

Component-based system for management of multilevel virtualization of networking resources

Robert Boczek

Dawid Ciepliński

April 5, 2011

Contents

1	Introduction	7
2	Context	9
2.1	QoS-aware networking	9
2.2	Resource virtualization approaches	9
2.3	Multilevel network virtualization	9
2.3.1	Virtual network resources	9
2.3.2	Fine-grained QoS control	9
2.3.3	Virtual appliances	9
2.3.4	„Network in a box” concept	9
2.4	Applications and benefits of virtual infrastructures	9
2.4.1	Testing and simulations	9
2.4.2	Improving server-side infrastructure scalability	9
2.4.3	Infrastructure as a service	9
2.4.4	The role of resource virtualization in the SOA stack	9
3	Requirements analysis	11
3.1	Functional requirements	11
3.1.1	Instantiation	11
3.1.2	Discovery	11
3.1.3	Accounting	11
3.2	Non-functional requirements	11
3.3	Underlying environment characteristics	11
3.4	General approach and problems it imposes	11
3.4.1	Load balancing / Deployment	11
3.4.2	Infrastructure isolation	11
3.4.3	Broadcast domain preservation	11
3.4.4	Constraints	11
4	Solaris OS as a resource virtualization environment	13
4.1	General information	13
4.2	Lightweight OS-level virtualization with Solaris Containers	13
4.3	Crossbow - network virtualization technology	13
4.4	Resource access control	15

5	The system architecture	17
5.1	High-level design	17
5.2	System components and their responsibilities	17
5.2.1	Assigner	17
5.2.2	Supervisor	17
5.2.3	Worker	17
5.3	Crossbow resources instrumentation	17
5.4	Domain model and data flows	17
6	Implementation	19
6.1	Implementation environment	19
6.2	Domain model transformation details	19
6.3	Low-level functions access	19
6.4	Building and running the platform	19
7	Case Study	21
7.1	Clustered GlassFish	21
7.1.1	Scenario description	21
7.1.2	GlassFish cluster integration	21
7.2	Multimedia server	21
7.2.1	Scenario description	21
7.2.2	Resource access requirements	21
7.2.3	Providing tunable and scalable virtual infrastructure	21
8	Summary	23
8.1	Conclusions	23
8.2	Achieved goals	23
8.3	Further work	23

