Virtual LAN Management Protocol (VLMP)
Draft RFC

David R. Cheriton
Stanford University and Granite Systems, Inc
cheriton@cs.stanford.edu

October 20, 1995

1 Status of This Memo

This memo is a draft specification of VLMP, a MAC- or datalink-level protocol for exchanging virtual LAN information between switches on an extended LAN such as switched/bridged Ethernet and other network technologies using Ethernet addresses such as FDDI. Distribution of this memo **will** be unlimited. Currently its distribution is restricted until reviewed and revised further. The protocol and document are still open to significant changes.

2 Introduction

A *virtual LAN* is a broadcast domain, i.e. a collection of endstations interconnected by one or more switches so that they can communicate as though they were connected by one physical broadcast LAN.

Virtual LANs allow the logical topology of interconnection to be separate from the physical topology. For example, an IP subnet [4] may span several different Ethernet switches. Ports and endstations on the different switches can communicate as though on the same "cable", transparent to the IP layer, just as with a bridge. However, ports and endstations on the same switch but associated with different virtual LANs do not see each others broadcast traffic. In some implementations, endstations in different virtual LANs are also precluded from communicating directly at layer 2. The existence of other virtual LANs on the same switches is also transparent to the endstations on this virtual LAN except possibly for performance. Unlike a bridge or hub, endstations on the same switch but in different virtual LANs cannot communicate with each other. A virtual LAN spanning two or more switches is referred to as a *distributed virtual LAN*.

Switches implementing distributed virtual LANs must communicate to provide a consistent handling of virtual LAN communication across the switches. (Alternatively, the network administrator must manually configure each switch with fully consistent information about all the virtual LANs in which it participates.) An industry-wide protocol and virtual LAN model is required to allow switches from different vendors to interoperate without this manual mechanism.

The *Virtual LAN Management Protocol* (VLMP) is a protocol for communicating virtual LAN information between switches in a vendor-independent form. It allows switches from different vendors to interoperate even if the endstation or port membership in virtual LANs is specified in different ways on each switch.

The basic VLMP model is as follows. A switch identifies each virtual LAN by character string name. For each virtual LAN, there is a *group* of switches identified by the name of the virtual LAN whose members consist of those switches that need to forward packets for that virtual LAN. VLMP includes a group membership protocol portion that is used by a switch to join the group corresponding to a virtual LAN for to each virtual LAN the switch handles. VLMP also specifies a multicast call for disseminating virtual LAN membership information at the MAC address level. Optionally, a switch can use VLMP to query another switch to determine the virtual LANs to which a specified MAC address belongs. Finally, VLMP allows a switch to invalidate the virtual LAN information in other switches.

VLMP is specified as an RPC protocol, first as a procedure interface and then as a packet format using ONC RPC [9] and ONC XDR [8]. This approach follows that taken with EGMP [1] and has similar advantages. That is, it allows the protocol to be generated by an RPC stub generator[1]. It uses a standard familiar packet format to support versions and data representations, and it imposes a procedural structuring on the protocol.

The next section elaborates on the basic model of VLMP and the procedure interface. Section 4 describes the protocol in terms of conventional PDUs.

3 VLMP: Model and RPC Specification

VLMP is based on a model described below. It is specified at the top level as a set of remote procedure calls.

3.1 VLMP Model

In the VLMP model, *each Ethernet address is associated with one or more virtual LANs.* This mapping may be specified by an explicit list of MAC addresses corresponding to a virtual LAN, indirectly by the association of the address with a port which is explicitly specified as part of a virtual LAN, or by association of the address with a virtual LAN through a higher-level mapping of network or routing domain to virtual LAN. As an example of the latter, an Ethernet address may be associated with a particular IP subnet which is mapped to a virtual LAN corresponding to all IP endstations with the same subnet number. This same MAC address may also be associated with other virtual LANs.

Each virtual LAN is identified by a variable-length character string name. For example, "defaultVLAN" is the name of the virtual LAN corresponding to the default setup of a switch or bridge not supporting virtual LANs. Virtual LAN names can carry semantics by various conventions. For example, a virtual LAN corresponding to IP subnet 36.249.0.0 would be called "ip/36.249.0.0". Similar standard prefixes are used for IPX and Appletalk. A name is restricted to a maximum of 128 printable non-whitespace 7-bit ASCII characters.

A switch is assumed to know the virtual LANs associated with the Ethernet address of a device that is directly connected to the switch. This switch is referred to as the *primary* switch for the

[1]This may require some modifications to the standard RPC run-time library to allow it to run at the Ethernet frame level, rather than at the UDP level

address. (A MAC-level device can be connected to a LAN segment with multiple switches attached. In this case, there can be multiple primary switches.)

A switch joins the switch group associated with each virtual LAN that it forwards packets for, for each port from which it can receive packets for this virtual LAN. A switch forwards packets for a given virtual LAN if it is the primary switch for endstations/ports in this virtual LAN or it is a transit switch for one or more switches that are members of the switch group corresponding to this virtual LAN.

In the general case, the switch joins this group on each of its ports for which this virtual LAN traffic is allowed. For each port, a switch records the membership requests it receives from other switches on this port. This information is used to determine the forwarding of a packet sent by a source in this virtual LAN when the packet is multicast or sent to the destination that is not known. Using the membership information, a switch can automatically determine the neighboring switches to which it should forward packets for a given virtual LAN. This protocol also notifies multiple switches in parallel on a LAN segment, allowing a single membership call per LAN segment, the same as EGMP.

Each switch periodically disseminates the membership information (i.e. the list of MAC addresses) for virtual LAN vl on each port it has received a join request for virtual LAN vl. This information includes both the information it has about local endstations as well as the membership information it has received from other switches for this virtual LAN. The latter information is disseminated on all ports with switch members in the virtual LAN other than the port on which it was received. With this mechanism, the virtual LAN information is disseminated across the spanning tree of each virtual LAN to all switches that handle that virtual LAN. With a link using the 802.10-based encapsulation and tagging approach, the switch can instead simply disseminate the SAID identifier value associated with each virtual LAN.

Optionally, a switch can query another switch for the list of virtual LANs in which an Ethernet address participates. The call returns a list of virtual LAN names. These virtual LAN names can then be mapped to descriptors for the corresponding virtual LANs local to the requesting switch. These records are used together with destination address and possibly protocol packet type to determine packet delivery within this switch. If the all-zeros Ethernet address is specified, the switch returns all the virtual LANs it handles.

