Computer Networks 54 (2010) 1926–1938

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

SocialVPN: Enabling wide-area collaboration with integrated social and overlay networks

Pierre St. Juste *, David Wolinsky, P. Oscar Boykin, Michael J. Covington, Renato J. Figueiredo

Advanced Computing and Information Systems Lab, University of Florida, P.O. Box 116200, Gainesville, FL 32611-6200, United States

ARTICLE INFO

Article history:
Available online 4 January 2010

Keywords:
Socialvpn
P2P networks
Overlay networks
Collaborative systems

ABSTRACT

Trusted collaborative systems require peers to be able to communicate over private, authenticated end-to-end channels. Network-layer approaches such as Virtual Private Networks (VPNs) exist, but require considerable setup and management which hinder the establishment of ad-hoc collaborative environments: trust needs to be established, cryptographic keys need to be exchanged, and private network tunnels need to be created and maintained among end users. In this paper, we propose a novel system architecture which leverages existing social infrastructures to enable ad-hoc VPNs which are self-configuring, self-managing, yet maintain security amongst trusted and untrusted third parties. The key principles of our approach are: (1) self-configuring virtual network overlays enable seamless bi-directional IP-layer connectivity to socially connected parties; (2) online social networking relationships facilitate the establishment of trust relationships among peers; and (3) both centralized and decentralized databases of social network relationships can be securely integrated into existing public-key cryptography (PKI) implementations to authenticate and encrypt end-to-end traffic flows. The main contribution of this paper is a new peer-to-peer overlay architecture that securely and autonomously creates VPN tunnels connecting social peers, where online identities and social networking relationships may be obtained from centralized infrastructures, or managed in a decentralized fashion by the peers themselves. This paper also reports on the design and performance of a prototype implementation that embodies the SocialVPN architecture. The SocialVPN router builds upon IP-over-P2P (IPOP) virtual networks and a PKI-based tunneling infrastructure, which integrates with both centralized and decentralized social networking systems including Facebook, the Drupal open-source content management system, and emailing systems with PGP support. We demonstrate our prototype's ability to support existing, unmodified TCP/IP applications while transparently dealing with user connectivity behind Network Address Translators (NATs). We also present qualitative and quantitative analyses of functionality and performance based on wide-area network experiments using PlanetLab and Amazon EC2.

1. Introduction

The Web 2.0 era has greatly facilitated collaboration on the Web as seen by the popularity of Wikis, blogs, and social networks. Internet users can effortlessly disseminate information to their peers with status updates through Twitter, share photos and videos through Facebook, or blog their political views on Blogspot. The ubiquitous adoption of the social web has become an attractive platform for bootstrapping collaborative systems; online social networks, in particular, along with their open development APIs (e.g. the OpenSocial API [1]) have allowed for the implementation of inherently

* Corresponding author. Tel.: +1 352 871 1679; fax: +1 352 392 5040.
E-mail addresses: ptony82@ufl.edu (P.St. Juste), davidiw@ufl.edu (D. Wolinsky), boykin@acis.ufl.edu (P. Oscar Boykin), research@michaelcovington.com (M.J. Covington), renato@acis.ufl.edu (R.J. Figueiredo).
URL: http://MichaelCovington.com (M.J. Covington).

1389-1286/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2009.11.019

more social and user-friendly systems. Hence, social networks are being integrated in a variety of collaborative systems such as distributed file storage with friends [2] or enhanced social web searches [3].

Despite the widespread usage of social networks, direct social connectivity, in other words, private peer-to-peer (P2P) connectivity between friends, is still a major hurdle for these collaborative systems due to the heterogeneous structure and constraints of the Internet such as limited IPv4 address space, Network Address Translators (NATs) [4] and private network configurations. Furthermore, most of these collaborative applications require the direct and constant involvement of a central administration to authenticate, control, secure and mediate interactions amongst peers. These centralized architectures necessitate complex support and management to meet continuous demands from its users, as well as significant infrastructure investments in order to robustly handle thousands and potentially millions of concurrent users. Peer-to-peer (P2P) systems, on the other hand, are architected to achieve scalability and availability in a distributed fashion without relying on centralized servers. However, they lack a comparable framework for authentication, access control, and security which are commonly available in centralized infrastructures. We thus advocate an approach where social networking infrastructures, are utilized to bootstrap secure social connections over P2P overlay networks. The synergy of these two models produces a scalable, secure and reliable system capable of supporting larger numbers of users with significantly less infrastructure support and management complexity. This combined system can be perceived in two ways: either as enhancing social networking capabilities with P2P connectivity or evolving P2P overlays into secure, social networking platforms.

In this paper, we present the concept of a social virtual private network (SocialVPN), an approach aimed at bridging the gap between social and overlay networking. At the heart of SocialVPN lies the ability to automatically establish *direct peer-to-peer Layer 3 network links* as a result of connections or friendships established through social networking infrastructures. In the context of this paper, we concern ourselves with a social networking infrastructure as a system which allows for the discovery of peers and the binding cryptographic public certificates (e.g. X.509 certificates or X.509 certificate fingerprints) to these identities. Hence, a social networking infrastructure can range from a full-fledged online social network such as Facebook to an encrypted Google talk chat session, and even a PGP-signed email exchanges amongst peers.

Assuming the aforementioned infrastructures are trusted, users can seamlessly leverage social networking relationships to establish private network-layer channels. In this context, social networking is key to enable a decentralized system where users are able to maintain their individual trust relationships with friendly interfaces. In this case, P2P overlay networking becomes the essential messaging substrate enabling the formation and maintenance of direct private tunnel without any centralized backend. SocialVPNs allow users to communicate securely using existing TCP/IP applications such as desktop sharing (e.g. VNC and RDP), shared file folders (e.g. Samba and NFS), audio/video-conferencing, multi-user games, and so on even in the presence of NATs and firewalls and without modification, a feature which is not currently supported by social networking infrastructures.

Towards this goal, we describe an overlay architecture which is novel in the following respects: (1) it enables automatic assignment and dynamic translation of virtual private IPv4 addresses and virtual DNS names to hosts in a non-intrusive manner which avoids conflicts with current network deployments and requires no user configuration; and (2) it supports automatic exchange and discovery of peer credentials (e.g. X.509 certificates) through multiple social networking infrastructures, allowing end-to-end authentication and encryption of all communication among trusted peers. In this approach, the only configuration required from users is the creation and management of *social connections*; the configuration and maintenance of *IP network connections* is self-managing and completely transparent to users. The SocialVPN connections are thus accomplished without burdening users with the complex, error-prone configuration typically required to bring up public key and network tunneling infrastructures in VPNs.

The initial concept for SocialVPN was presented in [5] which was based on virtual machine routers, IPSec and a peer-to-peer overlay. This paper describes an improved architecture of SocialVPN which does not depend on virtual machines or IPSec, and supports various certificate exchange models. Overall, we suggest a P2P architecture that securely and autonomously creates VPN tunnels connecting social peers, where online identities and social networking relationships may be obtained from centralized infrastructures, or managed in a decentralized fashion by the peers themselves. This SocialVPN architecture enables a communication overlay framework that facilitates the development and deployment of P2P collaborative systems without requiring modifications to existing applications.

We also describe and evaluate a prototype implementation based on the IPOP [6] virtual network, different social networking infrastructures (including the Facebook platform, a Web back-end based on the Drupal content management system, an PGP-signed email exchanges by peers), and an PKI-based IP packet encryption security systems using X.509 certificates. Experiments in both local- and wide-area networks are used to demonstrate the capabilities and measure the performance of these social IP links. The experiments are conducted in realistic, large-scale wide-area environments, including over 500 SocialVPN routers distributed across five continents over the PlanetLab infrastructure, and over 100 SocialVPN virtual endpoints deployed dynamically over the Amazon Elastic Cloud (EC2) infrastructure.

We measured the time and bandwidth overhead of creating and maintaining these IP network links as the number of social connections increases. Our results show that it took less than one second to connect to a peer over 90% of the time; that the bandwidth per node spent in overlay maintenance is of the order of 1 KB/s; and that the latency overhead (1 ms) and encrypted TCP/IP stream throughput (30 Mbit/s) of our user-level prototype provide acceptable performance levels for a variety of wide-area applications. We also quantify the resource requirements,

in terms of storage and HTTP requests, on the social networking backends to argue the advantage of P2P connectivity in social networks over central administration. Experiments show that SocialVPN supports common TCP/IP legacy software; such as SSH, VNC, RDP for remote access, VLC, and iTunes for media streaming, and NFS and SAMBA for remote file access.

