June 22, 2010

Software Datapath Acceleration for Stateless Packet Processing

FTF-NET-F0817

Ravi Malhotra
Software Architect
Agenda

► What can be accelerated
  • Stateless and stateful
  • Various applications
  • Sample TCP offload

► Soft Data Path Engine
  • Architecture
  • Feature set
  • Packet flow
  • DPE API

► Performance
  • Soft DPE advantage
  • Leverage key hardware offloads
Stateful Path and Stateless Packet Processing

► Most network packet processing protocols can be broken down into two paths
  • Stateless path, also known as the data path, requires quick and efficient switching/routing of packets
    ▪ Can be broken down into packet identification (classification) and forwarding
  • Stateful path, also known as the control path, requires more processing and has more inherent latency than the data path

► Stateful control path requires **90%** of the code and is used **10%** of the time.

► Stateless data path requires just **10%** of the code and is used **90%** of the time.

► This session focuses on how to accelerate the **10%** of the code in the stateless path to increase packet processing performance.
### Stateless Data Path for Different Applications

<table>
<thead>
<tr>
<th>Application</th>
<th>Data Path</th>
<th>Control Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer 2 bridging</td>
<td>FDB lookup, VLAN add/delete, Learning</td>
<td>Aging, STP</td>
</tr>
<tr>
<td>IPv4 forwarding</td>
<td>Dest-cache lookup, L2 modify</td>
<td>LPM route-table lookup, ARP, IP Options</td>
</tr>
<tr>
<td>NAPT</td>
<td>5-tuple lookup, IP/Port/L2 modify</td>
<td>Connection setup/destroy, policy, ALG</td>
</tr>
<tr>
<td>Firewall</td>
<td>Access control list, pin-holes</td>
<td>Stateful packet inspection, ALG</td>
</tr>
<tr>
<td>IPSec</td>
<td>5-tuple lookup, encap/decap + crypto</td>
<td>SA setup, security policy</td>
</tr>
<tr>
<td>QoS</td>
<td>Enforcement – sched, police, congestion, shaper</td>
<td>Policy, provisioning</td>
</tr>
</tbody>
</table>
Dynamic Connection Offloading with Soft DPA
(L4 TCP NAPT flow, no QoS)
Current – Linux Forwarding

<table>
<thead>
<tr>
<th>Platform</th>
<th>SoC</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1, P2</td>
<td>P1010, P1020, P2020, 85xx, 83xx</td>
</tr>
<tr>
<td>P3, P4, P5</td>
<td>P4080, P3040, P5020</td>
</tr>
</tbody>
</table>

- Data Packet flow
- Control Packet flow
- Configuration flow

Control Plane Applications (DHCP/DNS/IGMP etc)

Linux User-space

Linux Kernel

Linux Network Stack

eTSEC/QM Driver

eTSEC or FM-QM-BM

eTSEC or FM-BM-QM

Platform SoC
P1, P2 P1010, P1020, P2020, 85xx, 83xx
P3, P4, P5 P4080, P3040, P5020
Stateless Data Path Processing in QE

### Platform vs. SoC

<table>
<thead>
<tr>
<th>Platform</th>
<th>SoC</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1, P2</td>
<td>8323, 8360, 8569</td>
</tr>
</tbody>
</table>

**Control Plane Applications (DHCP/DNS/IGMP etc)**

- **Linux User-space**
- **Linux Kernel**

**Linux Network Stack**

- **Network Driver**
- **Control Logic**
- **DPE API**

**Interworking microcode**

- **UCC**

**Existing Solution for 8360 and 8323**

**RISC Microcode**

- **SEC**

**QE RISC Cores**

- **Network Interface**

**Network Interface**

**Control Packet flow**

**Configuration flow**

**Data Packet flow**
Stateless Data Path Processing in Software

### Overview

- **Data Packet flow**
- **Control Packet flow**
- **Configuration flow**

### Diagram Details

- **Control Plane Apps (DHCP/DNS/IGMP/IKE etc)**
- **VortiQa CP + NMS**
- **Linux Network Stack**
- **VortiQa Network Stack**
- **Soft Data Path Engine**
- **VeTSEC Driver**
- **VeTSEC**
- **Control Logic**
- **DPE API**
- **SEC Driver**

