Microsoft Research

Microsoft Research

Microsoft

Checking Beliefs in Dynamic Networks

Nuno Lopes, Nikolaj Bjørner, Patrice Godefroid, Karthick Jayaraman, George Varghese

Networks

- Business critical and complex

  ↳ Expensive bugs

- Fast protocol deployment in datacenters

  ↳ Frequent protocol changes

- A lot of legacy to maintain

  ↳ Operators don't have the full picture

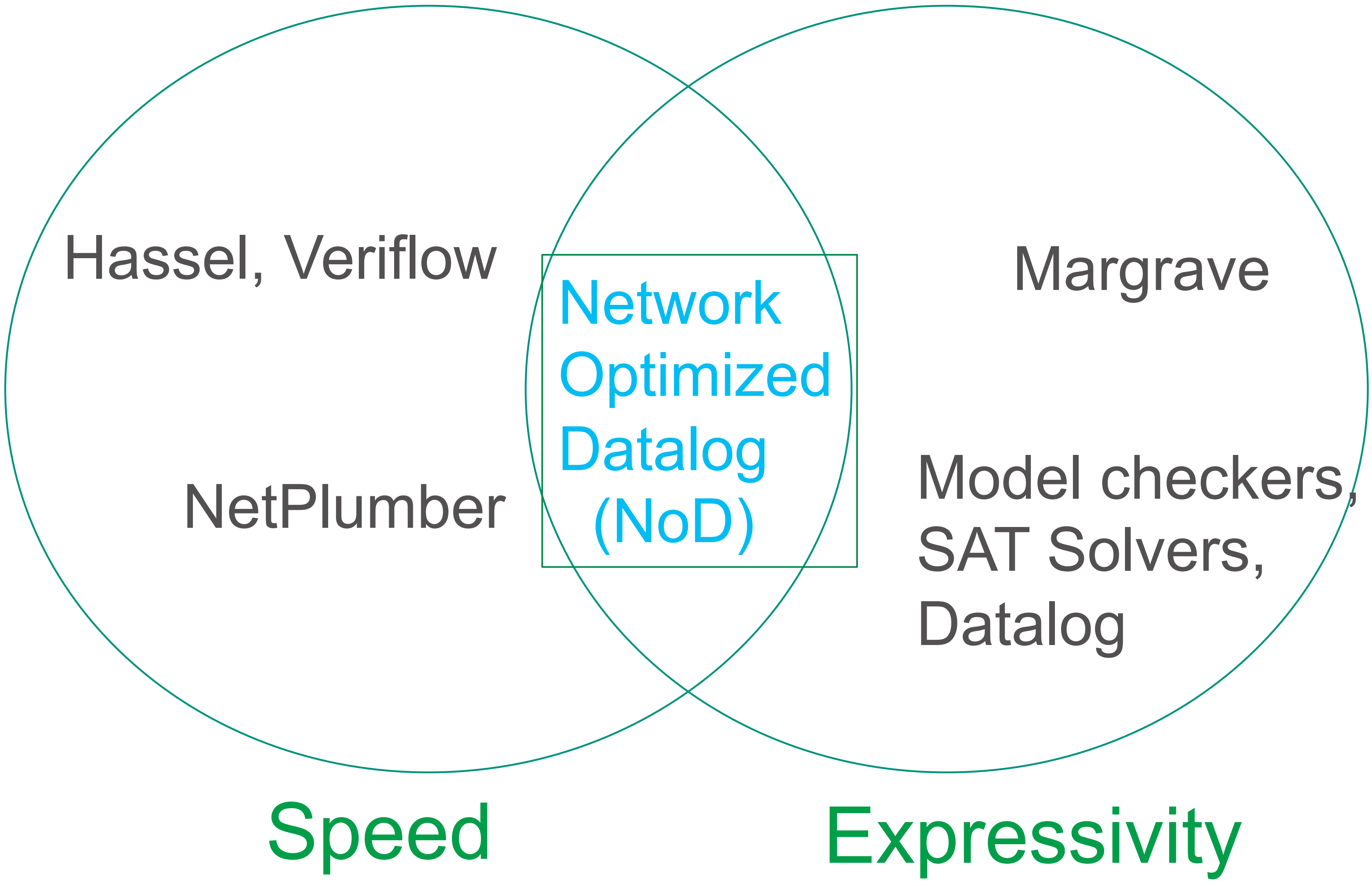
Network Verification to the Rescue

- Identify bugs
- Identify misbeliefs
- Increase confidence

This Paper in Context

- Implementation bugs (PIC)
  - E.g., protocol conformance
- Routing configuration errors (Batfish)
  - E.g., router filter error
- Dataplane configuration errors (NoD)
  - E.g., customer VMs can access controller

Existing Work versus Ours



Why Expressiveness Matters

- Network level
  - Enables modeling dynamic network behaviors such as new packet headers, new forwarding behaviors, failures, e.g.,
    - A P4 router adds a new header or a new forwarding behavior
- Specification level
  - Enables higher-level verification queries, e.g.
    - Customer VMs cannot reach fabric controller
    - All backup routers are equivalent