The rest of this paper is organized as follows. We provide details of the key motivations for our approach in Section 2. Section 3 present background and related works that address P2P connectivity and social infrastructures. We elaborate on the details the SocialVPN architecture in Section 4. Section 5 describes implementation, experiments, and result analysis of our prototype. We conclude in Section 6 with a summary and avenues for future work.

2. Motivation

The value in our proposed architecture comes from how it addresses various fundamental issues that need to be handled when designing social and collaborative systems. Specifically, we have identified the following challenges that SocialVPNs can address: P2P connectivity in social networks, support for legacy applications in overlay networks, network connectivity constraints, IPv4 address space limitations, and peer discovery and key management in overlay networks.

2.1. P2P connectivity in social networks

Current online social networks do not provide decentralized communication amongst peers. As a result, Web-based social networks (WBSNs) aggregate user content in centrally-administered domains requiring users to relinquish control of potentially private data. Although this centralization facilitates peer discovery and content sharing, it precludes point-to-point application access such as multi-media streaming, interactive multiplayer games, and desktop sharing because users can only interact through the predefined constructs supported by the WBSNs. Furthermore, WBSNs require massive infrastructures that can store huge amounts of user content (e.g. hundreds of millions of photos and videos) and serve the high numbers of simultaneous user requests, further increasing the cost and maintenance of these collaborative systems [7]. The SocialVPNs' ability to securely connect peers through IP-layer network links in a seamless and automatic manner can enable new methods of collaboration and sharing as well as application communication currently not supported by social networks. Consequently, this peer connectivity drastically reduces the centralized infrastructure requirements since it no longer needs to mediate every user interaction; for example, peers can share files directly with one another with control over their content where no intermediate storage is needed on social networking backends. These statements are quantitatively verified in our evaluation sections.

2.2. Support for legacy applications in overlay networks

Overlay networks have been very successful in delivering content in a P2P fashion – such as Skype for VoIP or BitTorent for file sharing. However, overlay networks typically connect users at the application layer, also effectively precluding a variety of legacy applications from being used – for instance, one cannot stream iTunes music or play a multiplayer game through Skype. There is a large number of software packages based on the Internet Protocol (IP), through the use of the Berkeley sockets API, but how can existing and legacy software be used with a new networking paradigm, such as one based on social connections? By means of a virtual networking, it is possible to produce a virtual local area network enabling the reuse of legacy TCP/IP based software.

2.3. Network connectivity constraints

Internet users today are regularly behind network address translation (NAT) devices using dynamically assigned IP addresses. Even though software exists to enable file transfer, conferencing, and collaboration, these tools typically require dedicated central servers in order to handle cases where users are not directly addressable by one another. An overlay network with NAT traversal support with self-optimizing direct connections can drastically simplify the development and deployment of collaborative services and applications.

2.4. IPv4 address space limitations

Network virtualization can provide a basis for addressing the three problems mentioned above; however, virtual networks require unique IP addresses at each endpoint. Public IPv4 addresses are scarce and often not available to end users, and private IPv4 addresses are not sufficient to enable each user of a typical social network to obtain a unique virtual IP address. For instance, the number of Facebook users (currently over 200 million) is larger than the number of IP addresses available in the 10.0.0.0 class A private address space [7]. Despite the fact that the IPv4 address space support billions of unique addresses, most users will only require direct communication with a few tens or hundreds of users [7]. Furthermore, the structure of these implicit communication networks is highly clustered [8] based on the small world phenomenon. We leverage these facts about social communication patterns to enable a dynamic IP assignment and translation scheme which avoids IPv4 address space conflicts with existing network configurations while maintaining support for legacy TCP/IP applications (Section 4.4).

2.5. Peer discovery and key management in overlay networks

Efficient distributed peer discovery and P2P key management is an open problem in P2P systems. Various key management models exist; some based on a central key distribution center, other based on IP-multicast or distributed-hash-tables [9,10]. We present several key distribution schemes that SocialVPN supports from a centralized

PKI-based model where all binding security credentials (i.e. X.509 certificates) are retrieved from a single trusted source to a web of trust model when users can perform certificate exchanges over PGP-signed emails. From a usability standpoint, a centralized social backend greatly simplifies the peer discovery and key management process, but it is not a requirement for making secure IP level network connections on SocialVPN (Section 4.2).

We believe that our approach is the first to address these common barriers to user connectivity in P2P collaboration. The SocialVPN design seeks to provide practical solutions to enabling private communication among social peers over P2P overlays; the next section provides background on the three main aspects of the architecture.

3. Background

The SocialVPN architecture integrates overlay-based virtual private networks (VPNs) and online social networks (OSNs), and allows for centralized and decentralized key management approaches.

3.1. Virtual private networks

Virtual private networks have been the popular choice for enabling wide-area access to resource in private organizational networks. Popular software products such as OpenVPN [11] are great tools that provide private IP layer tunneling in wide-area communication among peers. This approach suffers from two major drawbacks: a single point of failure, and error-prone configuration. In this client-server model, all encrypted IP tunneling is conducted through a publicly addressed VPN server which makes this approach highly centralized. Also, the complexities of configuration and key distribution make this approach unappealing when forming ad-hoc virtual organizations for wide-area collaboration. Various virtual networking projects for grid and cluster computing environments exist (such as VINE [12], VNET [13], or VIOLIN [14]), but they utilize routing tables that are managed and not self-configuring, making it difficult to establish private connections among peers in an ad-hoc P2P fashion.

Recently, P2P VPNs such as Hamachi [15], N2N [16], or ELA [17] have become popular peer-to-peer alternatives to centralized VPNs. In Hamachi, backend STUNT-like servers are used to enable NAT traversal and establish direct peer-to-peer connections among users; these servers also generate session keys for encryption and administer group access control. Group access control in Hamachi is done through shared secret keys where individual users do not initially control who has access to their network. This approach differs from SocialVPN architecture in two ways: (1) SocialVPN NAT traversal is not centralized because it uses existing nodes in the overlay to perform UDP hole punching for direct peer-to-peer connectivity, and (2) nodes negotiate their own session keys and manage access to their network locally without a centralized backend. N2N, on the other hand, does not require a centralized backend but it provides layer 2 networking and uses a different peer-to-peer network than the ring-structured overlay in SocialVPN [18]. The N2N peer-to-peer network uses supernodes that act as relay nodes for edge nodes that cannot communicate directly and they also store edge node information. N2N also requires more configuration; for example, automatic DHCP configuration is not yet available and pre-shared keys are used for link encryption. SocialVPN requires virtually no configuration and provides automatic DHCP and DNS services. ELA is also a peer-to-peer VPN with DHCP support, however it uses a different, hierarchical P2P overlay and SocialVPN uses a flat ring-based P2P overlay. Also, these VPNs do not integrate with online social infrastructures which is one of the key strengths of SocialVPN. SocialVPN also uses a dynamic IP translation mechanism which requires no global knowledge and coordination among the nodes for IP allocation which is usually required to avoid IP collisions since all endpoints in the aforementioned VPNs commonly use the same IP address space.

3.2. Online social networks

Online social networks have gained popularity as a means of allowing users to interact and collaborate, and have gained interest from the research community as a framework that can help address a variety of systems problems. Researchers are now able to obtain representative datasets of social graphs of hundreds of millions of real Internet users from different social networks which provides unprecedented opportunities to study social networks and their application in collaborative systems. Works such of Mislove et. al. [8] have looked at measuring and analyzing online social networks, while some [19] have striven to understand the trends and usage patterns of OSN applications. In the context of privacy in online social networking, Carminati et al. [20,21] have proposed access control models and certificate-based privacy models to help protect user content on OSNs. Golbeck et al. [22] proposes algorithms to calculate trust values that can be integrated into social applications.

Most of these works have focused on WBSN in which most user social interactions occurs within a Web browser; in the context of this paper, an online social network is not limited to these WBSNs. We define an online social network as a system with the following properties: (1) discovery and maintenance of peer relationships, (2) binding data (or information) about a peer, (3) capability to privately share information with social peers. For example, an instant message service can be considered a centralized online social network because it provides these services in the following manner: users identify each other through unique buddy names, they can look up the peer profile for binding user data, and information can be exchanged privately through the chat window over encrypted communications channels. In this perspective, SocialVPN can be integrated with a trusted instant messaging service and secure IP connections can be bootstrapped through the service.

