Software Transactional Networking:

Concurrent and Consistent Policy Composition

Dan Levin

Marco Canini, Petr Kuznetsov*, Stefan Schmid

TU Berlin/T-Labs, *Telecom ParisTech

3

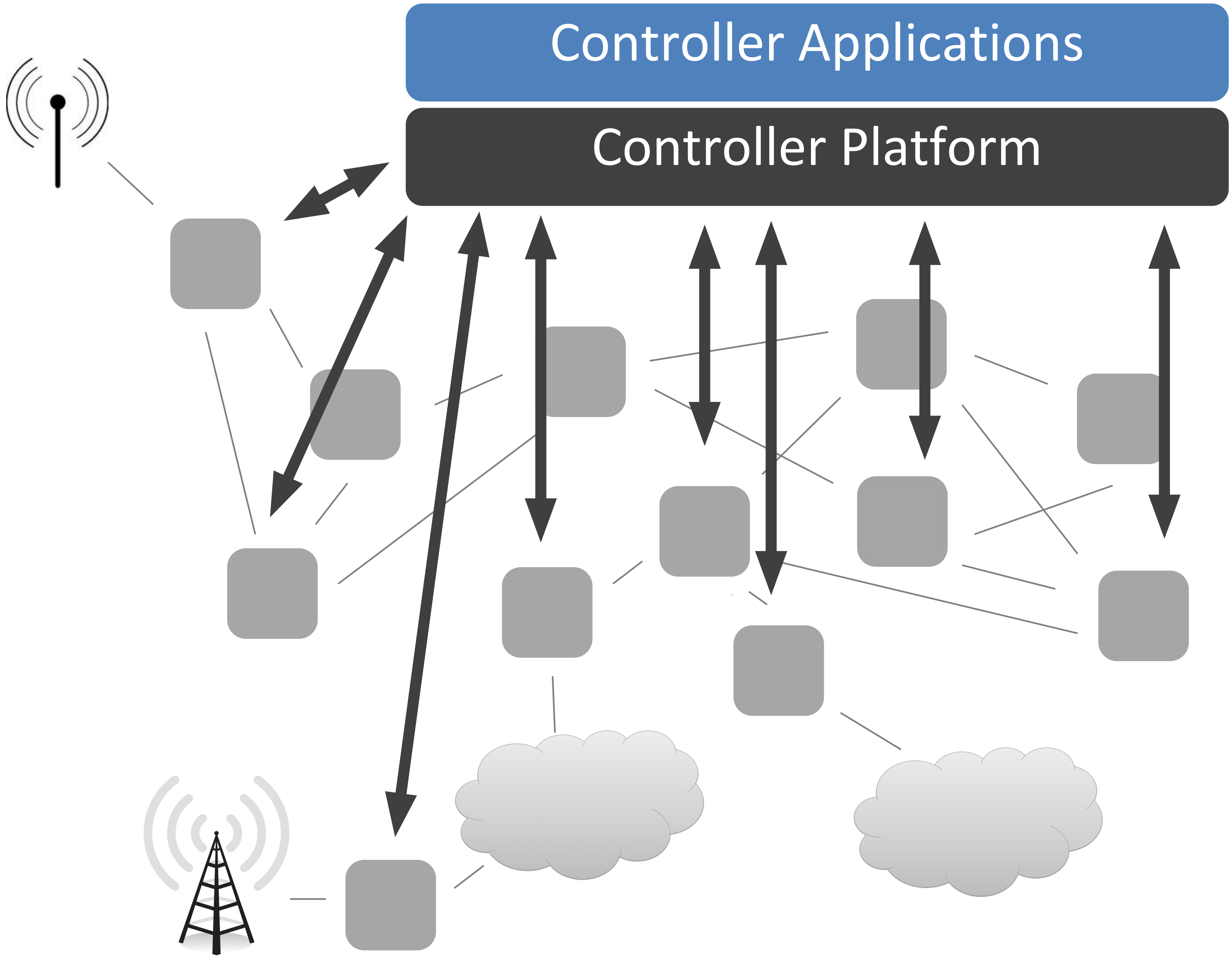
Software Transactional Networking:

