A Learning-based Approach for IP Geolocation

Brian Eriksson, Paul Barford, Joel Sommers[†], and Robert Nowak

University of Wisconsin - Madison, [†]Colgate University
bceriksson@wisc.edu, pb@cs.wisc.edu, jsommers@colagate.edu, nowak@ece.wisc.edu

Abstract. The ability to pinpoint the geographic location of IP hosts is compelling for applications such as on-line advertising and network attack diagnosis. While prior methods can accurately identify the location of hosts in some regions of the Internet, they produce erroneous results when the delay or topology measurement on which they are based is limited. The hypothesis of our work is that the accuracy of IP geolocation can be improved through the creation of a flexible analytic framework that accommodates different types of geolocation information. In this paper, we describe a new framework for IP geolocation that reduces to a machine-learning classification problem. Our methodology considers a set of lightweight measurements from a set of known monitors to a target, and then classifies the location of that target based on the most probable geographic region given probability densities learned from a training set. For this study, we employ a Naive Bayes framework that has low computational complexity and enables additional environmental information to be easily added to enhance the classification process. To demonstrate the feasibility and accuracy of our approach, we test IP geolocation on over 16,000 routers given ping measurements from 78 monitors with known geographic placement. Our results show that the simple application of our method improves geolocation accuracy for over 96% of the nodes identified in our data set, with on average accuracy 70 miles closer to the true geographic location versus prior constraint-based geolocation. These results highlight the promise of our method and indicate how future expansion of the classifier can lead to further improvements in geolocation accuracy.

1 Introduction

There are many ways in which the structural and topological characteristics of the Internet can be considered. One way that has significant implications for advertisers, application developers, network operators and network security analysts is to identify the *geographic location* of Internet devices (*e.g.,* routers or end hosts). Geographic location can mean the precise latitude/longitude coordinates of a device or a somewhat more coarse-grained location such as within a zip code, city, county or country.

There are a number of challenges in finding the geographic location of a given Internet device. The most obvious is that there is no standard protocol that provides the position of any device on the globe (although DNS entries can include a location record). Furthermore, Internet devices are not typically equipped with location identification capability (*e.g.,* GPS, although this may change in the future), and even if they did, some would consider this information private. Prior methods have

focused on identifying the geographic location of an Internet device based on its position relative to a set of active measurements from landmarks with known positions. While these methods have been shown to be capable of producing relatively accurate geographic estimates in some areas, inaccuracies remain for a variety of reasons. Principal among these is the fact of inconsistent density of specific measurements across the globe.

The goal of our work is to broadly improve IP geolocation accuracy over prior methods. Our hypothesis is that the large estimation errors caused by imperfect measurements, sparse measurement availability, and irregular Internet paths can be addressed by expanding the scope of information considered in IP geolocation. The estimation framework that we develop to test this hypothesis is to cast IP geolocation as a *machine learning-based classification* problem. This extensible approach enables information from multiple datasets to be fused such that areas that have low information content from one measurement can be compensated with better information content from other measurements.

To flesh out this framework in order to test our hypothesis, we must select both a classification method and a set of measurements that can be used to estimate IP geolocation. We develop a *Naive Bayes* estimation method that assigns a given IP target to a geographic partition based on a set of measurements associated with that IP target. Given the potentially large number of measurements to an IP target, probability likelihood estimation is simplified by a Naive Bayes approach. The network measurement data considered in this framework includes latency and hop count from a set of landmarks to an IP target. We also include population density in the framework as a demonstration of a non-network measurement that can help refine the estimates. The selection of this classifier/measurement combination was made to demonstrate the potential of this new approach, but is not meant to be definitive nor comprehensive.

To test and evaluate the capabilities of this initial instance of our learning-based approach, we consider geographic partitioning at the level of counties in the continental United States [1]. While considerable Internet topology lies outside the continental United States, the initial validation on this dataset will motivate future work on end hosts located outside the United States. We identified a target set of 114,815 spatially diverse nodes in the Internet through full mesh `traceroute` probing from Planetlab nodes, supplemental data from the iPlane [1] project, and careful alias resolution. For ground truth on the geographic location of these target nodes, we used the Maxmind database [2] as a validation set for our methodology. Of the 114,815 IP target nodes identified in our measurements, 16,874 were identified in the Maxmind database as being within the United States with known city locations. Due to its use as a commercial product, the exact underlying methodology for the Maxmind database is not available, although extensive use of user-survey geolocation information is known to be used.[2] For that set of 16K target

[1] Finer-grained partitioning on the order of zip codes or city blocks is certainly feasible in our framework, but county-level was selected due to the availability of data for test and evaluation.

