CURSOR: Configuration Update Synthesis Using Order Rules

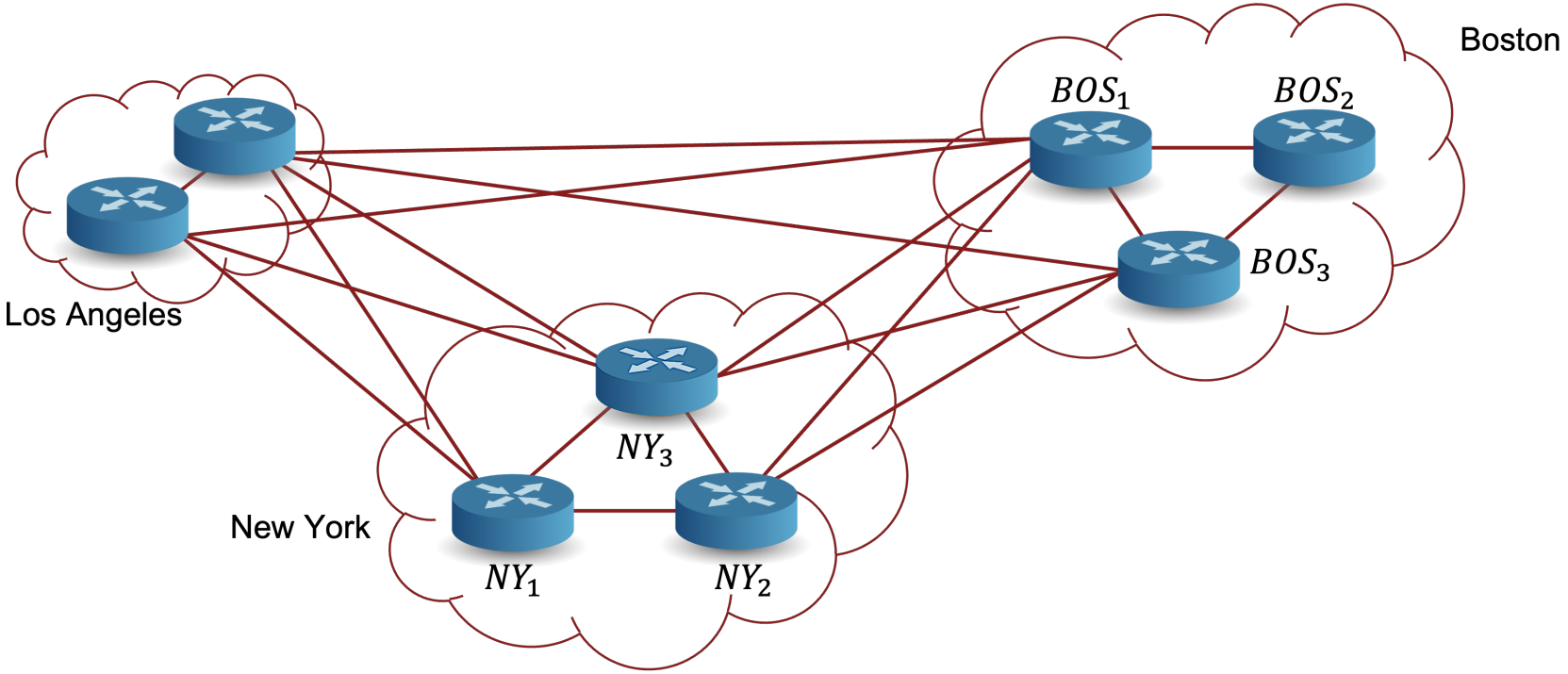
Zibin Chen and Lixin Gao

University of Massachusetts, Amherst

May 19, 2023

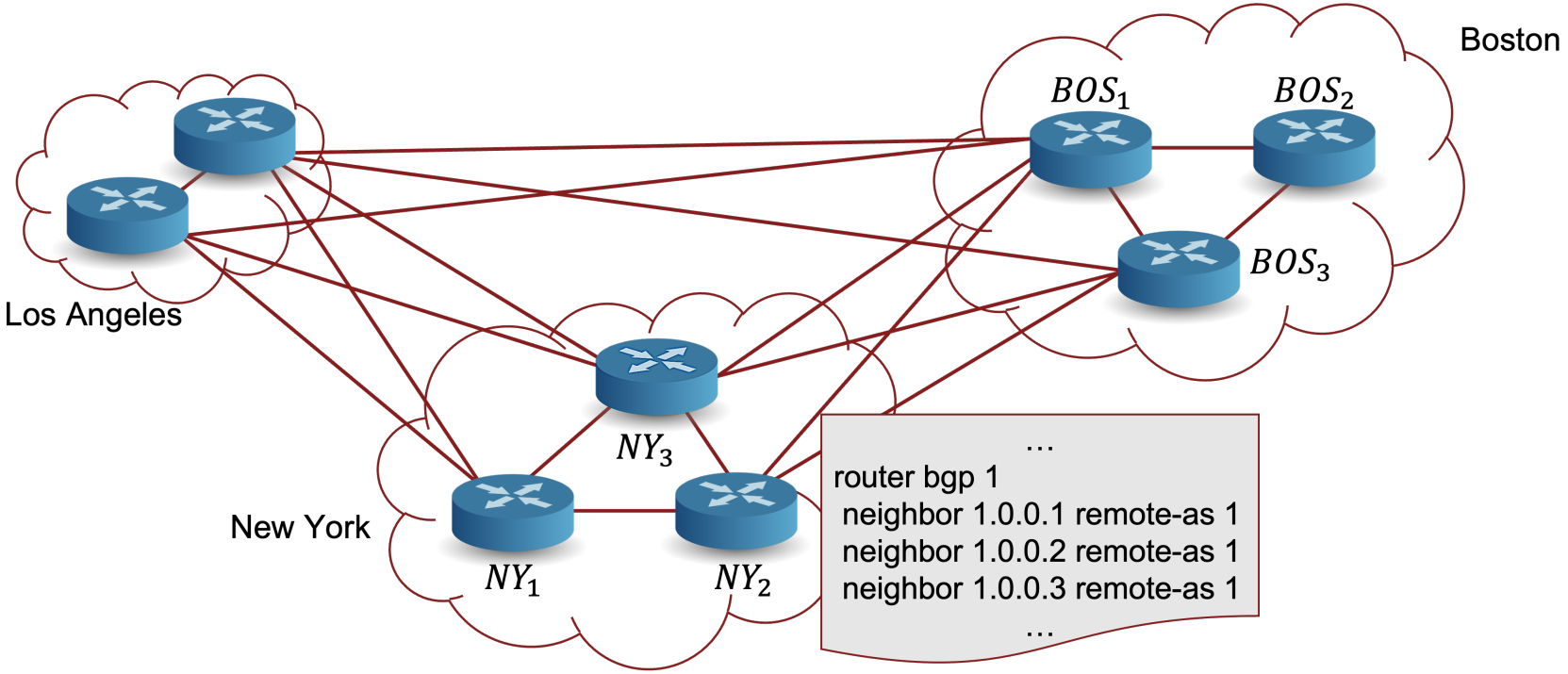
