proximity service

- Main idea
  - provide “glue” between experiments and sonar topology
  - mainly map sonars to storages and vice versa
  - determine existing tools/technologies that can be used
  - 3 approaches
    - Site-based – join based on common mapping to GOCDB/OIM sites
      - Already works for ATLAS – AGIS has such mapping automatically
      - There are caveats – each experiment has different site topology – each has different mapping to GOCDB/OIM – not all cases are one to one
    - GeoIP – join based on geographical distance
    - Traceroutes – join based on network distance (hops)
- Shawn McKee, Saul Youssef and M.B. are working on this

GeoIP

- The main idea: Determine geographical location for hosts from their IPs, use geographical distance (Great-circle, Vincenty) to calculate proximity
- Example
  - psum02.aglt2.org (perfSONAR) – long: -83.736719/lat: 42.276845
  - earth.crc.nd.edu (storage) – long: -86.2501/lat: 41.7007
  - Vincenty's distance = 217.87km
- Determining location
  - perfSONARs contain their geo location as part of toolkit's information
  - storages can be mapped using GeoIP service - DB which maps subnets to geo locations
    - GeoLite-City – open source – accuracy at city level, falls back to country level
    - Commercial provides, e.g. MaxMind– usually have better accuracy and provide more information (estimated accuracy/precision)
  - computed distance can be improved by subnet match – hosts on the same subnet (24) -> 0 distance

GeoIP prototype

- Prototype available at proximity.cern.ch
- Uses a custom GeoIP DB build from various sources
  - Contains only ATLAS (from AGIS) and CMS (from PhEDEx) storages
- Provides API/JSON – you can get nearest storages for a sonar and vice versa
  - Motivated by Shoal, returns n nearest records with geo information
  - Adds “origin” at the end
- You can tune
  - number of records returned (via count=7)
  - units (via unit=miles)
  - subnet match (via subnet=24)
- Examples
  - http://proximity.cern.ch/api/0.3/geoip/nearest?sonar=psum02.aglt2.org&count=5
  - http://proximity.cern.ch/api/0.3/geoip/nearest?se=lapp-se01.in2p3.fr&count=10


```
{
  "0": {
    "distance": 0,
    "hostname": "head01.aglt2.org",
    "geo": {
      "city": "Ann Arbor",
      "ip": "192.41.230.44",
      "time_zone": "America/Detroit",
      "long": -83.7113,
      "country_name": "United States",
      "host": "head01.aglt2.org",
      "country_code": "US",
      "lat": 42.3241,
      "accuracy": 3
    }
  },
  "1": {
    "distance": 217.8719075736171,
    "hostname": "earth.crc.nd.edu",
    "geo": {
      "city": "Notre Dame",
      "ip": "129.74.85.10",
      "time_zone": "America/Indiana/Indianapolis",
      "long": -86.2501,
      "country_name": "United States",
      "host": "earth.crc.nd.edu",
      "country_code": "US",
      "lat": 41.7007,
      "accuracy": 3
    }
```

GeoIP summary

- Prototype algorithms and database are easy to port and can be integrated in the already existing tools (within experiment)
  - Few lines of codes, most of the effort spent in building up the DB
- Prototype available for testing/experimental purposes only (not a production service) – we can decide to evolve it to a production service if there is interest
- Establishing WLCG-wide GeoIP service shouldn't be too complicated and can also have other benefits:
  - We already have many existing sources for this – GOCDB site locations, Shoal, sonars, LHCONE/LHCOPN routing – we can combine them to get a solid baseline
  - We could run a crowdsourcing campaign to fine tune the baseline (get additional subnets mapped)
  - Once established, DB would only require minor updates

