Cisco Nexus 7000 Hardware Architecture

BRKAR C-3470
Session Goal

- To provide you with a thorough understanding of the Cisco Nexus™ 7000 switching architecture, supervisor, fabric, and I/O module design, packet flows, and key forwarding engine functions.

- This session will **not** examine Unified I/O, DCB, FCoE, NX-OS software architecture, or other Nexus platforms.

- Related sessions:
  
  BRKARC-3471: Cisco NXOS Software Architecture
Agenda

- Chassis Architecture
  - Supervisor Engine Architecture
  - I/O Module Architecture
  - Forwarding Engine Architecture
  - Fabric Architecture
- Layer 2 Forwarding
- IP Forwarding
- IP Multicast Forwarding
- ACLs
- QoS
- NetFlow
Nexus 7010 Chassis

- Optional locking front doors
- System status LEDs
- ID LEDs on all FRUs
- Front-to-back airflow
- Integrated cable management with cover
- Air exhaust
- Locking ejector levers
- System fan trays
- Fabric fan trays
- Supervisor slots (5-6)
- Crossbar fabric modules
- Power supplies
- I/O module slots (1-4, 7-10)
- Two chassis per 7’ rack
- Air intake with optional filter
- Common equipment removes from rear
- ID LEDs on all FRUs
- Front-to-back airflow
Nexus 7018 Chassis

- Integrated cable management
- Supervisor slots (9-10)
- Power supply air intake
- Power supplies
- Common equipment removes from rear
- Crossbar fabric modules
- Optional front door
- Side-to-side airflow
- System status LEDs
- System fan trays

Supported in NX-OS release 4.1(2) and later
Agenda

- Chassis Architecture
- Supervisor Engine Architecture
- I/O Module Architecture
- Forwarding Engine Architecture
- Fabric Architecture
- Layer 2 Forwarding
- IP Forwarding
- IP Multicast Forwarding
- ACLs
- QoS
- NetFlow
Supervisor Engine

- Performs control plane and management functions
- Dual-core 1.66GHz Intel Xeon processor with 4GB DRAM
- 2MB NVRAM, 2GB internal bootdisk, compact flash slots
- Out-of-band 10/100/1000 management interface
- Always-on Connectivity Management Processor (CMP) for lights-out management
- Console and auxiliary serial ports
- USB ports for file transfer
Management Interfaces

Management Ethernet

- 10/100/1000 interface used exclusively for system management
- Belongs to dedicated “management” VRF
  - Prevents data plane traffic from entering/exiting from mgmt0 interface
  - Cannot move mgmt0 interface to another VRF
  - Cannot assign other system ports to management VRF

Connectivity Management Processor (CMP) Ethernet

- Connects to standalone, always-on microprocessor on supervisor engine
  - Runs lightweight software with network stack
  - Completely independent of NX-OS on main CPU
- Provides ‘lights out’ remote management and disaster recovery via 10/100/1000 interface
  - Removes need for terminal servers
Supervisor Engine Architecture
Agenda

- Chassis Architecture
- Supervisor Engine Architecture
- I/O Module Architecture
- Forwarding Engine Architecture
- Fabric Architecture
- Layer 2 Forwarding
- IP Forwarding
- IP Multicast Forwarding
- ACLs
- QoS
- NetFlow
8-Port 10GE I/O Module
N7K-M108X2-12L

- 8-port 10G with X2 transceivers
- 80G full-duplex fabric connectivity
- Two integrated forwarding engines (120Mpps)
  Support for “XL” forwarding tables (licensed feature)
- 8 ports wire-rate L3 multicast replication
- 802.1AE LinkSec

N7K-M108X2-12L
8-Port 10G XL I/O Module Architecture
N7K-M108X2-12L

EOBC

Fabric ASIC

To Fabric Modules

To Central Arbiter

Front Panel Ports

10G MAC

Linksec

1

2

3

4

5

6

7

8

Forwarding Engine

Replication Engine

VOQs

CPU

LC
32-Port 10GE I/O Module
N7K-M132XP-12

- 32-port 10G with SFP+ transceivers
- 80G full-duplex fabric connectivity
- Integrated 60Mpps forwarding engine
- Oversubscription option for higher density (up to 4:1)
- 8 ports wire-rate L3 multicast replication
- 802.1AE LinkSec
**Shared vs. Dedicated Mode**

**Dedicated mode**
- First interface in port group gets 10G bandwidth
- Other three interfaces in port group disabled

**Shared mode**
- Four interfaces in port group share 10G bandwidth

"Port group"- group of contiguous even or odd ports that share 10G of bandwidth (e.g., ports 1,3,5,7)

**rate-mode shared** (default)

**rate-mode dedicated**
32-Port 10G I/O Module Architecture
N7K-M132XP-12
48-Port 1G I/O Modules
N7K-M148GT-11, N7K-M148GS-11, N7K-M148GS-11L

- Three 1G I/O module options:
  - 48 10/100/1000 RJ-45 ports (N7K-M148GT-11)
  - 48 1G SFP ports (N7K-M148GS-11)
  - 48 1G SFP ports with XL forwarding engine (N7K-M148GS-11L)

- Integrated 60Mpps forwarding engine

- 46G full duplex fabric connectivity
  - Line rate on 48-ports with some local switching

- 48 ports wire-rate L3 multicast replication

- 802.1AE LinkSec
48-Port 1G I/O Modules Architecture
N7K-M148GT-11, N7K-M148GS-11, N7K-M148GS-11L

Front Panel Ports:
- 1-12
- 13-24
- 25-36
- 37-48

- Fabric ASIC
- VOQs
- Forwarding Engine
- Replication Engine
- Linksec
- EOBC
- LC CPU

