Federated Election Count Protocol

Shivam Patel

"The accumulation of all powers, legislative, executive, and judiciary, in the same hands, whether of one, a few, or many, and whether hereditary, selfappointed, or elective, may justly be pronounced the very definition of tyranny." - Madison

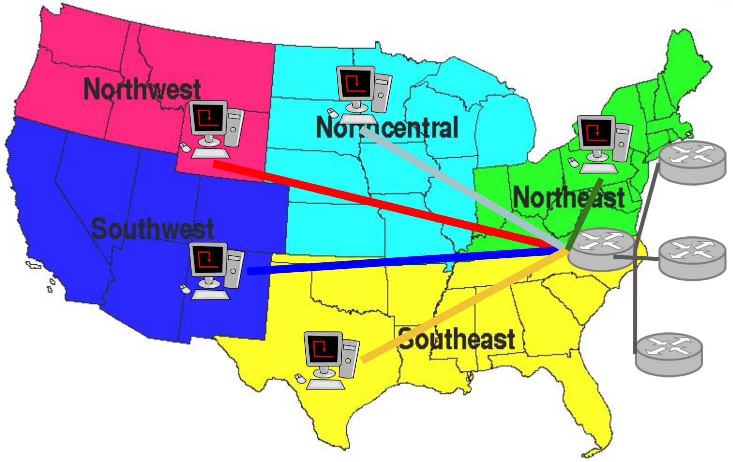
1) Problem & Motivation

- It is clear that due to problems of scale, arbitrary policy, and loss of the original ethics that were once vibrant, the consumer internet has degenerated from its original goals of openness, freedom, and decentralization.
- A necessary component of future Federated/Confederated(social media, content sharing) networks will be a way for all members in the network, to come to know about global governance decisions without naive all to all broadcasts.
- Assuming enforcement is solved for, one possible solution to the problem of efficiently sharing "decisions" is have the counting of "elections" offloaded to the network devices. A basic algorithm is explored here.

2) Proposal

- P4 is a programming language that allows for the programming of networking hardware/devices such as switches.
- In particular this allows for the design and implementation of new networking protocols. As well as offloading of computation and state management to network devices.
- Inspired by the Paxos consensus algorithm, a protocol imitating its insights is implemented here, using some of the aforementioned techniques.

3) Demo Scenario and Explanation



- An election from a few years ago, along with a map of the united states is encoded in a csv file.
- The candidates are Professors Boris and Nik competing for president.
- A graph representing a map of the US is constructed by states as nodes and states having an edge if they share a border.
- An all pairs shortest paths matrix is constructed, and regions are determined by, for a given state representing the region, its members are the set of all states within shortest path k of it, for some natural number k, such that each state is in some region, and that there may exist overlaps.
- csv file of votes is partitioned, and given to each region's host, algorithm begins by hosts sending the electoral votes and outcome of each of its member states to the central switch via FEC packets.
- Central switch relays votes to legislative switches for observation and duplicate elimination, legislative switches update their internal state to seen if not seen, and return packet with updated phase to central switch, if this is the first time a given state's votes have been seen a registers. Otherwise the legislative switch drops the packet,
- Once for a given state, the central switch has seen the corresponding packet for exactly the 2nd time, it broadcasts the result to all regional hosts, else drops.
- End result of the algorithm is the regional hosts have the same log of election results of all states in the United States, hence are able to determine a winner.

4) Results and Future Adjustments

- Implementation was reasonably successful. Due to time constraints there were some unhandled concurrency and network congestion bugs.
- In particular since a quick and dirty multi-threading was used, and implementation of hosts was done in python, python having a language peculiarity called the Global Interpreter Lock(GIL) caused some of the hosts to never report the 50th vote, i.e. never completely terminate.
- As well since more advanced features of P4 beyond state retention were not used, the broadcasting was done via a special broadcaster host, which caused a large amount of network congestion which led to its own problems.
- These issues can be solved using multi processing rather than multi threading for the regional hosts, and using multi cast groups in the switches to eliminate the need for a specialized broadcasting host.