Concurrent and Consistent Policy Composition

2

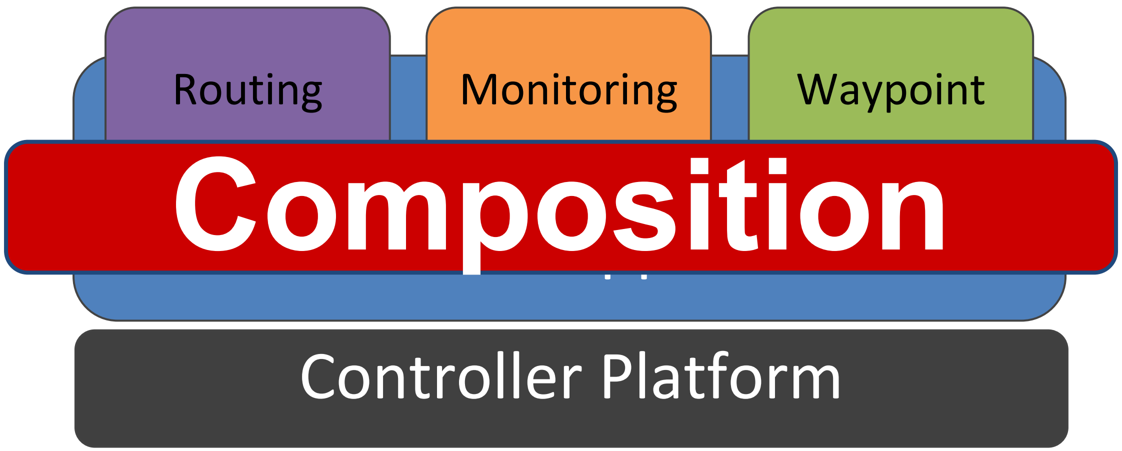
1

Network Policy Specification



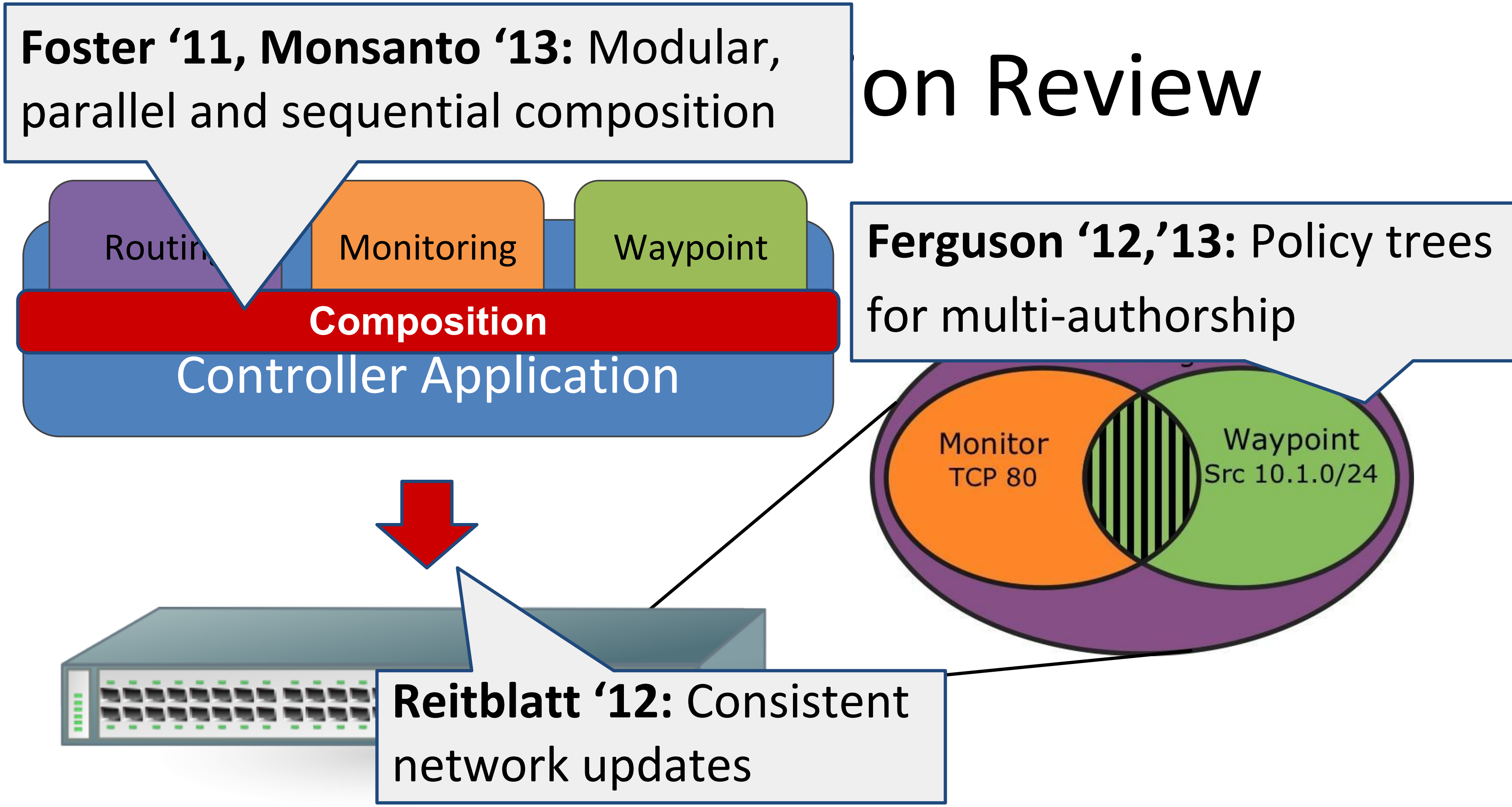
Network-wide Policy: Not Monolithic

Policy may originate from **multiple authors**, defined across **multiple functional modules**.



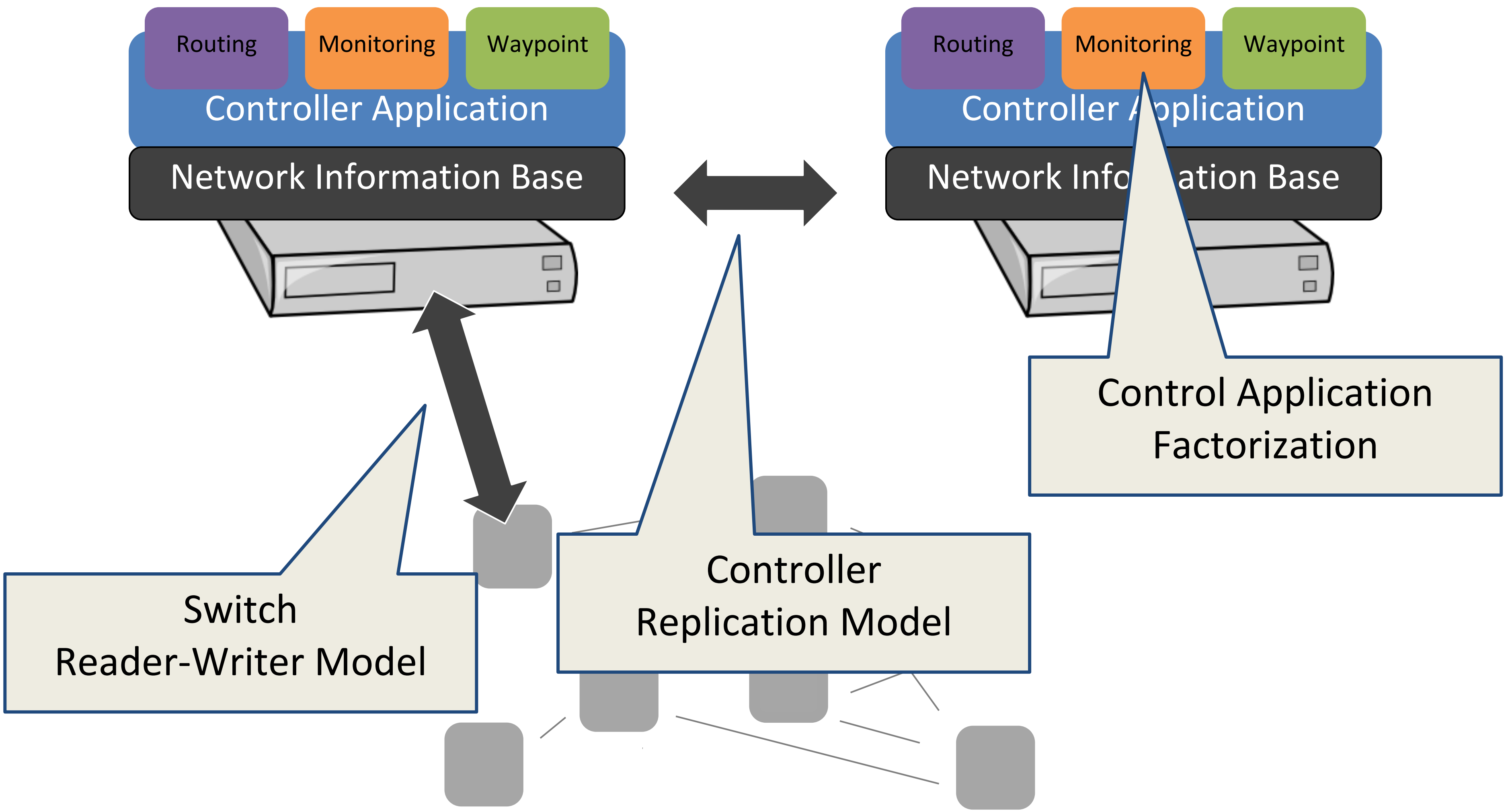
...necessitates policy **composition prior to network update.**

on Review



1. Precedence must be defined across policy sources
2. Packet forwarding rule priorities must be defined, and respect policy source precedence

Now, consider policy composition in the distributed control plane...



