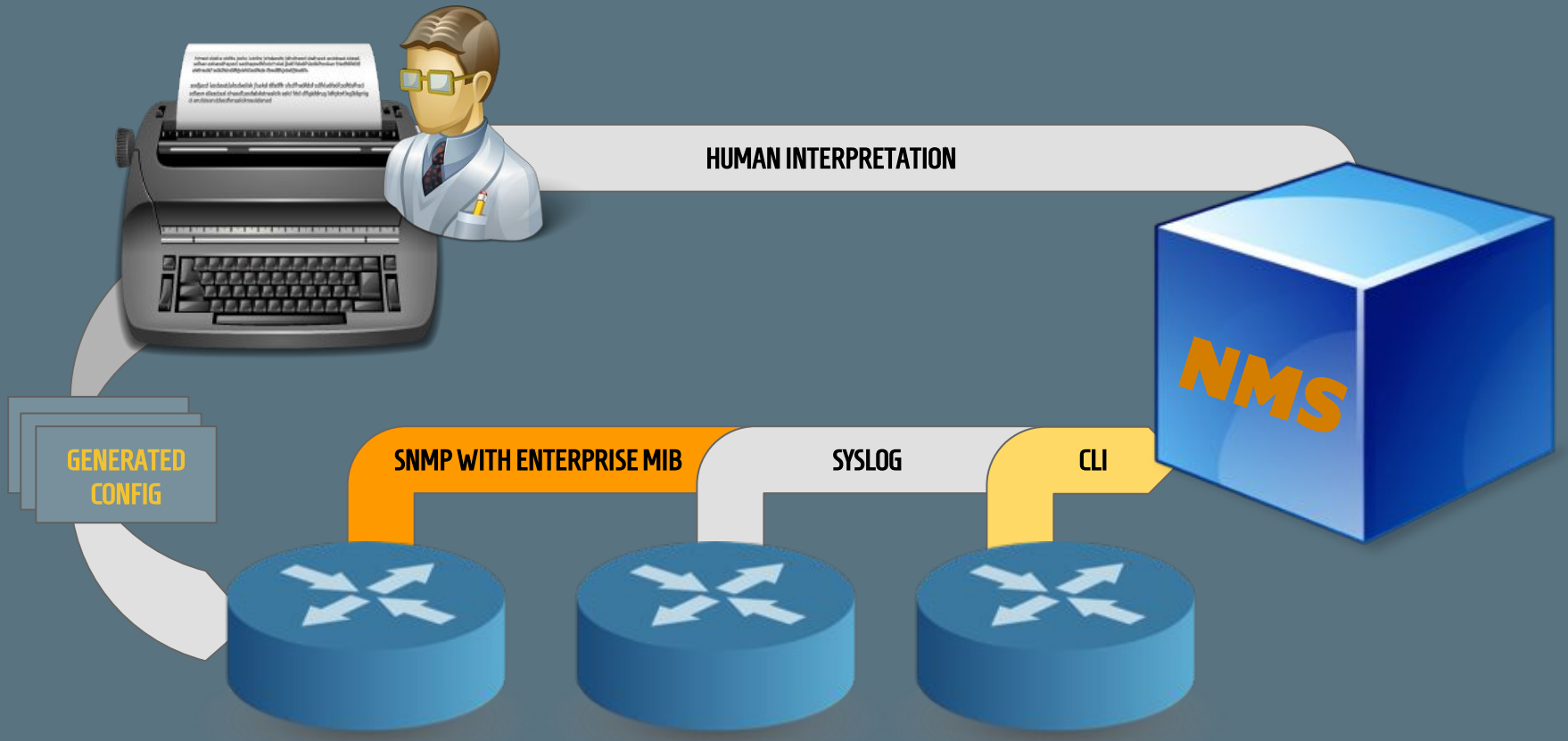
Scalability

Demonstrated break points - disincentive to deploy

Widely deployed - but not wildly successful.

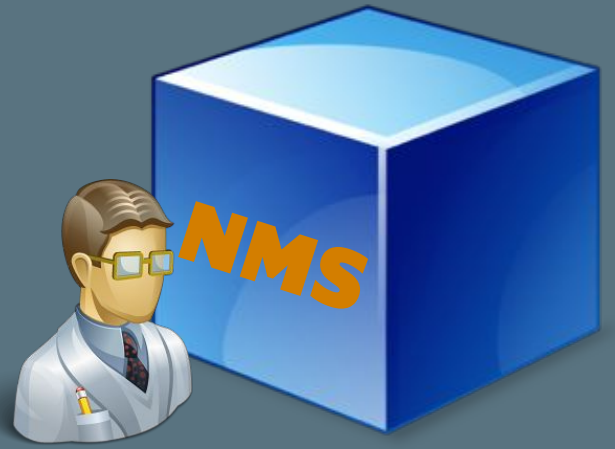
Almost every network operator uses SNMP - a success!
But... it is rarely used alone, and almost never using the 'standard' management modules
Has it survived simply due to lack of competition?

WHERE HAVE WE ENDED UP?



WHAT DOES THE NMS ACTUALLY DO TODAY?

- Usually divided into ‘Monitoring’ and ‘Configuration’ elements.
- Limited processing, and prioritisation of alarms and metrics.
- Config change automation - glues vendor specific CLI together.



- We’re stuck with networks that are human managed.

