Implementing Software Virtual Routers on Multi-core PCs using Click

Mickaël Hoerdt, Dept. of computer engineering
Université catholique de Louvain la neuve
mickael.hoerdt@uclouvain.be

In collaboration with:

N. Egi, P. Papadimitriou, L. Mathy (Lancaster Uni.)
M. Handley, A. Greenhalgh (UCL) F. Huici (NEC Europe)
• Traditionally, high performance networking devices:
  - Built as custom hardware
  - Made of various specialized “CPUs”
  - Specialized multi-cores

• Software routers
  - Familiar programming environment
  - Easily extensible
  - Cheap
  - Generic packet processing capability
  - Run on (slow ?) commodity hardware ?
Background

• But recent advances in commodity HW architectures
  
  - Multi-core (With very high clock frequency)
  
  - Buses (usual bottlenecks) are disappearing

• What about implementing virtual routers with all those available CPU cycles?
- Memory closer to the CPU: fast and small
- Memory Further from the CPU: slow and big
- Huge memory latency difference between L1 and Main memory (up to a factor 100)
### Background

#### Some numbers:

<table>
<thead>
<tr>
<th>Setup</th>
<th>Performances</th>
<th>Bottleneck</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Mode Linux IPv4 performances (64 bytes)</td>
<td>About 59 Kpps</td>
<td>Context switch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kernel space – user space</td>
</tr>
<tr>
<td>Xen – Guest Domain IPv4 performances (64 bytes)</td>
<td>About 150 Kpps</td>
<td>Context switch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hypervisor space – user space</td>
</tr>
<tr>
<td>Xen – Privileged Domain IPv4 performances (64 bytes)</td>
<td>About 800 Kpps</td>
<td>Single core cycles limitation</td>
</tr>
<tr>
<td>Linux - kernel IPv4 performances (64 bytes)</td>
<td>About 800 Kpps</td>
<td>Single core cycles limitation</td>
</tr>
<tr>
<td>6 running cores Raw performances (64 bytes)</td>
<td>About 7.2 Mpps</td>
<td>Memory latency</td>
</tr>
<tr>
<td>6 running cores IPv4 performances (64 bytes)</td>
<td>About 4.4 Mpps</td>
<td>Memory latency</td>
</tr>
</tbody>
</table>
Software Virtual Routers: Data plane

Aim: To avoid memory domain switch per packet

Legend
- Single Memory Segment
- Physical NIC
- Packets

Vrouter1's Domain
- Control Plane

Vrouter2's Domain
- Control Plane

Vrouter3's Domain
- Control Plane

Forwarder domain
- Click

Hypervisor domain

Data planes are off-loaded
To a single memory domain
Forwarder Domain Architecture: overview

**Legend**
- **Single memory space**
- **Static part of the Forwarding engine**
- **Dynamic part of the Forwarding engine**
- **Packets**
- **Physical NIC**
Forwarder Domain Architecture: input processing

Shared hardware queues
+ scales well with the number of supported virtual routers
- requires software classification: subject to unfairness

Broadcast ethernet traffic is replicated on all vouters
Forwarder Domain Architecture: input processing

Shared hardware queues
+ scales well with the number of supported virtual routers
- requires software classification: subject to unfairness

Dedicated hardware queues
+ Can drop packets before they hit memory: achieve fairness
- Limited number of supported virtual routers
Forwarder Domain Architecture: input processing

How can the CPUs core always be ready to poll everywhere?

Core 1

PollDevice → VRouter 1

Core 2

PollDevice → VRouter 2
Forwarder Domain Architecture: input processing

How can the CPUs core always ready to poll everywhere?

By exploiting NICs with multiple hardware queues:

Core 1

Core 2
Forwarder Domain Architecture: output processing

A Single ToDevice per hardware queue
+ Avoid costly cache misses
- Limited to a single core
Forwarder Domain Architecture: output processing

A Single ToDevice per hardware queue
+ Avoid costly cache misses
- Limited to a single core

Several ToDevice per hardware queue
+ Can exploits all the cores cycles
- Spinlock can trigger cache misses

Spinlock Perf loss

<table>
<thead>
<tr>
<th>Cores</th>
<th>Perf Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>-15%</td>
</tr>
<tr>
<td>4</td>
<td>-33%</td>
</tr>
<tr>
<td>6</td>
<td>-45%</td>
</tr>
</tbody>
</table>
Exploiting several hardware queues
  + Can exploit all the cores cycles
  - Limited number of supported vouters.

