Colin McCann June 9, 2011

Geolocating IP addresses for the IXmaps database

The IXmaps tracegen program records IP addresses into traceroutes and stores them in our local database. In order to map these traceroutes onto Google Earth, we need to know the physical location of the routers that correspond with the IP addresses. Our software assigns these addresses using a commercial service called Maxmind. Maxmind is reasonably accurate for edge routers, to within several city blocks. However, their success rate when locating core routers drops to nearly zero. We have therefore supplemented their data using a multilayered strategy to manually assign physical addresses to core routers. This document outlines the logic and process of this strategy, demonstrates example cases and provides current information about parsing specific ISP hostnames.

While this geolocation process likely provides more accurate longitude and latitude than is often provided by Maxmind, it generally remains reliable only at a city level; it places routers at a generic city location instead of in a particular building. The levels of accuracy are noted in the IXmaps database and on individual routes as geoprecision (city level, postal code level, building level and Maxmind). Further extension of the processes may allow us greater levels of geoprecision in the future.

At the time of writing, this process has been applied to roughly 57% of the IP addresses in the IXmaps database. This corresponds to roughly 70% of all hops in the database. Of the remaining 43% of IP addresses, most are either edge nodes (accurately located by Maxmind) or do not appear very frequently in the database.

Our geolocation process combines a variety of techniques. This document outlines the three major ones: the traceroute-landmark method, the hostnaming parsing method and the pre/post hop and latency method. While they are not mutually exclusive (in fact, they should support each other), each of these methods has a different associated level of certainty. These levels are connoted by a gl_override code in the database; traceroute-landmark has gl_override=1 (or others, see below), parsing has gl_override=2 and pre/post/latency has gl_override=3. In general, a gl_override of 1 has the highest level of certainty, followed by 2, then 3. See database table glo_reason for more details. These levels of certainty are one element that is used to establish the levels of geoprecision discussed above.

1. Traceroute-landmark method of geolocating routers (gl_override=1 and 5<=gl_override<=27)

Logic and process

The logic to this method is as follows: we send traceroutes to IP addresses that we know are located in a particular city. The IP addresses in the last couple of hops are likely to be in that city. We become increasingly certain of this under the following conditions (among others): greater number of traceroutes, greater number of different locations where traceroutes are initiated, greater number of times that IP is observed in traceroutes with city-specific targets. We can also observe patterns in the associated hostnames. Our level of confidence can be augmented by previous observations about patterns in hostnames, although each will be specific to an ISP.

The process can be divided into two steps: evidence and generalization. These steps are applied to a set of traceroutes targeting a city and an ISP. Thus, for the case of each city/ISP (or ASN) pairing, we must establish evidence of a pattern in IP addresses and then use that pattern to generalize to other IP addresses with that city/ISP. This process is detailed below.

1. Create batches that contain URLs known to be located in the target city. Educational institutions, museums, galleries, theaters, zoos, major unique attractions, local newspapers, etc. are all good examples. Run several batches of traceroutes (TRs) to the target city; this decreases the effect of anomalies and increases confidence in the data. Accuracy and confidence also increase if batches targeting the city are run from different origin locations. In general, we would estimate that a minimum of 2-3 batches should be run for each location, ideally from different originating locations.

2. Query the ixmaps database (DB) to determine which of those traceroutes contain IP addresses with the target ISP as part of the final hops. The closer

to the end of the traceroute (i.e. towards the final hop), the greater the confidence we have that the IP address is located in the target city.

3. Collect and note these TRs. There should be a minimum of 3 with a similar hostname structure (we have capped the number at 15 in the database, but more is obviously always better). Routes with same origin and destination should be included only once. A greater number of collected routes results in a higher level of confidence about the resulting geolocation of the IP addresses. Further, a mix of destinations, origins, submitters, dates (and other variables) is best for producing robust evidence; more variety gives a higher level of certainty, although there is currently no formalized method of documenting this variety.