Network Configurations

- Network operators configure each router separately.

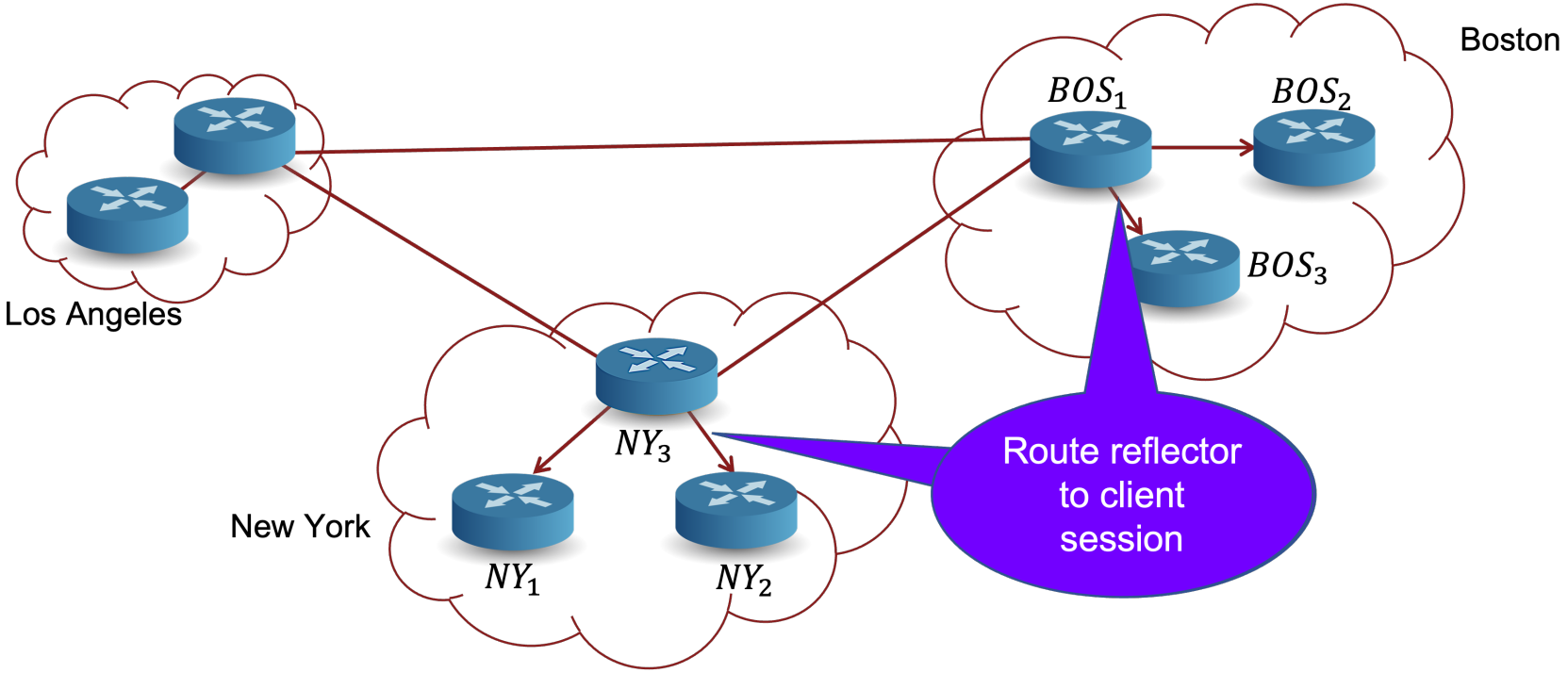


Network Configurations

- Network operators configure each router separately.

