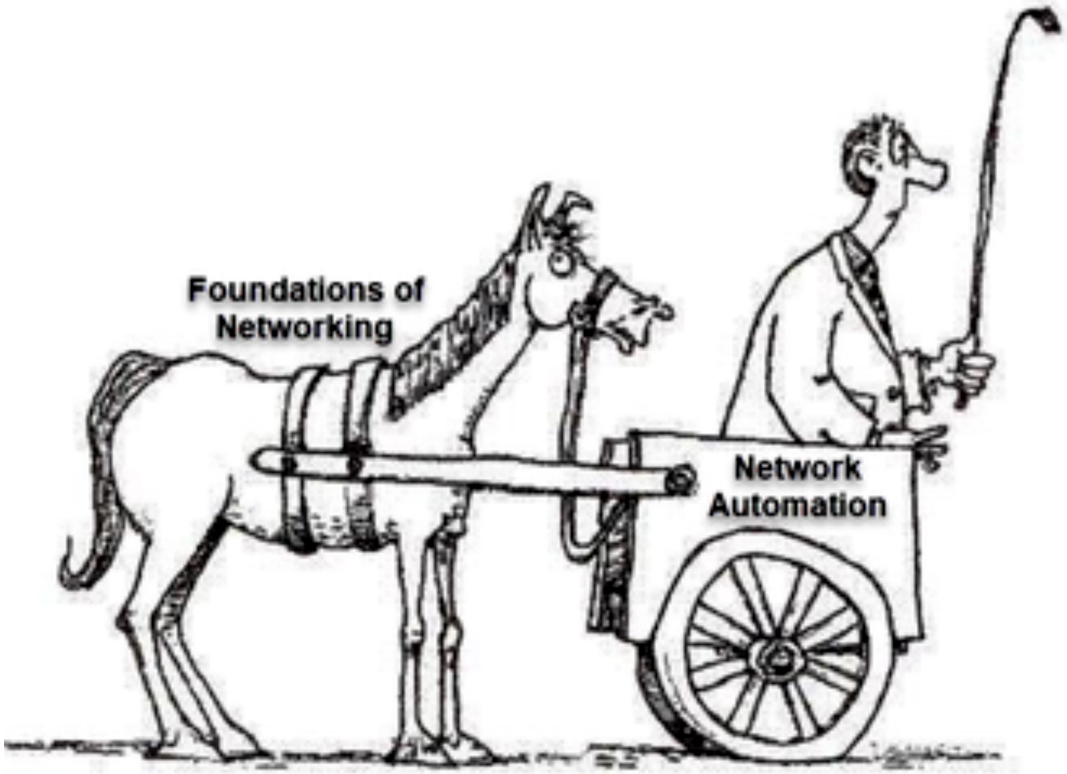
apstra

Intent Based Networking - the technology

NANOG™

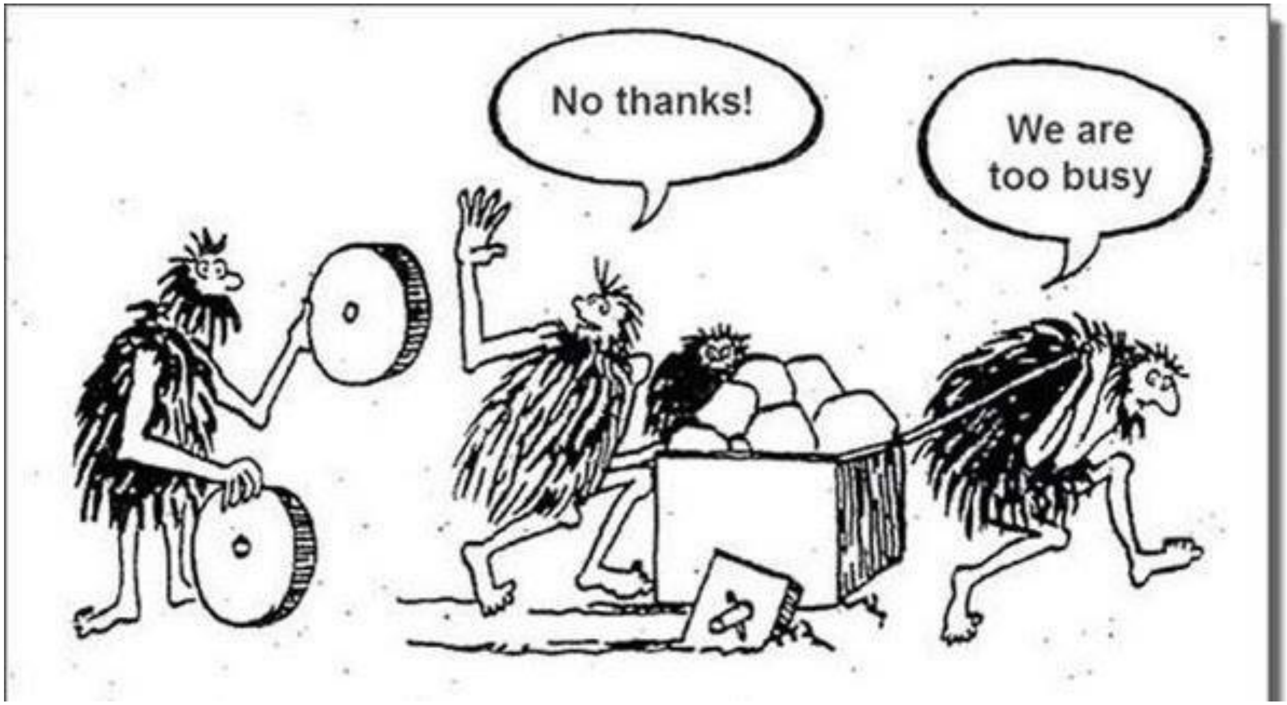
Jeff Tantsura
Head of Networking Strategy @Apstra
Chair IETF Routing and RIFT working groups, IAB

Why IBN?



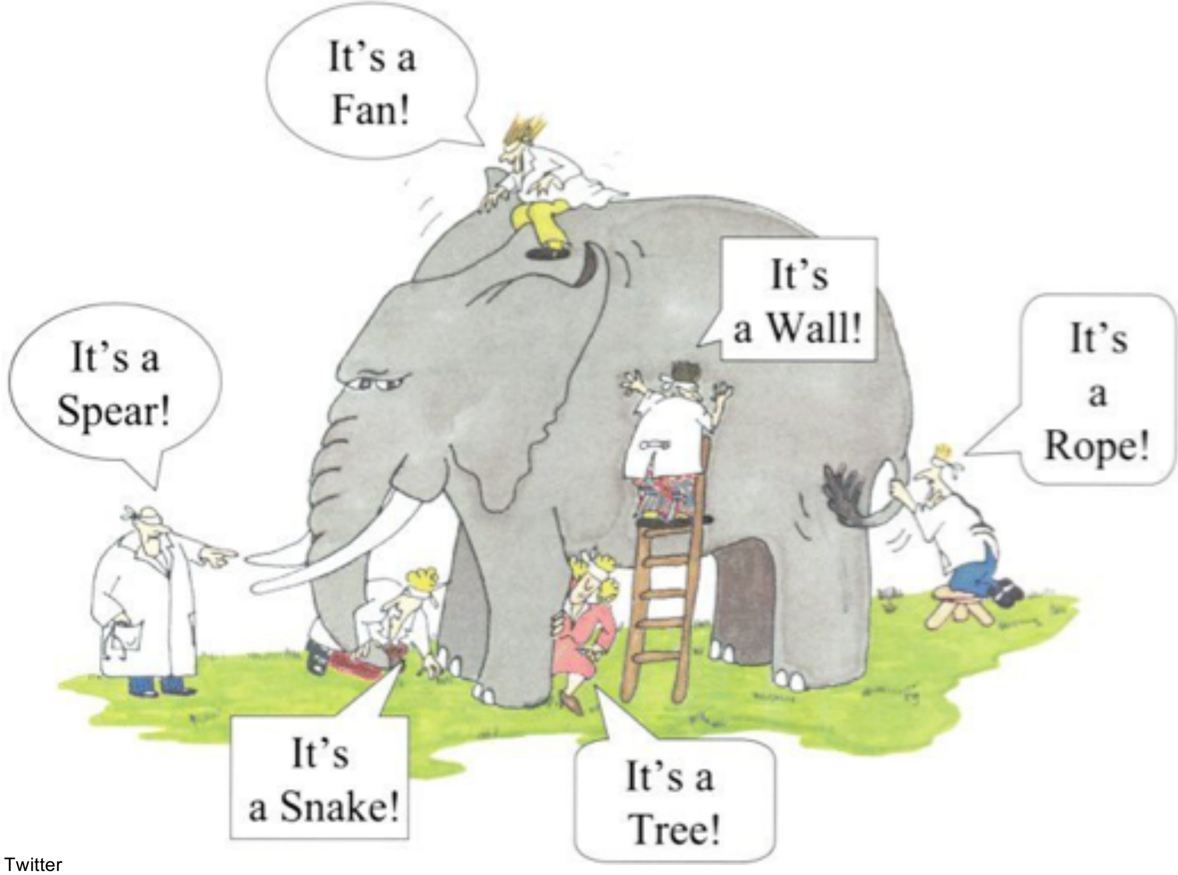
Source: Twitter

Why IBN?



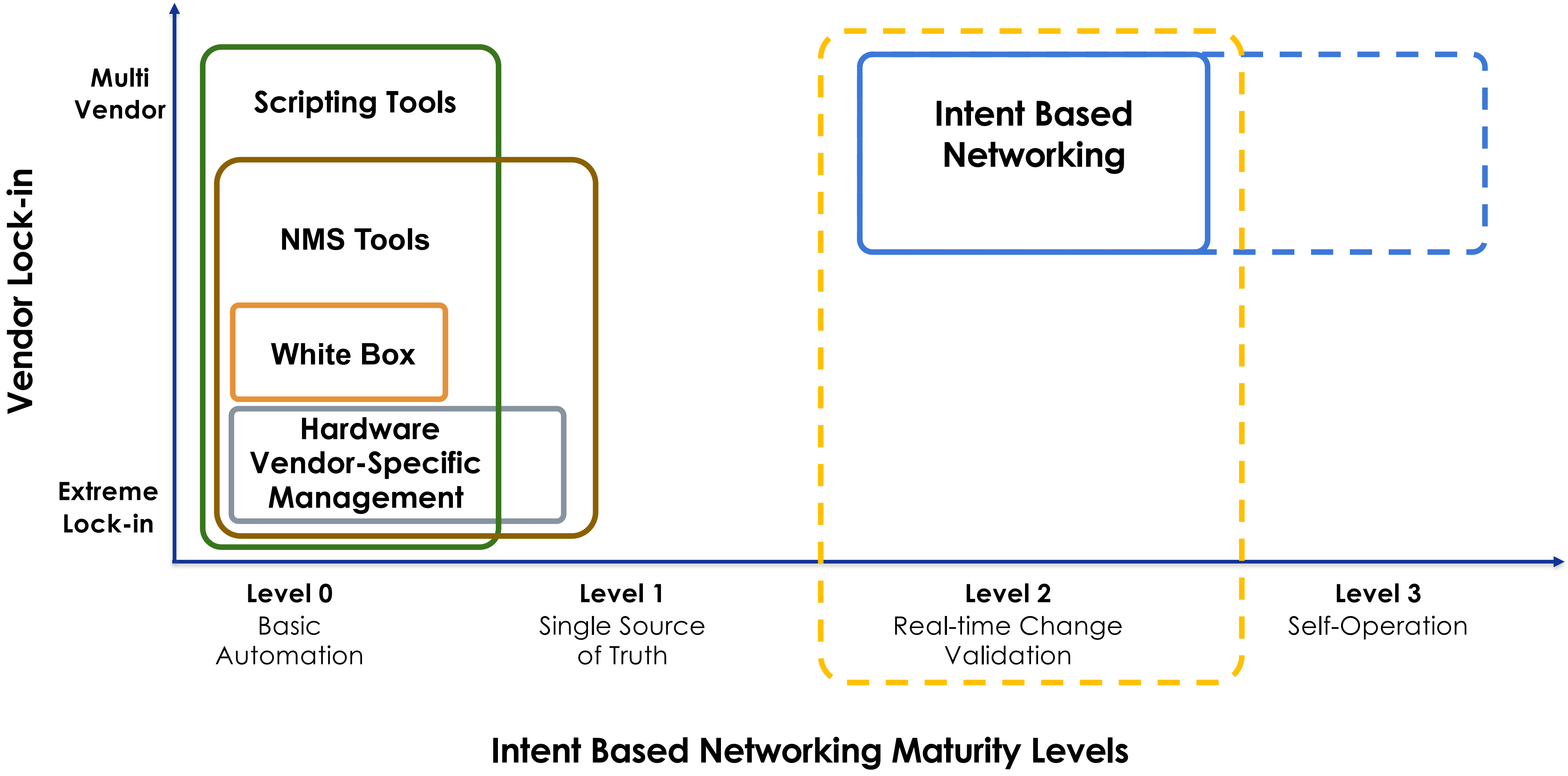
Source: Twitter

IBN Landscape

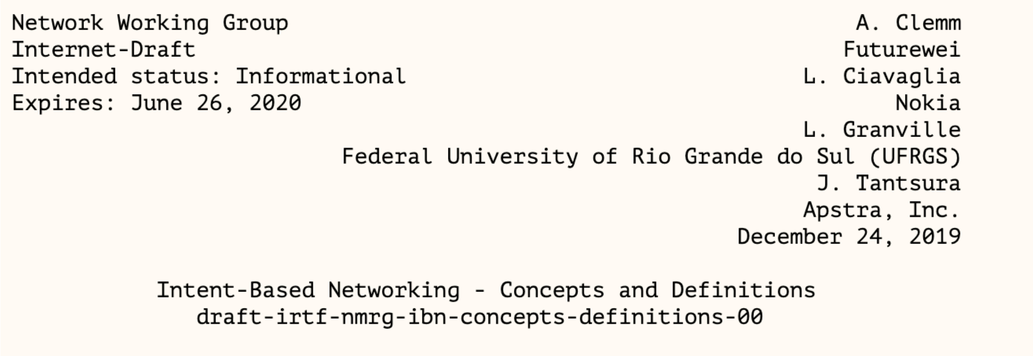


Source: Twitter

IBN Landscape



IBN standardization – just the beginning



```
Network Working Group                                          A. Clemm
Internet-Draft                                                Futurewei
Intended status: Informational                              L. Ciavaglia
Expires: June 26, 2020                                            Nokia
                                                            L. Granville
                        Federal University of Rio Grande do Sul (UFRGS)
                                                             J. Tantsura
                                                            Apstra, Inc.
                                                       December 24, 2019

           Intent-Based Networking - Concepts and Definitions
              draft-irtf-nmrg-ibn-concepts-definitions-00
```

← Has recently been adopted as the rg document

Outlines number of fundamental IBNS building blocks and their relationship:

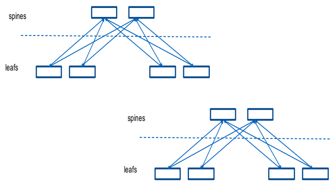
- SSoT: Single Source of Truth - A functional block in an IBN system that normalizes users intent and serves as the single source of data (normalized intended state) for every consumer.
- IBA: Intent Based Analytics - Analytics that are defined and derived from user' intent and used to validate the intended state.
- PDP: Policy Decision Point – part of intent definition, technology agnostic.
- PEP: Policy Enforcement Point – technology/device aware (e.g ACL or FW rule).

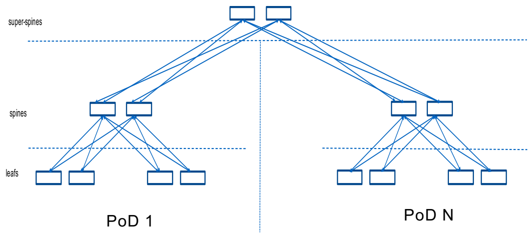
IBN – why DC is a good starting point?

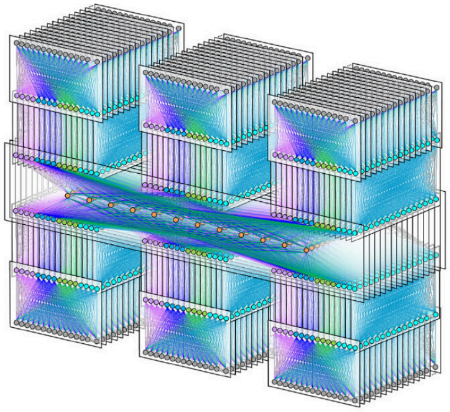
Ability to reason about Intent is a fundamental property of an IBNS!

Complex systems fail in mysterious ways ;-)

CLOS topologies are extremely regular/uniform and mutate (expand) in a very predictable way. PoD structure provides clear boundaries.





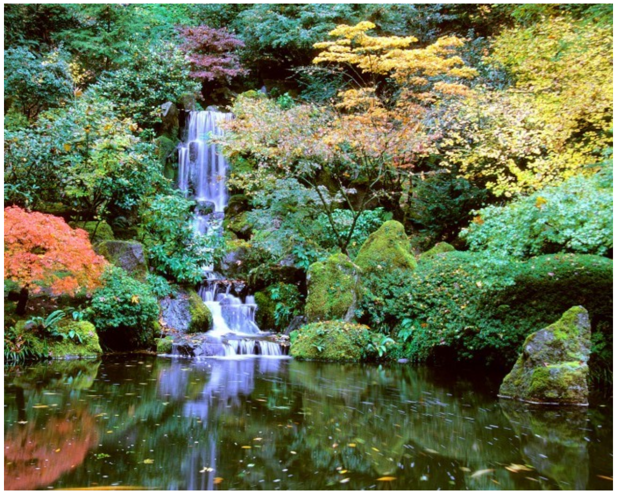


https://engineering.fb.com/data-center-engineering/f16-minipack/

IBN – why DC is a good starting point?

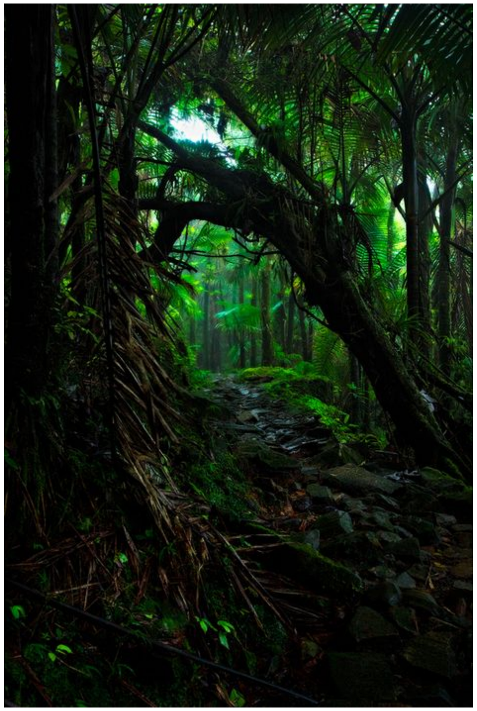
Telco WAN evolution

Day 0



https://www.travelportland.com/article/portland-japanese-garden/

Day 2



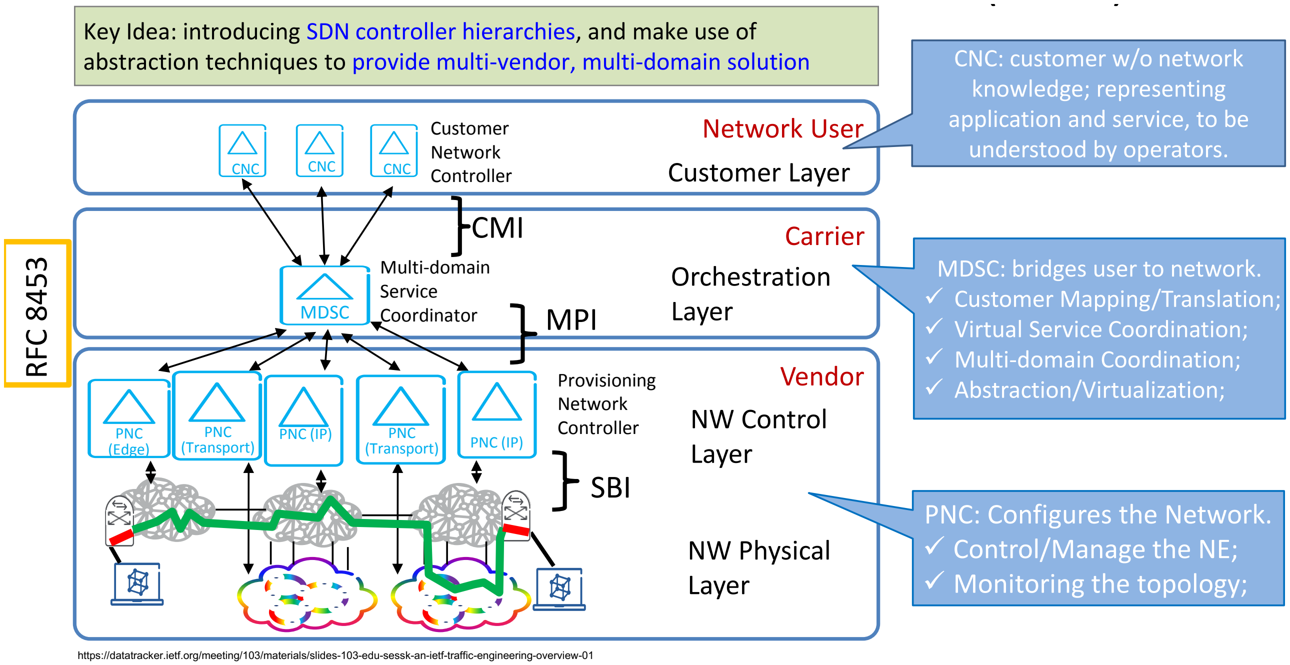
https://www.pinterest.com/pin/300756081335600951

IBN – why DC is a good starting point?

apstra

IBN in Telco WAN – is there a hope?

IETF TEAS ACTN framework is a step in the right direction



Key Idea: introducing SDN controller hierarchies, and make use of abstraction techniques to provide multi-vendor, multi-domain solution

CNC: customer w/o network knowledge; representing application and service, to be understood by operators.

MDSC: bridges user to network.
- ✓ Customer Mapping/Translation;
- ✓ Virtual Service Coordination;
- ✓ Multi-domain Coordination;
- ✓ Abstraction/Virtualization;

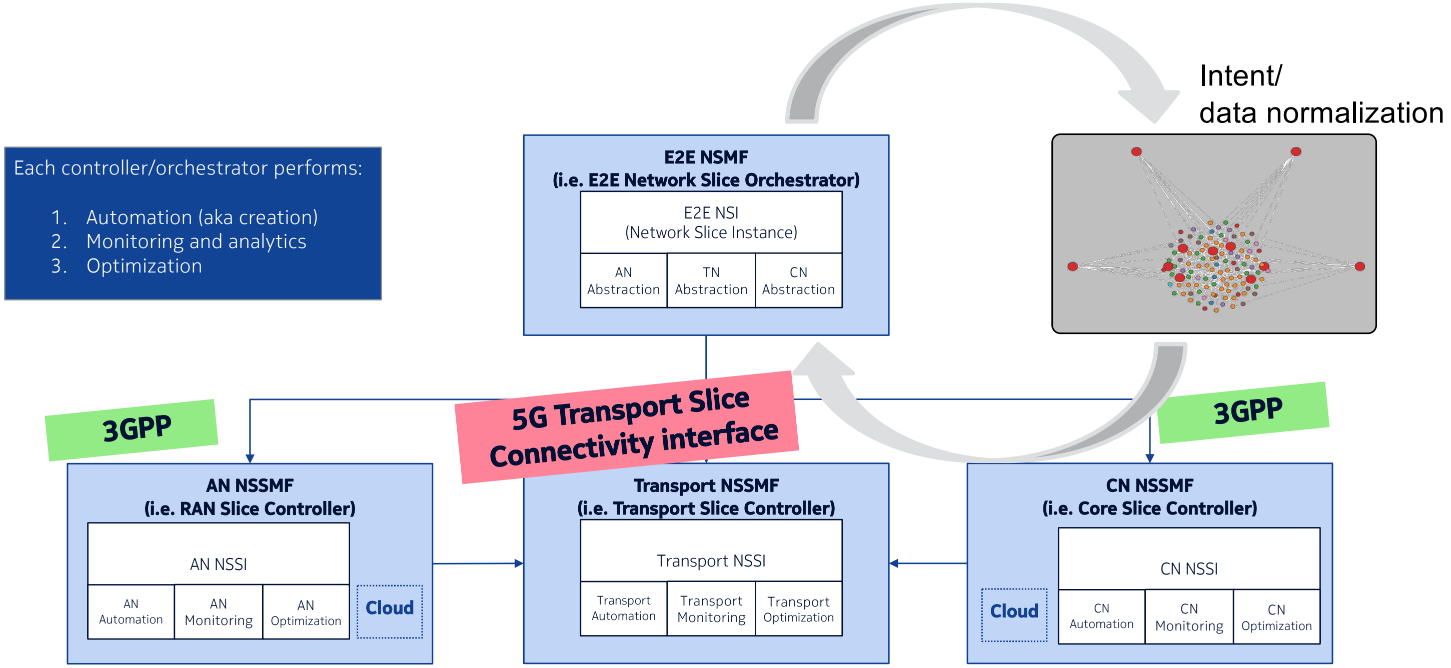
PNC: Configures the Network.
- ✓ Control/Manage the NE;
- ✓ Monitoring the topology;

https://datatracker.ietf.org/meeting/103/materials/slides-103-edu-sessk-an-ietf-traffic-engineering-overview-01

IBN – why DC is a good starting point?

