Operations and Management Area Working Group
Internet-Draft
Intended status: Standards Track
Expires: 1 January 2026

J. Evans
O. Pylypenko
Amazon
J. Haas
Juniper Networks
A. Kadosh
Cisco Systems, Inc.
M. Boucadair
Orange

30 June 2025

Information and Data Models for Packet Discard Reporting
draft-ietf-opsawg-discardmodel-08

Abstract

This document defines an information model and a corresponding YANG data model for packet discard reporting. The information model provides an implementation-independent framework for classifying packet loss to enable automated network mitigation of unintended packet loss. The YANG data model specifies an implementation of this framework for network elements.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://o-pylypenko.github.io/draft-ietf-opsawg-discardmodel/draft-ietf-opsawg-discardmodel.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-opsawg-discardmodel/.

Discussion of this document takes place on the Operations and Management Area Working Group mailing list (mailto:opsawg@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/opsawg/. Subscribe at https://www.ietf.org/mailman/listinfo/opsawg/.

Source for this draft and an issue tracker can be found at https://github.com/o-pylypenko/draft-ietf-opsawg-discardmodel.

Status of This Memo

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

This Internet-Draft will expire on 1 January 2026.

Copyright Notice

Table of Contents

1. Introduction 3
2. Terminology 4
3. Problem Statement 5
4. Information Model 6
  4.1. Structure 6
  4.2. Sub-type Definitions 9
  4.3. "ietf-packet-discard-reporting-sx" YANG Module 10
5. Data Model 24
  5.1. Structure 24
  5.2. Implementation Requirements 26
  5.3. Usage Examples 27
  5.4. "ietf-packet-discard-reporting" YANG Module 28
6. Security Considerations 31
  6.1. Information Model 31
  6.2. Data Model 31
7. IANA Considerations 32
8. Contributors 32
9. Acknowledgments 33
10. References 33
  10.1. Normative References 33

10.2. Informative References 33
Appendix A. Where Do Packets Get Dropped? 36
Appendix B. Example Signal-to-mitigation Action Mapping 37
Appendix C. Implementation Experience 38
Appendix D. Full Information Model Tree 39
Appendix E. Full Data Model Tree 45
Authors' Addresses 49

1. Introduction

The primary function of a network is to transport and deliver packets according to service level objectives. For network operators, understanding both where and why packet loss occurs within a network is essential for effective operation. Device-reported packet loss provides the most direct signal for identifying service impact. While certain types of packet loss, such as policy-based discards, are intentional and part of normal network operation, unintended packet loss can impact customer services. To automate network operations, operators must be able to detect customer-impacting packet loss, determine its root cause, and apply appropriate mitigation actions. Precise classification of packet loss is crucial to ensure that anomalous packet loss is easily detected and that the right action is taken to mitigate the impact. Taking the wrong action can make problems worse; for example, removing a congested device from service can exacerbate congestion by redirecting traffic to other already congested links or devices.

Existing metrics for reporting packet loss, such as ifInDiscards, ifOutDiscards, ifInErrors, and ifOutErrors defined in MIB-II [RFC2863] and the YANG Data Model for Interface Management [RFC8343], are insufficient for automating network operations. First, they lack precision; for instance, ifInDiscards aggregates all discarded inbound packets without specifying the cause, making it challenging to distinguish between intended and unintended discards. Second, these definitions are ambiguous, leading to inconsistent vendor implementations. For example, in some implementations ifInErrors accounts only for errored packets that are dropped, while in others, it includes all errored packets, whether they are dropped or not. Many implementations support more discard metrics than these, however, they have been inconsistently implemented due to the lack of a standardised classification scheme and clear semantics for packet loss reporting. For example, [RFC7270] provides support for reporting discards per flow in IPFIX using forwardingStatus, however, the defined drop reason codes also lack sufficient clarity to facilitate automated root cause analysis and impact mitigation, e.g., the "For us" reason code.

This document defines an information model and corresponding YANG data model for packet loss reporting which address these issues. The information model provides precise classification of packet loss to enable accurate automated mitigation. The data model specifies a YANG implementation of this framework for network elements, while maintaining consistency through clear semantics.

The scope of this document is limited to reporting packet loss at Layer 3 and frames discarded at Layer 2. This document considers only the signals that may trigger automated mitigation actions and not how the actions are defined or executed.

Section 3 describes the problem space and requirements. Section 4 defines the information model and classification scheme. Section 5 specifies the corresponding YANG data model and implementation requirements together with a set of usage examples, and the complete YANG module definition. The appendices provide additional context and implementation guidance.

2. 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] [RFC8174] when, and only when, they appear in all capitals, as shown here.

A packet discard accounts for any instance where a packet is dropped by a device, regardless of whether the discard was intentional or unintentional.

Intended discards are packets dropped due to deliberate network policies or configurations designed to enforce security or Quality of Service (QoS). For example, packets dropped because they match an Access Control List (ACL) denying certain traffic types.

Unintended discards are packets that were dropped, which the network operator otherwise intended to deliver, i.e. which indicates an error state. There are many possible reasons for unintended packet loss, including: erroring links may corrupt packets in transit; incorrect routing tables may result in packets being dropped because they do not match a valid route; configuration errors may result in a valid packet incorrectly matching an ACL and being dropped.

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

3. Problem Statement

The fundamental problem for network operators is how to automatically detect when unintended packet loss is occurring and determine the appropriate action to mitigate it. For any network, there are a small set of potential actions that can be taken to mitigate customer impact when unintended packet loss is detected:

1. Take a problematic device, link, or set of devices and/or links out of service.

2. Return a device, link, or set of devices and/or links back into service.

3. Move traffic to other links or devices to alleviate congestion or avoid problematic paths.

4. Roll back a recent change to a device that might have caused the problem.

5. Escalate to a network operator as a last resort when automated mitigation is not possible.

The ability to select the appropriate mitigation action depends on four key features of packet loss:

FEATURE-DISCARD-LOCATION: Determines which devices, interfaces and/or flows are impacted.

FEATURE-DISCARD-RATE: The rate and/or magnitude of the discards, indicating the severity and urgency of the problem.

FEATURE-DISCARD-DURATION: The duration of the discards which helps to distinguish transient from persistent issues.

FEATURE-DISCARD-CLASS: The type or class of discards, which is crucial for selecting the appropriate of mitigation - for example: error discards may require taking faulty components out of service; no-buffer discards may require traffic redistribution; policy discards typically require no automated action

While FEATURE-DISCARD-LOCATION, FEATURE-DISCARD-RATE, and FEATURE-DISCARD-DURATION are implicitly supported by MIB-II [RFC2863] and the YANG Data Model for Interface Management [RFC8343], FEATURE-DISCARD-CLASS requires a more detailed classification scheme than they define. The following information model defines such a classification scheme to enable automated mapping from loss signals to appropriate mitigation actions.

4. Information Model

The information model is defined using YANG [RFC7950], with Data Structure Extensions [RFC8791], allowing the model to remain abstract and decoupled from specific implementations in accordance with [RFC3444]. This abstraction supports different data model implementations - for example, in YANG, IPFIX [RFC7011], gMNI [gMNI] or SNMPv3 [RFC3411] - while ensuring consistency across implementations. Using YANG for the information model enables this abstraction, leverages the community's familiarity with its syntax, and ensures lossless translation to the corresponding YANG data model, which is defined in Section 5.

4.1. Structure

The information model defines a hierarchical classification scheme for packet discards, which captures where in a device the discards are accounted (component), in which direction they were flowing (direction), whether they were successfully processed or discarded (type), what protocol layer they belong to (layer), and the specific reason for any discards (sub-types). This structure enables both high-level monitoring of total discards and more detailed triage to map to mitigation actions.

