Progress Report
Week of August 10, 2020

Edris Qarghah

MANIAC LAB

Enrico Fermi Institute
University of Chicago

Weekly Goals

- Real Network
  - Learn about ElasticSearch, the structure of our data and how to query it.
  - Dive into Sushant's code to see what I can learn and re-use.
  - Determine unique pairs of PerfSonar nodes that are actively transmitting to one another.
  - 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.
- Toy Network (Postponed)
  - 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

Daily Log

Monday, August 10

- Network Analytics Discussion meeting.
  - Manjari discussed issues with flask app. Suggestion was to develop from Windows.
  - Petya had laptop issues. Worked on PS_Dash with problematic hosts. Issues with visualizing data with chart that was difficult to configure. Was creating too many instances of base class, working to create a singleton.
  - Soundar was working on the ingester and tracking high water marks in RabbitMQ (message count was going up but wasn't reflected in the interface). Plans to include timestamp logs for RMQ bus message count.
  - Tommy worked on Dashboard visualization stuff in Kibana. Dropdown selection menus were not populating with all source and destination host IPs (thanks to Ilija). Throughput
  - I discussed my work from last week and Shawn added the following thoughts:
    - Router 1 to 2 displays an IP address based on the end of cable connection (router 2). Going Router 2 to 1 does the opposite, so it can be hard to determine when a route is the same just being traversed from a different direction. To crack this, we need to know how people name routers.
    - How do we represent the network based on our limited sampling of it? We need an abstraction of the network that we can reason about, much like the Toy Network.
    - Perhaps we could create a database of hops as vectors. Each hop could have metadata and real data. Time dependency makes this complicated.
- Read about Nada's work synchronizing clocks.
- Spoke with Ilija about working with the real data and pulling Sushant's work:
  - Pull six hours of trace data and see how stable paths are. (One or two paths have almost..)
  - Review previous work (path is stable if all traffic..)
  - Stars/Firewall blocking ICMB packets
  - Distribution of number of hops per path.
  - Unique sources and destinations
  - Clean subset of data (even if that means tossing out half)

Tuesday, August 11

- Review Sushant's code:
  - Ran code connecting to ES and pulling basic statistics.
  - Read through queries to try to understand how they were working.
  - Tried to parse project structure to understand the function of various modules.
  - Read up on Neo4j because that was a component I couldn't get working.
- Met with Ilija to discuss Sushant's work:
  - There are some changes to elasticsearch:
    - search first parameter must be "index="
    - "true" now True
  - Elastic search querying language structure:
    - Items in 'bool' need to be true in order to be pulled
    - must = and, should = or
    - Nested aggregation under "aggs"
  - Data pruning considerations:
    - dest_production has to be true (we're not interested in non-production stuff)
    - If there are < 1000 measurements, it is not sufficiently significant to use
    - If n_hops <= 1, something funky is going on and those hosts should be removed.
    - Remove "looping" paths.
    - Paths are stable if they have the same SHA1; pull only stable ones!
  - trace_derived (ps_trace but trimmed down)
    - Updated every one minute
    - Aggregate data form January of 2018 to present
    - Fast way to get unique sources and destinations
  - For each hop, how many paths does it belong to. Which hop changed and what paths were effected.
  - ps_trace: "hops" array
  - ", *" means ICMB packet was blocked
  - asns: administrative domain of hops (we have a mapping of these to actual institutions)
  - "path_complete=true" means no stars
  - rtts (relay time): time it took from source to first, second, third, etc. (remember: different packets per endpoint)
  - Looping: IP appears more than once
  - Make heatmap: sources on one axis, dest on the other, flag everything that was removed.
  - route-sha1: String from all hops and makes sha1. Unique variable for each path, makes it fast to find the same path (use it to filter so only different/unique paths show).
  - src and dest can be ipv4 and 6 for the same locations (dest_host to src_host).
- Updated Sushant's code to make use of the insights from meeting with Ilija.

Wednesday, August 12

- Continued working on Sushant's code, but was uncertain what to do with it (as I wasn't sure if I could commit it back to Analytics or whether I should fork it).
- Created my own independent network analytics project.
  - Learned about IPython extensions and autoreload.
  - Created a connection to elasticsearch.
- Day was cut short due to an outage.

Thursday, August 13

- Recovered work from the previous day.
- Developed rudimentary project structure for `network_analytics`
- SAND Working Meeting
  - Tommy shared some visualizations he worked on that may be relevant to my work. It allows exploring a connection between src-dest pairs and the routes between them.
- Wrote a query to find unique pairs via double aggregation (on src then on dest). This was problematic as it quickly hit the bucket limit at only 60 srcs and 59 dests (when there are 386 of them, roughly).
- Read about Query DSL.
- Read about `copy_to` as a means of addressing double aggregation.
- Learned about terms queries which I could use if I transform the output of a query to match the format.
- Wrote a function to convert python lists of strings into ES query-friendly string describing a list (e.g., for use with `terms`).

Friday, August 14

- Finished reports.
- Discussed query issues and limitations with Tommy.
  - He suggested looking into script fields to address the double aggregation issue (i.e., make a single field that contains both source and destination).
  - Provided an example of a scripted field already written for `ps_trace`.
- Learned about the Painless scripting language for ES.
- Spoke with Ilija about week's work and he had the following feedback:
  - No need to do aggregation on `trace_derived`: it's already aggregated!
  - Searching has limits on the number of results it can return (10,000?); Scanning is a different approach to getting data that doesn't have those limitations (can take days).

Achievements

I learned about...

- IPython extensions and autoreload.
- Query DSL.
  - Query vs Aggs
  - Bool queries and use of should/must vs filter
  - Score
  - Term, terms and ranges
  - `copy_to` and script fields
  - Nested aggregation and bucket limits
- Our data in Kibana.
  - The various fields in `ps_trace` and `trace_derived`
  - There are 386 PS nodes that serve as both src and dest, if we filter out those that are only connected by one or fewer hops and those that do not have at least 1000 records.

I created...

- A `network_analytics` project to contain my work with the real network:
  - Created a rudimentary structure for the project.
  - Created a `main.ipynb` file to serve as my primary testing ground.
  - Wrote queries to get data about PS node pairs and the paths connecting them.

Roadblocks

Questions

- How do I make aggregate columns (e.g., `copy_to`) in ElasticSearch using Python?
- How do I get data in aggregates other than counts?
- What does all the seemingly superfluous stuff returned in queries mean (e.g., "_shards", the first set of "hits" when using both "query" and "aggs")?

Problems

- Currently I'm ruling out individual entries that don't meet my criteria (i.e., low hop average, low record count), but I really need to be doing this with the aggregate data for a path; I'm just not sure how to.

Challenges

- Trying to parlay the queries I had access to (either from Sushant or Inspect) into queries of my own design.
- There were times when there must have been an issue with my ElasticSearch syntax, but it ran just fine, but would produce no results. I largely switched to using "filter" instead of "must" (which apparently is more efficient anyway, because it doesn't calculate a score for results).

Plans for Next Week

- Finish determining what stable paths exist:
  - Improve method of finding PS pairs.
  - Rule out paths that have multiple route-sha1s
- Convert hop data into discrete edges.
  - Determine alternative names for edges (whether based on directionality or IPV4 vs IPV6).
- Determine how many paths a specific edge is on.
- Identify changes from expected paths:
  - Determine if statistics are meaningfully different (e.g., significantly increased packetloss).
  - Identify edges that differ between iterations.