

Social VPNs: Integrating Overlay and Social Networks for Seamless P2P Networking

Renato J. Figueiredo, P. Oscar Boykin, Pierre St. Juste, David Wolinsky
Advanced Computing and Information Systems Lab, University of Florida, Gainesville, FL 32611

Abstract

In this paper we introduce Social VPNs, a novel system architecture which leverages existing social networking infrastructures to enable ad-hoc VPNs which are self-configuring, self-managing, yet maintain security against untrusted parties. The key principles in our approach are: (1) self-configuring virtual network overlays enable seamless bi-directional IP-layer connectivity among parties linked by means of social connections; (2) social networking infrastructures greatly facilitate the establishment of trust relationships among parties, and these can be seamlessly integrated with existing public-key cryptography implementations to authenticate and encrypt traffic flows on overlay links end-to-end; and (3) knowledge of social connections can be used to improve the performance of overlay routing.

This paper describes the architecture of such Social VPNs and a prototype implementation which integrates the Facebook API, IP-over-P2P virtual networks, and the IPsec security infrastructure in a virtual router. We demonstrate the ability of the prototype to support existing, unmodified TCP/IP applications while transparently dealing with the increasingly common case of users connected to the Internet through Network Address Translators (NATs), and present qualitative and quantitative analysis of its functionality and performance.

1 Introduction

A large percentage of Internet users now routinely interact with social networking web-sites [1, 7]. Social networking infrastructures have been extremely successful at allowing users to discover and establish *social links* to one another, but provide little support for allowing users to connect with their peers through *network links*. The main overall contribution of this paper is a novel architecture for *Social VPNs* — a virtual

network which integrates social and overlay networking to bridge this gap. The main objective of a social VPN system is simple: to securely interconnect Internet users, where peer-to-peer IP-layer network tunnel links are created, automatically, as a result of connections established through social network infrastructures. To achieve this objective, the architecture we present addresses the following technical challenges:

How to enable easy discovery of peers and secure exchange of credentials for private communication? The social VPN leverages programmable, authenticated interfaces provided by social networking infrastructures to discover friends and their public keys (Section 2.1).

How to provide IP-layer connectivity in an environment constrained by scarce availability of IPv4 address spaces, private networks, and network address translators? The social VPN leverages overlay networking techniques to create private networks that can scale to large numbers of users, do not conflict with address spaces of existing resources, and support existing IPv4 applications (Section 2.2).

How to interface with existing resources in a non-intrusive and user-friendly manner? The social VPN leverages virtualized network interfaces and self-configuring techniques to establish both numeric IP addresses and names to those devices without requiring any user interventions (Section 2.3).

In this paper we make a case for a system which merges overlay and social networking to provide peer-to-peer IP connectivity among users who can seamlessly leverage social networking relationships to establish network-layer private channels. In this context, social networking is key to enable a system where each user is able to establish and maintain their individual trust relationships with friendly interfaces. Furthermore, P2P overlay networking is key to enable a system where private tunneling is setup and maintained without requiring central administration. The resulting social VPN is useful because it enables users to communicate, securely, using existing TCP/IP applications which are not currently supported by so-

cial networking infrastructures. These include desktop sharing (e.g. VNC and RDP), audio streaming (e.g. iTunes), shared file folders, audio/video-conferencing, multi-user games, to name a few.

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 names and virtual private IPv4 addresses to hosts in a manner which avoids conflicts with current network deployments and requires no user configuration; 2) it supports automatic generation, exchange and discovery of peer credentials through a social networking infrastructure, 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 social VPN functionality is thus accomplished without burdening users with the complex, error-prone configuration typically required to bring up public key and network tunneling infrastructures in VPNs.

This paper also describes a prototype implementation¹ which demonstrates the feasibility of instantiating social VPNs in realistic wide-area environments, where users are subject to firewalls and network address translators. The prototype encapsulates all the software required for overlay routing, address translation and secure tunneling in easy-to-deploy virtual routers. We present qualitative and quantitative performance analyses of the social VPN prototype for several relevant applications and configurations, and present directions for future work, including approaches for supporting efficient multicast-based discovery leveraging information from the social networking topology.

2 System Architecture

2.1 Peer discovery and Key Exchange

The peer discovery process uses social networking APIs to query relationships (already established by users) to allow an application to automatically build and maintain lists of peers. The core primitives used are: (1) the ability to query one's list of social connections, and (2) the ability to store and retrieve one's unique overlay identifier. These are functions already supported in social networking APIs such as Facebook's.