[2] Due to its dependence on user generated data, updating the Maxmind database requires extensive user surveying that is not needed with our learning-based methodology.

nodes, we then gathered hop count and latency measurements from 78 PlanetLab nodes located in the United States, which were the starting point for our assessment.

We selected a subset of target nodes[3] for training our classifier, with the training set nodes having both known measurements to the monitors and known geolocation. With the remaining nodes, we compare the geolocation estimates of both our learning-based approach and Constraint-Based Geolocation (CBG) [3] (the current state-of-the-art geolocation algorithm using ping measurements) validated against the locations found using the Maxmind database. We find that our estimator is able to provide better location estimates that CBG for 96% of the nodes and on average provide an estimate that is 70 miles closer to the true location. We believe that these results make a compelling case for future development of learning-based methods for IP geolocation.

2 Learning-based IP Geolocation

Given a single target IP address, *can we determine the geographic location of the target IP?* Consider a single target IP address with a set of measurements from a set of monitors with known geolocation to this target IP address. For the purposes of this work, the measurement set $\mathcal{M}$ ($= \{m_1, m_2, ..., m_M\}$) is the collection of both latency and hop count values going from the monitor set. Without loss of generality, now consider a set of possible counties in the continental United States ($\mathcal{C}$), such that the target is located in some county $c \in \mathcal{C}$. This changes the underlying problem to, *Given the measurement set $\mathcal{M}$, can we estimate which county $c \in \mathcal{C}$ the target IP is located in?* The best classifier would choose the county ($\widehat{c}$) that the target is most probably located in given the measurement set, $\widehat{c} = \arg\max_{c \in \mathcal{C}} P(c \mid \mathcal{M})$. Using Bayes Theorem [4] ($P(A|B) = \frac{P(B|A)P(A)}{P(B)}$), therefore we can restate the classifier as:

$$\widehat{c} = \arg\max_{c \in \mathcal{C}} P(c \mid \mathcal{M}) = \arg\max_{c \in \mathcal{C}} P(\mathcal{M} \mid c)\, P(c)$$

Where the value $P(\mathcal{M})$, the probability of observing the set of measurements, can be ignored due to this value being constant across any choice of county c.

Next, we expand our estimation framework to consider features other than measurements from monitors to IP targets. Given that the targets in this paper are routers, we can use the work in [5] to inform where these routers should be geographically located. Specifically, the value $P(c)$, the probability of classifying a target in county c, will be chosen using the results showing that the number of routers in a specific geographic location is strongly correlated with the population of that geographic location. Therefore, we can estimate the probability of classifying into a given county to be the population of that county divided by the total population in all the counties under consideration.

$$\widehat{P}(c_i) = \frac{\text{Population of } c_i}{\sum_{j \in \mathcal{C}} \text{Population of } c_j} \tag{1}$$

[3] We consider IP addresses and nodes to be equivalent in this paper since even if alias resolution on routers is imperfect, it should not affect our empirical results.

How can we estimate the value $P(\mathcal{M} \mid c)$, the probability likelihood of a measurement set $\mathcal{M}$ being observed given the target is located in county c? Given a set of training data, a set of IP addresses with known measurement sets $\mathcal{M}$ and locations c, we could use off-the-shelf techniques (kernel density estimators, histograms, etc.) to estimate the multivariate likelihood density $P(\mathcal{M} \mid c)$. A problem is that the set $\mathcal{M}$ is most likely of high dimensions (with dimensionality equal to the number of hop count and latency measurements observed to this target, in this case, on the order of 100), and most density estimator techniques have an error rate that increases quickly with the dimension of the problem [4].
If all of the values of M were statistically independent from each other, then the likelihood density could be restated as:

$$\begin{aligned} P(\mathcal{M} \mid c) &= P(\{m_1, m_2, ..., m_M\} \mid c) \\ &\approx P(m_1 \mid c)\, P(m_2 \mid c) ... P(m_M \mid c) \end{aligned} \tag{2}$$

This converts the problem from estimating one M-dimensional density to estimating M one-dimensional densities. However, it should be assumed that there is a large degree of correlation between measurements, with prior work in [6] showing correlation between hop count measurements, and work in [7] showing correlation between latency measurements. The risk of assuming statistical independence between measurements is informed by empirical studies on highly dependent data in [8]. That work shows that for classification, there is little penalty for assuming statistical independence even when the measurements are highly statistically dependent. This is due to classification performance depending only on the most-probable class (in this case, county region) likelihood probability being greater than other class likelihood probabilities, not the goodness-of-fit of our estimated likelihood probability to the true likelihood probability.