Each switch caches the virtual LAN names corresponding to a given MAC address based on the disseminated membership information and possibly callbacks. To limit the inconsistency that can arise from this information changing in the primary switch, VLMP provides an update notification operation that allows one switch to inform others potentially caching the virtual LAN membership information for a given MAC address that the information has changed.

VLMP calls and responses are multicast, so if there are multiple switches interested in this information, only one request is made in the typical case, using the same techniques as EGMP.

In operation, when a switch receives a packet from another switch, it looks up the virtual LAN for this packet, based on the source address, or, if encapsulated and tagged (as in the IEEE 802.10 proposal by Cisco), based on the SAID. It then uses this information to map the packet to a set of virtual LANs and then delivers the packet to the corresponding ports, the same as a packet generated local to the switch. If the relevant virtual LAN information is missing for a given address, the switch may optionally flood the packet, drop the packet or queue the packet and perform a callback.

In general, the switch does not know whether a packet was sent by another switch or by a directly connected endstation. In this case, it issues the query for virtual LAN information and assumes it is directly connected endstation if it fails to get a response. Section 5.1 discusses using EGMP to concurrently determine if the packet source on the LAN segment is a endstation as well as port modes that can be used to optimize this query.

A switch may optionally re-request virtual LAN membership information if it is otherwise going to drop the packet because the source does not appear to be in any local virtual LANs and the switch has not requested this information recently. This can be implemented by a recent-request cache. The cache saves the response of recent requests. The cache simply returns the last response that was received if the request is in the cache, rather than actually sending to the other switch. The basic lookup mechanism can then just query through the cache on each miss on an address.

With this approach, the indirectly receiving switch has correct and complete virtual LAN information *most of the time*. The exception occurs when a change in virtual LAN membership occurs for a MAC address and the switch fails to receive a notification to that effect before it processes a packet from that MAC address. If the change is to delete a membership, it is possible a packet may be delivered to a particular virtual LAN and endstation when, by strict consistency, it should not be. However, this misdelivery can only occur within a short period of time of the change because the membership information is otherwise invalidated by an `onUpdate` call. The revocation of rights in computer systems in general can rarely occur instantaneously, and this situation is yet one more instance of delayed revocation. For example, the authenticator used in various standard systems, including Kerberos, incorporates a timestamp or lifetime that causes the authenticator to become invalid after some time period. It is often not supported to revoke authorization sooner than the expiration of the timestamp. A similar latency for revocation seems acceptable with virtual LANs. (Conversely, it is rarely appropriate to complicate the protocol and increase overhead to provide faster revocation at the MAC layer than supported by higher-level protocols. After all, a determined endstation can often select an internetwork route from endstation A to endstation B to access some resource even if direct virtual LAN connectivity between them has been disabled.) Recall also that revocation only needs to operate in user-perceived times, not packet times.

If the change is to add a membership, it is possible that a packet is not delivered to a virtual LAN endstation to which it should be delivered. However, if a switch does re-request this information and provide a small cache of recent re-requests to prevent thrashing the sending switch, the probability of this situation becomes very small. One such scenario is where the address is known to belong to one virtual LAN but not to another to which it was recently added, and the second virtual LAN contains other endstations, and the packet was multicast. Then, the receiving switch performs a partial delivery. However, with multiple notifications of the update to the virtual LAN information, the probability of this loss is comparable to the probability of packet drop from other factors in many networks. Therefore, either the packet is recovered by a higher-level protocol, such as transport layer retransmission, or else the higher-level protocol should be tolerant of such packet losses in general. Because the information should be refreshed within the timeout interval, this loss situation should be transient, allowing a retransmitted packet or subsequent packets to get through. Finally, like with deletion, there is merit in allowing some time for this level of authorization change to propagate.

3.1.1 Using (Inter)Network Layer Information

Switches that implement network-layer routing as well as MAC-level switching can use virtual LAN information to automatically configure virtual LAN behavior to match the network layer structure. Using this information, a switch/router can determine whether a packet is deliverable based on its protocol type and its network-layer header. For example, a virtual LAN named as "ip/36.249.0.0" can be automatically interpreted as specifying the virtual LAN of all IP endstations with an IP address on this Internet network/subnetwork. When a packet is received with an IP Ether type and an IP source address on this IP subnet, the switch can restrict the delivery to endstations that are on this same IP subnet. The network layer information eliminates the need to use VLMP to call back to the source.

In this use, the subnet number with zeroes in the variable positions of the address is assumed to uniquely identify the subnet. If the switch can handle IP subnets, the switch is assumed to be able to determine the subnet mask from this subnet name. Otherwise, it may be necessary to encode the subnet mask into the virtual LAN character string name.

3.1.2 VLMP versus EGMP

The focus of VLMP is similar to that of EGMP in the sense that a virtual LAN is effectively a group of ports (from a delivery standpoint) just as EGMP maps a multicast address to a group of ports, possibly distributed across multiple switches. Moreover, VLMP is similar to inter-switch EGMP in defining a group of switches, corresponding to those handling a virtual LAN with VLMP, and corresponding to those handling a multicast address with EGMP. However, this group is naturally identified by the character string name of the virtual LAN with VLMP and by a multicast address with EGMP. Moreover, VLMP manages the group of switches associated with a virtual LAN, not the MAC addresses. It relies on each switch to determine the virtual LANs associated with the MAC addresses that appear in the source address field of each packet that it forwards. VLMP simply supports querying the virtual LAN associated with each MAC address. Thus, in effect, the group of MAC addresses in a virtual LAN is defined in a hierarchical form — at the top level by the group of switches handling the virtual LAN and at the second level, by the local membership determination of the switches.

3.2 Virtual LAN Information

The virtual LAN information is structured as

```
struct VLANInfo {
     MACAddr addr_;
     VLANName nameArray_[MaxNames];
  };
```

The `addr_` field specifies the MAC-level address and the `nameArray_` lists the names of the virtual LAN with which this address is associated. The virtual LAN membership information is structured as

```
struct VLAmemmhip_;
Info {
     VLANName nameArray_];
protected virtual v_;
     MACAddr addr_;
  };
```

The `addr_` field specifies the MAC-level address and the `nameArray_` lists the names of the virtual LAN with which this address is associated.

3.3 VLMP Procedures

VLMP contains the following procedures in its interface.

`void vlanMembership( VLANName vl, MACAddrList eal )` - Notify other switches that the specified virtual LAN contains the Ethernet addresses specified by the MACAddrList *eal*.

`VLANInfo vlans( MACAddr who )` - Return the `VLANInfo` for the endstation identified by `who`. The `MACAddr who` parameter specifies the Ethernet address being queried.

