Understanding Buffer Misses and Failures

Document ID: 14620

Contents

Introduction

This document discusses buffer misses and failures on the Routing Processor (RP).

Prerequisites

Requirements

There are no specific requirements for this document.

Components Used

This document is not restricted to specific software and hardware versions.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Conventions

Refer to Cisco Technical Tips Conventions for more information on document conventions.

Buffer Misses and Failures

The RP divides its processor memory into pools. Each pool contains a number of memory blocks of equal size. These memory blocks are called buffers.

Buffer Pools

There are six buffer pools:

- Small¡04 bytes buffers
- Middle¦00 byte buffers
- Big¡524 byte buffers

* VeryBig¤520 byte buffers
* Large¥024 byte buffers
* Huge¡8024 byte buffers

For example, if an interface processor needs to pass a 20 byte packet to the RP, it asks for a Small buffer. If an interface processor needs to pass a 500 byte packet to the RP, it asks for a Middle buffer, and so forth.

Note: The interface processor must ask for a buffer of a certain size.

When the interface processor asks for a buffer, this occurs:

* If a free buffer exists within the requested pool, the buffer is granted. Otherwise, the request generates a miss and the buffer algorithm tries to create more buffers for that pool .
* When IOS fails to get a Small buffer, it does not drop the packet. It increments the failed counter and falls through to the next level buffer, which is the Middle buffer and requests a buffer there. If it fails to get a Middle buffer, it requests the next level buffer, which is a Big buffer. This process continues until it hits the Huge buffer pool. If it fails to get a Huge buffer, then it drops the packet.
* When you use the IBM feature set, a miss almost always generates a failure.
* Although the IBM features may be process−switched, the code to get a buffer to pass a packet from an interface to the RP executes at interrupt level.
* Buffers can not be created at interrupt level; consequently, a miss queues its request for more buffers to the RP.
* Because an additional buffer can not be created on the spot, the buffer request fails, and the packet is dropped.

Buffer failures are one of the most common reasons for packet drops. When packet drops occur because of buffer failure, this occurs:

* After a buffer failure, the RP has an outstanding request to create more buffers of the appropriate size for the particular pool.
* While the RP is servicing the create buffers request, there may be additional failures in the pool.
* The RP may even fail to create more buffers, because of memory constraints in the system when the extra buffers are required.
* Essentially, the create buffers operation could take several microseconds, in which packets are continually dropped because of the buffer shortage.
* In addition, if buffers are used as quickly as they are created, the RP could be forced to spend more time on buffer creation than on packet processing.
* This may cause the RP to begin to drop packets so quickly that performance degrades and sessions are lost.

Fortunately, as this document discusses, buffer failure problems are not difficult to identify and resolve. This **show buffers** command output shows the current state of the router s buffer pools:

```
dspu−7k#show buffers

Buffer elements:
     500 in free list (500 max allowed)
     2370 hits, 0 misses, 0 created

Public buffer pools:
Small buffers, 104 bytes (total 16, permanent 10):
     11 in free list (0 min, 10 max allowed)
     1770 hits, 33 misses, 22 trims, 28 created
     9 failures (0 no memory)
Middle buffers, 600 bytes (total 90, permanent 90):
     89 in free list (10 min, 200 max allowed)
     590 hits, 0 misses, 0 trims, 0 created
```

```
     0 failures (0 no memory)
Big buffers, 1524 bytes (total 90, permanent 90):
     90 in free list (5 min, 300 max allowed)
     126 hits, 0 misses, 0 trims, 0 created
     0 failures (0 no memory)
VeryBig buffers, 4520 bytes (total 10, permanent 10):
     10 in free list (0 min, 300 max allowed)
     50 hits, 0 misses, 0 trims, 0 created
     0 failures (0 no memory)
Large buffers, 5024 bytes (total 10, permanent 10):
     10 in free list (0 min, 30 max allowed)
     0 hits, 0 misses, 0 trims, 0 created
     0 failures (0 no memory)
Huge buffers, 18024 bytes (total 2, permanent 0):
     0 in free list (0 min, 13 max allowed)
     2 hits, 2 misses, 0 trims, 2 created
     0 failures (0 no memory)
```

