Internet Engineering Task Force
Internet Draft
Intended status: Informational
Expires: February 2017

P. Unbehagen, Ed.
D. Romascanu
J. Seligson
C. Keene
Avaya

August 2016

A Framework for Automatic Attachment of Network Objects to Core Networks

draft-romascanu-opsawg-auto-attach-framework-01.txt

Abstract

This informational document describes a method that allows for the automatic attachment of network objects (e.g. end stations, network devices, sensors, automation elements) to a core network based on the individual services that are run or configured on the objects, and the mapping of the services to the managed paths in the network. The framework proposed by this document describes the operations that need to happen in order to have the network objects connected to the network ('attached') in an automatic manner and start providing their functionality and services without any requirement or dependency between the protocol stack on the network objects and the method used to build the bridging or routing paths in the network core.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on February 18, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Large networking deployments are often faced with the problem of connecting 'legacy' end stations to an upgraded network infrastructure, or with the demand to interconnect large numbers of network objects that perform different tasks, transmit information, or and controlled remotely across this network infrastructure. This informational document describes a method that allows for the automatic attachment of network objects (e.g. end stations, network devices, sensors, automation elements) to a core network based on the individual services that are run or configured on the objects, and the mapping of the services to the managed paths in the network. The framework proposed by this document describes the operations that need to happen in order to have the network objects connected to the network ('attached') in an automatic manner and start providing their functionality and services without any requirement or dependency between the protocol stack on the network objects and the method used to build the bridging or routing paths in the network core.

2. Terminology and Abbreviations

AAC - Auto Attach Client agent that resides on a non-SPB/PBB capable element that uses LLDPDUs to request I-SID assignment for the VLANs which have been configured on its network port.

AAS - Auto Attach Server agent that processes VLAN to I-SID requests from AAC elements that are connected to a SPB BEB

Element - Any end device or network node that may implement the auto attach functionality

LAN - Local Area Network

LLDP - Link Layer Discovery Protocol

MUD - Manufacturer User Description

VLAN - Virtual Local Area Network

3. 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 RFC 2119 [RFC2119].

4. Auto Attachment Framework, Model and Components

This section provides an overview of the behavior of auto attachment functionality. The scheme proposed in this document is applicable at the link layer or at the network layer. Is the currently subject of proposed standardization work in the IEEE 802.1 Working Group [AA], and it is consistent with work proposed in the IETF in the Network Virtualization Overlays (nvo3) and Interface to the Routing System (i2rs) Working Groups.

The purpose of Auto Attach is to allow an end-device to connect to a networking device at the edge of a bridged or routed network in the core without any further knowledge or assumption of the protocol(s) being run in the core. The end-device is called an AA client (AAC) and the networking device at the edge of the core network is called the AA server (AAS). An AA Client is a device that does not support the bridging or routing protocol in the core but supports some form of binding definition between the applications or services that it is running and the format of the data packets that it sends to the network (for example VLAN tags, or tunnel identifiers), if connectivity permits, has the ability to advertise this data to a directly connected AA Server. An AA Server is network device that potentially accepts externally generated service to tags or tunnels assignments that can be used for automated configuration purposes. The client identifies itself to the server and then requests service to tags binding(s). The server will either accept or reject each binding request. If accepted, any traffic on the (locally significant) VLAN or tunnel is forwarded through the routed network at the parameters required by the service.

The simplification brought by the auto-attachment scheme consists of the use of widely deployed protocols on the first hop connection between the AAC and the AAS which allow for the interaction between the two entities to happen in an automatic manner, without requiring manual configuration of attachment information at multiple locations. AACs that utilize this automated method for service assignment pass the assignment information to the AASs where the mappings are processed and approved or rejected. Specific actions are taken on both entities based on the outcome of the mapping request.

```
                Auto Attachment Controller
                     (SDN/Policy Server)

                     |                  |
              (6)    |           (6)    |
                     V                  V
                +--------+         +--------+         +---------+
                |estation|         |        |         |         |
                |  AAC   |---------|  AAS   |---------|  core   |
                |  (3)   |         |   (3)  |         | Network |
                +--------+         +--------+         +---------+
                  <----------(1)-------->  <-------(2)------>
                  <----------(4)-------->  <-------(5)------>
```

