Programmable Networks with Synthesis

Ahmed ElHassany  Petar Tsankov  Laurent Vanbever  Martin Vechev
Network Misconfigurations are Common

Amazon server outage affects millions of companies and causes online chaos

Amazon's massive AWS outage was caused by human error

CloudFlare apologizes for Telia screwing you over

Level3 switch config blunder blamed for US-wide VoIP blackout

The summer of network misconfigurations

What Makes Network Configuration Hard?

High-level, global routing requirements

Low-level, local router configurations

Network N1

Network N2

Network N3
Example

R1: Packets from N1 to N2 must follow the path A → D
R2: Packets from N1 to N3 must follow the path A → B → C → D
Example

R1: Packets from N1 to N2 must follow the path A → D
R2: Packets from N1 to N3 must follow the path A → B → C → D
### Example

**R1:** Packets from **N1** to **N2** must follow the path $A \rightarrow D$

**R2:** Packets from **N1** to **N3** must follow the path $A \rightarrow B \rightarrow C \rightarrow D$

- **Packets to **N3** not forwarded correctly**
Example

Network N1

Network N2

Network N3

Static routes table

<table>
<thead>
<tr>
<th>Network</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>N3</td>
<td>B</td>
</tr>
</tbody>
</table>

Add a static route to A configuration

Configure A to prefer static routes over OSPF

R1: Packets from N1 to N2 must follow the path A → D
R2: Packets from N1 to N3 must follow the path A → B → C → D
Example

B non-deterministically forwards packets for N3 to either A or C

Static routes table

<table>
<thead>
<tr>
<th>Network</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>N3</td>
<td>B</td>
</tr>
</tbody>
</table>

- **Network N1**
  - Packets from **N1** to **N2** must follow the path **A ➔ D**
  - **R1**: Packets from **N1** to **N2** must follow the path **A ➔ D**

- **Network N3**
  - Packets from **N1** to **N3** must follow the path **A ➔ B ➔ C ➔ D**
  - **R2**: Packets from **N1** to **N3** must follow the path **A ➔ B ➔ C ➔ D**
Example

Network N1:
- R1: Packets from N1 to N2 must follow the path A → D
- R2: Packets from N1 to N3 must follow the path A → B → C → D

Static routes table:

<table>
<thead>
<tr>
<th>Network</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>N3</td>
<td>B</td>
</tr>
</tbody>
</table>

Lower cost to 5
**Example**

**Static routes table**

<table>
<thead>
<tr>
<th>Network</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>N3</td>
<td>B</td>
</tr>
</tbody>
</table>

**Router configurations must be such that:**

1. A prefers static routes over OSPF
2. A has a static route to B for N3
3. A → D → N2 must be lowest cost from A to N2
4. B → C → D → N3 must be lowest cost from B to N3

**R1:** Packets from N1 to N2 must follow the path A → D

**R2:** Packets from N1 to N3 must follow the path A → B → C → D

Lower cost to 5
Current Practice

Initially not configured

Network topology

Routing requirements

Operators manually configure each router

All routers are configured
Current Practice

Initially not configured

Problems and Challenges

- *Diversity* in protocol expressiveness
- Protocol *dependencies*
- No *correctness* guarantees

All routers are configured
Wanted: Programmable Networks with Synthesis

Operators manually configure each router

Network topology
Routing requirements

Wanted: Programmable Networks with Synthesis
Wanted: Programmable Networks with Synthesis

Automatically configure routers with synthesis
Wanted: Programmable Networks with Synthesis

- Network topology
- Routing requirements

How is the behavior of routers captured?

How to find a configuration that conforms to the requirements?

Automatically configure routers with synthesis

All routers are configured

Operators manually configure each router

All routers are configured
Programmable Networks with *Synthesis*: Dimensions

<table>
<thead>
<tr>
<th>Deployment scenarios</th>
<th>Datacenter</th>
<th>ISP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Incremental</td>
<td>Enterprise</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Routing protocols</th>
<th>Deterministic</th>
<th>Probabilistic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OSPF, BGP, MPLS</td>
<td>ECMP, Gossip</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Paths</th>
<th>Reachability</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Isolation</td>
<td>Waypointing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Synthesis Techniques</th>
<th>Enumerative learning</th>
<th>Probabilistic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Symbolic execution <em>(SyNET</em>)</td>
<td>Constraint solving</td>
</tr>
</tbody>
</table>

*SyNET*: http://synet.ethz.ch
Capturing Network Behavior

Key idea: Express routing protocols, along with their dependencies, in *stratified Datalog*
Datalog (Graph Reachability)

**Input**

\[ \text{link}(n1, a) \]
\[ \text{link}(a, b) \]
...

