Demystifying the Performance of XDP BPF

Oliver Hohlfeld\textsuperscript{2}, Johannes Krude\textsuperscript{1}, Jens Helge Reelfs\textsuperscript{1}, Jan Rüth\textsuperscript{1}, Klaus Wehrle\textsuperscript{1}

\textsuperscript{1}RWTH Aachen University, \textsuperscript{2}Brandenburg University of Technology

https://comsys.rwth-aachen.de/
Offloading Packet Processing

- $\geq 10 \text{ GBit/s}$ challenge network stacks
- **Common solution:** Offloading from user-space to
  - CPU based SmartNIC
  - Device driver
  - Host OS in case of virtualization

This Paper: Comparison of same eBPF Programs at

- Netronome Agilio CX 2x10GbE & Linux 4.18.10 on Core i7-7700
Offloading Packet Processing

- \( \geq 10 \text{ GBit/s} \) challenge network stacks
- **Common solution: Offloading from user-space to**
  - CPU based SmartNIC
  - Device driver
  - Host OS in case of virtualization

- **XDP & eBPF**
  - Safe code execution with native performance
  - Executed within the device-driver or on the NIC
  - Can steer packets directly to user-space, bypassing most of the stack
Offloading Packet Processing

- ≥ 10 GBit/s **challenge network stacks**
- **Common solution:** Offloading from user-space to
  - CPU based SmartNIC
  - Device driver
  - Host OS in case of virtualization

- **XDP & eBPF**
  - Safe code execution with native performance
  - Executed within the device-driver or on the NIC
  - Can steer packets directly to user-space, bypassing most of the stack

This Paper: Comparison of same eBPF Programs at □, △ & ○

- **Netronome Agilio CX 2x10GbE & Linux 4.18.10 on Core i7-7700**
Program: `return XDP_DROP;`

- **Minimum sized packets** →
  - NIC
  - Device driver

- **Throughput Baseline: Drop All**

- **Drop rate [pkts/s]**
  - 0
  - 5 M
  - 10 M
  - 15 M

- **CPU cores [#]**
  - 1
  - 2
  - 3
  - 4

• Offloading improves performance
• Traffic distribution must be taken into account
Program: `return XDP_DROP;`

- Throughput Baseline: Drop All
- `minimum sized packets` → kernel-bypass
- NIC device driver

### Multiple Flows

<table>
<thead>
<tr>
<th>CPU cores [#]</th>
<th>0</th>
<th>5 M</th>
<th>10 M</th>
<th>15 M</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Offloading improves performance
- Traffic distribution must be taken into account
Throughput Baseline: Drop All

Program: `return XDP_DROP;`

- Offloading improves performance

### Multiple Flows

<table>
<thead>
<tr>
<th>CPU cores [#]</th>
<th>0</th>
<th>5 M</th>
<th>10 M</th>
<th>15 M</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Traffic distribution must be taken into account

**minimum sized packets** → [ ] NIC  △ device driver  ☐ kernel-bypass
Throughput Baseline: Drop All

Program: `return XDP_DROP;`

\[\text{minimum sized packets} \rightarrow \text{NIC} \quad \text{kernel-bypass} \rightarrow \text{device driver}\]

- Offloading improves performance
Throughput Baseline: Drop All

Program: `return XDP_DROP;`

- Offloading improves performance
- Traffic distribution must be taken into account
Processing Complexity

Program: for (i = 1; i < n; i++)
    do_something();
return XDP_DROP;
Processing Complexity

Program: for (i = 1; i < n; i++)
    calculate_checksum();
return XDP_DROP;
Processing Complexity

Program: for (i = 1; i < n; i++)
    calculate_checksum();
return XDP_DROP;

Performance depends on the processing complexity.
Most beneficial if task is smaller than the removed overhead.
Slow NIC CPU gets easily overloaded.
Processing Complexity

Program: for (i = 1; i < n; i++)
        calculate_checksum();
    return XDP_DROP;

- Performance depends on the processing complexity

Graph showing packet rate vs. checksum calculations.
Program: for \( i = 1; \ i < n; \ i++ \)
\[
\text{calculate_checksum();}
\]
\[
\text{return XDP_DROP;}
\]

- Performance depends on the processing complexity
- Most beneficial if task is smaller than the removed overhead
Program: for (i = 1; i < n; i++)
calculate_checksum();
return XDP_DROP;

• Performance depends on the processing complexity
• Most beneficial if task is smaller than the removed overhead
Program: for (i = 1; i < n; i++)
    calculate_checksum();
return XDP_DROP;

- Performance depends on the processing complexity
- Most beneficial if task is smaller than the removed overhead
- Slow NIC CPU gets easily overloaded
Reducing Latency

Program: `payload.dns.flags |= NXDOMAIN; swap(src,dst); return XDP_TX;`
Reducing Latency

Program: `payload.dns.flags |= NXDOMAIN; swap(src,dst); return XDP_TX;`

- Latency is highly influenced by the packet rate.
- Most latency is introduced before the device driver.

