Joseph Shumway
830003458

CSCE 181
Spring 2022

Time-Optimal Self-Stabilizing Leader Election in Population Protocols
Eric Severson
https://www.youtube.com/watch?v=_uqDoX60Oho

Eric started his keynote by talking about what a population protocols model is, being a system of n agents, each with a state. The model's interactions are many paired interactions between random agents where the they can either change or stay the same. These interactions are described using a transition rule table as shown in the video. This model was originally created for mobile sensor networks such as ZebraNet (wildlife tracking). More recently though, it's been highly useful in modeling chemical reaction networks and the dynamics of well mixed solutions. For example, how two types of molecules react with each other when mixed uniformly. This model is still being considered in other fields such as social media (for example, what happens when pairs of people or groups meet and interact?).

The topic of today though is how this model is used in distributed computing. Eric then talked about how this would be useful to find a way to model an election leader problem, where all of the states start out as Leaders, but must always converge to a single leader agent whos status is stable. This means that all further interactions will keep the leader as the only leader in the model. In this model, it is expected that $O(n^2)$ interactions are necessary until there is convergence. In parallel time, there will be n interactions per timestep, so O(n) time. The time's lower bound is provable to be S = 1/2log(log(n)) states. With this many states, it can be solved in O(logn) time.

The algorithm has to be able to be run on any arbitrary initial state, including one where all the states are followers. This means that a rule such as (L, L) -> (L, F) can't be used since if there are only followers, they can't elect a leader. Another way to tackle this problem is to use a bucket/ranking system. First create a rank for the same n number of states and shift each state forward (with overflow looping back to the first rank) if it is not the first one in that rank. This means that every state gets a rank, and the first rank is considered the leader. Coincidentally, this also solves the strictly harder ranking problem. The time analysis for this approach is O(n) for each interaction, leading to $O(n^2)$ total time.

Their new first proposed silent protocol takes O(n) time and O(n) states. Their second non-silent protocol takes O(logn) time, but exponential states. Both of these solve the ranking problem. For the first protocol, the silent states (those where we don't know their state) reset their values based on a counter that is around logn. They then propagate their value minus one to another state, which does the same and so on. This orders the states. They then construct a binary tree where all the states are ranked, with the root being rank 1 aka the leader. In the second protocol, the states are either collecting or resetting. When collecting, they have a name and a roster for collecting names. All the agents meet in O(logn) time, gathering other agent's names in their roster. The agent's roster's rank is its name's order in the roster.