The abstract structure of the IM is depicted in Figure 1. The full YANG tree diagram of the IM is provided in Appendix D.

module: ietf-packet-discard-reporting-sx

```
  structure packet-discard-reporting:
    +-- control-plane {control-plane-stats}?
    |  +-- traffic* [direction]
    |  |  ...
    |  +-- discards* [direction]
    |     ...
    +-- interface* [name] {interface-stats}?
    |  +-- name        string
    |  +-- traffic* [direction]
    |  |  +-- direction    identityref
    |  |  +-- l2
    |  |  |  ...
    |  |  +-- l3
    |  |  |  ...
    |  |  +-- qos
    |  |     +-- class* [id]
    |  |        ...
    |  +-- discards* [direction]
    |     +-- direction    identityref
```

```
|     +-- l2
|     |  ...
|     +-- l3
|     |  ...
|     +-- errors
|     |  +-- l2
|     |  |  ...
|     |  +-- l3
|     |  |  ...
|     |  +-- internal
|     |     ...
|     +-- policy
|     |  +-- l2
|     |  |  ...
|     |  +-- l3
|     |     ...
|     +-- no-buffer
|        +-- class* [id]
|           ...
+-- flow* [direction] {flow-reporting}?
|  +-- direction    identityref
|  +-- traffic
|  |  +-- l2
|  |  |  ...
|  |  +-- l3
|  |  |  ...
|  |  +-- qos
|  |     +-- class* [id]
|  |        ...
|  +-- discards
|     +-- l2
|     |  ...
|     +-- l3
|     |  ...
|     +-- errors
|     |  +-- l2
|     |  |  ...
|     |  +-- l3
|     |  |  ...
|     |  +-- internal
|     |     ...
|     +-- policy
|     |  +-- l2
|     |  |  ...
|     |  +-- l3
|     |     ...
|     +-- no-buffer
|        +-- class* [id]
```

```
|          ...
+-- device {device-stats}?
   +-- traffic
   |  +-- l2
   |  |  ...
   |  +-- l3
   |  |  ...
   |  +-- qos
   |     +-- class* [id]
   |        ...
   +-- discards
      +-- l2
      |  ...
      +-- l3
      |  ...
      +-- errors
      |  +-- l2
      |  |  ...
      |  +-- l3
      |  |  ...
      |  +-- internal
      |     ...
      +-- policy
      |  +-- l2
      |  |  ...
      |  +-- l3
      |     ...
      +-- no-buffer
         +-- class* [id]
            ...
```

Figure 1: Abstract IM Tree Structure

The discard reporting can be organized into several types: control plane, interface, flow, and device. In order to allow for better mapping to underlying data models, the IM supports a set of "features" to declare the supported type.

A complete classification path follows the pattern: component/direction/type/layer/sub-type/sub-sub-type/.../metric. Appendix A illustrates where these discards typically occur in a network device. The elements of the tree are defined as follows:

* Component:

  - control-plane: discards of traffic to or from a device’s control plane.

- interface: discards of traffic to or from a specific network interface.
- flow: discards of traffic associated with a specific traffic flow.
- device: discards of traffic transiting the device.

* Direction:
  - ingress: counters for incoming packets or frames.
  - egress: counters for outgoing packets or frames.
* Type:
  - traffic: counters for successfully received or transmitted packets or frames.
  - discards: counters for packets or frames that were dropped.
* Layer:
  - l2: Layer 2 traffic and discards, i.e., frame and byte counts.
  - l3: Layer 3 traffic and discards, i.e., packet and byte counts.

The hierarchical structure allows for future extension while maintaining backward compatibility. New discard types can be added as new branches without affecting existing implementations.

The corresponding YANG module is defined in Section 4.3.

4.2. Sub-type Definitions

discards/policy/: These are intended discards, meaning packets dropped by a device due to a configured policy, including: ACLs, traffic policers, Reverse Path Forwarding (RPF) checks, DDoS protection rules, and explicit null routes

discards/error/: These are unintended discards due to errors in processing packets or frames. There are multiple sub-classes.

discards/error/l2/rx/: These are frames discarded due to errors in the received Layer 2 frame, including: CRC errors, invalid MAC addresses, invalid VLAN tags, frame size violations and other malformed frame conditions

discards/error/l3/rx/: These are discards which occur due to errors in the received packet, indicating an upstream problem rather than an issue with the device dropping the errored packets, including: header checksum errors, MTU exceeded, invalid packet errors, i.e., incorrect version, incorrect header length, invalid options and other malformed packet conditions

discards/error/l3/rx/ttl-expired: These are discards due to TTL (or Hop limit) expiry, which can occur for the following reasons: normal trace-route operations, end-system TTL/Hop limit set too low, routing loops in the network.

discards/error/l3/no-route/: These are discards which occur due to a packet not matching any route in the routing table, e.g., which may be due to routing configuration errors or may be transient discards during convergence.

discards/error/internal/: These are discards due to internal device issues, including: parity errors in device memory or other internal hardware errors. Any errored discards not explicitly assigned to other classes are also accounted for here.

discards/no-buffer/: These are discards due to buffer exhaustion, i.e. congestion related discards. These can be tail-drop discards or due to an active queue management algorithm, such as RED [RED93] or CODEL [RFC8289].

An example of possible signal-to-mitigation action mapping is provided in Appendix B.