Division of labour

Chapter 1

Introduction

Chapter 2

Context

Chapter overview

2.1 QoS-aware networking

2.2 Resource virtualization approaches

2.3 Multilevel network virtualization

2.3.1 Virtual network resources

2.3.2 Fine-grained QoS control

2.3.3 Virtual appliances

2.3.4 „Network in a box” concept

2.4 Applications and benefits of virtual infrastructures

2.4.1 Testing and simulations

2.4.2 Improving server-side infrastructure scalability

2.4.3 Infrastructure as a service

2.4.4 The role of resource virtualization in the SOA stack

Summary

Chapter 3

Requirements analysis

Chapter overview

3.1 Functional requirements

3.1.1 Instantiation

3.1.2 Discovery

3.1.3 Accounting

3.2 Non-functional requirements

3.3 Underlying environment characteristics

3.4 General approach and problems it imposes

3.4.1 Load balancing / Deployment

3.4.2 Infrastructure isolation

3.4.3 Broadcast domain preservation

3.4.4 Constraints

Summary

Chapter 4

Solaris OS as a resource virtualization environment

Chapter overview

4.1 General information

4.2 Lightweight OS-level virtualization with Solaris Containers

4.3 Crossbow - network virtualization technology

One of the most important condition in terms of network virtualization is that network traffic should be insulated between virtual machines. This kind of isolation can be achieved by having a dedicated physical NIC, network cable and port from the switch to the virtual machine itself. Moreover, switch must also ensure sustainability on every port. In every other case virtual machines will definitely interfere between each other. In a particular case when we have to share physical NIC between virtual machines the most promising solution is to virtualize NIC hardware and the second layer of the OSI/ISO stack where sharing is fair and interferences will be avoided. These approach was adapted in the Crossbow architecture in OpenSolaris OS. Traffic separation is achieved by fundamental blocks of new architecture which are Virtual NICs (VNICs) created by dividing NIC into many VNICs. A VNIC can be created over NIC or Etherstub (more about them later) and be dynamically controlled by the bandwidth and CPU resources assigned to it. The crossbow architecture has introduced fully paralyzed network stack structure. Each stack could be seen as fully independent lane (without any shared locks, queues, and CPUs) therefore network isolation is guaranteed. Key concept is hardware classification performed by the NIC over which VNIC was created. Each lane has a dedicated buffer for Transmit (Tx) and Receive (Rx) ring. In case when load exceeds assigned limit packets must be dropped as it is wiser to drop them then to expend OS CPU resources.

