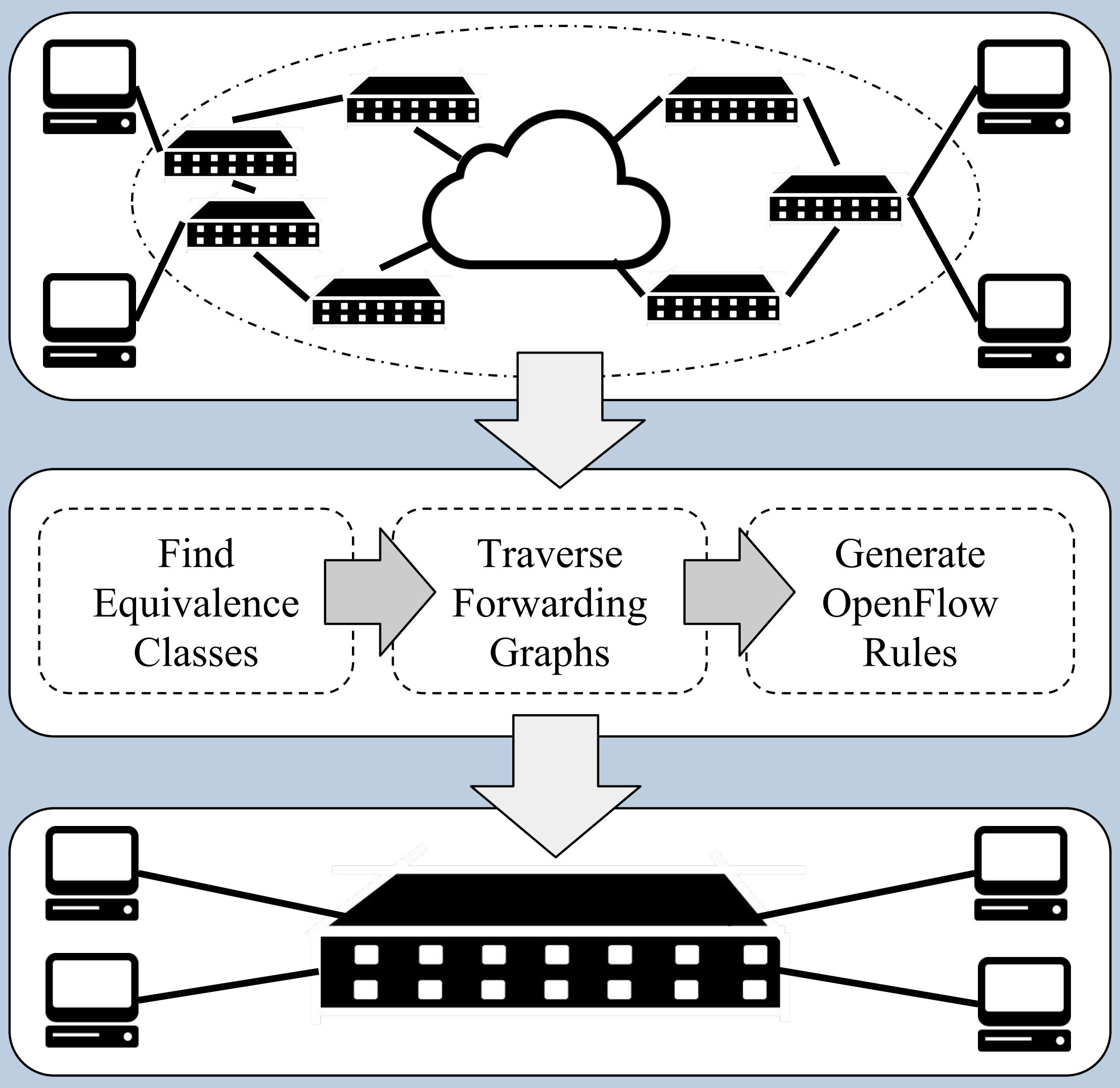
Simulation of a Software-Defined Network as One Big Switch

