Edge-based Performance Analysis

Susan Hinrichs, Eric Schwartz, François Pesce, Huan Yang, Dan States – ATS Summit Fall 2015

Observation

- The entry point to the network (Edge) is an excellent measurement point for overall system performance measurement
  - Everything goes through the edge
  - Need observation into the traffic at the edge
- ATS offers many great points for gathering performance data
  - Overall installation metrics and performance counters
  - Detailed access logs

Contributors over the year

- Dan States
- François Pesce
- Eric Schwartz
- Huan Yang
- Susan Hinrichs
- Josh Blatt
- Bryan Call
- Pushkar Pradhan
- Yuni Kim
- Josh Juen
- Shraddha Advani
- Yu Zou
- Wei Sun
- East Chao
- Dave Thompson

Topics

- Latency Model
- Mirroring metrics onto log entries
- Latency Maps / Heat Maps / Isochrone
- WhyHigh / YHigh

Latency Model

Latency Model

- Create a simple model of latency for service
  - Build model for measurable elements like RTT, cache hit rate, and connection start up.
- Use model to predict how changes to underlying network characteristics will affect overall service latency
- Initially concentrating on small data transfers

Latency Model

- Latency = Time from User Agent sending request to User Agent Receiving Response
  - Total Average Latency = Time from user agent to Edge + Time from edge to data center = UA_to_Edge_t + Edge_to_DC_t
- UA_to_Edge_t = Connection_overhead + data_exchange_t
- Edge_to_DC_t is similar, but only applies if the data is not in cache
  - Edge_to_DC_t = (1 – cache_hit_rate)*(connection_overhead’ + data_exchange_t’)

RTT and latency measures

- t = “best” latency from client to Edge
- Client may not get routed to best Edge entry point
  - A less optimal route adds time d to the latency
  - brp is the percentage of the time client is routed to non-optimal edge entry
- Average latency = at = (brp)*(t + d) + (1-brp)*t = brp*d + t
  - RTT = 2 * at
- For small data exchanges data_exchange_t can be approximated
  - data_exchange_t = RTT = 2 * at

Connection Overhead per transaction

- Connection Overhead = average time spent on connection setup
- Four cases
  - Reuse existing connection (ex_conn_per) – No overhead
  - Open a new TCP connection (no SSL) (tcp_only_per) – 1 RTT
  - Open a new SSL connection but reuse previously negotiated session (ssl_restart_per) – 2 RTT
  - Open a brand new SSL connection and session (ssl_full_per) – 3 RTT
- Average overhead is probability of each case times time of each case
  - Connection_overhead = tcp_only_per * RTT + ssl_restart_per * 2 RTT + ssl_full_per * 3RTT

Q1 Results

- Used RTT logs, DNS, logs, and ATS metrics to gather some initial results in Q1
  - Very small time duration logs
  - No doubt very much over-generalizing these results
  - Used to decide where to attack performance in the following quarters

Q1 Connection overhead percentages

- UA to Edge information gathered from ATS metrics
- Edge to DC information gathered from ysar requests per connection

UA to edge	Percentage	Edge to DC	Percentage
ssl_full_per	11.2%	ssl_full_per' (ycs)	20%
ssl_restart_per	11.8%	ssl_full_per' (ycpi)	10%
tcp_only_per	7.5%	ex_conn_per' (ycs)	80%
ex_conn_per	69.5%	ex_conn_per' (ycpi)	90%

Example use of model to evaluate impact of change

- Analyze impact of SSL Handshake latency increase
  - Say you have a technology to offload sensitive crypto operations to a more security location
    - Connection_overhead = tcp_only_per * RTT + ssl_restart_per * 2 RTT + **ssl_full_per * 3RTT**
    - It adds the RTT (proxy_RTT) from the Edge to the Crypto Proxy box to the cost of a full ssl handshake
    - Connection_overhead_proxy = tcp_only_per * RTT + ssl_restart_per * 2 RTT + **ssl_full_per * (3RTT + proxy_RTT)**
  - Say proxy_RTT = 100ms and RTT = 50ms
    - Connection_overhead = 32.35ms
    - Connection_overhead_proxy = 43.55ms

