A Network-State Management Service

Peng Sun

Ratul Mahajan, Jennifer Rexford,

Lihua Yuan, Ming Zhang, Ahsan Arefin

Princeton & Microsoft

Complex Infrastructure

Microsoft Azure

Number of	2010	2014
Data Center	A few	10s
Network Device	1,000s	10s of 1,000s
Network Capacity	10s of Tbps	Pbps

Variety of vendors/models/time

Management Applications

- Traffic Engine
- Load Balan
- Link Corruption Mitig
- Device Firmware Upgrade

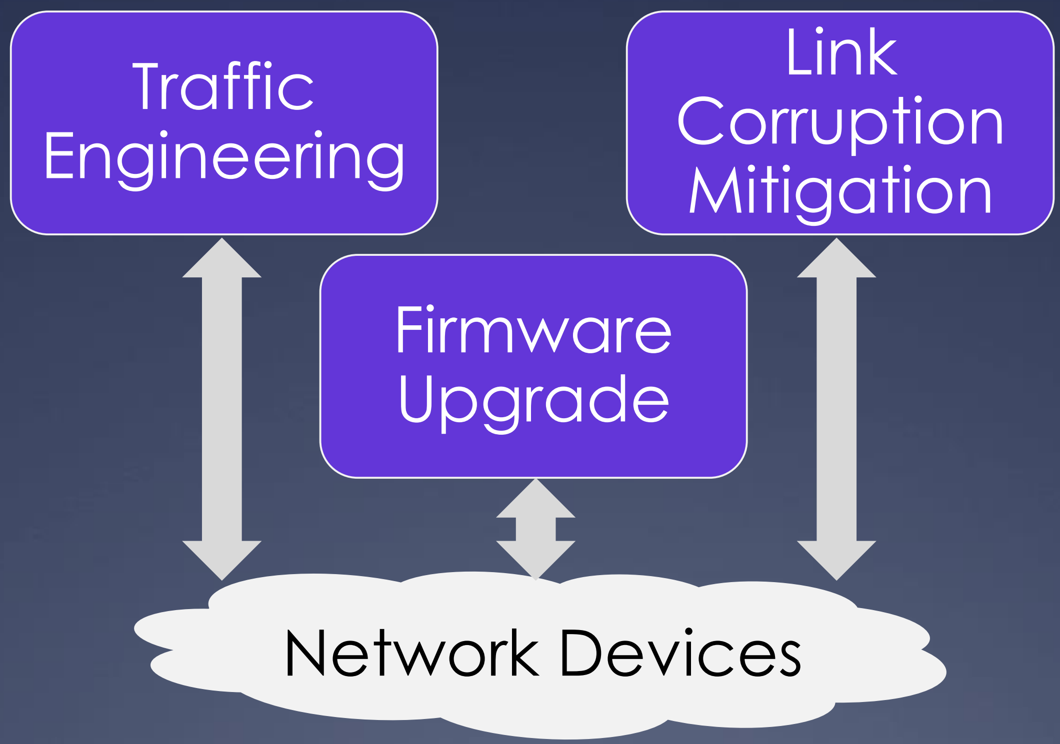
● ● ● ● ● ●

Our Question

How to safely run ***multiple*** management applications on ***shared*** infrastructure

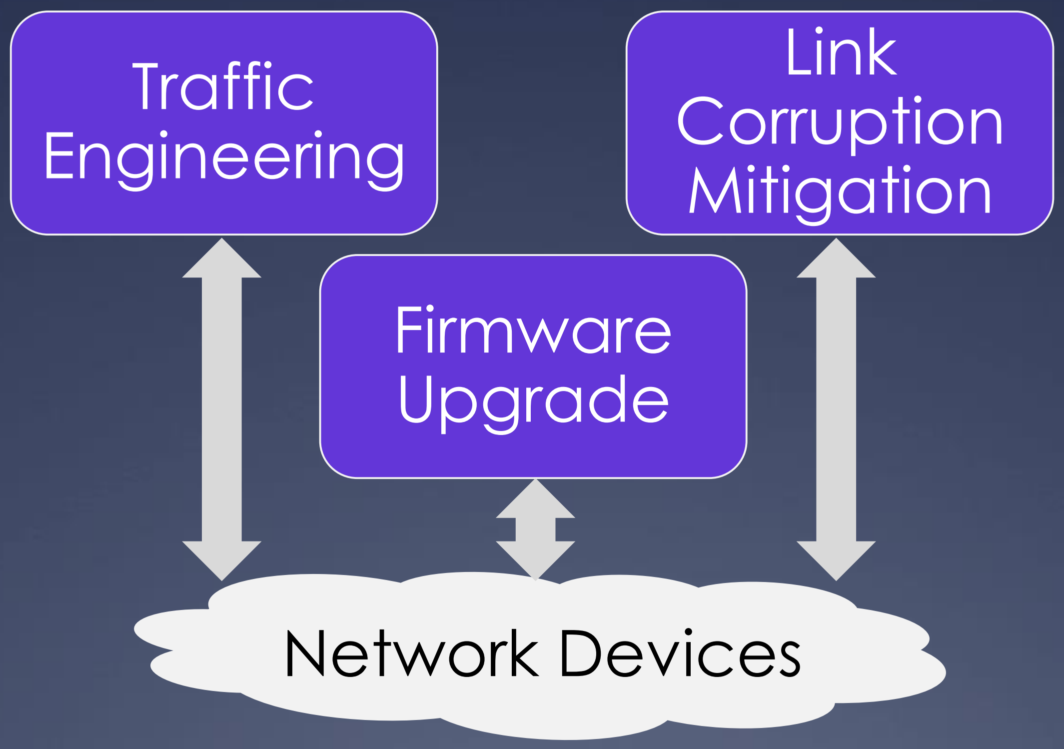
Naïve Solution

- Run independently

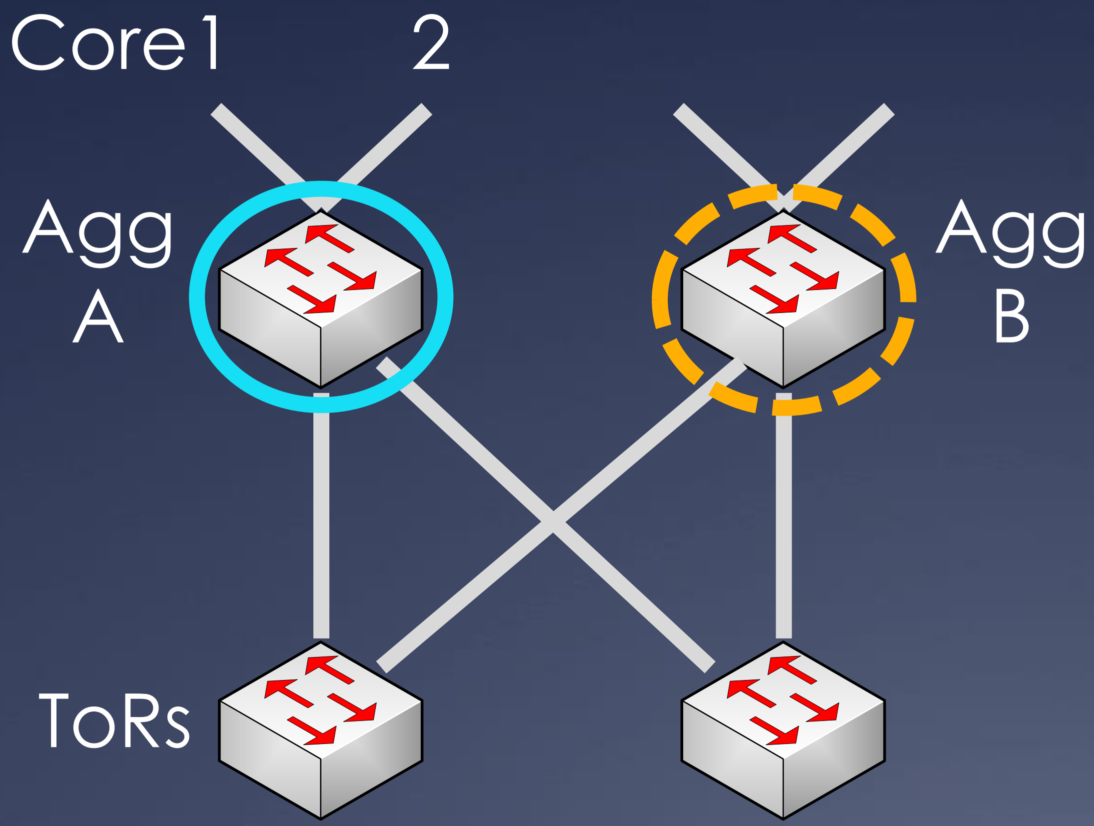


Naïve Solution

- It does not work due to 2 problems



Problem #2: Safety Violation



Link-corruption-mitigation shuts down faulty Agg A

Firmware-upgrade schedules Agg B to upgrade

Potential Solution #1

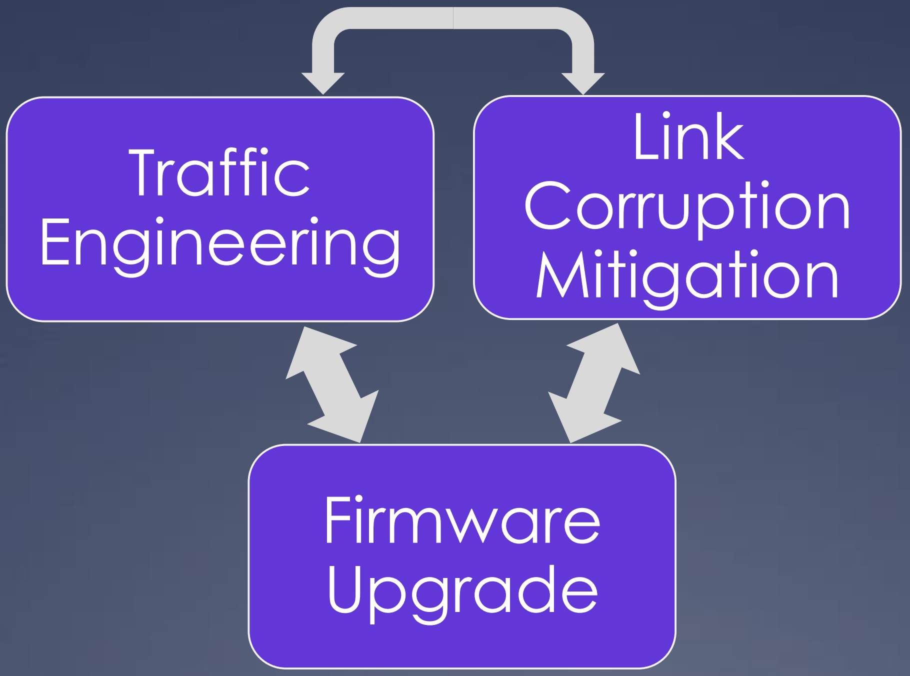
- One monolithic application
- Central control of all actions

Too Complex to Build

- Difficult to develop
  - Combine all applications that are already individually complicated
- High maintenance cost
  - for such huge software in practice

Potential Solution #2

- Explicit coordination among applications
- Consensus over network changes



Still Too Complex

- Hard to understand each other
  - Diverse network interactions

Application	Routing	Device Config
Traffic Engineering	✓	✗
Firmware upgrade	✗	✓

Main Enemy: Complexity

- Application development
- Application coordination

What We Advocate

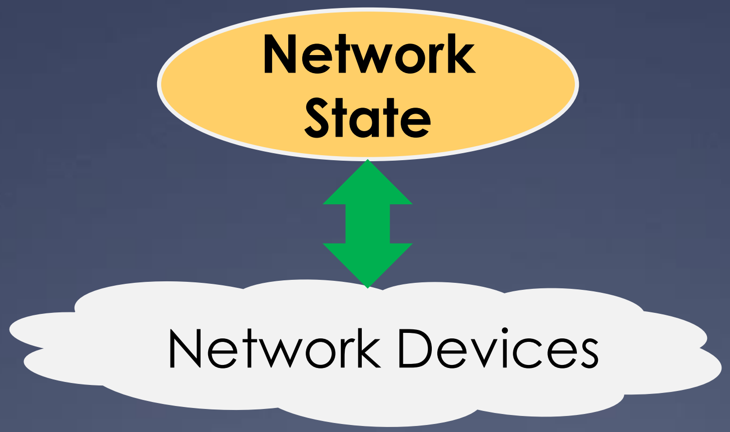
- Loose coupling of applications
- Design principle:
  - Simplicity with safety guarantees
- Forgo joint optimization
  - Worthwhile tradeoff for simplicity
  - Applications could do it out-of-band