4. Many ISPs name their router IP addresses with a consistent, parsable hostname. These characteristics allows us to generalize the findings from the previous steps to other IPs with the same ASN and city. For example, hostnames for Cogent (ASN 174) routers in Chicago contain the strings “ord” and “cogent.” Thus, we can go back and query the DB for all hostnames with these characteristics, and be reasonably confident that they are all geolocated in Chicago. Further examples will be given below. It is obviously only necessary to update the IP addresses that are incorrectly geolocated by Maxmind. It can be difficult to decide when this is the case. Seeing a visualization of the route can help, as certain hops will look counterintuitive. For example, if a route jumps from Chicago to New York to Chicago to Dallas, it can be used as partial evidence that the IP address in New York is incorrectly located. Thus, queries can be further limited by latitude and longitude characteristics. These vary from ASN to ASN; at the least, latitude and longitude of 38.0/-97.0 and of the ISP's head office are incorrect. Further details are provided in Section 2.

5. Once these hostnames are located in the DB, the details are extracted into a spreadsheet. At the least, those details must include ASN, hostname and IP address. In addition, the spreadsheet must include fields for city, latitude and longitude. These are not taken from the DB, but rather assigned. Latitude and longitude will only be exact to city level. Wikipedia/Geohack is used as a reference for this data (i.e. for Chicago, http://toolserver.org/~geohack/geohack.php?

pagename=Chicago&params=41_50_13_N_87_41_4_W_type:city_region:US-IL).

Notes:
This will always be an incomplete process. Some ASNs do not parse predictably, some are too small to be worth spending time on, new IPs will be created and old ones deleted, etc. This process is only intended to address the most egregious errors. Further refinements are certainly possible, but it will depend on the value added vs. the cost of labour. If the process is automated (as discussed later), running the script on the DB every couple of weeks should be sufficient – assuming current levels of data production.

A reverse of this technique will also produce useful data. In that case, instead of targeting a city/ISP and sending TRs towards it, we can look at the city of origin/ISP pairing. For example, the first hops in TRs sent from Chicago are likely to be located in Chicago. The process is the same beyond that.

Regarding confidence – it is difficult to be completely certain that a router/IP address is geolocated in a particular location. This process does not look to establish anything with complete certainty. Rather, the greater the amount of data that is received, the greater the likelihood that the geolocation is correct. The total number of traceroutes gathered in step 3 is one reflection of certainty (others include consistency of hostname patterns, comparisons to other data sets, etc.)

It may be difficult to decide on how early/late in the traceroute is acceptable for the target IP to be considered to be in the target city. In general, the last hop before the final AS number in the route is acceptable (i.e. if the route has 12 hops and 9-12 have a single ASN, 8 would be acceptable for 'final hop' and could therefore be used for evidence that the IP is in the city). This is due to the fact that long haul ISPs are likely to hand off in an IXP located in the target city. This situation is also a good place for other methodologies to be applied. For example, if there is a large jump in the latency between hop 7 and hop 8, and then hops 9-12 all have the same latency as 8, we can assume that hop 8 is in the same city.

There is not always a one to one correlation between hostname and IP. For example, AT&T sometimes has many different IPs for one hostname (likely due to load balancing).

Queries use the “hostname like '%xxx%'” structure because we have only established evidence that IP addresses with that type of hostname are located in a particular city. This cannot be extended to all IP addresses in the city/ISP pair. For example, some AT&T IPs are not given a hostname (these just have a dotted quad). These cannot be corrected using this methodology.

Queries should use multiple identifiers. At the very least, use two different “hostname like '%xxx%'”. This decreases potential errors (which is always a major concerns when human-generated data is entered into a DB). For example, Cogent hostnames often contain another ISP name (ie verio.ord03.atlas.cogentco.com); if a Verio query only selects by “hostname like '%verio%'”, it will also incorrectly pull these Cogent hostnames as well. Multiple identifiers help correct this. Further, ASN is not always a useful identifying characteristic; many ISPs use more than one ASN and some ASNs are listed as -1 or 0 in the DB.

The ixmaps DB is constructed to contain two fields for each of latitude and longitude. *mm_lat* and *mm_long* are for data pulled from Maxmind. The fields *lat* and *long* are for the newly generated data. This is only relevant if the user is involved with inputting the data into the DB.

