[Proceeding] LMAP: A Protocol to Automate the Setup of Logical Networks

LMAP: A PROTOCOL TO AUTOMATE THE SETUP OF LOGICAL NETWORKS

Riccardo Scandariato, Fulvio Risso and Patricia Lago
Dipartimento di Automatica e Informatica, Politecnico di Torino
Corso Duca degli Abruzzi, 24, 10129 Torino - Italy

ABSTRACT

This paper presents the Logical Membership Announcement Protocol (LMAP), a new signaling protocol that handles the construction of logical topologies (e.g. overlay networks) in a straightforward way. LMAP defines both a way to disseminate membership information into the network and a way to determine the adjacencies that must be established between members to create the logical topology. LMAP represents a building block for upper-layer distributed services (e.g. Virtual Private Networks, BGP peering, and more) that rely on a logical topology to achieve their functionality. Moreover, appropriate protocol extensions can be used to transport information that is meaningful only to the service application.

This paper presents the key points behind LMAP, it describes how LMAP works, it makes a comparison with other proposals, and it shows some preliminary results proving its robustness and its excellent scalability.

I. INTRODUCTION

A logical network is a network that groups together a set of entities (e.g. users) that are somehow connected between them. One of the most common applications of logical networks is the *overlay network*, i.e. a network with custom characteristics (quality of service, security) spanning over another 'base' network, which has a wider acceptance and which provides data exchange between two endpoints.

Significant examples of overlay networks are the early days' Internet (a data network built on top of the telephony network), the modern x-bones (MBone, 6Bone X-Bone), or Virtual Private Networks (VPNs) [3].

A general solution to setup logical networks will have to address two important problems. First, it needs a simple and straightforward mechanism to disseminate membership information. Term "membership" implies the establishment of a logical association between a networked entity (e.g. an host) and a *community*. A community is directly related to a group of applications (the *service*) that need to exchange data between them (e.g. a set of routing daemons participating to a BGP domain). Moreover, membership dissemination is intended as a collaborative process through which each entity declares its membership and receives membership information from peers involved in the same community.

The second problem concerns the automatic computation of the *logical topology* the service is relying on. A logical topology describes how members can mutually interact within the community. For instance, members can interact directly, when they exchange information without the mediation of other members (e.g. two directly connected peers), or indirectly, when the communication requires third-party members (like two clients connected through a gateway).

Although these two steps are the basic building blocks for the creation of a logical network, they are usually not enough in order to make services operative. For example, a VPN host [3] requires the knowledge of all the members involved in the same VPN (the community). Further, members must be able to exchange data by means of special paths. That is, the VPN service needs to map the logical topology into a physical network configuration (e.g. tunnel creation). The exploitation of topological information is usually service-dependant. For instance, differently from the VPN case, BGP peers [8] do not create tunnels; instead, they setup TCP connections in order to exchange routing data.

One solution to the problems above can be found in the Logical Membership Announcement Protocol (LMAP), a simple, general and extendible signaling protocol that we designed to automate both the dissemination of membership information and the setup of logical topologies within a community. LMAP is designed to be a general-purpose protocol and it takes no assumptions about the type of service it is supporting, the semantics of the membership itself, or the way the logical topology will be actually exploited. Furthermore, its modular architecture accommodates for ad-hoc extensions to the base protocol, enabling applications to encapsulate service-dependant parameters within protocol messages.

This paper presents the design and implementation of the LMAP protocol. Section II discusses the related work and points out that other solutions, compared to LMAP, are targeted to specific services. While Section III presents the overview of LMAP basic functionalities, Section IV presents some advanced features like its plug-and-play capabilities, the support for stub networks, and more. Section V presents the applicability of LMAP to VPN scenarios. Finally, conclusive remarks are given in Section VI.

II. RELATED WORK

To date, the setup of a logical network is mainly carried out manually; few efforts have been done to automate the dissemination and discovery of membership and to seamlessly configure logical topologies. Some proposals exist indeed, but they are usually targeted to specific services; hence they do not provide a general approach in order to be portable across different services.

We evaluate existing proposals against the following lists of characteristics that are required for a complete topology setup solution:

1. *Discovery*: automatic discovery of other members in the network;
2. *Topology*: full control of topology that has to be deployed;
3. *Configuration*: intuitive configuration of members;

4. *Extensibility*: applicability to a wide range of distributed services.

