Internet Geolocation and Evasion

James A. Muir P. C. van Oorschot

School of Computer Science
Carleton University, Ottawa, Canada
{jamuir,pvo}@scs.carleton.ca

8 April 2006

Abstract

Internet geolocation technology (IP geolocation) aims to determine the physical (geographic) location of Internet users and devices. It is currently proposed or in use for a wide variety of purposes, including targeted marketing, restricting digital content sales to authorized jurisdictions, and security applications such as reducing credit card fraud. This raises questions about the veracity of claims of accurate and reliable geolocation, and the ability to evade geolocation. We begin with a state-of-the-art survey of IP geolocation techniques and limitations, and examine the specific problems of (1) approximating a physical location from an IP address; and (2) approximating the physical location of an end client requesting content from a web server. In contrast to previous work, we consider also an *adversarial model*: a knowledgeable adversary seeking to evade geolocation. Our survey serves as the basis from which we examine tactics useful for evasion/circumvention. The adversarial model leads us to also consider the difficulty of (3) extracting the IP address of an end client visiting a server. As a side-result, in exploring the use of proxy servers as an evasionary tactic, to our surprise we found that we were able to extract an end-client IP address even for a browser protected by Tor/Privoxy (designed to anonymize browsing), provided Java is enabled. We expect our work to stimulate further open research and analysis of techniques for accurate and reliable IP geolocation, and also for evasion thereof. Our work is a small step towards a better understanding of what can, and cannot, be reliably hidden or discovered about IP addresses and physical locations of Internet users and machines.

1 Introduction and Motivation

The Internet connects hosts from all across the world. Sometimes it is desirable to know where, geographically, a particular host is. Informally, *Internet geolocation* is the problem of determining the physical location (to some level of granularity) of an Internet user. This is often also called *IP geolocation*, since every host directly connected to the Internet is identified by a unique IP address. A growing number of companies (e.g., Akamai, Digital Envoy, MaxMind, Quova, and Verifia) now maintain and licence databases which map IP addresses to geographic locations.

The development of IP geolocation technology is being driven by a number of applications; among the most lucrative is targeted advertising: if a host serving a web page is able to determine that a visiting client's IP address is in, say, Vancouver, then the server can embed in that page advertisements targeted to Vancouver customers (e.g., umbrellas). Other suggested applications of IP geolocation include automated redirection to nearby servers and web analytics (i.e., analyzing web page access logs to extract marketing data). Web sites often tailor content (other than advertisements) based on geographic location. For example, content served by Google undergoes automated country redirection based on client IP address.

We are particularly interested in security applications, and those in which users may have incentive to evade geolocation. This includes cases in which privacy is important (e.g., for personal or humanitarian

reasons) or related to either detecting or hiding illegal activity. IP geolocation is used for reducing losses due to credit card fraud (fraud rates are known to vary significantly for transactions originating from certain areas in the world), spam filtering (based on countries involved in relaying a message), securing remote network logins, and restricting distribution of digital content (e.g., to jurisdictions authorized by local regulations). It is also being promoted for use in reducing identity theft.[1]

The first and most basic question related to IP geolocation is whether it is actually possible – to what level of granularity, and for what fraction of IP addresses or end users.[2] In this paper, we are interested in exploring what techniques are currently available for IP geolocation and to what degree they can be evaded. An important consideration here is the potential use of proxy servers and anonymizing tools. In contrast to our work, the academic literature to date on IP geolocation techniques (e.g., [17, 15, 6]) has generally implicitly assumed that no evasive action is being taken.

A second interesting question is an ethical one: should IP geolocation technology be used, in light of the potential for abuse. We leave this second issue to privacy advocates, human rights activists, and legal scholars (e.g., see [10]). Our main focus is on technical capabilities, so that we may develop a better understanding of what is, and is not, currently possible technologically.

Geolocation and legal questions. Two recent court cases highlight the uncertainty surrounding the capabilities of Internet geolocation technology, and the difficulties this leaves the courts in. Aside from privacy, a major legal question is the ability to use geolocation technology to censor content download.

The case of Yahoo! v. The League Against Racism and Anti-Semitism [20] involved the sale of Nazi objects by Internet. An expert panel, consisting of Vinton Cert (U.S.), Bennet Laurie (U.K.) and François Wallon (France), agreed that Yahoo! could not render impossible access by French citizens to the offending materials for several reasons, including: (1) nationality based on IP address is only correct for 70% of French citizens, and (2) this is easily circumvented. However, the expert panel reported receipt of several submissions from commercial organizations asserting their technology could enforce the French court's censorship request.

The Nitke v. Ashcroft [14] case challenged the Communications Decency Act, and involved the question of posting on the web images considered in some regions to violate obscenity laws. Obscenity is determined by "local community standards"; thus a downloading client's location is relevant. The question arose as to the feasibility of mandating use of geolocation technology to enforce regional censorship of content (blocking downloads based on state or district). Expert testimony diverged on the accuracy of geolocation technology software, varying from 60-95%.

Our Contributions. We begin by providing a state-of-the-art survey of known techniques for mapping a numeric IP address to a geographic location, the first (to our knowledge) in the open literature integrating published literature [17, 15, 6] and patents [16, 8, 1]. From this foundation, we then examine these techniques from the viewpoint of an adversary, identify individual limitations, and provide the first academic study (to our knowledge) considering how a knowledgeable individual might try to evade Internet geolocation. We identify and explore the relationship between three distinguishing sub-problems related to Internet geolocation: user geolocation, IP geolocation, and IP address extraction. We consider several evasionary tactics involving the use of non-local IP addresses. For one of these, using a proxy (e.g., Tor/Privoxy [5, 18]), we find that provided Java is enabled, we are still able to extract an end-user IP address (contrary to the natural expectation of Tor users).[3] We also propose a new IP geolocation technique based on timing of HTTP refreshes, which has advantages over previous methods.

Organization. The remainder of the paper is organized as follows. Section 2 surveys and classi-

[1] See "Business Uses / Fraud Detection & Prevention" at `http://www.quova.com`.

[2] Needless to say, commercial organizations offering IP geolocation services make strong marketing claims.

[3] We communicated with two of the main Tor designers [4] and, apparently, this kind of attack is not entirely surprising; but, as of this writing, it was not prominently warned against on the EFF web site, `http://tor.eff.org`.

fies known IP geolocation technologies along with their respective limitations. Section 3 considers how a knowledgeable adversary might attempt to circumvent these technologies. Section 4 explores two new geolocation tools which may be of use even in the face of evasionary tactics: the use of a Java applet capable of extracting an IP address even from clients protected by Tor/Privoxy, and a new timing-based geolocation technique using HTTP refreshes. Section 5 discusses related work. Section 6 provides concluding remarks.

2 Survey of IP Geolocation Techniques

In this section we survey the state-of-the-art of IP geolocation, including nine separate approaches which we classify into three categories as indicated in Table 1. Along with each approach we also include our preliminary comments on limitations; these are discussed further and summarized in §3.

Our discussion follows a "cookbook" style, allowing one to carry out a number of these techniques from their own machine. For completeness, we include several techniques which are well-known to some, but facilitate subsequent discussion on evasion.

Many of the techniques are in use commercially, often in combination. Exactly which techniques can be used depend upon context since, in different situations, different information is available. In essence, the more (correct) information there is, the more clues there are to determine a location. However, regardless of context, for this section we assume at least an IP address as a minimal starting point.

category	approach	section
A. information intentionally registered in databases	*whois* look-up by IP address	2.1
	whois look-up by Autonomous System	2.2
	whois look-up by domain name	2.3
	DNS LOC records	2.4
B. information leaked	geographic codes in domain name	2.5
	user- or application-submitted information	2.6
C. network routing and/or timing information	approximation through round-trip times using `ping`	2.7
	inference based on routing data; e.g., BGP or `traceroute`	2.8
	reconnaissance of ISP networks	2.9

TABLE 1: **Classification of IP Geolocation Approaches.**

2.1 `whois` Look-up by IP Address

Information about an IP address is most easily obtained by look-up in pubic *whois* databases, which are maps between logical Internet identifiers (autonomous system numbers, domain names and IP addresses) and real-world entities. This allows one to determine the entity to whom a given IP address is registered (e.g., an ISP, company, organization, end user, etc.). Contact information provided for the entity usually includes email address, telephone number and mailing address. Geographic location can be inferred from telephone numbers and mailing addresses. One intended use of this information is to help users diagnose and resolve network problems; these databases list a "point of contact" for a particular part of the network.

ICANN controls the IP address space. IANA, previously in control, is now controlled by ICANN. A summary of IPv4 address space allocation can be found at `http://www.iana.org/assignments/ipv4-address-space`. IANA previously allocated blocks of IP addresses directly to entities. Now, ICANN/IANA allocates address blocks to five Regional Internet Registries (RIRs) who, in turn allocate sub-blocks to entities in their respective regions. The whois servers of the RIRs are: `whois.afrinic.net`, `whois.apnic.net`, `whois.arin.net`, `whois.lacnic.net`, `whois.ripe.net`.

