Intent Based Networking - the technology

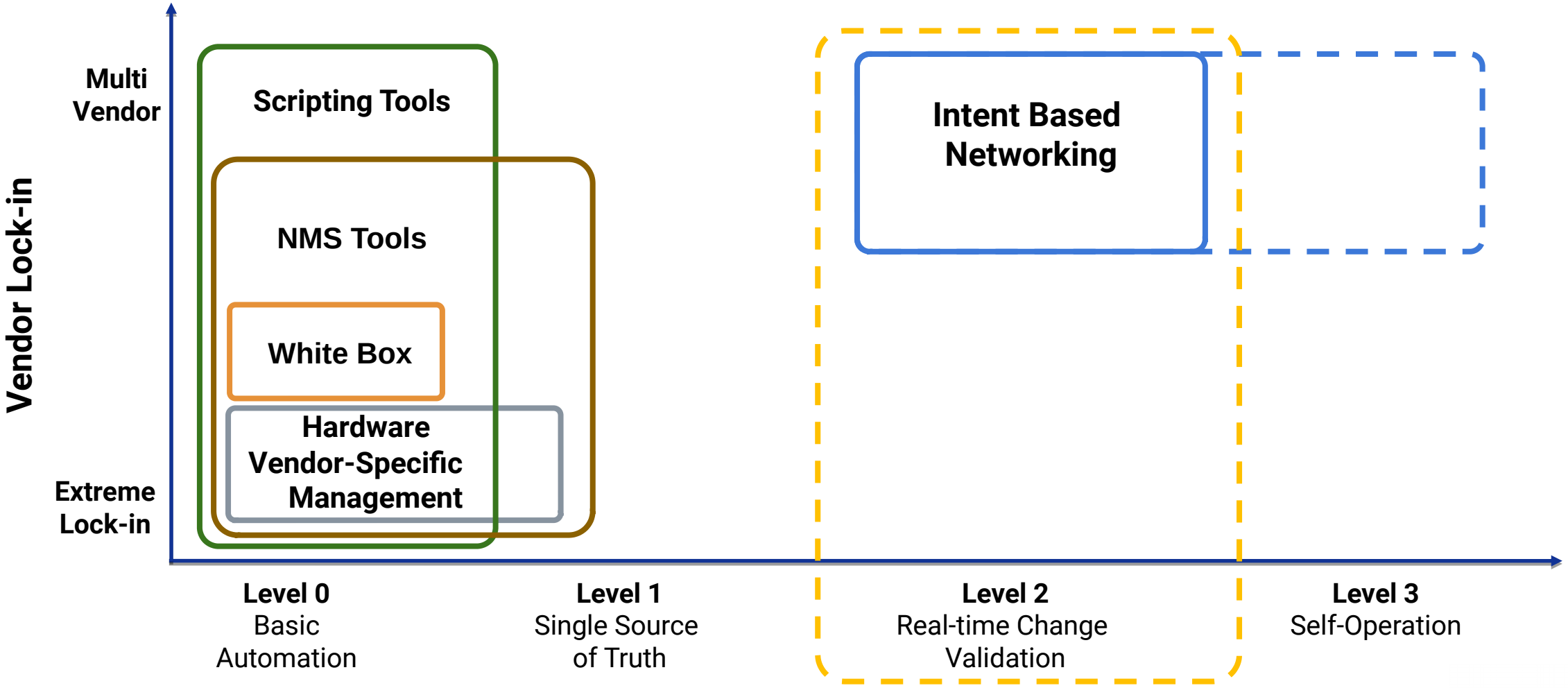
Jeff Tantsura
Head of Networking Strategy @Apstra

apstra

Why nmrg?

- nmrg is the home for IBN
- Need to define how an IBN system interacts with the world:
  - SouthBound
    - IETF/OC did a great job on YANG models, Netconf, Restconf, gRPC
    - Fallback to “native” API’s/CLI
  - Northbound (intent consumption) is not well defined, requires research and eventually standardization

IBN Landscape



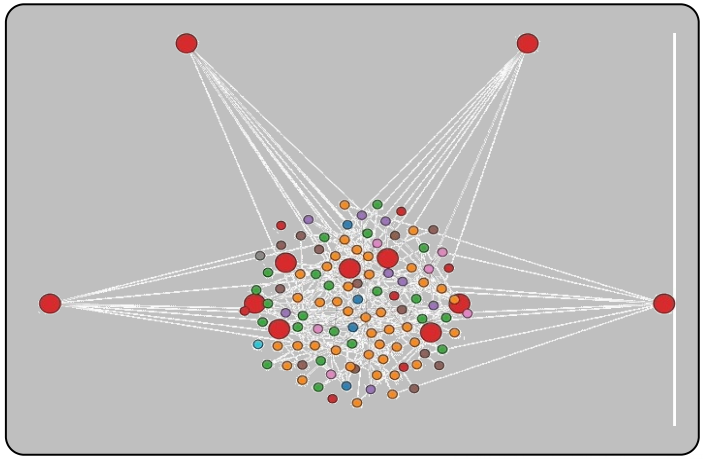
apstra

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*”



apstra

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
  - Unplanned change – impact of the change, difference between intended and operational states, how to get to intended state

apstra

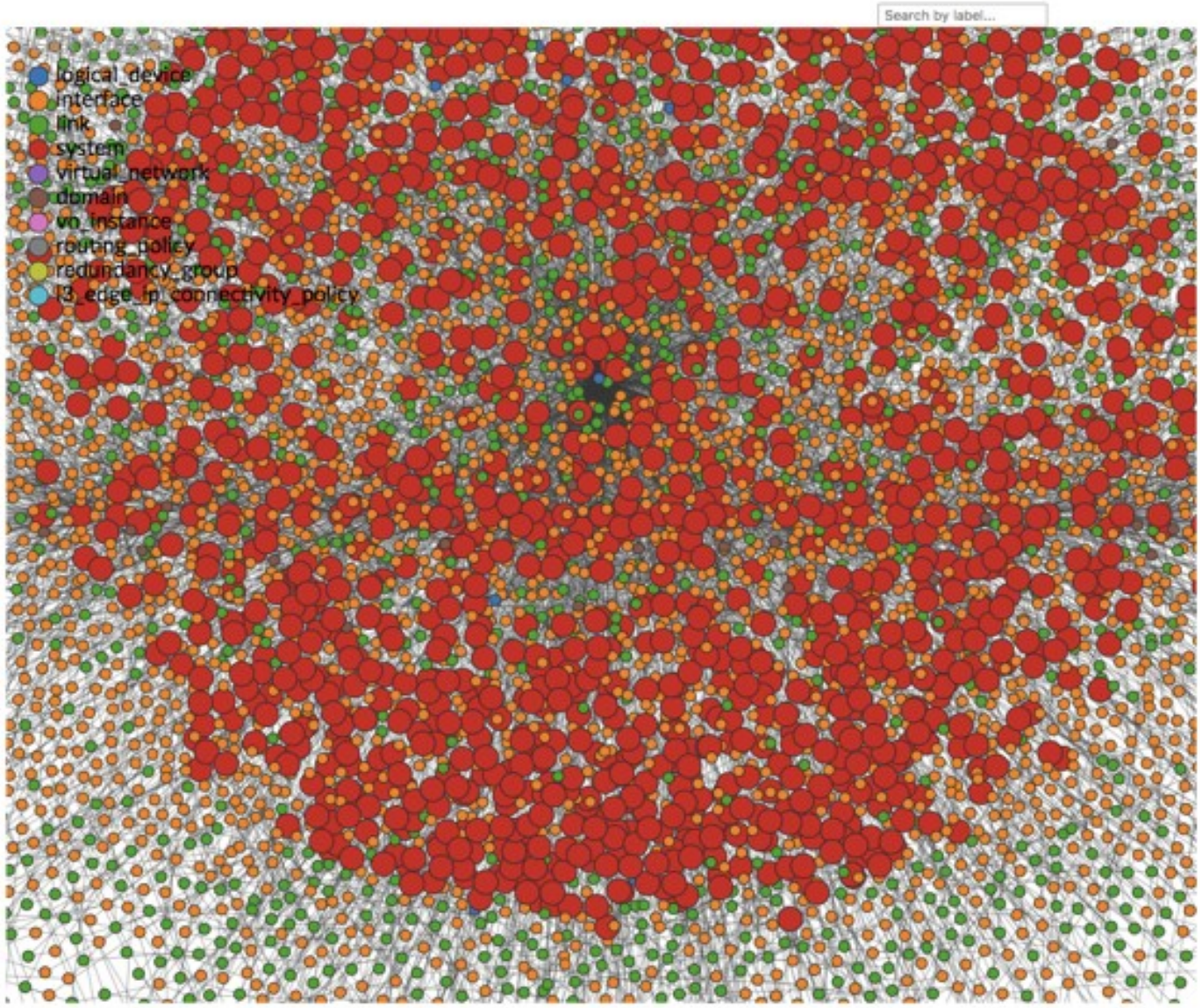
Architectural Goals of IBN

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 (IBA)
  - highly optimized SNR (signal to noise ratio) in analytics

apstra

Dealing With Scale?



apstra

Composition

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia

Article Talk

Function composition (computer science)

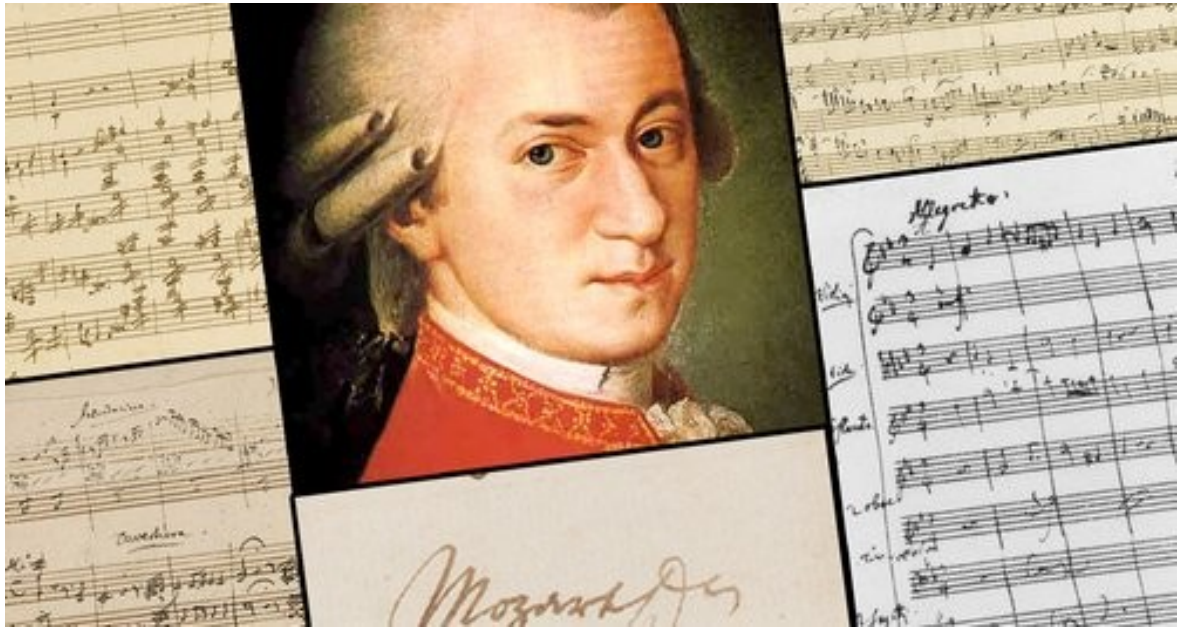
From Wikipedia, the free encyclopedia

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.



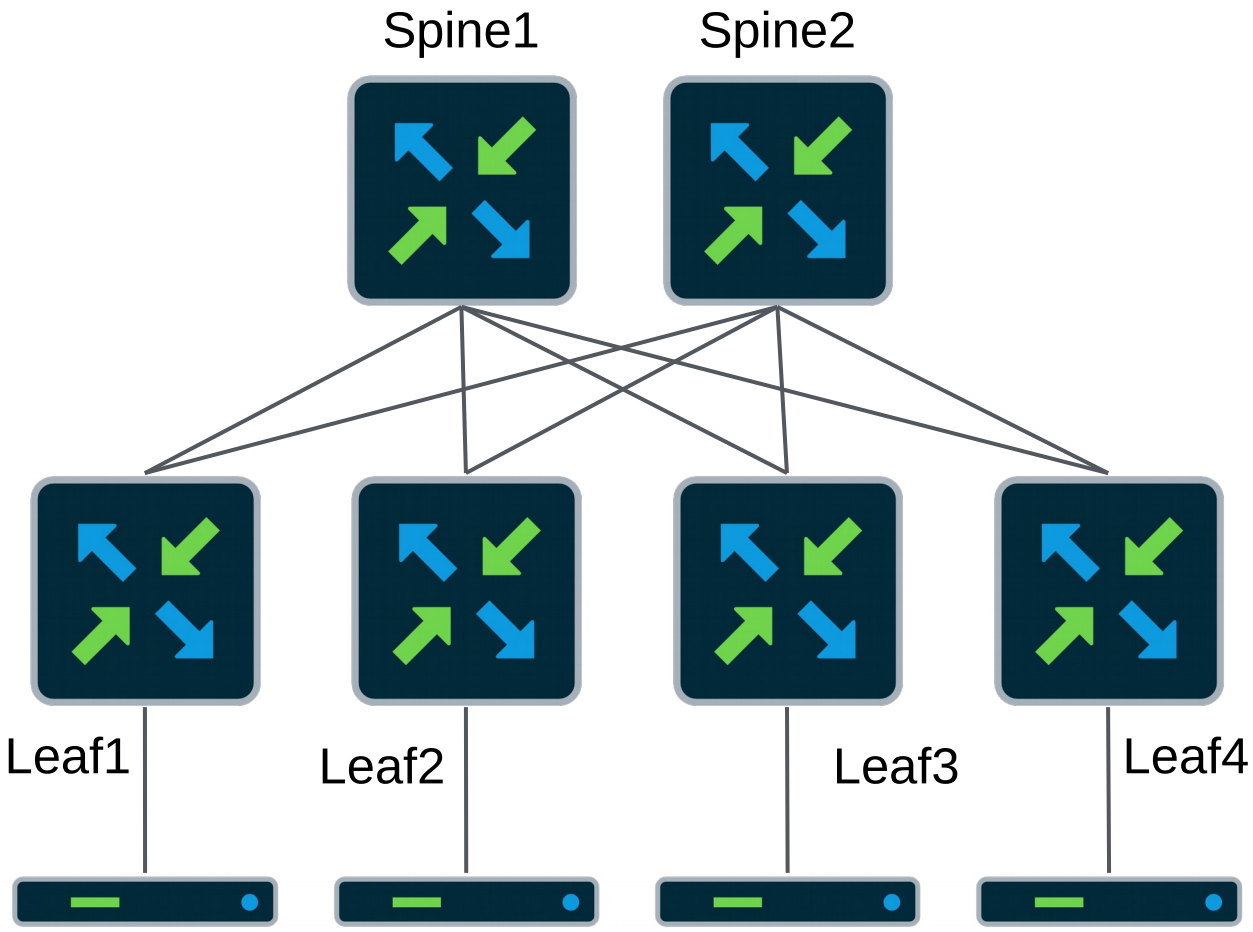
apstra

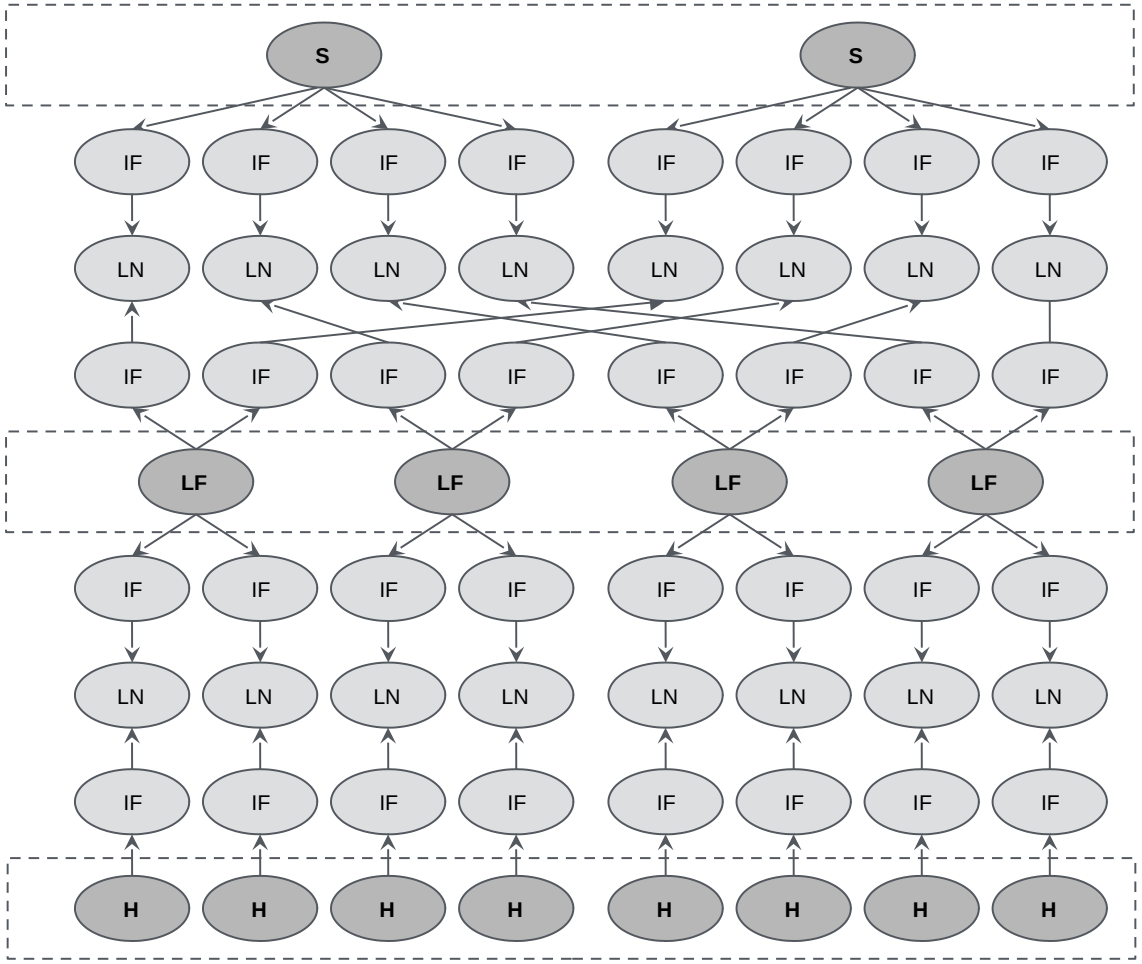
Why model a graph?

