Snowcap: Synthesizing Network-Wide Configuration Updates

Tibor Schneider    Rüdiger Birkner    Laurent Vanbever

SIGCOMM’21, August 24, 2021
Network reconfigurations happen often

*Daily*  Routing policy adaptations¹

*Monthly*  Traffic engineering adjustments¹

*Yearly*  Major network redesign²


Network reconfigurations happen often and cause incidents

Alibaba revealed that 56% of the incidents are caused by configuration updates\(^3\)

Careless migration can cause unallowed and unnecessary traffic shifts

Objective $\phi$: Traffic from $r_x$ to $E$ must traverse $r_{fw}$. 
Careless migration can cause unallowed and unnecessary traffic shifts

Objective $\phi$:
Traffic from $r_x$ to $E$ must traverse $r_{fw}$. 
Careless migration can cause unallowed and unnecessary traffic shifts

Objective $\phi$: Traffic from $r_x$ to $E$ must traverse $r_{fw}$. 

Diagram showing network nodes $r_x$, $r_1$, $r_2$, and $E$. The link weights are as follows:
- $r_x$ to $r_1$: 10
- $r_1$ to $r_2$: 5
- $r_2$ to $E$: 10
- $r_{fw}$ to $r_1$: 8
Careless migration can cause unallowed and unnecessary traffic shifts

Objective $\phi$: Traffic from $r_x$ to $E$ must traverse $r_{fw}$.

a eBGP session: $E \rightarrow r_1$
Careless migration can cause unallowed and unnecessary traffic shifts.

Objective $\phi$: Traffic from $r_x$ to $E$ must traverse $r_{fw}$.

- **a** eBGP session: $E \rightarrow r_1$
- **b** Link weight $r_x \rightarrow r_1$: $10 \sim 20$
- **c** Link weight $r_{fw} \rightarrow r_1$: $5 \sim 9$
Careless migration can cause unallowed and unnecessary traffic shifts.

Objective $\phi$: Traffic from $r_x$ to $E$ must traverse $r_{fw}$.

- eBGP session: $E \rightarrow r_1$
- Link weight $r_x \rightarrow r_1$: 10 $\sim$ 20
- Link weight $r_{fw} \rightarrow r_1$: 5 $\sim$ 9
Careless migration can cause unallowed and unnecessary traffic shifts

Objective $\phi$: Traffic from $r_x$ to $E$ must traverse $r_{fw}$.

- **a** eBGP session: $E \rightarrow r_1$
- **b** Link weight $r_x \rightarrow r_1$: 10 $\sim$ 20
- **c** Link weight $r_{fw} \rightarrow r_1$: 5 $\sim$ 9
Careless migration can cause unallowed and unnecessary traffic shifts

Objective $\phi$:
Traffic from $r_x$ to $E$ must traverse $r_{fw}$.

- **a** eBGP session: $E \rightarrow r_1$
- **b** Link weight $r_x \rightarrow r_1$: 10 $\rightarrow$ 20
- **c** Link weight $r_{fw} \rightarrow r_1$: 5 $\rightarrow$ 9
Careless migration can cause unallowed and unnecessary traffic shifts

Objective $\phi$: Traffic from $r_x$ to $E$ must traverse $r_{fw}$.

- a. eBGP session: $E \rightarrow r_1$
- b. Link weight $r_x \rightarrow r_1$: 10 $\rightarrow$ 20
- c. Link weight $r_{fw} \rightarrow r_1$: 5 $\rightarrow$ 9
Careless migration can cause unallowed and unnecessary traffic shifts

Objective $\phi$: Traffic from $r_x$ to $E$ must traverse $r_{fw}$.

- a) eBGP session: $E \rightarrow r_1$
- b) Link weight $r_x \rightarrow r_1$: 10 $\sim$ 20
- c) Link weight $r_{fw} \rightarrow r_1$: 5 $\sim$ 9
Careless migration can cause unallowed and unnecessary traffic shifts

Objective $\phi$: Traffic from $r_x$ to $E$ must traverse $r_{fw}$.