Each city/ASN pairing has an associated gl_override code (currently codes 5-27), while the IP addresses corrected using the generalized hostname patterns produced as a result of the traceroute-landmark method are given the gl_override code of 1. Corrections that are assigned based on hostname patterns from other sources (Sarangworld, ISP websites, logic, experience) are given the gl_override code of 2.

Example case: Chicago-AT&T (7018) (gl_override = 5)

I created a batch file for Chicago containing Chicago landmarks and uploaded it to the /trsets folder on the ixmaps server (it is important to follow the exact formatting for the .trset files). I then ran the batch with the IXmaps tracegen program several times from several different locations with different ISPs in Toronto.

I then queried the DB for all TRs with traceroute_ids in the range of the 150 or so generated, that also contained ASN 7018 (AT&T), using some variation on the following query:

```
#
select traceroute_id,i.ip_addr,hostname from ip_addr_info as i join tr_item as t on
i.ip_addr=t.ip_addr where asnum=7018 and traceroute_id between 3914 and
4155 group by traceroute_id,i.ip_addr,hostname order by traceroute_id;
#
```

In this case, I chose the range of 3914-4155 because those traceroute_ids corresponded to the routes generated when I ran the tracegen. I found that many hostnames contained the strings “cgcil” and “att”. I noted the relevant traceroute_ids and went to check on the website (under 'submitters') to see which of those routes have AT&T routers in their final hops. I noted about 5 of these (tr_ids: 3923, 3938, 3997, 4143, 4154), chosen because they contained different destinations. This serves as 'sufficient evidence' that the associated IP addresses are geolocated in Chicago.

I then looked at all of the hostnames that contained the strings “cgcil” and “att” using the query:

```
#
select hostname,ip_addr,lat,long from ip_addr_info where hostname like '%cgcil
%' and hostname like '%att%' order by hostname;
#
```

I noted that the latitude and longitude (L/L) for each entry was 38.0/-97.0, indicating that it was incorrect. I then copy-pasted the hostname and associated IP addresses into the spreadsheet (see 'IXmaps references Dec 2010.ods' as an example). I then added latitude, longitude and gl_override (taken from the Wikipedia entry on the city of Chicago). This data can be imported to the database using the dbupdate script. The data can also be generalized and inserted into the db_cleanup script, as described in the following section.

2. Current data regarding ISP-specific hostname parsing (gl_override=1 and gl_override=2)

While the traceroute-landmark method has the greatest level of certainty, it is very time consuming. It is therefore not recommended for each city/ISP pairing. Rather, we extrapolate ISP hostname patterns after examining sufficient pairings (currently, between 20 and 40) and do mass corrections of IP addresses that have those characteristics. These are assigned a gl_override code of 1. We can also look at other methods for deducing hostname patterns, including ISP websites, reference lists (see below), logic, experience, etc. Geocorrected IP addresses that use these methods are assigned a gl_override code of 2.

What follows is a list of hostname patterns for each ISP, with examples. For a complete list of each string being used to parse hostnames in the ixmaps database, see the script /home/ixmaps/db_cleanup. This script also contains the required postgresql updates for each ISP.
(see also http://www.sarangworld.com/TRACEROUTE/showdb-2.php3 and http://ecee.colorado.edu/~mathys/ecen1200/hwcl05/router_acronyms.html for older but comprehensive lists of hostname codes).

AT&T (7018)
Generally incorrectly geolocated by Maxmind at 38.0/-97.0.
Hostname contains "att" and "[city+state code]":
- Chicago = "cgcil"
- Dallas = "dlstx"
- San Francisco = "sffca"

Examples: cr1.dlstx.ip.att.net, cr2.cgcil.ip.att.net
Note: many AT&T hostnames have multiple IP addresses

Level 3 (3356)
Incorrectly geolocated by Maxmind at different L/Ls, but often at 38.0/-97.0.
Hostname contains "Level3" and "[City Name]"
Examples: ae-1-100.ebr1.Chicago2.Level3.net, ae-0-11.bar1.SanFrancisco1.Level3.net
Note: Unlike many ISPs, Level3 frequently uses capital letters (note: selects from the database are case sensitive)