THE COST OF FAILURE

NMSes typically end up complex, and fragile:

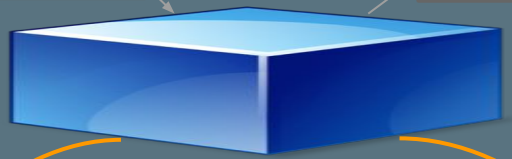
- **Per-vendor integrations:** high cost of change, and many 'specials'.
- **Fragile:** often end up relying on screen-scrapes and non-defined APIs.
- **Imperative**: 'turn-by-turn' understanding of the network required.
- **Esoteric:** deal with the nuts-and-bolts, not the service or application.

WHERE WOULD WE LIKE TO BE?

Programmatic consumers dealing with abstracted services that the network provides

Service changes

Service metrics



Intended state

Metrics

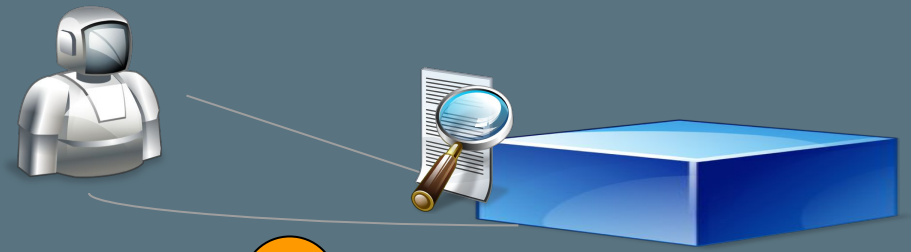
Events

Vendor A

Vendor B

Normalised APIs across vendors for streamed telemetry and declaration of intended state

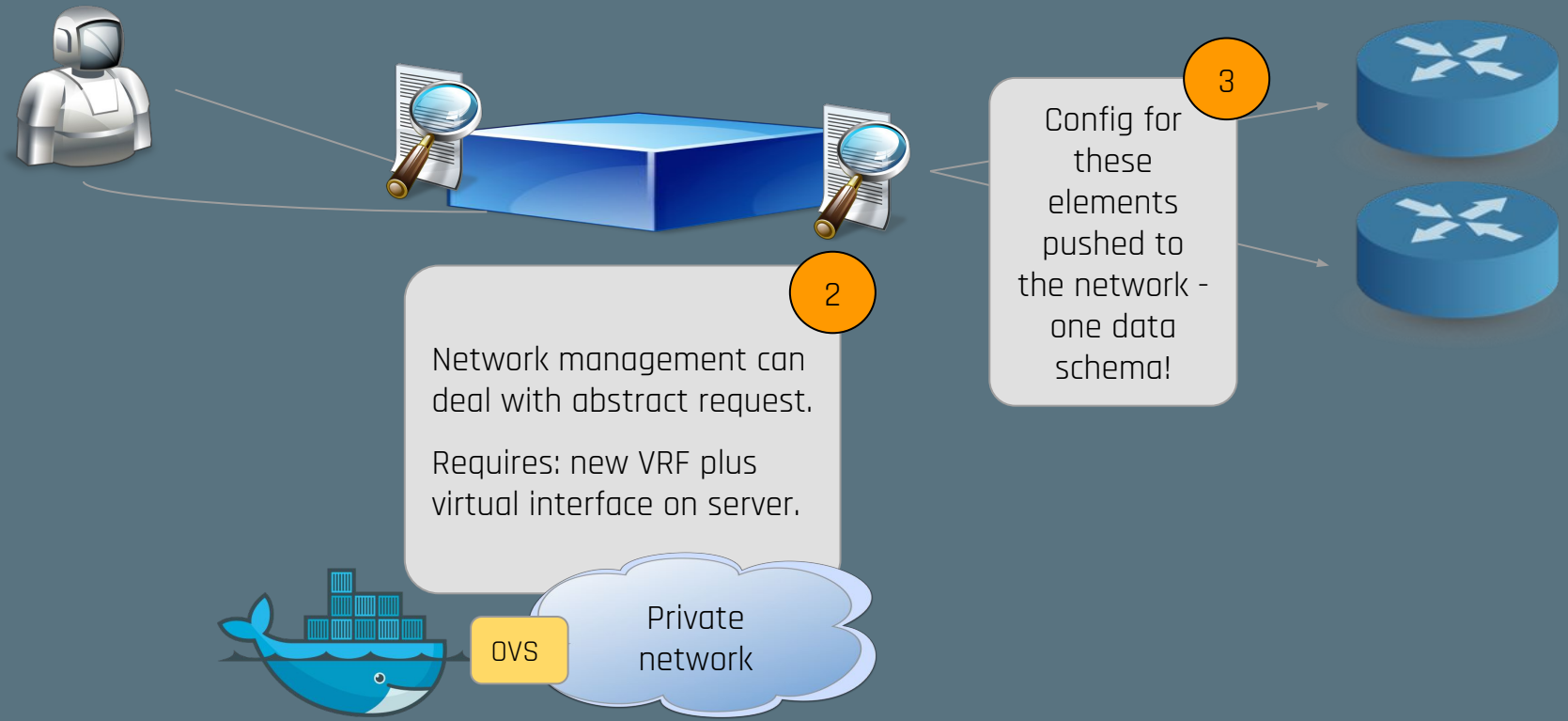
AUTOMATED NETWORK MANAGEMENT: THE TARGET (I)



