Router Architecture and Cisco IOS Internals

Session RST-301

Agenda

- CEF
- PXF
- CPU
- Crashes, Tracebacks, Hangs
- Memory Leaks, Corruption
Switching Paths

- Process switching
- Cache-based switching (fast switching)
  - Optimum switching
  - Flow switching
- CBUS/SSE switching/distributed fast switching
- CEF switching
- Distributed CEF switching
  - CPU-based
  - HW-based
- MLS flow switching
- MLS CEF flow switching
- PXF switching
CEF—Failure Reasons

- Why does CEF fail?
- How do you know it is CEF?
- What should we look for?

CEF—Status

- Is CEF operational?

```
1200RTR-8# sh ip cef summ
IP Distributed CEF with switching (Table Version 359)
  42 routes, 0 reresolve, 0 unresolved (0 old, 0 new)
  42 leaves, 32 nodes, 38656 bytes, 183 inserts, 141 invalidations
  0 load sharing elements, 0 bytes, 0 references
  2 CEF resets, 149 revisions of existing leaves
  181 in-place modifications
  refcounts: 5964 leaf, 5932 node

Adjacency Table has 4 adjacencies
1200RTR-8#
```
CEF—Status

- How to check if CEF is operational on the linecards:

```
12008RTR-8#sh cef line
CEF table version 359, 42 routes
Slot CEF-ver MsqSent XdrSent Seq MaxSeq LowQ MedQ HighQ Flags
0 355 6063 6567 4 28 0 0 0 up, sync
3 355 6063 6565 4 28 0 0 0 up, sync
5 355 6063 6566 4 28 0 0 0 up, sync
2 355 5217 5714 4 28 0 0 0 up, sync
1 355 749 956 4 28 0 0 0 up, sync
```

CEF—Resetting

- Clear options per linecard are:

```
12008RTR-8#clear cef line ?
<0-15> linecard slot number
Adjacency Clear adjacency information
Interface Clear interface related information
Prefix Clear Cisco Express Forwarding table
<cr>
```

- Check if tables are synchronized

```
X = <inserts> - <invalidations>
Must match between LC and RP
```
CEF—Status

LC-Slot0>sh ip cef summ
IP Distributed CEF with switching (Table Version 45)
42 routes, 0 reresolve, 0 unresolved (0 old, 0 new)
42 leaves, 32 nodes, 38656 bytes, 236 inserts, 194 invalidations
0 load sharing elements, 0 bytes, 0 references
3 CEF resets, 149 revisions of existing leaves
0 in-place modifications
refcounts: 5964 leaf, 5932 node

Received 6068 messages (Priority - 5919, High - 43 Medium - 2 Low - 104)
Messages from RP waiting for processing - High 0 Medium 0 Low 0
Sent 709011 messages, 1639120 xdrs. Linecard to RP queue size 0

Adjacency Table has 4 adjacencies
LC-Slot0>

• X=236–194=42

CEF—IPC

• IPC errors
  LC disabled if no IPC msg received in 5 minutes

12012RTR-3#sh ipc ?
  nodes  Show participating nodes
  ports  Show local IPC ports
  queue  Show the IPC retransmission queue
  status  Show status of local IPC server
12012RTR-3#sh ipc queue
There are 0 IPC messages waiting for acknowledgement in the transmit queue.
There are 0 messages currently in use by the system.

12012RTR-3#sh ipc stat
IPC System Status:
1000 IPC message headers in cache
20394229 messages in, 30660 out, 20390418 delivered to local port,
1946 acknowledgements received, 2242512 sent,
0 NACKs received, 0 sent,
0 messages dropped on input, 0 messages dropped on output
0 no local port, 0 destination unknown, 0 no transport
0 missing callback or queue, 0 duplicate ACKs, 1 retry,
0 message timeouts.
0 ipc_output failures, 0 mtu failures,
0 msg alloc failed, 0 emer msg alloc failed, 0 no origs for RPC replies
0 pak alloc failed, 0 memd alloc failed
0 no hq, 0 failed opens, 0 hardware errors
No regular dropping of IPC output packets for test purposes
**CEF—IPC**

12012RTR-3#sh ipc nodes
There are 4 nodes in this IPC realm.

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Name</th>
<th>Last Sent</th>
<th>Last Heard</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>Local</td>
<td>IPC Master</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1080000</td>
<td>GSR</td>
<td>GSR Slot 8</td>
<td>23</td>
<td>41</td>
</tr>
<tr>
<td>1070000</td>
<td>GSR</td>
<td>GSR Slot 7</td>
<td>23</td>
<td>29</td>
</tr>
<tr>
<td>1090000</td>
<td>GSR</td>
<td>GSR Slot 9</td>
<td>23</td>
<td>1140</td>
</tr>
</tbody>
</table>

---

**CEF—Interfaces**

- **Checking an interface**

12008RTR-8#sh cef int
GigabitEthernet3/0 is up (if_number 8)

- Per packet loadbalancing is enabled
- IP unicast RPF check is disabled
- Hardware idb is GigabitEthernet3/0
- Fast switching type 1, interface type 27
- IP Distributed CEF switching enabled
- IP Null turbo vector
CEF

- FIB is copy of routing table

```
12008RTR-8#sh ip cef
Prefix             Next Hop          Interface
0.0.0.0/0           attached          ATM2/0
                    attached          POS0/2
0.0.0.0/32          receive          POS0/2
2.2.2.2/32          77.6.8.6         POS0/2
```

CEF—Adjacencies

- Adjacency types
  - Cached
  - Glean
  - Receive
  - Punt
  - Null
  - Discard
  - Drop
CEF—Adjacencies

12008RTR-8# show ip cef 57.4.6.0 detail
57.4.6.0/24, version 333, cached adjacency to POS0/2
  0 packets, 0 bytes
    via 77.6.8.6, POS0/2, 0 dependencies
      next hop 77.6.8.6, POS0/2
      valid cached adjacency
12008RTR-8#
12008RTR-8# show ip route 57.4.6.0
Routing entry for 57.4.6.0/24
  Known via "ospf 1", distance 110, metric 6, type intra area
  Redistribution via ospf 1
  Last update from 77.6.8.6 on POS0/2, 1d07h ago
Routing Descriptor Blocks:
  * 77.6.8.6, from 4.4.4.4, 1d07h ago, via POS0/2
    Route metric is 6, traffic share count is 1