4.3. "ietf-packet-discard-reporting-sx" YANG Module

The "ietf-packet-discard-reporting-sx" module uses the "sx" structure defined in [RFC8791].

```
<CODE BEGINS>
module ietf-packet-discard-reporting-sx {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-packet-discard-reporting-sx";
  prefix plr-sx;

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }
  import ietf-yang-structure-ext {
```

```
      prefix sx;
      reference
        "RFC 8791: YANG Data Structure Extensions";
    }

    organization
      "IETF OPSAWG (Operations and Management Area Working Group)";
    contact
      "WG Web:   https://datatracker.ietf.org/wg/opsawg/
       WG List:  mailto:opsawg@ietf.org

       Author:   John Evans
                 <mailto:jevanamz@amazon.co.uk>

       Author:   Oleksandr Pylypenko
                 <mailto:opyl@amazon.com>

       Author:   Jeffrey Haas
                 <mailto:jhaas@juniper.net>

       Author:   Aviran Kadosh
                 <mailto:akadosh@cisco.com>

       Author:   Mohamed Boucadair
                 <mailto:mohamed.boucadair@orange.com>";
    description
      "This module defines an information model for packet discard
       reporting.

       Copyright (c) 2025 IETF Trust and the persons identified as
       authors of the code.  All rights reserved.

       Redistribution and use in source and binary forms, with or
       without modification, is permitted pursuant to, and subject
       to the license terms contained in, the Revised BSD License
       set forth in Section 4.c of the IETF Trust's Legal Provisions
       Relating to IETF Documents
       (https://trustee.ietf.org/license-info).

       This version of this YANG module is part of RFC XXXX; see the
       RFC itself for full legal notices.";

    revision 2024-06-04 {
      description
        "Initial revision.";
      reference
        "RFC XXXX: Information and Data Models for Packet Discard
                   Reporting";
```

```
  }

  /*
   * Features
   */

  feature interface-stats {
    description
      "Indicates support of interface statistics on this
       device.";
  }

  feature device-stats {
    description
      "Indicates support of global device statistics on this
       device.";
  }

  feature control-plane-stats {
    description
      "Indicates support of control plane statistics on this
       device.";
  }

  feature flow-reporting {
    description
      "Indicates support of flow reporting on this device.";
  }

  /*
   * Identities
   */

  identity direction {
    description
      "Defines a direction for the reported statistics.";
  }

  identity ingress {
    base direction;
    description
      "Reports statistics for the received from the network
       packets.";
  }

  identity egress {
    base direction;
    description
```

```
        "Reports statistics for the sent to the network
         packets.";
    }

    identity address-family {
      description
        "Defines a type for the address family.";
    }

    identity ipv4 {
      base address-family;
      description
        "Identity for an IPv4 address family.";
    }

    identity ipv6 {
      base address-family;
      description
        "Identity for an IPv6 address family.";
    }

    identity all {
      base address-family;
      description
        "Identity for all address families.";
    }

    /*
     * Groupings
     */

    grouping basic-packets {
      description
        "Grouping for Layer 3 packet counters.";
      leaf packets {
        type yang:counter64;
        description
          "Number of Layer 3 packets.";
      }
    }

    grouping basic-packets-bytes {
      description
        "Grouping for Layer 3 packet and byte counters.";
      uses basic-packets;
      leaf bytes {
        type yang:counter64;
        description
```

```
          "Number of Layer 3 bytes.";
      }
    }

    grouping basic-frames {
      description
        "Grouping for Layer 2 frame counters.";
      leaf frames {
        type yang:counter64;
        description
          "Number of Layer 2 frames.";
      }
    }

    grouping basic-frames-bytes {
      description
        "Grouping for Layer 2 frame and byte counters.";
      uses basic-frames;
      leaf bytes {
        type yang:counter64;
        description
          "Number of Layer 2 bytes.";
      }
    }

    grouping l2-traffic {
      description
        "Layer 2 traffic counters.";
      uses basic-frames-bytes;
    }

    grouping ip {
      description
        "Layer 3 traffic counters per address family.";
      list address-family-stat {
        key "address-family";
        description
          "Reports per address family traffic counters.";
        leaf address-family {
          type identityref {
            base address-family;
          }
          description
            "Specifies an address family.";
        }
        uses basic-packets-bytes;
        container unicast {
          description
```

```
            "Unicast traffic counters.";
          uses basic-packets-bytes;
        }
        container multicast {
          description
            "Multicast traffic counters.";
          uses basic-packets-bytes;
        }
      }
    }

    grouping l3-traffic {
      description
        "Layer 3 traffic counters.";
      uses ip;
    }

    grouping qos {
      description
        "Quality of Service (QoS) traffic counters.";
      container qos {
        presence "QoS statistics are available.";
        description
          "QoS traffic counters.";
        list class {
          key "id";
          min-elements 1;
          description
            "QoS class traffic counters.";
          leaf id {
            type string;
            description
              "QoS class identifier.";
          }
          uses basic-packets-bytes;
        }
      }
    }

    grouping traffic {
      description
        "All traffic counters.";
      container l2 {
        description
          "Layer 2 traffic counters.";
        uses l2-traffic;
      }
      container l3 {
```

```
        description
          "Layer 3 traffic counters.";
        uses l3-traffic;
      }
      uses qos;
    }

    grouping errors-l2-rx {
      description
        "Layer 2 ingress frame error discard counters.";
      container rx {
        description
          "Layer 2 ingress frame receive error discard
           counters.";
        leaf frames {
          type yang:counter64;
          description
            "The number of frames discarded due to errors
             with the received frame.";
        }
        leaf crc-error {
          type yang:counter64;
          description
            "The number of frames discarded due to CRC error.";
        }
        leaf invalid-mac {
          type yang:counter64;
          description
            "The number of frames discarded due to an invalid
             MAC address.";
        }
        leaf invalid-vlan {
          type yang:counter64;
          description
            "The number of frames discarded due to an invalid
             VLAN tag.";
        }
        leaf invalid-frame {
          type yang:counter64;
          description
            "The number of invalid frames discarded due to other
             reasons, not limited to: malformed frames, frame-size
             violations.";
        }
      }
    }

    grouping errors-l3-rx {
```

```
      description
        "Layer 3 ingress packet error discard counters.";
      container rx {
        description
          "Layer 3 ingress packet receive error discard
           counters.";
        leaf packets {
          type yang:counter64;
          description
            "The number of Layer 3 packets discarded due to
             errors in the received packet.";
        }
        leaf checksum-error {
          type yang:counter64;
          description
            "The number of received packets discarded due
             to a checksum error.";
        }
        leaf mtu-exceeded {
          type yang:counter64;
          description
            "The number of received packets discarded due to
             MTU exceeded.";
        }
        leaf invalid-packet {
          type yang:counter64;
          description
            "The number of invalid packets discarded due to other
             reasons, not limited to: invalid packet length, invalid
             header fields, invalid options, invalid protocol version,
             invalid flags or control bits, malformed packets.";
        }
      }
      leaf ttl-expired {
        type yang:counter64;
        description
          "The number of received packets discarded due to
           expired TTL.";
      }
      leaf no-route {
        type yang:counter64;
        description
          "The number of packets discarded due to not matching
           a valid route.";
      }
      leaf invalid-sid {
        type yang:counter64;
        description
```

```
            "The number of packets discarded due to an invalid
             Segment Routing (SR) SID.";
        }
        leaf invalid-label {
          type yang:counter64;
          description
            "The number of packets discarded due to an invalid
             MPLS label.";
        }
      }

      grouping errors-l3-int {
        description
          "Internal error discard counters.";
        leaf packets {
          type yang:counter64;
          description
            "The number of packets discarded due to internal
             errors.";
        }
        leaf parity-error {
          type yang:counter64;
          description
            "The number of packets discarded due to parity
             errors.";
        }
      }

      grouping errors-l2-tx {
        description
          "Layer 2 transmit error discard counters.";
        container tx {
          description
            "Layer 2 transmit frame error discard counters.";
          leaf frames {
            type yang:counter64;
            description
              "The number of Layer 2 frames discarded due to
               errors when transmitting.";
          }
        }
      }

      grouping errors-l3-tx {
        description
          "Layer 3 transmit error discard counters.";
        container tx {
          description
```

```
          "Layer 3 transmit packet error discard counters.";
        leaf packets {
          type yang:counter64;
          description
            "The number of Layer 3 packets discarded due to
             errors when transmitting.";
        }
      }
    }

    grouping errors {
      description
        "Error discard counters.";
      container l2 {
        description
          "Layer 2 frame error discard counters.";
        uses errors-l2-rx;
        uses errors-l2-tx;
      }
      container l3 {
        description
          "Layer 3 packet error discard counters.";
        uses errors-l3-rx;
        uses errors-l3-tx;
      }
      container internal {
        description
          "Internal error discard counters.";
        uses errors-l3-int;
      }
    }

    grouping policy-l2 {
      description
        "Layer 2 policy frame discard counters.";
      leaf frames {
        type yang:counter64;
        description
          "The number of Layer 2 frames discarded due
           to policy.";
      }
      leaf acl {
        type yang:counter64;
        description
          "The number of frames discarded due to Layer 2 ACLs.";
      }
    }
```

```
    grouping policy-l3 {
      description
        "Layer 3 policy packet discard counters.";
      leaf packets {
        type yang:counter64;
        description
          "The number of Layer 3 packets discarded due to policy.";
      }
      leaf acl {
        type yang:counter64;
        description
          "The number of packets discarded due to Layer 3 ACLs.";
      }
      container policer {
        description
          "The number of packets discarded due to policer
           violations.";
        uses basic-packets-bytes;
      }
      leaf null-route {
        type yang:counter64;
        description
          "The number of packets discarded due to matching
           a null route.";
      }
      leaf rpf {
        type yang:counter64;
        description
          "The number of packets discarded due to failing
           Reverse Path Forwarding (RPF) check.";
      }
      leaf ddos {
        type yang:counter64;
        description
          "The number of packets discarded due to DDoS
           protection policies.";
      }
    }

    grouping discards {
      description
        "Discard counters.";
      container l2 {
        description
          "Layer 2 frame discard counters.";
        uses l2-traffic;
      }
      container l3 {
```

```
        description
          "Layer 3 packet discard counters.";
        uses l3-traffic;
      }
      container errors {
        description
          "Error discard counters.";
        uses errors;
      }
      container policy {
        description
          "Policy-related discard counters.";
        uses policy;
      }
      container no-buffer {
        description
          "Discard counters due to buffer unavailability.";
        uses qos;
      }
    }

    grouping policy {
      description
        "Policy-related discard counters.";
      container l2 {
        description
          "Layer 2 policy frame discard counters.";
        uses policy-l2;
      }
      container l3 {
        description
          "Layer 3 policy packet discard counters.";
        uses policy-l3;
      }
    }

    grouping device {
      description
        "Device-level traffic and discard counters.";
      container traffic {
        description
          "Traffic counters.";
        uses traffic;
      }
      container discards {
        description
          "Discard counters.";
        uses discards;
```

```
      }
    }

    grouping interface {
      description
        "Interface-level traffic and discard counters.";
      list traffic {
        key "direction";
        description
          "Traffic counters.";
        leaf direction {
          type identityref {
            base direction;
          }
          description
            "Specifies a direction.";
        }
        uses traffic;
      }
      list discards {
        key "direction";
        description
          "Discard counters.";
        leaf direction {
          type identityref {
            base direction;
          }
          description
            "Specifies a direction.";
        }
        uses discards;
      }
    }

    grouping control-plane {
      description
        "Control plane packet counters.";
      list traffic {
        key "direction";
        description
          "Total control plane packets.";
        leaf direction {
          type identityref {
            base direction;
          }
          description
            "Specifies a direction.";
        }
```

```
        uses basic-packets-bytes;
      }
      list discards {
        key "direction";
        description
          "Control plane packet discard counters.";
        leaf direction {
          type identityref {
            base direction;
          }
          description
            "Specifies a direction.";
        }
        uses basic-packets-bytes;
        container policy {
          description
            "Number of control plane packets discarded due to policy.";
          uses basic-packets;
        }
      }
    }

    /*
     * Main structure definition
     */

    sx:structure packet-discard-reporting {
      description
        "Specifies the abstract structure of packet discard
         reporting data.";
      container control-plane {
        if-feature "control-plane-stats";
        description
          "Control plane packet counters.";
        uses control-plane;
      }
      list interface {
        if-feature "interface-stats";
        key "name";
        description
          "Indicates a list of interfaces for which packet
           discard reporting data is provided.";
        leaf name {
          type string;
          description
            "Indicates the name of the interface.";
        }
        uses interface;
```

```
      }
      list flow {
        if-feature "flow-reporting";
        key "direction";
        leaf direction {
          type identityref {
            base direction;
          }
          description
            "Specifies a direction.";
        }
        description
          "Flow packet counters.";
        uses device;
      }
      container device {
        if-feature "device-stats";
        description
          "Device level packet counters.";
        uses device;
      }
    }
  }
  <CODE ENDS>
```