`void onUpdate( VLANInfo vli )` - Update the cached copy of virtual LAN information with new virtual LAN information, `vli`, or invalidate the local copy if `vli` is null. The `VLANInfo` structure consists of an Ethernet address and a list of virtual LAN names, as shown previously.

In addition to the above, VLMP uses the GMP procedures listed below to be a member of a virtual LAN and receive to packets for that given virtual LAN.

`void join( Description desc )`

`void leave( Description desc )`

In this use, the `Description desc` is a a time field followed by a list of virtual LAN names, corresponding to the virtual LANs to which the membership applies. The GMP procedures carry their normal semantics, specialized to virtual LAN names. That is, a `join` call including a virtual LAN name *vln* sent on LAN segment *ls* indicates to all switches on this LAN segment that the calling switch wants to join the group of switches that are handling packets for the virtual LAN identified by *vln*. A `leave` call specifying *vln* indicates that the calling switch wants to be removed from this group. A VLMP interrogator switch per LAN segment periodically issues a leave-all (zero-length list of virtual LAN names), to get all switches on this LAN segment to re-join the virtual LAN groups of interest.

3.4 Role of the RPC System

These procedures are mapped to packets by a remote procedure call system. VLMP uses the ONC RPC and XDR standards to map to packets, except the packet is an Ethernet packet rather than a UDP or TCP packet. The RPC system is also expected to provide authentication and security if that is required in the application domain.

4 VLMP Protocol Description

VLMP uses the Ether type field value allocated for Ethernet-level ONC RPC (yet to be allocated).

All VLMP packets are transmitted to a single well-known (yet to be allocated) Ethernet multicast address referred to as the VLMP address. This address is logically the group address for the (switch) servers handling VLMP. Endstations do not receive or send VLMP packets.

Each Ethernet switch acts as a server for VLMP. The switch does not forward (VLMP) packets addressed to this one address to other LAN segments, so endstations and switches on other LAN segments do not receive VLMP traffic not local to their segment. Normally VLMP traffic is confined to backbone links or networks between the switches, and does not occur on the links purely connecting endstations to the switches.

4.1 VLMP Structs

The format for a `VLANInfo` struct is

```
+------------+-----+-------+---------+-------+----------+-..+---+
| endstation | pad | count | vlanName | count | vlanName |...| 0 |
+------------+-----+-------+---------+-------+----------+...+---+
      6         2      4       count
```

The `count` is a 32-bit unsigned count of the number of bytes in the `vlanName`, followed by `count` bytes of `vlanName`. Zero or more virtual LAN names (count plus characters) can follow, terminated by a zero `count` field.

The format for a `VLANMembership` struct is

```
+-------+---------+-------+--------+-----+--------+-----+-...+---+
| count | vlanName | count | MACAddr | pad | MACAddr | pad | ...| 0 |
+-------+---------+-------+--------+-----+--------+-----+-...+---+
   4       count      4        6       2       6       2
```

The *vlanName* specifies the virtual LAN and the list of *MACaddr*'s specify the endstations in the virtual LAN.

The format for a VLMP membership description byte string is

```
+-------+--------+---------+--------+----------+-..+---+
| delay | count0 | vlanName | count1 | vlanName |...| 0 |
+-------+--------+---------+--------+----------+...+---+
   4        4      count0      4       count1
```

The delay field specifies the time in microseconds before the state really changes, allowing other switches to determine the time to respond, just like with VLMP.

4.2 Structure of the Virtual LAN Name Space

A virtual LAN name is structured with standard prefixes to indicate attributes of the virtual LAN. The current attributes include protocol type and the null prefix. The standard prefixes are:

1. ip - IP-specific virtual LAN.
2. ipx - IPX-specific virtual LAN.
3. at - Appletalk-specific virtual LAN.

(Additional standard prefixes can be defined.) The standard prefixes are separated from the other portions of the name by "/"'s.

By encoding these attributes in the virtual LAN name, VLMP can accommodate new attributes easily. Moreover, switches that do not support an attribute can be configured based on the naming of the virtual LANs to interoperate with switches specifying new attributes, or not, depending on the naming of the virtual LANs. For example, a virtual LAN can be named `ip/36.249.0.0` even if the switch does not support IP subnet virtual LANs, allowing it to interoperate with a switch that does support this type of virtual LAN.

4.3 ONC RPC Description

The ONC RPC description of VLMP is two programs, one for server and one for client, namely: *These RPC descriptions have not been adequately checked or debugged yet.*

```
typedef unsigned long Time;
typedef char VLANName<128>;
typedef unsigned short MACAddr[3];
typedef struct {
    Time        delay_;
    VLANName nameList_<128>;
  } Description;

typedef struct {
    MACAddr  addr_;
    VLANName nameList_<128>;
  } VLANInfo;
typedef MACAddr MACAddrList<128>;

const VLMP_SERVER_PROG = 0x13333333;
const VLMP_CLIENT_PROG = 0x13333334;

#ifdef SERVER_PROG
program VLMP_SERVER {
    version VLMP_SERVER_1 {
        void VLMPPing(void) = 0;
        void join( Time, Description ) = 1;
        void leave( Time, Description ) = 2;
        VLANInfo vlans( MACAddr ) = 5;
      } = 1;
  } = VLMP_SERVER_PROG;

#else

program VLMP_CLIENT {
    version VLMP_CLIENT_1 {
        void VLMPPing(void) = 0;
```

```
        void leave( Time, Description ) = 2;
        void vlanMembership( VLANName, MACAddrList ) = 6;
        void onUpdate( VLANInfo ) = 7;
      } = 1;
  } = VLMP_CLIENT_PROG;

#endif
```

Following ONC RPC conventions, procedure 0 in both server and client is a null "ping" procedure.

"Compiling" this description through the standard `rpcgen` program produces RPC stubs that generate and handle the VLMP packet formats, which are described directly below for version 2 ONC RPC.

4.4 VLMP Packet Formats

VLMP packet format is the ONC RPC request and reply messages that correspond to the VLMP procedure declarations.

The following is the ONC request packet used by the `vlans` call. (ONC RPC uses the term "message" rather than packet, but with VLMP, each message is a separate packet and we are describing packet formats, so we use the term "packet" instead.)

```
|<----- 32 bits ----->|
+---------------------+
|  xid                |
+---------------------+
|  msg_type           |
+---------------------+
|  rpcvers            |
+---------------------+
|  prog               |
+---------------------+
|  vers               |
+---------------------+
|  proc               |
+---------------------+
|  auth_flavor        |
+---------------------+
|  auth_length (0)    |
+---------------------+
|  MACAddr            |
+- - - - - - - - - - -+
|                     |
+---------------------+
```

Following ONC RPC conventions, the packet is sent in big-endian network order.

