MP1 Performance Study

Comparison of 3 gossip protocols and number of nodes vs. convergence time

By Matt Scott, Roland Afzelius, and Liam Potts

Methodology:

The program was written to dynamically set the number of nodes that it will use for a run. For each number of nodes - two through nine - Push, Pull, and Push-Pull protocol was run twenty times each and the average number of total runs was recorded. The number of cycles to terminate was chosen as the dependent variable instead of literal time, so the result would not be influenced by the wait time (because the wait time is not relevant to the main question).

Packages Used:

fmt: used for printing to the terminal
math/rand: used for generating random numbers used in calling random nodes
sync/atomic: sync in general was used for aiding with concurrency (waitgroups), atomic was used for atomically incrementing global counters to ensure we properly tracked operations
time: used for creating delays to aid concurrency (allows goroutines to catch up to one another)

Procedures:

Although a central repository is used, this was to simplify the reusability of functions. Each node carries its own copy of the channels used for contacting nodes, and none know if the nodes they are contacting are infected or not, so they are all operating independently. The channel maps were implemented in a way that the key associated with a map is the name of the node that receives messages through it.

Push

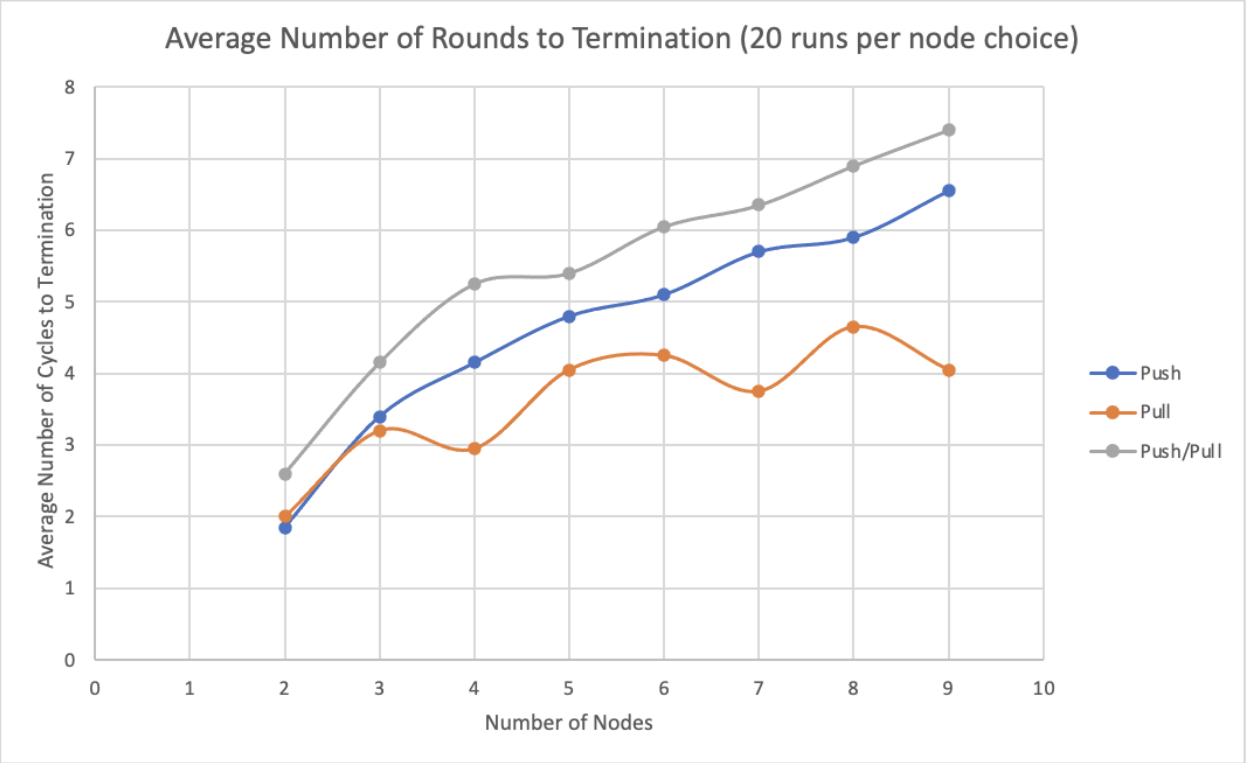
Push protocol is implemented by having infected nodes push their message through the contactChannel map. After a brief delay to maintain synchronicity, all nodes check their channel for messages, and uninfected nodes update their message if they received a new one.

Pull

Pull protocol is implemented by having uninfected nodes send their name through the nameChannel map to a random node. Nodes then check their nameChannel for requests, and if they have received any, send their message through the contactChannel using the name received as the key in the contactChannel map. Finally, all nodes check their contactChannel for messages and uninfected nodes update if they receive a new message.

Push/Pull

Push/pull protocol runs push protocol until half of the nodes in system are infected, then switches to pull protocol.



Discussion:

The data graphed above, supports the hypothesis that for gossip protocols with 2-9 nodes, Push-Pull gossip will take on average more runs than push gossip, which takes more runs to termination than pull gossip.

This is consistent with the time complexities of gossip that were shown in class. Push time complexity is O(N* logN) because every cycle requires each node to send at least one request (which is where the N comes from) and logN (base 2) cycles are needed to update all the nodes.

Pull time complexity is O(log log N) due to the quadratic convergence phase (as discussed in the class slides). Since O(N*logN) is slower than O(log log N), it makes sense that the data gathered shows that the pull method terminates faster on average.

Finally, as for Push/Pull it would make sense if the average number of cycles before termination was in between the push and pull methods, since it is a combination of both. The fact that Push/Pull took the longest is not easy to understand, but since Push/Pull is Push first, it makes sense that it is relatively parallel to the Push line.

References

Class slides referred to:
https://drive.google.com/file/d/1leb1smC1cuu03wu2F_8BEmIy7sZX58C5/view