MPLS WG
Internet-Draft
Intended status: Standards Track
Expires: April 30, 2015

K. Kompella
R. Balaji
Juniper Networks, Inc.
G. Swallow
Cisco Systems

October 27, 2014

Label Distribution Using ARP
draft-kompella-mpls-larp-02

Abstract

This document describes extensions to the Address Resolution Protocol to distribute MPLS labels for IPv4 and IPv6 host addresses. Distribution of labels via ARP enables simple plug-and-play operation of MPLS, which is a key goal of the MPLS Fabric architecture.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The term "server" will be used in this document to refer to an ARP/ L-ARP server; the term "host" will be used to refer to a compute server or other device acting as an ARP/L-ARP client.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document describes extensions to the Address Resolution Protocol (ARP) [RFC0826] to advertise label bindings for IP host addresses. While there are well-established protocols, such as LDP, RSVP and BGP, that provide robust mechanisms for label distribution, these protocols tend to be relatively complex, and often require detailed configuration for proper operation. There are situations where a simpler protocol may be more suitable from an operational standpoint. An example is the case where an MPLS Fabric is the underlay technology in a Data Centre; here, MPLS tunnels originate from host machines. The host thus needs a mechanism to acquire label bindings to participate in the MPLS Fabric, but in a simple, plug-and-play

manner. Existing signaling/routing protocols do not always meet this need. Labeled ARP (L-ARP) is a proposal to fill that gap.

[TODO-MPLS-FABRIC] describes the motivation for using MPLS as the fabric technology.

1.1. Approach

ARP is a nearly ubiquitous protocol; every device with an Ethernet interface, from hand-helds to hosts, have an implementation of ARP. ARP is plug-and-play; ARP clients do not need configuration to use ARP. That suggests that ARP may be a good fit for devices that want to source and sink MPLS tunnels, but do so in a zero-config, plug-and-play manner, with minimal impact to their code.

The approach taken here is to create a minor variant of the ARP protocol, labeled ARP (L-ARP), which is distinguished by a new hardware type, MPLS-over-Ethernet. Regular (Ethernet) ARP (E-ARP) and L-ARP can coexist; a device, as an ARP client, can choose to send out an E-ARP or an L-ARP request, depending on whether it needs Ethernet or MPLS connectivity. Another device may choose to function as an E-ARP server and/or an L-ARP server, depending on its ability to provide an IP-to-Ethernet and/or IP-to-MPLS mapping.

2. Overview of Ethernet ARP

In the most straightforward mode of operation [RFC0826], ARP queries are sent to resolve "directly connected" IP addresses. The ARP query is broadcast, with the Target Protocol Address field (see Section 5 for a description of the fields in an ARP message) carrying the IP address of another node in the same subnet. All the nodes in the LAN receive this ARP query. All the nodes, except the node that owns the IP address, ignore the ARP query. The IP address owner learns the MAC address of the sender from the Source Hardware Address field in the ARP request, and unicasts an ARP reply to the sender. The ARP reply carries the replying node's MAC address in the Source Hardware Address field, thus enabling two-way communication between the two nodes.

A variation of this scheme, known as "proxy ARP" [RFC2002], allows a node to respond to an ARP request with its own MAC address, even when the responding node does not own the requested IP address. Generally, the proxy ARP response is generated by routers to attract traffic for prefixes they can forward packets to. This scheme requires the host to send ARP queries for the IP address the host is trying to reach, rather than the IP address of the router. When there is more than one router connected to a network, proxy ARP enables a host to automatically select an exit router without running

any routing protocol to determine IP reachability. Unlike regular ARP, a proxy ARP request can elicit multiple responses, e.g., when more than one router has connectivity to the address being resolved. The sender must be prepared to select one of the responding routers.

Yet another variation of the ARP protocol, called 'Gratuitous ARP' [RFC2002], allows a node to update the ARP cache of other nodes in an unsolicited fashion. Gratuitous ARP is sent as either an ARP request or an ARP reply. In either case, the Source Protocol Address and Target Protocol Address contain the sender's address, and the Source Hardware Address is set to the sender's hardware address. In case of a gratuitous ARP reply, the Target Hardware Address is also set to the sender's address.

3. L-ARP Protocol Operation

The L-ARP protocol builds on the proxy ARP model, and also leverages gratuitous ARP model for asynchronous updates.