CEF—Adjacencies

12008RTR-8# show ip cef 42.42.42.0 255.255.255.0
42.42.42.0/24, version 21, attached, connected
  0 packets, 0 bytes
    via GigabitEthernet3/0, 0 dependencies
      valid glean adjacency
12008RTR-8# show ip route 42.42.42.0
Routing entry for 42.42.42.0/24
  Known via "connected", distance 0, metric 0 (connected, via interface)
Routing Descriptor Blocks:
  * directly connected, via GigabitEthernet3/0
    Route metric is 0, traffic share count is 1
**CEF—Adjacencies**

RTR12410-1#show ip int brie | include 12.10.10.1
SRP7/0          12.10.10.1     YES manual up           up
RTR12410-1#show ip cef 12.10.10.1 det
12.10.10.1/32, version 132025, receive

12008RTR-8#sh ip cef 224.0.0.0 240.0.0.0 int
224.0.0.0/4, version 363
0 packets, 0 bytes, Precedence routine (0)
  via 0.0.0.0, 0 dependencies
    next hop 0.0.0.0
    valid punt adjacency

12008RTR-8#sh ip cef 5.0.0.0
5.0.0.0/8, version 361, attached
0 packets, 0 bytes
  via Null0, 0 dependencies
  valid null adjacency

**CEF—Layer 2 rewrites**

12008RTR-8#show ip arp 42.42.42.1
Protocol    Address   Age (min)   Hardware Addr  Type   Interface
Internet  42.42.42.1              5   0030.f2c9.8338   ARPA   GigabitEthernet3/0

12008RTR-8#show adjacency internal | begin 42.42.42.1
IP         GigabitEthernet3/0 42.42.42.1(5)
  9 packets, 1098 bytes
    0000000000000000000000000000000000000000000000000000000000000000
    0030F2C983380010F6D8B8600800
    03:55:37
    Fast adjacency disabled
    IP redirect enabled
    IP mtu 1500 (0x0), index 4
    Output qs (0x0/0x4000)
    rawq slot 0 port 0 channel 0
    Output info 0x1E019260, Encap 0x1
    Fixup disabled
**CEF—Load Balancing**

- Load balancing with CEF
  
  SA/DA pair default
  
  Per packet (not supported with certain hardware based switching schemes)

```
12008RTR-8#sh ip cef 0.0.0.0 0.0.0.0 internal
0.0.0.0/0, version 405, attached, per-destination sharing
0 packets, 0 bytes
  via 42.42.42.1, 0 dependencies, recursive
  traffic share 1
  valid adjacency
via 77.6.8.6, 0 dependencies, recursive
  traffic share 1
  valid adjacency

0 packets, 0 bytes switched through the prefix
tmstats: external 0 packets, 0 bytes
  internal 0 packets, 0 bytes
Load distribution: 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 (refcount 1)
```

<table>
<thead>
<tr>
<th>Hash</th>
<th>OK</th>
<th>Interface</th>
<th>Address</th>
<th>Packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Y</td>
<td>GigabitEthernet3/0</td>
<td>42.42.42.1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Y</td>
<td>POS0/2</td>
<td>point2point</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Y</td>
<td>POS0/2</td>
<td>point2point</td>
<td>0</td>
</tr>
</tbody>
</table>
 Packet accounting with loadsharing

RTR12410-1(config)#ip cef accounting loadbalance-hash

CEF traffic statistics

RTR12410-1#show cef line <slot>
LC-SlotX#show cef int stat
LC-SlotX#show cef int <interface> Y
12008RTR-8#sh int [<interface>] stat

CEF—Statistics

RTR12410-1#show cef line 2
CEF linecard slot number 2, status up, sync
Linecard CEF version number 148331
Sequence number 14739, Maximum sequence number expected 14763, Seq Epoch 4
Send failed 0, Out Of Sequence 0, drops 0
Linecard CEF reset 4, reloaded 4
240702 elements packed in 177755 messages(7666234 bytes) sent
9 elements cleared
linecard in sync after reloading
0/0/0 xdr elements in LowQ/MediumQ/HighQ
2940/9/51 peak elements on LowQ/MediumQ/HighQ
Input packets 372394, bytes 233127656
Output packets 349988, bytes 255498679, drops 0
### CEF—Statistics

**LC-Slot0#sh cef int stat**

<table>
<thead>
<tr>
<th>Interface</th>
<th>Pkts In</th>
<th>Chars In</th>
<th>Pkts Out</th>
<th>Chars Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>GigabitEthernet3/0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>POS5/0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Ethernet0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Null0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ATM2/0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Loopback0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>POS0/0</td>
<td>51</td>
<td>2856</td>
<td>46</td>
<td>2576</td>
</tr>
<tr>
<td>POS0/1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>POS0/2</td>
<td>23497</td>
<td>140932</td>
<td>11</td>
<td>640</td>
</tr>
</tbody>
</table>

---

**LC-Slot2#show cef int pos 0**

POS2/0 is down (if_number 7)

Corresponding hwidb fast_if_number 7
Corresponding hwidb firstsw->if_number 7
Internet address is 12.4.10.2/30
ICMP redirects are always sent

Input fast flags 0x0, Output fast flags 0x0, Flags 0x41
ifindex 6(6)
Slot 2 Slot unit 0 VC -1
Transmit limit accumulator 0x0 (0x0)
IP MTU 4470
Switching statistics:
  Input: 372357 packets, 233124496 bytes
  Output: 349973 packets, 255494639 bytes
  0 input drops, 0 output drops
CEF—Accounting

- CEF accounting

```
12008RTR-8(config)#ip cef accounting?
non-recursive Enable accounting for traffic through non-recursive prefixes
per-prefix Enable per prefix accounting
prefix-length Enable prefix length accounting
```