To Fabric Modules
To Central Arbiter
Agenda

- Chassis Architecture
- Supervisor Engine Architecture
- I/O Module Architecture
- **Forwarding Engine Architecture**
- Fabric Architecture
- Layer 2 Forwarding
- IP Forwarding
- IP Multicast Forwarding
- ACLs
- QoS
- NetFlow
Forwarding Engine Hardware

- Hardware forwarding engine(s) integrated on every I/O module
- 60Mpps per forwarding engine
  - Layer 2 bridging with hardware MAC learning
- 60Mpps per forwarding engine IPv4 and 30Mpps IPv6 unicast
- IPv4 and IPv6 multicast support (SM, SSM, bidir)
- RACL/VACL/PACLs
- Policy-based routing (PBR)
- Unicast RPF check and IP source guard
- QoS remarking and policing policies
- Ingress and egress NetFlow (full and sampled)

<table>
<thead>
<tr>
<th>Hardware Table</th>
<th>M1 Modules</th>
<th>M1-XL Modules without License</th>
<th>M1-XL Modules with License</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIB TCAM</td>
<td>128K</td>
<td>128K</td>
<td>900K</td>
</tr>
<tr>
<td>Classification TCAM (ACL/QoS)</td>
<td>64K</td>
<td>64K</td>
<td>128K</td>
</tr>
<tr>
<td>MAC Address Table</td>
<td>128K</td>
<td>128K</td>
<td>128K</td>
</tr>
<tr>
<td>NetFlow Table</td>
<td>512K</td>
<td>512K</td>
<td>512K</td>
</tr>
</tbody>
</table>
“Scalable Services” License

- Forwarding engines on M1-XL I/O modules always have “XL” capacity
- Access to additional capacity controlled by presence of “Scaleable Services” license
  License applies to entire system (per-chassis)

N7K# show license usage

<table>
<thead>
<tr>
<th>Feature</th>
<th>Ins</th>
<th>Lic</th>
<th>Status</th>
<th>Expiry Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCALABLE_SERVICES_PKG</td>
<td>Yes</td>
<td>-</td>
<td>In use</td>
<td>Never</td>
<td>-</td>
</tr>
<tr>
<td>LAN_ADVANCED_SERVICES_PKG</td>
<td>Yes</td>
<td>-</td>
<td>In use</td>
<td>Never</td>
<td>-</td>
</tr>
<tr>
<td>LAN_ENTERPRISE_SERVICES_PKG</td>
<td>Yes</td>
<td>-</td>
<td>In use</td>
<td>Never</td>
<td>-</td>
</tr>
</tbody>
</table>

N7K#
Forwarding Engine Architecture

Forwarding engine chipset consists of two ASICs:

- **Layer 2 Engine**
  - Ingress and egress SMAC/DMAC lookups
  - Hardware MAC learning
  - IGMP snooping and IP-based Layer 2 multicast constraint

- **Layer 3 Engine**
  - IPv4/IPv6 Layer 3 lookups
  - ACL, QoS, NetFlow and other processing
  - Linear, pipelined architecture—every packet subjected to both ingress and egress pipeline
  - Enabling features does not affect forwarding engine performance
Forwarding Engine Pipelined Architecture

- Ingress NetFlow collection
- Ingress ACL and QoS classification lookups
- Unicast RPF check
- Ingress MAC table lookups
- IGMP snooping lookups
- IGMP snooping redirection

Layer 2 Engine
- Egress MAC lookup
- IGMP snooping lookups

Layer 3 Engine
- Egress policing
- FIB TCAM and adjacency table lookups for Layer 3 forwarding
- ECMP hashing
- Multicast RPF check

- Ingress policing

Packet Headers from I/O Module Replication Engine
- Egress pipeline

Final lookup result to I/O Module Replication Engine
- Ingress pipeline
- Egress pipeline

Presentation_ID 22
© 2010 Cisco and/or its affiliates. All rights reserved.
Agenda

- Chassis Architecture
- Supervisor Engine Architecture
- I/O Module Architecture
- Forwarding Engine Architecture
- Fabric Architecture
- Layer 2 Forwarding
- IP Forwarding
- IP Multicast Forwarding
- ACLs
- QoS
- NetFlow
Crossbar Switch Fabric Module

- Each fabric module provides 46Gbps per I/O module slot
  - Up to 230Gbps per slot with 5 fabric modules
- Currently shipping I/O modules do not leverage full fabric bandwidth
  - Maximum 80G per slot with 10G module
  - Future modules leverage additional available fabric bandwidth
- Access to fabric controlled using QoS-aware central arbitration with VOQ

N7K-C7010-FAB-1

N7K-C7018-FAB-1
Fabric Module Capacity

230Gbps
per slot bandwidth

Nexus 7018
I/O Module Capacity

1G modules
- Require 1 fabric for full bandwidth
- Require 2 fabrics for N+1 redundancy

230Gbps per slot bandwidth
- 4th and 5th fabric modules provide additional redundancy and future-proofing

10G modules
- Require 2 fabrics for full bandwidth
- Require 3 fabrics for N+1 redundancy
Access to Fabric Bandwidth

- Access to fabric controlled using central arbitration
  
  Arbiter ASIC on supervisor engine provides fabric arbitration

- Bandwidth capacity on egress modules represented by Virtual Output Queues (VOQs) at ingress to fabric
  
  I/O modules interface with arbiter to gain access to VOQs
What Are VOQs?