Overview of Statesman

- Network operating system for safe multi-application operation
- Uses network state abstraction
  - Three views of network state
  - Dependency model of states

The "State" in Statesman

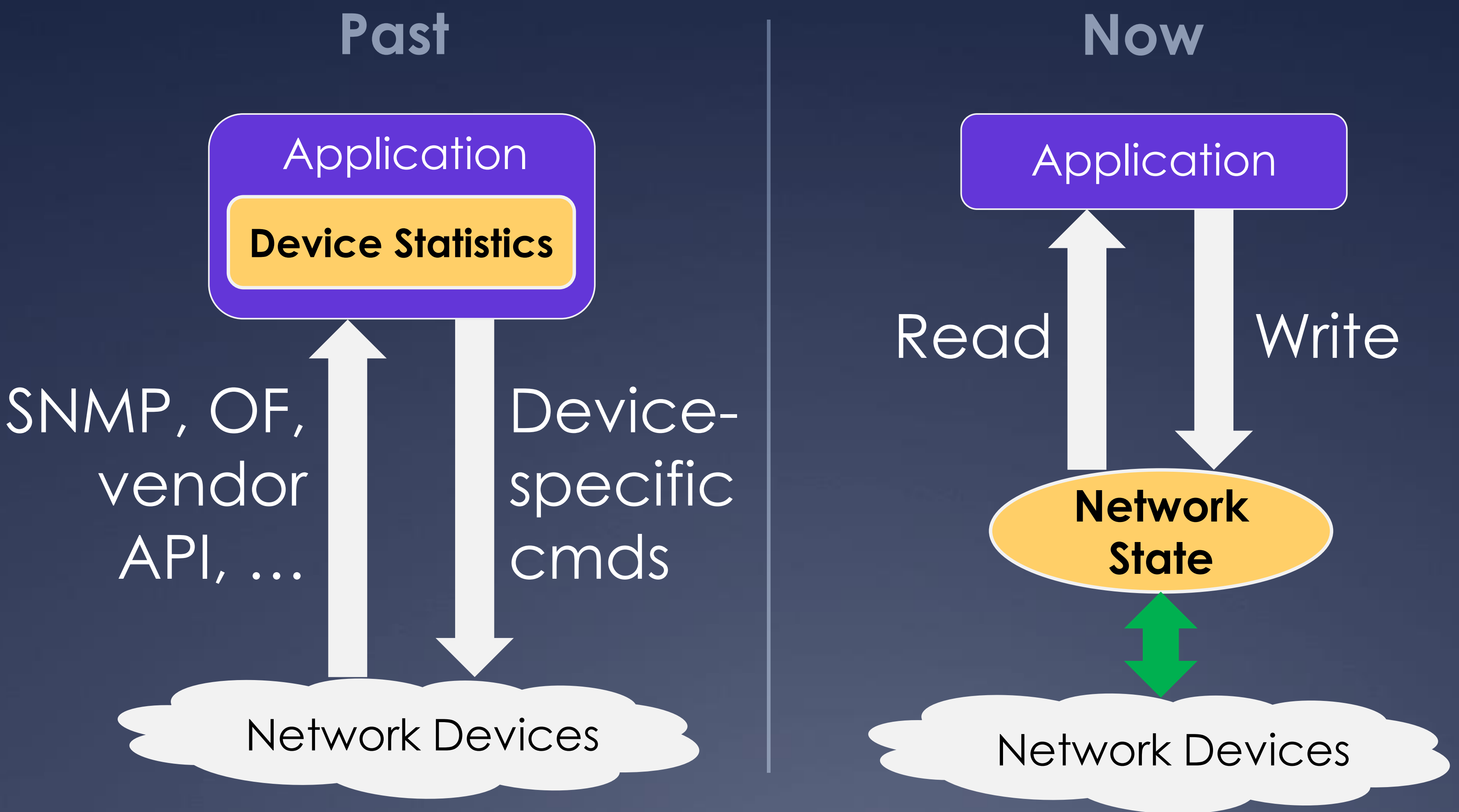
- Complexity of dealing with devices
  - Heterogeneity
  - Device-specific commands



State Variable Examples

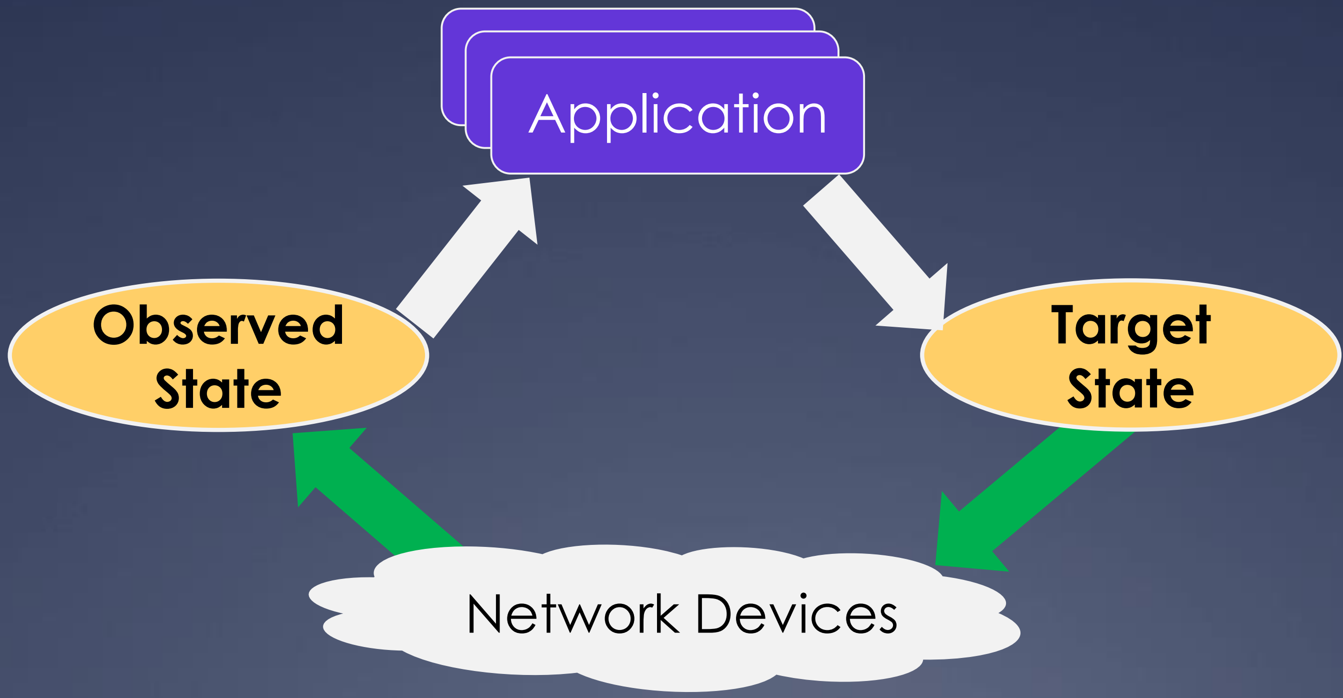
State Variable	Value
Device Power Status	Up, down
Device Firmware	Version number
Device SDN Agent Boot	Up, down
Device Routing State	Routing rules
Link Admin Status	Up, down
Link Control Plane	BGP, OpenFlow, …

Simplify Device Interaction

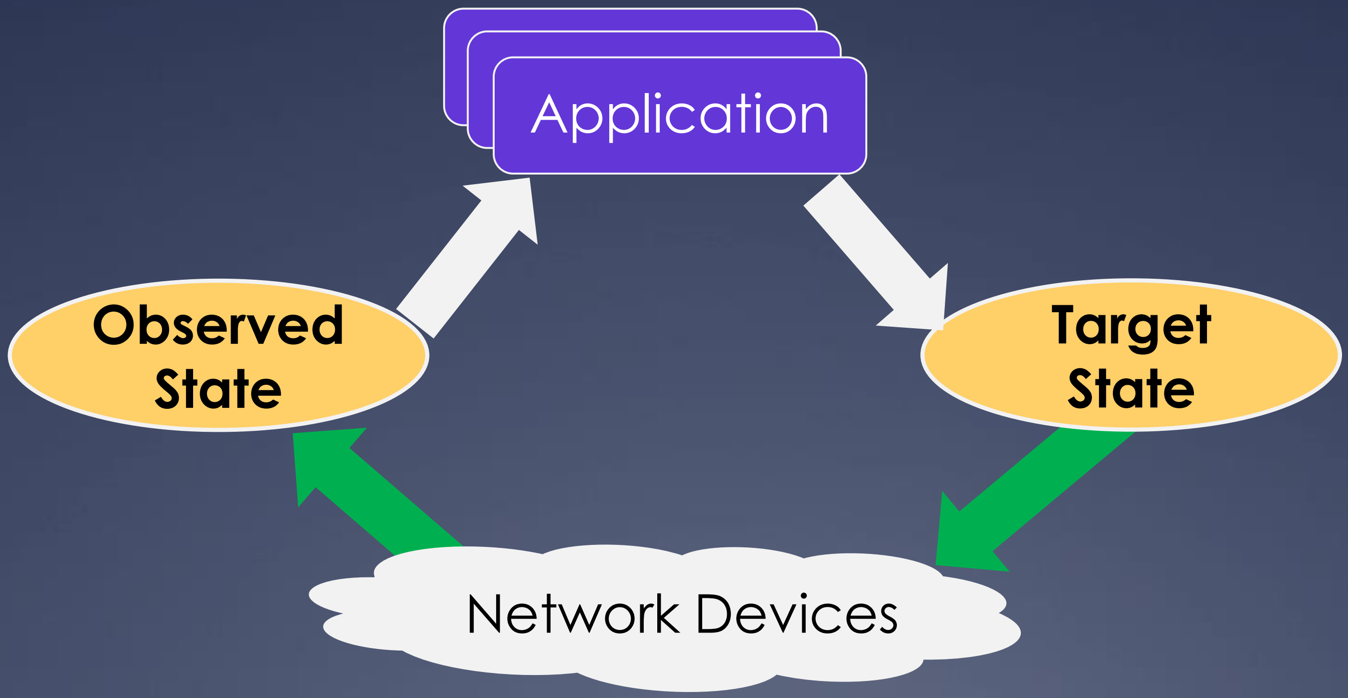


Views of Network State

Observed State	Actual state of the whole network
Target State	Desired state to be updated on the whole network