CEF—Accounting

```
12008RTR-8#clear ip cef ?
  * clear stats for all cisco express forwarding entries
    A.B.C.D prefix of entry whose stats are to be cleared
12008RTR-8#clear ip cef 33.0.0.0 ?
    A.B.C.D prefix mask of entry to clear
    prefix-statistics clear stats for cisco express forwarding prefixes

12008RTR-8(config)#ip cef traffic-statistics ?
    load-interval Set measurement period
    update-rate Specify update rate for nonrecursive prefix stats
<cr>
```
CEF—Accounting

12008RTR-8#sh cef not-cef-switched
CEF Packets passed on to next switching layer
Slot No_adj No_encap Unsupp'ted Redirect Receive Options Access Frag
RP 0 0 0 0 0 0 0 0
3 0 0 0 78 0 2256 0 0

12008RTR-8#sh cef drop
CEF Drop Statistics
Slot Encap_fail Unresolved Unsupported No_route No_adj ChkSum_Err
RP 0 0 0 0 2440 0
3 0 0 0 0 0 0

CEF—Debugs

RTR12410-1#debug ip cef ?
accounting Accounting events
drops Packets dropped by CEF
events IP CEF table events
fragmentation IP CEF fragmentation
hash IP CEF hash events
interface IP CEF interface events
interface-ipc Interface events related IPC
ipc IP CEF IPC events
prefix-ipc IP-prefixes related IPC
receive Packets received by IP CEF
subblock IP CEF subblock events
table IP CEF table changes
CEF

• CEF FAQ
  At least 64 megs, usually 128 or 256 megs of memory needed on LC
  No forwarding interruption during download of route table changes
  Only switching path in many hardware based platforms
  Allows Internet to scale

CEF—Inconsistency Checks

• CEF inconsistency checks
  Enabled via CSCds61128
  Off by default via CSCdt35866
  May want the fix for CSCdt18447, CSCuk23290
  • Helps troubleshoot any CEF issues
CEF—Inconsistency Checks

RTR12410-1(config)#ip cef table ?
adjacency-prefix Set adjacency-prefix characteristics
consistency-check Set consistency checking characteristics
event-log Set table event-log characteristics
resolution-timer Specify background resolution timer

RTR12410-1(config)#ip cef table event-log ?
machine Log events matching specified prefix/mask
size Number of event entries
<cr>

RTR12410-1#show ip cef events ?
A.B.C.D Show events relating to this prefix
detail Display full information
new Display new events since last show
summary Display summary of event log
within Show events within specified time
| Output modifiers
<cr>

RTR12410-1#show ip cef events trace

RTR12410-1#show ip cef events summar
CEF table events summary:
Storage for 10000 events (320000 bytes), 40/0 events recorded/ignored
Matching all events
Last event occurred 00:01:02.444 ago.
CEF—Inconsistency Checks

RTR12410-1(config)#ip cef table ?
   adjacency-prefix   Set adjacency-prefix characteristics
   consistency-check  Set consistency checking characteristics

RTR12410-1#show ip cef inconsistency records detail
Table consistency checkers (settle time 65s)
lc-detect: running
  0/0/0 queries sent/ignored/received
scan-lc: running [100 prefixes checked every 60s]
  0/0/22199 queries sent/ignored/received
scan-rp: running [100 prefixes checked every 60s]
  22099/0/0 queries sent/ignored/received
scan-rib: running [1000 prefixes checked every 60s]
  26488/0/26488 queries sent/ignored/received
Inconsistencies: 0 confirmed, 0/4 recorded
CEF—Information to Gather

- What to gather when a CEF bug hits
  
  `show ip cef <problem prefix>`
  
  `exec all show ip cef <problem prefix>`
  
  `show ip cef inc rec detail`
  
  `exec slot <problem slot> show ip cef inc rec detail`
  
  `show tech cef`

CEF—Recovery

- Recovering from a CEF bug
  
  If a prefix is missing from a linecard, then use
  `clear cef linecard <slot>`
  
  If a prefix is missing from the RP, then use
  `clear ip route *`
  
  Reset the consistency checkers with `clear ip cef inconsistency`
CEF—GSR Engine 1/2/4 LCs

- Check if switching asic and LC agree
- Check adjacency information on RP and rewrite information on egress LC

```
RTR12410-1#show adj int | begin <a.b.c.d>
LC-Slot7#show contr rewrite | begin <a.b.c.d>
```

CEF—GSR Engine 2 LC

- Hardware switched
  Inconsistency between PSA and LC

```
LC-Slot5#show ip psa 1.0.106.1
Leaf FCR 1 0x78003D80 found 3 deep
psa ip loadbalance 16 paths
Hash 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16: psa adjacency: 0x701BA0C0
   [8-7] loq 9C00 mtu 4400 oq 4000 ai 1 oi 12019100 oacl FFFF (encaps
   size 18)
gather 90 (bufhdr size 32 ip profile 4)
counters 0, 0 reported 0, 0.
```
CEF—GSR Engine 2 LC

• Loq—local output queue
• Oq—output queue
• IP MTU, gather header, oacl index

CEF—GSR Engine 2 LC

• loq = 90C0 = 1001 0000 1100 0000
  - Bits 0–2 is for the RED queue which is 0
  - Bits 3–5 is for the COS queue which is 0
  - Bits 6–9 is for the port on output LC which is 3
  - Bits 10–13 is for the output slot which is 4
  - Bits 14–15 is for denoting whether the queue is unicast or multicast
  - The value 10 for these bits indicate a unicast queue
CEF—GSR Engine 2 LC

- oq = 40C0 = 0100 0000 1100 0000
  
  Bits 0–2 is for RED queue which is 0
  Bits 3–5 is for COS queue which is 0
  Bits 6–9 is for channel/port which is 3
  Bits 10–13 are not used
  Bits 14–15 are for denoting unicast or multicast queue
  The value 01 indicates unicast queue

CEF—GSR Engine 2 LC

LC-Slot5#show ip psa ?