Main efforts were done in the area of overlay networks. In particular, key contributions can be found in the X-Bone project [10] and the Resilient Overlay Networks project (RONs) [1].

X-Bone is a distributed architecture for dynamic creation of virtual networks. The architecture is made up of two main modules: a resource daemon (RD), which resides on every network node, and a centralized overlay manager (OM), which coordinates tunnel setup and network route configuration.

The administrator requests the creation of a new overlay to the OM through a web graphical interface. Then, the OM creates the overlay in two steps. First, it invites the network nodes to participate to the new overlay by means of multicast messages. Second, the OM configures the nodes (which answered the invitation) through a secure TCP connection. Finally, the RD performs the actual node configuration (e.g. interface setup).

According to our list of characteristics, we can observe that from the *discovery* perspective no member discovery (in the strict sense) is used by X-Bone, since members are centrally configured. Hence, there is no need for members to know each other. Rather, discovery is used (though multicast) by the OM in order to contact the RDs.

Concerning *topology*, the X-Bone architecture is oriented toward the creation of the overlay in a 'do-what-I-mean' fashion (e.g. an overlay creation request can be expressed as the following statement: "Create a star of 6 nodes"). Hence, X-Bone does not offer a punctual control on which nodes will be involved in the overlay (multicast invitations are issued until a sufficient number of nodes replies). Additionally, X-Bone creates overlays according to few topology patterns, which mainly are busses, rings, and stars: hence, complex topologies cannot be managed easily. In X-Bone, no *configuration* is needed at the node side, since configurations are managed centrally by the OM, and pushed in nodes through a TCP connection. Obviously, this simplifies network management, but it introduces a single point of failure (the OM). At last, *extensibility* has not been addressed since X-Bone is only applicable to VPN scenarios.

Beside the comparison above, X-Bone presents some interesting features, such as the ability of creating stacked (i.e. hierarchical) overlays, or its use of TTL-scoped multicast messages (used by X-Bone for invitation messages), interesting to isolate signaling traffic.

The RON project focuses on the use of overlay networks as a mean of increasing robustness of wide area networks. In particular RON proposes a fast mechanism to recover link or path failures by exploiting alternative routes offered by the overlay (overcoming the slow convergence time of traditional routing protocols, e.g. BGP). A resilient network is made up of few nodes (RON routers) in a wide network. Nodes execute a link-state routing protocol to disseminate reachability information between them, and repeatedly probe available paths (i.e. virtual link). If a fault is identified on an adjacency the traffic is re-routed on an alternative path by means of the overlay.

Concerning our list of characteristics, *discovery* is provided through a dynamic announcement-based membership protocol. A new RON node has to know at least one peer in the overlay; afterwards, announcements are broadcasted to other peers through the overlay network itself. The solution for *topology* is quite limited: topology is built by a routing protocol, thus only full meshes are allowed (each node in a resilient network of N nodes has N-1 virtual links). The topology setup *configuration* is quite simple, since just a peer must be provided to the RON node. More complex is the configuration of routing policies used at the edge of a RON node; but this is a service-dependent issue (therefore not applicable to this work, being LMAP service-independent). At last, RON does not address *extensibility*: the solution is specifically targeted to fault-recovery, although it can be adapted to the VPN case.

Another topology setup mechanism (again, peculiar to VPNs) was suggested by the IETF in [7]. It proposes to carry some additional information in BGP (an extended community attribute) in order to define the role (hub, spoke, or mesh) of the BGP speaker. BGP routers create an adjacency only if it is compatible with their attributes (e.g. an hub brings up a new adjacency with a spoke). However, this proposal does not allow the dynamic discovery of BGP peers, since they must be explicitly configured. Moreover the use of a simple role attribute is of no help in the creation of complex topologies. In summary, we can observe that no membership *discovery* is provided and peers must be configured explicitly (by means of their IP address). Further, only simple *topologies* (meshes, stars) can be created, the same *configuration* burden of BGP is required and the solution only works for BGP-based VPNs (therefore no *extensibility* is addressed).

III. PROTOCOL OVERVIEW

LMAP is a signaling protocol that automates the dissemination of membership information. Such information defines the set of network systems that are participating to a given community. Moreover LMAP decreases the configuration effort in managing the logical topology on which a service must be deployed.

