On Scheduling Consistent Software-Defined Network Updates

Klaus-Tycho Foerster, Research Group Communication Technologies, Dagstuhl Seminar 1801, March 2018
Network Updates

• The Internet: Designed for selfish participants
  ◦ Often inefficient (low utilization of links), but robust

• But what happens if the WAN is controlled by a single entity?
  ◦ Examples: Microsoft & Amazon & Google ...
  ◦ They spend hundreds of millions of dollars per year
Network Updates

Think: Google, Amazon, Microsoft
Software-Defined Networking

• Possible solution:
  ◦ Software-Defined Networking (SDNs)

• General Idea: Separate data & control plane in a network

• Centralized controller updates networks rules for optimization
  ◦ Controller (control plane) updates the switches/routers (data plane)

• Centralized controller implemented with replication, e.g. Paxos
old network rules

network updates

new network rules
On Scheduling Consistent Software-Defined Network Updates, 2018-03-09

*old network rules*

*network updates*

*new network rules*
old network rules

network updates

new network rules

possible solution: be fast!

e.g., B4 (Google)
possible solution: be consistent!

e.g.,
- per-router ordering [Vanbever et al., 2012]
- two phase commit [Reitblatt et al., 2012]
- dependency graphs [Mahajan & Wattenhofer 2013, Jin et al., 2014]
- ....
old network rules

network updates

new network rules
Toy Example
Toy Example
Toy Example

![Diagram](image-url)
Toy Example

On Scheduling Consistent Software-Defined Network Updates, 2018-03-09
Toy Example
Appears in Practice

“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
Ordering Solution: Go backwards through the new routing tree
Ordering Solution: Go backwards through the new routing tree
Ordering Solution: Go backwards through the new routing tree
Ordering Solution: Go backwards through the new routing tree
Ordering Solution: Go backwards through the new routing tree
Ordering Solution: Go backwards through the new routing tree

- Always works for single-destination rules
  - Also for multi-destination with sufficient memory ("split")
- Schedule length: tree depth (up to $\Omega(n)$)
  - Optimal scheduling algorithms?
Greedy? Update as many as possible per round

- Always works 😊

- Maximizing is NP-hard 😞

- Single greedy update: $O(1)$ rounds $\Rightarrow$ $\Omega(n)$ rounds 😞 😞

- In general: Does a 3-round schedule exist? NP-hard 😞 😞 😞
Scheduling Loop-free Network Updates: It's Good to Relax! [Ludwig et al., 2015]

Two key ideas:
1. destination-based source-destination pairs \(<s, d>\)
2. no forwarding loops no loops between \(<s, d>\)
Scheduling Loop-free Network Updates: It's Good to Relax!

- Non-relaxed? $\Omega(n)$ rounds

- Relaxed?

\[ s \rightarrow \cdots \rightarrow d \]
Scheduling Loop-free Network Updates: It's Good to Relax!

- Non-relaxed? $\Omega(n)$ rounds

- Relaxed?

Round 1

Diagram showing the scheduling process.
Scheduling Loop-free Network Updates: It's Good to Relax!

- Non-relaxed? \( \Omega(n) \) rounds

- Relaxed?

Round 1
Scheduling Loop-free Network Updates: It's Good to Relax!

- Non-relaxed? $\Omega(n)$ rounds

- Relaxed?
Scheduling Loop-free Network Updates: It's Good to Relax!

- Non-relaxed? $\Omega(n)$ rounds

- Relaxed?
Scheduling Loop-free Network Updates: It's Good to Relax!

- Non-relaxed? $\Omega(n)$ rounds

- Relaxed? Just 3 rounds
Scheduling Loop-free Network Updates: It's Good to Relax!

- Non-relaxed? $\Omega(n)$ rounds

- Relaxed? Just 3 rounds
  - In general: $O(\log n)$ rounds (“Peacock”)

(a) The graph $G_1$ with 16 nodes. When Peacock selects the edge from 0/8 to 4/8 as a shortcut, pruning results in the graph in Fig. 10b.

(b) After two rounds with Peacock, isomorphic to $G_0$ in Fig. 10c.

(c) The graph $G_0$ with 8 nodes. 0/8 to 4/8 is the next shortcut.

(d) To the left, the output of Peacock on $G_0$ after two rounds. To the right, after two more rounds, selecting the first forward edge as a shortcut each time.

(e) The resulting updated graph, expanded into 16 nodes again.
Scheduling Loop-free Network Updates: It's Good to Relax!