- Virtual Output Queues (VOQs) on ingress modules represent bandwidth capacity on egress modules.
- If VOQ available on ingress to fabric, capacity exists at egress module to receive traffic from fabric.
  - Central arbiter determines whether VOQ is available for a given packet.
  - Bandwidth capacity represented by “credits.”
  - Credits are requested by I/O modules and granted by arbiter.
- VOQ is “virtual” because it represents EGRESS capacity but resides on INGRESS modules.
  - It is still PHYSICAL buffer where packets are stored.
- Note: VOQ is not equivalent to ingress or egress port buffer or queues.
  - Relates ONLY to ASICs at ingress and egress to fabric.
Benefits of Central Arbitration with VOQ

- Ensures priority traffic takes precedence over best-effort traffic across fabric
  - Four levels of priority for each VOQ destination

- Ensures fair access to bandwidth for multiple ingress ports transmitting to one egress port
  - Central arbiter ensures all traffic sources get appropriate access to fabric bandwidth, even with traffic sources on different modules

- Prevents congested egress ports from blocking ingress traffic destined to other ports
  - Mitigates head-of-line blocking by providing independent queues for individual destinations across the fabric

- In future, will provide lossless service for FCoE traffic across the fabric
  - Can provide strict priority and backpressure (blocking instead of dropping) for certain traffic classes, such as SAN traffic
Agenda

- Chassis Architecture
- Supervisor Engine Architecture
- I/O Module Architecture
- Forwarding Engine Architecture
- Fabric Architecture
- Layer 2 Forwarding
- IP Forwarding
- IP Multicast Forwarding
- ACLs
- QoS
- NetFlow
Layer 2 Forwarding

- MAC table is 128K entries (115K effective)
- Hardware MAC learning
  CPU not directly involved in learning
- All modules have copy of MAC table
  New learns communicated to other modules via hardware “flood to fabric” mechanism
  Software process ensures continuous MAC table sync
- Spanning tree (PVRST or MST) or Virtual Port Channel (VPC) ensures loop-free Layer 2 topology
Layer 2 Forwarding Architecture

- Layer 2 Forwarding Manager (L2FM) maintains central database of MAC tables
- L2FM keeps MAC table on all forwarding engines in sync
- L2FM-Client process on I/O modules interfaces between L2FM and hardware MAC table

```
n7010# sh processes cpu | egrep PID|l2fm
PID    Runtime(ms)  Invoked   uSecs  1Sec   Process
 3848     1106    743970580     0      0  l2fm
n7010# attach mod 9
Attaching to module 9 ...  
To exit type 'exit', to abort type '$.'
Last login: Mon Apr 21 15:58:12 2009 from sup02 on pts/0
Linux lc9 2.6.10 mv1401-pc_target #1 Fri Mar 21
23:26:28 PDT 2009 ppc GNU/Linux
module-9# sh processes cpu | egrep l2fm
  1544       6396    388173    16   0.0  l2fmc
module-9#
```
Hardware Layer 2 Forwarding Process

- MAC table lookup in Layer 2 Engine based on \{VLAN,MAC\} pairs
- Source MAC and destination MAC lookups performed for each frame
  - Source MAC lookup drives new learns and refreshes aging timers
  - Destination MAC lookup dictates outgoing switchport
L2 Packet Flow

1. Receive packet from wire
2. LinkSec decryption
3. 1st stage ingress port QoS
4. Submit packet headers for lookup
5. L2 SMAC/DMAC lookups
6. ACL/QoS/NetFlow lookups
7. Return result
8. VOQ arbitration and queuing
9. Credit grant for fabric access
10. Transmit to fabric
11. Return credit to pool
12. Receive from fabric
13. Return buffer credit
14. Submit packet headers for egress L2 lookup
15. Egress port QoS
16. LinkSec encryption
17. Transmit packet on wire

© 2010 Cisco and/or its affiliates. All rights reserved.
Agenda

- Chassis Architecture
- Supervisor Engine Architecture
- I/O Module Architecture
- Forwarding Engine Architecture
- Fabric Architecture
- Layer 2 Forwarding
- **IP Forwarding**
- IP Multicast Forwarding
- ACLs
- QoS
- NetFlow
IP Forwarding

- Nexus 7000 decouples control plane and data plane
- Forwarding tables built on control plane using routing protocols or static configuration
  - OSPF, EIGRP, IS-IS, RIP, BGP for dynamic routing
- Tables downloaded to forwarding engine hardware for data plane forwarding
IP Forwarding Architecture

- Routing protocol processes learn routing information from neighbors
- IPv4 and IPv6 unicast RIBs calculate routing/next-hop information
- Unicast Forwarding Distribution Manager (UFDM) interfaces between URIBs on supervisor and IP FIB on I/O modules
- IP FIB process programs forwarding engine hardware on I/O modules
  - FIB TCAM contains IP prefixes
  - Adjacency table contains next-hop information

```
module-9# sh processes cpu | egrep fib
PID   Runtime(ms)  Invoked  uSecs  lSec  Process
1534   80042       330725   242   0.0  ipfib
```

```
module-9# sh processes cpu | egrep ospf
PID   Runtime(ms)  Invoked  uSecs  lSec  Process
20944 93           33386880  0      0  ospf
```

```
module-9# sh processes cpu | egrep u.*rib
3573  117         44722390  0      0  u6rib
3574  150         34200830  0      0  urib
```

```
module-9# sh processes cpu | egrep ufdm
3836 1272        743933460  0      0  ufdm
```
Hardware IP Forwarding Process

- FIB TCAM lookup based on destination prefix (longest-match)
- FIB “hit” returns adjacency, adjacency contains rewrite information (next-hop)
- Pipelined forwarding engine architecture also performs ACL, QoS, and NetFlow lookups, affecting final forwarding result
IPv4 FIB TCAM Lookup

1. Generate TCAM lookup key (destination IP address)

2. Compare lookup key

   10.1.1.10

   - 10.1.1.2
   - 10.1.1.3
   - 10.1.1.4
   - 10.10.0.10
   - 10.10.0.100
   - 10.10.0.33
   - 10.1.1.xx
   - 10.100.1.xx

3. Load-Sharing Hash

   - Index, # next-hops
   - Index, # next-hops
   - Index, # next-hops
   - Index, # next-hops
   - Index, # next-hops
   - Index, # next-hops
   - Index, # next-hops
   - Index, # next-hops