Traces

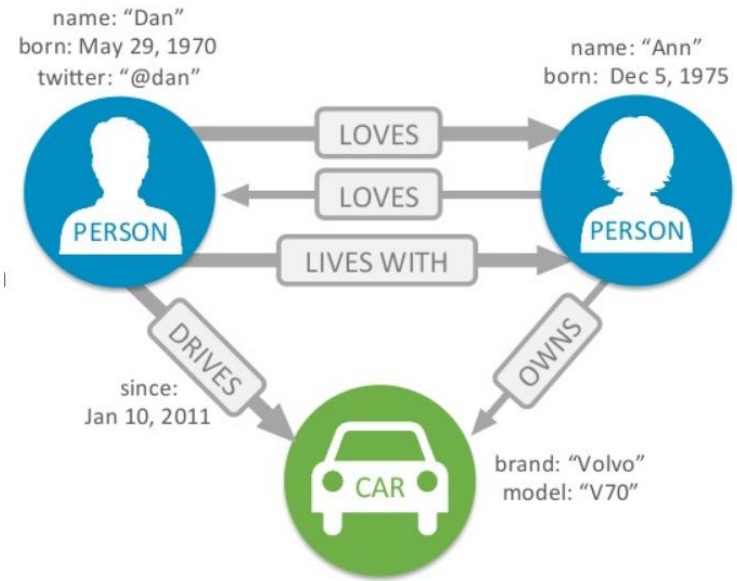
- The main idea: Use traceroutes to create a graph model of the network, use graph traversal algorithms to determine proximity of nodes

- Graph model of the network has potential for other use cases such as path analysis, correlations with other metrics, auto-detection of weak links, visual debugging

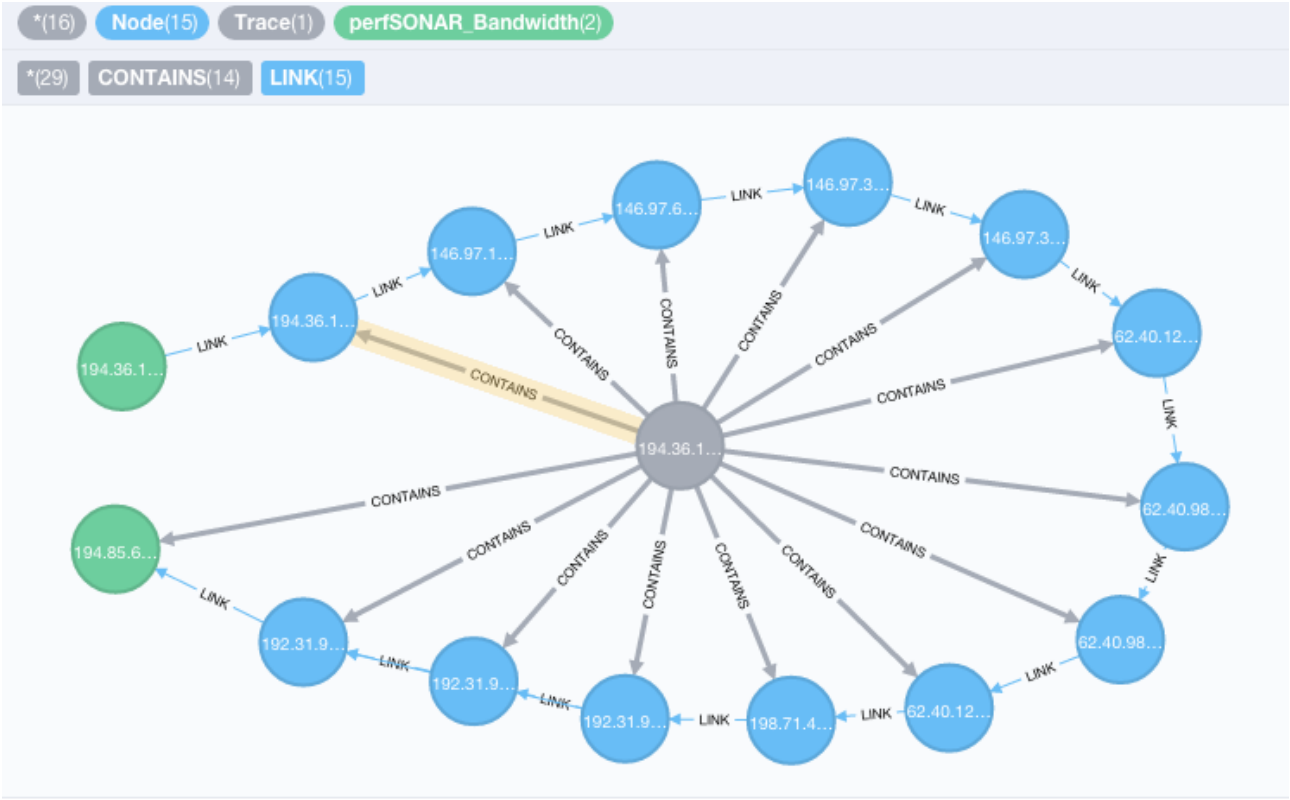
- perfSONAR can be used as the source for the traceroutes
  - Currently runs full mesh traceroutes between 110 sonars (12K traces/hour), ideal for baseline, other sources can be added since traceroute/tracepath
- Graph databases such as Neo4j can be used to store and query the model
  - Designed to support graphs with billions of nodes and relationships
  - Runs 4M+ relationship traversals/second
  - Can import 100K-1M records/second

