802.3 YANG
Base Interface Statistics

Rob Wilton

Cisco

2017 Jan 9 Pleminary - Remote

802.3 YANG Base Interface Module

Agenda:

- Introduction
- Proposed Ethernet Interface Statistics
- Questions/issues

Introduction

- Aim: To discuss/agree on base Ethernet interface statistics in YANG
- Also agree what should go in the description for an individual counter:
  - Specifically how much repetition?
- After agreement, I'll write them up for the next revision of the model

Types of counter

- All interfaces have ietf-interfaces statistics:
  - No need/benefit in any duplication
- So Ethernet statistics are in addition, in a separate ethernet/statistics container, contains:
  - Etherlike MIB style stats
  - A subset of RMON Ethernet counters
- I propose that it does not contain any of:
  - CSMA/CD specific counters (*they go in separate module*)
  - Flow control counters (*moved under flow control?)*
  - FEC/BER related counters (*would go with those features?*)
  - LPI? (*not sure about this one!)*

IETF interface YANG statistics

(For reference. Every Ethernet interface always has these)

```
+--ro statistics
   +--ro discontinuity-time    yang:date-and-time

   +--ro in-octets?            yang:counter64 = (total good bytes, inc fcs chars)
   +--ro in-unicast-pkts?      yang:counter64 = good uni pkts    (not drop/error/
   +--ro in-broadcast-pkts?    yang:counter64 = good bcast pkts           unknown)
   +--ro in-multicast-pkts?    yang:counter64 = good mcast pkts      “
   +--ro in-discards?          yang:counter32 = e.g. QoS/ACL drops
   +--ro in-errors?            yang:counter32 = e.g. Frame errors
   +--ro in-unknown-protos?    yang:counter32 = e.g. Unknown proto drops.

   +--ro out-octets?           yang:counter64
   +--ro out-unicast-pkts?     yang:counter64
   +--ro out-broadcast-pkts?   yang:counter64
   +--ro out-multicast-pkts?   yang:counter64
   +--ro out-discards?         yang:counter32
   +--ro out-errors?           yang:counter32
```

IETF interface YANG counters

Reference YANG description for a counter (it is based on IFMIB)

```
leaf in-unicast-pkts {
            type yang:counter64;
            description
              "The number of packets, delivered by this sub-layer to a
               higher (sub-)layer, that were not addressed to a
               multicast or broadcast address at this sub-layer.

               Discontinuities in the value of this counter can occur
               at re-initialization of the management system, and at
               other times as indicated by the value of
               'discontinuity-time'.";
            reference
              "RFC 2863: The Interfaces Group MIB - ifHCInUcastPkts";
          }
```

RMON counters

I propose that we use a subset of the historical RMON MIB Ethernet counters that we want:

- Advise IETF that this is what we are hoping to do, by:
  - Emailing IEEE/IETF liaison alias + NETMOD WG.
  - Also raising at Jan 30 IEEE/IETF coordination meeting to:
    - Check whether any WG is planning, likely, or anticipated to develop an RMON YANG model (noting that they could always just exclude Ethernet statistics anyway).
    - Check that they don’t have any concerns with us proceeding and defining the necessary RMON Ethernet subset that is still applicable.
  - Note - I do not anticipate any concern from IETF.

Existing RMON MIB Ethernet counters

(For reference purposes only, defined in RFC 2819)

```
etherStatsDropEvents               Counter32,  // Drop due to lack of resources
etherStatsOctets                   Counter32,  // Total bytes (good + bad)
etherStatsPkts                     Counter32,  // Total pkts (good + bad)
etherStatsBroadcastPkts            Counter32,  // Total good bcast pkts
etherStatsMulticastPkts            Counter32,  // Total good mcast pkts
etherStatsCRCAlignErrors           Counter32,  // 64 <= pkt <= 1518, bad CRC/align
etherStatsUndersizePkts            Counter32,  // pkt < 64, good CRC
etherStatsOversizePkts             Counter32,  // pkt > 1518, good CRC
etherStatsFragments                Counter32,  // pkt < 64, bad CRC
etherStatsJabbers                  Counter32,  // pkt > 1518, bad CRC
etherStatsCollisions               Counter32,  // Collision estimate
etherStatsPkts64Octets             Counter32,  // 64 byte pkts
etherStatsPkts65to127Octets        Counter32,  // 65 – 127 byte pkts
etherStatsPkts128to255Octets       Counter32,  // 128 – 255 byte pkts
etherStatsPkts256to511Octets       Counter32,  // 256 – 511 byte pkts
etherStatsPkts512to1023Octets      Counter32,  // 512 – 1023 byte pkts
etherStatsPkts1024to1518Octets     Counter32,  // 1024 – 1518 byte pkts
```