The peer discovery process also enables unique names to be assigned to resources connected to the social VPN. Unique fully qualified domain names (FQDN) in a virtual name space can be derived from the user's account, social network name, and overlay name. In order to allow users to be connected to multiple social networks, and to have multiple devices connected to the social VPN, the peer discovery mechanism needs to be able to look up multiple infrastructures. Name resolution is further elaborated in Section 2.3.

While various options are possible to implement the security infrastructure for a social VPN, our approach is based on public key cryptography (PKI). In doing so, we can reuse existing tools for processing x.509 certificates, and we can seamlessly integrate with an IP-layer VPN technology which is widely used (IPsec). PKI infrastructures are well-understood, and robust implementations are available; however, the management of keys is complex, error-prone and overwhelming to end users who are not familiar with the security theory in which PKI is based. Here the social networking infrastructure is leveraged to allow key exchanges to take place automatically and transparently to users, as the following example illustrates.

Suppose friends Alice and Bob wish to communicate over a social VPN. The social VPN software at Alice's endpoint generates a host private key and uses it to self-sign a host certificate, and uses APIs to authenticate and publish her public key to a data store hosted by the social networking site under a well-known entry associated with her account (e.g. *alice:social_vpn_key*). Bob runs the same social VPN software; by the same means, his public key is also published in the social networking infrastructure under *bob:social_vpn_key*. The social VPN software at each endpoint keeps a list of current friends, which is periodically updated (e.g. by polling the social networking infrastructure for the current list of friends). For each new friend f_i found, the VPN automatically downloads their respective public key $f_i:social_vpn_key$ and places it in a repository accessible to the network tunneling stack. Thus, as a side effect of Alice and Bob becoming friends in the social network, Alice's VPN downloads *bob:social_vpn_key* and Bob's VPN downloads *alice:social_vpn_key*. The social networking site authenticates both Alice and Bob using password authentication. Alice and Bob use software that authenticates the social networking site using SSL/TLS and a signed x.509 certificate (as is standard in web browsers). Thus, the key distribution is as secure as the social networking site.

Once public keys are obtained, the exchange of shared session keys for communication over a network

¹Available from <http://socialvpn.org>.

tunnel can be performed on-demand and in a point-to-point fashion with protocols such as IKE (Internet Key Exchange [5]). Take IPsec/IKE as an example of a VPN framework. Alice’s VPN router configures itself to enforce that any communication between *IPg_alice* and *IPg_bob* requires IPsec processing and pointing to Alice’s private key and Bob’s public key; a reciprocal configuration is done at Bob’s VPN router. Until both endpoints are configured, IP-layer communication between Alice and Bob is blocked by the IPsec stack at either endpoint. Once both endpoints are configured with their respective certificates, the IPsec stack negotiates session key exchanges transparently to them and establishes a private tunnel. Thereafter, any IP packets Alice and Bob exchange through the social VPN overlay are authenticated and encrypted end-to-end.

2.2 Solving the addressability problem in IPv4

Deployments of social VPNs need to accommodate user bases that can be quite large — there are currently tens of millions of users registered with major social networks. 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 address spaces do not scale to very 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 social VPN can both scale to number of users larger than the limit imposed by an IPv4 private address range and avoid address space conflicts with end user networks.

The key idea behind this approach exploits the fact that, while the *total* number of participants of a social networking infrastructure can be very large (tens of millions of users), the number of relationships a *single user* has at any point in time is significantly smaller (typically tens to hundreds). While the number of relationships a user 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 in 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 collision, as follows:

The social VPN maintains, at each user’s endpoint, a private IP address space that is sized to accommodate the expected maximum number of social connec-

tions 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 social VPN router such that it avoids collision with any existing network interfaces of a social VPN user’s machine. The social VPN endpoint has a local virtual address *IPl*, and each social connection peer *p* is mapped to a locally unique virtual IP address *IPp*.

The virtual network overlay operates with addresses assigned to a separate space. Each social VPN endpoint is associated with an identifier *IDg* which is globally unique within the overlay. For example, the IPOP [3] virtual overlay allows the use of (IP,namespace) tuples associating IPv4 addresses to a namespace identifier (a string) as globally unique identifiers. This enables easy integration with IPv4 virtual network devices while supports multiple disjoint private IPv4 address spaces within the overlay [4].