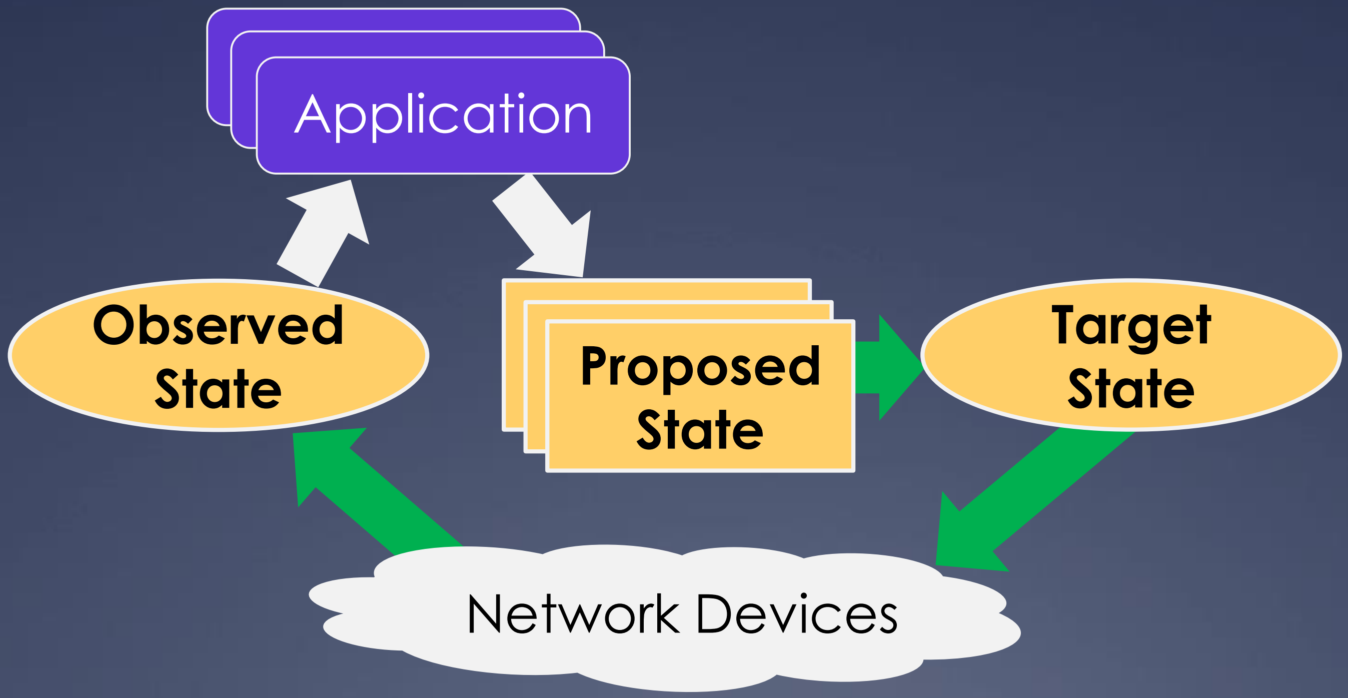


Two Views Are Not Enough



Two Views Are Not Enough

One More View	
Proposed State	A group of entity-variable-values desired by an application



How Merging Works

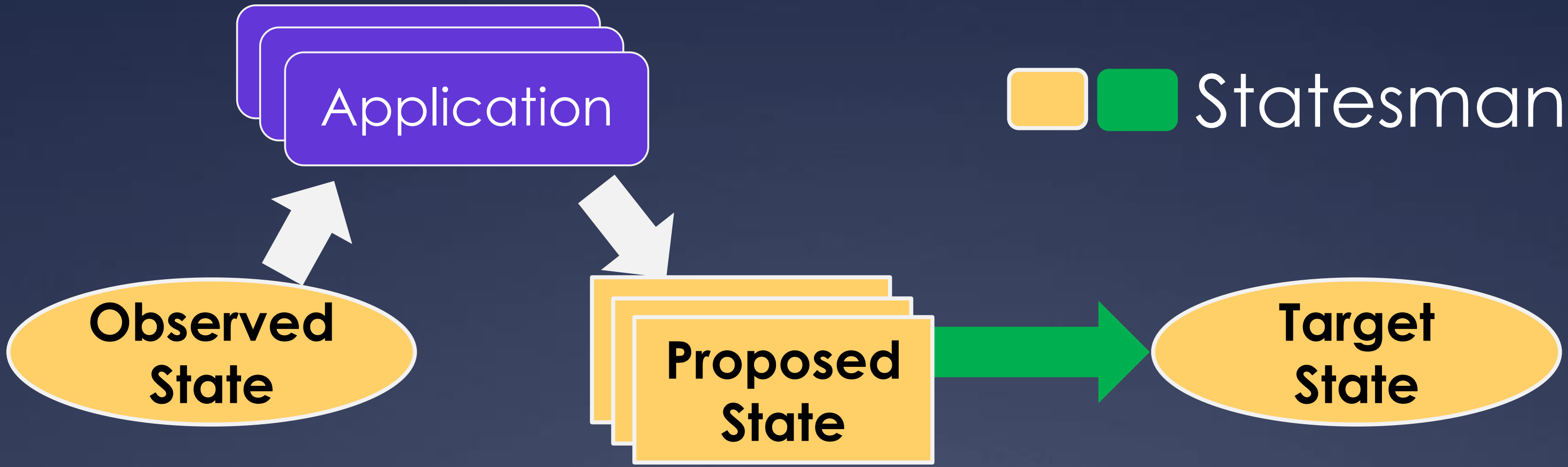
- Combine multiple proposed states into a safe target state
- Conflict resolution
  - Last-writer-wins
  - Priority-based locking
  - *Sufficient for current deployment*
- Safety invariant checking
  - Partial rejection & Skip update

Choose Safety Invariants

- Our current choice
  - Connectivity: Every pair of ToRs in one DC is connected
  - Capacity: 99% of ToR pairs have at least 50% capacity

Recap of Three-View Model

- Simplify network management



What we see from the network

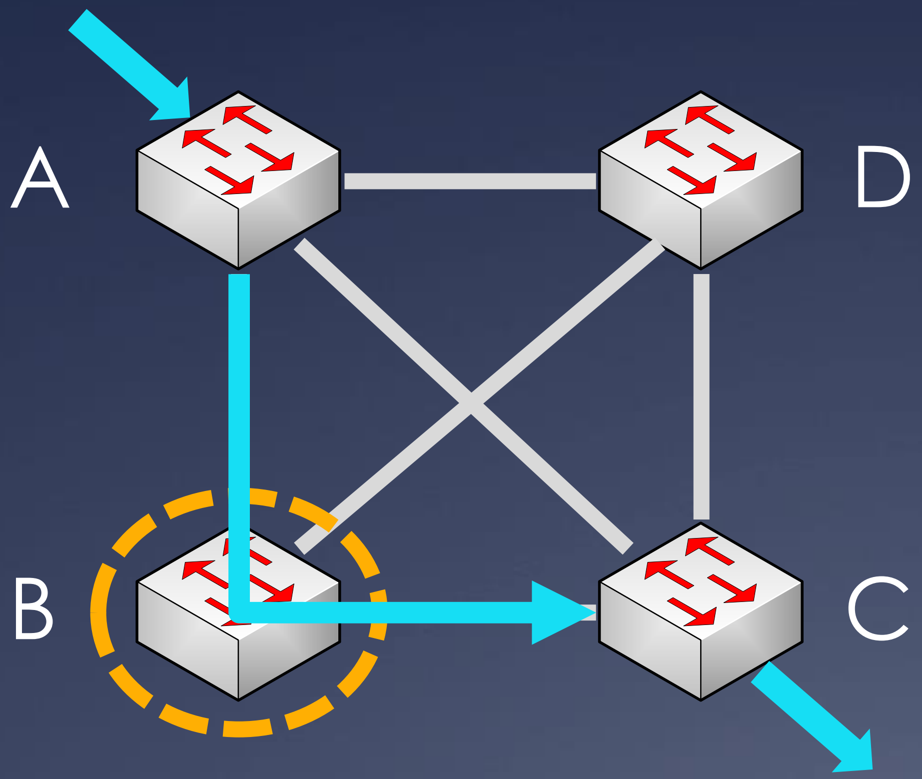
What we want the network to be

What can be actually done on the network

Yet Another Problem

- What’s in Proposed State
  - Small number of state variables that application cares
- Implicit conflicts arises
  - Caused by state dependency

