UNIVERSITY OF CALIFORNIA

SANTA CRUZ

In Pursuit of Privacy on a Public Internet

A thesis submitted in partial satisfaction
of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

Marc Mendonca

March 2012

The Thesis of Marc Mendonca
is approved:

Table of Contents

List of Figures

List of Tables

Abstract

In Pursuit of Privacy on a Public Internet

by

Marc Mendonca

User privacy on the Internet has been an increasing concern in recent years. With the proliferation and sophistication of information services, data mining, and search engines, a simple network address may be used to reveal a great deal of information about a user, including location, identity, and behavior.

A new network architecture paradigm known as "Software-Defined Networking" (SDN) has recently garnered attention in both industry and academia. Defined by a separation of data and control planes, it offloads routing decisions from the switching hardware and provides an innovative framework upon which new protocols and services can be deployed.

In this thesis, we examine how SDN can be employed by service providers to offer endpoint privacy. We identify existing approaches to privacy and determine that they make unacceptable tradeoffs between performance and anonymity. We argue an acceptable level of privacy can be provided to most users, with noticeably lower latency and throughput impact, by working with the network provider; to that end, we introduce *AnonyFlow*, an in-network anonymization service designed to efficiently and seamlessly provide privacy to users as they communicate with other endpoints and services. We design, implement, and evaluate a prototype of *AnonyFlow*, based on an OpenFlow SDN deployment, that achieves endpoint anonymity without compromising on throughput or latency.

Acknowledgments

This thesis would not have been written without the advice and support offered by a number of people, and I would like to express my gratitude and acknowledge their contributions here.

I thank my supervisors, Katia Obraczka (UCSC), Thierry Turletti (INRIA), and Srini Seetharaman (Deutsche Telekom), for their guidance and invaluable feedback on my work.

I am grateful to the numerous colleagues that I have worked with over the past couple of years at Deutsche Telekom R&D Lab USA, the Planète team at INRIA Sophia Antipolis, and the Inter-Networking Research Group at UCSC. In addition to helping me refine my ideas, they have provided an unforgettable environment in which I have grown as a researcher.

I would also like to acknowledge the School of Engineering staff for their friendly assistance. Many thanks to Carol Mullane for helping me navigate the administrivia related to graduate school.

Finally, I would like to thank my parents for their years of endless love and support.

1 Introduction

As data mining, geolocation services, targeted advertising, and data brokers become more pervasive, it is possible to learn a large amount from information flowing through the network, in particular network addresses stamped on each packet. This coupled with considerable increase in privacy and security breaches have sparked renewed interest in services and tools that provide user anonymity. However, as will become clear in Section 3, current anonymization approaches typically incur prohibitive performance penalties.

It can be argued that the bigger threat to privacy for everyday Internet users is unscrupulous or overzealous endpoints and Web services, and not network infrastructure providers, who are typically restricted from disclosing data. In fact, our claim is that infrastructure providers would be quite willing to protect their users against any potential attacks and/or threats as they may be held accountable and liable for security breaches, such as identity theft and other intrusions of user privacy. While there have been some previous attempts to enlist providers to offer privacy services[39], they required the deployment of specialized gateways and faced challenges arising from having multiple ingress/egress points in the network. While the previous work was flow-based, it used a cryptographic approach that relied on key rotations to prevent certain attacks. This caused outages on long-lived flows as gateways were unable to keep track of the per-flow state when keys rotate on a fixed-time period.

Recently, a new paradigm of network architecture known as "Software-Defined Networking" (SDN) has emerged. Defined by a separation of data and control planes, it offloads routing decisions from the switching hardware and provides an innovative framework upon which new protocols and services can be deployed. Approaching issues raised by the previous attempt listed above, such as handling multiple ingress/egress points, from a SDN viewpoint vastly simplifies the problem. The deployment is expedited as the service can be defined completely in software without the addition of specialized gateways. Furthermore, when using a controller to handle new flows, the issue of replicating flow-table state across routers becomes trivial.

Many of the problems facing earlier distributed solutions are solved by maintaining a centralized view.

Other popular approaches to privacy, as described in Section 2, offer unacceptable delays (as we will demonstrate in Section 5) or reduced flexibility in terms of routing or user control. This decreased performance, coupled with new opportunities presented by SDN deployments, motivated our use of the SDN framework to achieve flexible privacy without compromising performance.

In this thesis, we present the issue of network privacy with this viewpoint in mind. We present *AnonyFlow* [30], an in-network anonymization service designed to efficiently and seamlessly provide privacy to users as they communicate with other endpoints and services. As an anonymization tool, *AnonyFlow* can also be used as a building block for a variety of services, such as PO Box [38], anonymization of network traces [37], as well as a way to provide separation between location and identification [31].

By enlisting cooperation from infrastructure providers and thus adopting an "in-network" approach to anonymization, endpoint privacy is provided in a seamless, user-transparent way. Unlike approaches such as Onion Routing [22], *AnonyFlow* incurs negligible overhead; for example, as shown by our performance evaluation experiments in Section 5, *AnonyFlow*'s impact on user-perceived latency is close to zero. Additionally, *AnonyFlow* requires no changes to endpoints which facilitates its deployment considerably. *AnonyFlow* is able to provide intra-domain anonymity, as well as dynamic, on-demand addresses. This ability to provide disposable, flow-based identifiers prevents malicious endpoints from tracking behavior and launching attacks on users.

To evaluate *AnonyFlow*'s functionality and performance in managed networks, we implemented a simple prototype using the OpenFlow platform [29]. OpenFlow enables execution of network-level services through a *controller*, which dictates the behavior and actions of switches under its jurisdiction. This enables the implementation of in-network, on-the-fly, packet morphing actions. Our evaluation over the hardware-based testbed showed that *AnonyFlow* provides higher flexibility in IP anonymization with no impact on the end-to-end latency and a minor deterioration in TCP throughput in wide-area networks. By design, *AnonyFlow* places a degree of trust in the network infrastructure provider who operates the OpenFlow controller. As discussed above, we contend that network infrastructure providers have all the incentives to provide anonymity to its users in a transparent fashion, with minimal impact on performance.

The remainder of this thesis is organized as follows: We examine the background of web privacy and software-defined networking in Section 2, followed by a definition of our anonymity model and goals in Section 3. Section 4 presents the basic architecture and operation of several common approaches to privacy; we also provide a detailed overview of our solution, *AnonyFlow* [30]. Finally, we present a performance and security evaluation of the approaches in Section 5, and conclude the thesis in Section 6.

2 Background

In this section, we will canvass the background of areas related to our study. First, we will examine related work in network privacy. Then, we will survey the state of the emerging field of software-defined networking.

2.1 Network Privacy

First, we briefly review other solutions to endpoint privacy, as well as different approaches to network addressing and identification.