- **a** eBGP session: $E \rightarrow r_1$
- **b** Link weight $r_x \rightarrow r_1$: $10 \sim 20$
- **c** Link weight $r_{fw} \rightarrow r_1$: $5 \sim 9$
Careless migration can cause unallowed and unnecessary traffic shifts.

Objective $\phi$: Traffic from $r_x$ to $E$ must traverse $r_{fw}$.

- **a** eBGP session: $E \rightarrow r_1$
- **b** Link weight $r_x \rightarrow r_1$: $10 \sim 20$
- **c** Link weight $r_{fw} \rightarrow r_1$: $5 \sim 9$
Careless migration can cause unallowed and unnecessary traffic shifts

Objective $\phi$: Traffic from $r_x$ to $E$ must traverse $r_{fw}$.

- eBGP session: $E \rightarrow r_1$
- Link weight $r_x \rightarrow r_1$: 10 $\sim$ 20
- Link weight $r_{fw} \rightarrow r_1$: 5 $\sim$ 9
Careless migration can cause unallowed and unnecessary traffic shifts.

Objective $\phi$: Traffic from $r_x$ to $E$ must traverse $r_{fw}$.

- **a** eBGP session: $E \rightarrow r_1$
- **b** Link weight $r_x \rightarrow r_1$: $10 \rightarrow 20$
- **c** Link weight $r_{fw} \rightarrow r_1$: $5 \rightarrow 9$
Careless migration can cause unallowed and unnecessary traffic shifts

Objective $\phi$: Traffic from $r_x$ to $E$ must traverse $r_{fw}$.

- **a** eBGP session: $E \rightarrow r_1$
- **b** Link weight $r_x \rightarrow r_1$: $10 \sim 20$
- **c** Link weight $r_{fw} \rightarrow r_1$: $5 \sim 9$
Careless migration can cause unallowed and unnecessary traffic shifts.

Objective $\phi$: Traffic from $r_x$ to $E$ must traverse $r_{fw}$.

- **a** eBGP session: $E \rightarrow r_1$
- **b** Link weight $r_x \rightarrow r_1$: $10 \sim 20$
- **c** Link weight $r_{fw} \rightarrow r_1$: $5 \sim 9$
A few best practices exist
Best practices are not enough
Best practices are not enough

Migration from iBGP full-mesh to route-reflection:

\[ \geq 50\% \text{ chance to violate reachability} \]
Best practices are not enough

Migration from iBGP full-mesh to route-reflection:

<table>
<thead>
<tr>
<th>Chance to Violate Reachability</th>
</tr>
</thead>
<tbody>
<tr>
<td>≥ 50%</td>
</tr>
<tr>
<td>Random order</td>
</tr>
<tr>
<td>70%</td>
</tr>
</tbody>
</table>
Best practices are not enough

Migration from iBGP full-mesh to route-reflection:

<table>
<thead>
<tr>
<th></th>
<th>$\geq 50%$ chance to violate reachability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random order</td>
<td>70%</td>
</tr>
<tr>
<td>Best practice order</td>
<td>25%</td>
</tr>
</tbody>
</table>
Existing systems are limited

<table>
<thead>
<tr>
<th>Safety guarantees</th>
<th>Best practice</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Existing systems are limited

<table>
<thead>
<tr>
<th></th>
<th>Safety guarantees</th>
<th>Any protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best practice</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>
Existing systems are limited

<table>
<thead>
<tr>
<th></th>
<th>Safety guarantees</th>
<th>Any protocols</th>
<th>Any scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best practice</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
Existing systems are limited

<table>
<thead>
<tr>
<th>Safety guarantees</th>
<th>Any protocols</th>
<th>Any scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best practice</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>“Offline” systems</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>


Existing systems are limited

<table>
<thead>
<tr>
<th>Best practice</th>
<th>Safety guarantees</th>
<th>Any protocols</th>
<th>Any scenarios</th>
<th>Low overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Offline” systems</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Existing systems are limited

<table>
<thead>
<tr>
<th>Safety guarantees</th>
<th>Any protocols</th>
<th>Any scenarios</th>
<th>Low overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Best practice</strong></td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>“Offline” systems</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>“Online” systems</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>