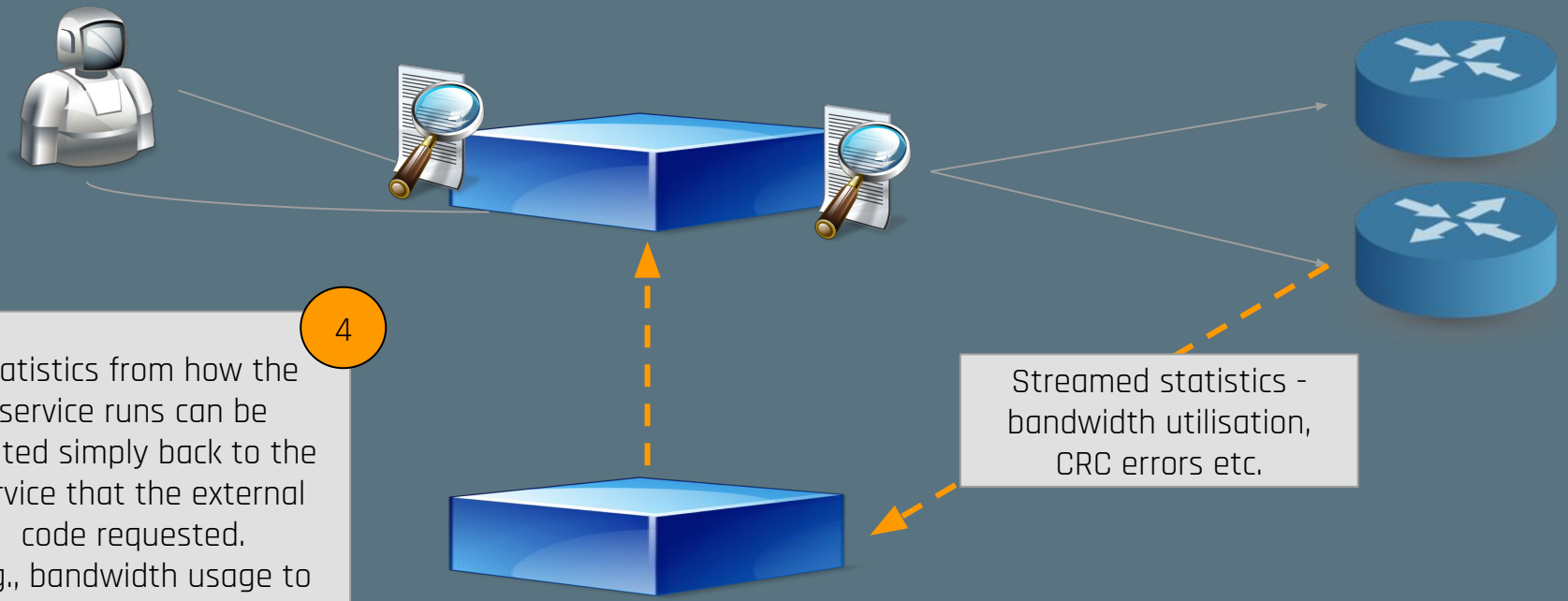
1

External code can request a new service - e.g., network connectivity to a new Docker container that has been turned up.

AUTOMATED NETWORK MANAGEMENT: THE TARGET (II)



AUTOMATED NETWORK MANAGEMENT: THE TARGET (III)

