排除故障在IOS XP的输入丢弃

目录

简介

本文描述如何排除故障在接口的输入丢弃在XR路由器。

先决条件

要求

本文档没有任何特定的要求。

使用的组件

此条款包括ASR 9000系列路由器、CRS系列路由器和GSR12000系列路由器。

本文档中的信息都是基于特定实验室环境中的设备编写的。本文档中使用的所有设备最初均采用原始（默认）配置。如果您的网络处于活动状态，请确保您了解所有命令的潜在影响。

背景信息

在IOS XP的输入丢弃比那有完全不同的含义在IOS的输入丢弃。当移植IOS到IOS XP并且开始发现其在show interface时的输入丢弃计数器它能迷惑您。

在IOS中，一个输入丢弃归结于该的接口输入队列全双工的获得。这意味着过多的数据包被踢了对交换的进程的CPU，并且没有能处理他们足够快速。Input queue被加强，直到获得全双工，并且有一些丢包。

在IOS XP，没有一个输入丢弃的严格定义。因此它基本上是至决定的组件的开发员他们是否要增加输入丢弃计数器，当他们决定丢弃数据包时。此处关键事是那在某种程度上代码，路由器决定丢弃数据包，以便意味着很可能路由器不应该转发数据包和路由器自觉地决定丢弃它。 所以，这与拥塞

没有涉及类似在IOS。然而，它相当是由路由器接收，并且不应该转发的数据包，因此路由器决定丢弃它，并且是很可能不是原因警报。虽然，您不能分辨是否它是某事忧虑，直到您完全了解增加输入丢弃计数器，并且不是简单的那的这数据包，不幸地。

示例：

- XR路由器连接到发送一些网桥协议数据单元的交换机(BPDU) andUDLD数据包。XR路由器没有生成树亦不在其第3层配置的UDLD建立接口它如此丢弃这些帧，并且增加在show interface的输入丢弃计数器。在这种情况下，没什么忧虑，因为它是要执行丢弃的正确的事这些帧，因为功能没有配置。

- ASR 9000有一思科快速转发(CEF)条目错误被编程的由于bug，以便不指向有效邻接。在这种情况下， ASR 9000线卡(LC)的网络处理器意识到路由器未命中loadinfo并且增加上传到接口输入丢弃计数器的网络处理器(NP)丢弃计数器。

当输入丢弃报告时，问题将推测这些是否是合法丢包类似在示例1或一问题的结果类似在示例2。

问题：增量在输入丢弃

本文获得被增加和如何的原因能检查的输入丢弃它是否是该原因：

控制器丢包

残帧，帧校验序列，中止，第一输入第一输出(FIFO)溢出，在SDH/SONET (POS)丢包的巨人数据包。

```
RP/0/RP0/CPU0:equinox#show controllers poS 0/2/0/0 framer statistics
POS Driver Internal Cooked Stats Values for port 0
================================================
Rx Statistics                   Tx Statistics
-------------                   -------------
Total Bytes:    71346296        Total Bytes:    67718333
Good Bytes:     71346296        Good Bytes:     67718333
Good Packets:   105385          Good Packets:   67281
Aborts:         0               Aborts:         0
FCS Errors:     0               Min-len errors: 0
Runts:          0               Max-len errors: 0
FIFO Overflows: 0               FIFO Underruns: 0
Giants:         0
Drops:          0

RP/0/RP0/CPU0:equinox#
```

以太网(gige， tengige…)接口，检查某事类似：

show controllers gigabitethernet 0/0/0/18 stats

检查是否有在增加以速率和在show interface的输入丢弃计数器一样的控制器stats的一个计数器。其中一些错误计数器必须也在show interface。

未知的目的地中等访问控制地址(DMAC)或dot1q vlan

数据包和目标MAC地址一起不是那个接口或有子接口没匹配的虚拟局域网的。这些能发生，当如此有在未知单播MAC地址L2域的某次泛滥连接的XR路由器对那L2域有不是其控制器之一的目标MAC地址的时接收帧。 也是可能的，当IOS路由器发送在其gige接口时的以太网Keepalive，因此这些Keepalive增加在XR路由器的输入丢弃，因为他们没有XR路由器的目标MAC地址。或者，当接口连接到有在XR路由器/子接口配置的和更多dot1q VLAN的另一个设备，以便XR路由器接收有一未知dot1q标记的帧。