Existing systems are limited

<table>
<thead>
<tr>
<th></th>
<th>Safety guarantees</th>
<th>Any protocols</th>
<th>Any scenarios</th>
<th>Low overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Best practice</strong></td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td><strong>“Offline” systems</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td><strong>“Online” systems</strong></td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Snowcap</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>


Snowcap performs network reconfigurations automatically and safely

Input

$C_i$  $C_f$

initial & final configurations

$\phi$

hard spec

$f$

soft spec

Snowcap

Live Network
Snowcap performs network reconfigurations automatically and safely

Input

\[ C_i \]  
\[ C_f \]

initial & final configurations

\[ \phi \]

hard spec

\[ f \]

soft spec

Compute the **difference**

\[ \rightarrow \]

set of commands.

Live Network

Snowcap
Snowcap performs network reconfigurations automatically and safely

\[
C_i \quad C_f
\]

\[
\phi \quad \text{hard spec}
\]

\[
f \quad \text{soft spec}
\]

initial & final configurations

Describe how network properties change

Live Network

Snowcap

Compute the set of commands.
Snowcap’s specification language is extremely flexible

- **Basic Policies** for each flow:
  Reachability, isolation, redundancy, waypointing.

- **Linear Temporal Logic (LTL):**
  Express how the policy changes during migration.
Linear Temporal Logic captures policy changes

Firewall migration from $r_1$ to $r_2$

\[
\left( \bigwedge_{x \in \text{Flows}} r_1 \in \text{path}_x \ \text{UG} \ r_2 \in \text{path}_x \right)
\]

All flows can switch at different times.
Linear Temporal Logic captures policy changes

Firewall migration from $r_1$ to $r_2$

\[
\left( \bigwedge_{x \in \text{Flows}} r_1 \in \text{path}_x \text{ UXG } r_2 \in \text{path}_x \right)
\]

All flows are allowed to bypass the firewall for a short time.
Linear Temporal Logic captures policy changes

Firewall migration from \( r_1 \) to \( r_2 \)

\[
\left( \bigwedge_{x \in Flows} r_1 \in path_x \right) \text{ UG } \left( \bigwedge_{x \in Flows} r_2 \in path_x \right)
\]

All flows must change at the same time.
Snowcap performs network reconfigurations automatically and safely

Minimize cost function → good solution
Not all correct orderings are created equally
Not all correct orderings are created equally
Not all correct orderings are created equally

(cost: 2)

(b a)

(cost: 3)

(b a c)

(cost: 2)

(b a)
Not all correct orderings are created equally

\[
\begin{align*}
&\text{cost: 2} \\
&\text{cost: 0} \\
&\text{cost: 3}
\end{align*}
\]
Not all correct orderings are created equally

\[
\begin{align*}
\text{cost: } & 2 \\
\text{cost: } & 0 \\
\text{cost: } & 1 \\
\text{cost: } & 3
\end{align*}
\]
Snowcap performs network reconfigurations automatically and safely

Input:

- \( C_i \) (initial configuration)
- \( C_f \) (final configuration)
- \( \phi \) (hard specification)
- \( f \) (soft specification)

Find a valid and good ordering of commands and apply them one-by-one.

Snowcap

Live Network
It’s all about navigating the search space of possible reconfiguration orderings

The search space is both

- **sparse**; and
- **huge**.
It’s all about navigating the search space of possible reconfiguration orderings.

The search space is both

- **sparse**; and
- **huge**.
It’s all about navigating the search space of possible reconfiguration orderings.

The search space is both

- **sparse**; and
- **huge**.

---

13 / 27
It’s all about navigating the search space of possible reconfiguration orderings

The search space is both
- **sparse**; and
- **huge**.
1. **Exploration**
   How does Snowcap solve simple scenarios?

2. **Counter-example-guided search**
   How does Snowcap solve more difficult scenarios?

3. **Evaluation**
   How efficient and effective is Snowcap?
1. Exploration
   How does Snowcap solve simple scenarios?