3.3. Key management

Cryptographic key management in peer-to-peer networks is a requirement for forming authenticated

end-to-end communication channels and access control [23,10]. Previous work on security frameworks for collaborative computing provides a usage-control model which incorporates a hybrid model based on attribute acquisition and event-updates to control decisions for resource access [24]. Domingo-Ferrer [25] proposes the use of public-key cryptography in social networks to reduce the overhead of managing private relationships which alleviate the requirements of the social networking infrastructure.

To the best of our knowledge, previous work has not studied the possibilities of using trusted social backend to simplify the peer-to-peer key management problem. In centralized online social networks, key management is simpler since all peers communicate through a common trusted entity, only the entity's public key needs to be distributed. Through the use of the server's X.509 certificate, the social backend authenticates peers through private SSL/TLS channels. Clients are then able to securely share private information through these authenticated social infrastructures. In the case of peer-to-peer communication, two approaches are possible. One approach is to exchange shared session secret keys through the centralized backend, but this requires constant communication with the backend to refresh session keys periodically. The other option is to use the secure backend channel for a one-time public key exchange (more specifically X.509 certificate) between the peers. Once the peers have obtained each others cryptographic public keys, they can directly negotiate and generate symmetric session keys for encrypted peer-to-peer communication. The latter approach is desirable from bandwidth and scalability standpoints, because it does not necessitate constant interaction with a centralized backend, and is the approach taken in the SocialVPN architecture.

4. The SocialVPN architecture

The SocialVPN architecture contains these five key components: (1) connection privacy through encrypted end-to-end authenticated channels, (2) peer discovery and certificate exchange through a trusted social backend, (3) direct peer connectivity and legacy application support through IP-over-P2P overlay networking, (4) dynamic IP allocation and translation which avoids network conflicts, and (5) unique fully-qualified domain name assignment to peer resources. We will describe each of these components and how they fit together in the SocialVPN design.

4.1. Establishing connection privacy

Connection privacy is a necessity for secure wide-area collaboration, and it is usually a feature missing in P2P overlay networks. While various options are possible to implement the security infrastructure for a SocialVPN, our approach is based on public-key cryptography which supports the public key infrastructure (PKI) model. In doing so, we can reuse existing tools for processing X.509 certificates, and we can seamlessly integrate with datagram security technology which is widely used such as IPSec or DTLS [26]. SocialVPN builds upon an IPSec-like datagram security model which works as follows: (1) Peers exchange X.509 certificates through a trusted medium, (2) the retrieved X.509 certificates serve as the trust anchors (list of trusted CA certificates) for bootstrapping secure connections, and (3) asymmetric public keys help generate symmetric session keys to encrypt the IP traffic between endpoints, using well-known protocols and implementations such as Diffie–Hellman key exchange and IPsec. This model thus allow for the creation of authenticated, private end-to-end tunnels which protects from third parties intrusion.

In order to bootstrap a secure SocialVPN connection, self-signed X.509 certificates are exchanged. Because these X.509 certificates are self-signed, they must be acquired through trusted means, because each peer ultimately becomes the certificate authority (CA) of their own certificate. In all PKI-based infrastructures, a basic requirement is that the CA certificates serve as the trust anchors and must be acquired securely. On first run, an X.509 certificate is generated containing a peer's security credentials such as name, email, country, organizational unit, organization, and the P2P address (in the SubjectAltName field). The P2P address is a unique 160-bit identifier uniformly assigned to each node on the peer-to-peer overlay; it forms the basis for the peer-to-peer structure and message routing. Peers are added to the SocialVPN system by retrieving their X.509 certificates from a trusted source. To create a social link, both peers need to add each other's certificate in their SocialVPN router, if *peer1* adds *peer2's* certificate, the secure network link will not form until *peer2* adds *peer1's* certificate. The reciprocity ensures both SocialVPN endpoints have acknowledged the social relationship by explicitly adding each certificate to their node; this process is automated when SocialVPN connects to an online social network. The information in the certificate, specifically email, P2P address and public key, are used by the SocialVPN router to create the IP-to-P2P address mapping as well as the DNS-to-IP mapping; meaning the X.509 certificate contains all of the credentials necessary to form the private P2P network link.

4.2. Peer discovery and certificate exchange

Peer discovery in SocialVPN is the process of obtaining a list of unique peer identifiers (e.g. email addresses) that represent social peers. Any system that can generate such a list can thus be considered a social networking infrastructure, provided that the users of the system trust the infrastructure. Once the social peers are identified, the X.509 certificates bound to the peer identities are obtained through a trusted backend, which may or may not be part of the same social networking infrastructure. In general, the SocialVPN model requires the following services from social networking infrastructures: (1) the ability to query for a list of peers, (2) the ability to retrieve binding information about the peer, and (3) and the ability to exchange public X.509 certificates among peers. These services could be provided by a single social networking backend or could be distributed over various decentralized components.

As previously mentioned, the SocialVPN security is based on the PKI model. PKI infrastructures are well-understood, and robust implementations are available; however, the management of keys is complex, error-prone and overwhelming to an end user who is not familiar with the security theory in which PKI is based. Here, the SocialVPN router, by querying an online social networking infrastructure, handles the management and distribution of X.509 certificates transparently in a secure manner. In the following sections, we describe three possible methods of how SocialVPN performs peer discovery and certificate exchange through three different online social networking infrastructures.

4.2.1. Centralized model: single server provides identity, relationships, and certificate data store

In this model, peer certificates are automatically exchanged through the trusted centralized social backend to form direct, private connections. The centralized model is the easiest to manage and design, and we will illustrate this with our Facebook prototype example. Facebook is a Web-based social network (WBSN), and it provides peers with the ability to create relationships, bind data to their identity through user profiles, and share trusted content with their peers. By our earlier definition, Facebook meets all the criteria of an online social network. Through the Facebook Platform API, peers are able to authenticate themselves, store the X.509 certificate on the Facebook datastore, and they are also able to retrieve X.509 certificates of their social peers as well (see Fig. 1, top left). As a result, the Facebook Platform along with the SocialVPN infrastructure are able to function as a PKI allowing for the trusted exchange of X.509 certificates. Once each SocialVPN endpoint is able to acquire each other's certificates, they are able to form a secure P2P link. The X.509 certificate contains two fields of binding information the peer identifier (i.e. email) and the P2P address for overlay routing.

4.2.2. Semi-centralized model: one server provides identities and relationships; distributed certificate data store

The semi-centralized model is similar to the centralized model except for certificate storage and the possibility of supporting multiple social networking infrastructure back-ends. The X.509 certificate's fingerprints are securely stored on the social networking back-end. Peers then share public security credentials (i.e. X.509 certificate) through the distributed-hash-table (DHT) available in P2P overlay (see Fig. 1, top right). Using the DHT for storage and only storing the fingerprints in the trusted backend allows for a more scalable design because less storage is required from the backend. Assuming the peer discovery and identity bound certificate fingerprints are obtained through trusted means, the certificate exchange can take place over the DHT as long as the public X.509 certificate integrity is confirmed. Once the certificates are safely acquired and trusted on both ends by verifying the fingerprints, they are utilized to form the secure IP tunnels between the two peers.

4.2.3. Decentralized model: decentralized identity, relationship and certificate data store

In the decentralized model, independent components are utilized together to serve as an online social networking infrastructure. For example, a list of contacts uniquely identified by email addresses; therefore, a trusted address book can provide peer discovery. We also utilize the concept of an identity provider, as any infrastructure which provides profile information based on a unique identifier; this serves to meet the second criterion. For instance, Gmail account users possess public profiles that users can update with general information about themselves. In the secure public profiles, users can publish their certificate fingerprints thus allowing other peers to obtain their trusted fingerprints for these identity providers. With the trusted fingerprints, the DHT certificates become verifiable allowing for the bootstrap of a secure connection.

