Transiently Secure Network Updates

Arne Ludwig¹, Szymon Dudycz², Matthias Rost¹, Stefan Schmid³
TU Berlin¹, University of Wroclaw², Aalborg University³
Network Updates

How to transition from old to new path?

While not discarding any packets!
Network Updates Happen

Error prone task
manual updates per device, despite global goals

Misconfiguration on switches that caused a “bridge loop”. [2012]

A network change was [...] executed incorrectly [...] re-mirroring storm [2011]
Model

Software-Defined Networking (SDN)
- Separate control from data plane
- Logically centralized network view (controller)
Software-Defined Networking (SDN)

- Separate control from data plane
- Logically centralized network view (controller)
- Not only destination based (match-action rules)
Model

Controller

Old policy

Network update

New policy

Controller

SDN enabled switch

A

SDN enabled switch

B

SDN enabled switch

A

SDN enabled switch

B
Model

Controller

Old policy

Network update

New policy

Controller

A
SDN enabled switch

B

A
SDN enabled switch

B
Strong Consistency

Two-phase commit [REI12] → Either old or new policy
Strong Consistency

Two-phase commit [REI12] → Either old or new policy

Tagging packets at ingress port
Strong Consistency

Two-phase commit [REI12] → Either old or new policy

Cons:
- Needs more switch memory
- Problematic with middleboxes (changed headers)
The Challenge: Transiently Secure Updates

- Consider dynamic updates without tagging [Mahajan et al., HotNets '13]

- Consistent forwarding state needs to be secured:
  - Ensure reachability by forbidding loops
  - Ensure traversal of waypoints, e.g. firewalls
Asynchronous Updates: Timing matters
Asynchronous Updates: Timing matters
Asynchronous Updates: Timing matters
Asynchronous Updates: Timing matters
Asynchronous Updates: Timing matters
Asynchronous Updates: Timing matters

We have to be selective which switches to update
Asynchronous Updates: Round model

Round:
Set of parallel updates.
Asynchronous Updates: Round model

Round:
Set of parallel updates.

Controller

ACK
Asynchronous Updates: Round model

Round:
Set of parallel updates.
Asynchronous Updates: Round model

Round:
Set of parallel updates.
Model Representation

Solid lines = current path
Dashed lines = new path
(Flow-specific path)
Model Representation

Solid lines = current path
Dashed lines = new path
(Flow-specific path)
Model Representation

- Solid lines = current path
- Dashed lines = new path
  (Flow-specific path)
Model Representation

Solid lines = current path
Dashed lines = new path
(Flow-specific path)

Safe to be updated
Safe to be left untouched
Model Representation

Solid lines = current path

Dashed lines = new path
Consistency Properties
Property: Strong Loop Freedom (SLF)
Property: Strong Loop Freedom (SLF)
Property: Strong Loop Freedom (SLF)

State

Temporary Forwarding Graph

\[ s \rightarrow v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow d \]

\[ s \rightarrow v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow d \]
Property: Strong Loop Freedom (SLF)

State

Temporary Forwarding Graph
Property: Strong Loop Freedom (SLF)

State

Temporary Forwarding Graph

Temporary forwarding graph – i.e. the union of previously and newly enabled edges – does not contain any directed loop.
Property: Strong Loop Freedom (SLF)

State

Temporary forwarding graph
– i.e. the union of previously and newly enabled edges
– does not contain any directed loop.
Property: Strong Loop Freedom (SLF)

State

Temporary Forwarding Graph
Property: Strong Loop Freedom (SLF)

State

Temporary Forwarding Graph
Property: **Relaxed Loop Freedom (RLF)**

**State**

```
 s  v_1  v_2  v_3  d
```

**Temporary Forwarding Graph**

```
 s  v_1  v_2  v_3  d
```

```
Property: **Relaxed Loop Freedom (RLF)**

Connected component of the temporary forwarding graph containing the source does not contain directed loops.
Property: **Relaxed Loop Freedom (RLF)**

Connected component of the temporary forwarding graph containing the source does not contain directed loops.

Finitely many packets

---

**State**

