On Consistent Updates in Software Defined Networks

Ratul Mahajan, Microsoft Research
Roger Wattenhofer, Microsoft Research & ETH Zurich
The Blind Spot of SDNs

Current State

Future State
Example

SDN Controller
Example

SDN Controller

```
u ---- x ---- d
 |      |      |
 v     y     |
```

```
 u ---- x
 |
 v ---- y ---- d
```
Example

SDN Controller
Example

SDN Controller
Example

[Reitblatt et al., SIGCOMM 2012]
Example

SDN Controller

[Reitblatt et al., SIGCOMM 2012]
Example

SDN Controller
Dependencies

Version Numbers
[Reitblatt et al.]

+ stronger packet coherence
– version number in packets
– switches need to store both versions

“Better” Solution
[This paper]
Minimum SDN Updates?
Minimum Updates: Another Example

\[ uvw \rightarrow u \]

\[ w \rightarrow v \]

or

\[ w \rightarrow u \]

or

\[ w \rightarrow v \]
Minimum vs. Minimal
Minimum vs. Minimal

No node can improve without hurting another node
In the paper, we present an algorithm to compute such a minimal dependency forest.
Main Contribution

For a given consistency property, what is the minimal dependency possible?
## Consistency Space

<table>
<thead>
<tr>
<th></th>
<th>None</th>
<th>Self</th>
<th>Downstream subset</th>
<th>Downstream all</th>
<th>Global</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Eventual consistency</strong></td>
<td>Always guaranteed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Drop freedom</strong></td>
<td>Impossible</td>
<td>Add before remove</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Memory limit</strong></td>
<td>Impossible</td>
<td>Remove before add</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Loop freedom</strong></td>
<td>Impossible (Lemma 6)</td>
<td>Rule dep. forest (§2.2)</td>
<td>Rule dep. tree (§2.1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Packet coherence</strong></td>
<td>Impossible (Lemma 7)</td>
<td></td>
<td>Per-flow ver. numbers</td>
<td></td>
<td>Global ver. numbers [8]</td>
</tr>
<tr>
<td><strong>Bandwidth limit</strong></td>
<td>Impossible (Lemma 8)</td>
<td></td>
<td></td>
<td>Staged partial moves [5]</td>
<td></td>
</tr>
</tbody>
</table>
It’s *not* just how to compute new rules.

It is also how to gracefully get from *current* to *new* configuration, respecting consistency.
Architecture

- Rule generator
- New rules
- Update plan generator
- Update DAG
- Plan optimizer and executor

Routing policy → Consistency property → Network characteristics
Update DAG

- Insert rule $r$ at node $u$
- Insert rule $t$ at node $w$
- Wait 10s
- Remove rule $q$ at node $x$
- Logical OR
- Remove rule $s$ at node $v$
- Insert rule $p$ at node $y$
Multiple Destinations using Prefix-Based Routing

- No new “default” rule can be introduced without causing loops
- Solution: Rule-Dependency Graphs!
- Deciding if simple update schedule exists: [Vanbever et al., TON 2012]
Breaking Cycles

Insert $u \rightarrow w$ → Remove $u \rightarrow v$ → Insert $v \rightarrow u$

Remove $w \rightarrow u$ ← Insert $w \rightarrow v$ ← Remove $v \rightarrow w$
Breaking Cycles

Insert $u \rightarrow w$

Remove $u \rightarrow v$

Insert $v \rightarrow u$

Remove $v \rightarrow w$

Insert at $w$: dest $v: w \rightarrow v$

Remove at $w$: dest $v: w \rightarrow v$

Insert $w \rightarrow v$

Remove $w \rightarrow u$

Insert $w \rightarrow v$

Remove $v \rightarrow u$

Remove $v \rightarrow w$
Architecture

Routing policy

Rule generator

New rules

Update plan generator

Update DAG

Plan optimizer and executor

Consistency property

Network characteristics
Breaking Cycles

Insert $u \rightarrow w$

Remove $u \rightarrow v$

Insert $v \rightarrow u$

Insert at $w$: dest $v: w \rightarrow v$

Remove at $w$: dest $v: w \rightarrow v$

Remove $w \rightarrow u$

Insert $w \rightarrow v$

Remove $v \rightarrow w$
Are Minimal Dependencies Good?
Are Minimal Dependencies Good?

…it depends
Are Minimal Dependencies Good?

...it depends

(But Plan optimizer and executor will fix it.)
Architecture

Routing policy

Rule generator

New rules

Update plan generator

Consistency property

Update DAG

Network characteristics

Plan optimizer and executor
Evaluation
## Summary

<table>
<thead>
<tr>
<th></th>
<th>None</th>
<th>Self</th>
<th>Downstream subset</th>
<th>Downstream all</th>
<th>Global</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eventual</td>
<td>Always guaranteed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>consistency</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Drop freedom</td>
<td>Impossible</td>
<td>Add before remove</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory limit</td>
<td>Impossible</td>
<td>Remove before add</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loop freedom</td>
<td>Impossible (Lemma 6)</td>
<td>Rule dep. forest</td>
<td>Rule dep. tree</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(§2.2)</td>
<td>(§2.1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Packet coherence</td>
<td>Impossible (Lemma 7)</td>
<td>Per-flow ver.</td>
<td>Global ver.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>numbers</td>
<td>numbers [8]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bandwidth limit</td>
<td>Impossible (Lemma 8)</td>
<td></td>
<td></td>
<td>Staged partial</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>moves [5]</td>
<td></td>
</tr>
</tbody>
</table>

### Diagram:

- **Routing policy**
  - Rule generator
  - New rules
  - Update plan generator
  - Update DAG
  - Plan optimizer and executor

- **Consistency property**
At ETH Zurich, we’re looking for a colleague in networking! Please ask me for details.
Thank You!
Questions & Comments?

www.disco.ethz.ch