5. Data Model

This data model implements the Information Model defined in Section 4 for the interface, device and control-plane components. This is classified as a Network Element model as defined by [RFC1157].

5.1. Structure

There is a direct mapping between the Information Model components and their data model implementations, with each component in the hierarchy represented by corresponding YANG containers and leaf data nodes. The abstract tree is shown in Figure 2.

module: ietf-packet-discard-reporting

```
  +--ro control-plane! {control-plane-stats}?
  |  +--ro traffic* [direction]
  |  |  ...
  |  +--ro discards* [direction]
  |     ...
  +--ro interface* [name] {interface-stats}?
  |  +--ro name        string
  |  +--ro traffic* [direction]
```

```
|  |  +--ro direction    identityref
|  |  +--ro l2
|  |  |  ...
|  |  +--ro l3
|  |  |  ...
|  |  +--ro qos
|  |     +--ro class* [id]
|  |        ...
|  +--ro discards* [direction]
|     +--ro direction    identityref
|     +--ro l2
|     |  ...
|     +--ro l3
|     |  ...
|     +--ro errors
|     |  +--ro l2
|     |  |  ...
|     |  +--ro l3
|     |  |  ...
|     |  +--ro internal
|     |     ...
|     +--ro policy
|     |  +--ro l2
|     |  |  ...
|     |  +--ro l3
|     |     ...
|     +--ro no-buffer
|        +--ro class* [id]
|           ...
+--ro device! {device-stats}?
   +--ro traffic
   |  +--ro l2
   |  |  ...
   |  +--ro l3
   |  |  ...
   |  +--ro qos
   |     +--ro class* [id]
   |        ...
   +--ro discards
      +--ro l2
      |  ...
      +--ro l3
      |  ...
      +--ro errors
      |  +--ro l2
      |  |  ...
      |  +--ro l3
      |  |  ...
```

```
|  +--ro internal
|     ...
+--ro policy
|  +--ro l2
|  |  ...
|  +--ro l3
|     ...
+--ro no-buffer
   +--ro class* [id]
      ...
```

Figure 2: Abstract DM Tree Structure

The full tree structure is provided in Appendix E.

5.2. Implementation Requirements

The following requirements apply to the implementation of the data model and are intended to ensure consistent implementation across different vendors and platforms while allowing for platform-specific optimisations where needed. While the model defines a comprehensive set of counters and statistics, implementations MAY support a subset of the defined features based on device capabilities and operational requirements. However, implementations MUST clearly document which features are supported and how they map to the model.

Requirements 1-11 relate to packets forwarded or discarded by the device, while requirement 12 relates to packets destined for or originating from the device:

1. All instances of Layer 2 frame or Layer 3 packet receipt, transmission, and discards MUST be accounted for.

2. All instances of Layer 2 frame or Layer 3 packet receipt, transmission, and discards SHOULD be attributed to the physical or logical interface of the device where they occur. Where they cannot be attributed to the interface, they MUST be attributed to the device.

3. An individual frame MUST only be accounted for by either the Layer 2 traffic class or the Layer 2 discard classes within a single direction or context, i.e., ingress or egress or device. This is to avoid double counting.

4. An individual packet MUST only be accounted for by either the Layer 3 traffic class or the Layer 3 discard classes within a single direction or context, i.e., ingress or egress or device. This is to avoid double counting.

5. A frame accounted for at Layer 2 SHOULD NOT be accounted for at Layer 3 and vice versa. An implementation MUST indicate which layers traffic and discards are counted against. This is to avoid double counting.

6. The aggregate Layer 2 and Layer 3 traffic and discard classes SHOULD account for all underlying frames or packets received, transmitted, and discarded across all other classes.

7. The aggregate QoS traffic and no-buffer discard classes MUST account for all underlying packets received, transmitted, and discarded across all other classes.

8. In addition to the Layer 2 and Layer 3 aggregate classes, an individual discarded packet MUST only account against a single error, policy, or no-buffer discard subclass.

9. When there are multiple reasons for discarding a packet, the ordering of discard class reporting MUST be defined.

10. If Diffserv [RFC2475] is not used, no-buffer discards SHOULD be reported as class[id="0"], which represents the default class.

11. When traffic is mirrored, the discard metrics MUST account for the original traffic rather than the reflected traffic.

12. Traffic to the device control plane has its own class. However, traffic from the device control plane MUST be accounted for in the same way as other egress traffic.

5.3. Usage Examples

If all of the requirements listed in Section 5.2 are met, a "good" unicast IPv4 packet received would increment:

* interface/ingress/traffic/l3/v4/unicast/packets

* interface/ingress/traffic/l3/v4/unicast/bytes

* interface/ingress/traffic/qos/class[id="0"]/packets

* interface/ingress/traffic/qos/class[id="0"]/bytes

A received unicast IPv6 packet discarded due to Hop Limit expiry would increment:

* interface/ingress/discards/l3/v6/unicast/packets

* interface/ingress/discards/l3/v6/unicast/bytes

* interface/ingress/discards/l3/rx/ttl-expired/packets

An IPv4 packet discarded on egress due to no buffers would increment:

* interface/egress/discards/l3/v4/unicast/packets

* interface/egress/discards/l3/v4/unicast/bytes

* interface/egress/discards/no-buffer/class[id="0"]/packets

* interface/egress/discards/no-buffer/class[id="0"]/bytes

A multicast IPv6 packet dropped due to RPF check failure would increment:

* interface/ingress/discards/l3/v6/multicast/packets

* interface/ingress/discards/l3/v6/multicast/bytes

* interface/ingress/discards/policy/l3/rpf/packets

5.4. "ietf-packet-discard-reporting" YANG Module

The "ietf-packet-discard-reporting" module imports "ietf-packet-discard-reporting-sx" module.

```
<CODE BEGINS>
module ietf-packet-discard-reporting {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-packet-discard-reporting";
  prefix plr;

  import ietf-packet-discard-reporting-sx {
    prefix plr-sx;
    reference
      "RFC XXXX: Information and Data Models for Packet Discard
                 Reporting";
  }

  import ietf-netconf-acm {
    prefix nacm;
    reference
      "RFC 8341: Network Configuration Access Control Model";
  }
```

```
     organization
       "IETF OPSAWG (Operations and Management Area Working Group)";
     contact
       "WG Web:   https://datatracker.ietf.org/wg/opsawg/
        WG List:  mailto:opsawg@ietf.org

        Author:   John Evans
                  <mailto:jevanamz@amazon.co.uk>

        Author:   Oleksandr Pylypenko
                  <mailto:opyl@amazon.com>

        Author:   Jeffrey Haas
                  <mailto:jhaas@juniper.net>

        Author:   Aviran Kadosh
                  <mailto:akadosh@cisco.com>

        Author:   Mohamed Boucadair
                  <mailto:mohamed.boucadair@orange.com>";
     description
       "This module defines a data model for packet discard reporting.

        Copyright (c) 2025 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Revised BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (https://trustee.ietf.org/license-info).

        All revisions of IETF and IANA published modules can be found
        at the YANG Parameters registry
        (https://www.iana.org/assignments/yang-parameters).

        This version of this YANG module is part of RFC XXXX; see the
        RFC itself for full legal notices.";

     revision 2025-03-03 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: Information and Data Models for Packet Discard
                    Reporting";
     }
```

```
     /*
      * Features
      */

     feature control-plane-stats {
       description
         "Indicates support of control plane statistics on this
          device.";
     }

     feature interface-stats {
       description
         "Indicates support of interface statistics on this
          device.";
     }

     feature device-stats {
       description
         "Indicates support of global device statistics on this
          device.";
     }

     /*
      * Main structure definition
      */

     container control-plane {
       if-feature "control-plane-stats";
       nacm:default-deny-all;
       presence "Control plane statistics are available.";
       config false;
       description
         "Control plane packet counters.";
       uses plr-sx:control-plane;
     }
     list interface {
       if-feature "interface-stats";
       key "name";
       nacm:default-deny-all;
       config false;
       description
         "Indicates a list of interfaces for which packet discard
          reporting data is provided.";
       leaf name {
         type string;
         description
           "Indicates the name of the interface.";
       }
```

```
      uses plr-sx:interface;
    }
    container device {
      if-feature "device-stats";
      presence "Device-level statistics are available.";
      nacm:default-deny-all;
      config false;
      description
        "Device level packet counters.";
      uses plr-sx:device;
    }
  }
  <CODE ENDS>
```