Cogent (174)
Generally incorrectly geolocated by Maxmind at 38.9144/-77.0763. This maps to a housing block near Georgetown University, likely a postal code level location of Cogent head office, located in Washington, D.C. (http://www.cogentco.com/us/about_about.php). In this case, we correct both 38.9144/-77.0763 and 38.0/-97.0.
Hostname contains "cogent" and "[airport code]"
Examples: te4-4.ccr02.dfw06.atlas.cogentco.com, te4-2.ccr01.ord07.atlas.cogentco.com

NTT/Verio (2914)
Unlike other ISP subsidiaries (Verizon and it's many variants), NTT and Verio can be treated as the same company. They use the same ASN, the hostnames parse similarly, etc. Both are grouped under 'NTT/Verio' in the spreadsheet and script. For simplicity, they are referred to using their ASN.
They are incorrectly geolocated by Maxmind at 39.569/-104.858, which is a postal code level geolocation of the NTT head office in Colorado (as per http://www.verio.com/contact/offices/)
Hostname contains "ntt"/"verio" and "[CLLI code]":

- Chicago = chcgil
- Dallas = dllstx
- Miami = miamfl

Examples: as-3.r20.dllstx09.us.bb.gin.ntt.net, ae-3.r21.chcgil09.us.bb.gin.ntt.net, ge-3-6.a00.chcgil07.us.ce.verio.net
Note: NTT is one of a set of ISPs that can be parsed using CLLI codes. http://www.telcodata.us/search-switches-by-clli-code can be useful when dealing with CLLI codes.

AboveNet (6461)
Generally incorrectly geolocated by Maxmind at 41.0291/-73.758, which maps to a highway in White Plains, NY. This is likely a postal code level geolocation of the AboveNet head office, which is several city blocks away (http://www.above.net/contact/contact-offices.php).
Hostname contains "above" and "[airport code]"
Examples: xe-0-3-0.cr2.ord2.us.above.net, ge-0-3-4.mpr1.dfw2.us.above.net

ALTER (Verizon/MCI, alter) (701)
Often incorrectly geolocated by Maxmind at 38.0/-97.0, but many other L/Ls as well. ALTER is a subsidiary of Verizon/MCI, dealing with businesses. Hostnames parse somewhat unpredictably. They often contain an airport code, but can contain a city code instead (or neither. Further, they contain "ALTER" or "alter."
Hostname contains "ALTER" and "[CITY CODE]"/"[AIRPORT CODE]"
Examples: 0.ge-5-1-0.XL4.IAD8.ALTER.NET,
0.ge-1-0-0.BR1.CHI13.ALTER.NET

Savvis (3561)
Usually incorrectly geolocated by Maxmind at 38.65/-90.5334, which roughly corresponds to the Savvis head office in Town and Country, MO (as per http://www.savvis.net/en-US/Contacts/Pages/GlobalOffices.aspx).
Hostnames are parsable, but are somewhat unpredictably. They will usually contain the city, sometimes capitalized, sometimes not. They will occasionally contain the state code and occasionally contain a reference to a carrier hotel.
Hostnames contain "savvis" and "[city name]"/"[City Name]"/"[city name+carrier hotel]"/"[City Name+Carrier Hotel]"
Examples: cr1-bundle-pos1.losangeles.savvis.net, hr1-te-8-0-0.NewJerseynj2.savvis.net, ber1-ge-7-2.chicagoequinix.savvis.net

Peer 1 (13768, often -1 in the DB)
Often incorrectly geolocated by Maxmind at 40.6888/-74.0203 (Manhatten, NY). This does not correspond to the Peer 1 head office, which is located in Vancouver. However, they're consistently wrong.
Hostnames contain "peer1" and "[city code]" and "[colocation centre code]."
Peer1 very kindly lists all of their colocation centres at (http://www.peer1.com/infrastructure/data_centers.php). All that is required is looking up the city and finding the exact address based on that information. www.peeringdb.com is also useful for confirmation (this site is generally quite useful).
Examples: 10ge.xe-0-0-0.tor-1yg-cor-1.peer1.net, oc48-po6-0.mtl-bvh-dis-2.peer1.net, 10ge-ten1-2.mia-89p-cor-2.peer1.net

Bell (577, often -1 in the ixmaps DB)
Often incorrectly geolocated by Maxmind at 60/-95. This L/L is the Canadian equivalent of the American 38/-97 (the exact physical centre of Canada).

