Progress Report
Week of July 13, 2020

Edris Qarghah

Enrico Fermi Institute
University of Chicago

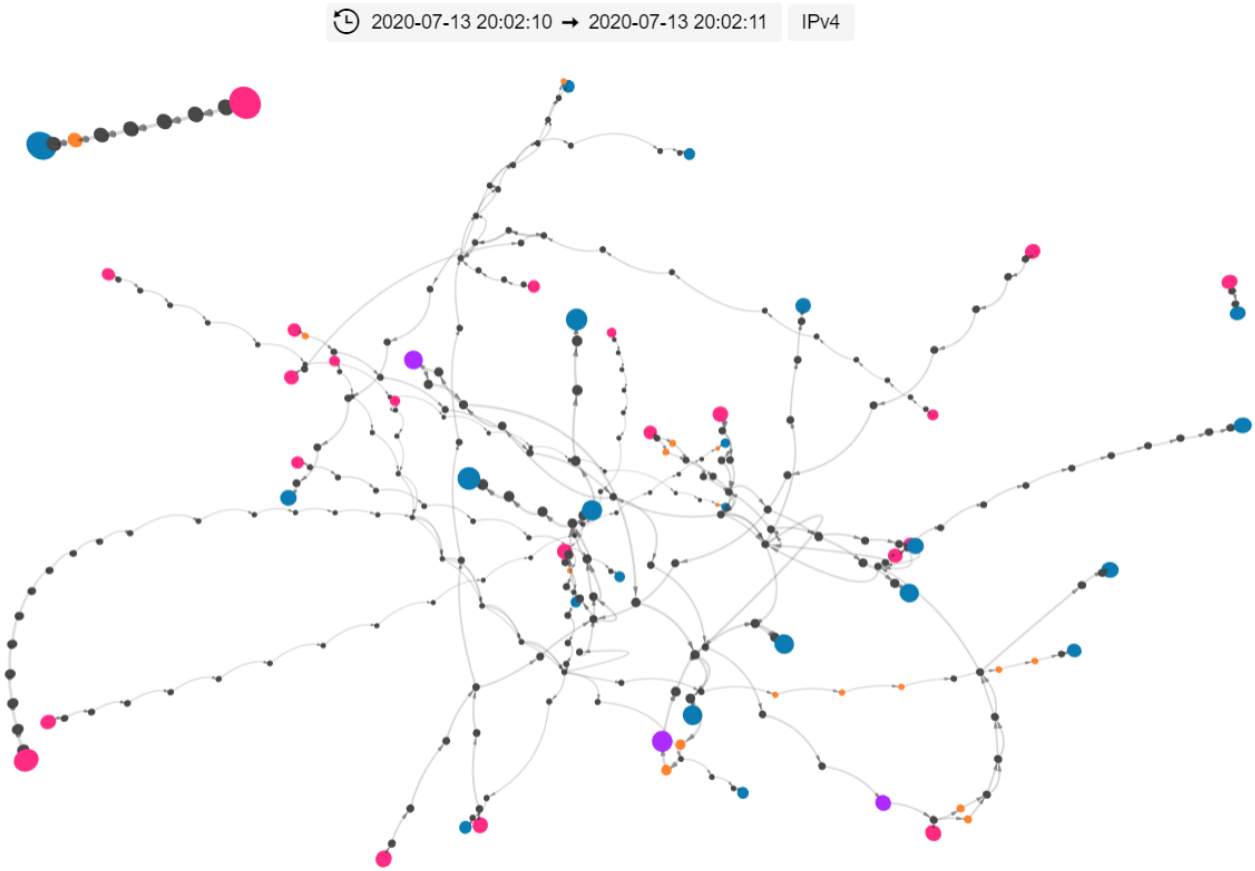
Weekly Goals

- Toy Network
  - Create a more robust network generation process (e.g., ensuring MST).
  - Simulate network activity.
  - Measure activity from specific nodes and generate a resulting time series.
  - Develop tools for tomography and anomaly detection.
  - Improve visualization of networks.
- Explore real network to understand how to better model it.
  - Hub and spoke approach to network generation?

