Lost In the Edge: Finding Your Way With Signposts

Charalampos Rotsos, Heidi Howard, David Sheets, Richard Mortier[1]
Anil Madhavapeddy, Amir Chaudhry, Jon Crowcroft
University of Cambridge, [1]University of Nottingham

Abstract

The *de facto* architecture of today's Internet services all but removes users' ability to establish inter-device connectivity except through centrally controlled "cloud" services. Whilst undeniably convenient, the centralised data silos of the cloud remain opaque and an attractive target for attackers. A range of mechanisms exist for establishing secure peer-to-peer connections, but are inaccessible to most users due to the intricacy of their network configuration assumptions. Users effectively give up security, privacy and (when peers are both on the same LAN) low-latency simply to get something useable.

We observe that existing Internet technologies suffice to support efficient, secure and decentralized communication between users, even in the face of the extreme diversity of edge connectivity and middlebox intervention. We thus present *Signpost*, a system that explicitly represents individual users in a network-wide architecture. Signpost DNS servers create a "personal CDN" for individuals, securely orchestrating the many different available techniques for establishing device-to-device connectivity to automatically select the most appropriate. A DNS API gives application compatibility, and DNSSEC and DNSCurve bootstraps secure connectivity.

1 Introduction

> *"DNS servers can play games. As long as they appear to deliver a syntactically correct response to every query, they can fiddle the semantics."*
>
> — RFC3234 [7]

Our requirements for inter-user and inter-device communication have grown in recent years, with the dramatic increase in the number of personal devices driving the need for networked resource sharing mechanisms. At the same time, the growth of social networking has transferred a large part of users' social activity onto the Internet. Our online identities are closely associated with our real life identities, and individuals are increasingly vulnerable to online digital threats, either from malicious third-party attacks or mass-surveillance[1].

Many Internet-wide services try to address the connectivity requirements by using the public cloud to route all device interconnection via third-party services, which is unsatisfactory. Users offload their data to centrally controlled third-party infrastructure, defining user-based access policies under the assumption that they have full control over access to their data. However, cloud services introduce an additional application-layer hop to the communication process, which as a result becomes susceptible to data eavesdropping and censorship, by both the service provider and intermediate routing entities. Unfortunately, the interests of service providers and users are not always aligned. Service providers have been observed to capriciously modify their terms of service [23], and provide very weak user service-level agreements that minimise legal responsibility [11]. Governments also legally force service providers to censor information [20, 27], block services [10] and even demand encryption keys [29].

The success of cloud services can be attributed in large part to the current state of the Internet, which is broadly divided into two halves: a high performance, global *core network* and multiple, heterogeneous and complex *edge networks*. User content is stored in the core network and served from content delivery networks via high-speed network links [28] from globally accessible hosts.

In contrast, edge networks are highly heterogeneous, regulated, often constrained in their connectivity towards the core, and plagued by transparent, programmable middleboxes. As a result, key properties of the Internet abstraction, such as bi-directional connectivity, have become things of the past. Home network NATs hide devices from the Internet and ISPs regulate application functionality [12, 16], while enterprise network admin-

[1] http://guardian.co.uk/world/2013/jun/06/us-tech-giants-nsa-data

istrators restrict unwanted communications with coarse-grained firewall policies [33]. The edge network is thus where the end-to-end architectural principles of the original Internet have been largely abandoned [6], to be replaced by use of centrally controlled services as a stepping stone to inter-device connectivity. Alternative peer-to-peer (P2P) approaches require extensive re-implementation of existing Internet services and require a critical mass of users, while their performance is susceptible to user resource altruism [25] and their cooperative nature is subject to privacy threats [9].

We present *Signpost*, a network architecture for distributed, authenticated and encrypted user communications. Signpost automates the configuration and orchestration of off-the-shelf network connection mechanisms, providing an abstraction layer to deploy distributed applications that can operate through the modern quagmire of middleboxes, NATs and firewalls. Users explicitly give their devices unique names in the DNS hierarchy within their own zone. Signpost DNS servers then establish optimal ad-hoc paths between pairs of devices in response to DNS lookups. Our prototype uses DNSSEC [2] for authentication and DNSCurve [3] for encrypting requests over existing resolver infrastructure, ensuring that users never need to manually configure any tunnels.

We next elaborate some use-cases (§2) to motivate our design goals, and then describe the Signpost architecture and how to reuse existing Internet services to re-establish a user-centric abstraction for identity and connectivity (§3). We conclude with related work (§4) and discussion of several challenges our design poses (§5).