Graph database concepts

- Graph database stores data in a graph, graph contains nodes and relationships
- Nodes
  - The objects in the graph
  - Can have name-value properties
  - Can have labels (types)
- Relationships
  - Relate nodes by type and direction
  - Can have name-value properties
- Graph query language (cypher), e.g.
  - match (x:Person)-[DRIVES]->(y) return x,y



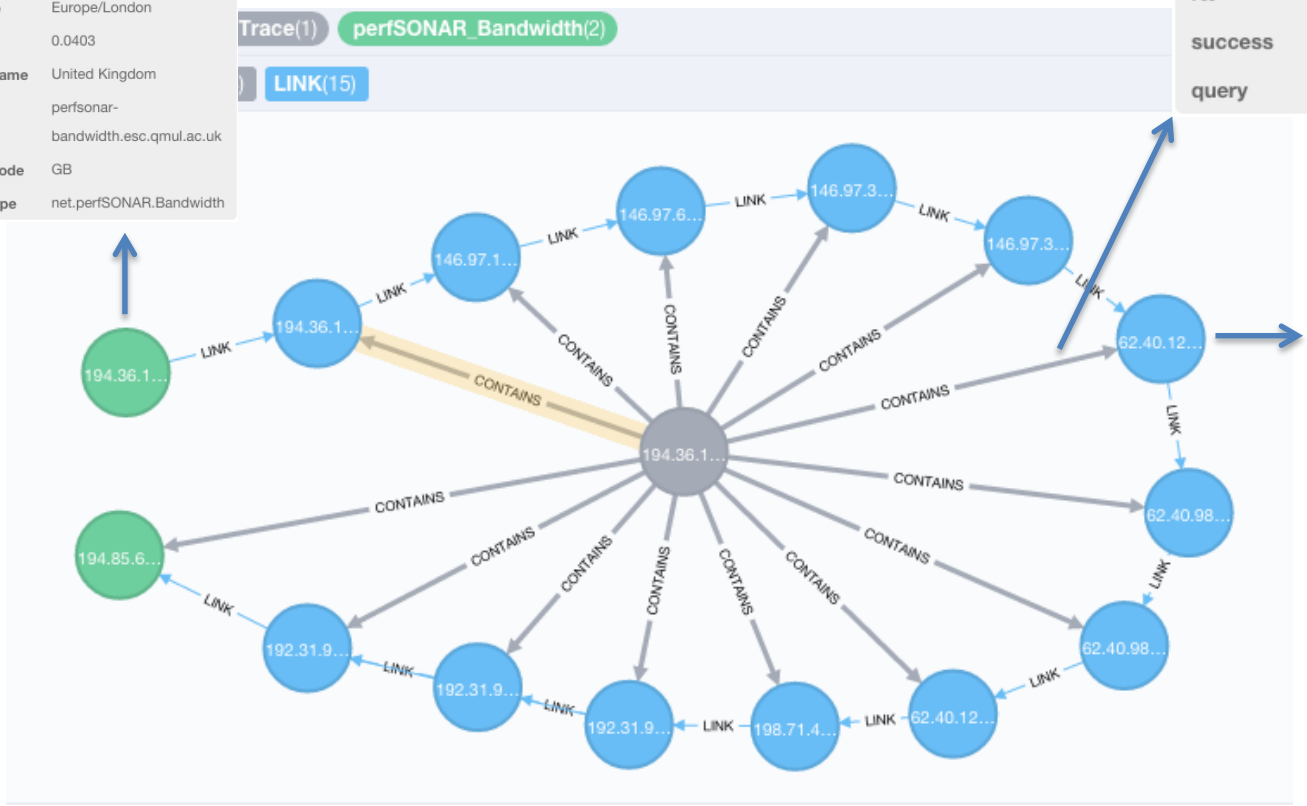
Traceroute representation



Properties

ip	194.36.11.37
hostname	perfsonar-bandwidth-v4.esc.qmul.ac.uk
city	London
lat	51.5231
time_zone	Europe/London