For example, the response to the command (or equivalent web form request) `whois -h whois.arin.net -- 134.117.225.13` indicates that IP addresses in the block 134.117.0.0/16 are registered to "Carleton University". From the listed organizational address information alone, one might infer (correctly) that 134.117.225.13 is a machine at Carleton University in Ottawa, Canada. Entering the postal code K1S 5B6 into a web based mapping application (e.g., Mapquest, Google Maps) converts a postal code to an approximate latitude and longitude.

Limitations:

1. (geographic precision) Not all target hosts are located at or near the address of the organization to which the host's IP address is registered. For example, one author's home IP address in Ottawa falls in the address block 70.24.0.0/13. ARIN lists the registrant for this block as: Rogers Cable Inc. (ROCA), One Mount Pleasant, Toronto, Ontario, M4Y 2Y5, Canada. So this technique would geolocate the author's PC (with very poor precision) to Toronto, 450 km from its true Ottawa location (albeit, still with correct country-level resolution). More generally, this technique may map large address blocks to a single location, even when not all addresses in the block are used by individual devices at that location.
2. (falsifiability) The whois database data is provided by registrants, who may submit false or incorrect data.
3. (freshness) The whois database records usually indicate a "last-updated" date. Data which has not been updated for several years may no longer be (completely) correct.

2.2 `whois` Look-up by Autonomous System (AS)

AS numbers – another logical identifier administered by ICANN useful for IP geolocation – are globally unique 16-bit integers used by routing protocols like BGP. Each RIR holds blocks of AS numbers allocated by ICANN/IANA (see `http://www.iana.org/assignments/as-numbers`). Organizations apply to their RIR to obtain an AS number. Each RIR's whois databases contain information on AS numbers they have assigned, and to whom. Every publicly routeable IP address is associated with an AS number. Given an IP address, we can determine the number of the AS in which it resides and then obtain details about the AS from a public whois database.

One method to obtain AS information for a given IP address is from a *BGP routing table*, which contains IP prefixes and sequences of AS numbers; for each IP prefix listed there is at least one sequence of AS numbers (often, several). The AS numbers describe a delivery route *(AS path)* for traffic destined for IP addresses having a particular prefix; the rightmost entry in a sequence is the AS originating those IP addresses. There are publicly viewable BGP tables; the University of Oregon Route Views Project (`http://routeviews.org`) provides several. Routeviews provides telnet access to a Cisco router, which one can log into using `telnet route-views.routeviews.org` with username `rviews`. Queries such as `show ip bgp 134.117.27.71` then return information from its BGP table. The response from this example query indicates that the router knows 53 different routes that it can use to deliver traffic to the addresses in the block 134.117.0.0/16, each route terminating with AS 29773 – the AS where the host 134.117.27.71 originates. Downloading the complete (very large) BGP table, e.g., from `http://archive.routeviews.org/`, allows one to extract AS numbers using standard Unix text processing tools.

Route Views offers another method (perhaps more convenient to some) of looking up AS numbers for a given IP address, based on a *Domain Name System* (DNS) look-up. To obtain the number for the AS which originates 134.117.27.71, we query the DNS database for information on a special domain name composed of the IP address written right to left with suffix `aspath.routeviews.org`. For example, the command `dig 71.27.117.134.aspath.routeviews.org ANY` returns the DNS entry for `71.27.117.134.aspath.routeviews.org` which contains a TXT record. The first string therein

is an AS path (itself extracted from a BGP table). The rightmost number in the path identifies the AS that originates 134.117.27.71 – in this case, 29773, as per the previous method.

Next we look up who AS 29773 is registered to. From `http://www.iana.org/assignments/as-numbers` we find it was allocated to ARIN. We then query the ARIN whois server for details about the registrant using `whois -h whois.arin.net -- AS29773` which returns a database record. Within the record, based on the address for the organization to which the AS is registered, we might infer (correctly) that 134.117.27.7 is in Ottawa, Canada.

Limitations:

1. (geographic precision) Not all target hosts are located at or near the address of the organization which registered their AS number. A large AS can originate many IP prefixes covering a wide geographic area. For example, ARIN's whois record for AS 1239 indicates `OrgName=Sprint, City=Reston, State=Virginia`. Reston has population under 100,000. Thus, it is incorrect to infer that all IP addresses originated by AS 1239 are in Reston, as a snapshot of the BGP routing table used by the router `route-views.routeviews.org` indicates that AS 1239 originates eleven /14 address blocks or $11 \cdot 2^{32-14} = 2{,}883{,}584$ IP addresses.
2. (falsifiability) The whois database may contain false or incorrect data. Typical AS operators would seem to have little incentive to deliberately publish inaccurate information in a whois AS record, since interoperability and quick diagnosis of connectivity problems are priorities. But operators of some ASs might, and insider attacks are always possible.
3. (freshness) The whois database record for an AS may be outdated. However, this may be less likely for an AS record than for an IP address record.

2.3 `whois` Look-up by Domain Name

Often an IP address will map to a *domain name* in the DNS database. Since DNS domain names must be registered, details about registrants are often available in public whois databases. This can be useful for IP geolocation.

Domain names consist of a sequence of dot-separated character-strings called *labels*. The right-most label is the *top-level domain* (TLD), and the label to its left is the *second-level domain*. The sequence of labels, read right-to-left, describes a hierarchy; domain names with more labels are subdomains of domain names with fewer labels. Each immediate subdomain of a TLD must be registered. *Registrar* companies sell domain name registration services, and register subdomains of one or more TLDs. The organization responsible for a TLD maintains a registry of all its second-level domains.

To locate a host with a given IP address, we first check if the IP address maps to a domain name using a *reverse DNS look-up* with a command line tool like `nslookup` or `dig`. The query `dig -x 134.117.225.13` returns an "answer section" indicating that 134.117.225.13 currently maps to `dante.ccsl.carleton.ca`.

We next query IANA to find the organization responsible for the `.ca` TLD. The response to the query `whois -h whois.iana.org -- ca` includes `Whois Server (port 43): whois.cira.ca` and also `URL for registration services: http://www.cira.ca/`. We query this whois server for details about the second-level domain: `whois -h whois.cira.ca -- carleton.ca`. The response to this query finally gives us an address (with postal code) for the entity that registered `carleton.ca`, in this case `Carleton University, CCS, Ottawa, Ontario K1S 5B6 Canada`. So we might infer (correctly) that 134.117.225.13 is also at or near this address. This last response also lists `Registrar: Internic.ca Corp` as the registrar used to register `carleton.ca`. We can also query their whois server for details using `whois -h whois.internic.ca -- carleton.ca`.

Limitations:

1. (completeness) Not all IP addresses map to a domain name.
2. (geographic precision) For target hosts (IP addresses) which map to a domain name, not all are located at or near the address listed in the registration record for that domain name. Moreover, very large classes of Internet hosts can be mapped (incorrectly) to a single location; this is especially problematic for domain names of ISPs (e.g., not all hosts with domain names ending in `aol.com` are in Dulles, Virginia).
3. (public availability of records) Not all registrars make registrant details publicly available. For example, for the domain name `cr.yp.to`, we can try to determine information on the registrant using `whois -h whois.iana.org -- to` and `whois -h whois.tonic.to -- yp.to` but the `tonic.to` whois server does not reveal much information. After investigating the web site `www.tonic.to`, we find an explanation at `http://www.tonic.to/faq.htm`: *Tonic does not maintain a whois database that provides registrant information, as many of our customers consider the public display of this information invasive of their privacy. In fact, we will never sell a mailing list of our customers.* So, it appears `tonic.to` will not provide the registrant's address. However, in this case, the host `cr.yp.to` has a second name, `dancer.math.uic.edu`, which resolves to the same IP address, and the registrant details for the latter are publicly available.
4. (falsifiability) Domain name whois registrant records may contain intentionally false data.
5. (freshness) Domain name whois registrant records may be stale.

