MOR: Monitoring and Measurements through the Onion Router

Demetris Antoniades[1], Evangelos P. Markatos[1], and Constantine Dovrolis[2]

[1] Institute of Computer Science
Foundation for Research & Technology Hellas
{danton,markatos}@ics.forth.gr
[2] College of Computing, Georgia Institute of Technology
{dovrolis}@cc.gatech.edu

Abstract. A free and easy to use distributed monitoring and measurement platform would be valuable in several applications: monitoring network or server infrastructures, performing research experiments using many ISPs and test nodes, or checking for network neutrality violations performed by service providers. In this paper we present MOR, a technique for performing distributed measurement and monitoring tasks using the geographically diverse infrastructure of the Tor anonymizing network. Through several case studies, we show the applicability and value of MOR in revealing the structure and function of large hosting infrastructures and detecting network neutrality violations. Our experiments show that about 7.5% of the tested organizations block at least one popular application port and about 5.5% of them modify HTTP headers.

1 Introduction

A common request of researchers, administrators and simple users, is easy access to a number of geographically distributed machines. Such access would facilitate experimentation and better understanding of several network configurations, or checking for network neutrality violations. In this extent a freely available, distributed monitoring and measurement platform would be of great value.

For many years now researchers have been using the Planetlab [10] infrastructure for conducting distributed experiments. Planetlab offers access to machines located in many different educational institutions around the world. Through a Unix-like system it allows its users to run experimental code on these machines. In this way, researchers are able to run distributed programs in many different locations, and check the network communication of their applications in real network environments. Being a (mostly) educational infrastructure, Planetlab omits commercial networks and Internet Service Providers (ISPs) with very different policies, configurations and infrastructures. Furthermore, access to Planetlab is limited to researchers. Administrators wanting to check recent configuration changes, and end users aiming at checking the quality of the service they pay for, lack a geographically distributed system freely available for this kind of daily experiments.

In this paper we present MOR, a technique for performing geographically distributed monitoring and measurement experiments. MOR utilizes the infrastructure of

the Tor anonymity network [12]. Tor is a free software aiming to protect the privacy of its users, by directing users' traffic through a distributed infrastructure. This infrastructure is built by voluntarily deployed nodes in organizations, institutions and homes. To support our idea we provide a proof-of-concept implementation of such a monitoring and measurement technique. Using our technique we examine a number of case studies that show the applicability and value of MOR. The provided case studies range from examining the structure and function of large hosting infrastructures and detecting network neutrality violations. Our main contributions can be summarized as follows:

- *We propose a technique for performing large-scale distributed measurements using the Tor anonymity network.*
- We *demonstrate the feasibility of our technique* by providing a *proof-of-concept implementation*, and provide several different *use cases*.
- We explore the extent of *port blocking* by various organizations over the Globe. Our results show that 11 out of 149 tested organizations (7.5%) block outgoing connections on ports of widely used services such as ftp(21), ssh(22) and telnet(23).
- We explore the extent to which organizations *alter HTTP headers*, and find that 9 out of 166 tested organizations (5.5%) alter, suppress or add HTTP headers.
- We explore the extent of *Skype blocking* by organizations. Interestingly enough, we find one ISP which consistently blocks Skype.

The rest of the paper is organized as follows: Section 2 gives a small introduction to Tor. Section 3 shows how we can use Tor in order to perform distributed network monitoring. We provide a number of case studies for our proposed technique in Section 4. Finally, we place our work in the appropriate context by presenting the related work in Section 5 and we conclude the paper with a discussion on the advantages and disadvantages of our approach in Section 6.

2 The Tor Network

Tor [12] is currently the most widely deployed anonymous communication system, with an estimation of more than 100,000 daily users around the globe [16]. These users range from ISP clients, military and company employees, to journalists and law enforcement officers [15]. Used mainly for web traffic, Tor is carefully designed in order to provide anonymity for low latency services.