```
 s --|  v1 |  v2 |  v3 |  d
```

**Temporary Forwarding Graph**

```
 s --|  v1 |  v2 |  v3 |  d
```
Property: **Relaxed Loop Freedom (RLF)**

State

Temporary Forwarding Graph
**Property: Relaxed Loop Freedom (RLF)**

State

Temporary Forwarding Graph

Observation: RLF requires one round less than SLF.
Property: Waypoint Enforcement (WPE)

Increasing number of middleboxes [Sherry et al., SIGCOMM '12]
Property: Waypoint Enforcement (WPE)

'Waypoint (e.g. firewall) may never be bypassed.'

Solid lines = current path
Dashed lines = new path

\[ s \xrightarrow{} v_1 \xrightarrow{} v_2 \xrightarrow{} d \]
Property: Waypoint Enforcement (WPE)

There may not exist a path bypassing the waypoint in the Temporary Forwarding Graph.
Property: Waypoint Enforcement (WPE)
Overview

**Task:** Minimize overall update time, while
- ensuring Loop Freedom (LF)
- ensuring Waypoint Enforcement (WPE)

**Theory**
- LF + WPE may conflict
- Deciding LF + WPE is NP-hard
- other 'negative' results

**Practice**
- Mixed-Integer Programming Formulations
- Qualitative and Quantitative Analysis
Theory:
LF and WPE may conflict
LF and WPE may Conflict
LF and WPE may Conflict

Violates WPE
LF and WPE may Conflict

Violates WPE
LF and WPE may Conflict

Violates WPE

Violates LF
LF and WPE may Conflict

\[
\begin{align*}
&\text{Violates WPE} \quad \text{Violates LF}
\end{align*}
\]
LF and WPE may Conflict

Violates WPE
Violates LF
LF and WPE may Conflict

Some update problems are unsolvable when considering LF and WPE.
LF and WPE may Conflict

Some update problems are unsolvable when considering LF and WPE. Independent of whether RLF or SLF is considered.
LF and WPE may Conflict

Some update problems are unsolvable when considering LF and WPE.
Can we determine these cases easily?
Theory:
Deciding whether an Update Schedule exists is NP-hard
Deciding existence of Schedule is NP-hard

- Proof by 3-SAT reduction
  - Given a 3-SAT formula we construct a network update instance and show that there exists an update schedule iff. the formula is satisfiable.
Deciding existence of Schedule is NP-hard

- Proof by 3-SAT reduction
  - Given a 3-SAT formula we construct a network update instance and show that there exists an update schedule iff. the formula is satisfiable.
  - 3-SAT Clause $\mathcal{K}_1 \land \mathcal{K}_2 \land \ldots \land \mathcal{K}_m$ over Variables $x_1, x_2, \ldots, x_k$
  - Here: we only sketch the idea.
Construction of 3-SAT Reduction: Outline

\[ \mathcal{K}_1 \xrightarrow{r_{k+1}^1} \mathcal{K}_2 \xrightarrow{r_k^1} \cdots \xrightarrow{r_1^1} \mathcal{K}_m \xrightarrow{r_{k+1}^m} \mathcal{K}_m \xrightarrow{r_k^m} \mathcal{K}_m \]

\[ \text{wp} \xrightarrow{x_1} \mathcal{K}_1 \xrightarrow{x_2} \cdots \xrightarrow{x_k} \mathcal{K}_m \xrightarrow{\delta} \mathcal{K}_m \]

\[ \text{wp} \xrightarrow{u_1} \mathcal{K}_1 \xrightarrow{u_2} \mathcal{K}_2 \xrightarrow{u_3} \mathcal{K}_3 \xrightarrow{wp} \]
Construction of 3-SAT Reduction: Variable Gadgets
Construction of 3-SAT Reduction: Variable Gadgets

One node for each positive occurrence of variable $x_j$

One node for each negative occurrence of variable $x_j$
Construction of 3-SAT Reduction: Clause Gadgets

part before waypoint

wp

part after wp
Construction of 3-SAT Reduction: Clause Gadgets

part before waypoint

part after wp