The fields of this call packet format are described below, with all but `MACAddr` following standard ONC RPC values.

xid - the transaction identifier, incremented on each VLMP call from each source, starting from 1.

msg_type - 0 for call.

rpcvers - 2, current version of ONC RPC.

prog - 0x1f333333 (to be assigned for VLMP)

vers - 1, first version of VLMP.

proc - `vlans`, specified as 5

auth_flavor - the value of 0 for standard `AUTH_NULL`.

auth_length - 0, because of the null authentication.

MACAddr - A 8-byte field containing a 48-bit Ethernet address in the high-order 6 bytes specifying the endstation on which virtual LAN information is being requested.

The following is the VLMP return packet format for a `vlans` accepted call.

```
|<----- 32 bits ----->|
+---------------------+
|  xid                |
+---------------------+
|  msg_type           |
+---------------------+
|  reply_stat         |
+---------------------+
|  auth_flavor        |
+---------------------+
|  auth_length (0)    |
+---------------------+
|  accept_stat        |
+---------------------+  +---------+
|  MACAddr            |  |   low   |
+- - - - - - - - - - -+  +---------+
|                     |  |   high  |
+---------------------+  +---------+
|  count              |
+---------------------+
|  vlanName           |
+---------------------+
|                     |
.       . . .         .
+---------------------+
|  count (0)          |
+---------------------+
```

The fields of this return packet format are described below,

xid - the transaction identifier, matching the call xid to which this is a return message.

msg_type - 1 for REPLY.

reply_stat - 0 for MSG_ACCEPTED (and otherwise it is a rejected message — see below.)

auth_flavor - the value of 0 for standard `AUTH_NULL`.

auth_length - 0, because of the null authentication.

accept_stat - the standard ONC RPC values, namely `SUCCESS (0)`, `PROG_UNAVAIL`(1), `PROG_MISMATCH` (2), `PROC_UNAVAIL` (3) and `GARBAGE_ARGS`(4).

low,high - are only used with `PROG_MISMATCH` (2) to indicate the low and high versions of the program that are supported by the server, as with standard ONC RPC.

MACAddr - Ethernet address of endstation, padded out with an extra 2 bytes to create an 8-byte field.

count - a count of the number of bytes in the following vlan name, or 0, if no more names.

vlanName - "count" bytes of name of a virtual LAN. (Zero or more virtual LAN names may follow, the last one having a zero "count" field, and zero bytes following.)

The following is the VLMP return packet format for a rejected call.

```
|<----- 32 bits ----->|
+---------------------+
|  xid                |
+---------------------+
|  msg_type           |
+---------------------+
|  reply_stat         |
+---------------------+
|  reject_stat        |
+---------------------+
|  low                |
+---------------------+
|  high               |
+---------------------+
```

The fields of this return packet format for rejected call are described below,

xid - the transaction identifier, matching the call xid to which this is a return message.

msg_type - 1 for REPLY.

reply_stat - 1 for MSG_DENIED. (and 0 for an accepted message, as described above.)

reject_stat - the standard ONC RPC values, namely `RPC_MISMATCH (0)` or `AUTH_ERROR` (1).

low - used with `RPC_MISMATCH` (0) to indicate the lowest supported RPC version number. With a `reject_stat` of `AUTH_ERROR`, this field is interpreted as the reason for authentication failure, using the standard ONC RPC values of `AUTH_BADCRED` (1), `AUTH_REJECTEDCRED` (2), `AUTH_BADVERF` (3), `AUTH_REJECTEDVERF` (4), and `AUTH_TOOWEAK` (5). (The use of `low` is described in this form rather than introducing yet another return packet format, as would be required to be totally consistent with ONC RPC conventions.) VLMP implementations need not initially support authentication so `AUTH_ERROR` should not occur. However, an implementation should recognize when it does arise and return an indication to the higher-level client software. Authenticated memberships may be required on some networks in the future.

high - only used with `RPC_MISMATCH` to indicate the highest supported RPC version number.

The following is the ONC request packet used by the `vlanMembership` call.

```
|<----- 32 bits ----->|
+---------------------+
|  xid                |
+---------------------+
|  msg_type           |
+---------------------+
|  rpcvers            |
+---------------------+
|  prog               |
+---------------------+
|  vers               |
+---------------------+
|  proc               |
+---------------------+
|  auth_flavor        |
+---------------------+
|  auth_length (0)    |
+---------------------+
|  count1             |
+---------------------+
|  vlanName           |
+---------------------+
|  count2             |
+---------------------+
|  MACAddr            |
+- - - - - - - - - - -+
|                     |
+---------------------+
|  MACAddr            |
+- - - - - - - - - - -+
|                     |
```

```
+---------------------+
.                     .
.       . . .         .
.                     .
+---------------------+
|  MACAddr            |
+- - - - - - - - - - -+
|                     |
+---------------------+
```

The fields of this call packet format are described below, omitting the standard ONC RPC values, which are the same as for the previous request packet. (The "proc" value specifies 6 for `onUpdate`.)

count1 - a 4-byte field indicating the length of the virtual LAN name.

vlanName - "count1" bytes of name of the virtual LAN whose membership information is provided.

count2 - a count of the number of bytes in the following MAC address list.

MACAddr - a MAC address that is included in the virtual LAN specified by *vlanName*.

The following is the VLMP request packet for the `onUpdate` call.

```
|<----- 32 bits ----->|
+---------------------+
|  xid                |
+---------------------+
|  msg_type           |
+---------------------+
|  rpcvers            |
+---------------------+
|  prog               |
+---------------------+
|  vers               |
+---------------------+
|  proc               |
+---------------------+
|  auth_flavor        |
+---------------------+
|  auth_length (0)    |
+---------------------+
|  MACAddr          |
+- - - - - - - - - - -+
|                     |
+---------------------+
|  count              |
+---------------------+
|  vlanName           |
```

```
+---------------------+
|                     |
.       . . .         .
+---------------------+
|  count  (0)         |
+---------------------+
```

The fields of this call packet format are described below, omitting the standard ONC RPC values, which are the same as for the previous request packet. (The "proc" value specifies 6 for `onUpdate`.)

MACAddr - an 8-byte field containing a 48-bit Ethernet address in the high-order 6 bytes, specifying the endstation to which the virtual LAN membership information applies.

count - a count of the number of bytes in the following vlanName, or 0, if no more names.

vlanName - "count" bytes of name of a virtual LAN. (Zero or more virtual LAN names may follow, the last one having a zero "count" field, and zero bytes following.)