Tor is based on the idea of Onion Routing [14], with the approach having its roots in the idea of Mix Networks proposed by Chaum [9] in early 1980s. Onion Routing is built on the concept that a message from a source to a destination will first travel via a sequence of arbitrary selected proxies (*Onion Routers*). In Tor this sequence (*circuit*) is selected at random when a connection request is received (*stream*). The last node in the circuit, called an *Exit node*, is the one that will perform the actual communication with the service of interest on behalf of the user. Before the source node transfers any message to the system, it will first encrypt it with the public key of all the intermediate proxies, creating a succession of layers like an *"onion"*. Any intermediate node will then decrypt the message, with its private key, and pass it on to the next node of the circuit. When the Exit node decrypts the message, it will have the actual request data

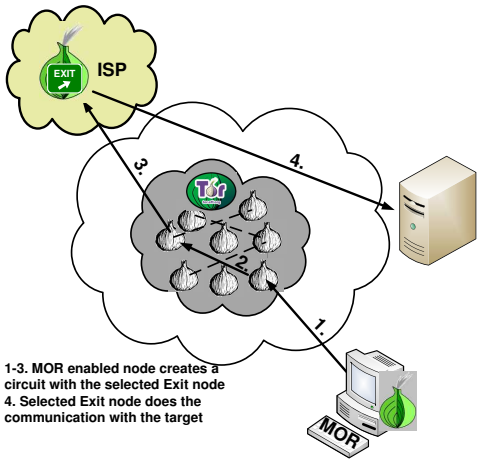

Fig. 1. Basic steps for routing experimental traffic through the Tor network.

and will be able to proceed by communicating with the requested service. The response will follow the same procedure (onion creation and "un-peeling") and reverse path back to the client.

In order for Tor to succeed in providing acceptable and efficient anonymity for its users, it needs to create an overlay with a large number of proxies. In this way, latency is minimized due to the even distribution of the load to the proxies, and anonymity is improved due to the wide combination of nodes for building circuits [11]. From the information provided by the directory servers of Tor about 1600 nodes were connected to the overlay in mid September'09. Almost half of the nodes (660) were accepting to forward traffic to the outer Internet, functioning as *Exit nodes*. These nodes were located in 48 different countries all over the word and registered by 312 different Autonomous Systems (AS). Such a large and geographically diverse overlay implicitly offers free access to the Internet through a large number of different operational networks in an easy and publicly available form. The work presented in this paper gives a first, to our knowledge, approach of using Tor for performing monitoring and measurements tasks.

3 Using Tor as a monitoring and measurement platform

Participation in the Tor network is freely available and minimal effort is needed to install and configure a proxy. Though, to be able to perform experiments using Tor's infrastructure one needs to properly instrument circuit creation and connection attaching. Fortunately, the Tor community provides extended documentation for the proxy control protocol and a python library for instrumenting the proxy [22].

For any given application we want to run through Tor a basic sequence of steps has to be followed. First we select the "experimental-node-set", a proper set of Exit nodes fitting the requirements of the application. Consider a Web-based application, the experimental-node-set will exclude all Exit nodes blocking outgoing traffic to port

80 in their configuration policy. After we have this experimental-node-set we follow the sequence of steps shown in Figure 1, iterating over each different Exit node in the set. First we create a circuit with the selected Exit node (steps 1-3). Since Tor does not allow for single-node circuits, this circuit includes at least one additional proxy. This intermediate proxy can be arbitrary selected, though we prefer to select a stable (based on its given status) router, to minimize any interference to our experiment. After the circuit is properly created, we proceed with creating the data socket for handling the required communication stream. On the first packet, the stream is attached to the previously created circuit. After these two steps are successfully completed, all traffic of the data stream is routed through the selected Exit node and our monitoring application can proceed as it would in the absence of Tor. Thus, it will send any data and wait for the response. The target host can be any online host in the Internet able to respond to our request, or a controlled host in a laboratory that would be instrumented to respond to our requests and/or log any incoming traffic from the Tor network for post analysis.