4. Offset

   - Adj Index
   - Next-hop 1 (IF, MAC)
   - Next-hop 2 (IF, MAC)
   - Next-hop 3 (IF, MAC)
   - Next-hop 4 (IF, MAC)
   - Next-hop 5 (IF, MAC)
   - Next-hop 6 (IF, MAC)
   - Next-hop 7 (IF, MAC)

5. Return lookup result

   - Result

Hit in FIB returns result in FIB DRAM

Adjacency index identifies ADJ block

Hash selects exact next hop entry

Ingress unicast IPv4 packet header

Generate TCAM lookup key

Flow Data

Adjacency Table

FIB TCAM

FIB DRAM

Flow

Data
“Routing” vs. “Forwarding”

- “Routing” information refers to unicast RIB contents in supervisor control plane
- “Forwarding” information refers to FIB contents at I/O module
Displaying Routing and Forwarding Information

- **show routing [ipv4|ipv6] [<prefix>] [vrf <vrf>]**
  
  Displays software routing (URIB) information
  
  Can also use traditional `show ip route` command

- **show forwarding [ipv4|ipv6] route module <mod> [vrf <vrf>]**
  
  Displays hardware forwarding (FIB) information on per-module basis

- **show forwarding adjacency module <mod>**
  
  Displays hardware adjacency table information on per-module basis
n7010# sh routing ipv4 10.100.7.0/24

IP Route Table for VRF "default"

10.100.7.0/24, 1 ucast next-hops, 0 mcast next-hops

  *via 10.1.2.2, Ethernet9/2, [110/5], 00:02:30, ospf-1, type-1

n7010# show forwarding ipv4 route 10.100.7.0/24 module 9

IPv4 routes for table default/base

+------------------+------------------+---------------------+
| Prefix            | Next-hop         | Interface            |
+------------------+------------------+---------------------+
| 10.100.7.0/24     | 10.1.2.2          | Ethernet9/2          |
+------------------+------------------+---------------------+

n7010# show forwarding adjacency 10.1.2.2 module 9

IPv4 adjacency information, adjacency count 1

+----------------+----------------+----------------+
| next-hop       | rewrite info   | interface      |
+----------------+----------------+----------------+
ECMP Load Sharing

- Up to 16 hardware load-sharing paths per prefix
- Use `maximum-paths` command in routing protocols to control number of load-sharing paths
- Load-sharing is per-IP flow or per-packet
  - Use caution with per-packet load-balancing!
- Configure load-sharing hash options with global `ip load-sharing` command:
  - Source and Destination IP addresses
  - Source and Destination IP addresses plus L4 ports (default)
  - Destination IP address and L4 port
- Additional randomized number added to hash prevents polarization
  - Automatically generated or user configurable value
- Configure per-packet load-sharing with interface `ip load-sharing per-packet` command
  - Ingress interface determines if load-sharing is per-flow or per-packet!
ECMP Prefix Entry Example

n7010# sh routing ipv4 10.200.0.0
IP Route Table for VRF "default"
10.200.0.0/16, 2 ucast next-hops, 0 mcast next-hops
  *via 10.1.1.2, Ethernet9/1, [110/5], 00:03:33, ospf-1, inter
  *via 10.1.2.2, Ethernet9/2, [110/5], 00:00:13, ospf-1, inter
n7010# sh forwarding ipv4 route 10.200.0.0 module 9
IPv4 routes for table default/base

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Next-hop</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.200.0.0/16</td>
<td>10.1.1.2</td>
<td>Ethernet9/1</td>
</tr>
<tr>
<td></td>
<td>10.1.2.2</td>
<td>Ethernet9/2</td>
</tr>
</tbody>
</table>

n7010#
Identifying the ECMP Path for a Flow

show routing [ipv4|ipv6] hash <sip> <dip> [<sport> <dport>] [vrf <vrf>]

n7010# sh routing hash 192.168.44.12 10.200.71.188

Load-share parameters used for software forwarding:

load-share type: 1

Randomizing seed (network order): 0xebae8b9a

Hash for VRF "default"

Hashing to path *10.1.2.2 (hash: 0x29), for route:

10.200.0.0/16, 2 ucast next-hops, 0 mcast next-hops

  *via 10.1.1.2, Ethernet9/1, [110/5], 00:14:18, ospf-1, inter

  *via 10.1.2.2, Ethernet9/2, [110/5], 00:10:58, ospf-1, inter

n7010#

Same hash algorithm applies to both hardware and software forwarding
L3 Packet Flow

Receive packet from wire

1. 4:1 Mux + Linksec

2. LinkSec decryption
   - 1st stage ingress port QoS

3. 10G MAC

4. Submit packet headers for lookup

5. L2 ingress and egress SMAC/DMAC lookups

6. L3 FIB/ADJ lookup
   - Ingress and egress ACL/QoS/NetFlow lookups

Fabric Module 1

- Fabric ASIC
- VOQs
- Replication Engine
- Forwarding Engine
- Layer 2 Engine
- Layer 3 Engine

Fabric Module 2

- Fabric ASIC
- VOQs
- Replication Engine
- Forwarding Engine
- Layer 2 Engine
- Layer 3 Engine

Fabric Module 3

- Fabric ASIC
- VOQs
- Replication Engine

Receive from fabric

7. Return result

8. VOQ arbitration and queuing

9. Credit grant for fabric access

10. Transmit to fabric

11. Receive from fabric
   - Return buffer credit

12. Return credit to pool

13. Submit packet headers for egress L2 lookup

14. L2-only SMAC/DMAC lookup

15. Egress port QoS

16. LinkSec encryption

17. Transmit packet on wire

Supervisor Engine

Central Arbiter

HDR = Packet Headers
DATA = Packet Data
CTRL = Internal Signaling

Receive packet from wire
Agenda

- Chassis Architecture
- Supervisor Engine Architecture
- I/O Module Architecture
- Forwarding Engine Architecture
- Fabric Architecture
- Layer 2 Forwarding
- IP Forwarding
- IP Multicast Forwarding
- ACLs
- QoS
- NetFlow
IP Multicast Forwarding

