RTGWG
Internet-Draft
Intended status: Standards Track
Expires: July 15, 2018

X. Ding
F. Zheng
Huawei

January 11, 2018

YANG Data Model for ARP
draft-ding-rtgwg-arp-yang-model-00

Abstract

This document defines a YANG data model to describe Address Resolution Protocol (ARP) configurations. 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 July 15, 2018.

Table of Contents

1. Introduction

This document defines a YANG [RFC6020] data model for Address Resolution Protocol [RFC826] implementation and identification of some common properties within a device containing a Network Configuration Protocol (NETCONF) server. Devices that are managed by NETCONF and perhaps other mechanisms have common properties that need to be configured and monitored in a standard way.

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.

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. YANG models can be used with network management protocols such as NETCONF [RFC6241] to install, manipulate, and delete the configuration of network devices.

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 module 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 ARP cache entry timeout, and clearing dynamic entries from the ARP cache.

This data model has one top level container, ARP, which consists of several second level containers. Each of these second level containers describes a particular category of ARP handling, such as defining static mapping between an IP address (32-bit address) and a Media Access Control (MAC) address (48-bit address).

```
module: ietf-arp
    +--rw arp
       +--rw arp-static-tables
       |  +--rw arp-static-table* [ip-address]
       |     +--rw ip-address     inet:ipv4-address-no-zone
       |     +--rw mac-address    yang:mac-address
       +--ro arp-statistics
          +--ro global-statistics
          |  +--ro requests-received?     uint32
          |  +--ro replies-received?      uint32
          |  +--ro gratuitous-received?   uint32
          |  +--ro requests-sent?         uint32
          |  +--ro replies-sent?          uint32
          |  +--ro gratuitous-sent?       uint32
          |  +--ro drops-received?        uint32
          |  +--ro total-received?        uint32
          |  +--ro total-sent?            uint32
          |  +--ro arp-dynamic-count?     uint32
          |  +--ro arp-static-count?      uint32
          +--ro local-statistics
             +--ro arp-if-statistics* [if-name]
                +--ro if-name      -> /if:interfaces/interface/name
                +--ro requests-received?     uint32
                +--ro replies-received?      uint32
                +--ro gratuitous-received?   uint32
                +--ro requests-sent?         uint32
                +--ro replies-sent?          uint32
                +--ro gratuitous-sent?       uint32
  augment /if:interfaces/if:interface/ip:ipv4/ip:neighbor:
  augment /if:interfaces-state/if:interface/ip:ipv4/ip:neighbor:
    +--ro vrf-name?      arp:routing-instance-ref
    +--ro expire-time?   uint32
  augment /if:interfaces/if:interface:
    +--rw expire-time?               uint32
    +--rw arp-learn-disable?         boolean
    +--rw proxy-enable?              boolean
    +--rw probe-interval?            uint8
    +--rw probe-times?               uint8
    +--rw probe-unicast?             boolean
    +--rw arp-gratuitous?            boolean
    +--rw arp-gratuitous-interval?   uint32
```

```
      +--rw arp-gratuitous-drop?       boolean
      +--rw arp-if-limit* [vlan-id]
         +--rw vlan-id            uint16
         +--rw limit-number       uint32
         +--rw threshold-value?   uint32
    augment /if:interfaces-state/if:interface:
      +--ro requests-received?     uint32
      +--ro replies-received?      uint32
      +--ro gratuitous-received?   uint32
      +--ro requests-sent?         uint32
      +--ro replies-sent?          uint32
      +--ro gratuitous-sent?       uint32
```

4. YANG Module

This section presents the YANG module for the ARP data model defined in this document.

```
<CODE BEGINS> file "ietf-arp@2018-1-11.yang"
module ietf-arp {
  namespace "urn:ietf:params:xml:ns:yang:ietf-arp";
  prefix arp;

 // import some basic types

  import ietf-inet-types {
    prefix inet;
  }

  import ietf-yang-types {
    prefix yang;
  }

  import ietf-interfaces {
    prefix if;
  }

  import ietf-ip {
    prefix ip;
  }

  import ietf-network-instance {
    prefix ni;
  }
```

