On the Design and Implementation of Structured P2PVPNs

David Isaac Wolinsky, Kyungyong Lee, Yonggang Liu, P. Oscar Boykin, Renato Figueiredo
University of Florida
{davidiw, klee, yonggang, boykin, renato}@acis.ufl.edu

Linton Abraham
Clemson University
labraha@clemson.edu

ABSTRACT

In recent years, P2P VPNs have become quite popular by allowing users to connect directly with each other bypassing the overhead of communicating through a third party proxy. These P2P VPNs require connecting to a central server for authentication, NAT traversal, and proxying in the off chance NAT traversal fails. This significantly improves upon classical, centralized VPNs, though it adds a new complexity either maintenance of all-to-all connections during run-time or the involvement of a centralized authentication entity for each live connection attempt. For this solution, we propose a completely run-time decentralized P2P model based upon a structured P2P system. In this paper, we will describe the components of this model as well as present and evaluate our reference implementation. A decentralized P2PVPN has an intuitive and simplistic setup, reduces the requirements for connectivity, offers better proxy selection in lieu of NAT traversal, and provides an opportunity for more intuitive trust solutions. For evaluation, we will compare system and networking overheads of the different VPN technology focusing on latency, bandwidth, CPU, and memory.

1. INTRODUCTION

A Virtual Private Network (VPN) provides the illusion of a Local Area Network (LAN), namely direct communication, over a wide area network such as the Internet while guaranteeing secure and authenticated communication amongst participants. Common uses of VPNs include accessing company or academic network resources while traveling abroad, playing LAN based video games over the Internet, connecting distributed resources from multiple sites, and securing your Internet traffic while in unsecure locations. In the context of this paper, we focus on VPNs that provide connectivity between individual resources and so all resources that need symmetric connectivity will need to be configured wth VPN software.

While traditional VPNs enable such distributed connectivity they do so at the cost of maintaining a central server, which becomes the conduit for all traffic, becoming a performance bottleneck and potentially removing end-to-end security. To alleviate this, there have been three directions 1) support for multiple VPN servers for a single VPN [10, 4], 2) the use of P2P connections for bypassing central communication that rely on run-time central authentication [7, 8], and 3) the use of unstructured P2P networks to form VPNs based upon shared secrets without user authentication and limitations on network size [5, 2, 9]. In this paper, we present a novel approach to forming secure, scalable, efficient, and self-configuring VPNs through the the use of Structured P2P systems that has no reliance on centralized systems after initialization. Structured P2P technology enables users to communicate directly with all users without knowing anything beyond their virtual IP bypassing the need for centralization while providing all-to-all communication without maintaining all-to-all connectivity with participants. Interesting applications of P2P include efficient wide area multicast, data distribution, storage, chat applications, and even IP connectivity.

Current generation P2PVPNs do not scale well, provide features such as full-tunneling of network traffic, such as forwarding Internet traffic, nor do they have intuitive ability for scalable multicast or broadcast. P2PVPNs rely on direct connectivity and in general will not work if NAT (Network Address Translation) traversal between peers is unsuccessful. Unstructured P2P based VPNs have similar issues, though have the ability to reuse the unstructured overlay to relay packets though typically relying on all-to-all connectivity in the system. Furthermore, unstructured P2P systems currently lack the ability to police participants in the system.

The problems we seek to address with our P2PVPN model include:

- reducing the role of centralization for user authentication in a VPN
- managing participants in a live system
- supporting full-tunneling of Internet traffic in a P2P system
- handling relay selection in lieu of unsuccessful NAT traversal
- supporting multicast and broadcast communication

A rudimentary overview of our solutions to the above problems follows and will be covered in depth in the rest of this paper. To provide fully decentralized run-time connectivity and policing, we use an automated certificate authority based upon the use of user groups. In the case of full-tunneling, P2PVPNs introduce significantly more complexity since a simple routing table swap as done in central VPNs no longer work, as such we investigate three different mechanisms for tunneling all Internet traffic to our full-tunnel endpoint(s) besides our P2P traffic. When nodes cannot directly communicate, they seek to connect to peers that are mutually physically close to each other and use them to relay communication. For efficient multicast and broadcast communication, we rely on the use of bootstrapping a private P2P system whose members are only participants of the VPN.