Jiaqi Yan, Xin Liu, Dong (Kevin) Jin
{jyan31, xliu125}@hawk.iit.edu, dong.jin@iit.edu

Problem Statement



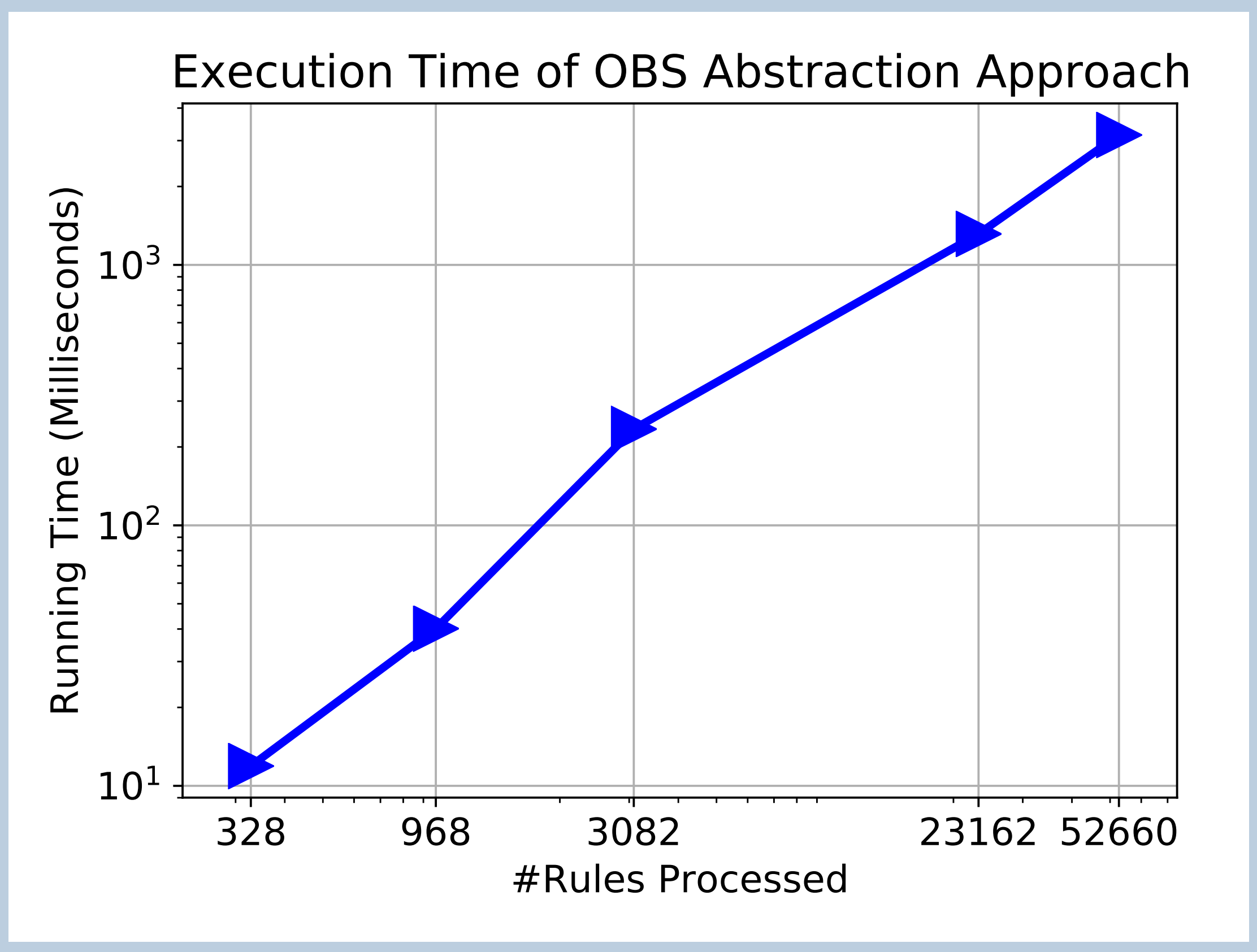
Researchers have developed multiple SDN simulation and emulation platforms to expedite the adoption of many emerging SDN-based applications to production systems.
However, the scalability of those platforms is often limited by the underlying physical hardware resources, which inevitably affects the simulation fidelity in large-scale network settings. In this work, we present a model abstraction technique that effectively transforms the network devices in an SDN-based network to one virtualized switch model.