2.4 Voluntary DNS LOC records

The DNS database can be used to publicly advertise the geographic location of a host, allowing hosts providing accurate information to be easily located by a few DNS queries. For example, assume a target host `crc.ca`. The response to the DNS query `dig crc.ca ANY` lists a number of *nameservers* (including `ns1.crc.ca`) which are DNS servers that act as authoritative sources of (often further) information for `crc.ca` and its subdomains. Repeating the DNS query to the server `ns1.crc.ca`, as `dig @ns1.crc.ca crc.ca ANY`, yields a response including three further resource records: SOA, LOC, TXT. This LOC record includes: 45 20 30.000 N 75 52 58.600 W 400.00m 1m 10000m 10m. This gives (see RFC 1876 [3, sec.3]) latitude, longitude, altitude, size, horizontal precision, vertical precision.[4] The location specified for `crc.ca` is 45deg 20min 30sec north latitude, 75deg 52min 58.6sec west longitude, altitude 400 meters above sea level. The host `crc.ca` is enclosed in a sphere of diameter 1m centered at this point, but these coordinates are relative to a horizontal circle of error 10,000m in diameter and a vertical circle of error 10m diameter. Despite these error bounds, the latitude and longitude data can be used as an estimate of host location. This could be compared to a database of city latitudes and longitudes to determine the city closest to this point. Such coordinates can be map-located using various on-line websites. For example, the following gives a detailed view of the local geography: `http://maps.google.com/maps?q=45+20'+30%22,-75+52'+58%22&t=h` (here `%22` encodes a double quote character; negative degrees indicates west longitude). The Communications Research Centre (CRC) facility is roughly 400 meters north of the point displayed on the map.

A database of hosts, containing approximately 800 entries, which have DNS LOC records can be found at `http://dns-loc.mapper.ofdoom.com/bulk/`.

Limitations:

[4] The latter three fields permit data which can convey privacy preferences. A location can be specified as being only within a large sphere – perhaps the size of a city or country – if it is undesirable to publicize precise locations.

1. (completeness) Very few hosts have LOC records ([8] estimates fewer than 1%).[5]
2. (falsifiability) The information contained in a DNS LOC record is unverified. It is submitted by users who may choose to publish deliberately misleading data.

2.5 Geographic Codes within Domain Names

Assuming that the IP address of a target host maps to a domain name, the domain name itself may provide geographic information. Identifying and correctly interpreting any *geographic codes* present in a domain name may reveal the location of a target host.

As a well-known example, 245 of the current 264 TLDs are *country code* top level domains (ccTLDs), each consisting of two letters: e.g., Anguilla (`.ai`), Australia (`.au`). If a target host has a domain name that ends with `.au`, then we might guess that it is in located in Australia. The complete list of ccTLDs is given at `http://www.iana.org/cctld/cctld-whois.htm`.

The organizations which manage each ccTLD have their own individual procedures and regulations which potential registrants must satisfy before they are allowed to register a domain name. Some of these regulations are designed to ensure that there is a direct connection between the registrant and the country. For example, the Canadian Internet Registration Authority (CIRA), which administers the `.ca` TLD, requires that registrants satisfy the "Canadian Presence Requirements For Registrants" (see `http://www.cira.ca`). Because of this policy, if a target host has a domain name which ends in `.ca`, then we are likely correct to conclude that *the registrant* of that domain name has a presence of some form in Canada. But, it is important to note that, even though *the registrant* has a presence in Canada, the regulations imposed by CIRA do not control the geographic placement of *the registrant's computers*.

Non-country code TLDs can also encode geographic information. Of the 18 *generic* TLDs, three – `.gov`, `.edu` and `.mil` – are reserved exclusively for U.S. organizations. Other well-known codes abbreviate U.S. state names or Canadian province names and often occur as second-level domains. For example, the second-level and top-level domains of `www.city.toronto.on.ca` stand for "Ontario, Canada", and in `www.ci.ontario.ca.us` they stand for "California, USA". Thus, full city names can be encoded in domain names. Note that the meaning of some geographic codes depends on context (*Ontario* is the name of both a city in California and a Canadian province).

ISPs often use geographic codes in naming routers inside their networks – not by any policy on domain name registration, but simply as a common convention. For example, Internet traffic leaving one author's PC passes through a router named `gw03.rchrd.phub.net.cable.rogers.com`. Local knowledge of Ottawa may allow one to deduce that `rchrd` is a geographic code corresponding to "Richmond Road", suggesting that `gw03.rchrd.phub.net.cable.rogers.com` may be located some place on Richmond Road. Independent of city, we might guess that `gw03` stands for "gateway 03" and `phub` for "primary hub". Indeed, Rogers systematically uses geographic codes in the naming of routers (see `http://www.dslreports.com/faq/10758`).

As ISPs follow their own naming conventions, it may be difficult to identify the geographic codes in a router's domain name, especially for someone unfamiliar with the geographic area in question. Some ISPs use airport codes to indicate a router's geography (e.g., `.ewr1` as a label within the domain name of a router in Newark, N.J.).

Limitations:

1. (completeness) Not all target hosts have IP addresses which map to domain names; and even if one does, that domain name might not contain a geographic code.

[5] When the CAIDA NetGeo database [17] was constructed, DNS LOC records were not queried for by default since their low occurrence did not justify the work necessary to check for them.

2. (geographic precision) The target host may not always be located at or near the location indicated by a geographic code. Also for ccTLDs, some countries are physically very large.
3. (falsifiability) Domain names are selected by registrants and may include (intentionally or unintentionally) misleading geographic codes.
4. (misinterpretation) Geographic codes can be difficult to identify and interpret. Not all country code TLDs have regulations which enforce a connection between registrants and country (e.g., the `.md` ccTLD was created for the Republic of Moldova, but it is marketed to the healthcare industry worldwide.

2.6 Application- or User-Submitted Data

A simple way to find out the geographic location of a user visiting a web site is to ask them. The entry of numerous commercial web sites involves a location question ("pick your country", "enter your postal code"). Location data, once entered, allows a site to associate it with a client IP address. A single web request of a local weather forecast, or flight booking, may create an entry in an IP geolocation database. Aside from user-volunteered data, applications may also leak location information. The following HTTP headers were generated by one author's web browser:

```
Host: www.ccsl.carleton.ca
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-CA; rv:1.7.12)
                Gecko/20051010 Firefox/1.0.7 (Ubuntu package 1.0.7)
Accept: text/xml,application/xml,application/xhtml+xml,text/html;
                q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-ca,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://www.google.ca
```

From this, the visited server might deduce that `en-ca` denotes an English (en) user in Canada (ca). Requesting time of day from the browser (e.g., by JavaScript) allows a region to be narrowed down by timezone. Locality can also be extracted from an HTTP `Accept-Charset` header. The header `Accept-Charset: EUC-JP,utf-8;q=0.7,*;q=0.7`, generated by a different web browser, may indicate that the user is in Japan (but this might instead indicate the language preference of someone outside of Japan). Similar information exists in the email headers generated by some email clients (e.g., the "windows-1250" character set is associated with Central Europe). Applications running on a user's machine may be able to determine location information from the operating system. For example, users initializing or installing Windows XP may be asked: "To help services provide you with local information, such as news and weather, select your present location: [Drop down list of countries]".

Limitations:

1. (completeness) Application data is not always available. A user visiting a web site can filter unnecessary HTTP headers (e.g., using Privoxy [18]).
2. (falsifiability) Any application or user submitted data may be intentionally falsified.

2.7 Ping time measurements

The well-known command line tool `ping` can be used to send an ICMP "echo message" to a host. The round-trip time (RTT) until a reply is received can be measured; e.g., `ping -c 4 www.usenix.org` produces four RTTs. Contrary to popular belief, the geographic location of a host can [8, 15, 6] be approximated to a fine granularity from RTTs; we review two ways to do so.

The time for IP packets to travel between two hosts at fixed locations varies. Two reasons are 1) data is processed by routers at non-constant speeds (routers which relay the IP packets queue and forward data depending on their respective loads); and 2) routers may forward packets along different paths (different paths between two hosts may differ in the number of routers and/or physical path length). Despite this variability, an *absolute minimum* RTT between any two hosts exists, based on the best-case processing times of the routers and end-hosts, and the time the packet spends travelling through the physical layer of the network. This value would give a noise-free indication of the distance an IP packet travels through the network. While we would not expect conditions allowing such a measurement, this value can be approximated by capturing several RTT measurements (say, 10–15) and taking the minimum of the observed values.

The first method we review [8, 15] is based on the observation: *hosts that exhibit similar network delays to other fixed hosts tend to be co-located*. To exploit this, designate two families of hosts: (a) *probe machines*, $P_1, P_2, P_3, \ldots$, used to measure minimum RTTs to other hosts; and (b) *landmark machines*, $L_1, L_2, L_3 \ldots$, of known geographic location. For each landmark machine, build a *delay vector* consisting of the minimum RTTs from each probe to that landmark, e.g., by "pinging" each landmark a number of times from each probe machine. For example, given three probe machines and five landmark machines (see Fig. 1), five vectors are built. The vector for L_1 contains three minimum RTT measurements, one determined by each probe machine. Now, given a target host T of unknown geographic location, estimate the landmark nearest to T by measuring minimum RTTs to T from each probe machine. These measurements define a new vector, which in our example has three entries denoted (t_{T1}, t_{T2}, t_{T3}). Now determine which of the five delay vectors is closest to (t_{T1}, t_{T2}, t_{T3}) using a vector distance function (e.g., Euclidean [15], Mahalanobis [8], or "city-block" distance [21]), then infer that T is near the corresponding landmark.