在CRS修复的物理层接口模块(PLIM)，您可能查找这样丢包在：

```
RP/0/RP0/CPU0:pixies-uk#sh contr plim asic statistics interface tenGigE 0/1/0/3 location
0/1/CPU0
Wed Aug 22 16:07:47.854 CEST
                                Node: 0/1/CPU0


TenGigE0/1/0/3 Drop
------------------------------------------------------------------------------
RxFiFO Drop       : 0                   PAR Tail Drop     : 0
PAR Err Drop      : 0                   Invalid MAC Drop  : 86
TxFIFO Drop       : 0                   Invalid VLAN Drop : 11
```

或者在tengige或gige控制器stats ：

```
RP/0/RP0/CPU0:pixies-uk#sh contr ten 0/1/0/3 stats
Wed Aug 22 16:22:42.059 CEST
Statistics for interface TenGigE0/1/0/3 (cached values):

Ingress:


    Input drop overrun          = 0
    Input drop abort            = 0
    Input drop invalid VLAN     = 11
    Input drop invalid DMAC     = 0
    Input drop invalid encap    = 0
    Input drop other            = 86
```

> **Note**:**其他**被增加而不是**输入丢弃无效DMAC**至少在CRS的8端口tengige修复的PLIM的Bug CSCub74803存在，**输入丢弃**。

共享端口适配器(温泉) (CRS， XR 12000)，有无效MAC的数据包将由SPA l2-tcam丢弃，因此您能找到在**show controllers TenGigE a/b/c/d全部的**这些丢包：

```
    Input drop other            = 107

   l2-tcam Invalid DA Drops: 107
```

在ASR 9000，**输入丢弃无效**在控制器stats的**其他**计数器没有被增加的**DMAC**和**输入丢弃**。因此方式认可在ASR 9000的这些丢包将找到NP处理与输入丢弃的接口：

```
RP/0/RSP0/CPU0:obama#sh int gig 0/0/0/30 | i "input drops"
Wed Aug 22 16:55:52.374 CEST
     1155 packets input, 156256 bytes, 1000 total input drops
RP/0/RSP0/CPU0:obama#sh contr np ports all location 0/0/CPU0
Wed Aug 22 16:56:01.385 CEST

                Node: 0/0/CPU0:
----------------------------------------------------------------

NP Bridge Fia                       Ports
-- ------ --- -----------------------------------------------------
0  0      0   GigabitEthernet0/0/0/30 - GigabitEthernet0/0/0/39
1  0      0   GigabitEthernet0/0/0/20 - GigabitEthernet0/0/0/29
2  1      0   GigabitEthernet0/0/0/10 - GigabitEthernet0/0/0/19
3  1      0   GigabitEthernet0/0/0/0 - GigabitEthernet0/0/0/9
RP/0/RSP0/CPU0:obama#
```

因此您能看到接口gig 0/0/0/30由0/0/CPU0的NP 0处理。
请检查NP0 NP计数器在0/0/CPU0的：

```
RP/0/RSP0/CPU0:obama#sh contr np counters np0 location 0/0/CPU0
Wed Aug 22 16:56:19.883 CEST

                Node: 0/0/CPU0:
----------------------------------------------------------------

Show global stats counters for NP0, revision v3

Read 26 non-zero NP counters:
Offset  Counter                                         FrameValue   Rate (pps)
--------------------------------------------------------------------------------
  22  PARSE_ENET_RECEIVE_CNT                              1465           0
  23  PARSE_FABRIC_RECEIVE_CNT                            2793           0
  24  PARSE_LOOPBACK_RECEIVE_CNT                          2800           0
  28  MODIFY_FABRIC_TRANSMIT_CNT                            80           0
  29  MODIFY_ENET_TRANSMIT_CNT                            1792           0
  32  RESOLVE_INGRESS_DROP_CNT                            1000           0
  35  MODIFY_EGRESS_DROP_CNT                              1400           0
  36  MODIFY_MCAST_FLD_LOOPBACK_CNT                       1400           0
  38  PARSE_INGRESS_PUNT_CNT                               465           0
  39  PARSE_EGRESS_PUNT_CNT                                155           0
  45  MODIFY_RPF_FAIL_DROP_CNT                            1400           0
  53  PARSE_LC_INJECT_TO_FAB_CNT                            80           0
  54  PARSE_LC_INJECT_TO_PORT_CNT                          864           0
  57  PARSE_FAB_INJECT_UNKN_CNT                            155           0
  67  RESOLVE_INGRESS_L3_PUNT_CNT                          465           0
  69  RESOLVE_INGRESS_L2_PUNT_CNT                          464           0
  70  RESOLVE_EGRESS_L3_PUNT_CNT                          1400           0
  93  CDP                                                  464           0
  95  ARP                                                    1           0
 109  DIAGS                                                154           0
 221  PUNT_STATISTICS                                     9142           1
 223  PUNT_DIAGS_RSP_ACT                                   155           0
 225  PUNT_DIAGS_RSP_STBY                                  155           0
```

```
 227   NETIO_RP_TO_LC_CPU_PUNT                                155           0
 373   L3_NOT_MYMAC                                          1000           0
 565   INJECT_EGR_PARSE_PRRT_PIT                              928           0

RP/0/RSP0/CPU0:obama#
```

