Linsey Szabo
Computer Networks
Fall 2023

Project 2 Design Document

Keep Alive Message Implementation

Every 2 seconds, each node transmits a keep alive message to its neighbors. Each message follows the following format: "keepalive.<name of sending node>.<UUID of sending node>.<host of sending node>.<port of sending node>.<metric between sending node and specific neighbor>." By starting each keep alive message with "keepalive" this makes identification of keep alive's very simple on the receiving end. Additionally, I chose to send all this information in my keep alive's for "automatic" detection when the addneighbor command is invoked. While link state advertisements contain node names and metrics, they do not contain more detailed information like host and port. This is fine for making a map, but neighbors do need that extra information for the neighbors command. Therefore, I used keep alive's to send this necessary information.

On the receiving end, a keep alive message is detected by the "keepalive" header. After parsing it, that neighbor's last keep alive time is recorded. If a node receives a keep alive message from a node it didn't already have as one of its peers, that peer is added as a neighbor. Due to the details sent in my keep alive's, this node would then have the necessary information for the neighbors command. Lastly, all nodes use the last keep alive time and the current time to check for dead nodes. If a node's last keep alive time is more than 15 seconds ago, it is considered dead and is then removed from that node's peers.

LSA Implementation

LSA's are sent every 2 seconds. Each messages follows the following format: "LSA.<name of sending node>.< UUID of sending node>.<sending node's neighbor 1>:<metric 1>, …,< sending node's neighbor n>:<metric n>.<sequence number>." By starting each LSA with "LSA" it makes parsing each on the receiving end. By including the UUID, it enables each node to construct a map of each UUID to each name for the map command to work correctly. Additionally, each node keeps track of a dictionary that maps UUID's to a received LSA message and corresponding sequence number. Therefore, upon receiving an LSA, if the sequence number is less than or equal to the sequence number stored, it's discarded. Otherwise, the LSA and sequence number are recorded in this dictionary.

To send the LSA's, I construct my own LSA based on my current neighbors, and I also loop through the dictionary that has stored received LSA's and their sequence numbers. These are then sent to all neighbors.

Libraries

- Threading: used to simultaneous transmit and receive messages from other nodes, version 3.7, https://docs.python.org/3/library/threading.html

- Time: used to keep track of last received keep alive, version 3.6, https://docs.python.org/3/library/time.html
- Heapq: used to implement a priority queue for Dijkstra’s, version 3.5, https://docs.python.org/3/library/heapq.html