On Consistent Updates in Software Defined Networks

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

SDN Controller
Example

SDN Controller
Example

SDN Controller
Example

SDN Controller

[Reitblatt et al., SIGCOMM 2012]
Example

SDN Controller

[Reitblatt et al., SIGCOMM 2012]
Example

SDN Controller
Dependencies

Version Numbers
[Reitblatt et al.]

“Better” Solution
[This paper]

+ stronger packet coherence
– version number in packets
– switches need to store both versions
Minimum SDN Updates?
Minimum Updates: Another Example

\[ \begin{align*}
  u & \rightarrow w & \rightarrow d & \rightarrow v \\
  v & \rightarrow w & \rightarrow d & \rightarrow u
\end{align*} \]

or

\[ \begin{align*}
  w & \rightarrow u \\
  w & \rightarrow v
\end{align*} \]
Minimum vs. Minimal
No node can improve without hurting another node

Minimum vs. Minimal
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</strong></td>
<td><strong>Drop</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>consistency</strong></td>
<td>consistency</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Impossible</strong></td>
<td><strong>Add before remove</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory limit</td>
<td><strong>Impossible</strong></td>
<td><strong>Remove before add</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loop freedom</td>
<td><strong>Impossible (Lemma 6)</strong></td>
<td>Rule dep. forest (§2.2)</td>
<td>Rule dep. tree (§2.1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Packet coherence</td>
<td><strong>Impossible (Lemma 7)</strong></td>
<td></td>
<td></td>
<td>Per-flow ver. numbers</td>
<td>Global ver. numbers [8]</td>
</tr>
<tr>
<td>Bandwidth limit</td>
<td><strong>Impossible (Lemma 8)</strong></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

Routing policy

Rule generator

New rules

Update plan generator

Consistency property

Update DAG

Network characteristics

Plan optimizer and executor
Update DAG

- Insert rule $r$ at node $u$
- Remove rule $s$ at node $v$
- Remove rule $q$ at node $x$
- Insert rule $p$ at node $y$
- Insert rule $t$ at node $w$

Logical OR

Wait 10s
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 $u \rightarrow w$ \quad Remove $u \rightarrow v$ \quad Insert $v \rightarrow u$

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

Remove $v \rightarrow w$
Architecture

Routing policy

Rule generator

New rules

Consistency property

Update plan generator

Update DAG

Network characteristics

Plan optimizer and executor
Breaking Cycles

Insert \( u \rightarrow w \)

Remove \( v \rightarrow u \)

Insert \( v \rightarrow w \)

Remove \( w \rightarrow u \)

Insert at \( w \):
dest \( v: w \rightarrow v \)

Remove at \( w \):
dest \( v: w \rightarrow v \)
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 → Update DAG → Plan optimizer and executor

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

- Consistency property

- Network characteristics
Evaluation
Evaluation
## Summary

<table>
<thead>
<tr>
<th>Eventual consistency</th>
<th>None</th>
<th>Self</th>
<th>Downstream subset</th>
<th>Downstream all</th>
<th>Global</th>
</tr>
</thead>
<tbody>
<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 (§2.2)</td>
<td>Rule dep. tree (§2.1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Packet coherence</td>
<td>Impossible (Lemma 7)</td>
<td>Per-flow ver. numbers</td>
<td>Global ver. numbers [8]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bandwidth limit</td>
<td>Impossible (Lemma 8)</td>
<td></td>
<td>Staged partial moves [5]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Diagram:

1. **Routing policy**
   - Rule generator
   - New rules

2. **Consistency property**
   - Update plan generator
   - Update DAG

3. **Network characteristics**
   - Plan optimizer and executor
At ETH Zurich, we’re looking for a colleague in networking! Please ask me for details.
Thank You!
Questions & Comments?

www.disco.ethz.ch