Daily Log

Monday, July 13

- Network Analytics Discussion meeting.
  - Met Shawn, Petya, Soundar and Tommy and learned a bit about what each is working on (apparently Painless is NOT painless).
  - Shared my progress from last week.
  - Added my weekly progress report presentation to SAND drive (meeting notes and personal folders) and will do so in the future.
  - Discussed the possibility of having multiple classes of nodes with differing levels of connectedness (e.g., hubs that have many connections and destinations that have only a few).
- Explored PerfSonar visualization for a better sense of what a network actually looks like.



- Ilija provided a helpful overview of how packet tracing works and I read up on it to supplement/solidify my understanding:
  - TCP packets have a Time To Live (TTL) which determines how many hops they can make before being discarded. This prevents packets from travelling the internet forever, as each hop decrements the TTL counter by 1.
  - When a TTL counter reaches zero, it is dropped and a message is sent back to the source to indicate where that happened.
  - Trace Routes leverage this by sending repeated packets to the same destination with incrementally higher TTL (i.e., the first one dies after the first hop, the second after the second hop, etc.). As each time TTL expires an ICMP "time exceeded" message is sent back with the info of where it happened, we can map out nodes that are one, two, three, etc. hops away, until we arrive at the destination.
  - Notes:
    * These are separate packets, so we are not guaranteed that the third hop that the TTL=3 packet died at is the same as the third hop made by the TTL=4 packet. As such, the path drawn by a trace route may depict hops between nodes that are not actually connected to one another!
    * It's possible that a node is visited twice on a trace-route, which is called a self-loop. This could happen, for example the TTL=6 packet took a different route and died at the same node as TTL=3, or it could be that there was a routing error that really meant that it passed through the node at TTL=3 but the fifth node decided that node was the best route and sent it back there again.
    * Some nodes (i.e., those behind firewalls at CERN and the like) will not send return messages when dropping a packet, but may still forward a packet that still has TTL. The dropped packets that didn't receive a response are orange in the PerfSonar visualization.
- Discussed with Ilija the shape of the network (i.e., some bottleneck links like the transatlantic one, some highly connected regions, like Europe).
- Reviewed Suchant's work and diagrams again. He was using ElasticSearch to explore the paths that were being taken on the network and understand them, so that he could identify paths that were performing poorly and flag them.

Tuesday, July 14

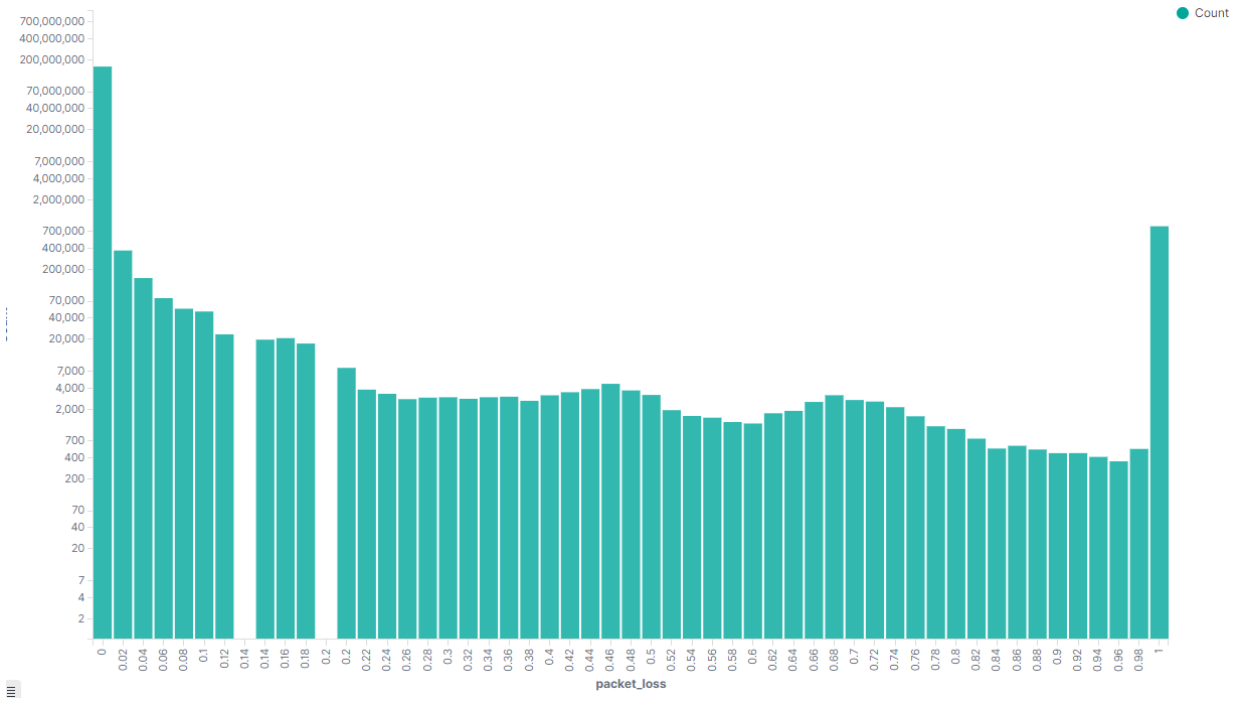
- Finalized and pushed refactoring from previous day.
- Spoke with Ilija about packet loss and read about it further to solidify my understanding:
  - Packet loss is a measure of link quality that can be diminished for a wide variety of reasons (e.g., induction errors, temperature swing, dirty fiber).
  - When a packet is lost, the bandwidth of that route is temporarily reduced by 50%. With stable performance, it is increased back again in 5% increments, which can take a long time to get back up to full capacity.
  - A link with 1% packet loss is effectively dead, unless the latency is VERY small. Data is buffered on both sides of the communication and a lost packet means that the packet needs to be resent, messing with the buffering (which is FILO).
  - *Big Takeaway:* The detrimental impact of packet loss is proportional to the latency of the connection.
- Created SSH key and profile for CI-connect and requested access to slate training.
- Uploaded passport photo to Workday.
- Had an unproductive diversion trying to update the Jupyter lab python kernel on the ML platform. Deemed the pay-off of having 3.6+ not worth sinking additional time into it.
- Learned about k-connected graphs and ensured that my graphs are at least 1-connected.
- Developed a means of creating "hub" nodes recursively to serve as a backbone for the network.

