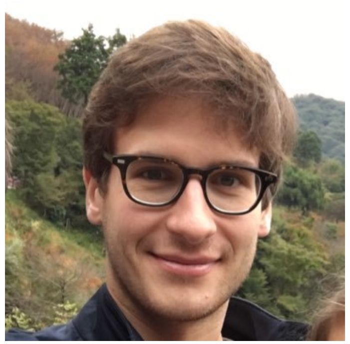
SRL
SOFTWARE RELIABILITY LAB

Networked Systems
ETH Zürich

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

MARCH 1, 2017 12:39PM

AMAZON'S giant servers crashed today causing chaos for millions of companies and people that use the cloud service and affected everythin[g] larger web sites to people's smarthomes and even library catalogues.

Amazon's massive AWS outage was caused by human error

One incorrect command and the whole internet suffers.

BY JASON DEL REY | @DELREY | MAR 2, 2017, 2:20PM EST

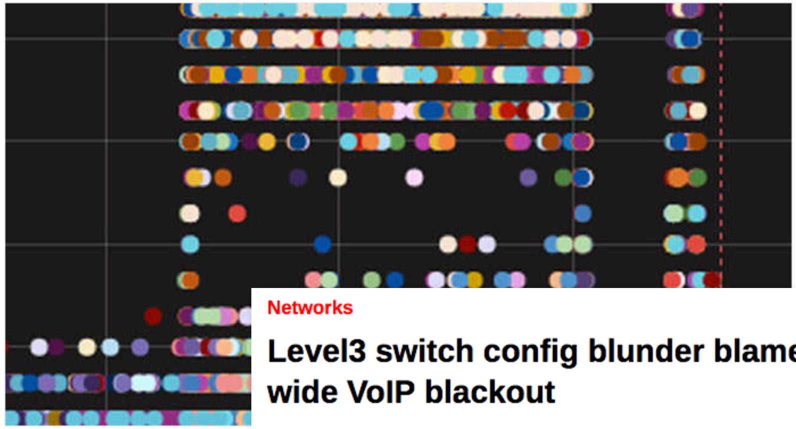
TWEET SHARE LINKEDIN

Sean Gallup / Getty

Amazon today blamed human error for the the big AWS outage that took down a bunch of large internet sites for several hours on Tuesday afternoon.

CloudFlare apologizes for Telia screwing you over

Unhappy about massive outage



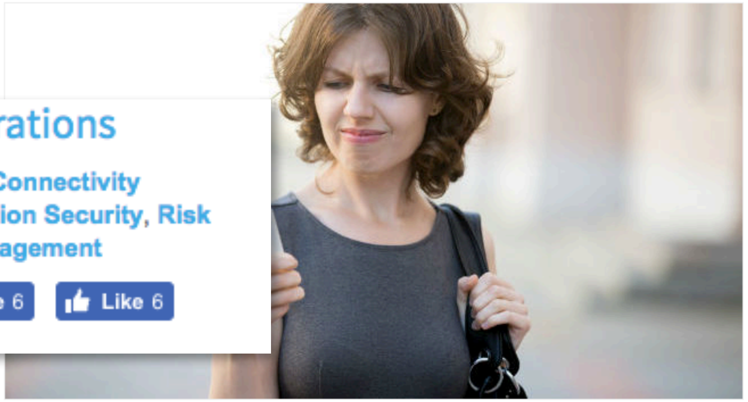
Ouch, th

21 Jun 2016 at 20:34, Kieren Mc

Networks

Level3 switch config blunder blamed for US-wide VoIP blackout

Network dropped calls because it was told to



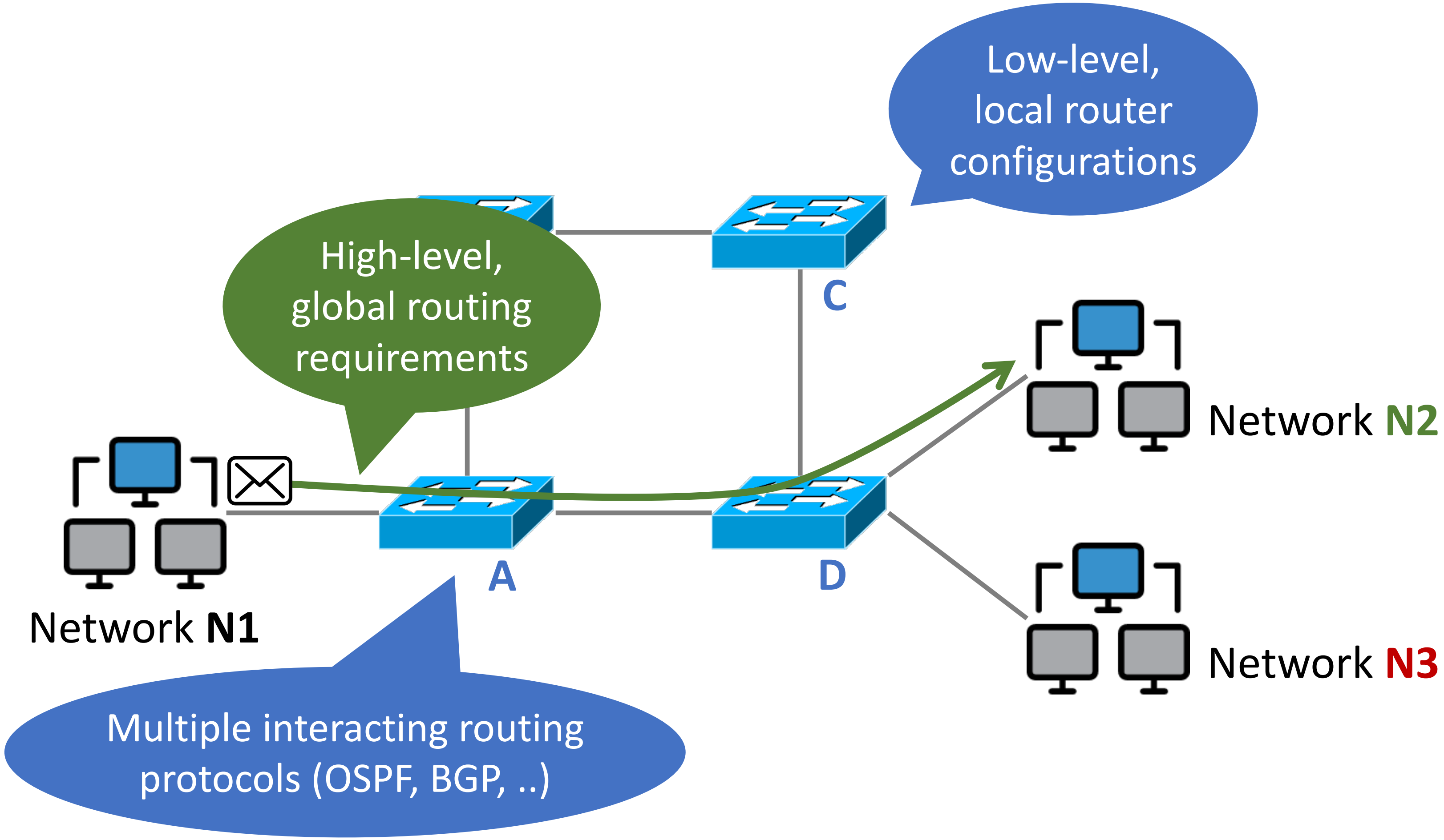
5 Oct 2016 at 11:33, Shaun Nichols

The summer of network misconfigurations

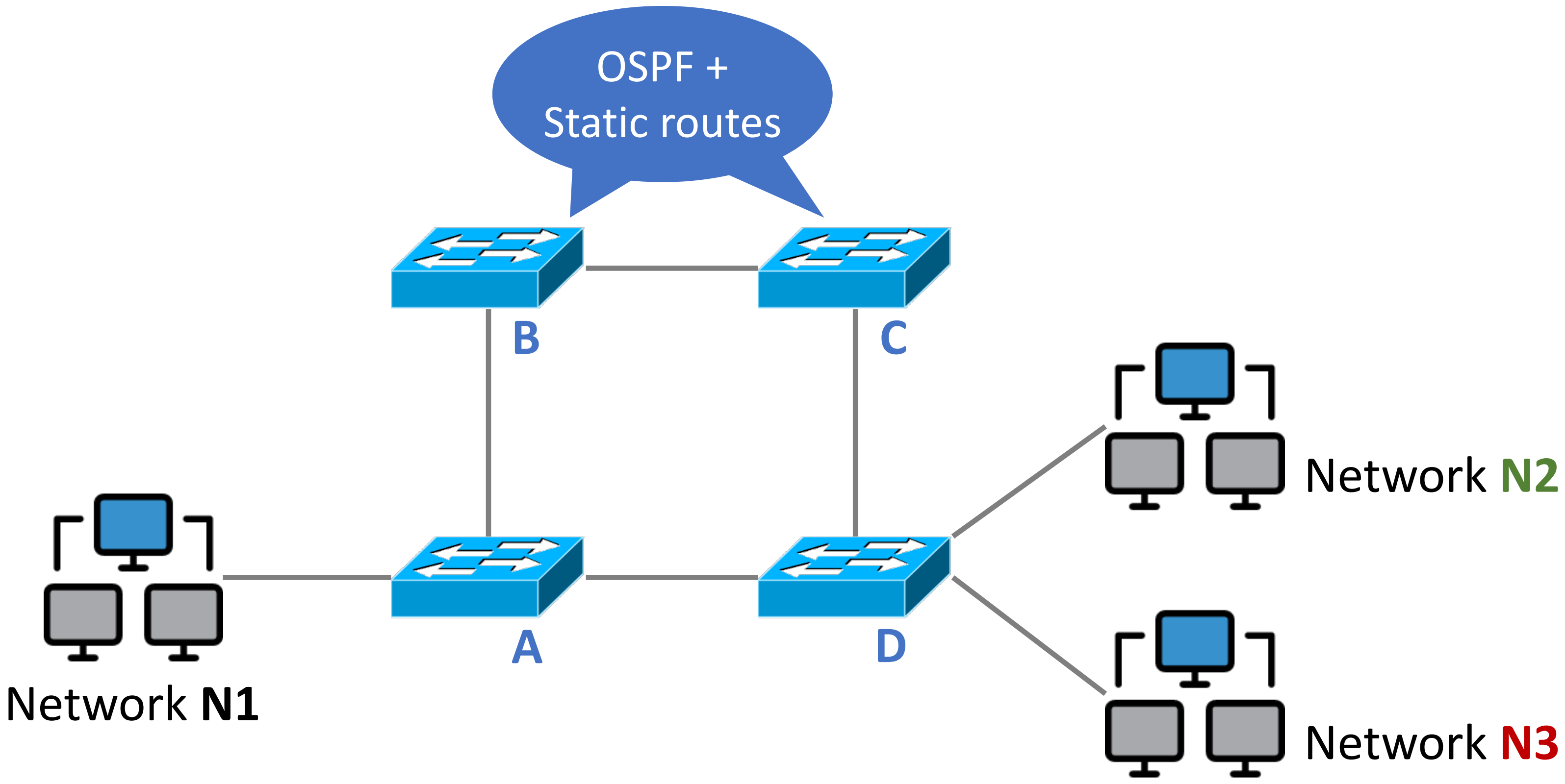
by Joanne Godfrey on August 11, 2016 in Application Connectivity Management, Firewall Change Management, Information Security, Risk Management and Vulnerabilities, Security Policy Management

EMAIL Share 32 Tweet Share 6 Like 6

What Makes Network Configuration Hard?