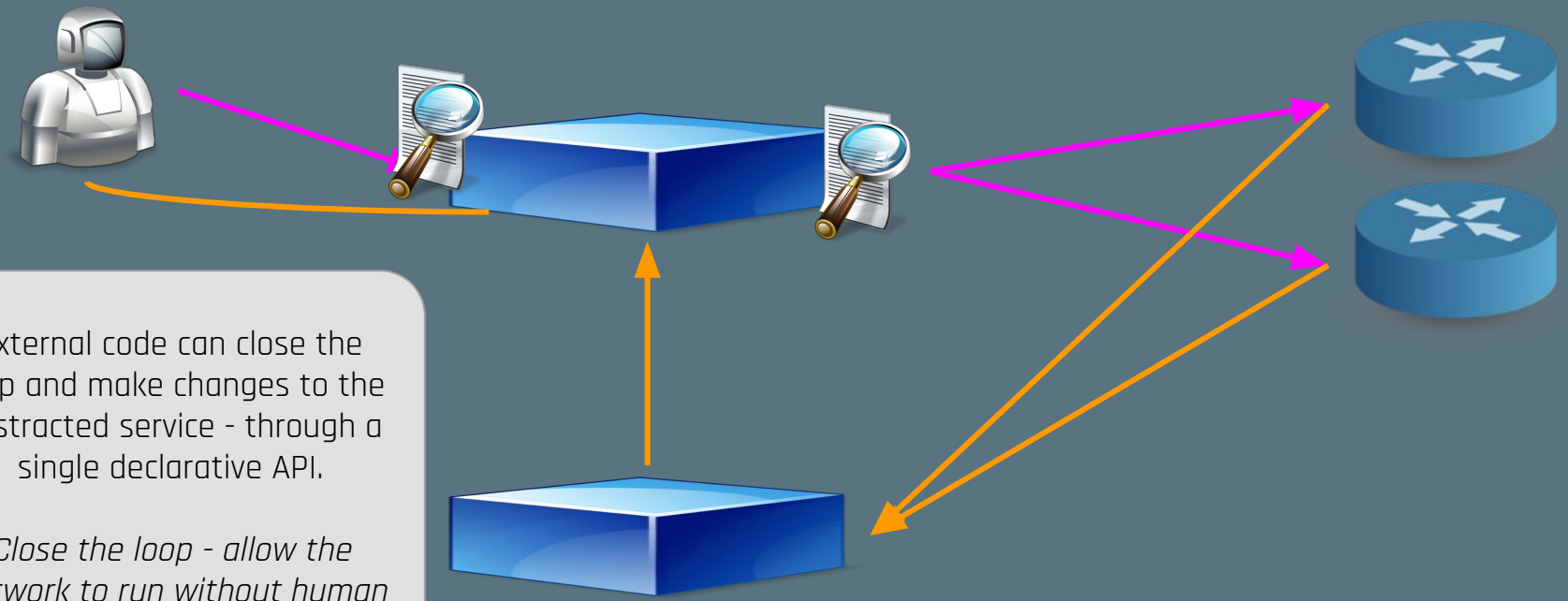


4

Statistics from how the service runs can be related simply back to the service that the external code requested.
(e.g., bandwidth usage to the new container)

Streamed statistics - bandwidth utilisation, CRC errors etc.

AUTOMATED NETWORK MANAGEMENT: THE TARGET (IV)

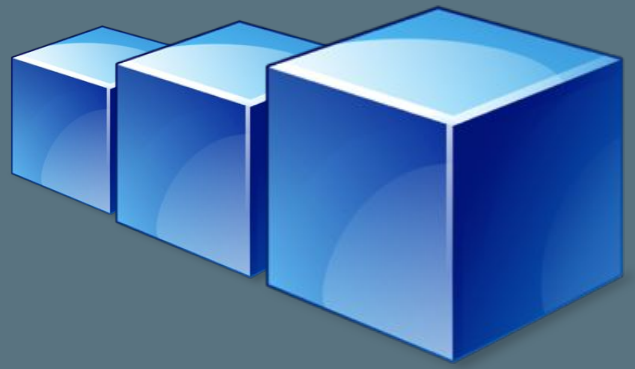


External code can close the loop and make changes to the abstracted service - through a single declarative API.

Close the loop - allow the network to run <u>without</u> human decision makers!

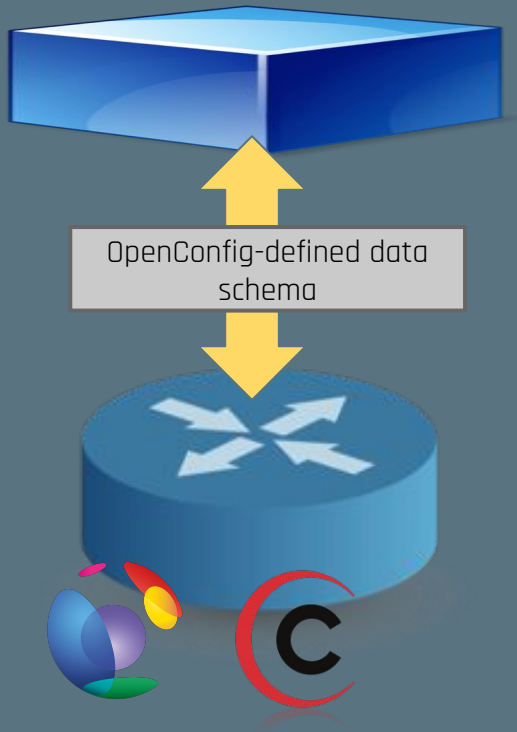
AIMS FOR 'EVOLVED' NETWORK MANAGEMENT

- **Single normalised data model** encompassing configuration and state.
- **Declarative API to change state** - where we want to be, not how to get there.
- Designed to be **interacted with programmatically** - optimise for code, not humans.
- **Real-time streaming API** for events and state updates.



HOW MIGHT WE GET THERE?
OPENCONFIG - START WITH THE USERS.

- Defining data models (schemas) for representing config and state
- Network entities (e.g., interfaces) and applications (e.g., BGP)
- Open source (Apache license)
- Only network operators



YANG:
AN IETF DATA MODELLING LANGUAGE

```
module "example-bgp" {
 ...
 container bgp {
    leaf as-number { type uint32; }
    list neighbors {
       key "neighbor-address";
       leaf neighbor-address { type ip-address; }
       leaf session-state {
         config false;
         type bgpstate; }
    }
}
```