IBN in Telco WAN – is there a hope? What about 5G?

E2E Network Slicing is too complex, transport part of it is a perfect candidate for IBN



Each controller/orchestrator performs:

1. Automation (aka creation)
2. Monitoring and analytics
3. Optimization

Intent/
data normalization

E2E NSMF
(i.e. E2E Network Slice Orchestrator)

E2E NSI
(Network Slice Instance)

AN Abstraction | TN Abstraction | CN Abstraction

5G Transport Slice Connectivity interface

3GPP

AN NSSMF
(i.e. RAN Slice Controller)

AN NSSI

AN Automation | AN Monitoring | AN Optimization

Cloud

Transport NSSMF
(i.e. Transport Slice Controller)

Transport NSSI

Transport Automation | Transport Monitoring | Transport Optimization

3GPP

CN NSSMF
(i.e. Core Slice Controller)

CN NSSI

CN Automation | CN Monitoring | CN Optimization

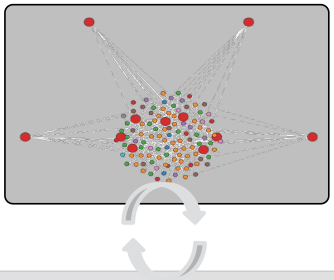
Cloud

https://datatracker.ietf.org/meeting/105/materials/slides-105-teas-sessa-03-5g-transport-slice-connectivity-interface-00

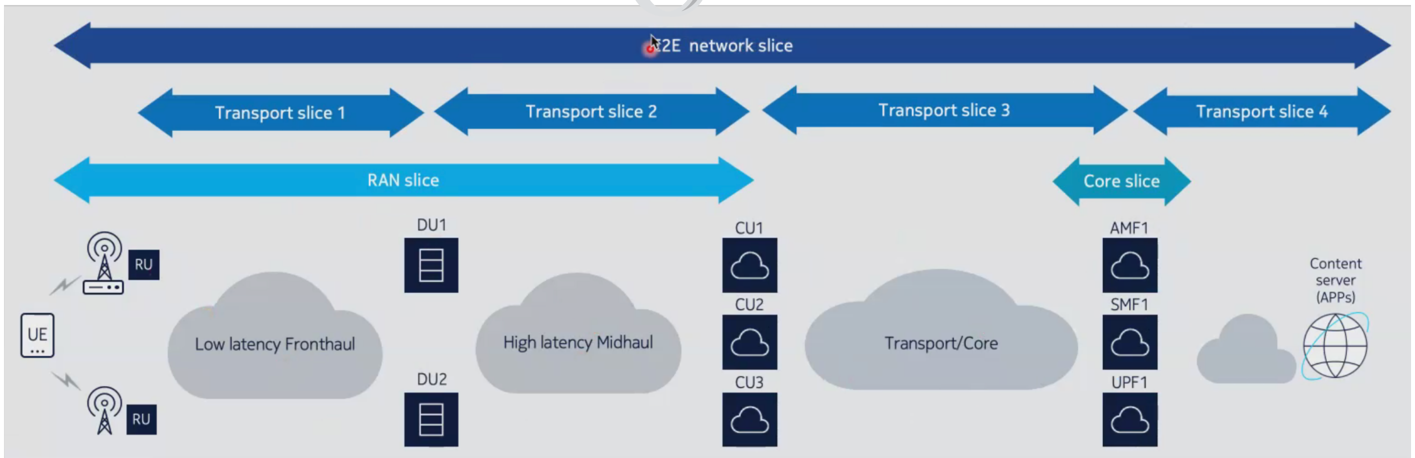
IBN – why DC is a good starting point?

IBN in Telco WAN – 5G NS

E2E Network Slicing is too complex, transport part of it is a perfect candidate for IBN



Intent/
data normalization

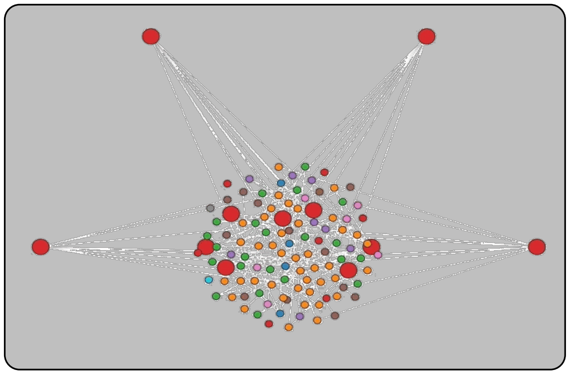


IBN Design Philosophy

Networks managed as a whole system, not individual components

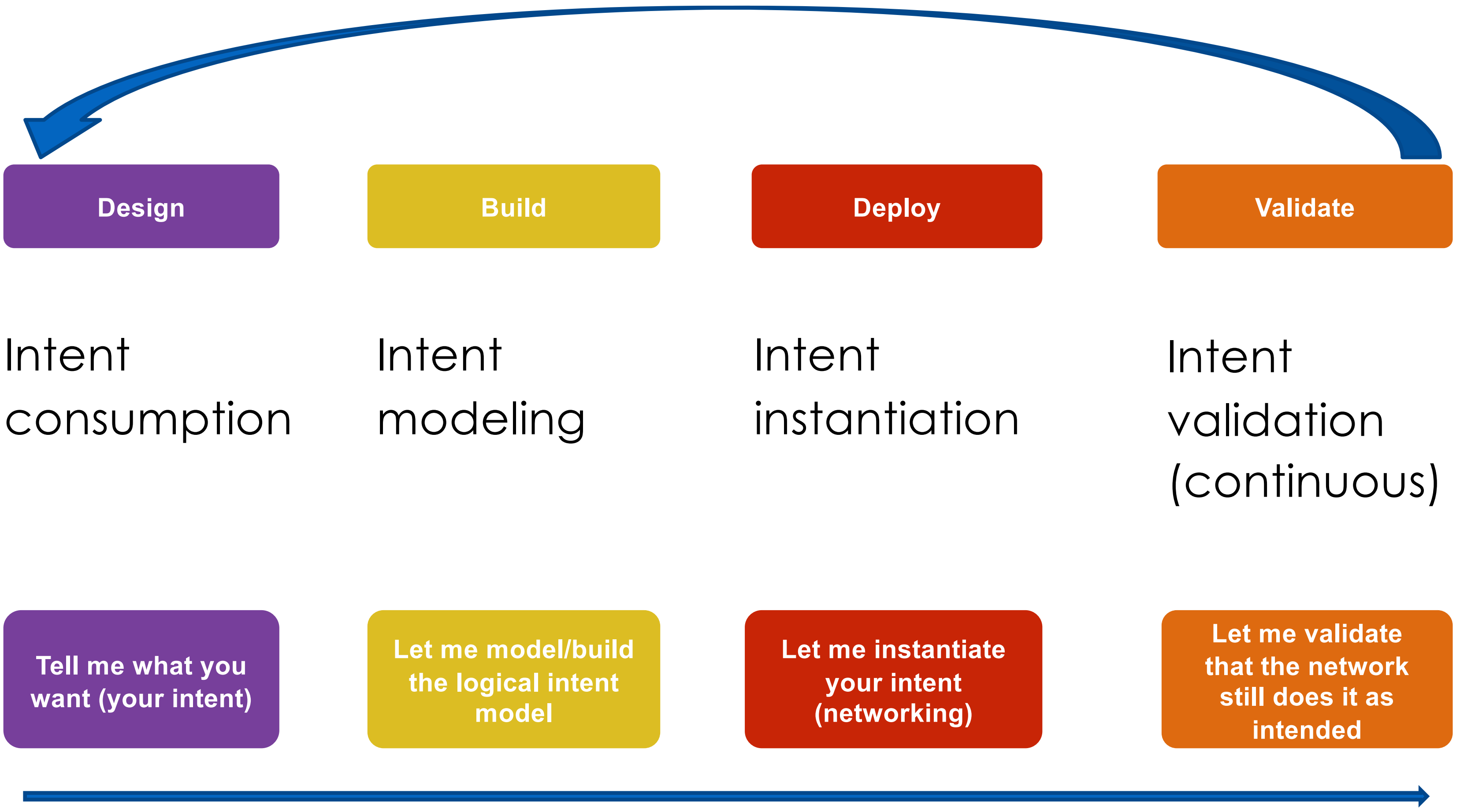
Successful networks are defined by the outcomes produced by the whole system

Intent Based Networking is about "*what*" not "*how*"



More details in: draft-irtf-nmrg-ibn-concepts-definitions

IBN life cycle



Design

Intent consumption

Tell me what you want (your intent)

Build

Intent modeling

Let me model/build the logical intent model

Deploy

Intent instantiation

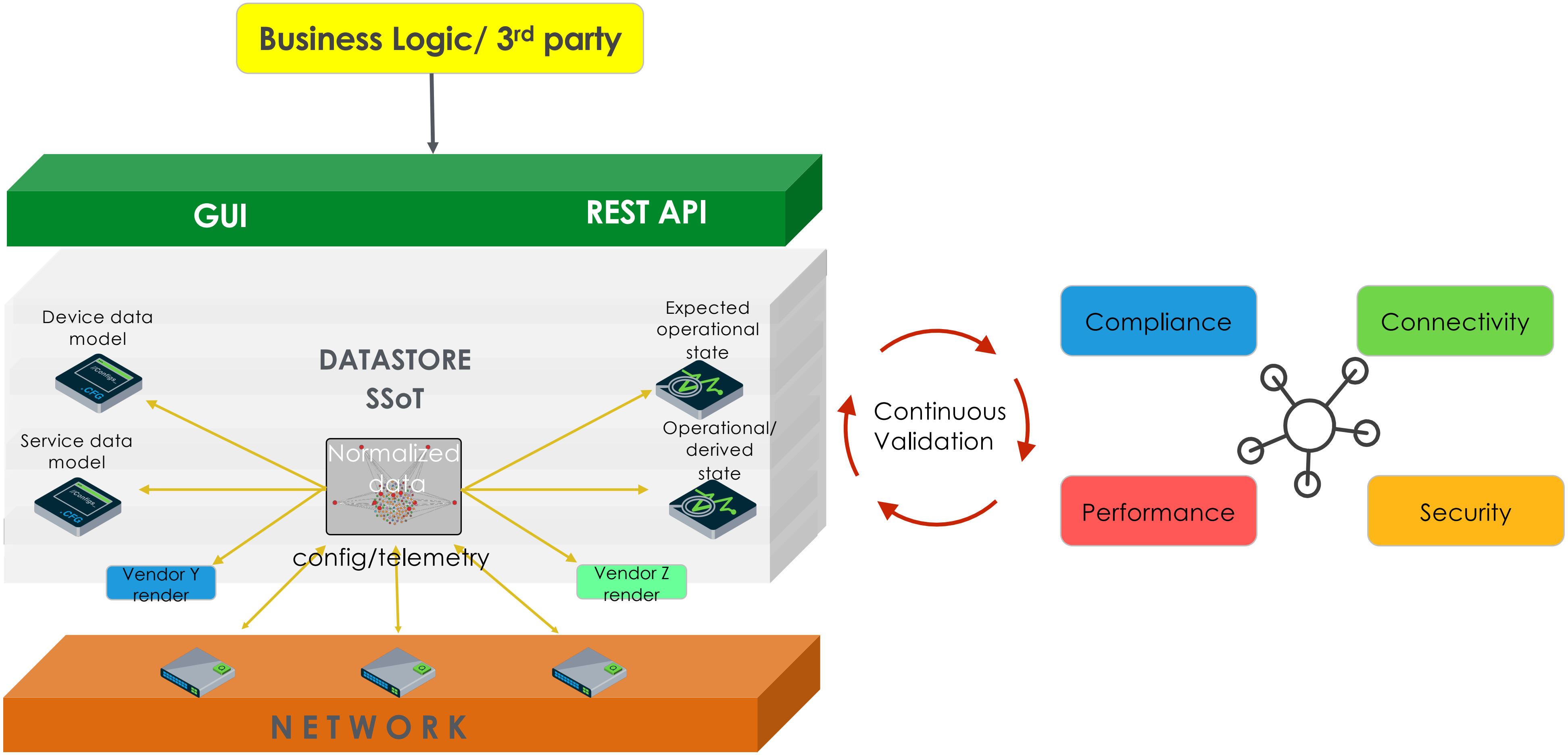
Let me instantiate your intent (networking)

Validate

Intent validation (continuous)

Let me validate that the network still does it as intended

Day in the IBNS life



Business Logic/ 3rd party

GUI

REST API

DATASTORE

SSoT

Device data model

Service data model

Normalized data

Expected operational state

Operational/ derived state

config/telemetry

Vendor Y render

Vendor Z render

NETWORK

Continuous Validation

Compliance

Connectivity

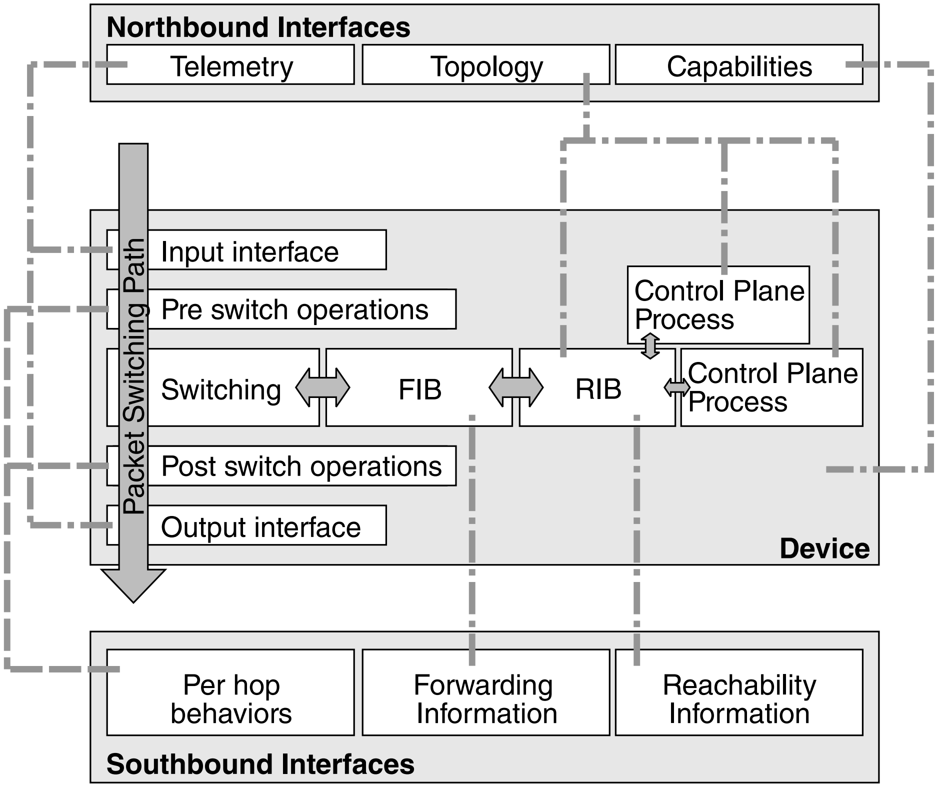
Performance

Security

Programmable Network and Interfaces

Network

A network is programmable when the control and data planes provide an interface that allows the state of the network to be modified and monitored through a machine readable data-driven API's



Source: Navigating Network Complexity: Book by Jeff Tantsura and Russ White

Programmable Network and Interfaces

apstra

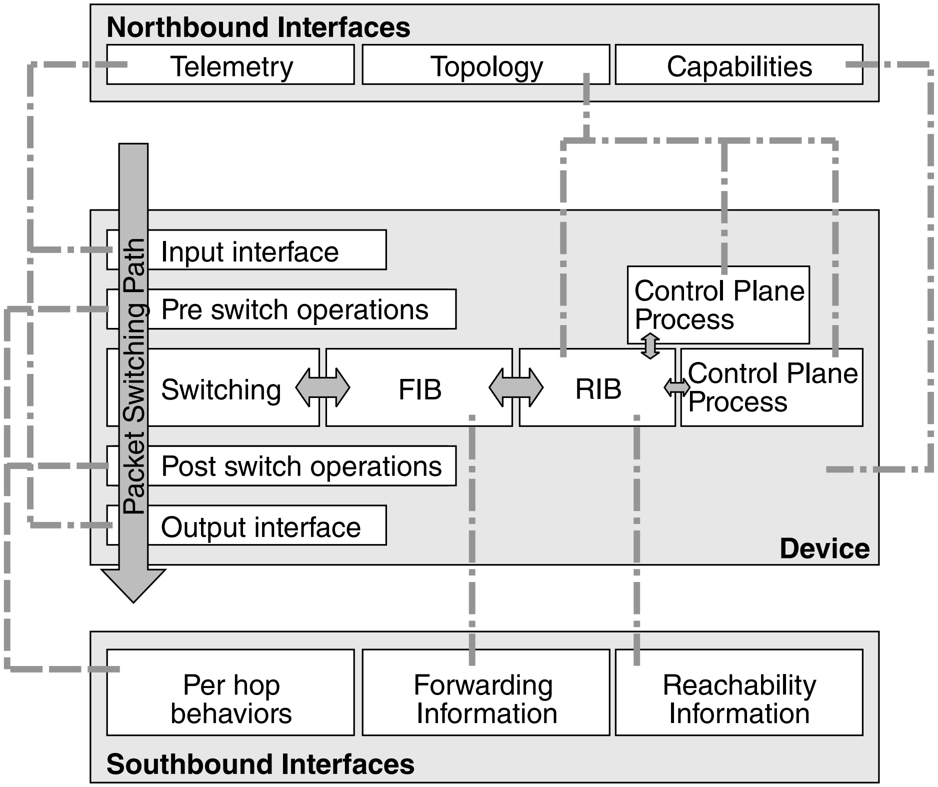
Interfaces

Capabilities: Provides information what can be programmed or controlled. This included schema's and metadata, or rather information about how the control structures are organized, and how information is presented by network devices to the controller.

Inventory: Provides information about what devices are installed where in the network, potentially including any information about physical connections.

Topology: Provides information about the state of links connecting network devices. This includes all the artifacts of the topology described.

Telemetry: Includes operational state, counters, and other information about the current network state. This includes but not limited by: resources used/available, queue depths, delay, jitter, etc…



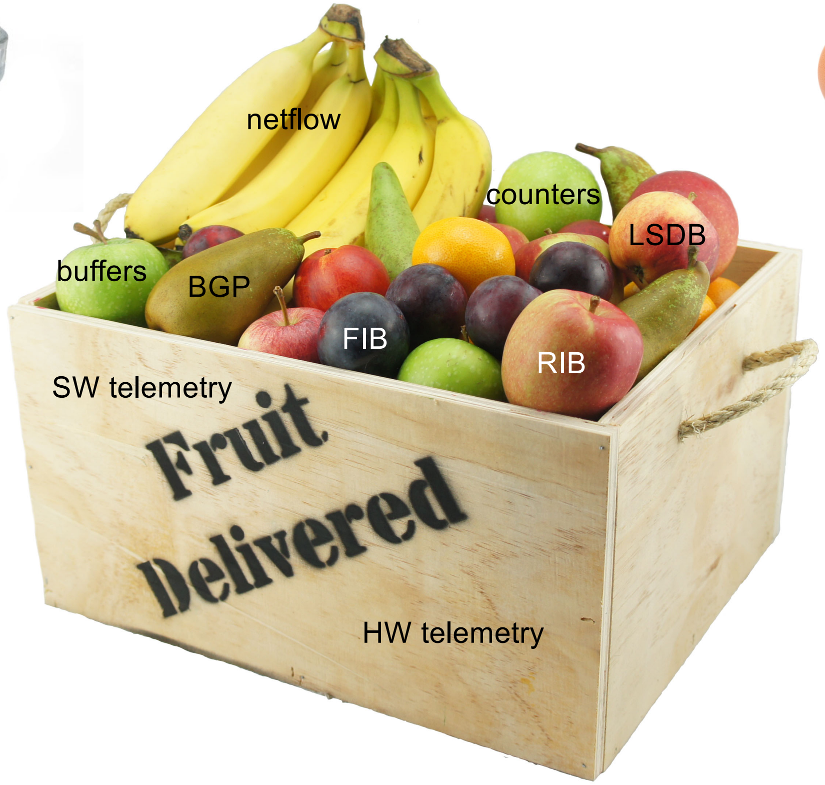
Source: Navigating Network Complexity: Book by Jeff Tantsura and Russ White

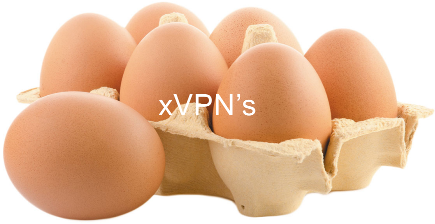
The need for data normalization

Source: food.com

➔ Fruitcake

The need for data normalization





Source: Twitter

Architectural Goals of IBN

Problems to be solved:

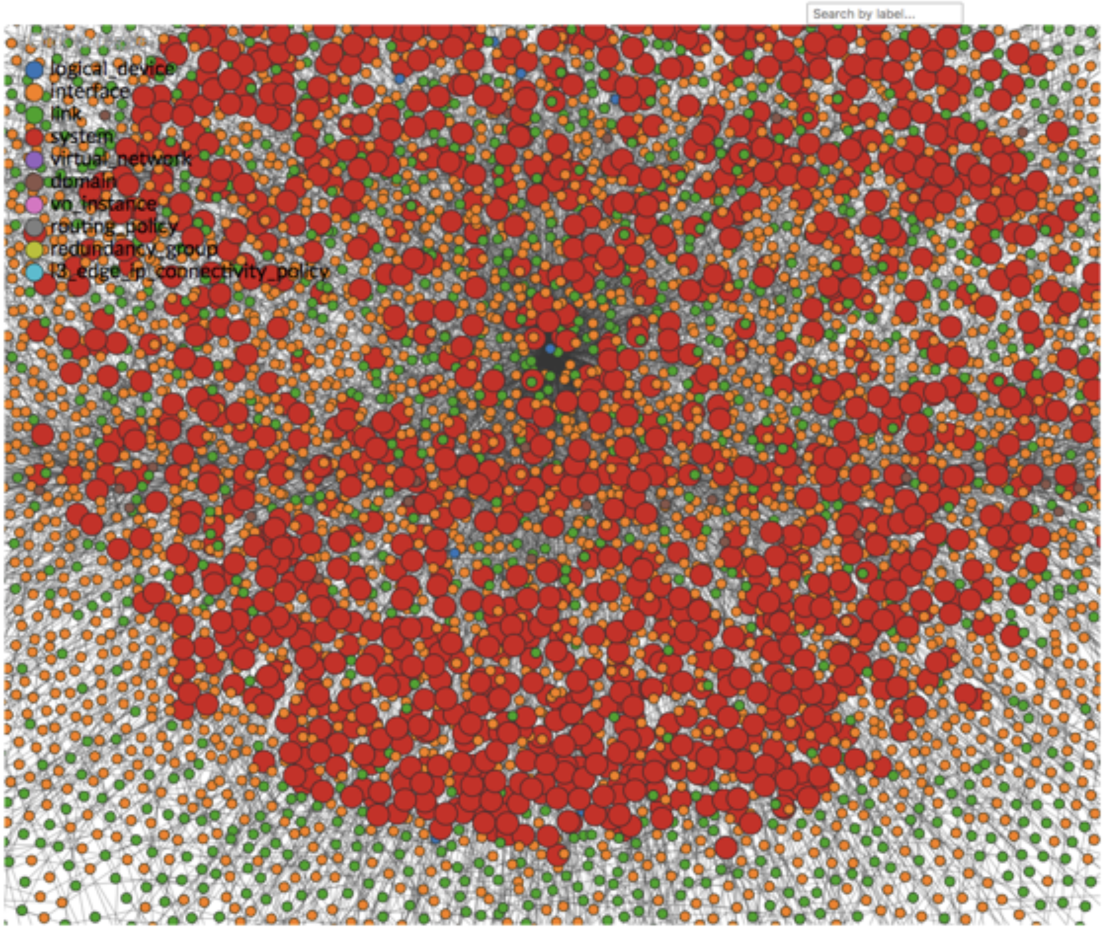
- Composition/decomposition @scale
- Dealing with changes:
  - Planned change – can I achieve desired (future) state while preserving original intent (meeting SLO's)
  - Unplanned change – impact of the change, difference between intended and operational states, how to get to intended state (remediation/notification)

Architectural Goals of IBN

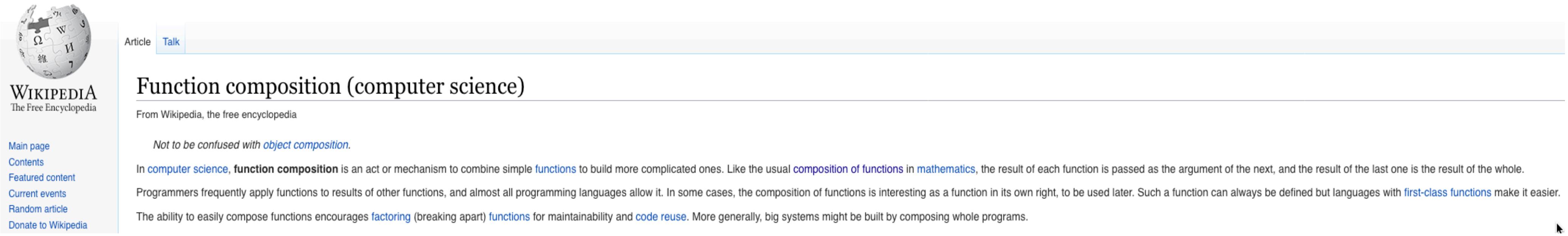
Problems to be solved:

- Closed loop validation:
  - continuously validate *outcomes* against the *intent* to ensure that the *composition* is working as intended
  - extract more knowledge by collecting less data thru IBA (Intent Based Analytics)
  - highly optimized SNR (signal to noise ratio) in analytics

Dealing With Scale?