- 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

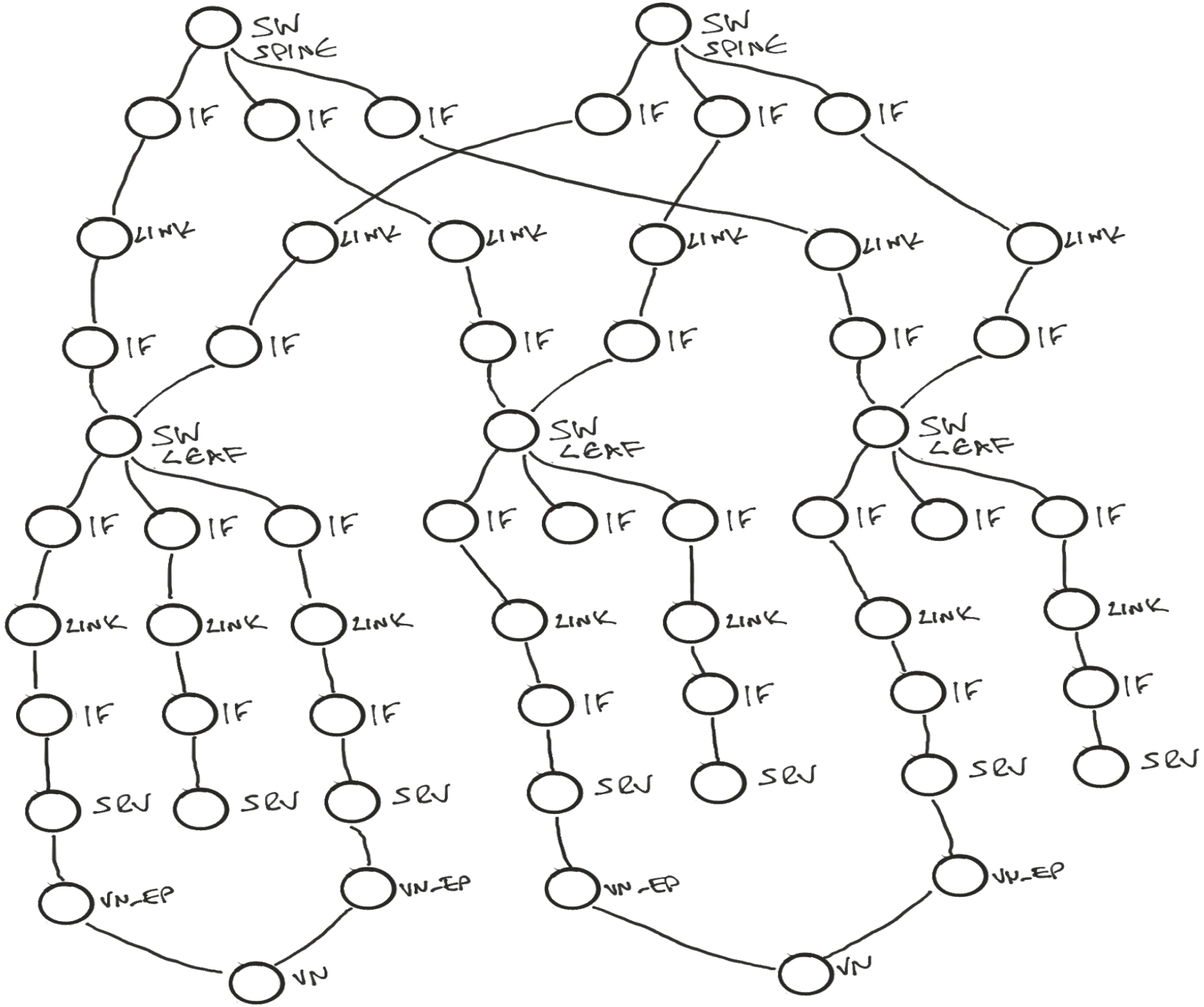
Intent-> Graph composition

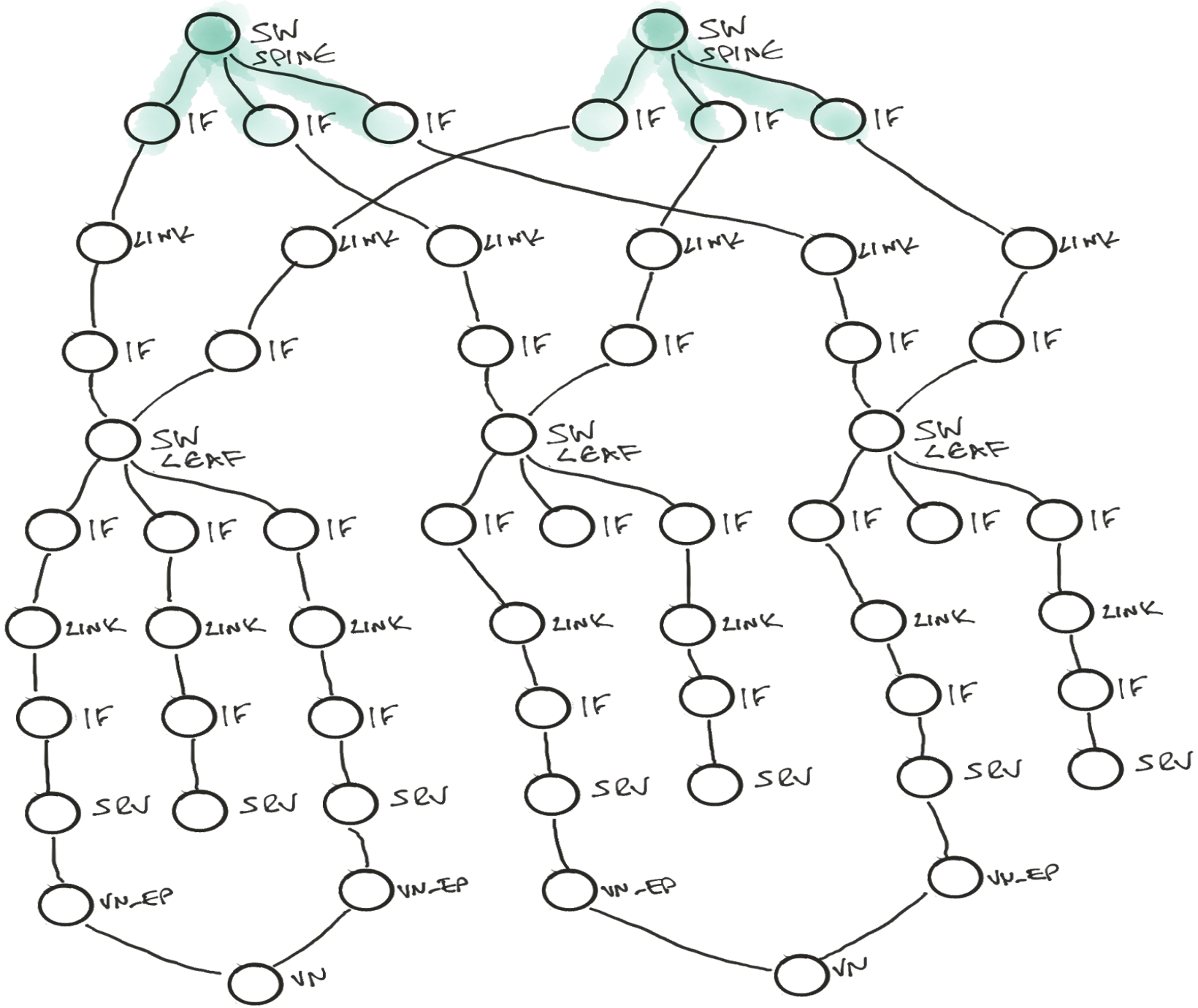




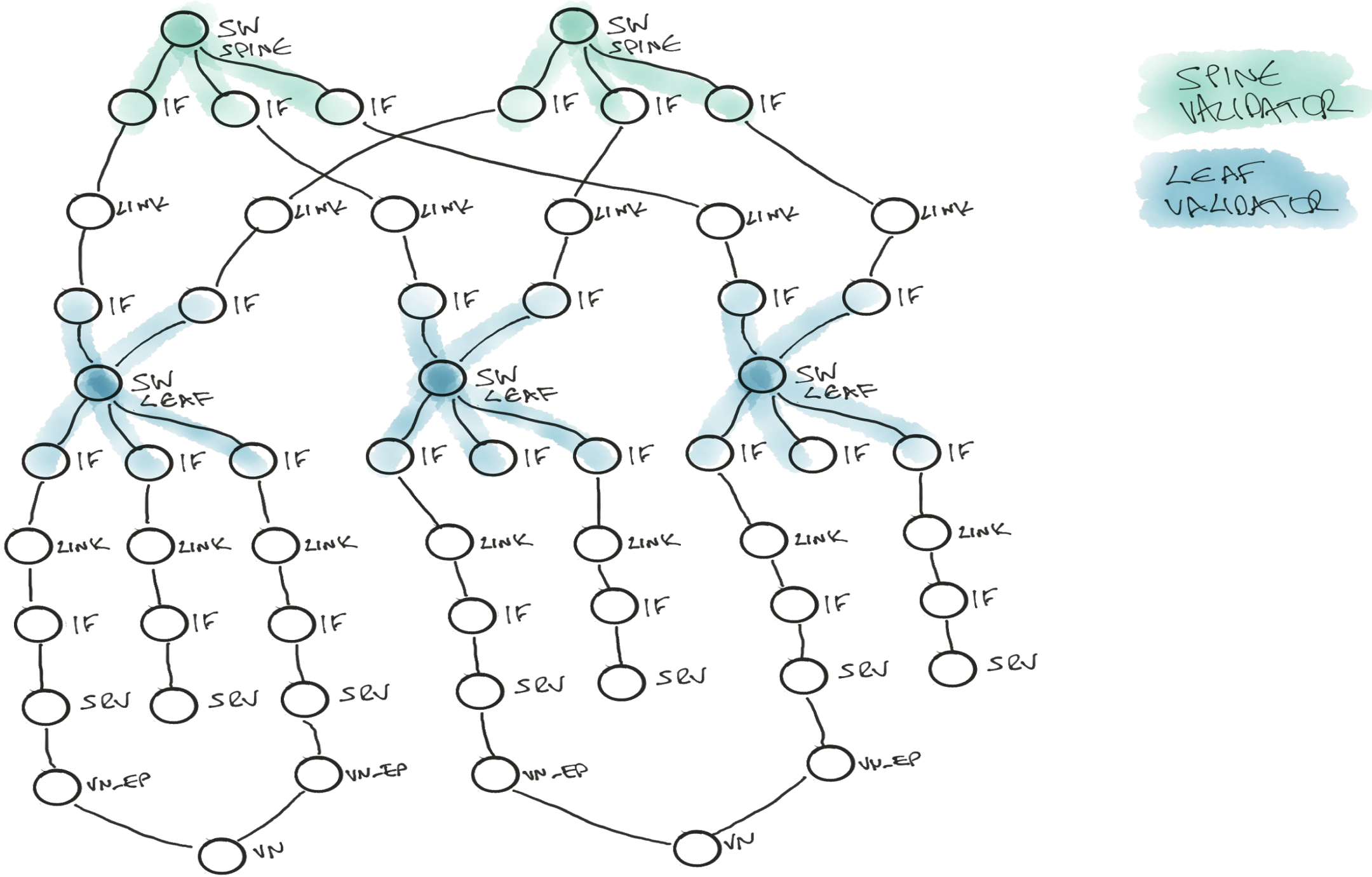
apstra

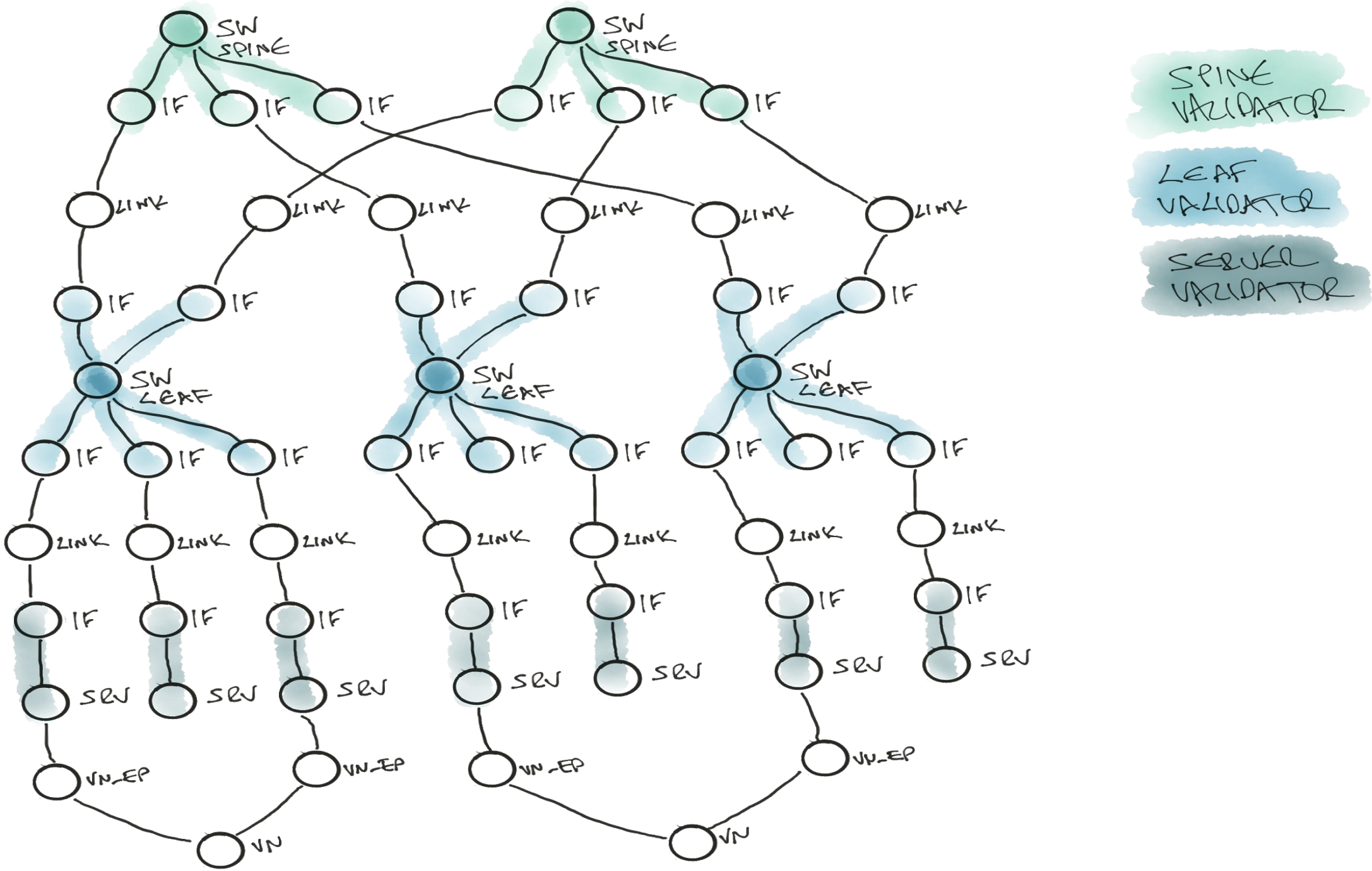
Function composition

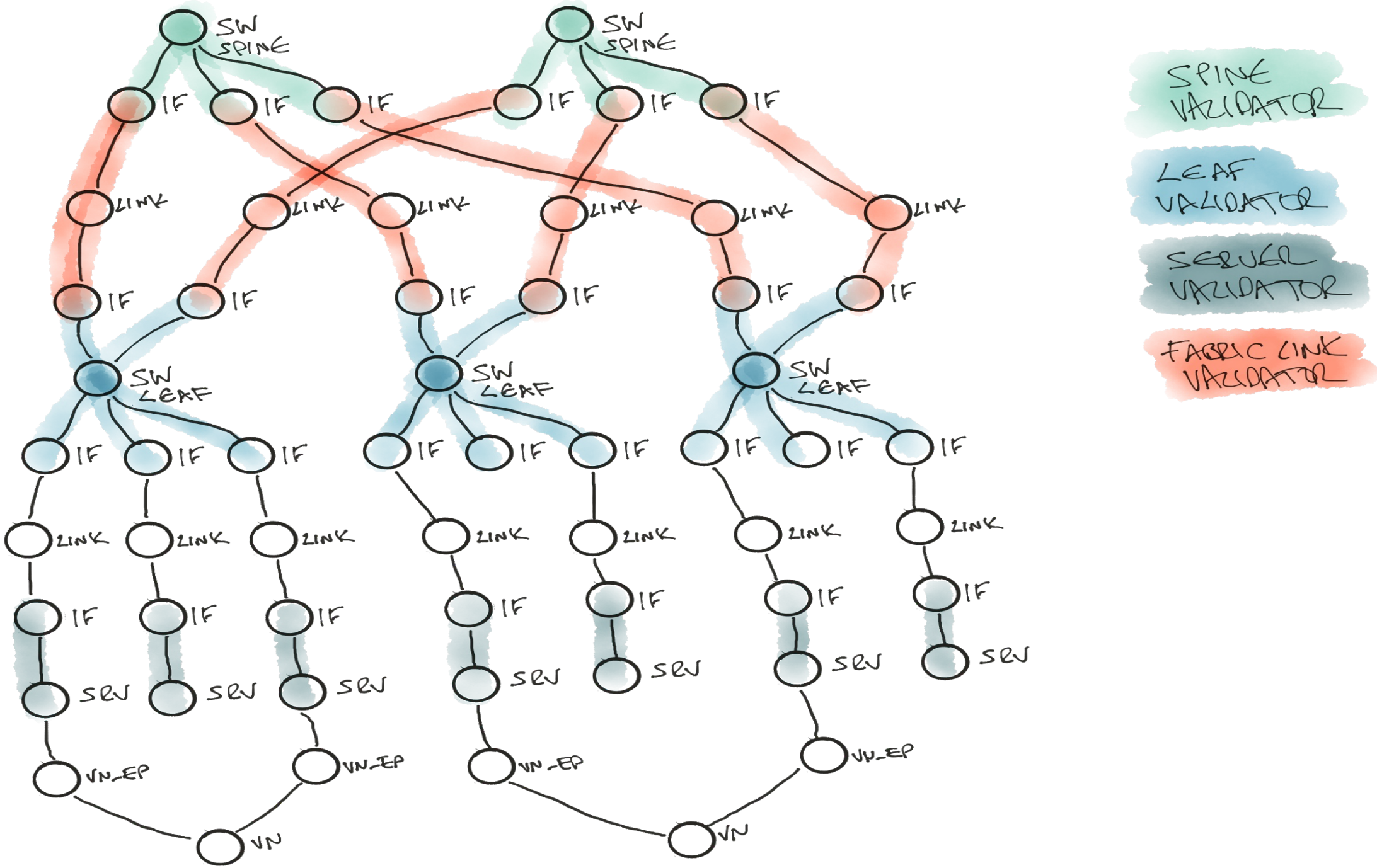

SW SPINE
SW SPINE
IF
IF
IF
IF
IF
IF
LINK
LINK
LINK
LINK
LINK
LINK
IF
IF
IF
IF
IF
IF
SW LEAF
SW LEAF
SW LEAF
IF
IF
IF
IF
IF
IF
IF
IF
IF
LINK
LINK
LINK
LINK
LINK
LINK
LINK
IF
IF
IF
IF
IF
IF
IF
SRV
SRV
SRV
SRV
SRV
SRV
SRV
VN_EP
VN_EP
VN_EP
VN_EP
VN
VN

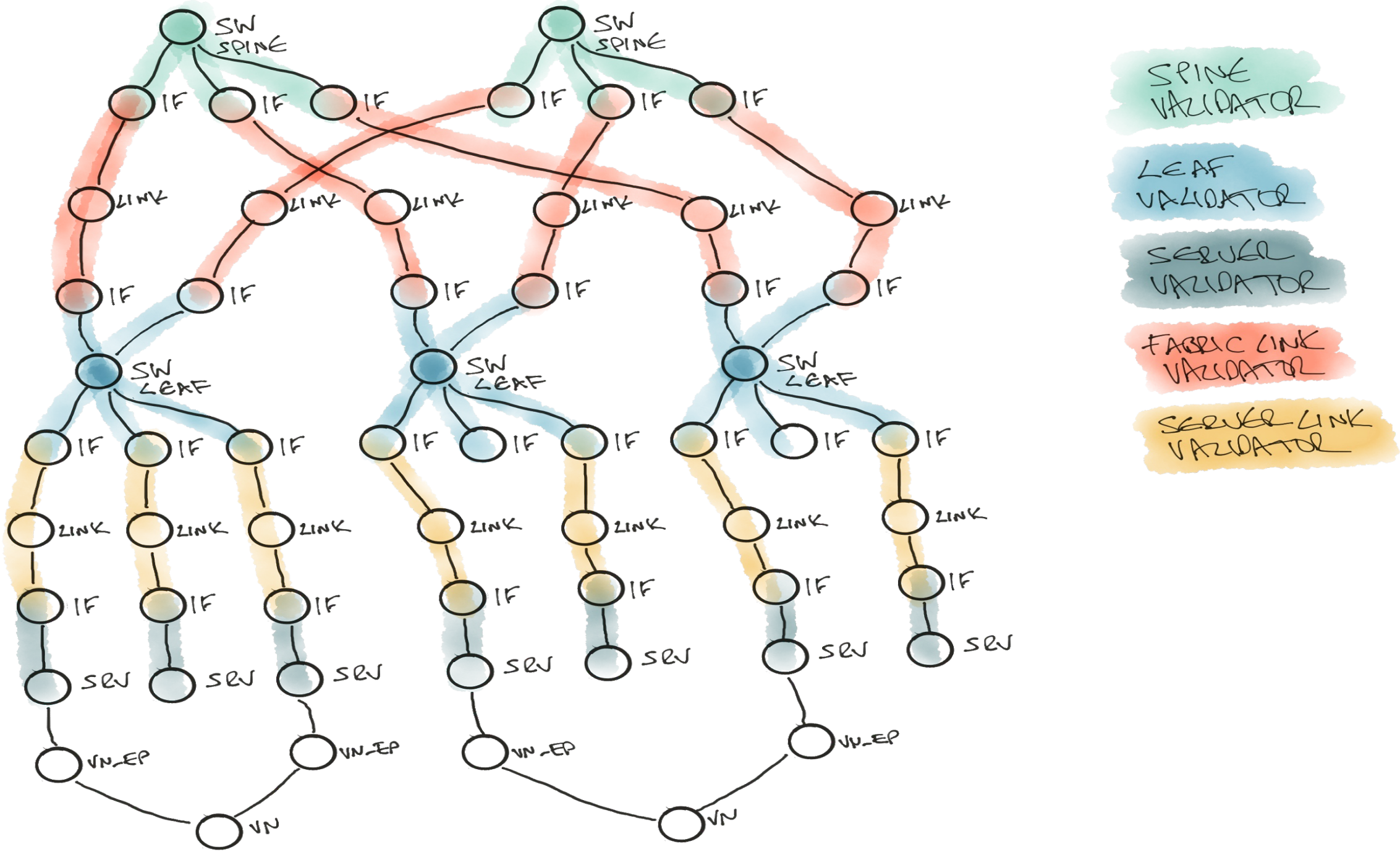


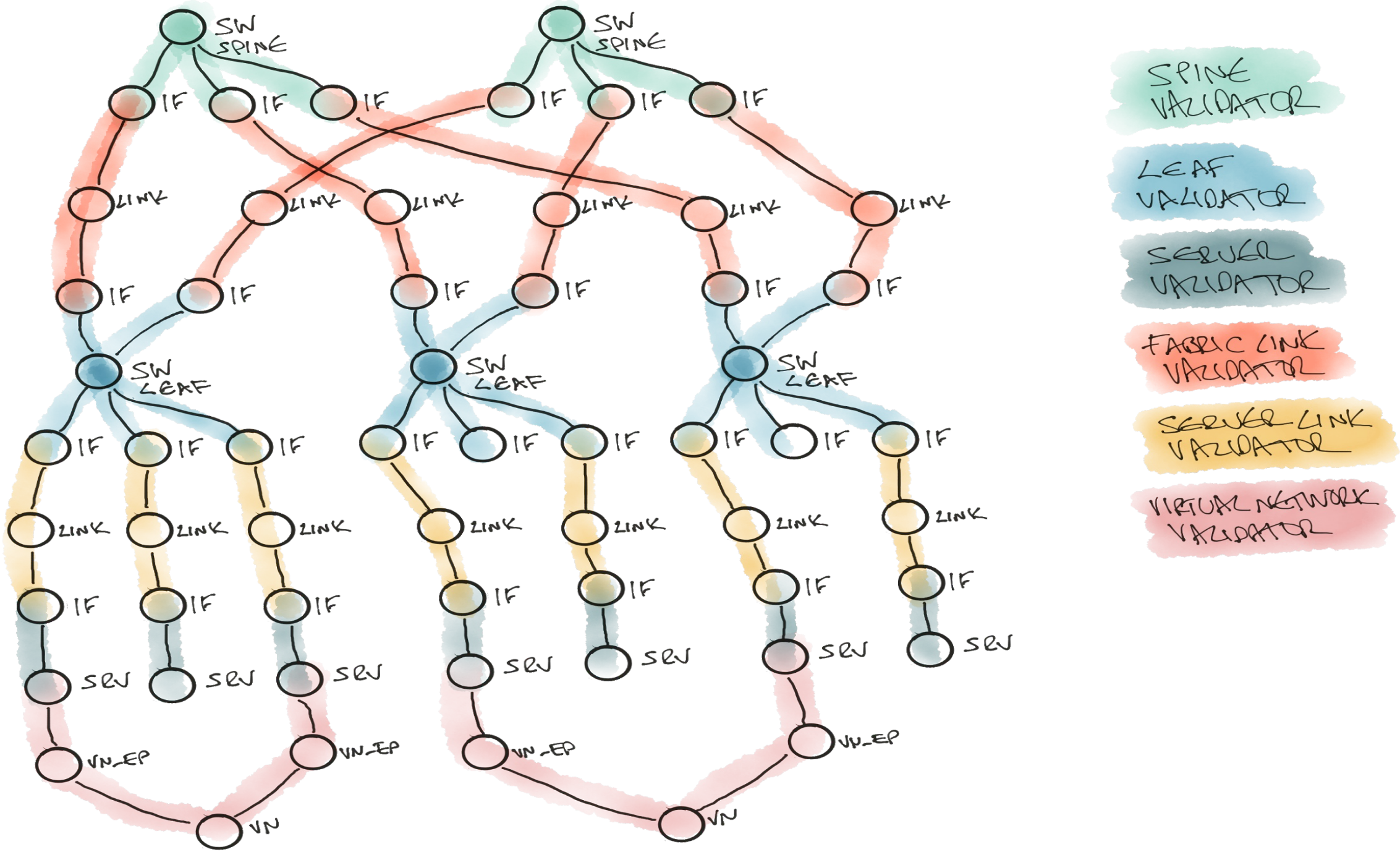