- Non-relaxed? $\Omega(n)$ rounds

- Relaxed? Just 3 rounds
  - In general: $O(\log n)$ rounds ("Peacock")
  - But: Peacock instances with $\Omega(\log n)$ rounds
Open Questions for scheduling loop free updates:

For both models: Approximation algorithms for #rounds?

Relaxed:

• Optimal #rounds: NP-hard or in P?
• Do $O(1)$ rounds always suffice?
  ° (if yes: at least 7)

Non-relaxed:

• NP-hard for $O(1) < k < \Omega(n)$ rounds?
Open Questions for scheduling loop free updates:

For both models: Approximation algorithms for #rounds?

Relaxed:

• Optimal #rounds: NP-hard or in P?
• Do $O(1)$ rounds always suffice?
  ◦ (if yes: at least 7)

Non-relaxed:

• NP-hard for $O(1) < k < \Omega(n)$ rounds?

More open questions and specifics:

Survey of Consistent Software-Defined Network Updates
Klaus-Tycho Foerster, Stefan Schmid, Stefano Vissicchio
arXiv 1609.02305v2, 2018
Different model: “Flows“

- Identified by a “tag“ in the packet header, update via
  - Install new tag rules
  - Switch from tag to tag at source

- Goal: Respect link capacity constraints (“stronger“)

- Without “tags“:
  - Congestion-Free Rerouting of Flows on DAGs
    Saeed Akhoondian Amiri, Szymon Dudycz, Stefan Schmid, Sebastian Wiederrecht
    arxiv 1611.09296(v3), 2016/7
A Small Sample Network
Green wants to send as well
Congestion!
This would work
So let's go back
But Red is a bit Slow..
Congestion Again!
So let's go Back ...
First, Red switches
Then, Blue ...

Round 2
And then, Green ...
How hard is this (feasibility)?

Flows may only take *old* or *new* paths:

- NP-hard via reduction from Partition

Intermediate flow allocations not restricted to *old* and *new*:

- NP-hard already for just 2 unit size flows*°

Not clear if the problem is in NP! (It is known to be in EXPTIME°)

---

*: “no tags“: *Congestion-Free Rerouting of Flows on DAGs*. (Akhoondian Amiri et al., arxiv 1611.09296(v3), 2016/7)
Splittable flows? In P!

Fundamental idea:

• Try to free up some capacity on all edges at full capacity
• Can be done using augmenting flows in the residual network

Open Problems for scheduling flow migration

• What happens when we can pick the new paths?
  ◦ Only studied so far for a single destination [Brandt et al., PMC 2017]

• Unsplittable flow migration:
  ◦ In general: NP-, PSPACE-, or EXPTIME-complete?
    - (flows might need to switch back and forth repeatedly [Foerster, NCA 2017])
  ◦ Just old/new paths: Complexity for unit size flows?
  ◦ ”Interesting“ polynomial cases?
Open Problems for scheduling flow migration

• What happens when we can pick the new paths?
  ◦ Only studied so far for a single destination [Brandt et al., PMC 2017]

• Unsplittable flow migration:
  ◦ In general: NP-, PSPACE-, or EXPTIME-complete?
  ◦ Just old/new paths: Complexity for unit size flows?
  ◦ “Interesting” polynomial cases?

More open questions and specifics:
Survey of Consistent Software-Defined Network Updates
Klaus-Tycho Foerster, Stefan Schmid, Stefano Vissicchio
arXiv 1609.02305v2, 2018
Decentralized Updates for „Tree-Ordering“

• So far: every round:
  ◦ Controller computes and sends out updates
  ◦ Switches implement them and send acks
  ◦ Controller receives acks
Decentralized Updates for „Tree-Ordering“

• So far: every round:
  ◦ Controller computes and sends out updates
  ◦ Switches implement them
  ◦ Controller receives acks

• Alternative: Use dualism to so-called *proof labeling schemes*
Decentralized Updates for Tree Ordering

When should I update?
Decentralized Updates for a Tree Ordering

Once my parent updates!
Decentralized Updates for Tree Ordering

Once my parent updates!

Send parent ID
Decentralized Updates for „Tree-Ordering“
Decentralized Updates for Tree Ordering

I’ll update too!

I updated
Decentralized Updates for „Tree-Ordering“

+ Only one controller-switch interaction per route change
+ New route changes can be pushed before old ones done (*include “version#”*)
+ Incorrect updates can be locally detected (*include depth in tree*)

- Requires switch-to-switch communication   e.g., [Nguyen et al., SOSR 2017]

On Scheduling Consistent Software-Defined Network Updates

Klaus-Tycho Foerster, Research Group Communication Technologies, Dagstuhl Seminar 1801, March 2018