



iWebGate DMZ Server

VLAN/MP2P: VLAN over Multi Peer-to-Peer

Revision: 1.0.0

Edition: Pre-Development

Public Release: 31st March, 2010

Author: Charlie Gargett,
Technical Director,
iWebGate Technology

Availability: Public Release

Copyright © 2010. iWebGate Technology Ltd. All Rights Reserved.

The copyright holder of this document hereby provides the reader (“You”) permission to reproduce in part or in full, transmit and/or store this document in part or in full in a public retrieval system in any format, provided that You conspicuously and appropriately publish on each copy and/or fragment this copyright notice and disclaimer of warranty in full and intact; keep intact all the notices that refer to this copyright notice and to the absence of any warranty; and give any other recipients of this document and/or fragment a copy of this copyright notice along with such document or documents.

Failure to comply with the above clause is deemed to be an infringement of Australian and/or International Copyright Laws. Copyright infringement is a serious matter under such Copyright Laws.

This document, the technology it describes, and all other related materials are subject to change without notice at the total discretion of the author and/or copyright holder.

In no way does the content of this document in part or in whole represent a commitment on the part of the copyright holders and/or iWebGate Technology Ltd or subsidiaries, and are in no way held liable for the failure of any part of software or hardware resulting from the use, misuse, ability or inability to use the information and procedures outlined in this document by qualified or unqualified persons alike. Employment of procedures outlined within this document should only be performed by appropriately qualified persons.

If deemed necessary, clarification of this notice can be obtained from the copyright holder by making such a request in writing and forwarding to iWebGate Technology Ltd, Level 29, 221 St Georges Tce Perth, Western Australia, 6000.

The copyright holder hereby acknowledges any/all copyright and/or trademark holdings mentioned throughout this publication.

Table of Contents

Abstract.....	7
1. Introduction.....	9
2. VLAN/MP2P vs Traditional VPN.....	10
3. Software Architecture.....	13
4. UDP vs TCP Peer-to-Peer Communications.....	15
5. Packet Header Structure.....	16
6. Implementation.....	19
6.1. Broker Server Function.....	19
6.2. Peer Function.....	21
6.3. The Firewall Hole Punching Process.....	25
6.4. Network Topology.....	27
7. Network Security.....	28
7.1. Peer Security.....	28
7.1.1. Peer Point Data Encryption.....	29
7.1.2. Layer 2 Stateful Packet Inspection.....	29
7.1.3. Routing Strategies (Broadcast and Multicast).....	30
7.1.4. Peer De-Registration Policy.....	31
7.2. Broker Server Security.....	31
7.2.1. Man-in-the-Middle Attack Prevention.....	32
7.2.2. Pre-Registration Certificate Check.....	33
7.2.3. Peer Overload Protection.....	33
8. Component Service Protocol.....	34
9. What You Wont Find in VLAN/MP2P.....	37
10. Paper Conclusion.....	37

Abstract

With ever increasing demand for doing business on today's and tomorrow's Internet, competent Network Engineers place securing private networks, whether they be small or large, at the top of their priority list. This introduces the problem of how to ensure users have accessibility to the LAN from the Internet and how to provide uninterrupted access to Internet resources without having to allocate public IP addresses to each computer which is just not viable given the current shortage.

Two potential solutions were introduced for consideration: IPv6 and IPv4 NAT. The latter soon found its way into just about every firewall available on the market today and looks to be the answer for now, but it has the adverse effect of preventing two disparate peers from communicating directly with each other without assistance.

Peer-to-Peer (P2P) technology has improved awareness of firewall “hole punching” techniques that address the inability for peers to communicate directly. However, it often means new software has to be developed or existing software needs to be modified in order to take advantage of P2P communications which can sometimes be costly, inefficient and result in proprietary protocols that other software may not understand.

This document discusses the use of software to provide transparent P2P communications at the network level removing the need for applications to be rewritten and allow them to communicate natively without the need to be concerned about firewalls. The software employs technology borrowed from VLAN, P2P and L2TP VPN software to achieve multi-peer-to-peer (MP2P) communications.

Welcome to VLAN over Multi-P2P (VLAN/MP2P).

1. Introduction

VPN using Peer-to-Peer technology is by no means a new concept and has been around since the first few years of the new millennium. It provides an extremely simple, robust and lightweight means of allowing two peers to communicate natively without the need for complex set-ups or large or private bandwidth requirements.

Unfortunately, promotion of such technology is relatively unnoticed in comparison to that of other VPN technologies such as L2TP/IPsec which has been heavily promoted by companies such as Microsoft. Aside from the awareness issues associated with VPN over P2P, there are other disadvantages that have hindered the growth of the technology.

- Most VPN over P2P systems are hosted by third parties to help remove the complexities of hosting a “rendezvous” server (known as a Broker Server in this software) needed to broker connections between peers. Such services have experienced difficulty in gaining the trust of companies looking for reliable VPN services.
- Most hosted systems only allow for communication between two peers at a time. This can be restrictive where multiple peers need to communicate with each other in more of a VLAN configuration instead of VPN.
- Licensing of hosted services can often be restrictive where per-user costs can become expensive for small-medium businesses where working from home is encouraged or required along with the need to support “Road Warriors”. Some providers also place restrictions on the number of users that can be configured as part of their service.
- Self-hosted solutions can be complex to deploy and require the contracting of specialist service providers to get systems functioning. This can often include the need to provide additional services in order to make it easier for people to use.
- Peers are unable to configure multiple Peer instances (membership to more than one network) using the resources of just one process. In order for a computer to handle more than one peer instance, multiple processes need to be run where each process handles one peer instance requiring more local resources.

VLAN/MP2P addresses each of these issues through providing an easy to deploy and use system that allows remote users to quickly establish connections between each other. It addresses the need to be able to configure a single node with multiple Peer instances in one resource friendly process without restrictive licensing requirements.

Network Engineers are able to deploy the service within their own

infrastructure eliminating the trust issue and allowing users to take advantage of a far more reliable, high performance remote access solution.

2. VLAN/MP2P vs Traditional VPN

The VLAN/MP2P software has been designed with the Demilitarised Zone in mind whereby the network on which the Broker Server resides does not automatically become a part of the VLAN that is created ensuring that other servers and resources are not exposed. The Broker server should always be run within a semi-trusted security zone and not on the private trusted network. The key to this is that the Broker Server never becomes a part of any VLAN it brokers connections for.

Traditional VPN solutions often house the VPN server on the trusted network and ultimately have greater potential to expose the trusted network unnecessarily and often unwittingly if the VPN services is not properly configured and protected. Placing the VPN Server within the DMZ is better but still has the potential to expose other resources within that zone.

A VPN is commonly used to provide remote users with access to resources normally only available to local LAN users.

A virtual private network (VPN) is a computer network in which some of the links between nodes are carried by open connections or virtual circuits in some larger network (e.g., the Internet) as opposed to running across a single private network. [1]

The virtual network is tunnelled through the parent network (commonly the Internet) by encapsulating the private traffic inside additional protocol headers in order to reach the remote destination via the parent network. Authentication and encryption is nearly always required in order to ensure the private connections and data are secured.

Two distinct types of VPN can be implemented, infrastructure (permanent) and ad-hoc or “Road Warrior” style VPN.

Infrastructure VPN is usually permanently deployed and is designed in such a way that routing traffic between two distinct physical LANs is done via the VPN connection. This is often referred to as LAN-to-LAN or site-to-site VPN and uses dedicated hardware deployed to maintain the connection requiring reliable and high-speed connections to the parent network.