TBD: Description of packet formats for the GMP portion of VLMP.

4.5 Protocol Procedures

VLMP uses the basic model of GMP (see Appendix), specialized to groups of switches identified by virtual LAN names. For each port, a switch joins the group of switches handling a virtual LAN *vl* if it has other ports that receive packets for that virtual LAN, either because of directly connected endstations in this virtual LAN, or because of a switch on the LAN segment connected to this port that is handling packets for this virtual LAN. The specialized form of GMP is used by the switches to join these groups.

VLMP periodically multicasts virtual LAN membership information for virtual LAN *vl* specified as a list of MAC addresses on every port that it has received a join request for this virtual LAN.

VLMP also provides a `vlans` call that the client invokes on the VLMP multicast address[2]. The server responds with a return packet containing the virtual LAN name for the requested Ethernet address, or else indicating a problem with the request. When the virtual LAN membership of an address changes and there has been recent requests for this information, the server sends an `onUpdate` datagram call to the VLMP multicast address. This call can specify the revised virtual LAN information for that address.

The following descriptions elaborate on these basic procedures.

4.5.1 The `join` Call

A switch invokes a `join` call when it initializes and has one or more virtual LANs in its configuration. The call specifies a list of virtual LAN names, one for each virtual LAN group that the switch wishes to join. It reinvokes this call when a new distributed virtual LAN is added or on timeout of the previous membership, as indicated by receiving a `leave` call.

[2]This multicasting of the request is an alternative to using the multicast address to determine the unicast address of the sender and then sending to that unicast address. The low frequency of VLMP calls and the benefits of other clients monitoring these calls makes multicasting the call the preferred approach.

The delay field of the descriptor parameter is specified as 0, to indicate indefinite membership until a leave is requested. In the preferred implementation, a virtual LAN may be specified as local to a switch, eliminating the need to join the corresponding group on each port.

4.5.2 `join` Call Handling

A switch receiving a VLMP `join` call on port p checks whether port p is specified as allowing connection to the specified virtual LANs. If so, it flags port p as connecting to the specified virtual LANs (for broadcast, multicast and locating unicast addresses) and it sends a response to the call.

If it does not implement any of the specified virtual LANs or port p is not enabled to export these virtual LANs, it ignores the request. (It would seem better to provide a negative response in the latter case, but there may be other switches on the same LAN segment that are able to accept the request.)

4.5.3 The `leave` Call

A switch issues an `leave` call on a port to remove itself from one or more virtual LAN groups on the attached LAN segment.

The `leave` call can have a null list of virtual LAN names, in which case it is viewed as specifying all virtual LAN names, and is referred to as the leave-all call.

A particular switch per LAN segment, the *interrogator* switch, periodically issues a leave-all on this LAN segment. This call causes switches on this LAN segment that need to continue to belong to one or more virtual LAN groups to rejoin these groups.

If it receives no `join` call specifying some virtual LAN vln after the leaveAllDelay time period, the switch issues another `leave` call specific to this virtual LAN and waits for the leaveDelay time period. If no `join` call is received before this time period ends and there are no locally connected endstations, the switch removes this port from the virtual LAN. This rejoining activity garbage-collecting old memberships that are no longer wanted.

The election of the interrogator is handled the same as EGMP, trying to operate with just one interrogator per LAN segment. That is, the designated switch (or bridge) for the spanning tree algorithm functions as the interrogator. This choice synchronizes the virtual LAN information with any changes to the network topology. Use of VLMP is allowed with other routing approaches besides the standard spanning tree, but the details of this operation are relegated to further study.

4.5.4 `leave` Call Handling

When a switch receives a `leave` call, it determines the set of virtual LANs specified in the leave call that correspond to groups to which this switch belongs. It then invokes one or more `join` calls to reaffirm its memberships for the specified virtual LANs. If the call is a leave-all call, it rejoins all of the groups it is still interested in on that port.

The switch behavior uses the same techniques as IGMP to avoid join call implosion, just applied at the datalink layer. That is, in more detail, when a node receives a leave call designating one or more virtual LANs to which it is interested:

1. It starts a join call timer set to a randomly-chosen value between zero and leaveAllDelay microseconds, where leaveAllDelay is a timeout value in microseconds chosen by the interrogator.

When the timer expires, a join call packet is transmitted containing the list of the virtual LAN names in which the switch is still interested that were listed in the leave call (or all such groups if a leave-all call) that have not already been rejoined to on the same LAN segment since the leave call was received. Thus, join calls are spread out over a leaveAllDelay microsecond interval instead of all occurring at once.

2. If a node hears a join call for a virtual LAN to which it belongs on that network, the node marks this virtual LAN as rejoined.

The switch does not forward packets destined to the VLMP address between LAN segments of the switch, *i.e.,* different ports. Thus, in the normal case, only one join call is generated per leave call per joined multicast group on each LAN segment connected to the switch, namely the one generated by the node whose delay timer expires first.

The above description covers the switch behavior as a VLMP client, requesting membership in virtual LAN groups. The switch also responds to a leave call as a server that is maintaining membership information requested by other switches. The behavior associated with this role is described below.

A switch checks the set of virtual LAN memberships that it has recorded for this port (to indicate that it should treat the attached LAN segment as part of these virtual LANs). For each such membership that is mentioned in the leave call (or all memberships if it is a leave-all call), the switch records the membership as a candidate for deletion. It then delays for a leaveDelay or leaveAllDelay period of time, depending on whether it was a leave-specific or leave-all request.

If the switch is an interrogator, after leaveAllDelay microseconds, the switch sends a second leave-specific call specifying the virtual LAN names that were listed in the original leave call that have not been mentioned in a subsequent join on the same LAN segment. If the switch is not an interrogator, it simply waits for this second leave-specific call from the interrogator.

For each virtual LAN for which there is no join in response to the interrogator's leave call after the specified delay period of time, the switch deletes the membership, causing it to no longer regard that port as connecting to other switches handling this virtual LAN.

If a non-interrogator switch fails to receive the interrogator leave call, it elects itself as an interrogator and sends the second leave call itself.

4.5.5 The `vlanMembership` Call

A switch invokes the **`vlanMembership`** call for a virtual LAN vl on each port p that has switch members in the correponding switch group (as reported through VLMP group membership). The call provides the list of MAC addresses that the switch knows to be in this virtual LAN. This list includes MAC addresses that the switch learned from other switches by this VLMP procedure. Ideally, it does include membership information that was learned from vlanMembership calls received on port p.