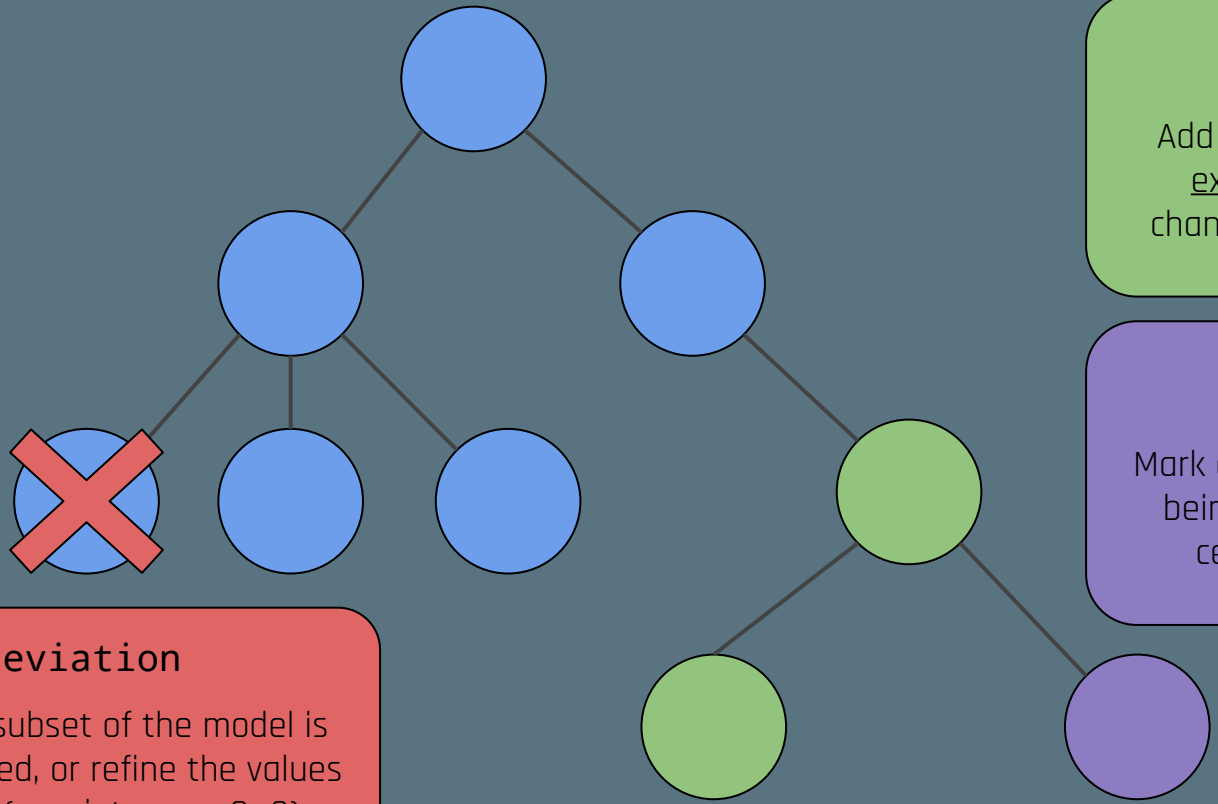
Schema definition language divided into logical modules

Defines a tree structure where 'containers' and 'lists' encapsulate data items

Data items (leaves) are typed, and can contain data such as default values

Used to model both 'configuration' (writeable) and 'state' (read-only) data

YANG: THINKING ABOUT EXTENSIBILITY



`augment`

Add new elements to the existing tree without changing existing content.