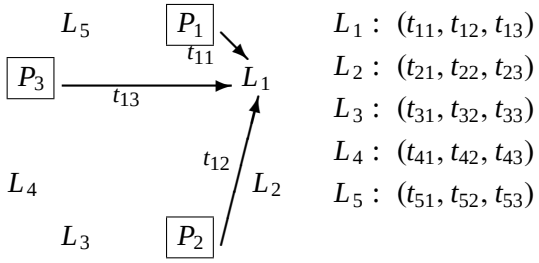

FIGURE 1: Probe machines, landmarks and delay vectors.

This method uses a *finite* number of locations – equal to the number of landmarks – to which a target host can resolve. Using more landmarks increases the granularity. This can be done, e.g., by having probes at known geographic location also serve as landmarks; or using third-party hosts as landmarks (possibly from the list of hosts having DNS LOC records; see §2.4). The only requirements on landmarks are that their geographic locations be known and that they respond to pings.

The second approach [6] maps a target host into a *continuous* space of locations. Begin by designating a family of Internet hosts that share properties of both the landmark and probe machines of the first method. The geographic location of each of these hosts must be known and they must also be able to send out "pings"; denote these hosts by LP_1, LP_2, LP_3, etc. To determine the location of a target host, T, each LP_i generates a *distance constraint* based on its observed minimum RTT to T, providing *upper bound* r_i on the length of the shortest geographic path between LP_i and T. This yields constraints $\mathsf{dist}(LP_i, T) \leq r_i$, $i = 1, 2, 3$, etc. For example, hosts LP_1, LP_2, LP_3, might generate these constraints with $r_1 = 682$, $r_2 = 2730$, and $r_3 = 170$ (km). The first asserts that T is located within 682 km of LP_1, implying as a feasible region the interior of a circle of radius 682 km centered at LP_1; the second, that T is located within a radius of 2730 km of LP_2. These two together imply as a feasible region the intersection of two circles; the feasible region for all three is the intersection F of three circles, ideally small but non-empty. A reasonable estimate for the location of T may be the center of F; its area can provide a confidence rating to the estimate. Adding more

constraints from other LP_i reduces the area of the feasible region. In experimental trials reported [6], the location of several target hosts was approximated using either 41 or 94 constraints (for targets in Western Europe and Continental U.S., resp.); the median error distances reported were below 25 km and 100 km, resp.

The second method requires the ability to convert minimum RTTs into upper bounds r_i on the distance to target; in contrast, the first used RTTs directly. It was proposed [6] that each LP_i develop its own customized conversion function, $r(t) = a_i t + b_i$ for constants a_i, b_i, by having each LP_i measure minimum RTTs to several landmarks (of known location) and thereby collect several data points consisting of minimum RTT vs. actual-distance pairs, (t, r). LP_1 could generate this data by "pinging" each of LP_2, LP_3, etc. The constants, a_i, b_i are chosen so that $r(t) = a_i t + b_i$ never under-estimates the actual distance to the landmarks. If the distance to the target is never under estimated, then F will never be empty. In practice, despite the calibration of the conversion functions, F may be empty. This might be considered a feature: if F is empty, then a location-determining routine might return a failure indication (rather than produce a poor estimate).

Limitations:

1. (completeness) Not all target hosts respond to ICMP echo messages. Indeed, hosts configured not to are increasingly common; but nearby hosts that do may be found, e.g., by `traceroute`.
2. (invasiveness) If 10-15 pings are needed to get a minimum RTT, assembling 40 distance constraints pings a target host 400-600 times – potentially viewed as an attack. Alternatively, ping requests can be staggered (e.g., over 24 hours), albeit limiting real-time applications.
3. (geographic precision) This method is a poor fit for target hosts with Internet access through high-latency connections (e.g., dial-up, satellite). The low error distances reported in the literature are promising, but tests involved well connected hosts (e.g., university campuses).
4. (falsifiability) A target host may influence round-trip times by delaying its replies.

2.8 Inference based on routing data

If it is difficult to determine the location of a target host, it may help to consider hosts "near" it. This can be done using routing information. The path that an IP packet follows can be determined using `traceroute`, which sends a target a sequence of IP packets with *time to live* (TTL) fields set as follows. The first has a TTL set to the default value of 1; this is incremented for each subsequent packet. An intermediate router along the path to the target should decrement the TTL upon receiving any packet, then either forward it (if TTL > 0), or send back an ICMP "Time Exceeded" message. The latter reveals the path the packet(s) travel.

For example, the output generated by `traceroute -I -q 1 131.106.3.253` ends with:

```
 9  core-01-so-0-1-0-0.chcg.twtelecom.net (66.192.244.21)  51.929 ms
10  core-02-ge-0-2-1-2.lsag.twtelecom.net (66.192.251.7)  87.743 ms
11  tagg-01-ge-2-3-0-506.snfr.twtelecom.net (66.192.250.121)  88.625 ms
12  206.169.168.46 (206.169.168.46)  95.380 ms
13  gw2.usenix.org (131.106.1.55)  89.181 ms
14  db.usenix.org (131.106.3.253)  94.433 ms
```

Note that `traceroute` also calculates round-trip times. Address 131.106.3.253 maps to the domain name `db.usenix.org` (see "hop" 14 above). While this domain name does not contain a geographic code, the router at hop 11 does: `snfr`, likely standing for "San Francisco". Since this is the last locatable host in the path (with respect to geolocation by geographic codes), and it is only three hops away from the target, we might infer that `db.usenix.org` is near San Francisco.

One may make other inferences from this route. Suppose our target is instead host 206.169.168.46 (listed at hop 12), which is sandwiched between `tagg-01-ge-2-3-0-506.snfr.twtelecom.net`

and `gw2.usenix.org`. Domain name registration data indicates `gw2.usenix.org` is in Berkeley, California. Thus, we might infer 206.169.168.46 is between San Francisco and Berkeley.

Another strategy, called "clustering" [15] or "blocking" [1], uses routing information to generalize the geographic location of a particular IP address to a *block* of IP addresses. The entries listed in publicly available BGP routing tables can be used for this purpose, although doing so results in many AS-level inferences which tend to be too general. BGP routes for smaller IP blocks provide better inferences; BGP *exception routes* are particularly useful.

The more specific routing information is, the better for inference purposes. For example, routing tables from intra-domain routing protocols (e.g., RIP) are of value to locators, as is the size of the subnet on which a target host resides (as revealed by the host's *subnet mask*). Methods of obtaining RIP tables and subnet masks are discussed in §2.9.

The data in a target host's DNS record may suggest other hosts which are located near it. For example, an MX record lists a host's mail server. Requesting a DNS *zone transfer* from a target host's authoritative name server can reveal many new hosts which may be geographically close; while most name servers will not allow zone transfers to foreign hosts, some do (e.g., `usenix.org`).

Limitations:

1. (inductive nature) Making an inference requires existing location data.
2. (completeness) With respect to `traceroute`, some routers are configured not to send ICMP error messages.

2.9 Network Reconnaissance

One way to determine geographic information for a large number of IP addresses is to obtain (purchase) network topology data from ISPs, i.e., a description of the geographic layout of an ISP's network and internal routing policies. This has been called "forming strategic alliances" [1]. A related method, though less scalable and available only to some organizations, is subpoena or seizure of (e.g., ISP) records.

Another avenue [1, 16] to investigate an ISP's network is to open a dial-up account with them. The phone number of the access point gives a cross-check of the location advertised by the ISP; looking the number up at `http://www.nanpa.com` provides city-level location information. Once connected to an ISP's network, any negotiated configuration information can be recorded, e.g., the assigned IP address and subnet mask. Disconnecting and then reconnecting may reveal new configuration information (e.g., a different IP address).

One specific tactic [1], for the extraction of RIP data, is to run a RIP server on the dial-up machine connected to an ISP's network, in the hope that an existing RIP server might accept the new server as legitimate and communicate routing information.

Some of an ISP's internal routers can be discovered by performing traceroutes to targets outside the network. Often, ISPs will run synchronization and management protocols on their network devices, e.g., NTP (Network Time Protocol) and SNMP. Public NTP and SNMP queries sent to these devices may extract data including: timezone, other hosts on the network, device type/manufacturer, location, net mask. Even devices which do not run NTP or SNMP may respond to ICMP timestamp and net mask queries.

Limitations:

1. (completeness) Not all ISPs provide dial-up access,[6] although most large ISPs seem to.
2. (legality) Extracting internal routing information from an ISP without consent may be prohibited by law.

[6] For example, this does not seem to be a priority for companies which provide "satellite Internet".