Proposed Ethernet Counters (Ingress)

(Combined Etherlike MIB and RMON MIB)

```
This counters are in addition to the ietf-interfaces statistics.

in-octets-total             counter64, // Total received bytes (good + bad)
in-pkts-total               counter64, // Total received pkts (good + bad)
in-pkts-errors-fcs          counter64, // 64 <= pkt <= 1518, bad CRC or alignment
in-pkts-errors-runt         counter64, // pkt < 64, good frame
in-pkts-errors-fragment     counter64, // pkt < 64, bad frame
in-pkts-errors-giant        counter64, // pkt > MRU, good frame
in-pkts-errors-jabber       counter64, // pkt > MRU, bad frame
in-pkts-drop-unknown-mac    counter64, // good frame, dropped due to unknown DMAC
in-errors-symbol?           counter64, // symbol errors (should this go elsewhere)?
in-errors-unknown-opcode?   counter64, //               (Should this go elsewhere)?
in-pkts-64                  Counter64,
in-pkts-65-127              Counter64,
in-pkts-128-255             Counter64,
in-pkts-256-511             Counter64,
in-pkts-512-1023            Counter64,
in-pkts-1024-mru            Counter64,
```

Proposed Ethernet Counters (Egress)

(Combined Etherlike MIB and RMON MIB)

```
This counters are in addition to the ietf-interfaces statistics.

out-octets-total             counter64, // Total trans bytes (good + bad)
out-pkts-total               counter64, // Total trans pkts (good + bad)

out-pkts-64                  Counter64,
out-pkts-65-127              Counter64,
out-pkts-128-255             Counter64,
out-pkts-256-511             Counter64,
out-pkts-512-1023            Counter64,
out-pkts-1024-mru            Counter64,
```

Ethernet Statistics questions (1 of 2)

- I’ve assumed that we also have a state leaf returning the actual MRU that is used to determine if a frame is a giant/jabber.
- Should all error/drop pkt counters be 32 bit or 64 bit?
  - IETF interface error/drop counters are 32 bit, but looks like this could wrap very fast for a 100G interface with 64 byte packets.
- Do we need separate *runts* from *fragments* and *giants* from *jabbers*?
  - Could just have a single bad alignment/FCS counter.
- Do we need unicast/broadcast/multicast packet counts separate from the IETF interface equivalent?

Statistics questions (2 of 2)

- Should symbol errors and unknown opcodes go in a separate container?
- Do we still need histogram counters?
  - Ideally would want to extend buckets beyond 1518 up to 8K.
  - Could make entirely generic:
    - Generic list of (min frame size, max frame size, count)
    - Description would give some recommendations of common bucket definitions.
    - Benefit: Increased flexibility.
    - Downsides: increases complexity and memory usage.
  - Could be defined by 802.3 or perhaps elsewhere (as an augmentation)
  - Should these be part of the same Ethernet statistics container?

Ethernet Statistics Description Example

```
leaf in-pkts-errors-fcs {
  type yang:counter64;
  units frames;
   description
     "A count of receive frames that do not pass the FCS check, regardless of whether or not the
      frames are an integral number of octets in length, and regardless of the frame length.

      This count effective comprises aFrameCheckSequenceErrors and aAlignmentErrors added together.

      Note: Coding errors detected by the Physical Layer for speeds above 10 Mb/s will cause the
      frame to fail the FCS check.

      A frame that is counted by an instance of this object is also counted by the corresponding
      instance of 'in-errors' leaf defined in the ietf-interfaces YANG module (RFC 7223).

      Discontinuities in the values of this counter can occur at re-initialization of the management
      system, and at other times as indicated by the value of the 'discontinuity-time' leaf defined
      in the ietf-interfaces YANG module (RFC 7223).";
    reference
      "IEEE 802.3, 30.3.1.1.6, aFrameCheckSequenceErrors";
}
```

Statistics Description Questions

- Should I try and simplify these descriptions, or keep the text? The additional text may help ensure more consistent implementations.
- Discontinuity (and other common semantics) could be described in the parent container description statement, and then just referenced here.
- Any opinions?

Thank you!