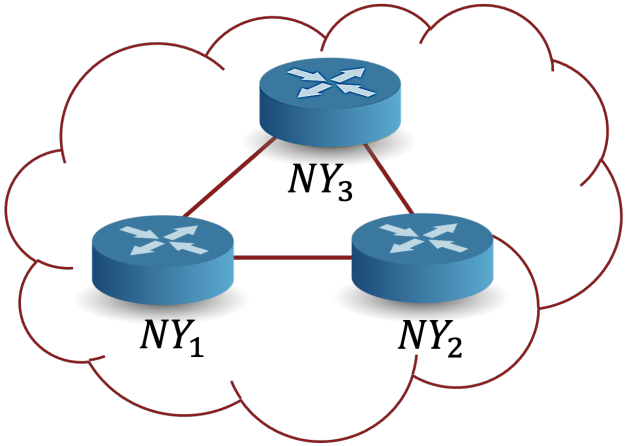


Configuration Update Scenario

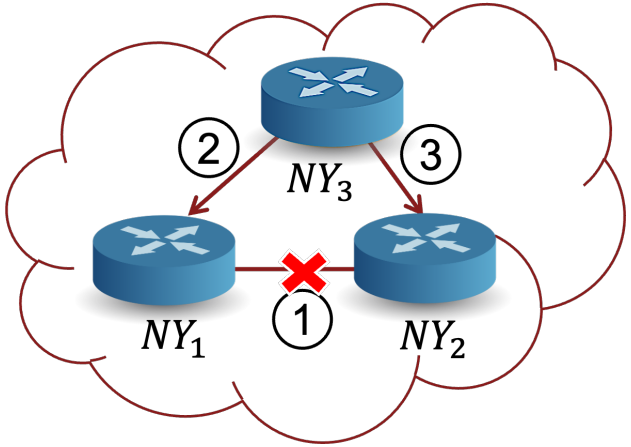


Updates Cannot be Applied at the Same Time

Fully meshed iBGP



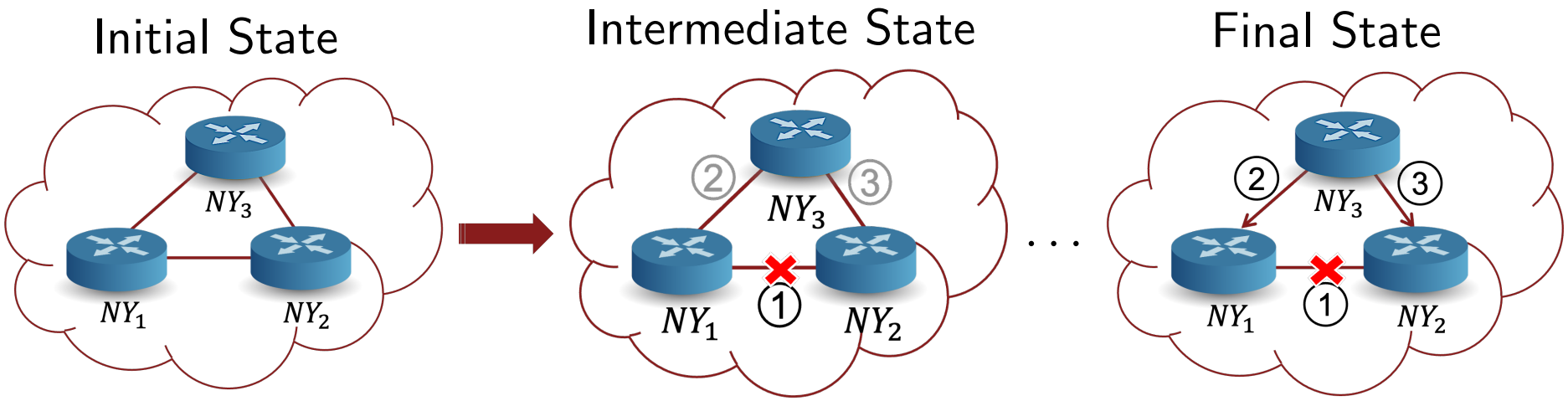
iBGP with Route Reflection



Configuration updates:

- ① : $NY_1 \rightarrow NY_2$ disconnect BGP session.
- ② : $NY_3 \rightarrow NY_1$ change to route-reflector-to-client session.
- ③ : $NY_3 \rightarrow NY_2$ change to route-reflector-to-client session.

Order of Applying Updates Matters



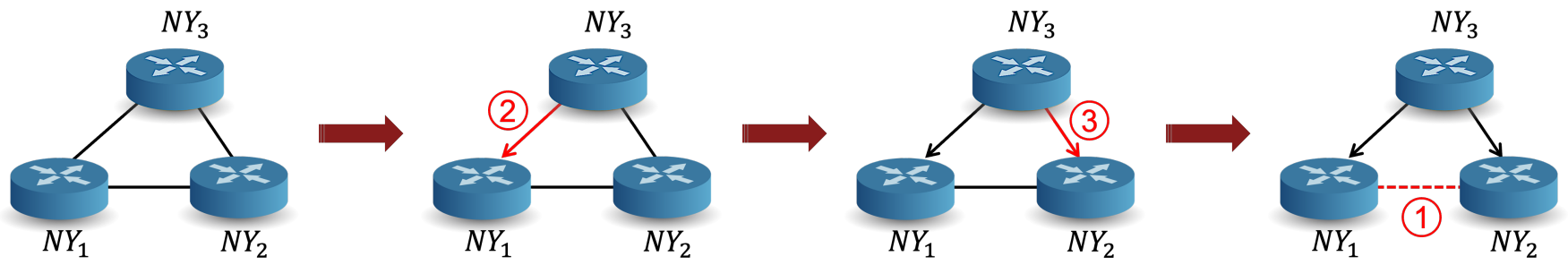
NY_2 does *not* receive routes learned by NY_1

Applied updates:

- 1 : $NY_1 \rightarrow NY_2$ disconnect BGP session.
- ~~2 : $NY_3 \rightarrow NY_1$ change to route-reflector-to-client session.~~
- ~~3 : $NY_3 \rightarrow NY_2$ change to route-reflector-to-client session.~~

