CS 244 Project Proposal

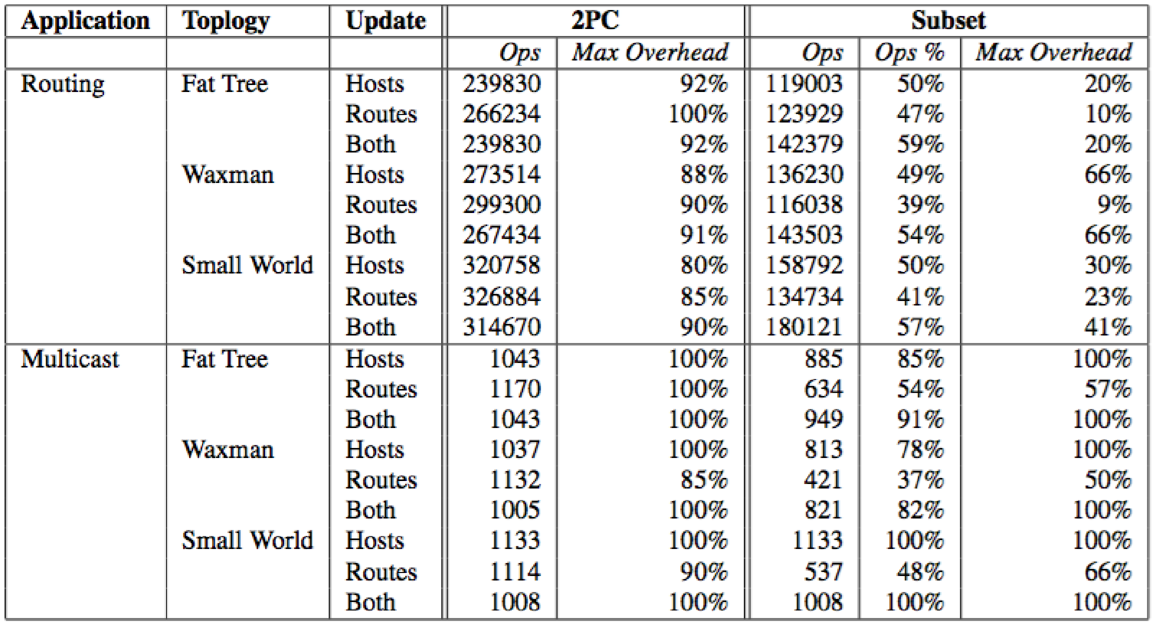
Spring '16, Sean Choi, Eyal Cidon

Paper 1 *Abstractions for Network Update*

One of the paper that we would like to replicate is the paper on the abstraction on network updates. The paper discusses the fundamental problem of difficulties in updating the network states. The author points out the following facts. Even with the arrival of SDN that hugely simplified the management of network configurations, the changes are still very low level and very prone to errors. Also, there are network states that cannot be exhaustively considered due to the complexity and the size of the network, such as considering a packet in flight at a given point in time.

In order to solve this, the authors provide an abstraction to provide an high-level abstraction that allow the programmers to simply update the configuration of the entire network. The primary abstraction is that each packets and the packets within the same flow must be processed by a single global network configuration. Given such abstractions, the authors provide several update mechanisms using the features in OpenFlow to support it. Then the author provides a theoretical model that the abstractions have been correctly implemented, as well as a verification tool to verify the properties.

The main result we would like to see is something like the following graph.



Application	Toplogy	Update	2PC		Subset		
			Ops	*Max Overhead*	*Ops*	*Ops %*	*Max Overhead*
Routing	Fat Tree	Hosts	239830	92%	119003	50%	20%
		Routes	266234	100%	123929	47%	10%
		Both	239830	92%	142379	59%	20%
	Waxman	Hosts	273514	88%	136230	49%	66%
		Routes	299300	90%	116038	39%	9%
		Both	267434	91%	143503	54%	66%
	Small World	Hosts	320758	80%	158792	50%	30%
		Routes	326884	85%	134734	41%	23%
		Both	314670	90%	180121	57%	41%
Multicast	Fat Tree	Hosts	1043	100%	885	85%	100%
		Routes	1170	100%	634	54%	57%
		Both	1043	100%	949	91%	100%
	Waxman	Hosts	1037	100%	813	78%	100%
		Routes	1132	85%	421	37%	50%
		Both	1005	100%	821	82%	100%
	Small World	Hosts	1133	100%	1133	100%	100%
		Routes	1114	90%	537	48%	66%
		Both	1008	100%	1008	100%	100%