In the **show buffers** output:

- `Total` identifies the total number of buffers in the pool, which include used and unused buffers.
- `Permanent` identifies the permanent number of allocated buffers in the pool. These buffers are always in the pool and can not be trimmed away.
- `In free list` identifies the number of buffers currently in the pool that are available for use.
- `Min` identifies the minimum number of buffers that the RP should attempt to keep in the free list:
    - The `min` parameter is used to anticipate demand for buffers from the pool at any given time.
    - If the number of buffers in free list falls below the `min` value, the RP attempts to create more buffers for that pool.
- `Max-allowed` identifies the maximum number of buffers that are allowed in the free list:
    - The `max-allowed` parameter prevents a pool from monopolizing buffers that it does not need anymore. It also frees this memory back to the system for further use.
    - If the number of buffers in the free list is greater than the `max-allowed` value, the RP should attempt to trim buffers from the pool.
- `Hits` identifies the number of buffers that have been requested from the pool. The `hits` counter provides a mechanism to determine which pool must meet the highest demand for buffers.
- `Misses` identifies the number of times that a buffer has been requested and the RP detected in which pool additional buffers were required. In other words, the number of buffers in the free list has dropped below `min` level. The `misses` counter represents the number of times the RP has been forced to create additional buffers.
- `Trims` identifies the number of buffers that the RP has trimmed from the pool, when the number of buffers in the free list exceeded the number of `max-allowed` buffers.
- `Created` identifies the number of buffers that have been created in the pool. The RP creates buffers in these situations:
    - When demand for buffers has increased until the number of buffers in the free list is less than the `min` buffers.
    - A `miss` occurs because there are no buffers in the free list.
    - Both of the previous situations.
- `Failures` identifies when IOS fails to get a Small buffer, it does not drop the packet. It increments the failed counter and falls through to the next level buffer, which is the Middle buffer and requests a buffer there. If it fails to get a middle buffer, it requests the next level buffer, which is a Big buffer. This process continues until it hits the Huge buffer pool. If it fails to get a Huge buffer, then it drops the packet.
- `No memory` identifies the number of failures caused by insufficient memory to create additional buffers.

You can can examine the characteristics of each pool, to determine which pools (if any) are encountering problems. The parameters for a pool can be tuned to allow the router to be better prepared to handle the load, if the pool seems to exhibit these characteristics:

- The number of misses and creates increment at a high rate (as a percentage of hits).
- There is a consistently low number of buffers in the free list.
- The number of `failures` or `no memory` increment.

buffers Configuration Command

With the **buffers** configuration command, you can tune these parameters for each buffer pool:

- **initial** Temporary buffers that are allocated at system reload.
- **max−free** Maximum number of free buffers.
- **min−free** Minimum number of free buffers.
- **permanent** Number of permanent buffers.

Initial Buffers

Tune **initial** buffers to accommodate the burst of session−establishment traffic after router reload.

```
buffers small initial 250
```

These buffers are eventually trimmed and returned to the system.

The initial buffers are designed to handle session establishment, which is always process−switched.

During session establishment, the fastswitching cache (used by other route protocols) is populated; process−switched buffers are no longer required and may be returned to the system.

To tune initial buffers may not be the correct solution for the IBM feature set, because almost all packets (after session establishment) are process−switched and require the additional buffering anyway.

Note: For the IBM process−switched features, you should tune **permanent** buffers rather than tune the temporary initial buffers.