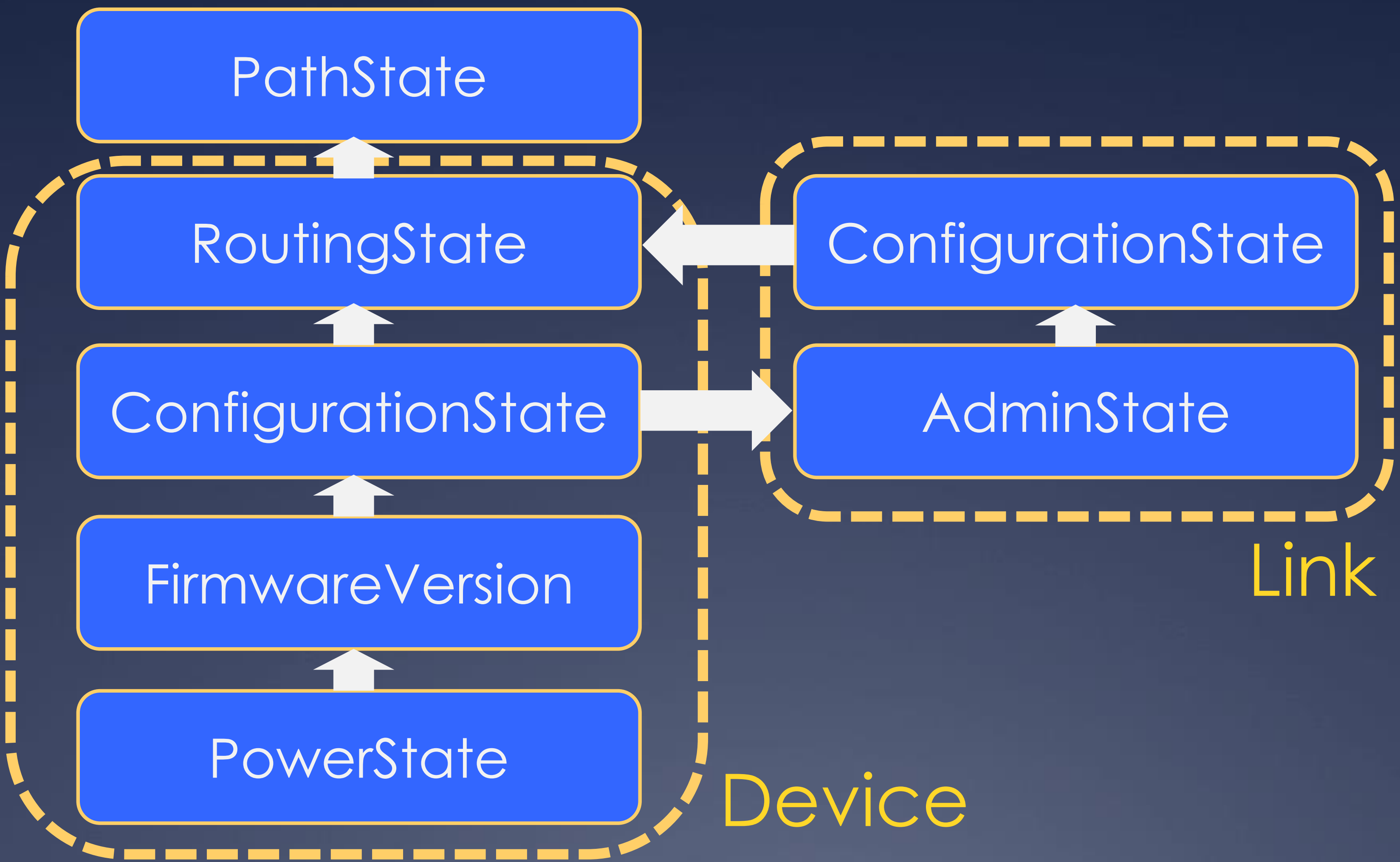
Implicit Conflict



TE writes new value of routing state of B for tunneling traffic

Firmware-upgrade writes new value of firmware state of B

Dependency Relations

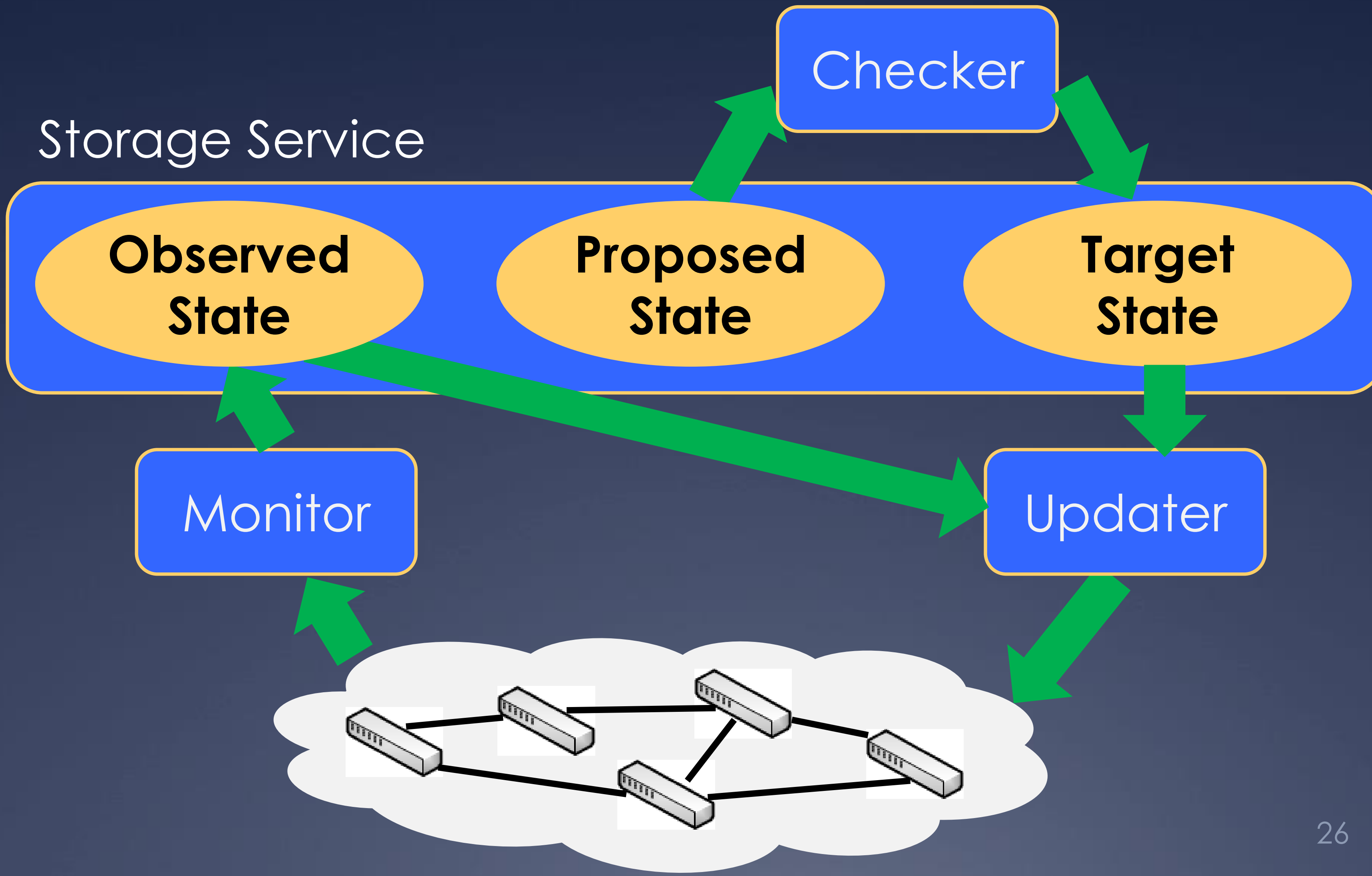


Build in Dependency Model

- Statesman calculates it internally

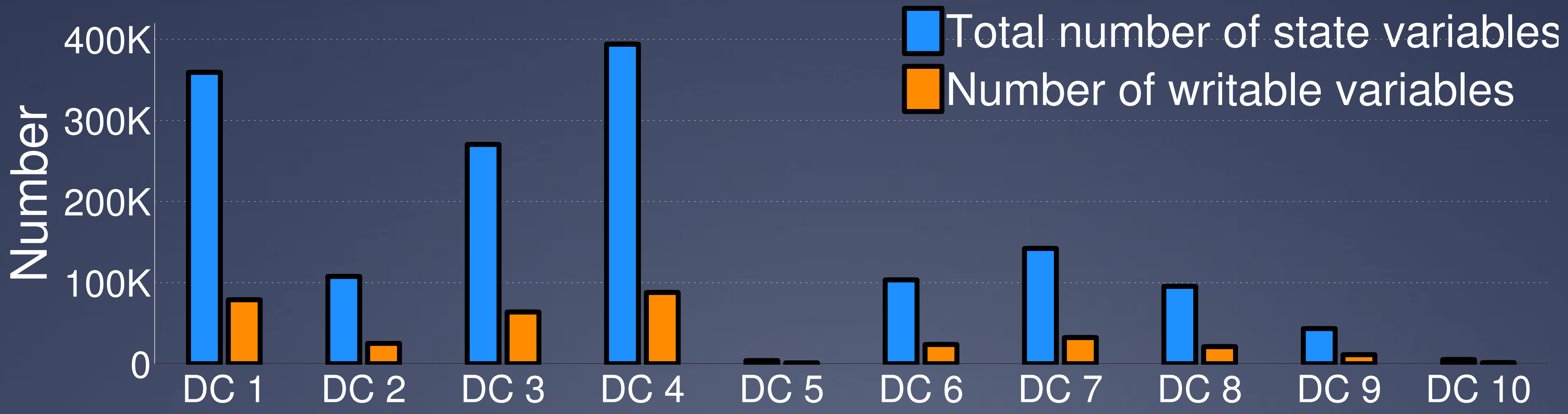
- Only exposes the result for each state variable
    - Whether the variable is controllable

Statesman System



Deployment Overview

- Operational in Microsoft Azure for 10 months
- Cover 10 DCs of 20K devices



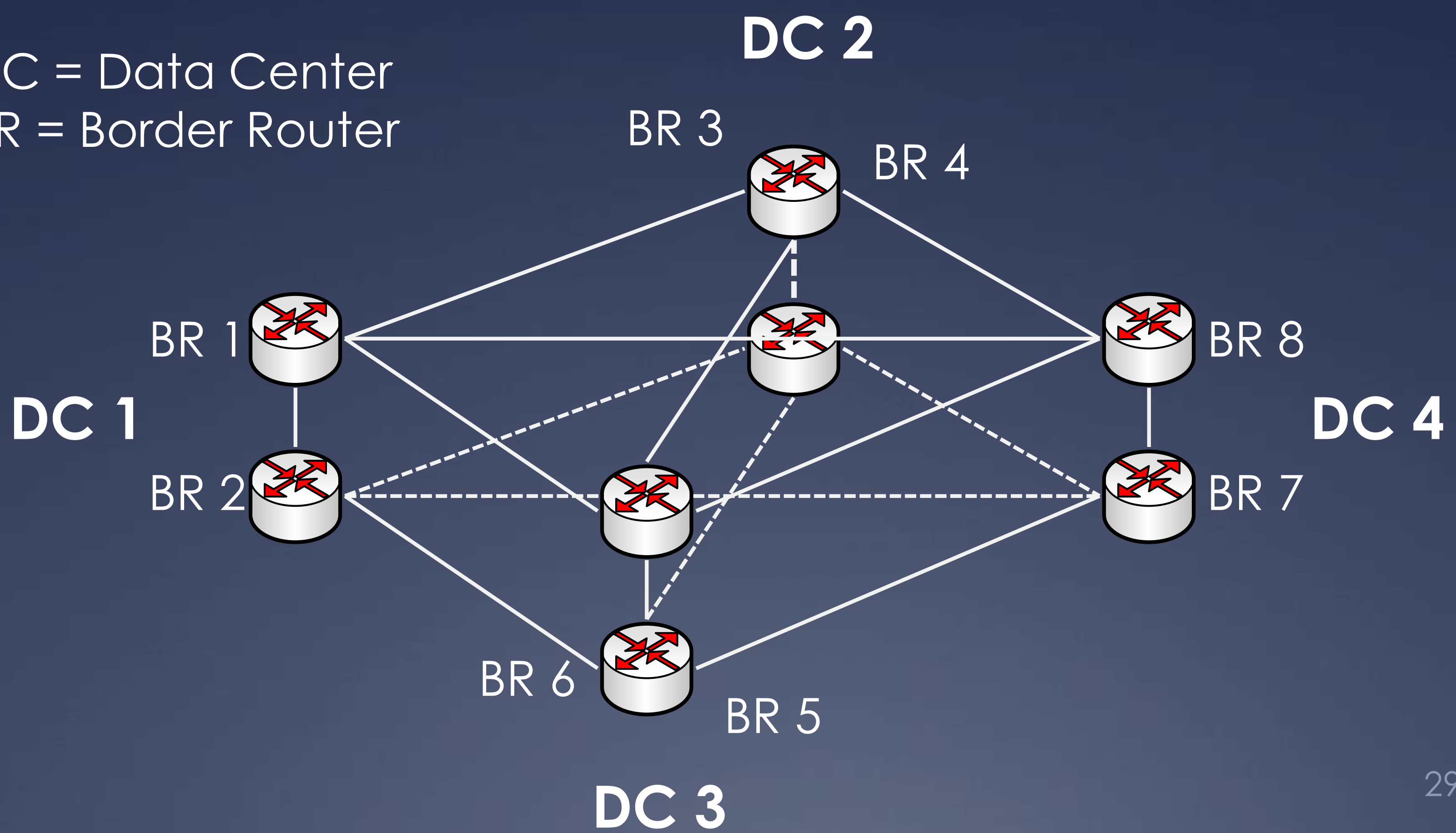
Production Applications

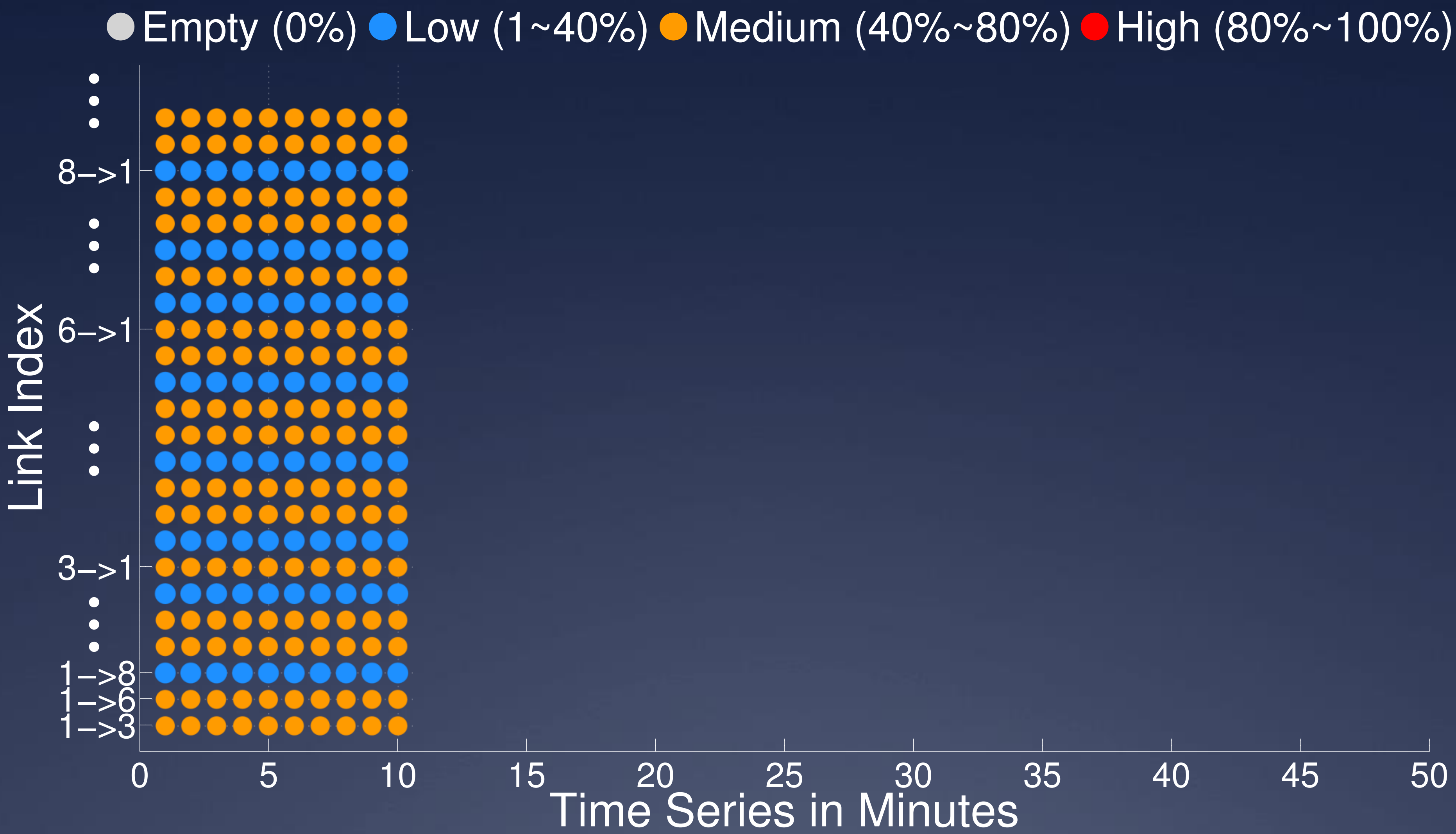
- 3 diverse applications built
  - Device firmware upgrade
  - Link corruption mitigation
  - Traffic engineering
- Finish within months
- Only thousands of lines of code