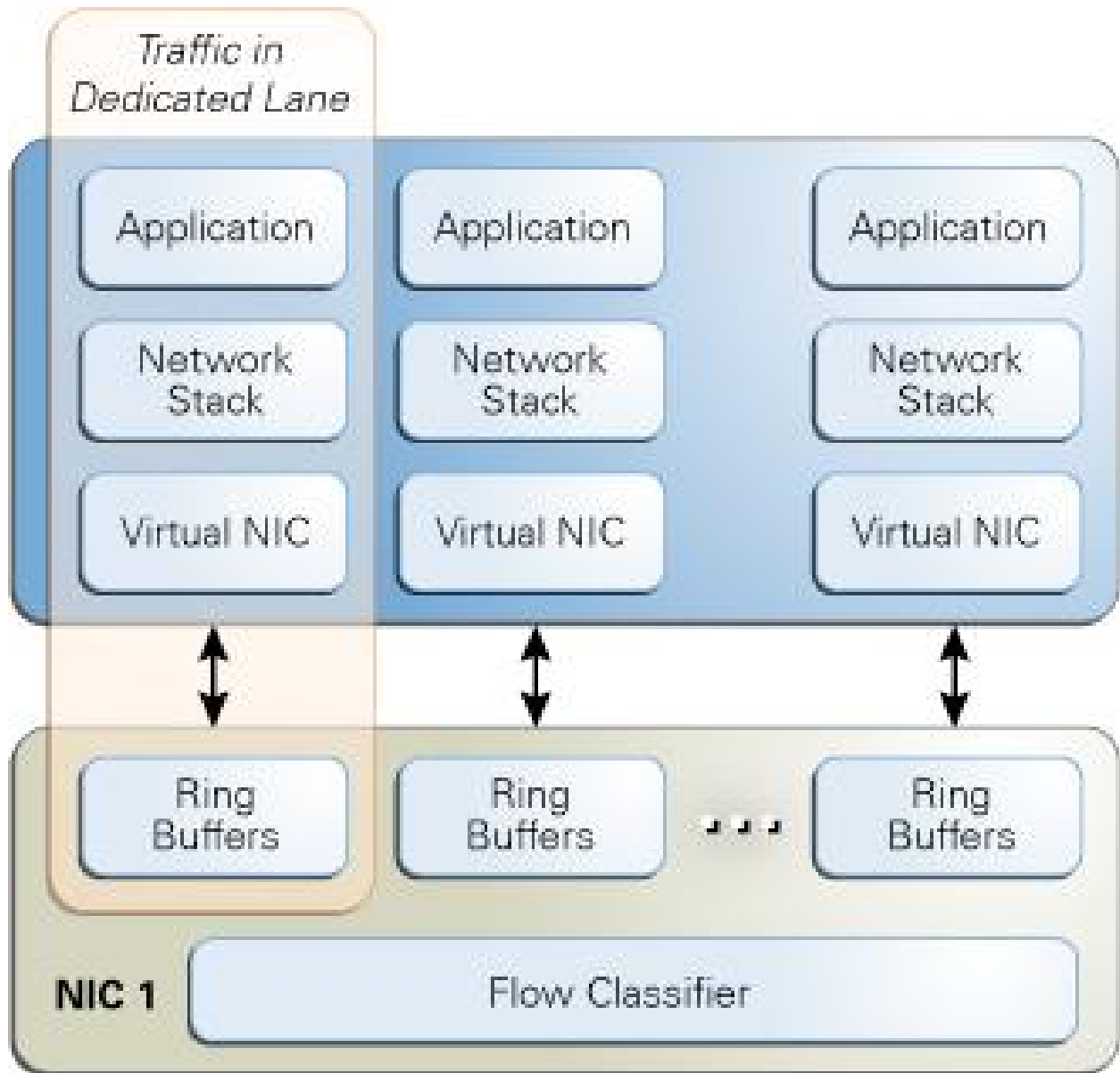


Figure 4.1: Dedicated lines in the Crossbow architecture

A virtualization lane can be one of two types, hardware-based or software-based. @todo dopisac Hardware-based virtualization lanes i Software-based virtualization lanes

@todo opisac szczegoly VNIC'ow

@todo opisac Dynamic Polling and Packet Scheduling

@todo opisac bandwidth partitioning

Etherstubs

As it was mentioned above, the MAC layer provides the virtual switching capabilities which allow VNICs to be created over existing physical NICs. In some cases, creating virtual networks without the use of a physical NIC is more welcomed than creating over physical NICs. In that case VNICs would be defined on the top of pseudo NICs. The Crossbow provides these kind of elements

which are called Etherstubs. These components could be used instead of NICs during creation of VNICs.

@todo examples of creating vnic and etherstubs **dladm** is the admin command for managing NICs, VNICs and Etherstubs. Below we present a few examples of creating VNICs, Etherstbus and how to assigned bandwidth and priority to theses elements.

1. `dladm create-vnic vnic1 -l e1000g0` - creates new VNIC **vnic1** over existing NIC **e1000g0**
2. `dladm create-etherstub ether00` - creates new Etherstub **ether00**
3. `dladm show-linkprop vnic11` - lists all properties assigned to **vnic11** link
4. `dladm set-linkprop -pmaxbw=1000 vnic11` - assigns 1Mbps bandwidth limit to **vnic11** link
5. `dladm set-linkprop -ppriority=low vnic11` - assigns low priority to **vnic11** link

Here we have just presented some basic commands. For more examples see **man dladm**

4.4 Resource access control

Summary

Chapter 5

The system architecture

Chapter overview

5.1 High-level design

5.2 System components and their responsibilities

5.2.1 Assigner

5.2.2 Supervisor

5.2.3 Worker

5.3 Crossbow resources instrumentation

5.4 Domain model and data flows

Summary

Chapter 6

Implementation

Chapter overview

6.1 Implementation environment

6.2 Domain model transformation details

6.3 Low-level functions access

6.4 Building and running the platform

Summary

Chapter 7

Case Study

Chapter overview

7.1 Clustered GlassFish

7.1.1 Scenario description

7.1.2 GlassFish cluster integration

7.2 Multimedia server

7.2.1 Scenario description

7.2.2 Resource access requirements

7.2.3 Providing tunable and scalable virtual infrastructure

Summary

Chapter 8

Summary

Chapter overview

8.1 Conclusions

8.2 Achieved goals

8.3 Further work

Bibliography

- [1] Crossbow: From Hardware Virtualized NICS To Virtualized Networks,
<http://conferences.sigcomm.org/sigcomm/2009/workshops/visa/papers/p53.pdf>
- [2] Virtual switching in Solaris, <http://hub.opensolaris.org/bin/download/Project+crossbow/Docs/virtual-switching-in-solaris.pdf>
- [3] Oracle Solaris 11 Express Network Virtualization and Network Resource Management,
<http://www.oracle.com/technetwork/articles/servers-storage-admin/sol11ecrossbow-186794.pdf>