Hostnames contain “bell” and “[city name]” or “[CITY NAME]”
Examples: bx4-toronto12_so-1-0-0.net.bell.ca, bx4-toronto12_so-1-0-0.net.bell.ca, bx1-saintjohnnb_POS0-0.net.bell.ca
Note: Bell has a large number of non-parsable hostnames (generally dotted quads), so a pre/post/latency approach is often more useful for attempting to correctly locate their routers. For further details, see section 3. Also, it can be difficult to isolate Bell ip_addrs when doing selects. Use a combination of hostname like '%bell%', AS number=577 and AS number=-1.

Telus (852)
Often incorrectly geolocated by Maxmind at 49.25/-122.95, which is near the Telus head offices in Vancouver (as per http://about.telus.com/media/Factsheet-ROB.html). Hostnames contain “telus” and “[city+state code]”/“[CITY+STATE CODE]:

- Toronto: toroon
- Nanaimo: NNIMBC
- Chicago: chcgil

Examples: toroonxngr00.bb.telus.com, edtnabxmdr00.bb.telus.com, chcgildtgr00.bb.telus.com
Note: Telus does not use CLLI codes (although some Telus codes are the same as the CLLI for the city). 4 letters for city, 2 letters for province is the norm.

Sprint (1239)
Often incorrectly geolocated by Maxmind at 38/-97, but other L/Ls as well. Fairly predictably parsable.
Hostnames contain “sprint” and “[city code]”

- Omaha: oma
- Akron: akr
- Cheyenne: che

Examples: sl-crs1-che-0-0-0-0.sprintlink.net, sl-crs1-stk-0-12-0-1.sprintlink.net

XO (2828)
Often incorrectly located by Maxmind at 38/-97, but other L/Ls as well.
Hostnames contain “xo” and “[cityname-statecode]”
Examples: ae0d0.mcr2.chicago-il.us.xo.net, ae0d1.cir1.seattle7-wa.us.xo.net, ae0d0.mcr1.la-ca.us.xo.net

Note: Many hostnames contain a 'ptr' string. This does not correspond to any city (pre/post/latency method reveal that different hostnames with 'ptr' are located in different cities).

Qwest (209, often -1 in the DB)
Generally incorrectly located by Maxmind at 38/-97 (occasionally others as well).
Hostnames contain "qwest" and "[city code]"/"[airport code]"

- Atlanta: atl
- Washington: dca, dcp, dcx
- New York: jfk

Examples: tpa-edge-14.inet.qwest.net, iah-edge-05.inet.qwest.net

Roadrunner (7017, 7843, 11351, 12271, 13343, 20001)
Sometimes incorrectly located by Maxmind at 38/-97, but many others as well.
Different ASNs correspond to different hostname parsings (and different areas of the US).
Hostnames contain:
"tbone.rr" and ".[city code]"/"[airport code]"

- New York: nyc
- Washington: dca
- Los Angeles: lax

"nyroc.rr" and "[city+state code]"

- Albany: albyny
- Syracuse: syrcny

"nyc.rr" and "[city code]"/"[city+state code]"

- New York: "nyc"
- New Jersey: "nwrknj"

"socal.rr" and "[city+state code]"

- Cameron Park: cnpkca
- Tustin: tustca

TATA (6453)
Generally incorrectly geolocated by Maxmind at 38/-97.
Many of these routers can be located to a building level. The following links are used in the process:
http://www.tatacommunications.com/enterprise/datacenter/colocation.asp

http://www.tatacommunications.com/datacenter/tcx2/images/Updated%20Datacenter%20Map.png
http://www.tatacommunications.com/map/gfp.html
http://www.peeringdb.com
http://lg.as6453.net/bin/lg.cgi (shows what the 3 digit codes before the city names refer to)
http://www.tatacommunications.com/downloads/enterprise/ethernet/8-Ethernet_Locations.pdf
Hostnames contain "as6453" and "[3 digit code]-[CityName]"
These 3 digit codes are listed in the above links; they correspond to a carrier hotel or TATA data center, the locations of which can also be found in the above links
Examples:
if-1-0-0-40.core1.DTX-Dallas.as6453.net – IXP (1950 N Stemmons) in Dallas
if-10-1-808.mse1.NW8-NewYork.as6453.net – IXP (60 Hudson) in New York
Vlan24.icore1.MTT-Montreal.as6453.net – TATA data center (45.482146/-73.540078) in Montreal