Challenge

- Determine an order of applying updates



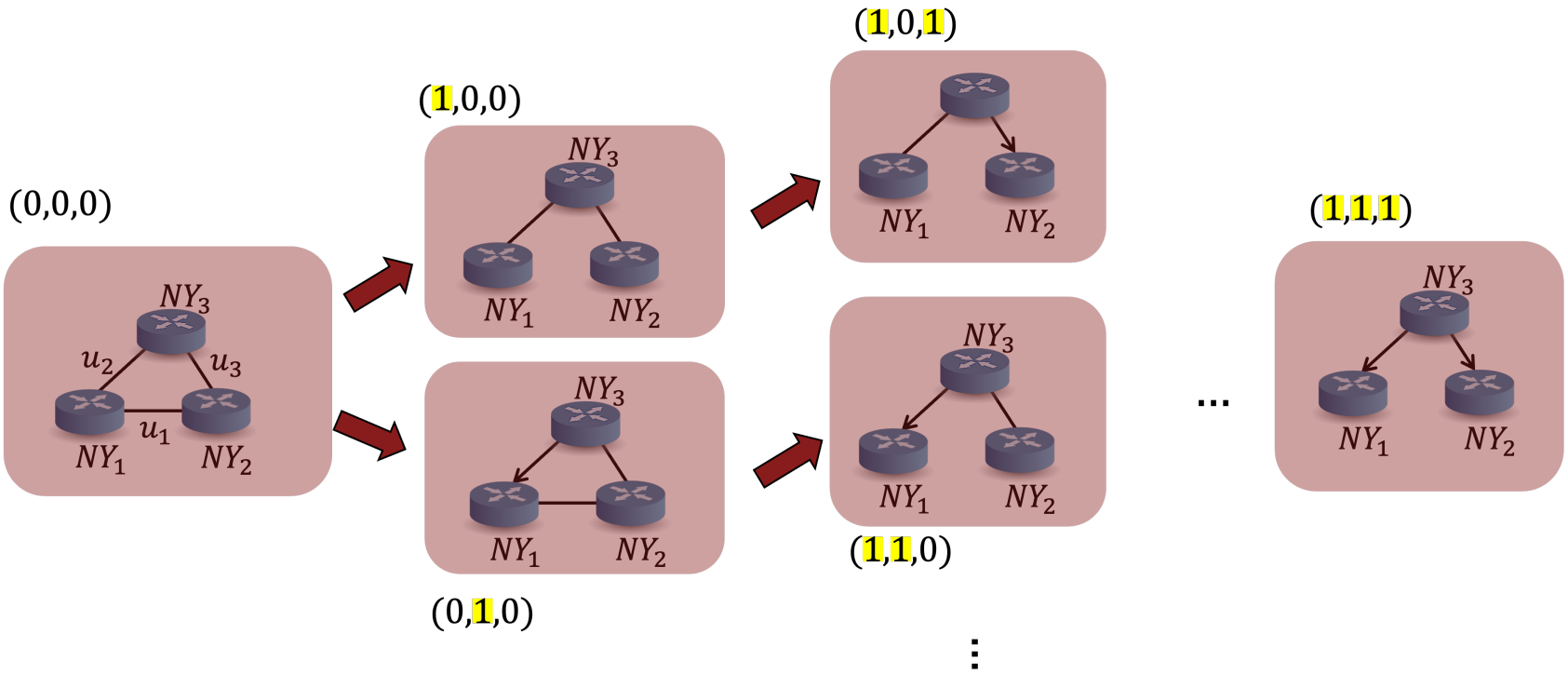
- Enumerating order space
  - Configuration updates can involve hundreds of routers changing their configurations
  - Doesn't scale: n updates have $n!$ possible orders of applying them

Related Work

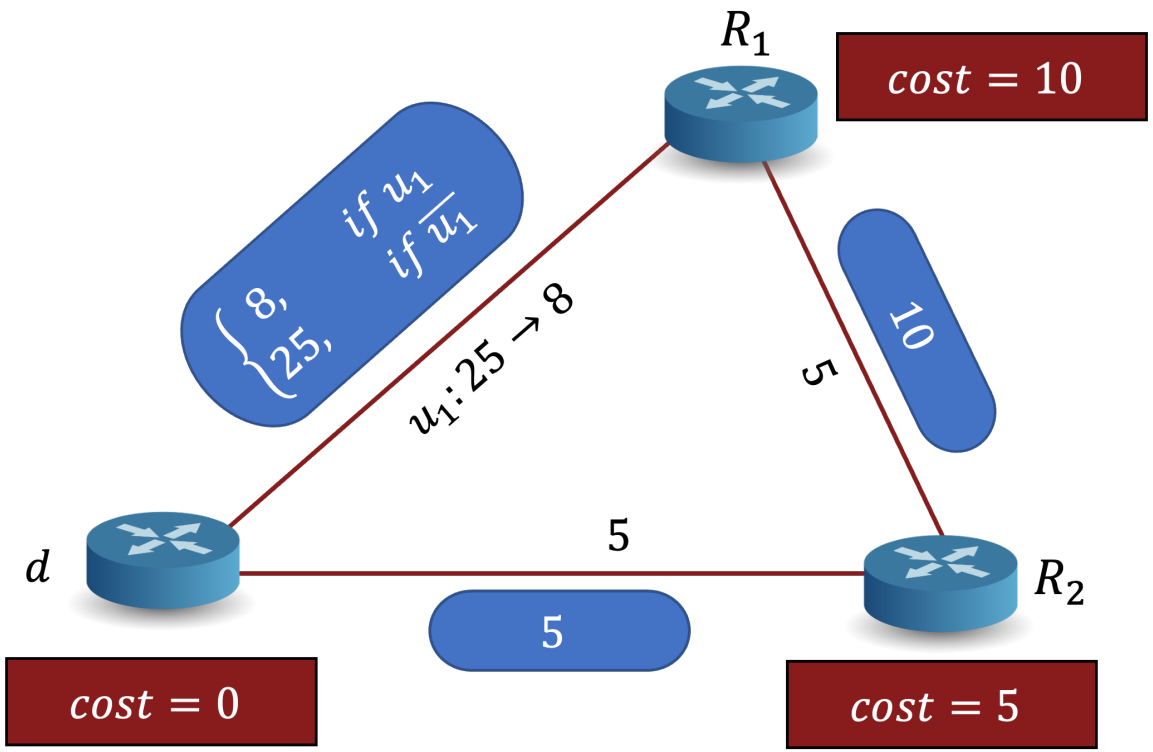
- Data Plane Verification
  - HSA (NSDI 12)
  - Delta-Net (NSDI 17)
  - Aquila (SIGCOMM 21)
- Control Plane Verification
  - Plankton (NSDI 20)
  - Tiramisu (NSDI 20)
- Configuration Update Synthesis
  - Snowcap (SIGCOMM 21)

