A Distributed and Robust SDN Control Plane for Transactional Network Updates

Marco Canini (UCL)

with
Petr Kuznetsov (Télécom ParisTech),
Dan Levin (TU Berlin),
Stefan Schmid (TU Berlin & T-Labs)

28 Apr 2015
INFOCOM 2015
Network Policy Specification
Network Policy Specification

1. <Match, Action>
2. <Match, Action>
3. <Match, Action>
4. <Match, Action>
5. <Match, Action>
6. ....
7. ....

forward from X to Y
Network Policy Specification

Controller Platform

Routing  Load Balancing  Monitoring  Access Control  Waypoint

Composition

Controller Platform

Policy composition assembles data plane updates as a semantically sound set of rules
Policy Composition Review

**Foster ’11, Monsanto ’13:** Modular, parallel and sequential composition

- **Reitblatt ’12:** Consistent network updates

- **Ferguson ’13:** Policy trees for multi-authorship

**Composition**
In the general case, policies might conflict.

Examples:
- Overlapping domains and same precedence
- Scarce flowtable resources

Goal: Must avoid conflicting policies!
Centralized Network Control?

Fully centralized $\rightarrow$ Inadequate availability, scalability and responsiveness
Distributed Network Control

Centralized Network Policy

Controller Platform

Routing

Load Balancing

Monitoring

Access Control

Waypoint

Composition?

Consistency and concurrency

Faults and asynchronous communication

INFOCOM 2015
Now, consider policy composition in the distributed control plane...
Why should the programmer care?

• We believe the programmer should not!
• Enter Software Transactional Networking
  – Let a dedicated component implement a general solution to all hard-to-solve, low-level concurrency and fault tolerance issues to policy composition
Consistency: Linearizability of updates

We don’t control traffic!

Manipulate the network as though there is no concurrency
Software Transactional Networking: Consistent Policy Composition (CPC)

1. All-or-nothing semantics
2. Tolerate up to $f$ controller node crash failures
3. Non conflicting policies eventually installed and at least one policy commits (among conflicting ones)
4. Ensure policy updates affect traffic as a sequential composition of their policies
Conceptualizing CPC

Every controller node receives and participate in installing every policy update.
Conceptualizing CPC

1. Internal ports match on tag $\tau$

2. Ingress ports apply tag $\tau$

1. Internal ports match on tag $\tau$

Policy $\pi$ unique tag $\tau$
Conceptualizing CPC

Atomic read-modify-write primitive
RMW model

- Controllers access ports with atomic read-modify-write primitive $\text{RMW}(g,v)$:
  - read current state $v'$
  - apply and return $g(v,v')$

- **Intuition**: do not update if policy update conflicts with currently installed policy

- In the paper: **Theorem**: 1-resilient read-write CPC is impossible
FixTag: upper bound algorithm

Operation:

1. Unique tag per path
2. Broadcast policy $\pi$ to all other controllers
3. Update ingress ports in predefined order
4. ... add rule to tag all packets matching $\text{dom}(\pi)$ with the tag corresponding to the path $\pi^{(i)}$ for ingress port $i$

Upsides: wait-free (tolerates all failure patterns)
- Controllers only synchronize through the data plane

Downsides: tag complexity linear in # possible policies and paths
- May grow super-exponential in the size of the network
Can we lower the tax complexity?

• No, if we get no feedback from the network
  – Tag $\tau$ cannot be reused if a packet tagged with $\tau$ is still “in flight”

• Suppose, we can correctly evaluate the set of active tags
  – Correct (but asynchronous) oracle

• Single-controller scenario: one bit is enough!
  – Upon policy update $\pi_i$, wait until ($i \mod 2$)-traffic is over, and use tag $i \mod 2$

• Two or more controllers: inherent price of concurrency?
  – Between constant and super-exponential?

• Yes, if controllers coordinate the use of tags
ReuseTag: linear complexity

• Proportional to the level of resilience:
  – Up to $f$ failures: $f+2$ tags needed (proved optimal)

• Controllers use replicated state machine that imposes a total order on the policy updates and ensures coordinated use and reuse of tags
  – All requests are serialized, even non-conflicting ones
Summary

• Software Transactional Networking framework for consistent policy composition (CPC) in distributed SDN control planes
  – Transactional interface to manipulate the network as though there is no concurrency
  – Policies compose or conflict (and abort)
    • Formal model of the problem is in the paper

• Two CPC algorithms
  – FixTag
  – ReuseTag: \(f+2\) tags (minimal number)
Backup
Acknowledgements

Petr Kuznetsov  
Dan Levin  
Stefan Schmid

Supported by ARC grant 13/18-054 from Communauté française de Belgique, and European Union’s Horizon 2020 ENDEAVOUR project (grant agreement 644960)