Composition



Function composition (computer science)

From Wikipedia, the free encyclopedia

Not to be confused with object composition.

In computer science, **function composition** is an act or mechanism to combine simple functions to build more complicated ones. Like the usual composition of functions in mathematics, the result of each function is passed as the argument of the next, and the result of the last one is the result of the whole.

Programmers frequently apply functions to results of other functions, and almost all programming languages allow it. In some cases, the composition of functions is interesting as a function in its own right, to be used later. Such a function can always be defined but languages with first-class functions make it easier.

The ability to easily compose functions encourages factoring (breaking apart) functions for maintainability and code reuse. More generally, big systems might be built by composing whole programs.

Why model a graph?

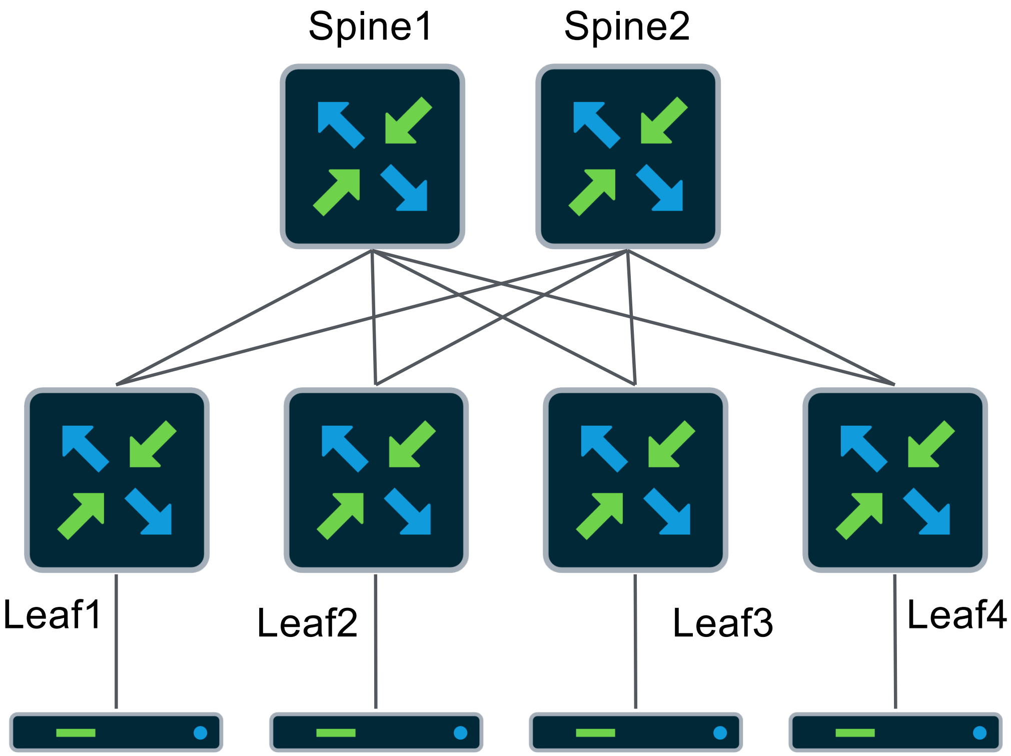
- Networks are intuitively the connected set of **nodes** and **relationships**

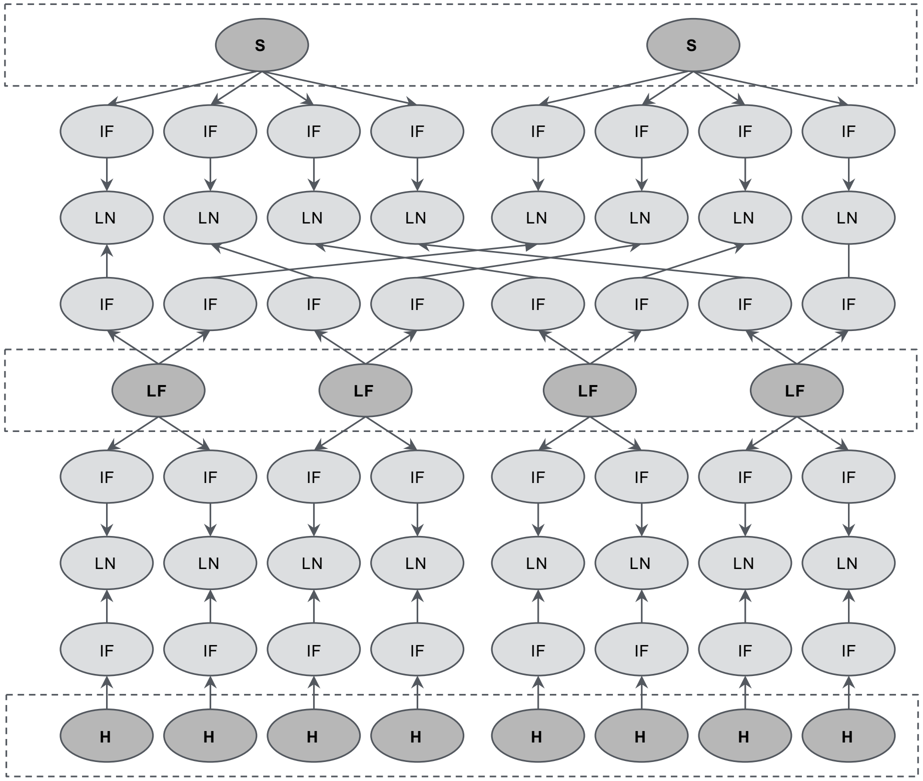
- As network requirements **change** the model can be easily **extended**

- Efficiently run **queries** that were **not anticipated** at model design time

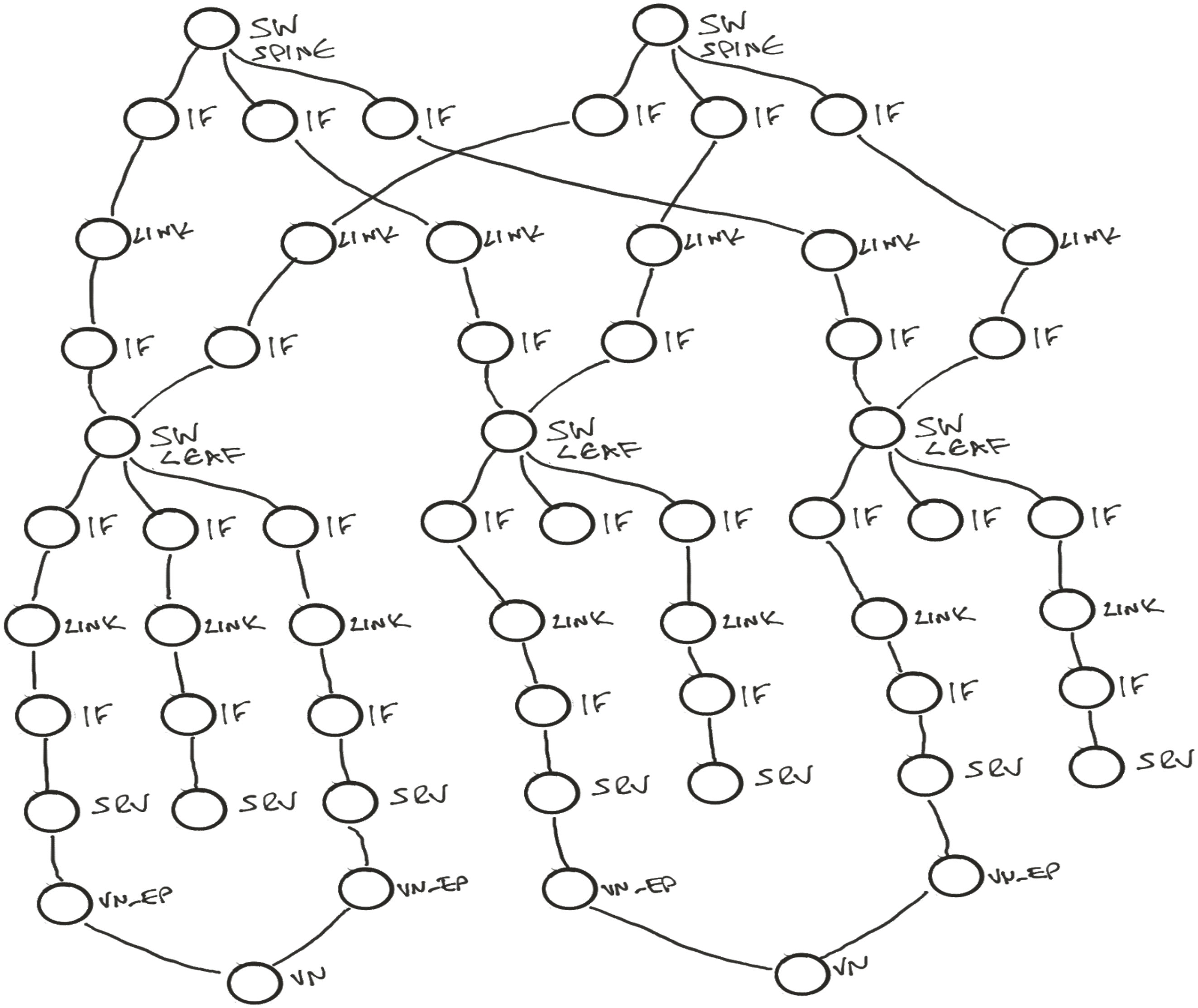
  Hint: you **will not** know all the queries at model definition time

Intent-> Graph composition

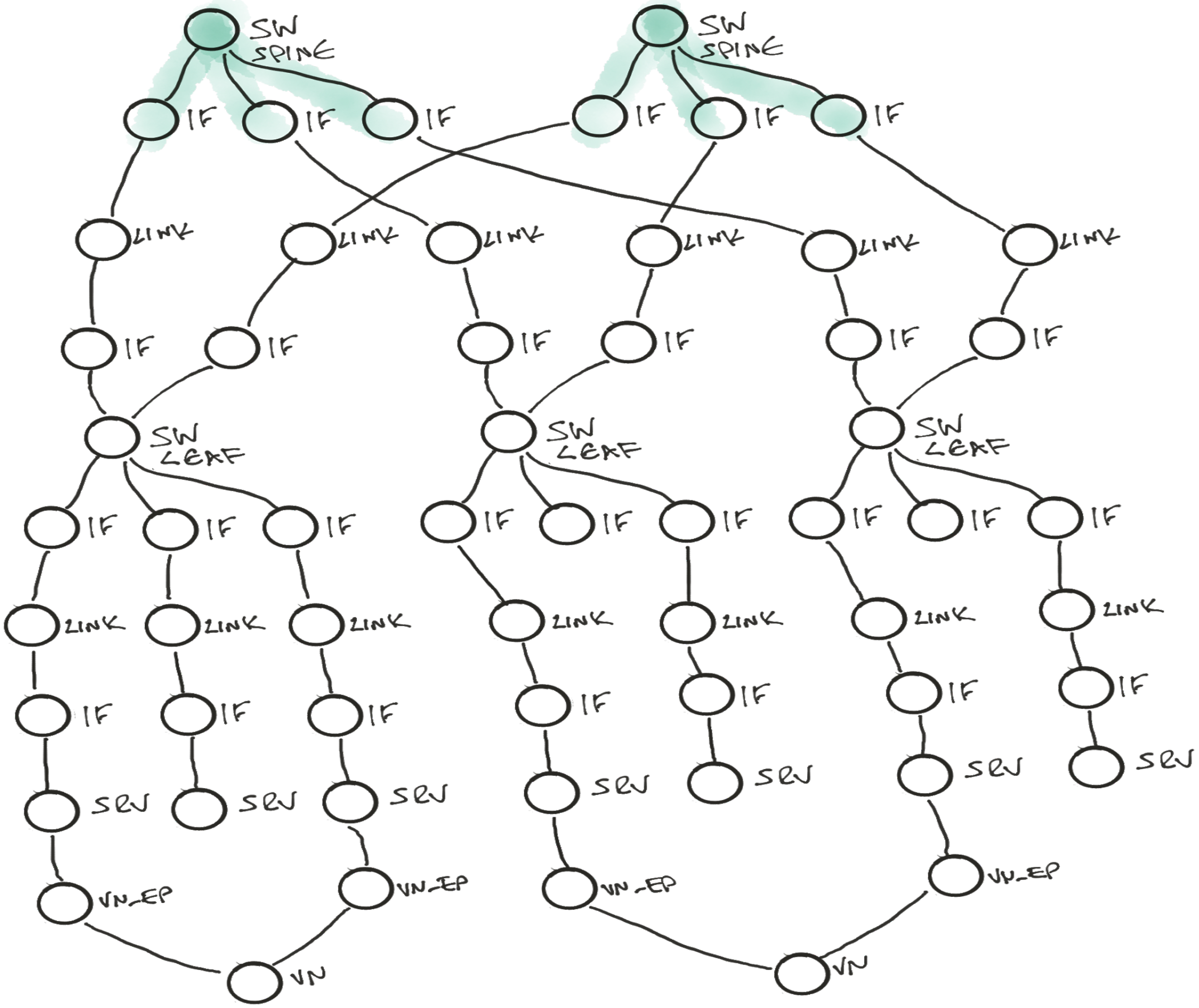




Function composition

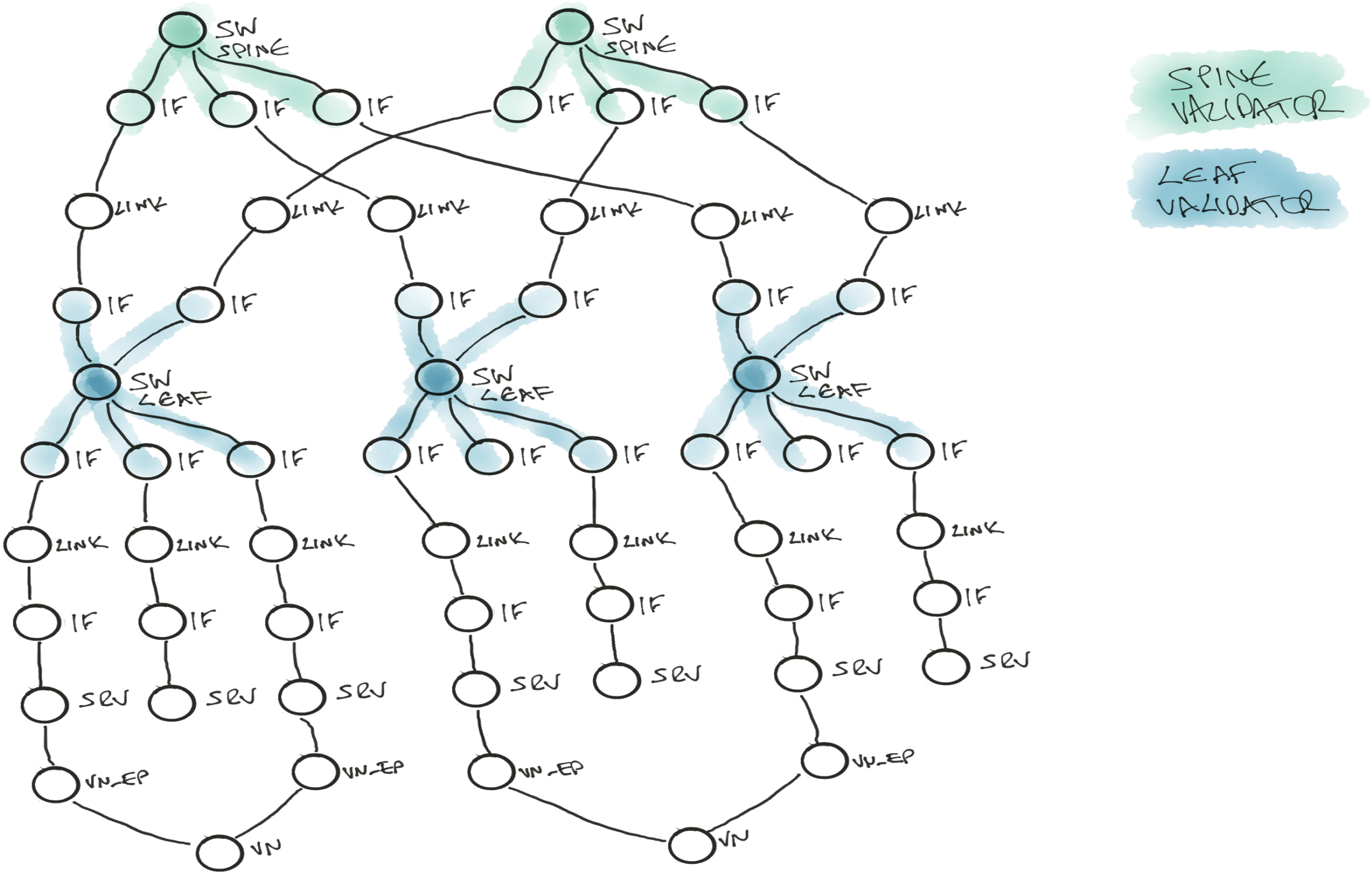

SW SPINE
SW SPINE
IF
IF
IF
IF
IF
IF
LINK
LINK
LINK
LINK
LINK
LINK
IF
IF
IF
IF
IF
IF
SW LEAF
SW LEAF
SW LEAF
IF
IF
IF
IF
IF
IF
IF
IF
IF
LINK
LINK
LINK
LINK
LINK
LINK
LINK
IF
IF
IF
IF
IF
IF
IF
SRV
SRV
SRV
SRV
SRV
SRV
SRV
VN_EP
VN_EP
VN_EP
VN_EP
VN
VN