- Forwarding tables built on control plane using multicast protocols
  - PIM-SM, PIM-SSM, PIM-Bidir, IGMP, MLD

- Tables downloaded to:
  - Forwarding engine hardware for data plane forwarding
  - Replication engines for data plane packet replication
### IP Multicast Forwarding Architecture

- Multicast routing processes learn routing information from neighbors/hosts
- IPv4 and IPv6 multicast RIBs calculate multicast routing/RP/RPF/OIL information
- Multicast Forwarding Distribution Manager (MFDM) interfaces between MRIBs on supervisor and IP FIB on I/O modules
- IP FIB process programs hardware:
  - FIB TCAM
  - Adjacency table
  - Multicast Expansion Table (MET)

```
n7010# sh processes cpu | egrep pim|igmp|PID
PID    Runtime(ms)  Invoked   uSecs  1Sec   Process
3842    109    32911620      0      0  pim
3850    133    33279940      0      0  igmp

n7010# sh processes cpu | egrep m.?rib
PID    Runtime(ms)  Invoked   uSecs  1Sec   Process
3843    177    33436550      0      0  mrib
3847    115    47169180      0      0  m6rib

n7010# sh processes cpu | egrep mfdm
PID    Runtime(ms)  Invoked   uSecs  1Sec   Process
3846    2442    743581240      0      0  mfdm

module-9# sh processes cpu | egrep fib
PID    Runtime(ms)  Invoked   uSecs  1Sec   Process
1534    80153    330725    242    0.0  ipfib
```
Hardware Programming

IP FIB process on I/O modules programs hardware:

- **FIB TCAM**
  Part of Layer 3 Engine ASIC on forwarding engine
  Consists of (S,G) and (*,G) entries as well as RPF interface

- **Adjacency Table (ADJ)**
  Part of Layer 3 Engine ASIC on forwarding engine
  Contains MET indexes

- **Multicast Expansion Table (MET)**
  Part of replication engine ASIC on I/O modules
  Contains output interface lists (OILs), i.e., lists of interfaces requiring replication
**Multicast FIB TCAM Lookup**

- **Ingress multicast packet header**
- **Generate Lookup Key**
  - 10.1.1.10, 239.1.1.1

- **Compare lookup key**

- **FIB TCAM**
  - 10.1.1.12, 239.1.1.1
  - 10.1.1.10, 232.1.2.3
  - 10.4.7.10, 225.8.8.8
  - 10.1.1.10, 239.1.1.1
  - 10.6.6.10, 239.44.2.1

- **Generate TCAM lookup key (source and group IP address)**

- **FIB DRAM**
  - RPF, ADJ Index

- **Adj Index**
  - RPF, ADJ Index
  - RPF, ADJ Index

- **Forwarding Engine**
  - MET Index
  - MET Index
  - MET Index
  - MET Index

- **Replication Engine**
  - OIFs
  - OIFs
  - OIFs
  - MET

- **Replicate**

- **OIFs**

- **Adjacency Table**

- **Result**

- **Return lookup result**

- **Hit in FIB returns result in FIB DRAM**

- **Identifies multicast adjacency entry**

- **Replication for each OIF in MET block**

- **MET index used to find OIFs for replication**

- **HIT!**
Displaying Multicast Routing and Forwarding Information

- `show routing [ipv4|ipv6] multicast [vrf <vrf>] [<source-ip>] [<group-ip>] [summary]`
  
  Displays software multicast routing (MRIB) information
  Can also use traditional `show ip mroute` command

- `show forwarding [ipv4|ipv6] multicast route [source <ip>] [group <ip>] [vrf <vrf>] module <mod>`
  
  Displays hardware multicast forwarding (FIB) information on per-module basis
Displaying Multicast Routing and Forwarding Information (Cont)

n7010# sh routing multicast 10.1.1.2 239.1.1.1
IP Multicast Routing Table for VRF "default"

(10.1.1.2/32, 239.1.1.1/32), uptime: 00:40:31, ip mrib pim
   Incoming interface: Ethernet9/1, RPF nbr: 10.1.1.2, internal
   Outgoing interface list: (count: 2)
      Ethernet9/17, uptime: 00:05:57, mrib
      Ethernet9/2, uptime: 00:06:12, mrib

n7010# sh routing multicast 239.1.1.1 summary
IP Multicast Routing Table for VRF "default"

Total number of routes: 202
Total number of (*,G) routes: 1
Total number of (S,G) routes: 200
Total number of (*,G-prefix) routes: 1
Group count: 1, average sources per group: 200.0

Group: 239.1.1.1/32, Source count: 200

<table>
<thead>
<tr>
<th>Source</th>
<th>packets</th>
<th>bytes</th>
<th>aps</th>
<th>pps</th>
<th>bit-rate</th>
<th>oifs</th>
</tr>
</thead>
<tbody>
<tr>
<td>(*,G)</td>
<td>767</td>
<td>84370</td>
<td>110</td>
<td>0</td>
<td>0 bps</td>
<td>2</td>
</tr>
<tr>
<td>10.1.1.2</td>
<td>9917158</td>
<td>1269395810</td>
<td>127</td>
<td>4227</td>
<td>4 mbps</td>
<td>2</td>
</tr>
<tr>
<td>10.1.1.3</td>
<td>9917143</td>
<td>1269393890</td>
<td>127</td>
<td>4227</td>
<td>4 mbps</td>
<td>2</td>
</tr>
<tr>
<td>10.1.1.4</td>
<td>9917127</td>
<td>1269391824</td>
<td>127</td>
<td>4227</td>
<td>4 mbps</td>
<td>2</td>
</tr>
</tbody>
</table>

<...>
n7010# sh forwarding ipv4 multicast route group 239.1.1.1 source 10.1.1.2 module 9