在NP计数器的所以L3_NOT_MYMAC意味着路由器接收在一个第3层接口的一帧与不是其中一个接口的目标MAC地址。并且路由器下降它正如所料，并且这报告作为在show interface的输入丢弃。在数据包的ASR 9000接收与在ASR 9000的子接口没配置的dot1q vlan，**输入丢弃未知802.1Q**计数器在**show controllers gigabitethernet没有**被增加**0/0/0/30 stats**。步骤是相同的如上所述为未知DMAC ：识别哪位NP处理接口然后检查此NP计数器。您看到NP抵抗增加的**UIDB_TCAM_MISS_AGG_DROP**在那种情况下。

被丢弃的数据包由于无法识别的上层协议

一个是直接的检测，尽管有这些丢包的一个计数器一条线路在**show interface的输入丢弃之下**：

```
RP/0/RSP0/CPU0:obama#sh int gig 0/0/0/18
Wed Aug 22 17:14:35.232 CEST
GigabitEthernet0/0/0/18 is up, line protocol is up

  5 minute input rate 4000 bits/sec, 0 packets/sec
  5 minute output rate 5000 bits/sec, 0 packets/sec
     7375 packets input, 6565506 bytes, 1481 total input drops
     1481 drops for unrecognized upper-level protocol
```

您能看到此处所有输入丢弃归结于无法识别的上层协议。
那含义该数据包接收与以太网协议路由器不是感兴趣。因此意味着功能在邻居配置(或在主机连接对第2层域连接对该接口)，以便发送我们有在XR路由器没配置的协议的帧。
示例：BPDU、Intermediate System to Intermediate System (ISIS)，连接较少网络协议(CLNP)，IPv6、UDLD、思科设备发现协议(CDP)、VLAN中继协议(VTP)、动态中继协议(DTP)，链路层发现协议(LLDP)等….

当这些功能在XR接口时没有配置， XR方框然后丢弃他们正如所料。要欲知什么样的帧增加此计数器，您将必须比较哪些功能在有在邻居启用的功能的XR路由器启用(它可以是路由器或交换机)，或者在所有设备启用的功能连接对第2层域连接对该接口(较不容易)。如果XR路由器连接到交换机，您能尝试在发送到在接口的XR路由器与输入丢弃数据包的该交换机的一个间距。请参阅：
ASR9000/XR ：无法识别的上层协议错误的丢包

在ASR 9000的NP丢包

在网络进程(NP)的丢弃计数器ASR 9000报告作为输入丢弃，当他们适用于接收在接口和被丢弃时的数据包。这不为分组交换机在CRS和XR 12000的引擎(PSE)丢包发生：他们没有算作是输入丢弃。

因此，如果有在ASR 9000的输入丢弃，并且他们不匹配这些原因之一，然后您将执行**show controllers np端口所有位置0/<x>/CPU0**找到NP处理与输入丢弃的接口然后检查其NP计数器与请**显示contr np计数器np<y>位置0/<x>/CPU0**。

您能管道传送输出只保持丢弃计数器用一命令类似**嘘contr np计数器np<y>位置0/<x>/CPU0|我丢弃**，但是这可以是危险的，因为丢弃计数器有时没有在其名称的丢弃。您参见与L3_NOT_MYMAC的一好的实例。那么可能**丢弃的**管道**|丢弃|没有|EXCD**.