2 Motivation & Threat Model

Local networks, including home networks, support an ecosystem of applications and protocols, e.g., NFS, IPP and VNC, enabling users to freely share data and resources between devices; while service discoverability mechanisms, e.g., Bonjour and SSDP, help to automate device and network configuration. A number of open-source tools exist that can create end-to-end communication channels with specific security properties across the Internet. Our goal is to link these mechanisms together programatically to enable direct device communication *without* relying on a cloud-based solution and connectivity to the global Internet. We next elaborate this motivation by considering two distinct scenarios.

2.1 Alice: Secure, private file sharing

Alice has a mobile phone and a desktop PC. Her phone normally achieves Internet connectivity using her mobile provider's 3G network while also opportunistically taking advantage of known WiFi networks when in proximity. Her desktop PC is located at her work, behind a NAT and a stateful firewall which implements a "drop by default" policy. Alice has no control over the network policy and Universal Plug and Play (UPnP) port forwarding is not supported. When both devices are connected to the enterprise network, Alice can access files on her desktop from her phone using SMB. When she is away from her desk, she must remember to use her corporate VPN client to enable this functionality, although this sometimes interferes with her use of other services at her remote location. Alice wants to securely access confidential work-related files, independently of her location, in accordance with her employer's security policies.

Threats. If Alice uses a third-party file-sharing service, she is vulnerable to the service provider snooping the contents, and also to law enforcement pressuring the provider to release the data to them. Furthermore, Alice is vulnerable to a denial-of-service if her provider decides to lock her online account, and the client access software installed on her computer could potentially remote wipe even her local copy of files. The contents of the files also become physically spread across remote datacenters, and so vulnerable to larger scale data leaks.[2]

Implications. Alice can bypass the threats above by keeping her data local – she just needs a mechanism to enable her smartphone to establish secure connectivity with her desktop PC over the lower-layer channels as she roams around different edge networks. It must be backwards compatible with existing applications and devices to fit her existing user experience.

2.2 Bob: Private audio streaming

Another individual, known here as Bob, needs to communicate with a friend, Charlie. Bob's government censors some Internet services through DNS and URL filtering and Deep Packet Inspection. Bob wishes to chat privately with Charlie, hiding such conversations from his government-controlled ISP. Bob's laptop is located within his NATed home network, and connects to the Internet via dynamic public IP addresses acquired by his home DSL router.

Threats. Charlie is unable to disseminate information to Bob using well-known 3rd party audio chat services as they are censored or intercepted by Bob's government. Bob knows that he can use Tor to hide his conversation, but still needs to ensure that the audio stream is encrypted end-to-end.

Implications. Bob and Charlie require an authenticated, globally-accessible rendezvous service through which they can meet and privately negotiate subsequent

[2]E.g., In June 2011 the popular Dropbox service exposed all user data without authentication for four hours. https://blog.dropbox.com/2011/06/yesterdays-authentication-bug

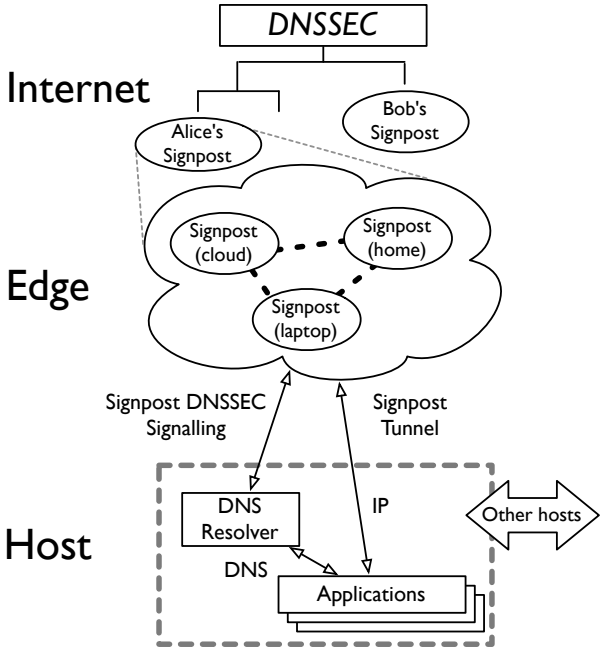

Figure 1: Signpost architecture.

channel parameters. Such a rendezvous service must accommodate aliases so that they can obfuscate the fact of their communication. Finally, to avoid potentially dangerous errors, particularly as they roam across different networks, they need the service to automatically select among all available underlying channels, and implement the most appropriate, without relying on centrally controlled cloud-hosted services that are easy targets for surveillance.