Application Scenarios

Our technique is useful if users only care about the end-to-end behavior rather than the details within the network, applicable but not limited to the following scenarios:

- Simulate a large-scale **network of networks**.
- Deal with **synchronization** problem in hybrid simulation and emulation testbed
- Share SDN network models between collaborators as **plug-in**

Approach Overhead

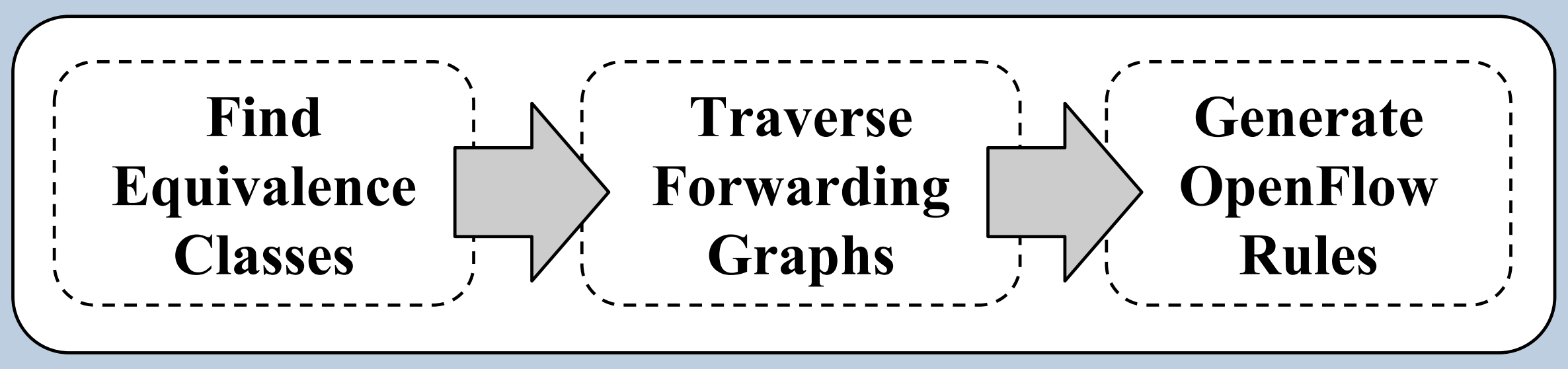


As the network state keeps evolving, it is essential to constantly update the abstracted big-switch model to reflect the changes, preferably in an online fashion. We recorded the running time for transforming various tree networks of different depth and fanout into the one-big-switch network model. The model abstraction process is lightweight. For example, it took about 3.15 seconds to process 52,660 rules.