Traditional network address translation (NAT)[44] provides a certain degree of privacy, but the public IP address can still typically be traced back to a single household, organization, or ISP. Additionally, it provides no privacy benefit to intranetwork communication between users behind the NAT box.

Simple anonymizing proxies, such as the Anonymizer[9], provide endpoint privacy but require trust in the proxy. Also, there is additional overhead of working at a higher layer or through tunneling, as well as the delay of routing traffic through the proxy rather than following the most efficient route to the destination.

Virtual Private Networks[4] are another popular solution used to hide network identity from the opposite endpoint. While widely supported and deployed, it has similar drawbacks as an anonymizing proxy - the user must trust the VPN service provider and traffic must flow through the provider network, which may not be the most efficient route to the destination.

Stronger overlay-based anonymity approaches include Onion routing[22], e.g., Tor[18], and Web mixes, e.g., JAP[6] and Tarzan[20]. While such solutions are still considered “low-latency” connection-oriented approaches compared to slower message-based anonymity systems[42], they still exhibit non-trivial overhead and noticeable delay (as we will discuss in later sections).

There also exist several privacy-preserving peer-to-peer networks. Freenet[14] is a distributed data store providing strong degree of anonymity and censorship-resistance, albeit with noticeable overhead. OneSwarm[28] leverages social networking and flexible permissions to offer a low-cost privacy option for P2P file sharing.

The BLIND framework[49] provides location privacy in IP networks through the use of public key endpoint identifiers and NAT-based forwarding agents.

There are several approaches that use the idea of a location-independent identifier. Mobile IP[38] was designed to allow users to communicate with a global identifier when changing network locations, through the use of "home" and "foreign" agents that store information about location-based addresses and redirect packets through IP tunnels. The Host Identity Protocol[34] was designed to separate the end-point identifier and location roles of IP addresses. Existing between the internetworking and transport layers, the HIP architecture is intended to decouple internetworking from the higher layers and replace all references to IP addresses within applications. Similarly, the Locator Identifier Separation Protocol (LISP)[31] separates endpoint identifies (EIDs) from routing locators (RLOCs). Endpoints address destinations via their EID, while the LISP border routers handle the network lookup and either encapsulate or rewrite the packet destination to the corresponding RLOC. While HIP approaches the issue by modifying the endstation, LISP uses a network-based location/identity split that we integrate into our design.

Molina-Jiménez and Marshall[32] proposed an approach that assigned temporary, random, IP and MAC addresses to users requiring anonymity on the Internet. Similarly in wireless LANs, Gruteser et al.[24] presented a mechanism to enhance location privacy through the use of disposable interface identifiers. We expand on these ideas in our system to provide clients with temporary identifiers that are linked to specific flows.

Just as traditional telephone companies are able to provide "caller-ID blocking" as an additional service to their customers, ISP's should be able to offer a service that automatically hides the IPs of their users without modification at the client-side. The Address Hiding Protocol (AHP)[39] by Raghavan et al attempts to do this. The mechanism is similar to the design of CPP[48], a system that encrypts IPv6 address to provide location privacy. AHP requires specially designed gateways that use time-based keys to keep in sync. This can cause collisions in cases of long-lived flows, and requires a somewhat involved design for handling multiple ingress/egress

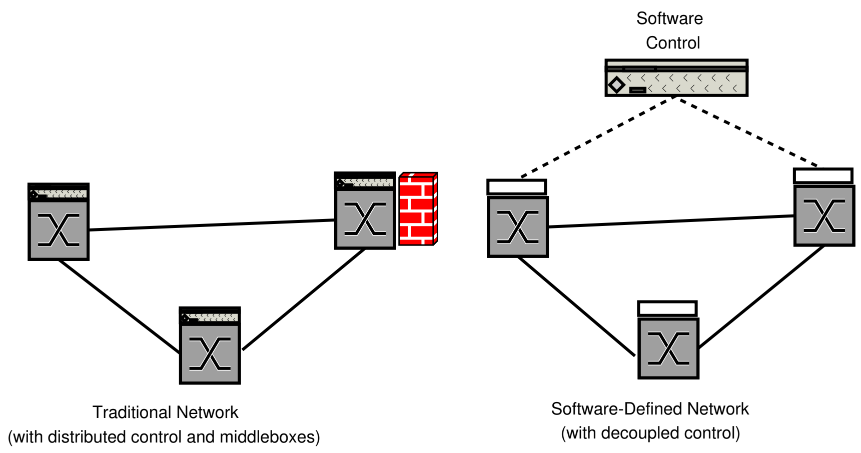


Figure 1: The SDN architecture decouples control logic from the forwarding hardware, and enables the consolidation of middleboxes, simpler policy management, and new functionalities.

points. We agree with the high-level goals of AHP, and hope to show how they can be more easily implemented and deployed using a software-defined networking approach.

2.2 Software-Defined Networking

An important enabling technology for our work is the emerging field of Software-Defined Networking (SDN). In this section, we will describe SDN and the problem it attempts to solve. We look at the history of SDN, from early ideas through recent developments, with a large focus on the OpenFlow[29] standard. We examine implications to other fields, and attempt to look at the future.

Software-Defined Networking was developed to facilitate rapid innovation and enable simple programmatic control of the network datapath. As visualized in Figure 1, the separation of the forwarding hardware from the control logic allows easier deployment of new protocols and applications, straightforward network visualization and management, and consolidation of various middleboxes into software control. Instead of enforcing policies and running protocols on a convolution of scattered devices, the network is reduced to "simple" forwarding hardware and the decision-making network controller(s). The forwarding hardware consists of (1) a *flow table* containing an entry and an action to take on active flows; and (2) an abstraction layer that securely communicates with a *controller* about new entries that are not currently on the flow table.

The decoupled system has been compared to an operating system[25], in which the controller provides a programmatic interface to the network, where applications can be written to perform management tasks and offer new functionalities. A layered view of this model can be viewed in Figure 2. This view assumes the control is centralized and applications are written as if the network is a single system. While this simplifies policy enforcement and management tasks, the bindings must be closely maintained between the control and the network forwarding elements.

One concern with the reactive centralized control model proposed by OpenFlow is scalability. Preliminary results by the OpenFlow developers show that a "controller based on a low-cost desktop PC could process over 10,000 new flows per second enough for a large college campus"[29]. Furthermore, any delay is only incurred by new flows - existing connections in the flow table are processed at line rate in the forwarding hardware. Nevertheless, the above model still exhibits scalability limitations and motivated proactive approaches, such as DIFANE[50], that push rules from the controller to a hierarchy of switches, such that the controller rarely needs to be consulted about new flows and traffic is kept in the data-plane. In their experiments, the DIFANE model reduces the added delay of the first packet from 10ms to under 1ms, while increasing the throughput a switch was able to handle from 20k flows/sec to 75k flows/sec[50].