Construction of 3-SAT Reduction: Clause Gadgets

part before waypoint

part after wp
Construction of 3-SAT Reduction: Clause Gadgets

Part before waypoint

Part after wp
Construction of 3-SAT Reduction: Clause Gadgets

Updating these switches violates WPE
Construction of 3-SAT Reduction: Clause Gadgets

Updating these switches violates LF
Construction of 3-SAT Reduction: Clause Gadgets

Clause gadget is tangled, as long as neither of these nodes is updated.
Construction of 3-SAT Reduction: Connection Clause with Variable Gadgets

Clause Gadget  Next Clause Gadget  Variable Gadget
Construction of 3-SAT Reduction: Connection Clause with Variable Gadgets

To untangle clauses, a consistent assignment of truth values to variables must be found.
Construction of 3-SAT Reduction: Untangling Clauses

- Trigger updates in variable gadgets depending on the truth value of the variable.
- Enable now bypassed backward rules from within variable gadgets.
- For each clause, select (arbitrarily) one of the valid assignments.
Construction of 3-SAT Reduction: Untangling Clauses

1) Trigger updates in variable gadgets depending on truth value of the variable
Construction of 3-SAT Reduction: Untangling Clauses

1) Trigger updates in variable gadgets depending on truth value of the variable

True
Construction of 3-SAT Reduction: Untangling Clauses

1) Trigger updates in variable gadgets depending on truth value of the variable
Construction of 3-SAT Reduction: Untangling Clauses

1) Trigger updates in variable gadgets depending on truth value of the variable
Construction of 3-SAT Reduction: Untangling Clauses

1) Trigger updates in variable gadgets depending on truth value of the variable

2) Enable now bypassed backward rules from within variable gadgets
Construction of 3-SAT Reduction: Untangling Clauses

1) Trigger updates in variable gadgets depending on truth value of the variable

2) Enable now bypassed backward rules from within variable gadgets
Construction of 3-SAT Reduction: Untangling Clauses

1) Trigger updates in variable gadgets depending on truth value of the variable

2) Enable now bypassed backward rules from within variable gadgets
Construction of 3-SAT Reduction: Untangling Clauses

1) Trigger updates in variable gadgets depending on truth value of the variable

2) Enable now bypassed backward rules from within variable gadgets
Construction of 3-SAT Reduction: Untangling Clauses

1) Trigger updates in variable gadgets depending on truth value of the variable

2) Enable now bypassed backward rules from within variable gadgets

3) For each clause select (arbitrarily) one of the valid assignments
Construction of 3-SAT Reduction: Untangling Clauses

1) Trigger updates in variable gadgets depending on truth value of the variable

2) Enable now bypassed backward rules from within variable gadgets

3) For each clause select (arbitrarily) one of the valid assignments. This untangles all clauses.

4) (start updating remaining nodes)
Main Result

\[(x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor x_4 \lor x_3) \land (\neg x_4 \lor x_2 \lor \neg x_3) \land (\neg x_1 \lor x_5 \lor x_6) \land (x_2 \lor \neg x_5 \lor \neg x_6)\]

3-SAT formula is satisfiable

iff.

constructed network update instance is updateable
Main Result