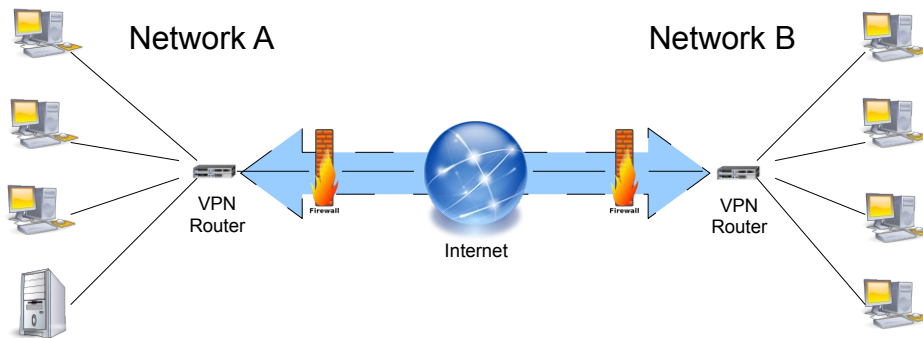


Figure 2.1 – Infrastructure VPN

Ad-hoc VPN (also referred to as “Road Warrior”) is a more of an “as needed” connection from a single host to a private LAN infrastructure allowing the remote user to become part of the LAN, but without the full benefit of the LAN (eg speed and sometimes the broadcasting capability). Typically, users will connect to a VPN service provided by a specific server (eg Windows Server or Cisco VPN Server) using proprietary VPN client software (eg Windows Vista or Cisco VPN Client).

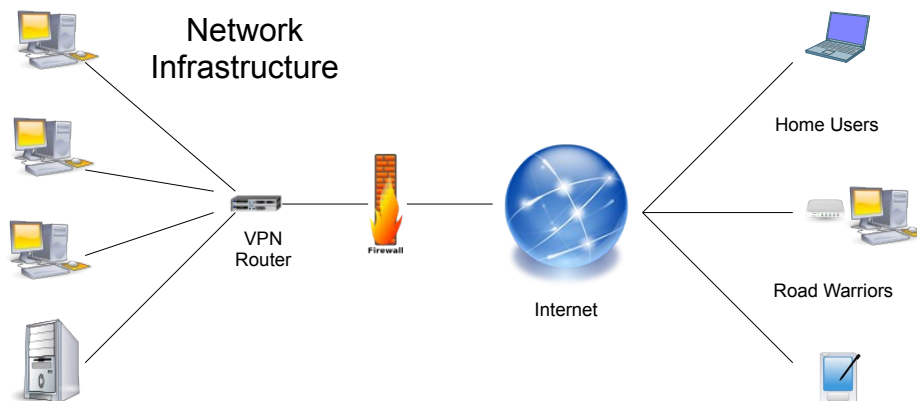


Figure 2.2 – Ad-Hoc or “Road Warrior” VPN

Traffic on both types of VPN needs to be encrypted to ensure it remains secure. Infrastructure VPN services usually provide this encryption at the

Layer 2 level where the VPN negotiation usually takes place and employs proprietary encryption technology to maintain data security. Ad-hoc VPN often employs encryption at the Layer 3 level using protocols such as IPsec or SSL/TLS.

Regardless of the VPN type used, VPN deployment and maintenance requires central infrastructure that needs to be configured and maintained by Network Engineers to which users must adhere and relinquish control of the set up of their VPN to that of one or two people.

These VPNs also require traffic to be routed through the VPN server infrastructure and thus form a star topology. This can have an adverse impact on two remote peers who wish to communicate directly with each other where all traffic is routed via the central VPN service. The system also requires that the server connection be maintained at all times. A server drop-out will inevitably drop the entire VPN service and the nodes that are connected to it.

For ad-hoc VPN, protocols other than traditional UDP or TCP are used to carry out VPN tasks. This introduces real problems for the remote user as they may reside behind a firewall that simply doesn't understand these protocols or has no way of allowing the protocol through (although this is less common). Worse still, the user simply has no control of the firewall they reside behind which is configured to block such VPN traffic (eg, Hotels, public wireless access points, etc).

In developing VLAN/MP2P, the intention was to avoid all of the issues related with traditional VPN by including the mixed benefits of P2P and VLAN. The P2P routines are also dropped to the Layer 2 level to transparently provide benefits to applications running on a peer:

- P2P provides for direct peer communications where traffic does not need to go via the VPN server. This breaks free from the traditional star topology of most VPN services and removes a lot of the traffic burdens lumbered onto the VPN server. Multiple Peer instances can be configured into one process allowing membership to multiple VLANs without excessively consuming local resources.
- VLAN provides the benefit enjoyed by computers attached to a broadcast domain regardless of their physical location. This can greatly assist in determining the location of peers and broadcasting messages to all of the peers that are members of a VLAN community; while
- Layer 2 provides the benefit of transparently providing peer-to-peer communication to applications that are not specifically written to communicate in such a way. New applications do not need to be written and existing ones do not need to be modified in order to benefit from P2P communication. Encryption is also applied at this level to ensure all traffic is encrypted including traffic that is not

encrypted by the application itself.

3. Software Architecture

VLAN/MP2P loosely couples the philosophy of VLAN technology to Peer-to-Peer techniques in order to achieve the binding of peers to each other from disparate locations around the Internet. The goal is to achieve the same broadcasting benefits enjoyed by computers in a LAN or VLAN which can simplify the process of locating remote peers and establishing communications with them.

A virtual LAN, commonly known as a VLAN, is a group of hosts with a common set of requirements that communicate as if they were attached to the same broadcast domain, regardless of their physical location. [2]

Each peer within a single VLAN/MP2P community has two basic common requirements:

1. to be able to communicate with other peers directly (just like a VLAN or LAN) with a minimum of fuss
2. to be able to broadcast information to each Peer within the VLAN community regardless of their physical location

VLANs are essentially a Layer 2 construct where a single VLAN closely corresponds to a particular IP subnet (Layer 3). This mapping, in essence, allows network administrators to collect nodes together in a single VLAN grouped by their IP subnet. Traffic transmitted at the Layer 2 level (usually delivered in accordance to a node's hardware or MAC address) is restricted to those nodes which are a member of the VLAN.

VLAN/MP2P employs the same technique whereby Peers provide their assigned hardware address as part of a registration with a Broker Server. The Broker Server maps the public IP address of the node to the provided hardware address and uses the mapping to determine where to send packets destined for a specified hardware address. VLAN/MP2P will collect the Peer members together into a single IP subnet as is done using managed VLAN switches and a Peer may broadcast packets at the pseudo Layer 2 level to other peers using the Broker Server.

Like *L2TP* [3], VLAN/MP2P conceptually works at the Layer 2 level while the transport is actually done at the Layer 3 level. VLAN/MP2P solely employs UDP based P2P technology as a means of communicating with peers on the same VLAN which primarily means that it has the capability of handling NAT traversal and firewall hole punching without any need to reconfigure NAT firewalls.

While the P2P technique might still be proprietary to the underlying software, applications do not need to be rewritten to make the most of the

P2P features it offers (including firewall hole punching) because the peer negotiation is done at several layers lower than the application layer making it transparent to any application running on the node. In fact, each Peer instance presents itself as a virtual network interface (VNI) known as a TAP adapter which is offered by the operating system kernel (see figure 3.1).

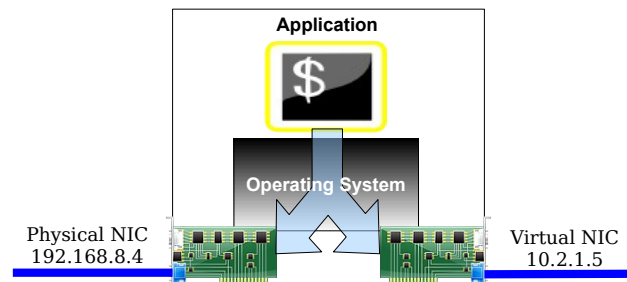


Figure 3.1 – The application view of VLAN/MP2P

Each Ethernet frame that arrives on the VNI from the operating system kernel is intercepted by the peer software, encrypted and encapsulated in a small VLAN/MP2P header that determines where the message needs to be sent. This message is transmitted back into the OSI stack at Layer 3 as a UDP datagram before being sent down to the true Layer 2 interface (see figure 3.2).