Wednesday, July 15

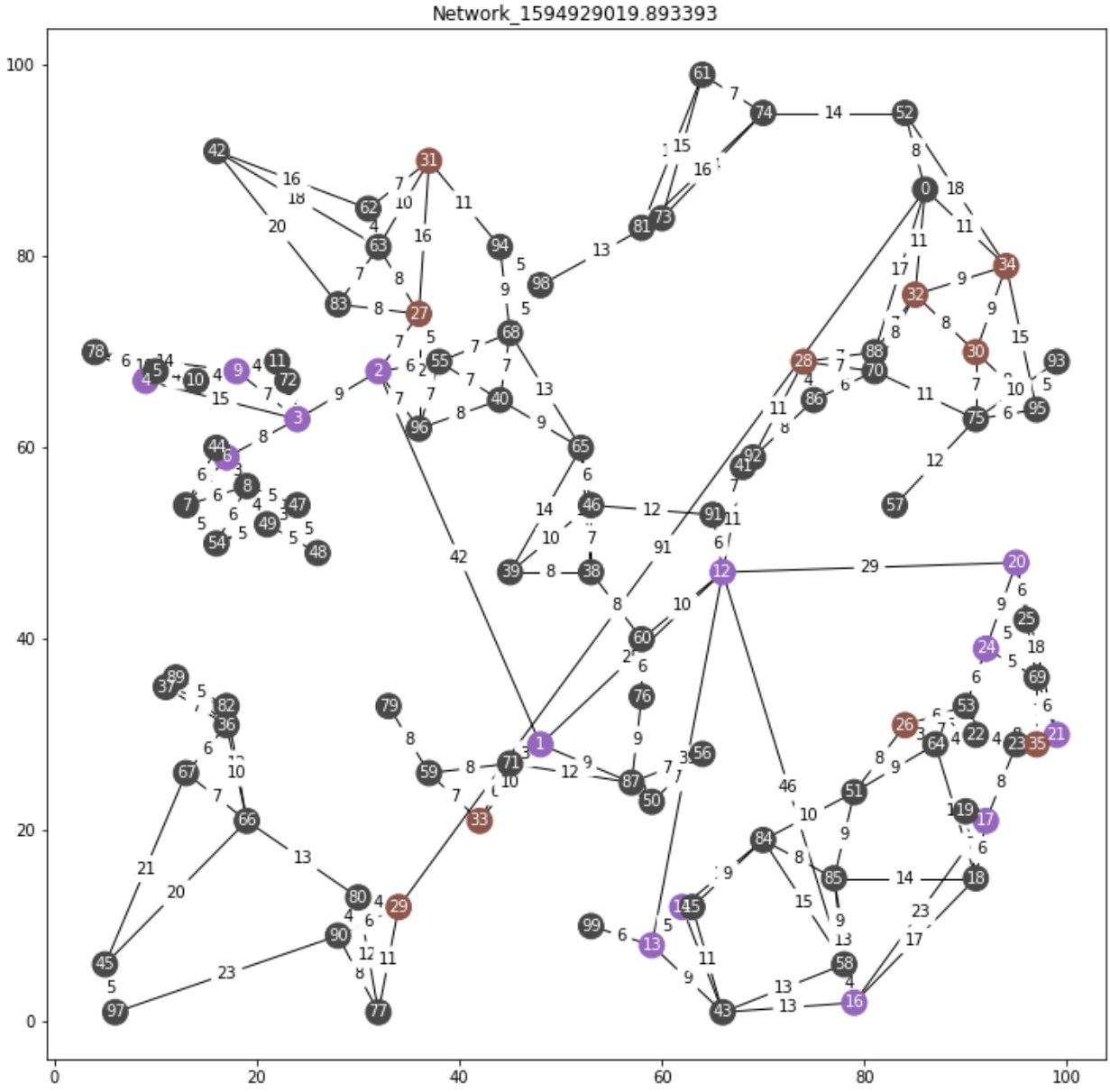
- Accidentally lost all work from previous day because I overwrote ML instance. Created a new one, rewrote and enhanced code, including:
  - Ensuring k-connectivity.
  - Creating hub nodes recursively.
  - Allowing parameters to be set for network generation.
  - Generating names and saving all graphs that are generated.
- Installed Pandoc to try to enable using Jupyter lab as documentation but was not able to get latex to render.
- Attended SLATE tutorial.