4.5.6 `vlanMembership` Call Handling

When a switch receives a vlanMembership call, it looks up the virtual LAN name vln specified in the call. If it does not handle this virtual LAN, the rest of the information may be ignored and the call is dropped. (The switch may choose to retain this information so that it can quickly drop packets that it receives that are for virtual LANs it does not handle.)

If the switch does handle the virtual LAN designated by vln, it updates its records of MAC addresses in this virtual LAN from those specified in the call.

If the switch contains an address that has not been reconfirmed recently by messages from the switches that holds these MAC address mappings, the address can be designated as scrap and used for a variety of outdoor projects.

4.5.7 The `vlans` Call

The client calls the `vlans` procedure to retrieve the virtual LAN information about a specified endstation address. The switch typically invokes this call upon receiving a packet with a source address for which the switch is not caching the virtual LAN information.

The client must re-request this information if it is invalidated by an `onUpdate` call from the providing switch. A switch responds to the request if it has the valid virtual LAN information being requested. This is typically just the primary switch for the specified Ethernet address (or one on the route to the primary switch) so only one or a small number of responses are expected and there is thus no need for random delays to delay with a large number of potential responses, such as in IGMP and EGMP. The client may also receive responses to other requests by other switches for the same address and update its virtual LAN information for this endstation address accordingly.

On receiving a response to a `vlans` call[3], if the switch is interested in this endstation virtual LAN information, it updates its local cache of this information using that contained in the packet.

4.5.8 `vlans` Call Handling

Upon receiving a `virtuaLANs` call, a switch determines whether it is a primary switch for the endstation address or the request was received on a port other than the one it would use to send to the endstation. If neither are true, it ignores the request. If the address is unknown, the switch can use the EGMP unicast to locate the address on its ports.

If it is a primary switch for the address, it looks up the virtual LAN information about this endstation address and returns it to the requesting switch. It also notes that it has provided this information. Then, if this information is changed within this period of time, it invokes a multicast `onUpdate` call.

If the primary switch does not have complete information on the virtual LAN membership, then the local delivery of packets may be incomplete or incorrect as well. For example, the switch may know of the MAC address from configuration information, but not know the virtual LAN(s) for this MAC address until it has mapped this MAC address to a port. However, this incompleteness should only arise on initialization of the switch and persist for only a short period of time, assuming the endstations are marginally active.

If the switch receives the request on a port other than the one it would use to send to the endstation, the switch responds to the VLMP query just like a primary switch if it has the virtual LAN information. (A switch does not forward packets sent to the VLMP address so any VLMP request a switch receives must be generated local to one of the LAN segments to which this switch is directly connected.) Given that a query is normally in response to receiving a packet, the responding switch should have just earlier forwarded this packet, and so should have valid virtual LAN information on

[3]The switch identifies such a response if it did not send it by creating a call record for the call if it receives a multicast `vlans` call sent by another switch, and it is interested in the response.

the source address that it can pass on to the requesting switch. (The switch can otherwise ignore the request or else request the information from a switch closer to the primary switch in response to the query.) It then flags its information to indicate that it needs to forward notifications of update on along the spanning tree away from the source address (or it can maintain more detailed information of which switch or LAN segment needs to be notified). In this way, the intermediate switch acts as a cache of the virtual LAN information to the switch that is further removed from the (source) address. It effectively short-circuits the callbacks for virtual LAN information as a packet traverses multiple hops. This callback on each hop with potentially a packet drop at each such stage is assumed to work OK because other routers behave like this, and with suitable memory in each switch, this behavior should arise infrequently.

This provision is for a non-primary switch to respond allows VLMP to support indirect multi-switch operation in which the destination switch is not directly connected by a backbone to the source switch.

A switch is assumed to know which of its ports forward to other ports based on the standard IEEE spanning tree algorithm or some other proprietary routing algorithm. There can be a spanning tree for virtual LAN as well. Some mechanism of this nature is required between switches to avoid packet forwarding loops, especially for broadcast.

A switch can be configured to either forward or not forward (transit) traffic between switches for virtual LANs that it is not handling directly itself. In the normal case, the switch needs to be configured to pass virtual LAN traffic for such virtual LANs. Otherwise, virtual LANs on two switches connected by a third that does not have these virtual LANs are not allowed to communication across the third switch. A switch should simply provide options to control and help manage this traffic.

4.5.9 `onUpdate` Call

When a switch changes the virtual LAN membership for an endstation and it has recently responded to one or more VLMP `vlans` calls for this address on port p, the switch multicasts an `onUpdate` call out this port specifying this MAC address and its associated virtual LAN names, for each such port p. The `onUpdate` call is retransmitted shortly after the first call to guard against single packet drop. The delay before retransmission is recommended to be 30 milliseconds.

A switch may invoke the `onUpdate` call on all ports if it does not record on which ports it has received `vlans` calls for this address. It may also invoke this call on update of virtual LAN information for any MAC address if it does not record the MAC addresses for which it has recently provided information.

A call with a zero-length list of virtual LAN names is specified to request that the switch(es) invalidate their virtual LAN information for this MAC address.

The broadcast address in the `onUpdate` call is specified to mean invalidate all the virtual LAN information that has been learned from `vlans` calls to this switch. In this case, a zero-length virtual LAN name list is specified.

4.5.10 `onUpdate` Call Handling

When a switch receives an `onUpdate` call, it looks up the address in its virtual LAN information cache. If it has an entry for this information, it updates its cache with this information. If the virtual LAN name list is zero-length, it simply invalidates the virtual LAN information so it will callback on the next reception of a packet from this MAC address or until the membership information is again disseminated by the relevant switch.

If there is no entry in the cache for this address, the switch can ignore the call or heuristically add the information to its cache, an implementation choice.

If the call specifies the broadcast address, it invalidates all virtual LAN information that it has learned from the calling switch. It can invalidate additional virtual LAN information if it does not record which information was learned from this switch, versus other switches or even other LAN segments.

There is no attempt here to deal with old delayed packets such as one that provides older information than that in the cache because packets are not generally reordered on a LAN segment.

4.6 Limits on Security

VLMP does not directly provide support for secure communication in the sense that any switch can report virtual LAN membership for a endstation that is not valid, and an intruder can respond to VLMP queries. An Ethernet sender can even use a false source address.

It is expected that a secure version of ONC RPC or other RPC system would be used for a secure version of VLMP. This secure layer would provide authentication of the switch and protection against impostors. The authentication of the switch could serve as a basis for determining whether to trust the switch for the virtual LAN information on a particular virtual LAN.