long	0.0403
country_name	United Kingdom
host	perfsonar-bandwidth.esc.qmul.ac.uk
country_code	GB
service_type	net.perfSONAR.Bandwidth

ip	62.40.125.18
ttl	9
rtt	114.88
success	1
query	1

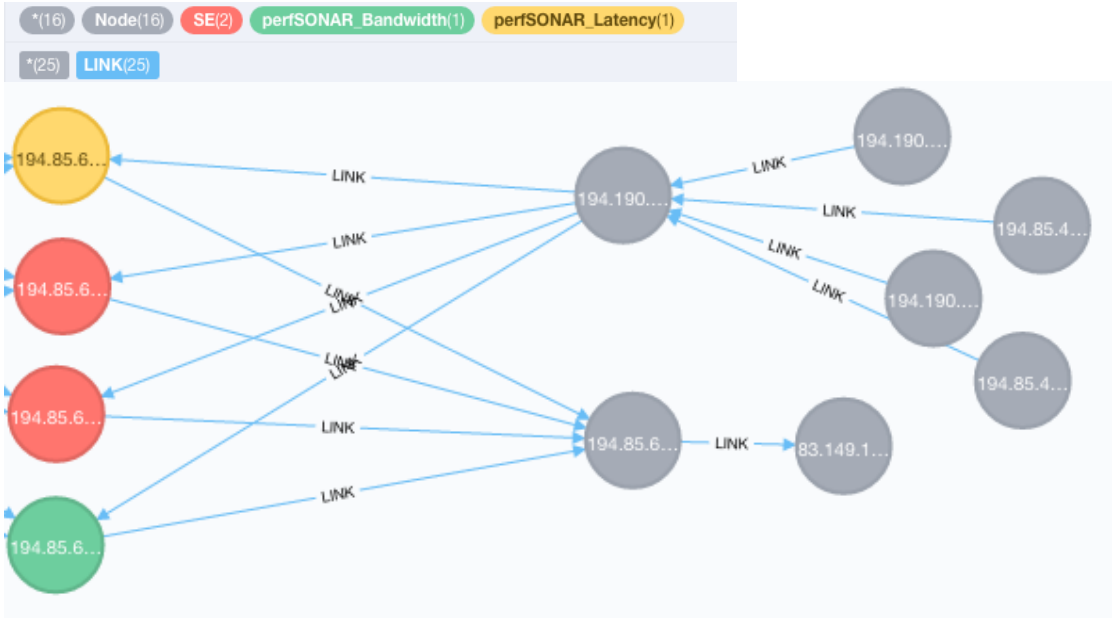


ip	62.40.125.18
name	GEANT
hostname	abilene-wash-gw.mx1.fra.de.geant.net
handle	DANT-RIPE
description	IP allocation for GEANT network infrastructure
address	City House, 126-130 Hills Road Cambridge CB2 1PQ, UK
cidr	62.40.112.0/20
country	GB
range	62.40.112.0 - 62.40.127.255
misc_emails	dancert@dante.net
abuse_emails	abuse@dante.net