SPINE VALIDATOR


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


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


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


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


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

Decomposition

WIKIPEDIA
The Free Encyclopedia

Decomposition (computer science)

From Wikipedia, the free encyclopedia

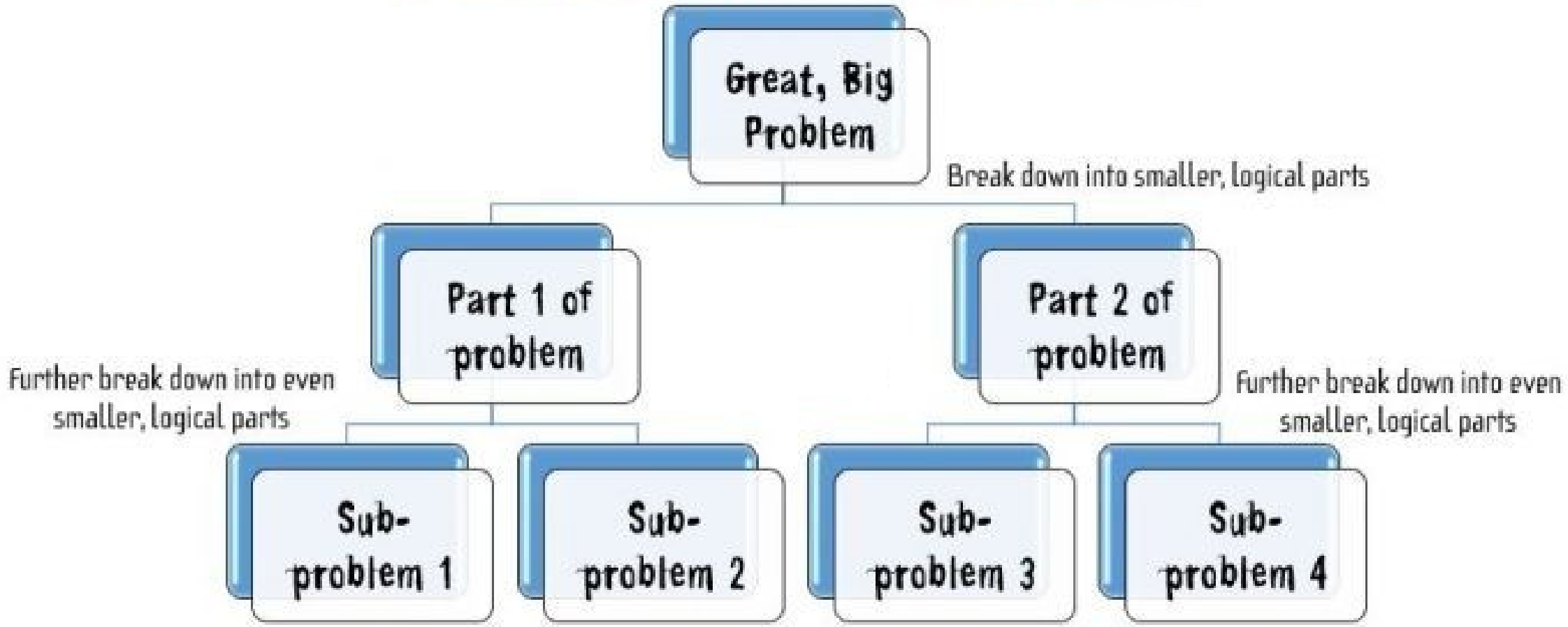
Decomposition in computer science, also known as **factoring**, is breaking a complex problem or system into parts that are easier to conceive, understand, program, and maintain.