However, it seems far better to use secure communication at the application level RPC and only rely on the virtual LAN mechanism to provide basic traffic management. With this approach, impostors providing false virtual LAN information can only deny service by stopping traffic that should go through and overloading endstations or LAN segments with traffic that should not be allowed. In both these cases, network monitoring tools should be able to quickly identify the problem, allowing the network administrator to take corrective action without any compromise of security.

4.7 Use with 802.10 Approach

The approach of using the 802.10 SAID field, as proposed by Cisco, to identify virtual LAN membership can use VLMP to communicate the meaning of the 32-bit SAID field between switches. That is, a switch supporting the 802.10 approach can use VLMP to query the sending switch for the virtual LAN name(s) associated with the value of the SAID field by embedding this value in the Ethernet address parameter of this call. The embedded value is a multicast address with the SAID value in the low-order 32-bits. The high-order 16 bits have the multicast and local address bits set, but are otherwise all zero. A switch using VLMP can then interoperate with a switch using the 802.10 encapsulation scheme if it can send and receive packets in this encapsulated form. (It may also have to store an indication of the virtual LANs on that port that are using encapsulation.)

The same approach as described directly above for interpreting an Ethernet address as a virtual LAN specifier can also be used to pass the SAID value(s) associated with a particular virtual LAN.

If a switch is already learning source addresses from packets received from other switches, the 802.10 approach simply avoids having a field associated with every (source) address record inside the switch that points to the virtual LAN membership. In effect, the SAID approach carries this pointer in the packet at the cost of the extra encapsulation.

The 802.10 encapsulation is more effective when there are designated backbone ports connecting to segments with no endstations, and the switches use what some have called negative filtering. In this approach, the switch simply sends what is not known to be local to the backbone and drops what it receives from the backbone which it does not know to be local. The switch does not learn addresses on the backbone at all. In this case, the SAID provides an identifier for the virtual LAN without the switch having to record all backbone MAC addresses. However, for shared-memory switches, maintaining a record for each backbone switch is relatively inexpensive in space for even quite large configurations, and avoid the cost of 802.10 encapsulation.

A packet tagging/encapsulation scheme such as the proposed IEEE 802.10 scheme is also motivated by situations in which one endstation uses the same MAC address on multiple virtual LANs. For example, the user may bridge between two virtual LANs for non-routable protocols such as Netbios and DEC LAT while routing other protocols such as IP, IPX, etc. Similarly, some server nodes configured with multiple Ethernet interfaces may use the same MAC address on each port.

We regard the former case to be solvable using protocol-specific virtual LANs, with the Netbios virtual LAN including all the ports while the IP virtual LAN on the switch can subdivide the ports to limit the range of broadcast. This structure of overlapping virtual LANs allows backwards compatibility with old protocols while avoiding the expense and performance costs with having packets go through a separate bridge component. With regards to the multiple uses of the same Ethernet address, it seems feasible to just return to, and enforce more carefully, the original uniqueness requirement on Ethernet addresses, rather than compromising the switch design because of this misuse of MAC addresses.

5 Implementation

VLMP has both a client and a server implementation, although both are generally on a switch. (The client implementation may also be appropriate for a network management tool, for extracting virtual LAN information from switches.)

5.1 Client VLMP

Client VLMP is implemented at an interface level that handles the source address lookup when a packet arrives on an interface. Rather than just looking up the address and adding it to the port mapping table if not present as in a conventional learning switch, a VLMP switch also checks the virtual LAN information it has for the endstation. A VLMP query is issued to get the missing information.

In the preferred implementation, a port can be in one of several modes that dictates how it determines virtual LAN information:

backbone - query using `vlans` for each source address for which it does not have the virtual LAN information. This port designation means that all packets arriving on this port are forwarded by another switch.

endstation - never query using VLMP. Determine virtual LAN information from local rules on virtual LAN configuration. This port designation means that all packets arriving on this port are received directly from an endstation.

mixed - Use VLMP to query for virtual LAN information, but also issue an endstation EGMP query on this unicast address (a EGMP leave-specific call to solicit a join call). A response to the latter indicates a directly connected endstation whose virtual LAN information should be determined by local rules. A response to the former indicates the information is being provided by a switch. This mode means that both switches and endstations are present on the LAN segment connected to the port.

The switch should not receive both types of response even when the LAN segment contains both other switches and the endstation because a switch does not respond to requests on the same port as it would send to this endstation. If it does receive both types of response, it assumes the endstation is directly connected and uses the local rules for determining membership in virtual LANs.

A switch may assume that a port is an endstation port until it receives one or more VLMP join operations, indicating the presence of other switches. A port must be explicitly configured as a backbone port by the network administrator.

In the case of a miss, the switch can either buffer the packet until a response is received or drop the packet. Buffering the packet is preferred but is not required in the protocol. Dropping the packet may significantly detract from the performance of the protocol if the memory of the switch is limited, limiting the amount of virtual LAN information that is cached, or the number of endstations generating inter-switch virtual LAN traffic is large.

A requesting switch should do the best it can with virtual LAN attributes and virtual LAN memberships that it does not implement. Generally, a switch should deliver to a superset of the endstations if it does not implement such restrictions, such as protocol-specific virtual LANs. With these virtual LAN attributes, the client switch would have to have virtual LANs named with the convention for an attribute that the switch does not support. In this case, it is reasonable to assume that communication was intended by the network administrator that set these names up, and provide the superset. Following the obvious alternative of not using these embedded attributes in the name when not supported by the switch would prevent communication because the names would not match.

5.2 Server VLMP

The server VLMP may be implemented in the switch or associated agent that is managing the switch. It may be implemented at a high-level, well removed from the packet forwarding machinery, but should be able to respond quickly to requests, so the requester is not excessively burdened with buffering packets and so, in the extreme, the virtual LAN information is available when the packet is retransmitted, if the packet is dropped.

In some switch designs, it may be feasible for the packet forwarding mechanism to generate and send the `vlans` call packet in response to a miss yet have a higher-level agent handle the response. The response packet includes the Ethernet address and so is self-contained.

A server should send out two onUpdate-alls to each LAN segment when it reboots. This action ensures that other switches are not continuing to hold virtual LAN information from the switch from a previous incarnation. (For example, the switch may have been halted and rebooted with a new virtual LAN configuration.)

A server needs to implement all of VLMP to be VLMP-compliant.

5.3 Reference Implementation

TBD!

6 Scalability and Performance

VLMP requires that a switch have a record that maps the source (MAC) address of an incoming packet to a list of virtual LANs, with the switch calling back to the sending switch to get this information if this record is not present when a packet with this source address is received. As a configuration using VLMP is scaled, the switches incur a growing space or a time overhead, or both. This section briefly explores the scalability of VLMP.

