Pyretic: Composing SDNs

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

I. Paper summary

THIS paper introduces *Pyretic* which is north bound high level controller API programming language. It creates parallel modules handling different tasks on the same traffic. User doesn't need deal with low level forwarding rules or their consistency. Pyretic lets the user simply define policies and then it converts them to efficient forwarding rules. In particular, instead of implementing a policy by incrementally installing physical rule after physical rule on switch after switch, a Pyretic policy is specified for the entire network at once, via a function from an input located packet (i.e., a packet and its location) to an output set of located packets. The output packets can have modified fields and usually end up at new locations. The programmer does not need to worry about which OpenFlow rules are used to move packets from place to place.

Pyretic policy language allows using boolean predicates using which one can do parallel as well as sequential processing. It allows for three types of abstractions in the form of Policy Language, Network object and Packet model. It treats packets like a dictionary and allows arbitrary addition of virtual fields. It also allows for static, dynamic and recurring policies and supports rich variety of topology abstraction.

II. Significant Contributions

- There were several works before this paper that tried to adopt modularity approach like slicing application controls in FlowVisor. However, those work do not address how to build a single application out of multiple, independent, reusable network policies that affect the processing of the same traffic. To solve these problems, this paper proposed new abstractions for building application out of multiple, independent modules that jointly manage network traffic.
- Pyretic creates Network objects which offer both information hiding and protection, while offering the familiar abstraction of a network topology to each module. Each network object has three key elements, namely a topology, a policy, and a mapping for derived networks. So, Pyretics provides several constructs for implementing these functionalities facilitating transforming topologies and transforming policies.
- Overall, Pyretic has been a breakthrough in the sense that it provides an intuitive and easy way for user to implement network policies without going into low level details.

III. Unresolved Issues

- Since the language is based on Python, the controller couldn't scale very well because of computing limit of the host machine.
- They mention that their implementation was slow. How much of this is due to being in Python and interpreted versus inherent properties in abstracting away the network, using stacks for fields, and maintaining virtual fields?
- The paper does not talk about how the high-level policy rules translate to low-level forwarding table rules. This remains