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction

Help
About Wikipedia
Community portal
Recent changes

DECOMPOSITION

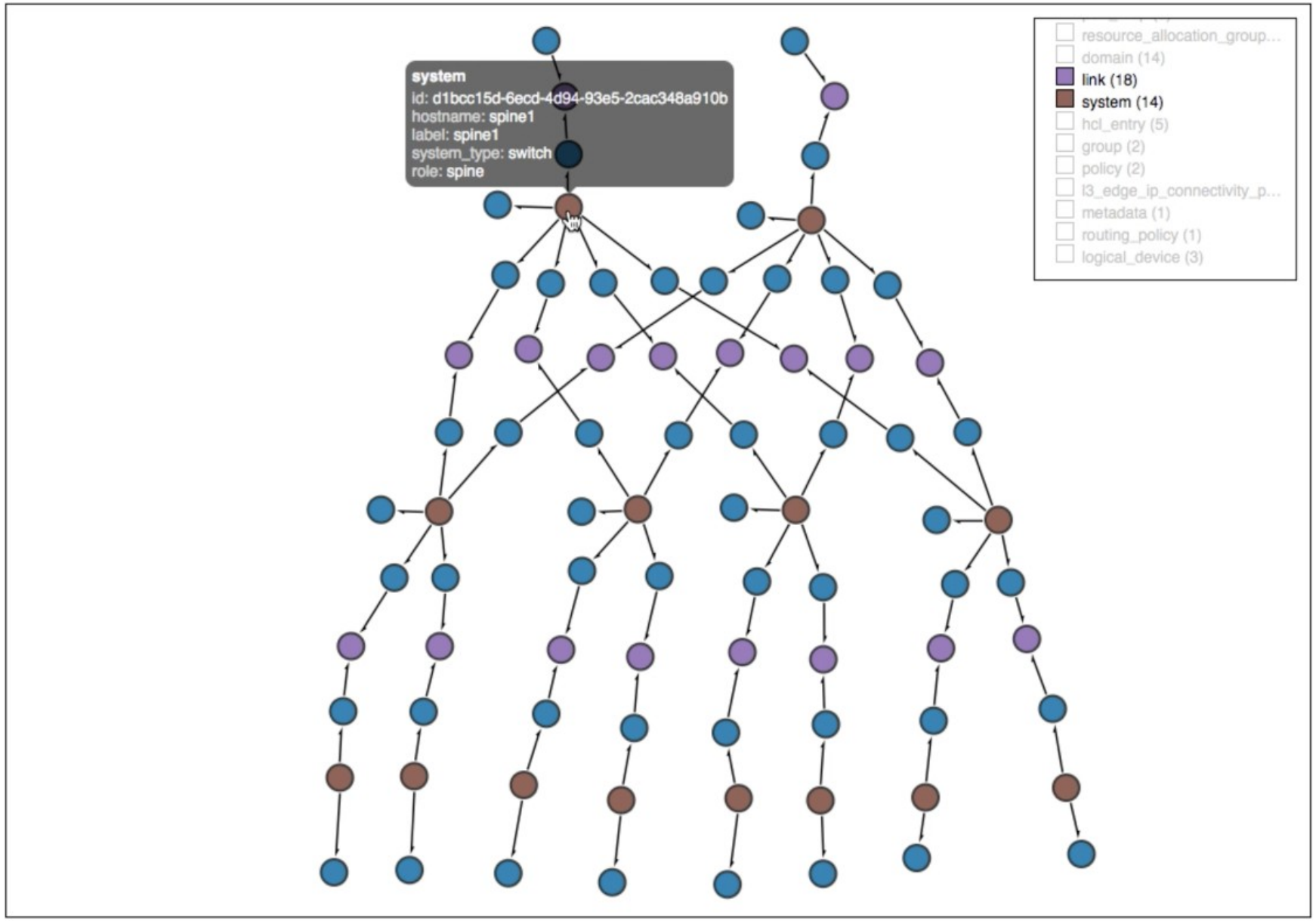


Decomposition: walking the graph

Query:

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

Execute Query



Query:

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

Execute Query

Close

Query

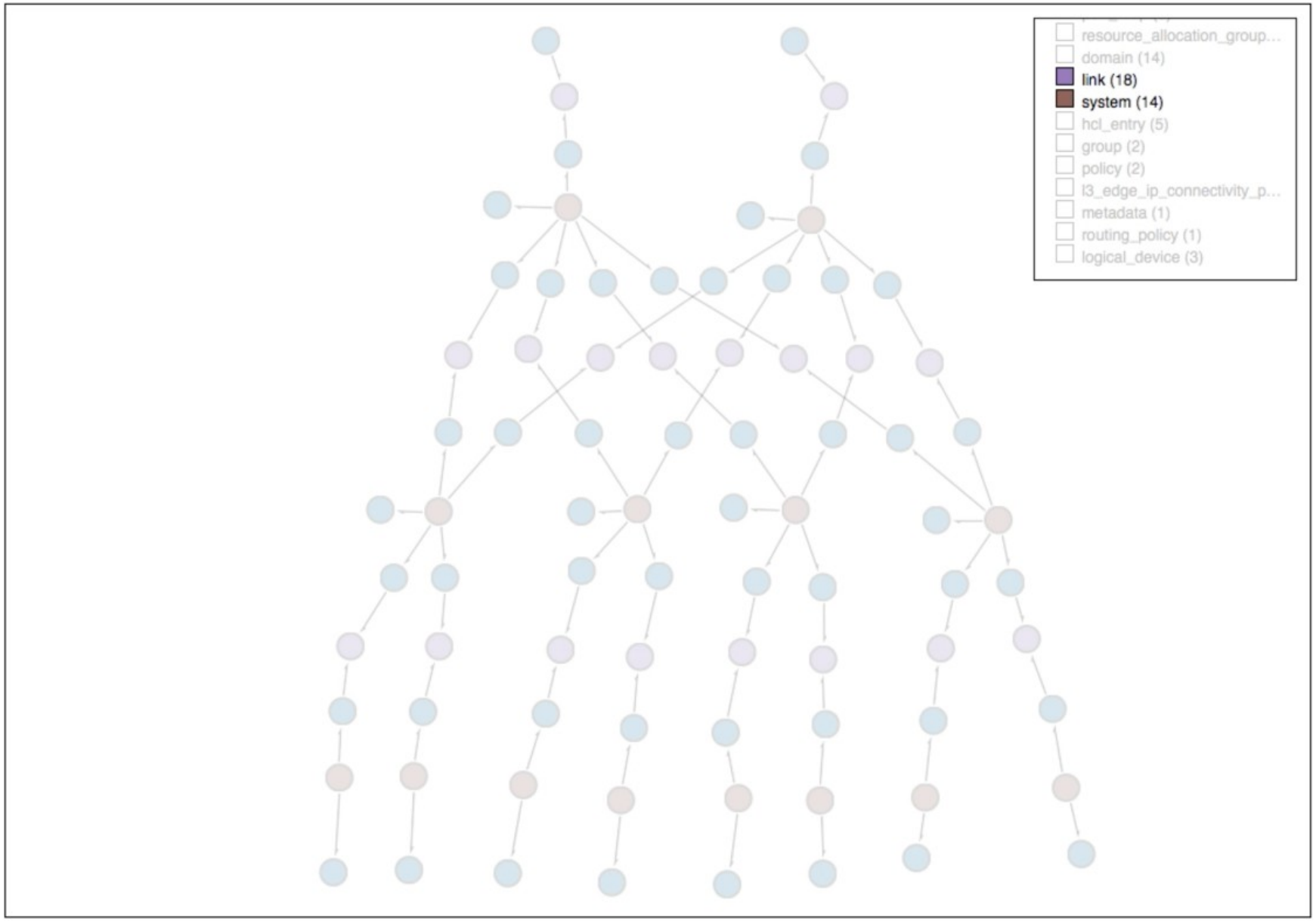
match(node("system", role="spine") .out()
.node("interface") .out() .node("link") .in_()
.node("interface") .in_() .node("system", role="leaf"))

Steps

start 0 1 2 3 4 5 6 7 8

<Start>

Paths (0)



Query:

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

Execute Query

Close

Query

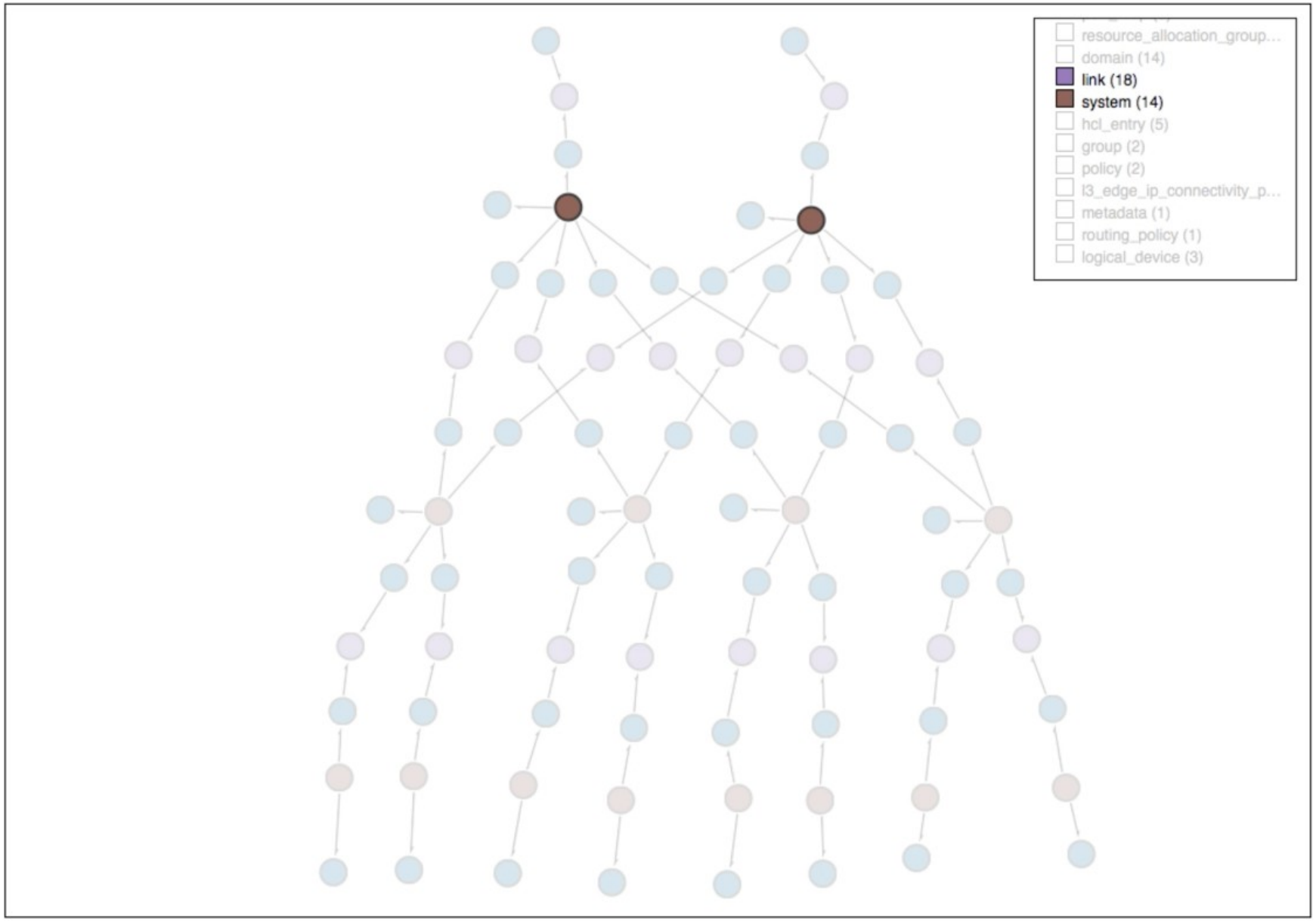
match(node("system", role="spine") .out()
.node("interface") .out() .node("link") .in_()
.node("interface") .in_() .node("system", role="leaf"))

Steps

start 0 1 2 3 4 5 6 7 8

<FindNodeAction type=system role=== spine>

Paths (2)



Query:

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

Execute Query

Close

Query

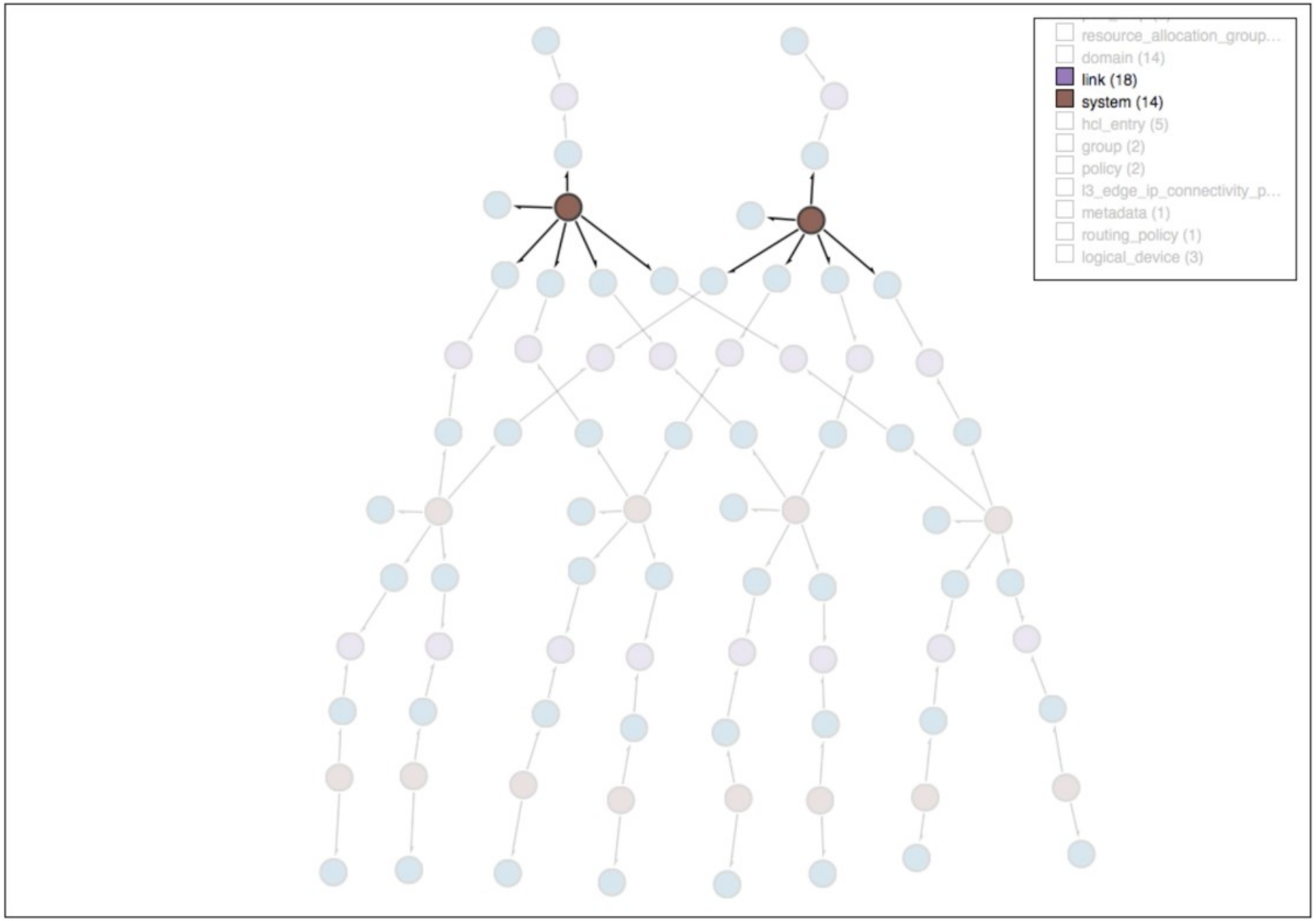
match(node("system", role="spine") .out()
.node("interface") .out() .node("link") .in_()
.node("interface") .in_() .node("system", role="leaf"))

Steps

start 0 1 2 3 4 5 6 7 8

<NodeOutRelationshipAction index=0>

Paths (14)



Query:

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

Execute Query

Close

Query

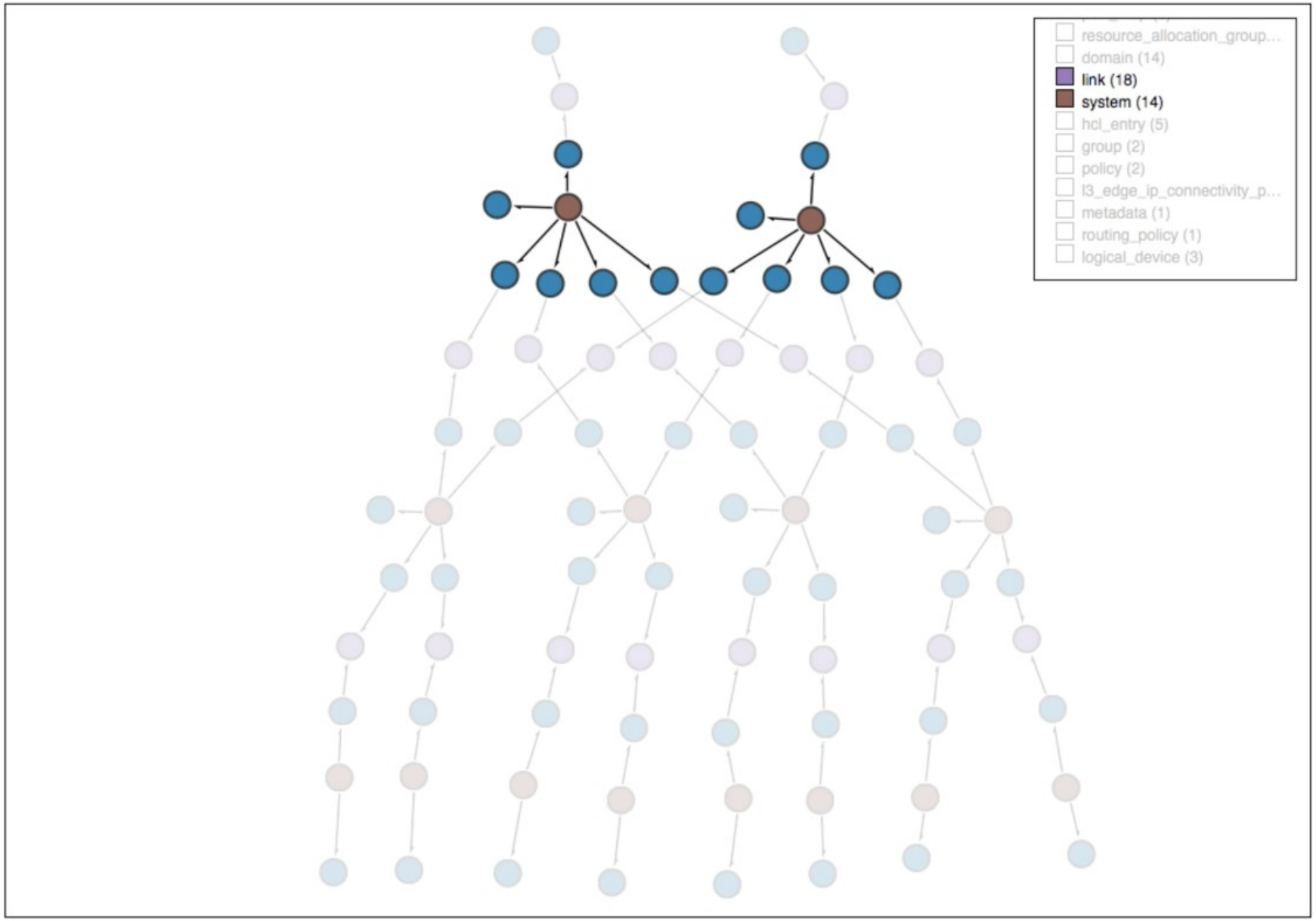
match(node("system", role="spine") .out()
.node("interface") .out() .node("link") .in_()
.node("interface") .in_() .node("system", role="leaf"))