Then, at each social VPN end-point, pair-wise *IDg-IPp* mappings are established for each social network peer. Source and destination network address translation (SNAT/DNAT) is employed to map outgoing packets from *IPl* to *IPp*. The overlay network routes packets destined to *IPp* using the global identifier *IDg*. Address translation is performed at the IP level, and thus applies to all TCP/UDP ports.

The following example illustrates this mapping process. It assumes IPv4 overlay identifiers as in IPOP. Consider a configuration where users Alice and Bob are connected by the social network infrastructure named *social* and the IPOP overlay by means of a social VPN router. In the example, Alice’s local virtual IP address *IPl_alice* is 192.168.0.2, and Bob’s *IPl_bob* is 172.16.0.2. Bob communicates with Alice through peer address *IPp_alice* 172.16.0.3; he connects to all his social VPN peers through peer addresses mapped to the address space 172.16.0.0/255.255.0.0. Alice communicates with Bob through peer address *IPp_bob* 192.168.0.3; she can connect to up to 65,535 social peers over the IP space 192.168.0.0/255.255.0.0. The name *alice.social.ipop* maps to the peer addresses 172.16.0.3 at Bob’s endpoint, and *bob.social.ipop* maps to 192.168.0.3 at Alice’s endpoints. The peer virtual IP addresses are translated to *IPg* global addresses by the social VPN router through the use of rules for destination and source IP address translation. Assume an overlay namespace *ns* and that global identifiers for Alice and Bob are *10.10.10.10:ns* and *10.20.20.20:ns*, respectively.

For example, Bob sends a packet to Alice from address 172.16.0.2 to address 172.16.0.3. The packet’s source and destination are translated such that when it reaches the IPOP overlay it is viewed as a packet

originated from *IPg_bob* (10.20.20.20) and destined to *IPg_alice* (10.10.10.10) within namespace *ns*. The IPOPOP overlay routes the packet and delivers to Alice’s social VPN router, where another translation process occurs: the source is translated from *IPg_bob* (10.20.20.20) to Alice’s peer IP entry for Bob (192.168.0.3), and the destination is translated from *IPg_alice* (10.10.10.10) to *IPL_alice* (192.168.0.2). Packets flowing in the reverse direction (Alice to Bob) are translated in a similar fashion.

It is anticipated that the Internet will increasingly support IPv6, which will alleviate some of the challenges pointed out above. Nonetheless, the approach described in this paper can easily be integrated with an IPv6 infrastructure. In essence, IPv6 deployments lift some of the restrictions that need to be dealt with by the virtual network (the 128-bit address space obviates the need for dynamic address translation and the mandatory support for IPsec facilitates deployment of the end-to-end authentication and privacy) but not all. Key components of our architecture (peer discovery, key exchange and virtual name-to-address mappings) are still necessary to implement a social VPN. It is also possible to bootstrap an IPv4 social VPN over an IPv6 infrastructure, which would be appealing from the standpoint of supporting legacy IPv4 applications during a transition to IPv6 networks.

2.3 Being User Friendly: Naming and Packaging

The ability to refer to peers with intuitive names (rather than numeric IP addresses) is key. The name spaces of a social VPN are private and decoupled from the Internet. Name resolution is also used as a means of establishing dynamic mappings of IP addresses. This translation can be performed by a loop-back DNS proxy which uses social networking APIs to query for VPN/overlay unique identifiers given a user name.

Address resolution in a social VPN keeps two translations associated with a virtual FQDN: (1) a map to a global identifier *IDg* which is unique in the network overlay; and (2) a map to a virtual address *IPl* which is locally unique within a virtual IP address space allocated to a user. The reason for keeping two mappings is to avoid collision with the address spaces local to different participants, as discussed in Section 2.2.

The social VPN software stack can be packaged in a plug-and-play system that is easy to deploy by end users. We have created two prototypes, one using Linux-based virtual appliance routers running on a system VM (e.g. VMware, VirtualBox), and one using a .NET virtual machine. Both expose a virtual

network interface (using a “tap” device) where the IP address and name resolution are configured automatically through DHCP and a loop-back DNS server.

3 Analysis

3.1 Implementation

We have implemented and evaluated a prototype social VPN based on the IPOPOP [3] virtual network and the Facebook API. The prototype implements the techniques described in Section 2 as follows:

Peer discovery: we used the Facebook social network in this prototype. Facebook provides a set of Web services that can be accessed through a REST-based interface — all methods are accessed through HTTP GET and POST requests. We used a C#-based developer toolkit library and created a desktop application which authenticates to Facebook, advertises one’s keys and discovers other users who also run the social VPN software. The Facebook DataStore facility was used to store and query for user virtual IP addresses and public keys. This information is stored in an object named *IpopData* as tuples (user name, virtual IP, RSA public key). This object is associated with a user ID; when a user starts the social VPN, he or she authenticates with Facebook. IPOPOP is started and obtains a virtual IP address, an RSA private/public key is automatically generated, and the user’s (name, virtual IP, public key) tuple is stored in an *IpopData* object. Other peers discover this information by querying their list of friends, and the *IpopData* objects associated with each friend ID.