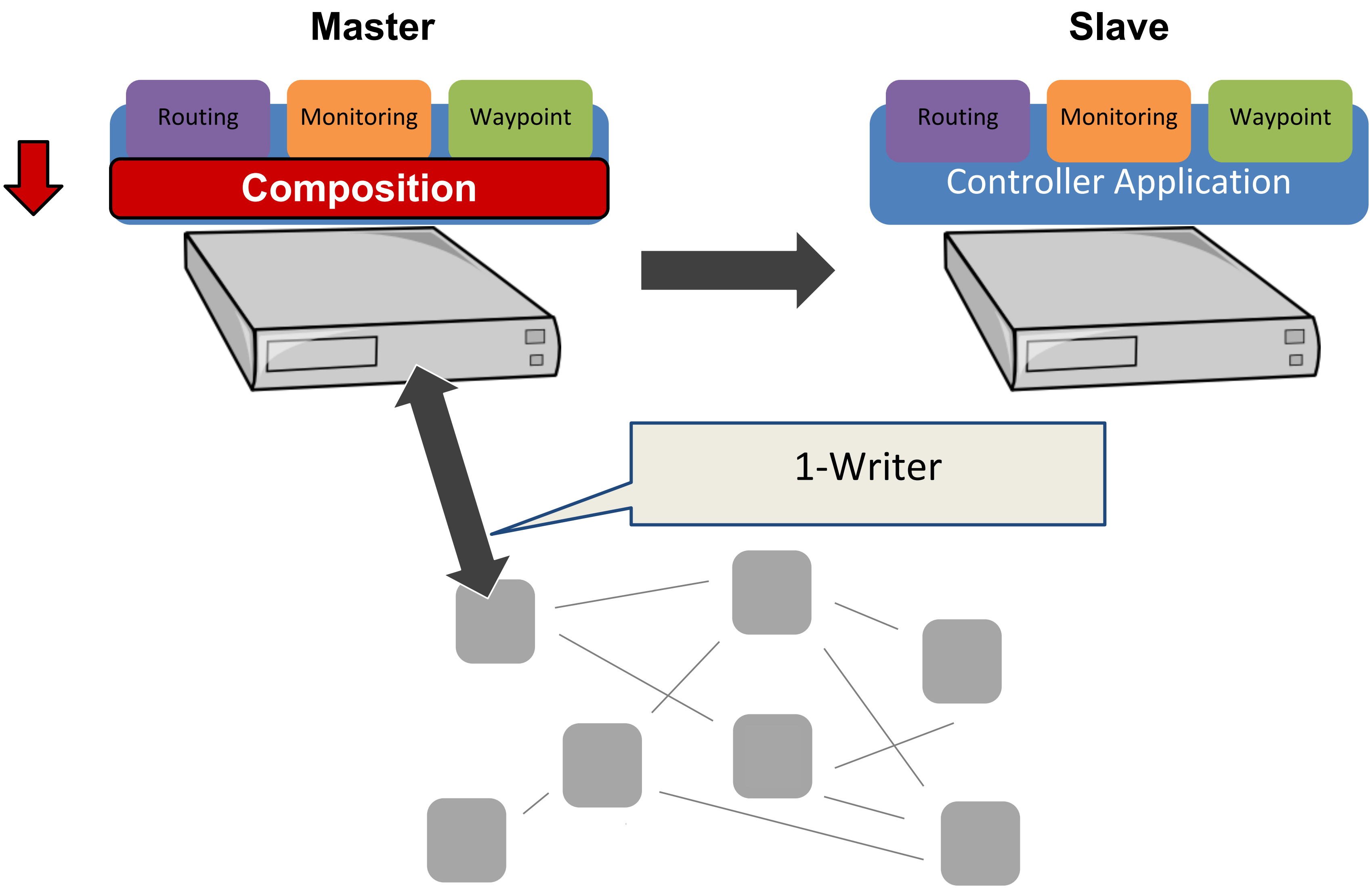
Research Question #1

How does the design of the
distributed control plane
affect policy composition with respect to:

1. The consistency of the composition
2. The semantics of consistent network update

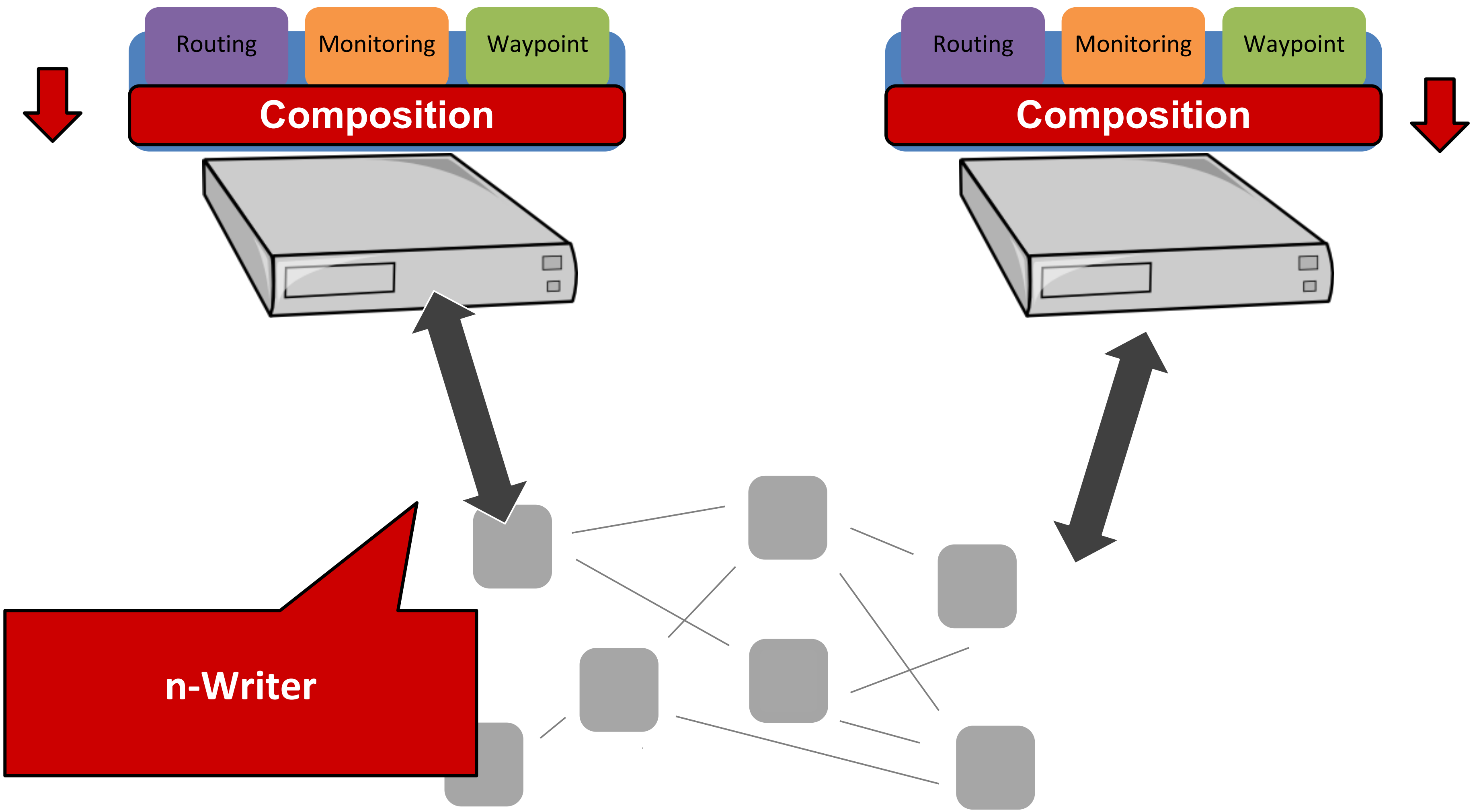
Distributed Control Models

1: Hot Standby Replication



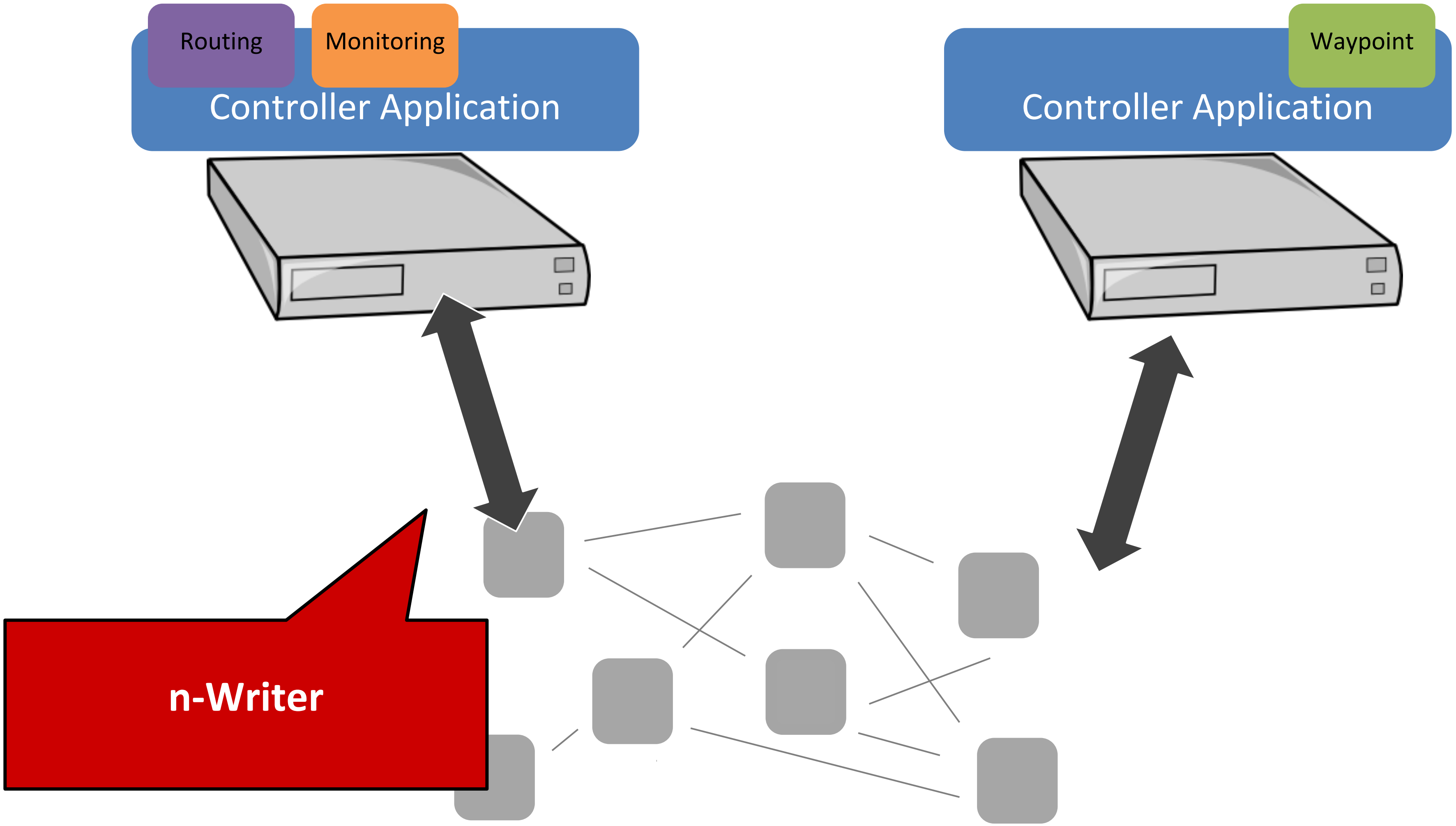
Distributed Control Models

2: Sharding by Disjoint Flow-Space

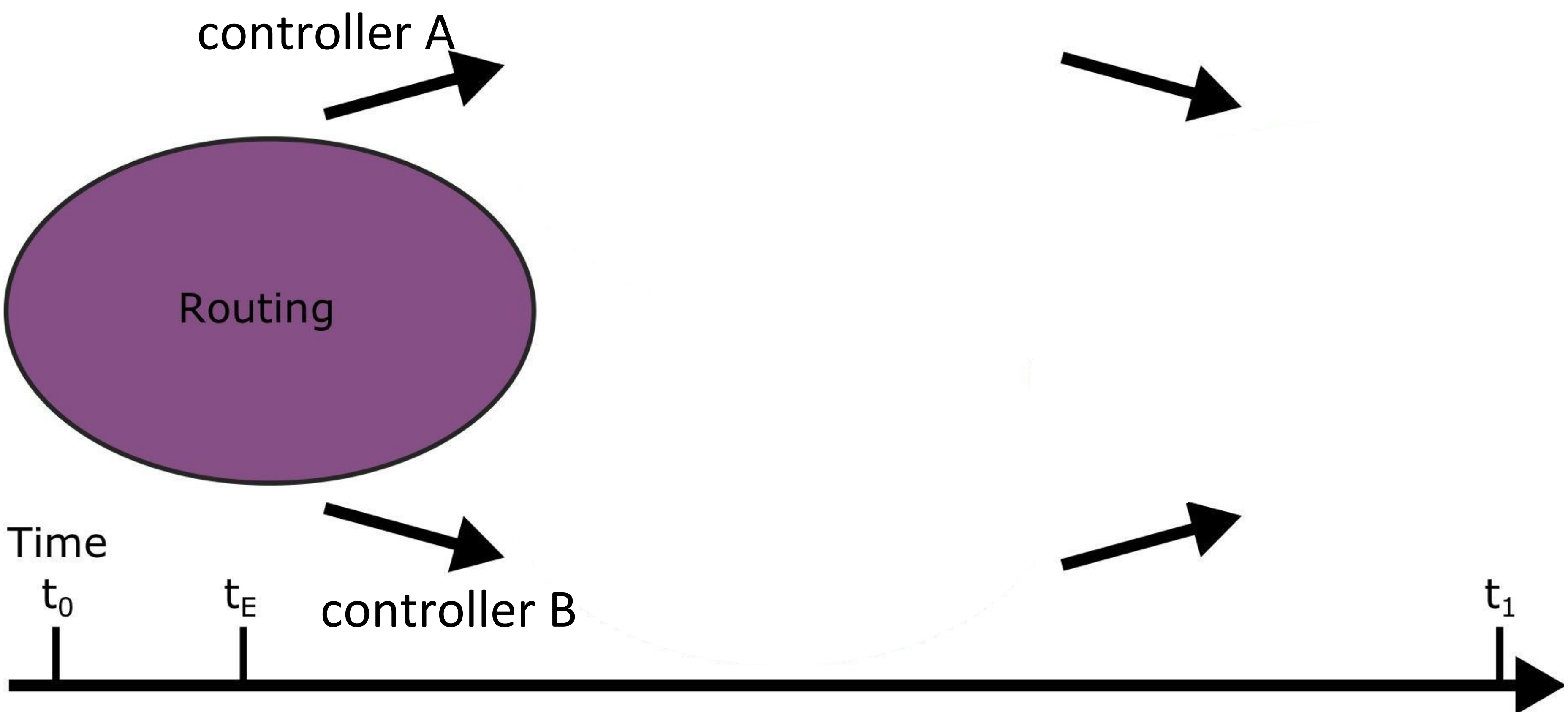