3 Signpost Architecture

Figure 1 gives an overview of the Signpost architecture, which has elements in the network core and edge and, optionally, the end host. The goal is to map a domain name hierarchy to a personal cloud of devices spread across this network graph. Not all devices are created equal and many, e.g., the Apple iPhone, only permit limited modification due to vendor restrictions. The application interface is based on zero-TTL DNS, with responses dependent on the querying node's identity, and lightweight flows established to build a network model.

We define three types of Signpost devices: *(i) Signpost-enabled* devices that can be fully software patched and so use the architecture to its fullest while remaining backwards-compatible with existing applications; *(ii) Signpost-aware* devices that cannot be fully patched but can host applications that take advantage of local Signpost-enabled devices to route via them; and *(iii) Signpost-controllers*, well connected instances in the network core, e.g., on Amazon EC2, bridging the control channel between devices and hiding device locations from other users.

We will next describe the name hierarchy (§3.1), followed by the connectivity (§3.2) and control (§3.3) functionality, and the API (§3.4).

3.1 Naming

The majority of Internet-connected devices are nameless from a network perspective by either residing behind a NAT or with only an assigned name that map to transient addresses, e.g. via DHCP. The first step to unlocking global connectivity is assigning devices stable names in the global naming hierarchy, and providing an authenticated mechanism to resolve these device names to concrete network addresses. The Domain Name System offers a widely deployed, available and scalable naming service. It is available in every Internet edge network and middleboxes have been designed around it. The second step is to use DNS to establish a delay-tolerant channel that conveys application connectivity intentions and provides an excellent point of integration between applications and the Signpost architecture.

Using DNS as a control channel requires careful thought about the interactions with the existing resolution infrastructure. We are confident this use will be acceptable as Content Delivery Networks (CDNs) have also used DNS to redirect requests to local caches for many years [28], and we simply extend this to represent users on the network instead of content providers. The incidence of low time-to-live DNS packets has also steadily increased for the same reason, and studies have shown that a further increase would not be harmful [4],

The other main deviation from conventional DNS is the need to identify the query's ultimate source, not just the resolver via which the last query was performed. On the server side, the Signpost architecture requires every user to own a DNS zone and to provide an authoritative Signpost-controller that serves responses for devices registered in this zone. On the client side we adopt a two-pronged strategy as DNS clients can perform *iterative* or *recursive* resolution, and may not be Signpost-aware:

When **Signpost-aware** clients communicate with a Signpost-controller, they use iterative resolution to connect directly to the origin server. DNSSEC extensions introduced public key cryptography primitives: `RRSIG` and `SIG(0)` records [13] contain signed information, while `DNSKEY` and `DS` records establish a global, public and authenticated key infrastructure. The Signpost DNS client signs iterative requests with a `SIG(0)` record identifying the origin, and replies with an `RRSIG` record. If iterative resolution is not possible, e.g., due to a firewall blocking outbound UDP, a lightweight DNS

tunnel is established [1] that encodes the signatures in the query.[3] This initial query yields a DNSSEC-authenticated DNSCurve nameserver to which all future name resolutions are encrypted and authenticated.

Once a request has been authenticated, Signpost enhances the default name lookup process via an *effectful-naming* mechanism. An authenticated name resolution request triggers a *tactic engine* which probes the network to establish a path between the devices, or (as a low-latency first resort) routes them via a core-network tunnel. This network probing, described in more detail later (§3.2), requires a control channel between participating devices. This control channel is a low overhead signalling mechanism that lets devices negotiate connection parameters and potentially reconfigure their edge networks via, e.g., UPnP or NAT punching. This channel is established via the Signpost-controller, via either SSL (if outgoing TCP works) or a DNS tunnel. Signpost uses DNS tunnelling only when connectivity is limited, and its resource impact is limited to a small amount of control traffic between the authoritative server of the user and the local resolver of the device.