2.1 Kernel Density Estimators

The next step in our learning-based framework is to estimate the one-dimensional densities, $P(m_i \mid c)$, the probability of the measurement value m_i being observed given that the target is located in county c. Consider a set of training data, where for each training target, both the measurement set $\mathcal{M}$ and the geolocation county c is known. Given the known monitor placement, for the entire training set we can determine the distance vector $\mathbf{d} = \{d_1, d_2, ..., d_M\}$, where d_i is the distance between the monitor associated with measurement m_i and county c. These measurements with distance ground truth can then be used to learn the density (the probability of observing measurement m_i given that the target is located d_i distance away from the monitor associated with measurement m_i).
Simple density estimators, such as histograms, can be used, but the lack of smoothness in the estimated density can hurt performance. Instead, we will look to use Kernel Density Estimators [4], which use the summation of smooth kernel functions to estimate the density. This smoothness in the estimated density allows improved estimation of the true density given the limited size of our training set.

For hop count measurements, a one-dimensional density will be estimated at each hop count value ranging from one hop away from a monitor to ten hops away (it is assumed that any distance longer than ten hops will not help in estimating distance). For latency measurements, due to the limited amount of training data, the measurements are aggregated together separated by 10ms, with a single estimated one-dimensional density for 0-9ms, a separate one-dimensional density for 10-19 ms, 20-29ms, etc. An example of a kernel estimated density for latency measurements can be seen in Figure 1 along with the resulting probability distribution across the US counties for observing this latency measurement to a monitor with known geolocation.

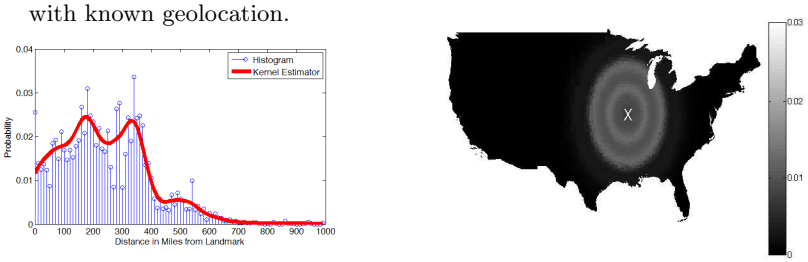

Fig. 1. (Left) - Probability for latency measurements between 10-19ms being observed given a target's distance from a monitor. Stem plot - Histogram density estimation, Solid line - Kernel density estimation. (Right) - The kernel estimated probability of placement in each county given latency observation between 10-19ms from a single monitor marked by 'x'.

2.2 Estimation Weights

The amount of location information from latency measurements is likely to be of more use than the location information derived from hop count measurements or population data. Therefore we introduce two weights λ_{hop} and λ_{pop} as the weights on the hop count measurements and the population density data respectively. Informed by the geolocation improvement by using measurement weights in the Octant framework [9], the ordering of the measurements should also imply some degree of importance, as the location of the monitor with the shortest latency measurement to the target should inform the classifier more than the monitor with the 30-th closest latency measurement. Therefore, we will also weight the ordering of measurement values by an exponential, such that the i-th latency measurement is weighted by $\exp(-i \cdot \gamma_{lat})$ and the j-th hop count measurement is weighted by $\exp(-j \cdot \gamma_{hop})$. The weight parameter values $(\lambda_{hop}, \lambda_{pop}, \gamma_{lat}, \gamma_{hop})$ will be found by the weight values that minimize the sum of squared distance errors between the training set of IPs known locations and the Naive Bayes estimated locations.