Distributed Control Models

3: Sharding by Policy

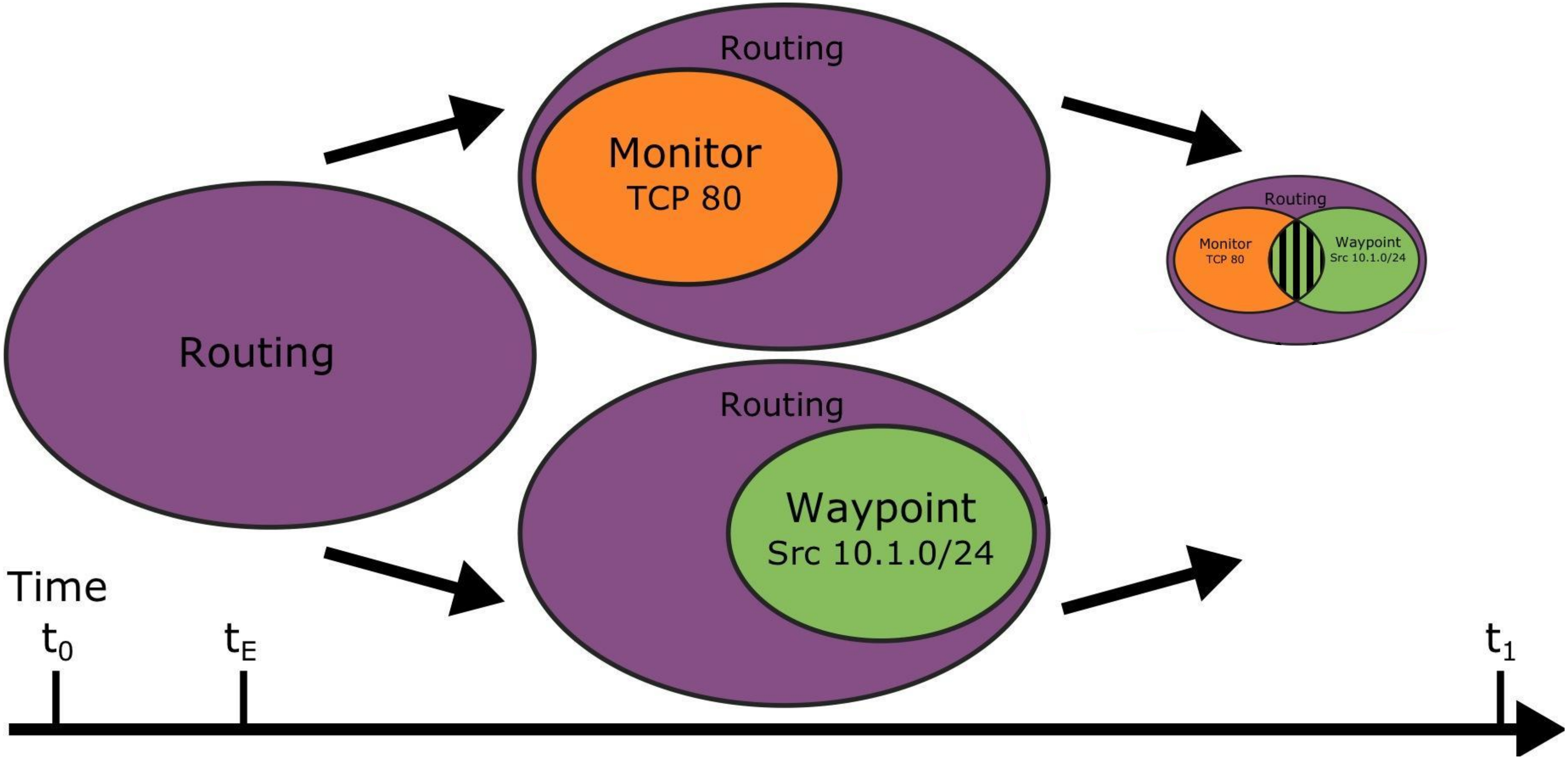


Concurrency Gone Wrong



Impossible to guarantee a deterministic outcome **without policy synchronization**

Consistent Policy Composition



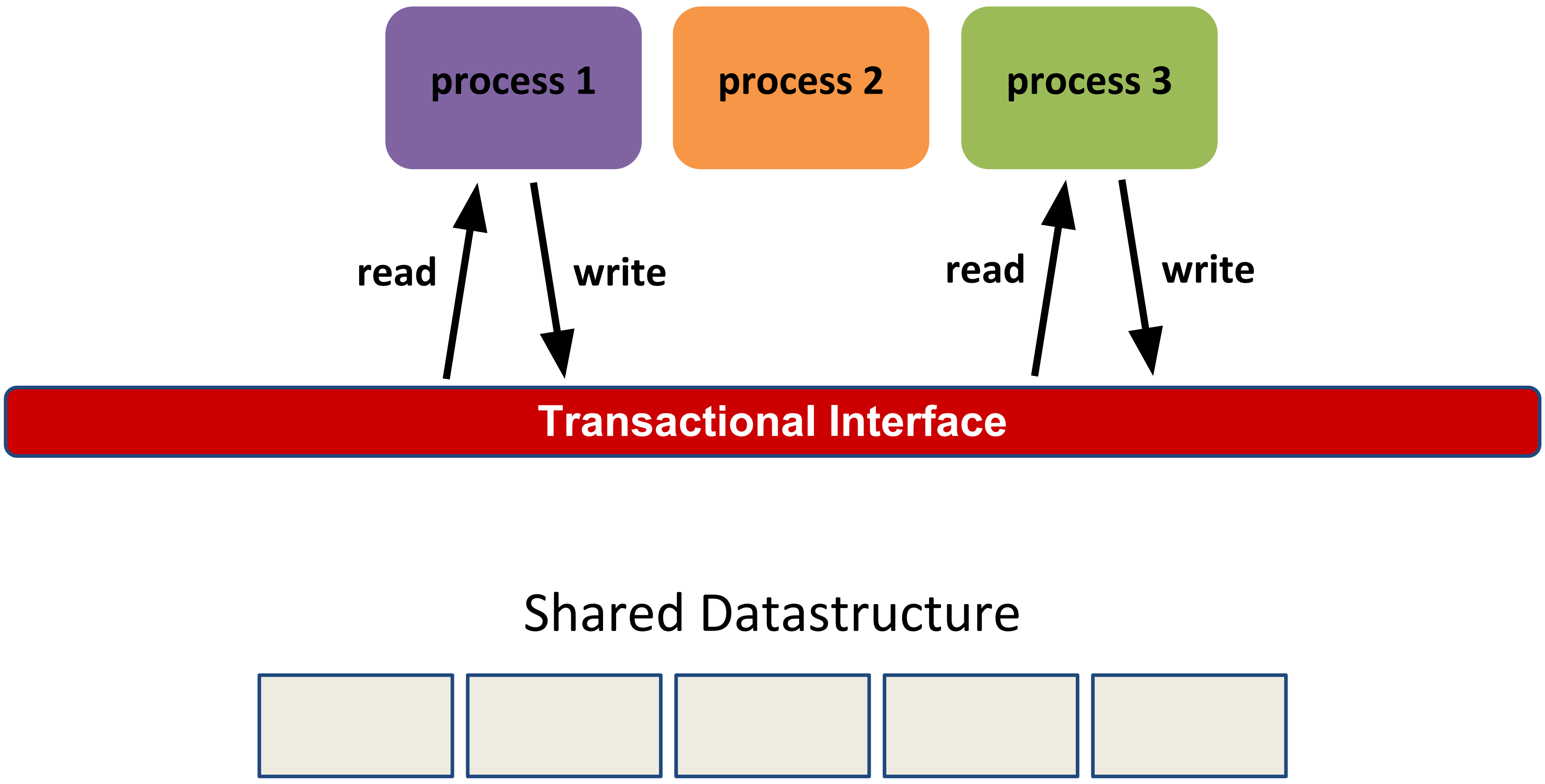
Deterministic Policy Composition which respects precedence of multi-authorship