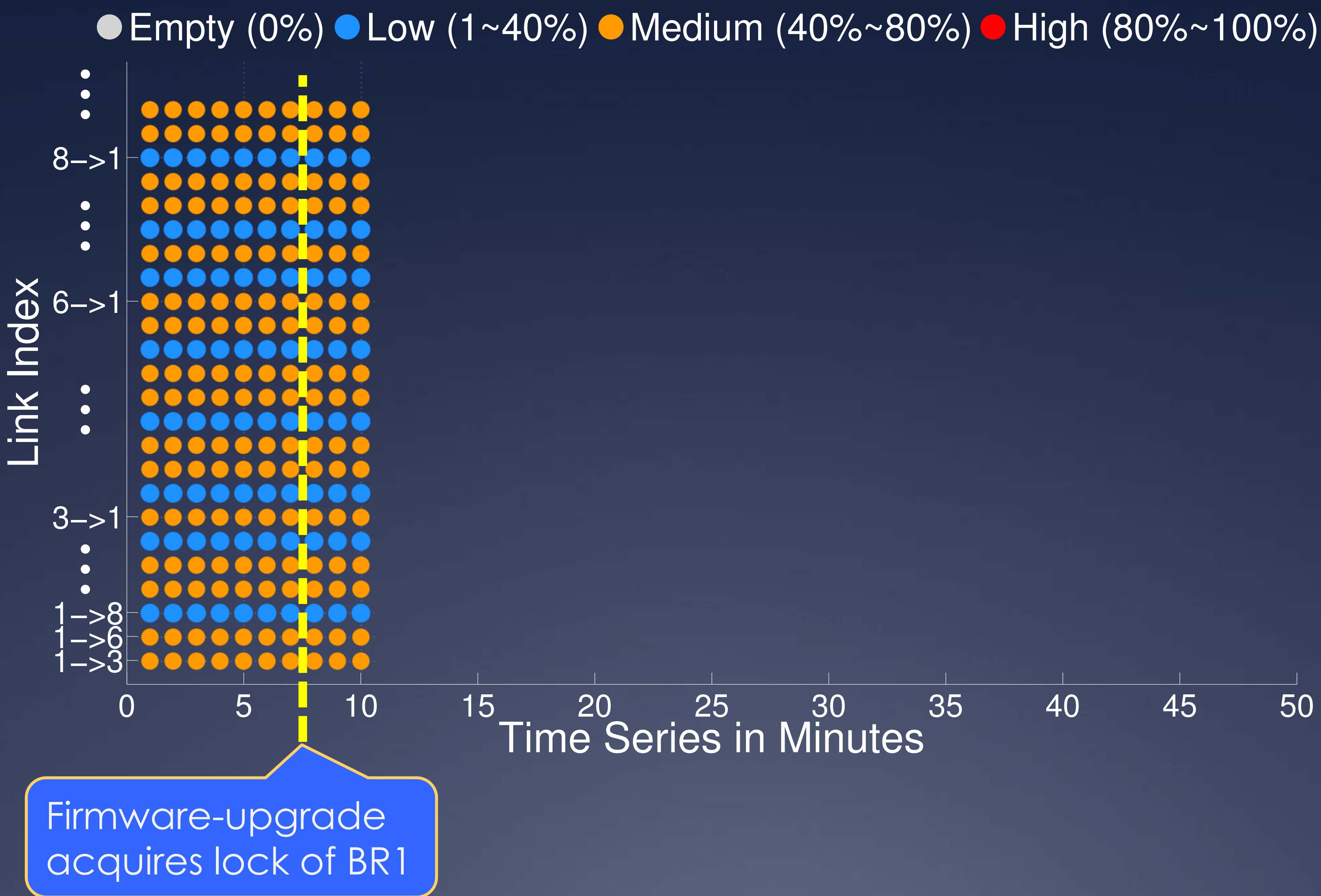
Case #1: Resolve Conflict

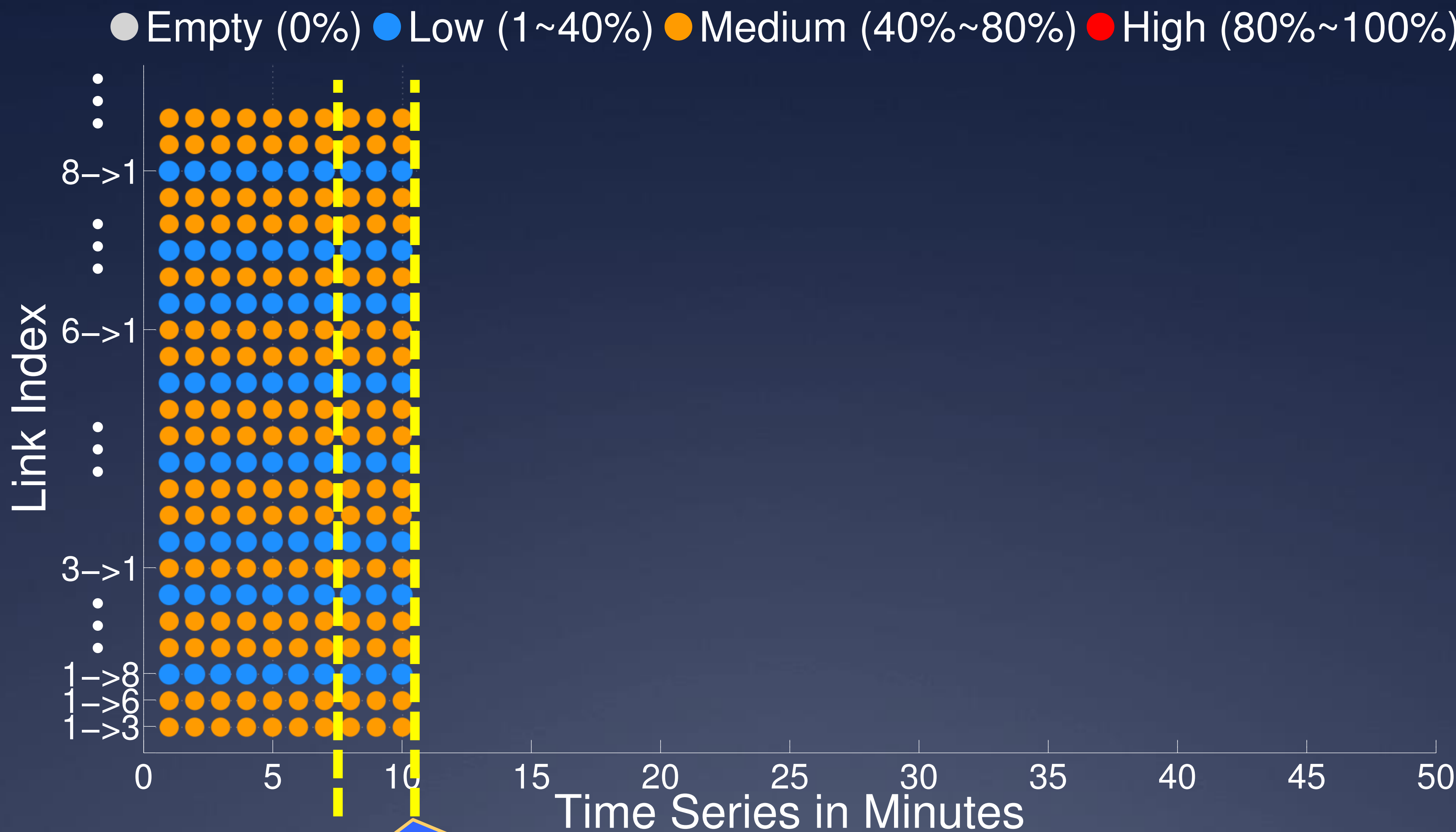
Inter-DC TE &
Firmware-upgrade

DC = Data Center
BR = Border Router

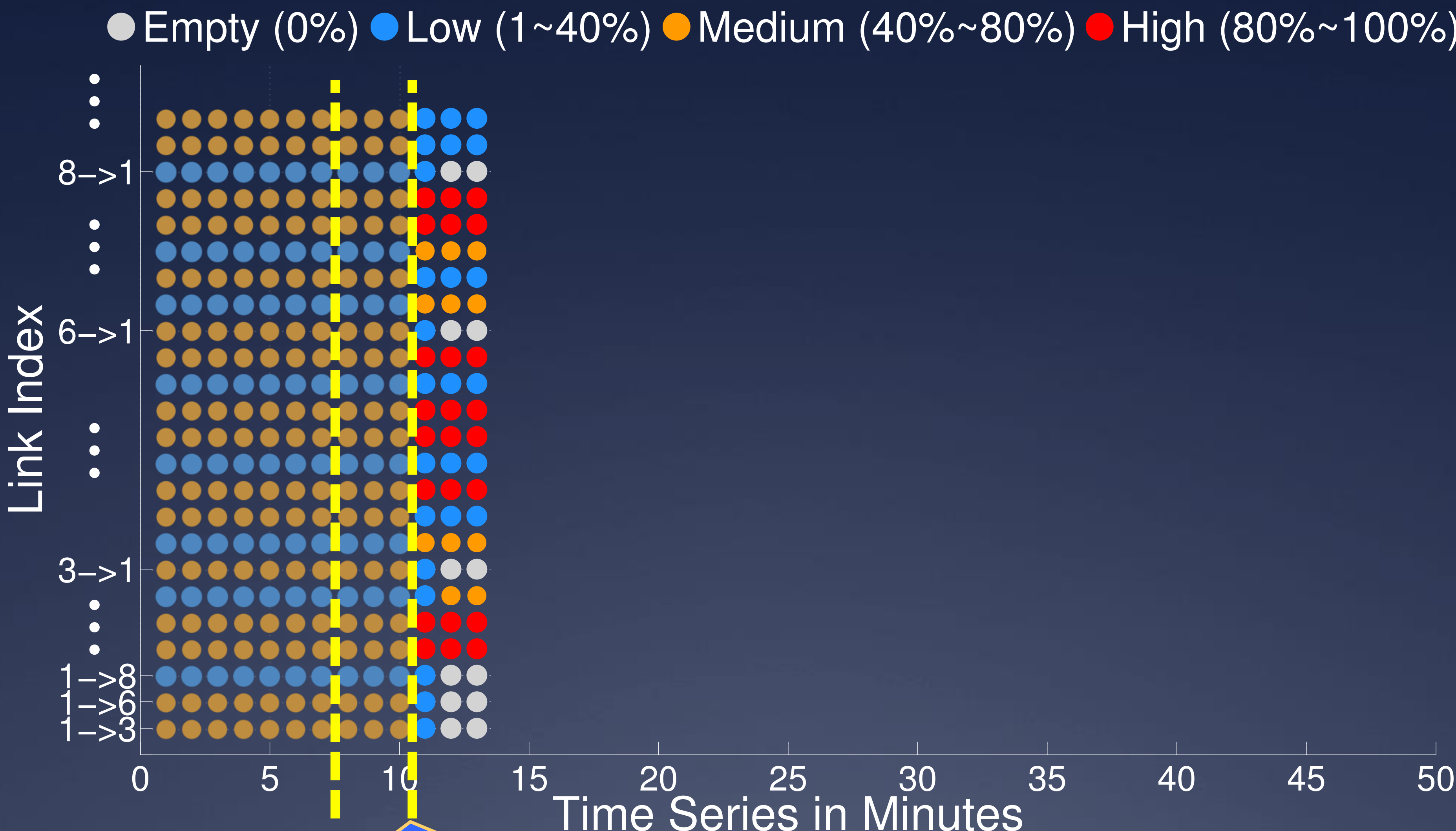



Empty (0%)
Low (1~40%)
Medium (40%~80%)
High (80%~100%)
Link Index
8–>1
6–>1
3–>1
1–>8
1–>6
1–>3
0
5
10
15
20
25
30
35
40
45
50
Time Series in Minutes