Steps

start 0 1 2 3 4 5 6 7 8

<RelationshipTargetAction index=1 type=interface>

Paths (12)



Query:

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

Execute Query

Close

Query

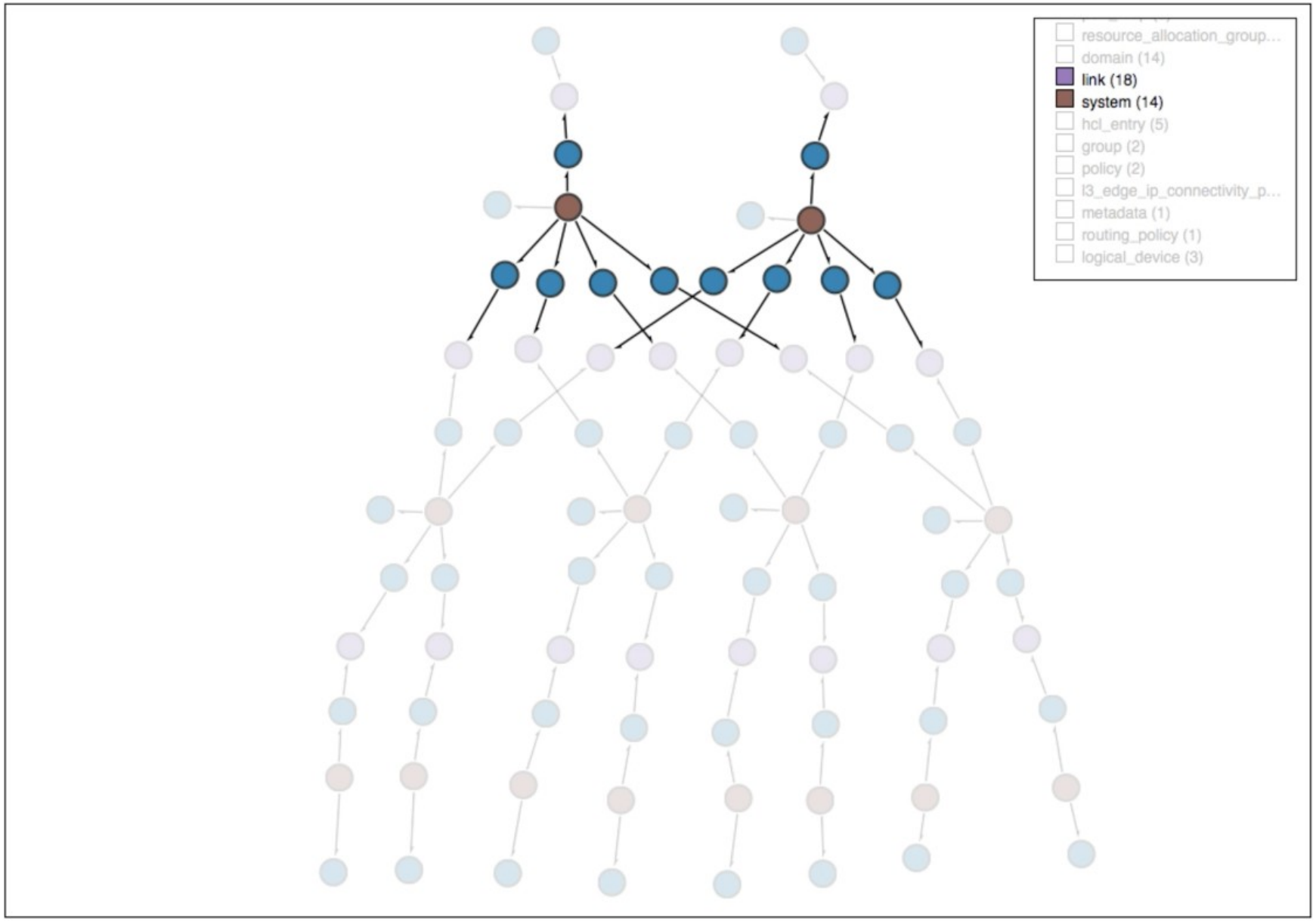
match(node("system", role="spine") .out()
.node("interface") .out() .node("link") .in_()
.node("interface") .in_() .node("system", role="leaf"))

Steps

start 0 1 2 3 4 5 6 7 8

<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")
)
```

Execute Query

Close

Query

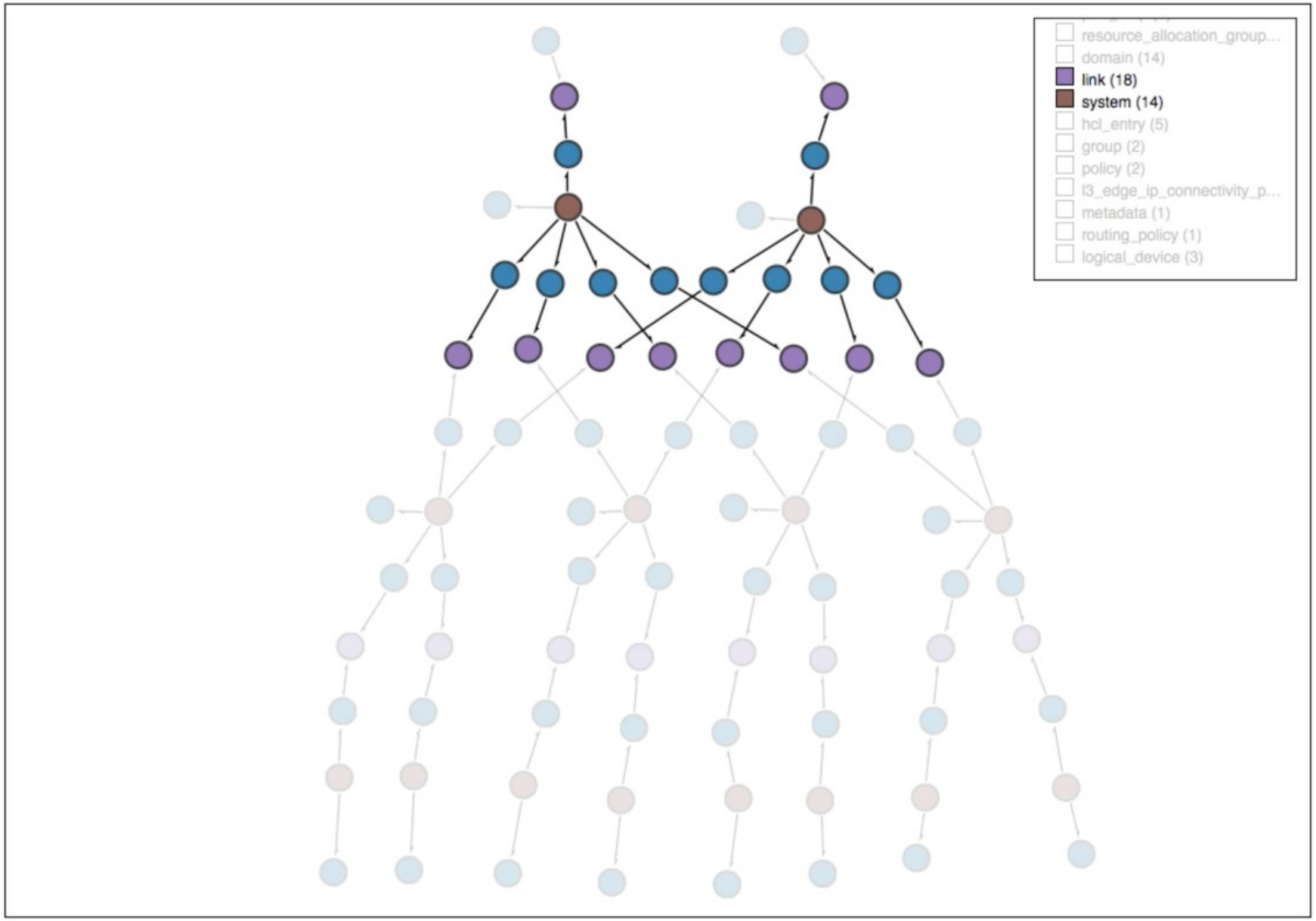
match(node("system", role="spine") .out()
.node("interface") .out() .node("link") .in_()
.node("interface") .in_() .node("system", role="leaf"))

Steps

start 0 1 2 3 4 5 6 7 8

<RelationshipTargetAction index=3 type=link>

Paths (10)



Query:

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

Execute Query

Close

Query

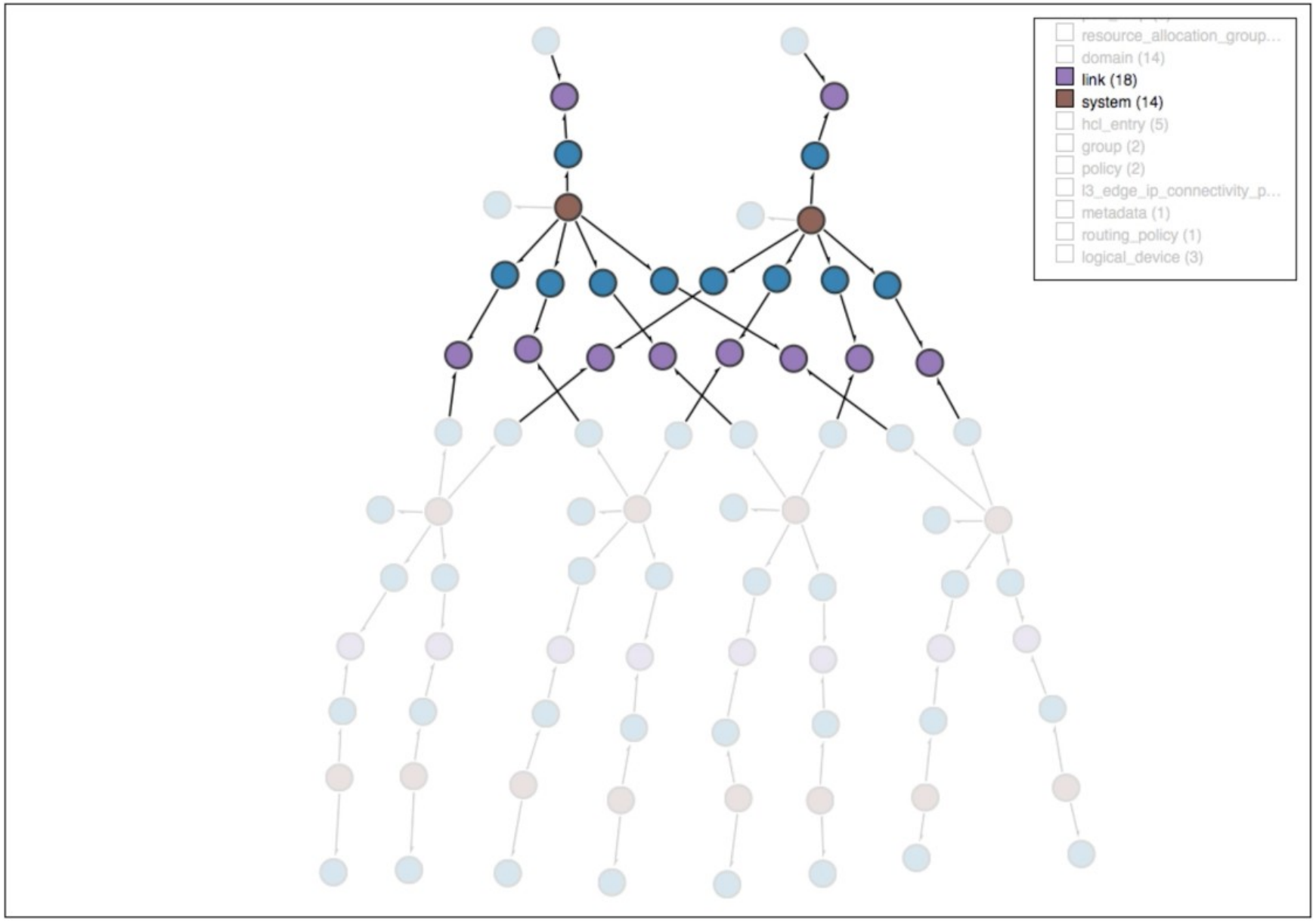
match(node("system", role="spine") .out()
.node("interface") .out() .node("link") .in_()
.node("interface") .in_() .node("system", role="leaf"))

Steps

start 0 1 2 3 4 5 6 7 8

<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")
)
```

Execute Query

Close

Query

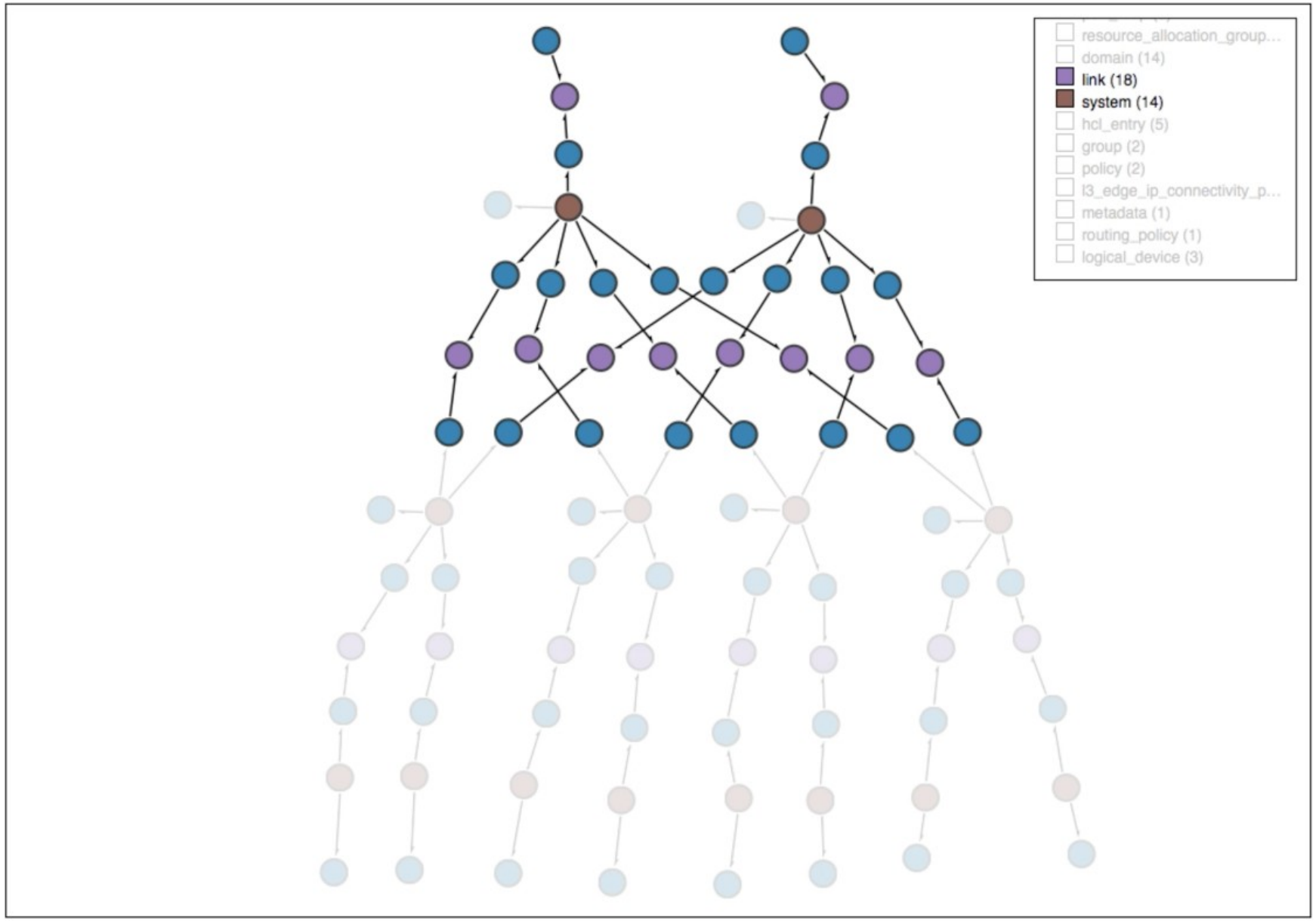
match(node("system", role="spine") .out()
.node("interface") .out() .node("link") .in_()
.node("interface") .in_() .node("system", role="leaf"))