Further refinements

- The rest of this year we worked on pushing metrics into the logs (see next section).
  - Still working on getting updates deployed
- Moved access logs into the grid for more regularly scheduled analysis over broader set of logs
- In future need to bring congestion and bulk data into the model

Mirroring Metrics

Mirroring Metrics

- ATS Metrics are very useful
  - Cache Hit Rate
  - SSL connections
    - Number of successful handshakes
    - Number of errors for each particular type of error
- ATS Metric granularity is at an ATS installation

Mirroring Metrics

- Would be nice to look at some of these metrics at different granularities
  - E.g, Probability of full SSL handshakes per transaction
    - For client geographic region
    - Time of day
    - Type of client (mobile vs wired)

Mirroring Metrics

- Spent Q2 and Q3 adding fields to access logs
  - %<cqtr> - client -> ats tcp reuse (in the process of fixing for HTTP2 and SPDY)
  - %<cqssl> - client -> ats ssl status
  - %<cqssr> - client -> ats ssl session reuse status
  - %<pitag> - updated HTTP2 to provide http2 pitag for protocol logging
  - %<sstc> - ats -> origin transaction count (used for tcp reuse)
  - %<pqssl> - ats -> origin ssl status
  - %<{MILESTONE2-MILESTONE1}msdms> - difference between two milestones in milliseconds
  - %<{MILESTONE1}ms> - time of milestone
  - %<cqssv> - client negotiated SSL/TLS version
  - %<cqssc> - client negotiated SSL/TLS cipher suite

Mirroring Metrics

- With these metrics in the logs, can do post processing to analyze the metrics at different granularities
  - Used by the Grid Based Latency Model analysis