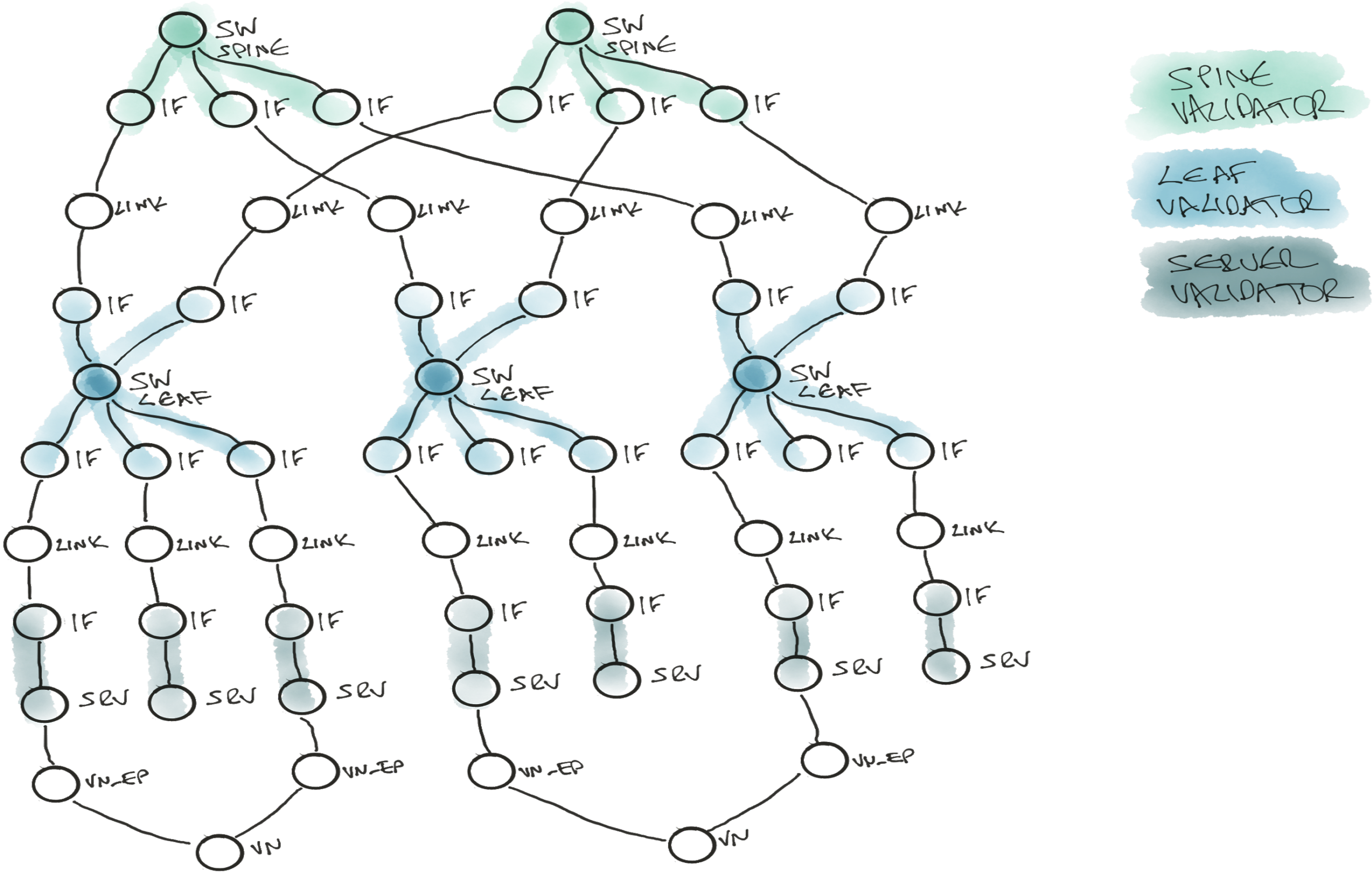
apstra

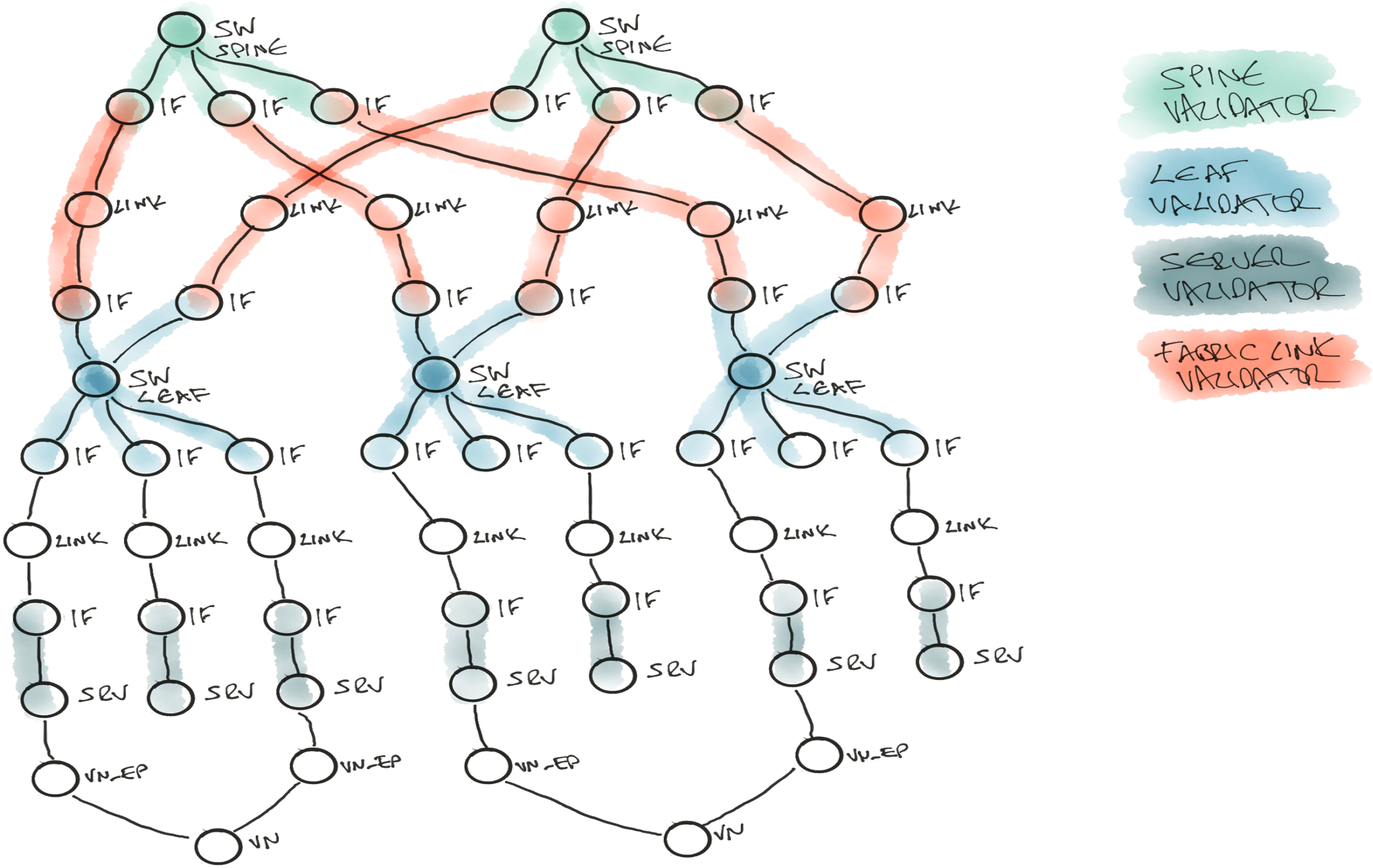


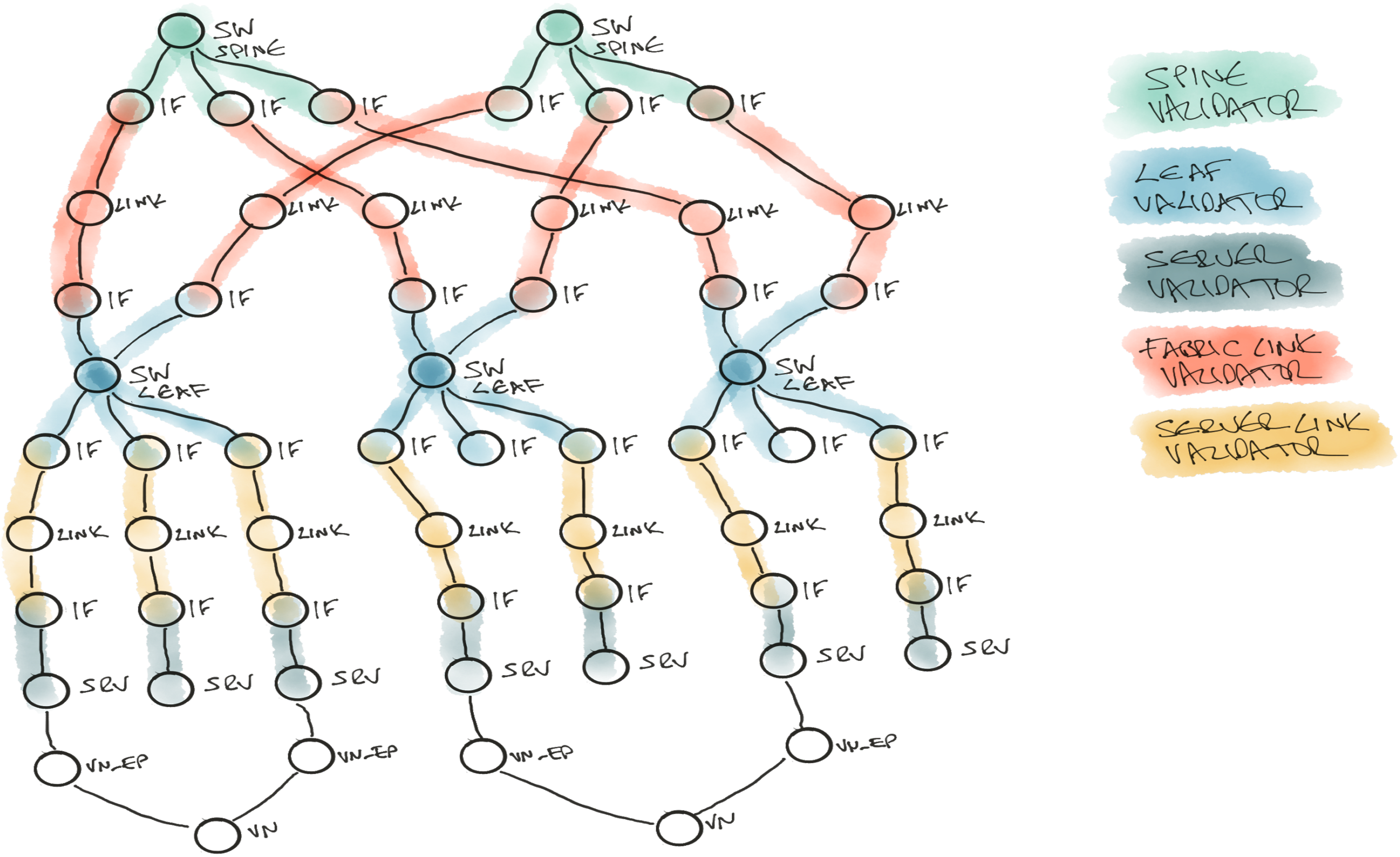
SPINE VALIDATOR

apstra

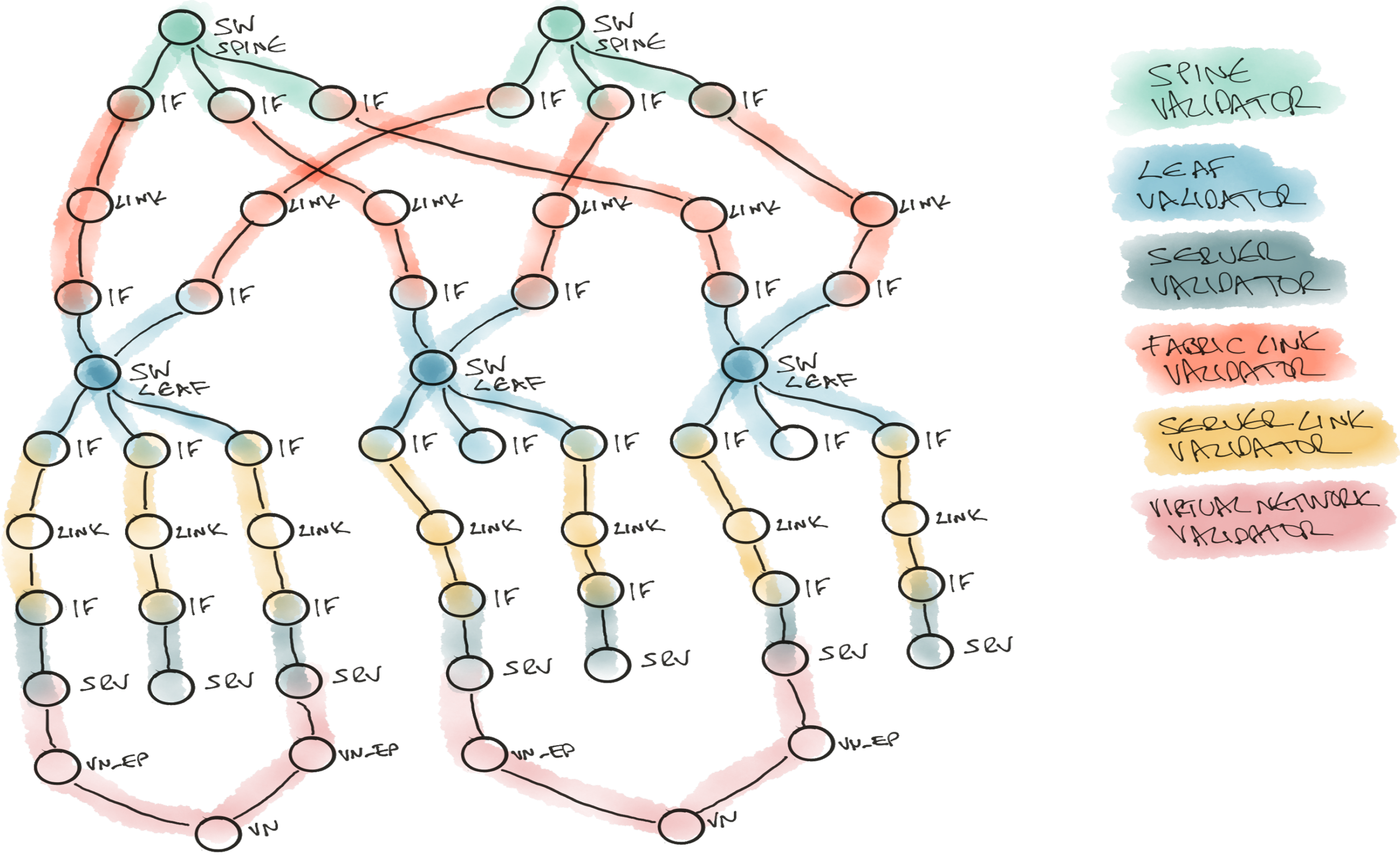

SW SPINE
SW SPINE
IF
LINK
SW LEAF
SW LEAF
SW LEAF
SRV
VN_EP
VN
VN
SPINE VALIDATOR
LEAF VALIDATOR


SW SPINE
SW SPINE
IF
IF
IF
IF
IF
IF
LINK
LINK
LINK
LINK
LINK
LINK
IF
IF
IF
IF
IF
IF
SW LEAF
SW LEAF
SW LEAF
IF
IF
IF
IF
IF
IF
IF
IF
IF
LINK
LINK
LINK
LINK
LINK
LINK
LINK
IF
IF
IF
IF
IF
IF
IF
SERV
SERV
SERV
SERV
SERV
SERV
SERV
VN_EP
VN_EP
VN_EP
VN_EP
VN
VN
SPINE VALIDATOR
LEAF VALIDATOR
SERVER VALIDATOR

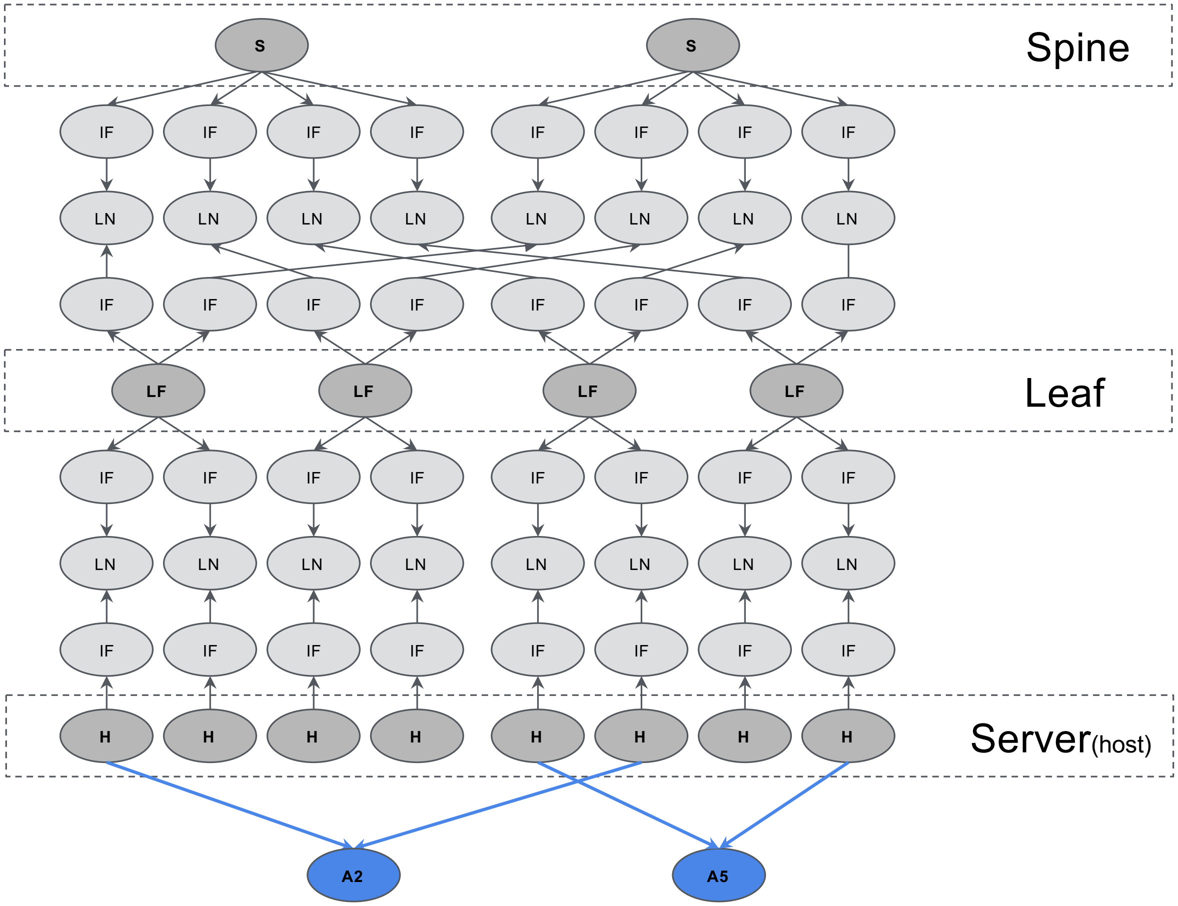

SW SPINE
SW SPINE
IF
LINK
SW LEAF
SW LEAF
SW LEAF
SERV
VN_EP
VN
VN
SPINE VALIDATOR
LEAF VALIDATOR
SERVER VALIDATOR
FABRIC LINK VALIDATOR


SW SPINE
SW SPINE
IF
LINK
IF
SW LEAF
SW LEAF
SW LEAF
IF
LINK
IF
SERV
VN_EP
VN
VN
SPINE VALIDATOR
LEAF VALIDATOR
SERVER VALIDATOR
FABRIC LINK VALIDATOR
SERVER LINK VALIDATOR

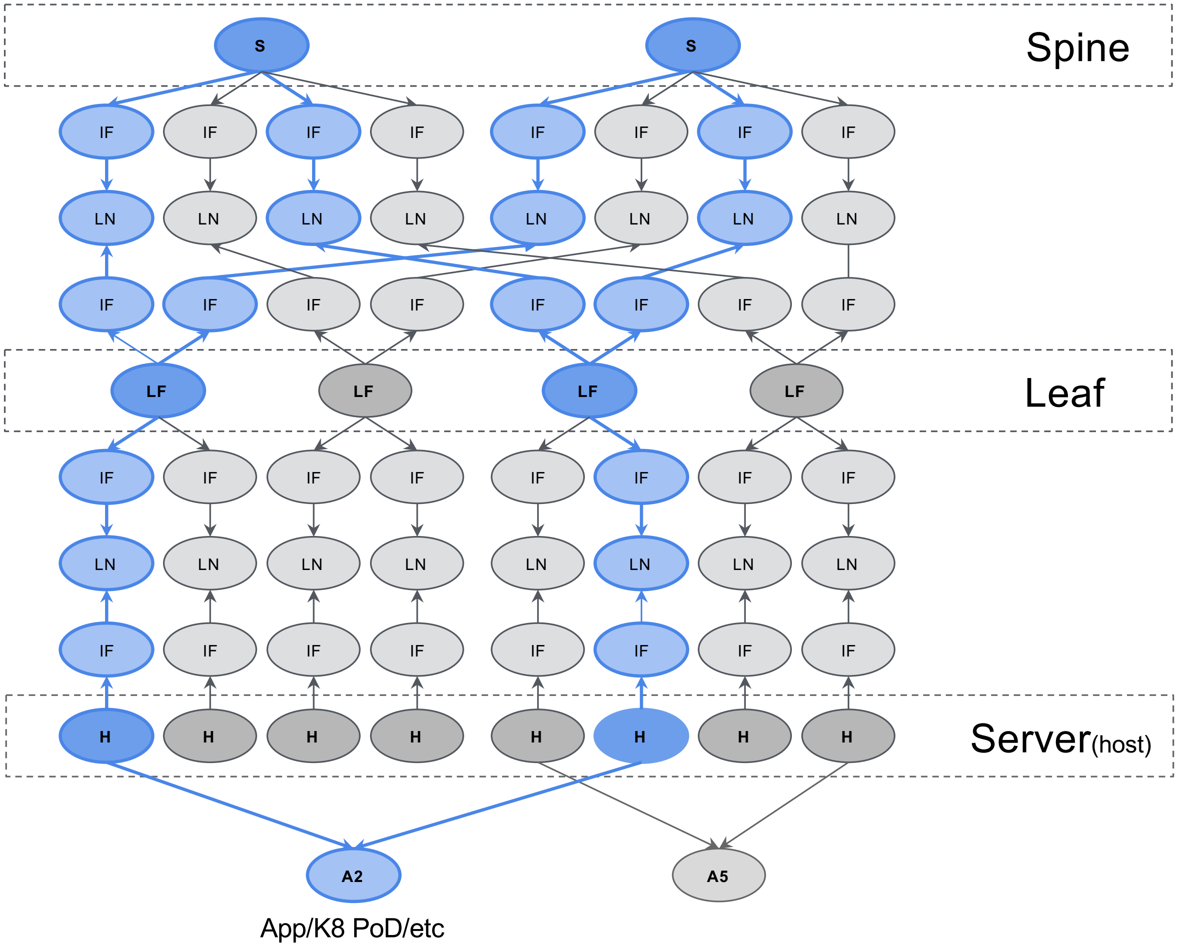
apstra


SW SPINE
SW SPINE
IF
IF
IF
IF
IF
IF
LINK
LINK
LINK
LINK
LINK
LINK
IF
IF
IF
IF
IF
IF
SW LEAF
SW LEAF
SW LEAF
IF
IF
IF
IF
IF
IF
IF
IF
IF
LINK
LINK
LINK
LINK
LINK
LINK
LINK
IF
IF
IF
IF
IF
IF
IF
SERV
SERV
SERV
SERV
SERV
SERV
SERV
VN_EP
VN_EP
VN_EP
VN_EP
VN
VN
SPINE VALIDATOR
LEAF VALIDATOR
SERVER VALIDATOR
FABRIC LINK VALIDATOR
SERVER LINK VALIDATOR
VIRTUAL NETWORK VALIDATOR

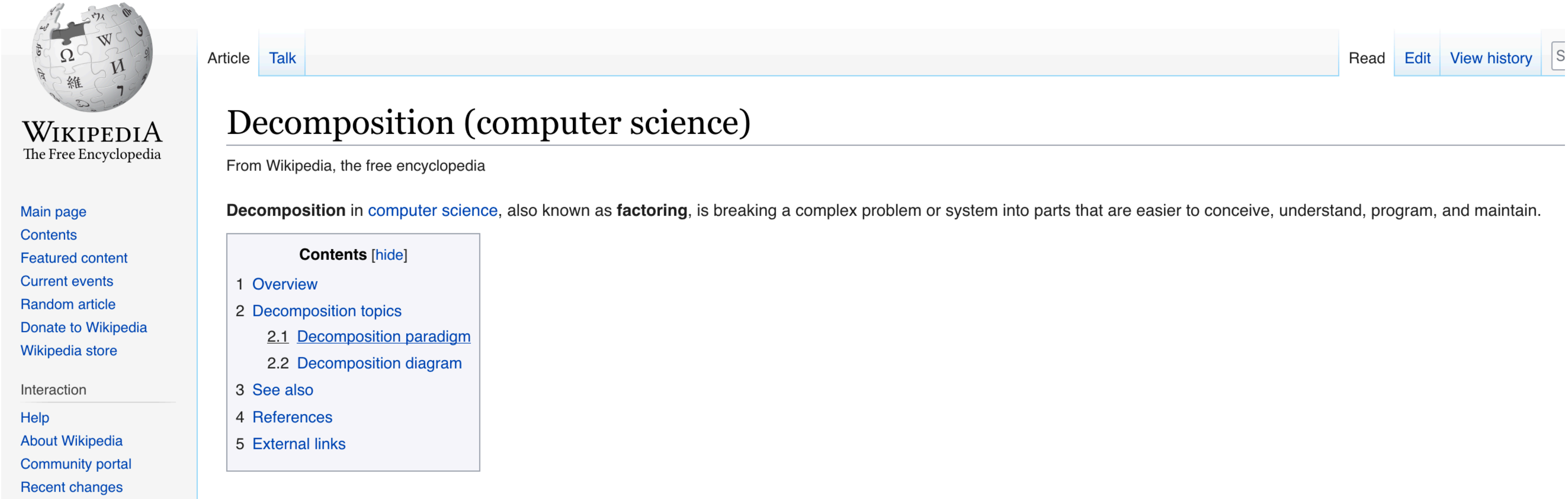
Resulting Model



Query: Links that carry “A2” traffic



Decomposition



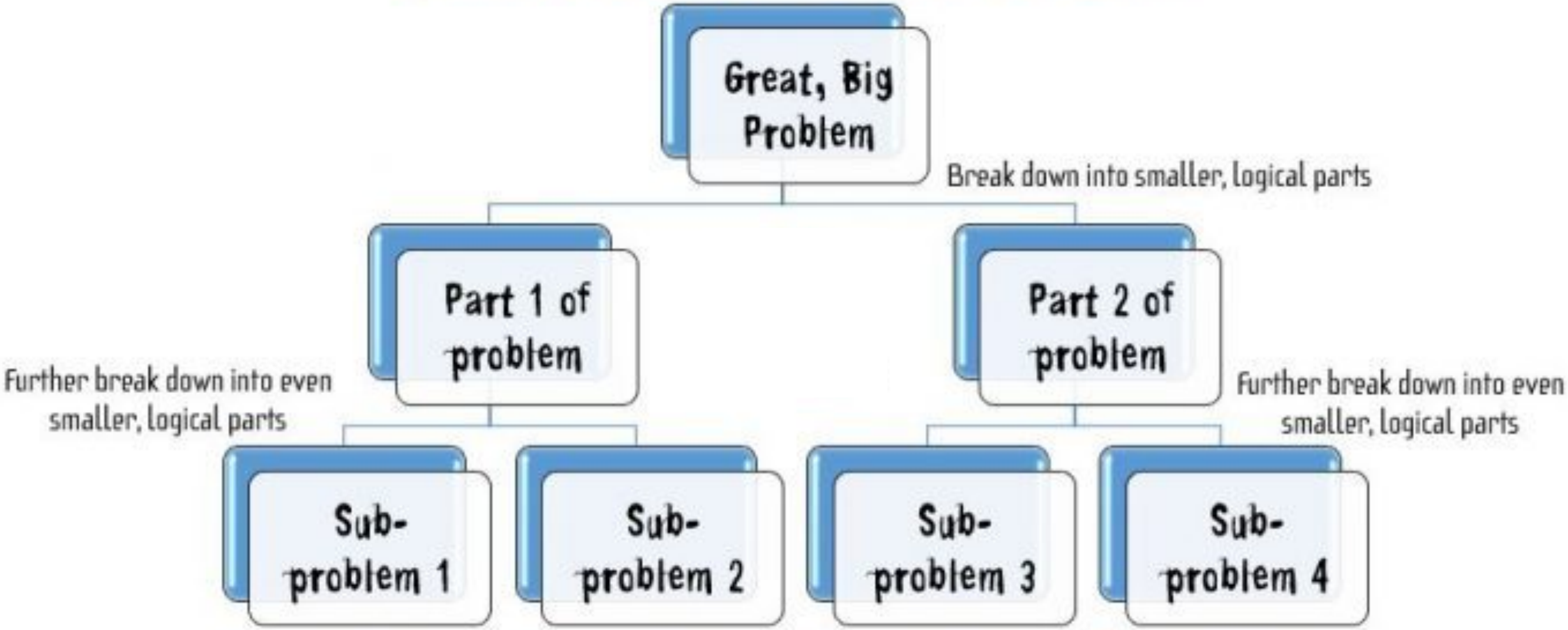
Decomposition (computer science)

From Wikipedia, the free encyclopedia

Decomposition in computer science, also known as **factoring**, is breaking a complex problem or system into parts that are easier to conceive, understand, program, and maintain.