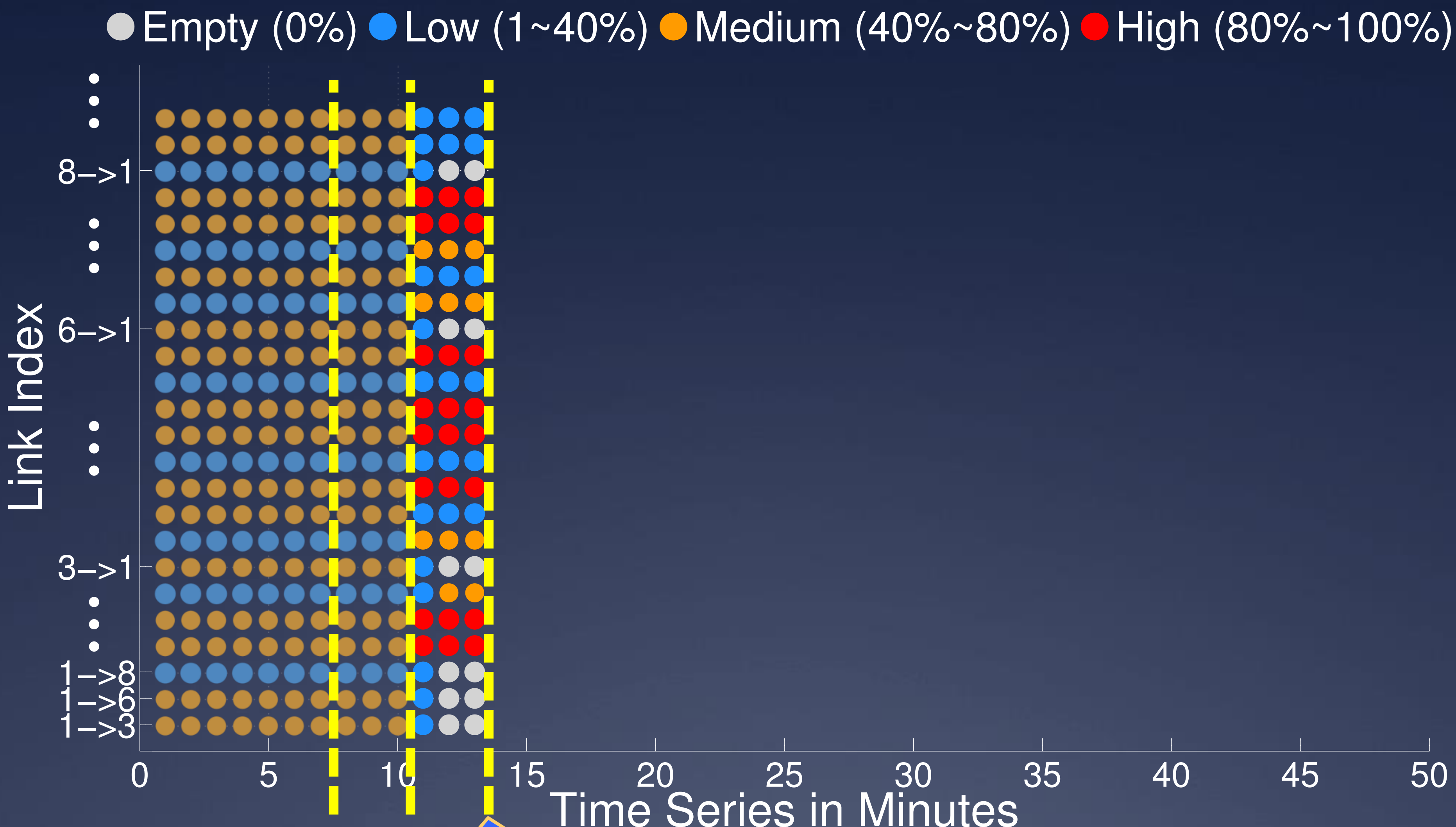

Empty (0%)
Low (1~40%)
Medium (40%~80%)
High (80%~100%)
Link Index
8−>1
6−>1
3−>1
1−>8
1−>6
1−>3
0
5
10
15
20
25
30
35
40
45
50
Time Series in Minutes
Firmware-upgrade acquires lock of BR1



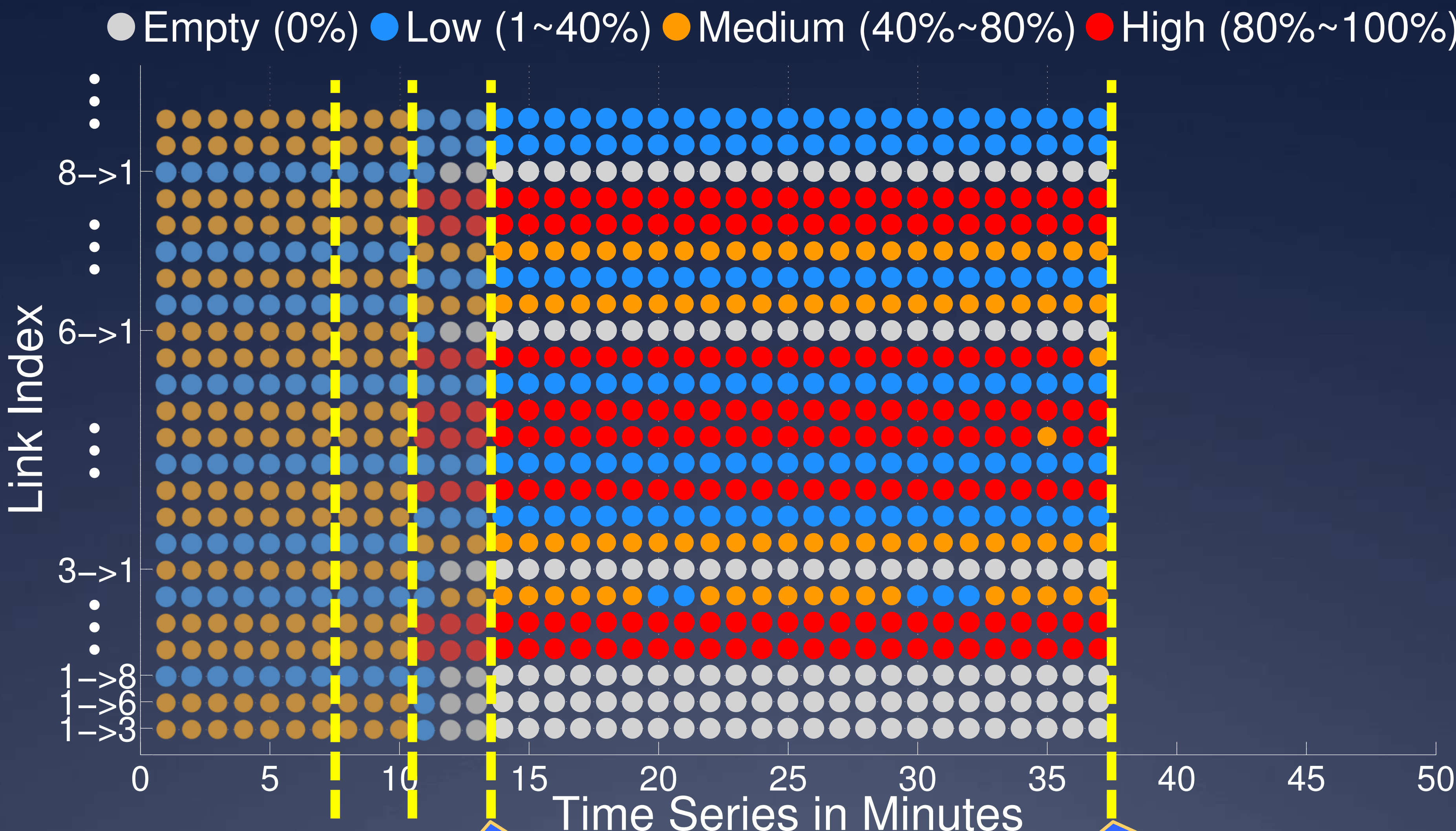
TE fails to acquire lock, and moves traffic away