Max−Free Buffers

Tune **max−free** buffers so that the value is equal to or greater than the permanent buffers. If all permanent buffers are in the free list, then the RP should not try to trim permanent buffers. Max−free can be used to ensure that unused buffers that are created during irregular bursts are returned to the system memory.

```
buffers small max−free 175

buffers small permanent 125
```

Min−Free Buffers

Tune **min−free** buffers so that the value represents the estimated minimum number of buffers required at any time. Min−free can be used to anticipate buffer shortage conditions and to ensure that a minimum number of buffers are always available.

```
buffers small min−free 50
```

Permanent Buffers

Tune **permanent** buffers so that the value represents the estimated number of buffers required for normal processing.

```
buffers small permanent 125
```

Permanent buffers are used to accommodate the normal buffer requirements (including frequent bursts) of the router. Determination of the normal buffer requirements is an interactive process, where the **show buffer** output should show the total buffers used in a pool at a given time. Permanent buffers should be tuned with regard to the consistent "total" buffers required. When you tune permanent buffers, you should focus on the reduction of creates and the elimination of misses and failures.

Additional show Commands

There are two other **show** commands that you can use to identify problems with buffer allocation:

- **show interfaces** *interface−identifier*
- **show source−bridge**

This **show interfaces** *interface−identifier* sample command output includes a counter for no buffer:

```
dspu−7k#show interfaces channel 4/2

Channel4/2 is up, line protocol is up
  Hardware is cxBus IBM Channel
  MTU 4472 bytes, BW 98304 Kbit, DLY 100 usec, rely 255/255, load 1/255
  Encapsulation CHANNEL, loopback not set, keepalive not set
  Virtual interface
  Last input 0:00:04, output 0:00:04, output hang never
  Last clearing of "show interface" counters never
  Output queue 0/40, 0 drops; input queue 0/75, 8 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     646 packets input, 27760 bytes, 8 no buffer
     Received 0 broadcasts, 0 runts, 0 giants
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     328 packets output, 16959 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets, 0 restarts
     0 output buffer failures, 0 output buffers swapped out
```

In the **show interfaces** *interface−identifier* command output:

- The `no buffer` counter increments when the interface fails to obtain a buffer for an inbound packet.
- Both the `no buffer` and `drops` (input queue) counters increment when the interface fails to obtain a buffer for an inbound packet.
- A `no buffer` counter that increments in the **show interfaces** output correlates to the `misses` counter that increments in the **show buffers** output. The appropriate buffer pool may be tuned.

This **show source−bridge** sample command output includes an interface counter for throttles, when source−route bridging (SRB) is configured for the interface:

```
dspu−7k#show source−bridge

Local Interfaces:                        receive            transmit
       srn bn  trn r p s n  max hops     cnt:bytes          cnt:bytes      drops
Ch4/2  666  1   99 *   f    7  7  7      652:26020            6:266           0
```

```
Global RSRB Parameters:
 TCP Queue Length maximum: 100

Ring Group 99:
  This TCP peer: 150.10.20.2
   Maximum output TCP queue length, per peer: 100
  Peers:                 state     bg lv  pkts_rx  pkts_tx  expl_gn   drops TCP
   TCP 150.10.20.1       open         *3      261      266        0       0   0
   TCP 150.10.20.2       -            *3        0        0        0       0   0
  Rings:
   bn: 1  rn: 888  locvrt ma: 4000.7000.fff1 Buff Ring888          fwd: 0
   bn: 1  RN: 666  local  ma: 4000.0c48.2e80 Channel4/2            fwd: 261
   bn: 1  RN: 88   remote ma: 4000.4000.fff1 TCP 150.10.20.1       fwd: 322
   bn: 1  RN: 250  remote ma: 4000.300f.7c09 TCP 150.10.20.1       fwd: 0

Explorers: ------- input -------             ------- output -------
         spanning  all-rings     total      spanning  all-rings     total
Ch4/2           0          0         0             0          1         1

  Local: fastswitched 0         flushed 0         max Bps 256000

         rings      inputs         bursts         throttles output drops
         Ch4/2           0              0                 8            0
```

In the **show source−bridge** command output:

- The `throttles` counter increments when the interface fails to obtain a buffer for an inbound packet.
- The `throttles` counter that increments in the **show interfaces** command output correlates to a `misses` counter that increments in the **show buffers** command output. The appropriate buffer pool may be tuned.

Related Information

- **Buffer Tuning for all Cisco Routers**
- **Technical Support & Documentation − Cisco Systems**

Updated: Sep 09, 2005 Document ID: 14620