The web of trust security model can also be integrated with SocialVPN to provide another example of a decentralized certificate exchange model. In this scenario, a PGP key is used to sign a self-signed X.509 certificate. The reason for the double signing is because the PKI-based model and PGP pages models are not compatible with each other. This hybrid approach allows us to take advantage of existing PKI based security infrastructures while allowing us to leverage the decentralized benefits of the web of trust PGP design. Once again, the sole requirement for creating SocialVPN IP links is the trusted exchange of X.509 certificates. Users can easily email each other their X.509 certificates, as long as the email is PGP-signed. If the receiver also has the sender's PGP key as part of their PGP keyring, they can easily verify the X.509 certificate integrity with PGP. Once the email has been verified, the user can input the trusted X.509 certificate to the SocialVPN system manually through the user interface. In this model, the user becomes the source for peer discovery, the trusted peer's PGP key contains the identity binding information, and the public key exchange is done through the email messaging system. As in the other two cases, once public key certificates are exchanged, the process of generating and exchanging symmetric keys for VPN tunneling is transparent and is accomplished by P2P messaging among the endpoints, e.g. to follow a protocol such as Diffie–Hellman. It is possible to envision this same model applied to an instant messaging scenario, where users can share each other's certificates through an encrypted chat session. The common theme here is the establishment of trusted, and not necessarily private, out-of-band communication channels for one-time exchange of binding security credentials.

4.3. Legacy application connectivity through IP-over-P2P overlay

The majority of networking applications are based on the TCP/IP protocol; hence, these applications cannot easily leverage P2P overlay networks for connectivity, since P2P libraries are usually incorporated at the application layer. Allowing unmodified applications network connectivity through a P2P overlay is a valuable feature that can expedite the design and deployment of wide-area collaborative systems. In this section, we describe how we move P2P

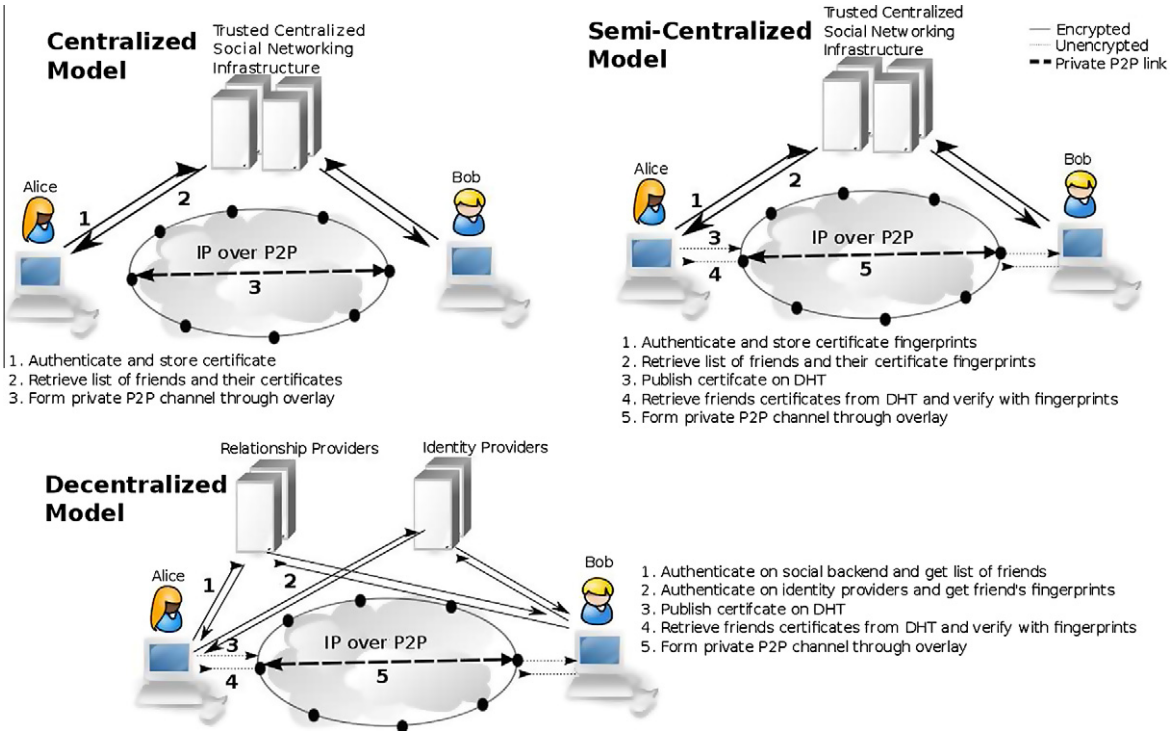


Fig. 1. Certificate exchange models. Top left: In the centralized model, the list of friends and their certificates are obtained from a single social networking backend. Top right: In the semi-centralized model, the list of friends and their certificate fingerprints are obtained from one or more centralized social networking backends; certificates are stored and retrieved from a distributed data store (DHT). Bottom: In the decentralized model, the list of friends can be obtained from multiple social backends; certificate fingerprints are retrieved from multiple identity providers, and certificates are exchanged over the DHT. Peers themselves can verify identities locally, e.g. following on a PGP-based web of trust model. n the last two cases, the certificate fingerprints are used to verify the integrity of DHT-acquired certificates.

overlay communication from the application layer to the network layer. While the IP-over-P2P layer in the SocialVPN architecture can conceivably be designed on top of various P2P substrates, our discussion to follow is based on the IPOP overlay. A capability of IPOP that is key in the SocialVPN context is decentralized NAT traversal; these and other aspects of the IPOP overlay are described in previous work [6,18].

Virtual network interfaces can be utilized to capture and inject IP packets from and to a host operating system kernel [27]. These captured packets are then tunneled as normal application P2P traffic through an optimized structured overlay [18]. On the sending side, the virtual IP routers are able to retrieve IP packets from legacy applications through the virtual network interface, and send these IP packets to the appropriate P2P node on the overlay. On the receiving end, the router receives an IP packet from the P2P network, and injects it back into the host operating system; thus enabling Layer 3 level communication between applications. As shown in Fig. 2, the SocialVPN virtual IP routers maintain a mapping of IP-to-P2P addresses where a P2P address is bound to a particular peer based on information obtained from the peers credentials' exchange (i.e. X.509 certificates containing P2P addresses). Therefore, the routers possess a list of P2P addresses representing social peers on the P2P network and will only route IP packets to or accept IP packets mapped to social peers' P2P addresses.

The structured P2P overlay network manages the connectivity between the peers through self-configuration and self-organization as nodes join and leave the P2P network, and employs decentralized NAT traversal techniques to connect nodes that are not directly addressable over the Internet [18,28]. When socially connected peers are identified on the overlay network, direct IP tunnels are formed and maintained to allow low latency IP communication amongst peers. In our analysis, we measure the cost of maintaining the social connections on the P2P overlay as the number of connections increase.

4.4. Dynamic IP allocation and translation

Deployments of SocialVPNs need to accommodate user bases that can be quite large – there are currently hundreds of millions of users registered with WBSNs. This presents a challenging problem because VPN endpoints require unique IP addresses and is thus subject to several constraints. IPv6 infrastructure and applications are not widespread; public IPv4 address spaces are scarce; private IPv4 addresses do not scale to large numbers, and can collide with local address spaces of users who are increasingly bound to private networks behind NAT devices. Here we describe an approach where, through address translation, a SocialVPN can scale to numbers of users larger than the limit imposed by IPv4 private address range while avoiding address space conflicts with end user networks.