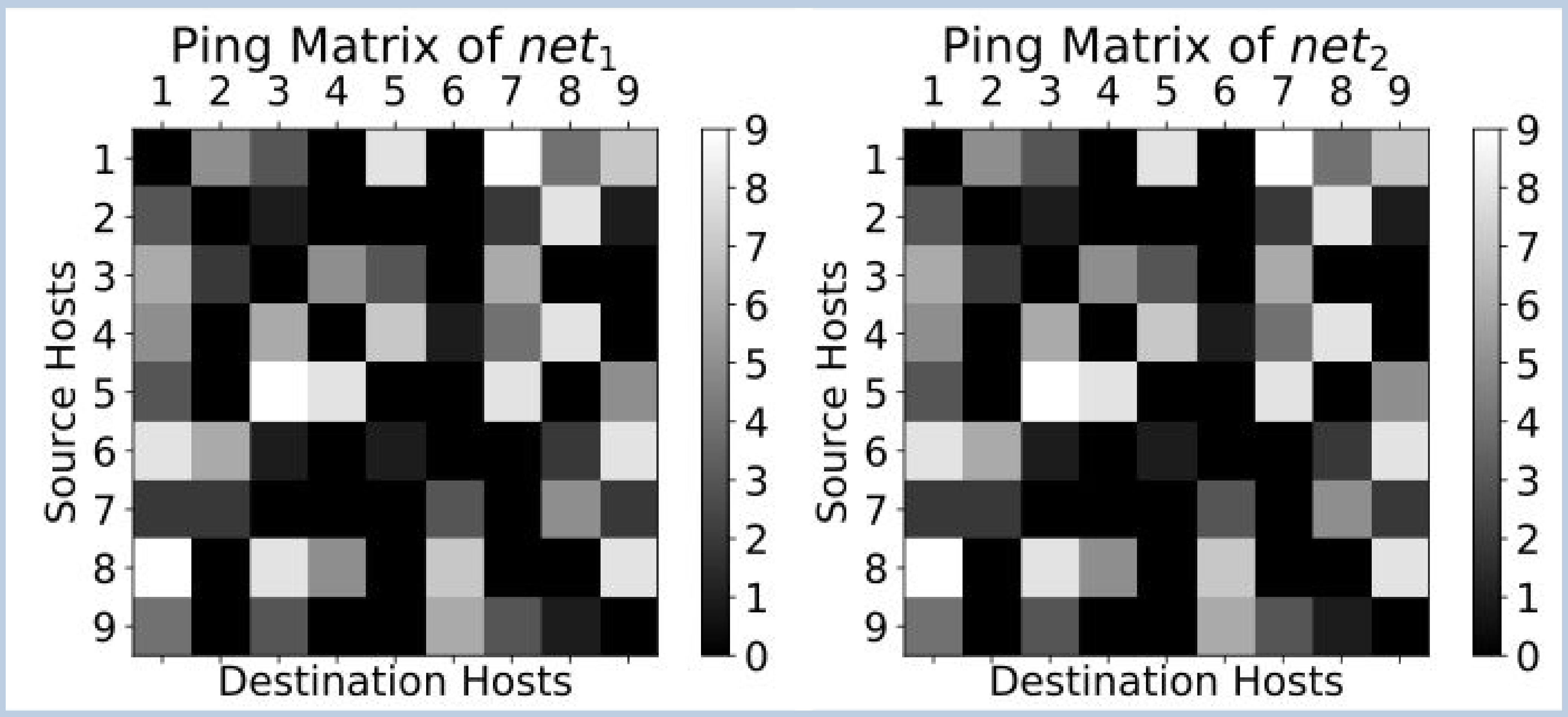
Three-Step Model Abstraction Approach

To effectively transform a static SDN data plane configuration to "one-big-switch" model that preserves the same end-to-end forwarding behavior, we need to identify how every packet is processed in the snapshot, and how to correctly configure the big-switch model to reflect the identical forwarding logic. We develop a three-step model abstraction method, summarized as follows.

- **Identifying Equivalence Classes (EC).** We partition all possible packets in the network into mutually exclusive sets and the packets belongs to the same EC are processed in the same way. Those sets are identified according to the matching field of **all** the OpenFlow rules on **all** the SDN switches in the original network.
- **Creating Forwarding Graphs.** We model the forwarding behavior of each packet set using the topology information as well as the local information stored on SDN switches (e.g., port mapping, rule priorities, etc), and generate a graph-based model to represent the forwarding behavior.
- **Generating OpenFlow Rules of the Big Switch.** We generate the OpenFlow rules for the big switch in order to preserve the end-to-end forwarding logic. This step includes (1) constructing the port-to-host mapping, (2) generating the rules by matching the packet header of each set, and (3) forwarding the packet to the correct output port.