Comcast (7922)
Maxmind often incorrectly locates Comcast routers at 39.1029/-94.3826 or 40.7242/-111.879. The later is generally when the correct location is a carrier hotel. It is unclear why either of these L/Ls are chosen by Maxmind, as Comcast is headquartered in Philadelphia.
Hostnames contain "comcast" and "[city name].[state code]"/"[carrier hotel address].[state code]"
Examples:
pos-1-15-0-0-cr01.mclean.va.ibone.comcast.net, pos-0-2-0-0-pe01.1950stemmons.tx.ibone.comcast.net

Shaw (6327)
Incorrectly located by Maxmind at 60/-95. Fairly unpredictable.
Hostnames contain "shawcable" and "[city code]"/"[state code]"/"[region code]"

- Vancouver: vc
- Chicago: il
- Victoria: gv

Examples: rc1nr-pos0-7-0-0.wp.shawcable.net, rc2so-tge0-1-2-0.cg.shawcable.net

Internap (22212 and many others)
Incorrectly located by Maxmind at many different L/Ls. Extremely unpredictable.
Hostnames contain “pnap”/“internap” and “[city code]”

- Seattle: sea
- Boston: bsn
- Palo Alto: pao

Examples: acs007-wdc005-844-cr1.acs007.internap.net, sagonet-1.border2.acs002.pnap.net
Note: many hostnames will contain two or three city codes. It is likely that these correspond to 'coming from' and 'going to' and 'located at.'
Internap is sometimes [from]-[to].[location] and sometimes [to]-[from].[location]. It is not certain that [location] is the router location. It may be [from]-[to].[from]
Roughly half of PNAP L/Ls are located correctly by Maxmind. Therefore, fix only entries with lat=33.7516 and lat=33.8004. PNAP hostnames also sometimes contain carrier hotel information (look up at https://www.peeringdb.com/private/facility_list.php if locational information is required)

Cenic (2152 and 2153)
This is a university network in California, incorrectly located by Maxmind at 33.8188/-118.038.
Hostnames contain “cenic” and “[city code]”

- Los Angeles: lax
- Oakland: oak
- Sunnyvale: svl

Examples: dc-lax-core2—lax-core1-10ge.cenic.net, ucb—oak-dc2-ge.cenic.net
Note: hostnames generally contain (at least) two pieces of locational information. Hostnames parse to [where it's going]--[where it's at] (or maybe [where it's going]--[where it came from]). The question really is 'where is the router in question actually located.' The argument could be made for either former or latter. This highlights a general issue with this type of geolocation. Often, all routers could be shifted forward or back in the representation of traceroutes. However, for the purposes of IXmaps this is not overly relevant (we want the

general route and what locations it passes through, not the ability to say 'this router with this IP is located in this precise location').
Cenic in LA is located 1 Wilshire as per http://www.peeringdb.com and http://doc.cenic.org/tools/view.pl?table=Site&clli=LSANCA02

Telia (1299)
Incorrectly located by Maxmind at 47/8 in Switzerland. Telia's head office is in Sweden. 47/8 may be the European equivalent of 38/-97 and 60/-95.
Hostnames contain "telia" and "[city code]"

- New York: nyk
- Chicago: chi
- London: ldn

nyk-b4-link.telia.net, ldn-bb1-pos7-1-0.telia.net

Global Crossing (3549)
Generally incorrectly located by Maxmind at 38/-97. Fairly inconsistent.
Standard hostnames contain "gblx" and "[CITY CODE]"/"[AIRPORT CODE"
Examples: te3-4-10G.ar5.SEA1.gblx.net, po1-20G.ar6.DCA3.gblx.net
Note: Global Crossing hostnames will often contain location specific information, generally the name of the company that connects to the backbone through the target router or addresses. Examples: FIBER-TECHNOLOGIES-WBS.GigabitEthernet3-0-9.ar1.CLE1.gblx.net, Onecleveland.ge-3-0-0.ar2.cle1.gblx.net

