The Power of Two in Consistent Network Updates: Hard Loop Freedom, Easy Flow Migration

Klaus-Tycho Förster and Roger Wattenhofer
“some switches can ‘straggle,’ taking substantially more time than average (e.g., 10-100x) to apply an update”

Jin et al., SIGCOMM 2014
“Tree Ordering”
“Tree Ordering”
“Tree Ordering”
Software-Defined Networking

Centralized controller updates networks rules for optimization

Controller (control plane) updates the switches/routers (data plane)
old network rules

network updates

new network rules
old network rules

network updates

new network rules
old network rules

network updates

new network rules

possible solution: be fast!

e.g., B4 [Jain et al., 2013]
possible solution: be consistent!

e.g.,
- per-router ordering [Vanbever et al., 2012]
- two phase commit [Reitblatt et al., 2012]
- SWAN [Hong et al., 2013]
- Dionysus [Jin et al., 2014]
- ....
possible solution: be consistent!
Dynamic Updates

Idea: Update as many edges as you can
Dynamic Updates
Dynamic Updates

network updates

a — b — d

...
Dynamic Updates

network updates
Dynamic Updates

network updates

greedy maximum update
a & b update → all others wait
2 nodes update
Dynamic Updates

**greedy maximum update**
- a & b update → all others wait
- 2 nodes update

**maximal update**
- a waits → all others update
- all but 1 update
Find maximal update?

• Let’s go more general
Find maximal update?

• Let’s go more general
• Delete all cycles in a graph
Find maximal update?

• Let’s go more general
• Delete all cycles in a graph
Find maximal update?

- Let’s go more general
- Delete all cycles in a graph
Find maximal update?

- Let’s go more general
- Delete all cycles in a graph
- **NP-hard** to approximate
  - *Feedback Arc Set*
Find maximal update?

- Let’s go more general
- Delete all cycles in a graph
- **NP-hard** to approximate
  - *Feedback Arc Set*
- And it’s equivalent 😞
Find maximal update?

• Let’s go more general
• Delete all cycles in a graph
• **NP-hard** to approximate
  – *Feedback Arc Set*
• And it’s equivalent 😞
Dynamic Updates

Maximize \#edges updated \approx Feedback Arc Set

\Rightarrow Approximate within \( O(\log n \log \log \log n) \)

Better approximation bound for Feedback Arc?

\Rightarrow Implies better bound for \#edges
Scheduling Updates

But how long until all edges updated?
Scheduling Updates

But how long until all edges updated?

Ludwig et al. (2015): NP-hard for 3-schedule
Scheduling Updates

But how long until all edges updated?

Ludwig et al. (2015): NP-hard for 3-schedule

Our result (with 2 destinations)

NP-hard for any sublinear schedule
Scheduling Updates

Idea: Delay updates
Scheduling Updates

Idea: Delay updates
Scheduling Updates

Idea: Delay updates
Idea: Delay updates
Scheduling Updates

Iterate over and over and over....

\[ v \rightarrow w \]

\[ d, d' \]

\[ w' \rightarrow v \rightarrow w \]

\[ d, d' \]
Loop Freedom Overview

• Maximize updated #edges per update
  – NP-hard

• Sublinear schedule checking for 2 destinations
  – NP-hard
CONGESTION AHEAD

NEXT 20 YEARS
How to Move Flows?

\[
\begin{align*}
S_1 & \rightarrow v \rightarrow w \rightarrow T_2 \\
S_2 & \rightarrow v \rightarrow w \rightarrow T_2 \\
S_1 & \rightarrow T_1 \\
S_2 & \rightarrow T_1
\end{align*}
\]
How to Move Flows?
How to Move Flows?
How to Move Flows?
How to Move Flows?
How Hard in General w/o Splitting?

• Previous work: **Fastest** Migration is NP-hard
How Hard in General w/o Splitting?

• Previous work: Fastest Migration is NP-hard
• Our work: Deciding is NP-hard
How Hard in General w/o Splitting?

• Previous work: **Fastest** Migration is NP-hard
• Our work: **Deciding** is NP-hard
  – Reduction from Partition
Issues with Splitting
Issues with Splitting
Issues with Splitting
Packets bypass Waypoints!
2-Splittable Flows

- Keep both flow paths at the same time
- Easy updates: Change allocations @sources
High-Level Algorithm Idea

• Establish new paths at size 0
High-Level Algorithm Idea

- Establish new paths at size 0
- Only if >0 for all paths can be obtained:
  - Consistent migration possible
High-Level Algorithm Idea

• Establish new paths at size 0
• Only if >0 for all paths can be obtained:
  – Consistent migration possible
    • By changing allocations over multiple steps
High-Level Algorithm Idea

• Establish new paths at size 0
• Only if $>0$ for all paths can be obtained:
  – Consistent migration possible
    • By changing allocations over multiple steps
Summary
The Power of Two in Consistent Network Updates: Hard Loop Freedom, Easy Flow Migration

Klaus-Tycho Förster and Roger Wattenhofer
“Data plane updates may fall behind the control plane acknowledgments and may be even reordered.”
Kuzniar et al., PAM 2015

“...the inbound latency is quite variable with a [...] standard deviation of 31.34ms...”
He et al., SOSR 2015

“some switches can ‘straggle,’ taking substantially more time than average (e.g., 10-100x) to apply an update”
Jin et al., SIGCOMM 2014