![Graph showing response time vs. packet rate]
Reducing Latency

Program: `payload.dns.flags |= NXDOMAIN;
swap(src,dst);
return XDP_TX;`

- Latency is highly influenced by the packet rate
- Most latency is introduced before the device driver
Reducing Latency

Program: `payload.dns.flags |= NXDOMAIN; swap(src, dst); return XDP_TX;`

- Latency is highly influenced by the packet rate
- Most latency is introduced before the device driver
Offloading From Virtual Machines

- Virtualization adds layers
  - Hypervisor: Xen 4.9.2
  - Open vSwitch
  - vNIC driver
⇒ More offloading potential

virtual machine

- kernel-bypass
- AF_XDP

NIC

△ host device driver
Offloading From Virtual Machines

- Virtualization adds layers
  - Hypervisor: Xen 4.9.2
  - Open vSwitch
  - vNIC driver

⇒ More offloading potential

Program: `return XDP_DROP;`

---

**virtual machine**

- kernel-bypass
- AF_XDP
- VM device driver
- NIC
- host device driver

---

<table>
<thead>
<tr>
<th>CPU cores</th>
<th>0</th>
<th>5 M</th>
<th>10 M</th>
<th>15 M</th>
</tr>
</thead>
<tbody>
<tr>
<td>drop rate [pkts/s]</td>
<td>0</td>
<td>5 M</td>
<td>10 M</td>
<td>15 M</td>
</tr>
</tbody>
</table>
Offloading From Virtual Machines

• Virtualization adds layers
  ▶ Hypervisor: Xen 4.9.2
  ▶ Open vSwitch
  ▶ vNIC driver

⇒ More offloading potential

Program: return XDP_DROP;

• Only minimal benefit in offloading within a VM
Offloading From Virtual Machines

- Virtualization adds layers
  - Hypervisor: Xen 4.9.2
  - Open vSwitch
  - vNIC driver

  More offloading potential

Program: `return XDP_DROP;`

- Only minimal benefit in offloading within a VM
- But offloading to the host requires isolation
Influence on VM Host

Program: `return XDP_DROP;`

- **Kernel-bypass**
- **VM device driver**
- **AF_XDP**
- **NIC**
- **Host device driver**

- **CPU usage [%]**
  - **VM**
  - **VM share**
  - **Host share**

- **Diagram:**
  - CPU usage comparison between VMs and host with different device drivers.

- **Key Points:**
  - Offloading can reduce CPU usage in VM and host.
  - But may shift CPU usage to the host.
  - Introduces fairness and accounting problems.
Influence on VM Host

Program: return XDP_DROP;

- Offloading can reduce CPU usage in VM and host

• Offloading can reduce CPU usage in VM and host
Influence on VM Host

Program: return XDP_DROP;
Program:
for (i = 1; i < n; i++)
calculate_checksum();
return XDP_DROP;

- Offloading can reduce CPU usage in VM and host
- But may shift CPU usage to the host
  - Introduces fairness & accounting problems
Influence on Neighboring VMs

Program:

```c
for (i = 1; i < n; i++)
    calculate_checksum();
return XDP_DROP;
```

- The NIC can introduce delays for non-offloaded applications.
Influence on Neighboring VMs

Program:
```c
for (i = 1; i < n; i++)
    calculate_check_checksum();
return XDP_DROP;
```

- VM 1:
  - Program: DNS Ping
  - Kernel-bypass
- VM 2:
  - Program: DNS Ping
  - VM device driver

The NIC can introduce delays for non-offloaded applications.
Influence on Neighboring VMs

Program:

```c
for (i = 1; i < n; i++)
    calculate_checksum();
return XDP_DROP;
```

- **VM 1**
  - **Program:** DNS Ping
  - **Device:** VM device driver

- **VM 2**
  - **Device:** host device driver
  - **Device:** kernel-bypass

The NIC can introduce delays for non-offloaded applications.
Influence on Neighboring VMs

Program:

```c
for (i = 1; i < n; i++)
    calculate_checksum();
return XDP_DROP;
```

• The NIC can introduce delays for non-offloaded applications
• **Offloading performance depends on specific scenario**
  - Offloading may be beneficial
  - But, device driver offloading may not be sufficient
  - But, slow NIC CPU may get overloaded
  - But, offloading may affect traffic of non-offloaded applications
Conclusion

- Offloading performance depends on specific scenario
  - Offloading may be beneficial
  - But, device driver offloading may not be sufficient
  - But, slow NIC CPU may get overloaded
  - But, offloading may affect traffic of non-offloaded applications

- More measurements and results in our paper