Key exchange and private tunneling: The virtual IP addresses and RSA public keys retrieved during peer discovery are used to configure IPsec for private tunneling. The RSA public keys are stored in the router appliance, and a configuration is automatically generated for each friend using the Racoon IPsec tool. For example, at Alice’s appliance, Racoon is configured to require tunneling through IPsec for outgoing packets from *IPg_alice* to *IPg_bob* and incoming packets in the reverse direction, and to point to Bob’s public key. Negotiation of shared session keys for the IPsec tunnel occurs on-demand with the IKE protocol when packets flow from Alice to Bob.

IP address translation: The virtual IP addresses retrieved during peer discovery are used to establish IP mappings between *IPL_bob* and *IPg_bob*. We have implemented two approaches to IP address translation, one using packet rewriting within IPOPOP in a standalone social VPN router application, and one using IPtables

DNAT/SNAT mappings applied to network interfaces within a router virtual appliance.

Name resolution: The user names retrieved in the peer discovery process are used to populate a loop-back DNS server which is part of the Facebook application and which is configured to listen on the virtual tap device installed on the user’s computer. This virtual DNS server maps names composed from Facebook user IDs to the locally assigned private IP address *IPL_bob*. We derive a user-friendly FQDN for a user by concatenating their first name and last initial (e.g. *bob_s.facebook.ipop*). If a name conflict is detected within one’s social network using this naming scheme, a sequence of steps is taken to reach a unique name, including using the full last name (e.g. *bob_smith.facebook.ipop*) and concatenating the user’s unique ID to the name. Support for users connected to the social VPN through multiple computers is allowed; each social VPN application has a unique 160-bit P2P identifier, and an additional string in the host name is prepended to identify a user’s machine (e.g. *pc3.bob_smith.facebook.ipop*).

Virtual network interface setup: The social VPN prototype configures a virtual network interface device on the host upon startup. To this end, the social VPN application inspects the network configuration of the user’s host, discovers an unused private network address space (by scanning subnets in the 10.0.0.0, 172.16.0.0, and 192.168.0.0 ranges), and assigns an address within this space for the social VPN host (*IPL_bob*).

3.2 Experiments

We bootstrapped a wide-area deployment with a total of 430 IPOP router nodes on PlanetLab resources in North and South America, Europe, Asia and Australia to conduct experiments with the social VPN prototype. Virtual appliances running the social VPN router connecting to this overlay were deployed on hosts behind three different NATs (two distinct ISP providers, and one university network). Hosts tested included x86-based desktops and laptops running Linux, Windows and MacOS and virtual appliances running on VMware and VirtualBox.

We qualitatively tested several applications, and quantitatively assessed the latency overhead and the bandwidth achieved by our prototype. The following applications were successfully tested between social VPN nodes: VNC-based shared remote desktop sessions; file sharing using Samba and NFS; a 3D first-person game (Warsow); a light-weight Web server (Cherokee); music sharing and chat over Bonjour with

IP translations (n)		n=1	n=10	n=100
latency	(no IPsec)	5.23	5.30	5.60
(ms)	(with IPsec)	5.34	5.54	5.70
throughput	(no IPsec)	6.72	6.20	5.78
(Mb/s)	(with IPsec)	6.01	5.81	5.71

Table 1. Average ICMP round trip latencies (100 trials) and iperf TCP throughput (30-second transfer).

iTunes and Pidgin; and voice-over-IP with Ekiga.

We have also measured the latency overhead and the throughput of the prototype. In this setup, two desktop machines (1-core 2.5GHz and 3.2GHz Athlon CPUs with 512MB RAM) in a 100Mbit/s switched Ethernet local-area network were connected through the social VPN. We measured the round-trip latency of ICMP ping messages and TCP throughput achieved with the iperf tool. In the experiments, we varied the number of IPtables mappings and IPsec tunnels (n=1, n=10, and n=100) to assess the impact of having different numbers of peers connected to a social network. A summary of the results is shown in Table 1. We observe a latency overhead of around 5ms and throughput of around 6Mbit/s. An increase in the number of translations results in slightly increased round-trip latency and decreased throughput. Overall, the observed level of performance is acceptable for wide-area environments with residential broad-band connections targeted by our architecture. Nonetheless, several performance improvements are being pursued.