3 Circumventing Geolocation

We now consider an adversary who is specifically attempting to evade geolocation, and is knowledgeable in the sense of understanding both the details of geolocation technologies and which ones are being utilized by *locators* (parties trying to geolocate them). The adversary seeks to hide, or limit information disclosing, their true geographic location, and may even provide false information to misdirect a locator to a false location conclusion (locators who might otherwise continue searching for location information, may end the search given concrete, albeit false, evidence). The adversary's payoff may be, e.g., to view location-restricted content, log-on to a network, commit credit card fraud or escape the legal consequences of an action.

3.1 Limitations of Individual Geolocation Approaches

The limitations of the individual geolocation approaches listed in §2 directly suggest approach-specific evasion strategies. Rather than revisit each of these limitations, we provide Table 2 as a coarse summary and here simply mention a few examples. An adversary who decides to register a domain name for their IP address might deliberately edit any geographic codes from that domain name. An existing DNS LOC record, if any, might be removed from public view. To strip regional identifiers from HTTP requests resulting from web browsing, a filtering program could be used. Information may be provided which misdirects locators, e.g., by registering a domain name containing geographic codes for a foreign region, or publishing in a DNS LOC record the GPS coordinates of a distant city.

	geographic precision	falsifiability	freshness	completeness	public availability	misinterpretation	invasiveness/legality	inductive nature
1. whois IP look-up	●	●	●					
2. whois AS look-up	●	●	●					
3. whois domain name look-up	●	●	○	●	●			
4. DNS LOC records		●		●				
5. domain name geographic codes	●	●		●		●		
6. user/application submitted data		●		●				
7. RTTs captured via ping	●	●		●			●	
8. inference based on routing data				○				●
9. ISP network reconnaissance				○			○	
10. RRTs captured via HTTP refresh	●	●					●	

TABLE 2: **Limitations of IP Geolocation Approaches.** "○" denotes a partial limitation; e.g., for ISP network reconnaissance, one sub-approach has a legality issue, while another does not. We include approach 10 (from §4.2), but not that of §4.1, as the latter does only IP address extraction, not geolocation.

We observe from Table 2 that while each approach has limitations, combining complementary approaches can ameliorate these limitations. On the other hand, those seeking to evade geolocation can utilize techniques to avoid disclosure of local IP addresses, as detailed in §3.3 and motivated by §3.2.

3.2 Three Distinguishing Problems for Geolocation

Before further discussion of the adversarial model, we distinguish several problems to provide perspective.

Problem 1 (user geolocation). *Determine the geographic location of an Internet user, given a connection attempt or content request initiated by that user.*

Problem 2 (IP geolocation). *Determine the geographic location of the Internet device using a given IP address.*

User geolocation (Problem 1) is the main problem of interest to locators (e.g., consider a web server which processes on-line credit-card purchases). However, of the 9 techniques surveyed in §2, *all but one* (deriving location from application- or user-submitted data) deal with Problem 2 – geolocating the Internet *device* which uses a given IP address. Despite the fact that a user's device must supply a return IP address whenever requesting content from an Internet host, the distinction between these two problems is not trivial. Given a user sitting directly in front of an Internet-connected computer C which they use to make a credit-card purchase at a web site, the return IP address A submitted to the site is not necessarily that of C; indeed, it can be arranged (see §3.3) that A corresponds to a computer far away. This leads to the following observation.

Observation 1. *It is the difference between Problem 1 and Problem 2, or confusion between them, that is at the heart of the controversy (recall §1) over whether Internet geolocation "works".*

On the one hand, Internet experts claim that Internet geolocation is trivially avoidable (cf. testimony of Finkelstein and Laurie [14]). On the other hand, companies (e.g., Digital Element[7]) make claims such as being able to determine what city an IP address originates from with 94% accuracy. Aside from language ambiguity in such a statement (e.g., 94% of *all* IPv4 addresses can be located to a city, vs. 94% of U.S. end-users surveyed reported that their city was correctly determined), these claims are not necessarily contradictory if interpreted as statements about different problems. Using techniques from §2 in combination, Internet geolocation companies *do* seem to be able to answer Problem 2 reasonably well (apparently, e.g., at the country or city level, for most IP addresses). However, as we will emphasize in §3.3, a counterpoint is that this technology *cannot* answer Problem 1 for all end-users.

To effectively answer Problem 1 (user geolocation), 8 of the 9 techniques in §2 require the locator to have the IP address of the end-user's device (or that of a device nearby). This leads to the third problem.

Problem 3 (IP address extraction). *Determine the IP address of an Internet end-user's device, given a content request initiated by that user.*[8]

If we can determine the IP address of the computer sitting in front of the end-user (i.e., Problem 3), and solve Problem 2 (IP geolocation), then this solves Problem 1 (user geolocation). For the majority of end-users, browsing the web as they normally would, when they visit a web site the IP address recorded by the web server is the IP address of the computer in front of them. So, in the majority of cases, Problem 3 is solved.[9] Unfortunately, adversaries are not as accommodating as most end-users, as discussed in §3.3 and §3.4.

3.3 Employing non-local IP addresses

Here we discuss three approaches for an adversary to arrange that their Internet traffic reveals a non-local IP address (i.e., hides their actual IP address, and region of sensitivity) when received by locators.

A) Long distance dial-up. Despite slower speeds, accessing the Internet through PSTN modem dial-up remains as an available option. An important advantage of dial-up is that it permits a user to access the

[7] `http://www.digital-element.net/ip_intelligence/ip_intelligence.html`

[8] We assume that an end-user has a publicly routeable IP address. While we do not directly address complications related to private-use IP addresses as commonly used with Network Address Translation (NAT), the IP address of a NAT device may suffice to allow geolocation of the devices it serves (e.g., home networks).

[9] It is this situation where the technology of companies like Digital Element and Quova works best.

Internet from different service areas. For example, a business traveller from Toronto who has purchased dial-up access from an ISP (e.g., Bell Canada) can, while in Montreal, access the Internet through a local Montreal telephone number. Locators would identify the resulting IP address with Montreal (e.g., the ISP may have sold such information). However, users outside the Montreal region may also call the Montreal access number. Adversaries can make long distance and/or international telephone calls to access the Internet, thus appearing to originate from a region of their choice. Moreover, a number of companies specialize in providing world-wide dial-up Internet access.[10]

B) Proxies. A *proxy* is a program which acts as an intermediary between a client and a server (see [2] for a more detailed definition), usually running on a host separate from both. This host is also commonly referred to as "a proxy". Academic literature [15, 6] has stated that proxies represent a "fundamental limitation" to IP geolocation. We submit that proxies actually make little difference to Problem 2 (IP geolocation) of §3.2, but do present some difficulty for Problem 1 (user geolocation) – see §4.1 re: IP address extraction.

For example, an adversary who wants to access a locator's web page may, instead of sending their HTTP request directly to the locator's web server, send it to a proxy. The proxy will then pull down the web page on behalf of the adversary and relay it back to them. The IP address recorded by the locator will be that of the proxy, not the (end-user) adversary.

Not all proxies work in the same way. Proxies can be classified according to what level of the protocol stack they interpret and whether or not they maintain a cache. A *SOCKS proxy* [12] works at the Transport layer and it does not maintain a cache.[11] A SOCKSv5 proxy can interpret any TCP or UDP traffic (including any traffic generated by a web browser). Squid[12] is a popular open source *caching proxy* which can interpret HTTP and other web related protocols (i.e., Squid is an HTTP proxy and more). With respect to keeping a client's IP address anonymous, a SOCKS proxy is usually preferred. Proxies like Squid often relay the client's IP address to a server by adding an `X-Forwarded-For:` header to their HTTP traffic.

Anyone with `ssh` access to a remote machine (e.g., `anon.machine.example`) can, through port forwarding, use this machine as a SOCKS proxy to browse the web through. After logging in using the command `ssh -D 8888 user@anon.machine.example`, a user simply needs to configure their browser to use the SOCKS server `localhost:8888` (in Firefox, this can be set under Preferences / General / Connection Settings). Now any TCP traffic sent to port 8888 on the user's local machine is forwarded over an encrypted connection to `anon.machine.example` where it is processed by an `ssh` SOCKS proxy.

C) Remote sessions. In the two previous techniques, an adversary's browser and any other network-accessing applications are run on their local machine. If instead these applications are run on a remote machine, the IP address attached to the resulting network traffic will be that of the remote machine; thus, the IP address of the end-user device is not revealed. Windows XP users can run programs on a remote Windows XP machine using the "Remote Desktop" function. Linux users can run graphical programs on remote linux machines using the X11 forwarding function of `ssh`. Logging in using `ssh -X user@anon.machine.example` allows any graphical programs started on `anon.machine.example` to be displayed by the X11 Windows server running on the user's local machine. VNC (Virtual Network Computing) software can be used to carry out remote sessions independent of platform.