**Program**

\[ \text{path}(X, Y) \leftarrow \text{link}(X, Y) \]
\[ \text{path}(X, Y) \leftarrow \text{link}(X, Z), \text{path}(Z, Y) \]

**Query**

\[ \text{path}(n1, n2)? \]

Can we capture the network’s forwarding plane?
Datalog (Shortest-path Routing)

**Input**

`\text{link}(n_1, a, 10)`

Captures the router configuration

**Program**

\[
\text{path}(\text{Router}, \text{Net}, \text{Net}, \text{Cost}) \leftarrow \text{link}(\text{Router}, \text{Net}, \text{Cost}),
\]

\[
\text{path}(\text{Router}, \text{Net}, \text{NextHop}, C_1 + C_2) \leftarrow \text{link}(\text{Router}, \text{NextHop}, C_1),
\]

\[
\text{path}(\text{NextHop}, \text{Net}, X, C_2)
\]

\[
\text{sp}(\text{Router}, \text{Net}, \text{NextHop}, \text{min}(C)) \leftarrow \text{path}(\text{Router}, \text{Net}, \text{NextHop}, C)
\]

\[
\text{fwd}(\text{Router}, \text{Net}, \text{NextHop}) \leftarrow \text{sp}(\text{Router}, \text{Net}, \text{NextHop}, C)
\]

**Query**

\[
\text{fwd}(a, n_2, ?)
\]

Checks a property on the forwarding plane

Add OSPF cost to links
Routing Requirements

Paths

Packets for traffic class TC must follow the path \[ r_1 \rightarrow \cdots \rightarrow r_n \]

\[ \text{fwd}(r_1, tc, r_2) \land \cdots \land \text{fwd}(r_{n-1}, tc, r_n) \]
Routing Requirements

Paths
Packets for traffic class $TC$ must follow the path
$r_1 \rightarrow \cdots \rightarrow r_n$
$f_{wd}(r_1, tc, r_2) \land \cdots \land f_{wd}(r_{n-1}, tc, r_n)$

Traffic isolation
The paths for two distinct traffic classes $tc_1$ and $tc_2$ do not share links in the same direction
$\forall R_1, R_2. f_{wd}(R_1, tc_1, R_2) \Rightarrow \neg f_{wd}(R_1, tc_2, R_2)$
Routing Requirements

Paths
Packets for traffic class $TC$ must follow the path $r_1 \rightarrow \cdots \rightarrow r_n$

Traffic isolation
The paths for two distinct traffic classes $tc_1$ and $tc_2$ do not share links in the same direction

Reachability
Packets for traffic class $tc$ can reach router $r_2$ from router $r_1$
Routing Requirements

**Paths**
Packets for traffic class $TC$ must follow the path $r_1 \rightarrow \cdots \rightarrow r_n$.

**Traffic isolation**
The paths for two distinct traffic classes $tc_1$ and $tc_2$ do not share links in the same direction.

**Reachability**
Packets for traffic class $tc$ can reach router $r_2$ from router $r_1$.

**Loop-freeness**
The forwarding plane has no loops.
Network-wide Configuration Synthesis
Network-wide Configuration Synthesis

Network specification $N$  
(OSPF, BGP, MPLS, ...)

Routing requirements $R$  
(isolation, reachability, reliability)

Synthesis problem:  
Find a configuration $C$ such that $N$ configured with $C$ satisfies $R$

Network-wide configuration $C$  
(protocol configurations for routers)

Datalog program $P$

Datalog query $Q$

(Input) Synthesis problem:  
Find an input $I$ such that $P, I \models Q$

Datalog input $I$
Network-wide Configuration Synthesis

**Network specification** \( N \)
(OSPF, BGP, MPLS, ...)

**Routing requirements** \( R \)
(isolation, reachability, reliability)

**Synthesis problem:**
Find a configuration \( C \) configured with \( N \) such that \( R \)

**Datalog program** \( P \)

**Datalog query** \( Q \)

**Network-wide configuration** \( C \)
(protocol configurations for routers)

**Datalog input** \( I \)

**Problems:**
- No input synthesis tools for Datalog
- Problem is undecidable
Input Synthesis for Datalog

Key idea: Reduce to solving logical constraints
Input Synthesis for Datalog (via Constraint Solving)

Datalog program $P$

\[
\begin{align*}
\text{path}(X,Y) & \leftarrow \text{link}(X,Y) \\
\text{path}(X,Y) & \leftarrow \text{link}(X,Z), \text{path}(Z,Y)
\end{align*}
\]