Research Question #2

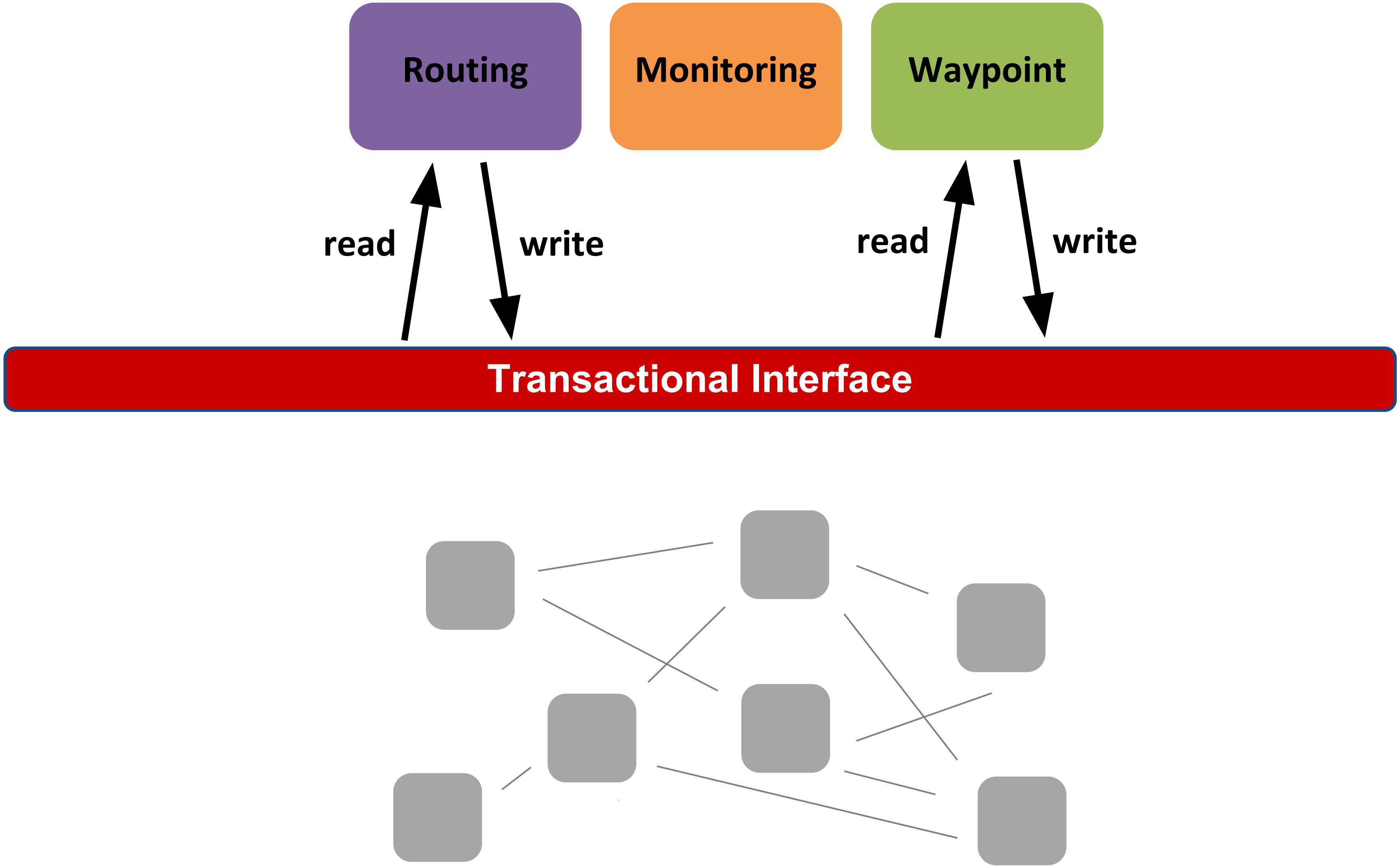
Can we realize a distributed policy composition interface that...

is agnostic to:	guarantees:
Control Distribution model	Consitent update semantics
Switch reader-writer model	Consitent policy composition