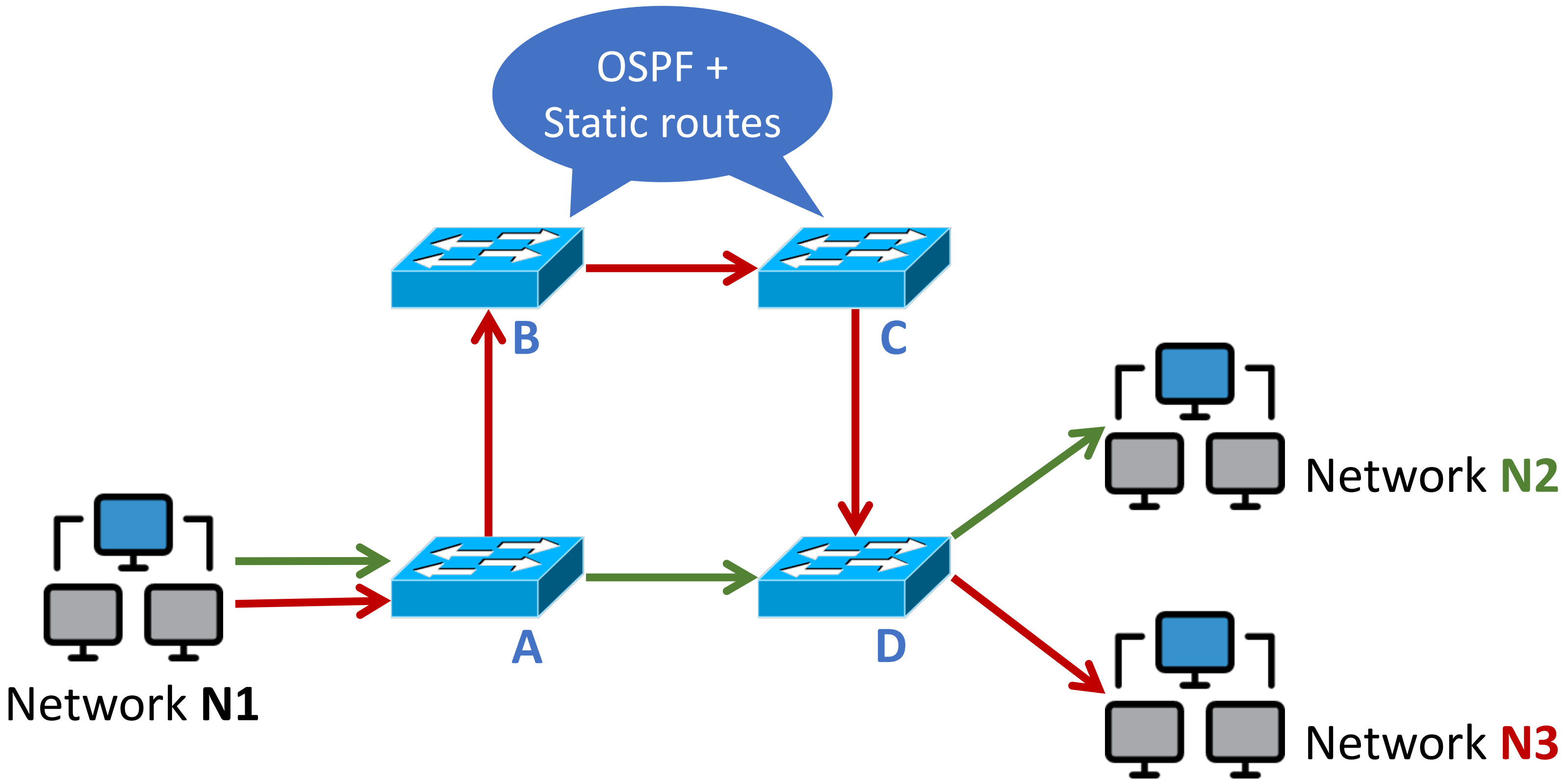
Example



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

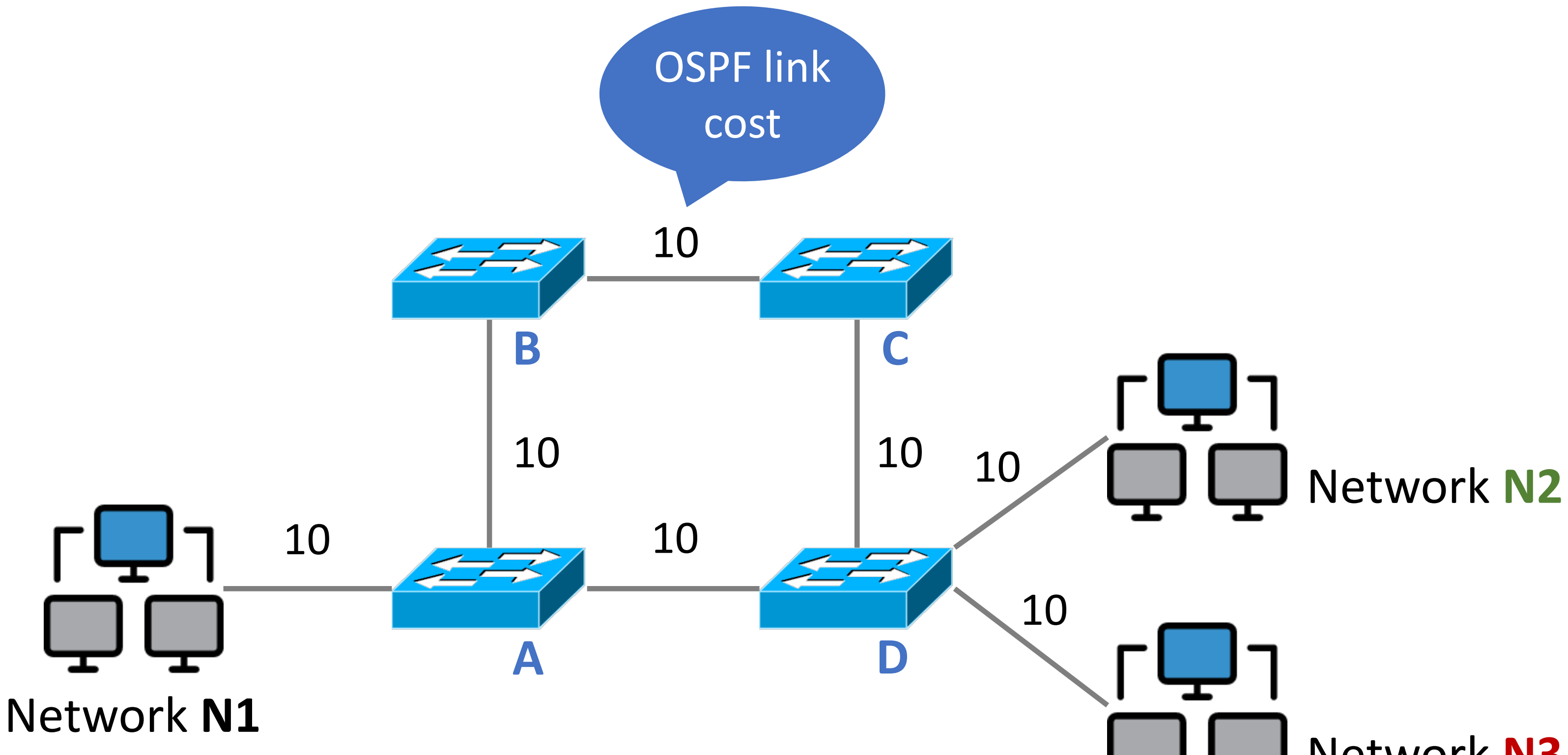
Example



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**

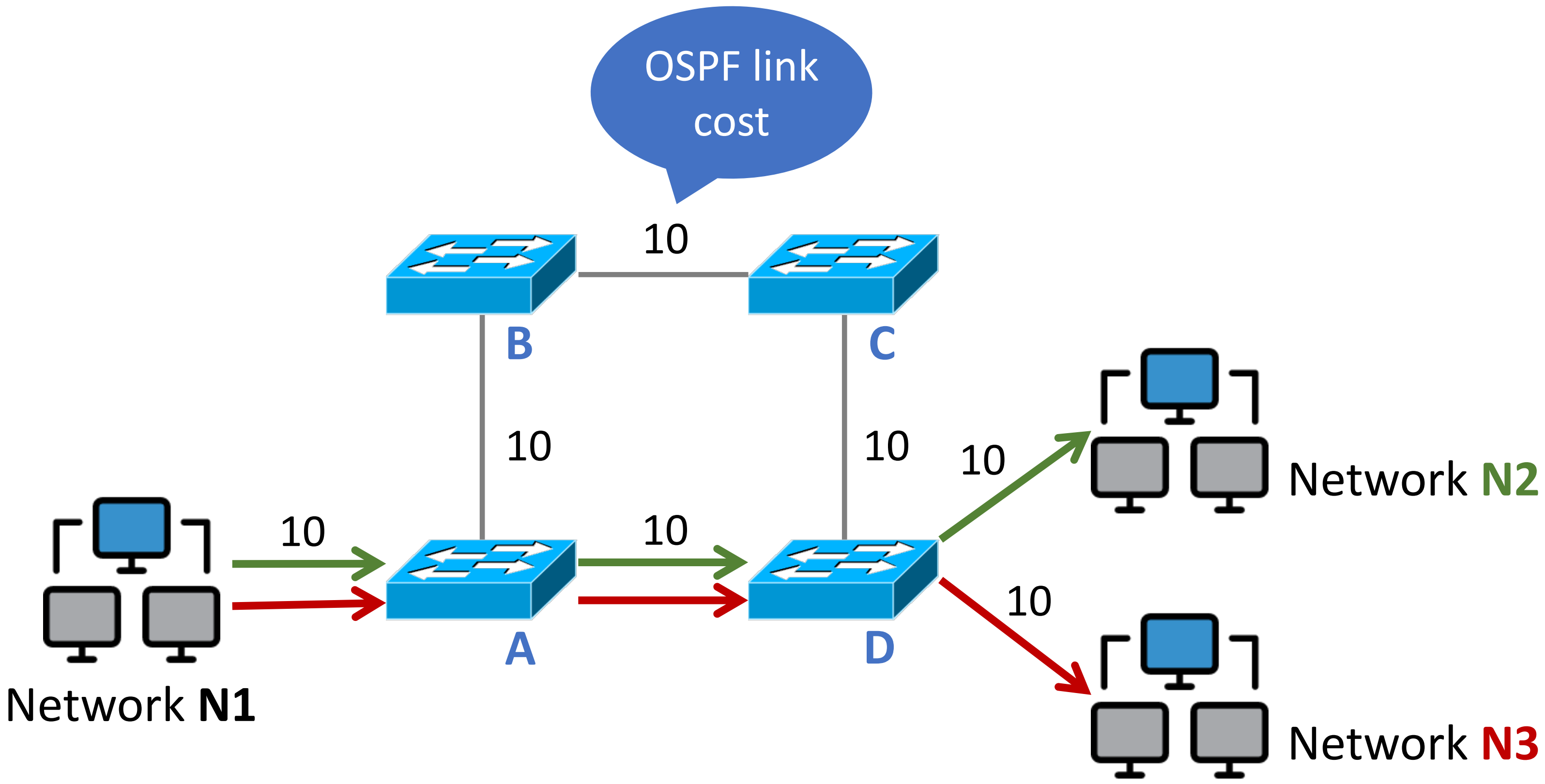
Example



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