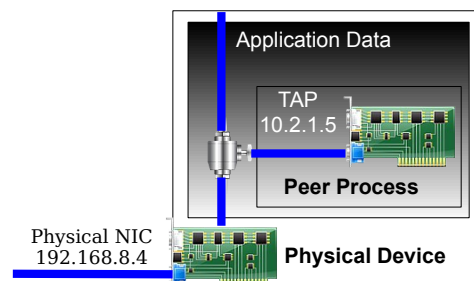


Figure 3.2 – The operating system view of VLAN/MP2P

The Peer component process determines where the message needs to be sent by interpreting the original Ethernet frame to determine the destination hardware (MAC) address and then figures out whether or not the packet can be sent directly to the remote Peer or if it must be relayed through the Broker Server (if it is a broadcast frame or if the Peer is

unknown at the time of transmission).

For most ad-hoc VPN connections, a single client instance is all that is needed, but there are situations that demand for a single node to join more than one network while keeping configuration and resource use at a minimum.

Similar VPN/P2P software [4] attempts to address this issue through executing a single client process for every network the node wishes to join. Each process will require its own slice of memory and its own separate UDP port on which to communicate with other nodes. Local resources can quickly be consumed and can also place additional load on the translation table in the NAT firewall.

This is where the “M” in MP2P comes in. Multi Peer-to-Peer provides a means of allowing a single Peer component process to act as more than one Peer (multiple Peer instances). The single Peer component process will establish a TAP interface for each Peer instance and optionally use the global UDP port on which P2P communication will take place. The TAP interface or port over which traffic arrives at the node is used to determine which of the local Peer instances the incoming message is for.

This requires less resources on the local computer, may require less entries in the NAT translation table and is ideal for IT support staff who need to remotely support more than one customer from the same computer or for contract software developers who need access to more than one remote system.

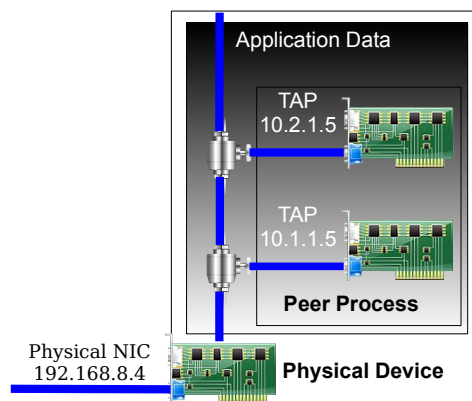


Figure 3.3 – Multi-peer configuration

4. UDP vs TCP Peer-to-Peer Communications

While TCP provides a more stable transport for messages between hosts on local or disparate networks, VLAN/MP2P makes use of UDP as its primary peer communication protocol for a variety of reasons.

- UDP hole punching techniques are better supported by the majority of NAT firewalls available on the market today [5]. TCP hole punching is becoming more supported, but is somewhat more complex than that of its counterpart and requires the use of more ports. UDP can provide simultaneous multi-peer connections on a single port exercising efficient use of local resources and placing less load on NAT translation tables.
- Research [6] also shows that UDP has far better support for hairpin connections at the NAT device than that of TCP. In a number of instances, support for TCP hairpin connections was non-existent which most certainly hampers the Peer connection process.
- In terms of P2P, TCP shares the same traversal pitfalls as UDP but presents additional ones in certain instances where Simultaneous TCP Open and other TCP specific techniques are employed. Additionally, operating systems must support address and port reuse (SO_REUSEADDR & SO_REUSEPORT) in order to function correctly. Some socket implementations still do not support such options.
- Layer 2 deals in Ethernet frames which are more analogous with UDP datagrams making UDP more appropriate for dealing with Ethernet frame re-transmissions. UDP packet sizes can be kept to similar sizes to that of the original Ethernet frame derived from the MTU settings of the TAP adapter.
- UDP is more tolerant of high-latency network connections (such as satellite) and can still be just as reliable over higher speed, lower-latency networks such as LAN.
- From an application level, TCP connections are still being honoured so end points still benefit from stateful connections. **Translation to UDP is performed transparently to the application so the benefits of both protocols can apply.**

5. Packet Header Structure

Each packet transmitted over UDP connections, whether it originates from a Peer or a Broker Server, is prepended with a packet header that contains information about the Peers involved in the message as well as other instructional information.

The structure of the packet header is as follows:

TGT_MAC	SRC_MAC	V	O	A	R	R	R	COMMUNITY NAME	PEER_NAME	IP1	IP2	P1	P2
---------	---------	---	---	---	---	---	---	----------------	-----------	-----	-----	----	----

The first two fields are 6 byte blocks that contain the target and source hardware (MAC) addresses of the Peers associated with the packet. These

are used by the Broker Server to route packets (instead of the IP addresses) to the correct destination.

These are followed by seven (7) single byte fields that represent the protocol version (V) where the version must match that which is understood by the Broker Server or remote Peer. Any mismatch in protocol version will cause the Peer or Broker Server to silently drop the packet.

The following byte (O) depicts the origin of the message where a non-zero byte depicts the packet originated from the Broker Server and was not forwarded on behalf of another Peer.

The next byte (A) determines the action required by the Peer or the Broker Server receiving the packet. Actions can match one and only one of the following:

Value	Macro Name	Description
0	ACTION_PACKET	<p>Packet must be forwarded on to the Peer specified in the TGT_MAC field on behalf of the Peer specified in the SRC_MAC field.</p> <p>The packet payload contains the data read from the source Peer's TAP interface (encrypted by the Peer).</p>
1	ACTION_REGISTER	<p>A Peer is registering or re-registering (if already registered) with the Broker Server. No forwarding of this packet is required.</p> <p>The packet payload should contain the PKI encrypted authentication credentials of the user for new registrations and re-registrations alike.</p>
2	ACTION_DEREGISTER	<p>The Broker Server has requested that the Peer specified in the TGT_MAC field de-registers the Peer specified in the SRC_MAC field. This packet will always originate from the Broker Server so if the Origin header byte is zero, the packet should be ignored.</p> <p>The packet payload should be empty.</p>
3	ACTION_REGISTER_REQ	<p>Receipt of a packet with this action set will nearly always arrive at a Peer via the Broker Server originating from the Peer specified in SRC_MAC. The Peer is requesting negotiation of direct Peer-to-Peer communication and is always sent in conjunction with a ACTION_PROBE packet sent directly to the Peer.</p> <p>The packet payload should be empty.</p>
4	ACTION_REGISTER_ACK	<p>A packet with this action set can come from either the Broker Server (Origin header byte set to non-zero) or a remote Peer (Origin header byte set to zero).</p>

		<p>The packet is received from the Broker Server when a Peer sends a ACTION_REGISTER packet to the Broker Server. Failure to receive this packet after sending such a packet should force the Peer to assume the Broker Server is off-line or is unreachable.</p> <p>The packet is received from the remote Peer (specified in SRC_MAC) when a ACTION_REGISTER_REQ packet is sent to the remote Peer via the Broker Server. Receipt of this packet from a remote Peer indicates that direct Peer-to-Peer negotiation was successful.</p> <p>The packet payload should be empty.</p>
5	ACTION_REGISTER_NACK	<p>Receipt of this packet indicates explicit non-acknowledgement of either a ACTION_REGISTER packet sent to the Broker Server (Origin header flag on) or an ACTION_REGISTER_REQ packet sent to a remote Peer via the Broker Server (Origin header flag off). The Peer instance receiving this packet should take appropriate action as a response.</p> <p>The packet payload should be empty.</p>
6	ACTION_PROBE	<p>ACTION_PROBE packets serve two purposes and should always be silently dropped by any Peer receiving one. The Broker Server should never receive one of these packets.</p> <p>A Peer will send this type of packet directly to a Peer right after it sends a ACTION_REGISTER_REQ packet via the Broker Server to the same Peer. This opens the specified port in a well-behaved firewall in preparation for receipt of a ACTION_REGISTER_ACK from the remote Peer completing the Peer-to-Peer negotiation.</p> <p>The other purpose of this packet is to keep firewalls open for continued direct peer communication. These will cease if a remote Peer is de-registered. If these packets are received they should always be dropped silently.</p> <p>The packet payload should be empty.</p>
7	ACTION_BKEY_REQ	<p>A packet with this action set can only be sent from a Peer instance to the Broker Server. It instructs the Broker Server to send the X.509 certificate containing the server's public key to the requesting Peer instance.</p> <p>The packet payload should be empty.</p>
8	ACTION_BKEY_ACK	<p>Received by a Peer instance and must always come from a Broker Server. The PKI for VLAN/MP2P is only used for verification of the Broker Server and for</p>

		<p>encryption of the authentication credentials. Data encryption is performed using OpenSSL's symmetric cipher API.</p> <p>The packet payload will contain the X.509 certificate of the Broker Server which the Peer should check against an appropriate CA Certificate store to verify the Broker Server's identity before sending encrypted authentication credentials.</p>
9	ACTION_BKEY_NACK	<p>Indicates that an error has occurred with loading or attempting to send the Broker Server certificate to the Peer instance requesting it. This should be treated as an error for the Peer instance receiving the packet and the Peer instance should be de-registered.</p> <p>The packet payload should be empty.</p>

The next four (4) bytes (R) are reserved for later use and should be set to 0 and ignored. Likely use of one of these four is to set the platform type on which the source Peer is running.