TE fails to acquire lock, and moves traffic away

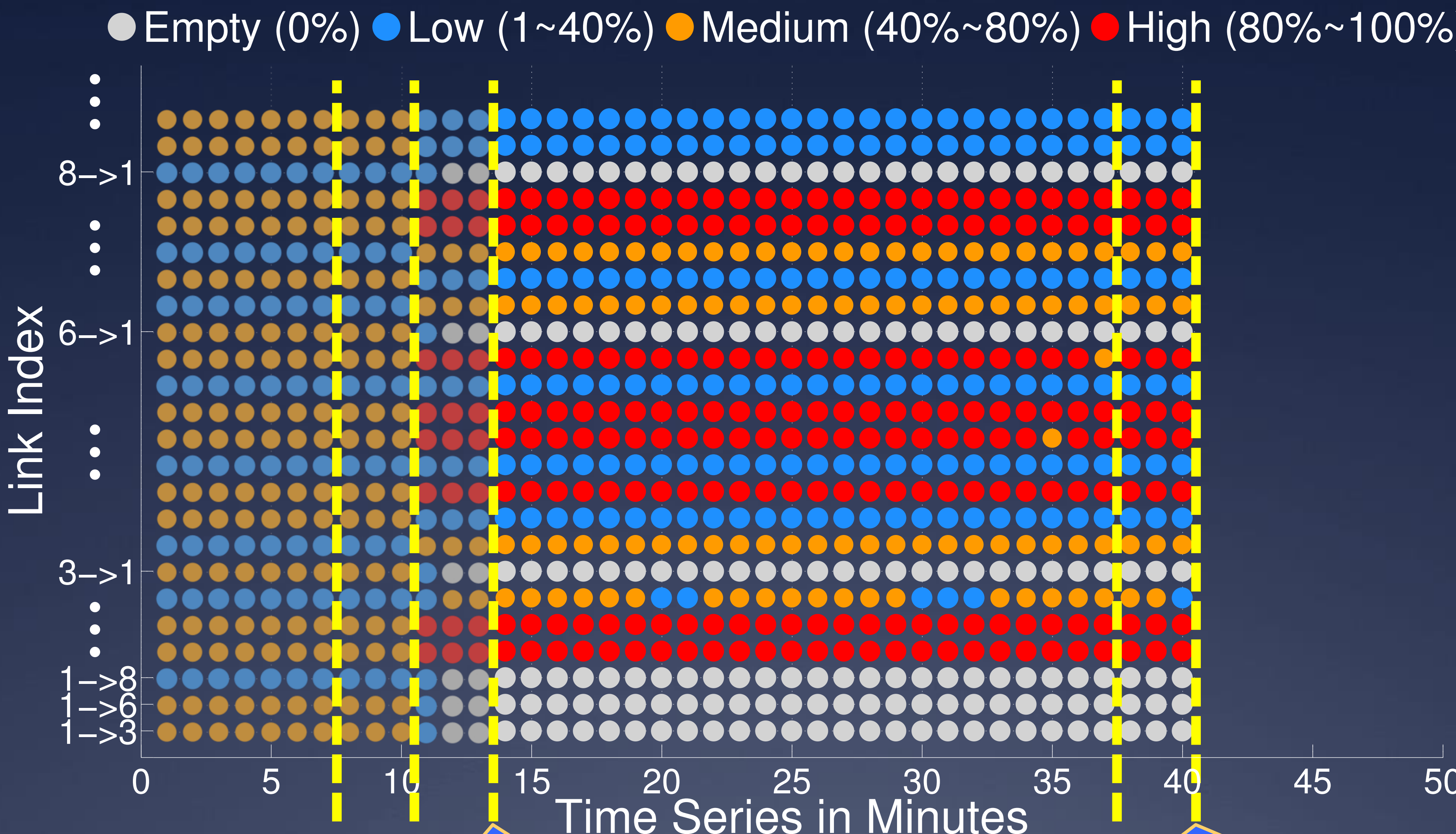


BR1 firmware upgrade starts



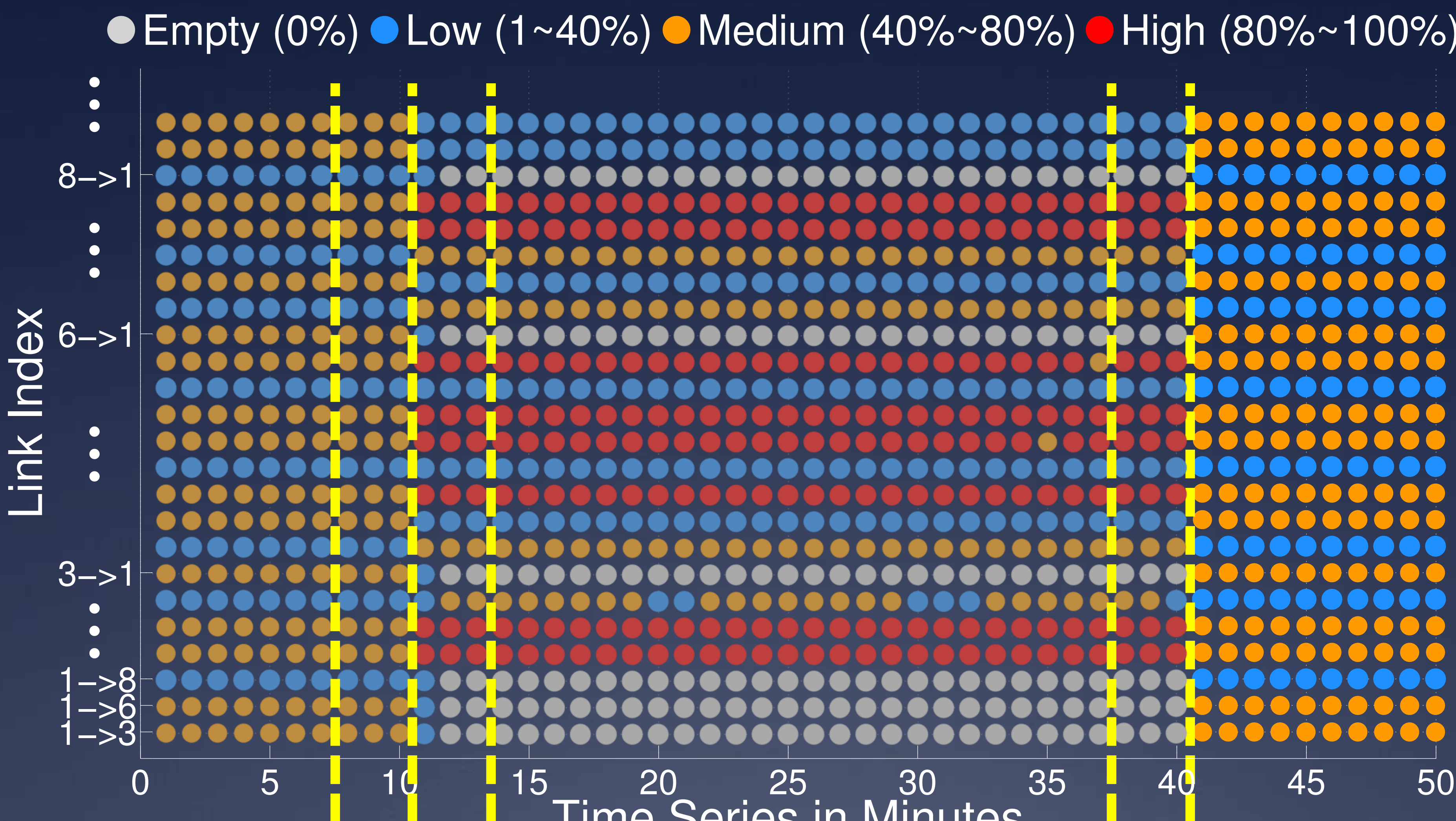
BR1 firmware upgrade starts

BR1 firmware upgrade ends. Lock released.



BR1 firmware upgrade starts

TE re-acquires lock, and moves traffic back



BR1 firmware upgrade starts

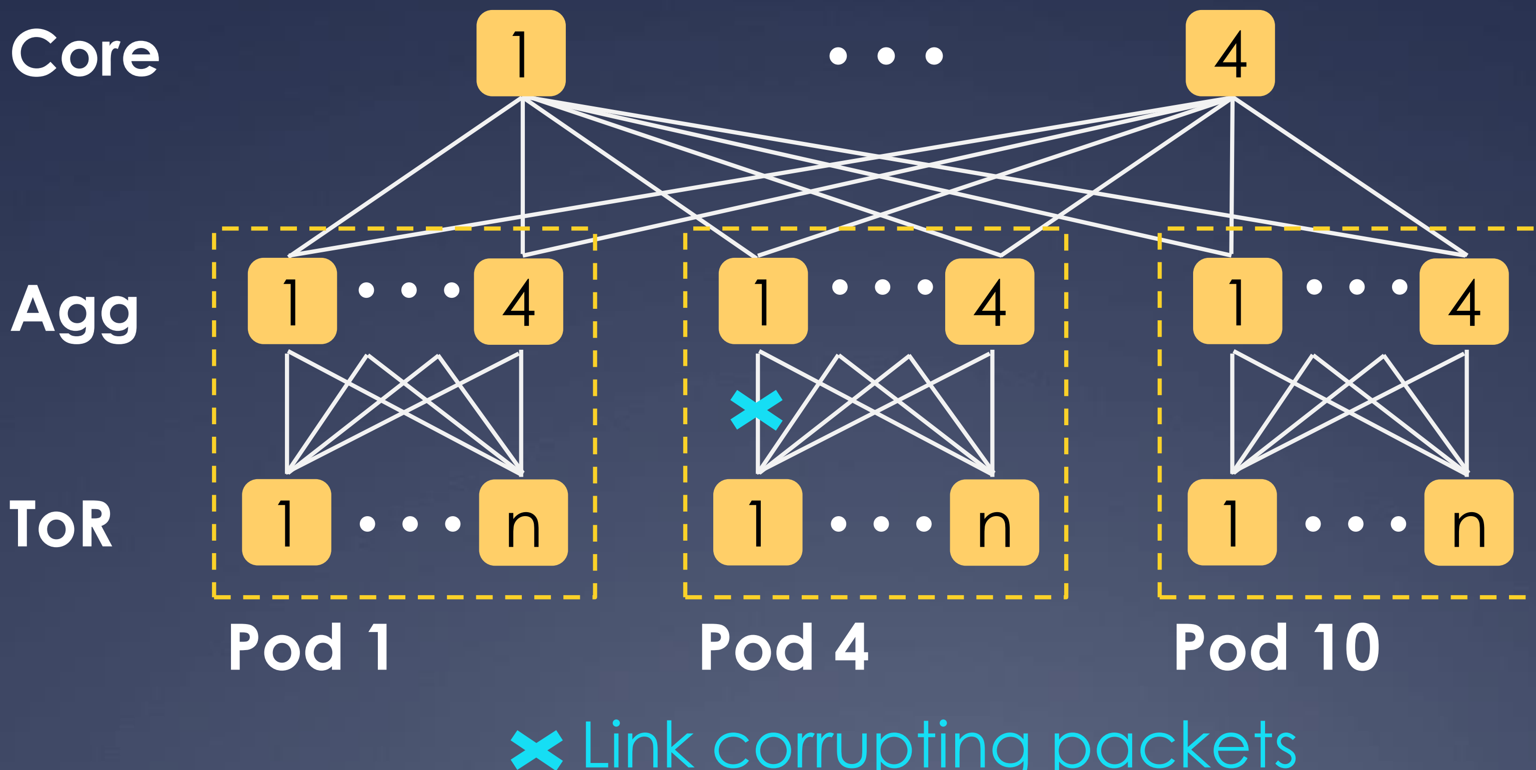
TE re-acquires lock, and moves traffic back

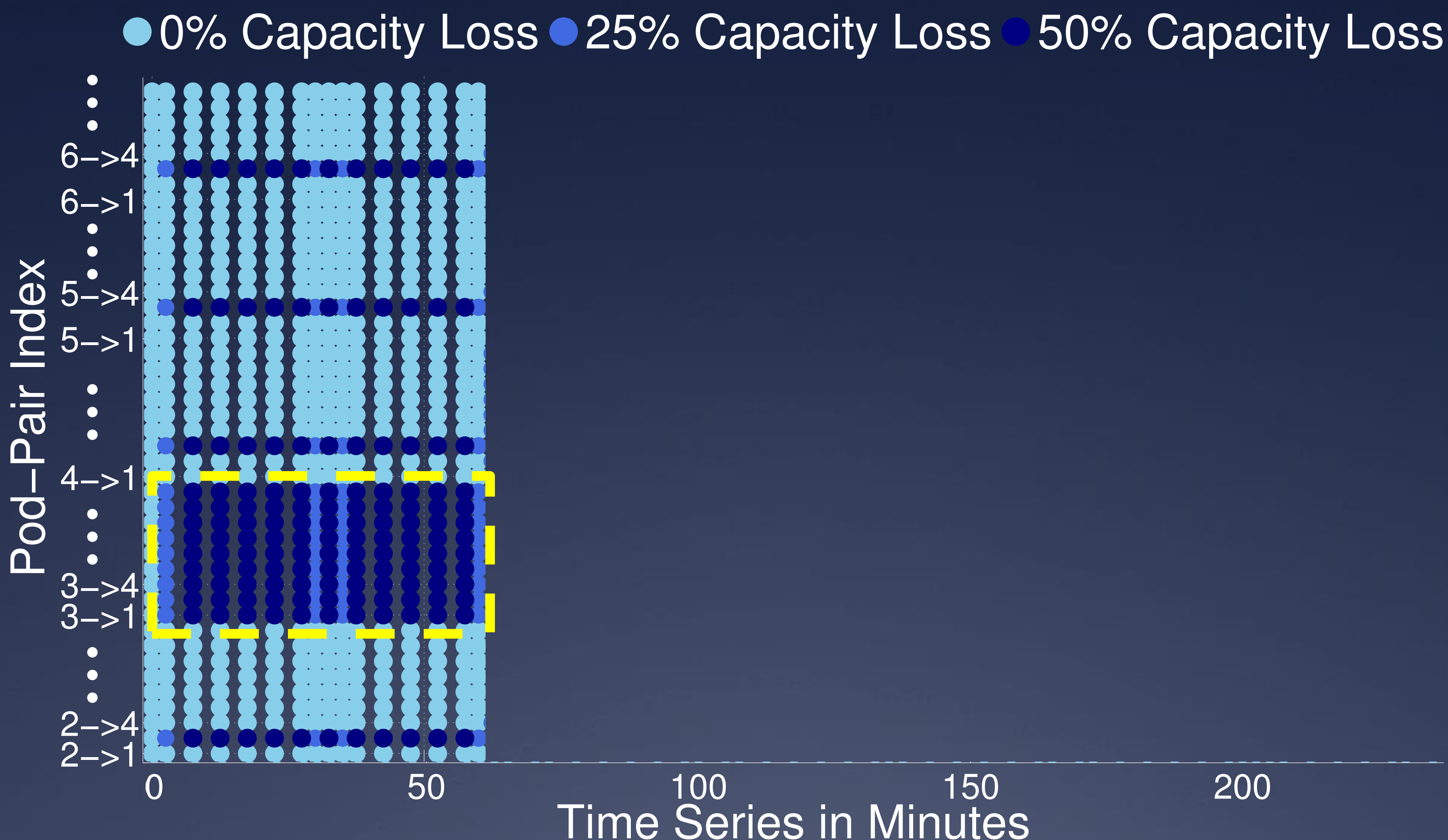
Case #1 Summary

- Each application:
  - Simple logic
  - Unaware of the other
- Statesman enables:
  - Conflict resolution
  - Necessary coordination