Contents [hide]

DECOMPOSITION



Great, Big Problem

Break down into smaller, logical parts

Part 1 of problem

Part 2 of problem

Further break down into even smaller, logical parts

Further break down into even smaller, logical parts

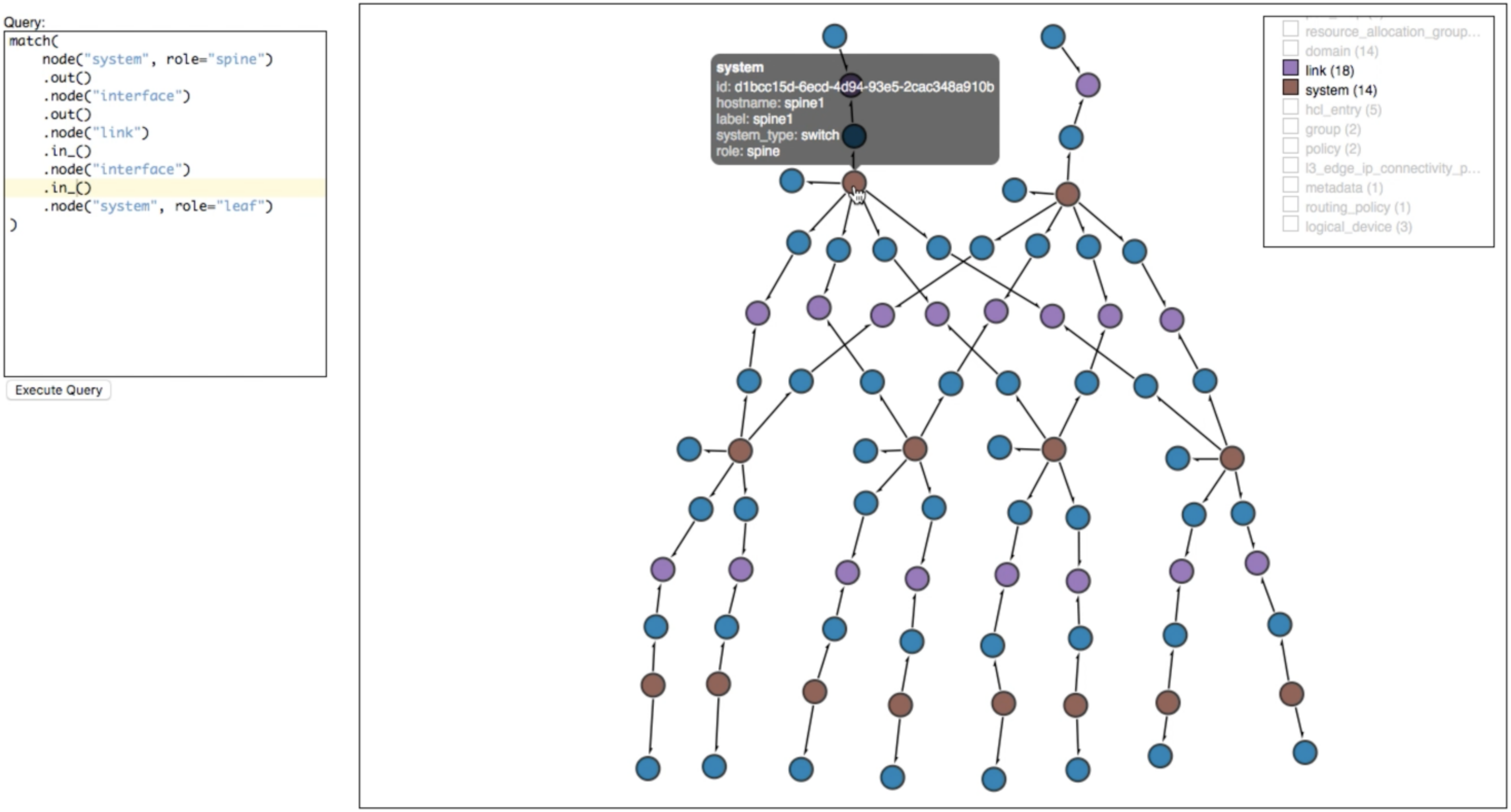
Sub-problem 1

Sub-problem 2

Sub-problem 3

Sub-problem 4

Decomposition: walking the graph

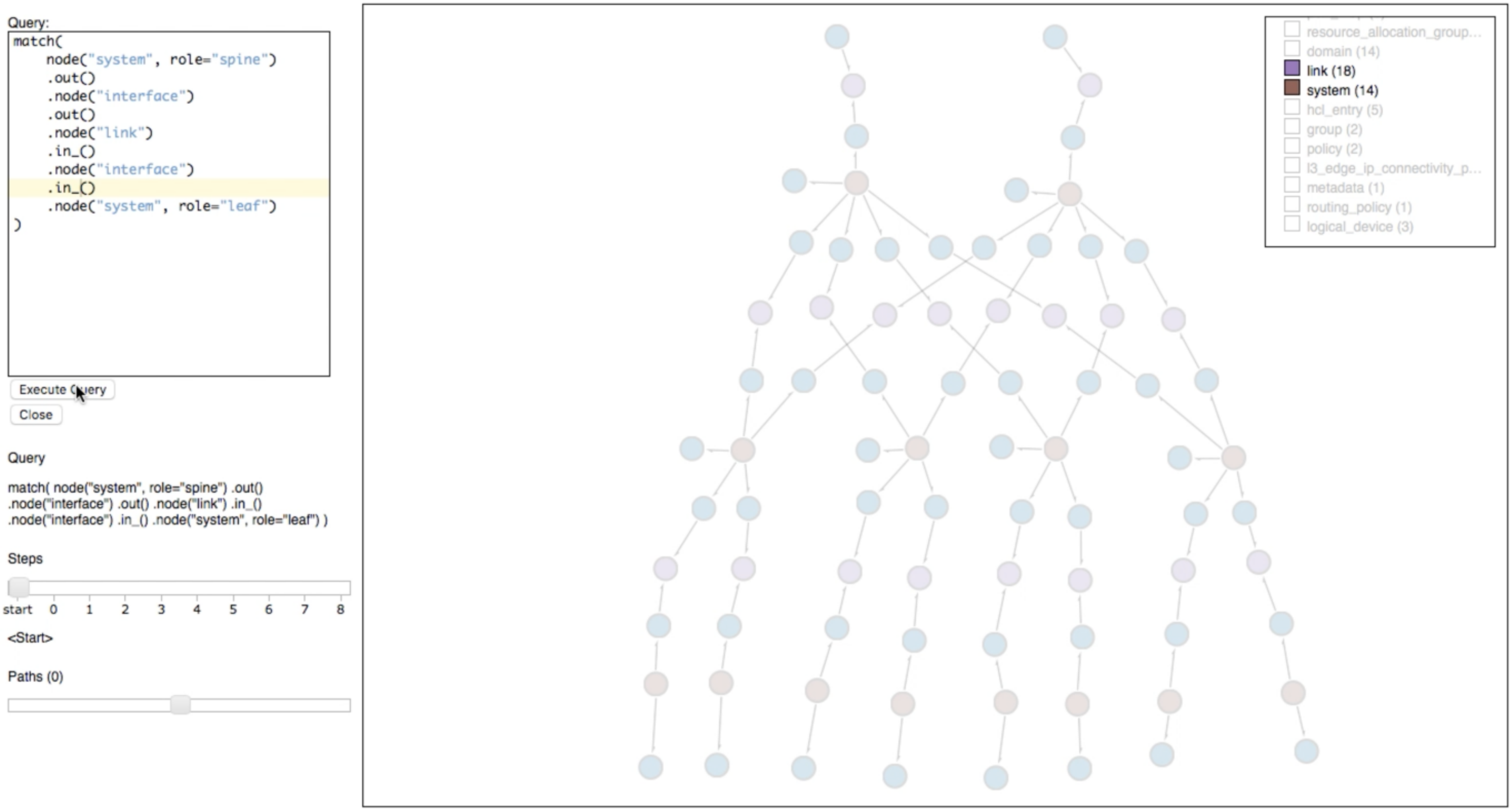


Query:

```
match(
    node("system", role="spine")
    .out()
    .node("interface")
    .out()
    .node("link")
    .in_()
    .node("interface")
    .in_()
    .node("system", role="leaf")
)
```

Execute Query

system
id: d1bcc15d-6ecd-4d94-93e5-2cac348a910b
hostname: spine1
label: spine1
system_type: switch
role: spine


Query:

```
match(
    node("system", role="spine")
    .out()
    .node("interface")
    .out()
    .node("link")
    .in_()
    .node("interface")
    .in_()
    .node("system", role="leaf")
)
```

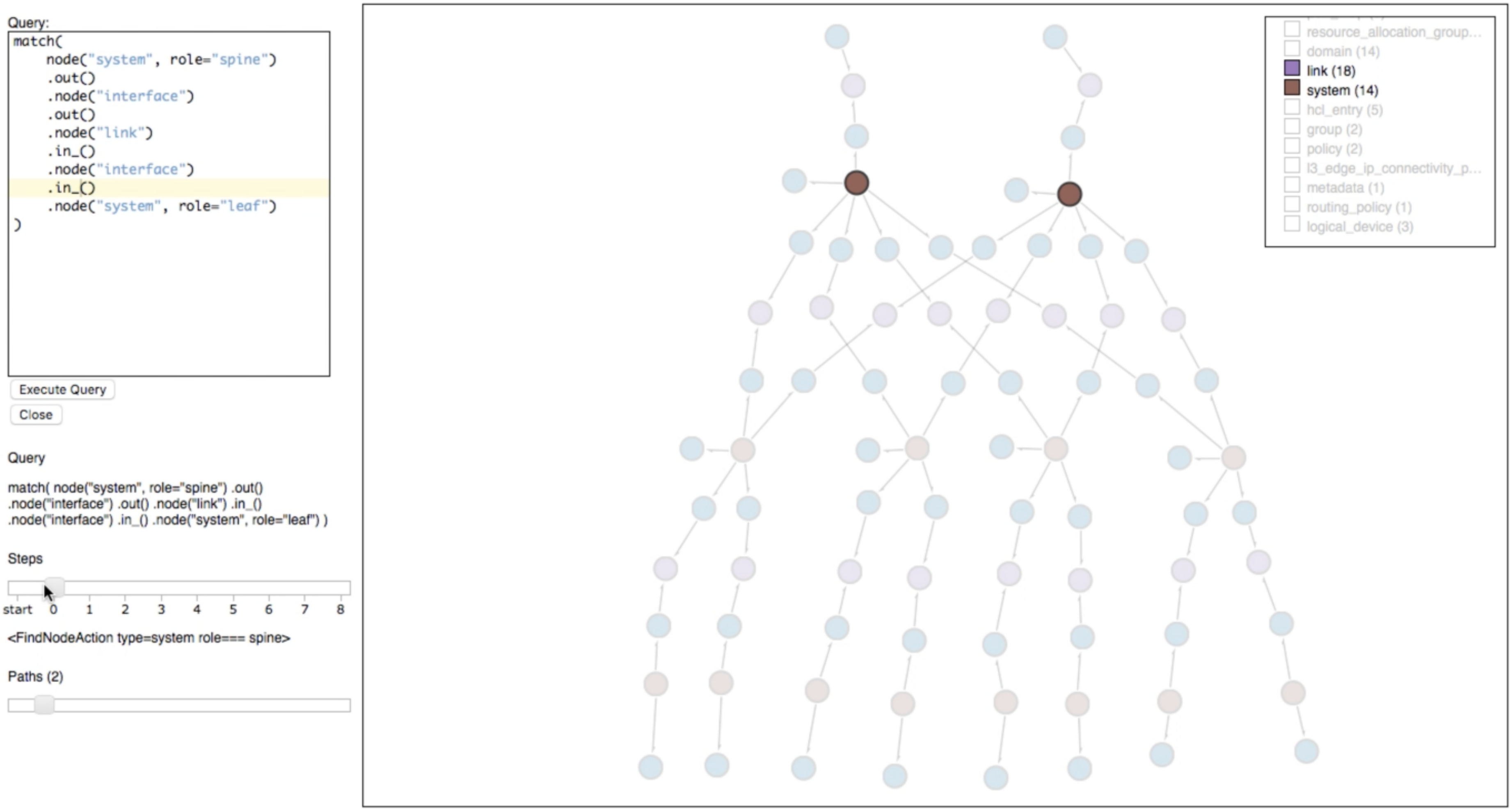
Query

match(node("system", role="spine") .out() .node("interface") .out() .node("link") .in_() .node("interface") .in_() .node("system", role="leaf"))

Steps

<Start>

Paths (0)



Query:

```
match(
    node("system", role="spine")
    .out()
    .node("interface")
    .out()
    .node("link")
    .in_()
    .node("interface")
    .in_()
    .node("system", role="leaf")
)
```

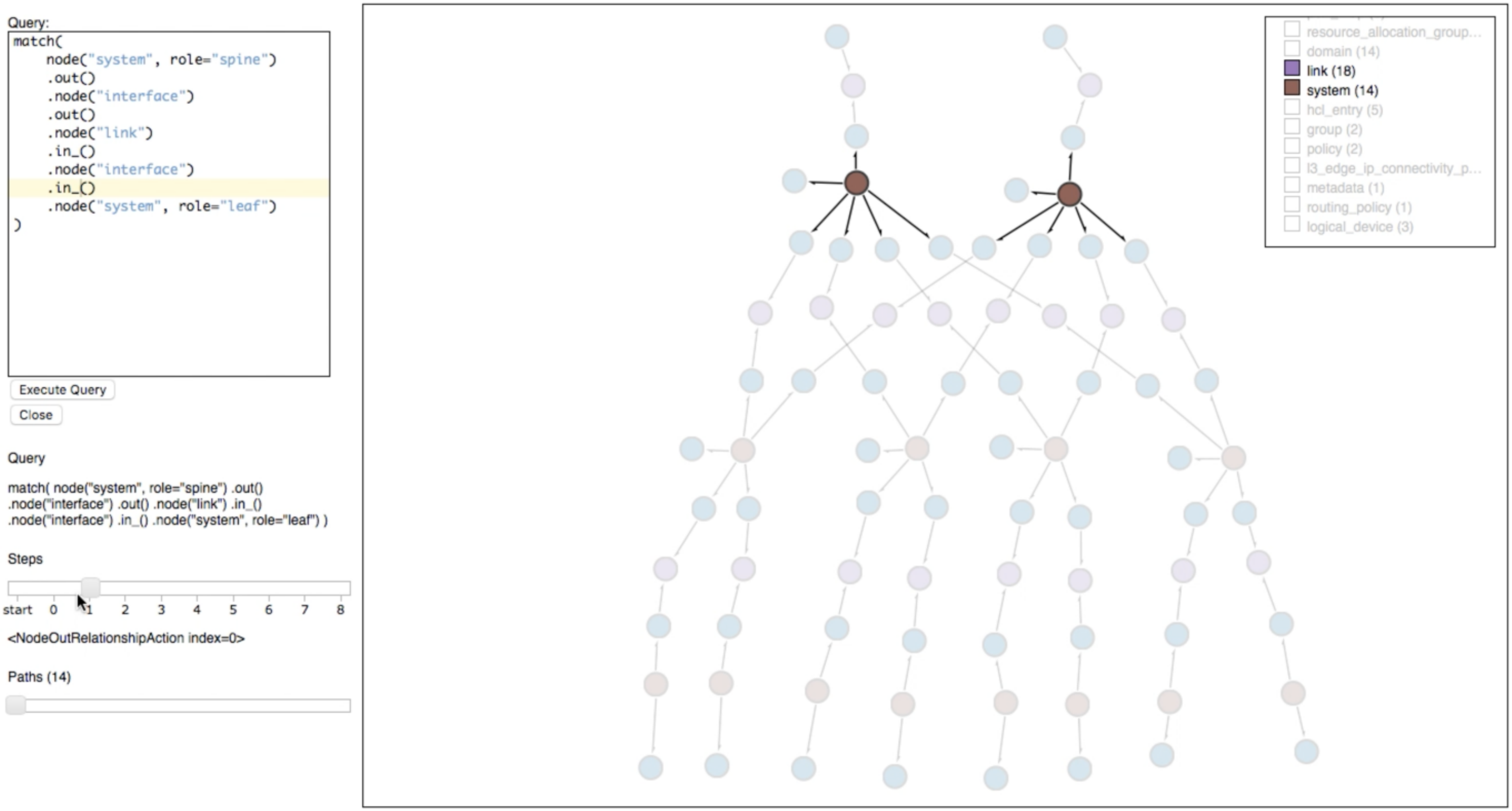
Query

match(node("system", role="spine") .out() .node("interface") .out() .node("link") .in_() .node("interface") .in_() .node("system", role="leaf"))

Steps

<FindNodeAction type=system role=== spine>

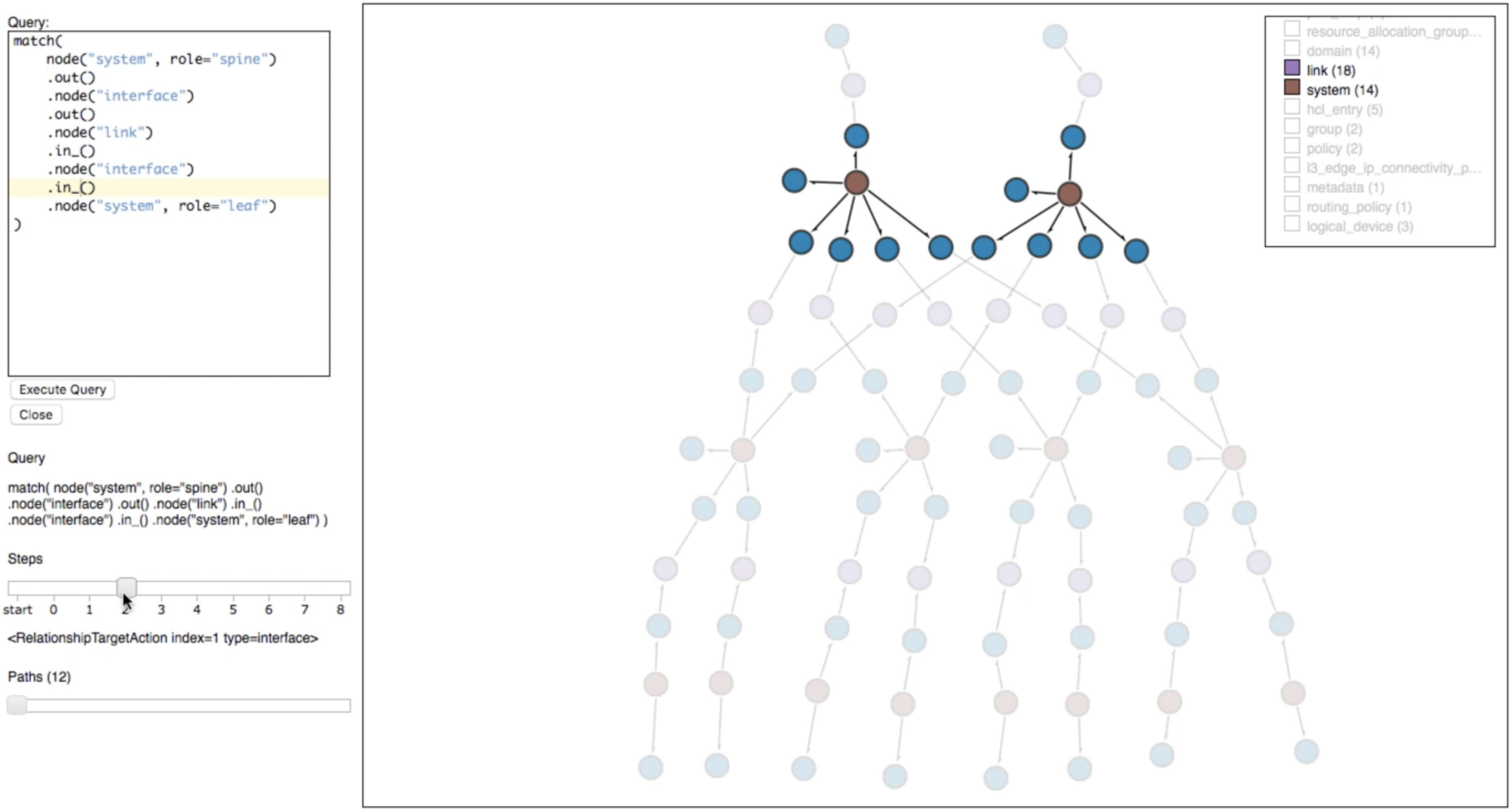
Paths (2)



```
match(
    node("system", role="spine")
    .out()
    .node("interface")
    .out()
    .node("link")
    .in_()
    .node("interface")
    .in_()
    .node("system", role="leaf")
)
```

match(node("system", role="spine") .out() .node("interface") .out() .node("link") .in_() .node("interface") .in_() .node("system", role="leaf"))

<NodeOutRelationshipAction index=0>



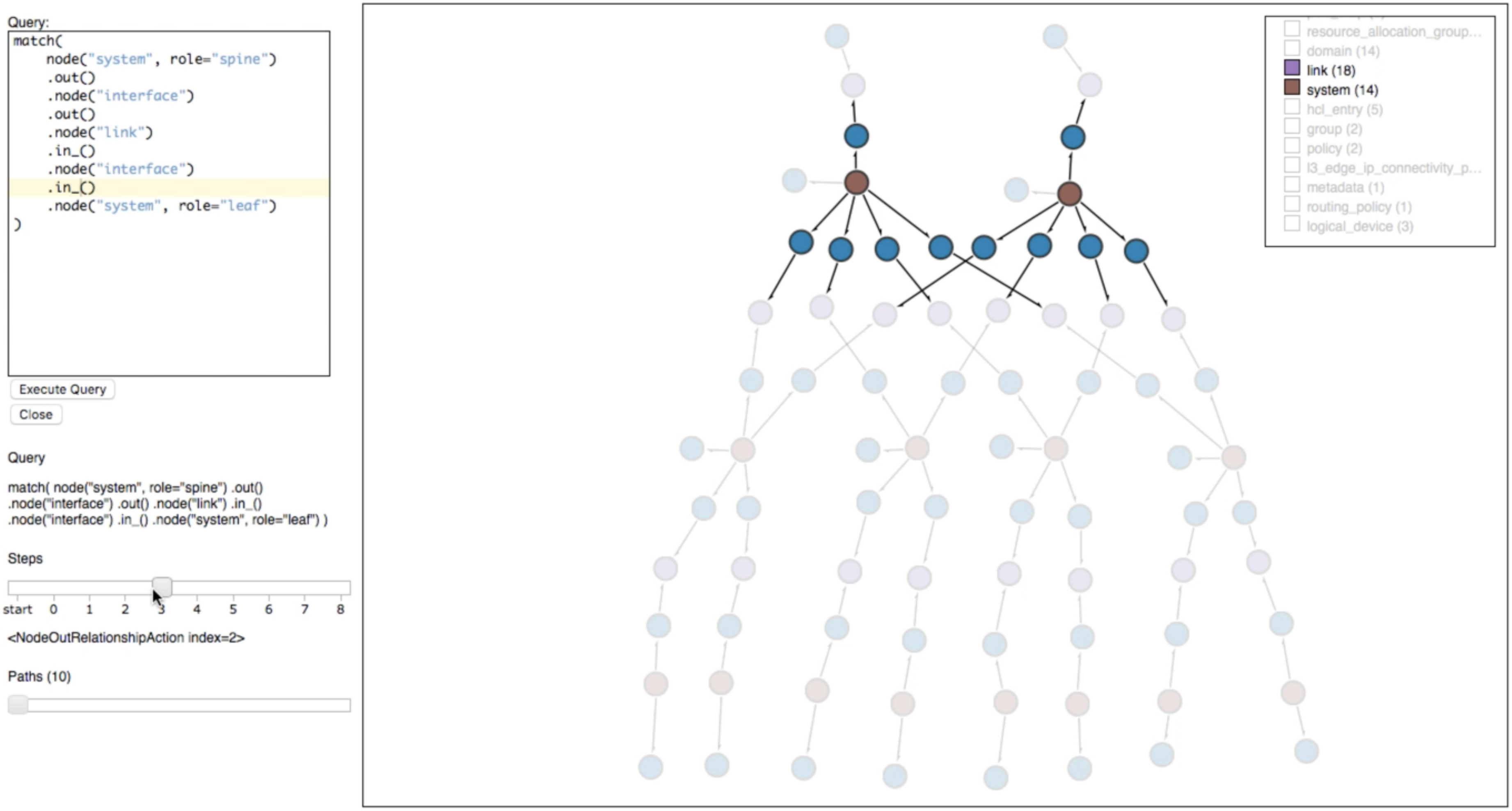
Query:

```
match(
    node("system", role="spine")
    .out()
    .node("interface")
    .out()
    .node("link")
    .in_()
    .node("interface")
    .in_()
    .node("system", role="leaf")
)
```

Query

match(node("system", role="spine") .out() .node("interface") .out() .node("link") .in_() .node("interface") .in_() .node("system", role="leaf"))

<RelationshipTargetAction index=1 type=interface>



Query:

```
match(
    node("system", role="spine")
    .out()
    .node("interface")
    .out()
    .node("link")
    .in_()
    .node("interface")
    .in_()
    .node("system", role="leaf")
)
```