2. Counter-example-guided search
   How does Snowcap solve more difficult scenarios?

3. Evaluation
   How efficient and effective is Snowcap?
The exploration algorithm is based on DFS traversal

- **a** eBGP session: $E \rightarrow r_1$
- **b** Link weight $r_x \rightarrow r_1$: 10 $\sim$ 20
- **c** Link weight $r_{fw} \rightarrow r_1$: 5 $\sim$ 9
The exploration algorithm is based on DFS traversal.
Sequences with a known, bad prefix are not explored

- eBGP session: $E \rightarrow r_1$
- Link weight $r_x \rightarrow r_1$: 10 $\sim$ 20
- Link weight $r_{fw} \rightarrow r_1$: 5 $\sim$ 9
Greedy minimization of the cost function

- eBGP session: $E \rightarrow r_1$
- Link weight $r_x \rightarrow r_1$: $10 \sim 20$
- Link weight $r_{fw} \rightarrow r_1$: $5 \sim 9$
Greedy minimization of the cost function

- eBGP session: \( E \rightarrow r_1 \)
- Link weight \( r_x \rightarrow r_1: 10 \leadsto 20 \)
- Link weight \( r_{fw} \rightarrow r_1: 5 \leadsto 9 \)

\[ a \quad b \quad c \]
\[ 0 \quad 10 \quad 0 \]
\[ 8 \quad 9 \quad 20 \]
\[ r_{fw} \]
\[ r_1 \]
\[ E \]
Greedy minimization of the cost function

eBGP session: $E \rightarrow r_1$

Link weight $r_x \rightarrow r_1$: $10 \rightarrow 20$

Link weight $r_{fw} \rightarrow r_1$: $5 \rightarrow 9$
Greedy minimization of the cost function

- eBGP session: $E \rightarrow r_1$
- Link weight $r_x \rightarrow r_1$: 10 $\sim$ 20
- Link weight $r_{fw} \rightarrow r_1$: 5 $\sim$ 9
Greedy minimization of the cost function

- **a** eBGP session: $E \rightarrow r_1$
- **b** Link weight $r_x \rightarrow r_1$: $10 \sim 20$
- **c** Link weight $r_{fw} \rightarrow r_1$: $5 \sim 9$
Greedy minimization of the cost function

- eBGP session: $E \rightarrow r_1$
- Link weight $r_x \rightarrow r_1$: 10 $\sim$ 20
- Link weight $r_{fw} \rightarrow r_1$: 5 $\sim$ 9
Greedy minimization of the cost function

- eBGP session: $E \to r_1$
- Link weight $r_x \to r_1$: $10 \sim 20$
- Link weight $r_{fw} \to r_1$: $5 \sim 9$
Greedy minimization of the cost function

- eBGP session: $E \rightarrow r_1$
- Link weight $r_x \rightarrow r_1$: $10 \sim 20$
- Link weight $r_{fw} \rightarrow r_1$: $5 \sim 9$
DFS Exploration works well in *most* cases

However: What if we get stuck? Bad decision *early* may cause problems *later*.

→ Actively find the problem!
DFS Exploration works well in *most* cases

**However:** What if we get stuck? Bad decision *early* may cause problems *later*.

→ Actively find the problem!
1. **Exploration**
   How does Snowcap solve simple scenarios?

2. **Counter-example-guided search**
   How does Snowcap solve more difficult scenarios?

3. **Evaluation**
   How efficient and effective is Snowcap?
Snowcap uses counter-example-guided search to resolve difficult dependencies

- performs normal exploration until a dead end
- follows a divide-and-conquer approach
Initial configuration
Initial configuration
Final configuration
Final configuration
DFS traversal first applies $a$ followed by $b$
DFS traversal first applies $a$ followed by $b$. 

![Diagram showing DFS traversal paths]
DFS traversal first applies \( a \) followed by \( b \)
Either $c$ and $d$ will now cause forwarding loops.
Either $c$ and $d$ will now cause forwarding loops
Either $c$ and $d$ will now cause forwarding loops
Reduce to minimal reproducing sequence

current ordering: $a \ b \ c$
Reduce to minimal reproducing sequence

current ordering: $ab$c
Reduce to minimal reproducing sequence

current ordering: \[ b \text{ c} \]
Reduce to minimal reproducing sequence
Solve the minimal sequence using the DFS algorithm
Extend with yet unused commands

current ordering: \[d, c\]
Solve the extended sequence

current ordering: \(d c\)
Keep the found solution for future iterations

current ordering: \(d \ c\)

\[
\begin{align*}
\text{reduce} & \quad \downarrow \\
\text{solve} & \quad \xmark \\
\text{extend} & \\
\end{align*}
\]
1. Exploration
How does Snowcap solve simple scenarios?