Considering space overhead first, a switch normally creates a record for the source address of each packet that it accepts. VLMP requires that this record have an additional pointer to a list of the virtual LANs that the address is in. Thus, we consider the VLMP overhead to be at most 4 bytes per MAC address, not the full size of this record. Moreover, in some switch implementations, this record would have a pointer to the list of virtual LANs for the MAC address to allow it to quickly identify the associated virtual LANs on receiving a packet from a directly connected device. For these implementations, the VLMP space overhead is essentially zero. (If a switch defines virtual LANs based on ports, this pointer would not be required. If the switch defines virtual LANs using network layer information, such as IP subnet, then a non-primary switch can also determine the virtual LAN from this network layer information and eliminate the need for both the VLMP information and the callback, at least for types of network-layer traffic that support this form of virtual LAN.)

These MAC address records are normally required even when the switch is using the IEEE 802.10 approach because the switch uses these records to determine a port for the unicast destination address. Thus, even a non-primary switch needs to maintain these records for the addresses of packets it forwards.

The primary time overhead of VLMP is the blocking of a packet and callback to the sender when a packet is received for which there is no local virtual LAN information. This action delays the packet delivery, imposes a processing overhead on both the sending and receiving switch and uses network bandwidth for the callback. However, the dissemination of the routing information should suppress and satisfy most of the hardware people.

If we assume each MAC address record is in the range of 50 bytes, one megabyte of memory for these records could cover approximately 20,000 endstations. If it took 20 switches to cover this number of endstations and a megabyte of memory cost \$35, the total cost of the memory for these records would be about \$700, an insignificant cost relative the total cost of the switches, which would likely be approximately \$200,000 to \$300,000. Moreover, with typically client-server traffic, only switches near the root of the client-server traffic would see the full range of the addresses.

With this provision of memory for these records, the VLMP callbacks would occur primarily on initialization and thus would impose very little overhead in an operating network.

The 20,000 endstation LAN is larger than any network that the author is aware of, although Boeing and Microsoft may come close. This, this analysis suggests that VLMP would allow scaling well beyond the level that has been used to date.

As a counter argument to this analysis, some switches use a distributed architecture suggesting that this state has to be replicated per port. However, packets for a given source address normally only arrive on one port, so this cached information can be maintained per port without significant replication of state at each port.

Some switches use what some have called *negative filtering* in which a switch has one or more designated backbone links and forwards all traffic not recognized as to a local destination to the backbone. It similarly rejects all traffic from the backbone which is addressed to a destination it does not recognize. In this scheme, there is no need (ignoring virtual LANs for the moment) to maintain a record for remote source addresses, contradicting one of the base assumptions of the above analysis. We make several points on this front.

First, changing these switches to use a record per MAC address seems easier than supporting encapsulation, which requires changing the basic transmission and reception mechanisms. Second, the encapsulated approach restricts the topologies or else doubles the packet traffic for multicast. That is, either there are designated links that only contain other encapsulating switches or else each multicast packet on the link needs to be sent once unencapsulated for endstations on this link and once encapsulated for switches on this link. (We do not consider assuming all endstations can handle encapsulated packets as a practical assumption.) It would be necessary transmit unicast packets in both forms as well if the switch could not determine whether the endstation was present on the link or on the other side of the switch, or if not all switches support the encapsulation scheme. Moreover, similar multiple transmissions appear to be necessary when a source host is in multiple virtual LANs. Finally, VLMP can be used in conjunction with an encapsulation scheme such as the Cisco-proposed IEEE 802.10 scheme to map SAID's to virtual LAN names. A switch supporting encapsulated as well as non-encapsulated packets (and VLMP) can interoperate with both kinds of switches.

7 Concluding Remarks

VLMP is a protocol that allows the switches on an extended Ethernet to determine the virtual LAN membership of a given Ethernet address that is connected through another switch supporting VLMP. It is based on the model of identifying virtual LANs by character string name, and allowing communication between endstations that have a virtual LAN in common, based on comparing these names. The model is extended slightly to accommodate additional attributes of virtual LAN, such as protocol-specific virtual LANs, by a convention on the naming of these virtual LANs, namely a standard prefix to names identifying these attributes.

As a standard protocol to be used by switches from all vendors, VLMP provides interoperability in a multi-vendor environment. Moreover, VLMP allows individual switches to implement and administer the membership by different (possibly proprietary) mechanisms. The key requirement is that all switches support naming of virtual LANs in a common name space and that they implement EGMP.

VLMP is specified for Ethernet. It can be directly used with other network technologies that use Ethernet address formats, such as FDDI. Its design could be readily adapted to other switched network technology with similar distributed virtual LAN requirements, emulated LANs. We leave it to those working with other MAC protocols and other network technologies to adapt VLMP if needed and so desired. Hopefully, the design of VLMP as an RPC-generated protocol will facilitate such adaptation.

Acknowledgment

Comments by John Wakerly of Alantec significantly contributed to the content and presentation. Mick Seaman of 3COM suggested the dissemination of virtual LAN membership rather than relying exclusively on callback for this information.

References

[1] D. Cheriton and S. Deering and K. Duda, Ethernet Group Management Protocol (EGMP), draft RFC, September 1995.

[2] D.R. Cheriton, Recursive Structuring of an RPC Protocol Architecture, Proceedings of SIGCOMM'88, ACM, Stanford, CA 1988.

[3] S. Deering, "Host Extensions for IP Multicasting", RFC 1112, Stanford, Aug. 1989.

[4] J. Mogul and J. Postel, "Internet Standard Subnetting Procedure", RFC 950, Stanford University and USC/ISI, Aug. 1985.

[5] D. Plummer, An Ethernet Address Resolution Protocol or, Converting Network Protocol Addresses to 48-bit Ethernet Addresses for Transmission on Ethernet Hardware, RFC 826, Symbolics, Nov. 1982.

[6] J. Postel, "Multi-LAN Address Resolution", RFC 925, USC/ISI, Oct. 1984.

[7] C.Smoot and J. Quarterman, "Using ARP to implement transparent subnets gateways", RFC 1027, Oct. 1987.

[8] Sun Microsystems Inc, "XDR: External Data Representation Standard", RFC 1014, Sun Microsystems Inc, June 1987.

[9] Sun Microsystems Inc, "RPC: Remote Procedure Call Protocol Specification Version 2", RFC 1057, Sun Microsystems Inc, June 1988.

A GMP Protocol

TBD - see EGMP specification for now.