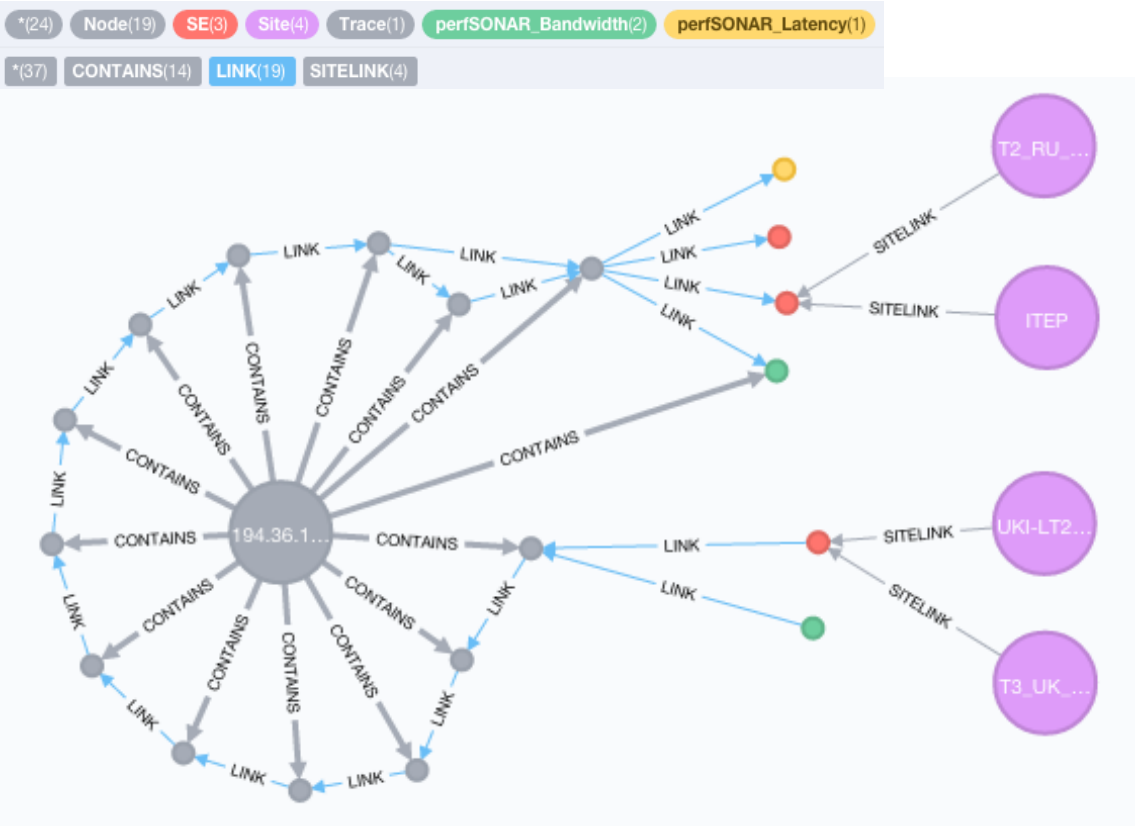
CONTAINS ip: 194.36.11.3 ttl: 1 rtt: 0.519 success: 1 query: 1