In this memo, we will refer to L-ARP clients (that make L-ARP requests) and L-ARP servers (that send L-ARP responses). In Figure 1, H1, H2 and H3 are L-ARP clients, and T1, T2 and T3 are L-ARP servers. T is a member of the MPLS Fabric that may not be an L-ARP server. Within the MPLS Fabric, the usual MPLS protocols (IGP, LDP, RSVP-TE) are run. Say H1, H2 and H3 want to establish MPLS tunnels to each other (for example, they are using BGP MPLS VPNs as the overlay virtual network technology). H1 might also want to talk to a member of the MPLS Fabric, say T.

```
                                . . . . . .
                               .           .
                      H1 --- T1             T4
                         \   .     MPLS      .
                          \  .               .
                           \ .    Fabric     .
                      H2 --- T2             T3 --- H3
                              .            .
                               . . . . . .
```

Figure 1

3.1. Basic Operation

A node (say H1) that needs an MPLS tunnel to a destination (say H3) broadcasts over all its interfaces an L-ARP query with the Target Protocol Address set to H3. A node that has reachability to H3 (such as T1 or T2) sends an L-ARP reply with the Source Hardware Address set to a locally-allocated MPLS label plus its Ethernet MAC address.

After receiving one or more L-ARP replies, H1 can select either T1 or T2 to send MPLS packets that are destined to H3. As described later, the L-ARP response may contain certain parameters that enable the client to make an informed choice of the routers.

As with standard ARP, the validity of the MPLS label obtained using L-ARP is time-bound. The client should periodically resend its L-ARP requests to obtain the latest information, and time out entries in its ARP cache if such an update is not forthcoming. Once an L-ARP server has advertised a label binding, it MUST NOT change the binding until expiry of the binding's validity time.

The mechanism defined here is simplistic; see Section 4.

3.2. Asynchronous operation

The preceding sections described a request-response based model. In some cases, the L-ARP server may want to asynchronously update its clients. L-ARP uses the gratuitous ARP model [RFC2002] to "push" such changes.

In a pure "push" model, a device may send out updates for all prefixes it knows about. This naive approach will not scale well. This memo specifies a mode of operation that is somewhere between "push" and "pull" model. An L-ARP server does not advertise any binding for a prefix until at least one L-ARP client expresses interest in that prefix (by initiating an L-ARP query). As long as the server has at least one interested client for a prefix, the server sends unsolicited (aka gratuitous, though the term is less appropriate in this context) L-ARP replies when a prefix's reachability changes. The server will deem the client's interest in a prefix to have ceased when it does not hear any L-ARP queries for some configured timeout period.

3.3. Client-Server Synchronization

In an L-ARP reply, the server communicates several pieces of information to the client: its hardware address, the MPLS label, Entropy Label capability and metric. Since ARP is a stateless protocol, it is possible that one of these changes without the client knowing, which leads to a loss of synchronization between the client and the server. This loss of synchronization can have several bad effects

If the server's hardware address changes or the MPLS label is repurposed by the server for a different purpose, then packets may be sent to the wrong destination. The consequences can range from suboptimally routed packets to dropped packets to packets being

delivered to the wrong customer, which may be a security breach. This last may be the most troublesome consequence of loss of synchronization.

If a destination transitions from entropy label capable to entropy label incapable (an unlikely event) without the client knowing, then packets encapsulated with entropy labels will be dropped. A transition in the other direction is relatively benign.

If the metric changes without the client knowing, packets may be suboptimally routed. This may be the most benign consequence of loss of synchronization.

3.4. Applicability

L-ARP can be used between a host and its Top-of-Rack switch in a Data Center. L-ARP can also be used between a DSLAM and its aggregation switch going to the B-RAS. More generally, L-ARP can be used between an "access node" and its first hop MPLS-enabled device in the context of Seamless MPLS [reference]. In all these cases, L-ARP can handle the presence of multiple connections between the access device and its first hop devices.

ARP is not a routing protocol. The use of L-ARP should be limited to cases where the L-ARP client has a small number of one-hop connections to L-ARP servers. The presence of a complex topology between the L-ARP client and server suggests the use of a different protocol.

3.5. Backward Compatibility

Since L-ARP uses a new hardware type, it is backward compatible with "regular" ARP. ARP servers and clients MUST be able to send out, receive and process ARP messages based on hardware type. They MAY choose to ignore requests and replies of some hardware types; they MAY choose to log errors if they encounter hardware types they do not recognize; however, they MUST handle all hardware types gracefully. For hardware types that they do understand, ARP servers and clients MUST handle operation codes gracefully, processing those they understand, and ignoring (and possibly logging) others.

4. For Future Study

The L-ARP specification is quite simple, and the goal is to keep it that way. However, inevitably, there will be questions and features that will be requested. Some of these are:

1. Keeping L-ARP clients and servers in sync. In particular, dealing with:

    A. client and/or server restart

    B. lost packets

    C. timeouts

2. Withdrawing a response.

3. Dealing with scale.

4. If there are many servers, which one to pick?

5. How can a client make best use of underlying ECMP paths?

6. and probably many more.

