RTGWG
Internet-Draft
Intended status: Standards Track
Expires: December 30, 2018

X. Ding
F. Zheng
Huawei
R. Wilton
Cisco Systems

June 28, 2018

YANG Data Model for ARP
draft-ding-rtgwg-arp-yang-model-02

Abstract

This document defines a YANG data model to describe Address Resolution Protocol (ARP) configurations. The data model performs as a guideline for configuring ARP capabilities on a system. It is intended this model be used by service providers who manipulate devices from different vendors in a standard way.

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). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on December 30, 2018.

Table of Contents

1. Introduction

This document defines a YANG [RFC7950] data model for Address Resolution Protocol [RFC826] implementation and identification of some common properties within a device. Devices have common properties that need to be configured and monitored in a standard way. This document is intended to present universal ARP protocol configuration and many vendors can implement it.

The data model convers configuration of system parameters of ARP, such as static ARP entries, timeout for dynamic ARP entries, interface ARP, proxy ARP, and so on. It also provides information about running state of ARP implementations.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, [RFC2119].

The following terms are defined in [RFC6241] and are not redefined here:

- client
- configuration data
- server
- state data

1.2. Tree Diagrams

A simplified graphical representation of the data model is presented in Section 3.

- Brackets "[" and "]" enclose list keys.
- Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

Tree diagrams used in this document use the notation defined in [RFC8340].

2. Problem Statement

This document defines a YANG [RFC7950] configuration data model that may be used to configure the ARP feature running on a system. Data model "ietf-ip" [I-D.ietf-netmod-rfc7277bis] covers the address mapping functionality. However, this functionality is strictly dependent on IPv4 networks, and many ARP related functionalities are missing, e.g. device global ARP entries and control, configuration related to dynamic ARP learning, proxy ARP, gratuitous ARP, etc.

The data model makes use of the YANG "feature" construct which allows implementations to support only those ARP features that lie within their capabilities. It is intended this model be used by service providers who manipulate devices from different vendors in a standard way.

This model can be used to configure the ARP applications for discovering the link layer address associated with a given Internet layer address.

3. Design of the Data Model

This data model intends to describe the processing that a protocol finds the hardware address, also known as Media Access Control (MAC) address, of a host from its known IP address. These tasks include, but are not limited to, adding a static entry in the ARP cache, configuring dynamic ARP learning, proxy ARP, gratuitous ARP. There are two kind of ARP configurations: global ARP configuration, which is across all interfaces on the device, and per interface ARP configuration.

3.1. ARP Caching

ARP caching is the method of storing network addresses and the associated data-link addresses in memory for a period of time as the addresses are learned. This minimizes the use of valuable network resources to broadcast for the same address each time a datagram is sent.

There are static ARP cache entries and dynamic ARP cache entries. Static entries are manually configured and kept in the cache table on a permanent basis. Dynamic entries are added by vendor software, kept for a period of time, and then removed. We can specify how long an entry remains in the ARP cache. If we specify a timeout of 0 seconds, entries are never cleared from the ARP cache.

3.2. proxy ARP

Proxy ARP [RFC1027] can be configured to enable the switch to respond to ARP queries for network addresses by offering its own Ethernet media access control (MAC) address. With proxy ARP enabled, the switch captures and routes traffic to the intended destination.

3.3. gratuitous ARP

Gratuitous ARP requests help detect duplicate IP addresses. A gratuitous ARP is a broadcast request for a router's own IP address. If a router or switch sends an ARP request for its own IP address and no ARP replies are received, the router- or switch-assigned IP address is not being used by other nodes. However, if a router or switch sends an ARP request for its own IP address and an ARP reply is received, the router- or switch-assigned IP address is already being used by another node.