Key Idea: Symbolic Execution

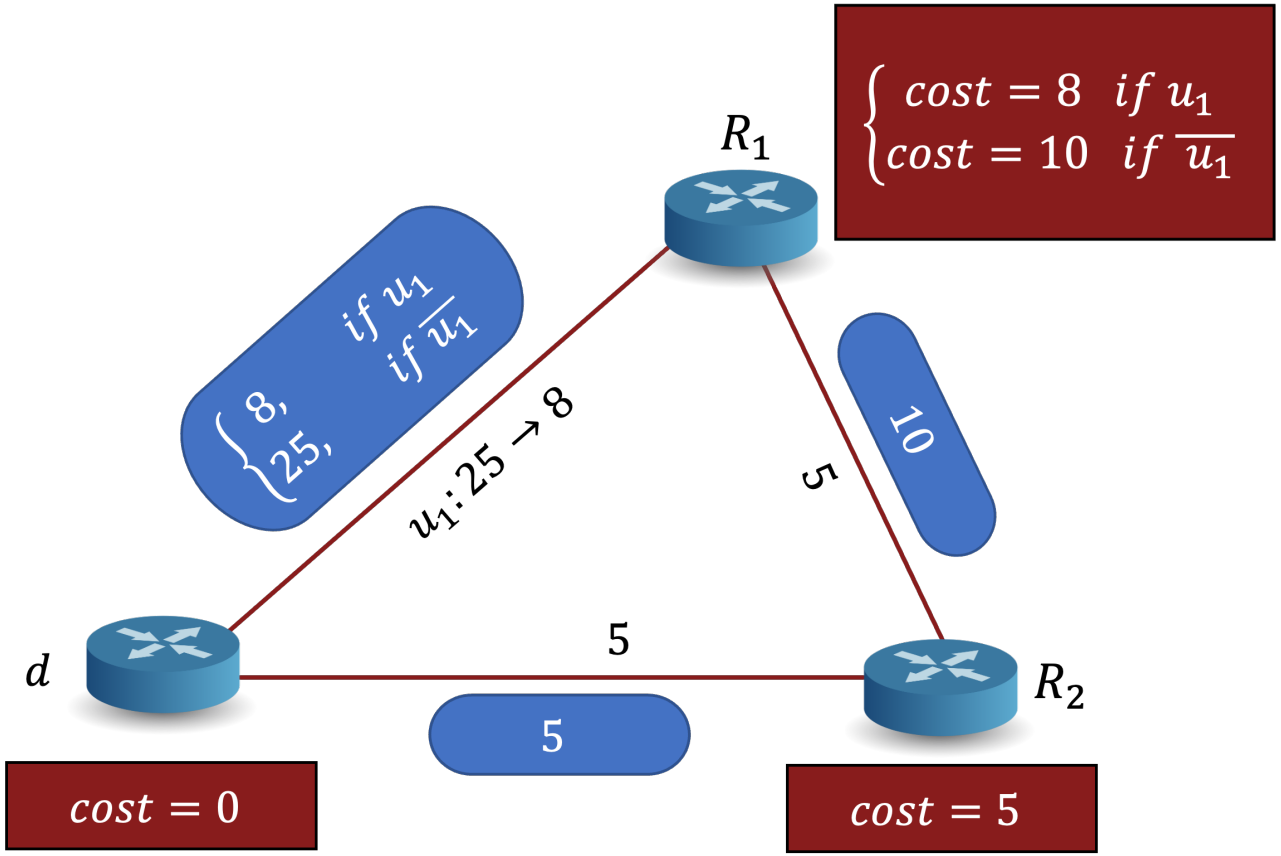
- State encodings: (u_1, u_2, u_3)