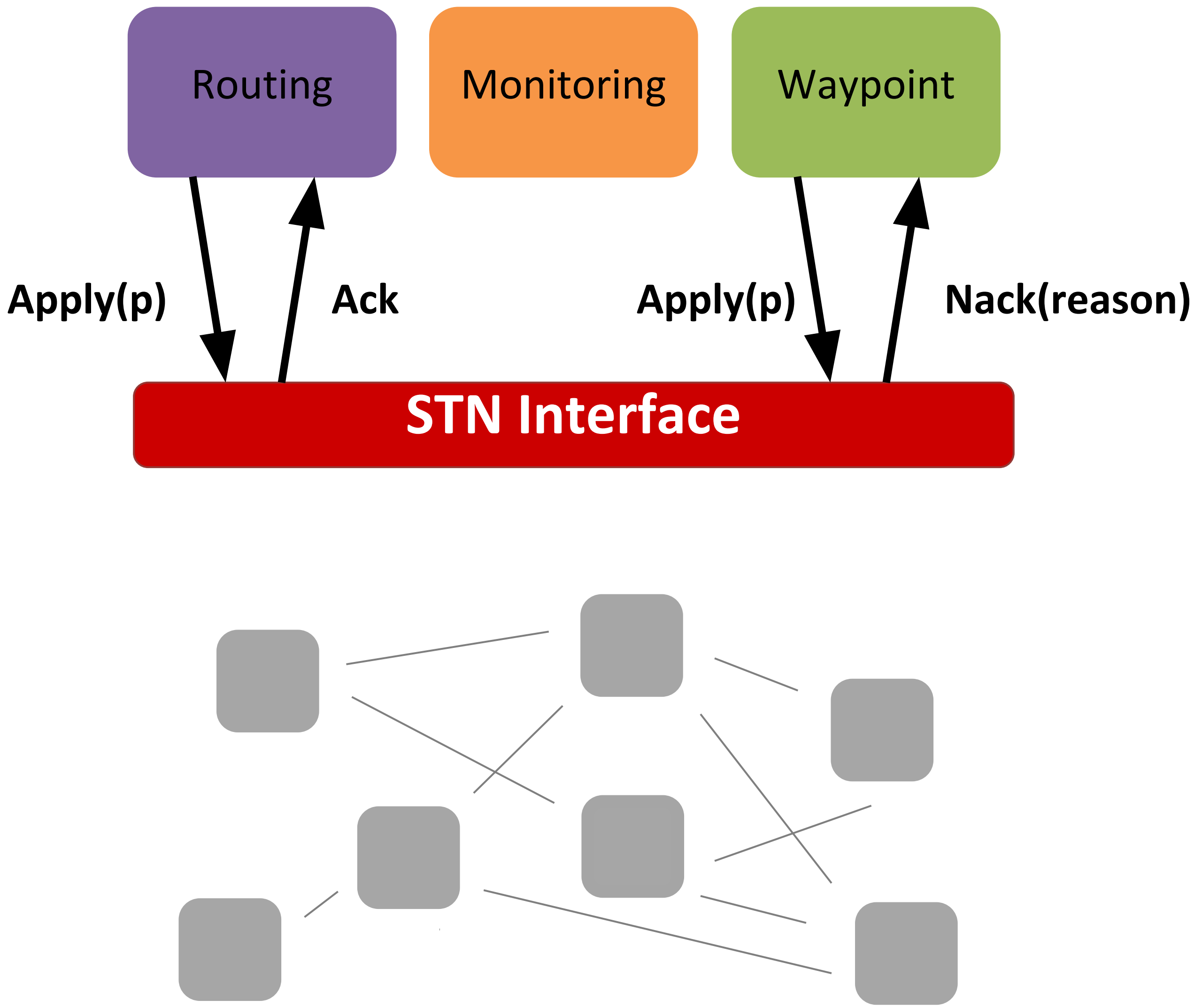
Software Transactional **Memory**



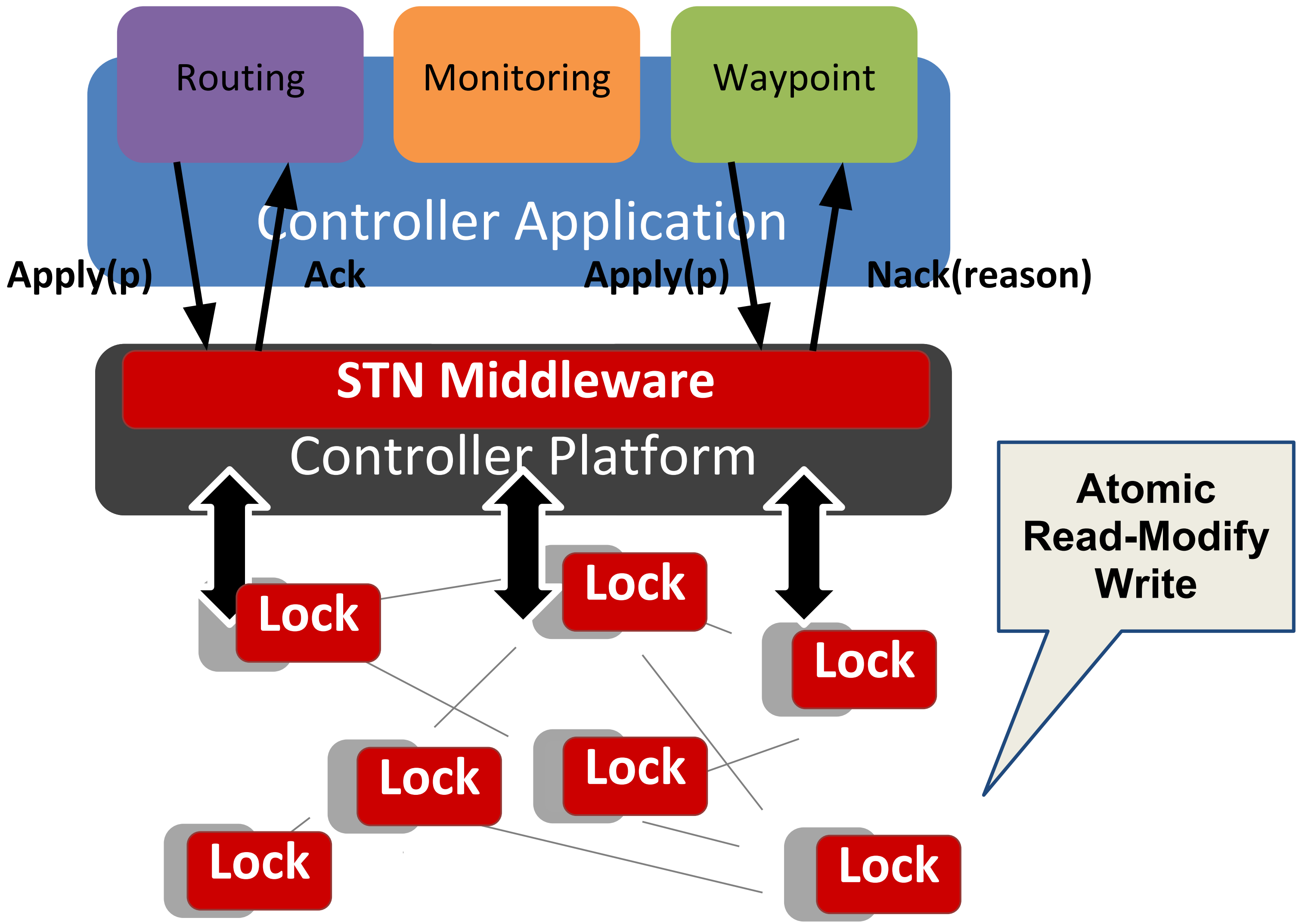
Software Transactional **Networking**



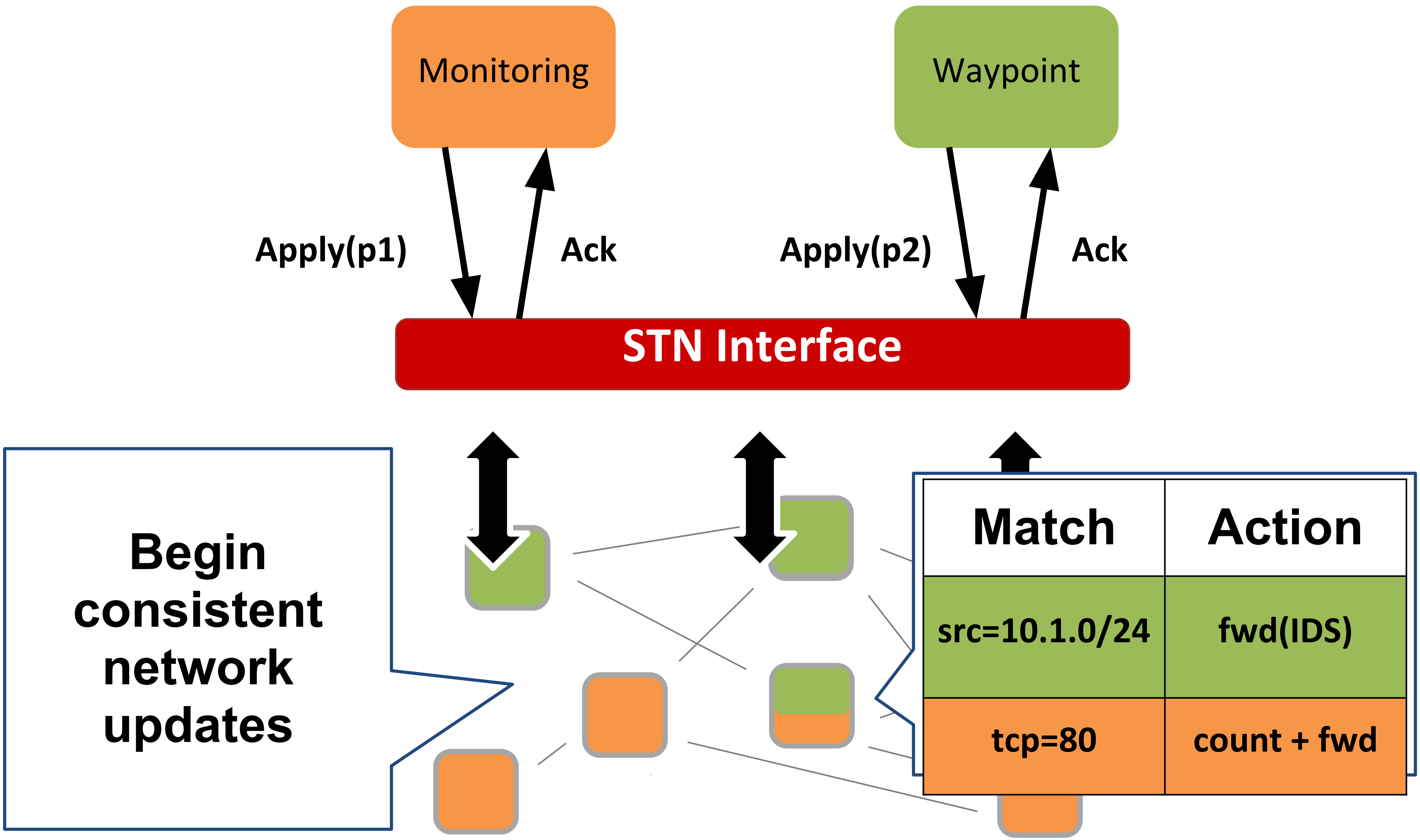
The STN Interface



Conceptualizing STN

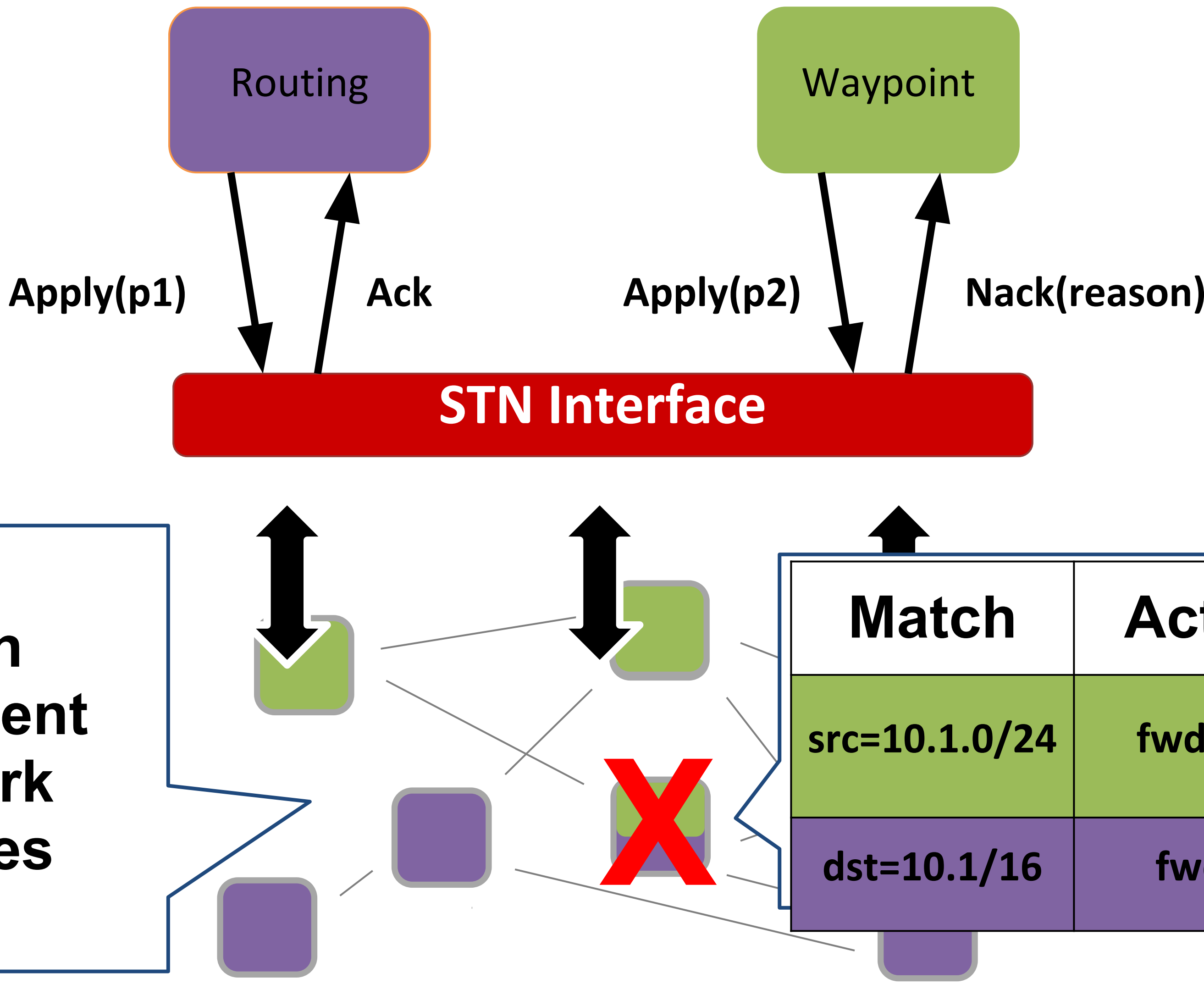


STN in Action (Ack Case)



Match	Action
src=10.1.0/24	fwd(IDS)
tcp=80	count + fwd

STN in Action (Nack Case)



Begin consistent network updates

Match	Action
src=10.1.0/24	fwd(IDS)
dst=10.1/16	fwd(2)

STN fine-grained locking algorithm

```
apply(policy p, policy tree f):
    tag = newTag(p)
    for s in switches():
        if s.rules.doCompose(p, f):
            tag' = concurrentTag(s, tag)
            s.addTagRules(p, tag')
        else remove tag' rules, nack(reason)
    for s in ingressSwitches():
        s.addRule(match=*, action=push(tag'))
```

Open Issues

- For our simple algorithm, tag and forwarding-rule state grows exponentially $O(2^n)$ for n-concurrency
- Different Update Consistency and Policy Composition Semantics
- Atomic read-modify-write primitive at switch
- Controller fault tolerance

Summary

Full technical report for more details
http://arxiv.org/abs/1305.7429

Internship Opportunities @T-Labs

- Interface for distributed policy composition
- Framework to reason about concurrent policy composition and consistent update

Backup Slides

Policy Composition Review

Priority	Match	Action
1	dst=10.1/16, *	fwd(1)
1	dst=10.2/16, *	fwd(2)
2	dst=10.1/16,dport=80, *	fwd(1)
2	dst=10.2/16,dport=80, *	fwd(2)
3	dst=10.1/16, src=10.1.0/24, dport=80, *	fwd(IDS)
...	...	...

Routing: dst=10.1/16 → fwd(1)
dst=10.2/16 → fwd(2)

Monitor: tcp_port=80 → count

Waypoint: src=10.1.0/24 → fwd(IDS)

Composition is the **"cross-product"** of rules

Distributed Control Models

2: Sharding by Region