SDN has been a recent focus of academia and industry, especially the recently formed Open Networking Foundation (ONF), the standardization body behind the OpenFlow protocol [29]. While the current efforts behind ONF have received the greatest attention, it is worth noting that the idea of programmable switches and decoupled control logic has been around for many years.

Some of the earliest ideas of programmable networks came out of the active networking[45, 46] research of the 1990s. The two main approaches involved (1) user-programmable switches, with in-band data transfer and out-of-band management channels; (2) capsules, in which every message sent by a user contained a program fragment which was interpreted by the routers. Despite a large amount of research attention, the active

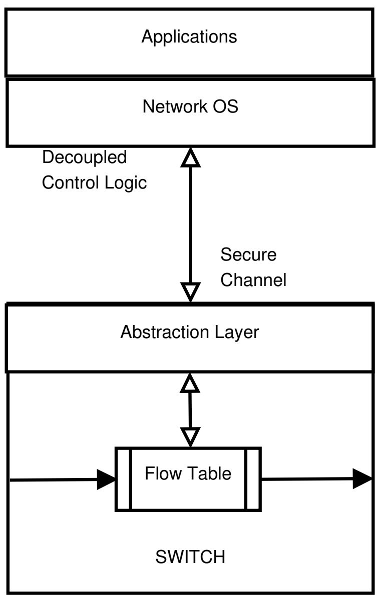


Figure 2: The separated control logic can be viewed as a network operating system, upon which applications can be built to "program" the network.

networking approach never transferred to widespread industry deployment, possibly due to practicality concerns over the security and performance of early protocols[33]. In contrast, the SDN approach taken by the Open Networking Foundation has not focused on *user* control of the network and has had industry involvement from the start.

A more recent idea of giving control mechanisms a global view of the network was proposed by the 4D project[40, 23, 11], which separated decision logic from protocols governing interaction between network elements.

The immediate predecessor to OpenFlow was Ethane[13], a network-policy controller. The idea of using a single, centralized controller to manage policy and security evolved into the broader concept of software-defined networking. To put Ethane in the context of OpenFlow today, the policy and security rules would likely be implemented as an application on top of a controller/"network operating system" such as NOX[25], Maestro[12], Beacon[5], SNAC, Helios, BigSwitch, etc.

An interesting type of proxy controller, called Flowvisor[43], can be used to add a level of network virtualization to OpenFlow networks and allow multiple controllers to manage the same set of physical switches. Initially developed to allow experimental research to be conducted on deployed networks alongside production traffic, it also facilitates and demonstrates the easy of deploying new services in SDN environments.

The applications of software-defined networks are quite broad, ranging everywhere from high performance data-centers[15, 26], to improving wireless mesh network hand-offs[16]. An area of future research we have been exploring is the use of SDN techniques in multi-hop wireless and infrastructure-less networks. To-date, current solutions have targeted infrastructure networks and are unsuited to disruption due to their centralized nature. We hope to adapt the flexibility, programmability, and control offered by SDN to environments that are inherently prone to delay and disruption. For the remainder of this thesis, we will focus on how the SDN paradigm can be applied to solve the issue of network privacy.

3 Model and Goals

In this section, we define the usability and threat model that is addressed in this thesis, and set design goals that we believe a privacy service should meet. We start by briefly surveying web usability work to determine user-tolerance for delay.

3.1 Web Usability

Web usability studies[7, 21, 35, 41] have generally agreed that the tolerable waiting time for a page load peaks at roughly 5-8 seconds without feedback. Additionally, long page loads negatively affect user satisfaction and perceived service quality. When using an anonymity system, a user's conceptual model[8], which affects QoS tolerance, may allow for a higher page load times and delay if the user expects an overhead for the secure environment. Nevertheless, a recent usability study[19] of Tor found that the expected Web user cancellation rate was 6 times greater, indicating high degree of user dissatisfaction and frustration. DNS requests in Tor were 40 times slower than direct connections. Although anonymity solutions such as Tor still have their applications, we argue that lower-latency approaches may have an important role both as stand-alone services as well as building blocks for applications that require endpoint anonymity.

Beyond the performance aspect, we are motivated by the observations of Dingledine and Mathewson[17]. The privacy service must be usable, even by users who are less familiar with technology. Futhermore, a larger anonymity set has the potential to provide better privacy. Expanding on this idea, as AHP[39] did, we believe a network-centric service enabled by network providers would reduce client misconfigurations and user confusion while increasing endpoint privacy.

3.2 Threat Model

Before describing our design goals, we briefly overview the threat model we employ. Unlike systems that wish to offer full anonymity, we place a measure of trust in the managed network provider and the network privacy service. In our work, the main "adversary" is the other endpoint. To a lesser extent, we also wish to hide information from third-party switches outside the managed network when traffic crosses the Internet.

In the network today, an intrusive endpoint may attempt to track user behavior based on the IP address of the connection. By correlating network logs with user actions, the "anonymity" that many users believe is implicit on the Internet is destroyed as usage patterns such as when the user connects, how often, to what services, etc. can be extracted. Furthermore, the proliferation of services such as WHOIS and IP geolocation allows third-party providers to learn a great deal about the location and possibly the real identity of the user. Besides passively monitoring user activity, active attacks may take place based on the information gleaned from IP address, where user experience may be altered or user access may even be blocked. While it is possible that user profiling may be used for a benign purpose, they can also lead to censorship or gross violations of user privacy. Our ideal privacy service attempts to decouple network identifiers from location and identity in order to provide users with truly free and universal Internet experience.

One final threat that should be noted is the "adversary within". As with any promise of privacy, it is possible that some users may abuse the service to attack others or perform illegal actions. While the managed nature of a service may easily allow privileges to be revoked, care must be taken lest the service itself becomes the main adversary to privacy.

3.3 Design Goals

We believe privacy services should protect users from endpoint logging at the least, while minimizing performance impact. Below, we list the main design goals for a lightweight privacy service:

- *Endpoint privacy* – the other endpoints should not be able to track source endpoint behavior or location based on the address they receive.
- *Minimal performance impact* – the privacy service should have minimal impact on the user's perceived latency when accessing Internet-based services.
- *Network-based design* – the privacy service should require minimal change to

the endpoints, and strive to eliminate user confusion and misconfiguration.

- *Disposable identifiers* – the privacy service should be able to provide dynamic, on-demand identifiers and therefore prevent the other endpoint from tracking behavior or launching attacks on the host's network address.

We should also point out that we do not try to address the following issues:

- *Data security* – in an orthogonal approach, applications are allowed to enforce their desired level of protection from eavesdropping and tampering through data encryption at the application layer. Likewise, we leave it to the users to select applications that are privacy-aware and will not leak identity information to the other endpoint.
- *Steganography* – a high-performance, lightweight service cannot attempt to achieve unobservability, such that a third-party is unable to tell that a message exists. A user may attempt to do so in parallel, perhaps by hiding messages within other data, such as digital media[27, 10].
- *Complete anonymity* – as previously highlighted, we assume that network providers are trusted entities and require their cooperation in concealing end-user identity. If a user intends to access restricted content or perform illicit actions in their jursidiction, a stronger level of anonymity should be pursued.