6. Security Considerations

6.1. Information Model

The information model defined in Section 4.3 specifies a YANG module using [RFC8791] data extensions. It defines a set of identities, types, and groupings. These nodes are intended to be reused by other YANG modules. The module by itself does not expose any data nodes that are writable, data nodes that contain read-only state, or RPCs. As such, there are no additional security issues related to the YANG module that need to be considered.

6.2. Data Model

This section is modeled after the template described in Section 3.7 of [I-D.ietf-netmod-rfc8407bis].

The YANG module specified in Section 5.4 defines a data model that is designed to be accessed via YANG-based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. These YANG-based management protocols (1) have to use a secure transport layer (e.g., SSH [RFC4252], TLS [RFC8446], and QUIC [RFC9000]) and (2) have to use mutual authentication.

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are no particularly sensitive writable data nodes.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or

notification) to these data nodes. Specifically, the following subtrees and data nodes have particular sensitivities/ vulnerabilities:

control-plane, interfaces, and devices: Access to these data nodes would reveal information about the attacks to which an element is subject, misconfigurations, etc.

Also, an attacker who can inject packets can infer the efficiency of its attack by monitoring (the increase of) some discard counters (e.g., policy) and adjust its attack strategy accordingly.

7. IANA Considerations

IANA is requested to register the following URI in the "ns" subregistry within the "IETF XML Registry" [RFC3688]:

URI: urn:ietf:params:xml:ns:ietf-packet-discard-reporting-sx
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:ietf-packet-discard-reporting
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

IANA is requested to register the following YANG module in the "YANG Module Names" subregistry [RFC6020] within the "YANG Parameters" registry:

Name: ietf-packet-discard-reporting-sx
Namespace: urn:ietf:params:xml:ns:ietf-packet-discard-reporting-sx
Prefix: plr-sx
Maintained by IANA? N
Reference: RFC XXXX

Name: ietf-packet-discard-reporting
Namespace: urn:ietf:params:xml:ns:ietf-packet-discard-reporting
Prefix: plr
Maintained by IANA? N
Reference: RFC XXXX

8. Contributors

Nadav Chachmon
Cisco Systems, Inc.
170 West Tasman Dr.
San Jose, CA 95134
United States of America
Email: nchachmo@cisco.com

9. Acknowledgments

The content of this document has benefitted from feedback from JR Rivers, Ronan Waide, Chris DeBruin, and Marcos Sanz.

10. References

10.1. 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/rfc/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/rfc/rfc3688>.

[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/rfc/rfc6020>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/rfc/rfc8341>.

[RFC8791] Bierman, A., Bj旦rklund, M., and K. Watsen, "YANG Data Structure Extensions", RFC 8791, DOI 10.17487/RFC8791, June 2020, <https://www.rfc-editor.org/rfc/rfc8791>.

10.2. Informative References

[gMNI] Marrow, C., "gRPC Network Management Interface, IETF 98, March 2017, <https://datatracker.ietf.org/meeting/98/materials/slides-98-rtgwg-gnmi-intro-draft-openconfig-rtgwg-gnmi-spec-00>", n.d..

[I-D.ietf-netmod-rfc8407bis] Bierman, A., Boucadair, M., and Q. Wu, "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", Work in Progress, Internet-Draft, draft-ietf-netmod-rfc8407bis-28, 5 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-rfc8407bis-28>.

[RED93] Jacobson, V., "Random Early Detection gateways for Congestion Avoidance", n.d..

[RFC1157] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol (SNMP)", RFC 1157, DOI 10.17487/RFC1157, May 1990, <https://www.rfc-editor.org/rfc/rfc1157>.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, DOI 10.17487/RFC2475, December 1998, <https://www.rfc-editor.org/rfc/rfc2475>.

[RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, DOI 10.17487/RFC2863, June 2000, <https://www.rfc-editor.org/rfc/rfc2863>.

[RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, DOI 10.17487/RFC3411, December 2002, <https://www.rfc-editor.org/rfc/rfc3411>.

[RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, DOI 10.17487/RFC3444, January 2003, <https://www.rfc-editor.org/rfc/rfc3444>.

[RFC4252] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, DOI 10.17487/RFC4252, January 2006, <https://www.rfc-editor.org/rfc/rfc4252>.

[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/rfc/rfc6241>.

[RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <https://www.rfc-editor.org/rfc/rfc7011>.

[RFC7270] Yourtchenko, A., Aitken, P., and B. Claise, "Cisco-Specific Information Elements Reused in IP Flow Information Export (IPFIX)", RFC 7270, DOI 10.17487/RFC7270, June 2014, <https://www.rfc-editor.org/rfc/rfc7270>.

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

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

[RFC8289] Nichols, K., Jacobson, V., McGregor, A., Ed., and J. Iyengar, Ed., "Controlled Delay Active Queue Management", RFC 8289, DOI 10.17487/RFC8289, January 2018, <https://www.rfc-editor.org/rfc/rfc8289>.

[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/rfc/rfc8340>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/rfc/rfc8343>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

Appendix A. Where Do Packets Get Dropped?

Understanding where packets are discarded in a network device is essential for interpreting discard signals and determining appropriate mitigation actions. Figure 3 depicts an example of where and why packets may be discarded in a typical single-ASIC, shared-buffered type device. While actual device architectures vary between vendors and platforms, with some using multiple ASICs, distributed forwarding, or different buffering architectures, this example illustrates the common processing stages where packets may be dropped. The logical model for classifying and reporting discards remains consistent regardless of the underlying hardware architecture.

Packets ingress on the left and egress on the right:

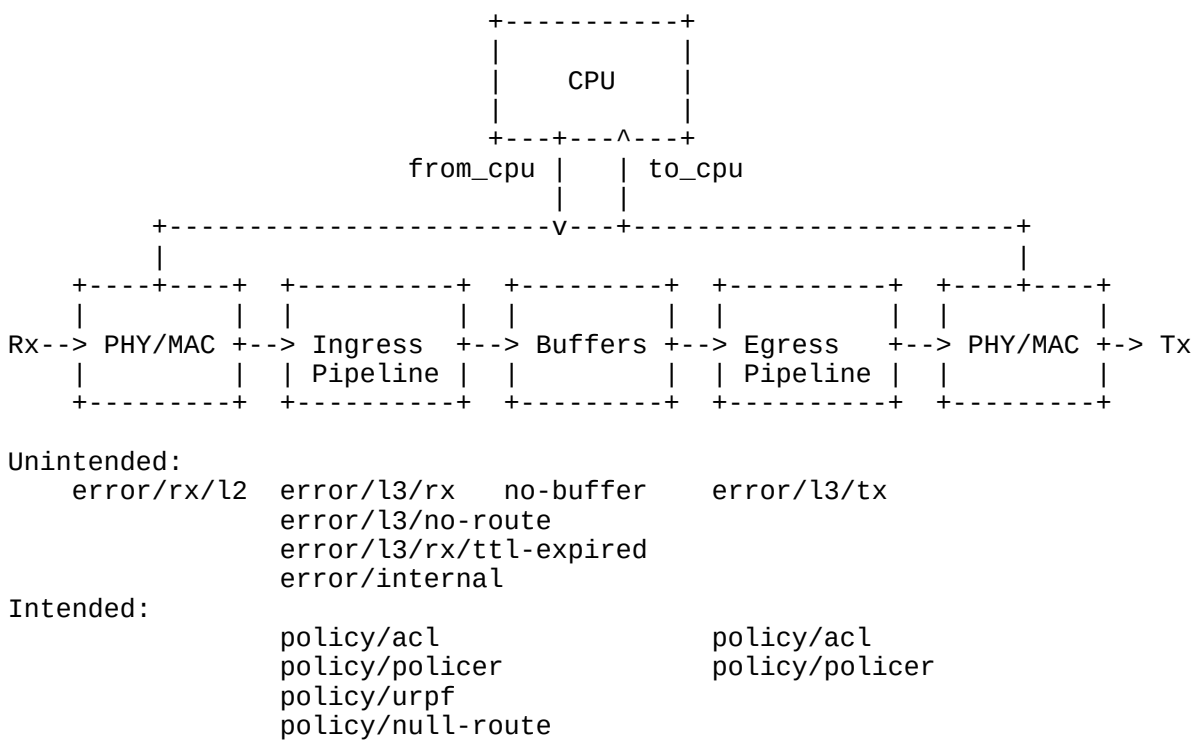

Figure 3: Example of where packets get dropped

See Appendix B for examples of how these discard signals map to root causes and mitigation actions.

Appendix B. Example Signal-to-mitigation Action Mapping

The effectiveness of automated mitigation depends on correctly mapping discard signals to root causes and appropriate actions. Table 1 gives example discard signal-to-mitigation action mappings based on the features described in section 3.

DISCARD-CLASS	Discard cause	DISCARD-RATE	DISCARD-DURATION	Unintended?	Possible actions
ingress/discards/errors/l2/rx	Upstream device or link error	>Baseline	O(1min)	Y	Take upstream link or device out-of-service
ingress/discards/errors/l3/rx/ttl-expired	Tracert	<=Baseline		N	no action
ingress/discards/errors/l3/rx/ttl-expired	Convergence	>Baseline	O(1s)	Y	No action
ingress/discards/errors/l3/rx/ttl-expired	Routing loop	>Baseline	O(1min)	Y	Roll-back change
.*/policy/.*	Policy			N	No action
ingress/discards/errors/l3/no-route	Convergence	>Baseline	O(1s)	Y	No action
ingress/discards/errors/l3/no-route	Config error	>Baseline	O(1min)	Y	Roll-back change
ingress/discards/errors/l3/no-route	Invalid destination	>Baseline	O(10min)	N	Escalate to operator
ingress/discards/errors/internal	Device errors	>Baseline	O(1min)	Y	Take device out-of-service
egress/discards/no-buffer	Congestion	<=Baseline		N	No

					action
egress/discards/no-buffer	Congestion	>Baseline	O(1min)	Y	Bring capacity back into service or move traffic

Table 1: Example Signal-Cause-Mitigation Mapping

The 'Baseline' in the 'DISCARD-RATE' column is both DISCARD-CLASS and network dependent.

Appendix C. Implementation Experience

This appendix captures practical insights gained from implementing this information model across multiple vendors' platforms, as guidance for future implementers.

1. The number and granularity of discard classes defined in the information model represent a compromise. It aims to provide sufficient detail to enable appropriate automated actions while avoiding excessive detail, which may hinder quick problem identification. Additionally, it helps to limit the quantity of data produced per interface, constraining the data volume and device CPU impacts. While further granularity is possible, the defined schema has generally proven to be sufficient for the task of mitigating unintended packet loss.

2. There are many possible ways to define the discard classification tree. For example, we could have used a multi-rooted tree, rooted in each protocol. Instead, we opted to define a tree where protocol discards and causal discard classes are accounted for orthogonally. This decision reduces the number of combinations of classes and has proven sufficient for determining mitigation actions.

3. NoBuffer discards can be realized differently with different memory architectures. Whether a NoBuffer discard is attributed to ingress or egress can differ accordingly. For successful auto-mitigation, discards due to egress interface congestion should be reported on egress, while discards due to device-level congestion (e.g. due to exceeding the device forwarding rate) should be reported on ingress.

4. Platforms often account for the number of packets discarded where the TTL has expired (or Hop Limit exceeded), and the device CPU has returned an ICMP Time Exceeded message. There is typically a policer applied to limit the number of packets sent to the device CPU, however, which implicitly limits the rate of TTL discards that are processed. One method to account for all packet discards due to TTL expired, even those that are dropped by a policer when being forwarded to the CPU, is to use accounting of all ingress packets received with TTL=1 as a proxy measure.

5. Where no route discards are implemented with a default null route, separate discard accounting is required for any explicit null routes configured, in order to differentiate between interface/ingress/discards/policy/null-route/packets and interface/ingress/discards/errors/no-route/packets.

6. It is useful to account separately for transit packets discarded by ACLs or policers, and packets discarded by ACLs or policers which limit the number of packets to the device control plane.

7. It is not possible to identify a configuration error - e.g., when intended discards are unintended - with device discard metrics alone. For example, additional context is needed to determine if ACL discards are intended or due to a misconfigured ACL, i.e., with configuration validation before deployment or by detecting a significant change in ACL discards after a configuration change compared to before.

8. Aggregate counters need to be able to deal with the possibility of discontinuities in the underlying counters.

9. In cases where the reporting device is the source or destination of a tunnel, the ingress protocol for a packet may differ from the egress protocol (e.g., if IPv4 is tunneled over IPv6). Some implementations may attribute egress discards to the ingress protocol.