Thursday, July 16

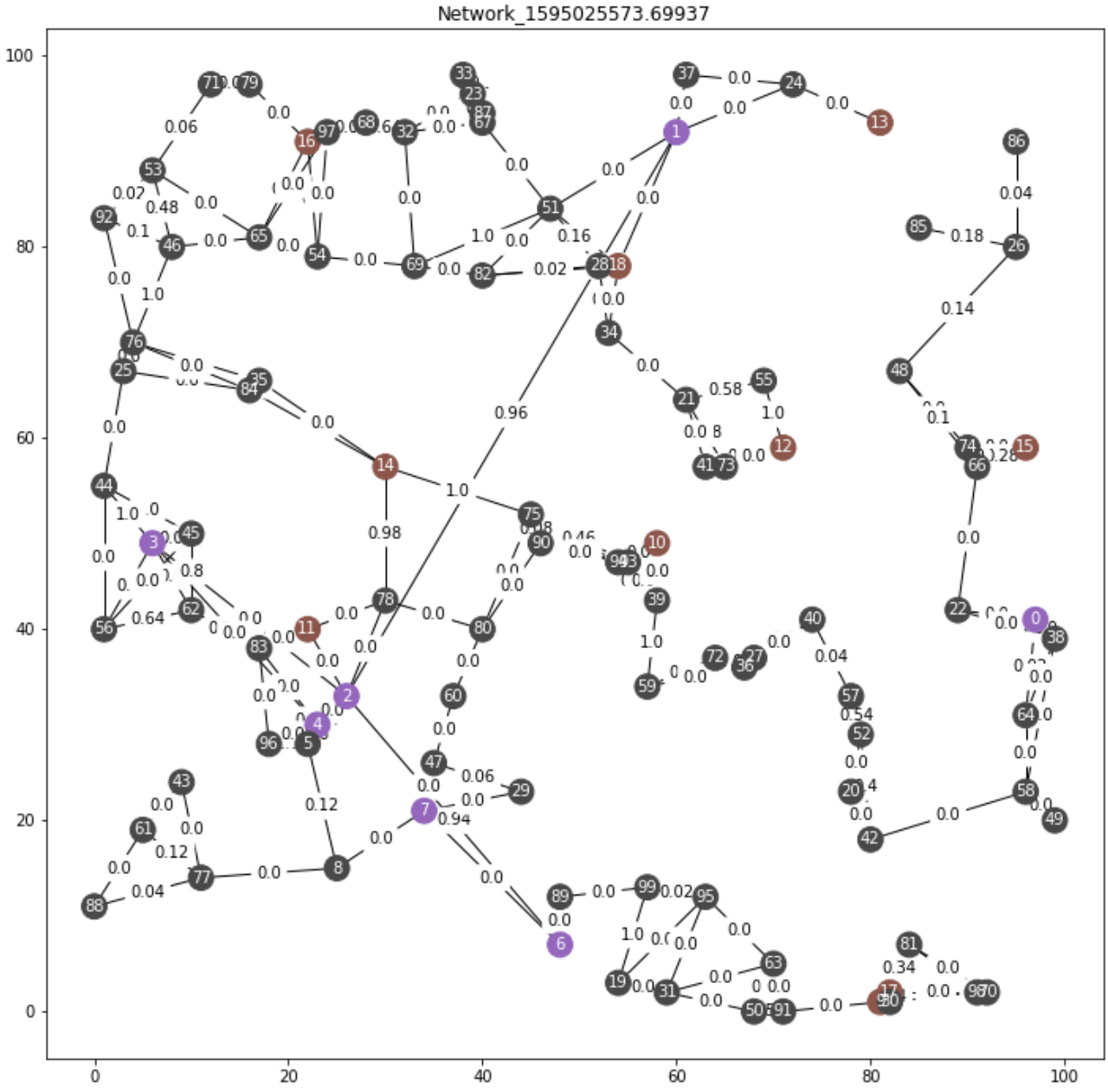
- Discussed packetloss distribution with Ilija:
  - Packets are sent 10 times a second, for a total of 600 over a minute to determine packetloss rates.
  - Ilija showed me to access useful data visualizations in Kibana (pictured below, log of count of packetloss rate (i.e., count of times in 30 days that 2% of packets were lost).