4 Approaches to Web Privacy

In this section, we will examine the architecture and operation of several web privacy approaches, including our own *AnonyFlow* [30].

4.1 Traditional

We first look at several "common" privacy services that apply to the threat model listed above and are familiar to one of ordinary skill in computer networks. Although some of the services were not originally designed to address the concern of user privacy, they have been deployed and used for those purposes today.

Network Address Translation (NAT)

Network Address Translation (NAT)[44] simply refers to the process of modifying the address information of in-transit packets based on pre-defined rules, usually to go from one address space to another, e.g., when interconnecting networks. There are many different variations on NAT, some of which are more suited to endpoint privacy and security than others, e.g., port address translation (PAT) dynamically allocates a network and port address to a flow, whereas static NAT allows the permanent allocation of a one-to-one address assignment for a host. As a tool, NAT has long been used to "masquerade" a private network behind a limited pool of global IP addresses, for the purposes of preserving the limited address space and providing some measure of privacy and security.

Anonymizing Web Proxy

Proxy servers have long been used for many purposes, including anonymization[9]. An anonymizing proxy will simply send a request on behalf of a client to a server and return the response; most importantly, it will allow the server to believe that the request originated at the proxy. Beyond the performance associated with using a proxy, the user must trust the proxy server to not eavesdrop or modify unencrypted application data.

Tor Onion Routing

Onion routing[22, 18], takes the ideas of anonymizing proxy servers to the next level, and attempts to hide the client from even the proxy. Basically, proxies are chained together and packets are encrypted such that each proxy only has knowledge about the next and previous hop; therefore, no single proxy is aware if the previous hop was the true message source. While this provides a high level of security, the overhead associated with encrypting/decrypting the data multiple times and visiting multiple proxies can slow down performance significantly. Additionally, full data encryption is not provided unless supported at the application layer, as traffic that exits the Tor network is unencrypted, leaving the exit proxy aware of the message contents and the ultimate destination.

Virtual Private Network (VPN)

VPNs[4] have long been used to provide a user remote access to an organizational network, but have been increasingly used to bypass censorship and act as secure proxies in recent years. There are many implementations of secure VPNs, but they all typically require authentication and will encrypt traffic between the remote host and the private provider network; traffic within the provider network or destined for outside endpoints may not be encrypted. Although VPNs may obfuscate traffic details from the network provider of the remote endpoint, the user shifts his trust to the VPN provider.

4.2 *AnonyFlow*

We designed our service to reflect the goals defined in Section 3. We will explain the architecture and operation of *AnonyFlow*, and introduce our implementation.

4.2.1 Architecture

AnonyFlow is designed to provide endpoint privacy by concealing the source identifier from the other side of the connection. Additionally, the in-network based approach allows a level of accountability that can be used to revoke access to malicious users.

Identifiers

Before presenting *AnonyFlow*'s architectural components and operation, we describe the different identifiers *AnonyFlow* employs to represent users. Note that we discuss our work in the context of a global deployment, but our work is equally applicable if we operate on Layer 2 identifiers within a local network.

- *Machine IP address* – the address assigned to the machine, and what would normally be seen by the other endpoint if *AnonyFlow* is disabled. When using *AnonyFlow*, it is rewritten as soon as possible to another form, and only changed back when delivering messages back to the machine.
- *AnonID* – the identifier that the other endpoint receives as it provides no information of the location or identity of the machine. They may be used on a temporary basis and discarded at the end of a flow; alternatively, it can be used as a more permanent identifier for a service that wishes to remain anonymous.
- *Network IP address* – this address is routable back to the managed network of the machine, and is necessary for traversing the Internet. It is associated with a given AnonID and can be changed on demand.

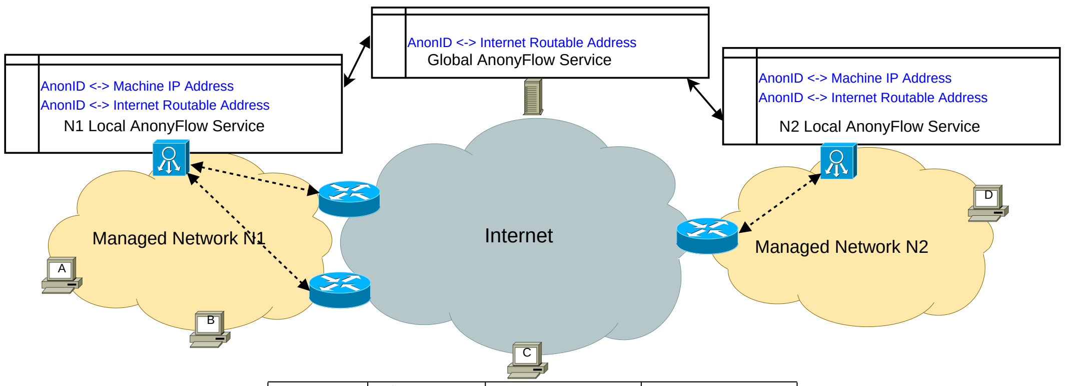
Components

Each managed network participating in the *AnonyFlow* service can be thought of as a *local domain.* The *AnonyFlow* service itself consists of a *local* and *global* component. At the border of each local domain, AnonIDs must be translated to an identifier (e.g., IP address) that would allow flows to traverse intermediate routers.

- *AnonyFlow conduit* – consults with the local *AnonyFlow* service, rewrites IP addresses to/from AnonIDs, and forwards resulting packets towards destination.
- *Local AnonyFlow service* – handles user join/leave and local mappings, and communicates with global service.

- *Global AnonyFlow service* – handles network lookup for AnonIDs outside local managed network.

Example Operation


AnonID <-> Internet Routable Address
Global AnonyFlow Service
AnonID <-> Machine IP Address
AnonID <-> Internet Routable Address
N1 Local AnonyFlow Service
AnonID <-> Machine IP Address
AnonID <-> Internet Routable Address
N2 Local AnonyFlow Service
Managed Network N1
Internet
Managed Network N2
A
B
C
D

Entity	AnonID	Network IP	Machine IP
A	1	11.11.X.X	11.11.0.1
B	2	11.11.X.X	11.11.0.2
C	-	12.12.0.1	12.12.0.1
D	3	13.13.X.X	13.13.0.1

Figure 3: Example of *AnonyFlow*'s Operation.

We use the scenario depicted in Figure 3 to illustrate *AnonyFlow*'s operation and examine the series of events that occur when host A opens a connection to the hidden service on host B. First, A sends a packet with source '11.11.0.1' and destination '2' that will pass through the *AnonyFlow* conduit on network $N1$. The conduit will consult with the *AnonyFlow*'s local service of $N1$ for the first packet of the flow. The local service determines that the source address is associated with AnonID '1', and the destination AnonID '2' is within the same network. The conduit will then rewrite the source address to '1' and the destination to '11.11.0.2' and set up rules to forward the flow to the destination.