Example Beliefs

Policy Template	Example
Protection Sets	Customer VMs cannot access controllers
Reachable Sets	Customer VMs can access other VMs
Consistency	ECMP/Backup routes should have identical reachability
Middlebox	Forward path connections through middlebox should reverse
Locality	Packets between two hosts in the same cluster should stay within the cluster

Solution

Network-Optimized Datalog (NoD)

- Datalog for the specification of:
  - Data-plane/control-plane
  - Verification properties
- Tool for efficient verification
  - Available in open-source SMT solver Z3

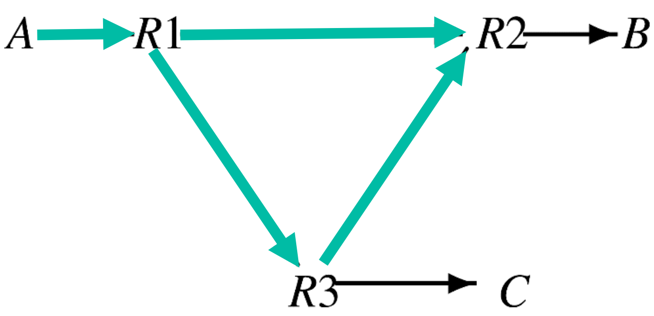
Why Datalog?

- Good expressiveness/efficiency tradeoff
- Supports packet rewriting, load balancing
- Provides all (symbolic) solutions for "free"
  - Unlike SAT solvers or model checkers

Modeling Networks using Datalog

- Each matching rule in the FIB and each ACL rule becomes a Datalog rule
- State is set of packets at each router
- Packets start at sources; Datalog runs to fixed-point -> packets at destinations

Networks as Datalog Programs



in	*dst*	*src*	*rewrite*	*out*
$R1$	$10\star$	$01\star$		$R2$
$R1$	$1\star\star$	$\star\star\star$		$R3$
$R2$	$10\star$	$\star\star\star$		B
$R3$	$\star\star\star$	$1\star\star$		C
$R3$	$1\star\star$	$\star\star\star$	$dst[1] := 0$	$R2$

Dataplane

$$R1(dst, src) \quad :- \quad A(dst, src)$$

Guards

$$G_{12} \quad := \quad dst = 10\star \wedge src = 01\star$$

$$G_{13} \quad := \quad \neg G_{12} \wedge dst = 1\star\star$$

$$G_{2B} \quad := \quad dst = 10\star$$

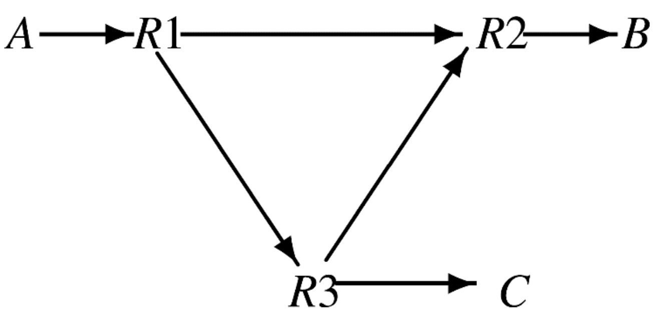
$$G_{3C} \quad := \quad src = 1\star\star$$

$$G_{32} \quad := \quad \neg G_{3C} \wedge dst = 1\star\star$$

$$Id \quad := \quad src' = src \wedge dst' = dst$$

$$Set0 \quad := \quad src' = src \wedge dst' = dst[2]\ 0\ dst[0]$$

Example of Reachability



in	dst	src	$rewrite$	out
$R1$	$10\star$	$01\star$		$R2$
$R1$	$1\star\star$	$\star\star\star$		$R3$
$R2$	$10\star$	$\star\star\star$		B
$R3$	$\star\star\star$	$1\star\star$		C
$R3$	$1\star\star$	$\star\star\star$	$dst[1] := 0$	$R2$

Compute all packets sent by A that reach B

$$
\begin{array}{lcl}
 & & A(dst, src) \\
R1(dst, src) & :- & A(dst, src) \\
R2(dst', src') & :- & R1(dst, src) \wedge G_{12} \wedge Id \\
R2(dst', src') & :- & R3(dst, src) \wedge G_{32} \wedge Set0 \\
R3(dst', src') & :- & R1(dst, src) \wedge G_{13} \wedge Id \\
B(dst', src') & :- & R2(dst, src) \wedge G_{2B} \wedge Id \\
C(dst', src') & :- & R3(dst, src) \wedge G_{3C} \wedge Id \\
 & ? & B(dst, src)
\end{array}
$$

Result:

$$
10\star01\star \cup (10\star\star\star\star \setminus (10\star01\star \cup \star\star\star1\star\star))
$$

$$
= 10\star0\star\star
$$

So what's wrong with Datalog?