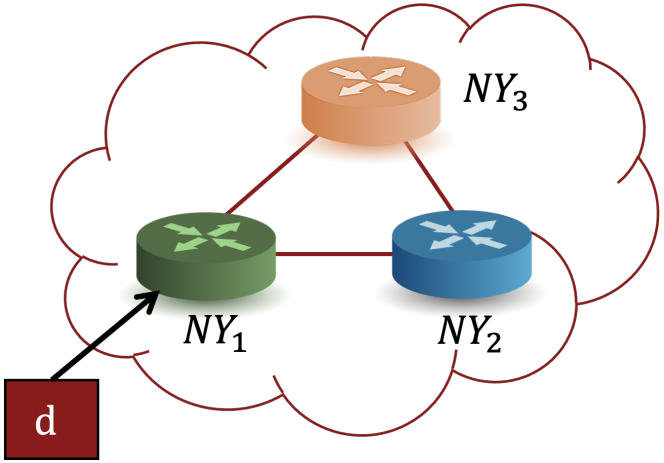
Symbolic Execution for OSPF

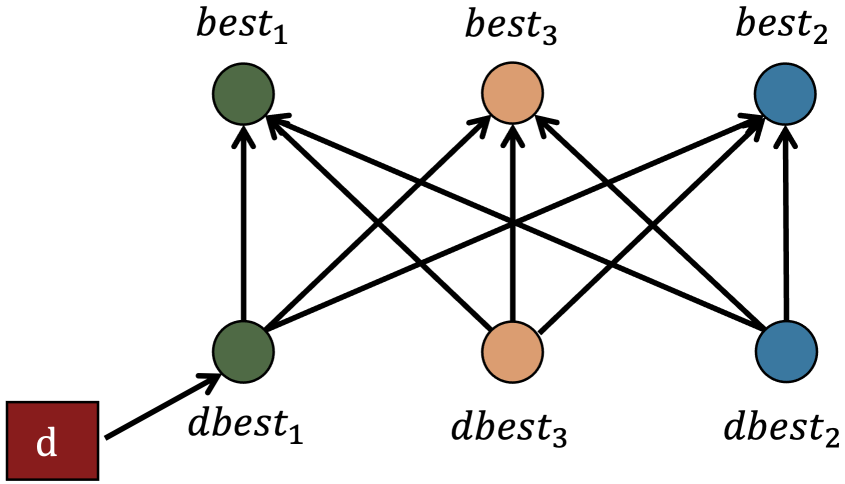


Symbolic Execution for OSPF

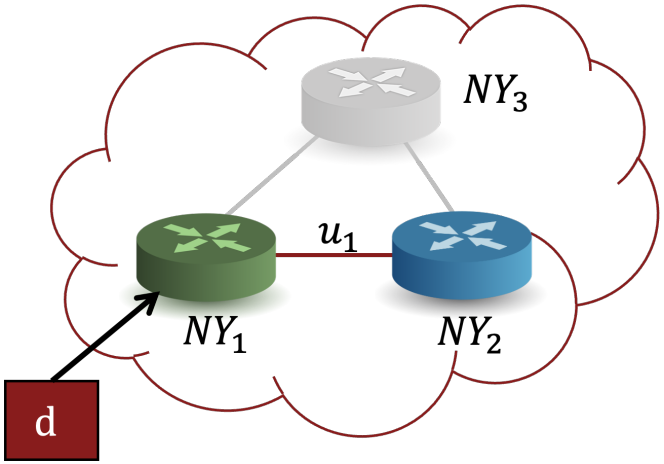


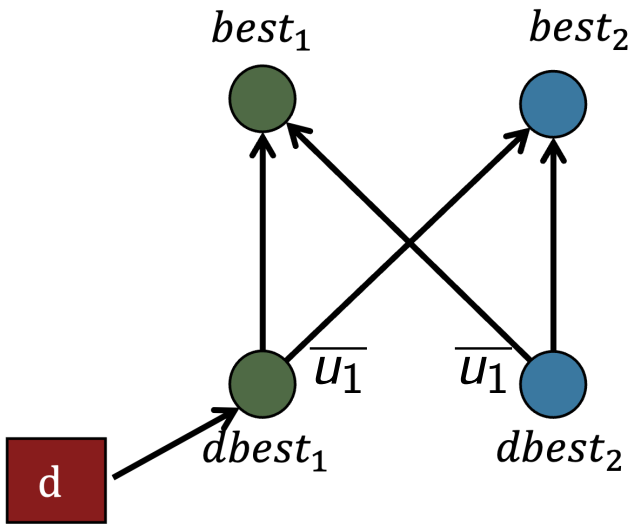
Symbolic Execution for BGP





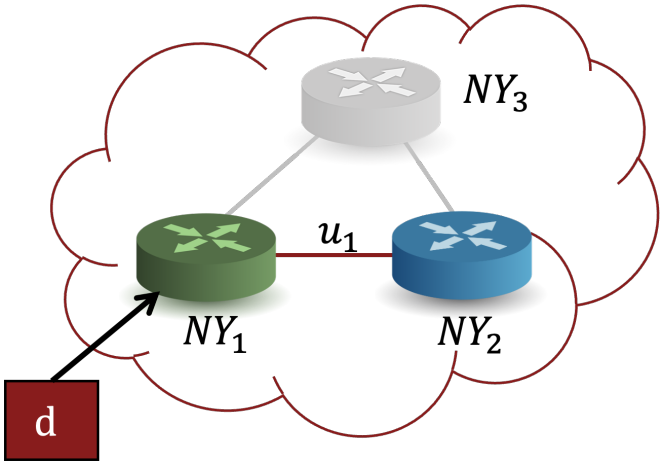
Symbolic Execution for BGP

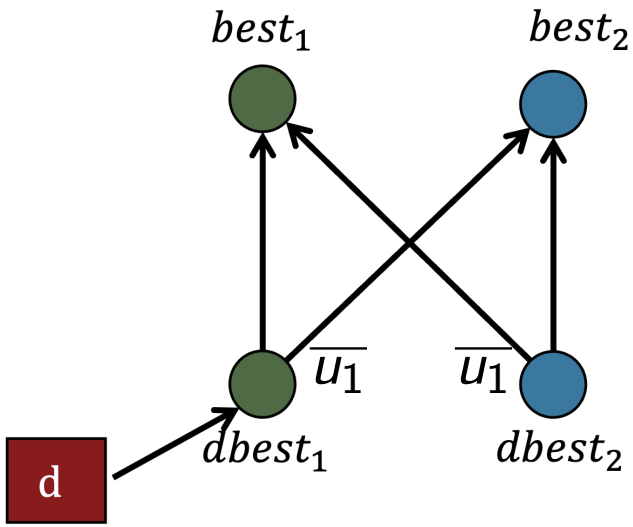




Symbolic Execution for BGP

$$\begin{cases} d\text{: via } NY_1 \text{ if } \overline{u_1} \\ d\text{: null} \quad \text{ if } u_1 \end{cases}$$