If the destination AnonID is in another network, for instance, when host A communicates with host D, there are a few more steps. The local service must do a global lookup of the destination AnonID to determine an address routable to the destination network. It must also assign a routable address to the source, in a manner similar to NAT. Finally, when the flow arrives at a conduit in the destination network, it will rewrite the source address to AnonID '1' and the destination to the

Entity	AnonID	Network IP	Machine IP
B	X		
C		X	
D	X		
N1 Service	X	X	X
N2 Service	X	X	
Global Service	X	X	

Table 1: Identifiers of host A as observed by other agents.

machine address of D.

In the final case of the destination being outside the *AnonyFlow* namespace, such as when host A communicates with host C, our system resembles a traditional yet more flexible NAT service. The conduit rewrites the source address with an address routable to the source network so that the destination is able to trace the message directly back to the source network.

In Table 1, we summarize the identifiers of host A that each network entity is able to observe when host A communicates with host B, C, and D.

4.2.2 Implementation

Although there are a number of ways to enable *AnonyFlow* in a managed network, we use the OpenFlow platform in our reference implementation. OpenFlow provides the means for rapid deployment and execution of network services by enabling a remote controller to modify the behavior of switches and routers. By providing direct access to the switch flow table, the OpenFlow API allows services to achieve custom routing and packet processing.

In our OpenFlow implementation, each local *AnonyFlow* domain consists of a network managed by a single OpenFlow controller. *AnonyFlow* conduits correspond to OpenFlow-enabled switches, while the local *AnonyFlow* service is integrated with the OpenFlow controller. *AnonyFlow*'s global service exists outside of the OpenFlow infrastructure as a directory service.

5 Evaluation

5.1 Performance

First, we demonstrate the performance of existing privacy approaches identified in the previous section in a series of web measurements; we believe the results illustrate the opportunity that can be addressed by a lightweight approach such as *AnonyFlow*. We then evaluate the performance of *AnonyFlow* along with several of the common approaches on a network testbed with OpenFlow switches.

5.1.1 Web Measurements

We measured the performance of existing privacy options, including web proxies and Tor onion routing, and found most to exhibit a significant performance overhead, which caused response times to stress or exceed recommendations from the web usability studies. We conducted these measurements over a period of several weeks in the Deutsche Telekom Los Altos Laboratory, comparing to a base;ome of non-anonymized networking. The *proxy* measurements use a list of commercial web proxies, while the *VPN* measurements use the UC Santa Cruz VPN. When setting up *Tor* circuits, we picked 3 random relays from around the world for each new connection.

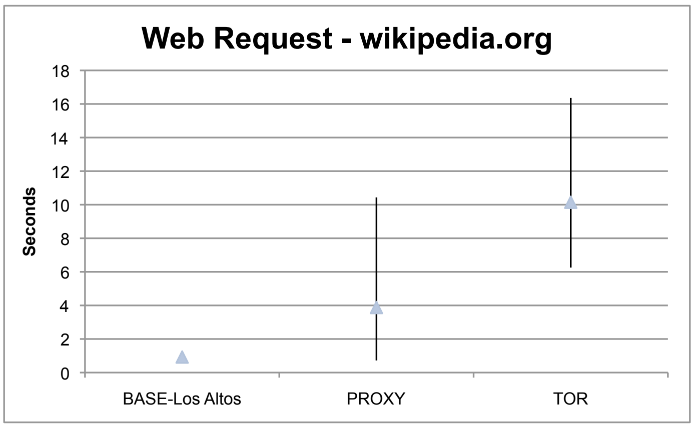


Figure 4: Webpage load time using common privacy tools.

In Figure 4, we measure the time it takes to load the front page of wikipedia.org. As may be expected, the actual performance of the privacy options depended on the

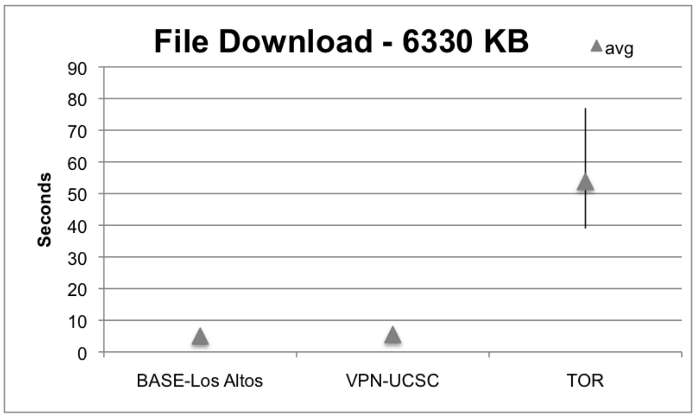


Figure 5: File download time using common privacy tools.

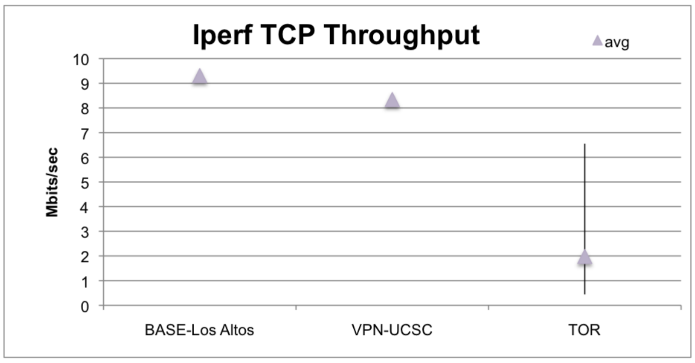


Figure 6: TCP throughput using common privacy tools.

load and location of the relays in relation to the endpoints. On average, proxies had a response time more than 4 times slower than a direct connection, while Tor circuits were more than 10 times slower. Furthermore, the Tor results fell outside what would commonly be considered tolerable waiting times for a regular web service.

In Figure 5, we looked at the download time of a 6330 Kilobyte file from UCSC to the Los Altos laboratory with Tor and the UCSC VPN. As expected, Tor was about 10 times slower on average. Despite the fact that the VPN provider was located on the same network as the file server, the VPN connection was 10% slower on average than the direct connection. Although the difference is small when compared to Tor, it is not insignificant; performance would be expected to degrade further depending on the distance from the endpoints to the VPN provider.

Finally, we compared TCP throughput between UCSC and Los Altos using *iperf* in Figure 6. Much like the file download, the VPN throughput was about 10% less than a direct connection on average, while the Tor throughput was much lower.