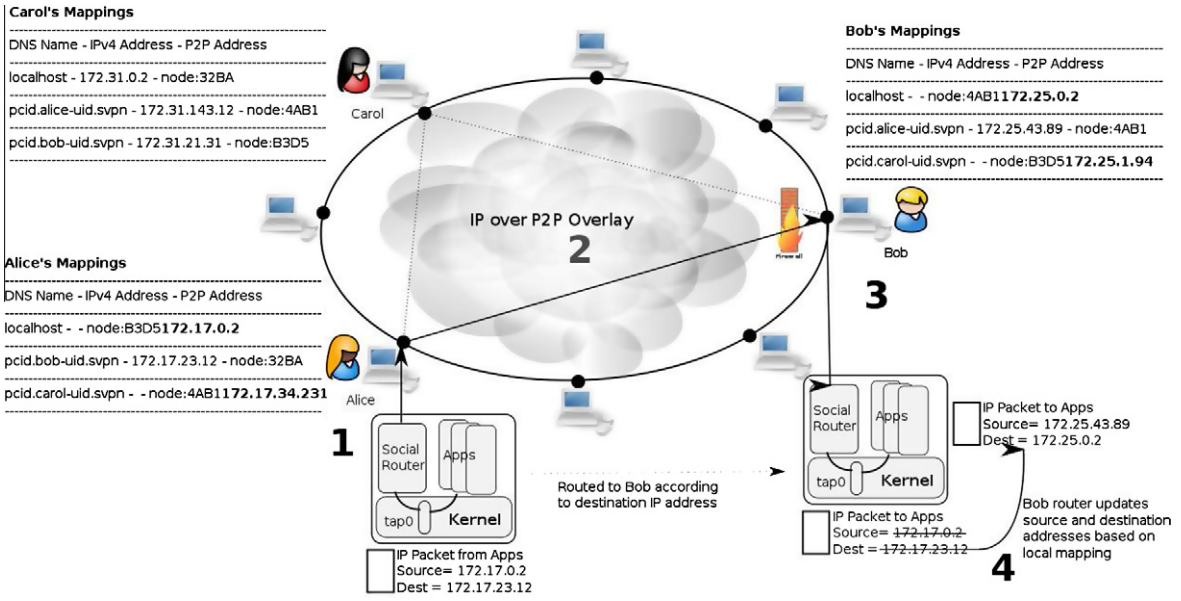


Fig. 2. SocialVPN architecture. After the peer discovery and certificate exchange through a social backend, peers form direct, encrypted channels where applications can communicate through TCP/IP. (1) Applications send IP packets to tap0 virtual NIC through the kernel and the user-level social router captures the IP packets. (2) Social router checks the destination IP which maps to friend Bob, encrypts the IP packet with the symmetric key (previously established for the IP-over-P2P tunnel after public key exchange) and sends the encrypted IP packet over the P2P tunnel through Bob's firewall. (3) Bob's social router receives the IP packet. It looks up in its local database information about the source (including Alice's symmetric key and virtual IP address); it then decrypts it, and updates the source and destination IP addresses according the Bob's local mapping. (4) Bob's router sends the translated IP packet to the applications through the kernel-based virtual NIC.

The key idea behind this approach exploits the fact that, while the total number of participants in a collaborative system such as an online social network can be very large (hundreds of millions of users), the number of relationships a *single user* has at any point in time is significantly smaller (typically hundreds to thousands). Nonetheless, while the number of relationships a user has is relatively small, it is larger than the number of network interfaces that operating systems typically are able to handle. Therefore, a solution that multiplexes social networking connections into a single virtual network interface with a single virtual IP address is desirable. Our approach accomplishes the goal of presenting a single IPv4 virtual network interface while avoiding address space collisions, as follows.

A SocialVPN maintains, at each user's endpoint, a private IP address space that is sized to accommodate the expected number of social connections a user may have. For instance, a 16-bit class B private address space supports tens of thousands of connections. This private address space is dynamically assigned locally by the SocialVPN router such that it avoids collision with any existing network interfaces of a SocialVPN user's machine. The following example (see Fig. 2) illustrates the address translation process.

For example, if a user has a physical network interface with an IP address 172.16.5.16 with a netmask 255.255.0.0, the virtual network interface used by the SocialVPN router would be automatically configured to use a non-conflicting IP address range such as 172.17.0.2 and peers would be allocated IP addresses in the 172.17.x.y range. Each peer is also assigned an IP address in the selected local address space, a mapping between IP address to peer's P2P addressed is maintained by the SocialVPN router. The SocialVPN router uses the IP-to-P2P mappings to route IP packets to the appropriate peers based on the destination IP (e.g. destination IP 172.17.34.231 maps to P2P address node:4AB1, so all IP packets with destination 172.17.34.231 go to peer with that node address, see Fig. 2).

Because of the dynamic IP allocation, IP packets need to be translated by the received SocialVPN router to match the receiver's IP-to-P2P mapping. Let's say Alice has a local virtual IP address of 172.17.0.2 and her friend Bob is dynamically assigned the IP address 172.17.23.12 by her local SocialVPN router. On Bob's local SocialVPN router, he has a local virtual IP address of 172.25.0.2 and Alice is dynamically assigned a virtual IP of 172.25.43.89. When Alice communicates with Bob over IP, Alice's SocialVPN router receives IP packets from the host OS with 172.17.0.2 as the source IP and 172.17.23.12 as the destination IP (Bob's address), and tunnels them directly to Bob over the P2P overlay. Bob's SocialVPN router hence receives the IP packets from the overlay and changes the source and destination IP addresses to the appropriate addresses assigned by Bob's SocialVPN router; meaning the source address changes from 172.17.0.2 to 172.25.43.89, and the destination address from 172.17.23.12 to 172.25.0.2 (see Fig. 2 step 4). Since only IP addresses are translated, all protocols above Layer 3 such as transport layer UDP and TCP ports information remain unchanged.

Due to this translation, IP addresses are not globally valid in the virtual network. For instance, Alice and Bob both

have a friend Carol, Alice assigns Carol an IP address of 172.17.34.231, but Bob gives Carol the 172.25.1.94 IP address. Since only Alice's SocialVPN router has the distinct local IP mapping for Carol, Bob's router could not resolve the 172.17.34.231 IP address to Carol's P2P address, in other words, if Alice tells Bob that he could ping Carol's machine at 172.17.34.231, Bob's ping messages would never reach Carol's machine. Although source and destination IP addresses are updated, UDP or TCP ports are not changed. This is similar to the network address translation performed by full cone NATs. Most client-server applications (e.g. Web browsing, file sharing, remote desktop) are able to work without changes with such kind of NAT. However, some protocols which exchange IP addresses in the payload of messages (e.g. FTP, SIP, mDNS) require packet inspection and IP translation in order to work correctly.

4.5. Global resource naming

As previously explained, the virtual IP addresses assigned to peers are only valid on the local peer's machine; hence peers cannot refer to each other by IP address. It is important from a usability and protocol design perspective to have a fully qualified domain name (FQDN) assigned to each peer. In our approach, the SocialVPN router generates globally unique DNS names based on each peer's unique identifier as shown in Fig. 2. A loopback DNS server, part of the SocialVPN router, resolves DNS requests for names in its domain to the proper IP address allocated to the peer. For instance, both Alice's and Bob's SocialVPN routers would assign Carol's machine a DNS name of pcid.carol-uid.svpn, Alice's DNS server would resolve pcid.carol-uid.svpn to 172.17.34.231, while Bob's DNS server would resolve the same DNS query to 172.25.1.94. Hence, both Alice and Bob would be able to ping Carol's machine using the same pcid.carol-uid.svpn DNS name.

5. Analysis

In this section we report on the functionality and performance of a SocialVPN router that implements the techniques described in the paper. To date, we have successfully integrated the SocialVPN router with the Facebook API, with an open-source Web back-end (Drupal), and with support for manual entry of self-signed certificates through the user interface. The latter approach supports users to copy/paste PGP-signed email messages containing SocialVPN certificates as the mechanism for discovery and public key exchange. The prototype router leverages the IPOP overlay, which supports both IP-over-P2P tunneling and a DHT. The user-level router is implemented in C# and works with the Mono and .NET runtimes, using the tap virtual network device. The prototype implements a user-level security stack on the overlay that is inspired by the IPsec protocol. A previous implementation that integrates with a kernel IPsec stack has been demonstrated in prior work. Deployments of the prototype on hundreds of wide-area PlanetLab nodes have been running continuously for months; in the experiments described below, we create a separate overlay on resources distributed across PlanetLab, Amazon EC2 and resources within our lab to assess the performance of the prototype and establish the feasibility of our approach quantitatively.

5.1. Connecting to multiple social infrastructures

Facebook exposes social services through the Facebook Platform API which uses a REST-like interface where all method calls are made through HTTP GET and POST requests. Through this API, Facebook supports the three main requirements of a social infrastructure expected by SocialVPN: (1) query a list of friends, (2) retrieve information bound to peer identity through the Facebook profile information, and (3) exchange X.509 certificates through the Facebook application datastore. In the case of the Drupal content management system, we used the services XMLRPC module along with the relationship module to enable the services necessary to function as an online social networking infrastructure. For the email scenario, we used Gmail and the FireGPG extension to Firefox, which enabled us to encrypt and sign email messages including attachments using PGP keys directly through the Firefox web browser. Hence, peers are able to email their X.509 certificates to each other and use PGP to sign and verify the X.509 certificates. Integration with these different backends serves to show that the SocialVPN can connect to multiple backends.

In our experiments, we are interested in determining the performance of the system when the DHT is employed as the certificate data store. In addition, we use a centralized social networking identity/relationship provider to facilitate the bootstrapping of a network during the experiments. The experimental setup is described in detail next.