The next two blocks are both 32 bytes long and contain the source Peer's VLAN ID or Community name and Node name respectively. The VLAN Community name is used by the Broker Server to ensure that broadcast or multicast packets are delivered only to member Peers. The host name is not currently used by the Broker Server at this stage and is included as a means of providing a human readable name for future developments.

The final 12 bytes represent the private and public end points (IP address and port pairs) of the source Peer in the following order:

```

IP1 = Private IP Address
IP2 = Public IP Address
P1  = Private UDP Port
P2  = Public UDP Port

```

Note that the private end point is the one assigned to the TAP interface and not the physical interface that might contain the real private end point of the node on the LAN.

6. Implementation

The VLAN/MP2P software package comes in two very simple components, the "broker" and "peer" component processes that represent the Broker Server and the Peer respectively.

6.1. Broker Server Function

The Broker Server reads its configuration from a specified file passed to it via the command line. The file need only contain a few parameters to get the Broker Service started:

```

UDP_PORT           = 7718
PIDFILE            = /var/run/broker.pid
LOGFILE            = /var/log/icbd/broker.log
PUBLIC_CERTFILE    = /home/data/certificates/http/public/cert.pem
PRIVATE_KEYFILE    = /home/data/certificates/http/private/key.pem

EFFECTIVE_USER     = 2
EFFECTIVE_GROUP    = 2

PEER_TIMEOUT       = 60
TIMEOUT            = 10
DEBUG_LEVEL        = 0
VERBOSE           = 0

```

Listing 6.1 – Example broker.conf File

UDP_PORT	The UDP port on which the Broker Server will listen for incoming traffic from Peers. If the Broker Server is to reside behind a NAT firewall, then this port should be appropriately redirected to the server. Default: 7718.
PIDFILE	Defines the location of the Broker Server's PID status file. Default: /var/run/broker.pid
LOGFILE	Defines the location of the Broker Server's log file. Default: /var/log/icbd/broker.log
PUBLIC_CERTFILE	Defines the location of the Broker Server's X.509 Signed Certificate file containing the Public Key. Default: /home/data/certificates/http/public/cert.pem
PRIVATE_KEYFILE	Defines the location of the Broker Server's Private Key file. Default: /home/data/certificates/http/private/key.pem
EFFECTIVE_USER	Defines the User ID of which the Broker Server process will run. Ignored on Windows. Default: 2
EFFECTIVE_GROUP	Defines the Group ID of which the Broker Server process will run. Ignored on Windows. Default: 2
PEER_TIMEOUT	The time period for which a Peer remains inactive (no traffic to or via the Broker Server) before the Broker Server sends de-registration notifications. Set to 0 to disable and allow Peers to maintain their own lists. Must be set to at least 20 seconds or 0 to disable. Default: 60 seconds
TIMEOUT	Sets the port listener time out period to allow the Broker Server

	to handle other tasks. Default: 10 seconds
DEBUG_LEVEL	Defines the level of information printed to the log file based on the following: 0: Logging Disabled 1: Fatal messages only 2: Fatal and Error messages 3: Fatal, Error and Warning messages 4: Fatal, Error, Warning and Info messages 5: Everything (forced by VERBOSE=1) Default: 0
VERBOSE	Tells the Broker Server to set DEBUG_LEVEL to 5 and outputs all messages to standard error. Default: 0

The Broker Server starts with simply establishing a listener on the specified UDP socket on either default port number 7718 or on the port specified through the configuration file.

Once established, it waits for incoming Peer registrations (REGISTER packets) and responds to each one with a REGISTER_ACK packet. REGISTER_ACK packets reply with the original REGISTER packet provided by the Peer but modifies it by adding the public IP address and port pair and sets the Peer action header to REGISTER_ACK. On receiving the REGISTER_ACK, the Peer not only confirms successful registration with the Broker Server, but is also able to retrieve the public IP address and port pair used by the NAT firewall or the node itself. It is this pair through which other peers will communicate.

A Peer will continue to transmit packets to the Broker Server until such time that it can communicate directly with other peers. The Peer will also continue to periodically send REGISTER packets to the Broker Server to let it know that it is still “alive”.

In the event that the Broker Server has not received a REGISTER packet from a Peer for a significant period of time and PEER_TIMEOUT is not set to zero (0), the Broker Server will signal to all other Peers in the VLAN community that the Peer has dropped out. This instructs the remaining Peers to de-register the Peer from their “Direct” or “Indirect” Peers list (see Peer Function) as it is likely that the firewall has closed communications with the missing Peer and should it re-appear at a later stage, appropriate P2P connectivity can be re-negotiated.

6.2. Peer Function

By comparison the Peer functionality is far more complex than that of the

Broker Server which is an indicator that the Broker Server is under far less load than traditional VPN servers. The Peer process is capable of handling more than one Peer configuration in a single process and for each configuration, an optional UDP port and TAP interface is assigned (it is possible and more common for a global UDP port to be used).

The Peer process is responsible for determining which Peer an incoming UDP packet is for and then transferring the packet to the respective TAP interface and vice-versa if an Ethernet frame arrives on the TAP interface.

The Peer process is also responsible for parsing the packet header of an incoming UDP packet and responding according to the details found in the header. Ethernet frames must also be read from the TAP interface, encrypted and then have the VLAN packet header attached ahead of the payload before being transmitted to the remote Peer.

Finally, the Peer is also responsible for employing the services of the Broker Server to negotiate direct Peer-to-Peer connectivity through appropriate NAT firewalls or on the direct LAN if deemed to be the best path.

On starting up, the Peer process reads in its settings from a single configuration file that contains some global directives and one or more Peer configurations. The configuration file is formatted similar to this:

```
LOCAL_PORT           = 7719
BIND_IPV4            = 0.0.0.0
LOGFILE              = /var/log/icbd/peer.log
PIDFILE              = /var/run/peer.pid
EFFECTIVE_USER       = 2
EFFECTIVE_GROUP      = 2
DEBUG_LEVEL          = 0
TIMEOUT              = 10
VERBOSE              = false

[PEER]
  COMMUNITY_NAME      = My Community Name
  KEYFILE             = /path/to/crypto.key
  PEER_REG_INTERVAL   = 10
  LOCAL_IPV4          = 10.1.1.5
  BROKER_ADDRESS      = 123.124.86.23
  BROKER_PORT         = 7718
  STRICT_CHECKS       = true
  ENABLE_ROUTING      = true

[PEER]
  COMMUNITY_NAME      = Your Community Name
  KEYFILE             = /path/to/crypto.key
  LOCAL_IPV4          = 10.2.1.5
  PEER_REG_INTERVAL   = 10
  BROKER_ADDRESS      = 58.91.6.3
  BROKER_PORT         = 7718
  DISABLED            = true
```