Latency Maps / Heat Maps / Isochrone Maps

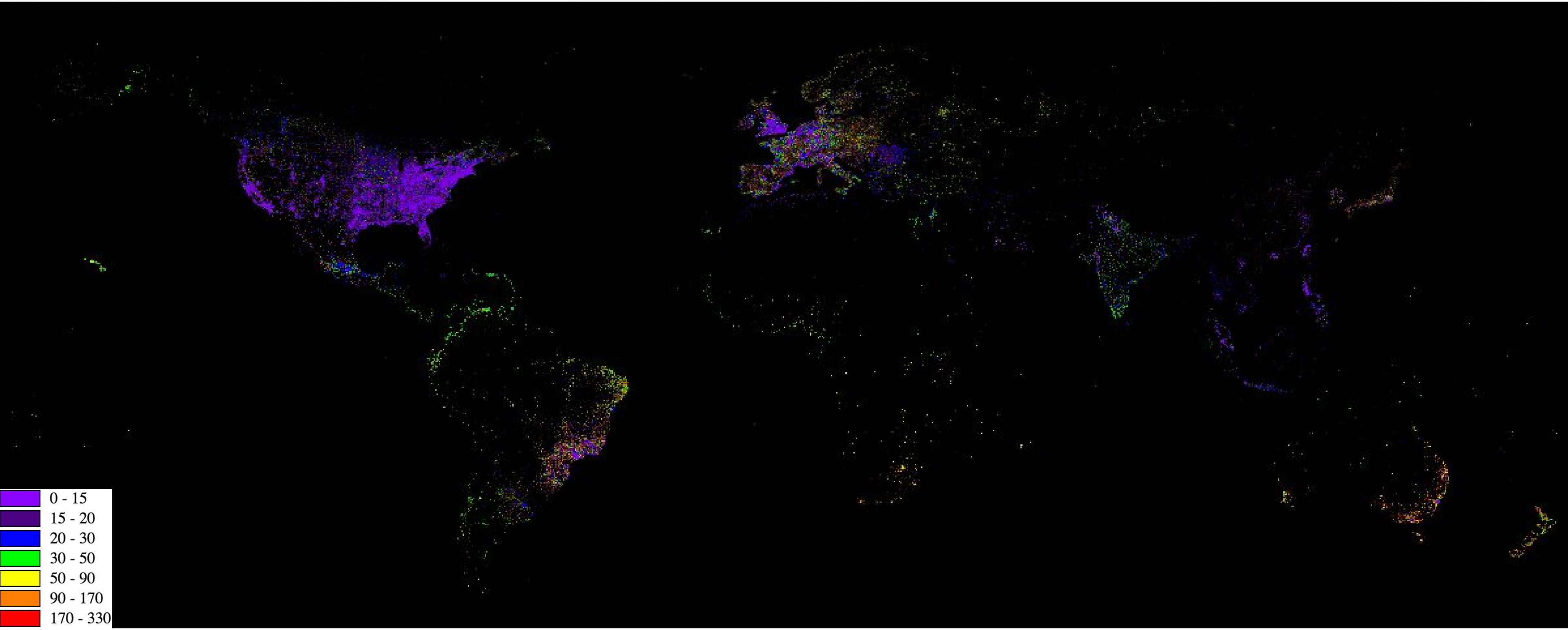
Latency Map

- Inject beacon to gather RTT measurements from all client areas to all Edge entry points
- Run systemtap script to gather RTT
- Analyze systemtap RTT logs to build latency map
  - Very useful for all kinds of analysis
  - Big pile of data
- Build a isochrone map to visualize
  - http://emptypipes.org/2015/05/20/europe-isochrone-map/
  - Can generalize to map RTT to services rather than train transit times.