(10.1.1.2/32, 239.1.1.1/32), RPF Interface: Ethernet9/1, flags:

  Received Packets: 10677845 Bytes: 1366764160

  Number of Outgoing Interfaces: 2

  Outgoing Interface List Index: 15

    Ethernet9/2 Outgoing Packets: 432490865 Bytes: 55358830720

    Ethernet9/17 Outgoing Packets: 419538767 Bytes: 53700962176

n7010#
Egress Replication

- Distributes multicast replication load among replication engines of all I/O modules with OIFs
- Input packets get lookup on ingress forwarding engine
- For OIFs on ingress module, ingress replication engine performs the replication
- For OIFs on other modules, ingress replication engine replicates a single copy of packet over fabric to those egress modules
- Each egress forwarding engine performs lookup to drive replication
- Replication engine on egress module performs replication for local OIFs
L3 Multicast Packet Flow

1. Receive packet from wire
2. LinkSec decryption
3. Transmit packet on wire
4. 2nd stage ingress port QoS
5. L2 ingress snooping lookup
6. L3 multicast FIB lookup
7. Return MET result
8. Replicate for fabric delivery
9. Submit packet headers for lookup
10. VOQ queuing
11. Transmit to fabric
12. Dequeue multicast distribution copy from fabric
13. Replicate for local OIF delivery
14. Submit packet headers for egress lookups
15. L3 multicast FIB lookup
16. L2 egress snooping lookup
17. Egress port QoS
18. LinkSec encryption
19. Transmit packet on wire
Unicast vs. Multicast on Fabric

Fabric consists of two parallel “fabric planes”:

- **Unicast traffic**
  - Centrally arbitrated
  - Round-robin load balanced over available fabric channels

- **Multicast traffic**
  - Locally arbitrated
  - Load balanced over available fabric channels using hash
Agenda

- Chassis Architecture
- Supervisor Engine Architecture
- I/O Module Architecture
- Forwarding Engine Architecture
- Fabric Architecture
- Layer 2 Forwarding
- IP Forwarding
- IP Multicast Forwarding
- ACLs
- QoS
- NetFlow
Security ACLs

- Enforce security policies based on Layer 2, Layer 3, and Layer 4 information
- Classification TCAM (CL TCAM) provides ACL lookups in forwarding engine
  - 64K hardware entries
- Router ACL (RACL) – Enforced for all traffic crossing a Layer 3 interface in a specified direction
  - IPv4, ARP RACLs supported
- VLAN ACLs (VACLs) – Enforced for all traffic in the VLAN
  - IPv4, MAC VACLs supported
- Port ACLs (PACLs) – Enforced for all traffic input on a Layer 2 interface
  - IPv4, MAC PACLs supported
ACL Architecture

- ACL manager receives policy via configuration
- ACL manager distributes policies to ACL/QoS Clients on I/O modules
- Clients perform ACL merge and program ACEs in Classification (CL) TCAM in forwarding engines

```
module-9# sh processes cpu | egrep aclmgr
PID    Runtime(ms)  Invoked   uSecs  1Sec   Process
3589   1662     516430000      0      0 aclmgr

module-9# sh processes cpu | egrep aclqos
1532   9885     671437     14    0.0 aclqos
```

```
n7010# sh processes cpu | egrep aclmgr
PID    Runtime(ms)  Invoked   uSecs  1Sec   Process
3589   1662     516430000      0      0 aclmgr
```
ACL CL TCAM Lookup

Packet header:
SIP: 10.1.1.1
DIP: 10.2.2.2
Protocol: TCP
SPORT: 33992
DPORT: 80

Generate TCAM lookup key (source/dest IPs, protocol, L4 ports, etc.)

ACL SRAM

ip access-list example
permit ip any host 10.1.2.100
deny ip any host 10.1.68.44
deny ip any host 10.33.2.25
permit tcp any any eq 22
deny tcp any any eq 23
deny udp any any eq 514
permit tcp any any eq 80
permit udp any any eq 161

Forwarding Engine

Result affects final packet handling

Hit in CL TCAM returns result in CL SRAM

Result

SIP | DIP | Protocol | SPORT | DPORT

Hit!

X="Mask"

Return lookup result

Security ACL

Comparison lookup key

Generate Lookup Key

Permit
Deny
Permit
Deny
Permit
Deny
Permit
Displaying Classification Resources

- `show hardware access-list resource utilization module <mod>`

```
n7010# sh hardware access-list resource utilization module 9
```

<table>
<thead>
<tr>
<th>Hardware Modules</th>
<th>Used</th>
<th>Free</th>
<th>Percent Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tcam 0, Bank 0</td>
<td>1</td>
<td>16383</td>
<td>0.000</td>
</tr>
<tr>
<td>Tcam 0, Bank 1</td>
<td>4121</td>
<td>12263</td>
<td>25.000</td>
</tr>
<tr>
<td>Tcam 1, Bank 0</td>
<td>4013</td>
<td>12371</td>
<td>24.000</td>
</tr>
<tr>
<td>Tcam 1, Bank 1</td>
<td>4078</td>
<td>12306</td>
<td>24.000</td>
</tr>
<tr>
<td>LOU</td>
<td>2</td>
<td>102</td>
<td>1.000</td>
</tr>
<tr>
<td>Both LOU Operands</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single LOU Operands</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCP Flags</td>
<td>0</td>
<td>16</td>
<td>0.000</td>
</tr>
<tr>
<td>Protocol CAM</td>
<td>4</td>
<td>3</td>
<td>57.000</td>
</tr>
<tr>
<td>Mac Etype/Proto CAM</td>
<td>0</td>
<td>14</td>
<td>0.000</td>
</tr>
<tr>
<td>Non L4op labels, Tcam 0</td>
<td>3</td>
<td>6140</td>
<td>0.000</td>
</tr>
<tr>
<td>Non L4op labels, Tcam 1</td>
<td>3</td>
<td>6140</td>
<td>0.000</td>
</tr>
<tr>
<td>L4 op labels, Tcam 0</td>
<td>0</td>
<td>2047</td>
<td>0.000</td>
</tr>
<tr>
<td>L4 op labels, Tcam 1</td>
<td>1</td>
<td>2046</td>
<td>0.000</td>
</tr>
</tbody>
</table>

```
n7010#```
Agenda

- Chassis Architecture
- Supervisor Engine Architecture
- I/O Module Architecture
- Forwarding Engine Architecture
- Fabric Architecture
- Layer 2 Forwarding
- IP Forwarding
- IP Multicast Forwarding
- ACLs
- QoS
- NetFlow
Quality of Service

- Comprehensive LAN QoS feature set
- Ingress and egress queuing and scheduling
  Applied in I/O module port ASICs
- Ingress and egress mutation, classification, marking, policing
  Applied in I/O module forwarding engines
- All configuration through Modular QoS CLI (MQC)
  All QoS features applied using class-maps/policy-maps/service-policies
### QoS Architecture

- QoS manager receives policy via configuration
- QoS manager distributes policies to ACL/QoS Clients on I/O modules
- Clients perform ACL merge and program hardware:
  - ACEs in Classification (CL)
  - TCAM in forwarding engines
  - Queuing policies in I/O module port ASICs