In all of these, it is important to realize that, whenever possible, a solution that places most of the burden on the server rather than on the client is preferable.

5. L-ARP Message Format

```
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |           ar$hrd              |            ar$pro             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |     ar$hln    |    ar$pln     |            ar$op              |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    //                     ar$sha (variable...)                    //
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    //                     ar$spa (variable...)                    //
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    //                     ar$tha (variable...)                    //
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    //                     ar$tpa (variable...)                    //
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    //                     ar$lst (variable...)                    //
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    //                     ar$att (variable...)                    //
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: L-ARP Packet Format

ar$hrd Hardware Type: MPLS-over-Ethernet. The value of the field used here is [HTYPE-MPLS-TBD]. To start with, we will use the experimental value HW_EXP2 (256)

ar$pro Protocol Type: IPv4/IPv6. The value of the field used here is 0x0800 to resolve an IPv4 address and 0x86DD to resolve an IPv6 address.

ar$hln Hardware Length: the value of the field used here is 6.

ar$pln Protocol Address Length: for an IPv4 address, the value is 4; for an IPv6 address, it is 16.

ar$op   Operation Code: set to 1 for request, 2 for reply, and 10 for ARP-NAK. Other op codes may be used, but this is not anticipated at this time.

ar$sha Source Hardware Address: In an L-ARP message, Source Hardware Address is the 6 octets of the sender's MAC address.

ar$spa Source Protocol Address: In an L-ARP message, this field carries the sender's IP address.

ar$tha Target Hardware Address: In an L-ARP query message, Target Hardware Address is the all-ones Broadcast MAC address; in an L-ARP reply message, it is the client's MAC address.

ar$tpa Target Protocol Address: In an L-ARP message, this field carries the IP address for which the client is seeking an MPLS label.

ar$lst Label Stack: In an L-ARP request, this field is empty. In an L-ARP reply, this field carries the MPLS label stack in the format below.

ar$att Attribute TLV: In an L-ARP request, this field is empty. In an L-ARP reply, this field carries attributes for the MPLS label stack in the format below.

Figure 3 describes the format of MPLS Label Stack carried in L-ARP. Figure 4 describes the format of Attribute TLV carried in L-ARP.

```
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |    # labels   |         MPLS Label (20 bits)          |E|Z|Z|Z|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         MPLS Label (20 bits)          |E|Z|Z|Z| ...           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: MPLS Label Stack Format

MPLS Label Stack: This field contains the MPLS label stack for the client to use to get to the target. Each label is 3 octets; the Length is 3*(number of labels). This field is valid only in an L-ARP request message.

E-bit: Entropy Capability

This field indicates whether the label stack of MPLS data packets sent with the label in this advertisement can contain Entropy Label or not. If this flag is set, the client has the option of inserting ELI and EL as specified in [RFC6790]. The client can choose not to insert ELI/EL pair, if it does not support Entropy Labels, or the local policy does not permit the client to insert ELI/EL. If this flag is clear, the client must not insert ELI/EL into the label stack when sending packets with the advertised L-ARP label.

Z These bits are not used, and SHOULD be set to zero on sending and ignored on receipt.

If other parameters are deemed useful in the L-ARP reply, they will be added as needed.

```
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |      Type     |    Length     |     Metric (4 octets) ...     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  ...  Metric                  | ...                           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: Attribute TLV

6. Security Considerations

TODO

7. IANA Considerations

TODO

8. Acknowledgments

Many thanks to Shane Amante for his detailed comments and suggestions. Many thanks to the team in Juniper prototyping this work for their suggestions on making this variant workable in the context of existing ARP implementations. Thanks too to Luyuan Fang, Alex Semenyaka and Dmitry Afanasiev for their comments and encouragement.

9. Normative References

[RFC0826] Plummer, D., "Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware", STD 37, RFC 826, November 1982.

[RFC2002] Perkins, C., "IP Mobility Support", RFC 2002, October 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, November 2012.

Authors' Addresses

Kireeti Kompella
Juniper Networks, Inc.
1194 N. Mathilda Avenue
Sunnyvale, CA 94089
USA

Email: kireeti.kompella@gmail.com

Balaji Rajagopalan
Juniper Networks, Inc.
Prestige Electra, Exora Business Park
Marathahalli - Sarjapur Outer Ring Road
Bangalore 560103
India

Email: balajir@juniper.net

George Swallow
Cisco Systems
1414 Massachusetts Ave
Boxborough, MA 01719
US

Email: swallow@cisco.com