Listing 6.2 – Example peer.conf File

** = Required Field*

LOCAL_PORT	The local UDP port number on which to listen for VLAN/MP2P traffic. LOCAL_PORT can also appear in [PEER] stanzas but if this one is defined, it will override the peer-specific setting. Default: 7719
BIND_IPV4	The local IPv4 address to which the local UDP port is bound. This prevents the UDP port from listening on TAP interfaces which could cause problems on some systems. Default: 0.0.0.0 (listen on all local addresses)
PIDFILE	Defines the location of the Peer node's PID status file. Default: /var/run/peer.pid
LOGFILE	Defines the location of the Peer node's log file. Default: /var/log/icbd/peer.log
EFFECTIVE_USER	Defines the User ID of which the Peer process will run. Ignored on Windows. Default: 2
EFFECTIVE_GROUP	Defines the Group ID of which the Peer process will run. Ignored on Windows. Default: 2
DEBUG_LEVEL	Defines the level of information printed to the log file based on the following: <ul style="list-style-type: none"> 0: Logging Disabled 1: Fatal messages only 2: Fatal and Error messages 3: Fatal, Error and Warning messages 4: Fatal, Error, Warning and Info messages 5: Everything (forced by VERBOSE=1) Default: 0
VERBOSE	Tells the Peer process to set DEBUG_LEVEL to 5 and outputs all messages to standard error. Default: 0
TIMEOUT	Sets the port listener time out period to allow the Peer to handle other tasks. Default: 10 seconds

Following the global directives are the individual Peer configurations. The Peer process will create a new Peer instance for every [PEER] directive it encounters and each Peer configuration must provide all the mandatory fields.

** = Required Field*

*COMMUNITY_NAME	The name of the VLAN ID or Community to create or join. If the Peer is the first to register this name, a new VLAN is created in the Broker Server. No default value.
-----------------	--

*KEYFILE	Specifies the path to the file containing the pre-shared encryption key that is used to encrypt all traffic over the Peer-to-Peer connections. Only peers in the VLAN community with this key will be able to decipher encrypted traffic. No default value.
*LOCAL_IPV4	Defines the IPv4 address that will be assigned to the TAP adapter interface. No default value.
LOCAL_PORT	The local UDP port number on which to listen for VLAN/MP2P traffic. LOCAL_PORT can also appear in Global directives and if defined, will override this peer-specific setting. At least one of the LOCAL_PORT directives must be defined. Default: 7719
PEER_REG_INTERVAL	Defines the re-registration interval (in seconds) for the Peer instance. This keeps the Peer alive in the Broker Server's known peers list and helps maintain firewall status. Default: 20
*BROKER_ADDRESS	Specifies the public IPv4 address of the Broker Server. No default value.
BROKER_PORT	Specifies the UDP port of the Broker Server. Default: 7718
STRICT_CHECKS	Specifies that the Peer should perform strict checks on the Public Certificate that the Broker Server issues for encryption of authentication credentials. If the certificate is not able to pass these checks, the registration of the Peer is aborted. Default: 1 (enabled)
DISABLED	Specifies whether or not the defined Peer instance should be disabled. This will prevent the Peer instance from being registered in the node's Peer process. If included, setting this directive to anything other than 0 will render the Peer instance disabled. Default: 0 (Peer is enabled)
ENABLE_ROUTING	Specifies whether or not the Peer instance will honour routing beyond the VLAN onto its own physical network. This should be enabled with caution and effectively turns the VLAN into a VPN service Default: 0 (Routing is disabled)

Once the Peer configuration file is loaded, the Peer process will check to see how it has been instructed to handle the UDP listener port. If a Global port has been defined, it will be used in place of port definitions provided in any of the Peer instances. Otherwise, each Peer instance will have its own UDP port opened.

The Peer process then iterates through each of the loaded Peer instances and establishes the associated TAP adapter interfaces. When the Peer registers with the Broker Server, the server will check the IPv4 address of

the Peer instance to see that it is not already being used in order to reduce the possibility of IP address clashes between two or more Peers.

Before registering with the specified Broker Server, the Peer component process will iterate through each Peer instance and attempt authentication by first requesting the server certificate containing the Public Key with which credentials will be encrypted. If `STRICT_CHECKS` is enabled for a Peer instance, the Peer component process will perform a verification check on the server certificate to ensure it has been properly signed by a trusted Certificate Authority before attempting to register the Peer instance.

As the Peer process establishes each TAP adapter interface (and UDP port), it registers the Peer instance with the specified Broker Server with a `REGISTER` packet and awaits for a `REGISTER_ACK` response indicating the Peer registration was successful. The return of a `REGISTER_NACK` or no response at all after a specific amount of time will instruct the Peer component process to de-register the Peer instance.

The Peer process keeps track of each of its local Peer registrations and attempts to re-register with the Broker Server every `PEER_REG_INTERVAL` seconds. This ensures that the Broker Server remains aware of the Peer instance and will not instruct remote Peers to de-register the Peer instance from their Peer lists ensuring direct Peer-to-Peer communication is maintained.



For best performance, users should only enable logging and/or verbose mode when attempting to debug connection problems. When running with logging enabled, VLAN connection performance can be substantially reduced with the additional processing required to write logs entries to file.

6.3. The Firewall Hole Punching Process

Once the initialisation process has completed and each Peer instance is registered, the Peer process awaits traffic destined for one of the local Peer instances. When the Peer process receives a packet on the UDP port, it determines which Peer instance the packet is for and begins processing it.

Initially, the Peer should receive a packet that requires writing to the TAP adapter interface and thus into the operating system kernel to be made available to applications. On receipt of such a packet, a number of processes occur:

1. The packet payload is written to the TAP adapter making it available to the operating system kernel and ultimately the application expecting to receive the data.
2. The Peer process will create a new remote Peer instance in memory and append the instance to its indirect Peers list. Packets from this remote Peer will continue to be sent via the Broker Server until such time as a REGISTER_ACK is received from the remote Peer.
3. A REGISTER_REQ packet is generated with the origin Peer of the original packet as the target and sent to the Broker Server for relaying.
4. A PROBE packet is sent directly to the public address of the origin Peer of the incoming packet knowing that it will most likely be dropped at the remote firewall. This opens a new entry into the translation table of the local NAT firewall which will allow new traffic from the origin Peer to come through. The first direct response from the remote peer will most likely be a REGISTER_ACK packet. This is the first phase of the firewall hole punching used in the peering process.

If the PROBE packet was successfully sent to the remote Peer's NAT firewall, the local NAT firewall will have opened up a translation table entry that should allow for the remote Peer to respond directly to the REGISTER_REQ packet.

The local Peer instance should then receive the REGISTER_ACK packet sent directly from the remote Peer. On receipt of such a packet, the local Peer process will promote the remote Peer instance from its indirect Peers list to its direct Peers list indicating that packets destined for that Peer can be sent directly and thus completing the peering process.

Consistent traffic between the two Peers from this point onwards will ensure that the NAT firewall maintains an opening in the firewall rules. However, should the traffic not be consistent enough, the firewall will close the hole created by the recent traffic and the two Peers will no longer be able to communicate directly, requiring the hole-punching procedure to be completed again.

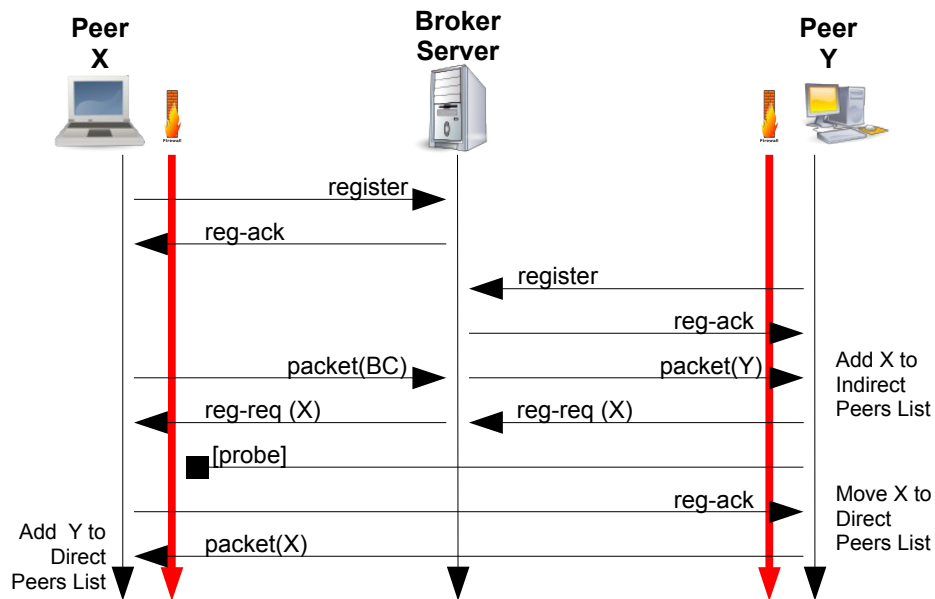


Figure 6.3 – Firewall Hole Punching

To help avoid firewall closure, small PROBE packets are sent every PEER_REG_INTERVAL seconds to the Peers on the direct Peers list. If the remote Peer receives the PROBE packet, it is silently dropped and not written to the remote Peer's TAP adapter interface.