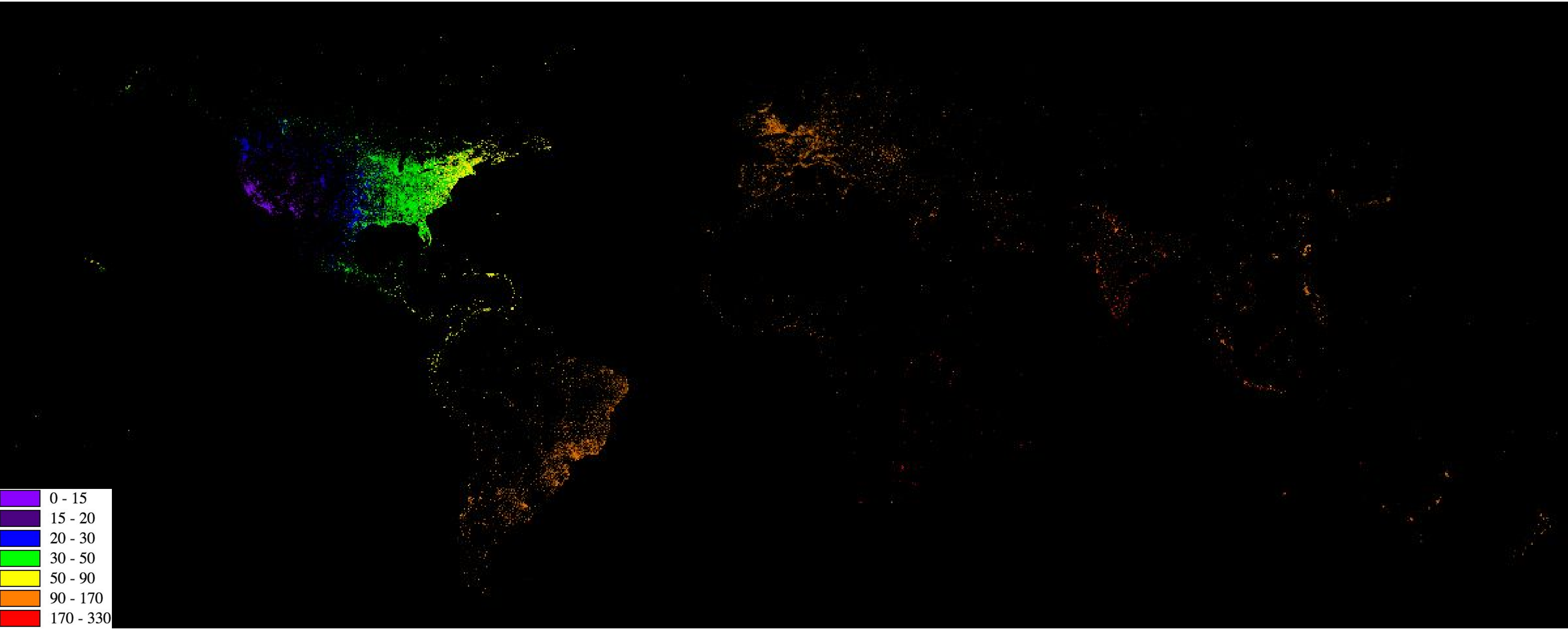
Isochrone Map

- Map four dimensions of data
  - X,Y – Location of clients
  - Hue – RTT time classifications (low, medium, high)
  - Saturation – Number of measurements
- Can present a variety of RTT data
  - Time from client to services via one edge entry point
  - Time for client to services for “best” entry point
  - What if scenarios where edge entry points are added or removed