Proximity

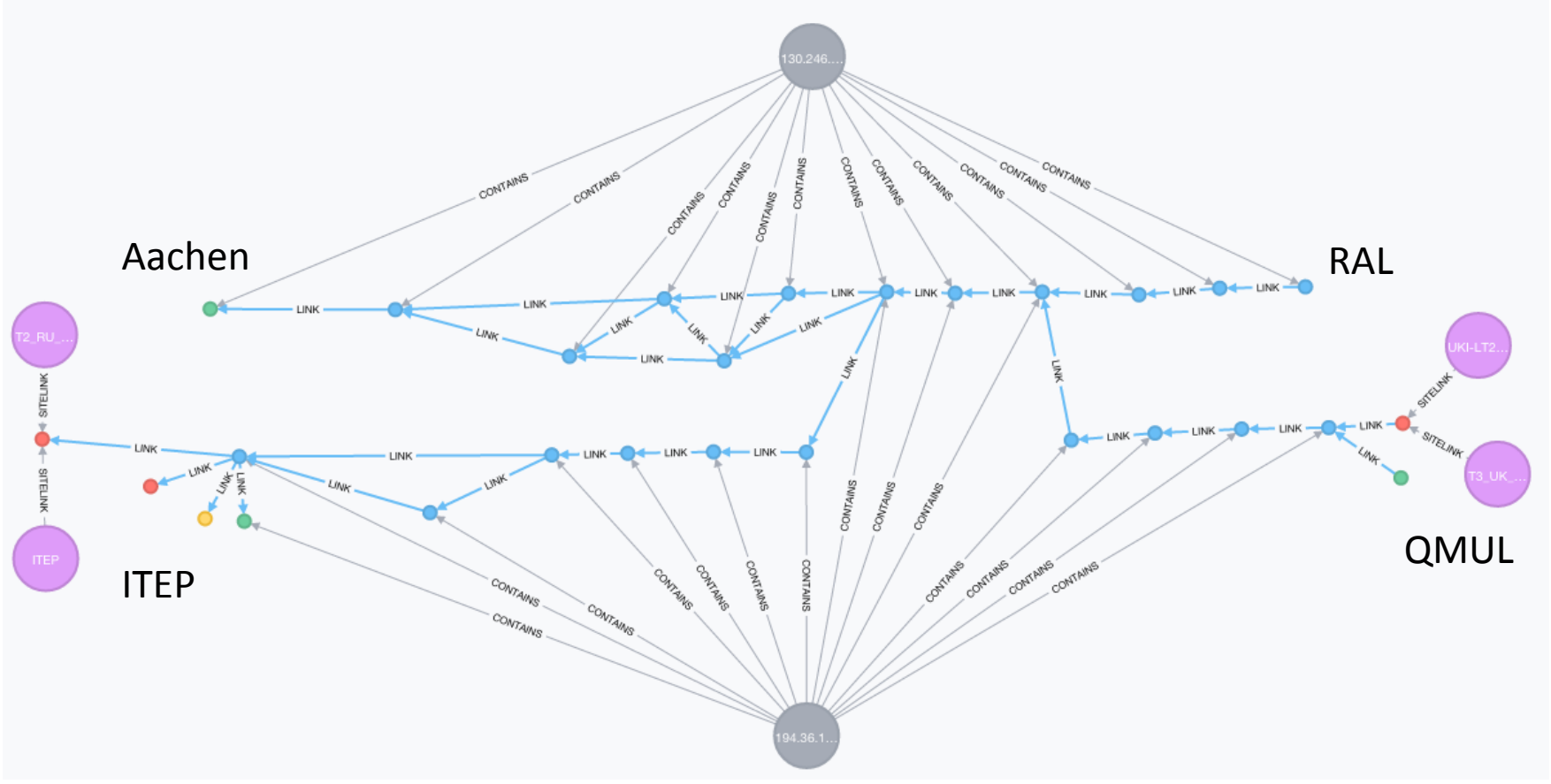
- Via graph query, what is n hops away from given node
  - (match (x:SE {host: "se3.itep.ru"})-[:LINK*1..2]-(y:SONAR) return x, y)



Site links


*(24)
Node(19)
SE(3)
Site(4)
Trace(1)
perfSONAR_Bandwidth(2)
perfSONAR_Latency(1)
*(37)
CONTAINS(14)
LINK(19)
SITELINK(4)
T2_RU_...
ITEP
UKI-LT2...
T3_UK_...
194.36.1...
LINK
CONTAINS
SITELINK