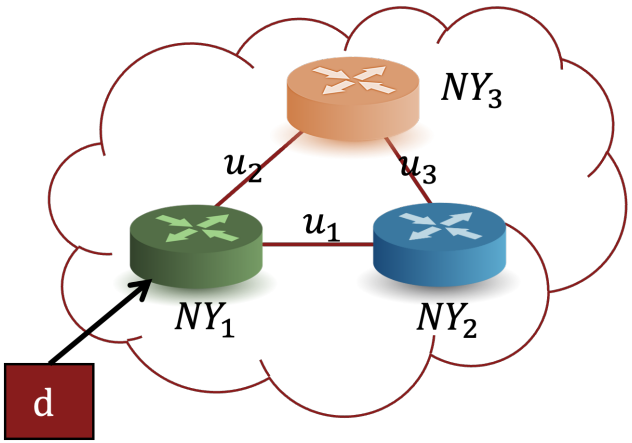


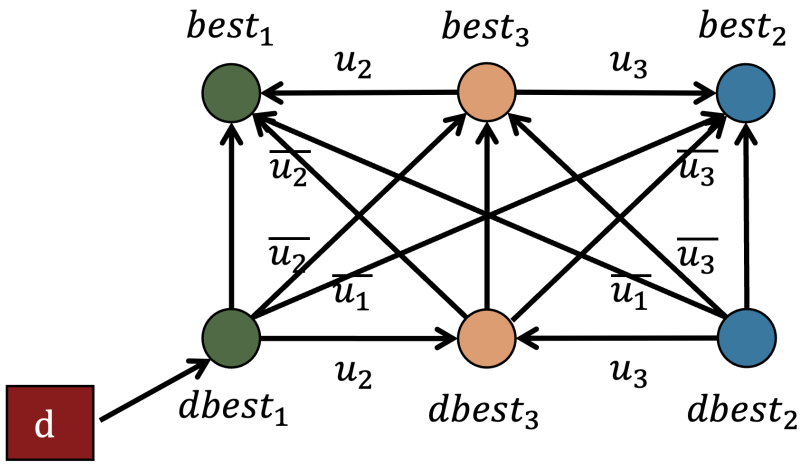


Symbolic Routing Table

- NY_1 receives route to d via eBGP.
- Symbolic routing table for NY_2 is

$$\begin{cases} \text{via } NY_1 & \text{if } \overline{u_1} \vee (u_2 \wedge u_3) \\ null & \text{if } (u_1 \wedge \overline{u_2}) \vee (u_1 \wedge \overline{u_3}) \end{cases}$$





Constructing Update Orders

- Symbolic routing table for NY_2

$$\begin{cases} \text{via } NY_1 & \text{if } \overline{u_1} \vee (u_2 \wedge u_3) \\ \underline{null} & \text{if } (u_1 \wedge \overline{u_2}) \vee (u_1 \wedge \overline{u_3}) \end{cases}$$

- Ensure reachability of NY_2
- Make $(u_1 \wedge \overline{u_2}) \vee (u_1 \wedge \overline{u_3}) = \text{false}$

$$u_1 \wedge \overline{u_2} = \text{false}, \quad \textit{and}$$
$$u_1 \wedge \overline{u_3} = \text{false}$$

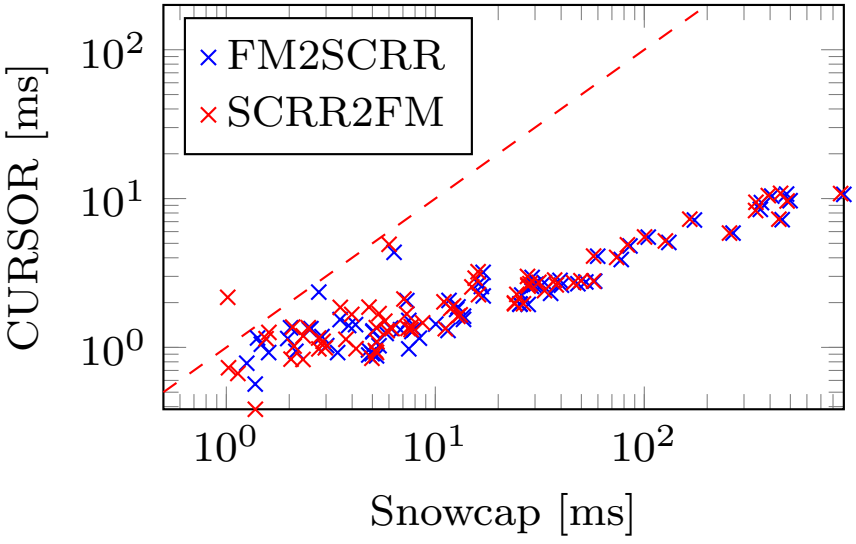
- Deriving rules
  - $u_1 \wedge \overline{u_2} \implies (u_1, u_2) \neq (1, 0) \implies u_2 \text{ before } u_1$
  - $u_1 \wedge \overline{u_3} \implies (u_1, u_3) \neq (1, 0) \implies u_3 \text{ before } u_1$
- Order: $u_2 \rightarrow u_3 \rightarrow u_1$