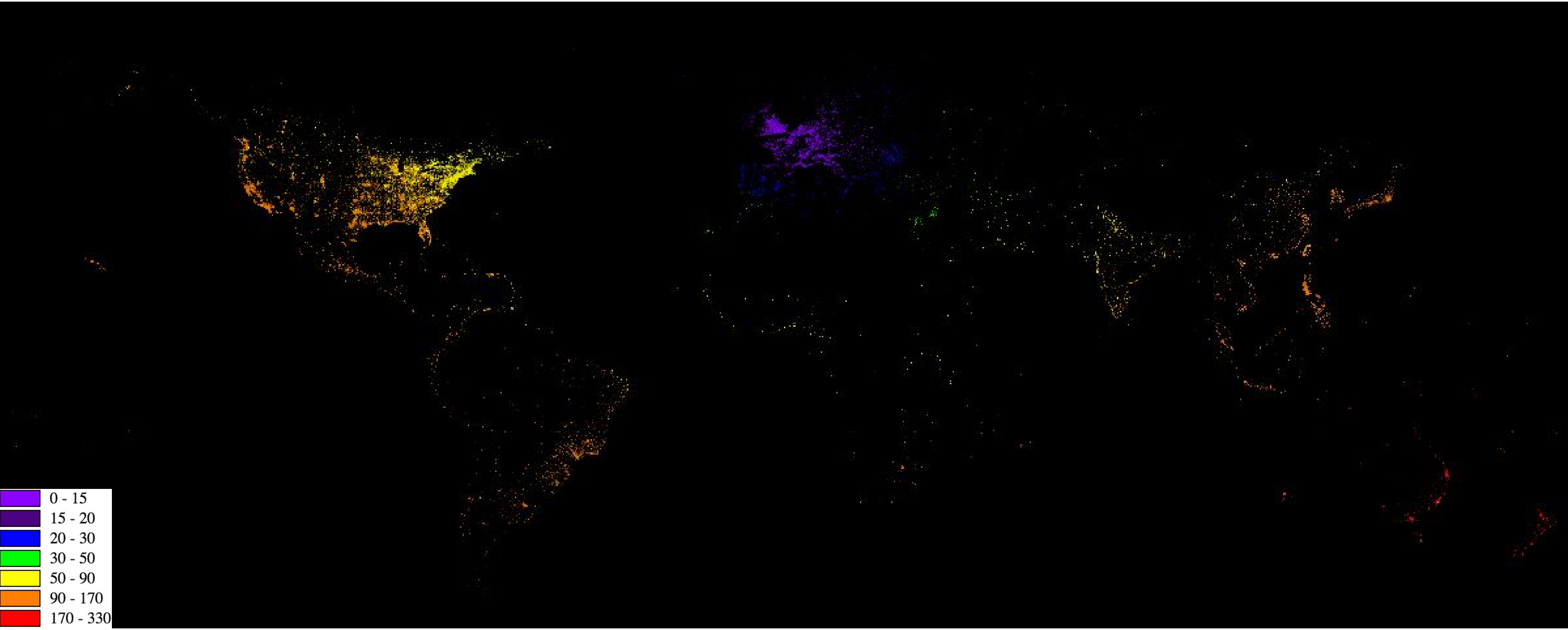
Latency Maps



Latency Maps: LAX

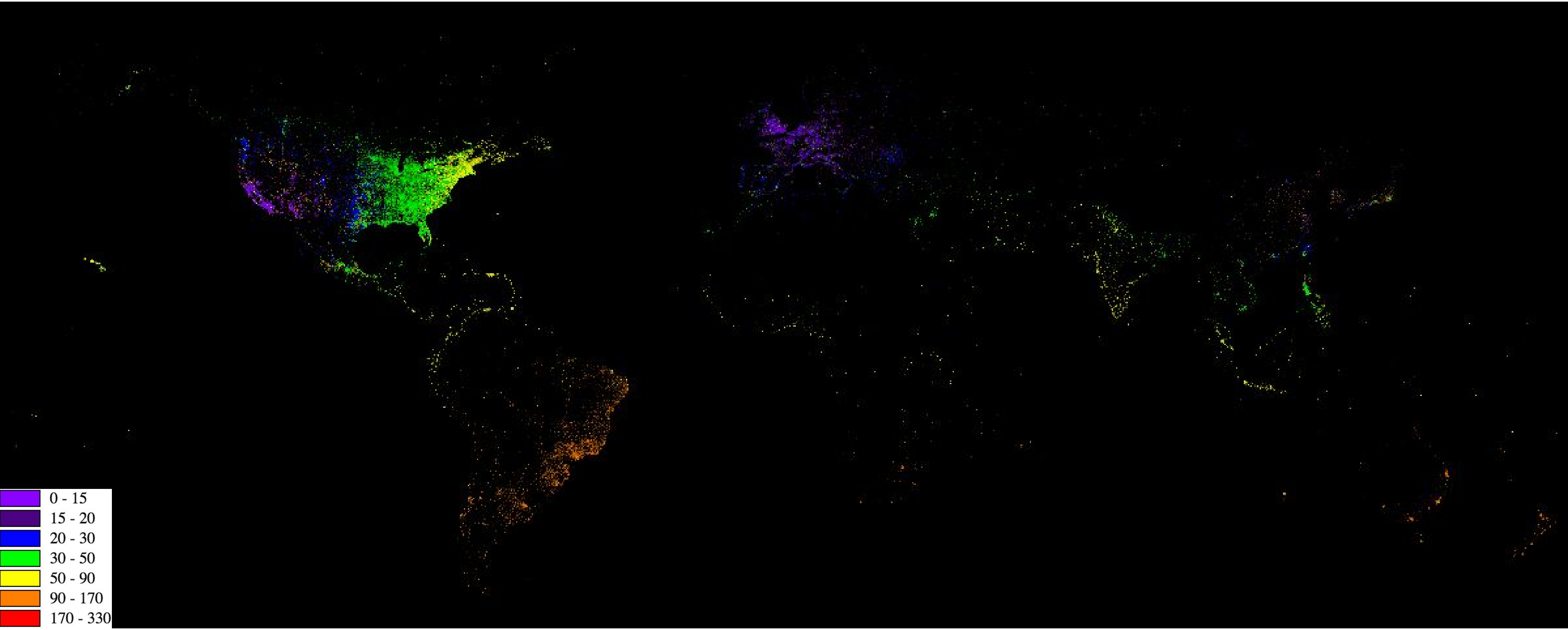


Latency Maps: FRA

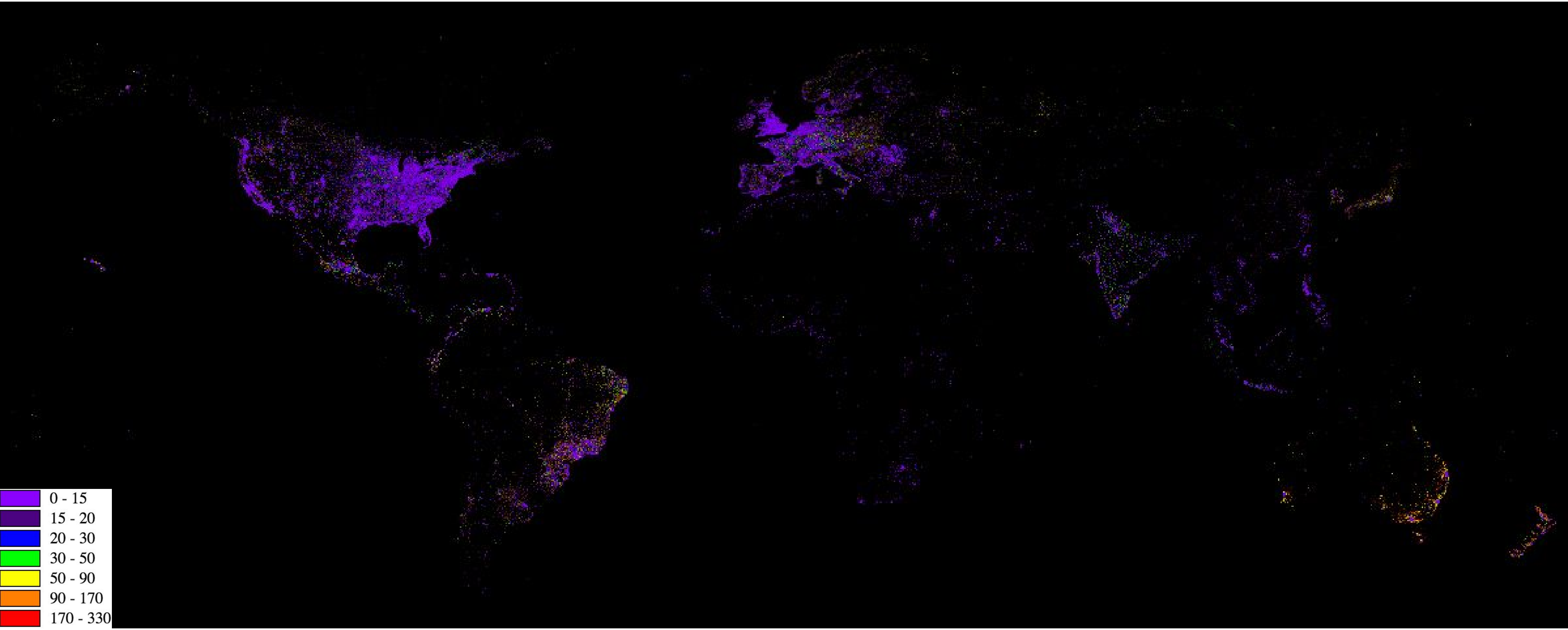


Latency Maps: PEK

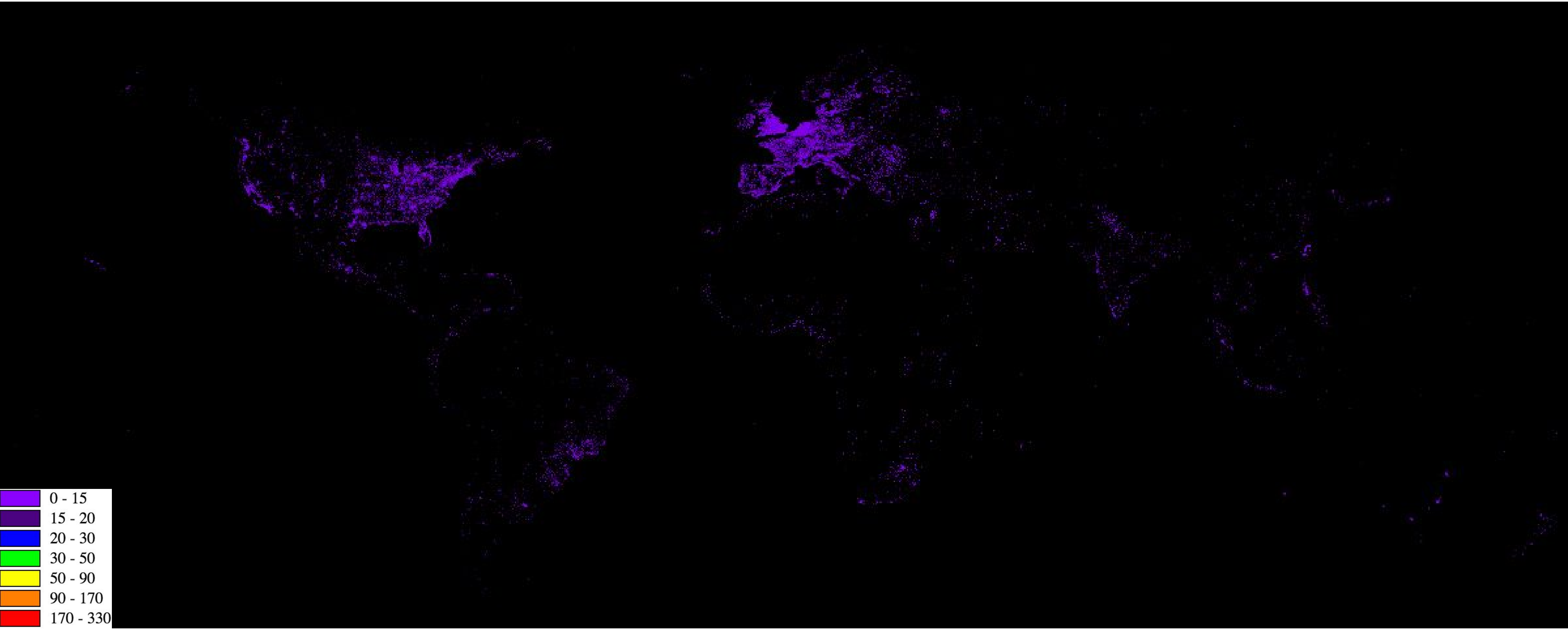
Latency Maps: Composite LAX/FRA/PEK



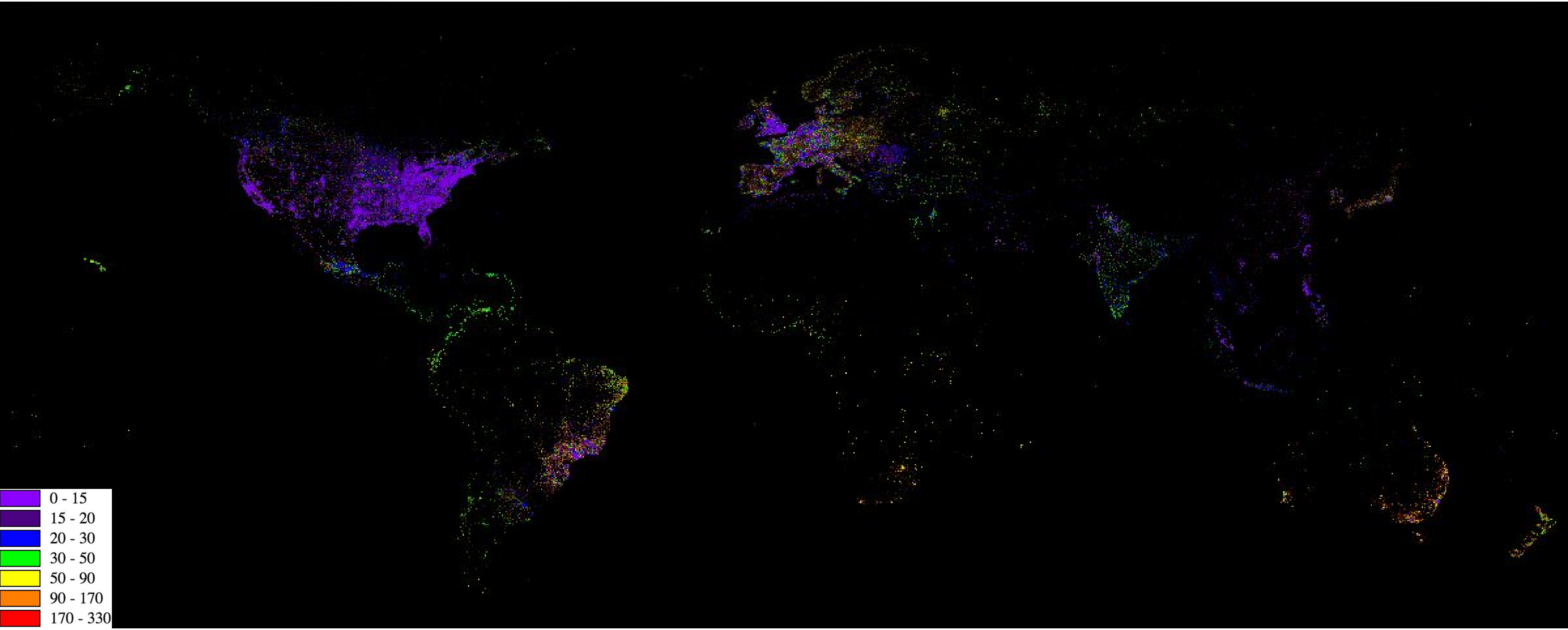
Troubleshooting with maps: case study world with IST



Troubleshooting with latency Maps: just IST



Troubleshooting with latency Maps: Everything but IST