Connecting with a **Signpost-unaware** client via a Signpost-controller is currently inherently less secure, though Signpost does attempt to reduce the window of vulnerability. The controller, e.g., Alice, publishes a time-limited DNS capability in the form of a large, secret, one-time-use subdomain. This is shared out-of-band with the user of the Signpost-unaware client, e.g., Bob. Then, when Bob wishes to communicate with Alice, he attempts to resolve the secret subdomain, initiating setup of a secure channel by Alice's *tactic engine* (see below). Assuming some channel can be constructed, its endpoint IP address is returned to Bob's device as the result of the DNS resolution of the secret subdomain. Subsequent communication is then as private as that channel allows. Alice's Signpost DNS server will fail any subsequent attempts to resolve this subdomain, as well as being alerted to potentially nefarious activity.

3.2 Connectivity

Establishing a stable channel between devices on different edge networks is a non-trivial task. Middlebox interference and NATs means all of the network limitations must be dynamically probed. The simplest solution – a tunnel routed via the cloud– is clearly not the optimal solution if devices are co-located on the same edge network or are in physical proximity. Signpost servers solve this discovery problem in a similar manner to public CDNs. The first time a client query is received, the first-available solution – typically a tunnel – is returned in the DNS reply. Meanwhile, the server and clients probe for alternative channels, to be included in any subsequent replies to the same client. Since all replies are given a zero TTL, subsequent queries are very likely.

[3]This would be simplified by permitting `SIG(0)` RRs in DNS queries, a feature we are preparing as an IETF draft proposal.

Tactics	*Purpose*
STUN/TURN, NAT Punch, UPnP	NAT traversal
Tor, I2P	Anonymity
iodine	Tunnelling through DNS
TUNS, SSH, IPsec, SSL, TLS	Encryption & authentication
OpenVPN, L2TP	Indirect encrypted tunnel
Privoxy	Web proxy
Multipath TCP	Multipath support

Table 1: Tactics to connect two devices.

We introduce two notions to build this: *tactics* and the *tactics engine*. The tactic abstraction encapsulates the logic a specific connection mechanism requires to probe, connect, forward and detect disconnection. A short list of tactics that can be used to establish connectivity in real networks are presented in Table 1.

The tactic engine abstracts the logic to establish a communication channel between two devices, using available tactics, and runs on all Signpost-enabled and Signpost-controller devices. Its role is to execute a range of tactics in parallel, to try to create a communication channel with specific security properties. Additionally, if an active channel is disconnected, the tactic engine is responsible for re-establishing connectivity, using the same or different set of tactics. Establishing connectivity through the tactic engine is fundamentally an optimisation problem, where constraints are defined by the policy and the objective function is defined by the implementation, e.g., tactic selection maximises network throughput or minimises path establishment latency.

Applications running on Signpost-aware devices can adapt their behaviour to make optimal use of the system using a library with an API similar to the POSIX resolver API. The Signpost-aware device should be co-located in the same network with a Signpost-enabled device and be able to receive multicast DNS-SD [8] notifications. The library uses the DNS-SD mechanism to discover nearby Signpost devices and redirect traffic to them.

3.3 Control

Empowering users with control in Signpost requires two key functions: policy expression and authentication. User policy is defined through simple local configuration, where users define network path security properties on a per-domain basis. The tactics engine ensures policy enforcement during path establishment at runtime. User authentication employs a public key cryptography

scheme and takes advantage of the DNSSEC key distribution mechanism to bootstrap private DNSCurve channels. For each Signpost, at least one DNSKEY RR is available in the global DNS graph, verifiable by any host via DNS tree traversal or preconfigured trust anchors.

In addition, Signpost defines a key hierarchy for timely control and revocation of device trust. At the top level of the key hierarchy, we require a Key Signing Key (KSK) for each user. This key is added in the existing DNSSEC key infrastructure through a DS RR hosted on the authoritative server of the parent domain. The KSK is updated very infrequently and is used solely to express trust on device security keys. Further, the top-level domain of a Signpost user must also define a Zone Signing Key (ZSK) which is signed by the KSK and authenticates device DNSKEY RR records. In addition, each Signpost device requires a Device Signing Key (DSK) which is used by the device to authenticate the control channel and bootstrap authenticating tactics. Using this hierarchy of keys and a zero TTL value for RR records, users can easily revoke their trust for a compromised device by removing DNSKEY records from the authoritative server.

The authentication mechanism assumes that the client device is already a member of the user's cloud of devices. We can introduce new devices using the resurrecting-duckling protocol [30]. To register a new device, a short-lived passphrase is generated by a Signpost-enabled device and is entered by the user into the new device. A keypair is then generated on the new device and the passphrase is encrypted with the new device's secret key. The public key and encrypted passphrase are then sent to the sponsoring device for registration.