3.4. ietf-arp Module

This module has one top level container, ARP, which consists of two second level containers, which are used for static entries configuration and global parameters control.

```
module: ietf-arp
    +--rw arp
       +--rw global-static-entries {global-static-entries}?
       |  +--rw static-entry* [ip-address]
       |     +--rw ip-address     inet:ipv4-address-no-zone
       |     +--rw mac-address    yang:mac-address
       +--rw global-control
          +--rw enable-learning?   boolean
          +--rw enable-proxy?      boolean
  augment /if:interfaces/if:interface:
    +--rw arp-dynamic-learning
       +--rw expire-time?     yang:timeticks
       +--rw learn-disable?   boolean
       +--rw proxy
       |  +--rw mode?   enumeration
       +--rw probe
       |  +--rw interval?   uint8
       |  +--rw times?      uint8
       |  +--rw unicast?    boolean
       +--rw gratuitous
       |  +--rw enable?     boolean
       |  +--rw interval?   uint32
       |  +--rw drop?       boolean
       +--ro statistics
          +--ro in-requests-pkts?      uint16
          +--ro in-replies-pkts?       uint16
          +--ro in-gratuitous-pkts?    uint16
          +--ro out-requests-pkts?     uint16
          +--ro out-replies-pkts?      uint16
          +--ro out-gratuitous-pkts?   uint16
  augment /if:interfaces/if:interface/ip:ipv4/ip:neighbor:
    +--ro remaining-expire-time?   uint32
```

4. ARP YANG Module

This section presents the ARP YANG module defined in this document. This YANG module imports typedefs from [RFC6991].