$$\left( x_1 \lor x_2 \lor x_3 \right) \land \left( \neg x_1 \lor x_4 \lor x_3 \right) \land \left( \neg x_4 \lor x_2 \lor \neg x_3 \right) \land \left( \neg x_1 \lor x_5 \lor x_6 \right) \land \left( x_2 \lor \neg x_5 \lor \neg x_6 \right)$$

3-SAT formula is satisfiable

iff.

constructed network update instance is updateable

Independent of whether RLF or SLF is considered.
Practice: Computing Update Schedules
Computing Update Schedules

- Finding a solution is NP-hard
- We employ Mixed-Integer Programming to compute solutions
  - evaluate computational hardness
  - quantitatively analyze feasibility
Computing Update Schedules

- LF and WPE are checked using Temporary Forwarding Graph
- Given decisions which switches to update, the state and the Temporary Forwarding Graph follow
Assign update of switch $v$ to a single round $r$:

$$x^r_v \in \{0, 1\}$$
Computing Update Schedules

Assign update of switch $v$ to a single round $r$:

$$x^r_v \in \{0,1\}$$

Represent forwarding state after round $r$ by

$$y^r_{u,v} \in [0,1]$$
Computing Update Schedules

Assign update of switch $v$ to a single round $r$:

$$x^r_v \in \{0, 1\}$$

Represent forwarding state after round $r$ by

$$y^r_{u,v} \in [0,1]$$

Represent Temporary Forwarding Graph by

$$y^{r-1 \lor r}_{u,v} \in [0,1]$$
Computing Update Schedules

Assign update of switch $v$ to a single round $r$:

$$x^r_v \in \{0, 1\}$$

Represent forwarding state after round $r$ by

$$y^r_{u,v} \in [0, 1]$$

Represent Temporary Forwarding Graph by

$$y^{r-1\lor r}_{u,v} \in [0, 1]$$

\[
1 = \sum_{r \in R} x^r_v \\
y^r_{u,v} = 1 - \sum_{r' \leq r} x^r_{u} \\
y^r_{u,v} = \sum_{r' \leq r} x^r_{u}' \\
y^{r-1\lor r}_{u,v} \geq y^{r-1}_{u,v} \\
y^{r-1\lor r}_{u,v} \geq y^r_{u,v} \\
y^{r-1\lor r}_{u,v} \leq \frac{l^r_v - l^r_u - 1}{|V| - 1} + 1 \\
\overline{a}^r_{s,w} = 1 \\
\overline{a}^r_{v,w} \geq \overline{a}^r_{u,w} + y^{r-1}_{u,v} - 1 \\
\overline{a}^r_{v,w} \geq \overline{a}^r_{u,w} + y^r_{u,v} - 1 \\
\overline{a}^r_d = 0
\]
Computing Update Schedules

Enforce SLF by employing Miller-Tucker-Zemlin Constraints by level variables:

\[ l_v^r \in [0, |V|-1] \]

\[
\begin{align*}
1 &= \sum_{r \in \mathcal{R}} x_v^r \\
y_u^r &= 1 - \sum_{r' \leq r} x_u^{r'} \\
y_{u,v} &= \sum_{r' \leq r} x_{u}^{r'} \\
y_{u,v}^r - 1 &\geq \sum_{r' \leq r} y_{u,v}^{r'} \\
y_{u,v}^r - 1 &\geq y_{u,v}^r \\
y_{u,v}^r &\leq \frac{l_v^r - l_u^r - 1}{|V| - 1} + 1
\end{align*}
\]

\[
\begin{align*}
\overline{a}_s^{r,w} &= 1 \\
\overline{a}_u^{r,w} &\geq \overline{a}_u^{r,w} + y_{u,v}^{r-1} - 1 \\
\overline{a}_v^{r,w} &\geq \overline{a}_u^{r,w} + y_{u,v}^r - 1 \\
\overline{a}_d^{r,w} &= 0
\end{align*}
\]
Computing Update Schedules

Enforce SLF by employing Miller-Tucker-Zemlin Constraints by level variables:

\[ l^r_v \in [0, |V| - 1] \]

\[ 1 = \sum_{r \in \mathcal{R}} x^r_v \]
\[ y^r_{u,v} = 1 - \sum_{r' \leq r} x^r_u \]
\[ y^r_{u,v} = \sum_{r' \leq r} x^{r'}_u \]
\[ y^r_{u,v} \geq y^r_{u,v} \]
\[ \sum_{r'} y^r_{u,v} \geq \sum_{r'} y^{r'}_{u,v} \]
\[ y^r_{u,v} \leq \frac{l^r_v - l^r_{u,v} - 1}{|V| - 1} + 1 \]

State

\[ \text{Temporary Forwarding Graph} \]

\[ \overline{a}^r_{s,w} = 1 \]
\[ \overline{a}^r_{v,w} \geq \overline{a}^r_{u,w} + y^r_{u,v} - 1 \]
\[ \overline{a}^r_{v,w} \geq \overline{a}^r_{v,w} + y^r_{u,v} - 1 \]
\[ \overline{a}^r_{d,w} = 0 \]
Computing Update Schedules

Enforce SLF by employing Miller-Tucker-Zemlin Constraints by level variables:

\[ l_v^r \in [0, |V| - 1] \]

Guarantee WPE by reachability constraints:

Nodes reachable from the source, without using waypoint w, are 'marked' by \( \overline{a}_v^{r,w} = 1 \)

\[
\begin{align*}
1 &= \sum_{r \in R} x_v^r \\
y_{u,v}^r &= 1 - \sum_{r' \leq r} x_u^{r'} \\
y_{u,v}^r &= \sum_{r' \leq r} x_u^{r'} \\
y_{u,v}^{r-1} + y_{u,v}^r &\geq y_{u,v}^r \\
y_{u,v}^{r-1} + y_{u,v}^r &\geq y_{u,v}^r \\
y_{u,v}^{r-1} + y_{u,v}^r &\leq \frac{l_v^r - l_u^r - 1}{|V| - 1} + 1 \\
\overline{a}_{s,w}^r &= 1 \\
\overline{a}_{v,w}^r &\geq \overline{a}_{u,w}^r + y_{u,v}^{r-1} - 1 \\
\overline{a}_{v,w}^r &\geq \overline{a}_{u,w}^r + y_{u,v}^r - 1 \\
\overline{a}_{d,w}^r &= 0
\end{align*}
\]
Computing Update Schedules

Guarantee WPE by reachability constraints:

Nodes reachable from the source, without using waypoint \( w \), are 'marked' by \( \overline{a}_{v \rightarrow w}^{r} = 1 \)

<table>
<thead>
<tr>
<th>State</th>
<th>Temporary Forwarding Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s ) ( v_1 ) ( v_2 ) ( d )</td>
<td>( s ) ( v_1 ) ( v_2 ) ( d )</td>
</tr>
<tr>
<td>( 1 ) ( 0 ) ( 0 ) ( 0 )</td>
<td>( 1 ) ( 0 ) ( 0 ) ( 0 )</td>
</tr>
</tbody>
</table>

\( 1 = \sum_{r \in \mathcal{R}} x_{v}^{r} \)

\( y_{u,v}^{r} = 1 - \sum_{r' \leq r} x_{u}^{r'} \)

\( y_{u,v}^{r} = \sum_{r' \leq r} x_{u}^{r'} \)

\( y_{u,v}^{r-1 \lor r} \geq y_{u,v}^{r-1} \)

\( y_{u,v}^{r-1 \lor r} \geq y_{u,v}^{r} \)

\( y_{u,v}^{r-1 \lor r} \leq \frac{l_{v}^{r} - l_{u}^{r} - 1}{|V| - 1} + 1 \)

\( \overline{a}_{s \rightarrow w}^{r} = 1 \)

\( \overline{a}_{v \rightarrow w}^{r} \geq \overline{a}_{u \rightarrow w}^{r} + y_{u,v}^{r-1} - 1 \)

\( \overline{a}_{v \rightarrow w}^{r} \geq \overline{a}_{u \rightarrow w}^{r} + y_{u,v}^{r} - 1 \)

\( \overline{a}_{d \rightarrow w}^{r} = 0 \)
Computing Update Schedules

- RLF can be realized similarly, but is more complex to compute.

\[
\begin{array}{l}
\text{Mixed-Integer Program 1: Optimal Rounds (-R-)} \\
\quad \text{(Obj)} \quad \min R \\
\quad 1 = \sum_{r \in \mathcal{R}} x_v^r \quad r \in \mathcal{R}, \quad \forall v \in \mathcal{V} \quad (1) \\
\quad y_{r,v}^u = 1 - \sum_{r' \leq r} x_{r,v}^{r'} \quad \forall r \in \mathcal{R}, \quad (u,v) \in \mathcal{E}_1 \quad (3) \\
\quad y_{r,v}^w = \sum_{r' < r} x_{r,v}^{r'} \quad \forall r \in \mathcal{R}, \quad (u,v) \in \mathcal{E}_2 \quad (4) \\
\quad \text{Decision Variant (D) A Flow Extension (F)} \\
\quad a_v^r = 1 \quad r \in \mathcal{R} \quad (5) \\
\quad d_v^r \geq a_v^r + y_{u,v}^{r-1} - 1 \quad \forall r \in \mathcal{R}, \quad (u,v) \in \mathcal{E} \quad (6) \\
\quad a_v^r \geq d_v^r + y_{u,v}^r - 1 \quad \forall r \in \mathcal{R}, \quad (u,v) \in \mathcal{E} \quad (7) \\
\quad y_{u,v}^{r,1} \geq a_v^r + y_{u,v}^{r-1} - 1 \quad \forall r \in \mathcal{R}, \quad (u,v) \in \mathcal{E} \quad (8) \\
\quad y_{u,v}^{r,1} \geq a_v^r + y_{u,v}^{r-1} \quad \forall r \in \mathcal{R}, \quad (u,v) \in \mathcal{E} \quad (9) \\
\quad y_{u,v}^{r,1} \leq \frac{l^v}{v} - \frac{l^w}{w} - \frac{1}{|\mathcal{V}| - 1} + 1 \quad \forall r \in \mathcal{R}, \quad (u,v) \in \mathcal{E} \quad (10) \\
\quad \overline{a}_v^{r,u} = 1 \quad r \in \mathcal{R}, \quad (u,v) \in \mathcal{WP} \quad (11) \\
\quad \overline{a}_v^{r,u} \geq \overline{a}_u^{r,u} + \overline{r} - 1 \quad r \in \mathcal{R}, \quad (u,v) \in \mathcal{WP} \quad (12) \\