2. Counter-example-guided search
How does Snowcap solve more difficult scenarios?

3. Evaluation
- How does Snowcap scale?
- How effective is Snowcap?
We evaluate Snowcap on a wide range of topologies and migration scenarios

- \( \approx 80 \) Topologies from Topology Zoo\(^6\)
- Common migration scenarios\(^7\)
- Random link weights and iBGP topologies.

Snowcap finds solutions within seconds

Migration from iBGP full-mesh to route-reflection.

≥ 50% chance to violate reachability

<table>
<thead>
<tr>
<th>Order Type</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random order</td>
<td>70%</td>
</tr>
<tr>
<td>Best practice order</td>
<td>25%</td>
</tr>
</tbody>
</table>
Snowcap finds solutions within seconds

Migration from iBGP full-mesh to route-reflection.

<table>
<thead>
<tr>
<th></th>
<th>50% chance to violate reachability</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random order</td>
<td>70%</td>
<td></td>
</tr>
<tr>
<td>Best practice order</td>
<td>25%</td>
<td></td>
</tr>
<tr>
<td>Snowcap</td>
<td>0%</td>
<td>at most 12s*</td>
</tr>
</tbody>
</table>

*for 3081 commands on 82 routers.
Snowcap’s runtime scales very well with increasing complexity
Snowcap’s runtime scales very well with increasing complexity
Snowcap’s runtime scales very well with increasing complexity
Snowcap’s runtime scales very well with increasing complexity
Snowcap’s runtime scales very well with increasing complexity.

- **Number of commands**
  - Random permutations: Time increases linearly.
  - Snowcap (DFS exploration): Time increases slowly.

- **Number of difficult dependencies**
  - Random permutations: Time increases dramatically.

Graphs show the comparison between random permutations and Snowcap's DFS exploration.
Snowcap’s runtime scales very well with increasing complexity.

---

**Time [s]**

- **Number of commands**
  - Random permutations
  - Snowcap (DFS exploration)

---

**Time [s]**

- **Number of difficult dependencies**
  - DFS exploration
  - Random permutations
Snowcap’s runtime scales very well with increasing complexity.
Snowcap is effective at finding good orderings

Cost (# of traffic shifts)

Network acquisition
Snowcap is effective at finding good orderings

Cost (# of traffic shifts)

Network acquisition

random permutations
Snowcap is effective at finding good orderings

Cost (# of traffic shifts)

Network acquisition
Snowcap is effective at finding good orderings

Cost (# of traffic shifts)

Network acquisition

Cost (# of traffic shifts)

Double all link weights
Predictable overhead for Snowcap’s greedy optimization

# of states explored

 Networks from Topology Zoo

79
Predictable overhead for Snowcap’s greedy optimization

# of states explored

Networks from Topology Zoo

Snowcap ($\phi$ only)
Predictable overhead for Snowcap’s greedy optimization

# of states explored

Networks from Topology Zoo

Snowcap ($\phi$ and $f$)

Snowcap ($\phi$ only)
Predictable overhead for Snowcap’s greedy optimization

![Graph showing # of states explored vs Networks from Topology Zoo]

- Snowcap (φ and f)
- Snowcap (φ only)
- Random order (φ only)

# of states explored

Networks from Topology Zoo
Snowcap guarantees safe network reconfigurations

Snowcap is open-source
https://snowcap.ethz.ch
Snowcap: Synthesizing Network-Wide Configuration Updates

Tibor Schneider  Rüdiger Birkner  Laurent Vanbever

https://nsg.ee.ethz.ch

Input

\[ C_i, C_f \]

\( \phi \)

hard spec

\( f \)

soft spec

Snowcap

Live Network