- Out-of-the-box implementations are slow
  - They work with a packet a time
- Our contributions:
  - Symbolic representation (dealing with sets of packets)
  - Efficient propagation of packets across routers

Symbolic Representation

- Packets represented as Difference of Cubes [NSDI'12]
- Generalized to support negation, useful e.g. to check consistency across backup routers

$$\bigcup_i \left(v_i \setminus \bigcup_j v_j \right)$$

$v{\downarrow}i$ X$\downarrow j$ ernary bit-vectors

Examples: $10{\star}01{\star} \cup (10{\star}{\star}{\star}{\star} \setminus (10{\star}01{\star} \cup {\star}{\star}{\star}1{\star}{\star}))$

$10{\star}0{\star}{\star}$

Fuse Internal Datalog Operators

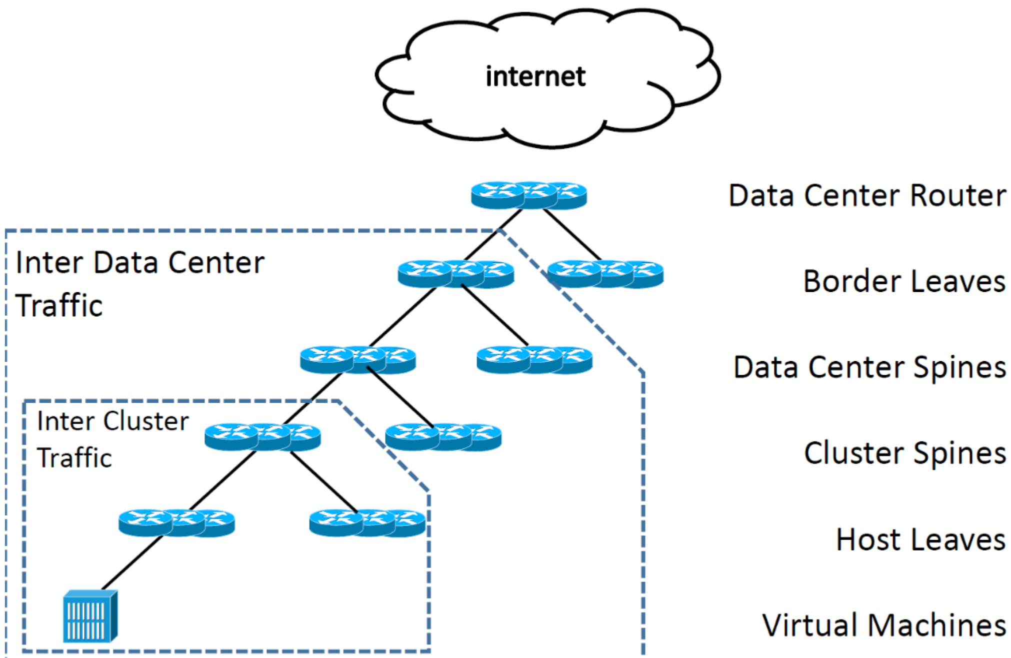
Evaluation

Evaluation questions

- Do beliefs help?
- How hard is it to add a new forwarding protocol?
- How does NoD performs compared with existing verification tools?
- Is this useful in practice?

Beyond Reachability: Locality

- Found multiple violations of traffic locality



Query	Cluster 1	Cluster 2	Cluster 3
C2C	12 (2)	13 (2)	11 (2)
B2DSP	11 (2)	11 (2)	11 (2)
B$\overline{2}$DSP	3 (1)	4 (1)	4 (1)
B2CSP	11 (2)	11 (2)	11 (2)
B$\overline{2}$CSP	11 (2)	12 (2)	11 (2)

Verification time in seconds

Checking Operators' Beliefs

- Operators cannot specify reachability at VM level for millions of VMs
- They have "beliefs" of which sets of stations can reach others
- Found exceptions to operator's beliefs
  - Customer VMs cannot access fabric controllers
- Process of belief refinement helps elicit specifications

Dynamism Example

- Experimental MPLS-like backbone with custom forwarding
- Took a few hours to model without any tool change
- Loop detection in < 1 second
- Identified 56 flows as black holes in 5 seconds

Performance Comparison

	Model Checker	SMT All Solutions	NoD	HSA
Stanford Unreach	12.2	0.1	**2.1**	0.1
Stanford Reachable	13.7	1121	**5.9**	0.9
Stanford Loop	11.7	290	**3.9**	0.2
Cloud	Time out	Time out	**15.7**	-
Cloud 2	8.5	Time out	**4.8**	-

Run time in seconds

Network Verification in Production

- Simplified version of NoD: SecGuru
  - Local checks on each router
- Deployed in Azure
- Finds ~1 problem per day

- Reduced legacy corporate ACL from 3,000 to 1,000 rules without outages

Conclusion

- NoD is expressive; takes as input:
  - Protocol specification -> Dynamism
  - Verification properties -> Beliefs
- More expressive than previous network verification tools, while competitive in speed
- Network operators' beliefs are fragile
- Code and benchmarks available on-line!

Microsoft Research

Microsoft