<CODE BEGINS>file "ietf-arp@2018-01-27.yang"

```
module ietf-arp {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-arp";
  prefix arp;

  import ietf-inet-types {
    prefix inet;
        reference "RFC 6991: INET Types Model";
  }

  import ietf-yang-types {
    prefix yang;
        reference "RFC 6991: yang Types Model";
  }

  import ietf-interfaces {
    prefix if;
    description
      "A Network Management Datastore Architecture (NMDA)
       compatible version of the ietf-interfaces module
       is required.";
  }
  import ietf-ip {
    prefix ip;
    description
      "A Network Management Datastore Architecture (NMDA)
       compatible version of the ietf-ip module is
       required.";
  }

  organization
    "IETF Routing Area Working Group (rtgwg)";
  contact
    "WG Web: <http://tools.ietf.org/wg/rtgwg/>
     WG List: <mailto: rtgwg@ietf.org>
     Editor: Xiaojian Ding
         dingxiaojian1@huawei.com
     Editor: Feng Zheng
         habby.zheng@huawei.com
     Editor: Robert Wilton
         rwilton@cisco.com";
  description
    "Address Resolution Protocol (ARP) management, which includes
     static ARP configuration, dynamic ARP learning, ARP entry query,
     and packet statistics collection.";

  revision 2018-01-27 {
    description
```

```
      "Init revision";
       //   NOTE TO RFC EDITOR:
     // Please replace the following reference
     // to draft-ding-rtgwg-arp-yang-model-02 with
     // RFC number when published (i.e. RFC xxxx).
    reference
         "draft-ding-rtgwg-arp-yang-model-02";
  }

      /*
      * Features
      */

    feature global-static-entries {
    description
      "This feature indicates that the device allows static entries
       to be configured globally.";
  }


  container arp {
    description
      "Address Resolution Protocol (ARP) management, which includes
        static ARP configuration, dynamic ARP learning, ARP entry
        query, and packet statistics collection.";

    container global-static-entries {
      if-feature "global-static-entries";
      description
        "Set a global static ARP entry, which is independent of the
interface.";
      list static-entry {
        key "ip-address";
        description
          "List of ARP static entries that can be configured globally.";
        leaf ip-address {
          type inet:ipv4-address-no-zone;
          description
            "IP address, in dotted decimal notation.";
        }
        leaf mac-address {
          type yang:mac-address;
          mandatory true;
          description
            "MAC address in the format of H-H-H, in which H is
                        a hexadecimal number of 1 to 4 bits.";
        }
      }
```

```
    }
```

```
    container global-control {
      description
        "Set global control parameters, which are independent of interface.";
        leaf enable-learning {
          type boolean;
                  default "true";
          description
            "Enables or disables global dynamic ARP learning.
                        If 'true', then enforcement is enabled.
                        If 'false', then enforcement is disabled.";
        }
        leaf enable-proxy {
          type boolean;
          default "true";
          description
            "Proxy ARP is enabled by default; perform this
                        task to globally disable proxy ARP on all interfaces.";
        }
    }
        }
   augment "/if:interfaces/if:interface" {
    description
      "Augment interface configuration with parameters of ARP.";
    container arp-dynamic-learning {
      description
        "Support for ARP configuration on interfaces.";
      leaf expire-time {
        type yang:timeticks {
          range "60..86400";
        }
        units "second";
        description
          "Aging time of a dynamic ARP entry.";
      }
      leaf learn-disable {
        type boolean;
        default "false";
        description
          "Whether dynamic ARP learning is disabled on an interface.
                  If the value is True, dynamic ARP learning is disabled.
                  If the value is False, dynamic ARP learning is enabled.";
      }

      container proxy {
        description
          "Configuration parameters for proxy ARP";
                leaf mode {
                  type enumeration {
```

```
                        enum DISABLE {
                          description
                                "The system should not respond to ARP requests
that
                                do not specify an IP address configured on the
local
                                subinterface as the target address.";
                        }
                        enum REMOTE_ONLY {
                          description
                                "The system responds to ARP requests only when
the
                                sender and target IP addresses are in different
                                subnets.";
                        }
                        enum ALL {
                          description
                                "The system responds to ARP requests where the
sender
                                and target IP addresses are in different
subnets, as well
                                as those where they are in the same subnet.";
                        }
                    }
                    default "DISABLE";
                    description
                          "When set to a value other than DISABLE, the local
system should
                          respond to ARP requests that are for target addresses
other than
                          those that are configured on the local subinterface
using its own
                          MAC address as the target hardware address. If the
REMOTE_ONLY
                          value is specified, replies are only sent when the
target address
                          falls outside the locally configured subnets on the
interface,
                          whereas with the ALL value, all requests, regardless of
their
                          target address are replied to.";
                    reference "RFC1027: Using ARP to Implement Transparent Subnet
Gateways";
                 }
               }

               container probe {
                 description
```

```
        "Common configuration parameters for all ARP probe.";
      leaf interval {
        type uint8 {
          range "1..5";
        }
        units "second";
        description
          "Interval for detecting dynamic ARP entries.";
      }
      leaf times {
        type uint8 {
          range "0..10";
        }
```

```
          description
            "Number of aging probe attempts for a dynamic ARP entry.
             If a device does not receive an ARP reply message after
             the number of aging probe attempts reaches a specified
             number,thedynamic ARP entry is deleted.";
        }
        leaf unicast {
          type boolean;
          default "false";
          description
            "Send unicast ARP aging probe messages for a dynamic ARP
             entry.";
        }
      }

      container gratuitous {
        description
          "Configure gratuitous ARP.";
        leaf enable {
          type boolean;
          default "false";
          description
            "Enable or disable sending gratuitous-arp packet on
             interface.";
        }
        leaf interval {
          type uint32 {
            range "1..86400";
          }
          units "second";
          description
            "The interval of sending gratuitous-arp packet on the
             interface.";
        }
        leaf drop {
          type boolean;
          default "false";
          description
            "Drop the receipt of gratuitous ARP packets on the interface.";
        }
      }

      container statistics {
        config false;
        description
          "IP ARP Statistics information on interfaces";
        leaf in-requests-pkts {
          type uint16;
```

```
          description
            "Total ARP requests received";
        }
        leaf in-replies-pkts {
          type uint16;
          description
            "Total ARP replies received";
        }
        leaf in-gratuitous-pkts {
          type uint16;
          description
            "Total gratuitous ARP received";
        }
        leaf out-requests-pkts {
          type uint16;
          description
            "Total ARP requests sent";
        }
        leaf out-replies-pkts {
          type uint16;
          description
            "Total ARP replies sent";
        }
        leaf out-gratuitous-pkts {
          type uint16;
          description
            "Total gratuitous ARP sent";
        }
      }
    }
  }


  augment "/if:interfaces/if:interface/ip:ipv4/ip:neighbor" {
    description
      "Augment neighbor list with parameters of ARP,
       eg., support for remaining expire time query on interfaces.";
    leaf remaining-expire-time {
      type uint32;
      config false;
      description
        "Remaining expire time of a dynamic ARP entry. ";
    }
   }


}
```