Steps

start 0 1 2 3 4 5 6 7 8

<RelationshipSourceAction index=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")
)
```

Execute Query

Close

Query

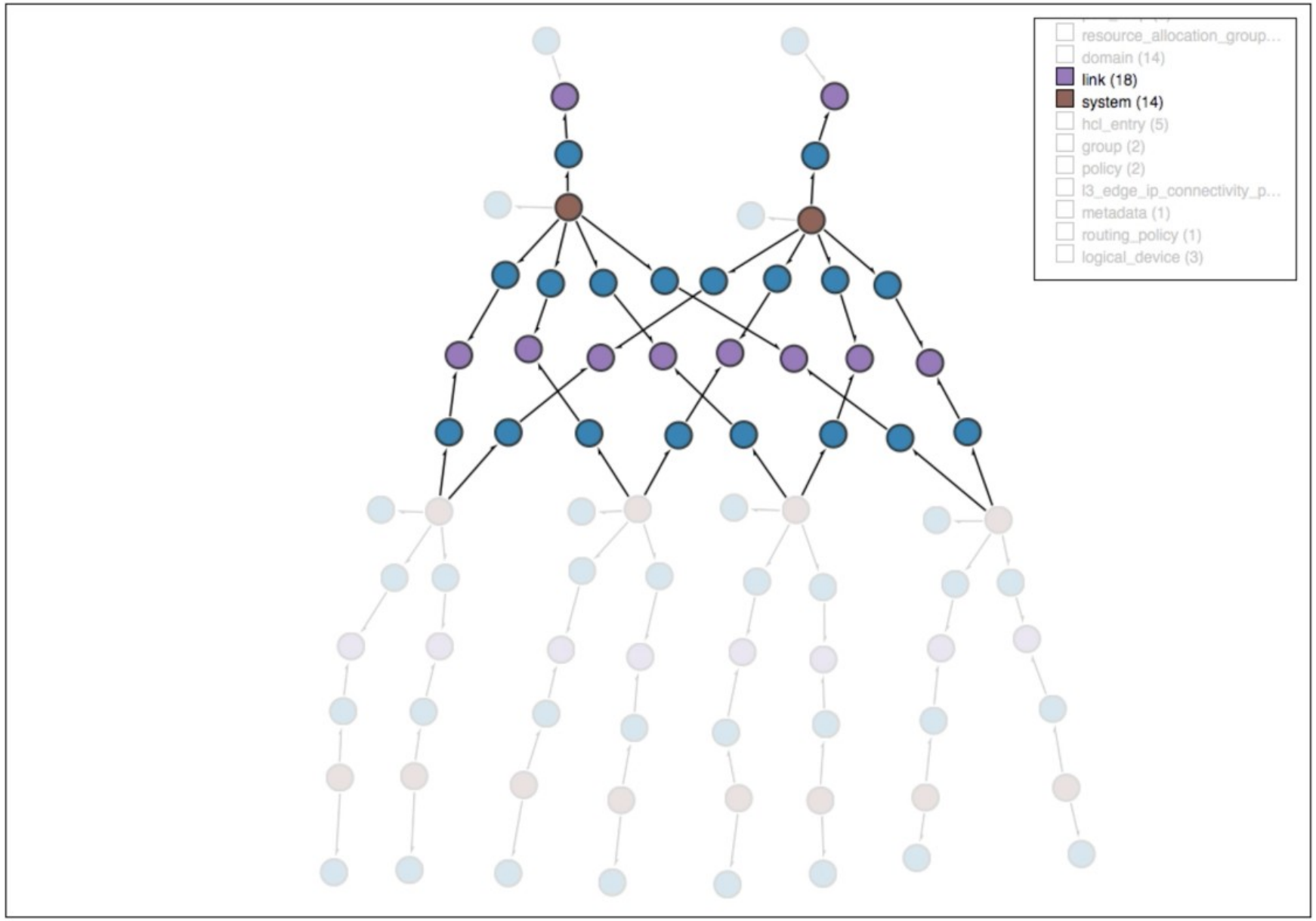
match(node("system", role="spine") .out()
.node("interface") .out() .node("link") .in_()
.node("interface") .in_() .node("system", role="leaf"))

Steps

start 0 1 2 3 4 5 6 7 8

<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")
)
```

Execute Query

Close

Query

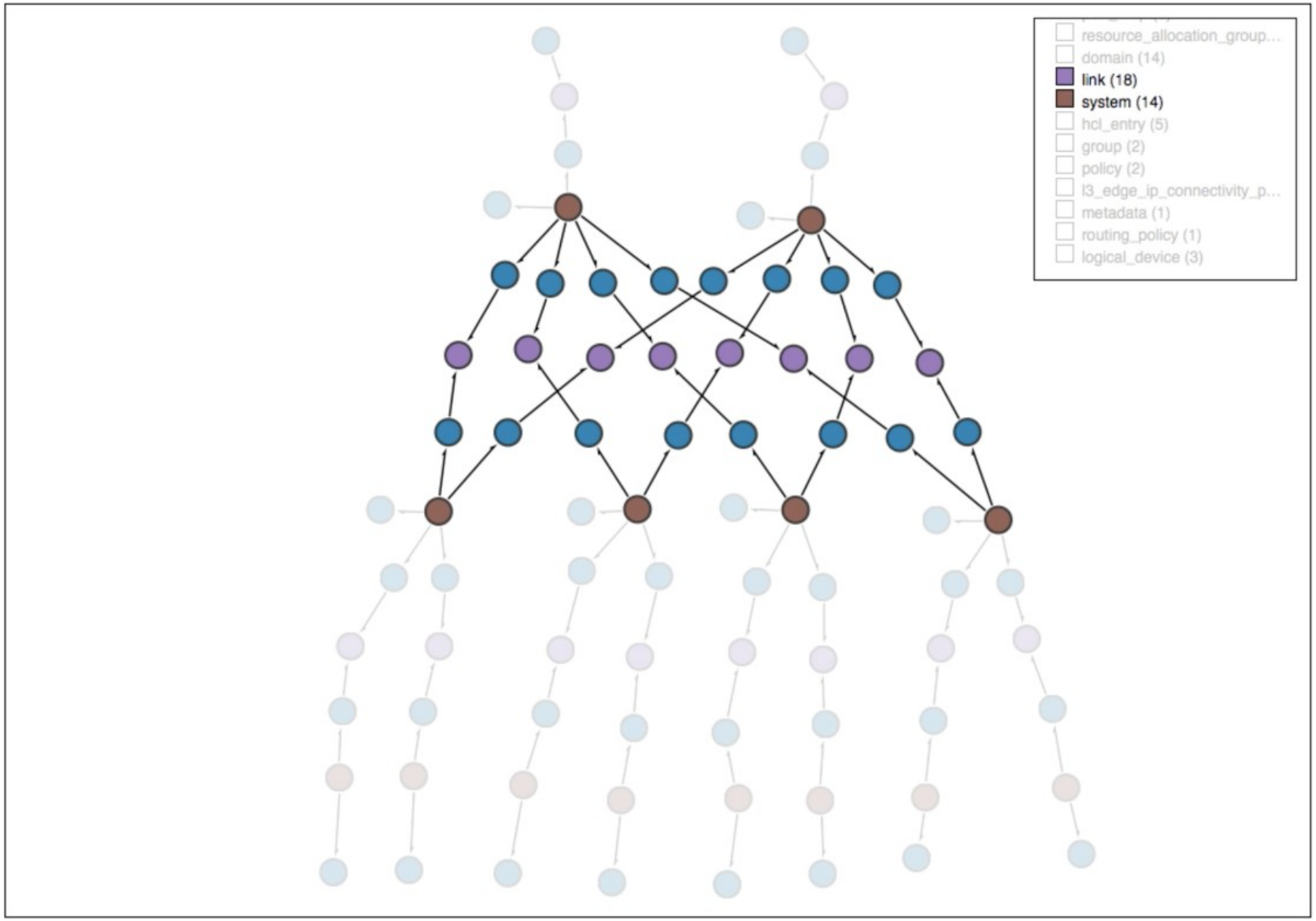
match(node("system", role="spine") .out()
.node("interface") .out() .node("link") .in_()
.node("interface") .in_() .node("system", role="leaf"))

Steps

start 0 1 2 3 4 5 6 7 8

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

Paths (8)



Query:

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

Execute Query

Close

Query

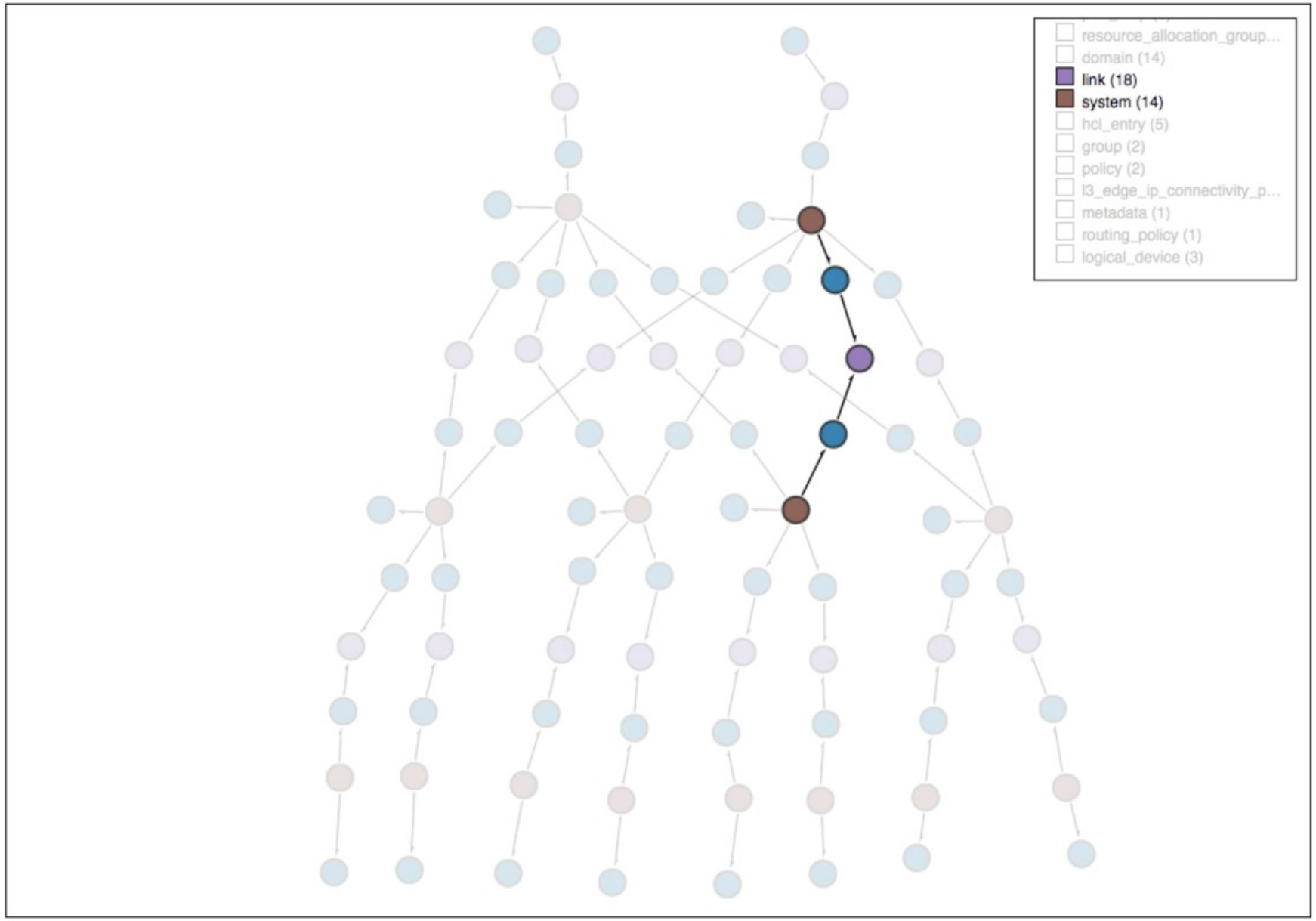
match(node("system", role="spine") .out()
.node("interface") .out() .node("link") .in_()
.node("interface") .in_() .node("system", role="leaf"))

Steps

start 0 1 2 3 4 5 6 7 8

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

Paths (8)



Query:

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

Execute Query

Close

Query

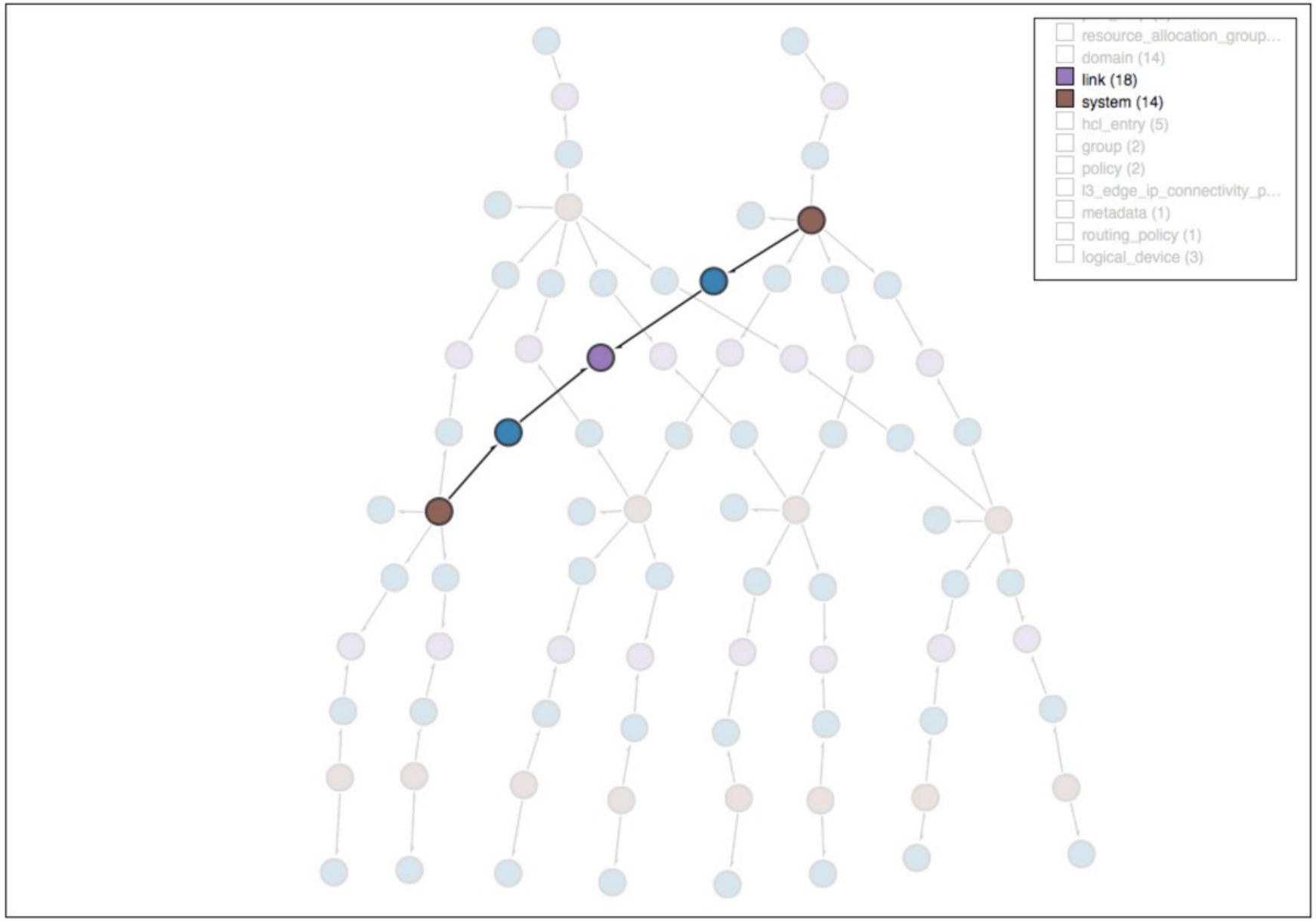
match(node("system", role="spine") .out()
.node("interface") .out() .node("link") .in_()
.node("interface") .in_() .node("system", role="leaf"))

Steps

start 0 1 2 3 4 5 6 7 8

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

Paths (8)



Query:

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

Execute Query

Close

Query

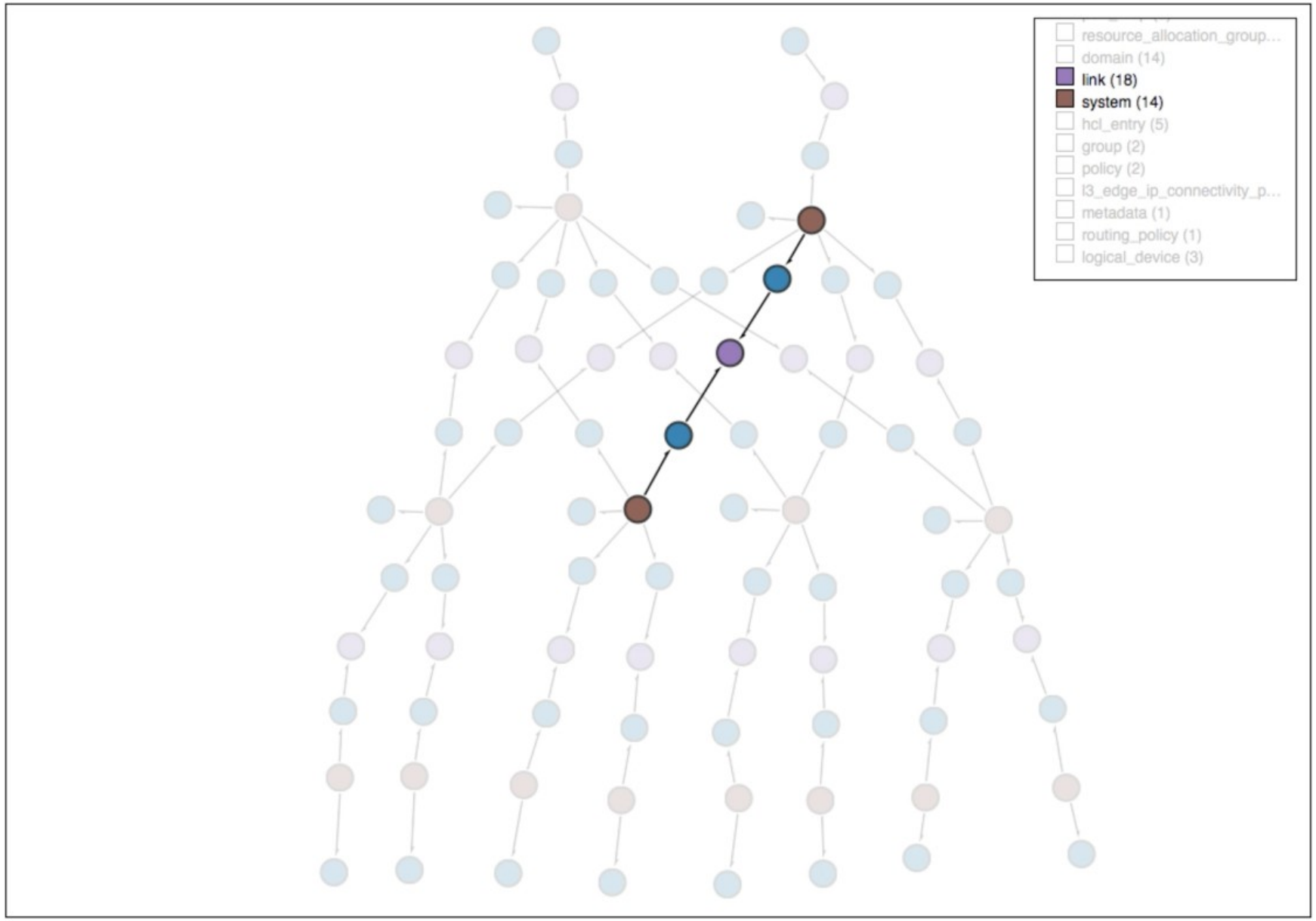
match(node("system", role="spine") .out()
.node("interface") .out() .node("link") .in_()
.node("interface") .in_() .node("system", role="leaf"))

Steps

start 0 1 2 3 4 5 6 7 8

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

Paths (8)



Query:

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

Execute Query

Close

Query

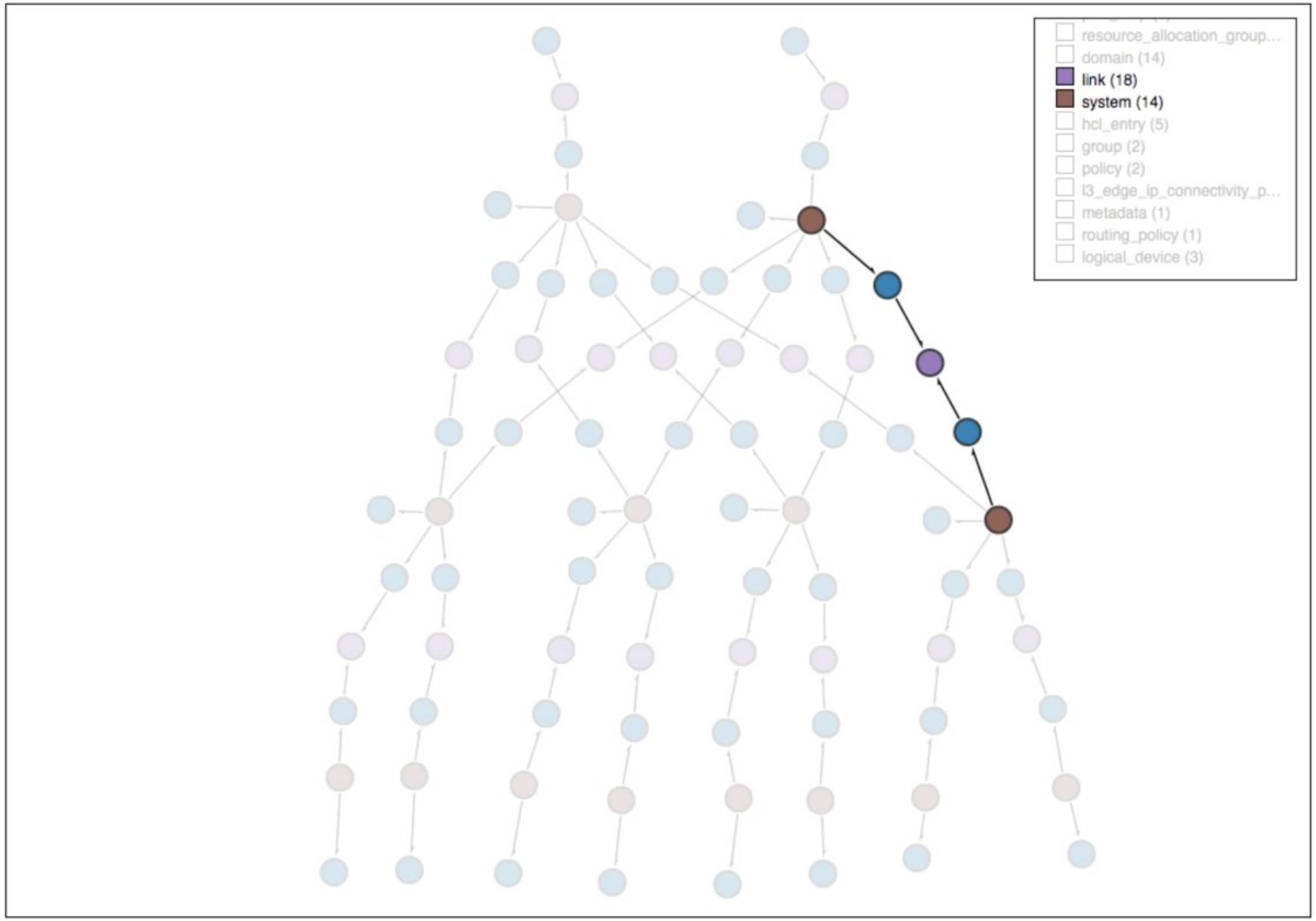
match(node("system", role="spine") .out()
.node("interface") .out() .node("link") .in_()
.node("interface") .in_() .node("system", role="leaf"))

Steps

start 0 1 2 3 4 5 6 7 8

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

Paths (8)



Query:

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

Execute Query

Close

Query

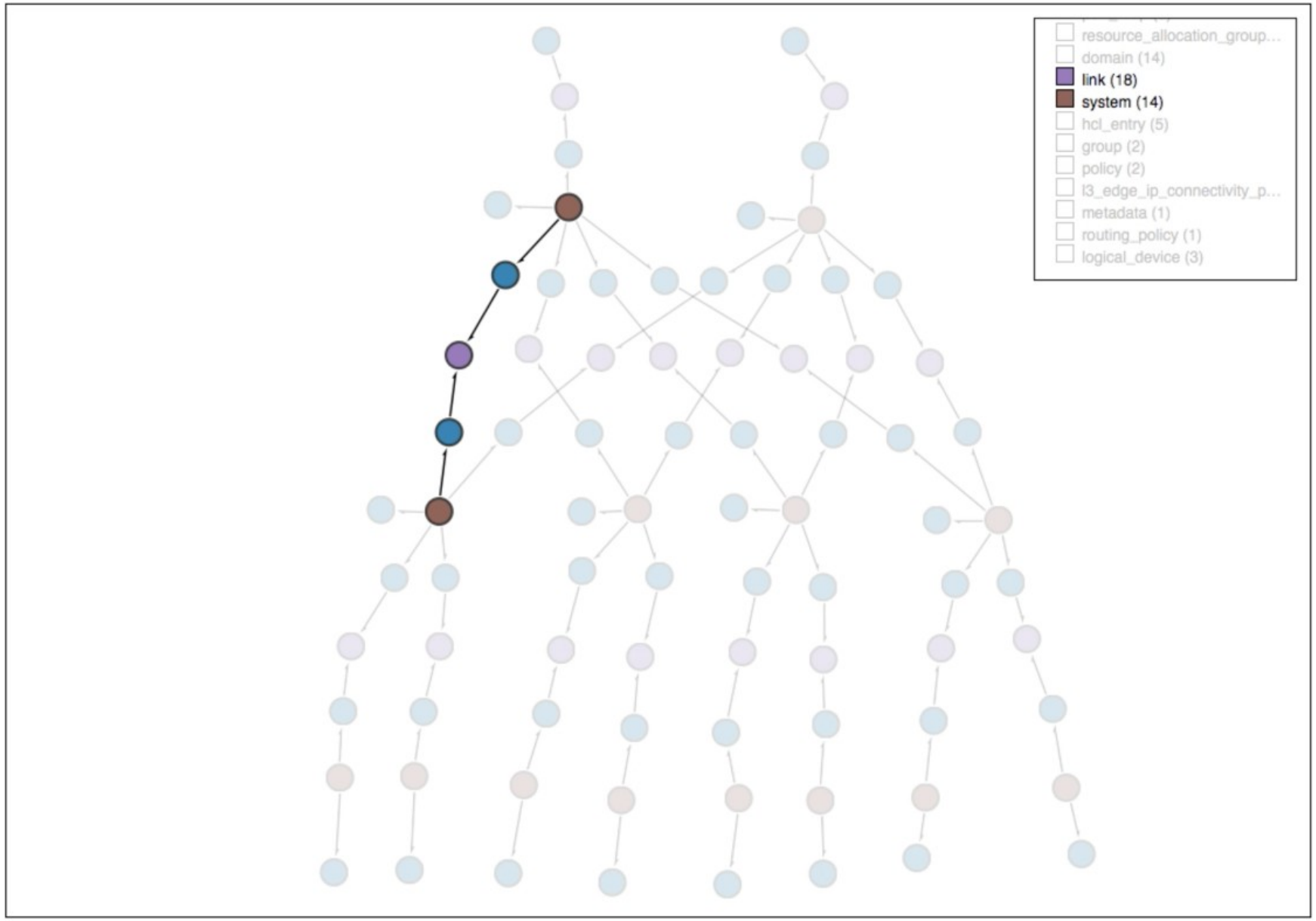
match(node("system", role="spine") .out()
.node("interface") .out() .node("link") .in_()
.node("interface") .in_() .node("system", role="leaf"))

Steps

start 0 1 2 3 4 5 6 7 8

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

Paths (8)



Query:

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

Execute Query

Close

Query

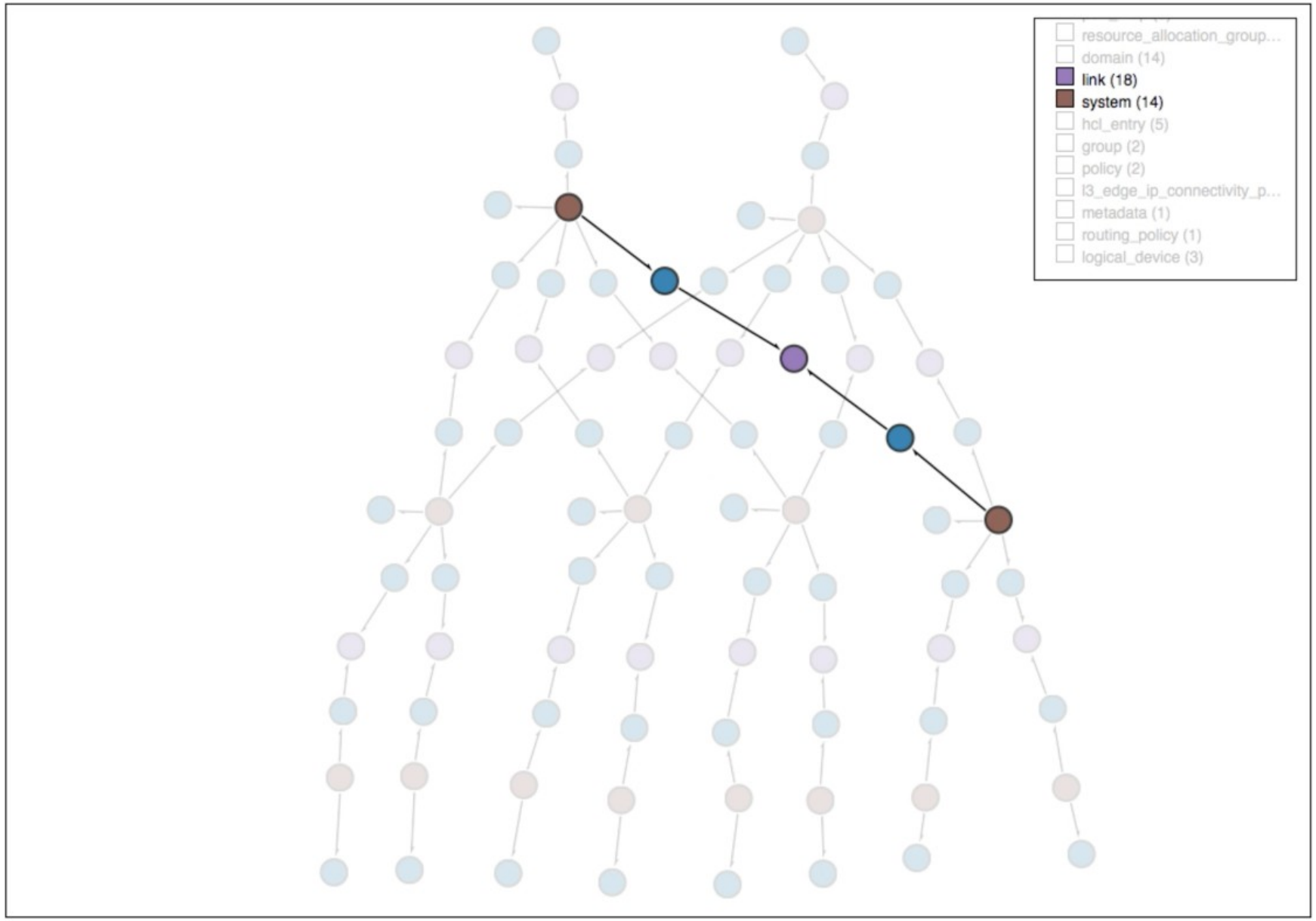
match(node("system", role="spine") .out()
.node("interface") .out() .node("link") .in_()
.node("interface") .in_() .node("system", role="leaf"))

Steps

start 0 1 2 3 4 5 6 7 8

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

Paths (8)



Query:

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

Execute Query

Close

Query

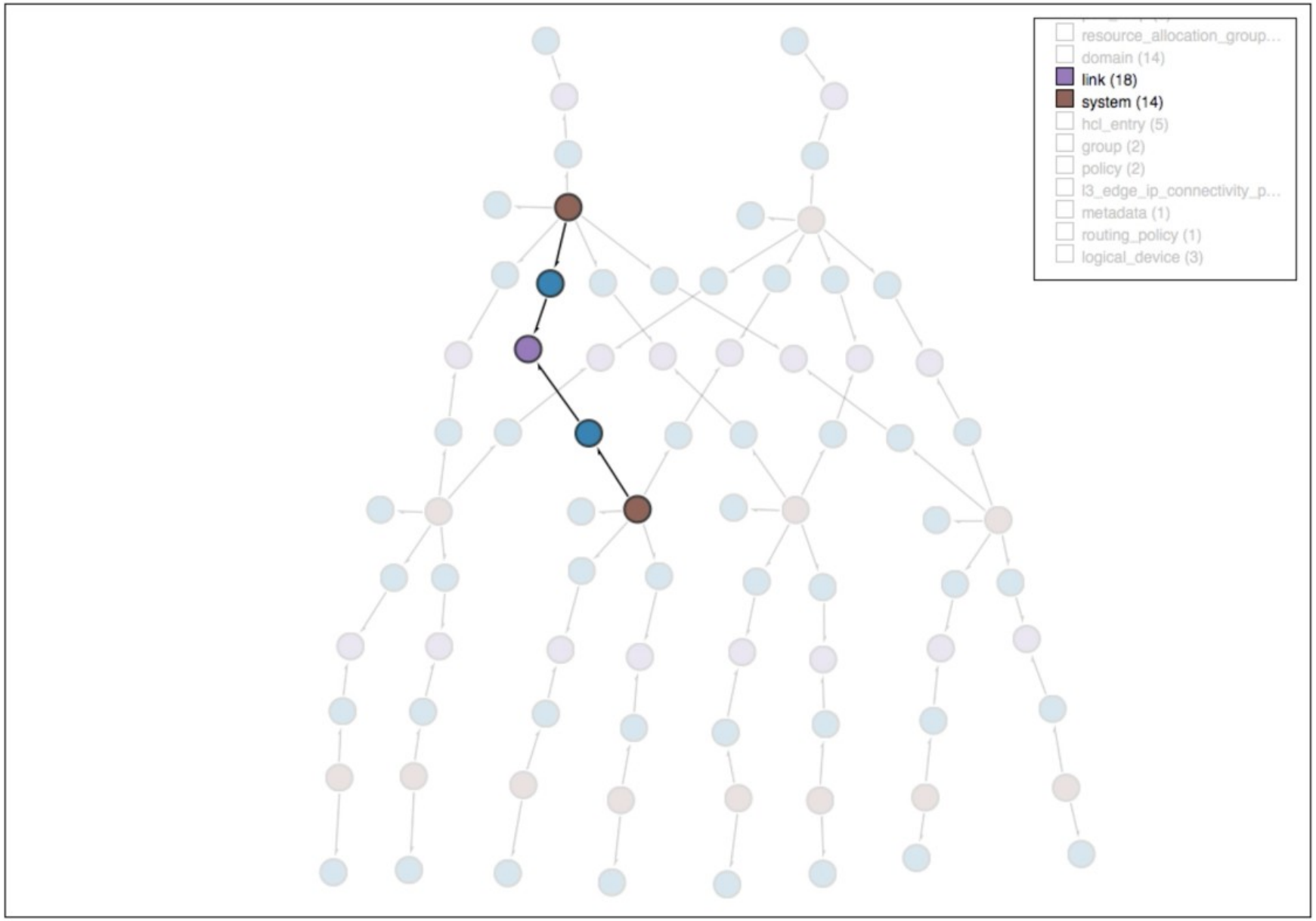
match(node("system", role="spine") .out()
.node("interface") .out() .node("link") .in_()
.node("interface") .in_() .node("system", role="leaf"))