Figure 1: Conceptual Auto Attach Model

(1) Auto-Attachment Primitives
(2) Routing Control Plane
(3) ’horizontal’ data model
(4) Service Tag
(5) Service Route/Tunnel ID
(6) ’vertical’ data model

Figure 1 depicts a conceptual example of the process where an AAC can use a one-hop protocol (1) e.g. LLDP that includes the Auto-Attachment Primitives to communicate the need to connect a service to the appropriate route or tunnel in the bridged or routed network which runs the core bridging or routing protocol (2). A ’horizontal’ data model (3) is shared by the AAC and AAS and synchronized after the initial exchange of information. If the binding requests are being accepted, the AAC will start sending traffic using the service tag (4), traffic which will be routed or tunneled appropriately in the bridged or routed paths (5). The policy data model (6) allows for the interaction between the network devices and the end-stations to a policy server, allowing for the integration of AA scheme in a policy-based service management environment.

An Auto Attach Client (AAC) can run in any device (end station, estation) that connects to a core network. One field of applications can be for example Internet of Things (IoT) devices that connect to a network. The service association is automated with relative low resources, allowing connection of the devices to the appropriate network services and applications.

The different classes of devices that run AACs may be configured by means that are specific for the respective classes of applications or services. A YANG module may instantiate the 'vertical' data model (6) of the Auto Attachment framework allowing for standard-based interaction with the Auto Attachment Controllers, which are instantiated as policy or Software Define Networks (SDN) servers.

4.1. Discussion

At least one proprietary implementation introduces the concept of an Auto Attach Proxy. Such an optional block placed between an AAC and AAS would allow for multiplexing and/or virtualizing the access to servers. At this phase we decided to leave this optional block for further study.

5. Auto Attachment Process

The auto-attachment process is composed of a one-hop two steps protocol that performs the following functions:

- Element Discovery
- Services Configuration

5.1. Element Discovery

The first stage of establishing AA connectivity involves element discovery. An Auto Attach agent resides on all capable elements. Server agents control the Auto Attach (AA) of service tags to routes or tunnels when enabled to accept and process such requests from AAC elements. Typically this is done through a global service setting and through per-port settings that control the transmission of information in the one-hop protocol (1) on the appropriate links that interface AAC's and AAS's.

Once the required AA settings are enabled on the elements (e.g., the AA service and the per-port AA settings) the AA agent on each element type, both AAC and AAS, advertises its capabilities (i.e., server/client) through protocol (1) primitives to each other.

Following discovery of AA capabilities by both the AAC and the AAS, the AA agent on each element is aware of all AA services currently provided by the network elements to which it is directly connected. Based on this information, an AAC agent can determine whether Auto

Attach data, namely locally administered assignments, should be exported to the AAS that is associated with an edge networking device to which it is attached to on its network uplink ports.

5.2. Services Configuration

Service mappings can be established when these two criteria are met:

1. AA Server found during discovery

   AAC - - AAS peering was established during the discov
ery
   process

2. Service Tags / Routes mapping are defined locally

   Assuming that an administrator has defined one or more tags / routes mappings, or AAC bindings have been received for processing

Each mapping assignment in an AA request received by the AAS is processed individually and can be accepted or rejected. An assignment may be rejected for a number of reasons, such as server resource limitations or, for example, restrictions related only to the source AAC. Rejected assignments are passed back to the originating AAC with a rejected state and, if appropriate, an indication as to why the rejection occurred. Limited state information may be maintained on the server related to rejected assignments.

Each route (or tunnel, or VLAN) that is associated with an accepted assignment is instantiated on the AAS bridge if it does not already exist.

The AAS agent is responsible for tracking which, if any, of these actions are performed so that settings can be cleared when they are no longer needed. This can occur, for example, when configuration changes on an AAC updates the received assignment list when an AAC associated with a downlink port changes or an AAC connection disappears entirely.

Each station tag (e.g. VLAN, subnet) that is associated with a service assignment must be defined on the client’s device. The port associated with the uplink connecting the AAC to the AAS must be a member of the VLAN or Subnet assignment lists that are sent to and accepted by the AAS. This allows tagged traffic on to pass through the edge networking device into the core routed network when required. To ensure that markings are maintained between devices, traffic on the uplink port MUST be tagged. If a tag has not been created before the assignment itself, it is automatically created by the AAC agent when a proposed assignment is accepted. Port tagging and the port VLAN or subnet membership update are also performed by the AAC automatically based on assignment acceptance. To ensure consistency, tags SHOULD NOT be deleted as long as they are referenced in any I-SID/VLAN assignments on the device.

