Provable Data Plane Connectivity with Local Fast Failover

Introducing OpenFlow Graph Algorithms

Michael Borokhovich (Ben Gurion Uni, Israel)
Liron Schiff (Tel Aviv Uni, Israel)
Stefan Schmid (TU Berlin & T-Labs, Germany)
Robust Routing Mechanisms

• Link failures today are not uncommon [1]

• Modern networks provide robust routing mechanisms
  • i.e., routing which reacts to failures
  • example: MPLS local and global path protection
**Fast In-band Failover**

- **Important that failover happens fast = in-band**
  - Reaction time in control plane can be orders of magnitude slower [1]

- **For this reason: OpenFlow Local Fast Failover Mechanism**
  - Supports conditional forwarding rules (depend on the local state of the link: live or not?)

- **Gives fast but local and perhaps “suboptimal” forwarding sets**
  - Controller improves globally later…

---

[1] Reference or citation.
Important that failovers happen fast = in-band
- Reaction time in control plane can be orders of magnitude slower

For this reason: OpenFlow Local Fast Failover Mechanism
- Supports conditional forwarding rules (depend on the local state of the link: live or not?)
- Gives fast but local and perhaps “suboptimal” forwarding sets
  - Controller improves globally later…

However, not much is known about how to use the OpenFlow fast failover mechanism. E.g.: How many failures can be tolerated without losing connectivity?
• Important that failover happens fast = in-band
  • Reaction time in control plane can be orders of magnitude slower
• For this reason: OpenFlow Local Fast Failover Mechanism

  • Supports conditional forwarding rules (depend on the local state of the link: live or not?)
  • Gives fast but local and perhaps "suboptimal" forwarding sets
  • Controller improves globally later…

However, not much is known about how to use the OpenFlow fast failover mechanism. E.g.: How many failures can be tolerated without losing connectivity?

How to use mechanism is a non-trivial problem even if underlying network stays connected: (1) conditional failover rules need to be allocated ahead of time, without knowing actual failures, (2) views at runtime are inherently local. How not to shoot in your foot with local fast failover (e.g., create forwarding loops)?
Contribution: Very Robust Routing Possible with OpenFlow

Theorem: «Ideal» Forwarding Connectivity Possible

There exist algorithms which guarantee that packets always reach their destination, independently of the number and locations of failures, as long as the remaining network is connected.
Contribution: Very Robust Routing Possible with OpenFlow

Theorem: «Ideal» Forwarding Connectivity Possible

There exist algorithms which guarantee that packets always reach their destination, **independently of the number and locations** of failures, as long as the remaining network is connected.

Three algorithms:
- Modulo
- Depth-First
- Breadth-First

Essentially classic **graph algorithms** (routing, graph search) implemented **in OpenFlow**. Make use of **tagging** to equip packets with meta-information to avoid forwarding loops.
Contribution: Very Robust Routing Possible with OpenFlow

Theorem: «Ideal» Forwarding Connectivity Possible

There exist algorithms which guarantee that packets always reach their destination, independently of the number and locations of failures, as long as the remaining network is connected.

Three algorithms:
- Modulo
- Depth-First
- Breadth-First

Essentially classic graph algorithms (routing, graph search) implemented in OpenFlow. Make use of tagging to equip packets with meta-information to avoid forwarding loops.

Analysis of their complexity: maximum stretch (route length compared to ideal route), number of tags, number of OpenFlow rules.
Overview of Contributions

High-Level Algorithms

Complexity Analysis

Flow-Table Implementations

Related Work

- Borokhovich, OPODIS’13
- [1] Liu et al. NSDI’13
- Graph-search literature
We expect that our algorithms scale up to 500-node networks (ignoring link capacities) (e.g., using our NoviKit 250 switches, with 32MB flow table space and full support for extended match fields).
We expect that our algorithms scale up to 500-node networks (ignoring link capacities) (e.g., using our NoviKit 250 switches, with 32MB flow table space and full support for extended match fields).

Inherent tradeoffs between robustness and network load of failover without tagging.

Same objective: ideal connectivity. But their link-reversal algorithms not applicable to OpenFlow: require dynamic state at router.

Lower bounds with implications on optimality of our algorithms.

Overview of Contributions

Flow-Table Implementations

Complexity Analysis

Related Work

• Borokhovich, OPODIS’13
• [1] Liu et al. NSDI’13
• Graph-search literature
• Fast failover: example of a function that should be kept in the data plane

• Our result shows that non-trivial functions can be computed in the OpenFlow data plane!

• Our algorithms: may serve in compilers for higher-level languages, e.g., FatTire