4 Case Studies

In this section we present a number of applications, that show the applicability and value of using Tor as an experimental platform. In our case studies we use MOR to reveal the structure and function of large hosting infrastructures and to detect network neutrality violations.

4.1 Examining content replication in a One-Click Hosting Service

Our first case of interest comes from the need to understand how files are replicated and served by *www.rapidshare.com*: one of the largest One-Click Hosting Services (OCH). OCH services enable users to upload and download very large (100-200 MBytes) files for a very low cost. Their low cost, dependable service and very high capacity, make OCH services very popular within file sharing communities. Thus, recently, OCH services have been used as an alternative to peer-to-peer file-sharing systems. From a technical point of view, OCH services can be considered as Content Distribution Networks (CDNs). However, OCH services host mostly large files, while CDNs, in addition to large files, host lots of small files, such as objects in web pages.

Since the files shared by users of *rapidshare.com* are several MBytes large, our interest is to understand how the service replicates and serves each file to its users. To explore this, we access the same file from many different locations (Tor Exit nodes) in order to derive the actual server(s) that provide the file each time. In our experiments, described in detail in [2], we build a list with more than 20,000 *Rapidshare* URLs, available on the Web, and repeatedly requesting them for download, by a group of 421 different *Exit nodes*. In this way, if server selection is done based on the client's IP address the address of the Exit node would be used by the decision algorithm.

Our experiments show that, although we observe more than 5,000 Rapidshare server IP addresses in total, *each* file is hosted only by *exactly* 12 *Rapidshare* servers. Our download attempt is always redirected to a single indexing server providing us with download URLs that point to 12 different servers hosting the actual file. Furthermore,

Description	Port	Blocked (%)	Description	Port	Blocked (%)	Description	Port	Blocked (%)
Sun-RPC	111	13.16	MS-SQL	1434	7.97	Telnet	23	7.38
IMAP	143	6.70	MySQL	3306	6.52	Unreal-Game	7777	6.52
Netmeet	1503	6.47	Shiva-VPN	2233	6.47	SNMP	161	6.43
FW1-VPN	259	6.43	Netmeet	1720	6.43	Bay-VPN	500	6.38
SSH	22	6.36	Skype	5060	6.34	FTP	21	6.33
DNS-Xfer	53	6.25	IMAPS	993	5.98	HTTP	80	5.83
HTTPS	443	5.73	POP3	110	5.43			

Table 1. Checked port numbers.

we observed no ISP specific policy decisions, since all download requests for a specific URL where (almost) equally distributed among the 12 download servers.

4.2 Network Neutrality

An important discussion regarding human rights on network access is the one related to network neutrality. Internet users expect neutral treatment of their traffic from their provider, regardless the application protocol, port number or content they aim to access. In that extent, we use MOR to present a number of use cases able to infer whether a user is receiving neutral treatment from her network provider. Note, though, that use of the described experiments is not limited to end users, but an administrator can also exploit the same setup to test network or firewall configurations.

Port Blocking Our first case study, regarding network neutrality, explores whether an organization blocks outgoing traffic from specific port numbers to the global Internet. We use a MOR client issuing access requests (TCP-SYN) to a controlled machine, located in our organization, for a number of different TCP ports. The controlled machine logs all incoming traffic using *tcpdump*, and is set out of the firewall of our organization, thus able to receive and respond to any incoming request.

In our experiments we use the port numbers defined as *"Ports of Interest"* in [7]. These ports span a large number of applications (web, p2p, e-mail, games, chat etc.). The full list of ports we use, and the description of each port, are depicted in Table 1. We probe each port 10 times from 236 different Tor *Exit nodes*. We consider a port to be blocked by an organization only if no TCP-SYN was received in any of the 10 tries.