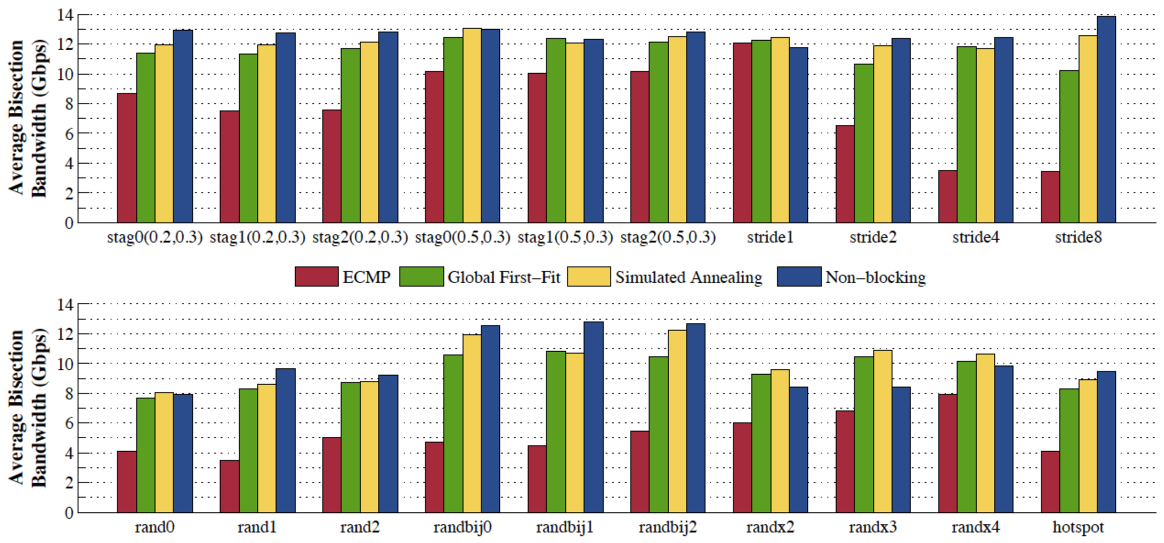
This table represents the comparison between the number of Openflow install operations between a two-phase update and Subset (the optimization in this paper). This experiment was run on Mininet topology. We hope to replicate the similar experiment as follows. First we would set up a Mininet topology and come up with upgrade plans for routing that will be mapped to both 2PC and Subset. Then, we will compare the number of operations on OpenFlow just as this table. We hope to see a similar reduction in number of operations.

Paper 2 *Hedera: Dynamic Flow Scheduling for Data Center Networks*

The rise of big data computing in data centers and the increased size and complexity of the data center has caused many application to be bottlenecked in the network. Traditionally most data center networks were designed as hierarchical trees. To accommodate growth networks are increased horizontally rather than vertically and new topologies are built as multi rooted trees that include many redundant paths between nodes. A flow scheduler is needed in order to fully utilize all the multipaths is the network. The commonly used scheduler ECMP statically determines the path for each flow using a hashing scheme, the issue with this is that occasionally a hash collision might happen causing a loss in throughput.

To deal with this the authors present a dynamic flow scheduler called Hedera. When a flow through a switch become large enough a central scheduler is updated. The scheduler then tries to math flows to non-conflicting paths and updates the switches. The scheduler uses Global First Fit and Simulated Annealing as heuristics to solve the NP complete problem finding flow routes in a general network while not exceeding the capacity of any links. Global First Fit simply places a large flow on the first available link the scheduler finds. Simulated Annealing forwards all of the flows to a specific host through a specific switch assigned to that host, the scheduler searches for the optimal solution under this constraint.

The goal of our project will be to replicate the results of these to heuristics and if time permits try another heuristic not tested by the authors. We would like to get a figure similar to figure 9 in the paper:


Average Bisection Bandwidth (Gbps)
stag0(0.2,0.3)
stag1(0.2,0.3)
stag2(0.2,0.3)
stag0(0.5,0.3)
stag1(0.5,0.3)
stag2(0.5,0.3)
stride1
stride2
stride4
stride8
ECMP
Global First–Fit
Simulated Annealing
Non–blocking
rand0
rand1
rand2
randbij0
randbij1
randbij2
randx2
randx3
randx4
hotspot