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

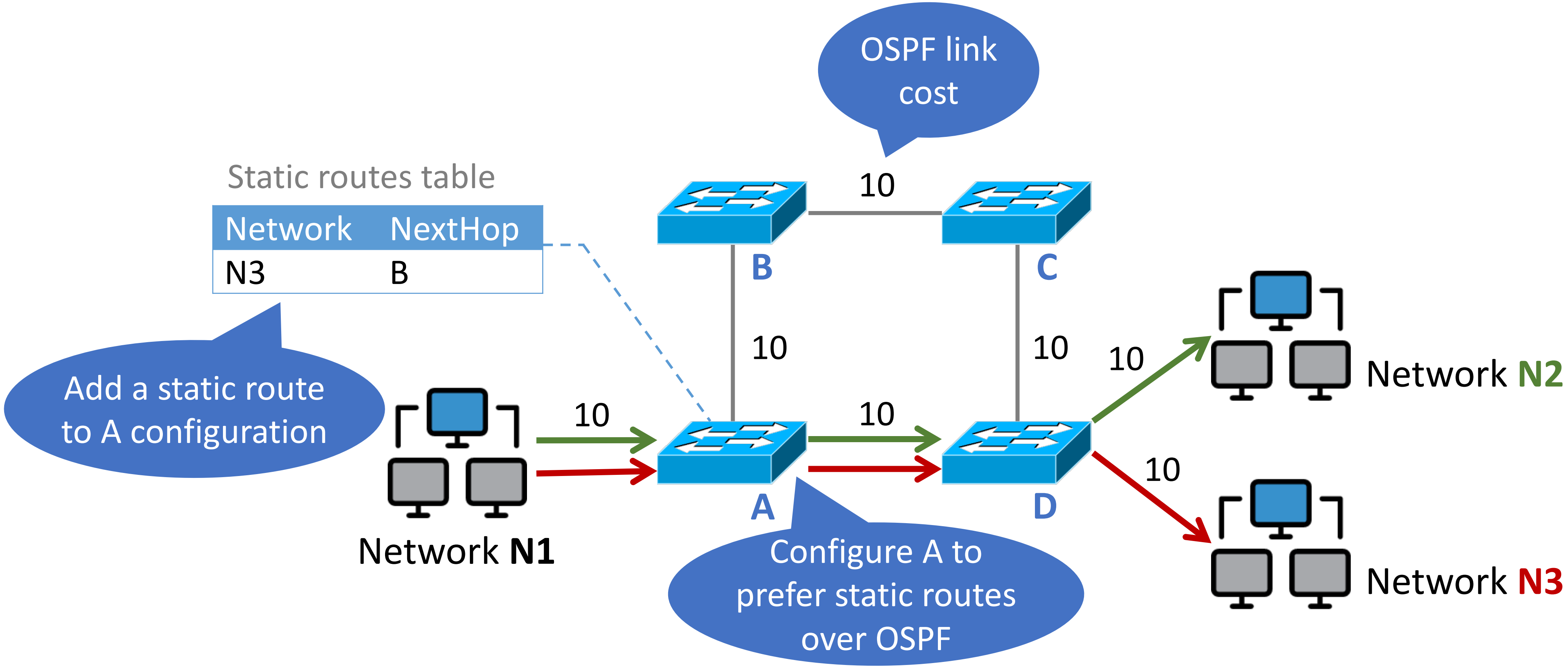
Example



✔ 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



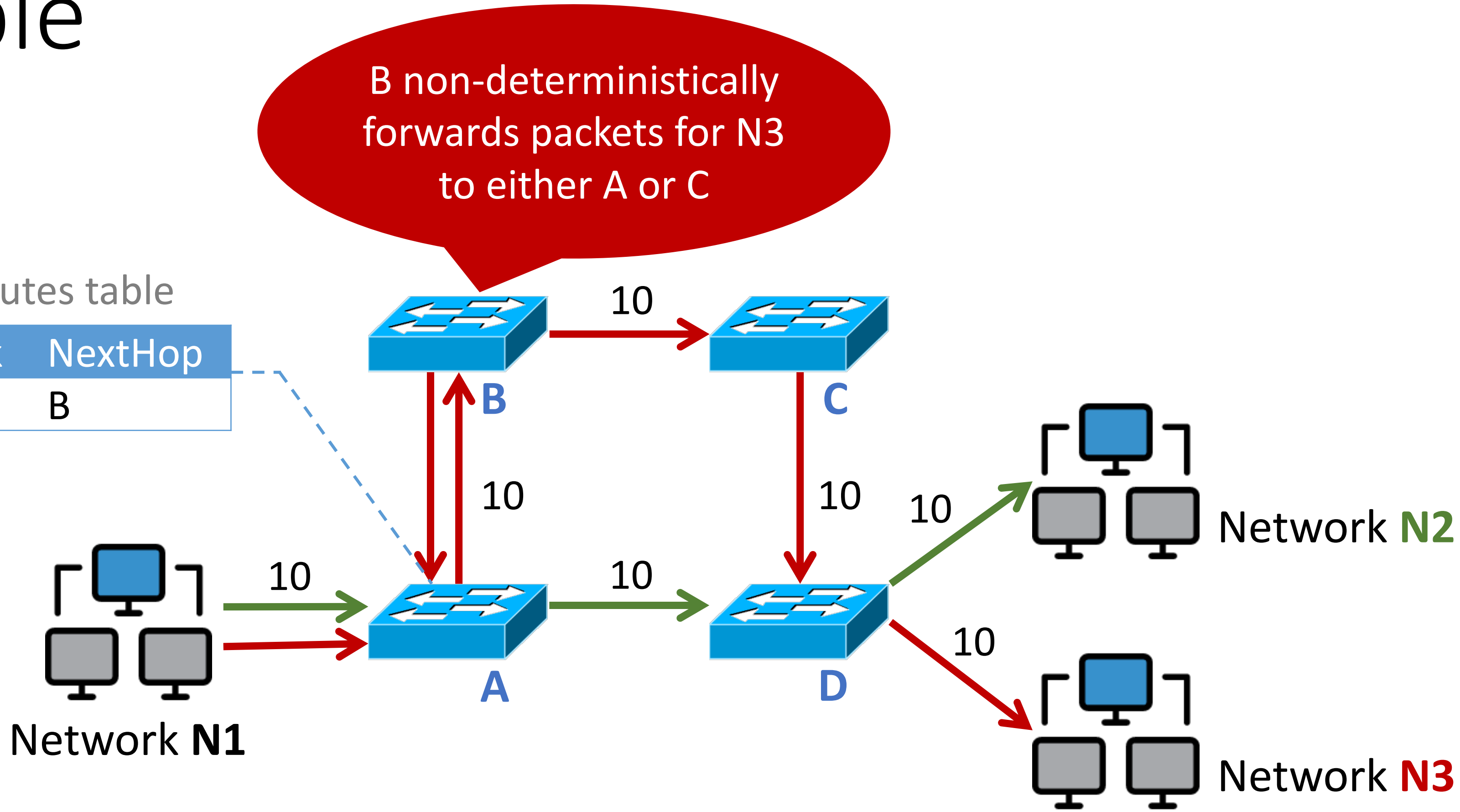
✔ R1: Packets from **N1** to **N2** must follow the path **A → D**