3.4 Backward compatibility

Signpost provides full backward compatibility to existing applications in Signpost-enabled devices. The dæmon exposes Signpost connectivity on the network layer of the OS as a local subnet. OpenFlow [24] functionality on end-hosts exposes network level programmability to Signpost-enabled devices and Signpost-controllers. By default, the dæmon forwards packets as normal to the local network interfaces, while for Signpost flows, the dæmon delegates control to the active tactic. Active tactics use the OpenFlow protocol to intercept, inject and control traffic, and can thus redirect flows to fit the network abstraction of the underlying connection mechanism. The dæmon exposes a local DNS resolver that intercepts all application resolution requests, and uses this to hook into subsequent traffic by returning a temporary loopback IP address for requests and using it as a cookie to associate traffic with that name lookup.

To provide some concrete examples of our architecture, let's consider Alice and the example given in §2.1. Alice has a Signpost-controller running in a globally visible location in the cloud with a stable public IP address. The controller is an authoritative name server for the domain `alice.io` and has a chain of trust with DNSSEC through to the root node. The server will present a different view of the network, based on the requester's identity.

With respect to the example of Bob and Charlie in §2.2, Charlie binds his desktop to `desktop.charlie.io` and Bob binds his laptop to `laptop.bob.io`. Using Signpost, Charlie and Bob can communicate as follows:

1. An application on Charlie's desktop does a local DNS query for `desktop.bob.io`, which is caught by the device's Signpost DNS resolver.
2. The Signpost DNS resolver signs the query and initiates the resolution.
3. The query is received by Bob's Signpost-controller as the authoritative server for `bob.io`.
4. Bob's Signpost-controller authenticates the source of the query and checks the connection policy for the specific domain.
5. Bob's Signpost-controller establishes a control channel with Charlie's Signpost-controller and negotiates the forwarding policy.
6. Both Signpost-controllers use their tactic engines and begin to create tunnels between themselves and with the respective devices.
7. Once a path is established, Bob's Signpost-controller responds to Charlie's desktop with an IP which is configured, through the OpenFlow service, to traverse bi-directionally from Charlie's desktop, to Charlie's Signpost-controller over to Bob's Signpost-controller and in the end to Bob's laptop, using the respective active tactics.
8. The application on Charlie's desktop can now send packets to Bob's laptop, and as the hosts move, the tactics engine will reconfigure the end-to-end path, while preserving the abstraction to applications.

This example describes a scenario where connectivity is established through the Signpost-controllers. Other scenarios are possible, notably when Bob himself does not have a Signpost-controller), but are elided for space.

4 Related Work

Building these personal device clouds ultimately relies on establishing mutual trust. Signpost authenticates hosts using the Public Key Infrastructure (PKI) of DNSSEC, although alternative PKI systems have been suggested due to DNSSECs reliance on single trusted entities. Perspectives [32] authenticates a host's public key by observing it from a range of network vantage points,

and the Unmanaged Internet Architecture (UIA) [15] similarly authenticates hosts by consensus. A common theme is the decoupling of host location from host identity, and ILNP [5] replaces IP addresses with separate locator and identifiers, allowing for improved mobility. Delegation-Oriented Architecture [31] uses flat endpoint identifiers like HIP [26] and UIP [14], unlike the user-friendly DNS names used in UIA and Signpost.

DNS currently accommodates mobile hosts through Dynamic DNS. Online services allow individuals to register DNS names for their devices and dynamically update the mappings to IP addresses. The popularity of these services highlights the demand for DNS names for an individual's devices but they have proven woefully insufficient for private devices due to the lack of access control for DNS resolution [17]. UIA uses DNS names like Signpost, e.g., `phone.alice` and `laptop.bob`, though UIA doesn't use the DNS. Instead, names are mapped to unique endpoint identifiers using gossip protocols through device clouds. In UIA, connection persistence is achieved by having each node maintaining connections at all times with a set of peers, while Signpost takes the approach of only creating connections as a side-effect of the DNS query. Like Signpost, Hamachi [21] aims to provide end-to-end connections via VPNs but Signpost provides a wide range of tactics supporting different properties according to the needs of the communication.