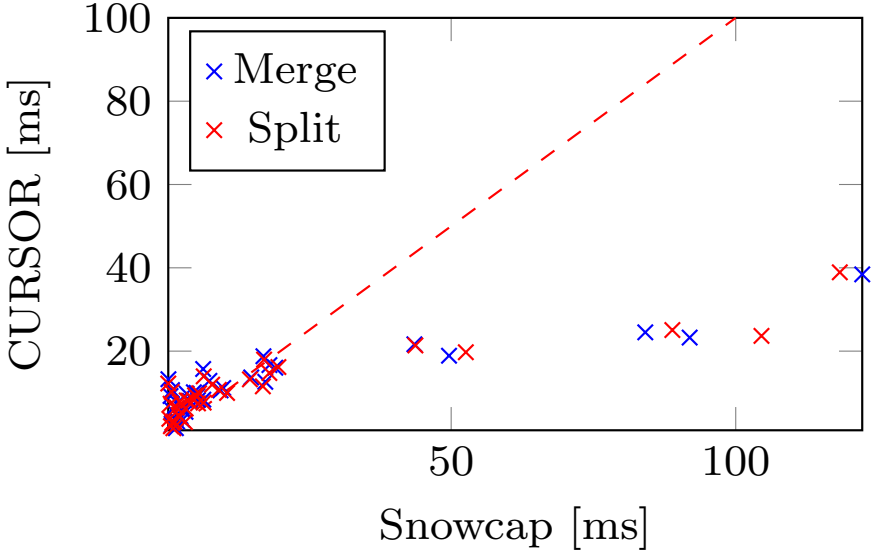
Evaluation

Experiment Setting:

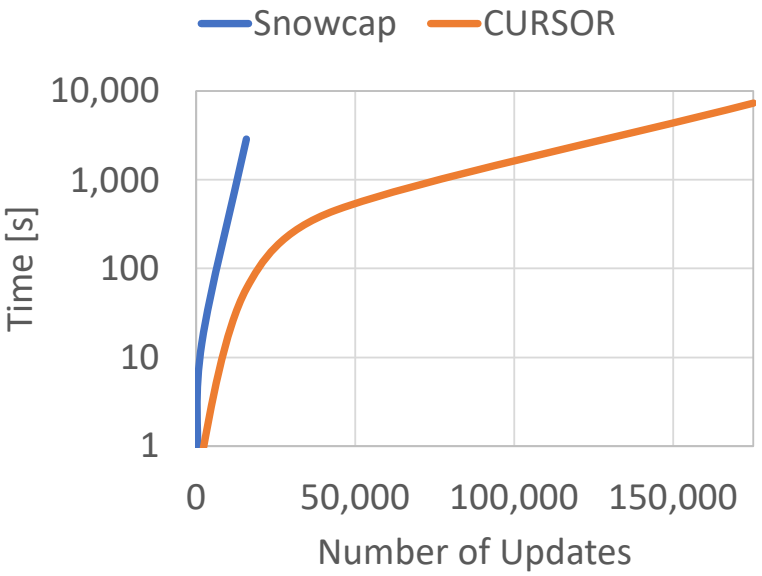
- Real-world topologies from Topology-Zoo & Synthesized topologies
- Configuration updates:
  - FM2RR/RR2FM: fully-meshed iBGP $\Leftrightarrow$ iBGP with route reflection
  - Merge/Split: Merge two networks/Split a network into two
- Properties:
  - Reachability/Waypoint

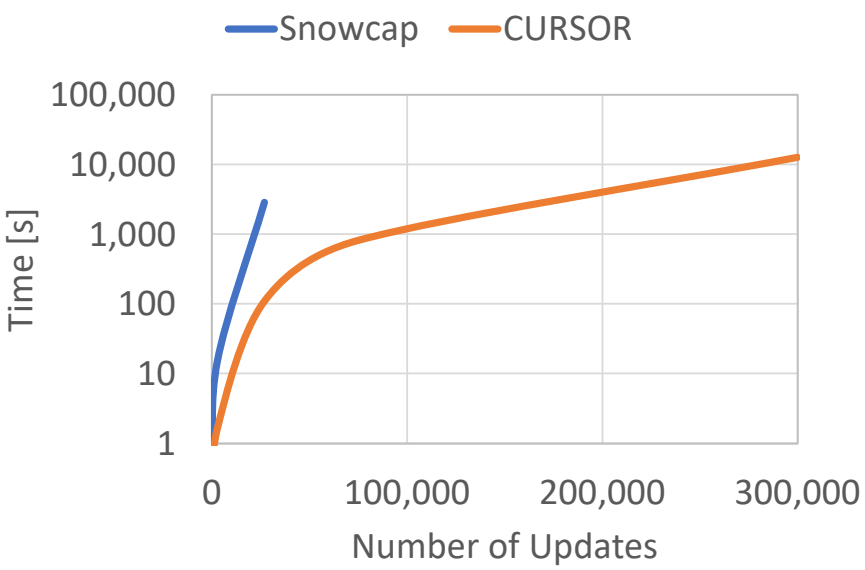
Time Efficiency





Scalability





Conclusion

- We propose CURSOR, a configuration update synthesis.
- CURSOR synthesizes configuration order by symbolically determining the routes.
- CURSOR can reduce the synthesis time by 1-2 orders of magnitude from the state-of-the-art approach.

Thanks!

University of Massachusetts Amherst