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

Example

Static routes table

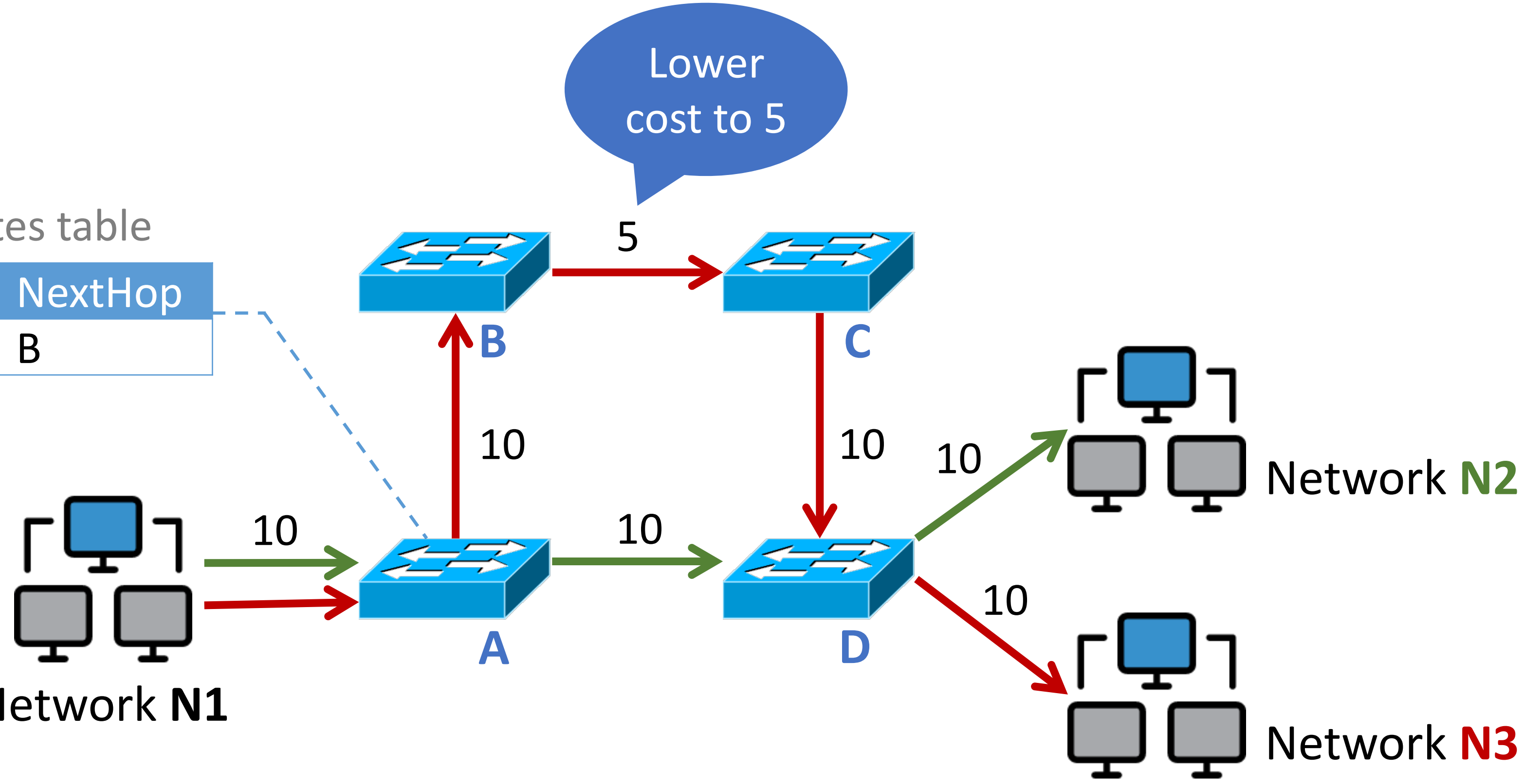
Network	NextHop
N3	B



✔ 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



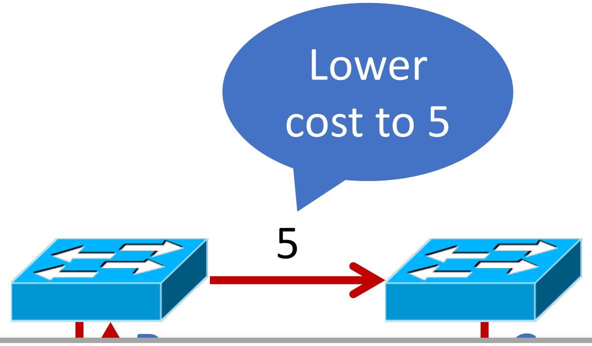
Network	NextHop
N3	B

- ✔ 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

Static routes table

Network	NextHop
N3	B



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**

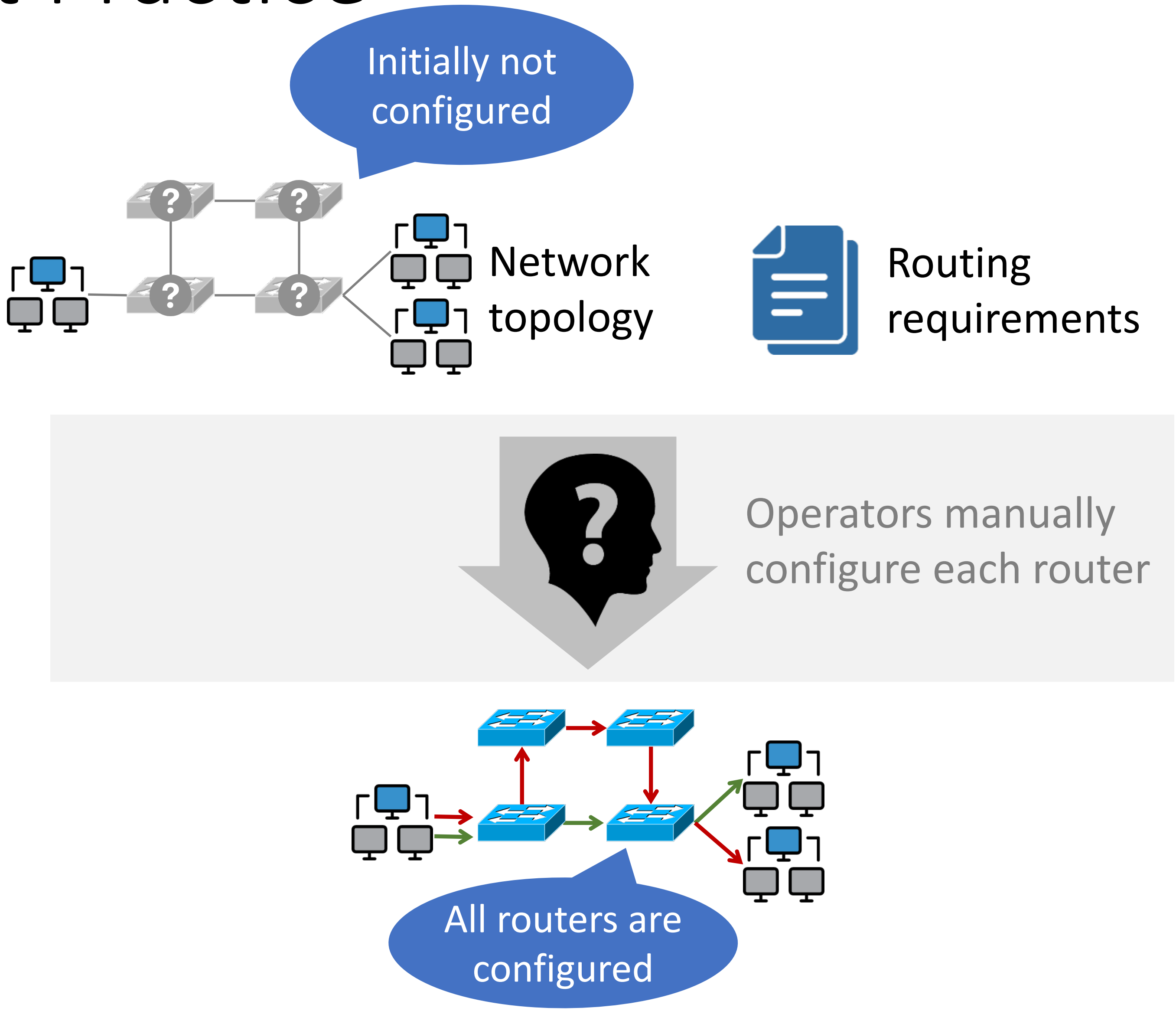
N2

N3

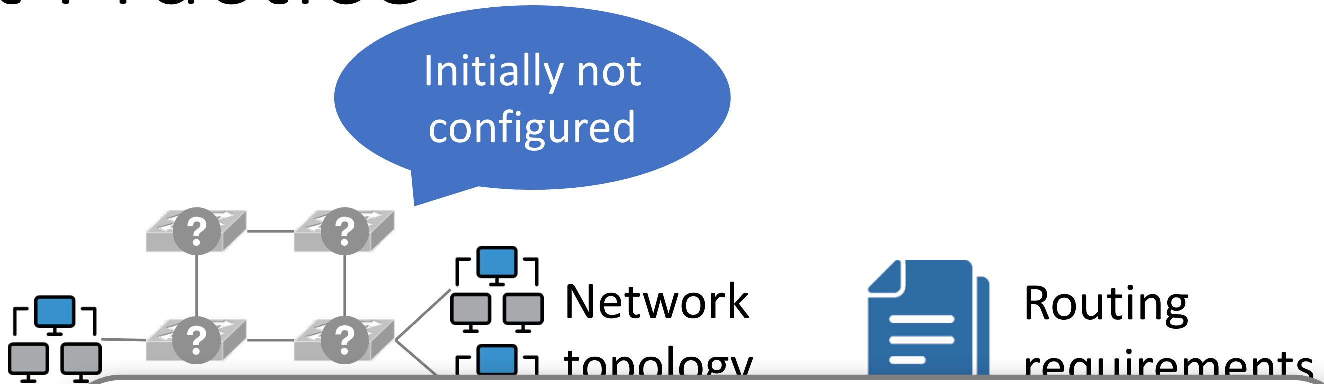
✔ R1: Packets from **N1** to **N2** must follow the path **A → D**