5.1.2 Testbed Measurements

To evaluate the performance of *AnonyFlow*'s OpenFlow-based implementation, we deploy it in a real network testbed with OpenFlow switches. We focus on IP identifier anonymization through header rewriting. This section describes our deployment and compares the performance of *AnonyFlow* with other related approaches.

Testbed

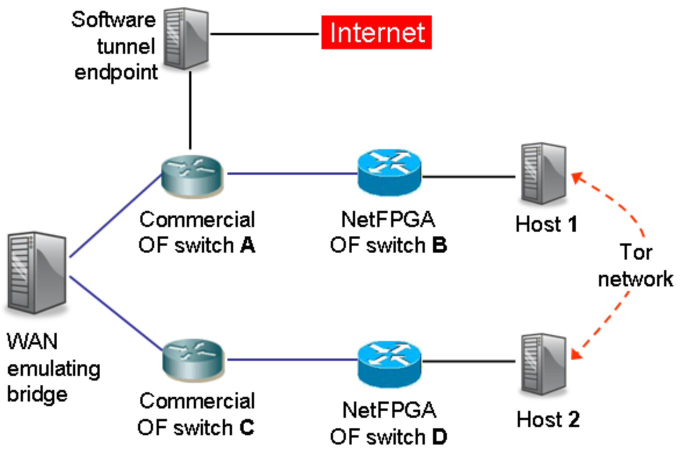


Figure 7: Lab network used to emulate a wide-area OpenFlow-enabled network. The NetFPGA-based OpenFlow switches at the edge take care of the required header rewriting actions.

As shown in Figure 7, the testbed has two logical sub-networks interconnected by a Linux bridge that runs the `netem` toolkit so as to emulate a wide-area network [2]. Each subnetwork consists of two commercial OpenFlow-enabled switches and two NetFPGA[36]-based OpenFlow switches, all of which are controlled by a remote NOX[25]-based *AnonyFlow* controller. This testbed provides two main features:

- Header re-writing capability, which allows re-writing the IP fields (i.e., src IP or dst IP or both) of all packets at line-rate, in the edge switches (i.e., NetFPGA-based one).
- TCP file transfer time characterization over a wide-area network with varying end-to-end round trip times.

Results

In Figure 4 we presented the latency characterization of common privacy-preserving tools. *AnonyFlow*'s performance is almost the same as the base case because the anonymization is done by an en-route switch. This happens at line speed and does not incur additional delays.

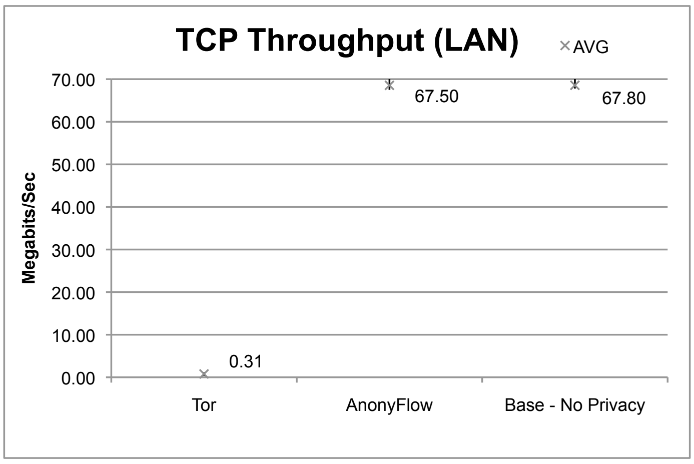


Figure 8: TCP Throughput (`iperf`) between two hosts in our testbed.

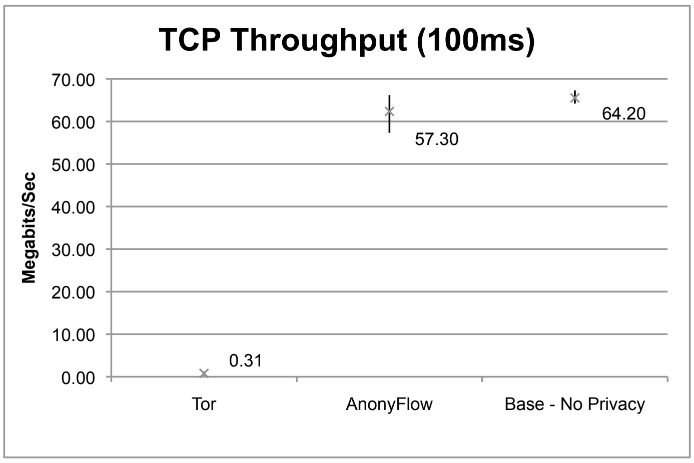


Figure 9: TCP Throughput (`iperf`) between two hosts with emulated 100ms delay.

In Figures 8 and 9 we present the TCP throughput characterization performed over the testbed shown in Figure 7. In this characterization, we ran `iperf`[1] between the two hosts (Host 1 and 2 in Figure 7) multiple times, with each run lasting 25 seconds. The mean, min and max values are shown in the two figures; the mean is denoted by a 'x' marker, and the vertical line around the mean value represents the min to max range. In Figure 8, we observe that *AnonyFlow* does not experience a throughput deterioration, while Tor achieves throughput that is a few orders of magnitude lower than the direct route. This is because Tor uses additional relay hops in the Internet without much guarantees on performance. We, then, add a 100ms delay, so as to emulate a wide-area network, between the 2 hosts and plot the performance in Figure 9, the same Tor results were used in Figure 9 because Tor traffic already goes over the real Internet. *AnonyFlow* experienced a 10.75% decrease in throughput. Thus, in the wide-area case, *AnonyFlow* provides a good tradeoff between performance (as measured by latency and throughput) and flexibility.

5.2 Security

In this section, we examine possible attacks on users and on the privacy service itself. In general, as specified in our design goals, we expect the privacy service to protect against a single passive adversary and not more exhaustive global attacks.

Passive attacks

- *Network address tracking* – a static IP address allows users to be tracked, and may also leak information about identity and location.

  NAT may hide your identity if the other user is on a different network, but will usually disclose enough information to locate the user's home network and does not typically offer control over the length of time a particular address is used or for which flows it would be used.

  An anonymizing proxy may obsfuscate your activity from the opposite user, but it requires trust in an additional third-party. When using a particular proxy, a user will also always be associated with the proxy address; therefore, traffic is easily traced back to the proxy, and the user may be blocked by the other endpoint due to the behavior of other proxy users. On the other hand, if the user is the only proxy user to regularly communicate with the other endpoint, then the adversary would still be able to track user behavior based on the proxy address.

  VPN services have similar drawbacks to proxies and NAT.

Tor offers a good amount of anonymity, provided the user correctly configures the system and avoids applications that send unencrypted identifying information. Third-party adversaries have been known to act as Tor exit routers and either sniff the unencrypted traffic or inject malicious scripts aimed at disclosing identity. In addition, just as with proxies, traffic from Tor exit routers have been known to be blocked by endpoints due to unregulated behavior by other Tor users.