### Platform Table

<table>
<thead>
<tr>
<th>Platform</th>
<th>SoC</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1, P2</td>
<td>P1010, P1020, P2020, 85xx</td>
</tr>
<tr>
<td>P3, P4, P5</td>
<td>N/A</td>
</tr>
</tbody>
</table>

### Notes

- **Completely re-used from QE based Platforms**
- **e500 Cores**

---

Freescale, the Freescale logo, AbWin, C5, CodeTEST, CodeWarrior, ColdFire, C-Ware, mobileGT, PowerQUICC, StarCore, and Symphony are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. BeeKit, BeeStack, CoreNet, the Energy Efficient Solutions logo, Flexis, MXC, Platform in a Package, Processor Expert, QorIQ, QUICC Engine, SMARTMOS, TurboLink and VortiQa are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © 2010 Freescale Semiconductor, Inc.
Soft Data Path Engine Feature List

► Stateless packet processing (all stateful processing including ALG, SPI firewall, ARP, routing, learning etc. done by control-path)

► Offloads following stateless processing
  • IPv4 forwarding
  • NAPT/firewall (ACL) processing
  • Layer 2 switching with VLAN
  • IPSec forwarding
  • Quality of service

► Support for the following interfaces:
  • Ethernet
  • VLAN
  • PPPoE
  • WLAN
Soft Data Path Engine Feature List (cont.)

► Maintenance
  • Per-flow statistics and aging

► Platform support
  • Multicore support over VeTSEC

► Provides a standard configuration across platforms
  • Integrates seamlessly with Linux networking stack and applications using SWANG package
  • Integrates seamlessly with VortiQa networking stack and customer network stacks

► Leverages hardware acceleration (hashing, scheduling, classification, security) where available
Data Path Engine API Architecture Overview
Soft Data Path Engine Performance Advantage

Results on P2020 RDB - 1200/600/400 : 2-core SMP Linux

Significant (2x to 5x) performance improvement over native Linux
Soft Data Path Engine Multicore Scaling

Results on P2020 RDB - 1200/600/400: 1-core non-SMP vs. 2-core SMP Linux

Scaling factor of > 1.8x when migrating from 1-core to 2-core
Soft Data Path Engine Flow Scaling

Results on P2020 RDB - 1200/600/400 : 2-core SMP Linux – 64 byte traffic

IPv4

NAPT

IPSec

Low performance degradation for handling multiple flows
Data Path Hardware Acceleration

Core(s)

Network Stack (SMP optimized)

Autonomous aware

Drivers/API

Look-Aside Offload

Generic Offload

Ingress Offload

Autonomous Processing

Egress Offload
## Hardware Acceleration Support

<table>
<thead>
<tr>
<th>Offload</th>
<th>Feature</th>
<th>Advantage</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ingress</strong></td>
<td>Hash calculation</td>
<td>Packet distribution to multiple cores, flow-pinning, table lookup</td>
</tr>
<tr>
<td></td>
<td>Coarse classification</td>
<td>Offload stateless ACL processing</td>
</tr>
<tr>
<td></td>
<td>Packet parsing</td>
<td>Avoid software overhead</td>
</tr>
<tr>
<td><strong>Generic</strong></td>
<td>Hardware buffer management</td>
<td>No buffer alloc/free operations in software</td>
</tr>
<tr>
<td></td>
<td>Hardware queue management</td>
<td>Simpler packet Rx/Tx, efficient stashing (to L1/L2), leaves room in cache for other data</td>
</tr>
<tr>
<td><strong>Egress</strong></td>
<td>Hardware QoS</td>
<td>Avoid software overhead, mitigate DoS attacks, prioritize CPU cycles</td>
</tr>
<tr>
<td><strong>Core</strong></td>
<td>Backside L2 cache</td>
<td>Faster access for multiple flow tables</td>
</tr>
<tr>
<td><strong>Look-Aside</strong></td>
<td>Protocol-aware cryptography</td>
<td>Offload protocol encapsulation/decapsulation, sequence tracking etc.</td>
</tr>
</tbody>
</table>
Hardware Acceleration Advantage

Hardware Acceleration provides upto 2.5x improvement
Summary

► Software data path engine
  • Optimized packet processing path
  • Consistent interface across platforms
  • Easy integration with network stacks
  • Single solution across QorIQ LE/ULE platforms

► Performance advantage
  • Flexibility to leverage hardware acceleration
  • Optimized for multicore scaling