您能清除接口计数器和NP计数器与**clear controller np计数器**NP抵抗增量以速率和输入丢弃一样的**np<y>位置0/<x>/CPU0**大致在同一时间发现。

例如，您获得意味着在NP计数器的IPV4_PLU_DROP_PKT CEF/PLU条目说数据包必须丢弃。 您没有一个默认路由并且有不可达的如此禁用的数据包不匹配具体的路由其中点击是丢弃条目的默认CEF处理程序。

如果在能解释输入丢弃的NP查找一个丢弃计数器，当他们增加在同一速率，但是NP丢弃计数器不是明显的，您能导航此页设法了解什么计数器含义：
[ASR9000/XR ：排除故障丢包和了解NP丢弃计数器]

> **Note**:在支持论坛的Xander的页包含第一代的丢弃原因线路卡(三叉戟)，并且有新的计数器名称对于新一代(台风)线路卡…，但是基于名称，您一定能查找一相似的计数器名称和在三叉戟。

Netio

您能收集**show netio idb <int>，**并且这给您接口输入丢弃和netio节点丢弃计数器：

```
RP/0/RP0/CPU0:ipc-lsp690-r-ca-01#show netio idb gigabitEthernet 0/2/0/1

GigabitEthernet0/2/0/1 (handle: 0x01280040, nodeid:0x21) netio idb:
---------------------------------
name:                     GigabitEthernet0_2_0_1
interface handle:         0x01280040
interface global index:   3
physical media type:      30
dchain ptr:               <0x482e0700>
echain ptr:               <0x482e1024>
fchain ptr:               <0x482e13ec>
driver cookie:            <0x4829fc6c>
driver func:              <0x4829f040>
number of subinterfaces: 4096
subblock array size:      7
DSNCNF:                   0x00000000
interface stats info:
  IN  unknown proto pkts:  0
  IN  unknown proto bytes: 0
  IN  multicast pkts:      0
  OUT multicast pkts:      0
  IN  broadcast pkts:      0
  OUT broadcast pkts:      0
   IN  drop pkts:           0 <============ cleared when added to input drop counter !!!
  OUT drop pkts:           0
  IN  errors pkts:         0
  OUT errors pkts:         0

Chains
--------------------
Base decap chain:
    ether                <30>  <0xfd018cd8, 0x482c736c>  <       0,        0>
```

```
Protocol chains:
---------------
<Protocol number> (name)  Stats
 Type  Chain_node       <caps num>  <function, context> <drop pkts, drop bytes>
 <snip>
 <13> (mpls)   Stats IN: 204 pkts, 23256 bytes; OUT: 0 pkts, 0 bytes
   Encap:
    mpls                <25>  <0xfcc7ddbc, 0x00000000>  <       0,        0>
    ether               <30>  <0xfd0189b4, 0x482c736c>  <       0,        0>
    l2_adj_rewrite      <86>  <0xfcaa997c, 0x4831a2e8>  <       0,        0>
    pcn_output          <54>  <0xfd0561f0, 0x48319f04>  <       0,        0>
    q_fq                <43>  <0xfd05f4b8, 0x48320fec>  <       0,        0>
    txm_nopull          <60>  <0xfcadba38, 0x4824c0fc>  <       0,        0>
   Decap:
    pcn_input           <55>  <0xfd0561f0, 0x4830ba8c>  <       0,        0>
    q_fq_input          <96>  <0xfd05f330, 0x48312c7c>  <       0,        0>
     mpls                <25>  <0xfcc7b2b8, 0x00000000>  <     152,    17328>
   Fixup:
    l2_adj_rewrite      <86>  <0xfcaa945c, 0x00000000>  <       0,        0>
    pcn_output          <54>  <0xfd0561f0, 0x48319f04>  <       0,        0>
    q_fq                <43>  <0xfd05f4b8, 0x48320fec>  <       0,        0>
    txm_nopull          <60>  <0xfcadba38, 0x4824c0fc>  <       0,        0>
```

在此处多协议标签交换(MPLS)节点的丢包可能归结于超时的MPLS存活时间(TTL) (在环路的情况下或，当客户执行traceroute)或分段要求和不分段(DF)设置的例如位。您能运行**调试mpls丢包**和**调试mpls错误**以接口的位置设法推测什么样的数据包增加此计数器。

被踢的组播信息包。当您没看到netio **IN丢弃pkts**，但是下面netio节点与可能解释**IN丢弃pkts的**一些丢包时，然后这也许是mcast被踢的数据包，并且您能使**deb mfib netio丢弃**推测什么样的数据包