WhyHigh / YHigh

WhyHigh

- WhyHigh is a Google system for identifying latency problems
  - http://research.google.com/pubs/pub35590.html
- Look for “inflated latencies”
  - Address prefixes that are geographically close to each other should have similar rtt to a data center
  - Address prefixes in the same geographic region with significantly different latency characteristics indicates that there is probably something wrong with communication to that provider.

WhyHigh

- Input
  - BGP tables (AS prefixes)
  - RTT
  - Geo Location data

Inflated Latency

- Inflated latency due to bad routing
  - One provider routes through more steps. Even best case is bad
  - Client Prefix Min RTT - Client Region Min RTT > 50ms

WhyHigh

- Built Hadoop scripts to aggregate RTT logs and BGP tables to identify inflated latencies.
- Currently generate spreadsheet of all inflated prefixes.
- YHigh result confirms that there is an issue with a network provider in Brazil.

Moving Forward

- Useful data, but still a lot of data
  - Investigating how to better highlight the biggest problems
  - Involve number of measurements (accuracy of data) or number of clients in prefix (impact of data)
- Improving accuracy
  - Our BGP data dumps are ad hoc. Working on more regular and up to date feeds
  - Need to break up aggregated BGP prefixes for our analysis. ISP may aggregate routing prefixes so they may include multiple geographic regions

Questions?