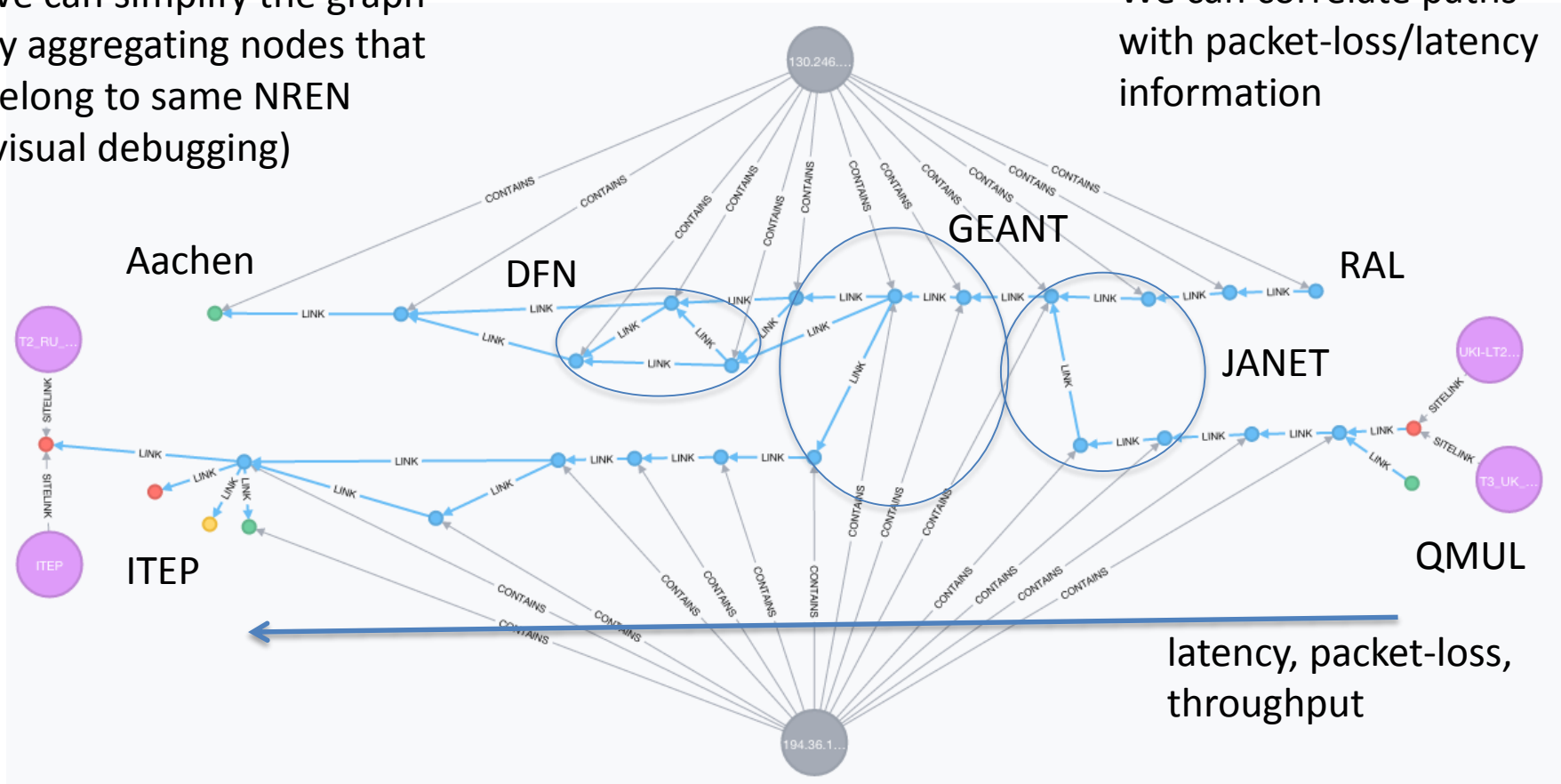
Path analysis



Path analysis

We can simplify the graph by aggregating nodes that belong to same NREN (visual debugging)

We can correlate paths with packet-loss/latency information



Prototype

- Available at proximity.cern.ch
  - Graph with sonar traceroutes, ipwhois, geoip, ATLAS/CMS sites and SEs linked (via common subnets only)
  - Neo4j access on request at proximity.cern.ch (only from CERN) and at http://192.41.231.184:7474/ (from anywhere, but needs to be updated)
  - Currently only for testing
- There is also an initial version of API
  - http://proximity.cern.ch/api/0.1/trace/nearest?host=perfsonar01-iep-grid.saske.sk&count=3
  - http://proximity.cern.ch/api/0.1/trace/tracepath?src=atlas-npt2.bu.edu&dst=heplnx130.pp.rl.ac.uk
  - http://proximity.cern.ch/api/0.1/trace/shortest_path?src=atlas-npt2.bu.edu&dst=heplnx130.pp.rl.ac.uk
  - Needs more work to be useful

Traces - Summary

- Initial prototype for proximity based on traceroutes available
  - There is also Spatial Extension to Neo4j which would provide GeoIP functionality – needs to be tested - common model for both GeoIP and traces
  - Need to gain more experience in graph databases and evolve loaders (also add other sources), validate the current model, evaluate performance
  - Evolve API to support more complex queries
- Positive experience in using graph databases (Neo4j)
  - There are others available, usually not as mature as Neo4J, but they're catching up fast (SPARK GraphX, Giraph, Blueprints, etc.)
- There is potential to offer more advanced and very useful functionality
  - Path analysis, aggregations,