5. Data Model Examples

This section presents a simple but complete example of configuring static ARP entries and dynamic learning, based on the YANG modules specified in Section 4.

5.1. Static ARP Entries

Requirement:
Enable static ARP entry global configuration (not rely on interface).

```
<config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
   <arp xmlns="urn:ietf:params:xml:ns:yang:ietf-arp">
      <static-tables>
         <ip-address> 10.2.2.3 </ip-address>
         <mac-address> 00e0-fc01-0000 </mac-address>
      </static-tables>
   </arp>
```

Requirement:
Enable static ARP entry configuration on interface (defined in draft [I-D.ietf-netmod-rfc7277bis]).

```
<config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
   <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
      <neighbor>
         <ip-address> 10.2.2.3 </ip-address>
         <mac-address> 00e0-fc01-0000 </mac-address>
         <if-name> GE1/0/1 </if-name>
      </neighbor>
   </ipv4>
```

5.2. ARP Dynamic Learning

Requirement:
Enable ARP dynamic learning configuration.

```
   <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
      <arp-dynamic-learning xmlns="urn:ietf:params:xml:ns:yang:ietf-arp-
dynamic-learning">
         <if-name> GE1/0/1 </if-name>
         <expire-time>1200</expire-time>
         <learn-disable>false</learn-disable>
         <proxy>
                   <mode>DISABLE</mode>
                </proxy>
         <probe>
            <interval>5</interval>
            <times>3</times>
            <unicast>false</unicast>
         </probe>
         <gratuitous>
            <gratuitous-enable>false<gratuitous-enable>
            <interval>60</interval>
            <drop>false</drop>
         <gratuitous>
      </arp-dynamic-learning>
```

6. Security Considerations

The YANG module defined in this document is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC6536] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

These are the subtrees and data nodes and their sensitivity/ vulnerability:

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations.

7. Acknowledgments

The authors wish to thank Alex Campbell and Reshad Rahman, Qin Wu, many others for their helpful comments.

8. References

8.1. Normative References

[I-D.ietf-netmod-rfc7223bis]
Bjorklund, M., "A YANG Data Model for Interface Management", draft-ietf-netmod-rfc7223bis-03 (work in progress), January 2018.

[I-D.ietf-netmod-rfc7277bis]
Bjorklund, M., "A YANG Data Model for IP Management", draft-ietf-netmod-rfc7277bis-03 (work in progress), January 2018.

[RFC1027] Carl-Mitchell, S. and J. Quarterman, "Using ARP to implement transparent subnet gateways", RFC 1027, DOI 10.17487/RFC1027, October 1987, <https://www.rfc-editor.org/info/rfc1027>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

8.2. Informative References

[RFC0826] Plummer, D., "An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware", STD 37, RFC 826, DOI 10.17487/RFC0826, November 1982, <https://www.rfc-editor.org/info/rfc826>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

Authors' Addresses

Xiaojian Ding
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: dingxiaojian1@huawei.com

Feng Zheng
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: habby.zheng@huawei.com

Robert Wilton
Cisco Systems

Email: rwilton@cisco.com