AnonyFlow makes it difficult to track behavior by using disposable, on-demand, flow-based identifiers when talking to endpoints within domains; otherwise, temporary IP addresses are assigned when communicating with traditional Web servers.

- *Timing correlation* – in some cases, the adversary can learn information by observing packet inter-arrival times. Unlike high-latency anonymity systems that introduce delays or alter the timing characteristics of a connection, none of the systems we examine try to hide timing information. Additionally, *AnonyFlow* attempts to maximize performance by sending messages using the most efficient routing path which is provided by the underlying network routing service.
- *Content analysis* – packet payloads may contain identifying information. We leave it to the user to select applications that provide data privacy. None of the services we examine provide full end-to-end encryption, with the exception of Tor in the case that the other endpoint is a hidden service.

Active attacks

- *Direct attacks on identifiers* – adversaries may attempt to launch a denial of service attack on the provided address or port scan for vulnerabilities on the user machine.

NAT - depending on the type and setup, NAT may or may not prevent other hosts from attacking the host machine using the global address provided by NAT.

An anonymizing proxy will usually prevent the opposite endpoint from contacting you.

VPN services are usually not designed to prevent communication, and may not stop an adversary.

Tor offers a good level of protection, provided the user correctly configures the system. A potential downside is that your real network address is disclosed to other Tor routers, which may be third-party adversaries.

AnonyFlow - the use of temporary, flow-based "AnonIDs" offers a good level of protection and indirection to endpoints as it is not possible to communicate with an identifier after it has been released or expired.

- *Denial of service on service* – an adversary may attempt to overwhelm a privacy service with false requests.

NAT - depending on the type and setup, NAT may or may not prevent other hosts from attacking the host machine using the global address provided by NAT.

An anonymizing proxy is vulnerable to DoS attacks.

VPN services are vulnerable to some extent.

Tor routers, which individually owned and operated PCs with routing software, may be as vulnerable to DoS attacks as any other endpoint; with the exception that their addresses are made publically available when participating in Tor routing service.

AnonyFlow - This may be dealt with on-the-fly by pushing rules that dump excessive flows from offenders at the nearest conduit.

- *Router subversion* – though not part of the threat model we presented, it is possible that an intermediate router may be subverted into provided some information to the adversary.

NAT - if the router is behind the NAT, it can disclose the true source of the message. Of course, a compromised NAT device can fully disclose all information.

An anonymizing proxy - if the router is between the proxy and the source, then it can disclose the true source to the adversary. As with NAT, a compromised proxy can fully disclose all information.

Tor is designed to prevent any one router from learning the message source; however, a compromised exit router may disclose message contents to an adversary.

AnonyFlow - if an intermediate router is compromised by the adversary, they may learn information about the hidden endpoint. In the worst case, the protection offered degenerates to a simple network address translation (NAT) service. If an Internet router is subverted it can provide information about the source and destination networks (though not the actual machines). Likewise, if a local privacy service colludes with the adversary, it can provide full information about local mappings but only the source network of other identifiers. To protect against subversion of switches within domains, we require authentication and use encrypted communication between switches and the privacy service.

- *Man-in-the-middle* – if an attacker is able to fully compromise a router, then they are able to listen and inject into the communication between two endpoints. Most of the approaches we examined do not provide any native data encryption when transporting flows; Tor and VPN are two exceptions, but even then the protection does not cover end-to-end if an application uses unencrypted data. If an intermediate router is compromised by the adversary, they may view and/or modify the content of the payload; it is left to the user to select applications that offer data privacy and integrity.
- *Governmental authority* – In most cases, endpoint privacy is provided within the network infrastructure rather than at the endpoints; therefore, it is subject to disclosure and/or shutdown by entities with authority over that infrastructure. As such, it is well suited to protecting the identity of users with legitimate activities but ill-advised for criminals or users living under oppressive regimes. A notable exception is Tor, which attempts to hide identity information from

the routers themselves.

Feature Comparison

In this thesis we presented *AnonyFlow*, an in-network endpoint anonymization service designed to provide privacy to users. Through experiments on a real network testbed, we show that our proof-of-concept OpenFlow-based prototype of *AnonyFlow* significantly outperforms Tor [3], an Onion Routing-based approach, and delivers similar performance when compared to non-anonymized network access, while providing more features than the other schemes (as summarized in Table 2).

Features	NAT	Proxy	Tor	*AnonyFlow*
Hide source from opposite endpoint	X	X	X	X
Hide source from relays			X	
Change identifier on-demand			X	X
L2 or Intra-domain anonymization		X		X
Provide hidden services			X	X
Optimal routing	X			X

Table 2: Feature comparision of 3 anonymization techniques.

6 Concluding Remarks

Internet privacy is a growing area of user trepidation, yet for many, a controlled disclosure of information would suffice over full anonymity. In this case, a service offered by the network provider would offer a simple and low-impact solution to maintain privacy while communicating with other endpoints.

The advent of software-defined networking (SDN), which creates a separation between control decisions and routing hardware, provides an opportunity for network service providers to rapidly deploy new protocols and services that would previously have required specialized middle boxes and complex synchronization.

Our solution, *AnonyFlow*, offers a lightweight endpoint privacy service with minimal performance impact.

Directions for future work include:

- *Scalability* – currently, *AnonyFlow*'s global mapping service that maps AnonIDs to networks is implemented using a centralized architecture. As the number of users and domains grow, a distributed service would reduce possible bottlenecks.
- *Increased privacy on the existing Internet* – the decision to follow the best routing path also reduces the protection offered when visiting endpoints outside *AnonyFlow* domains to the level offered by NAT solutions today. To better hide information about the "source" network, it may be sensible to route traffic through additional proxy domain(s).
- *Increased access to hidden services from the existing Internet* – currently, only users from within *AnonyFlow* domains are able to access hidden services offered by other *AnonyFlow* users. In the future, a proxy mechanism should be developed that allows outside users to utilize these services, much like tor2web[47] enables access to hidden services on the Tor network.
- *Placement* – it suffices to place OpenFlow switches with header rewriting capability in only a few strategic locations of the network. We plan to further investigate this option, which allows incremental deployment.

References

[1] iperf tool. http://iperf.sourceforge.net.

[2] Netem tool. http://www.linuxfoundation.org/collaborate/workgroups/networking/netem.

[3] Tor anonymity network. http://www.torproject.org.

[4] L. Andersson and T. Madsen. Provider Provisioned Virtual Private Network (VPN) Terminology. RFC 4026 (Informational), March 2005.

[5] Beacon. http://beaconcontroller.net/.

[6] O. Berthold, H. Federrath, and S. Kopsell. Web mixes: A system for anonymous and unobservable internet access. In *Designing Privacy Enhancing Technologies*, pages 115–129. Springer, 2001.