3.4 Summary Discussion of Circumventing Geolocation

Table 2 summarizes (along with §3.1) limitations of individual IP geolocation approaches, and suggests combinations of individual approaches which may overcome these limitations. Of the approaches listed, the

[10] For example, see `http://www.worldial.com/` and `http://intlaccess.web.aol.com/`

[11] SOCKS is short for SOCKetS, as in one end of a TCP connection.

[12] See `http://www.squid-cache.org/`

	A) Long distance dial-up	B) Proxy	C) Remote logins
1. whois IP look-up	●	●	●
2. whois AS look-up	●	●	●
3. whois domain name look-up	●	●	●
4. DNS LOC records	●	●	●
5. domain name geographic codes	●	●	●
6. user/application submitted data			●
7. round-trip times captured via ping	●	●	●
8. inference based on routing data	●	●	●
9. ISP network reconnaissance	●	●	●
10. RTTs captured via HTTP refresh	○	○	●

TABLE 3: **Evading Geolocation by Non-local IP Addresses.** Bullets indicate which geolocation methods (rows 1-10) may be evaded by specified techniques (columns A-C) which employ a non-local IP address.

most trustworthy information appears to be network topology information (§2.9). The fact that adversaries might falsify location information implies that locators should be wary of making inferences solely from potentially user-controlled information (including ping times). Table 3 summarizes how the three techniques of §3.3 (employing non-local IP addresses) fare against individual geolocation approaches; remote logins appear the most resistant technique.

We suggest that a conservative adversary should assume that if the traffic they send (e.g., resulting from an HTTP request) to a locator carries their usual IP address (i.e., solves Problem 3), then this will expose their location to discovery (assuming, as discussed in §3.2, that Problem 2 is solved reasonably well in practice). Thus we suggest that to evade geolocation, the IP address that an adversary allows to be submitted to a locator should be one outside their region of sensitivity. This can be done using techniques from §3.3: long-distance dial-up, remote proxies, and remote sessions.

However, these techniques themselves are not guaranteed to hide an end-user IP address – in some circumstances they may be defeated by more powerful (perhaps atypical) classes of locators. For example, a government-sanctioned organization (e.g., Interpol) may be able to subpoena records from an ISP or trace phone calls, and thereby learn the telephone number of a dial-up customer; the system logs of a remote proxy, or a machine used as a remote desktop, could be seized, revealing IP addresses of end-users.

Using a chain of proxies (i.e., connecting through one proxy to another) avoids a single point of failure allowing exposure. This idea, along with a number of other improvements, is utilized in the deployed anonymizing network *Tor* (The Onion Router [4]), which appears widely recommended for anonymous browsing. We caution that, even assuming use of Tor, there are practical issues related to browsing through a proxy, which unless carefully addressed, can expose an end-user's IP address; see §4.1 for further discussion.

Of the three IP-hiding techniques in §3.3, the cheapest and most convenient (for an adversary) would appear to be using a proxy. There are many "open proxies" on the Internet which can be accessed by anyone. However, users should beware that not all open proxies are benign. A malicious proxy might record web traffic with the intent of mining passwords and credit card numbers – thus such information should never be sent through an untrusted proxy. Long-distance dial-up has the advantage that users do not need to worry about any inadvertent network connections (e.g., DNS look-ups) exposing their local IP address.

4 Other Geolocation Techniques

Motivated by the preceding discussion of circumvention, we now present two new techniques which can be used to help geolocate Internet users.

4.1 Extracting IP Addresses through Java (Despite Proxies)

While proxies complicate the task of geolocating end-users, they do not necessarily preclude this. Here we present a new method for extracting an end-user IP address even for end-users protected by Tor/Privoxy.

Suppose a user accesses a locator's web page through an HTTP proxy. This is, for example, just how users of the AOL network access the web.[13] To geolocate this user, the locator wants to learn the IP address of the user, rather than that of the proxy. It has been suggested [16] (without details on how to create the applet) that by including a Java applet in the web page, the end-user's IP address A can be determined. This relies on the claimed ability of an applet to make a *non-proxied* connection back to the web server, exposing A. This claim has potentially serious implications for users relying on proxy servers for anonymity.

We explore the consequences for a user browsing the web via *Tor* [4, 5] with the most recent version of the *Java Runtime Environment* enabled in their browser.[14] Although (see below) previously known techniques can indeed extract end-user IP addresses from proxy-protected end-users in certain circumstances, we provide here the details for a new technique which works in situations beyond those addressed previously.

Tor is an anonymizing network designed to facilitate low-latency anonymous communication. Users install a free client program which, once activated, negotiates a secure pathway through the Tor network; a user's network traffic can be tunnelled through this pathway. The interface to the pathway on the user's machine is a SOCKS proxy run by the client which, by default, listens on local port 9050. There are two main methods of browsing the web through Tor. Method 1 is to configure a browser to use the SOCKSv5 proxy `localhost:9050`; any TCP traffic the browser generates – including web page requests – is then sent to port 9050 and tunnelled through Tor. A web server will then see page requests as originating from the exit node of the Tor pathway. However, this method is discouraged (e.g., by the EFF Tor web page). Instead, Method 2 is recommended: using the HTTP/HTTPS proxy Privoxy [18] between the user's browser and the Tor SOCKS interface, to allow identifying HTTP data to be filtered out by Privoxy. Otherwise, although *connections* to a server will not identify a user, *application data* sent over the connection might. Method 2 also prevents the user's browser from making inadvertent non-proxied DNS queries.

The new attack we present works whether Tor is utilized in Method 1 or Method 2. However, to simplify discussion, first assume a Tor user's browser is configured by Method 1 (SOCKSv5 proxy `localhost:9050`). Assume also that Java is enabled with JRE 5.0 installed. We first review a known method of causing an applet in some cases – depending on the browser and operating system – to make non-proxied network connections. Then we show how a user can defend against this technique, and finally describe a new technique which *always* causes an applet to make a non-proxied connection. The only defence (to our knowledge) against this new technique is to disable Java.

Suppose a Tor user downloads a web page containing a Java applet. The applet is permitted to open a network connection back to the server which originated it,[15] e.g., by the Java code:

```
int tcp_port = 80;
Socket S = new Socket(getCodeBase().getHost(), tcp_port);
```

This connection is administered by the JRE, which by default should inherit any proxy settings from the browser (i.e., `localhost:9050`). However, Internet Explorer (IE 6.0 with SP1 running on Windows XP) – and possibly other browsers (but not Firefox, in our tests) – seem unable to communicate these preferences to the JRE. With such a browser, a Tor user's real IP address is reported to the server by the code above. While this issue of proxy settings not being passed to the JRE is not widely known, it has been noted by some Tor enthusiasts.[16]

Rather than leave the communication of proxy settings to chance, the JRE can be explicitly informed of

[13] See `http://webmaster.info.aol.com/caching.html`.

[14] As of this writing, this is JRE 5.0 Update 6; see `http://java.com/en/download/index.jsp`.

[15] This is allowed by the Java applet security model; see `http://java.sun.com/sfaq/`

[16] For example, see `http://uk.geocities.com/osin1941/exposingtor.html`.

them using the Java Control Panel (JCP).[17] We assume now that a Tor user has entered the SOCKS proxy `localhost:9050` in the JCP network settings. This defends against the connection made by the code above. However, we now show that we can still arrange that the applet makes a non-proxied connection.

The Java 1.5 API[18] provides a new constructor for the class `Socket`, allowing programmers to specify proxy preferences for individual sockets. These override any other proxy settings, including those in the browser and JCP. We wrote an applet which executed the following code:

```
Socket S = new Socket(Proxy.NO_PROXY);
int tcp_port = 80;
InetSocketAddress A = new InetSocketAddress(getCodeBase().getHost(), tcp_port);
S.connect(A);
```

Our applet opens a non-proxied connection back to the server from which it originated. Our tests (using Windows with Firefox or IE; and Linux with Firefox) found that we could extract end-user IP addresses. This remained true for user browsers configured to go through Privoxy before Tor (i.e., using Method 2).

We end this section with a brief example of a previously known method of extracting an end-user IP address from Method 2 Tor users.[19] Suppose a Tor user is configured by Method 2, and their JRE has its default network settings (i.e., to obtain proxy settings from the browser). Even though Java applets should not be able to determine the IP address of the host on which they run [19], the following code can do so:

```
int tcp_port = 80;
Socket S = new Socket(getCodeBase().getHost(), tcp_port);
String IP = S.getLocalAddr();
```

This technique was implemented in an applet at the web site www.stayinvisible.com (see "Test Your IP"). For a web server to learn a Tor user's IP address by this technique, the value of `IP` (above) must be communicated to the server; this is easily arranged even if Privoxy is set to filter out a user's real IP address from HTTP traffic (e.g., the applet could encrypt `IP`). This attack fails if the end-user is behind a NAT device, as `IP` would be a private-use address (e.g., 192.168.0.1); however, the fact that a private-use IP address can be extracted contradicts the goals of the Java applet security model.