A.B.C.D prefix of entries to show
adjacency Display all prefixes resolving through adjacency
detail Display full information
CEF—GSR Engine 4 LC

LC-Slot1#show ip gen6-cef 1.0.105.251
HW Leaf :72800FD8, IP 1.0.105.251, LeafContents: 57600047C1704017 :Pkt 0,
Byt 0
Lbl_L1 0x7923E0B8, Bundle 1, Lbl Opcode 7, Is_Fast 0, Lbl_L2 ptr 0x79041200
LbL_L2 79041200 LbL10 Cos0, GIF 0, LQF 0, OQF 0, TagOp 0, Adj_L3 0x79000380
AdjL3 0x79000380, MTU 0, Slot 7, LQ 0, OQ 4000, BHdr 2, OI 0x12019100, Pkt
7A000
lCO = 0, Byt 0
Leaf 0x72800FD8 found 3 deep
Prefix Counters:
byte_counter 0, packet_counter 0.
fib->hwleaf->lbl_L1 7923E0B8 = 1708240
loadinfo 41DCB7C0, loadinfo->hwloadinfo 79041200 = 70
ip loadbalance 16 paths lbl_L2 79041200
adj_L3 79000380
Hash 1,2,3,4,5,6,7,8adj 41CD4D00, adj->hw_adjacency 41F37C80
[0-7] oq 4000 loq 0 slot 7 mtu 0 oi 12019100 bfr_hdr 2

CEF

Q and A
• Router crash
• PXF crash
  Show pxf crash
• PXF performance issues
• 10000 (ESR) vs 7200 (NSE)
• Will also be used in other platforms
• Performance issues
  Input interface (ignores/throttles)
  Entry to pxf
  Within pxf
  Drop decision in pxf
  Output interface
  Punting

show hardware pxf cpu cef       - ESR
show pxf feature cef entry      - NSE
• Shows a summary of PXF column memory used by CEF
  show hardware pxf cpu cef a.b.c.d.       - ES
• CEF information for a.b.c.d.
show hardware pxf cpu statistics [ip | drop | diversion]  - ESR
show pxf accounting summary  - NSE
  • Shows chassis-wide PXF forwarding statistics

show hardware pxf cpu statistics drop <interface>
  • Shows interface specific PXF drop statistics
show hardware pxf cpu statistics drop
  • Shows chassis-wide PXF drop statistics
show hardware pxf cpu statistics diversion

- Shows chassis-wide PXF diversion statistic

- PXF pipeline components maintain counters
  Monotonically increasing, never cleared

- To display PXF queueing statistics:
  show hardware pxf cpu queue [<interface>]
  Without <interface> parameter, returns queueing information for the to-RP packet diversion queue
PXF

show hardware pxf cpu context

• Shows how busy the FP is with the current traffic load
• PXF pipeline hung if counters idle

show pxf info

PXF

• Multiple platforms will make use of PXF path
• CLI will be made consistent between all existing and upcoming platforms
PXF

Q and A

Stretch!

CPU
CPU

- Traffic
- Process called repeatedly
- Process stuck in loop
- Alignment errors
- Interrupt level or process level
- CPUHOG

Interpreting the Results

- Show processes
- Show processes cpu
Show Processes

7200dsl#show processes
CPU utilization for five seconds: 0%/0%; one minute: 0%; five minutes: 0%

<table>
<thead>
<tr>
<th>PID</th>
<th>Qty</th>
<th>PC Runtime (ms)</th>
<th>Invoked</th>
<th>uSecs</th>
<th>Stacks</th>
<th>TTY</th>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hwe</td>
<td>600DF5C0</td>
<td>0</td>
<td>1</td>
<td>0 5640/6000</td>
<td>0</td>
<td>SSCOP Input</td>
</tr>
<tr>
<td>2</td>
<td>Lst</td>
<td>601F2034</td>
<td>34300</td>
<td>16747</td>
<td>2048 5764/6000</td>
<td>0</td>
<td>Check heaps</td>
</tr>
<tr>
<td>3</td>
<td>Cwe</td>
<td>601DC210</td>
<td>0</td>
<td>11</td>
<td>0 5652/6000</td>
<td>0</td>
<td>Pool Manager</td>
</tr>
<tr>
<td>4</td>
<td>Mst</td>
<td>6021E638</td>
<td>0</td>
<td>2</td>
<td>0 5612/6000</td>
<td>0</td>
<td>Timers</td>
</tr>
<tr>
<td>5</td>
<td>Mwe</td>
<td>6028E5A0</td>
<td>2104</td>
<td>8</td>
<td>263000 4644/6000</td>
<td>0</td>
<td>OIR Handler</td>
</tr>
</tbody>
</table>

Dissecting show processes

CPU utilization for five seconds: 25%/20%; one minute: 26%; five minutes: 24%

- X/Y: total cpu time / cpu time under interrupt
  total cpu = 25 = (process + interrupt) = (5 + 20)

<table>
<thead>
<tr>
<th>PID</th>
<th>Qty</th>
<th>PC Runtime (ms)</th>
<th>Invoked</th>
<th>uSecs</th>
<th>Stacks</th>
<th>TTY</th>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Process</td>
</tr>
</tbody>
</table>

- PID—process ID
- Q—Process queue priority. H, M, L.
- Ty—Scheduler test. *, E, S, rd, we, sa, si, sp, st, hg, xx
Dissecting Show Processes

- PC—Program Counter; can be decoded using stack decoder
- Runtime—total CPU time used (in ms)
- Invoked—number of times process was called
- uSecs—average duration of each invocation

Dissecting Show Processes

- Stacks—low water mark/total stack space available
- TTY—terminal that controls process
- Process—name of the running process
Show Processes CPU

- Provides the 5Sec, 1Min, 5Min breakdown for each process
- Useful for a quick snapshot
- Should be used in conjunction with show processes

Process Called Repeatedly

<table>
<thead>
<tr>
<th>Process Called Repeatedly</th>
<th>19</th>
<th>63524</th>
<th>1111022</th>
<th>57</th>
<th>25.73%</th>
<th>26.20%</th>
<th>26.10%</th>
<th>0 IP Input</th>
</tr>
</thead>
</table>

- Why is a process called repeatedly?
- IP Input
  - Excessive process switching of ip traffic
  - Could be unicast/multicast/broadcast traffic
  - Enable fastest switching paths for unicast/multicast
  - Broadcast control
- Show ip traffic
Process Called Repeatedly

- Could be because it is stuck in a loop
  
  Get several snapshots of the stack for the process id
  
  Show stack <pid>
  
  Sometimes you may need to get the stack via a
  
  Show memory <address>
  
  Stack can be decoded using rsym/srysm

  To decode dump of show memory, you will need to know how the stack is built

Alignment Errors

- Alignment errors, spurious memory access
  
- Show align
  
- Low traffic, high interrupt level CPU
Typical Causes of High CPU

- **IP Input**
  - We are process switching ip traffic—Why?
  - Is the router [turbo] fast-switching or CEF switching?
  - If the router is [turbo] fast-switching, is it getting cache trashing?
    - What is the available amount of free processor memory?
    - How many route flaps are we getting?
    - How many link flaps are occurring?

