Progress Report
Week of August 3, 2020

Edris Qarghah

Enrico Fermi Institute
University of Chicago

Weekly Goals

- Finalize tweaking of network distributions
- Create means of visualizing individual path changes
- Observe impacts of breaking an edge:
  - Use traceroutes to identify path changes
  - Identify edges that are no longer being used
  - Infer identity of broken edge
  - Display impact on latency/packetloss
- Observe impact of breaking multiple edges

Daily Log

Monday, August 3

- Network Analytics Discussion meeting.
- Made temporary histograms to confirm distribution matching.
  - Loaded histogram data to dataframes and scaled.
  - Tried alternative means of calculating latency (using the distribution).
  - Adjust packet-loss to account for the probable number of multiplications.
  - Added path hops, latency and packetloss to path lengths function.
  - Updated OWD distribution to account for offset.
- Finished distribution matching (got Ilija's sign-off).
- Fixed issue with lists containing one tuple being loaded as just a tuple.

Tuesday, August 4

- Added function to change edge values (i.e., latency, packetloss).
- Manually changed latency and noted change in shortest paths.
- Added shortest paths dictionary to the graph properties.
  - Encountered issues saving and loading shortest path dictionaries (GML does not like taking non-strings as dictionary keys).
- Created means of converting a shortest path list to a list of edges.
- Created preliminary means of getting changes to paths.
- Added capacity to only draw paths connecting specific PS pairs.

Wednesday, August 5

- Added helper functions to address issue with non-string keys (i.e., pair to string, string to pair).
- Improved graph path change function to account for bi-directionality of edges.
- Met with Ilija about results of killing individual edges. Suggestions:
  - Kill edges entirely (rather than simply increasing latency)
  - Systematically remove each individual edge in a graph (this may break the graph into multiple components).
- Prepared slide for OSG update meeting.
- Read about time clock sync issues in blockchain and distributed systems.
- Ensured that path comparisons still functioned if path could not be completed after edge removal.

Thursday, August 6

- Reviewed OSG update meeting presentation.
- SAND Informal Working Meeting.
  - Learned more about what the others are doing:
    - Tommy (UMich undergrad) is working on Kibana dashboards for network monitoring
    - Manjari (UMich undergrad) is working on a UI for network topology analysis (highly relevant to my work)
    - Petya (U Plovdiv PhD candidate) is working on a Plotly Dash ML web app to provide site summaries.
  - Tommy helped Petya with data pulling issues in ElasticSearch, which was instructive (she was pulling a set of IPs first as a source, then as a destination but was mysteriously getting the same result both ways).
- Wrote functions to systematically remove each edge in a network then tally the resulting path changes.
- Created histograms to display edge ambiguity.
- Modified packetloss histogram to allow comparison of multiple iterations of graphs.
- Updated hops histogram to actually be a histogram!
- Patched issue with histogram obscuring original distribution by setting alpha to .5. A better solution needs to be found.

Friday, August 7

- Finished reports.
- Caught an exception related to how shortest paths were being stored.
- When getting path changes, fixed how edges were being removed from the changed_edges dictionary.
- Updated graph drawing:
  - Adjusted perfsonar node color to make them easier to see (blue instead of brown).
  - Changed the default edge label to None (easier to see paths/colors that way).
- Add naming and autosaving of histograms.
- Updated slides following meeting to clear up ambiguity and structure them better.
- Added details about the process for determining deleted edges:
  1. Determine shortest paths between all PerfSONAR (PS) nodes in G.
  2. Delete an edge and save the network as a new graph, H.
  3. Determine shortest paths between PS nodes in H.
  4. Compare shortest paths between graphs:
     (a) Record edges in each G path that aren't in the corresponding H path (i.e., edges that are potentially the removed edge).
     (b) Confirm those edges were removed entirely from H (i.e., they are not on some other path in H).
     (c) Determine how many different paths each edge was removed from.
        i. It's likely that the edge removed from the most paths is the deleted edge.
        ii. If multiple edges were removed an equal number of times, there is ambiguity as to which was deleted.
  5. Repeat 2-4 for every edge in G.
- Caught and corrected issue with edge ambiguity (wasn't fully excluding edges that appear elsewhere in the output graph). This involved a significant rewrite of get_path_changes.
- Fixed font size on histograms.

Achievements

I learned about...

- The "clock problem" for blockchain/"timechain" and distributed systems.
- Other students on the SAND team and how their work relates:
  - Tommy (UMich undergrad) is working on Kibana dashboards for network monitoring
  - Manjari (UMich undergrad) is working on a UI for network topology analysis (highly relevant to my work)
  - Petya (U Plovdiv PhD candidate) is working on a Plotly Dash ML web app to provide site summaries.

I created...

- Means of identifying changed paths:
  - Edges difference between old/new trace-routes.
  - Accounting for lack of edge directionality.
- Means of narrowing down candidate edges:
  - Discarding edges that remain on other paths.
  - Discarding edges not removed from all paths (only works if we know only one. edge)
- Means of visualizing changes:
  - Graphs highlighting only relevant paths.
  - Histograms of candidate edges and impact to distributions.

Roadblocks

Questions

- How can I narrow down the number of potentially impacted edges?
- Is there a way to leverage edge adjacency to help make these determinations?

Problems

- Edges are indexed as tuples, making their use as keys in dictionaries difficult (ordering and saving as strings).
- The toy network has ambiguity with even very limited curated data, which will only be worse with real data.

Challenges

- Highlighting changes to a graph without things getting too cumbersome.
- Developing ways to summarize the impact of changes over many iterations that were intuitive/easy to understand.

Plans for Next Week

- Toy Network
  - Further static network tomography:
    - Dead vs. throttled edges
    - Adjusting for change to variable number of edges
    - Add bandwidth and measure bandwidth impact
    - Edge removal heatmap
  - Simulate network activity
  - Measure activity from specific nodes/generate Time Series
  - Develop tools for tomography and anomaly detection
- Real Network
  - Take snapshots of traceroutes at various times for comparison.
  - Try to determine whether traceroute variation can be used in the same manner as in toy network to identify dead/ailing edges.