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

  revision 2017-10-18 {
    description
      "Init revision";
    reference
      "RFC XXX: ARP (Address Resolution Protocol) YANG data model.";
  }

/*grouping*/

  grouping arp-prob-grouping {
    description
      "Common configuration for all ARP probe.";
    leaf probe-interval {
      type uint8 {
        range "1..5";
      }
          units "second";
      description
        "Interval for detecting dynamic ARP entries.";
    }
    leaf probe-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, the dynamic ARP entry is deleted.";
    }
    leaf probe-unicast {
      type boolean;
      default "false";
      description
```

```
        "Send unicast ARP aging probe messages for a dynamic ARP
                entry.";
    }
  }

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

  grouping arp-statistics-grouping {
    description "IP ARP Statistics information";
    leaf requests-received {
      type uint32;
      description "Total ARP requests received";
    }
    leaf replies-received {
      type uint32;
      description "Total ARP replies received";
    }
    leaf gratuitous-received {
      type uint32;
      description "Total gratuitous ARP received";
    }
    leaf requests-sent {
      type uint32;
```

```
      description "Total ARP requests sent";
    }
    leaf replies-sent {
      type uint32;
      description "Total ARP replies sent";
    }
    leaf gratuitous-sent {
      type uint32;
      description "Total gratuitous ARP sent";
    }
  }

  /* Typedefs */

  typedef routing-instance-ref {
    type leafref {
      path "/ni:network-instances/ni:network-instance/ni:name";
    }
    description
      "This type is used for leafs that reference a routing instance
          configuration.";
  }

  /* Configuration data nodes */

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

    container arp-static-tables {
        //config false;
      description
        "List of ARP entries that can be configured.";
      list arp-static-table {
                  key "ip-address";
        description
          "Short static ARP table. By default, the system ARP table is
                  empty, and address mappings are implemented by dynamic
                  ARP.";

        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.";
      }

    }
  }//End of arp-tables

  container arp-statistics {
    config false;
    description
      "List of ARP packet statistics.";
    container global-statistics {
      description
        "ARP packet statistics.";
      uses arp-statistics-grouping;
      leaf drops-received {
        type uint32 {
          range "0..4294967294";
        }
        description
          "Number of ARP packets discarded.";
      }
      leaf total-received {
        type uint32 {
          range "0..4294967294";
        }
        description
          "Total number of ARP received packets.";
      }
      leaf total-sent {
        type uint32 {
          range "0..4294967294";
        }
        description
          "Total number of ARP sent packets.";
      }
      leaf arp-dynamic-count {
        type uint32 {
          range "0..4294967294";
        }
        description
          "Number of dynamic ARP count.";
      }
      leaf arp-static-count {
        type uint32 {
```

```
            range "0..4294967294";
          }
          description
            "Number of static ARP count.";
        }
      }
          container local-statistics {
        list arp-if-statistics {
        key "if-name";
        description
          "ARP statistics on interfaces. ARP statistics on all
                  interfaces are displayed in sequence.";
        leaf if-name {
          type leafref {
            path "/if:interfaces/if:interface/if:name";
          }
          description
            "Name of an interface where ARP statistics to be
                       displayed reside.";
        }
        uses arp-statistics-grouping;
      }

           description
           "foo";
          }
        }
        }

   //End of arp-static-tables

         augment "/if:interfaces/if:interface/ip:ipv4/ip:neighbor"
                 {
            description
          "Long static ARP table has been defined in
                  /if:interfaces/if:interface/ip:ipv4/ip:neighbor";
             }
          augment "/if:interfaces-state/if:interface/ip:ipv4/ip:neighbor" {
            description
          "List of ARP entries that can be queried.";

                  leaf vrf-name {
          type arp:routing-instance-ref;
                     config false;
          description
            "Name of the VPN instance to which an ARP entry
                       belongs.";
```