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

Current Practice

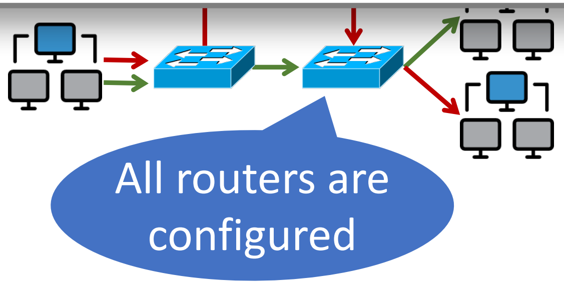


Current Practice

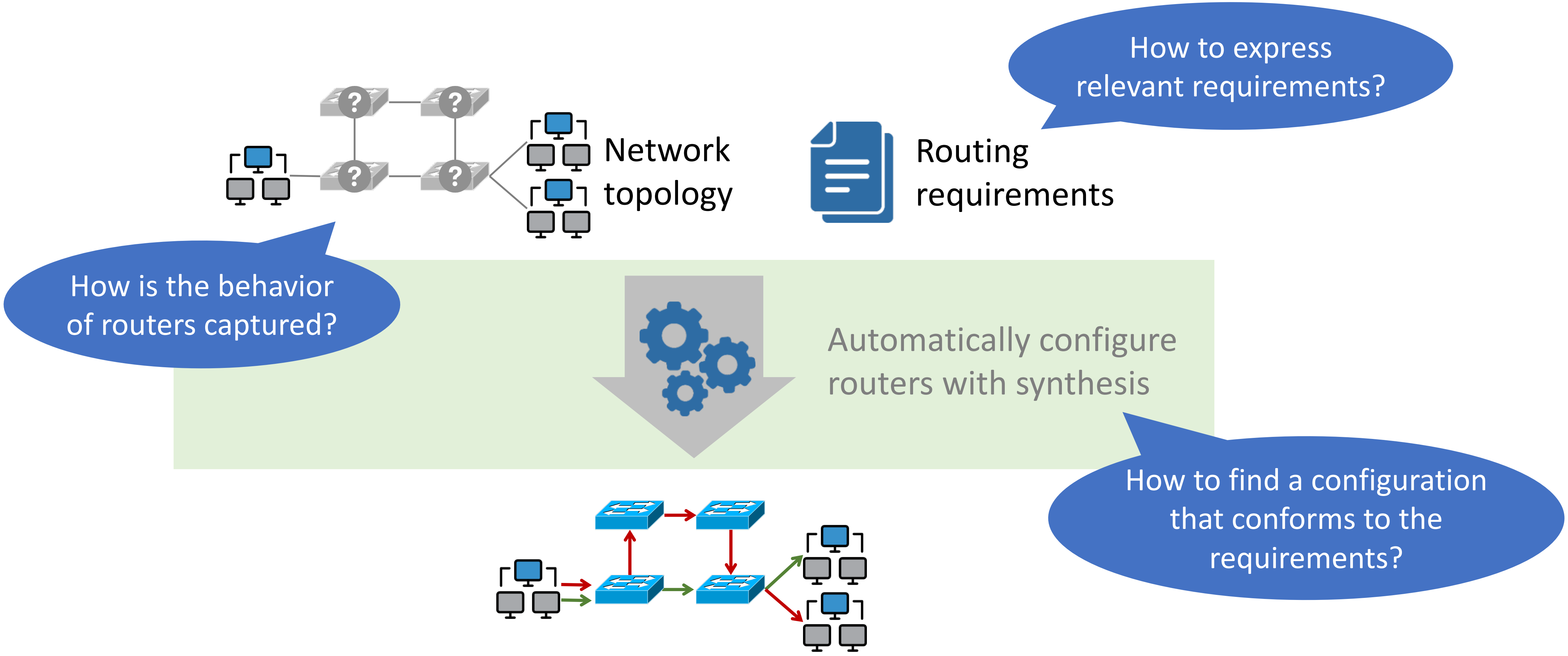


Problems and Challenges

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



Wanted: Programmable Networks with Synthesis



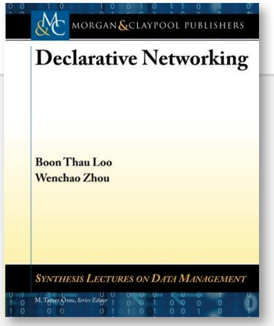
Programmable Networks with *Synthesis*: Dimensions

Deployment scenarios	Datacenter, Incremental, ISP, Enterprise
Routing protocols	**Deterministic**: Static routes, OSPF, BGP, MPLS; **Probabilistic**: ECMP, Gossip
Requirements	Paths, Reachability, Isolation, Waypointing, Failures, Congestion
Synthesis Techniques	Enumerative learning, Probabilistic, Constraint solving, Symbolic execution (**SyNET****), CEGIS

****SyNET**: http://synet.ethz.ch

Capturing Network Behavior

Key idea: Express routing protocols, along with their dependencies, in ***stratified Datalog***



Datalog Example

Input

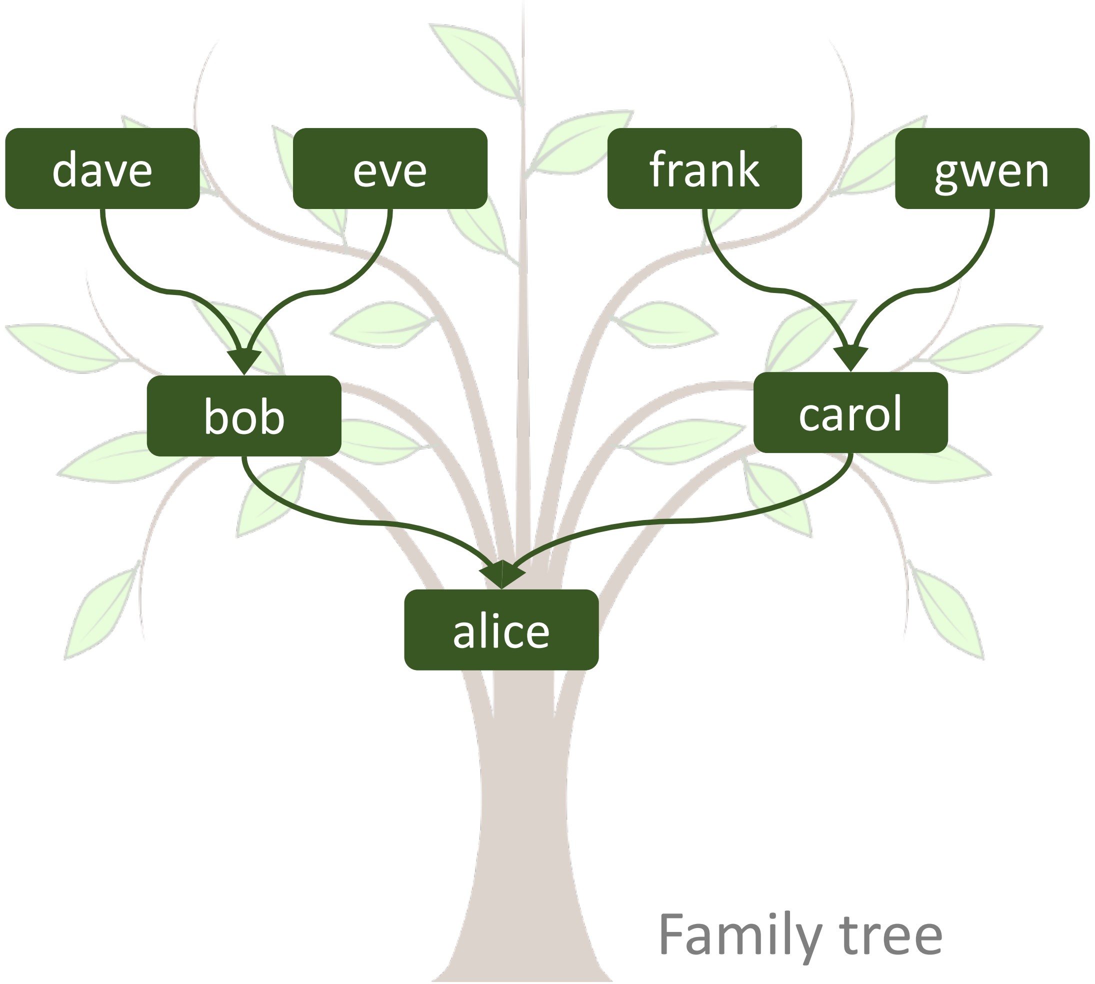
$parent(bob, alice)$
$parent(carol, alice)$
…

Program

$anc(X, Y) \leftarrow parent(X, Y)$
$anc(X, Y) \leftarrow parent(X, Z), anc(Z, Y)$

Query

$anc(dave, alice)?$



Family tree

How is this related to networks?

Datalog Syntax (1/2)

To define a Datalog program, we need:

Constants: $\mathcal{C} = \{alice, bob, carol, \dots\}$

Variables: $\mathcal{V} = \{X, Y, \dots\}$

Predicates: $\mathcal{P} = \{parent, anc, \dots\}$

The sets above can be used to construct the "atoms" of a Datalog program:

Ground atoms: $A_{\mathcal{P}(\mathcal{C})} = \{p(t_1, \dots, t_n) \mid p \in \mathcal{P}, \forall 0 \leq i \leq n.\, t_i \in \mathcal{C}\}$

Example: $parent(dave, alice)$

Atoms: $A_{\mathcal{P}(\mathcal{C},\mathcal{V})} = \{p(t_1, \dots, t_n) \mid p \in \mathcal{P}, \forall 0 \leq i \leq n.\ t_i \in \mathcal{C} \cup \mathcal{V}\}$

Example: $parent(X, Y)$

Datalog Syntax (1/2)

A ***Datalog program*** is a set of rules of the form

Positive literal

Negative literal

$$a \leftarrow l_1, \dots, l_n$$

where a is an atom and $l_1, \dots, l_n$ are literals of the form a or $\neg a$

Body

Head

$$anc(X, Y) \leftarrow parent(X, Z), anc(Z, Y)$$

A Datalog program is ***well-formed*** if for any rule in the program, all variables that appear in the head also appear in the body

Is this rule well-formed $anc(X, Y) \leftarrow parent(Y, Z)$?

Semantics of *Positive* Datalog Programs

Each Datalog program P is associated with a consequence operator T_P:

Interpretations: $\mathcal{I} = 2^{A_{\mathcal{P}}(\mathcal{C})}$ (an interpretation I is a set of ground atoms)

Substitutions: $\sigma: \mathcal{V} \rightarrow \mathcal{C}$ (a substitution maps variables to constants)

Consequence operator: $T_P: \mathcal{I} \rightarrow \mathcal{I}$

$$T_P(I) = \{\sigma(a) \mid a \leftarrow l_1, \dots, l_n \in P, \forall i \in [1 \dots n]. I \vdash \sigma(l_i)\}$$

where: $I \vdash l_i$ if $l_i = a$ and $a \in I$

$I \vdash l_i$ if $l_i = \neg a$ and $a \notin I$

A Datalog program P is ***positive*** if the negation operator does not appear in its rules

Is $\mathrm{T_P}$ monotone if P is a positive Datalog program?

The ***semantics*** of a positive Datalog program P is given by the least-fixed point of $\mathrm{T_P}$

How can we compute the least-fixed point of $\mathrm{T_P}$?

Example

Compute the least-fixed point of the following Datalog program:

$$
\begin{aligned}
&p(a) \leftarrow q(X) \\
&q(b) \leftarrow r(a), p(b) \\
&p(b) \leftarrow r(a)
\end{aligned}
$$

defined over the signature $\mathcal{C} = \{a, b, c\}, \mathcal{V} = \{X\}$, and $\mathcal{P} = \{p, q, r\}$

Datalog *Inputs*

We can split the set $\mathcal{P}$ of predicates into

1. ***input predicates***: predicates that do not appear in the head of rules, and
2. ***output predicates***: all remaining predicates

Which are the input/output predicates of this program?

$$anc(X, Y) \leftarrow parent(X, Y)$$
$$anc(X, Y) \leftarrow parent(X, Z), anc(Z, Y)$$

An ***input*** for a program P is an interpretation I that contains only atoms constructed using input predicates

The ***semantics*** of a positive Datalog program P given an ***input*** I for P is given by the smallest fixed point of $\mathrm{T_P}$ that contains I. Let's denote this by $[\![P]\!]_I$.

