PGA: Using Graphs to Express and Automatically Reconcile Network Policies

Nate Foster
Cornell University
Ithaca, NY, USA
jnfoster@cs.cornell.edu

Network operators must often combine several different policies into a single coherent policy that faithfully encodes the constraints of each input. For example, the operator of a large enterprise network might combine a policy requiring all web traffic to traverse a load balancer with another stipulating that all external traffic must traverse a firewall. The combined policy would need to route external web traffic across a load balancer and a firewall.

Composing policies correctly turns out to be challenging, especially when they are expressed in terms of low-level constructs such as access control lists or switch forwarding rules. High-level programming languages such as Pyretic and NetKAT offer composition operators that avoid these issues in some cases, but none of them guarantee that the joint intent of the policies being combined will be preserved in general. Instead, programmers must perform intricate rewritings on policies—decomposing the inputs into their constituent pieces, and then manually reassembling them to produce the desired result.

This paper presents a new framework, Policy Graph Abstraction (PGA), that is designed to overcome these challenges. In PGA, operators express policies in terms of intuitive, graph-based abstractions in which nodes represent groups of end hosts or network functions, and edges indicate allowed communication. Optional annotations constrain the types of traffic that may flow over edges, and restrict the policies the graph may be validly composed with. A syntactic policy composition operator combines several different policy graphs into a single graph that encodes the constraints of each input (or fails if it is impossible to do so). It works by normalizing the input policies so that each end host group is disjoint, and merging sequences of network functions so that all paths in the composed policy traverse the sequences specified in the inputs. PGA has been implemented in Python and evaluated on a collection of benchmarks including several synthetic applications and a real-world access control policy from a large enterprise. The composition of tens of thousands of policies can be computed in less than ten minutes. The scalability of the tool depends critically on PGA's use of end host groups, which decouple the expression of policy for various traffic classes from the mapping between individual end hosts and groups.

The SIGCOMM reviewers were excited about this paper because it addresses a tricky problem that arises in many organizations, especially ones in which there are multiple policy sub-domains. The idea of abstracting away low-level implementation details to enable more flexible composition is simple and powerful. The graph-based abstractions provided in PGA are elegant and closely match how many network operators think about policy at an intuitive level. PGA also resembles some of the group- and intent-based frameworks that are being actively developed in industry. Hence, it seems likely that the abstractions and algorithms that are developed in this paper will have broad appeal and lasting impact.

PGA is not the final word on high-level policy formalisms and there are many promising directions for future research. The paper does not deal with the issue of policy compilation—i.e., how to translate policies into low-level configurations for switches, middleboxes, and end hosts. It does not offer an independent description of the semantics of policies, which means the expressiveness of the framework is not easy to analyze and questions about properties of the composition operator—*e.g.*, is composition associative, commutative, and idempotent?—can only be answered by understanding the implementation. Although the initial experimental results that quantify the costs of composition are promising, it remains to be seen how well a system based on PGA would perform in practice.

Overall, PGA offers an elegant solution to the difficult problem of expressing and composing multiple policies. It represents a significant contribution to the literature on policy formalisms and should trigger a lot of discussion and follow-up work.