Steps

start 0 1 2 3 4 5 6 7 8

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

Paths (8)



Query:

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

Execute Query

Close

Query

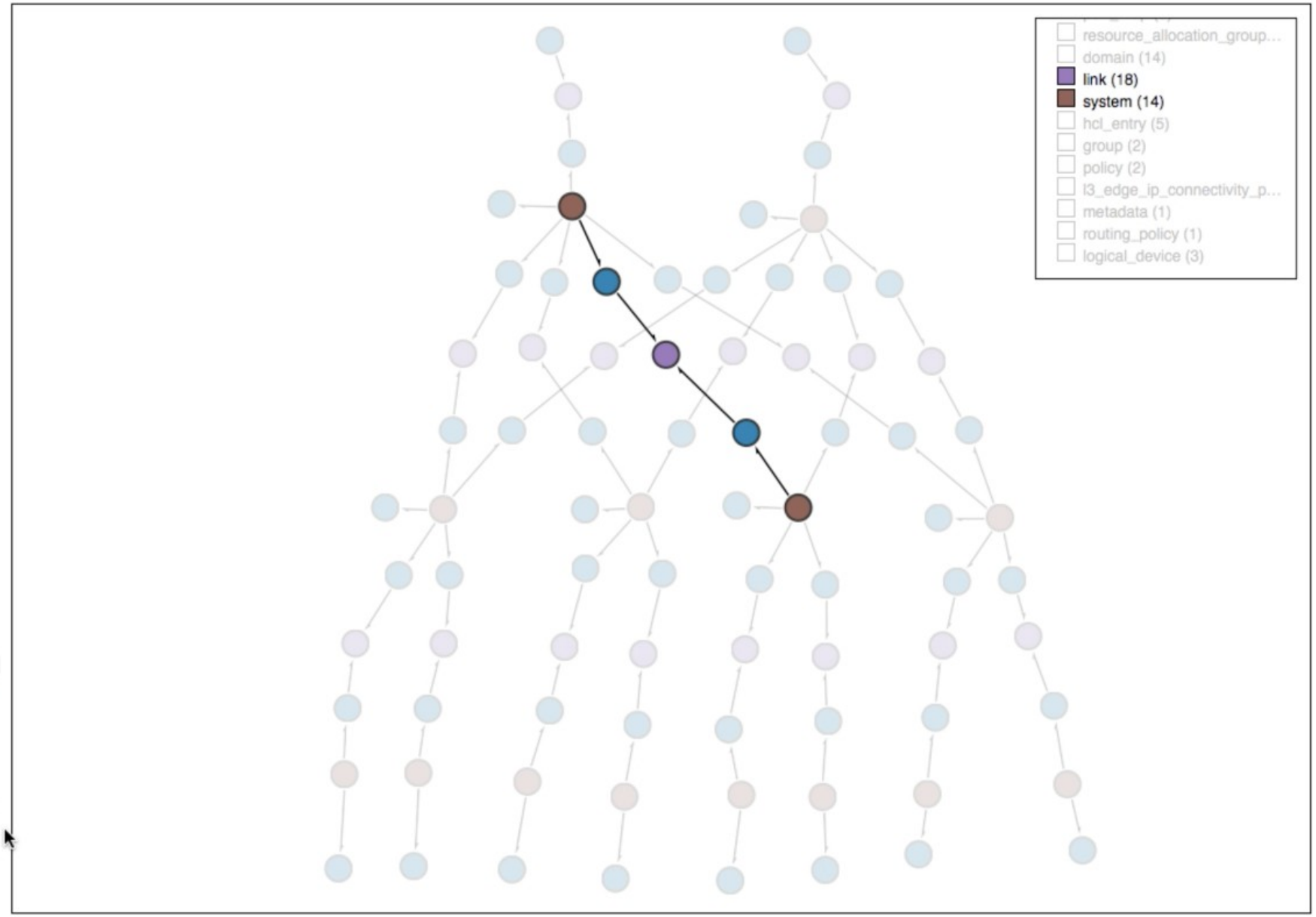
match(node("system", role="spine") .out()
.node("interface") .out() .node("link") .in_()
.node("interface") .in_() .node("system", role="leaf"))

Steps

start 0 1 2 3 4 5 6 7 8

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

Paths (8)



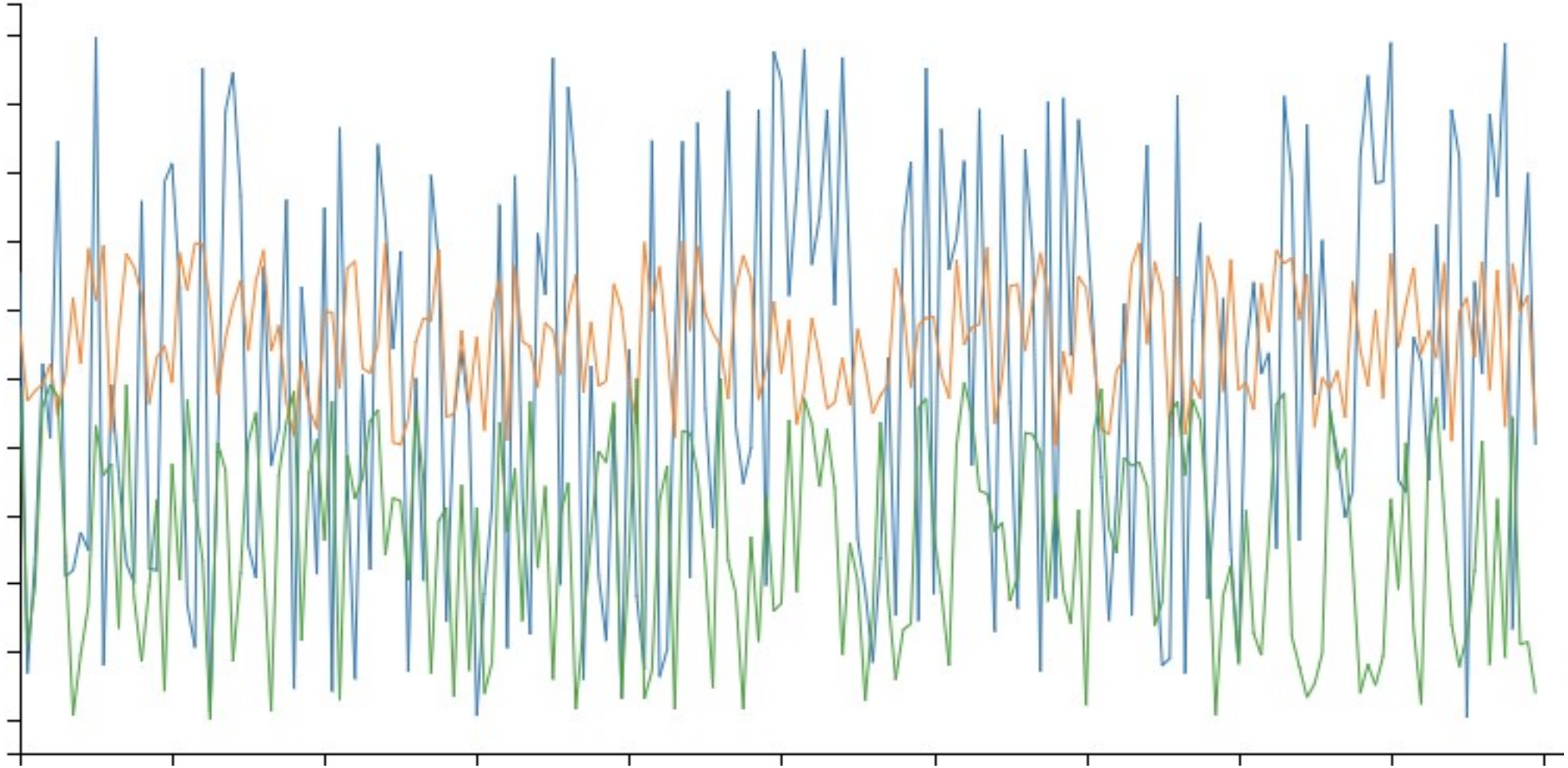
Intent Based Analytics

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

Was I looking for something?

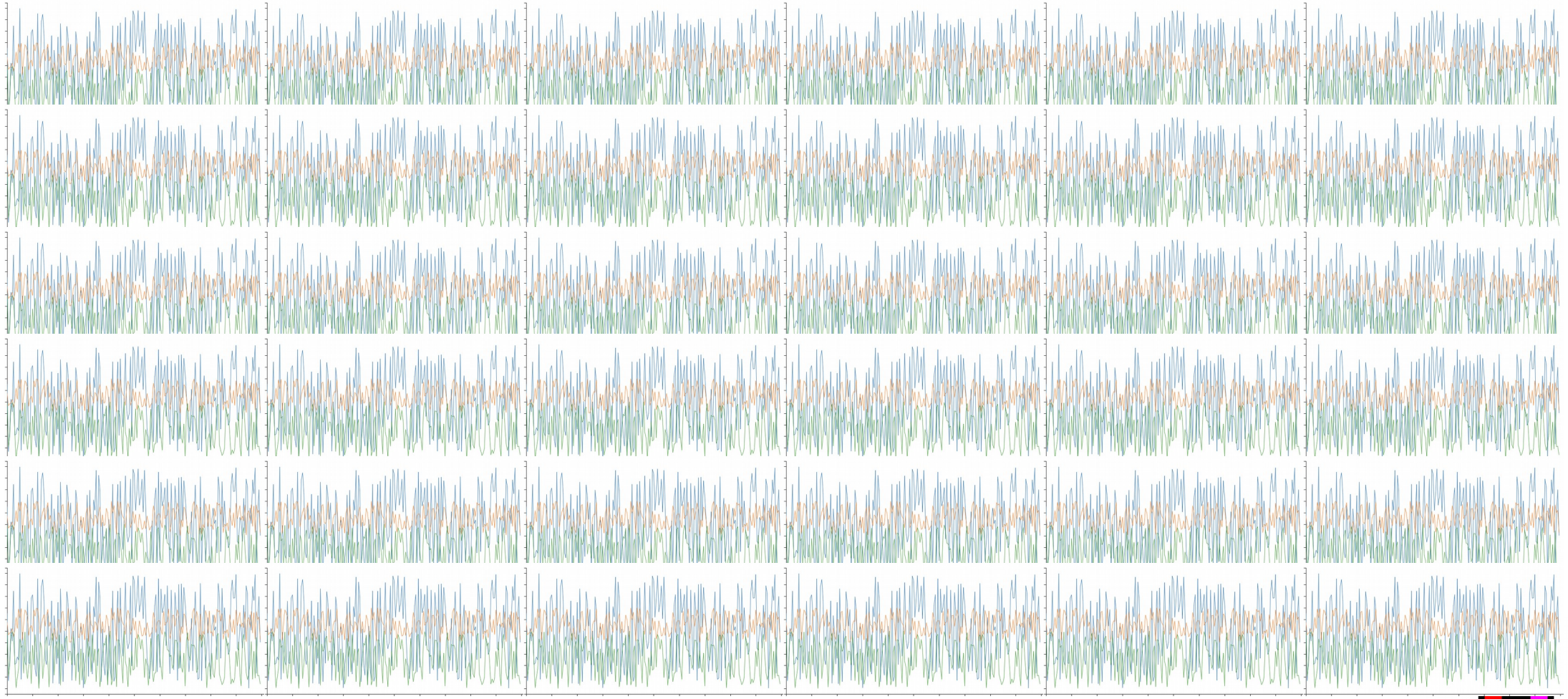
apstra

Gathering high def telemetry



apstra

For all my leaf1 interfaces



apstra

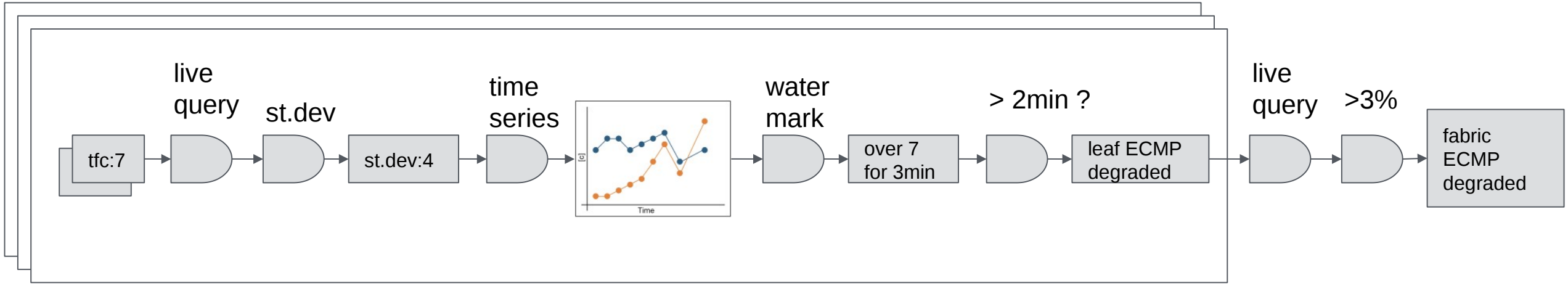
For all my leafs

apstra

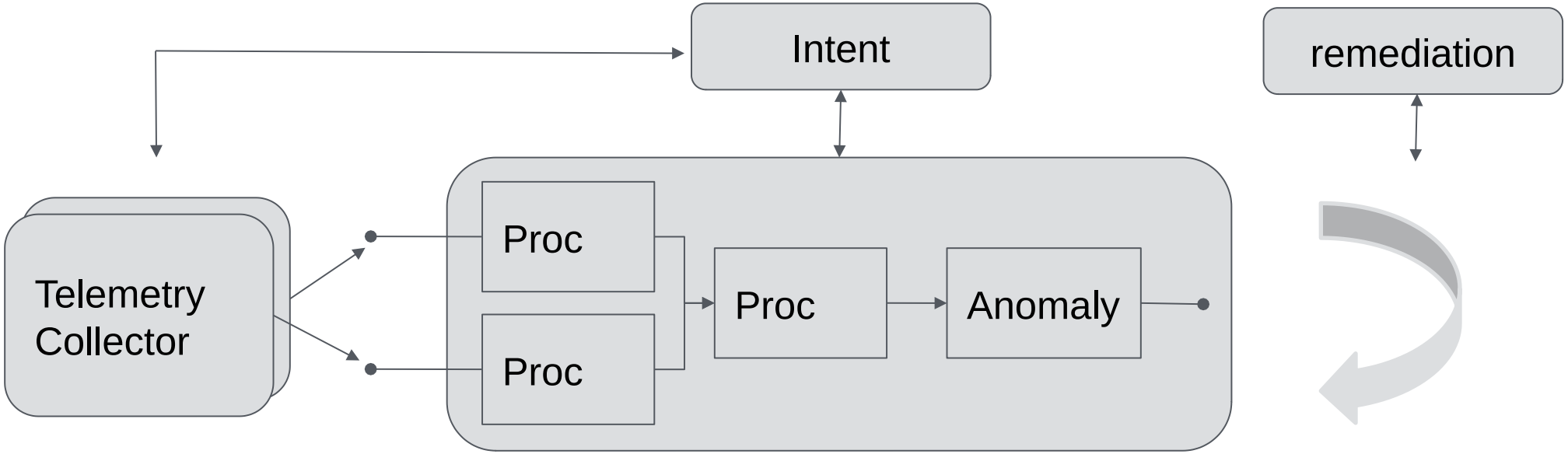
So that I have insight

apstra

IBA : ECMP fabric health



IBA – context aware analytics



Declaratively specified

Once specified, is in constant sync with intent

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

New telemetry is “wired-in”

apstra

Conclusion

- Basic automation, while hot topic - is the easiest step in the IBN journey
- Single source of truth is mandatory to be able to reason about any change
- Day 2 operations @scale:
  - context aware continues validation
  - dealing with changes
  - configuration drift
  - remediation

  is a much more complicated area to deal with

apstra

Questions

apstra

Thank You!

www.apstra.com

@ApstraInc

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

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

apstra®