\quad \overline{a}_v^{r,u} \geq \overline{a}_u^{r,u} + y_{u,v}^{r-1} - 1 \quad r \in \mathcal{R}, \quad (u,v) \in \mathcal{WP} \quad (13) \\
\quad \overline{a}_d^{r,u} = 0 \quad r \in \mathcal{R}, \quad (u,v) \in \mathcal{WP} \quad (14)
\end{array}
\]
Computing Update Schedules

- RLF can be realized similarly, but is more complex to compute.

- Objective: minimize #rounds

Mixed-Integer Program 1: Optimal Rounds (-R-)

\[
\begin{align*}
\text{min} & \quad R \\
\text{s.t.} & \quad R \geq r \cdot x_v^r \\
& \quad 1 = \sum_{r \in \mathcal{R}} x_v^r, \quad r \in \mathcal{R}, v \in V \\
& \quad y_{a,v}^r = 1 - \sum_{r' \leq r} x_{a,v}^{r'}^r, \quad r \in \mathcal{R}, (u,v) \in E_{x_1} \\
& \quad y_{a,v}^r = \sum_{r' < r} x_{a,v}^{r'}^r, \quad r \in \mathcal{R}, (u,v) \in E_{x_2} \\
& \quad a_v^r = 1, \quad r \in \mathcal{R} \\
& \quad a_v^r \geq a_u^r + y_{a,v}^{r-1}, \quad r \in \mathcal{R}, (u,v) \in E \\
& \quad a_v^r \geq a_u^r + y_{a,v}^r, \quad r \in \mathcal{R}, (u,v) \in E \\
& \quad y_{u,v}^{r-1} - y_{u,v}^r \geq a_u^r + y_{u,v}^{r-1} - 1, \quad r \in \mathcal{R}, (u,v) \in E \\
& \quad y_{u,v}^{r-1} - y_{u,v}^r \geq a_u^r + y_{u,v}^r - 1, \quad r \in \mathcal{R}, (u,v) \in E \\
& \quad y_{u,v}^{r-1} \leq \frac{|V| - 1}{|V| - 1} + 1, \quad r \in \mathcal{R}, (u,v) \in E \\
& \quad \overline{a}_v^{r,w} = 1, \quad r \in \mathcal{R}, w \in WP \\
& \quad \overline{a}_v^{r,w} \geq \overline{a}_u^{r,w} + y_{a,v}^{r-1}, \quad r \in \mathcal{R}, w \in WP, (u,v) \in E_{ WP}^{r-1} \\
& \quad \overline{a}_v^{r,w} \geq \overline{a}_u^{r,w} + y_{a,v}^r, \quad r \in \mathcal{R}, w \in WP, (u,v) \in E_{ WP}^{r} \\
& \quad \overline{a}_d^{r,w} = 0, \quad r \in \mathcal{R}, w \in WP
\end{align*}
\]
Computing Update Schedules

- RLF can be realized similarly, but is more complex to compute.
- Objective: minimize #rounds
- Some employed constraints are 'weak'; we propose:
  - Decision Variant (D)
  - A Flow Extension (F)

Mixed-Integer Program 1: Optimal Rounds (-R-)

\[
\begin{align*}
\text{min } & R \\
\text{subject to } & R \geq r \cdot x^r_v, \quad r \in R, \quad v \in V \\
& 1 = \sum_{r \in R} x^r_v, \quad v \in V \\
& y^r_{a, v} = 1 - \sum_{r' \leq r} x^{r'}_{a, v}, \quad r \in R, \quad (u, v) \in E_{\pi_1} \\
& y^r_{a, v} = \sum_{r' < r} x^{r'}_{a, v}, \quad r \in R, \quad (u, v) \in E_{\pi_2} \\
& a^r_e = 1, \quad r \in R \\
& a^e_w \geq a^e_u + y^r_{a, v} - 1, \quad r \in R, \quad (u, v) \in E \\
& a^e_w \geq a^r_e + y^r_{a, v} - 1, \quad r \in R, \quad (u, v) \in E \\
& y^{r \lor r}_{a, v} \geq a^r_e + y^{r \lor r}_{a, v} - 1, \quad r \in R, \quad (u, v) \in E \\
& y^{r \lor r}_{a, v} \geq a^{r \lor r}_e + y^{r \lor r}_{a, v} - 1, \quad r \in R, \quad (u, v) \in E \\
& y^{r \lor r}_{a, v} \leq \frac{y^r - y^{r \lor r}_{a, v}}{|V| - 1} + 1, \quad r \in R, \quad (u, v) \in E \\
& \overline{a}^r_{e, w} = 1, \quad r \in R, \quad w \in WP \\
& \overline{a}^r_{e, w} \geq \overline{a}^r_{e, v} + y^{r \lor r}_{a, v} - 1, \quad r \in R, \quad w \in WP, \quad (u, v) \in E_{\pi_{WP}}^{w} \\
& \overline{a}^r_{e, w} \geq \overline{a}^r_{e, v} + y^{r \lor r}_{a, v} - 1, \quad r \in R, \quad w \in WP, \quad (u, v) \in E_{\pi_{WP}}^{w} \\
& \overline{a}^r_{e, w} = 0, \quad r \in R, \quad w \in WP
\end{align*}
\]
Computing Update Schedules