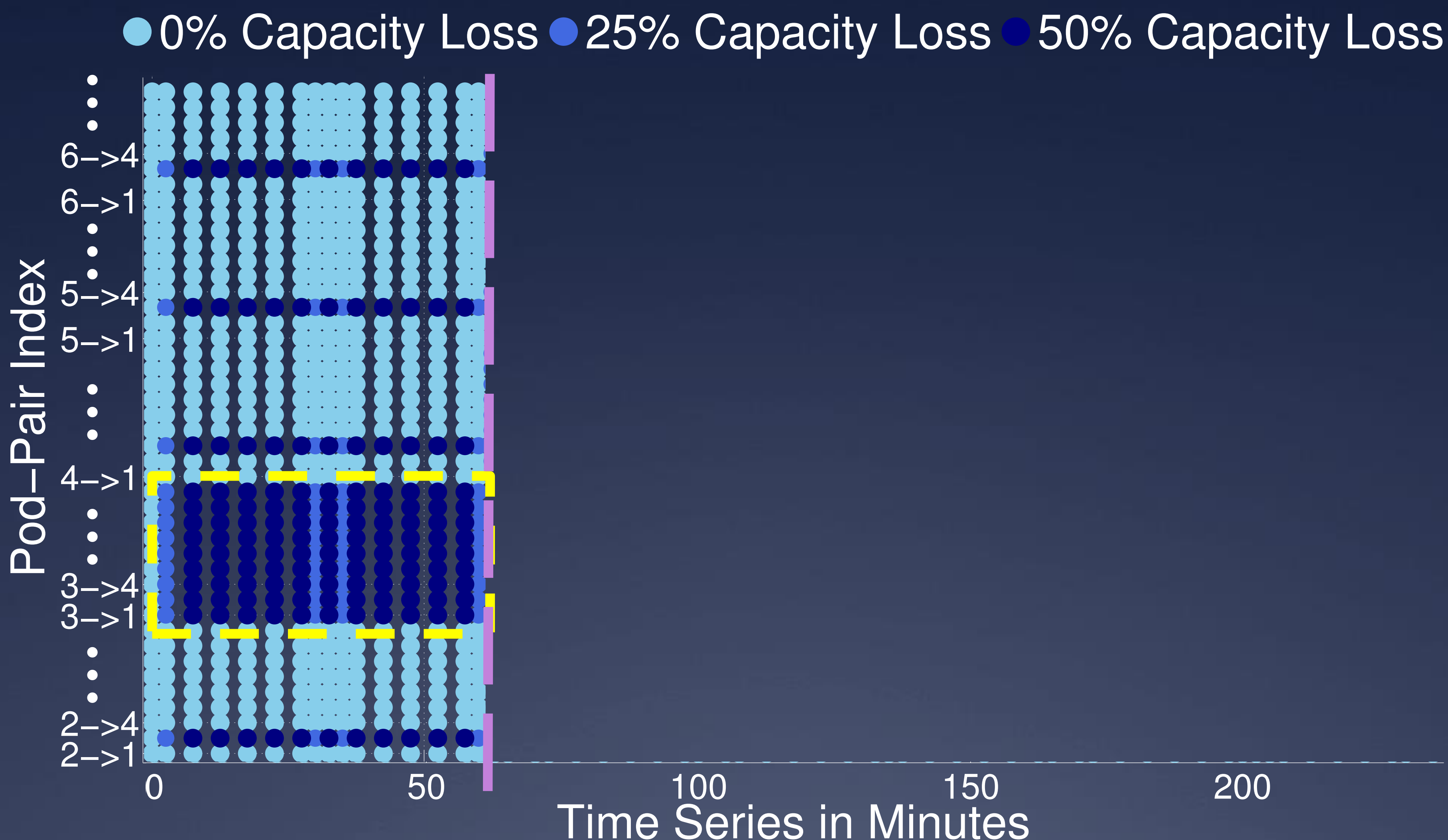
Case #2: Maintain Capacity Invariant

Firmware-upgrade &
Link-corruption-mitigation

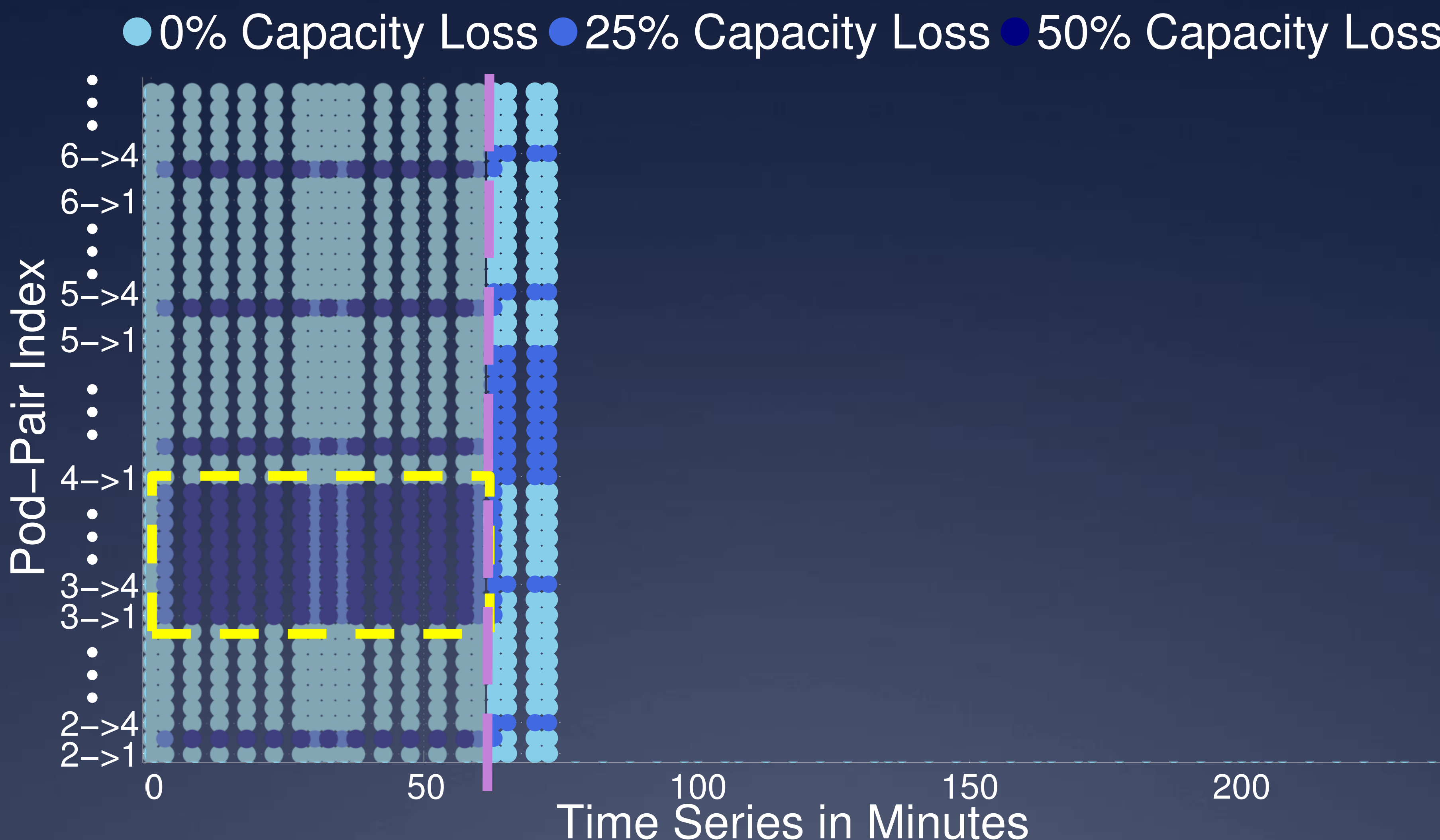




Upgrade proceeds in normal speed in Pod 3 and 5



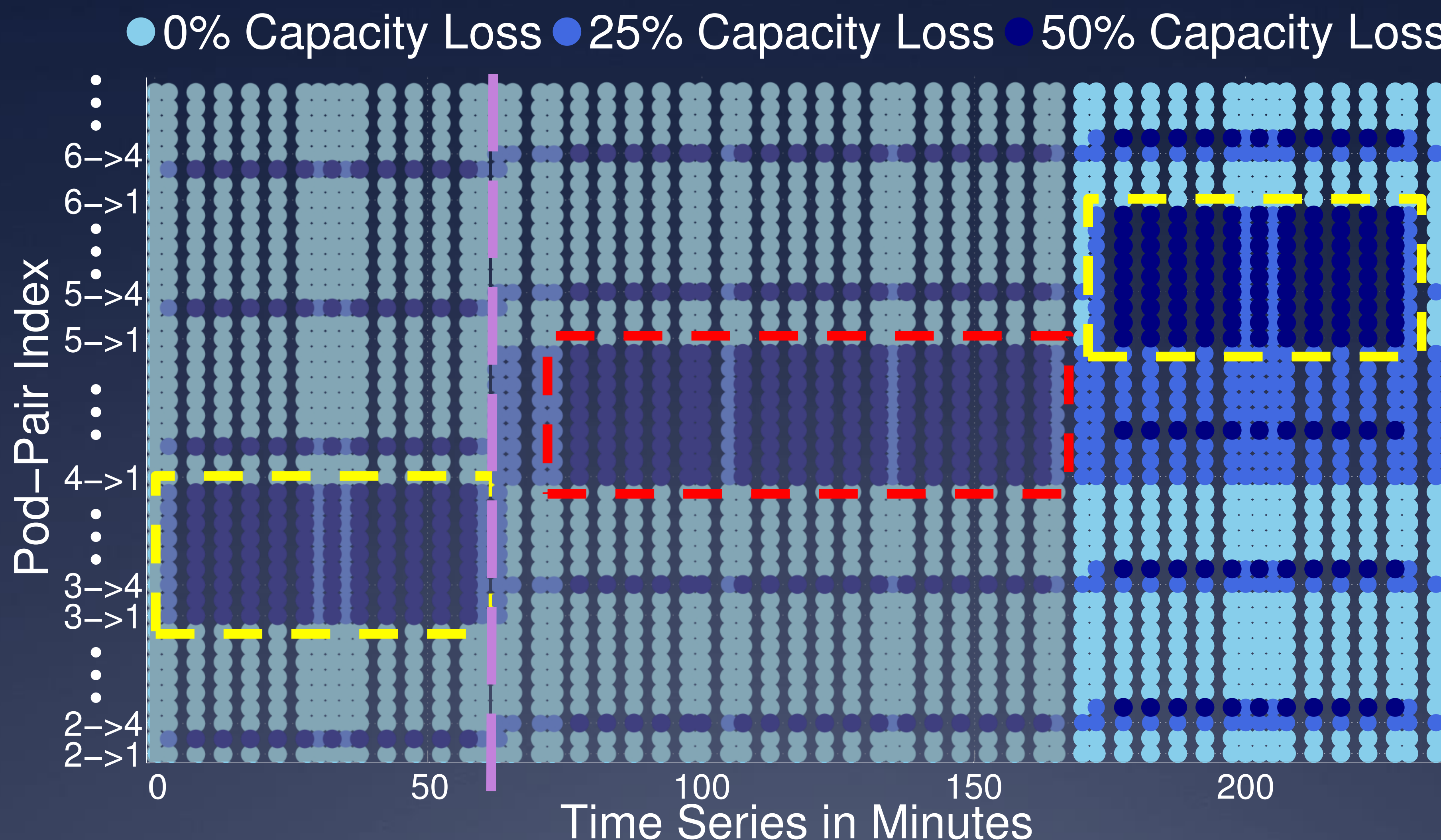
Upgrade proceeds in normal speed in Pod 3 and 5



Upgrade proceeds in normal speed in Pod 3 and 5

Upgrade proceeds in normal speed in Pod 3 and 5

Upgrade in Pod 4 is slowed down by checker due to lost capacity



Upgrade proceeds in normal speed in Pod 3 and 5

Upgrade in Pod 4 is slowed down by checker due to lost capacity

Case #2 Summary

- Statesman:
  - Automatically adjusts application progresses
  - Keeps the network within safety requirements

Conclusion

- Need network operating system for multiple management applications
- Statesman
  - Loose coupling of applications
  - Network state abstraction
- Deployed and operational in Azure

Thanks!

Questions?

Check paper for related works