2.3 Methodology Summary

Dividing the measurement set $\mathcal{M}$ into the set of latency measurements $\{l_1, l_2, ..., l_m\}$ and the set of hop count measurements $\{h_1, h_2, ..., h_m\}$ (where the total number of measurements $M = 2m$), our learning-based classifier from Equation 2 can be restated using the kernel density estimators (where instead of the true likelihood $P(m_i \mid c)$ we have the kernel estimated $\widehat{P}(m_i \mid c)$), the weight terms, and the monotonic properties of

the logarithm function as:

$$\widehat{c_i} = \arg\max_{c \in \mathcal{C}} \left(\lambda_{pop} \log \widehat{P}(c) + f_{hop} + f_{lat} \right) \quad (3)$$

Where $f_{hop} = \lambda_{hop} \sum_{j=1}^{m} \exp(-j \cdot \gamma_{hop}) \log \widehat{P}(h_j \mid c)$, and $f_{lat} = \sum_{j=1}^{m} \exp(-j \cdot \gamma_{lat}) \log \widehat{P}(l_j \mid c)$, and the term $\widehat{P}(c)$ for the 3,107 counties in the continental United States is found using Equation 1.
A summary of the complete methodology is seen in Algorithm 1. Note that all the computational complexity of this algorithm is on training the parameters $(\lambda_{hop}, \lambda_{pop}, \gamma_{lat}, \gamma_{hop})$. Each target is geolocated using only $O(M|\mathcal{C}|)$ number of multiplications, where $|\mathcal{C}|$ is the total number of location classes under consideration (in this paper, the number of counties in the continental United States), and M is the total number of measurements to the current target IP.

Algorithm 1 - Naive Bayes IP Geolocation Algorithm

Initialize:
- Measure the hop-count and latency measurements from every monitor to a training set with known geographic locations.
- Using a population density database, find $\widehat{P}(c)$ for all $c \in \mathcal{C}$ using Equation 1.
- Using kernel density estimators, estimate the one-dimensional distribution $\widehat{P}(m|c)$ for every measurement $m \in \mathcal{M}$.
- Find the optimal values for $\lambda_{hop}, \lambda_{pop}, \gamma_{lat}, \gamma_{hop}$ that minimize the sum of squared distance errors over the training set.

Main Body

1. For each target IP with unknown geography, estimate the location $\widehat{c}$ by Equation 3.

3 Experiments

3.1 Measurement Dataset

To assess our geolocation algorithm, we sought a large set of IP addresses of routers with as much spatial diversity as possible within the continental United States. Starting with the spatially diverse set of Planetlab [10] node locations, the full mesh `traceroute` probing between these nodes will find a very large set of router IP addresses with high spatial diversity. Existing data were provided by the iPlane project [1], which performs a `traceroute` from all available Planetlab hosting sites to a set of target prefixes obtained through the Routeviews project [11]. We used four weeks of iPlane data collected over the period of 12 December 2008 to 8 January 2009. In addition to the iPlane data, we collected `traceroute` data between a full mesh of Planetlab hosting sites, of which there were 375 at the time we collected these data. For performing traceroutes, we used the Paris traceroute tool [12], using it once in UDP mode and a second time in ICMP mode in order to discover as many routers as possible [13]. Options were set in the Paris traceroute tool so that it produced a low level of probes while taking somewhat longer to complete a given traceroute. We collected a full mesh of Planetlab traceroute measurements three separate times between December 11, 2008 and January 6, 2009. For these measurements, we were able to use about 225 Planetlab sites due to maintenance and other issues.

Using these two data sets, we were able to discover 125,146 unique router IPv4 addresses. A standard problem with traceroute-based studies is IP interface disambiguation, also known as *alias resolution*. Interfaces on a given Internet router are typically assigned separate IP addresses; identifying which addresses correspond to the same physical router same physical router is the challenge in alias resolution. To de-alias our data set, we used the alias database published by the iPlane project. This database builds on prior work in alias resolution, including the methods used by the Rocketfuel project [14]. Upon de-aliasing our set of router IP addresses, we identified 114,815 routers.

To construct the measurements used in our analysis (as described below), we required the hop counts and latency measurements to each identified router from all available Planetlab sites. In order to limit the overhead of probing for this hop count and latency data, we used the following approach. For each IP address, we sent a direct ICMP echo request packet (*i.e.*, a ping). In other work, it was observed that a majority of Internet hosts respond to ICMP echo request packets [15]; we also found this to be true. Indeed, more than 95% of all router IP addresses we identified responded. This should not be surprising considering the fact that these addresses were initially identified through active probing. For computing the hop count, we use the methodology of [16] on the echo response (note that this is the hop count of the reverse path).