```
n7010# sh processes cpu | egrep qos|PID
PID  Runtime(ms)  Invoked  uSecs  1Sec  Process
3849  1074  66946870  0  0  ipqosmgr

module-9# sh processes cpu | egrep aclqos
1532  9885  671437  14  0.0  aclqos
module-9#```
Port QoS - 8-Port 10G Module

- **Buffers**
  - 96MB ingress per port
  - 80MB egress per port

- **Queue Structure**
  - 8q2t ingress
  - 1p7q4t egress
8-Port 10G Module Buffering

Ingress

Port 1
96MB

Replication Engine

PortASIC

Port 1
80MB

Egress

1

8q2t

1p7q4t
Port QoS - 32-Port 10G Module

Buffers

- **Ingress** (two-stage ingress buffering)
  - Dedicated mode: 1MB per port + 65MB per port
  - Shared mode: 1MB per port + 65MB per port group

- **Egress**
  - Dedicated mode: 80MB per port
  - Shared mode: 80MB per port-group

Queue Structure

- 8q2t + 2q1t ingress
- 1p7q4t egress
32-Port 10G Module Buffering - Shared Mode

Ingress:
- Port Group (Fixed) 2q1t
- Ports 1, 3, 5, 7
  - Port 1: 1MB
  - Port 3: 1MB
  - Port 5: 1MB
  - Port 7: 1MB
  - Total: 65MB

Port ASIC
- Replication Engine
- Port 1, 3, 5, 7
  - Port 1, 3, 5, 7
  - Total: 80MB

Egress:
- Port Group 8q2t
- Port 1, 3, 5, 7
- Total: 80MB

4:1 Mux
- 1, 3, 5, 7 Port Group
32-Port 10G Module Buffering - Dedicated Mode

Ingress

Port 1 65MB

Port Group

1,3,5,7

P ort ASIC

Replication Engine

Port 1 80MB

Egress

4:1 Mux

1,3,5,7

Port Group

8q2t

(Fixed)

2q1t

Port 1 1MB

12345678
Port QoS - 48-Port 1G Modules

Buffers
- 7.56MB ingress per port
- 6.15MB egress per port

Queue Structure
- 2q4t ingress
- 1p3q4t egress
48-Port 1G Modules Buffering
Marking and Policing

- Classification uses CL TCAM in forwarding engine to match traffic
- After classification, traffic can be marked or policed
- Marking policies statically set QoS values for each class
- Policing performs markdown and/or policing (drop)
- Policers use classic token-bucket scheme
  
  Uses Layer 2 frame size when determining rate
- Note: policing performed on per-forwarding engine basis
  
  Shared interfaces (such as SVI/EtherChannel) and egress policies could be policed at \(<\text{policing rate} > \times \text{<number of forwarding engines}>\)
QoS CL TCAM Lookup

Packet header:
SIP: 10.1.1.1
DIP: 10.2.2.2
Protocol: TCP
SPORT: 33992
DPORT: 80

Generate TCAM lookup key (source/dest IPs, protocol, L4 ports, etc.)

SIP | DIP | Protocol | SPORT | DPORT
---|---|---|---|---
10.1.1.1 | 10.2.2.2 | 06 | 84C8 | 0050

Compare lookup key

Hit!

CL TCAM

xxxxxxx | 10.2.2.xx | xx | xxx | xxx
xxxxxxx | 10.4.24.xx | xx | xxx | xxx
10.1.1.xx | xxxxxxx | 06 | xxx | xxx
10.1.1.xx | xxxxxxx | 06 | xxx | xxx
xxxxxxx | 10.5.5.xx| 06 | xxx | 0017

Hit in CL TCAM returns result in CL SRAM

Result affects final packet handling

Result

CL SRAM

Policer ID 1
Policer ID 1
Remark DSCP 32
Remark DSCP 40
Remark IP Prec 3

QoS Classification ACLs

ip access-list police
permit ip any 10.3.3.0/24
permit ip any 10.4.12.0/24
ip access-list remark-dscp-32
permit udp 10.1.1.0/24 any
ip access-list remark-dscp-40
permit tcp 10.1.1.0/24 any
ip access-list remark-prec-3
permit tcp any 10.5.5.0/24 eq 23
Monitoring QoS Service Policies

\texttt{show policy-map interface \[\text{[[<interface>]] \[\text{type qos|queuing]\]}\]} brief

n7010\# show policy-map interface e9/1
Global statistics status : enabled
Ethernet9/1

Service-policy (qos) input: mark
policy statistics status: enabled

Class-map (qos): udp-mcast (match-all)
432117468 packets
Match: access-group multicast
set dscp cs4