Column *"Blocked"* of Table 1 shows the percentage of nodes that seem to be blocking each tested port number. Note that we currently have no indication whether the blocking is done by the Exit node hosting organization or somewhere in the path between that organization and our controlled machine. This kind of identification is left for future work. In all cases at least 5% of the nodes employ port blocking. From the results, we can see the largest percentage of blocking (more than 7%) to be for ports that are prone to Internet attacks, like MS-SQL and MySQL default ports (1434 and 3306 respectively) and Sun-RPC (111). Furthermore, it is interesting to see that a number of the tested organizations (more than 6%) block access to widely used services such as ftp (21), ssh (22), and telnet (23). We speculate that port blocking is done both for security

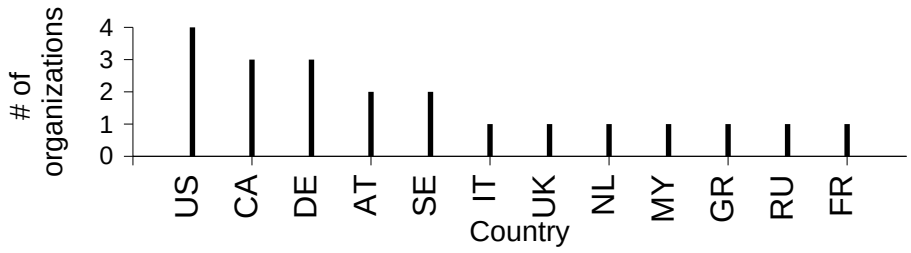

Fig. 2. Number of port blocking organizations per country

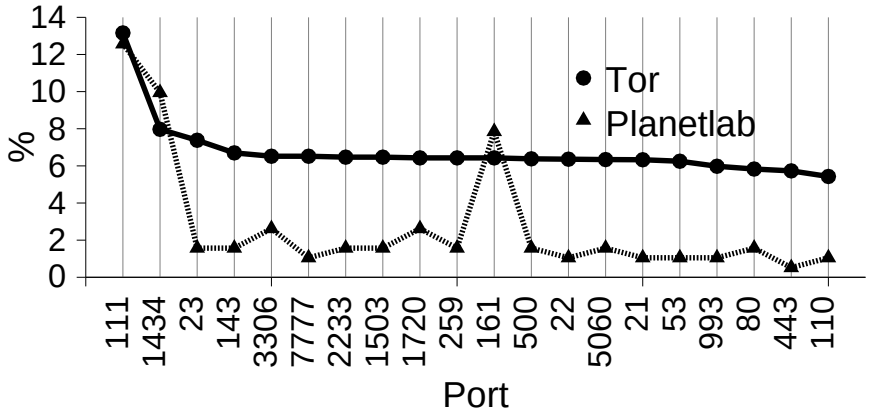

Fig. 3. Port blocking comparison between Tor and Planetlab

considerations (ssh, telnet) and traffic discrimination (skype). Figure 2 plots the number of organizations (ASNs) exploiting port blocking per country.

As a further step, we examined the same port numbers using the Planetlab infrastructure [10, 21]. We use 191 different Planetlab nodes, again sending 10 TCP-SYN connection requests for each port. Figure 3 shows the comparison between the two infrastructures. We can observe a similar percentage of blocking only in three port numbers (111, 1434 and 161), which are considered prone to Internet attacks. In all other cases we observe a larger percentage of blocking in the case of the Tor overlay. Comparing the autonomous system numbers (ASN) hosting Planelab and Tor nodes we found only 10 common ASNs. Most of the organizations hosting the planetlab node were universities and research institutions. On the other hand, the percentage of academic organization in the Tor ASNs was less than 2%. Thus, MOR, by utilizing the Tor infrastructure, provides access to nodes located in commercial providers.