Figure 6.3 illustrates how the firewall hole punching technique works.

In the event that the Peer process receives a DEREGISTER packet from the Broker Server, the specified Peer is de-registered and removed from the Peer's direct or indirect peers list. This instructs the Peer process to forward packets for the de-registered Peer via the Broker Server until the firewall can be penetrated again for direct peer communication.

6.4. Network Topology

Correct VLAN/MP2P set up allows for multiple Peers within a common VLAN community to communicate with each other quickly and efficiently regardless of their physical location. A single node can join more than one VLAN at one time allowing the Peer to communicate with other Peers in more than one VLAN without the need for complex routing tables. The result is a simple mesh network of point-to-point access between Peers as depicted in Figure 6.4.

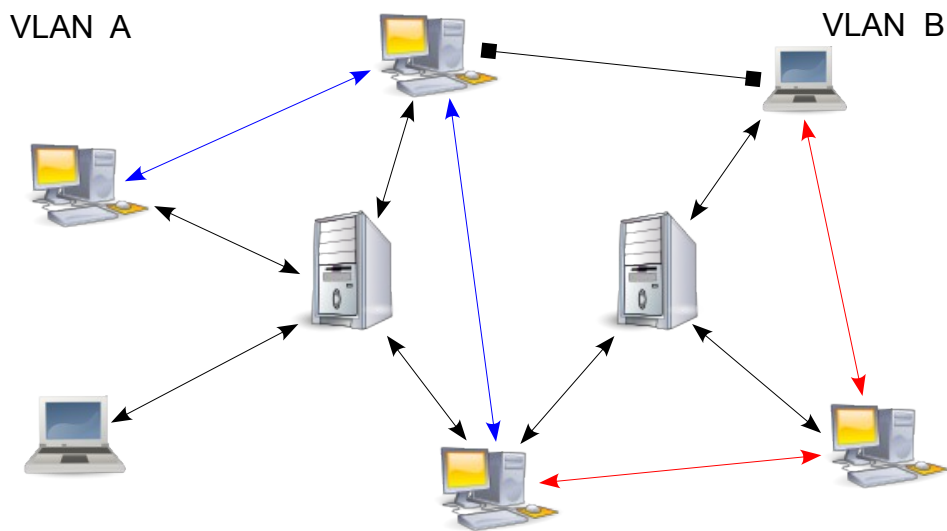


Figure 6.4 – Mesh Network Topology

Figure 6.4 demonstrates a valid VLAN/MP2P network topology that involves two Broker Servers serving two distinct VLAN communities, VLAN A connected in blue and VLAN B connected in red. The black connectors indicate the path taken for registration with a Broker Server or relaying of traffic through respective Broker Servers. The bottom-most node is configured with two Peer instances, one in each of the two VLANs so this node is able to communicate with any of the nodes on the entire diagram.

Other Peers within each VLAN are only able to communicate with other Peers registered within their respective VLAN community. This is because VLAN/MP2P, by default, does not allow for traffic routing through a Peer instance, so Peers are confined to direct Peer-to-Peer communication – which also helps to improve the security of the virtual network. For example, the two nodes connected with square end points indicate that any attempt to connect directly would be blocked by the nodes respective firewall.

If routing were to be enabled on the bottom-most node, then that Peer could act as a router or conduit between the two VLAN networks and/or the physical LAN to which that Peer is connected. This would allow any of the Peers in the diagram to communicate with each other (except the Broker Servers) and also (possibly) the nodes residing on the physical LAN to which that Peer is connected. Appropriate routing tables would need to be configured.

7. Network Security

The very concept of VLAN/MP2P in itself provides a secure means of communication between Peers that are connected to the same hostile network. The direct communication between two or more Peers means that no other computers (trusted or otherwise) need be involved in the connection.

Of course, it cannot be assumed that the data being transmitted between Peers is by any means secure if it is not encrypted. As VLAN/MP2P is a Layer 2 technology, it must assume that all the data being read from the TAP adapter of each Peer instance is unencrypted. Therefore, each frame must be encrypted by the local Peer process prior to being sent to the remote Peer.

We discuss *Peer Point Data Encryption* in the next section.

7.1. Peer Security

Peer security is the most important aspect of VLAN/MP2P deployments as it is through an ill-configured or ill-protected Peer that damage can be done or intruders could gain access to the local Node and potentially remote Peers through VLAN links.

It is important to remember not to confuse the Peer Point Data Encryption with the Public Key encryption required by the Broker Server for authentication. To help distinguish the two, Peer-to-Peer communication (including via the Broker Server) uses Peer Point Data Encryption with a PSK/IV for encrypting traffic. Peer-to-Broker authentication uses Public Key Encryption to encrypt authentication credentials before sending to the Broker Server.

7.1.1. Peer Point Data Encryption

VLAN/MP2P uses the well known and well trusted *OpenSSL*^{[7][8]} encryption libraries in order to encrypt traffic as it passes from the TAP adapter to the UDP port and vice-versa.

As the software uses the UDP protocol as a means of sending and receiving information between peers, true SSL/TLS cannot be used as the routines can only be applied to TCP based connections. This does not, however, prevent the use of symmetric cryptography for UDP messages.

When receiving messages from the TAP interface, the Peer process will first encrypt the frame using the 128-bit Pre-Shared Key (PSK) and 64-bit Initialisation Vector associated with the VLAN ID or community as read from a specified key file (see *Listing 7.1*). On successful encryption, the message is encapsulated in a Peer packet header and sent to the remote peer node.

```
D1 D4 FE 74 EC 76 CB E9
DA 1F 32 F4 9B 3E 48 8D
BA A7 F2 19 4F 83 D7 98
```

Listing 7.1 – Example Pre-Shared Key and IV Pair

On receiving a UDP message from a remote Peer, the local Peer will extract the encrypted payload from the message and decrypt the data using the same PSK/IV pair and write the decrypted message to the local TAP adapter.

The PSK/IV file must be correctly formatted in order to be read and applied as a valid key. The PSK/IV must be provided over 3 separate lines of eight (8) hexadecimal byte representations. Peers do not exchange the PSK/IV automatically meaning users must share the key amongst VLAN community members manually in order to communicate successfully. If a new key is generated, it must be re-shared again to keep all users up-to-date.

7.1.2. Layer 2 Stateful Packet Inspection

(NOTE: Packet Inspection will not be available in initial releases)

One of the greatest benefits of VLAN/MP2P is that it works at a pseudo Layer 2 level and intercepts traffic as it passes down the OSI levels allowing inspection of data packet headers at the TCP/UDP protocol level, the IP level and the Ethernet level.

If enabled, packet inspection allows the Peer user to determine which services are forwarded on to Peers and which services are to remain on the physical LAN only. This can assist the user in ensuring that only specific traffic is allowed out of their computer and delivered to Peers – and vice-versa.

Packet inspection also provides a means of tuning the Peer performance. If a user finds that the Peer connection is being flooded by traffic from a particular service, the ports used by that service can be blocked at the TAP interface to ensure that the traffic remains local only.

Packet inspection is not likely to be available until later releases of VLAN/MP2P. For now, users should use personal firewalls and/or port filtering software to prevent transmission of certain traffic over the VLAN/MP2P links.