10. While the classification tree is seven layers deep, a minimal implementation may only implement the top six layers.

Appendix D. Full Information Model Tree

The following YANG tree diagram shows the complete IM structure:

```
   module: ietf-packet-discard-reporting-sx

     structure packet-discard-reporting:
       +-- control-plane {control-plane-stats}?
       |  +-- traffic* [direction]
       |  |  +-- direction    identityref
       |  |  +-- packets?     yang:counter64
       |  |  +-- bytes?       yang:counter64
       |  +-- discards* [direction]
       |     +-- direction    identityref
       |     +-- packets?     yang:counter64
       |     +-- bytes?       yang:counter64
       |     +-- policy
       |        +-- packets?   yang:counter64
       +-- interface* [name] {interface-stats}?
       |  +-- name        string
       |  +-- traffic* [direction]
       |  |  +-- direction    identityref
       |  |  +-- l2
       |  |  |  +-- frames?   yang:counter64
       |  |  |  +-- bytes?    yang:counter64
       |  |  +-- l3
       |  |  |  +-- address-family-stat* [address-family]
       |  |  |     +-- address-family    identityref
       |  |  |     +-- packets?          yang:counter64
       |  |  |     +-- bytes?            yang:counter64
       |  |  |     +-- unicast
       |  |  |     |  +-- packets?   yang:counter64
       |  |  |     |  +-- bytes?     yang:counter64
       |  |  |     +-- multicast
       |  |  |        +-- packets?   yang:counter64
       |  |  |        +-- bytes?     yang:counter64
       |  |  +-- qos!
       |  |     +-- class* [id]
       |  |        +-- id         string
       |  |        +-- packets?   yang:counter64
       |  |        +-- bytes?     yang:counter64
       |  +-- discards* [direction]
       |     +-- direction    identityref
       |     +-- l2
       |     |  +-- frames?   yang:counter64
       |     |  +-- bytes?    yang:counter64
       |     +-- l3
       |     |  +-- address-family-stat* [address-family]
       |     |     +-- address-family    identityref
       |     |     +-- packets?          yang:counter64
       |     |     +-- bytes?            yang:counter64
       |     |     +-- unicast
```

```
|     |     |  +-- packets?   yang:counter64
|     |     |  +-- bytes?     yang:counter64
|     |     +-- multicast
|     |        +-- packets?   yang:counter64
|     |        +-- bytes?     yang:counter64
|     +-- errors
|     |  +-- l2
|     |  |  +-- rx
|     |  |  |  +-- frames?          yang:counter64
|     |  |  |  +-- crc-error?       yang:counter64
|     |  |  |  +-- invalid-mac?     yang:counter64
|     |  |  |  +-- invalid-vlan?    yang:counter64
|     |  |  |  +-- invalid-frame?   yang:counter64
|     |  |  +-- tx
|     |  |     +-- frames?   yang:counter64
|     |  +-- l3
|     |  |  +-- rx
|     |  |  |  +-- packets?          yang:counter64
|     |  |  |  +-- checksum-error?   yang:counter64
|     |  |  |  +-- mtu-exceeded?     yang:counter64
|     |  |  |  +-- invalid-packet?   yang:counter64
|     |  |  +-- ttl-expired?     yang:counter64
|     |  |  +-- no-route?        yang:counter64
|     |  |  +-- invalid-sid?     yang:counter64
|     |  |  +-- invalid-label?   yang:counter64
|     |  |  +-- tx
|     |  |     +-- packets?   yang:counter64
|     |  +-- internal
|     |     +-- packets?        yang:counter64
|     |     +-- parity-error?   yang:counter64
|     +-- policy
|     |  +-- l2
|     |  |  +-- frames?   yang:counter64
|     |  |  +-- acl?      yang:counter64
|     |  +-- l3
|     |     +-- packets?      yang:counter64
|     |     +-- acl?          yang:counter64
|     |     +-- policer
|     |     |  +-- packets?   yang:counter64
|     |     |  +-- bytes?     yang:counter64
|     |     +-- null-route?   yang:counter64
|     |     +-- rpf?          yang:counter64
|     |     +-- ddos?         yang:counter64
|     +-- no-buffer
|        +-- qos!
|           +-- class* [id]
|              +-- id         string
|              +-- packets?   yang:counter64
```

```
|              +-- bytes?     yang:counter64
+-- flow* [direction] {flow-reporting}?
|  +-- direction    identityref
|  +-- traffic
|  |  +-- l2
|  |  |  +-- frames?   yang:counter64
|  |  |  +-- bytes?    yang:counter64
|  |  +-- l3
|  |  |  +-- address-family-stat* [address-family]
|  |  |     +-- address-family    identityref
|  |  |     +-- packets?          yang:counter64
|  |  |     +-- bytes?            yang:counter64
|  |  |     +-- unicast
|  |  |     |  +-- packets?   yang:counter64
|  |  |     |  +-- bytes?     yang:counter64
|  |  |     +-- multicast
|  |  |        +-- packets?   yang:counter64
|  |  |        +-- bytes?     yang:counter64
|  |  +-- qos!
|  |     +-- class* [id]
|  |        +-- id         string
|  |        +-- packets?   yang:counter64
|  |        +-- bytes?     yang:counter64
|  +-- discards
|     +-- l2
|     |  +-- frames?   yang:counter64
|     |  +-- bytes?    yang:counter64
|     +-- l3
|     |  +-- address-family-stat* [address-family]
|     |     +-- address-family    identityref
|     |     +-- packets?          yang:counter64
|     |     +-- bytes?            yang:counter64
|     |     +-- unicast
|     |     |  +-- packets?   yang:counter64
|     |     |  +-- bytes?     yang:counter64
|     |     +-- multicast
|     |        +-- packets?   yang:counter64
|     |        +-- bytes?     yang:counter64
|     +-- errors
|     |  +-- l2
|     |  |  +-- rx
|     |  |  |  +-- frames?          yang:counter64
|     |  |  |  +-- crc-error?       yang:counter64
|     |  |  |  +-- invalid-mac?     yang:counter64
|     |  |  |  +-- invalid-vlan?    yang:counter64
|     |  |  |  +-- invalid-frame?   yang:counter64
|     |  |  +-- tx
|     |  |     +-- frames?   yang:counter64
```

```
|     |  +-- l3
|     |  |  +-- rx
|     |  |  |  +-- packets?          yang:counter64
|     |  |  |  +-- checksum-error?   yang:counter64
|     |  |  |  +-- mtu-exceeded?     yang:counter64
|     |  |  |  +-- invalid-packet?   yang:counter64
|     |  |  +-- ttl-expired?     yang:counter64
|     |  |  +-- no-route?        yang:counter64
|     |  |  +-- invalid-sid?     yang:counter64
|     |  |  +-- invalid-label?   yang:counter64
|     |  |  +-- tx
|     |  |     +-- packets?   yang:counter64
|     |  +-- internal
|     |     +-- packets?        yang:counter64
|     |     +-- parity-error?   yang:counter64
|     +-- policy
|     |  +-- l2
|     |  |  +-- frames?   yang:counter64
|     |  |  +-- acl?      yang:counter64
|     |  +-- l3
|     |     +-- packets?      yang:counter64
|     |     +-- acl?          yang:counter64
|     |     +-- policer
|     |     |  +-- packets?   yang:counter64
|     |     |  +-- bytes?     yang:counter64
|     |     +-- null-route?   yang:counter64
|     |     +-- rpf?          yang:counter64
|     |     +-- ddos?         yang:counter64
|     +-- no-buffer
|        +-- qos!
|           +-- class* [id]
|              +-- id         string
|              +-- packets?   yang:counter64
|              +-- bytes?     yang:counter64
+-- device {device-stats}?
   +-- traffic
   |  +-- l2
   |  |  +-- frames?   yang:counter64
   |  |  +-- bytes?    yang:counter64
   |  +-- l3
   |  |  +-- address-family-stat* [address-family]
   |  |     +-- address-family    identityref
   |  |     +-- packets?          yang:counter64
   |  |     +-- bytes?            yang:counter64
   |  |     +-- unicast
   |  |     |  +-- packets?   yang:counter64
   |  |     |  +-- bytes?     yang:counter64
   |  |     +-- multicast
```

```
|  |        +-- packets?   yang:counter64
|  |        +-- bytes?     yang:counter64
|  +-- qos!
|     +-- class* [id]
|        +-- id         string
|        +-- packets?   yang:counter64
|        +-- bytes?     yang:counter64
+-- discards
   +-- l2
   |  +-- frames?   yang:counter64
   |  +-- bytes?    yang:counter64
   +-- l3
   |  +-- address-family-stat* [address-family]
   |     +-- address-family    identityref
   |     +-- packets?          yang:counter64
   |     +-- bytes?            yang:counter64
   |     +-- unicast
   |     |  +-- packets?   yang:counter64
   |     |  +-- bytes?     yang:counter64
   |     +-- multicast
   |        +-- packets?   yang:counter64
   |        +-- bytes?     yang:counter64
   +-- errors
   |  +-- l2
   |  |  +-- rx
   |  |  |  +-- frames?          yang:counter64
   |  |  |  +-- crc-error?       yang:counter64
   |  |  |  +-- invalid-mac?     yang:counter64
   |  |  |  +-- invalid-vlan?    yang:counter64
   |  |  |  +-- invalid-frame?   yang:counter64
   |  |  +-- tx
   |  |     +-- frames?   yang:counter64
   |  +-- l3
   |  |  +-- rx
   |  |  |  +-- packets?          yang:counter64
   |  |  |  +-- checksum-error?   yang:counter64
   |  |  |  +-- mtu-exceeded?     yang:counter64
   |  |  |  +-- invalid-packet?   yang:counter64
   |  |  +-- ttl-expired?     yang:counter64
   |  |  +-- no-route?        yang:counter64
   |  |  +-- invalid-sid?     yang:counter64
   |  |  +-- invalid-label?   yang:counter64
   |  |  +-- tx
   |  |     +-- packets?   yang:counter64
   |  +-- internal
   |     +-- packets?        yang:counter64
   |     +-- parity-error?   yang:counter64
   +-- policy
```

```
            |  +-- l2
            |  |  +-- frames?   yang:counter64
            |  |  +-- acl?      yang:counter64
            |  +-- l3
            |     +-- packets?      yang:counter64
            |     +-- acl?          yang:counter64
            |     +-- policer
            |     |  +-- packets?   yang:counter64
            |     |  +-- bytes?     yang:counter64
            |     +-- null-route?   yang:counter64
            |     +-- rpf?          yang:counter64
            |     +-- ddos?         yang:counter64
            +-- no-buffer
               +-- qos!
                  +-- class* [id]
                     +-- id         string
                     +-- packets?   yang:counter64
                     +-- bytes?     yang:counter64
```