Tinet (3257)
Generally incorrectly located by Maxmind at 50/8.7. Fairly inconsistent.
Hostnames contain "tinet" and "[city code]"

- Toronto: tor
- New York: nyc
- Los Angeles: lax

Examples: xe-1-1-0.mtl10.ip4.tinet.net, ge-6-1-0.mia11.ip4.tinet.net
Note: some hostnames do not contain city codes; instead they have an ISP connection (i.e. the ISP with which this IP address is connecting Tinet). They take the form ["ISP name"]-gw (for gateway, I think) and cannot be geolocated accurately.

Hurricane Electric (6939)
Generally incorrectly located by Maxmind at 37.5155/-121.896. Fairly predictable.
Hostnames contain ".he." and "[airport code]"
Examples: 10gigabitethernet1-1.core1.sea1.he.net, 10gigabitethernet1-1.core1.dal1.he.net
Note: AS number is occasionally 0.

The following ISPs are smaller and/or do not appear frequently in the database. They are grouped together based on how hostnames are parsed, with notes for specific ISPs. See db_cleanup script for more details.

Paetec (1785), ThePlanet (21844), Integra (7385), GroupTelecom (6539).
These ISP hostnames parse based on CLLI codes. They're not completely consistent (some CLLI codes have one letter different than the standard).
Note: GroupTelecom is generally all capitals letters.
http://www.telcodata.us/search-switches-by-clli-code is useful for CLLI code data.

Rackspace (various), Limelight Networks (22822), Secureserver (26496), Attens (various), Viawest (13649, 32868), City-Guide (13680); also supplementing Layer (various), Megapath (4565).
These ISP hostnames parse based on city/airport codes. They are fairly consistent. There is a very wide range of Maxmind assigned L/Ls.

Earthlink, Lincon, TW Telecom, RCN, Abilene, Canarie NTN/canet, Cox, Layer, OARNET, Aliant.
These ISP hostnames parse in a variety of ways, often based on city codes. However, they usually require '-'s and '.'s to distinguish them (see db_cleanup for more details).
Note: http://www.canarie.ca/en/network/tools/about may be useful for specific Canarie locations

Corrections based on preceding/following hops and latency (gl_override=3)

While many ISPs have predictable codes contained within their router hostnames, some do not. From the data gathered, Bell and AT&T are the most frequently observed culprits in this regard. In addition, some other major ISPs (like Cogent, i.e.) have set the hostname of some routers as the IP address (dotted quad). (Note: this is frequently observed when the router is between two hops in the same city and the AS number changes (that is, traffic is passed from one ISP to another with the same city). We conjecture that these routers are therefore often situated in carrier hotels/IXPs, although we have not changed geoprecision from city level to building level to reflect that conjecture without further evidence).

Without parsable hostnames, the traceroute-landmark methodology cannot be utilized (nor can we use more conventional hostname parsing techniques, obviously). Instead, we can use a different kind of logic to modify the geolocation of routers, one based on preceding hops, following hops and latency.

Basically, if [hop x] is unknown, we use the information from [hop x+1] and [hop x-1]. In the simplest case, if [hop x-1] is in NY and [hop x+1] is in NY, we can assume that [hop x] is in NY. Similarly, if [hop x-2] and [hop x+2] (or [hop x+1] and [hop x-2], etc.) are in the same city, we can assume that [hop x] is in that city, with somewhat less certainty.

If the preceding and following hops are in different cities, it is more difficult to geolocate an IP address with certainty. We therefore make some further assumptions. If the lowest latency in [hop x] is roughly equal to the latency in [hop x-1], but not the latency in [hop x+1], we can guess that [hop x] is located in the same city as [hop x-1] (and the same logic can be used for [hop x+1]). For example, in tr_id 1486, I placed hop 10 in Halifax based on this logic. Hop 9 (in Montreal) has a lowest latency of 93, while hop 11 (in Halifax) has a lowest latency of 109. Since hop 10 has a lowest latency of 109, we can guess that it is located in Halifax. There is obviously somewhat less certainty here than in other types of geolocation of IP addresses. However, certainty can be increased by looking at multiple routes with the same IP address and seeing if they correspond (if the route follows the same path, consistent latencies can also be used as evidence).