A. Membership dissemination

Each community is identified by a globally unique identifier (the *Community ID*), and it is associated with a unique multicast network address (*community group*). Each member maintains the association between the two values; the assignment of the multicast groups has to be done by means of a special procedure that is outside the scope of this work[1]. Globally unique multicast addresses must be used only for communities that span several administrative domains. Otherwise, addresses can be reused across distinct domains, hence reducing the risk of address shortage.

Each LMAP node announces itself as a member of a given community by means of advertisement messages directed to the community group. Since each LMAP node (that can be either a router or an host) joins only the group of interest, it receives only the desired announcements: this choice avoids overloading nodes with useless membership information. Each announcement reaches all other network members in the same community and it is repeated periodically in order to refresh membership (*soft-state* model). The announcement can make use of a scope limiter (for

[1] For instance, when a new service is created, its multicast group can be advertised by means of SAP [4]. Alternately, a group address can be requested to a multicast address allocation server by means of MDCAP [5]. The assignment procedure should also consider scenarios in which a community crosses multiple administrative domains.

example the *time to live* contained into the IP packet) in order not to flood the entire network.

Note that for membership dissemination and topology construction to work properly, not all systems in the network must run LMAP, but only those that want to participate in the service. In general, systems in the network are classified as service members (i.e. systems that participate in a given community and adopt LMAP to announce their membership) and service-unaware systems.

B. Creation of the logical topology

The logical topology can be seen as a group of nodes and a set of adjacencies among them. Instead of explicitly declaring adjacencies, LMAP implicitly deduces them from the definition of the role played by members of a shared logical topology, e.g. the hub of a star. The immediate advantage of this approach is that administrators can identify roles instead of adjacencies, which are hard (and time-consuming) to define, and can be computed automatically by the protocol. For instance, in Figure 1 (in which the logical topology is made of labeled nodes and dashed lines between them) we can easily identify a mesh among nodes C, E, D, and the two leafs A and B attached to E.

Moreover, the definition of every kind of topology (full meshes, partial meshes, trees, etc.) can be easily established by a proper role configuration. Again, in the example above, we have a complex topology composed by a star (A, B, E) plus a full mesh (C, D, E).

In LMAP, member role information is carried inside membership announcements. LMAP defines two role-types: `hub` (the member will establish direct adjacencies with all other members) and `leaf` (the member will establish adjacencies only with hub members). Although the resulting topology seems to be simply a star, we will show that an arbitrary topology is permitted by properly defining the *role-ambit*, a parameter additional to the role-type. For instance, role "`Hub_01`" (hub number one) has a type equal to "Hub" and an ambit equal to "1".

Logical topology is established through a simple algorithm: an adjacency exists between two members if and only if the corresponding roles match according to one of the following *construction rules*:

- For the same role-ambit, both members have role-type `Hub`.
- For the same role-ambit, one member has role-type `Hub` and the other member has role-type `Leaf`.

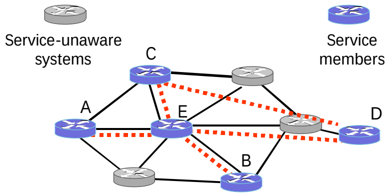

Figure 1. Network with LMAP members.

The logical topology will result from the roles assigned to the network members and it can be arbitrary. For instance, Table 1 shows a possible set of roles realizing the sample topology drawn with dashed lines in Figure 1.

Member	Role combo	Adjacencies
A	`Leaf_01`	`E (Hub_01)`
B	`Leaf_01`	`E (Hub_01)`
C	`Hub_02`	`D, E (Hub_02)`
D	`Hub_02`	`C, E (Hub_02)`

E	`Hub_01`
	`Hub_02`
		`A, B Leaf_01);`
		`C, D (Hub_02)`

Table 1. Role combos referred to Figure 1

As shown in Table 1, a member can be assigned with more than one role for the same community and that set of roles is called *role combo*. There is only a single rule that role combos must obey to: *a member cannot contemporary be* `Leaf` *and* `Hub` *for the same role ambit*. For instance, role combo "`Hub_01 Hub_02 Leaf_03`" is valid, while "`Hub_01 Hub_02 Leaf_02`" is not.