Datalog query $Q$

\[
\begin{align*}
\text{path}(a,c) \land \neg \text{link}(a,c)
\end{align*}
\]

Generate constraints

\[
\begin{align*}
\forall X,Y. & \text{path}_1(X,Y) \iff \text{link}(X,Y) \\
\forall X,Y. & \text{path}_2(X,Y) \iff (\text{link}(X,Y) \lor (\exists Z. (\text{link}(X,Z) \land \text{path}_1(Z,Y)))) \\
\text{path}_2(a,c) & \land \neg \text{link}(a,c)
\end{align*}
\]

Constraints $\psi$

Constraint solving

\[
\begin{align*}
\text{link}(a,b), \text{link}(b,c), \\
\text{path}_1(a,b), \text{path}_1(b,c) \\
\text{path}_2(a,b), \text{path}_2(b,c), \text{path}_2(a,c)
\end{align*}
\]

Model $M \models \psi$

Derive input

\[
\begin{align*}
\text{link}(a,b), \text{link}(b,c)
\end{align*}
\]

Datalog input $I$
Input Synthesis for Datalog (via Constraint Solving)

Datalog program \( \mathcal{P} \)

\[
\text{path}(X,Y) \leftarrow \text{link}(X,Y) \\
\text{path}(X,Y) \leftarrow \text{link}(X,Z), \text{path}(Z,Y)
\]

Datalog query \( \mathcal{Q} \)

\[
\text{path}(a,c) \land \neg \text{link}(a,c)
\]

\[\forall X,Y. \text{path}_1(X,Y) \iff \text{link}(X,Y)\]
\[\forall X,Y. \text{path}_2(X,Y) \iff (\text{link}(X,Y) \lor (\exists Z. (\text{link}(X,Z) \land \text{path}_1(Z,Y))))\]
\[\text{path}_2(a,c) \land \neg \text{link}(a,c)\]

Theorem: \( \mathcal{M} \models \psi \) if and only if \( \mathcal{P}, \mathcal{I} \models \mathcal{Q} \)

Constraints \( \psi \)

\[
\text{link}(a,b), \text{link}(b,c), \text{path}_1(a,b), \text{path}_1(b,c), \text{path}_2(a,b), \text{path}_2(b,c), \text{path}_2(a,c)
\]

Model \( \mathcal{M} \models \psi \)

Datalog input \( \mathcal{I} \)

\[
\text{link}(a,b), \text{link}(b,c)
\]
Generating Constraints

\[
\text{link}(N1, A, ?) \\
\text{...}
\]

\[
\text{path}(\text{Router, Net, Net, Cost}) \leftarrow \\
\text{link}(\text{Router, Net, Cost})
\]

\[
\text{path}(\text{Router, Net, NextHop, } C_1 + C_2) \leftarrow \\
\text{link}(\text{Router, NextHop, } C_1), \\
\text{path}(\text{NextHop, Net, } X, C_2)
\]

\[
\text{sp}(\text{Router, Net, NextHop, min(C)}) \leftarrow \\
\text{path}(\text{Router, Net, NextHop, } C)
\]

\[
\text{fwd}(\text{Router, Net, NextHop}) \leftarrow \\
\text{sp}(\text{Router, Net, NextHop, } C)
\]

\[
fwd(a, n_2, d) \land fwb(a, n_3, b) \land \ldots
\]

\[
\forall X, Y. \text{path}_1(X, Y) \iff \text{link}(X, Y) \\
\forall X, Y. \text{path}_2(X, Y) \\
\iff \left( \text{link}(X, Y) \lor \exists Z. (\text{link}(X, Z) \land \text{path}_1(Z, Y)) \right) \\
\text{path}_2(a, c) \land \lnot \text{link}(a, c)
\]
Generating Constraints

\( \text{link}(N1, A, ?) \)

\( \ldots \)

\( \text{path}(\text{Router}, \text{Net}, \text{Net}, \text{Cost}) \leftarrow \text{link}(\text{Router}, \text{Net}, \text{Cost}) \)

\( \text{path}(\text{Router}, \text{Net}, \text{NextHop}, C_1 + C_2) \leftarrow \text{link}(\text{Router}, \text{NextHop}, C_1), \text{path}(\text{NextHop}, \text{Net}, X, C_2) \)

\( \text{sp}(\text{Router}, \text{Net}, \text{NextHop}, \min(C)) \leftarrow \text{path}(\text{Router}, \text{Net}, \text{NextHop}, C) \)