Explicitly, our contributions made in this paper are:

- automated group-based certificate authority
- three different approaches to configuring full-tunneling
- intelligent selection of relays
- use of a private P2P VPN system bootstrapped of a general P2P system

The rest of this paper is organized as follows. Section II gives an overview of current VPN technologies and the efforts to decentralized. Section III introduces P2P structures and our previous work IPOP (IP over P2P). Section IV describes the contributions of this paper, namely a feature-full P2PVPN. In Section V, we discuss our implementation and present evaluation comparing centralized, P2P, and our VPN. Finally, we give some concluding remarks in Section VI.

2. VIRTUAL PRIVATE NETWORKS

There exist many different flavors of virtual networking, this paper focuses on those that are used to create or extend a virtual layer 3 network. A few examples of such technologies include Cisco's Systems VPN and AnyConnect VPN Client [?] as well as OpenVPN [10]. In this section, we begin by going in depth on client configuration of VPNs, overview concepts of server roles in centralized VPNs, and then roles of participants in P2P VPNs. Finally we conclude the section by presenting a table **??** which compares qualitatively the features of VPNs that fit in these categories.

2.1 Client VPN Configuration

In figure **??**, we abstract the common features of all VPNs with focus on the client. The key components of the client are 1) client software that communicates with the VPN overlay directly and 2) a virtual network device. During initialization VPN software begins by authenticating with some overlay agent, optionally it then queries the agent for information about the network such as the network address space, and then starts the virtual network device.

There are many different mechanisms for communicating with the overlay agent. For quick setup, a system may provide a shared secret password or key that is common for the entire network. A more user-friendly and manageable approach re-uses the shared secret mechanism and then adds user accounts and passwords, thus blocking unauthorized users from the VPN, while still making it somewhat difficult for brute force attacks to work, so long as the key remains private. For the strongest level of security, each client can be configured to have a signed-certificate that makes brute force attacks all but impossible. The tradeoffs come in terms of usability. While the use of uniquely sign-certificates may be the most secure, it can be quite difficult for novice computer users. A good balance found in many environments is the mixture of a shared secret and user account, where the shared secret is included with the installatoin of the VPN application where the application is distributed from a secured site.

Once the user has connected with the overlay, the virtual networking device needs to be configured enabling the user's machine to communicate with other participants in the VPN. This configuration varies by VPN, commonly though, this information contains the network address space, an allocation of an address for the user's machine, and potentially a remote peer for full-tunneling.

In order to communicate over the VPN transparently, there must exist a network device driver that allows common network APIs such as Berkeley Sockets and hence existing application to work without modification. There are many different types of virtual networking devices, though due to our focus on an open platform, we focus on TAP [6]. TAP allows the creation of one or more Virtual Ethernet and / or IP devices and is available for almost all modern operating systems including Windows, Linux, Mac OS/X, BSD, and Solaris. A TAP device exists as a file descriptor providing read and write operations. Incoming packets from the virtual network are written to the TAP device and the networking stack in the OS delivers the packet to the appropriate socket. Packets that are read from a TAP device are those that are sent by sockets to the virtual network.

The virtual network device is either configured using static addressing or dynamically through dynamic host configuration process (DHCP) [3, 1]. This causes a new routing rule that causes all packets sent to the virtual network address space to be sent to the virtual network device. When a packet is read from the TAP device, it can then be sent to the overlay via the client application. The overlay will deliver the packet to another end point, which can be a client or a server enabled with virtual networking stack. When receiving a packet, it will be written to the TAP device. In most cases, the IP layer header will remain unchanged while configuration will determine how the Ethernet header will be handled.

The described configuration so far, creates what is known as a split tunnel, or virtual network connection that only has passes traffic directly related to the virtual network and not Internet traffic. Another form of tunneling exists called full tunneling. Full tunneling allows a VPN client to securely forward all their Internet traffic through a VPN router. This enables a user to ensure all their Internet communication originates from a secure and trusted location and provides some level of security when a user is an insecure and potentially hostile environment, such as an open wireless network at coffee shop.

Most centralized VPNs implement full-tunneling by doing a routing rule swap, where the default gateway becomes an endpoint in the VPNs subnet and traffic for the VPN server is routed to the local network gateway. For example, on a typical home network, all traffic for the VPN server is sent via to the networks router and then via the Internet to the VPN server. All other traffic is sent to the virtual network device, then sent securely to the VPN server. In a P2P system, there becomes two new considerations 1) P2P traffic must not be routed to the VPN gateway and 2) there may be more than one VPN gateway. Allowing more than one VPN gateway per VPN allows distribution of the cost of maintaining a full-tunnel who will have 3 additional messages for each incoming packet. We discuss and provide solutions to this problem in 4.1.