5.2. Experiments

We conducted experiments to analyze some key metrics of our design: (1) the time to form the private links with the peers, (2) the bandwidth overhead cost of

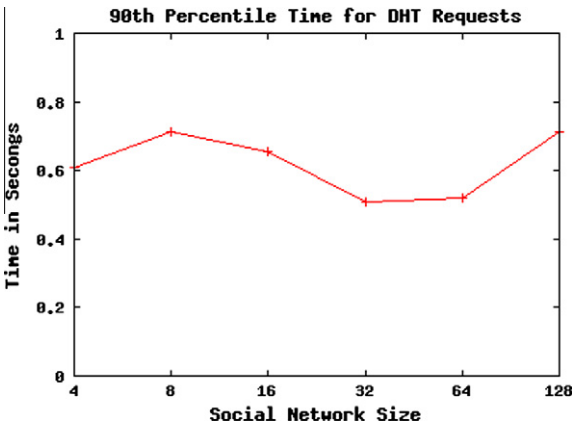


Fig. 3. DHT Retrieval Times for deployments of 16, 32, 64, and 128 SocialVPN nodes. For 16, 32 node deployments, 100% of DHT queries took less than 2 s. For 64-node deployment, 98% of 5311 requests took less than 2 s. For 128 node deployment, 94% of 18,782 requests took less than 2 s. F1 represents percentage which failed on first attempt, attempts succeeded after retries.

maintaining these private links on the P2P overlay, (3) the latency and bandwidth of the virtual private network, and (4) the server load on a centralized social backend. In our experimental setup, we utilized Amazon Elastic Cloud (EC2) to deploy virtual machines with one virtual core and 1.7 GB of RAM to experiment with varying social network sizes. These virtual machines are based on the Debian 5 Lenny Linux distribution running the 2.6.26 Linux kernel. The software is written in C#; hence, we used the Mono .NET Runtime version 1.9.1. Each virtual machine ran a SocialVPN node which connected to a pre-deployed P2P overlay network of 500 nodes running on PlanetLab [29]. We also deployed an initial 20 SocialVPN nodes in a computing cluster at the University of Florida (UF); these nodes represented peers that are already online when SocialVPN nodes join the network and also help against the clustering effect of Amazon EC2 nodes.

All of the SocialVPN nodes connected to the same social networking backend to obtain peer relationships and peer certificate fingerprints. We implemented a Django-based backend which provided the necessary services of a social networking infrastructure which was deployed on the Google App Engine Cloud Infrastructure [30]. The Google App Engine backend provided the tools to measure the number of HTTP requests made by the SocialVPN endpoints, as well as the storage and CPU requirements on the backend. Also in our experimental setup, we stored the X.509 certificate fingerprints on the online social networking backend, while the actual X.509 certificates were stored on the DHT to minimize the storage requirements on the backend.

5.2.1. Link creation time

We analyzed the time taken to form direct, private IP links once peers were discovered through the social networking backend. We examined two main steps: the X.509 certificate retrieval from the DHT, and the formation of the encrypted IP tunnels between peers, which includes the exchange of Diffie–Hellman messages over the P2P overlay to establish a pair of symmetric keys. Understanding the time taken to retrieve an X.509 certificate from the DHT is important to prove that a DHT is capable of supporting such a load with an acceptable retrieval time. The measurements taken were from deployments of 16, 32, 64, and 128 SocialVPN nodes deployed on Amazon EC2. These Amazon nodes were brought up simultaneously and ran for a 50-min period in each deployment scenario. In all cases, there was all-to-all social connectivity among the nodes; in other words, with 16 Amazon nodes, each node has 15 direct connections with the other Amazon nodes, plus the additional 20 direct connections with the nodes running at the UF cluster.

In Fig. 4, we show that as the number of peers increase it consistently takes less than 800 ms for 90% of the DHT requests, which is a reasonable timespan to obtain an X.509 certificate from a distributed datastore. Fig. 3 provides a histogram demonstrating the distribution of the certificate retrieval times from the DHT with the various network sizes. For network sizes of 16 and 32, all DHT retrieval times were below 1200 ms. A DHT request performed on our Chord-like structure overlay has a complexity of $log(N)$ hops where N is the number of nodes in the P2P network. Hence, with a P2P network size of

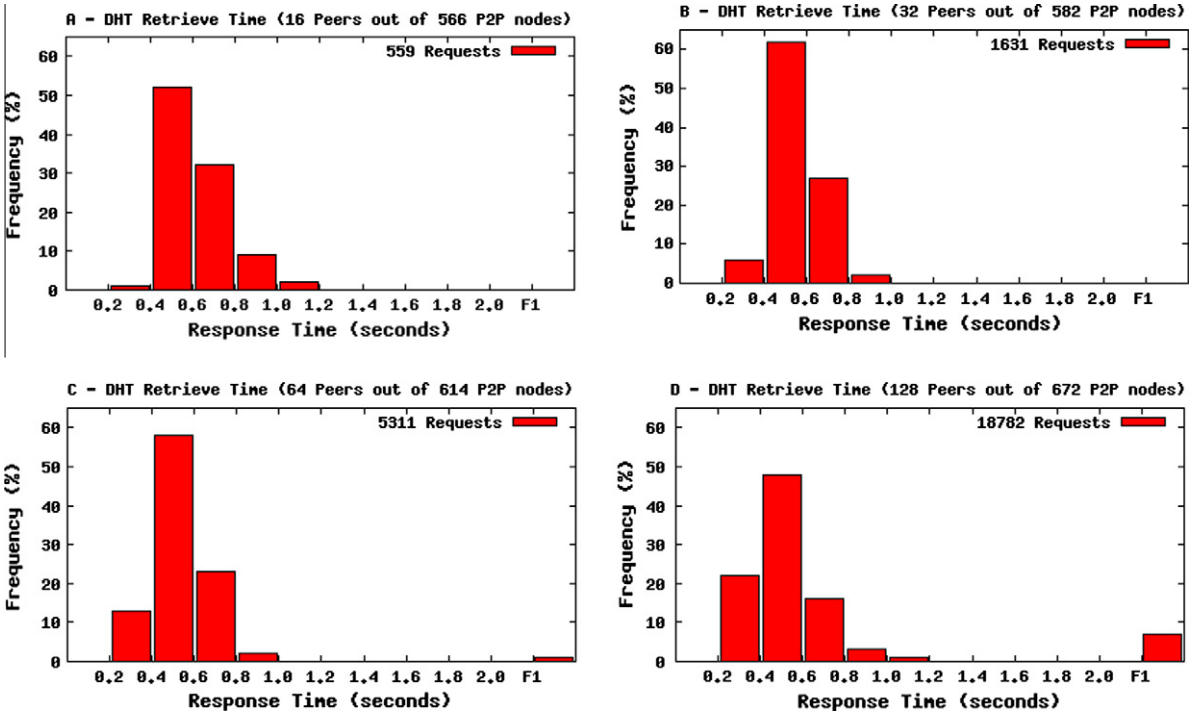


Fig. 4. 90th Percentile Time for deployments of 4–128 nodes. As network size increases, DHT lookup performance remains around 600 ms.

around 500–600 nodes, it takes an average of 9 hops to reach the node responsible for storing the ⟨*key*, *value*⟩ pair. Also, since our P2P network is deployed over PlanetLab which has globally dispersed nodes, it is not hard to see why a DHT request may take up to 1200 ms to return a value. In the 64 and 128 deployments, there was a 1% and 6% failure rates respectively for first time DHT requests caused by cases where nodes issue DHT lookups under ⟨*key*, *value*⟩ pairs that have not been successfully stored in the DHT. This happens because when 64 or 128 nodes join the P2P network simultaneously it takes longer for the P2P network to stabilize causing some DHT PUT request to initially fail; all DHT lookups eventually succeeded after subsequent retries.

Fig. 5 shows the connection times for network sizes 32, 64, and 128. In all cases, over 85% of the connections took less than 800 ms. The connection time involves peers creating direct paths to each other where NAT traversal is performed when necessary. With Amazon EC2 nodes, NAT traversal is not required between these nodes since they are on the same internal network. NAT traversal is necessary between the UF and Amazon nodes because the UF nodes are located behind a NAT. In each case, between 15% and 20% of the connections took over 1 s to connect due to the NAT traversal process between the UF and Amazon EC2 nodes.