How can we compute $[\![P]\!]_I$?

Datalog and Negation

What should be the semantics of this program?

$$p(X) \leftarrow \neg q(X)$$
$$q(X) \leftarrow p(X)$$
$$p(X) \leftarrow r(X)$$

Problem: For a Datalog program P with negation, the consequence operator T_P is not guaranteed to be monotone!

What about the semantics of this program?

$$p(X) \leftarrow \neg q(X)$$
$$q(X) \leftarrow r(X)$$

This Datalog program is called ***"stratified"***

Semantics of Stratified Datalog

A ***Datalog*** program P is ***stratified*** if its rules can be partitioned into sets $P_1, \ldots, P_n$ called strata, such that:

1. for every predicate p, all rules with p in their heads are in one stratum P_i
2. if a predicate symbol p occurs in a positive literal in P_i , then all rules with p in their heads are in a stratum P_j with $j \leq i$
3. if a predicate symbol p occurs in a negative literal in P_i , then all rules with p in their heads are in a stratum P_j with $j < i$

What is an example of a Datalog program that is/is not stratified?

The semantics of a stratified Datalog program P, with strata $P_1, \ldots, P_n$, and an input I for P, is given by the fixed-point M_n where
$M_0 = I$, and $M_i = [\![P_i]\!]_{M_{i-1}}$, for $i \in [1, n]$.

Is M_n unique for any stratified Datalog program? What if we partition the rules into different partitions?

Encoding Network Behavior in stratified Datalog

Datalog (2/3 Graph Reachability)

Input

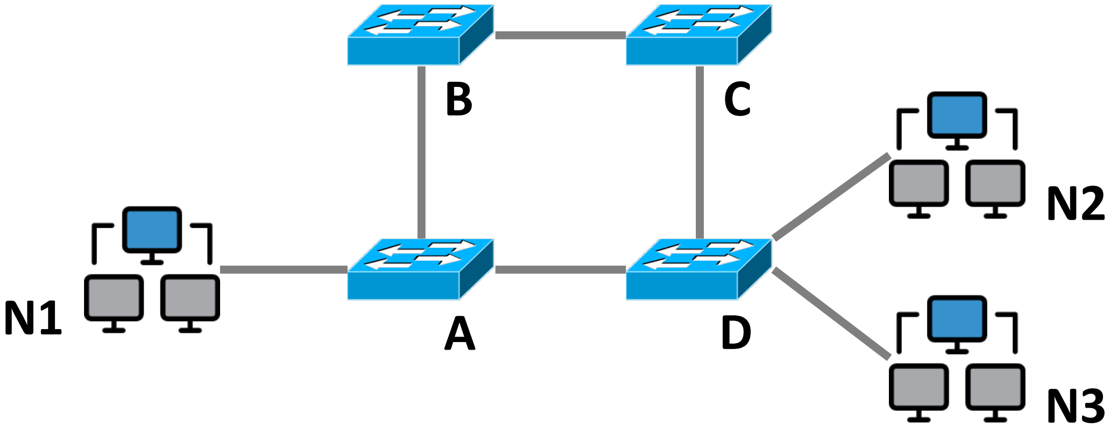
$link(n1, a)$
$link(a, b)$
…

Program

$path(X, Y) \leftarrow link(X, Y)$
$path(X, Y) \leftarrow link(X, Z), path(Z, Y)$

Query

$path(n1, n2)?$



Can we capture the network's forwarding plane?

Datalog (3/3 Shortest-path Routing)

Add OSPF cost to links

Input

$link(n_1, a, 10)$

Captures the router configuration

Program

$path(Router, Net, Net, Cost) \leftarrow$
$\quad link(Router, Net, Cost)$

$path(Router, Net, NextHop, C_1 + C_2) \leftarrow$
$\quad link(Router, NextHop, C_1),$
$\quad path(NextHop, Net, X, C_2)$

$sp(Router, Net, NextHop, min\langle C \rangle) \leftarrow$
$\quad path(Router, Net, NextHop, C)$

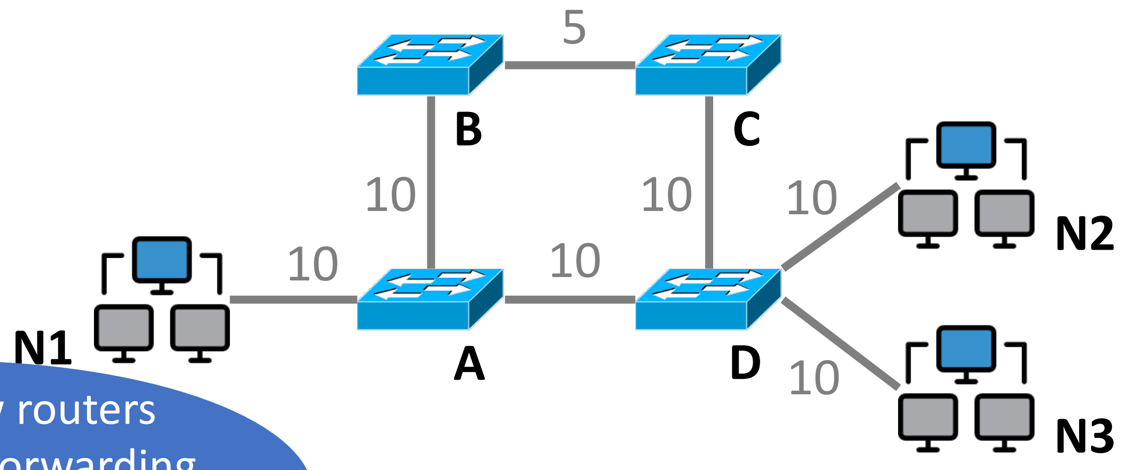
$fwd(Router, Net, NextHop) \leftarrow$
$\quad sp(Router, Net, NextHop, C)$

Captures how routers compute their forwarding entries using routing protocols

Query

$fwd(a, n_2, ?)$

Checks a property on the forwarding plane



Routing Requirements as Datalog Queries

Paths

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

$$fwd(r_1, tc, r_2) \land \cdots \land fwd(r_{n-1}, tc, r_n)$$

Routing Requirements as Datalog Queries

Paths

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

$fwd(r_1, tc, r_2) \wedge \cdots \wedge fwd(r_{n-1}, tc, r_n)$

Traffic isolation

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

$\forall R_1, R_2 . fwd(R_1, tc_1, R_2) \Rightarrow \neg fwd(R_1, tc_2, R_2)$

Routing Requirements as Datalog Queries

Paths

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

$fwd(r_1, tc, r_2) \land \cdots \land fwd(r_{n-1}, tc, r_n)$

Traffic isolation

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

$\forall R_1, R_2.\, fwd(R_1, tc_1, R_2) \Rightarrow \neg fwd(R_1, tc_2, R_2)$

Reachability

Packets for traffic class tc can reach router r_2 from router r_1

$reach(r_1, tc, r_2)$

Routing Requirements as Datalog Queries

Paths

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

$$fwd(r_1, tc, r_2) \land \cdots \land fwd(r_{n-1}, tc, r_n)$$

Traffic isolation

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

$$\forall R_1, R_2.\, fwd(R_1, tc_1, R_2) \Rightarrow \neg fwd(R_1, tc_2, R_2)$$

Reachability

Packets for traffic class tc can reach router r_2 from router r_1

$$reach(r_1, tc, r_2)$$

Loop-freeness

The forwarding plane has no loops

$$\forall TC, R.\, \neg reach(R, TC, R)$$

Analysis of Network Configurations in Datalog

Analysis of Network Configurations in Datalog

Network-wide configuration $\boldsymbol{C}$
(protocol configurations for routers)

Datalog ***input*** $\boldsymbol{I}$

Network specification $\boldsymbol{N}$
(OSPF, BGP, MPLS, …)

Datalog ***program*** $\boldsymbol{P}$

Routing requirements $\boldsymbol{R}$
(isolation, reachability, reliability)

Datalog ***query*** $\boldsymbol{Q}$

Analysis question:
Does the network $\boldsymbol{N}$ *configured with* $\boldsymbol{C}$ *satisfy the requirements* $\boldsymbol{R}$*?*

Query entailment:
Does $\boldsymbol{P}, \boldsymbol{I} \vDash \boldsymbol{Q}$ *hold?*

Analysis of Network Configurations in Datalog

Network-wide configuration C
(protocol configurations for routers)

Datalog *input* $\boldsymbol{I}$

Network specification N
(OSPF, BGP, MPLS, …)

Datalog *program* $\boldsymbol{P}$

Routing requireme
(isolation, reachability,

Theorem: Query entailment in Datalog is in **PTIME**

Analysis question:
Does the network $\boldsymbol{N}$ *configured with* $\boldsymbol{C}$ *satisfy the requirements* $\boldsymbol{R}$*?*

Query entailment:
Does $\boldsymbol{P}, \boldsymbol{I} \vDash \boldsymbol{Q}$ *hold?*

Network-wide Configuration Synthesis

Network-wide Configuration Synthesis

Network specification $\boldsymbol{N}$
(OSPF, BGP, MPLS, …)

Datalog ***program*** $\boldsymbol{P}$

Routing requirements $\boldsymbol{R}$
(isolation, reachability, reliability)

Datalog ***query*** $\boldsymbol{Q}$

Synthesis problem:
Find a configuration $\boldsymbol{C}$ *such that* $\boldsymbol{N}$ *configured with* $\boldsymbol{C}$ *satisfies* $\boldsymbol{R}$

(Input) Synthesis problem:
Find an input $\boldsymbol{I}$ *such that* $\boldsymbol{P}, \boldsymbol{I} \vDash \boldsymbol{Q}$

Datalog ***input*** $\boldsymbol{I}$

Network-wide configuration $\boldsymbol{C}$
(protocol configurations for routers)

Network-wide Configuration Synthesis

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

Datalog ***program*** P

Routing requirements R
(isolation, reachability, reliability)

Datalog ***query*** Q

Synthesis problem
Find a configuratio
configured with C s

Problems:

- No input synthesis tools for Datalog
- Problem is undecidable

problem:
ch that $P, I \vDash Q$

Network-wide configuration C
(protocol configurations for routers)