  - The distribution was heavily skewed toward 0 with a spike at 1, due to dead connections.
- Exported packetloss data, cleaned data and took the square root of the count in each band before using it to create a distribution (the count was too heavily skewed for an effect to be likely/visible on the scale we're working with, but the log flattened things out too much; the square root was a decent middle ground).
- Randomly selected packetloss rate for each edge based on the distribution derived from the Kibana data.
- Improved network graphs:
  - Added network name, so that interesting graphs can be pulled up again (as they are being saved.
  - Added color coding for PerfSonar, Hub nodes, sources and destinations (the last two haven't been implemented on the network side yet).

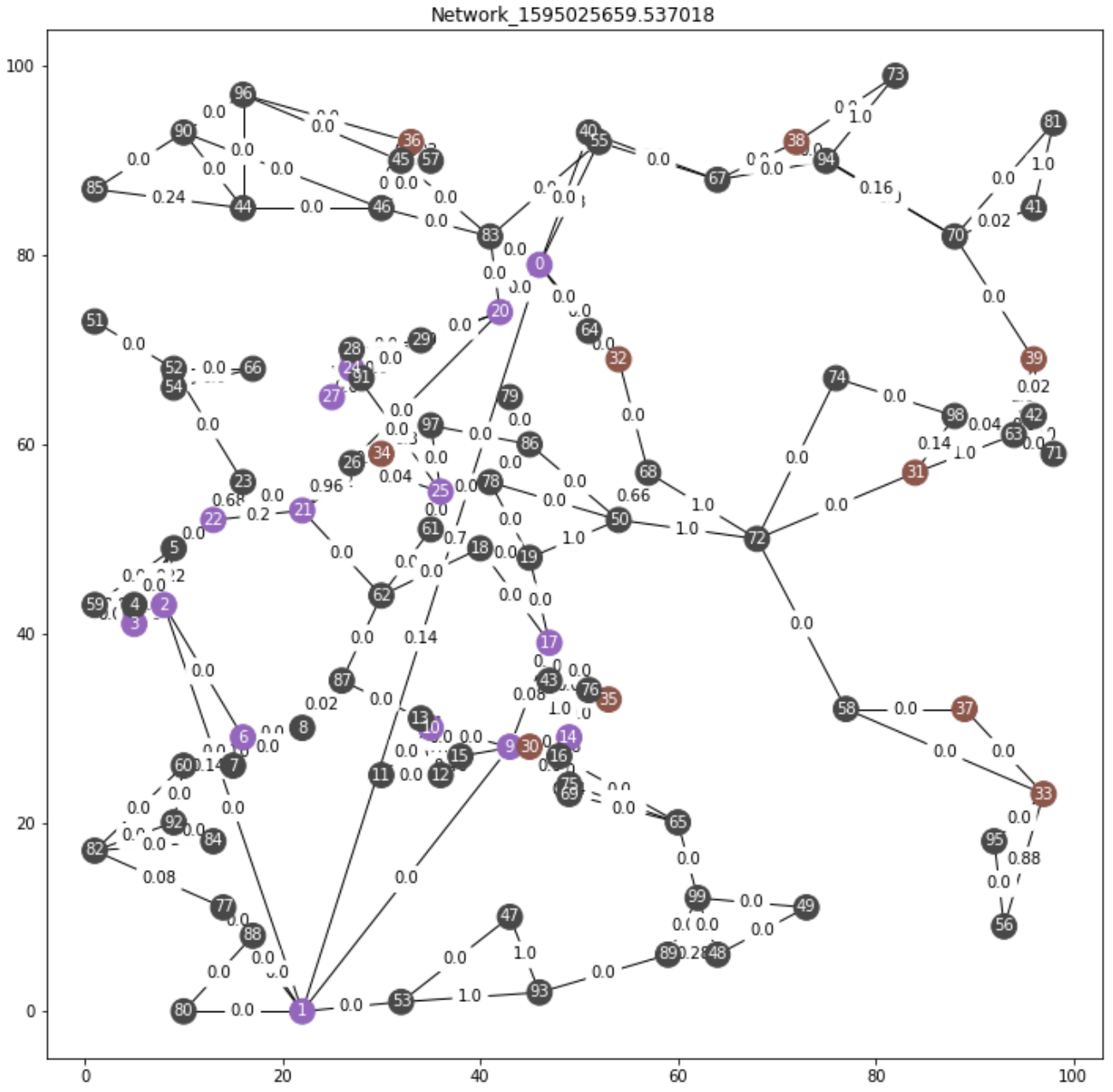


– Tested displaying packetloss rate instead of latency of edges.



Friday, July 17

- Fixed bug where first hub node wasn't being connected to it's children.



- Created a list of PerfSonar nodes and found the shortest distance between each pair, based on latency.
- Created progress report deck and updated this document.

Achievements

I learned about...

- Slate and its use.
- Packet tracing and related complications/considerations.
- Packet loss and its causes.
- Visualizations in Kibana.
- K-connected graphs.
- NetworkX's functionality around:
  - Node coloring.

  - Finding shortest path.
  - K-connected-edge augmentation.

I created...

- More robust networks:
  - Generation refactored into separate script and parameterized.
  - Always k-connected (configurable).