- **IP Input**
  - If the router is CEF switching, is CEF getting disabled?
    - How much memory does the router have—The linecards?
    - If DCEF is taking place, what’s the status of the CEF linecards?
  - Use show interface commands to determine if the traffic is unicast-multicast/broadcast
IP Input

- Switching path
- Multicast fast-switching/distributed multicast fast-switching
- Useful commands to troubleshoot fast-switching cache
  show ip cache [optimum | flow | verbose]
  debug ip cache [ACL]

IP Input

- show ip traffic
  Multiple snapshots; which counter is incrementing?
- debug ip error
  Turn off logging to console. Enable it and disable it quickly
- ip route-cache flow
  Monitor flows—Figure out what is going to CPU
Virtual-Exec/Exec

- Virtual-Exec/Exec processes will be high in CPU
  - When running debugs
  - Show commands
  - File System commands
  - Ensure that console logging is disabled before turning on chatty debugs

Feature Check

- ACLs
- Fancy queueing
- Compression
- Encryption
- Accounting
- Encapsulations
- Broadcast flooding
- Explorer forwarding
- Policy routing
- Secondary addresses
- Network address translation
CPU

Q & A

Crash, Traebacks, Hangs
Why Do Routers Crash?

- The router has detected an error it cannot fix
- Software tries to do something it shouldn’t
- Either of these could be HW or SW

Things the Router Cannot Fix

- Processor memory parity error (PMPE)
  
  This is usually seen on RSPs, but has been reported on 4700s, GSRs, anything MIPS that doesn’t have ECC
  
  “System was restarted by processor memory parity error at PC 0x601F3DD8, address 0x0”
Things the Router Cannot Fix

- PMPEs occur on VIPs, GSR LCs too
  - Look for either ‘System exception sig=20’ or ‘Packet Memory <Read> Parity error’
- Parity errors
  - Soft parity errors—transient
  - Hard parity errors—hardware

Things the Router Cannot Fix

- Error interrupts
  - GT64010 Errors
  - Packet SRAM errors
  - OIR error
  - Power supply failure
  - Fatal PCI error
- System restarted by “Watchdog Timeouts”
  (as opposed to SW watchdog timeouts)
Things the Router Cannot Fix

- Unexpected interrupt
  Can be HW or SW
- Stack trace has “validate_sum”

Things the Router Cannot Fix

- Out of memory should not crash, but...
  System was restarted by error—a software forced crash,
  PC 0x601C9814
  %SYS-2-MALLOCFAIL: 0emem allocation of 1490 bytes failed
  from 0x601FE6C4, pool Processor, alignment 0
  This is one of four things:
    Fragmentation problem
    Misguided process trying to over-malloc()
    Memory leak
    Legitimate use of memory
Things the Router Cannot Fix

• Somebody used memory that didn’t belong to them
  This is not always immediately detectable;
  Depending on the platform, the amount of memory protection may be limited

• Other internal consistency checks failed
  These usually show up as ‘software forced crash’, and have things like ‘abort’,
  ‘crashdump’, ‘checkheaps’, and ‘validate_memory’ in the stack
Software Tries to Do Something It Shouldn’t

• Read or write to an invalid memory location (check the memory map!)
• Read or write to a valid memory location with no response within a specific timeout
• These can be ‘bus error’, or ‘spurious access’ (older code may also report ‘SegV’)
  Spurious access == invalid read<16k
  Bus error/SegV == invalid write, or a read from an invalid hardware address

Software Tries to Do Something It Shouldn’t

• Read <16k == spurious access—log the error, don’t crash
• Read or write from other invalid addresses—bus error, crash
Software Tries to Do Something It Shouldn’t

• Be careful about “valid” and “invalid” hardware locations—the memory maps do not take into account how much memory you have on the box

  On a 2500, ‘00000000–0FFFFFF’ is 16MB of DRAM, but if I only have 8MB, then access to 0x00F43212 is invalid!

Software Tries to Do Something It Shouldn’t

• If you can’t find the map, show region is useful

• Show region may be different from memory map if virtual addresses are used
Software Tries to Do Something It Shouldn’t

• Show region
  Do not get show reg—this is different

```bash
isp-2503a#sh region
Region Manager:

<table>
<thead>
<tr>
<th>Start</th>
<th>End</th>
<th>Size(b)</th>
<th>Class</th>
<th>Media</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>0x00FFFFFF</td>
<td>16777216</td>
<td>Local</td>
<td>R/W</td>
<td>main</td>
</tr>
<tr>
<td>0x00001000</td>
<td>0x0001B69B</td>
<td>108188</td>
<td>IData</td>
<td>R/W</td>
<td>main:data</td>
</tr>
<tr>
<td>0x0001B69C</td>
<td>0x000633BB</td>
<td>294176</td>
<td>IBss</td>
<td>R/W</td>
<td>main:bss</td>
</tr>
<tr>
<td>0x000633BC</td>
<td>0x00FFEFFF</td>
<td>16366660</td>
<td>Local</td>
<td>R/W</td>
<td>main:heap</td>
</tr>
<tr>
<td>0x00FFF000</td>
<td></td>
<td>4096</td>
<td>Local</td>
<td>R/W</td>
<td>main:flhlog</td>
</tr>
<tr>
<td>0x03000000</td>
<td>0x037FFFFF</td>
<td>8388608</td>
<td>Flash</td>
<td>R/O</td>
<td>flash</td>
</tr>
<tr>
<td>0x0302471C</td>
<td>0x03417DF7</td>
<td>4142812</td>
<td>IText</td>
<td>R/O</td>
<td>flash:text</td>
</tr>
<tr>
<td>0x04000000</td>
<td>0x041FFFFF</td>
<td>2097152</td>
<td>Iomem</td>
<td>R/W</td>
<td>iomem</td>
</tr>
</tbody>
</table>
```