If there is not a marked difference in latencies (or other helpful information), we can be even less sure when geolocating. The following methods for establishing L/Ls have a low level of certainty and should only be attempted when necessary. We are currently assuming that, as a rule, smaller ISPs are less likely than large ISPs to move packets over large distances. In tr_id 3445, we modified hop 18 to a generic San Francisco L/L based on this logic. The latencies for hops 17/18/19 were similar, which is no help in placing hop 18 into either Dallas or San Francisco. However, we would argue that moving traffic from Dallas to SF is more likely to be done by AT&T than the much smaller AS#13867; AT&T retains the packets within their network until they reach SF where they are handed off to the smaller ASN. Therefore, hop 18 (an AT&T node) is placed in San Francisco.

We would also posit that if we see a section of route like the following: [hop x-1]:[city A/ISP 1] -> [hop x]:[unknown city/ISP 2] -> [hop x+1]:[city B/ISP 2], [hop x] is likely located in [city A]. This is due to the fact that a handoff between ISPs must occur in an IXP, data center, or similar. See tr_id 3012 as an example: hop 14 has been corrected to San Jose, despite no useful hostname/latency information. This is due to the fact that the handoff between ASN174 and ASN7018 occurs between hop 13 and hop 14 (likely in an IXP) and therefore hop 13 and hop 14 are more likely to be in the same city.

Similarly, all other things being equal, routers will be clumped nearer to a handoff between ISPs. If a sequence of routers with the same ISP contains a router with no locational information or latency differences (this is quite rare), then the router is likely second last in the sequence, is followed by a router that hands off the packets to a different ISP and should be located in the same city as the final router. So: [hop x-1]:[city A/ISP 1] -> [hop x]:[unknown city/ISP 1] -> [hop x+1]:[city B/ISP 1], then [hop x] is more likely to be located in city B. This is due to the fact that multiple routers are often needed in one location when handing off traffic. In tr_id 1802, hop 9 is corrected to the same location as hop 10 based on this rationale.

Notes:

- this type of geocorrection is labour intensive, since it cannot be done in bulk; each IP address needs to be researched independently. Therefore, we

generally only used it for IP addresses that appeared in many hops or appeared in our showcase routes
- we have been working under the assumption that Maxmind is reasonably good at locating the final hop to a city level of geoprecision
- hop 0 can be known when the origin of a route is specified. In fact, hop 0 is often known with greater certainty than any other hop. This is useful when attempting to correct IP addresses that appear in early hops.

The following types of selects are most used when using this methodology:

```
#
select traceroute_id,hop,t.ip_addr,lat,long,hostname,asnum,gl_override into temp
table cm1 from tr_item as t join ip_addr_info as i on t.ip_addr=i.ip_addr group by
traceroute_id,hop,t.ip_addr,lat,long,hostname,asnum,gl_override order by
traceroute_id,hop;
+
select traceroute_id,hop,ip_addr,lat,long,hostname,dest,submitter from cm1 as c
join traceroute as t on c.traceroute_id=t.id where asnum=7018 and hostname
like '%sea%' group by
traceroute_id,hop,ip_addr,lat,long,hostname,dest,submitter order by
traceroute_id,hop;
#
```

This selects all hops in all routes that have the particular characteristics (in this case, asnum 7018 and hostname like 'sea'). The routes can then be examined manually to see if pre/post/latency methods can be applied.

Final notes on specific IP addresses, sets of IP addresses or smaller ASNs

ASN# 20183: 209.196.50.135 209.196.16.35
This ASN belongs to VeriCenter (now owned by SunGard). VeriCenter does hosting, SunGard provides software to educational and public sector orgs (this is the connection to UFile, we believe). Vericenter head office is in Houston. http://investing.businessweek.com/research/stocks/private/snapshot.asp?privcapId=106202. IPs listed have been corrected from Houston to Atlanta based

on Pre/Post/Latency. We have not used a generic Atlanta L/L because pre/post hops have a specific L/L in Atlanta (which was used instead).