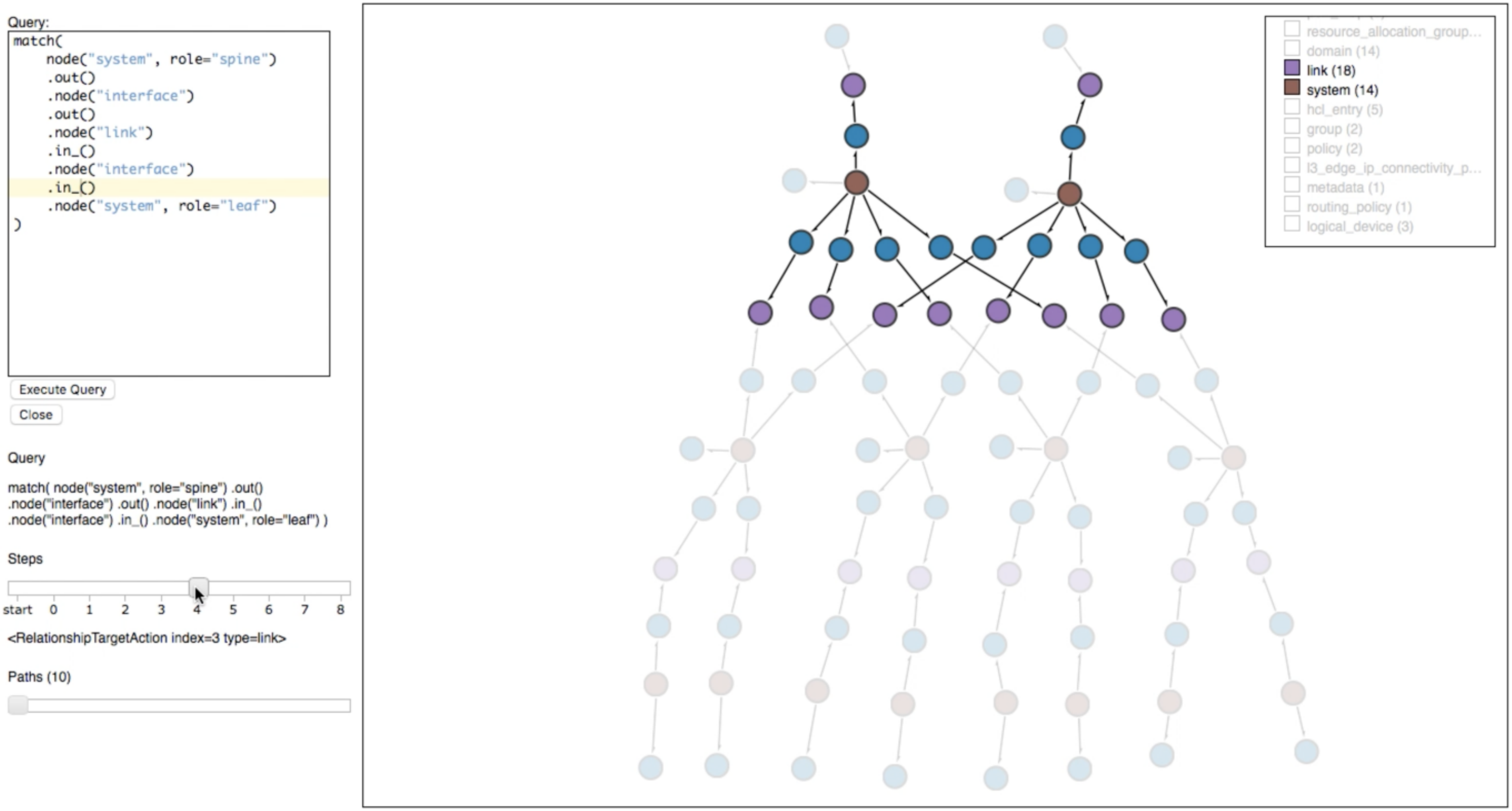
Query

match(node("system", role="spine") .out() .node("interface") .out() .node("link") .in_() .node("interface") .in_() .node("system", role="leaf"))

Steps

<NodeOutRelationshipAction index=2>

Paths (10)



Query:

```
match(
    node("system", role="spine")
    .out()
    .node("interface")
    .out()
    .node("link")
    .in_()
    .node("interface")
    .in_()
    .node("system", role="leaf")
)
```

Query

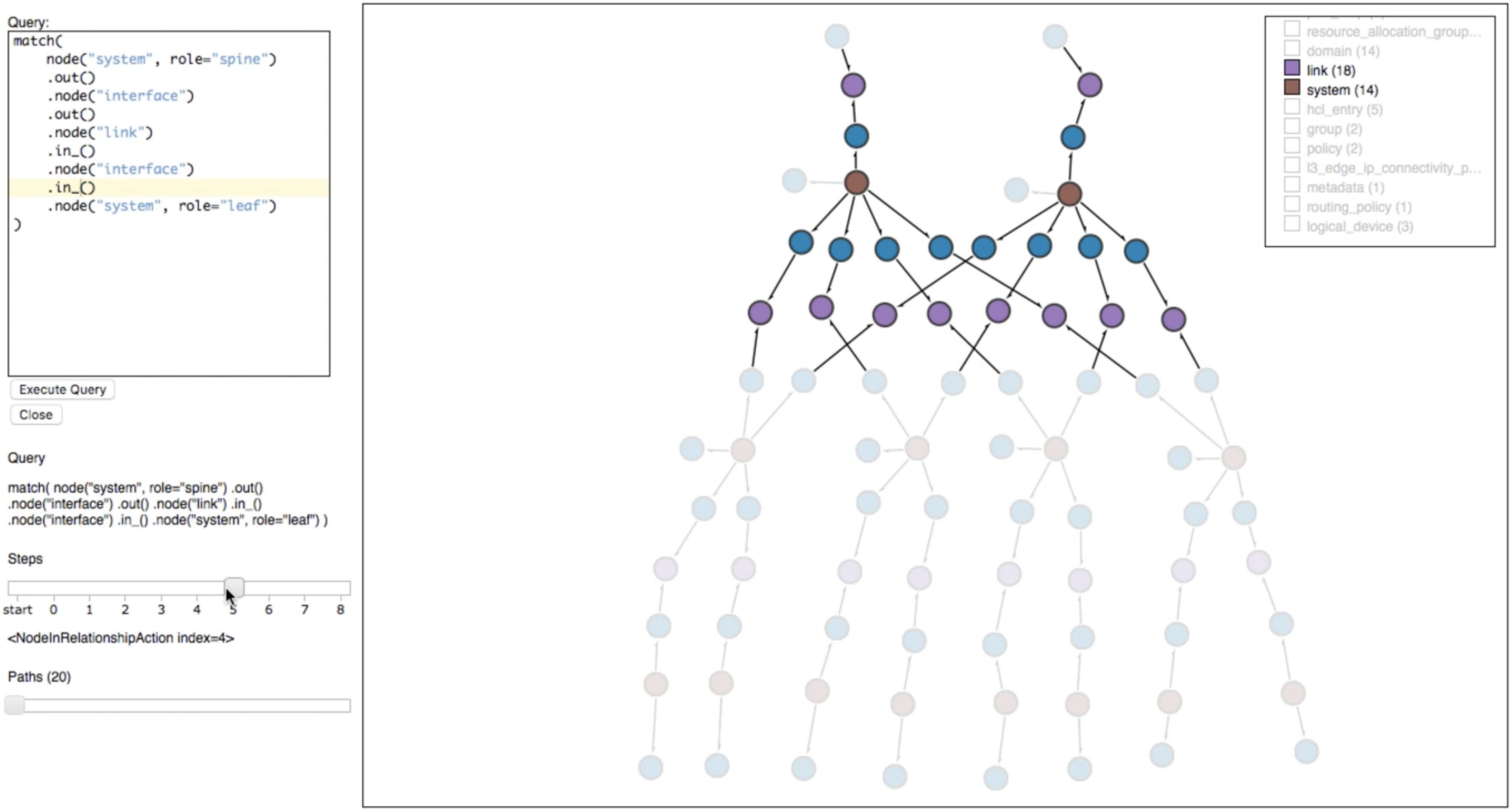
match(node("system", role="spine") .out() .node("interface") .out() .node("link") .in_() .node("interface") .in_() .node("system", role="leaf"))

Steps

<RelationshipTargetAction index=3 type=link>

Paths (10)

- ☐ resource_allocation_group...
- ☐ domain (14)
- 🟪 link (18)
- 🟫 system (14)
- ☐ hcl_entry (5)
- ☐ group (2)
- ☐ policy (2)
- ☐ l3_edge_ip_connectivity_p...
- ☐ metadata (1)
- ☐ routing_policy (1)
- ☐ logical_device (3)



Query:

```
match(
    node("system", role="spine")
    .out()
    .node("interface")
    .out()
    .node("link")
    .in_()
    .node("interface")
    .in_()
    .node("system", role="leaf")
)
```

Execute Query

Close

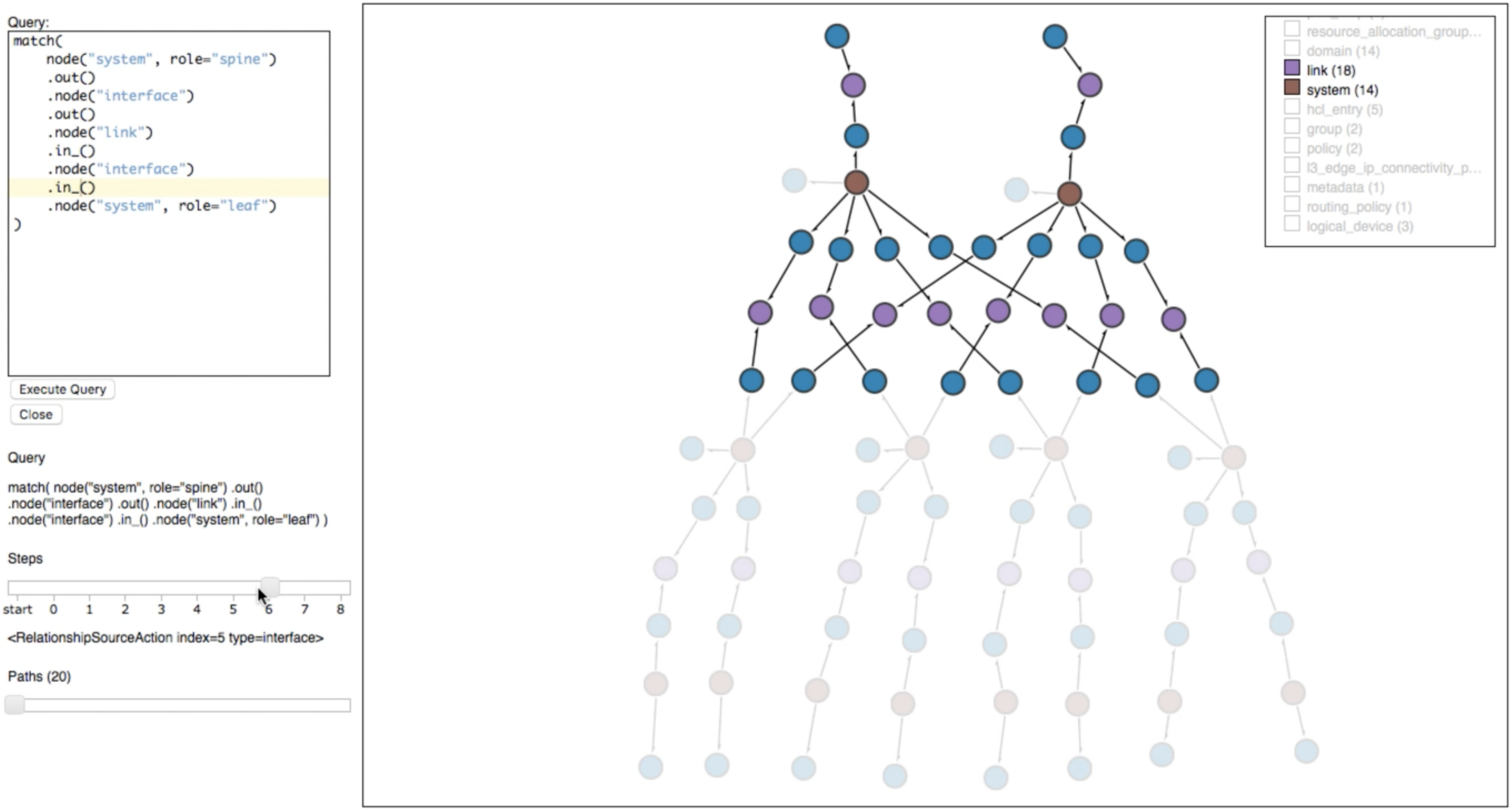
Query

match(node("system", role="spine") .out() .node("interface") .out() .node("link") .in_() .node("interface") .in_() .node("system", role="leaf"))

Steps

<NodeInRelationshipAction index=4>

Paths (20)



Query:

```
match(
    node("system", role="spine")
    .out()
    .node("interface")
    .out()
    .node("link")
    .in_()
    .node("interface")
    .in_()
    .node("system", role="leaf")
)
```

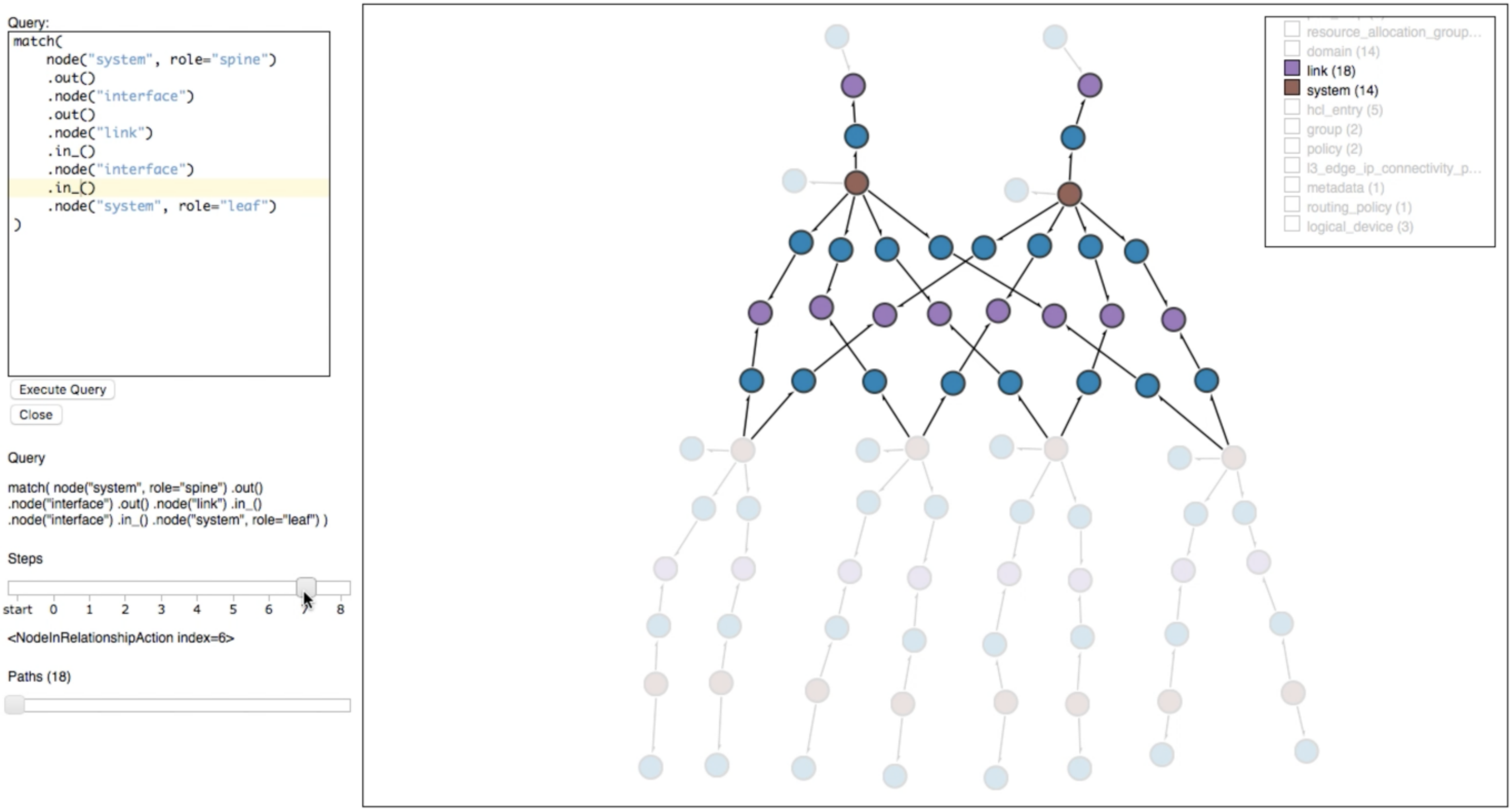
Query

match(node("system", role="spine") .out() .node("interface") .out() .node("link") .in_() .node("interface") .in_() .node("system", role="leaf"))

Steps

<RelationshipSourceAction index=5 type=interface>

Paths (20)



Query:

```
match(
    node("system", role="spine")
    .out()
    .node("interface")
    .out()
    .node("link")
    .in_()
    .node("interface")
    .in_()
    .node("system", role="leaf")
)
```

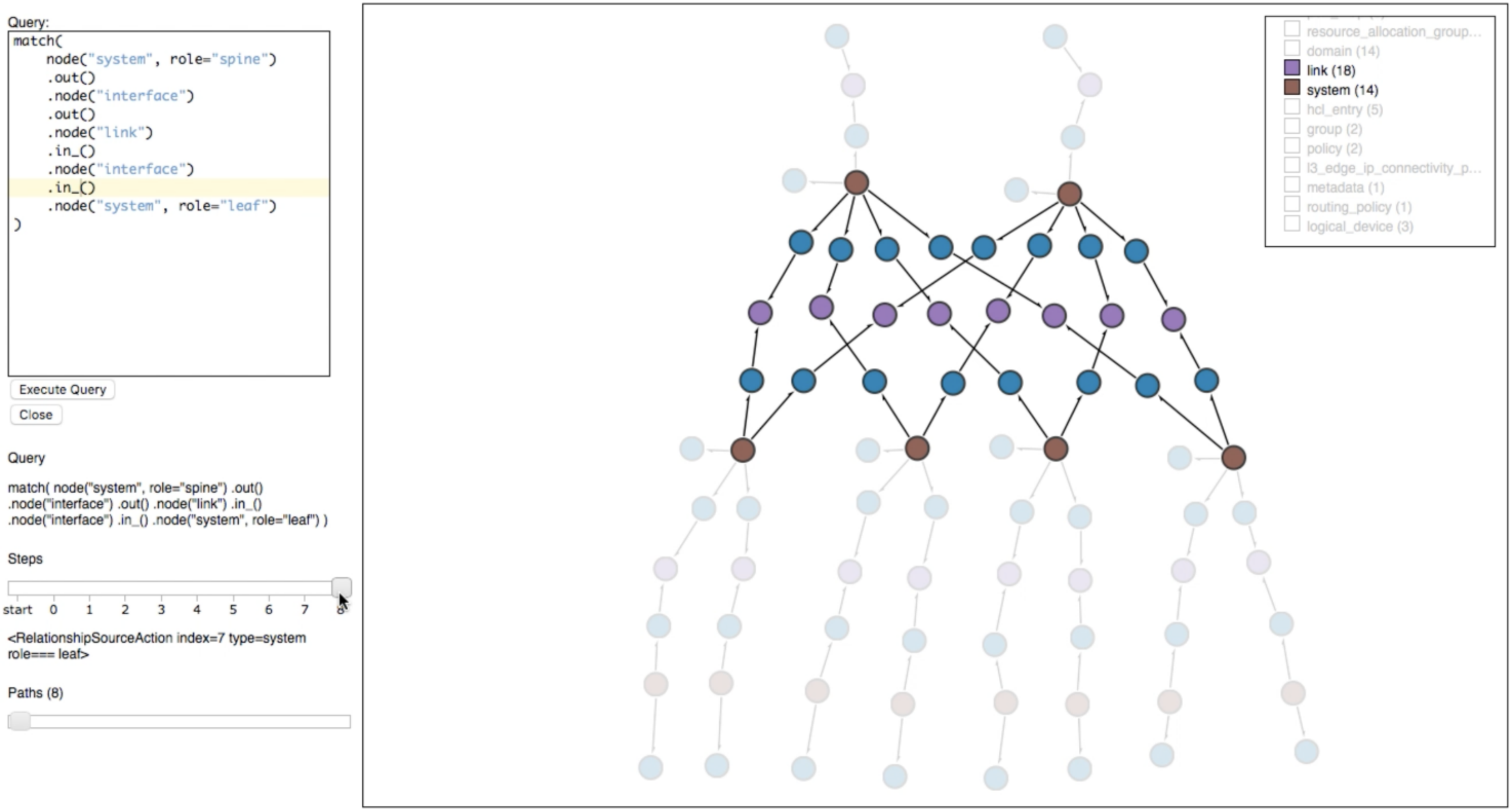
Query

match(node("system", role="spine") .out() .node("interface") .out() .node("link") .in_() .node("interface") .in_() .node("system", role="leaf"))

Steps

<NodeInRelationshipAction index=6>

Paths (18)



Query:

```
match(
    node("system", role="spine")
    .out()
    .node("interface")
    .out()
    .node("link")
    .in_()
    .node("interface")
    .in_()
    .node("system", role="leaf")
)
```

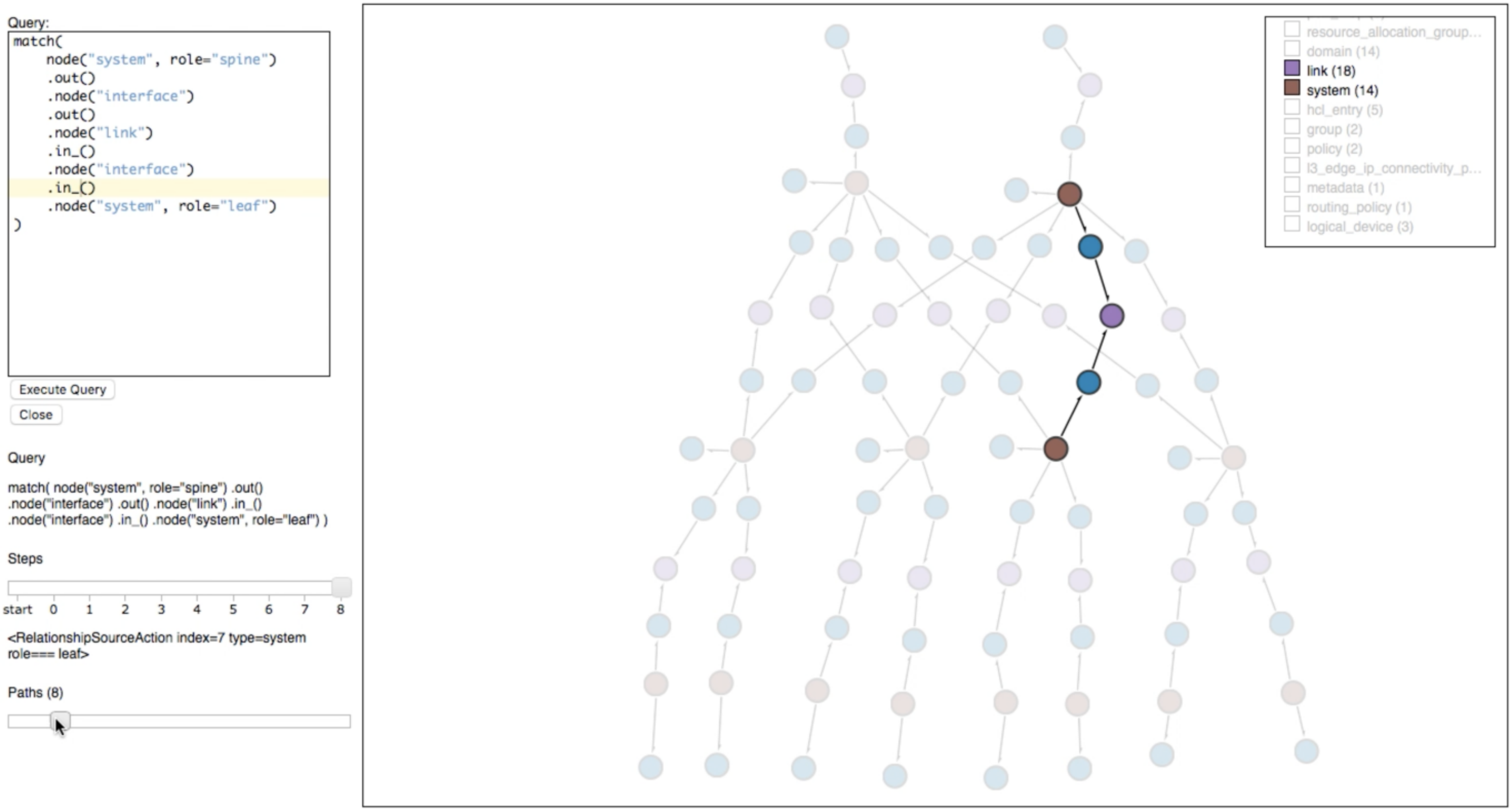
Query

match(node("system", role="spine") .out() .node("interface") .out() .node("link") .in_() .node("interface") .in_() .node("system", role="leaf"))