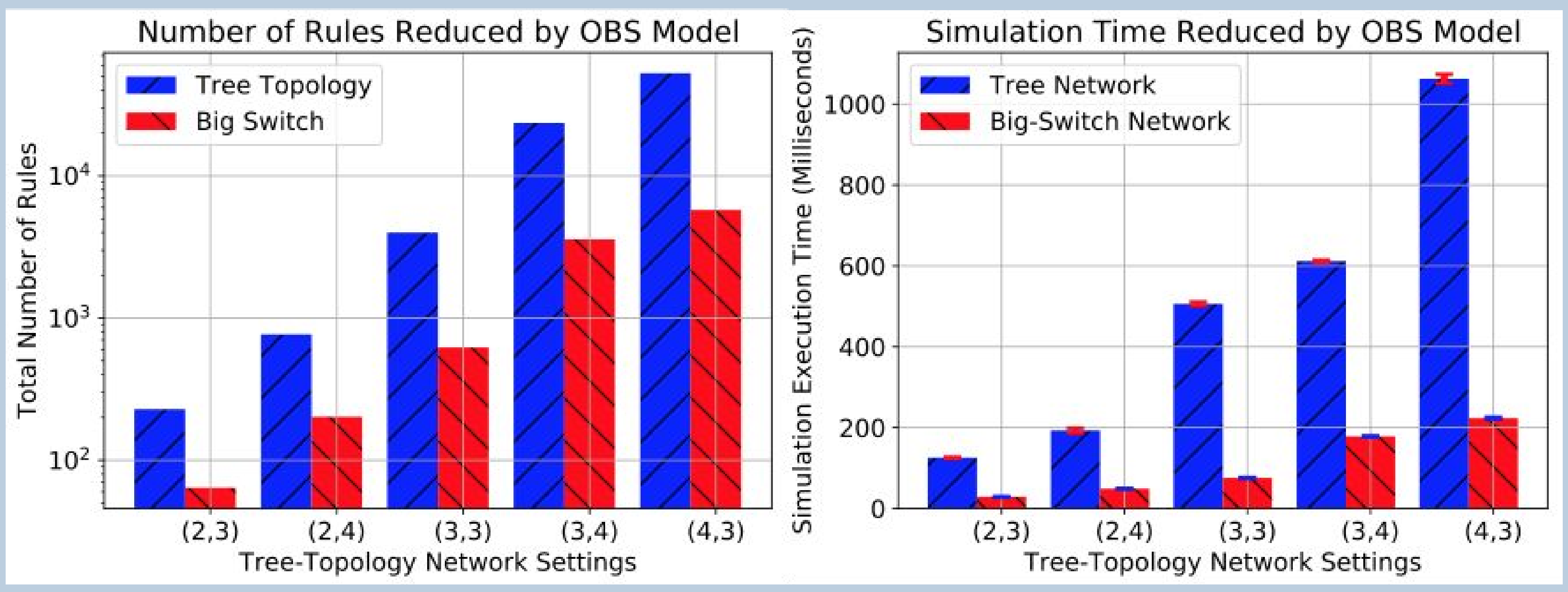


Preserving Forwarding Logic



net$_1$ emulates the original SDN-based network with a tree topology (d, f), where depth d and fanout f are tunable. **net$_2$** emulates the corresponding one-big-switch-based network for each net_1 We conduct ping test between all combination of host-ends. Matrix $\mathcal{R}_1$ and $\mathcal{R}_2$ represent the number of packets received at each host in net_1 and net_2, respectively. The gradient legend visualizes the number of received packets. Identical $\mathcal{R}_1$ and $\mathcal{R}_2$ indicate that our model abstraction technique preserves the network forwarding logic.

Performance Gain



Number of OpenFlow Rules. We compare the total number of rules installed on the switches in both net_1 and net_2 with the same experimental settings. The number of rules on the big switch is about 72% to 89% less than the number of rules in the original tree-topology network.
Simulation Running Time. When executing both models in the SDN simulator S3FNet, the big-switch-based network model saves about 75% to 85% running time as compared to simulating the corresponding original SDN-based network.