Maple: Simplifying SDN Programming Using Algorithmic Policies

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

I. Paper summary

THIS paper introduces *Maple*, a system that simplifies SDN programming by allowing a programmer to use a standard programming language to design an arbitrary, centralized algorithm (algorithmic policy) to decide the behaviors of an entire network, and providing an abstraction that the programmer-defined, centralized policy runs, conceptually, afresh on every packet entering a network. It includes a highly-efficient multicore scheduler that can scale efficiently to controllers with 40+ cores, as well as a novel tracing runtime optimizer that can automatically record reusable policy decisions, offload work to switches when possible, and keep switch flow tables up-to-date by dynamically tracing the dependency of policy decisions on packet contents as well as the environment (system state).

Maple introduces two key components in its design to efficiently implement the abstraction. The first is an optimizer, or tracing runtime, which automatically discovers reusable algorithmic policy executions at runtime, offloads work to switches when possible, and invalidates cached policy executions due to environment changes. The second is a runtime scheduler, or scheduler for short, which provides Maple scalable execution of policy misses generated by the many switches in a large network on multicore hardware. In addition, Maple allows a set of higher-level tools to be built on top of the basic abstraction namely, deployment portal which imposes constraints on top of Maple, and policy composer, which an SDN programmer can introduce.

II. Significant Contributions

- Like The design of Maple uses the concepts of layer. Maple divides the architecture into two levels. For user level, users dont need to adapt to a new programming model but use standard languages to design arbitrary algorithms for forwarding input packets. Meanwhile, for system level, the optimizer and run-time scheduler just need to focus on performance issues, correctness of rules installed in the OpenFlow switches.
- Mapple's algorithmic policies provide a simple, expressive programming model for SDN, eliminating a key source of errors and performance problems. Maple provides an implementation of algorithmic policies through several novel techniques, including: runtime tracing of algorithmic policies, maintaining a trace tree and compiling TT to flow tables to distribute processing to switches; using TT annotations to implement compiler optimizations such as rule and priority reductions.
- It introduces a novel SDN optimizer that discovers reusable forwarding decisions from a generic running control program which uses a trace tree that records the invocation of the programmer-supplied f on a specific packet then generalizes the dependencies and outcome to specific packet.

III. Unresolved Issues

- Scalability is an issue for this tree based approach. Tree structure depends on the policy framework code and even reordering conditions can lead to different trees. Thus, a badly written policy can easily cause the algorithm to explode. Thus, it requires some static analysis.
- The tool does not account for elements like load balancer. In case of non-deterministic code path, ideally controller should not cache, however this doesn't happen. Thus, further work is required to make the abstraction work for all the network elements and be implemented in production networks.
- Speed is again an issue as with all the other SDN controllers. Even though sing cores solves the problem partly, but still it requires to be quicker to be able to be used in production networks.