Appendix E. Full Data Model Tree

The following YANG tree diagram shows the complete data module structure:

```
module: ietf-packet-discard-reporting
  +--ro control-plane! {control-plane-stats}?
  |  +--ro traffic* [direction]
  |  |  +--ro direction    identityref
  |  |  +--ro packets?     yang:counter64
  |  |  +--ro bytes?       yang:counter64
  |  +--ro discards* [direction]
  |     +--ro direction    identityref
  |     +--ro packets?     yang:counter64
  |     +--ro bytes?       yang:counter64
  |     +--ro policy
  |        +--ro packets?   yang:counter64
  +--ro interface* [name] {interface-stats}?
  |  +--ro name        string
  |  +--ro traffic* [direction]
  |  |  +--ro direction    identityref
  |  |  +--ro l2
  |  |  |  +--ro frames?   yang:counter64
  |  |  |  +--ro bytes?    yang:counter64
  |  |  +--ro l3
  |  |  |  +--ro address-family-stat* [address-family]
  |  |  |     +--ro address-family    identityref
  |  |  |     +--ro packets?          yang:counter64
  |  |  |     +--ro bytes?            yang:counter64
```

```
|  |  |     +--ro unicast
|  |  |     |  +--ro packets?   yang:counter64
|  |  |     |  +--ro bytes?     yang:counter64
|  |  |     +--ro multicast
|  |  |        +--ro packets?   yang:counter64
|  |  |        +--ro bytes?     yang:counter64
|  |  +--ro qos!
|  |     +--ro class* [id]
|  |        +--ro id         string
|  |        +--ro packets?   yang:counter64
|  |        +--ro bytes?     yang:counter64
|  +--ro discards* [direction]
|     +--ro direction    identityref
|     +--ro l2
|     |  +--ro frames?   yang:counter64
|     |  +--ro bytes?    yang:counter64
|     +--ro l3
|     |  +--ro address-family-stat* [address-family]
|     |     +--ro address-family    identityref
|     |     +--ro packets?          yang:counter64
|     |     +--ro bytes?            yang:counter64
|     |     +--ro unicast
|     |     |  +--ro packets?   yang:counter64
|     |     |  +--ro bytes?     yang:counter64
|     |     +--ro multicast
|     |        +--ro packets?   yang:counter64
|     |        +--ro bytes?     yang:counter64
|     +--ro errors
|     |  +--ro l2
|     |  |  +--ro rx
|     |  |  |  +--ro frames?          yang:counter64
|     |  |  |  +--ro crc-error?       yang:counter64
|     |  |  |  +--ro invalid-mac?     yang:counter64
|     |  |  |  +--ro invalid-vlan?    yang:counter64
|     |  |  |  +--ro invalid-frame?   yang:counter64
|     |  |  +--ro tx
|     |  |     +--ro frames?   yang:counter64
|     |  +--ro l3
|     |  |  +--ro rx
|     |  |  |  +--ro packets?          yang:counter64
|     |  |  |  +--ro checksum-error?   yang:counter64
|     |  |  |  +--ro mtu-exceeded?     yang:counter64
|     |  |  |  +--ro invalid-packet?   yang:counter64
|     |  |  +--ro ttl-expired?     yang:counter64
|     |  |  +--ro no-route?        yang:counter64
|     |  |  +--ro invalid-sid?     yang:counter64
|     |  |  +--ro invalid-label?   yang:counter64
|     |  |  +--ro tx
```

```
|     |  |     +--ro packets?   yang:counter64
|     |  +--ro internal
|     |     +--ro packets?        yang:counter64
|     |     +--ro parity-error?   yang:counter64
|     +--ro policy
|     |  +--ro l2
|     |  |  +--ro frames?   yang:counter64
|     |  |  +--ro acl?      yang:counter64
|     |  +--ro l3
|     |     +--ro packets?      yang:counter64
|     |     +--ro acl?          yang:counter64
|     |     +--ro policer
|     |     |  +--ro packets?   yang:counter64
|     |     |  +--ro bytes?     yang:counter64
|     |     +--ro null-route?   yang:counter64
|     |     +--ro rpf?          yang:counter64
|     |     +--ro ddos?         yang:counter64
|     +--ro no-buffer
|        +--ro qos!
|           +--ro class* [id]
|              +--ro id         string
|              +--ro packets?   yang:counter64
|              +--ro bytes?     yang:counter64
+--ro device! {device-stats}?
   +--ro traffic
   |  +--ro l2
   |  |  +--ro frames?   yang:counter64
   |  |  +--ro bytes?    yang:counter64
   |  +--ro l3
   |  |  +--ro address-family-stat* [address-family]
   |  |     +--ro address-family    identityref
   |  |     +--ro packets?          yang:counter64
   |  |     +--ro bytes?            yang:counter64
   |  |     +--ro unicast
   |  |     |  +--ro packets?   yang:counter64
   |  |     |  +--ro bytes?     yang:counter64
   |  |     +--ro multicast
   |  |        +--ro packets?   yang:counter64
   |  |        +--ro bytes?     yang:counter64
   |  +--ro qos!
   |     +--ro class* [id]
   |        +--ro id         string
   |        +--ro packets?   yang:counter64
   |        +--ro bytes?     yang:counter64
   +--ro discards
      +--ro l2
      |  +--ro frames?   yang:counter64
      |  +--ro bytes?    yang:counter64
```

```
          +--ro l3
          |  +--ro address-family-stat* [address-family]
          |     +--ro address-family    identityref
          |     +--ro packets?          yang:counter64
          |     +--ro bytes?            yang:counter64
          |     +--ro unicast
          |     |  +--ro packets?   yang:counter64
          |     |  +--ro bytes?     yang:counter64
          |     +--ro multicast
          |        +--ro packets?   yang:counter64
          |        +--ro bytes?     yang:counter64
          +--ro errors
          |  +--ro l2
          |  |  +--ro rx
          |  |  |  +--ro frames?          yang:counter64
          |  |  |  +--ro crc-error?       yang:counter64
          |  |  |  +--ro invalid-mac?     yang:counter64
          |  |  |  +--ro invalid-vlan?    yang:counter64
          |  |  |  +--ro invalid-frame?   yang:counter64
          |  |  +--ro tx
          |  |     +--ro frames?   yang:counter64
          |  +--ro l3
          |  |  +--ro rx
          |  |  |  +--ro packets?          yang:counter64
          |  |  |  +--ro checksum-error?   yang:counter64
          |  |  |  +--ro mtu-exceeded?     yang:counter64
          |  |  |  +--ro invalid-packet?   yang:counter64
          |  |  +--ro ttl-expired?     yang:counter64
          |  |  +--ro no-route?        yang:counter64
          |  |  +--ro invalid-sid?     yang:counter64
          |  |  +--ro invalid-label?   yang:counter64
          |  |  +--ro tx
          |  |     +--ro packets?   yang:counter64
          |  +--ro internal
          |     +--ro packets?        yang:counter64
          |     +--ro parity-error?   yang:counter64
          +--ro policy
          |  +--ro l2
          |  |  +--ro frames?   yang:counter64
          |  |  +--ro acl?      yang:counter64
          |  +--ro l3
          |     +--ro packets?      yang:counter64
          |     +--ro acl?          yang:counter64
          |     +--ro policer
          |     |  +--ro packets?   yang:counter64
          |     |  +--ro bytes?     yang:counter64
          |     +--ro null-route?   yang:counter64
          |     +--ro rpf?          yang:counter64
```

```
           |     +--ro ddos?         yang:counter64
           +--ro no-buffer
              +--ro qos!
                 +--ro class* [id]
                    +--ro id         string
                    +--ro packets?   yang:counter64
                    +--ro bytes?     yang:counter64
```

Authors' Addresses

John Evans
Amazon
1 Principal Place, Worship Street
London
EC2A 2FA
United Kingdom
Email: jevanamz@amazon.co.uk

Oleksandr Pylypenko
Amazon
410 Terry Ave N
Seattle, WA 98109
United States of America
Email: opyl@amazon.com

Jeffrey Haas
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
United States of America
Email: jhaas@juniper.net

Aviran Kadosh
Cisco Systems, Inc.
170 West Tasman Dr.
San Jose, CA 95134
United States of America
Email: akadosh@cisco.com

Mohamed Boucadair
Orange
France
Email: mohamed.boucadair@orange.com