Datalog ***input*** I

Input Synthesis for Datalog

Key idea: Reduce to solving SMT constraints

Input Synthesis for Positive Datalog (First Attempt)

Datalog program $\boldsymbol{P}$

$path(X,Y) \leftarrow link(X,Y)$
$path(X,Y) \leftarrow link(X,Z), path(Z,Y)$

Datalog query $\boldsymbol{Q}$

$path(a,c) \ \wedge \ \neg link(a,c)$

Generate SMT constraints

$\forall X,Y.\, path(X,Y) \Leftarrow link(X,Y)$
$\forall X,Y.\, path(X,Y) \Leftarrow \exists Z.\, link(X,Z) \wedge path(Z,Y)$
$path(a,c) \ \wedge \ \neg link(a,c)$

Easy encoding… unfortunately it does not work

SMT Constraints $\boldsymbol{\psi}$

Solve SMT constraints

$path(a,c)$

Model $\boldsymbol{M} \vDash \boldsymbol{\psi}$

Derive input (by project on input predicates)

Datalog input $\boldsymbol{I}$

$\emptyset$

Unfortunately, we get $P, I \nvDash Q$

Input Synthesis for Positive Datalog (1/2)

$path(a,c)$ is a ***positive*** query

Datalog program $\boldsymbol{P}$

$path(X,Y) \leftarrow link(X,Y)$
$path(X,Y) \leftarrow link(X,Z), path(Z,Y)$

Datalog query $\boldsymbol{Q}$

$path(a,c) \ \land \ \neg link(a,c)$

Generate SMT constraints

Bounded unrolling for positive queries

$\forall X,Y.\, path_1(X,Y) \Leftrightarrow link(X,Y)$
$\forall X,Y.\, path_2(X,Y) \Leftrightarrow \big(link(X,Y) \lor (\exists Z.\, (link(X,Z) \land path_1(Z,Y))\big)$
$path_2(a,c) \ \land \ \neg link(a,c)$

SMT Constraints $\boldsymbol{\psi}$

Solve SMT constraints

$link(a,b), link(b,c),$
$path_1(a,b), path_1(b,c)$
$path_2(a,b), path_2(b,c), path_2(a,c)$

Model $\boldsymbol{M} \vDash \boldsymbol{\psi}$

Derive input

$link(a,b), link(b,c)$

Datalog input $\boldsymbol{I}$

Input Synthesis for Positive Datalog (2/2)

$path(a, c)$ is a ***negative*** query

Datalog program $\boldsymbol{P}$

$path(X, Y) \leftarrow link(X, Y)$
$path(X, Y) \leftarrow link(X, Z), path(Z, Y)$

Datalog query $\boldsymbol{Q}$

$\neg path(a, c) \ \wedge \ \neg link(a, c)$

Generate SMT constraints

No unrolling for negative queries

$\forall X, Y.\, path(X, Y) \Leftarrow link(X, Y)$
$\forall X, Y.\, path(X, Y) \Leftarrow \exists Z.\, link(X, Z) \wedge path(Z, Y)$
$\neg path(a, c) \ \wedge \ \neg link(a, c)$

SMT constraints $\boldsymbol{\psi}$

Solve SMT constraints

$\{\}$

Model $\boldsymbol{M} \vDash \boldsymbol{\psi}$

Derive input

$\{\}$

Datalog input $\boldsymbol{I}$

Input Synthesis for Positive Datalog (2/2)

$path(a, c)$ is a ***negative*** query

Datalog program $\boldsymbol{P}$

$path(X, Y) \leftarrow link(X, Y)$
$path(X, Y) \leftarrow link(X, Z), path(Z, Y)$

Datalog query $\boldsymbol{Q}$

$\neg path(a, c) \ \land \ \neg link(a, c)$

No unrolling for negative queries

Generate SMT constraints

$\forall X, Y. path(X, Y) \Leftarrow link(X, Y)$
$\forall X, Y. path(X, Y) \Leftarrow \exists Z. link(X, Z) \land path(Z, Y)$
$\neg path(a, c) \ \land \ \neg link(a, c)$

SMT Constraints $\boldsymbol{\psi}$

Solve SMT constraints

$\{\}$

Model $\boldsymbol{M} \vDash \boldsymbol{\psi}$

Derive input

$\{\}$

Datalog input $\boldsymbol{I}$

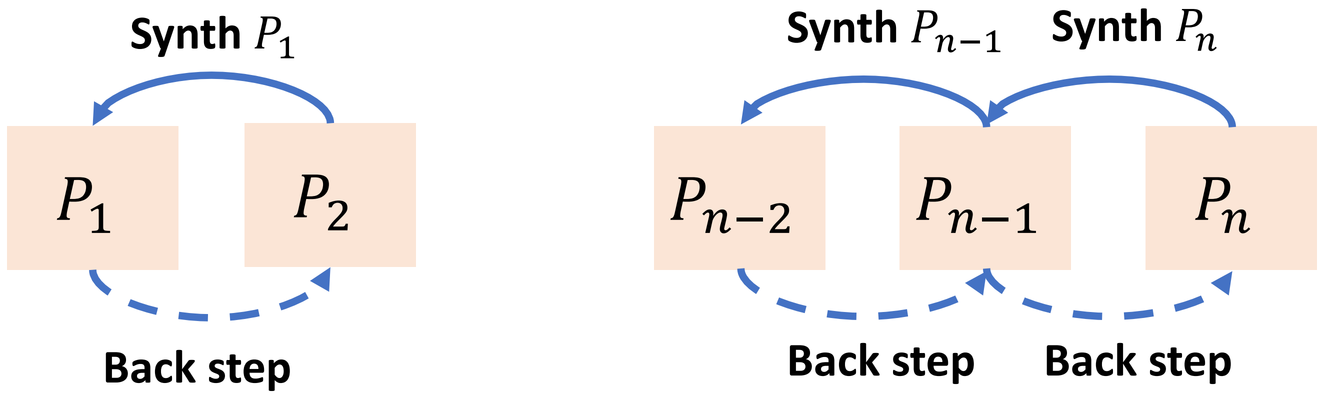
Summary:

- Unroll rules for positive queries, do not unrolling rules for negative queries
- Combine both kinds of constraints for constraints that contain both positive/negative queries.

Input Synthesis for *Stratified* Datalog

Suppose we have a program P with strata $P_1, \dots, P_n$, and a query Q.

High-level idea:

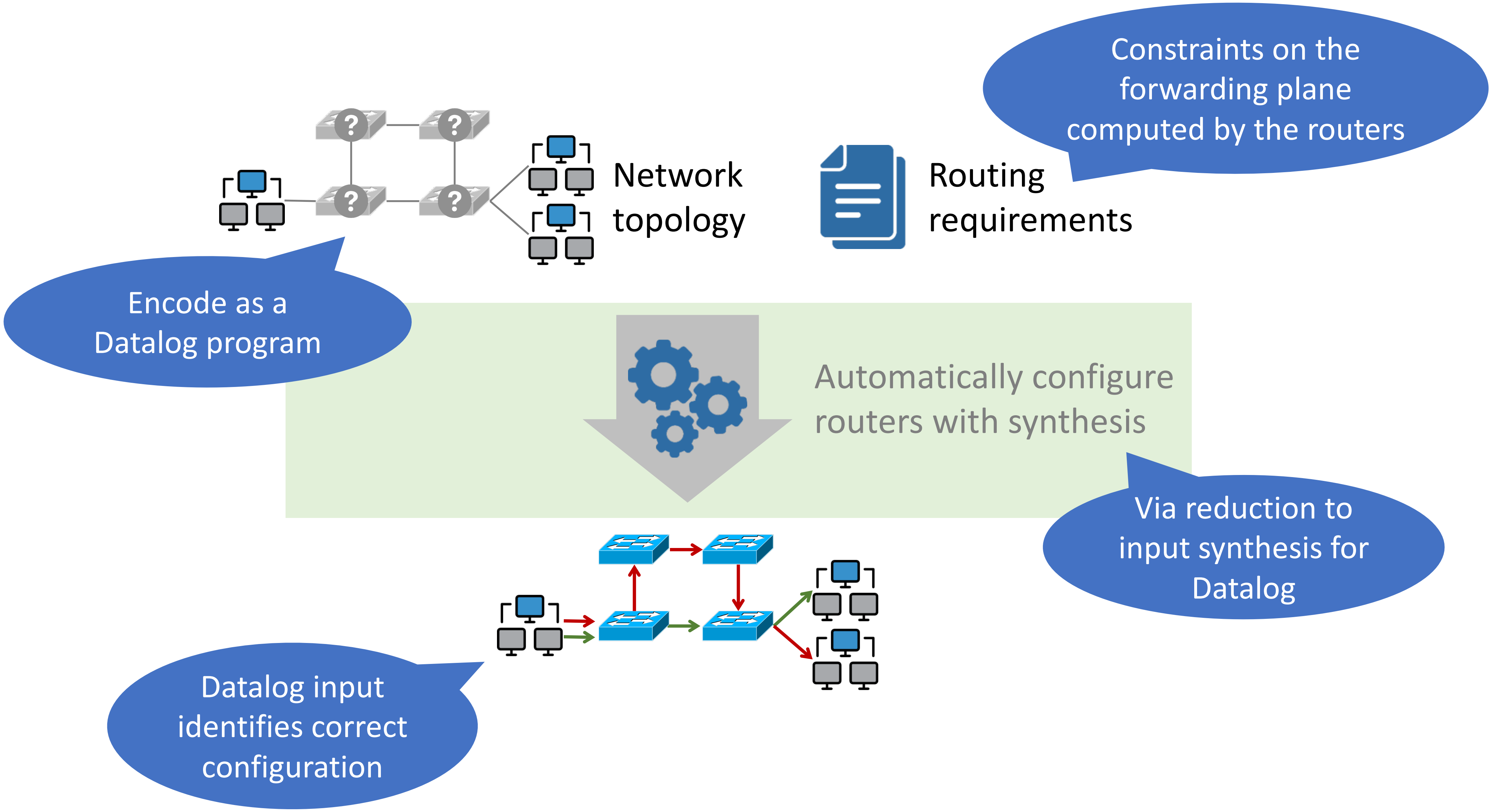


Synth P_n: Compute input I_n for stratum P_n such that $[\![P_n]\!]_{I_n}$ satisfies Q.