7.1.3. Routing Strategies (Broadcast and Multicast)

Some users of traditional VPN services are often surprised to find that while connected to the VPN service, they find that Internet access is either diminished (in terms of speed) or has become completely unavailable. This is due to the VPN client accepting the server's requirement to have the

default route on the computer set to that of the VPN server or some other server within the VPN meaning all traffic is sent via the VPN – including normal Internet traffic that would be better served via the local Internet connection.

VLAN/MP2P does not require replacement of the default route nor additional routing to be added to the Peer node routing tables other than that of the route assigned for the VLAN network over the new TAP interface. This means that users are still able to browse the Internet unhindered and performance is not diminished in any way. All Internet traffic is still sent via the usual default route of the Peer node so performance remains at an optimum level.

The other routing benefit provided by the VLAN/MP2P software is that it will happily honour all broadcasting protocols including NetBIOS over TCP/IP. Most Microsoft Windows networks work over this NetBIOS protocol and issue broadcast and multicast messages in an effort to discover information about the network infrastructure and other nodes on the local LAN.

As VLAN/MP2P is peer-to-peer based and does not allow routing beyond the Peer instance by default, the Peer can be thought of as simply an extension of the LAN and should be able to happily accept and handle broadcast and multicast traffic.

At the Layer 2 level, broadcasting is a means by which a computer can transmit a message to all other computers connected to a broadcast domain (usually a physical network or VLAN) regardless of their address. Broadcasts are usually performed by setting the destination Ethernet hardware address (MAC address) to FF:FF:FF:FF:FF:FF whereby each node on the network is expected to pick up the frame and respond to it if required.

Multicast is similar to broadcast in that a single computer is able to transmit a message on a broadcast domain which will be picked up by other nodes that have expressed an interest in picking up the message. Multicast messages in VLAN/MP2P are pretty much treated in the same way as a broadcast message and results in the message being sent to every Peer in the VLAN Community in the chance that the Peer is interested.

7.1.4. Peer De-Registration Policy

In order to retain access through firewalls and their visibility within the VLAN community network, Peers must maintain their registrations with the Broker Server to which they originally registered. Each Peer must also maintain a list of “direct” Peers that have recently had communication with the Peer instance.

For each Peer instance, the Peer component process checks the `PEER_REG_INTERVAL` setting and selects the shortest one defined to see if

the specified amount of time has passed since the last registration check was performed.

If a check is required, the Peer process iterates its known Peers list for remote Peer instances that should be de-registered from the direct and indirect Peers list.

De-registration is determined by checking the last time a remote Peer instance has communicated with the local Peer instance.

If a remote Peer on the local Peer's direct Peers list has not been “seen” for between `PEER_TIMEOUT` seconds, the remote Peer is de-registered from the appropriate list forcing the local Peer to have to re-attempt P2P negotiation the next time it encounters the remote Peer instance.

It is possible for a local Peer instance to receive a `DEREGISTER` packet from the Broker Server in which case the Peer instance matching the hardware address specified in the `SRC_MAC` field of the header will be immediately de-registered regardless of the list it was found on (if any).

The de-registration process ensures that any chance of potential attack by an impersonating Peer is minimised and also reduces the load on a node and amount of memory being used by the running Peer process.

7.2. Broker Server Security

The Broker Server does not require an encryption PSK/IV pair because it has no requirement to decrypt messages that it might relay between Peers. As only the Peer requires the encryption key pair, the server is never able to “eavesdrop” on messages that pass through it, ensuring that the integrity of the network security is maintained only by the Peers.

The Broker Server is, however, vulnerable to attack that, while it may not necessarily compromise the security of the Peer, can prevent the Broker Server from responding in a timely manner or even worse, allowing a means for unwanted users to establish VLAN communities of their own placing unwanted load on the Broker Server.

7.2.1. Man-in-the-Middle Attack Prevention

To combat unauthorised use and Man-in-the-Middle attacks on the Broker Server, the server requires Peers to register with appropriate credentials to ensure that the user is able to establish VLAN communities through the server. Additionally, a Public Key Infrastructure (PKI) is employed to help validate the identity of the Broker Server and ensure traffic is not being “eavesdropped” by a third party.

The credentials must be sent to the Broker Server in the packet payload as they need to be encrypted before being sent. Given the server has no knowledge of the encryption key (PSK/IV pair) that is in use between Peers,

it is not possible to decrypt the credentials encrypted using the PSK/IV key and it is ill-advised to send the credentials unencrypted.

To combat this problem, the Broker Server and the registering Peer must use a separate method that can be used to encrypt and decrypt (respectively) the packet payload containing the registration credentials. This is where PKI is applied and the Broker Server is configured with the location of its Private Key and Public Certificate (X.509) containing its Public Key. PKI also provides a means of validating the Broker Server's identity through third party signing checks on the public certificate issued by the server.

The public certificate is requested by a new Peer before attempting registration and the Broker Server will always issue the certificate regardless of who the Peer is. The VLAN/MP2P Peer will optionally check the validity of the certificate against a store of trusted CA certificates and will ultimately use the Public Key to encrypt the Peer's authentication credentials before sending them to the Broker Server.

Failure to establish a means of encryption or to provide correct credentials will prevent the Peer from being able to join a VLAN/MP2P community and therefore will not receive assistance from the Broker Server in relaying traffic nor negotiating Peer-to-Peer communication.

Each Peer instance can be configured with the `STRICT_CHECKS` setting enabled which will require the Peer component process to perform appropriate checks on the Certificate issued by the Broker Server. If the issued certificate cannot be verified as coming from a trusted issuer, then registration of that Peer instance is aborted.

A request by a Peer for a Broker Server's certificate will only happen once when the Peer instances are read from the configuration file and the location of the Broker Server is known. If the certificate is received, optionally validated and cached, it is then used to encrypt the authentication credentials during the Peer's initial registration.

Subsequent registrations will not request the Broker Server's certificate again and will use the previously cached one. If a Broker Server's certificate is renewed and re-created, the Peer will either need to be restarted or an instruction will need to be received through the Peer's service port requesting that the certificate be requested from the Broker Server again.

Alternatively, users can perform a manual pre-registration certificate check to verify the validity of a Broker and its certificate. See the next section for more information on this.

7.2.2. Pre-Registration Certificate Check

Before a user trusts that the Broker Server s/he believes the Peer process will be registering with is the right one, Peer instances must either be configured with `STRICT_CHECKS` enabled, or a pre-registration certificate

check should be done so a user can manually ensure that the certificate being issued by the Broker Server for encrypting authentication routines is valid.

This is analogous to best practice of checking the certificate of a web server prior to proceeding with logging into the service it offers. Users should view the certificate details and confirm that the certificate was issued by a trusted authority that confirms the identity of the web server.

The difference between web services and the VLAN/MP2P software with `STRICT_CHECKS` disabled is that, as part of the authentication process, users do not have the chance to check the certificate before the Peer will use the public key to encrypt the credentials and transmit them to the Broker Server.

To combat this situation, the Peer process can perform a pre-registration certificate check to give them the chance to confirm that the certificate that the Broker Server issues is valid so that the user can trust that the server is the one they expect it to be.

This pre-registration check does not necessarily need to be performed if the Peer instance has `STRICT_CHECKS` enabled as the Peer component process will automatically perform this check for the user by referencing the issuer of the certificate against some pre-trusted CA certificates (as is done with web servers and browsers).

7.2.3. Peer Overload Protection

Peer Overloading is an attack on a Peer but comes via the Broker Server where an unscrupulous user discovers the properties (hardware address, encryption key, VLAN ID and VLAN IP address) of a remote Peer that has already registered with a given Broker Server.

The user then attempts to impersonate the Peer by overwriting the registered public IP address and UDP port of the the original Peer with their own and thus receiving traffic that should have gone to the original Peer.

While it is important that legitimate users of the established VLAN keep their Peer properties and encryption keys secret to help prevent this type of attack, the Broker Server can enforce additional checks to ensure that the original registered public IP address of a Peer does not change.

To help with the issue of IP mobility in relation to this check, the Peer might be asked by the Broker Server to re-authenticate if it detects a change in origin of a Peer. The Peer must honour this by re-registering with appropriate credentials once again to ensure that the Peer is legitimate.

