I'm doing some independent work this semester with Professor David Walker, with whom I took functional programming last semester. My project is being implemented functionally, however the main focus is on rule optimization in network routers--specifically, routers within software-defined networks (SDNs).

SDNs

The difference between SDNs and traditional networks lies in the positioning and functionality of the control plane: typically, when you buy a bunch of routers from Cisco, they’re implemented as a control plane and a data plane, where the former decides (at a high level) where packets should go, and the latter actually processes the packets as they come in and ships them off according to the former’s directions. SDNs, on the other hand, strip away the control plane, creating a single master controller that communicates with every router, where the routers are now just data planes.

There are a ton of potential advantages to this type of networking. To give you a taste: say you’re trying to figure out how to get from New York to Miami. With traditional networks, you’re stuck on the ground in a swamp, asking nearby travellers if they can lend you some information. With SDNs, you’re looking down at the world from a high level, as if you have the world mapped out—finding a path is trivial.

Rule Optimization

Each router in the network is implemented as a *flow table*, which is a set of {*pattern, action}* pairs. Each pattern matches on some field of the incoming packets and, if it finds a correct match, the action is performed (where the actions are things like *forward to port 5* or *drop from network*).

But the pattern matching can get pretty complicated: you can match on multiple fields (for example, some routers match on *both* source IP and destination IP addresses), require multiple tables, etc.

The hardware that implements this matching is known as TCAM. It works in parallel to perform O(1) matching, but it makes up in cost what it gives you in speed—that is, you can’t get much of it because it’s highly expensive. As a consequence, you want to use as few rules as possible in your router.

A Family of Algorithms

That’s where my work comes in: I’m trying to develop a family of algorithms that optimizes the number of rules necessary in a given flow table. Included in this family would be:

* Optimizing lists that match on a single field
* Optimizing lists that match on multiple fields
* Optimizing lists by breaking a single table into multiple tables

These problems have been progressed to various degrees: the first is solved (and I’ve implemented the solution); the second is considered NP-complete for more than two fields, but a good approximate solution exists (which I’ve also implemented); and the third has not received much attention, which perhaps makes it the most exciting.

Going Forward

I’ve only been working on this project for a few weeks, but already it makes me more excited about the prospect of academia. The problem I’m trying to solve here is small enough that I’ve been able to gather and read all the relevant papers, with which I’m now comfortable enough to reference by author names. I’ve really dived into the task, and I’m incredibly excited to (hopefully) contribute something to the open problem.

I’ll be posting here about my work as I progress through the semester; for reference, it all culminates in a poster session in early March.