Synth $P_{n-1} \dots P_1$: Compute input I_i for stratum P_i such that $[\![P_i]\!]_{I_i}$ produces the input I_{i+1} synthesized by the previous step

Back step: Backtrack to step **Synth** P_i if the step **Synth** P_{i-1} returns **unsat**

Network-wide Configuration Synthesis



Software Synthesis @ SRL

SOFTWARE RELIABILITY LAB

http://www.srl.inf.ethz.ch/

Develop new synthesis techniques to solve practical system challenges

Intent

Desired properties

Software Synthesis

Oracle-guided

CEGIS

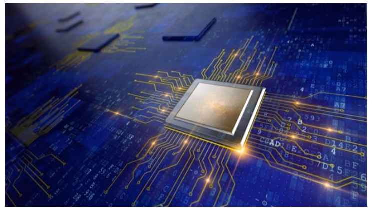
Constraint-based

Probabilistic

Big Code

Computer networks

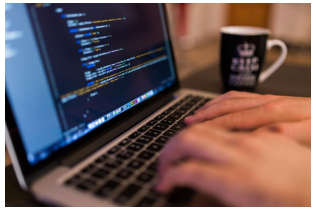
Security and privacy



Modern architectures

Datacenters

Data science



End-user programming

Application domains

Implementation

Implementation

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

- Written in **Python** ($\approx 4K$ lines of code)
- Protocols encoded in **stratified Datalog** (≈ 100 rules)
- Uses the **Z3** constraint solver. Relies on linear integer arithmetic theories (**LIA**).
- Outputs **CISCO** configurations
- Supports **BGP**, **OSPF**, and **static routes**

Network-specific optimizations

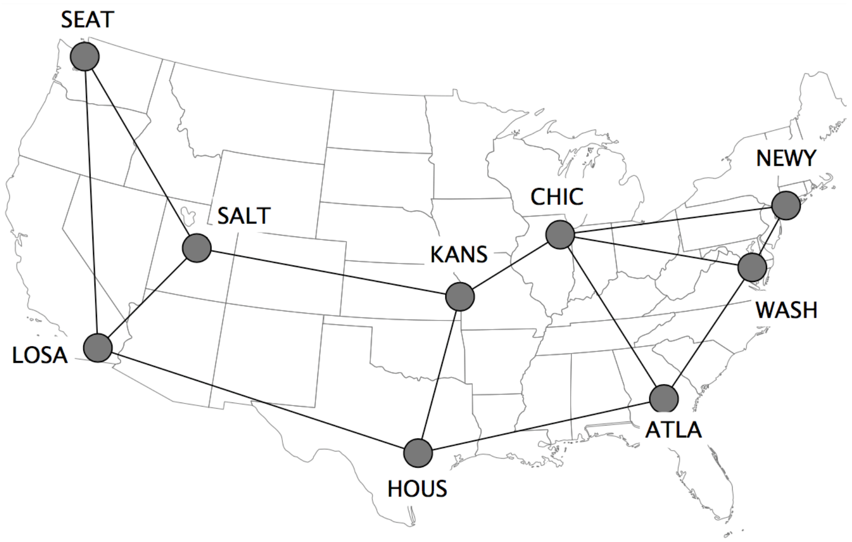
- Partial evaluator for Datalog
- Protocol-specific constraints

```
! 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"
!
```

Sample **CISCO** configuration output by SyNET

Experiments

Experiment



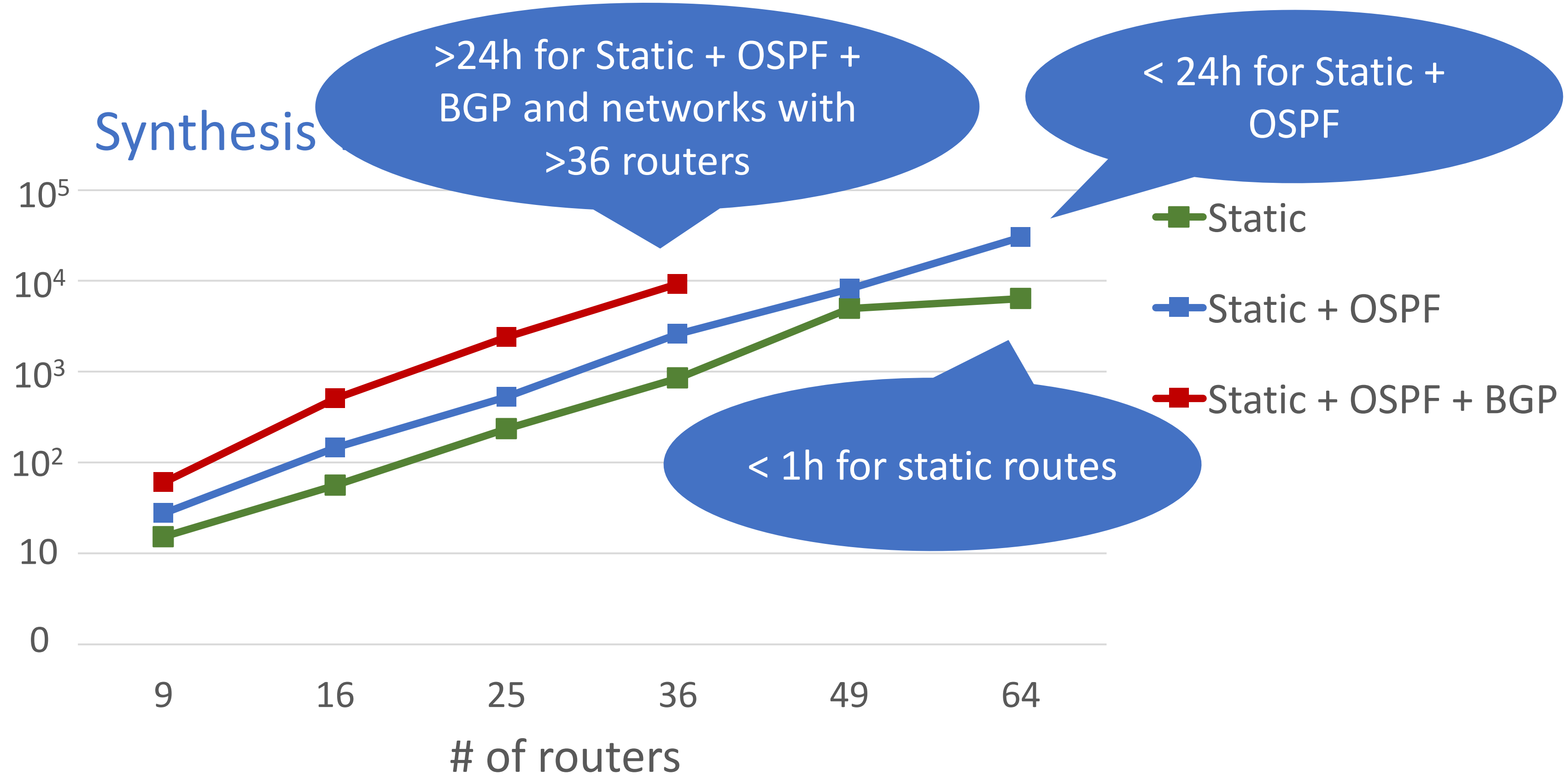
US-based network connecting major universities and research institutes

Protocols / # Traffic classes	1 class	5 classes	10 classes
Static	1.3s	2.0s	4.0s
Static + OSPF	9.0s	21.3s	49.3s
Static + OSPF + BGP	13.3s	22.7s	1m19.7s

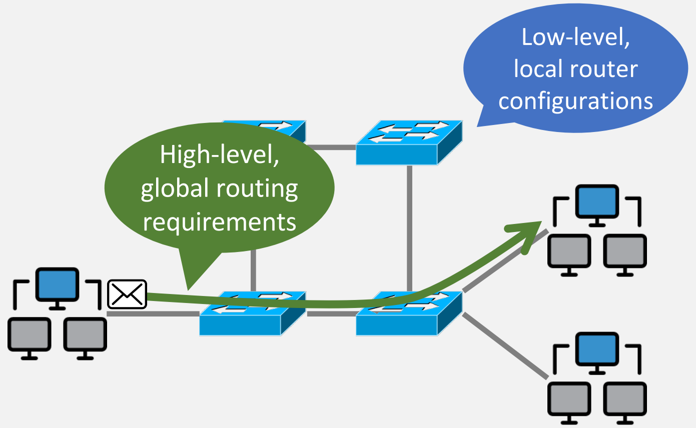
Synthesis Times

Scalability Experiment

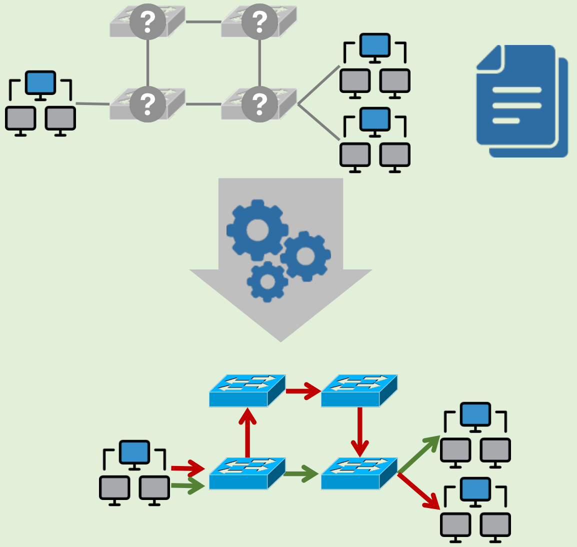
- Grid topologies with up to 64 routers
- Requirements for 10 traffic classes



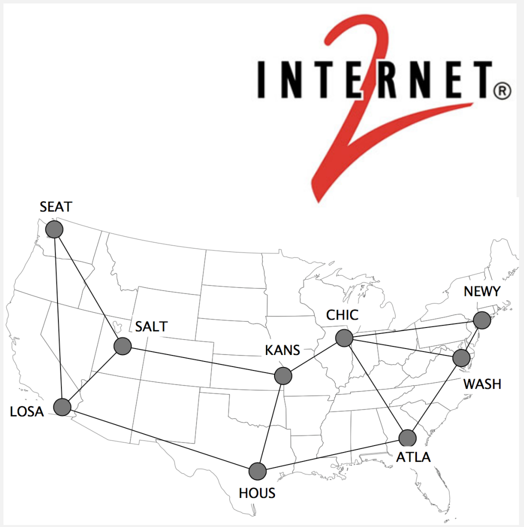
Summary: Programmable Networks with Synthesis



Global requirements vs local configurations



Network-wide configuration synthesis



Approach scales to realistic problems

For more details read this paper: https://arxiv.org/abs/1611.02537