4.2 Timing-based Geolocation using HTTP Refresh

Here we describe a novel technique for collecting timing information related to an end-user's location. This technique works even when an end-user's machine, and machines nearby, do not respond to ICMP echo requests. In fact, this timing information can be collected without knowledge of the end-user's IP address.

Consider a user U who is browsing the web as normal (without using any proxies). Suppose U's browser loads a web page named `start.html`, containing the HTML tag:

```
<meta http-equiv="refresh" content="5; url=./stop.html">
```

This causes U's browser to wait 5 seconds, and then retrieve the page `stop.html` from the web server S. Now change the delay constant 5 to 0. This causes U's browser to retrieve the page `stop.html` immediately after loading `start.html`. If S records the respective times t_0 and t_1 when `start.html` and `stop.html` are requested, then $t_1 - t_0$ measures the round-trip time (RTT) from S to U's machine.

High resolution RTTs can be computed from page refreshes as follows. Store the following code in a file named `timer.php`:

[17] In Linux, run the file `ControlPanel` found in the JRE install directory; in Windows, the file is named `javacpl`.

[18] See `http://java.sun.com/j2se/1.5.0/docs/api/`.

[19] Thus, Tor users following the "recommended" Privoxy/Tor configuration are vulnerable to a previously known IP address extraction technique (albeit not widely known, nor in the literature); this technique fails for Method 1 Tor users.

```
<html>
<head><?php $timestamp = urlencode(microtime()); ?>
<meta http-equiv="refresh" content="0; url=./timer.php?<?php echo $timestamp ?>">
</head>
<body>
<?php echo $timestamp ?>
</body>
</html>
```

When U requests `timer.php` from S, the PHP Hypertext Preprocessor computes a time stamp (stored in the variable `$timestamp`) and evaluates the relevant statements above. An example of the resulting HTML sent to U is:

```
<html>
<head>
<meta http-equiv="refresh" content="0; url=./timer.php?0.96204300+1138426754">
</head>
<body>
0.96204300+1138426754
</body>
</html>
```

Here the time is given in fractional plus whole seconds. On reading this HTML, U's browser will immediately re-fetch `timer.php` – thus entering an infinite cycle of page refreshes (broken by clicking the browser's "stop" button or pushing the Esc key). Page refreshes can be done in a background frame or window, to avoid user inconvenience; more stealthy techniques are also possible (e.g. using Ajax). Each page refresh generates a new timestamp which is recorded in the web server's log file, e.g.:

```
GET /timer.php?0.70288200+1138426755 HTTP/1.1
GET /timer.php?0.35810600+1138426756 HTTP/1.1
GET /timer.php?0.98025000+1138426756 HTTP/1.1
GET /timer.php?0.89433400+1138426757 HTTP/1.1
```

The differences between successive timestamps represent round-trip times (RTTs).

This method can be adapted to the techniques described in §2.7. Assume 10 given probe machines each must compute a minimum RTT to the user U. The first web page U loads redirects U to a page on probe machine P_1. P_1 contains 15 individual web pages which will be loaded successively in U's browser, resulting in 14 RTT measurements. The last page on P_1 sends U to a page on P_2, which has 15 analogous pages. This continues until each probe collects its RTTs. Note that probe machines need not have synchronized clocks; each probe computes differences between its own timestamps.

We note that this timing technique also provides some (albeit limited) location information about users surfing from behind a SOCKS proxy. However, the analysis techniques of §2.7 do not seem applicable here. Any timing traffic that probe machines send to such a user will, once they pass through the proxy, travel the same route; but to fix a location, diversity is required in the respective paths of probe machines to the target.

Limitations:

1. (invasiveness) Several HTTP refreshes (say, 10-15 per probe machine) may be required to determine a location. This might be viewed as an attack or as legitimate HTTP traffic.
2. (geographic precision) It may be difficult to locate target hosts with Internet access through high-latency connections (e.g., dial-up, satellite). We have not implemented this technique but we expect its error distances to be comparable to that of the ping-based methods.
3. (falsifiability) A target host may influence round-trip times by delaying its replies.

5 Related Work

Because of the broad applications and implications of geolocation technology, a number of (primarily non-academic) sources have addressed the topic. We briefly mention the most prominent of these.

Periakaruppan and Nemeth [17] describe a graphical traceroute tool displaying the location of intermediate routers. They deduce location from DNS LOC records, geographic codes in domain names and by querying the (now outdated) CAIDA NetGeo database. They also verify location estimates by multiplying minimum ping times by the speed of light in copper. Moore et al. [13] describe building the CAIDA NetGeo database, using DNS LOC look-up, whois data and domain name geographic codes. Location estimates are also verified using ping times and the speed of light in copper. The first empirical study of IP geolocation technology was carried out by Padmanabhan and Subramanian [15]. They reported on error distances resulting from experiments conducted using three geolocation techniques: geographic codes, ping time measurements (the first method reviewed in §2.7) and interpolation based on BGP data; the latter two of these were novel. The distance constraint based technique (see §2.7) was proposed by Gueye et al. [6]. Ziviani et al. [21] studied the similarity-based ping time method of geolocation, and consider the effect the placement of landmarks and probe machines on accuracy.

Three significant patents address IP geolocation technology. The Digital Envoy patent [16] describes how to use `traceroute`, `nslookup` and `whois` to mine geographic information; heuristics for assigning a confidence level (0–100) to location estimates; and network reconnaissance techniques to be conducted during dial-up sessions into ISPs (extraction of MAC addresses is mentioned). They specifically address the problem of determining an end-user's IP address through an HTTP proxy. The NSA patent [8] describes a similarity-based ping time technique which is essentially the same as that of Padmanabhan and Subramanian [15]. The extensive Quova patent [1] also uses `traceroute`, `nslookup` and `whois`, and assigns confidence factors to location estimates; the main difference from the Digital Envoy patent is the inclusion of several heuristics ("confidence maps") for assigning confidence factors. Also mentioned is the purchase of internal routing information from ISPs.

A 2002 report [9] published by the Information Technology Association of America (ITAA) condemns the use of Internet geolocation technology for the purposes of assigning European Union value added tax (VAT) rates according to the perceived location of e-commerce customers. It states that the constant reassignment of IP addresses (to different regions) by ISPs and large global companies is a continual source of inaccuracy. They speculate this problem will worsen with the transition to IPv6, and question the methods by which Internet geolocation companies determine accuracy rates. Regarding the latter, strong marketing claims are indeed made by commercial organizations. Quova (www.quova.com) claims their "GeoPoint" IP geolocation database achieves 99.9% country-level accuracy and 94.0% U.S. state-level accuracy. Digital Element (www.digital-element.net) claims their "NetAcuity" database achieves over 99% country-level accuracy and 94% city-level accuracy. Verifia claims their "NetGeo" database is over 99.5% accurate at country-level. In all these, a rigorous definition of "accuracy" is not immediately evident.

In expert testimony, a report [20] by Cerf, Laurie and Wallon to the French court in LICRA v. Yahoo! mentions several problems with Internet geolocation, and notes that a user can connect through long-distance dial-up to use a foreign IP address. Laurie and Wallon estimated that, by using IP filtering and querying users of ambiguous IP address, Yahoo! could identify as French roughly 90% of French Internet users; Cerf disputed this estimate and gave a separate recommendation.[20] Laurie's report [14] in his testimony in Nitke v. Ashcroft considers the possibility (and fantastic expense) of modifying the infrastructure of the Internet to support IP geolocation; Finkelstein [14] also gave testimony and mentioned several ways to evade geolocation (some are mentioned in §3). Finkelstein mentions *oppositional* geolocation (cf. *uncooperative* geolocation [8]), but we know of no open study of evasionary geolocation prior to the present work.

[20] Laurie offered additional comments on this case; see `http://www.apache-ssl.org/apology.html`.

Among non-commercial projects, www.hostip.info offers a free database populated with user-submitted location information; an API is available. A number of perl regular expressions are available for extracting geographic codes from domain names.[21] JavaInetLocator is a Java-based IP geolocation database constructed using whois data; it is also available as a perl module.[22]

6 Concluding Remarks

Since specific details of environments, attacker goals, and defender goals vary on a case-by-case basis, it is hard to make conclusive statements regarding Internet geolocation; consequently, any such statements should be carefully examined. Internet experts have stated that geolocation cannot be done reliably, while some commercial organizations have claimed that they can do it with 99% accuracy. Marketing exaggeration aside, confusion arises from ambiguity in the precise meanings intended by such statements, unstated underlying assumptions, and the intended applications. A quote attributed [11] to Andy Champagne, Akamai's Director of Network Analytics, although lacking full context here, is insightful: *This service isn't meant [for] people are who trying to be evasive. It's meant for the 99 percent of the general public who are just at home surfing.* Claims made by commercial IP geolocation services regarding their accuracy typically assume no evasive action by users; this is not particularly useful in adversarial applications. Related to this, our work emphasizes the importance, when evaluating claims from IP geolocation services, to keep in mind the differences between the problems of user geolocation, IP geolocation, and IP address extraction.