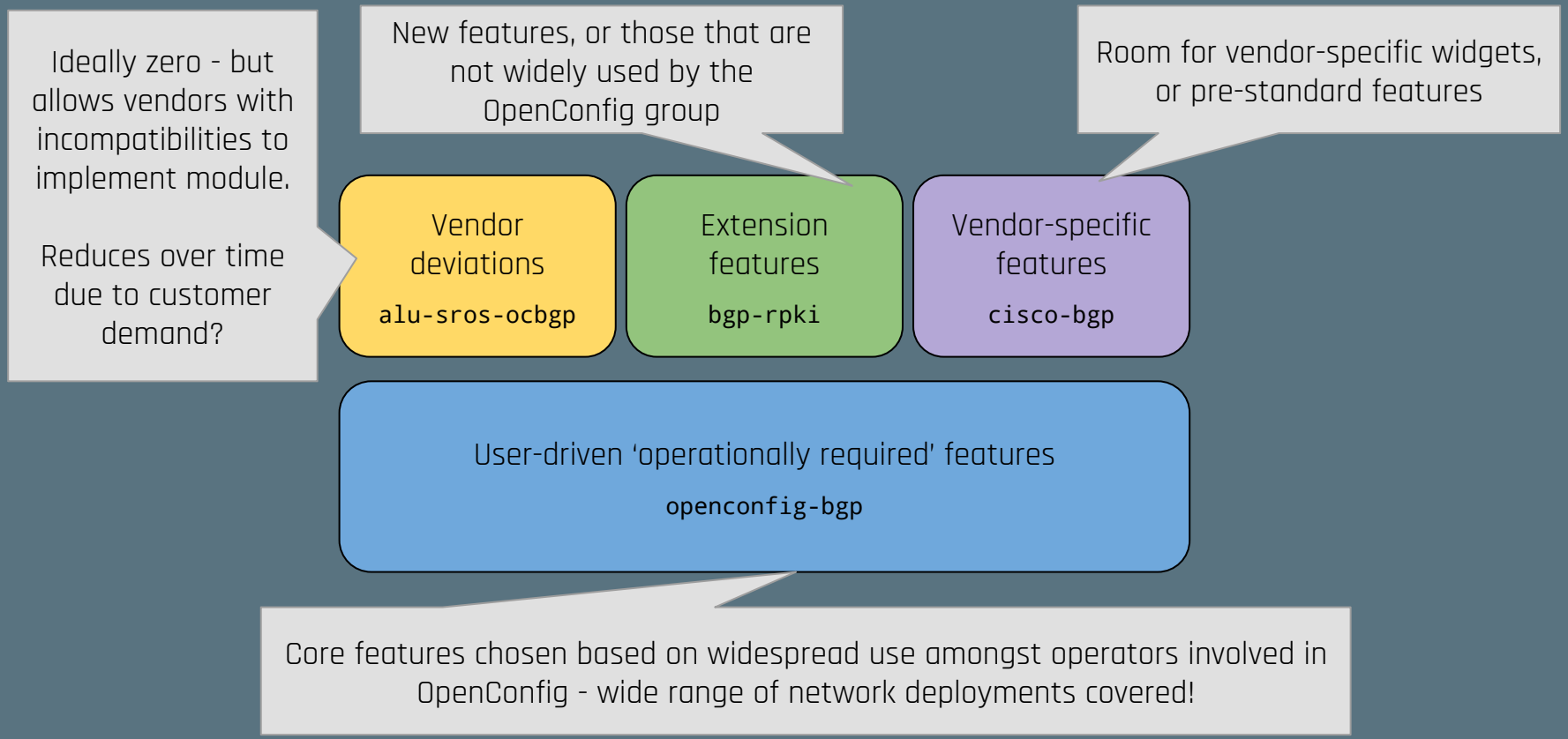
`if-feature`

Mark a subset of the tree as being dependent upon a certain 'feature flag'

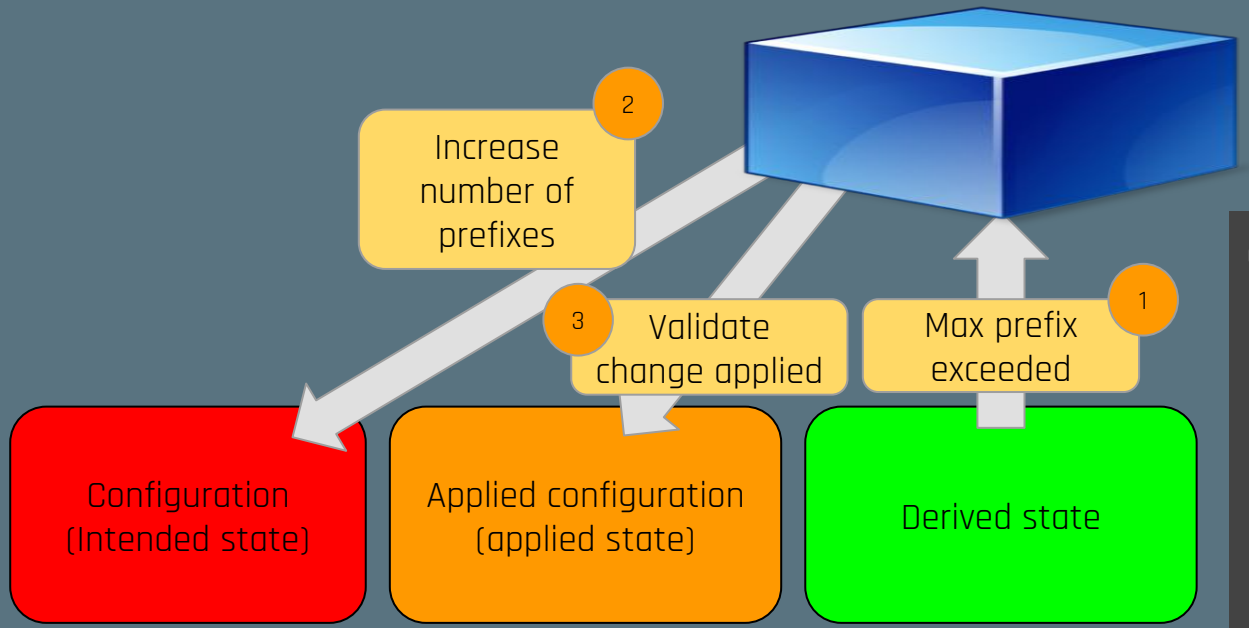
`deviation`

Indicate a subset of the model is not supported, or refine the values specified (e.g., int range 0...3)