HTTP Header Suppression In the next use case, we use MOR to study the suppression of *HTTP Headers* performed by different organizations. As shown in [5] most of the

```
'Accept-language': 'en-us'
'Accept-encoding':
  'gzip, deflate, compress;q=0.9'
'Host': '139.91.70.22'
'Accept': 'text/html,
  application/xhtml+xml,
  appplication/xml;q=0.9,*/*;q=0.8'
'User-agent': 'Mozilla/5.0 (X11; U;
  Linux i686; en-US; rv:1.9.0.3)
  Gecko/2008092510 Ubuntu/8.04
  (hardy) Firefox/3.0.3'
'Accept-charset':
  'iso-8859-1,utf-8;q=0.7,*;q=0.7'
'Connection': 'Close'
'Referer': '139.91.70.81'
'Cache-control': 'max-age=0'
```

Fig. 4. Actual Request HTTP Header

```
'Content-Length': '175'
'Accept-Ranges': 'bytes'
'Server': 'Apache/2.2.3 (Debian)
  DAV/2 SVN/1.4.2
  mod˙python/3.2.10 Python/2.4.4
  PHP/5.2.0-8+etch13
  mod˙ssl/2.2.3 OpenSSL/0.9.8c
  mod˙perl/2.0.2 Perl/v5.8.8'
'Last-Modified':
  'Sat, 11 Apr 2009 10:05:05 GMT'
'Connection': 'close'
'etag': '"44540-9e-9d84b640"'
'Date':
  'Sat, 18 Apr 2009 12:34:24 GMT'
'Content-Type': 'text/html;
  charset=UTF-8'
```

Fig. 5. Actual Response HTTP Header

time *HTTP Header* suppression comes from the network and not the browser, since some organizations may use intermediate proxies, and add or remove some headers, for caching, security and privacy reasons.

In our setup, we use a monitored machine, running an Apache HTTP server on port 80. Our MOR client issues HTTP requests for a simple web page hosted in our server. The initial *HTTP Header* contained in the request is shown in Figure 4. The HTTP server always responds with the header shown in Figure 5. [3] We used tcpdump to capture both the traffic sent and received from our client to the Tor proxy and also the traffic to and from the Web server.

Our experimental-node-set contains 166 different exit nodes. In the largest percentage of Exit Nodes we observed no difference in both request and response headers. In 5.5% of the cases, though, we had suppressed headers, addition of extra header fields and in some cases responses without even accessing the Web server, probably due to caching of a previous identical request from another Exit Node in the same network.

In most cases the altered, added or suppressed field reveals the existence of a proxy, either used as a centralized HTTP access point for an organization or for content caching purposes. In such cases we observe altering of the *"Connection"* header field in the request header, and addition of the *"Via"* header field in the response header. Furthermore, we observe cases where the organization completely removes the *"Referer"* or *"User-agent"* header, probably due to privacy policies. The organizations for which we observe alteration in the HTTP header fields are located in several countries, namely France, Germany, Argentina, USA, Canada and China.

As a further step we run our Web server on an arbitrary port number (20000). This experiment investigates whether an organization uses Deep Packet Inspection (DPI) to recognize any HTTP traffic, or whether the identification is based only on well known port numbers. In our results no altering or suppression was observed when accessing the server in a different port. Thus we can say that all used organizations do not employ DPI for HTTP traffic altering. As future work we plan to extend this study to identify traffic discrimination for file-sharing applications, like BitTorrent, through DPI.

[3] Note that in case of a difference in the HTTP Request the server will also respond differently.

Skype censorship Another issue of interest, regarding network neutrality violations, arises when an organization is restricting access or limiting the performance of an Internet application, based on the port numbers used, employing DPI techniques or specific host blocking. As an example, in this case study, we explore the extent to which organizations block access to the Skype IP telephony application.

Skype utilizes a p2p infrastructure to connect its clients. When the Skype application starts it first communicates with a centralized login server (`ui.skype.com`) which will verify the user's credentials and allow her to log in to the p2p network. After the log in phase, the user's Skype traffic, either chat, voice or video, is transferred through the application's p2p overlay [6].