Class-map (qos): udp (match-all)
76035663 packets
Match: access-group other-udp
police cir 2 mbps bc 1000 bytes pir 4 mbps be 1000 bytes
conformed 587624064 bytes, 3999632 bps action: transmit
exceeded 293811456 bytes, 1999812 bps action: set dscp dscp table cir-markdown-map
violated 22511172352 bytes, 153221133 bps action: drop

n7010#
Agenda

- Chassis Architecture
- Supervisor Engine Architecture
- I/O Module Architecture
- Forwarding Engine Architecture
- Fabric Architecture
- Layer 2 Forwarding
- IP Forwarding
- IP Multicast Forwarding
- ACLs
- QoS
- NetFlow
NetFlow

- NetFlow table is 512K entries (490K effective), shared between ingress/egress NetFlow
- Hardware NetFlow entry creation
  - CPU not involved in NetFlow entry creation/update
- All modules have independent NetFlow table
- Full and sampled NetFlow supported by hardware
NetFlow Architecture

- NetFlow manager receives configuration via CLI/XML
- NetFlow manager distributes configuration to NetFlow-Clients on I/O modules
- NetFlow-Clients apply policy to hardware

```
n7010# sh processes cpu | egrep nfm|PID
PID    Runtime(ms)  Invoked   uSecs  1Sec   Process
24016         1463  735183570      0      0  nfm

module-9# sh processes cpu | egrep nfp
1538        68842    424290    162    0.0  nfp
```

Hardware NetFlow Creation
Full vs. Sampled NetFlow

- NetFlow configured per-direction and per-interface
  Ingress and/or egress on per-interface basis
- Each interface can collect full or sampled flow data
- Full NetFlow: Accounts for every packet of every flow on interface, up to capacity of NetFlow table
- Sampled NetFlow: Accounts for M in N packets on interface, up to capacity of NetFlow table
### Viewing NetFlow Records

**show hardware flow ip [detail] module <mod>**

```plaintext
n7010# sh hardware flow ip interface e9/1 module 9
```

D - Direction; IF - Intf/VLAN; L4 Info - Protocol:Source Port:Destination Port
TCP Flags: Ack, Flush, Push, Reset, Syn, Urgent

<table>
<thead>
<tr>
<th>D IF</th>
<th>SrcAddr</th>
<th>DstAddr</th>
<th>L4 Info</th>
<th>PktCnt</th>
<th>TCP Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>I 9/1</td>
<td>010.001.001.002</td>
<td>010.001.002.002</td>
<td>006:01024:01024</td>
<td>0001403880</td>
<td>. . . S .</td>
</tr>
<tr>
<td>I 9/1</td>
<td>010.001.001.003</td>
<td>010.001.002.003</td>
<td>006:01024:01024</td>
<td>0001403880</td>
<td>. . . S .</td>
</tr>
<tr>
<td>I 9/1</td>
<td>010.001.001.004</td>
<td>010.001.002.004</td>
<td>006:01024:01024</td>
<td>0001403880</td>
<td>. . . S .</td>
</tr>
</tbody>
</table>

```plaintext
n7010# sh hardware flow ip interface e9/1 detail module 9
```

D - Direction; IF - Intf/VLAN; L4 Info - Protocol:Source Port:Destination Port
TCP Flags: Ack, Flush, Push, Reset, Syn, Urgent; FR - FRagment; FA - FastAging
SID - Sampler/Policer ID; AP - Adjacency/RIT Pointer
CRT - Creation Time; LUT - Last Used Time; NtAddr - NT Table Address

<table>
<thead>
<tr>
<th>D IF</th>
<th>SrcAddr</th>
<th>DstAddr</th>
<th>L4 Info</th>
<th>PktCnt</th>
<th>TCP Flags</th>
<th>ByteCnt</th>
<th>TOS</th>
<th>FR</th>
<th>FA</th>
<th>SID</th>
<th>AP</th>
<th>CRT</th>
<th>LUT</th>
<th>NtAddr</th>
</tr>
</thead>
<tbody>
<tr>
<td>I 9/1</td>
<td>010.001.001.002</td>
<td>010.001.002.002</td>
<td>006:01024:01024</td>
<td>0001706722</td>
<td>A . . . S .</td>
<td>0000218460416</td>
<td>0x000</td>
<td>N</td>
<td>Y</td>
<td>0x000</td>
<td>0x000000</td>
<td>02168</td>
<td>02571</td>
<td>0x000331</td>
</tr>
</tbody>
</table>
NetFlow Data Export

To NetFlow Collector

Fabric ASIC

Supervisor Engine

I/O Module

NetFlow Table

I/O Module

NetFlow Table

I/O Module

NetFlow Table

Generate NetFlow v5 or v9 export packets

Hardware Flow Creation

Aged Flows

Forwarding Engine

Aged Flows

Forwarding Engine

Aged Flows

Forwarding Engine

Main CPU

Switched EOBC

Mgmt Enet

To NetFlow Collector

via Inband

via mgmt0

NetFlow Collector

Generate NetFlow v5 or v9 export packets

Hardware Flow Creation

Hardware Flow Creation

Hardware Flow Creation
Conclusion

- You should now have a thorough understanding of the Nexus 7000 switching architecture, I/O module design, packet flows, and key forwarding engine functions...

- Any questions?
Q and A
Complete Your Online Session Evaluation

- Give us your feedback and you could win fabulous prizes. Winners announced daily.
- Receive 20 Passport points for each session evaluation you complete.
- Complete your session evaluation online now (open a browser through our wireless network to access our portal) or visit one of the Internet stations throughout the Convention Center.

Don’t forget to activate your Cisco Live Virtual account for access to all session material, communities, and on-demand and live activities throughout the year. Activate your account at the Cisco booth in the World of Solutions or visit www.ciscolive.com.