Designing a Platform for Flexible and Performant Virtual Routers on Commodity Hardware

Adam Greenhalgh, Dept. of Computer Science
University College London
a.greenhalgh@cs.ucl.ac.uk

In collaboration with:
N. Egi, M. Hoërdt, L. Mathy, P. Papadimitriou (ULancs)
M. Handley (UCL)
F. Huici (NEC Europe)
Talk overview

• Background
• Virtual Router Platform Design
  – Consolidation of forwarding paths
  – Hardware multi-queuing for interface sharing
• Packet Processing Scheduling
• Evaluation
• Related Work
• Conclusions
A Software router can:
- Forward 64 byte packets at 7Mp/s (million packets / second).
- Be sufficient for edge networks.
- Be faster than the cheap CISCOs.

Spare CPU cycles.

Why not run several (virtual) routers on a single box?
Challenges of virtual routers?

- Isolating VRs from each others.
- Minimise virtualization overhead.
- Fair scheduling of resources.
- Mapping CPU cores to VRs in high performance way.

[Diagram with ISOLATION, PERFORMANCE, FLEXIBILITY labels]
Why now? Enabling technologies

• Virtual machines on regular modern x86 PC hardware
  – XEN, VMware, etc.
  – Both Intel and AMD are adding virtualization extensions to their processors

• Inexpensive and powerful x86 PC hardware
  – Forwards several Gbp/s even with min. sized packets
  – Sufficient capacity for most small and medium size businesses

• Future hardware trend
  – Massively Multi-core x86 PC architecture
  – Advanced multi-queue NICs
Why now? Enabling technologies

• XEN
  – Para-virtualization
  – VMs with performance close to native hardware
  – Excellent hardware support
  – Exploits Intel’s and AMD’s virtualization technologies

• Click Modular Router
  – Running in the kernel
  – Simple and small elements
  – Plenty of elements ready to use
  – Implement new elements is simple
  – High performance (outperforms even native Linux)

• Mature open source Control Planes
  – Xorp, Quagga
• Main Features
  – Consolidation of FP\(s\)
    - Single forwarding domain (IDD)
    - Offloading FP\(s\) and merging in the IDD
  – Hardware multi-queueing for interface sharing
  – Packet Processing Scheduling
IDD and Guest Domains

IDD hosts the forwarding plane (Click)

Guest domains:
- host control planes (XORP, Quagga)
- may host FPs (Click)
Other Forwarding Configurations

- Forwarding with I/O Channels (FP1 ↔ FP3)
- Forwarding in Guest Domains with direct NIC mapping (FP4)
• 10Gb NIC **Device Driver** based on Intel’s 82598:

• Packets are *filtered* into *different* queues and *polled* by virtual routers

• Support for up to 16 queues (filtering on the **MAC** level)

• **Fairness** is attained
Goals and Principles for Scheduling

• Main Goals:
  – Keep packets at the same cache hierarchy
  – Utilize the spare CPU cycles efficiently
  – Share CPU resources among virtual routers fairly

• Basic Principles:
  – Schedule forwarding trees
  – A tree should be scheduled in a single core
  – Use one thread per core (avoid context switching)
  – Multiple trees can share a core
Forwarding Trees

- One tree per input interface is set up
- Output interfaces are shared (by modifying Click scheduler)
- A tree is scheduled in a single core
Fair Scheduling for Virtual Routers

- **Click Scheduling:**
  - Tickets are assigned to schedulable tasks
  - Inactive tasks gradually “lose” their tickets
  - Inactivity of trees can cause unfairness among virtual routers

- **Ticket Mitigation**
  - Preserve the resource (ticket) allocation among virtual routers
  - When a virtual router has one or more inactive tree:
    - Distribute the tickets assigned to this tree among the other trees of the same virtual router
    - Resource allocation among virtual routers is not “violated”
**System under Test**

- **Dell PowerEdge 2950 [SMP architecture]**
  - 2 Intel Xeon X5355 CPUs (quad-core @2.66GHz)
    - 32K L1d, 32K L1i, 4M L2 shared d&i cache (shared by 2 cores)
  - 8*1GB system memory
    - PC-5300
  - 12 Gigabit Ethernet ports
    - On 3 NICs
    - Each NIC had 4 PCIe lanes
    - Intel 82571EB controllers
  - Software
    - Click in Linux kernel
    - Xen VMM/Openvz
    - Oprofile, etc.
• 1 virtual router
• 2-6 active trees, each on assigned to a separate core
• 64-byte packets
- 6x6 virtual router
- 1-4 active trees on assigned to a single core
- 64-byte packets
• 2 2x2 virtual routers with different priorities (2 to 1)
• Col. 1: all trees are active, fairness is attained due to ticket allocation
• Col. 2: 1 tree is inactive for VR1 with ticket mitigation, fairness is attained
• Col. 3: 1 tree is inactive for VR1 without ticket mitigation, VR2 steals resources from VR1
Conclusion

• Modern commodity HW is a viable platform for well-performing software virtual routers

• Forwarding paths with trusted elements
  – Adequate isolation and fairness
  – Negligible performance loss

• Forwarding paths with untrusted elements
  – Co-exist with trusted forwarding paths
  – Some performance drop for the untrusted forwarding path

• HW assistance and proper mechanisms are needed to overcome the novel system issues
  – Multi-queue NIC
  – Multi-queue support in the virtualization technique
  – NUMA

• Can we use it for more than being a single Virtual Router?
Project page, including papers
http://nrg.cs.ucl.ac.uk/vrouter/