2.2 Centralized VPN Servers

2.3 Centralized P2P VPN Systems

2.4 P2PVPN Client / Server Roles

Unlike centralized systems, pure P2P systems have no concept of dedicated servers, not to say that a user cannot start an instance of the P2P VPN software purely for enhancing or enabling connecctivity. In these systems, all participants are members of a collective known as an overlay. Current generation P2P, decentralized VPNs use a P2P unstructured network, where there are no guarantees about distance and routability between peers. As a result participants tend to be connected to a random distribution of peers in the overlay. Finding a peer requires either global knowledge of the pool or at worst case broadcasting a lookup message to the entire overlay. While unstructured P2P systems have some scalability concerns, P2P systems in general allow for server-less systems. In the realm of VPNs, all client VPNs are also servers with varying different responsibilities depending on the VPN application, as we present in table **??**.

Typically, decentralized, P2P VPNs begin by attempting to connect to well known end points running the P2P overlay, a list of such end points is distributed with the application or some other out-of-band mechanism. In the case of P2PVPN, this involves communication with one or more BitTorrent trackers to find other members of the P2PVPN group. N2N requires knowledge of any existing peer in the system. It uses this endpoint to bootstrap more connections to other peers in the system, allowing the application to be an active participant in the overlay and

	VPN Type	Authentication Method	Peer Discovery	NAT Traversal	Availability
OpenVPN	Centralized with multiple servers	Certificates or passwords with a central server	Stored at central server(s)	Relay through server(s)	Open Source
CloudVPN	Centralized with multiple servers	CA-signed Certificates	Broadcast	Relay through server(s)	Open Source
Hamachi	Centralized P2P	Password at central server	Stored at central server	NAT traversal and relay through central server	personal use only, no private servers
GBridge	Centralized P2P	Password at central server	Stored locally	NAT traversal and relay through central server	free to use, close source, no private relays, Windows only
Wippien	Centralized P2P	Password at central server	Stored locally	NAT traversal, no relay support	Mixed Open / Closed source
N2N	Unstructured P2P	Shared secret	Broadcast look up	NAT traversal and support of relay through overlay	Open Source
P2PVPN	Unstructured P2P	Shared secret	Everyone knows about everyone else	No NAT traversal, support of relay through overlay	Open Source
tinc	Unstructured P2P	CA Certificates / Private key	Everyone knows about everybody	No NAT traversal, support of relay through overlay	Open Source
IPOP	Structured P2P	CA Certificates or pre-exchanged keys	DHT lookup	NAT traversal and relay through physically close peers	Open Source

Table 1: VPN Comparison

potentially be a bootstrap connection for other peers attempting to connect.

3. STRUCTURED PEER-TO-PEER SYSTEMS

4. COMPONENTS OF A P2PVPN

4.1 Full-Tunneling over P2P

5. EVALUATING VPN MODELS

6. CONCLUSIONS

7. REFERENCES

[1] S. Alexander and R. Droms. *RFC 2132 DHCP Options and BOOTP Vendor Extensions.*

[2] L. Deri and R. Andrews. N2N: A layer two peer-to-peer vpn. In *AIMS '08: Proceedings of the 2nd international conference on Autonomous Infrastructure, Management and Security*, pages 53–64, Berlin, Heidelberg, 2008. Springer-Verlag.

[3] R. Droms. *RFC 2131 Dynamic Host Configuration Protocol*, March 1997.

[4] E. Exa. CloudVPN. `http://e-x-a.org/?view=cloudvpn`, September 2009.

[5] W. Ginolas. P2PVPN. `http://p2pvpn.org`, August 2009.

[6] M. Krasnyansky. Universal tun/tap device driver. `http://www.kernel.org/pub/linux/kernel/people/marcelo/linux-2.4/Documentation/networking/tuntap.txt`, March 2007.

[7] LogMeIn. Hamachi. `https://secure.logmein.com/products/hamachi/vpn.asp`, July 2008.

[8] K. Petric. Wippien. `http://wippien.com/`, August 2009.

[9] G. Sliepen. tinc. `http://www.tinc-vpn.org/`, September 2009.

[10] J. Yonan. OpenVPN. `http://openvpn.net/`, March 2007.