Legitimate users should also be encouraged to change and re-share their encryption keys as often as possible to help ensure security levels are maintained.

8. Component Service Protocol

Equally as important as the primary role of each component of VLAN/MP2P is the ability to be able to figure out what the components are doing, what information they are holding and how to get them to bend to the user's will without having to interrupt services with restarts.

Each component of the software (the Broker Server process and the Peer process) establishes a Component Service Protocol (CSP) TCP port 7711 on the local loop-back adapter allowing the user or local third party software to interrogate or control the state of the process.

For example, the user may wish to:

- retrieve a list of Peers that their local Peer instances are aware of
- obtain detailed information about a single remote Peer instance that can be used for specific application routines and/or services
- temporarily suspend a given Peer instance making it temporarily unavailable to remote Peers
- clean up and terminate the process altogether.

The CSP port is only established on the local loop-back address as it should only be interfaced by the local user usually using locally installed software that understands the protocol.

Essentially, CSP is similar to that of SMTP or HTTP in that the protocol is implemented in plain text and could just as easily be interfaced by a telnet client than it could through an application specifically designed to understand CSP. The main difference is that CSP uses a request/response method based on simple XML and not a request/response method using specific structured formatting.

Authentication and encryption is not implemented in CSP because the assigned service port only listens on the local loop-back adapter interface (127.0.0.1) so it can be assumed to be secure as access can not be made beyond the physical machine.

The Component Service Protocol can also be considered as read-only in that no Peer instance settings can be dynamically adjusted through the service port and nor can information be uploaded into the Peer, however it is possible to stop and start the Peer process and/or suspend and resume individual Peer instances, although this is done through Peer control and not through Peer status interrogation.

Nothing need be done to initialise the CSP protocol once connected to the service port and instructions or requests can be made immediately which initiates a single request/response session. Multiple request/response sessions can be handled in sequence on a single connection so there is no need to have to close the connection until all sessions have been handled.

The CSP client will always begin a session with a request and the component process will end that session with a response, but will not close the connection. A request is formatted using basic XML and must be encapsulated in a `<request>` root element and contain at least a `<retrieve>` or `<control>` element along with some optional elements to complete the request. Listing 8.1 shows an example of a properly formatted request.

```
<?xml version="1.0"?>
<request>
  <retrieve>peer list</retrieve>
  <community>Community Name</community>
</request>
```

Listing 8.1 – Example of a Properly Formatted CSP Request.

There is no need to understand the detail of listing 8.1 as that discussion is beyond the scope of this document. For now, it is simply important to illustrate how simple it is to communicate through CSP.

As mentioned, requests must contain at least a `<retrieve>` or `<control>` element containing either the objects to be retrieved or the instruction to be executed by the component process. All other included elements depend on which of these two elements are included and the content of the element.

Each request is always followed up with a response from the component to which the request is sent and is similarly formatted as outlined in listing 8.2. Each response will contain the original `<retrieve>` or `<control>` element from the request along with the relevant requested information or status output resulting from the particular request.

```
<?xml version="1.0"?>
<response>
  <retrieve>peer list</retrieve>
  <community>Community Name</community>
  <body>
    <peer name="Peer A" ipaddress="10.2.1.6">
    <peer name="Peer B" ipaddress="10.2.1.7">
    ...
  </body>
</response>
```

Listing 8.2 – Example of a Properly Formatted CSP Response.

Obviously, errors can and do occur and are issued from the component using the same XML elements that describe the error as demonstrated in listing 8.3.

```
<?xml version="1.0"?>
<response>
  <retrieve>peer list</retrieve>
```

```
<community>Community Name</community>
<error>
  <code>XXX</code>
  <type>FATAL</type>
  <brief>Unknown Community Name</brief>
  <description>
    The supplied community name could not be
    identified by the Peer. Ensure that the
    community name in the request is correct
    and try again.
  </description>
</error>
</response>
```

Listing 8.2 – Example of a Properly Formatted CSP Error Response.

The Broker Server and Peer CSP services offer the same set of requests and controls however this will most likely change over time as new capabilities are added to the Peer component. XML provides the ability for new services and controls to be added without much work.

The CSP services support unlimited client connections which do not hinder the performance of the Peer's primary role of handling network traffic as the main CSP service is run in a parallel thread to the traffic handler. Additionally, each client connection is threaded to ensure that timely responses are made to each client connected to the service port – again with out hindering the performance of the main traffic handler.

For a detailed discussion of the CSP for Broker Servers and Peers, refer to the CSP Developer's Reference Manual.

9. What You Wont Find in VLAN/MP2P

In a word – compression. VLAN/MP2P currently performs well enough that the introduction of packet compression would literally be incapable of adding any additional performance benefit. The reason for this is that the bandwidth latency benefit gained by compression would be offset by the time it takes to both compress and then decompress the packet between reading it from the local TAP interface and writing it to the remote TAP interface.

Testing has demonstrated no noticeable degradation of service performance when the software is in use as compared with using the same service natively. In some instances, performance is improved as a direct benefit of being able to use P2P communications in place of VPN or port re-routing. The added bonus is the improvement in security and the removal of the need to reconfigure firewalls.

This does not mean that users of VLAN/MP2P cannot head the Peers with

WAN accelerators such as those provided by Riverbed Technology.

10. Paper Conclusion

VLAN/MP2P provides numerous benefits over traditional VPN infrastructure all of which have been discussed throughout this paper. A summary is provided here:

- Direct Peer-to-Peer communication without compromising security or reconfiguring firewalls.
- Each node running the Peer process has the ability to instantiate multiple Peer instances using the one UDP port reducing resources consumed without degrading performance. This also means a single node can be a member of more than one VLAN community at once using a single process.
- P2P is performed at the network level (effectively Layer 2) so that any application running above this layer can benefit from P2P techniques without the need to re-write a single line of code. New applications can be developed without the need to consider P2P integration.
- UDP packets are encrypted prior to leaving the Peer instance and decrypted on arrival using a 128bit pre-shared key with 64bit initialisation vector. The popular and trusted OpenSSL encryption API (EVP) has been used.
- Benefits of VLAN technology have been “borrowed” from traditional hardware devices and introduce these benefits into a wider area community network. Support for broadcasting and multi-casting is completely supported.
- VLAN/MP2P can be wholly hosted by a single company for its own use. Both the Broker Server and Peer software parts are provided in the solution so control vests completely with the organisation deploying the technology. Traffic is never routed through third party servers.
- Implementation can be performed solely by the end user. Network Administrators need only deploy the Broker Server and ensure it remain in service. Users then generate VLAN names and keys which are then shared amongst themselves and allocate themselves a unique IP address.
- Provides a means for nodes behind a dynamically assigned public IP address to be reach at a static, private IP address regardless of remote Peer locations.

- 1 Wikipedia, Virtual Private Network (a definition),
<http://en.wikipedia.org/wiki/VPN>.
As at October, 2008.
- 2 Wikipedia, Virtual LAN (a definition),
<http://en.wikipedia.org/wiki/VLAN>.
As at November, 2008.
- 3 IETF, Layer 2 Tunneling Protocol “L2TP”,
<http://www.ietf.org/rfc/rfc2661.txt>.
August, 1999.
- 4 L. Deri et al., N2N: A Layer 2 Peer-to-Peer VPN,
<http://luca.ntop.org/n2n.pdf>.
2007-08.
- 5 B. Ford et al., Test Results: P2P Communication Across NAT,
<http://www.bford.info/pub/net/p2pnat/>.
February 17th, 2005.
- 6 B. Ford et al., Test Results: P2P Communication Across NAT,
<http://www.bford.info/pub/net/p2pnat/>.
February 17th, 2005.
- 7 The OpenSSL Project Website, Secure Sockets Layer & Transport Layer Security,
<http://www.openssl.org/>.
As at November, 2008.
- 8 Network Security with OpenSSL.
Symmetric Cryptography: Encrypting with the EVP API.
J. Viega; M. Messier; P. Chandra,
O'Reilly Media, Inc.
June 17th, 2002.