Things to Gather

• Show tech
  If you can’t get that, then show {ver|stack|diag|run}, at least

• For crashinfo, bootflash:crashinfo
  Copy bootflash:crashinfo tftp or show file bootflash:crashinfo

• Show log
Things to Gather

- Stack decode
  
  http://www.cisco.com/stack/stackdecoder.shtml
  http://cco.cisco.com/support/bugtools/

- History up to the point of crash
  
  Has this happened before—How often—Has anything changed on the network—Are you sure?

Things to Gather

- Core dump
  
  Exception dump <x.x.x.x>
  
  Exception protocol {ftp|tftp}
  
  Debug sanity

- Not needed for most crashes
Stack Trace: Decoding

• Stack trace vs.—Traceback
  Same thing—a stack of functions that were
  executing when we found an error—sh stack
displays stack that was saved at time of crash

  Tracebacks routinely used to report stack
trace when a message needs to be reported—
not indicative of crash in this case

• When do you decode a stack trace?
  Most useful when you have ‘software forced
crash’, ‘bus error’, or the like

  Least useful when you have a PMPE
  With a PMPE, the fault is (almost) never software!
Stack Trace: Decoding

- How do you decode a stack trace?
  
  Run the ‘RA:’ column of `sh stack` and any tracebacks through stack decoder
  
  Note: if you have a crashinfo, look towards the top for the version you were running when you crashed; that’s more accurate than `sh ver/sh stack`

- Run from flash systems
- Run from RAM systems
Slave in slot 7 was restarted by bus error at PC 0x60182A38, address 0x5F4D4F4E

RSP Software (RSP-JSV-M), Version 11.2(16), RELEASE SOFTWARE (fc1)  
Compiled Tue 06-Oct-98 12:01 by ashah  
Image text-base: 0x600108A0, data-base: 0x60AF8000

Stack trace from system failure:
FF: 0x6105CDC8, RA: 0x60182A38
FF: 0x6105CE08, RA: 0x60154DE8
FF: 0x6105CE20, RA: 0x6018BBE0
FF: 0x6105CE60, RA: 0x60160DD8
FF: 0x6105CE78, RA: 0x60173D88
FF: 0x6105CE90, RA: 0x60173D74

Stack Trace: Decoding

vi pac.crash

Slave in slot 7 was restarted by bus error at PC 0x60182A38, address 0x5F4D4F4E

RSP Software (RSP-JSV-M), Version 11.2(16), RELEASE SOFTWARE (fc1)  
Compiled Tue 06-Oct-98 12:01 by ashah  
Image text-base: 0x600108A0, data-base: 0x60AF8000

Stack trace from system failure:
FF: 0x6105CDC8, RA: 0x60182A38
FF: 0x6105CE08, RA: 0x60154DE8
FF: 0x6105CE20, RA: 0x6018BBE0
FF: 0x6105CE60, RA: 0x60160DD8
FF: 0x6105CE78, RA: 0x60173D88
FF: 0x6105CE90, RA: 0x60173D74

Stack Trace: Decoding

srsym - pac.crash

/auto/nanometer/release/112/sym/112-16/rsp-jsv-mz.112-16.symbols.Z read in
Slave in slot 7 was restarted by bus error at PC 0x60182A38
0x60182A38 [pool_prune(0x60182914)+0x124], address 0x5F4D4F4E
Image text-base: 0x600108A0[&_start(0x600108a0)+0x0], data-base: 0x60AF8000[etext(0x60af76eb)+0x915]
FF: 0x6105CDC8[etext(0x60af76eb)+0x5656dd], RA: 0x60182A38[pool_prune(0x60182914)+0x124]
FF: 0x6105CE08[etext(0x60af76eb)+0x56571d], RA: 0x60154DE8[pak_pool_periodic(0x60154dc0)+0x28]
FF: 0x6105CE20[etext(0x60af76eb)+0x565735], RA: 0x6018BBE0[registry_list(0x6018bae0)+0x100]
FF: 0x6105CE60[etext(0x60af76eb)+0x565775], RA: 0x60160DD8[net_oneminute(0x60160dac)+0x2c]
FF: 0x6105CE78[etext(0x60af76eb)+0x56578d], RA: 0x60173D88[rk_process_dispatch(0x60173d74)+0x14]
FF: 0x6105CE90[etext(0x60af76eb)+0x5657a5], RA: 0x60173D74[rk_process_dispatch(0x60173d74)+0x0]
Stack Trace: Decoding

• Note that we decoded the ‘RA’ column, not ‘FP’
• Function stack reads bottom-to-top, caller-to-callee
• Look for bugs (but which symbols?)

Stack Trace: Decoding

• What symbols do I use to search for bugs?
  Start with the first few symbols on the top of the stack
  After you do enough crashes, you’ll start to see that there are symbols you can usually ignore (more on this later!)
Stack Trace: What Does It Mean?

• A stack is a place for IOS to save operating information about routines that it’s running

• When one function calls another, the called function (callee) is placed (pushed) on to the top of the stack

• When the function at the top of the stack returns, it is removed from (popped off of) the top of the stack

Stack Trace: What Does It Mean?

• A stack is a log of how you got to the function that ended up crashing the box

• It is very useful

• It is almost always 100% accurate

  Sometimes a stack can be corrupted, but this is pretty easy to tell
Stack Trace: What Does It Mean?