The usage of roles to specify the logical topology is much more powerful (and simpler) than other methods. For instance, a method often used consists in listing explicitly the members to which an adjacency must be established. However, this method requires a greater configuration effort. For instance, in a full mesh of N nodes, LMAP requires the configuration of the same role in all the N nodes (O(N) complexity), in place of (N-1) different adjacencies in all the N nodes ($O(N^2)$ complexity). Additionally, this method is error prone and time-consuming: adding a new member requires updating the configuration of all members that must establish an adjacency towards it. For instance, let us suppose a star topology where the hub is a single point of failure. Replacing the hub (or adding a backup hub for redundancy) requires changing the configuration of all the leaf members. LMAP straightforwardly solves the problem: once the new node is assigned with the original node's role, topology is automatically recomputed (with no changes in leafs).

LMAP nodes are not obliged to keep track of the entire logical topology: they have just to record their adjacencies (i.e. peers). Additionally, depending on how the LMAP protocol is implemented, it can keep track of either the full logical topology, or just local adjacencies. Note that keeping track of the full logical topology can be possibly unfeasible if the announcements were scope-limited.

LMAP does not foresee an explicit recovery algorithm in case of network failures (e.g. network partitioning). Nevertheless, the normal operating mode (i.e. based on properly configured timers) will guarantee at run-time the consistency of the LMAP database. On the contrary, LMAP explicitly foresees mechanisms that speed-up topology re-computation. For example, membership changes are immediately notified to the group. Additionally, LMAP defines an explicit *leave* message. Such features are particularly useful in dynamic environments where membership joining and leaving occur frequently, but they are not detailed in this paper (see [9] for further details).

C. LMAP Application Programming Interface

Application-level software that wants to exploit LMAP capabilities needs a standard interface to interact with the associated LMAP instance (i.e. a daemon implementing LMAP). To this aim, LMAP provides a simple set of primitives to `Join()` and `Leave()` a community, to `Add()` and

Remove() service-dependent parameters to LMAP announcements, and to Get() the information contained in the LMAP database (e.g. the current community topology). The service application, in turn, must provide the LMAP instance with a handle to a Notify() function. The latter is called by the LMAP instance in order to notify the service application of any change occurred in the LMAP database. A change can consist in a new announcement that has been received, a modification in the service parameters of a previously recorded member, and alike.

IV. ADVANCED FEATURES

This Section illustrates the advanced features of LMAP.

A. Plug and play

The LMAP protocol includes some mechanisms to ease the configuration of members, in order to assure plug and play capabilities. To this aim, the protocol reserves the role Hub_0. When an LMAP instance is executed with no role combo, it will automatically configure the Hub_0 as the default role. As a consequence, if no role is specified for all members, a full mesh topology is established between members. LMAP also reserves a default multicast address that can be shared among all the communities. If an LMAP instance is executed without any community group, it will automatically use the default group. Obviously, among all LMAP announcements sent over the default group, each LMAP instance needs to select only those related to its own community. To this aim the LMAP node can check the *Community ID Object*, a community identifier contained in each announcement. While distinct multicast addresses make LMAP more scalable, the default address leads LMAP to be possibly started without any setup information (except the Community ID).

B. Unicast support

Although multicast is a powerful way to disseminate memberships, it could not be widely available. For instance, some transit domains that interconnect multicast-aware networks could not support multicast forwarding. LMAP offers an option to overcame these issues by providing unicast support. In this case, members exchange the LMAP messages by using unicast packets. A small number of modifications are required in order to support unicast operations. Basically, unicast peering relationships must be explicitly configured in LMAP members that face a unicast-only domain. Unicast LMAP differs only in the way membership is disseminated (i.e. announcements only reach the configured peers), while logical topology computation still relies on roles. That is, an adjacency between two members is established on the basis of matching roles, even though a unicast peering is configured between them. Thereafter, the protocol follows the same timings of multicast mode, i.e. the member still refreshes membership; however refresh messages are replicated to all unicast peers. In order to simplify the configuration burden in the unicast mode, LMAP does not exclude the possibility to download the configuration (i.e. the unicast peering relationships) from a central server by means of a reliable TCP connection. The deployment of a configuration server is clearly discouraged because of scalability and reliability problems, but it could be helpful in some specific cases.

