draft-opsawg-evans-discardmodel

J. Evans, O. Pylypenko, Amazon
  J. Haas, Juniper Networks
A. Kadosh, Cisco Systems, Inc.
Problem statement 1

• The job of a network is to transport packets
• Packet loss is the primary signal of when a network is not doing its job
• But some level of packet loss is normal in TCP/IP networks
• How can we minimize anomalous packet loss through automated network operations?
Operational Context

Active monitoring

Passive monitoring

Event data

Signal

Detection

Triage

Correlation and root cause analysis

Anomaly detection

Triangulation + Impact detection

Auto-mitigation/remediation

Risk-Management

Auto-mitigation
Problem statement 2

• How can we report packet loss ...
  ... with sufficient accuracy that we can detect anomalies (even low-level)
  ... and sufficient context that we can apply appropriate auto-mitigation actions
    ... which device?
    ... what's the cause?
Working backwards from auto-mitigation

• There are only a relative small number of auto-mitigation actions
  • Take a device / link / set of devices and or links out of service
  • Put a device / link / set of devices and or links back into service
  • Roll-back a change
  • Move traffic
  • Escalate to Network Operators

• Precise signal of impact is important – taking the wrong action can be worse than taking no action
  • Taking a congested device out of service can make congestion worse
MIB-II (RFC1213, 1991)

• ifInDiscards
  • “The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space.”

• ifInErrors
  • “The number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol.”
Implementation Inconsistency

• All vendors support more discard metrics than this – but they are inconsistently implemented

• Experience across multiple implementations and hardware platforms:
  • Not reporting all discards – appears like a grey failure
  • Duplicate reporting across discard metrics
  • Same OID can account for different types of discard on different platforms
  • ifInErrors can include non-discarded “errors” and discarded errors
  • Interface metrics vs. platform metrics vs. something in between

• There are no clearly defined semantics for packet loss reporting
Experience defining a new packet discard classification scheme

- We defined discard classes working backwards from auto-remediation
- Defined discard semantics
- Mapped the underlying hardware drop counters to the discard classes
  - Across multiple hardware platforms
  - From 64 to 256 underlying hardware drop counters, depending on platform
Discard classification scheme

All discards*

Intended discards
- ACL
- Policier
- Null Route

Unintended discards
- No route
- No buffer
- TTL exceed
- Error
  - Rx
    - Local
    - Tx
  - Queue 0
  - Queue n
- Tx

* Also need packets sent
Semantics Matter

• TLDR:
  • Report all packet drops ...
  • ... once and only once ...
  • ... where they occur ...
  • ... in the right class

• Long version: 
  https://datatracker.ietf.org/doc/draft-opsawg-evans-discardmodel/
## Reason → Cause → Action mappings

<table>
<thead>
<tr>
<th>Drop reason</th>
<th>Direction</th>
<th>Drop Cause</th>
<th>Loss rate</th>
<th>Loss duration</th>
<th>Customer impacting?</th>
<th>Possible actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ErrorRxL2Discards</td>
<td>Ingress</td>
<td>Upstream device or link error</td>
<td>&gt;0(Anomaly)</td>
<td>O(1min)</td>
<td>Y</td>
<td>Take upstream link or device out-of-service</td>
</tr>
<tr>
<td>TTLDiscards</td>
<td>Ingress</td>
<td>Tracert</td>
<td>&lt;=Baseline</td>
<td></td>
<td>N</td>
<td>no action</td>
</tr>
<tr>
<td>TTLDiscards</td>
<td>Ingress</td>
<td>Convergence</td>
<td>&gt;Baseline</td>
<td>O(1s)</td>
<td>Y</td>
<td>no action</td>
</tr>
<tr>
<td>TTLDiscards</td>
<td>Ingress</td>
<td>Routing loop</td>
<td>&gt;Baseline</td>
<td>O(1min)</td>
<td>Y</td>
<td>Roll-back</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Implementation experience

• Number of discard classes is a compromise
  • Enough granularity to take the right action
  • Too much information – can slow down resolution rather than help to surface the problem quickly
  • Volume of data for per interface metrics

• Null route vs. no route discards
• To CPU ACL vs. transit ACL discards
• Responded TTL expired vs total TTL expired
• Cannot detect config error without additional context
Information model + semantics rather than data model [RFC3444]
Result of implementation experience
Possible subsequent data models for NETCONF/Yang or IPFIX
Related NANOG presentation:
  • https://youtu.be/FixkCbixgMM?feature=shared