Steps

<RelationshipSourceAction index=7 type=system role=== leaf>

Paths (8)



Query:

```
match(
    node("system", role="spine")
    .out()
    .node("interface")
    .out()
    .node("link")
    .in_()
    .node("interface")
    .in_()
    .node("system", role="leaf")
)
```

Query

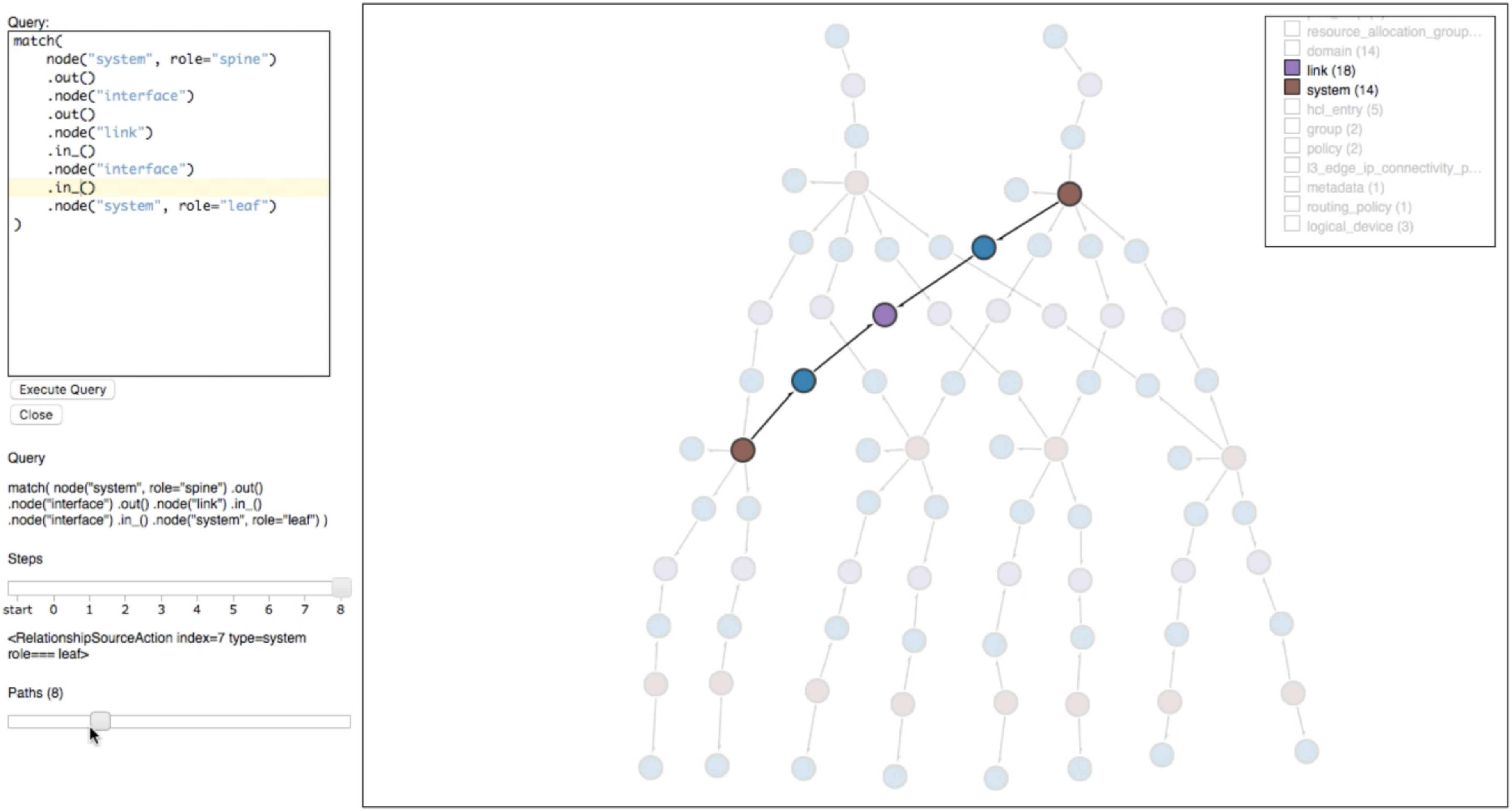
match(node("system", role="spine") .out() .node("interface") .out() .node("link") .in_() .node("interface") .in_() .node("system", role="leaf"))

Steps

<RelationshipSourceAction index=7 type=system role=== leaf>

Paths (8)

apstra



Query:

```
match(
    node("system", role="spine")
    .out()
    .node("interface")
    .out()
    .node("link")
    .in_()
    .node("interface")
    .in_()
    .node("system", role="leaf")
)
```

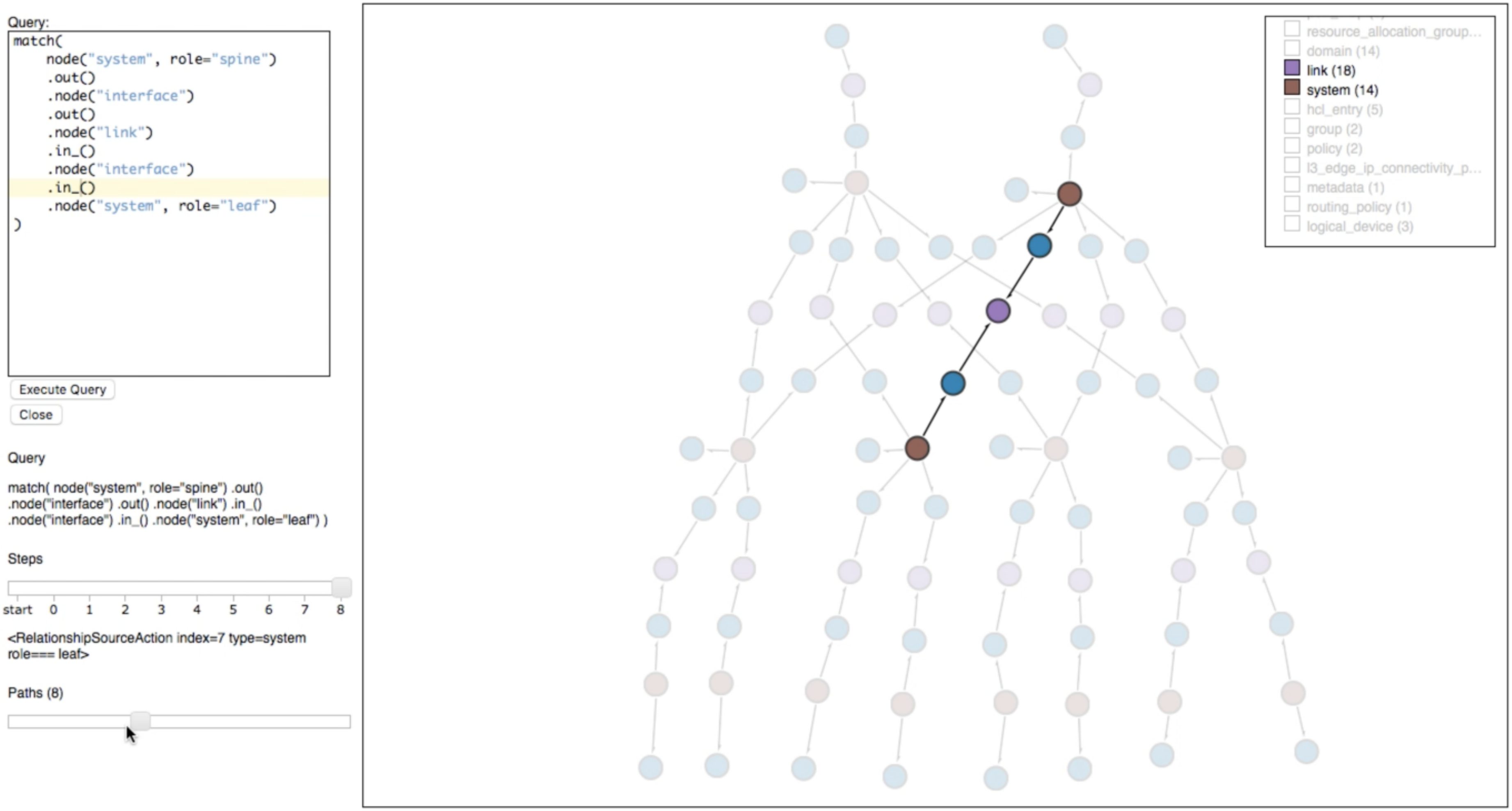
Query

match(node("system", role="spine") .out() .node("interface") .out() .node("link") .in_() .node("interface") .in_() .node("system", role="leaf"))

Steps

<RelationshipSourceAction index=7 type=system role=== leaf>

Paths (8)



Query:

```
match(
    node("system", role="spine")
    .out()
    .node("interface")
    .out()
    .node("link")
    .in_()
    .node("interface")
    .in_()
    .node("system", role="leaf")
)
```

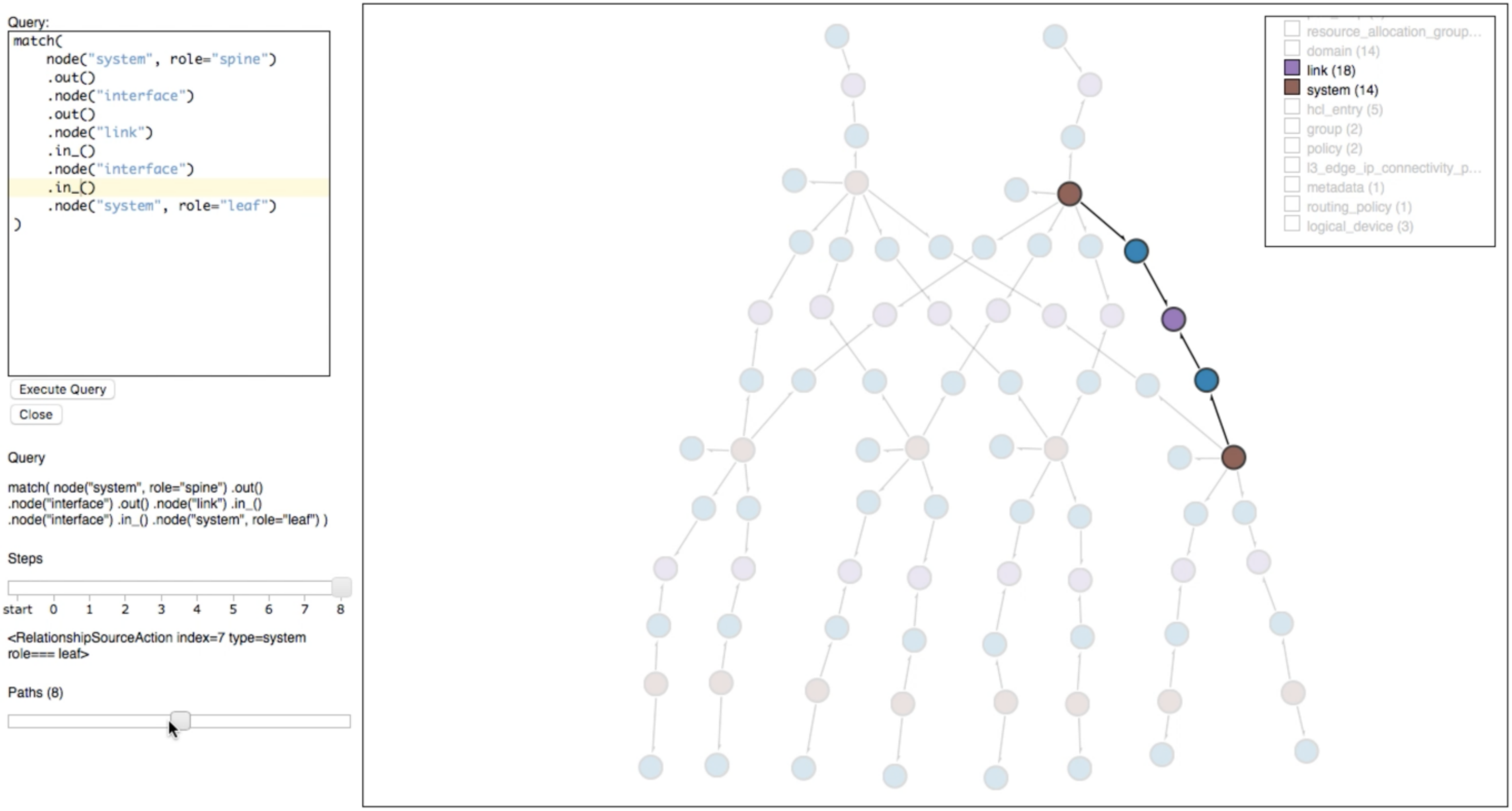
Query

match(node("system", role="spine") .out() .node("interface") .out() .node("link") .in_() .node("interface") .in_() .node("system", role="leaf"))

Steps

<RelationshipSourceAction index=7 type=system role=== leaf>

Paths (8)



Query:

```
match(
    node("system", role="spine")
    .out()
    .node("interface")
    .out()
    .node("link")
    .in_()
    .node("interface")
    .in_()
    .node("system", role="leaf")
)
```

Query

match(node("system", role="spine") .out() .node("interface") .out() .node("link") .in_() .node("interface") .in_() .node("system", role="leaf"))

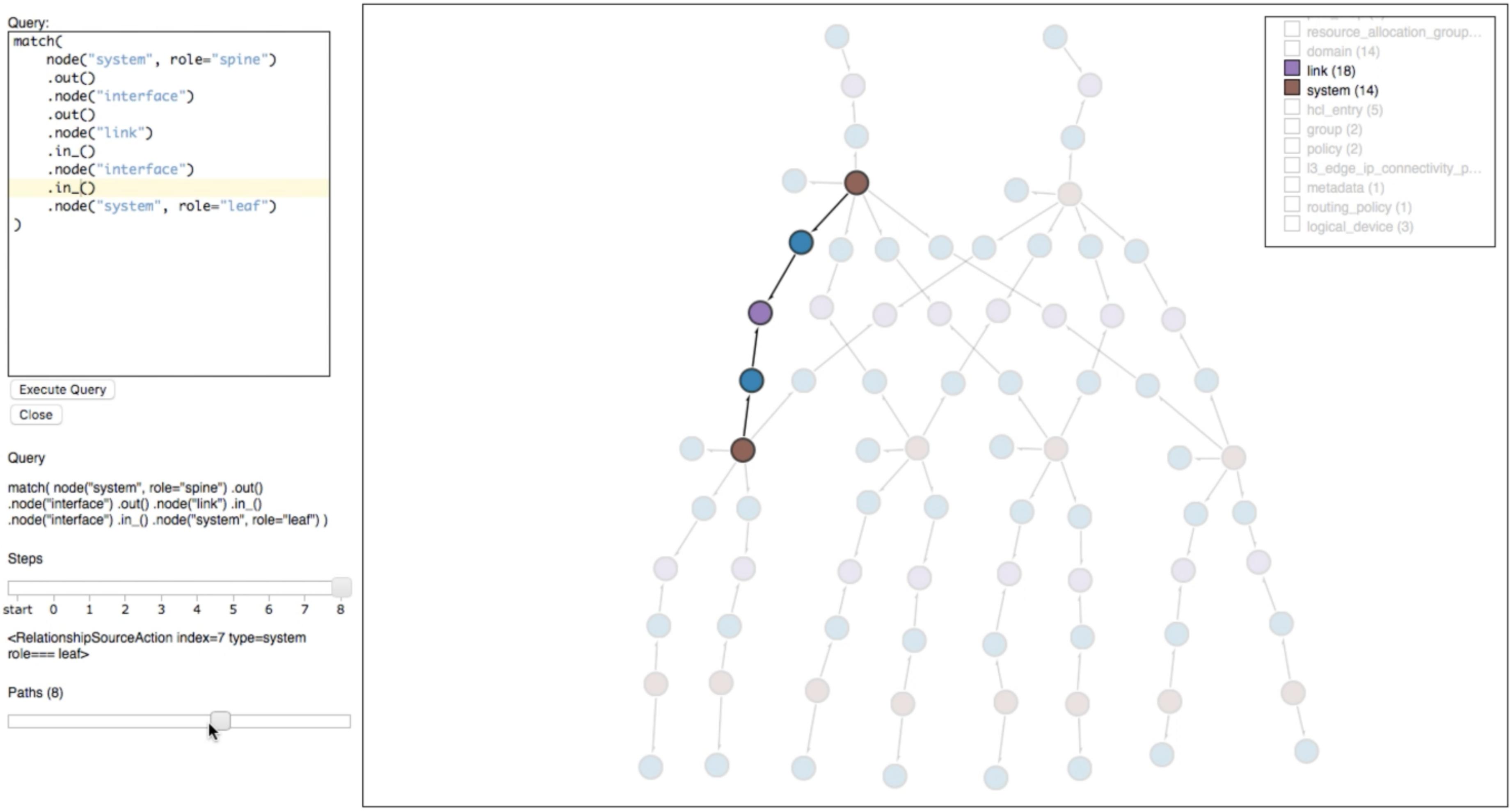
Steps

<RelationshipSourceAction index=7 type=system role=== leaf>

Paths (8)

□ resource_allocation_group...
□ domain (14)
🟪 link (18)
🟫 system (14)
□ hcl_entry (5)
□ group (2)
□ policy (2)
□ l3_edge_ip_connectivity_p...
□ metadata (1)
□ routing_policy (1)
□ logical_device (3)

apstra



Query:

```
match(
    node("system", role="spine")
    .out()
    .node("interface")
    .out()
    .node("link")
    .in_()
    .node("interface")
    .in_()
    .node("system", role="leaf")
)
```

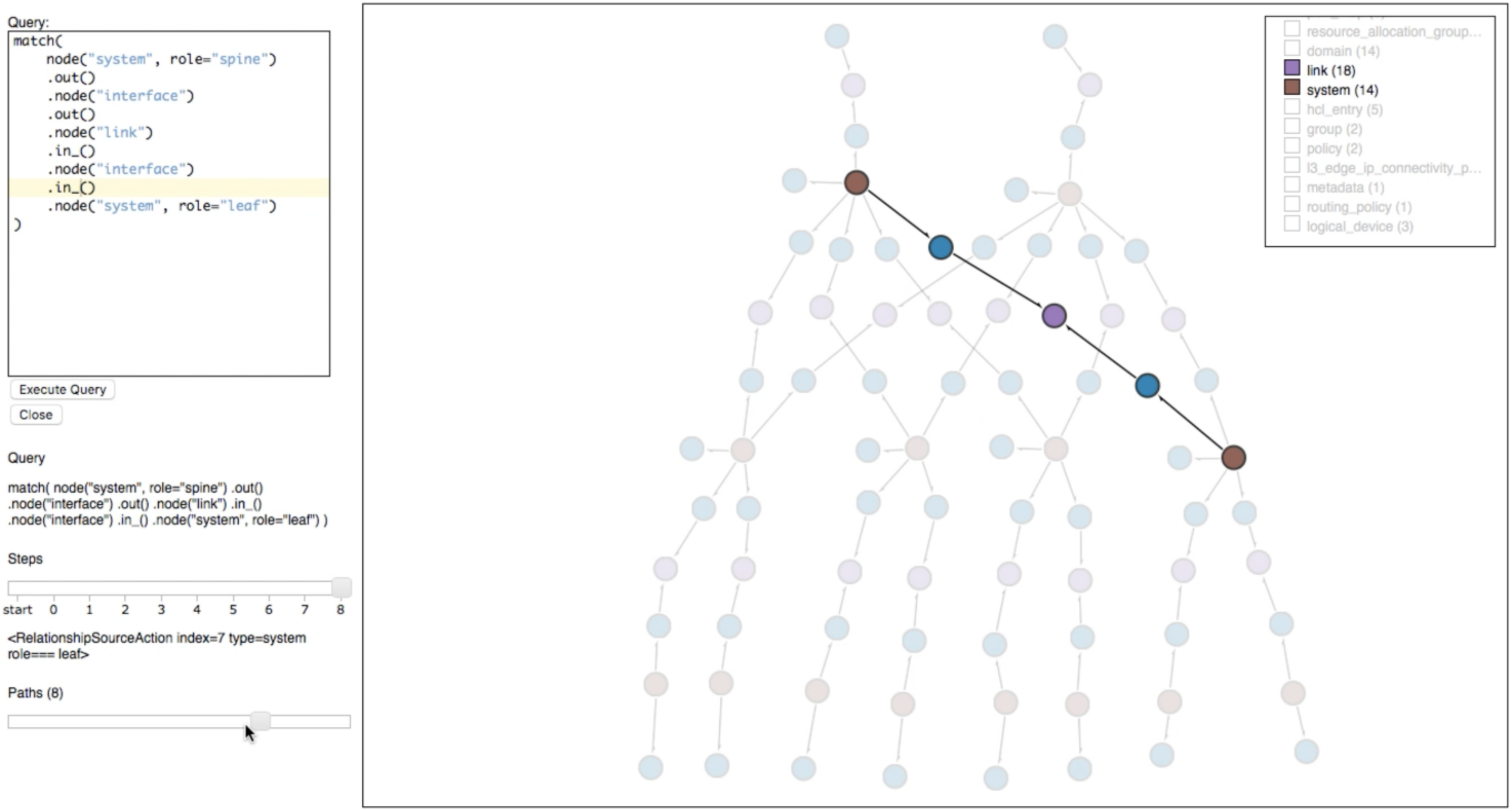
Query

match(node("system", role="spine") .out() .node("interface") .out() .node("link") .in_() .node("interface") .in_() .node("system", role="leaf"))

Steps

<RelationshipSourceAction index=7 type=system role=== leaf>

Paths (8)



Query:

```
match(
    node("system", role="spine")
    .out()
    .node("interface")
    .out()
    .node("link")
    .in_()
    .node("interface")
    .in_()
    .node("system", role="leaf")
)
```

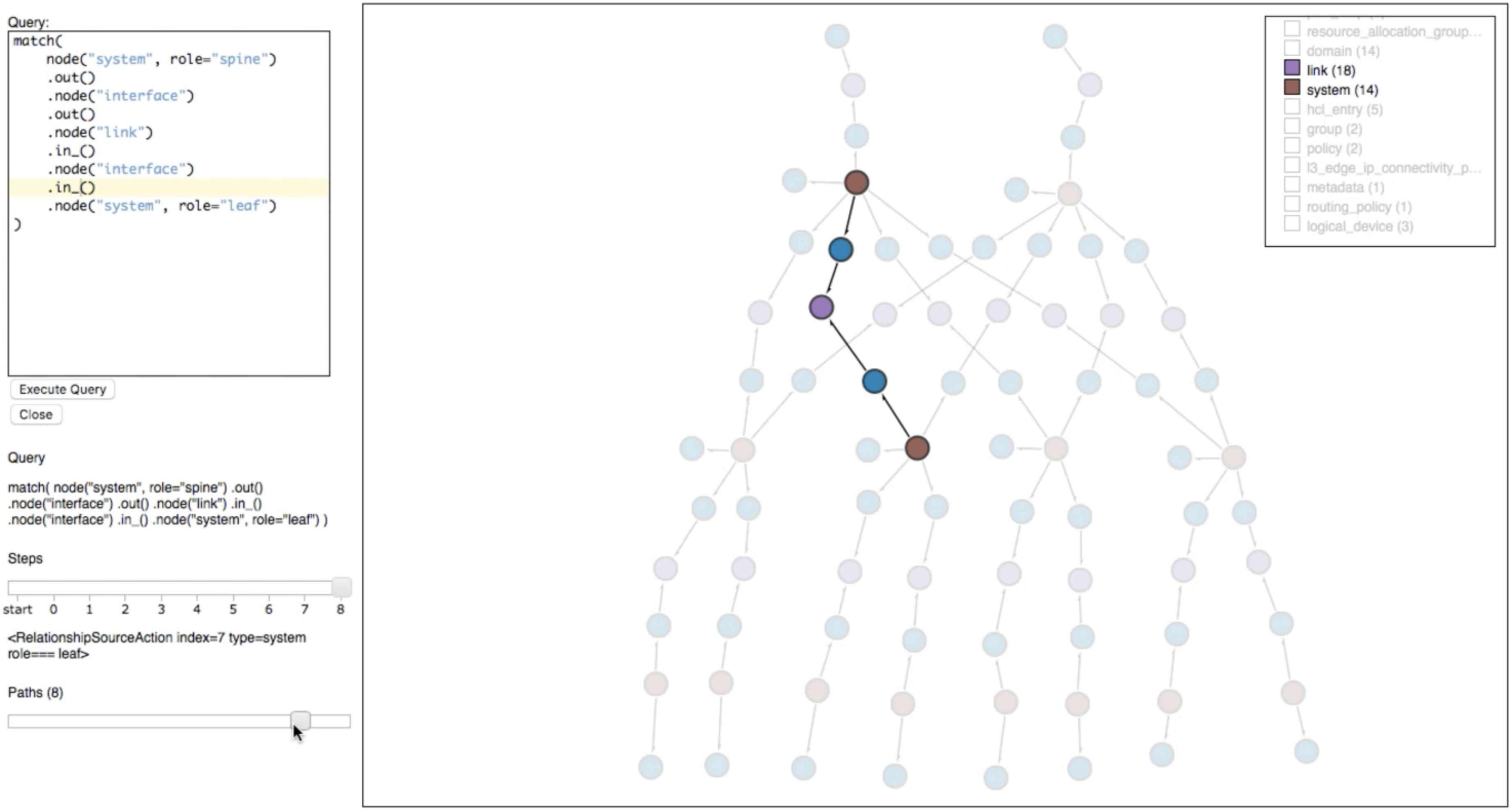
Query

match(node("system", role="spine") .out() .node("interface") .out() .node("link") .in_() .node("interface") .in_() .node("system", role="leaf"))