3.2 Results

For geolocation ground truth, we use the Maxmind database [2], which is rated to be 82% accurate within 25 miles for IPs located within the US. From our dataset of 114,815 disambiguated routers, Maxmind identified 16,874 routers located in the continental United States with known county location. Using 5-Fold Cross Validation [4], we test the performance of the methodology five times using 20% of the routers as our training set, leaving the remaining 80% of the routers to test the accuracy of our methodology.

We compare the geolocation results from our learning-based method to Constraint-Based Geolocation (CBG). To generate CBG geolocation estimates, we implemented the algorithm described in [3]. CBG is the current state-of-the-art IP geolocation methodology using only ping-based measurements. The basic intuition behind CBG is that each latency measurement to a set of monitors with known location can be considered a series of constraints, where given speed-of-light in fiber assumptions and self-calibration using a set of training data, we can determine a feasible geographic region given each latency measurement. Given a series of latency measurements, the possible geographic placement is considered the intersection of many constraint regions, with the estimated location behind the centroid of this intersection region.

To assess performance of both geolocation algorithms, we will consider the error distance to be the distance in miles between the centroid of our estimated classified county and the centroid of the ground truth (Maxmind) county. Performance of our learning-based Naive Bayes framework and the CBG method with respect to the empirical cumulative probability can be seen in Figure 2-(left). As seen in the figure, the geolocation

estimates produced by our learning-based framework are more accurate than CBG for 96% of the routers. On average the Naive Bayes location estimates are 70 miles closer to the true location than the CBG estimates.

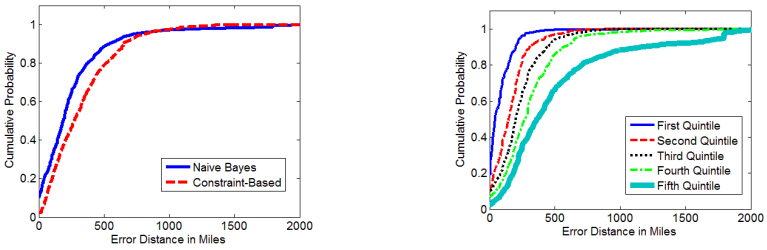

Fig. 2. (Left) - Empirical cumulative probability of error distance. (Right) - Breakdown of each quintile empirical cumulative probability error distance for our learning-based methodology.

3.3 Quintile Confidence Levels

Using Equation 3, the Naive Bayes framework will find $\widehat{P}(\widehat{c} \mid \mathcal{M})$, the estimated probability of each target being classified correctly by our learning-based framework given the set of measurements. This can be considered a level of confidence in the classification of each target IP. Using this confidence level, we can sort into quintiles, from a quintile set containing the 20% of the target IPs with the largest $\widehat{P}(\widehat{c} \mid \mathcal{M})$ values (*e.g.,* the targets we are most confident in accurately geolocating), to a quintile set containing the 20% of target IPs with the smallest $\widehat{P}(\widehat{c} \mid \mathcal{M})$ values (*e.g.,* the targets we are least confident in). Figure 2-(right) shows how this confidence level accurately predicts the quality of our classification, with the most confident 20% of the targets being classified far more accurately than any other quintile set. Therefore, in addition to estimating the geolocation of each target IP, we also have a level of confidence that directly corresponds to the accuracy of our prediction.

3.4 The Impact of Additional Information

To assess the impact of using multiple features in our learning-based framework, we generate geolocation estimates when both population density information is removed (setting the weight of using the population density to zero, $\lambda_{pop} = 0$) and when hop count information is removed (setting the weight of using the hop count data to zero, $\lambda_{hop} = 0$). Table 1 shows the average distance error for using various combinations of our measurements, and the results indicate that both the hop count data and the population density information significantly contribute to the improved performance of the methodology. Using only latency information, the Naive Bayes methodology still outperforms the CBG method due to the more accurate multiple latency density estimates used to classify the location of each end host instead of simply using the intersection of feasible latency regions as in the CBG methodology.