OPENCONFIG MODELLING APPROACH



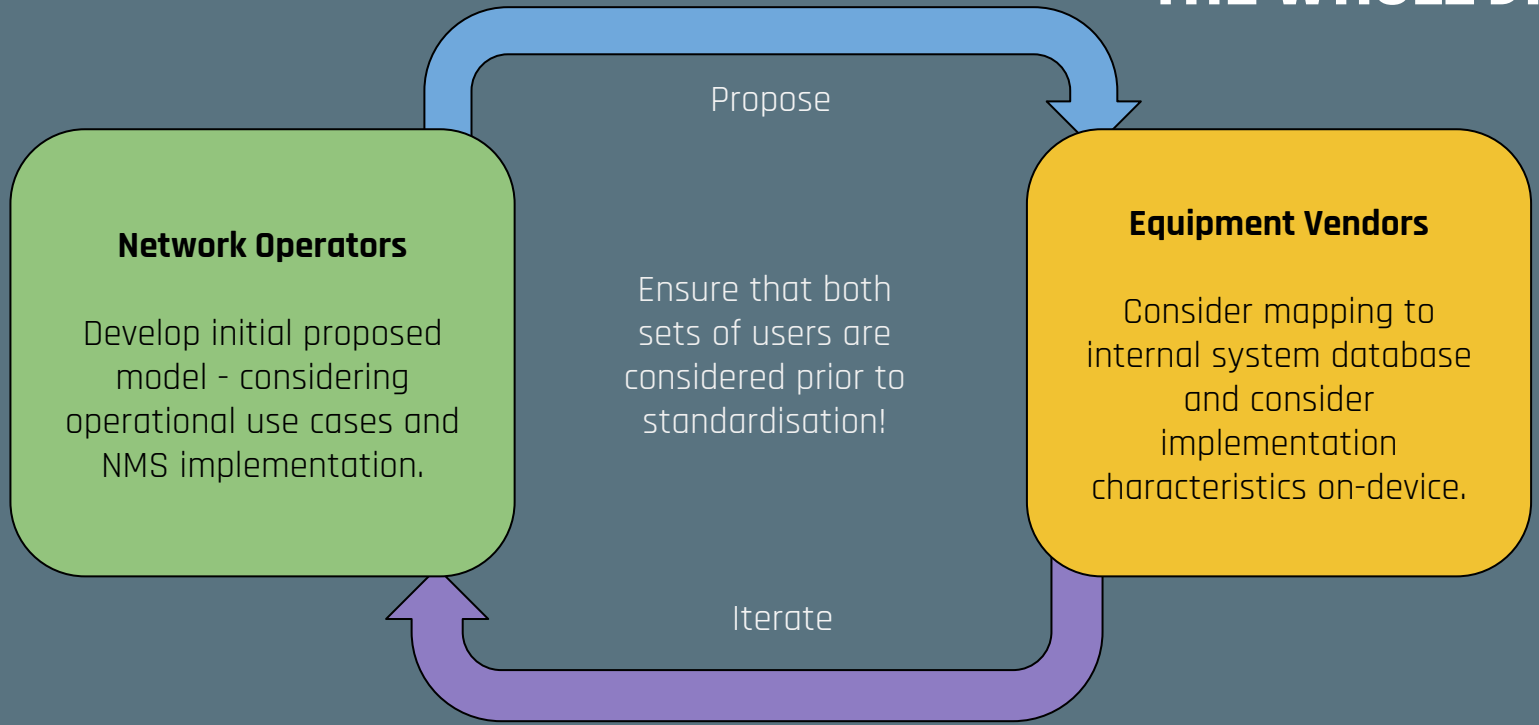
THINKING ABOUT USE CASES: OPERATIONAL STATE




```
container prefix-limit {
        container config {
                leaf max-prefix { ... };
        }
        container state {
                config false;
                leaf max-prefix { ... };
                leaf recvd-prefixes { ... };
        }
}
```

Common workflow: relate the running state of the network to the intended state, and ensure that changes in intention are applied. Models structured to consider such use cases.

OPENCONFIG: THINKING ABOUT THE WHOLE SYSTEM



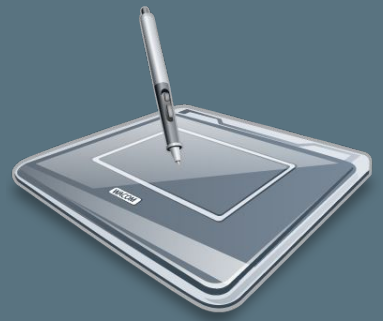
OPEN SOURCE: THINKING ABOUT MAKING THINGS USABLE

Generate language bindings

- Make libraries available that work for developers
- Java, Go, Python...

Programmatically create instance

- Simplify way that these instances can be created - think about users.

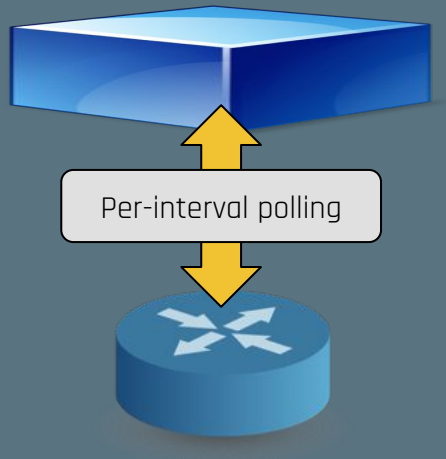


Serialise instance

- Agree on encodings that make sense
- Work to get these implemented!

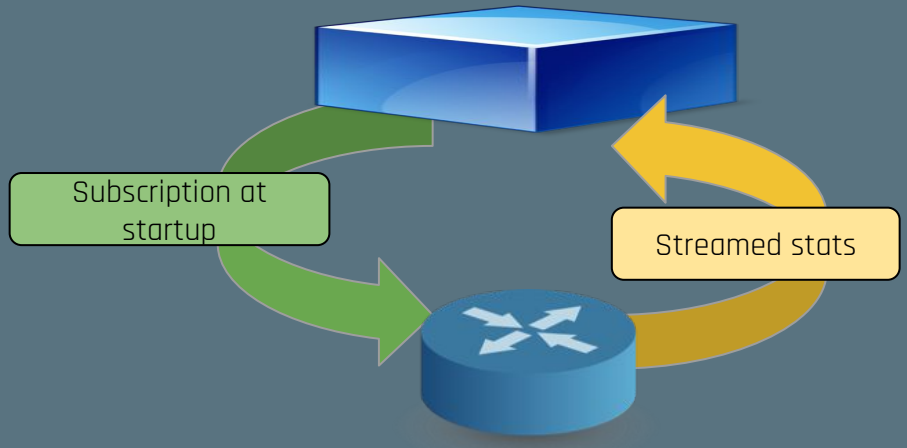
IMPROVING SCALABILITY THROUGH STREAMING TELEMETRY

SNMP Statistics Approach:



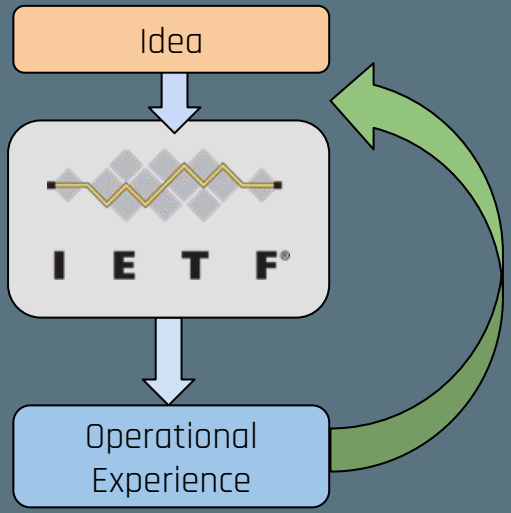
- Expensive whole-table polls
- Filtering required
- Query sent each polling interval

OpenConfig + Streaming Telemetry:



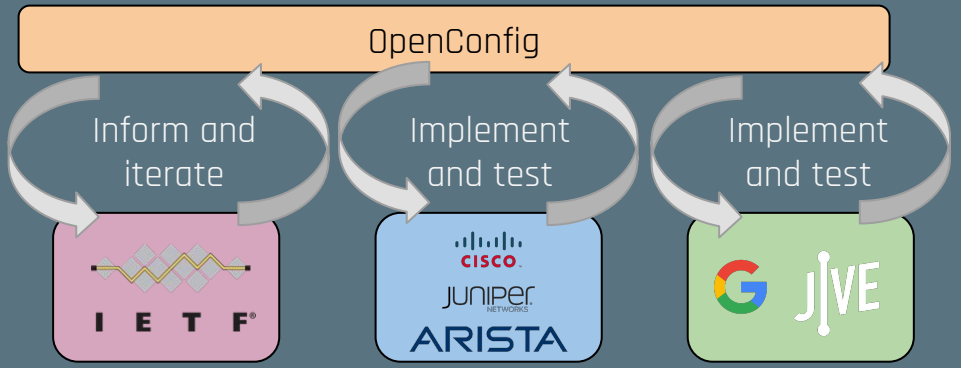
- Adopt publish/subscribe
- Subscriptions on a per-state path
- Events directly relatable to configured entities

CAN WE ITERATE BEFORE WE CARVE OUT THE STANDARDS?



1-3+ years!

- Long time to standardise - without wide ops experience
- Nothing is perfect first time - long iteration cycle.



- Improve the chances of success by testing in the real-world before standardisation
- Intention is to quickly capture issues preventing 'basic' success!
- Cisco, Juniper, Arista, Google, Jive already implementing!

COMPARING OC/YANG TO SNMP - RFC 5218

Meeting a real need

Improves network manageability

Incremental deployability

Co-exists with other management

Open code availability

Specific efforts to ensure tooling is available

Open maintenance

Open IETF standards process

Good technical design

Address the flexibility issues of SNMP

Extensibility

Standard modules designed to be extended

Scalability

Being tested through implementations early and often

Open licensing

Zero patents, no IPR claims, Apache licensed

Open questions

- Tooling that grows around the OpenConfig effort - what do operators that don't write code do?
- Feature diversity - what happens to those that have huge feature diversity - particularly does the approach continue to work for Enterprise?

By no means a "done deal" - but significant focus on improving what's there today!

CONCLUDING THOUGHTS...

New technologies/protocols will always have an adoption curve...

Some organisations have a huge cost of change from their existing systems - should not prevent us "trying something better".

Hype vs. demonstrated needs...

(Hopefully) Have not mentioned 'SDN' - proposal here is to adopt concepts from other infrastructure management into networks - with real benefits.

It's easy to see problems in the rear view mirror...

OpenConfig/YANG will not be perfect first time - there is a real need for networks to be treated like software - if we don't get it right first time, bump version and refactor!

THANKS! QUESTIONS?

E-Mail: rjs@rob.sh or rjs@jive.com
Twitter: @robshakir

ADDITIONAL RESOURCES

- OpenConfig: www.openconfig.net
- OpenConfig Models: https://github.com/openconfig/public
- Go bindings: https://github.com/openconfig/goyang
- Python bindings: https://github.com/robshakir/pyangbind
- Google on OpenConfig: https://www.youtube.com/watch?v=_XBwRydxj1M
- Tail-f YANG Tutorials: http://www.tail-f.com/education/
- OpenConfig on Juniper: https://vimeo.com/139447948
- OpenConfig on Cisco: https://vimeo.com/146123659
- OpenConfig on Arista: https://eos.arista.com/openconfig-the-emerging-industry-standard-api-for-network-elements/
- Network Field Day OpenConfig round-table: http://techfieldday.com/appearance/anees-shaikh-presents-open-network-management/