  - Nodes that serve as hubs and branch (forming a network backbone) recursively to a specified depth.
  - PerfSonar nodes (configurable).
- Better visualizations of networks:
  - Color coded nodes.
  - Figures named after graph, to make referencing/exploring graphs easier in the future.
- Weekly progress report slides and document (I know this is the same every week, but it is a pretty significant chunk of time and work that I am proud of).

Roadblocks

Questions

- The hub system seems to work well for ensuring there are some longer connections. What would be some other good ways to create clustering while maintaining broader connectivity?
- Should higher hubs (i.e., hubs connecting other hubs) be disconnected from lesser nodes entirely?
- How connected should nodes of various depths be?
- How can I ensure that PerfSonar nodes are well distributed and not too centrally located?
- How do I combine latency and packetloss edge data to get meaningful tomography?

Problems

- I accidentally overwrote my ML instance.
- I was unable to get Jupyter notebook updated to 3.6+ on my ML instance.
- I was unable to get LaTex to render from Jupyter notebook.

Challenges

- Depicting network branching.
- Having graph attributes reflected on the figures drawn (e.g., node coloring).
- More broadly, using built-in functions (i.e., shortest path algorithms) on non-standard/custom criteria.

Plans for Next Week

- Learn about tomography strategies.
- Develop tomography of static network (e.g., with latency/packet loss)
- Simulate network activity.
- Generate time series based on that activity.
- Learn how this connects to anomaly detection.
- Explore how the model can be applied to real network data.