5 Challenges & Open Issues

This paper describes an early prototype; we expect to continue to evolve the architecture as we gain deployment experience. Our extensive use of DNS is not an accident. Unlike most other Internet protocols, the middlebox architecture on the edge networks does not hinder DNS, so it is (almost) always available. The DNS(SEC) naming hierarchy difficult for a single political force to surreptitiously or selectively disrupt.

DNSSEC provides authenticated DNS responses, integrity and authenticated denial of existence. However, it doesn't provide availability and confidentiality, and user identity could be spoofed if the trust anchors are compromised. Signposts could use alternate name resolution methods [32] as they get more widely deployed.

DNSCurve uses elliptic-curve cryptography to provide query and response confidentiality in streamlined mode and payload confidentiality in `TXT` mode. Large-scale studies have shown that unknown DNS RRs are often blocked by resolvers [19], and brand new protocols fare even worse in the edge network [18]. Improving the edge's name resolution capabilities via Signpost's tactics engine is key to navigating this middlebox-plagued future.

Neither DNSSEC nor DNSCurve address anonymity directly, so the initial name lookup for a service is not currently anonymous, although the reply and subsequent data channels set up through Tor are anonymous. We are currently designing schemes to improve the anonymity properties of the initial name lookups as future work.

Backwards compatibility with existing devices is essential to the successful deployment of Signpost, particularly in a world where devices are becoming more locked down and providing ever more limited APIs to applications. Our initial use-case for Signpost is for a single user to gather their personal devices under one logical namespace, and not require anyone else to participate. The subsequent network effect arises from multiple users wishing to communicate securely, and thus selectively and transiently connecting their clouds. As the number of such connections grows, when coupled with obfuscation features such as use of aliases, it should enable us to provide effective anonymity in scenarios such as §2.2 by using Signpost servers as a mix-zone.

The biggest challenge is perhaps that of deployment and ease-of-use. Moving from centralised services requires non-technical users to have an infrastructure that is reliable and resilient to attack. We are using library OSes [22] and the recent "Raspberry Spring" of embedded ARM devices to make our release as easy-to-assemble as possible, but only a full deployment later this year will allow us to gather data to demonstrate this.

The open-source community has released some excellent tools to enable the creation of end-to-end tunnels, enabling authentication, encryption, mobility or anonymity in the face of censorship and surveillance. Signposts bridge the gap between development and deployment by using existing DNS infrastructure as a signalling channel, giving us a head start in the race against censorship.

Signpost is not just a point-solution for combating censorship; it is a framework that can encapsulate *all* current and future solutions. By explicitly naming users in this system, we add a security and usability link absent from previous attempts in this space. We are currently converting our prototype implementation into a distributable open-source package for release later in 2013. We are also making tactics scriptable so that the latest connectivity techniques can be integrated into tactics engine. This will allow ordinary users to overcome the currently insurmountable usability problems of keeping up-to-date with the latest advances in securing online communication.

Acknowledgments. This work was supported by Horizon Digital Economy Research, RCUK grant EP/G065802/1. We would like to thank Tim Harris, Malte Schwarzkopf, Derek McAuley, Marcel Waldvogel, Alan Mycroft and Steven Hand and our shepherd Nikita Borisov for feedback on earlier drafts.

References

[1] Iodine. http://code.kryo.se/iodine/.

[2] ARENDS, R., AUSTEIN, R., LARSON, M., MASSEY, D., AND ROSE, S. DNS Security Introduction and Requirements. RFC 4033, IETF, Mar. 2005.

[3] BERNSTEIN, D. J. Dnscurve: Usable security for dns. http://dnscurve.org/.

[4] BHATTI, S., AND ATKINSON, R. Reducing DNS caching. In *IEEE Computer Communications Workshops (INFOCOM WKSHPS)* (2011).

[5] BHATTI, S., AND ATKINSON, R. Identifier-Locator Network Protocol (ILNP) Architectural Description. RFC 6740, IETF, Nov. 2012.

[6] BLUMENTHAL, M. S., AND CLARK, D. D. Rethinking the design of the Internet: the end-to-end arguments vs. the brave new world. *ACM Transactions on Internet Technology* (Aug. 2001).

[7] CARPENTER, B., AND BRIM, S. Middleboxes: Taxonomy and Issues. RFC 3234, IETF, Feb. 2002.

[8] CHESHIRE, S., AND KROCHMAL, M. Dns-based service discovery. RFC 6763, IETF, Feb. 2013.