- RLF can be realized similarly, but is more complex to compute.
- Objective: minimize \#rounds
- Some employed constraints are 'weak'; we propose:
  - Decision Variant (D)
  - A Flow Extension (F)

(D)
Only one update per round.

(F)
Additional s-d flows for each round to improve relaxations.
Computing Update Schedules

- RLF can be realized similarly, but is more complex to compute.
- Objective: minimize # rounds
- Some employed constraints are 'weak'; we propose:
  - Decision Variant (D)
  - A Flow Extension (F)

\[
\sum_{v \in V} x_v^r = 1 \quad r \in R.
\]

\[
\sum_{e \in \delta^+(s)} f_e^r = 1 \quad r \in R \quad (D)
\]

\[
\sum_{e \in \delta^+(v)} f_e^r = \sum_{e \in \delta^-(v)} f_e^r \quad r \in R, v \in V \setminus \{s, d\} \quad (F)
\]

\[
f_e^r \leq y_e^r \quad r \in R, e \in E_{\pi_1} \cup E_{\pi_2} \quad (19)
\]

\[
\sum_{e \in \delta^-(w)} f_e^r \geq 1 \quad r \in R, w \in WP \quad (20)
\]

\[
a_v^r \geq f_v^{r-1} \quad r \in R \quad (22^*)
\]

\[
a_v^r \geq f_v^r \quad r \in R \quad (23^*)
\]
Practice: Computational Experiments
Computational Setup

- Generate update instances at random by permuting nodes
- 12,600 instances overall
  - 10 to 30 switches with 1 to 3 waypoints
  - 200 instances for each combination
- (We discard scenarios which can a priori be determined to be infeasible to update, e.g. when waypoints are reordered)
Computational Setup

- Consider 8 different MIP formulations
  \[ S(\text{LF}) \text{ vs. } R(\text{LF}) \]
  \[ \text{D(cision)} \text{ vs. } - \]
  \[ \text{F(low Extension)} \text{ vs. } - \]

- Use Gurobi 6.5.0 to solve the formulations using branch-and-bound

- Terminate computations after 600 seconds
Computational Study: Solvability

WP=1

scenarios [percent]

nodes

feasible
Computational Study: Solvability

WP=1
More than 80% feasible

0.0
0.2
0.4
0.6
0.8
1.0

0.0
0.2
0.4
0.6
0.8
1.0

10
15
20
25
30

nodes

scenarios [percent]

feasible
Computational Study: Solvability

More than 80% feasible
Computational Study: Solvability

More than 80% feasible

WP=1

Provably impossible to update

feasible

infeasible
Computational Study: Solvability

More than 80% feasible

Provably impossible to update

few undecided instances
Computational Study: Solvability

More than 80% feasible

nearly always optimal solutions

few undecided instances
Computational Study: Solvability

WP=1

WP=2

WP=3

- feasible
- \text{optimal}_{SLF}
- \text{optimal}_{RLF}
- unknown
- infeasible
Computational Study: Solvability

more provably unupdateable instances

June 17th, 2016
Computational Study: Solvability

more provably unupdateable instances
more undecided instances
Computational Study: Solvability

more provably unupdateable instances
more undecided instances
less optimal solutions
Computational Study: Solvability

more provably unupdateable instances
more undecided instances
less optimal solutions
still: more than 65% feasible
Computational Study: RLF vs. SLF

Required Rounds for Feasible Scenarios

ECDF

rounds

RLF,10

SLF,10
Computational Study: RLF vs. SLF

Required Rounds for Feasible Scenarios

ECDF

rounds

RLF,10
SLF,10
RLF,30
SLF,30
Computational Study: RLF vs. SLF

50% to 90% within 4-6 rounds
Computational Study: Formulation Performance
Computational Study: Formulation Performance

Runtime Infeasibility Detection