C. Support for stub networks

The LMAP protocol is designed to allow the construction of arbitrary (possibly complex) topologies. However, there are particular scenarios where simplicity of configuration would be more useful than flexibility in topology construction: e.g. services based on a client-server paradigm. Figure 2 shows clients in a stub network (e.g. a LAN) that want to participate to a service deployed in the core network (e.g. a provider). To limit network traffic, service requests/data originated by clients can be concentrated by a near-site gateway (labeled as server), rather than being directly sent in the core side. The gateway, in turn, announces itself over the core network (dashed arrow), by avoiding interactions between the two sides. Indeed, clients are not interested in the core topology of the service: they just want to know *which* is the gateway that will forward service traffic on their behalf. In this case, the topology is implicit and quite simple: clients just connect to the gateway, with the result of a starred topology (dashed lines). Furthermore, clients need a *seamless configuration* since they typically run on hosts owned by non-technical users.

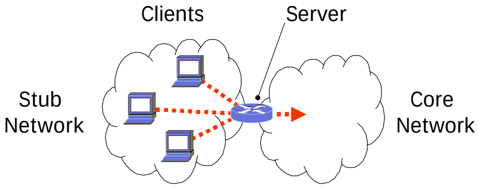

Figure 2. Stub network scenario

LMAP includes two additional features to implement the above scenario. First, members (server and clients) in the stub network use a well-known multicast address called *shared group* whose scope can be limited[2], e.g. to the LAN. The use of a well-known group makes client configuration easier and avoids multicast packets generated by the core network to be received in the stub side too. Second, to preserve the plug-and-play capabilities, the Leaf_0 role is also reserved for the default configuration of LMAP clients. That is, when a LMAP instance is executed specifying the client behavior, the protocol assumes the Leaf_0 as the default role. As a consequence, the only information that must be specified for client stations is the community through the Community ID.

As shown in Figure 2, the server node participates to both the core and stub side. Hence it must have two different role combos (one for each side), to which it applies the construction rules independently, by matching each combo with announcements received from the corresponding side. When an LMAP instance is executed specifying the server behavior, the protocol assumes Hub_0 as the default role for the stub side. Concerning the core side, the LMAP instance uses the configured combo/group parameters (if present) or it applies the default values as described in Section IV.A.

Further, the server may enforce some coupling mechanisms between the two sides: the server may use clients' announcements to activate its membership within the core network. Similarly, when no more announcements are received

[2] How scope is limited is an implementation issue. Groups can be confined by using administratively scoped addresses [6]. Alternatively, TTL thresholds can be configured on the server node.

from clients, the server may deactivate its membership; i.e. it may stop announcing itself, causing the pruning of the server (and of the attached stub network) from the logical topology. This behavior is conceptually akin to the interaction between IGMP and multicast routing protocols at the LAN-WAN edge.

D. Extensibility

LMAP is a supporting protocol for the upper-level services, able to carry service-dependent information. However, it makes no assumptions about the service itself. That is, LMAP implements a way to disseminate memberships and it distributes service information with no further elaboration.

LMAP messages are made of an envelope containing many objects, each one carrying a piece of information (e.g. a role). The base protocol defines its own set of objects, which are used for membership dissemination and topology setup and that are processed by the LMAP instance internally. Further, LMAP can transport service-specific data in its announcements though a generic *Service Object* that is completely transparent from the LMAP standpoint.

The Service object is passed by the service to the LMAP instance, and then inserted into announcements. The format of this object is application-specific, allowing a wide range of choices (from binary to XML). These objects are not modified or interpreted by the LMAP protocol: optional objects are extracted and delivered to the upper-layer service application upon reception of an announcement. The service application parses and processes objects' content according to the service-dependent logic.

For example, a service application running on a server member can elaborate and compact the information learned from clients over the shared group. Then, the compacted information can be passed back to the LMAP instance for core-side dissemination. In certain cases, in fact, better performances can be gained by aggregating service-specific objects coming from different clients into a bulked announcement over the core network. Such practice is transparent to LMAP, since service-specific objects are interpreted at the application level. However, in order to perform these elaborations, LMAP should be coupled (through the LMAP API) with the service module that resides on the network node.

V. EXTENSIONS FOR VPN PROVISIONING

This Section shows the applicability of LMAP to the VPN provisioning service. Section V.A briefly describes provisioned VPN. Further details can be found in [2]. Section V.B explains how LMAP can be used in the context of VPN provisioning, demonstrating LMAP extensibility.