- How do you know if your stack is corrupted?
  1) does the stack follow a logical path of execution—
      Hint: ip fastswitching code, followed by DLSw code
           followed by NMS code is probably bogus
  2) If the stack trace contains mainly ‘etext()’ in the RA
      column, there a good chance it is corrupt
  3) look in the code, see if the caller function is
      supposed to call the called function—This is usually
      pretty easy, but there are a few gotchas (function
      vectors, registry)

Stack Trace: Interpretation

- Cisco IOS function calls can be broken down
  into two main categories:
  
  Stuff that does stuff == “configuration-dependent”
  Stuff that supports the stuff that does stuff ==
  “IOS internals”
Stack Trace: Interpretation

• Stuff that does stuff
  IP, OSPF, DLSW, FRF.9, TEI negotiation, SVC call setup, etc…etc…
  Generally, it’s stuff you can disable on a router

Stack Trace: Interpretation

• Stuff that supports stuff that does stuff
  Timers (managed and watchdog), housecleaning stuff (checkheaps), memory management (malloc()), registry functions, software-forced crashing (abort(), crashdump())
  This stuff exists in all routers
Stack Trace: Interpretation

- Knowing how to properly interpret a stack can save you lots of time
- There are three main parts to a stack: the top few symbols, the middle few, and the bottom few

System was restarted by error - a Software forced crash, PC 0x602F27B0
7200 Software (C7200-P-M), Version 11.1(22)CC, EARLY DEPLOYMENT RELEASE SOFTWARE (fc1)
Compiled Mon 02-Nov-98 21:55 by jjgreen
Image text-base: 0x600088E0, data-base: 0x605AC000

Stack trace from system failure:    srsym output:
FP: 0x60C82F50, RA: 0x602F27B0  0x602F27B0:abort
FP: 0x60C82F50, RA: 0x602F0110  0x602F0110:crashdump
FP: 0x60C82F68, RA: 0x602883E4  0x602883E4:process_handle_watchdog
FP: 0x60C82F80, RA: 0x6027C48C  0x6027C48C:signal_receive
FP: 0x60C82FA0, RA: 0x60223BBC  0x60223BBC:process_forced_here
FP: 0x60C83000, RA: 0x6026C288  0x6026C288:mgd_timer_set_exptime_internal
FP: 0x60C83048, RA: 0x6026C7DC  0x6026C7DC:mgd_timer_set_exptime
FP: 0x60C83060, RA: 0x6026C8E0  0x6026C8E0:mgd_timer_start
Stack Trace: Interpretation

- The bottom few symbols are always IOS internals
  r4K_process_dispatch, scheduler, timers, etc...
- The middle few can be internals or config-dependent
- The top few are usually config-dependent

Stack Trace: Interpretation

- If your stack is not hosed, then the topmost symbol in your stack trace is what caused the router to crash
- If the router crashed intentionally (software-forced crash), then the topmost symbol is not at fault for the crash
Stack Trace: Interpretation

- In a ‘software-forced crash’, you’ll always see abort and crashdump; start your search with the symbols below that
- PMPEs are treated as hardware, unless you have a really really really good reason to suspect otherwise
- In other crashes, start with the top few symbols from the stack

Decoding Crashinfo

- It’s just your last X console messages, stack trace, register dump, memory blocks
- Decode all relevant tracebacks
- Process level stack trace == “show stack”
- Decode all alignment errors
Crashinfo

- What information is in Crashinfo?

  The following information is in crashinfo file to report the information at the point of crash:
  - Up to 32 KB of errmsg and command history show alignment
  - Malloc and free trace which has 128 entries
  - Process Level Stack trace
  - Process Level Context
  - Process Level Stack Dump
  - Process Level Register Memory Dump
  - The interrupt stacks (in newer images only)

Hangs

- Process stuck in infinite loop
- Process is waiting for some event
Hangs—Information to Gather

- Config-register 0x2
  Reload
- Send break
- Repeat 5–10 times
  Stack
  Context
  Cont

Hangs—Analysis

- Need access to source code
- Decode RA and attempt to determine which line of code the process is “hung”
- Scheduler allocate/scheduler interval
Crashes, Hangs and Tracebacks

Q & A

Memory
Memory Leaks

- Identify if normal memory consumption or is a memory leak
- Process does not free memory after use
- Processor pool memory leaks
- IO pool memory leaks
- Processor/IO memory leaks can also occur on VIP2s, VIP4s, GSR LCs, etc

Memory Leaks—Show Process Memory

```
esc-7204-VXR#show proc mem
Total: 121284864, Used: 13698680, Free: 107586184

PID TTY   Allocated Freed Holding Getbufs Retbufs Process
0   0     99908       1852 10543080 0 0 *Init*
```

- Total—memory left after IOS init
- Used—memory currently used
- Free—memory free
- PID—process ID
- TTY—exec line
Memory Leaks—Show Process Memory

- Allocated—memory allocated by process
- Freed—memory freed by process
- Holding—memory currently held by process
- Getbufs—packet buffers requested by process
- Retbufs—packet buffers returned by process
- Process—process name

Memory Leaks—Show Memory Summary

```
esc-7204-VXR#show mem summ

Head    Total(b)     Used(b)     Free(b)   Lowest(b)  Largest(b)
Processor   62455700   121284864    13698900   107585964   107351772

Processor memory
Alloc PC        Size     Blocks      Bytes    What
0x60009EAC        172          1        172    Init
```

- Head—address of start of memory
- Lowest—low water mark of free memory
- Largest—largest contiguous free memory block
Memory Leaks—Show Memory Summary

- Alloc PC—function call that allocated memory
- Size—number of bytes in each block
- Blocks—number of blocks allocated
- Bytes—total memory used by process
- What—name of process

Memory Leaks

May 19 16:16:49 EST: %SYS-2-MALLOCFAIL: Memory allocation of 4692 bytes failed from 0x60499270, pool I/O, alignment 32
-Process= "IP SNMP", ipl= 4, pid= 75
-Traceback= 604D8668 604D9AF8 60499278 60499638 60499B90 60579CE8 605CE49C

- Error message useful to figure out location of memory leak
- Process reporting mallocfail is the one that detected low memory condition, not necessarily process that caused leak
Memory Leaks—Information to Gather

• Usually too late by the time a “mallocfail” occurs
• Different information to gather for processor pool memory leaks and IO pool memory leaks
• Always include one copy of show tech