Forwarder Domain Architecture: output processing
Forwarder Domain Architecture: output processing

Exploiting several hardware queues

![Diagram showing throughput vs. number of ToDevices for different configurations.](image)
Forwarder Domain Architecture: output processing

Exploiting several hardware queues

Very significant Performance impact
Forwarder Domain Architecture: output processing

Exploiting several hardware queues

Cache hierarchy effect!
Forwarder Domain Architecture: switching

• We don't know
  On which outgoing interface a packet will be switched.

• But we want
  To always keep packets on the same cache hierarchy.

• Solution
  Software Tree based scheduling.
Forwarder Domain Architecture: switching

- Software Tree based scheduling requires elements replication
Forwarder Domain Architecture: switching

- Software Tree based scheduling requires elements replication

And element data replication too!
Building a shared forwarding path for vrouters is a trade-off between the desired:

- Desired Scalability depends on - Level of fairness and performances

- Fairness is obtained by:
  - Assigning tickets wisely to the Click scheduler

- Performances is obtained by:
  - Distributing the computation among CPU cores to maximize the number of available CPU cycles
  - Keeping packets as deep as possible inside the cache hierarchy to minimize memory latency
Forwarder Domain Architecture: Merging

So far we talked about input processing, output processing and switching.
So far we talked about input processing, output processing and switching.

But what about the syntactic glue between Input/output processing elements and the vrouters?

Goal: From Virtual Routers Click configuration designers perspective, the forwarder Domain architecture stays opaque.
Vrouters click configurations are the same as routers config Anything between Poll/From-Device and ToDevice
Vrouters click configurations are the same as routers config anything between Poll/From-Device and ToDevice.

Input processing elements are all ending by a “push” connection.
Forwarder Domain Architecture : Merging

Vrouters click configurations are the same as routers config anything between Poll/From-Device and ToDevice

Input processing elements are all ending by a “push” connection.

Output processing elements are all starting by a “pull” connexion.
Forwarder Domain Architecture : Merging

1. Suppress PollDevice/ToDevice from the config
1. Suppress PollDevice/ToDevice from the config

2. Prefix all the elements names with the Vrouter ID.
Forwarder Domain Architecture: Merging

1. Suppress PollDevice/ToDevice from the config

2. Prefix all the elements names with the Vrouter ID.

3. Find free slots and plug the config in the forwarder domain

Input processing
PollDevice → Classifier → vrouter-foo

Output processing
vrouter-foo → ToDevice
Forwarder Domain Architecture : Merging

All the vrouter handlers are stored under the assigned prefix directory within the click-fs.

Ex: /click/vrouter-foo/counter/count
    /click/vrouter-foo/counter/reset
    /click/vrouter-foo/counter/bit_rate

This directory is then exported to the virtual router address space
All the vrouter handlers are stored under the assigned prefix directory within the click-fs.

Ex: /click/vrouter-foo/counter/count
    /click/vrouter-foo/counter/reset
    /click/vrouter-foo/counter/bit_rate

This directory is then exported to the virtual router domain

To delete/update the vrouter config, all the elements prefixed with the vrouter-id are deleted from the Click config
How is the control plane traffic sent/received to/from vrouters?

Legend:
- Frontends VNIC
- Backends VNIC
- Physical NIC
- Control plane traffic
Forwarder Domain Architecture: Control plane

For the traffic coming from the vrouter control plane

- A mapping file that associates backends with the physical NICs help us to build the Click plumbing.

For the traffic going to the vrouter control plane

- Vrouter's use the equivalent of the “ToHost” element to indicate a interest in receiving traffic
Forwarder Domain Architecture: Future work

- Platform software packaging.

- Vrouter management.

- Automatic resource allocation and scheduling of concurrent vouters

- Performances requirements to physical resources mapping.