The login phase is done over the HTTP protocol by requesting a URL from the server. The URL contains the hashed user credentials and information about the running version of the application. We use this login method in order to identify organizations that block Skype users from logging into the system. For our experiments we extracted the URL from the latest version of a Linux Skype client. [4] We request the URL from Skype's login server through Tor, using 171 different Exit nodes. For every Exit node we request the URL 10 times and log the response result. We consider the organization not to be blocking access to Skype if we receive at least one valid response from the server. Our connection requests were 100% unsuccessful in only one case. This Exit node is hosted by a Kuwait ISP which has also been reported by others to block Skype. [5]

4.3 Further Possible Use Cases

Web-Page Censorship: Recent work has shown that a number of web clients receive altered pages during their browsing sessions [19]. These alterations may include advertisements, extra javascript code and even malware, that are either annoying (in the best case) or harmful to the user. Using our technique one can compare the page she receives from a Web server, with the pages received when the server is visited from a different geographical location and/or ISP.

Network Problems Diagnosis: Using MOR one can easily detect if an administered or desired service is working properly. For example a user can check if a service is non responsive also from other organizations or only from it's own network in order to report this to her administrators. Also online service administrators can check the visibility and correct functioning of their service when viewed from external networks.

DNS update speed: Using a SOCKS4a proxy one can direct DNS queries through the Tor network. Combined with our technique one can measure the time needed for a domain name update to become visible by the rest of the Internet.

5 Related Work

With Tor being increasingly popular during the last years, a number of researchers examined the network, targeting its performance [20], attacking the system [8, 18]. or trying to compromise its users' anonymity through traffic analysis [13, 17].

[4] `http://ui.skype.com/ui/2/2.0.0.72/en/getlatestversion?ver=2.0.0.72&uhash=1074a31ab9146cc11ab149c86a32dc920`

[5] http://www.248am.com/mark/kuwait/skype-blocked-by-qualitynet/

The goal of performing a variety of distributed experiments, led a number of researchers to use overlay networks from a different aspect than the one initially intended. Athanasopoulos *et al.* in [3] used the Gnutella overlay network to perform Distributed Denial of Service attacks on third party services. In a subsequent work the same authors illustrated the use of Gnutella in anonymously downloading a Web file [4]. Close to our work, Beverly *et al.* in [7] used the Gnutella Network to quantify the prevalence of port blocking from ISPs and institutions. In their setup, a super-peer in the Gnutella network was instrumented to redirect each contacting client to a specific port of a measurement host controlled by the authors. Recently, Barth *et al.* in [5] used advertisement networks to study the use and suppress of the *HTTP Referer* field. They used two advertisement networks to display custom advertisements to the users for 3 days. When the advertisement was displayed in the user's Web browser, it issued a number of HTTP requests to two servers controlled by the authors. Their observation showed that most of the times, the *Referer* HTTP field was suppressed in the network and not in the browser.

6 Discussion, Limitations and Conclusions

In this paper we propose a new technique for performing measurement and monitoring tasks. We propose the use of the Tor anonymizing network as a geographically distributed infrastructure. Using our proof-of-concept implementation, MOR, we present a number of case studies that demonstrate the applicability and value of our approach. Our experiments show that about 7.5% of the tested organizations block at least one popular application port and about 5.5% of them modify HTTP headers.

While our work actually leverages the Tor network, the applications we propose make careful use of the network adding limited overhead (i.e. single TCP-SYN packets and small Web requests). We expect MOR to act as a motivation for a number of users, interested in measurement and monitoring tasks, in adding more relays to the network.In this case, Tor will benefit from MOR users, since more proxies will increase the network's geographic diversity, improve anonymity and Tor's overall performance [1].

Limitations: Unfortunately, with the current state of the Tor network, a number of interesting tasks can not be implemented. Two main limitations are the lack of relaying non-TCP traffic and the limited throughput performance. Tor does not, for the time being, support relaying non-TCP traffic. In this extent a number of programs (i.e. traceroute) that use other IP protocols, can not be relayed through Tor. Due to this, a number of experiments based on this type of tools are not feasible. Furthermore, though Tor tries, and succeeds, to provide low latency anonymization, it is still not able to support high throughput applications. In this case experiments targeting on identifying path delay, loss and average/peak throughput are not guaranteed to provide accurate results. It is highly possible that the measured metric will be affected by the system itself and will not correspond to the actual value from the targeted network. Though these are fundamental limitations on the number of possible use cases of our technique, we believe that our work will encourage exploration for integrating the aforementioned metrics.