Steps

<RelationshipSourceAction index=7 type=system role=== leaf>

Paths (8)



Query:

```
match(
    node("system", role="spine")
    .out()
    .node("interface")
    .out()
    .node("link")
    .in_()
    .node("interface")
    .in_()
    .node("system", role="leaf")
)
```

Execute Query

Close

Query

match(node("system", role="spine") .out() .node("interface") .out() .node("link") .in_() .node("interface") .in_() .node("system", role="leaf"))

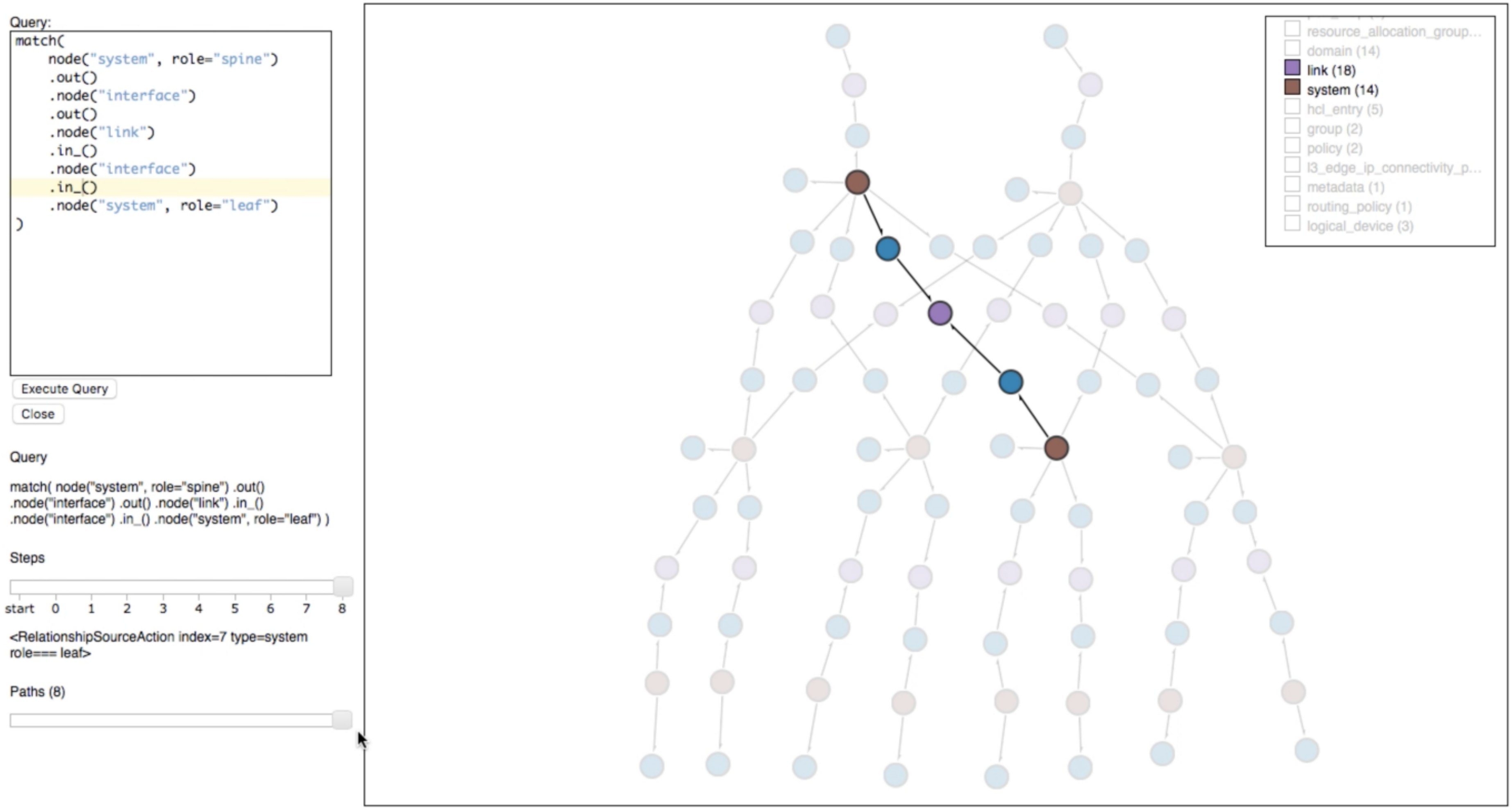
Steps

start 0 1 2 3 4 5 6 7 8

<RelationshipSourceAction index=7 type=system role=== leaf>

Paths (8)

apstra



Query:

```
match(
    node("system", role="spine")
    .out()
    .node("interface")
    .out()
    .node("link")
    .in_()
    .node("interface")
    .in_()
    .node("system", role="leaf")
)
```

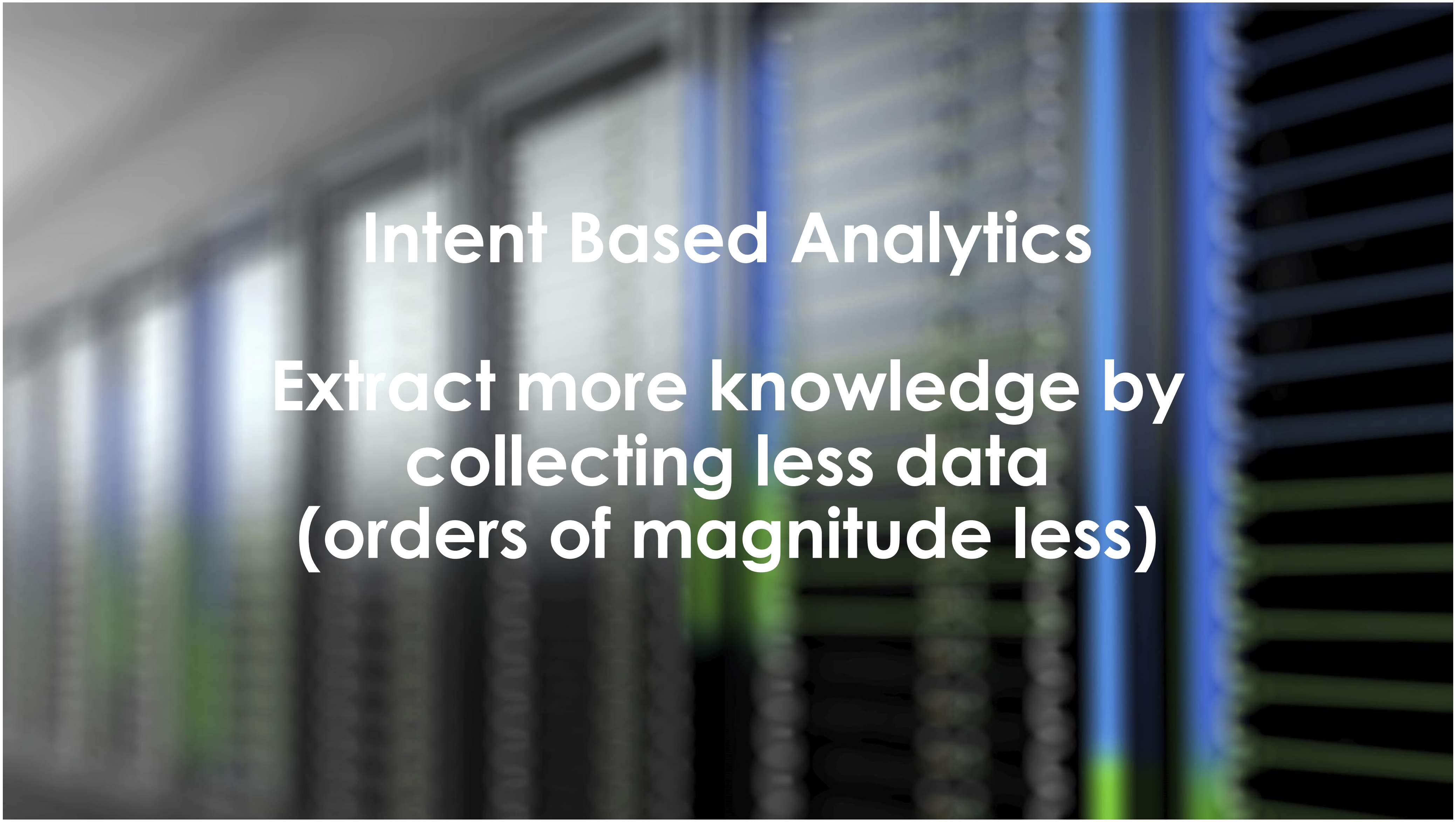
Query

match(node("system", role="spine") .out() .node("interface") .out() .node("link") .in_() .node("interface") .in_() .node("system", role="leaf"))

Steps

<RelationshipSourceAction index=7 type=system role=== leaf>

Paths (8)

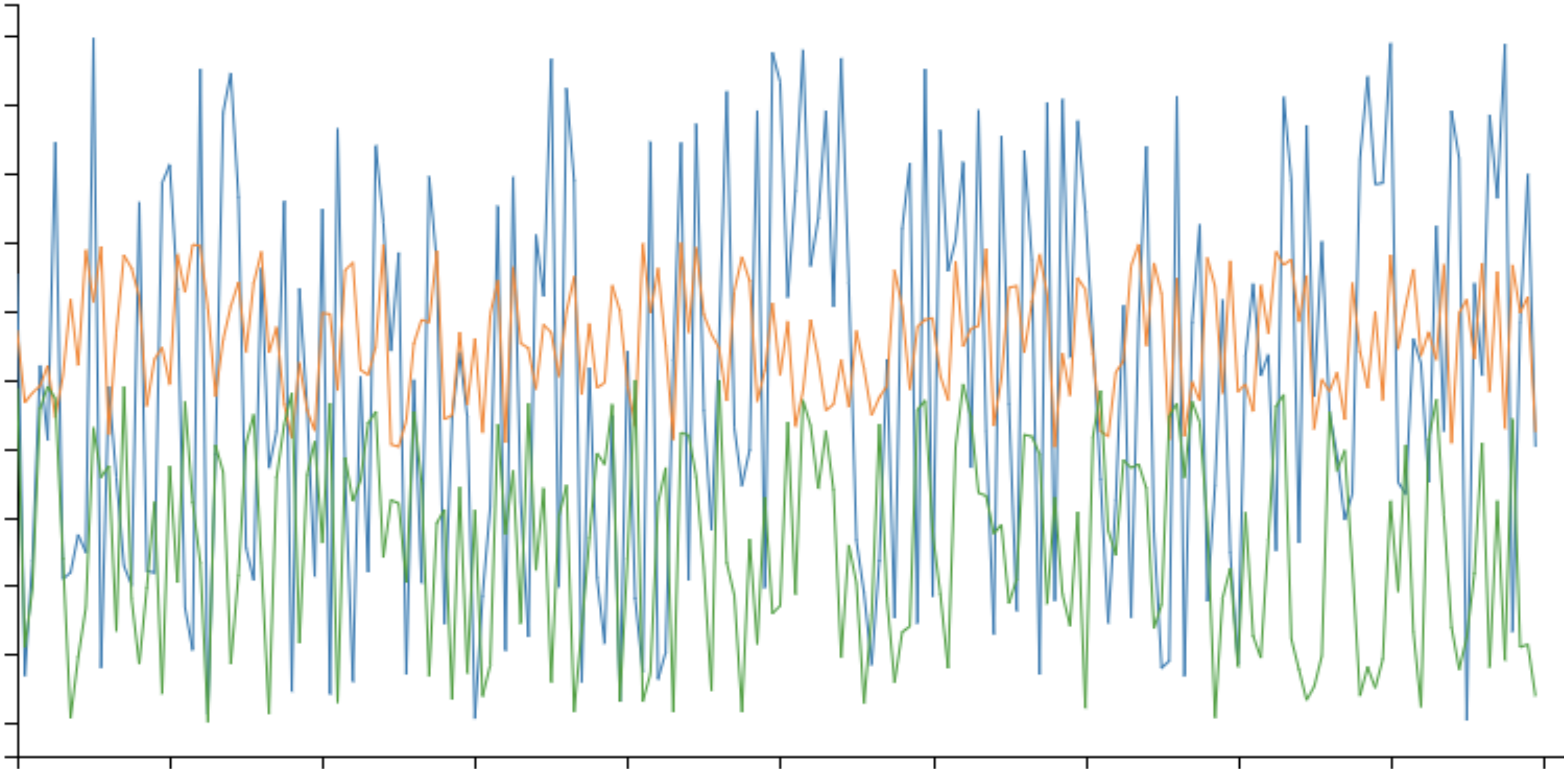


Intent Based Analytics

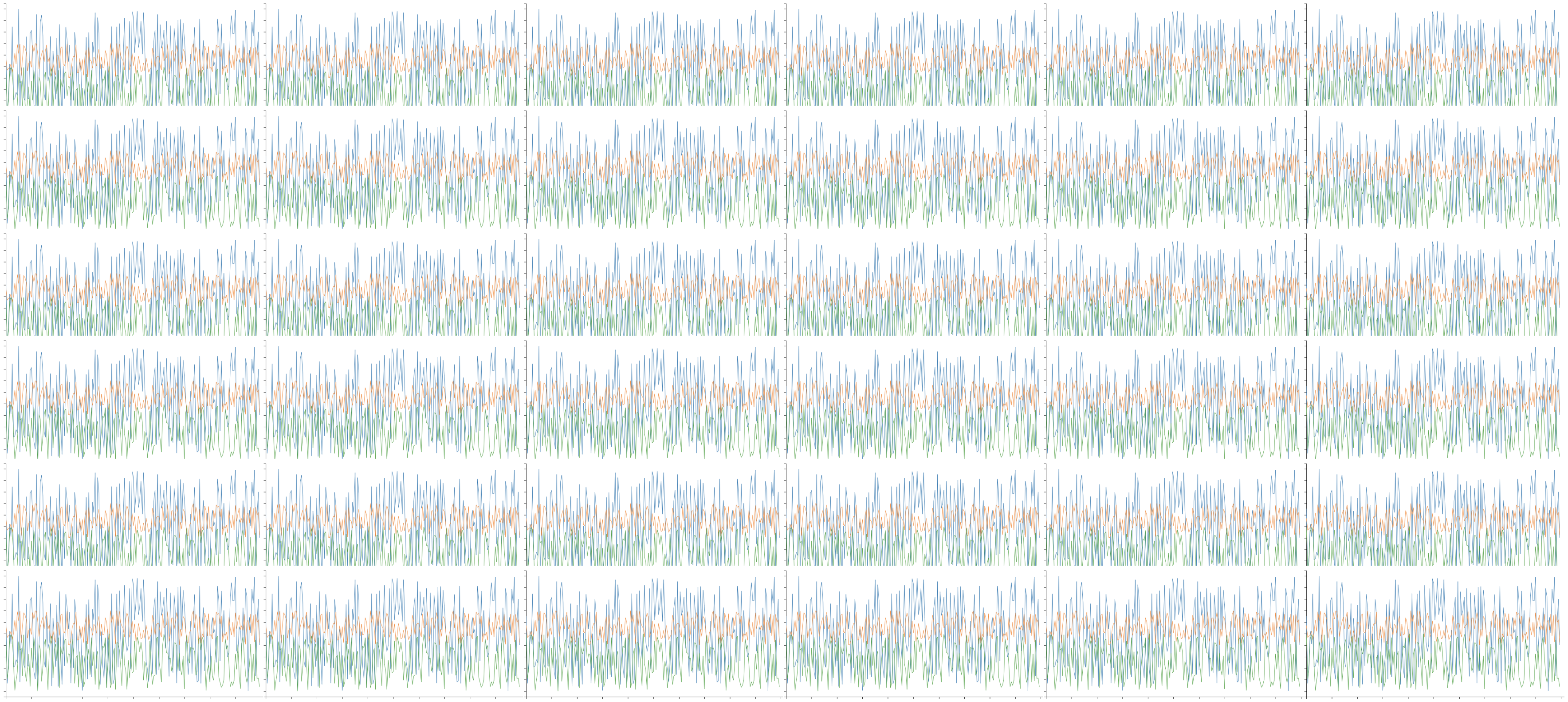
Extract more knowledge by collecting less data (orders of magnitude less)

Was I looking for something?

Gathering high def telemetry



For all my leaf1 interfaces

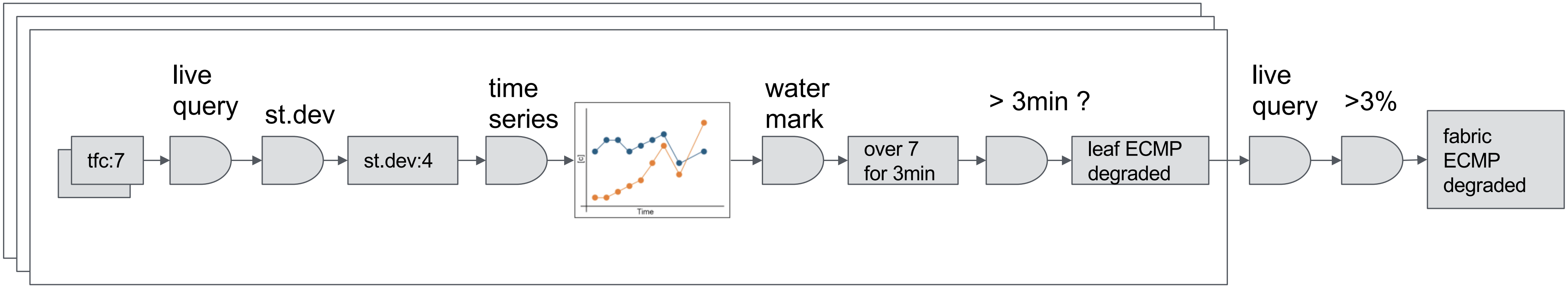


For all my leafs

So that I have insight

Question: Is my fabric ECMP imbalanced?

IBA : ECMP fabric health (load sharing across fabric links)



tfc:7

live query

st.dev

st.dev:4

time series

water mark

over 7 for 3min

> 3min ?

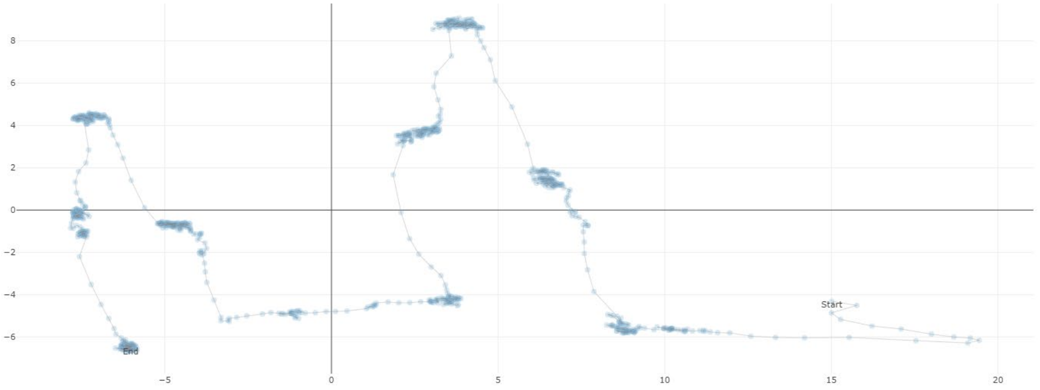
leaf ECMP degraded

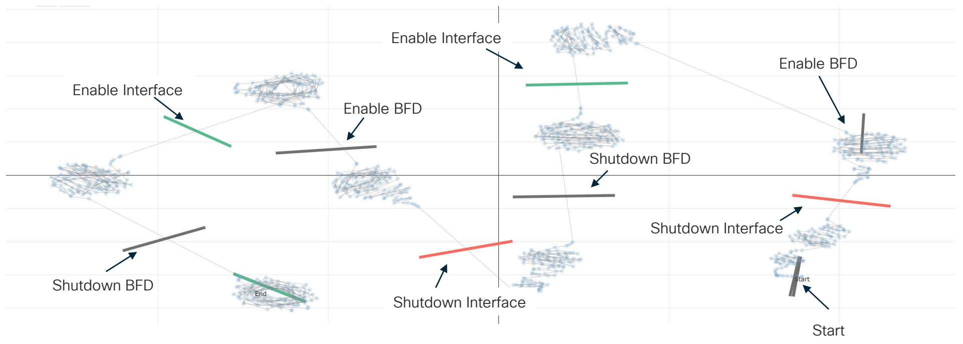
live query

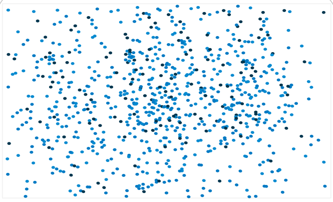
>3%

fabric ECMP degraded

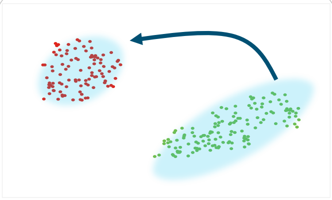
Would ML be helpful?



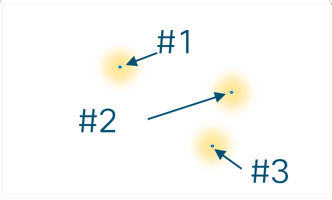




Holistic view:
Collect “all” counters
all the time.



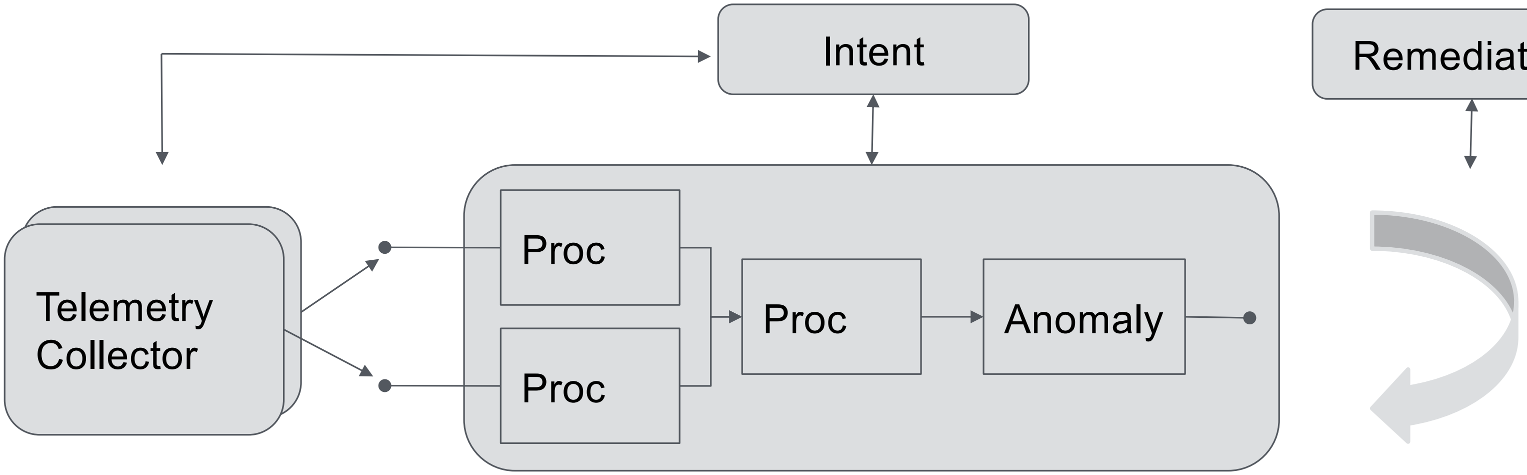
Macroscopic view:
Catch interesting
state changes.
Dim.-Redux, Cluster.



Microscopic view:
Choose counters
which best describe
the state change.

Source: BRKOPS-3825, Frank Brockners

IBA – context aware analytics



Declaratively specified, definition is de-coupled from instantiation

Once specified, is in constant sync with intent

Extracts knowledge out of the raw telemetry – context drives the content

New telemetry is “wired-in”

Conclusion

- Basic automation, while hot topic - is the first and easiest step in the IBN journey

- Single source of truth is mandatory for an IBN system to be able to reason about any change

- Day 2 operations @scale:
    - context aware continues validation
    - dealing with changes
    - configuration drift
    - remediation

is the most complicated area of technologies to deal with!

Questions

Thank You!

www.apstra.com

@ApstraInc

https://www.linkedin.com/company/apstra

https://www.facebook.com/apstrainc/

apstra®