[7] N. Bhatti, A. Bouch, and A. Kuchinsky. Integrating user-perceived quality into web server design. *Computer Networks*, 33(1):1–16, 2000.

[8] A. Bouch, A. Kuchinsky, and N. Bhatti. Quality is in the eye of the beholder: meeting users' requirements for internet quality of service. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 297–304. ACM, 2000.

[9] J. Boyan. The anonymizer. *CMC Magazine*, 1997.

[10] S. Burnett, N. Feamster, and S. Vempala. Chipping away at censorship firewalls with user-generated content. In *Proc. 19th USENIX Security Symposium, Washington, DC*, 2010.

[11] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe. Design and implementation of a routing control platform. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pages 15–28. USENIX Association, 2005.

[12] Z. Cai, AL Cox, and TSE Ng. Maestro: A system for scalable openflow control. Technical Report TR10-08, Rice University, December 2010.

[13] M. Casado, M.J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: Taking control of the enterprise. *ACM SIGCOMM Computer Communication Review*, 37(4):1–12, 2007.

[14] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing Privacy Enhancing Technologies*, pages 46–66. Springer, 2001.

[15] A.R. Curtis, J.C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee. Devoflow: Scaling flow management for high-performance networks. In *ACM SIGCOMM*, 2011.

[16] P. Dely, A. Kassler, and N. Bayer. Openflow for wireless mesh networks. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1–6. IEEE, 2011.

[17] 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), Cambridge, UK, June*, 2006.

[18] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th conference on USENIX Security Symposium-Volume 13*, pages 21–21. USENIX Association, 2004.

[19] B. Fabian, F. Goertz, S. Kunz, S. Müller, and M. Nitzsche. Privately waiting–a usability analysis of the tor anonymity network. *Sustainable e-Business Management*, pages 63–75, 2010.

[20] M.J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 193–206. ACM, 2002.

[21] D.F. Galletta. *Web site delays: How tolerant are users?* PhD thesis, Citeseer, 2002.

[22] D. Goldschlag, M. Reed, and P. Syverson. Onion routing. *Communications of the ACM*, 42(2):39–41, 1999.

[23] A. Greenberg, G. Hjalmtysson, D.A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang. A clean slate 4d approach to network control and management. *ACM SIGCOMM Computer Communication Review*, 35(5):41–54, 2005.

[24] M. Gruteser and D. Grunwald. Enhancing location privacy in wireless lan through disposable interface identifiers: a quantitative analysis. *Mobile Networks and Applications*, 10(3):315–325, 2005.

[25] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. Nox: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*, 38(3):105–110, 2008.

[26] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. Elastictree: Saving energy in data center networks. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, pages 17–17. USENIX Association, 2010.

[27] T.S. Heydt-Benjamin, A. Serjantov, and B. Defend. Nonesuch: a mix network with sender unobservability. In *Proceedings of the 5th ACM workshop on Privacy in electronic society*, pages 1–8. ACM, 2006.

[28] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson. Privacy-preserving P2P data sharing with OneSwarm. *ACM SIGCOMM Computer Communication Review*, 40(4):111–122, 2010.

[29] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[30] Marc Mendonca, Srini Seetharaman, and Katia Obraczka. A flexible in-network ip anonymization service. In *the IEEE ICC Workshop on Software Defined Networks*, June 2012.

[31] David Meyer. The locator identifier separation protocol (lisp). *The Internet Protocol Journal*, 11(1):23–36, March 2008.

[32] C. Molina-Jiménez and L. Marshall. True anonymity without mixes. In *wiapp*, page 32. Published by the IEEE Computer Society, 2001.

[33] J.T. Moore and S.M. Nettles. Towards practical programmable packets. In *Proceedings of the 20th Conference on Computer Communications (INFOCOM)*. Citeseer, 2001.

[34] R. Moskowitz and P. Nikander. Host Identity Protocol (HIP) Architecture. RFC 4423 (Informational), May 2006.

[35] F.F.H. Nah. A study on tolerable waiting time: how long are web users willing to wait? *Behaviour & Information Technology*, 23(3):153–163, 2004.

[36] Netfpga platform. `http://netfpga.org`.

[37] Ruoming Pang, Mark Allman, Vern Paxson, and Jason Lee. The devil and packet trace anonymization. *SIGCOMM Comput. Commun. Rev.*, 36:29–38, January 2006.

[38] C. Perkins. IP Mobility Support for IPv4, Revised. RFC 5944 (Proposed Standard), November 2010.

[39] B. Raghavan, T. Kohno, A. Snoeren, and D. Wetherall. Enlisting isps to improve online privacy: Ip address mixing by default. In *Privacy Enhancing Technologies*, pages 143–163. Springer, 2009.

[40] J. Rexford, A. Greenberg, G. Hjalmtysson, D.A. Maltz, A. Myers, G. Xie, J. Zhan, and H. Zhang. Network-wide decision making: Toward a wafer-thin control plane. In *Proc. HotNets*, pages 59–64. Citeseer, 2004.

[41] P.R. Selvidge, B.S. Chaparro, and G.T. Bender. The world wide wait: effects of delays on user performance. *International Journal of Industrial Ergonomics*, 29(1):15–20, 2002.

[42] A. Serjantov and P. Sewell. Passive attack analysis for connection-based anonymity systems. *Computer Security–ESORICS 2003*, pages 116–131, 2003.

[43] R. Sherwood, M. Chan, A. Covington, G. Gibb, M. Flajslik, N. Handigol, T.Y. Huang, P. Kazemian, M. Kobayashi, J. Naous, et al. Carving research slices out of your production networks with openflow. *ACM SIGCOMM Computer Communication Review*, 40(1):129–130, 2010.

[44] P. Srisuresh and K. Egevang. Traditional IP Network Address Translator (Traditional NAT). RFC 3022 (Informational), January 2001.

[45] D.L. Tennenhouse, J.M. Smith, W.D. Sincoskie, D.J. Wetherall, and G.J. Minden. A survey of active network research. *Communications Magazine, IEEE*, 35(1):80–86, 1997.

[46] D.L. Tennenhouse and D.J. Wetherall. Towards an active network architecture. In *DARPA Active NEtworks Conference and Exposition, 2002. Proceedings*, pages 2–15. IEEE, 2002.

[47] Tor2Web. `http://tor2web.org/`.

[48] J. Trostle, H. Matsuoka, M.M.B. Tariq, J. Kempf, T. Kawahara, and R. Jain. Cryptographically protected prefixes for location privacy in ipv6. In *Privacy Enhancing Technologies*, pages 142–166. Springer, 2005.

[49] J. Ylitalo and P. Nikander. Blind: A complete identity protection framework for end-points. In *Security Protocols*, pages 163–176. Springer, 2006.

[50] M. Yu, J. Rexford, M.J. Freedman, and J. Wang. Scalable flow-based networking with difane. In *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM*, pages 351–362. ACM, 2010.