```
        }

        leaf expire-time {

          type uint32 {
            range "1..1440";
          }
                    config false;
          description
            "Aging time of a dynamic ARP entry. ";
        }

          }

    augment "/if:interfaces/if:interface" {
      description
        "List of ARP Interface configurations.including the aging time,
                  probe interval, number of aging probe attempts, ARP
                  learning status, and ARP proxy.";

        leaf expire-time {
          type uint32 {
            range "60..86400";
          }
                  units "second";
          description
            "Aging time of a dynamic ARP entry.";
        }
        leaf arp-learn-disable {
          type boolean;
          default "false";
          description
            "Whether dynamic ARP learning is disabled.
                        If the value is True, dynamic ARP learning
                        is disabled. If the value is False, dynamic
                        ARP learning is enabled.";
        }
        leaf proxy-enable {
          type boolean;
          default "false";
          description
            "Enable proxy ARP.";
        }
        uses arp-prob-grouping;
        uses arp-gratuitous-grouping;

        list arp-if-limit {
          key "vlan-id";
```

```
          description
            "Maximum number of dynamic ARP entries that an
                     interface can learn. If the number of ARP
                         entries that an interface can learn changes
                         and the number of the learned ARP entries
                         exceeds the changed value, the interface cannot
                         learn additional ARP entries. The system prompts
                         you to delete the excess ARP entries.";

           leaf vlan-id {
            type uint16 {
              range "0..4094";
            }
            description
              "ID of the VLAN where ARP learning is restricted.
                           This parameter can be set only on Layer 2
                           interfaces and sub-interfaces. Ethernet, GE, VE,
                           and Eth-Trunk interfaces can be both Layer 3 and
                           Layer 2 interfaces. When they work in Layer 3 mode,
                           they cannot have VLANs configured. When they work
                           in Layer 2 mode, they must have VLANs configured.
                           Ethernet, GE, and Eth-Trunk sub-interfaces can be
                           both common and QinQ sub-interfaces.";
          }
          leaf limit-number {
            type uint32 {
              range "1..65536";
            }
            mandatory true;
            description
              "Maximum number of dynamic ARP entries that an
                              interface can learn.";
          }
          leaf threshold-value {
            type uint32 {
              range "60..100";
            }
            must "not(not(../limit-number))"{
                              description
                "Upper boundary must be higher than lower boundary.";
                             }
              description
                "Alarm-Threshold for Maximum number of ARP entries
                              that an interface can learn.";
          }
        }
      }
        augment "/if:interfaces-state/if:interface" {
```

```
        description
          "ARP statistics on interfaces. ARP statistics on all
                  interfaces are displayed in sequence.";
        uses arp-statistics-grouping;
      }
    // End of arp-statistics

}

<CODE ENDS>
```

5. Data Model Examples

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

5.1. Static ARP entries

Requirement:
Enable common static ARP entry configuration.

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

Requirement:
Enable long static ARP entry configuration.

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

5.2. ARP interfaces

Requirement:
Enable static ARP interface configuration.

```
<config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
   <arp xmlns="urn:ietf:params:xml:ns:yang:ietf-arp">
      <arp-interfaces>
            <if-name> GE1/0/1 </if-name>
            <expire-time>1200</expire-time>
                <arp-learn-disable>false</arp-learn-disable>
            <proxy-enable>false</proxy-enable>
                <probe-interval>5</probe-interval>
                <probe-times>3</probe-times>
                <probe-unicast>false</probe-unicast>
                <arp-gratuitous>false</arp-gratuitous>
                <arp-gratuitous-interval>60</arp-gratuitous-interval>
                <arp-gratuitous-drop>false</arp-gratuitous-drop>
                <arp-if-limits>
                   <vlan-id>3</vlan-id>
                   <limit-number>65535</limit-number>
                   <threshold-value>80</threshold-value>
            </arp-if-limits>
       </arp-interfaces>
   </arp>
```

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. Conclusions

TBD.

8. References

8.1. Normative References

[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>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[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., "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>.

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