An AAC must handle primary AAS loss and this requires maintenance of a server’s inactivity timer. In order to make this possible a time-alive mechanism needs to be implemented between any AAC and its primary AAS. If an interruption of communication is detected, the service is considered interrupted, the service tags / routes assignments accepted by the server are considered rejected. Assignment data is then defaulted (reverts to the ’pending’ state) and the AA agent, which resides on the AAC, removes related settings. If a back-up mechanism (alternate routes to the primary AAS, secondary AAS) exists it will be activated.

A "last updated" timestamp is associated with all active assignments on the AAS. When this value is not updated for a pre-determined amount of time, the service tags / routes assignment is considered obsolete. Obsolete assignment data and related settings are removed by the AAS.

The current routes / tags assignment list is advertised by an AAC at regular intervals. During processing of this data, an AAS must handle list updates and delete assignments from previous advertisements that are no longer present. Though these entries would be processed appropriately when they timeout, the AAS attempts to update the data in real-time and SHOULD initiate deletion immediately upon detection of this condition.

6. Security Considerations

It is important to provide an option to ensure that the aforementioned Auto Attach communication is secure in terms of data integrity (i.e., the data has not been altered in transit) and authenticity (i.e., the data source is valid).

There are several ways this can be ensured:

- Assume that the one-hop link between the end device and the network device makes use of certificate based authentication like IEEE 802.1AR [AR] certificates
- Check data integrity and perform source validation by using an optional keyed-hash message authentication code (HMAC) to protect

the Discovery and Configuration message exchanges. This type of message authentication allows communicating parties to verify that the contents of the message have not been altered and that the source is authentic. Use of this mechanism is optional and is controlled through a user-configurable attribute.

7. IANA Considerations

This memo includes no request to IANA.

Note: the section will be removed during conversion into an RFC by the RFC Editor.

8. Further and Related Work

- Standard extensions to the IEEE 802.1AB (LLDP) [LLDP] protocol are developed by the IEEE 802.1 Working Group. The relevant project is IEEE 802.1Qcj [AA] for 'Automatic Attachment to Provider Backbone Bridges (PBB) services'.
- Element Discovery could be implemented by using one of the protocols that implement the Manufacturer User Description Specification [MUD-SPEC]. The MUD LLDP Extension, the MUD URL DHCP Option and the MUD URC X.509 Extension defined in [MUD-SPEC] can be used for the purpose of instantiating the Discovery process.

9. Acknowledgments

The first version of this document that introduced the auto attach concept was co-authored by Nigel Bragg and Ludovic Beliveau.

We would like to thank the following people (in no particular order) for their contributions:

Zenon Kuc

Cristian Mema

Roger Lapuh

Craig Griffin

Chris Buerger

Keith Krajewski

Eliot Lear

This document was prepared using 2-Word-v2.0.template.dot.

10. References

10.1. Normative References

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

[LLDP] IEEE STD 802.1AB, "IEEE Standard for Local and Metropolitan Area Networks-- Station and Media Access Control Connectivity Discovery", 2005.

[AR] IEEE STD 802.1AR, "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", 2009.

10.2. Informative References

[AA] IEEE 802.1Qcj, "Standard for Local and Metropolitan Area Networks-Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks Amendment: Automatic Attachment to Provider Backbone Bridging (PBB) services"

[MUD-SPEC] Lear E. and R. Droms, "Manufacturer Usage Description Specification",draft-lear-ietf-netmod-mud-04 (work in progress), August 2016.

Authors' Addresses

Paul Unbehagen Jr., editor
Avaya
1300 W. 120th Avenue
Westminster, CO 80234
US

Email: unbehagen@avava.com

Dan Romascanu,
Avaya
Azrieli Center Holon
26, HaRokhmim Str., Bldg. D
Holon, 5885849

Israel

Phone: +972-3-645-8414
Email: dromasca@avaya.com

John Seligson
Avaya
4655 Great America Parkway
Santa Clara, CA 95054
US

Email: jseligso@avaya.com

Carl Keene
Avaya
600 Technology Park Dr
Boston, MA 01821
US

Email: ckeene@avava.com