5.2.2. Bandwidth cost

We also measured the bandwidth overhead of maintaining the social connections on the P2P overlay. The calculated bandwidth is the average number of bytes transferred per second across all the Amazon EC2 nodes for each 50-min deployment. Fig. 6 shows us that the bandwidth cost increases proportionally with the number of peers in the network starting at 0.2 KBytes per second to 1.2 KBytes per seconds from a network size of 4 friends to 128 friends. The P2P overlay uses bandwidth for maintenance such as routing table updates, probing, DHT operations, and nodes joining/leaving the overlay. Peers are proactively probed (every 15 s) to determine the status of the node. Hence, there is a small bandwidth overhead for maintaining the structured overlay and the direct social connections – a few percentage points of a typical broadband with access to hundreds of Kbit/s bandwidth.

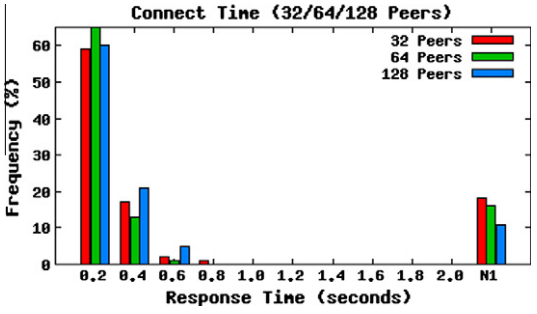


Fig. 5. Connect Time for deployments of 32, 64, and 128 SocialVPN nodes. Connection times remain stable as the network increases up to 128 nodes. N1 represents percentage with more complicated NAT environments, some nodes took up to several minutes before they could form direct connections.

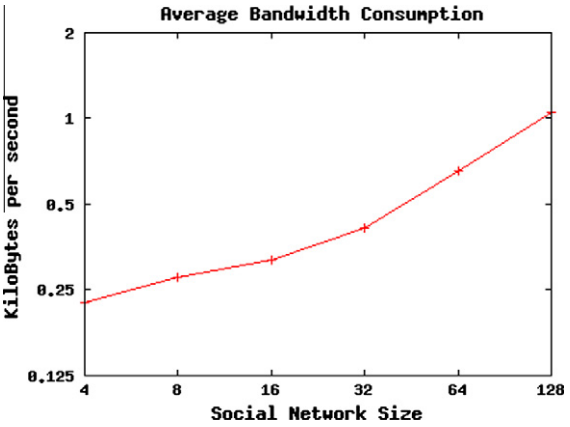


Fig. 6. Bandwidth cost as social network size increases from 4 nodes to 128 nodes. This is the bandwidth consumed by the SocialVPN router to maintain the structure the P2P network as well as the direct links between peers.

5.2.3. Application and network performance

We qualitatively tested several applications and quantitatively assessed the latency overhead and bandwidth achieved by our prototype. The following applications were successfully tested between SocialVPN nodes: VNC- and RDP-based shared remote desktop sessions, file sharing through Samba and NFS, multi-player 3D LAN game (Valve's CounterStrike), HTTP server with Apache, music sharing/streaming through iTunes, multicast-based service discovery (mDNS/SD) Bonjour [31], direct P2P chat with Pidgin over Bonjour, VoIP with Ekiga. We also measured the bandwidth and latency of our virtual network. We ran two SocialVPN nodes on the same cluster connected by a 1 GB Ethernet switch. We used the ping tool to measure the round-trip latencies of 100 ICMP request/reply packets and obtained an average latency of 1.1 ms. We also calculated the bandwidth with the Iperf network measurement tool by measuring the TCP throughput of a 30 s transfer and achieved a bandwidth of 30 Mbps. Overall, the observed performance is acceptable for wide-area environments using a collaborative system which is the target of our architecture. Nonetheless, performance improvements are actively being pursued.

5.2.4. Server load on backend

We argued that the hybrid SocialVPN approach leverages the benefits of P2P communication to alleviate the infrastructure demands of online social networks; therefore, we measured the number of HTTP requests, and bandwidth consumed by our experiments. The current SocialVPN system essentially makes three types of HTTP requests to the social networking backend: get friends, get fingerprints, and store fingerprint. These three HTTP requests are performed every five minutes, and when a new SocialVPN node joins the network, to synchronize the remaining nodes. According to the Google App Engine site monitoring tools taken over a 24-h period for 38 SocialVPN nodes, the server received a total of 44121 HTTP requests with a bandwidth cost of 300 MBytes. Each node made an average of 48 HTTP requests per hour, and consumed 293 KBytes of bandwidth per hour for communication with

the social network. We compared the cost of aggregating all the work on a single server. The per-node costs are trivial, but even for this small example, the server bandwidth costs would be non-negligible. With this low amount of resource usage on the social backend, peers are able to share files, securely chat, or stream media directly over the encrypted P2P virtual network. Providing these collaborative services through a conventional, centralized social network would require much more infrastructure than the demonstrated server load of SocialVPN.

6. Future work and conclusion

Social networking infrastructures can greatly facilitate the configuration and deployment of systems because they have proven quite effective in enabling users to discover and associate with their peers. This paper shows an architecture where social connections established through user-friendly Web-based infrastructures can effectively guide the creation of encrypted, authenticated connections at the computer network layer. This is achieved in a user-transparent manner and supporting a wealth of existing TCP/IP applications on top of existing networking infrastructure. To our knowledge, this is the first work to integrate social networks and wide-area peer-to-peer overlay networks. A prototype implementation of our approach and experiments with a deployment on PlanetLab and Amazon EC2 infrastructure has shown this approach to be feasible and promising.

In this work we described a decentralized method of implementing a self-configuring virtual private network which tremendously facilitates collaboration among peers over the Internet. Our approach combines various existing technologies such as social networking APIs, structured peer-to-peer overlays, and PKI certificate models to provide an easy-to-use, scalable, yet secure IP-level communication link among friends. By maintaining a structured peer-to-peer overlay as a messaging substrate, peers can initiate NAT traversal which enables direct IP traffic tunneling amongst two peers. The structured peer-to-peer overlay also provides a distributed datastore through a DHT; which in our case can be used for X.509 certificate publishing and retrieval. Our experiments show that our overlay can effectively support the certificate exchange operations necessary for the construction of encrypted IP tunnels where over 85% of the certificate requests took less than 800 ms in a overlay of over 500 nodes. The measured bandwidth overhead of maintaining the structured overlay as well as the direct peer-to-peer links (a total of over 150 connections) was under 3 KBytes/s. Finally, we described how various certificate exchange models from a full-fledged social networking backend such as Facebook, to the web of trust model of PGP can be integrated into the system. We have tested prototype implementations that used Facebook, a Drupal-based social networking backend, and a Google App Engine backend which all showed easy integration and minimal load on these various backends which further validates our claim of low resource requirements on a trusted backend.

The paper also presents several opportunities for research. In future work, we plan to investigate how information about social network links can improve overlay routing performance. In the SocialVPN context, there is a particular interest in optimizations that target communication patterns found in typical TCP/IP applications. For unicast applications such as client/server, social links may help reduce the latency in routing of connection setup messages needed for NAT traversal and guide the selection of overlay proxy nodes when NAT traversal is not possible. We believe support for multicast applications, in particular for resource discovery, can greatly enhance the functionality and usability of SocialVPNs. Multicast-based resource discovery implementations such as Bonjour [31] and UPnP, if supported by SocialVPN, would allow users to discover services at their social peers' resources with any configuration. Multicasting in such context may also benefit from information at the social networking layer for efficient messaging. Our prototype supports a simple multicast to enable Bonjour-based systems such as Apple's iTunes, or the Pidgin instant messenger, to discovery peers on the SocialVPN. Future work will consider this problem in more detail.