![Graph showing runtime infeasibility detection for different formulations and node counts.](image-url)
Computational Study: Formulation Performance

Combining Decision and Flow extension yields infeasibility certificates approx. 2 orders of magnitude faster.
Computational Study: Formulation Performance

Runtime First Solution

Runtime [s]

nodes

10 20 30 10 20 30 10 20 30 10 20 30 10 20 30 10 20 30 10 20 30
Computational Study: Formulation Performance

Runtime First Solution

Median time for finding first solution:
< 1 second for 10 and 20 nodes
Computational Study: Formulation Performance

Runtime First Solution

Median time for finding first solution:
< 1 second for 10 and 20 nodes
< 10 seconds for 30 nodes
Related Work

Loop Freedom

- Model and greedy algorithm [Mahajan et al., HotNets '13]
- NP-hardness of optimization, introduction of RLF [Ludwig et al., PODC '15]
- Updating multiple schedules at the same time [Dudycz et al., DSN '16 (to appear)]
- Hardness of computing maximum set of switches to update [Amiri et al., SIROCCO '16 (to appear)]

Waypoint Enforcement

- Introduction of WPE, impossibility and first MIP formulations [Ludwig et al., HotNets '14]
Conclusion

Problem
- Dynamic network updates ensuring LF and WPE

Theory
- LF + WPE may conflict
- LF + WPE is NP-hard to decide
- (other results)

Practice
- MIP Formulations for computing schedules
- Flow and Decision extensions to improve infeasibility detection

Evaluation
- Many scenarios are updateable using few rounds
- MIP formulations have reasonable runtimes
Backup
Theory:
Reordering Waypoints is impossible
Reordering Waypoints is impossible

update to

\[ s \quad wp_2 \quad wp_1 \quad d \]
Reordering Waypoints is impossible
Reordering Waypoints is impossible

\[
\begin{align*}
  &o \quad \ldots \quad \rightarrow \quad \bullet \quad \rightarrow \quad \ldots \quad \rightarrow \quad \bullet \quad \rightarrow \quad \ldots \quad o \\
  &s \quad wp_1 \quad wp_2 \quad d \\
  &o \quad \ldots \quad \rightarrow \quad \bullet \quad \rightarrow \quad \ldots \quad \rightarrow \quad \bullet \quad \rightarrow \quad \ldots \quad o \\
  &s \quad wp_1 \quad wp_2 \quad d \\
  &o \quad \ldots \quad \rightarrow \quad \bullet \quad \rightarrow \quad \ldots \quad \rightarrow \quad \bullet \quad \rightarrow \quad \ldots \quad o \\
  &s \quad wp_2 \quad wp_1 \quad d
\end{align*}
\]
Reordering Waypoints is impossible

There must exist an update bypassing the first waypoint.
Theory: WPE requires waiting
WPE requires waiting

State

Temporary Forwarding Graph

\[ s \rightarrow v_1 \rightarrow v_2 \rightarrow d \]

\[ s \rightarrow v_1 \rightarrow v_2 \rightarrow d \]
WPE requires waiting

State

Temporary Forwarding Graph

\[ s \quad v_1 \quad v_2 \quad d \]

\[ s \quad v_1 \quad v_2 \quad d \]
WPE requires waiting

State

Temporary Forwarding Graph

\[ s \rightarrow v_1 \rightarrow v_2 \rightarrow d \]

\[ s \rightarrow v_1 \rightarrow v_2 \rightarrow d \]
WPE requires waiting

State

Temporary Forwarding Graph

Packets still traversing link will bypass WP
WPE requires waiting

State

\[
\begin{array}{cccc}
  s & v_1 & v_2 & d \\
\end{array}
\]

Temporary Forwarding Graph

\[
\begin{array}{cccc}
  s & v_1 & v_2 & d \\
\end{array}
\]

WPE requires upper bound on link delays, if the relative ordering of nodes changes.
Construction of 3-SAT Reduction: Remaining Connections