A. Provisioned Virtual Private Networks

As depicted in Figure 3, a provisioned VPN is a connectivity service offered by a provider to its customers (sites or single users). Virtual connectivity among VPN clients (circled) is implemented inside the provider network by establishing a proper mesh of tunnels (dashed lines). Network nodes interconnected by tunnels are called VPN routers (darker) and they constitute the so-called "virtual topology" (i.e. the topology of the VPN). VPN routers are configured in order to route data generated by customer stations belonging to a VPN, to the correct destination. VPN traffic is forwarded through the tunnel mesh (see arrows).

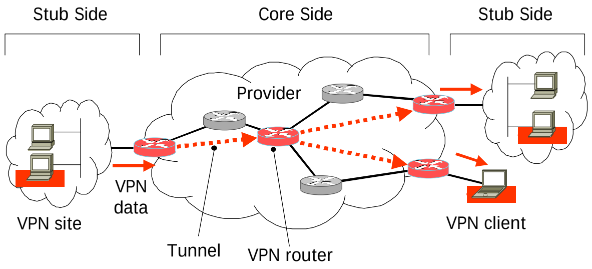

Figure 3. Provisioned VPN example

VPN routers are mainly placed at the edge of the provider network. These routers (called provider edges in the literature) act as *access* point for customers residing in the stub side. These routers identify which VPN the customer traffic belongs to on the basis of the incoming interface and/or traffic filters that match the traffic characteristics (i.e. values of TCP/IP header fields). After having classified the incoming traffic, they must decide to which outgoing tunnel (in the core side) they must forward the traffic, mainly on the basis of the destination address of packets. Forwarding to an outgoing tunnel implies the encapsulation and optionally the encryption of the traffic itself.

There can also be VPN routers placed in the *core* of the network. These routers act as tunnel switches, in that they switch customer traffic from an incoming tunnel to the proper outgoing tunnel (again on the basis of the destination address). Note that traffic classification is straightforward, since traffic comes in from tunnels that are implicitly associated to the corresponding VPNs.

Summarizing, a proper set up of a provisioned VPN requires configuring the following information:

- Membership: VPN routers must be aware of the set of VPNs they are serving.
- Dissemination: VPN routers must propagate their membership to other VPN routers in the core side.
- Topology setup: tunnel topology must be settled among VPN routers.
- Classification: access VPN routers must identify which ingress traffic belongs to which VPN, in order to deliver it accordingly.

LMAP provides an optimized solution to all the above items. Additionally, LMAP allows the decoupling of VPN setup steps from the VPN-layer routing. Hence, no changes are needed for routing protocols, since they do not carry VPN-related extra information.

B. Operating provisioned VPNs with LMAP

As depicted in Figure 3, a provisioned VPN requires configuration both in the stub side and in the core side.

Concerning the stub side, interaction between VPN clients and the access router does not (typically) require the setup of a tunnel topology. Mainly, clients have to signal their intention to join a given VPN. Additionally, a client may decide that just a part of its traffic must be routed on the VPN. In summary, it needs to inform the access router of both its intention to join a VPN, and which traffic belongs to the VPN. These operations can be carried out effortless by LMAP thanks to its support for stub networks: as soon as a client joins a VPN it sends an announcement to the shared group. Then the access router (acting as a server) receives the announcement and adds the site to the VPN topology. Obviously, if there were other clients already members of the same VPN, the access router can drop announcements of further clients because the site was already inserted in the topology.

The server gathers the whole traffic of a VPN client through the same network interface. Since the client may participate to multiple VPNs, it inserts proper *filtering information* into the Service Object of each announcement in order to specify which traffic belongs to which VPN. This indicates the server a way to recognize the different VPN flows.

Joining of customers to a given VPN is realized through standard LMAP signaling in the provider network. VPN routers (both access and core) must be configured with a proper set of roles according to the topology that the customers desire. Membership exchanges of LMAP announcements (sent through the per-VPN community group) lead to the setup of the tunnel mesh. Since VPN routers can use different tunneling technology, announcements can carry *tunneling parameters* in the Service Object. These parameters list all tunneling technologies (with a corresponding preference) that are supported by each member and they can be used to automatically agree on which tunneling technology (if any) to adopt during the tunnel setup.

Once tunnels have been established, a VPN-dedicated routing protocol can be activated on top of them. This does not require any modifications to the base routing protocol activated on the base network. As a result, members of the VPN will be able to exchange traffic securely with low administrative effort.