[9] CUEVAS, R., KRYCZKA, M., CUEVAS, A., KAUNE, S., GUERRERO, C., AND REJAIE, R. Is content publishing in bittorrent altruistic or profit-driven? Co-NEXT '10, ACM.

[10] DAINOTTI, A., SQUARCELLA, C., ABEN, E., CLAFFY, K. C., CHIESA, M., RUSSO, M., AND PESCAPÉ, A. Analysis of country-wide internet outages caused by censorship. IMC '11, ACM.

[11] DECLAN MCCULLAGH, C. N. Dropbox confirms security glitch–no password required. http://cnet.co/kvVbpz.

[12] DISCHINGER, M., MISLOVE, A., HAEBERLEN, A., AND GUMMADI, K. P. Detecting bittorrent blocking. IMC '08, ACM.

[13] EASTLAKE 3RD, D. DNS Request and Transaction Signatures (SIG(0)s). RFC 2931, IETF, Sept. 2000.

[14] FORD, B. Unmanaged Internet Protocol: taming the edge network management crisis. *SIGCOMM Comput. Commun. Rev. 34*, 1 (Jan. 2004).

[15] FORD, B., STRAUSS, J., LESNIEWSKI-LAAS, C., RHEA, S., KAASHOEK, F., AND MORRIS, R. Persistent personal names for globally connected mobile devices. OSDI '06, USENIX Association.

[16] GUARDIAN. Pirate bay blockade begins with virgin media. http://gu.com/p/37acx/tw.

[17] GUHA, S., AND FRANCIS, P. Identity trail: Covert surveillance using dns. In *Privacy Enhancing Technologies* (2007), Springer.

[18] HONDA, M., NISHIDA, Y., RAICIU, C., GREENHALGH, A., HANDLEY, M., AND TOKUDA, H. Is it still possible to extend TCP? IMC '11, ACM.

[19] KREIBICH, C., WEAVER, N., NECHAEV, B., AND PAXSON, V. Netalyzr: illuminating the edge network. In *Proceedings of the 10th annual conference on Internet measurement* (2010), IMC '10, ACM, pp. 246–259.

[20] LEE, L. T. USA patriot act and telecommunications: Privacy under attack, the. *Rutgers Computer & Tech. LJ 29* (2003), 371.

[21] LOGMEIN. Virtual Networking with LogMeIn Hamachi. https://secure.logmein.com/products/hamachi/.

[22] MADHAVAPEDDY, A., MORTIER, R., ROTSOS, C., SCOTT, D., SINGH, B., GAZAGNAIRE, T., SMITH, S., HAND, S., AND CROWCROFT, J. Unikernels: library operating systems for the cloud. ASPLOS '13, ACM.

[23] MCKEON, M. The evolution of privacy on facebook. http://mattmckeon.com/facebook-privacy/.

[24] MCKEOWN, N., ANDERSON, T., BALAKRISHNAN, H., PARULKAR, G., PETERSON, L., REXFORD, J., SHENKER, S., AND TURNER, J. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review 38*, 2 (2008).

[25] MILLER, J. L., AND CROWCROFT, J. The near-term feasibility of P2P MMOG's. NetGames '10, IEEE.

[26] MOSKOWITZ, R., NIKANDER, P., JOKELA, P., ED., AND HENDERSON, T. Host Identity Protocol. RFC 5201, IETF, Apr. 2008.

[27] NETCRAFT. Netcraft monitors wikileaks and operation payback targets. http://url.anonfiles.org/bp.

[28] NYGREN, E., SITARAMAN, R. K., AND SUN, J. The Akamai network: a platform for high-performance internet applications. *SIGOPS Oper. Syst. Rev. 44* (August 2010).

[29] REUTERS.COM. India demands full blackberry access. http://reut.rs/lQkPca.

[30] STAJANO, F. The resurrecting duckling. In *Security Protocols* (2000), Springer, pp. 183–194.

[31] WALFISH, M., STRIBLING, J., KROHN, M., BALAKRISHNAN, H., MORRIS, R., AND SHENKER, S. Middleboxes no longer considered harmful. OSDI'04, USENIX Association.

[32] WENDLANDT, D., ANDERSEN, D. G., AND PERRIG, A. Perspectives: improving ssh-style host authentication with multi-path probing. ATC'08, USENIX Association.

[33] YU, B., AND WANG, R. Research of access control list in enterprise network management. In *Informatics and Management Science VI*, W. Du, Ed., vol. 209 of *Lecture Notes in Electrical Engineering*. Springer London, 2013.