4 Related Work

The main prior work in IP geolocation that we compared and contrasted our learning-based methodology with is Constraint-Based Ge-

Table 1. Average Error Distance (in miles)

CBG Method	Naive Bayes (latency, hop count, pop.)	Naive Bayes (latency, hop count)	Naive Bayes (latency, pop.)	Naive Bayes (latency)
322.49	253.34	261.89	277.29	278.96

olocation [3]. More recent geolocation work in [9],[17] has found improvements over Constraint-Based Geolocation, but both methodologies require `Traceroute`-based measurements to the targets along with location hints acquired by unDNS [14] probes. One potential disadvantages of these methodologies is the dependency on DNS naming conventions, which have been shown to not always be reliable [18]. This requires sophisticated location validation and reweighting mechanisms to be developed and maintained. The focus of this work was to introduce our elegant learning-based geolocation framework and validate its performance using simple ping-based measurements. We leave the extension of our learning-based framework to these newer `Traceroute`-based methodologies as future work. To the best of our knowledge, this is the first work to frame IP geolocation as a machine learning problem.

5 Conclusions and Future Work

The goal of our work is to improve the accuracy of estimates of the geographic location of nodes in the Internet. Our work is based on the hypothesis that the ability to zero in on the geolocation of nodes is improved by considering a potentially broad set of features including both active measurements and more static characteristics associated with locations. To consider this hypothesis, we introduce a learning-based framework that enables geolocation estimates to be generated efficiently, and is flexible in the feature space that can be considered. In this initial study, we employ a Naive Bayes classifier and generate estimates from two types of empirical measurements in our framework (latency and hop counts) and one societal characteristic (population density). We then test the feasibility of our learning-based approach using an empirical dataset of over 16K target routers, and latency and hop count data to 78 monitors with known geographic locations. We show that our geolocation estimates are more accurate for 96% of the routers in our test set versus the estimates generated by a current state-of-the-art constraint-based geolocation method. We also show how the use of multiple features does indeed enhance the overall estimation accuracy. In future work, we plan to investigate additional features that improve the accuracy of our estimates, and the possible use of a multi-scale classification framework that narrows the classification region given classification confidence levels.

References

1. H. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iPlane: An Information Plane for Distributed Services," in *USENIX OSDI '06*, November 2006.

2. "Maxmind geolocation database http://www.maxmind.com."
3. B. Gueye, A. Ziviani, M. Crovella, and S. Fdida, "Constraint-based geolocation of internet hosts," in *IEEE/ACM Transactions on Networking*, December 2006.
4. L. Wasserman, "All of Nonparametric Statistics," May 2007.
5. A. Lakhina, J. Byers, M. Crovella, and I. Matta, "On the Geographic Location of Internet Resources," in *IEEE Journal on Selected Areas in Communications*, August 2003.
6. B. Eriksson, P. Barford, and R. Nowak, "Network Discovery from Passive Measurements," in *Proceedings of ACM SIGCOMM '08*, August 2008.
7. E. Ng and H. Zhang, "Predicting Internet Network Distance with Coordinate-baseed Approaches," in *Proceedings of IEEE INFOCOM '02*, April 2002.
8. I. Rish, "An Empirical Study of the Naive Bayes Classifier," in *Workshop on Empirical Methods in Artificial Intelligence*, 2001.
9. B. Wong, I. Stoyanov, and E. Sirer, "Octant: A comprehensive framework for the geolocation of internet hosts," in *USENIX NSDI 2007*, April 2007.
10. A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak, "Operating System Support for Planetary-Scale Network Services," in *USENIX NSDI '04*, March 2004.
11. "University of Oregon Route Views Project," http://www.routeviews.org/.
12. B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira, "Avoiding traceroute anomalies with Paris traceroute," in *Proceedings of ACM IMC '06*, October 2006.
13. M. Luckie, Y. Hyun, and B. Huffaker, "Traceroute Probe Method and Forward IP Path Inference," in *Proceedings of ACM IMC '08*, October 2008.
14. N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP Topologies with Rocketfuel," in *Proceedings of ACM SIGCOMM '02*, August 2002.
15. J. Heidemann, Y. Pradkin, R. Govindan, C. Papadopoulos, G. Bartlett, and J. Bannister, "Census and Survey of the Visible Internet," in *ACM IMC '08*, October 2008.
16. H. Wang, C. Jin, and K. Shin, "Defense against spoofed IP traffic using hop-count filtering," *IEEE/ACM Transactions on Networking*, vol. 15, no. 1, pp. 40–53, 2007.
17. E. Katz-Bassett, J. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe, "Towards IP Geolocation Using Delay and Topology Measurements," in *ACM IMC '06*, October 2006.
18. M. Zhang, Y. Ruan, V. Pai, and J. Rexford, "On the Geographic Location of Internet Resources," in *In Proceedings of USENIX Annual Technical Conference*, 2006.