Memory Leaks—Processor Pool Memory Leak

• Capture periodically (after a reboot)
  Term length 0
  Show clock
  Show process memory
  Show memory summary
• Since output can be extensive, capture from a telnet session (not console)
Memory Leaks—IO Pool Memory Leak

- Capture periodically (after a reboot)
  - Term length 0
  - Show clock
  - Show memory summary
  - Show buffers
  - Show buffers old
  - Show buffers old dump
  - Show buffers assigned
  - Show buffers assigned dump
  - Show buffers pool <pool name>
  - Show buffers pool <pool name> dump

- Capture from a telnet session

---

Show Memory Processor

---

Lists Elements
Interrupt Stack
TTY data
messages
Basic Memory Unit: blocktype_

- Allocated and in Use
- Allocated and on Freelist
  - Process Stacks
  - Pool Items
  - Chunks
  - SWIDBS, HWIBS, IDBS
  - QUEUES
  - Routing Tables, Switching Tables
  - All DYNAMIC Structures

Blockmagic
Allocate_pid
Allocate_Check
*Allocate_Name
Allocate_pc
*Next
*Previous
Size
Refcount
**Last_Last_Allocator
Data_Area

Freeblocktype Free_info[0]

Uchar Data[0]

Example of Memblock Breakdown

Router#sh mem  Dx:0e517c0
0e517ed: AB1234CD FFFFFFFE 00000000 00E33C14
0e517f0: 06233844 60E51824 60E5179C 0000000E
0e51800: 00000001 00C0000D S2656720 46756E59
0e51810: 74E56F5E 20300000 60D51004 60D51000
0e51820: FDC110DF ABC123CD FFFFFFFF 00000000
0e51830: 06233C14 06233644 60E51858 60E517F4
0e51840: 00C00000 00000001 00C0000D S2656720
0e51850: 46756E59 74E56F5E 20313230 00D51004
0e51860: 60D51000 FDC110DF AB1234CD FFFFFFFE
0e51870: 00000000 60A002C4 601E8680 60E518AC
0e51880: 60E51838 00000000 00000001 00C0000D
0e51890: 00000000 00C00001 60E5A859 00000001
0e518A0: 6055100D 60551000 FDC110DF AB1234CD
0e518B0: FFFFFFFE 00000000 60A054AC 60233044
0e518C0: 60E518F0 60E5187C 00000000 00000001
0e518D0: 061E5838 50617273 6520455F 64E57300
0e518E0: 00C00000 60E5185A 60D51020 FDC110DF

Redzone Located at: Data Area Location + Length of Data
The Model Holds in Physical Memory

Physical Memory

Sequential Creation of Blocks

Lower Memory

Higher Memory

Example Illustrating Sequential Memblocks

```
Router# mem Dx 60E517cD
60E517ED: AB1234CD FFFFFFE 00000000 60E517C4
60E517FD: 60233B44 60E51824 60E5179C 8000000E
60E51800: 00000001 0000000D 52656720 46756E63
60E5181D: 74696F6E 20380000 60D510D4 60D510D0
60E5182D: FDC110DF AB1234CD FFFFFFE 00000000
60E5183D: 60E5180E 60233844 60E51866 60E517F4
60E5184D: 0000000E 00000001 0000000D 52656720
60E5185D: 46756E63 74696F6E 20313200 60D510D4
60E5186D: 60D510D0 FDC110DF AB1234CD FFFFFFE
60E5187D: 00000000 60AFD2C4 601E6800 60E518AC
60E5188D: 60E51838 8000000E 00000001 0000000D
60E5189D: 00000000 00000101 60E5A8EE 00000001
60E518AD: 60D510D4 60D51000 FDC110DF AB1234CD
60E518BD: FFFFFFE 00000000 60AF84AC 60233844
60E518CD: 60E5180D 60E5187C 0000000E 00000001
60E518DD: 601E8538 50617273 6520424F 64657370
60E518ED: 0000000D 60E5115A 60D56F2C FDC110DF
```

MemBlockHeader

Data
FreeBlockType Header

Router#sh mem 0x60000000
5E23FC0: 0xBE000000 00000000 00000000 00000000
5EC2ED6: 0x00000000 00000000 00000000 00000000
5EE2EED6: 0x00000000 00000000 00000000 00000000
MagicBlock
Size and InUse
FreeMagic
FreeMagic
FreeMagic
FreeMagic
RedZone

Memory Need for a Small Message, 40 bytes

Router#sh mem 0x60000000
5E23FC0: 0xBE000000 00000000 00000000 00000000
5EC2ED6: 0x00000000 00000000 00000000 00000000
5EE2EED6: 0x00000000 00000000 00000000 00000000
MagicBlock
Size and InUse
FreeMagic
FreeMagic
FreeMagic
FreeMagic
RedZone

Set InUse Flag in the MEMBLOCK Size Byte
## Initialization of the Data Area

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blockmagic</td>
<td></td>
</tr>
<tr>
<td>Allocator_pid</td>
<td>Pid</td>
</tr>
<tr>
<td>Allocator_Check</td>
<td>Check</td>
</tr>
<tr>
<td>Allocator_Name</td>
<td>Name</td>
</tr>
<tr>
<td>*Allocator_Name</td>
<td>pc</td>
</tr>
<tr>
<td>*Next</td>
<td></td>
</tr>
<tr>
<td>*Previous</td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>00000018</td>
</tr>
<tr>
<td>Refcount</td>
<td>00000000</td>
</tr>
<tr>
<td><strong>Last_Last_Allocator</strong></td>
<td></td>
</tr>
<tr>
<td>Data_Area.data</td>
<td>0D0D0D0D</td>
</tr>
</tbody>
</table>

**CLEAR InUSE FLAG Set, MSB in SIZE**

**REDZONE**

## Major Steps in the Free Operation

- Clear the InUse flag
- Poison the data region with the poison value
  
  **POISON -> 0x0D0D0D0D**
  
  Note: This will overwrite REDZONE
- Attempt to merge with previous block
- Attempt to merge with next block
- Add block to freelist
Poison of the Data Area

Questions & Answers
Router Architecture and Cisco IOS Internal
Session RST-301

Please Complete Your Evaluation Form
Session RST-301