VI. CONCLUSIONS

This paper presented the LMAP protocol, a new signaling protocol that handles the construction of logical topologies.

The most important characteristic of LMAP, compared to related work, is its flexibility. LMAP is service independent, while neither X-Bone nor RON is.

Moreover LMAP is the only protocol that satisfies all the requirements needed for the construction of a logical network (i.e. *discovery, topology, configuration,* and *extensibility*). It does not require a new node to already know a peer; instead it automatically discovers other peers thanks to multicast announcements. Then, it builds seamlessly the logical topology, which can be arbitrarily complex, and it does not suffer of any single points of failure thanks to a collaborative and distributed approach to topology construction. Finally, member configuration is really simple and it supports several services (as exemplified in Section V.B for VPN support).

LMAP borrows some ideas from other existing protocols. For instance, the support for stub networks (Section IV.C) shows similarities with IGMP as each member announces itself on a multicast group and these announcements are “grouped” by the multicast access router. Membership dissemination uses some ideas introduced in the Link State routing protocols such as OSPF (each member sends announcements with a "flooding" technique), even if it does not require for each member to keep all the other members’ announcements.

A first prototyping implementation of LMAP has been tested on FreeBSD, Linux, and Win32. To give a first glance on initial performance evaluation and on-field operation of LMAP, some considerations follow:

- When a new member *joins* a certain topology, time convergence is of few seconds, and is directly dependent from the underlying multicast support.
- When an existing member *leaves* a topology, time convergence is equivalent to the scenario above, provided that the “bye” procedure is correctly followed. This procedure requires the generation of a predefined sequence of packets stating the intention to leave the community.
- Topology convergence is ensured thanks to a refresh mechanism that reissues announcements every three minutes.
- If an existing member abandons a topology without explicitly following the “bye” procedure, its entry remains valid for a period that is (at most) 3,5 times the refresh time (i.e. 10,5 minutes). This can represent a problem for appliances based on an always up-to-date topology. To meet this requirement, the refresh time can be configured in each LMAP instance.
- The size of a typical announcement for an average host playing two roles is about 100 bytes. If we consider a topology of 1,000 members, this requires to manage a traffic of about 4,5 Kbps for each host, which is far too low to represent any problem.
- Moreover, if we think of a service (e.g. VPNs) and we add service-dependent information (e.g. for VPN configuration) a typical message will be heavier: for instance, a single filtering information can be of 25 to 39 bytes in case of IPv4 or IPv6 respectively; a VPN message defining two filtering information can add 50 to 100 bytes to the size of the announcement message reported above (see previous point), thus doubling the necessary bandwidth (which continues to be negligible being less than 9 Kbps).

To conclude, we believe that LMAP represents a promising technology for constructing of logical topologies in an easy yet robust way. Its flexibility and general applicability represent an additional advantage, as exemplified for VPNs.

REFERENCES

[1] Andersen D., et al., “Resilient Overlay Networks”, 18th ACM SOSP, Banff, Canada, October 2001

[2] Callon R., et al., “A Framework for Provider Provisioned Virtual Private Networks”, draft-ietf-ppvpn-framework-04.txt, IETF Internet Draft, February 2002

[3] Gleeson B., et al., “A Framework for IP Based Virtual Private Networks” IETF RFC 2764, February 2000

[4] Handley M., et al., “Session Announcement Protocol” IETF RFC 2974, October 2000

[5] Hanna S., et al., "Multicast Address Dynamic Client Allocation Protocol (MADCAP)", IETF RFC 2730, December 1999

[6] Meyer D., “Administratively Scoped IP Multicast”, IETF RFC 2365, July 1998

[7] Ould-Brahim H., et al., “Using BGP as an Auto-Discovery Mechanism for Network-based VPNs”, draft-ietf-ppvpn-bgpvpn-auto-02.txt, IETF Internet Draft, July 2001

[8] Rekhter Y., et al., “A Border Gateway Protocol 4 (BGP-4)”, IETF RFC 1771, March 1995

[9] Scandariato R., Risso F., “Virtual Network Provisioning: LMAP”, Politecnico di Torino Technical Report DAI-SE-2001-07-26, July 2001

[10] Touch J., “Dynamic Internet Overlay Deployment and Management Using the X-Bone", 8th International Conference on Network Protocols, Osaka, Japan, November 2001