4 Related Work

Several related projects have considered the use of virtual networks to facilitate the deployment and management of wide-area distributed systems, including ViNe [9], VIOLIN [6], VNET [8]. The Internet Indirection Infrastructure (I3) project has also addressed support for legacy applications [2]. While these overlays could conceivably be used as the communication backbone of a social VPN, these projects have not addressed the use of social networks to facilitate peer discovery and key exchange.

This work is also related to P2P VPNs such as Hamachi². There are key differences in our approach. First, Hamachi users are responsible for creating virtual subnets and passwords are used to allow other users to join the subnet; hence, a user joining a sub-

²<http://www.logmein.com/hamachi>

net is connected to other subnet users without knowing their identity. In contrast, the social VPN guarantees a user is only connected to the identities they are socially connected with. This advantage of the social VPN is rooted on the fact that it provides dynamic IP address translation, which obviates the need for subnets as a mechanism for grouping users, and enables the system to scale while avoiding address collisions. For example, Hamachi uses a currently unallocated class-A address space (5.0.0.0); once this address space becomes allocated, their virtual network will collide with hosts on the Internet, making them inaccessible to their users.

5 Conclusions and future work

The main overall contribution of this paper is a novel social VPN architecture where social connections established through user-friendly Web-based infrastructures guides the creation of private virtual IP network links in a user-transparent manner. To our knowledge, this is the first work to integrate social and overlay IP networks to create self-configuring social VPNs. A prototype implementation of our approach and experiments with a deployment in a realistic wide-area environment have shown the technology behind this approach to be feasible and promising.

There are several existing applications that can benefit from a social VPN, and it can foster the development of new socially-enabled collaborative applications; future work focusing on user studies are to assessing usability and applications of this system. Another future work direction is to investigate how information about social network links can improve overlay routing performance. In the social VPN 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 that support for *multicast* applications, in particular for resource discovery, can greatly enhance the functionality and usability of social VPNs. Multicast-based resource discovery implementations such as Bonjour and UPnP enable users to discover services at their social peers' resources without 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 to discover peers on the social VPN. Future work will consider this problem in more detail.

6 Acknowledgments

Effort sponsored by the NSF under awards SCI-0537455 and OCI-0721867. 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] Nielsen//NetRatings report, May 2006.
- [2] A. Kubota K. Lakshminarayanan I. Stoica D. Joseph, J. Kannan and K. Wehrle. Ocala: An architecture for supporting legacy applications over overlays. In *Proc. USENIX Symp. on Networked Systems Design and Implementation (NSDI)*, pages 267–280, 2006.
- [3] A. Ganguly, A. Agrawal, P.O.Boykin, and R. Figueiredo. IP over P2P: Enabling self-configuring virtual IP networks for grid computing. In *Proc. IEEE Int. Parallel and Distributed Processing Symposium (IPDPS)*, Rhodes, Greece, June 2006.
- [4] A. Ganguly, D. Wolinsky, P. O. Boykin, and R. Figueiredo. Decentralized dynamic host configuration in wide-area overlay networks of virtual workstations. In *Proc. First Workshop on Large-scale, Volatile Desktop Grids (PCGrid)*.
- [5] D. Harkins and D. Carrel. The internet key exchange. Internet RFC 2409, Nov. 1998.
- [6] X. Jiang and D. Xu. Violin: Virtual internetworking on overlay infrastructure. In *Proc. 2nd Int. Symp. on Parallel and Distributed Processing and Applications (ISPA)*, pages 937–946, 2004.
- [7] M. Rankin Macgill A. Smith A. Lenhart, A. Maden. Teens and social media: The use of social media gains a greater foothold in teen life as email continues to lose its luster. Pew Internet and American Life Project, Dec 2007.
- [8] A. Sundararaj, A. Gupta, and P. Dinda. Dynamic topology adaptation of virtual networks of virtual machines. In *Proc. Seventh Workshop on Languages, Compilers and Run-time Support for Scalable Systems (LCR)*, Oct. 2004.
- [9] M. Tsugawa and J. A. B. Fortes. A virtual network (vine) architecture for grid computing. In *Proc. IEEE Int. Parallel and Distributed Processing Symposium (IPDPS)*, Rhodes, Greece, June 2006.