References

1. A. Acquisti, R. Dingledine, and P. Syverson. On the Economics of Anonymity. In *Proceedings of Financial Cryptography (FC '03)*, January 2003.
2. D. Antoniades et al. One-Click Hosting Services: A File-Sharing Hideout. In *Proceedings of the ACM/SIGCOMM conference on Internet Measurements*, November 2009.
3. E. Athanasopoulos, K. Anagnostakis, and E. Markatos. Misusing Unstructured P2P Systems to Perform DoS Attacks: The Network That Never Forgets. In *Proceedings of the 4th International Conference on Applied Cryptography and Network Security*, June 2006.
4. E. Athanasopoulos et al. GAS: Overloading a File Sharing Network as an Anonymizing System. In *Proceedings of the 2nd International Workshop on Security*, 2007.
5. A. Barth, C. Jackson, and J. Mitchell. Robust Defenses for Cross-Site Request Forgery. In *Proceedings of the 15th ACM conference on Computer and Communications security*, 2008.
6. S. Baset and H. Schulzrinne. An Analysis of the Skype Peer-To-Peer Internet Telephony Protocol. In *IEEE International Conference on Computer Communications*, 2006.
7. R. Beverly et al. The Internet's Not a Big Truck: Toward Quantifying Network Neutrality. In *Proceedings of the 8th Passive and Active Measurement Workshop*, Apr. 2007.
8. N. Borisov, G. Danezis, P. Mittal, and P. Tabriz. Denial of Service or Denial of Security? How Attacks on Reliability can Compromise Anonymity. In *Proceedings of the 14th ACM Conference on Computer and Communication Security*, October 2007.
9. D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2), 1981.
10. B. Chun et al. Planetlab: an overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.*, 33(3):3–12, 2003.
11. R. Dingledine and N. Mathewson. Anonymity Loves Company: Usability and the Network Effect. In *Proceedings of the Fifth Workshop on the Economics of Information Security (WEIS 2006)*, June 2006.
12. R. Dingledine, N. Mathewson, and P. Syverson. Tor: the Second-Generation Onion Router. In *Proceedings of the 13th conference on USENIX Security Symposium*, 2004.
13. N. Evans, R. Dingledine, and C. Grothoff. A Practical Congestion Attack on Tor Using Long Paths. In *Proceedings of the 18th USENIX Security Symposium*, August 2009.
14. D. M. Goldschlag, M. G. Reed, and P. F. Syverson. Hiding Routing Information. In *Proceedings of Information Hiding: First International Workshop*, May 1996.
15. Karsten Loesing. Measuring The Tor Network: Evaluation Of Client Requests to the Directories, 2009. `https://git.torproject.org/checkout/metrics/master/report/dirreq/directory-requests-2009-06-26.pdf`.
16. D. McCoy et al. Shining Light in Dark Places: Understanding the Tor Network. In *Proceedings of the 8th International Symposium on Privacy Enhancing Technologies*, July 2008.
17. S. J. Murdoch and G. Danezis. Low-Cost Traffic Analysis of Tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, May 2005.
18. V. Pappas et al. Compromising Anonymity Using Packet Spinning. In *Proceedings of the 11th Information Security Conference*, September 2008.
19. C. Reis et al. Detecting In-flight Page Changes with Web Tripwires. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, 2008.
20. R. Snader et al. A Tune-up for Tor: Improving Security and Performance in the Tor Network. In *Proceedings of the Network and Distributed Security Symposium*, February 2008.
21. N. Spring, L. Peterson, A. Bavier, and V. Pai. Using PlanetLab for Network Research: Myths, Realities, and Best Practices. *SIGOPS Oper. Syst. Rev.*, 40(1):17–24, 2006.
22. The Tor Project. TorCtl. `https://svn.torproject.org/cgi-bin/viewvc.cgi/torctl/`.