A NICE Way to Test OpenFlow Applications

Ashutosh Mittal (E-mail: amittal@kth.se)

I. Paper summary

THIS paper introduces *NICE* (No bugs In Controller Execution), a tool for automating the testing of OpenFlow applications that combines model checking and concolic execution. It uses a novel way to quickly explore the state space of unmodified controller programs written for the popular NOX platform. It augments model checking with symbolic execution of event handlers (to identify representative packets that exercise code paths on the controller). It also presents a simplified OpenFlow switch model (to reduce the state space), and effective strategies for generating event inter-leavings.

NICE consists of three parts: (i) a model checker, (ii) a concolic-execution engine, and (iii) a collection of models including the simplified switch and several end hosts. In order to handle the diverse inputs to the packet in handler, it constructs symbolic packets. Also, to minimize the size of the state space, it chooses a concrete (rather than symbolic) representation of controller state. Systematically exploring all code paths, for every feasible code path in the handler, the symbolic-execution engine finds an equivalence class of packets that exercise it. For each equivalence class, it instantiate one concrete packet and enables a corresponding send transition for the client. It also proposes domain-specific heuristics that substantially reduce the space of event orderings while focusing on scenarios that are likely to uncover bugs.

In testing three real world applications: a MAC-learning switch, in-network server load balancing, and energy efficient traffic engineering; NICE uncovered eleven bugs.

II. Significant Contributions

- Its novel way of combining model checking symbol execution, in comparison to the other model-checkers strikes a good balance between (i) capturing system concurrency at the right level of granularity, (ii) simplifying the state space and (iii) allowing testing of unmodified controller programs.
- NICE systematically explores the space of possible system behaviors, and checks them against the desired correctness properties. It also gives the freedom to configure the desired search strategy. In the end, it outputs property violations along with the traces to deterministically reproduce them.
- NICE prototype tests show that it is five times faster than other contemporary tools, its OpenFlow-specific search strategies reduce the state space by up to 20 times, and the simplified switch model brings a 7-fold reduction on its own.

III. Unresolved Issues

- *Concrete execution on the switch:* In identifying the equivalence classes of packets, the algorithm assumes the packets reach the controller. However, depending on the rules already installed in the switch, some packets in a class may reach the controller while others do not. This leads to some loss in both efficiency and coverage. Thus, symbolic forwarding still needs to be further optimized.
- *Concrete global controller variables:* In symbolically executing each event handler, NICE could miss complex dependencies between handler invocations. In some cases, one call to a handler could update the variables in a way that affects the symbolic execution of a second call, whereas symbolic execution of the second handler would start from the concrete global variables. Thus efficient representation of global variables symbolically is required.
- *Infinite execution trees in symbolic execution:* Infinite execution trees is an inherent limitation of symbolic execution. The paper tries to address this limitation at the expense of some loss in coverage. The trade off needs to pushed as farther as possible.