mmb: Flexible High-Speed Userspace Middleboxes

Korian Edeline, Justin Iurman, Cyril Soldani, Benoit Donnet
Montefiore Institute, University of Liège
Belgium
A middleboxed Internet

https://github.com/mami-project/roadshows
kernelspace vs userspace

Kernel:

Userspace:
kernelspace vs userspace

Kernel:
× Too slow for high-speed forwarding
× Missing optimizations (batching, caching, etc)

Userspace:
kernelspace vs userspace

Kernel:
✗ Too slow for high-speed forwarding
✗ Missing optimizations (batching, caching, etc)

Userspace:
✗ No direct access to NIC (context switch, sk_buff)
kernelspace vs userspace

Kernel:
✗ Too slow for high-speed forwarding
✗ Missing optimizations (batching, caching, etc)

Userspace:
✗ No direct access to NIC (context switch, sk_buff)
✔ DPDK (DMA, I/O batching)
kernelspace vs userspace

Kernel:
✗ Too slow for high-speed forwarding
✗ Missing optimizations (batching, caching, etc)

Userspace:
✗ No direct access to NIC (context switch, sk_buff)
✔ DPDK (DMA, I/O batching)
✔ Software optimizations
✔ Flexibility
## Kernel-Bypass Frameworks

<table>
<thead>
<tr>
<th>Framework</th>
<th>Modularity &amp; Flexibility</th>
<th>Optimization Techniques</th>
<th>Commodity Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click</td>
<td>●</td>
<td>○</td>
<td>✓</td>
</tr>
<tr>
<td>PF_RING</td>
<td>○</td>
<td>○</td>
<td>✓</td>
</tr>
<tr>
<td>RouteBricks</td>
<td>●</td>
<td>○</td>
<td>✓</td>
</tr>
<tr>
<td>mOS</td>
<td>○</td>
<td>○</td>
<td>✓</td>
</tr>
<tr>
<td>PacketShader</td>
<td>○</td>
<td>○</td>
<td>✗</td>
</tr>
<tr>
<td>DoubleClick</td>
<td>●</td>
<td>○</td>
<td>✓</td>
</tr>
<tr>
<td>FastClick</td>
<td>●</td>
<td>○</td>
<td>✓</td>
</tr>
<tr>
<td>MiddleClick</td>
<td>●</td>
<td>○</td>
<td>✓</td>
</tr>
<tr>
<td>ClickNP</td>
<td>●</td>
<td>●</td>
<td>✗</td>
</tr>
<tr>
<td>VPP</td>
<td>●</td>
<td>●</td>
<td>✓</td>
</tr>
</tbody>
</table>
Vector Packet Processing (VPP)

- DPDK
- RSS queues, Zero-Copy and more
- Packet vectors
- Modular node-based processing
- Low-level optimizations (caching, pipelining)
VPP Dual-Loop

while (n_left_from >= 2) {
    /* prefetch next iteration */
    if (PREDICT_TRUE(n_left_from >= 4)) {
        vlib_prefetch_buffer_header(b[2], STORE);
        vlib_prefetch_buffer_header(b[3], STORE);
    }

    process(b[0]);
    process(b[1]);

    b += 2;
    next += 2;
    n_left_from -= 2;
}

/* process remaining packets */
while (n_left_from > 0) {
    process(b[0]);

    b += 1;
    next += 1;
    n_left_from -= 1;
}
mmb: A VPP middlebox

Goals:

• Various middlebox policies (firewall, NAT, traffic engineering)

• Fast even with thousands rules

• Intuitive CLI
mmb: CLI grammar

# mmb <add-keyword> <match> ... <match>
    <target> ... <target>

<add-keyword> : add-stateless | add-stateful
<match> : <field> <condition> <value>
<target> : mod <field> <value> | add <field> <value>
            | strip [!] <field> | map <field> <value>
            | shuffle <field> | drop
mmb: forwarding graph

architecture

ip4-input

ip4-mmb-classify

MATCH

DROP

MISS

ip4-lookup

error-drop

ip4-mmb-rewrite
mmb: forwarding graph

- Classification: 
  \[(\text{Packet} \& \text{Mask}) \oplus \text{Key}\]

- Rewrite
  \[(\text{Packet} \& \text{Mask}) \mid \text{Key}\]
mmb: processing path
Performance Analysis

- **FastClick:**
  - Fast (multi-queue, ZC forwarding, batching, DPDK)
  - Click
- **eXpress Data Path (XDP):**
  - In-Kernel
  - eBPF
- **iptables**
Performance Analysis: Testbed

Direct

Indirect

PCI Passthrough

Bridged
Performance Analysis: Baselines

- VPP, FastClick, 4.15 > 99% of direct baseline
Performance Analysis: 5-tuples firewall

- Stateless matching on 5-tuples \((saddr, daddr, sport, dport, proto)\)
Performance Analysis: 5-tuples firewall

- Stateless matching on 5-tuples (saddr, daddr, sport, dport, proto)
- mmb & XDP at direct baseline
- FastClick matching (IPFilter) has performance issues
- Iptables 4.15 sustains direct baseline with up to 1,000 rules
Performance Analysis: stateful flow matching

- Stateful matching on 5-tuples (saddr, daddr, sport, dport, proto)
Performance Analysis: stateful flow matching

- Stateful matching on 5-tuples (saddr, daddr, sport, dport, proto)
- mmb & XDP at direct baseline
- FastClick at 85% direct baseline
- Iptables stateful is similar to stateless (with few rules).
Performance Analysis: TCP Options

- Matching on TCP Options
- Not applicable to iptables, FastClick & XDP
- Stable until 78 rules
Conclusion & Next steps

- mmb sustains line rate for different use cases
- Next Step: Payload reconstruction
- https://github.com/mami-project/vpp-mb
Thanks!
Performance Analysis: Testbed

- Intel Xeon E5-2620 2.1GHz, 16 Threads, 32GB RAM
- Intel XL710 2x40GB NICs
- Huawei CE6800 switch
- Debian 9
Performance Analysis: RTT

- mmb, kernel 4.15
- mmb, kernel 4.9
- XDP fw
- FastClick
- iptables
- iptables conntrack
Performance Analysis: CPU time

- Clocks per Node

- Use Cases: FW, Stateful, NAT, TCP-OPTS

- Categories: ip4-input, ip4-lookup, ip4-rewrite, ip4-mmb-rewrite, ip4-mmb-classify, FortyGigabitEthernet2/0/1-tx, FortyGigabitEthernet2/0/1-output, FortyGigabitEthernet2/0/0-tx, FortyGigabitEthernet2/0/0-output, dpdk-input