\( \text{fwd}(\text{Router}, \text{Net}, \text{NextHop}) \leftarrow \text{sp}(\text{Router}, \text{Net}, \text{NextHop}, C) \)

\( \text{fwd}(a,n_2,d) \land \text{fwd}(a,n_3,b) \land \ldots \)
Synthesis Algorithm (Naïve Approach)

Input

Box 1 (ψ₁)

Box 2 (ψ₂)

Box 3 (ψ₃)

Requirements

\[ M_1 \models ψ_1 \wedge I_2 \]

\[ M_2 \models ψ_2 \wedge I_3 \]

\[ M_3 \models ψ_3 \wedge \text{Reqs} \]
Naïve approach does NOT work

1. Not all valid SMT models are valid network configurations.
2. The search space of valid inputs is huge.
Synthesis Algorithm (Our Approach)

**Input Constraints**

*Ensure the synthesized configurations are valid*  

*Ex. OSPF Link costs are positive numbers*
Synthesis Algorithm (Our Approach)

**Input Constraints**

Ensure the synthesized configurations are valid

Ex. OSPF Link costs are positive numbers

---

**Domain specific constraints**

Reduce the search space

Ex. Routers only forward to neighbours
Synthesis Algorithm (Our Approach)

**Input Constraints**
*Ensure the synthesized configurations are valid*

*Ex. OSPF Link costs are positive numbers*

**Domain specific constraints**
*Reduce the search space*

*Ex. Routers only forward to neighbours*

**Learn from mistakes**
*Don’t reuse the same invalid input*

\[ M_{boxi} \models \psi \land l_{i+1} \land \neg l_i \]
Synthesis Algorithm (Our Approach)

**Input Constraints**
*Ensure the synthesized configurations are valid*  
*Ex. OSPF Link costs are positive numbers*

**Domain specific constraints**
*Reduce the search space*  
*Ex. Routers only forward to neighbours*

**Learn from mistakes**
*Don’t reuse the same invalid input*  
*\[ M_{boxi} \models \psi^i \land l_{i+1}^i \land \neg l_i^i \]*

**Partial evaluation**
*Don’t compute already known values*
Network-wide Configuration Synthesis

Encode as a Datalog program

Constraints on the forwarding plane computed by the routers

Automatically configure routers with synthesis

Datalog input identifies correct configuration

Routing requirements

Via reduction to input synthesis for Datalog
Implementation

The SyNET system ([http://synet.ethz.ch](http://synet.ethz.ch))

- Written in Python (~ 4K lines of code)
- Protocols encoded in stratified Datalog (~ 100 rules)
- Uses the Z3 constraint solver
- Outputs CISCO configurations
- Supports BGP, OSPF, and static routes

CISCO configuration output by SyNET

```plaintext
! A snippet from router A
interface f0/1
  ip address 10.0.0.2 255.255.255.254
  ip ospf cost 10
  description "To B"
interface f0/0
  ip address 10.0.0.0 255.255.255.254
  ip ospf cost 65530
  description "To C"
interface f1/0
  ip address 10.0.0.4 255.255.255.254
  ip ospf cost 65530
  description "To D"
```
Experiments
Experiment

US-based network connecting major universities and research institutes

<table>
<thead>
<tr>
<th>Protocols / # Traffic classes</th>
<th>1 class</th>
<th>5 classes</th>
<th>10 classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static</td>
<td>1.3s</td>
<td>2.0s</td>
<td>4.0s</td>
</tr>
<tr>
<td>Static + OSPF</td>
<td>9.0s</td>
<td>21.3s</td>
<td>49.3s</td>
</tr>
<tr>
<td>Static + OSPF + BGP</td>
<td>13.3s</td>
<td>22.7s</td>
<td>1m19.7s</td>
</tr>
</tbody>
</table>

Synthesis Times
Scalability Experiment

- Grid topologies with up to 64 routers
- Requirements for 10 traffic classes

![Graph showing synthesis time for different numbers of routers and routing protocols. The x-axis represents the number of routers, and the y-axis represents synthesis time in seconds. The graph shows that the synthesis time increases as the number of routers increases. There are three lines representing different routing configurations: Static + OSPF, Static + OSPF + BGP, and Static routes. The chart indicates that the synthesis time is less than 1 hour for Static routes and less than 24 hours for Static + OSPF.](image)
Summary: Programmable Networks with Synthesis

- **Global requirements vs local configurations**
- **Network-wide configuration synthesis**
- **Approach scales to realistic problems**