Acknowledgements

This research is sponsored by the National Science Foundation under Grant Nos. IIP-0758596 and CCF-0622106, the South East Alliance for Graduate Education and the Professoriate, the Florida Education Fund under the McKnight Doctoral Program. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] Opensocial, 2008. <http://code.google.com/apis/opensocial/>.
[2] Friendstore, 2008. <http://www.news.cs.nyu.edu/friendstore/>.
[3] A. Mislove, K.P. Gummadi, P. Druschel, Exploiting social networks for internet search, in: Proceedings of the 5th Workshop on Hot Topics in Networks (HotNets'06), 2006.
[4] B. Ford, P. Srisuresh, D. Kegel, Peer-to-peer communication across network address translators, CoRR abs/cs/0603074.
[5] R. Figueiredo, O. Boykin, P.S. Juste, D. Wolinsky, Social vpns: integrating overlay and social networks for seamless p2p networking, in: IEEE Workshop on Collaborative Peer-to-Peer Systems (COPS'08), 2008.
[6] A. Ganguly, A. Agrawal, P. Boykin, R. Figueiredo, Ip over p2p: enabling self-configuring virtual ip networks for grid computing, in: Parallel and Distributed Processing Symposium, 2006. IPDPS 2006, 20th International, 2006, 10 pp.
[7] Facebook statistics, May 2009. <http://www.facebook.com/press/info.php?statistics>.
[8] A. Mislove, M. Marcon, K.P. Gummadi, P. Druschel, B. Bhattacharjee, Measurement and analysis of online social networks, in: Proceedings of the 5th ACM/USENIX Internet Measurement Conference (IMC'07), 2007.
[9] F. Qiu, C. Lin, H. Yin, Ekm: An Efficient Key Management Scheme for Large-scale Peer-to-peer Media Streaming, vol. 1426, Springer, Berlin/Heidelberg, 2006. pp. 395–404.
[10] S. Rafaeli, D. Hutchison, A survey of key management for secure group communication, ACM Comput. Survey 35 (3) (2003) 309–329. doi:http://doi.acm.org/10.1145/937503.937506.
[11] Openvpn – the open source vpn., 2009. <http://openvpn.net/>.
[12] M. Tsugawa, A. Matsunaga, J.A.B. Fortes, Virtualization technologies in transnational dg, in: dg.o '06: Proceedings of the 2006 international conference on Digital government research, ACM Press, New York, NY, USA, 2006, pp. 456–457. doi:<http://doi.acm.org/10.1145/1146598.1146747>.

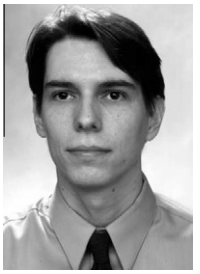
[13] A. Sundararaj, P. Dinda, Towards virtual networks for virtual machine grid computing, 2004. URL <http://citeseer.ist.psu.edu/645578.html>.
[14] I.X. Jiang, Violin: virtual internetworking on overlay. URL <http://citeseer.ist.psu.edu/714412.html>.
[15] Hamachi – instant, zero configuration vpn., May 2009. <http://https://secure.logmein.com/products/hamachi/vpn.asp?lang=en>.
[16] L. Deri, R. Andrews, N2n: a layer two peer-to-peer vpn., 2008. <http://luca.ntop.org/n2n.pdf/>.
[17] S. Aoyagi, M. Takizawa, M. Saito, H. Aida, H. Tokuda, Ela: a fully distributed vpn system over peer-to-peer network, in: International on Applications and the Internet, IEEE/IPSJ. IEEE Computer Society, Los Alamitos, CA, USA, 2005, pp. 89–92.
[18] A. Ganguly, P.O. Boykin, D.I. Wolinsky, R.J. Figueiredo, Improving peer connectivity in wide-area overlays of virtual workstations, in: HPDC '08: Proceedings of the 17th international symposium on High performance Distributed Computing, ACM, New York, NY, USA, 2008, pp. 129–140. doi:<http://doi.acm.org/10.1145/1383422.1383439>.
[19] M. Gjoka, M. Sirivianos, A. Markopoulou, X. Yang, Poking facebook: characterization of OSN applications, in: Proceedings of the 1st ACM SIGCOMM Workshop on Social Networks (WOSN'08), 2008.
[20] B. Carminati, E. Ferrari, A. Perego, Private relationships in social networks, in: ICDE Workshops, IEEE Computer Society, 2007, pp. 163–171. URL: <http://dblp.uni-trier.de/db/conf/icde/icdew2007.html/CarminatiFP07>.
[21] B. Carminati, E. Ferrari, A. Perego, Rule-based access control for social networks, in: R. Meersman, Z. Tari, P. Herrero (Eds.), OTM Workshops (2), Lecture Notes in Computer Science, vol. 4278, Springer, 2006, pp. 1734–1744. URL: <http://dblp.uni-trier.de/db/conf/otm/otm2006-w2.html/CarminatiFP06>.
[22] J. Golbeck, J. Hendler, Inferring binary trust relationships in web-based social networks, ACM Trans. Internet Technol. 6 (4) (2006) 497–529. doi:http://doi.acm.org/10.1145/1183463.1183470.
[23] X. Liu, H. Yin, C. Lin, Y. Deng, Efficient key management and distribution for peer-to-peer live streaming system, in: International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS 2007). 2007, pp. 638–642.
[24] X. Zhang, M. Nakae, M.J. Covington, R. Sandhu, Toward a usage-based security framework for collaborative computing systems, ACM Trans. Inform. Syst. Security. 11 (1) (2008) 1–36. doi:http://doi.acm.org/10.1145/1330295.1330298.
[25] J. Domingo-Ferrer, A public-key protocol for social networks with private relationships, in: Lecture Notes in Computer Science, (Modeling Decisions for Artificial Intelligence-MDAI'2007), vol. 4617, 2007, pp. 373–379.
[26] E. Rescorla, N. Modadugu, Datagram transport layer security, April 2006. <http://www.rfc-editor.org/rfc/rfc4347.txt>.
[27] Universal tun/tap driver, 2008. <http://vtun.sourceforge.net/tun/index.html>.
[28] P.O. Boykin, J.S.A. Bridgewater, J. Kong, K. Lozev, B. Rezaei, V.P. Roychowdhury, Brunet software library. <http://brunet.ee.ucla.edu/brunet/>.
[29] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, M. Bowman, Planetlab: an overlay testbed for broad-coverage services, SIGCOMM Comput. Commun. Rev. 33 (3) (2003) 3–12. doi:http://doi.acm.org/10.1145/956993.956995.
[30] Google app engine cloud infrastructure, 2009. <http://code.google.com/appengine/>.
[31] M. Krochmal, Multicast dns internet draft, September 2008. <http://files.multicastdns.org/draft-cheshire-dnsext-multicastdns.txt/>.

Pierre St Juste is a Ph.D. student under the supervision of Dr. Renato Figueiredo. He completed both his Bachelor of Science and his Master of Science degrees in Computer Engineering from the department of Electrical and Computer Engineering at the University of Florida. His research interests include social networks, virtual networks, overlay networks, autonomic computing, and virtualization.

David Wolinsky received a BS and MS in Electrical-Computer Engineering at the University of Florida and is currently working towards a Ph.D. His research focuses on virtual networking in grids and clouds, the use of P2P overlays for virtual networking, and autonomic aspects of virtual networks. Additionally, David currently works as a system administrator for the Archer project which comprises a decentralized, distributed grid for computer architecture research. David is also the current maintainer of IPOP (IP over P2P) and Grid Appliance projects.

P. Oscar Boykin is an Assistant Professor of Electrical and Computer Engineering at the University of Florida. His research interests include fault tolerant computing, quantum cryptography, quantum information and computation and peer-to-peer networking. Oscar Boykin received his Ph.D. in Physics from UCLA in 2002.

Michael J. Covington is currently an independent security researcher and consultant living in Portland, Oregon. Previously, Dr. Covington worked as a Senior Research Scientist in the Network Security Lab at Intel Corporation. His research is focused on improving security and reliability for next-generation systems and platforms. With more than seven patents pending and as the author of numerous papers that have been published in leading academic conferences and journals, Dr. Covington's research has explored formal access control modeling, cutting edge authentication techniques, and security approaches for pervasive computing environments. Dr. Covington received his Ph.D. and MSCS degrees from the Georgia Institute of Technology's College of Computing in Atlanta, Georgia. He also holds a B.S. degree from Mount Saint Mary's College in Emmitsburg, Maryland.



Renato J. Figueiredo is an Associate Professor at the Department of Electrical and Computer Engineering of the University of Florida. Dr. Figueiredo received the B.S. and M.S. degrees in Electrical Engineering from the Universidade de Campinas in 1994 and 1995, respectively, and the Ph.D. degree in Electrical and Computer Engineering from Purdue University in 2001. From 2001 until 2002 he was on the faculty of the School of Electrical and Computer Engineering of Northwestern University at Evanston, Illinois. His research interests are in the areas of virtualization, distributed systems, overlay networks, computer architecture, and operating systems.