Our research suggests no evidence that any single known method for Internet geolocation is *robust* – i.e., works for all IP addresses, all software and network configurations, and against adversarial end-users. Individual techniques generally have exploitable limitations, and/or provide less geographic precision than desired. However, this is countered by the wide variety of methods which can be employed in combination to reinforce confidence and precision, depending on the application. The simplest general guideline we offer is that *those relying on Internet geolocation services should not expect to succeed in all scenarios, all of the time; and those trying to evade geolocation should not expect to do so in all scenarios, all of the time.* While many obstacles can be used to complicate the task of geolocators, few guarantees exist for those wishing to evade gelocation due to the many possible ways location information may "leak out" or be extracted.

While geolocating a host is more art than science, despite its limitations, IP geolocation technology remains useful in many applications, in part due to the fact that most end-users take no evasive action. Current geolocation capabilities are well-suited (or at least adequate) in applications having the luxury of being able to rely on cooperative users, and those in which it suffices to be correct more often than not. For example, 70% accuracy (even country-level) may suffice to cut down fraud considerably. Geolocation technology also seems in many cases to be sufficient for technical compliance with legal regulations.

IP geolocation is a poor fit in cases where very high reliability and/or high geographic precision is required. Suitability should be examined on a case by case basis if evasive action may be expected. Applications requiring fine granularity remain problematic – e.g., resolving to a country level is far different than to a small number of meters, as is necessary for emergency 911 services. Use of long-distance dial-up and remote sessions appear to be powerful evasion techniques (but JavaScript may still be used to extract time zone information, and ISP server logs may be subpoenaed). We note that even if accurate IP geolocation is possible for 99% of IP addresses, if the remaining 1% is fixed and predictable by an adversary, and such that the adversary can place themselves within this subspace, then they can evade geolocation 100% of the time. We note that while would-be geolocators may purchase geolocation databases from commercial organizations for IP geolocation purposes, it seems that adversaries could also purchase or obtain such databases –

[21] See `http://www.sarangworld.com/TRACEROUTE/patterns.php3`.

[22] See `http://javainetlocator.sourceforge.net/` and `http://search.cpan.org/~nwetters/IP-Country-2.20/`

and through careful analysis, possibly identify database weaknesses (e.g., a 1% subset noted above).

It appears that some of the dangers of using Java from behind an anonymizing proxy were known previously, but not widely. The implications of allowing programmers to specify the proxy preferences of sockets in the Java 1.5 API do not appear to have been previously discussed. We believe that we are the first to point out the danger of this being used to extract IP addresses from behind anonymizing proxies, and suggest that those who promote the use of Tor should make this danger clear to users. From our results, it seems prudent to conclude that if Java is enabled, it is quite difficult to hide an end-client IP address; we believe that over time, this will remain true even if specific methods for IP address extraction are shut down.

The new timing-based geolocation proposal using HTTP refresh offers a number of advantages over ping-based techniques; one of the most important is its broad applicability. Although many users browse the web through computers which do not respond to ICMP echo requests, round-trip times to these users can be calculated using HTTP refreshes. Note that the basic technique does not rely on Java or JavaScript. Timing measurements collected through HTTP also seem to fit particularly well with dynamic-content web APIs like Ajax and Flash.

Acknowledgements. The authors thank Nigel Gourlay for bringing [6] to their attention. Also, helpful comments on an earlier version of this work were contributed by John Aycock, and by Anil Somayaji and other members of Carleton University's Digital Security Group. This report was funded in part by the Communications Security Establishment and by MITACS. The research of JAM was also supported by a Natural Sciences and Engineering Research Council of Canada (NSERC) Postdoctoral Fellowship. The research of PVO is also supported by NSERC under a Discovery Grant and as Canada Research Chair in Network and Software Security.

References

[1] M. ANDERSON, A. BANSAL, B. DOCTOR, G. HADJIYIANNIS, C. HERRINGSHAW, E. KARPLUS AND D. MUNIZ. Method and apparatus for estimating a geographic location of a networked entity. *United States Patent 6,684,250*. Assigned to Quova, Inc. Filed 3 April 2001. Issued 27 January 2004.

[2] I. COOPER, I. MELVE AND G. TOMLINSON. Internet web replication and caching taxonomy. *RFC 3040*, January 2001.

[3] C. DAVIS, P. VIXIE, T. GOODWIN, I. DICKINSON. A means for expressing location information in the domain name system. *RFC 1876*, January 1996.

[4] R. DINGLEDINE, N. MATHEWSON, P. SYVERSON. Tor: the second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004, pp. 303–320.

[5] Tor: Overview, `http://tor.eff.org/overview.html.en`

[6] B. GUEYE, A. ZIVIANI, M. CROVELLA, S. FDIDA. Constraint-based geolocation of Internet hosts. In *Proceedings of the Internet Measurement Conference 2004*, October 2004, pp. 288–293.
(extended version: `http://rp.lip6.fr/~gueye/`)

[7] OSIN. Exposing Tor Users' Real IPs,
`http://uk.geocities.com/osin1941/exposingtor.html`

[8] S. HUFFMAN AND M. REIFER. Method for geolocating logical network addresses. *United States Patent 6,947,978*. Assigned to the United States of America as represented by the Director, National Security Agency. Filed 29 December 2000. Issued 20 September 2005.

[9] INFORMATION TECHNOLOGY ASSOCIATION OF AMERICA. *Ecommerce taxation and the limitations of geolocation tools*, November 2002.

[10] Internet Engineering Taskforce Geographic Location/Privacy Working Group,
`http://www.ietf.org/html.charters/geopriv-charter.html`

[11] A. JESDANUN. "World Wide Web Narrowing?", Associated Press. CBS News, New York, 12 July 2004.

[12] M. LEECH, M. GANIS, Y. LEE, R. KURIS, D. KOBLAS AND L. JONES. SOCKS protocol version 5. *RFC 1928*, March 1996.

[13] D. MOORE, R. PERIAKARUPPAN, J. DONOHOE, AND K. CLAFFY. Where in the world is netgeo.caida.org? *International Networking Conference (INET) '00*, July 2000. (poster).

[14] BARBARA NITKE AND THE NATIONAL COALITION FOR SEXUAL FREEDOM v. JOHN ASHCROFT, Attorney General (U.S.A), U.S. District Court, Southern District of New York, case no. 01 Civ. 11476 (RMB), 2003-2004. (Finkelstein testimony: `http://www.sethf.com/nitke/ashcroft.php`)
(Laurie testimony: `http://www.apache-ssl.org/nitke.pdf`)

[15] V. PADMANABHAN AND L. SUBRAMANIAN. An investigation of geographic mapping techniques for Internet hosts. In *Proceedings of SIGCOMM 2001*, August 2001, pp. 173-185. (an extended version of this paper is contained in Subramanian's Master's Thesis)

[16] S. PAREKH, R. FRIEDMAN, N. TIBREWALA AND B. LUTCH. Systems and methods for determining collecting and using geographic locations of Internet users. *United States Patent 6,757,740*. Assigned to Digital Envoy, Inc. Filed 31 March 2000. Issued 29 June 2004.

[17] R. PERIAKARUPPAN AND E. NEMETH. Gtrace – a graphical traceroute tool. In *Proceedings of LISA '99: 13th Systems Administration Conference*, November 1999, pp. 69–78.

[18] Privoxy - Home Page, `http://www.privoxy.org`

[19] Chronology of security-related bugs and issues, `http://java.sun.com/sfaq/chronology.html`. (see "Privacy hole, related to IP addresses, fixed in May 1996 JDK 1.0.2 (March 17, 1997)")

[20] YAHOO! INC. V. LA LIGUE CONTRE LE RACISME ET L'ANTISEMITISME AND L'UNION DES ETUDIENTS JUIFS DE FRANCE, Tribunal de Grande Instance de Paris (County Court of Paris), (commenced April 2000; Interim Court Order, 20 November 2000).
`http://www.eff.org/legal/Jurisdiction_and_sovereignty/LICRA_v_Yahoo/`

[21] A. ZIVIANI, S. FDIDA, J. DE REZENDE, O. DUARTE Improving the accuracy of measurement-based geographic location of Internet hosts. *Computer Networks and ISDN Systems* **47** (2005), 503–523.