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

Robert Boczek Dawid Ciepliński

May 9, 2011

Contents

Division of labour

Chapter 1

Introduction

Chapter 2

Context

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

4.3.1 Crossbow architecture

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 paralized 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 dop them then to expend OS CPU resources.

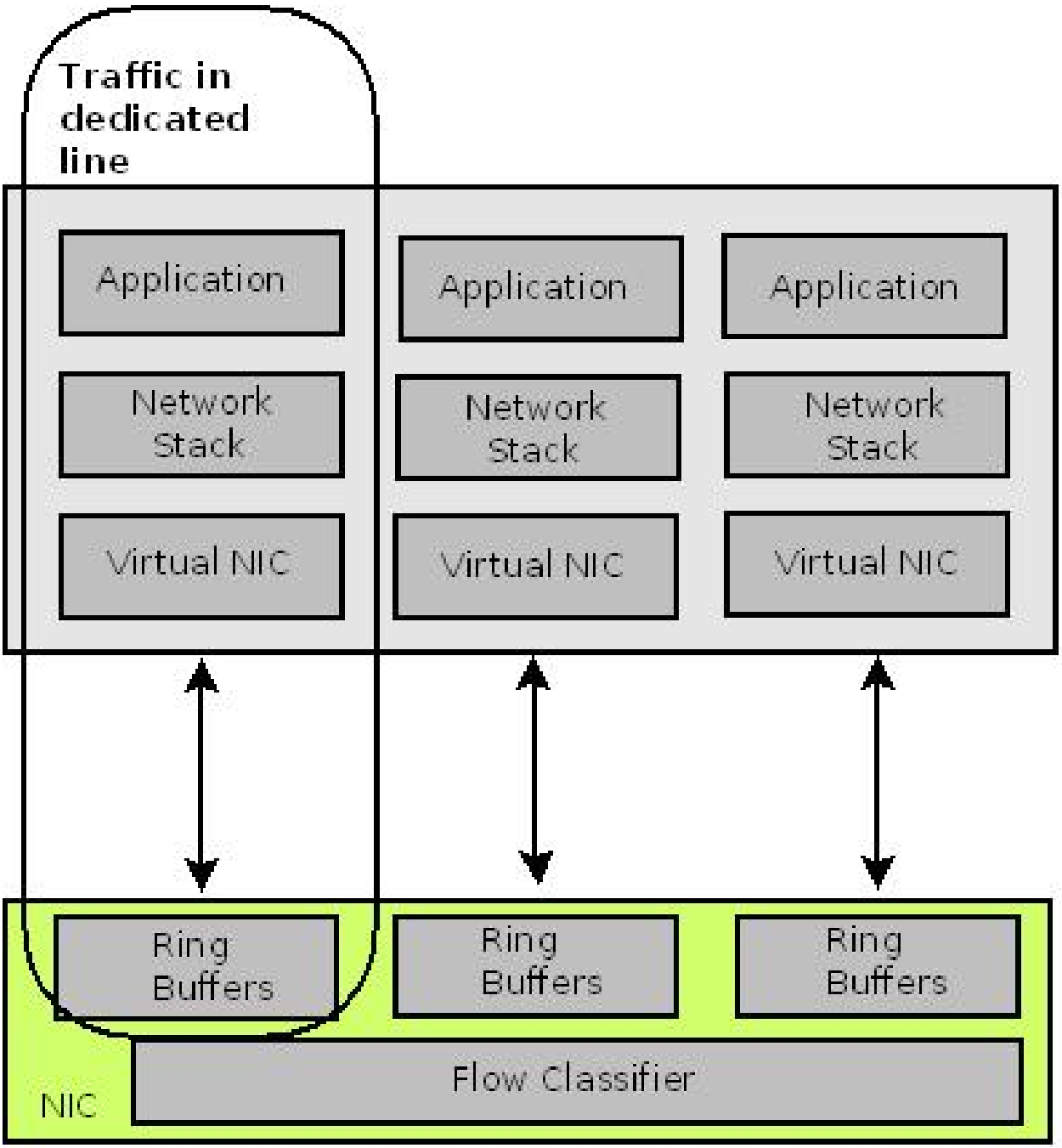

Figure 4.1: Dedicated lines in the Crossbow architecture

4.3.2 Virtualization lines

Virtualization is the most key component in the Crossbow architecture. Each lane consists some used to some concrete type of traffic. It usually would be composed of:

1. NIC resources(receive and transmit rings, interrupts, MAC address slots)

2. Driver resources(DMA bindings)

3. MAC layer resources (data structures, execution threads, locks)

A virtualization lane can be one of two types, hardware-based or software-based.

Hardware-based virtualization lanes

This type requires ability to partitioning resources from NIC. The minimum requirement is that a hardware-based lane should must have a dedicated receive ring. Other resources such as transmit lane can be exclusive or shared between lanes. Each virtual machine could have one or more lanes assigned and the incoming packets would be distributed among them based on even scheduling unless some administrative polices where created, such as priority or bandwidth limit.

Software-based virtualization lanes

In case when NIC runs out of hardware-based virtualization lane, receive and transmit rings may be shared by multiple VNICs. The number of software-based virtualization lanes also often called softrings is unlimited. The main disadvantage of software-based lanes is the lack of fairness and isolation which in fact is provided in hardware-based lanes. The received and sent rings may work also in mix mode, whereas some of the rings may be assigned to software and some may be assigned to hardware based lanes.

4.3.3 Dynamic polling

The Crossbow architecture proposed two types of working mode. Currently used mode is determined by traffic and load. Under low load, where the rate of arraving packets is lower than time of packet processing, lane works in the interrupt mode which means that receive ring generates an interrupt when new packet arrives. However, when the backlog grows, the line switches to dynamic polling mode in which a kernel thread goes down to the receive ring in the NIC hardware to extract all outstanding packets in a single chain. Key aspect is that every virtualization lane works independently and transparently from each other. Usually only three threads are used per lane:

1. Poll thread which goes to the NIC hardware to get all packet chain

2. Woker thread which is responsible for protocol processing (IP and above) or delivers packets to virtual machine. Thread performs also any additional transmit work which is a natural requirement some concrete protocol, such as processing TCP packets that require sending ACK packets.

3. Transmit thread that is activated when if packets are being sent after transmit side flow control relief discharge, or after retreiving transmit descriptor. Application or virtual machine can transmit any packets without performing queuing because of flow control or context switching.

4.3.4 Crossbow elements

VNics

Virtual NICs (VNICs) that are related to their own lane are the key element in crossbow architecture. There is no difference between NIC and VNIC in administration, they are all treated as data links.

Every VNIC has an assigned lane and flow classifier which classifies received packets by VNIC's MAC address and sometimes by the VLAN tag. If created with a VLAN tag, protocols like GVRP or MVRP may be used to register the VLAN tag with the physical switches too.

In terms of sharing bandwidth, Crossbow enables administrative control of bandwidth for every single VNIC. The bandwidth of the link is implemented by regulating the periodic intake of incoming packets per dedicated lane. The network stack allows only as many packets as it was assigned to specific VNIC. The lane picks more packets when the next period begins. In case of regulating the speed of transmited bandwidth it is much easier as the network stack can either control the application that is generating the stream of packets or just drop the excessive amount of packets. These mechanisms are also used in flows QoS described and discussed later in this paper.

Virtual switching

Virtual switches created always implicitly when the first VNIC is defined under existing NIC could never be accessed directly nor be visible by any user (even administrator).

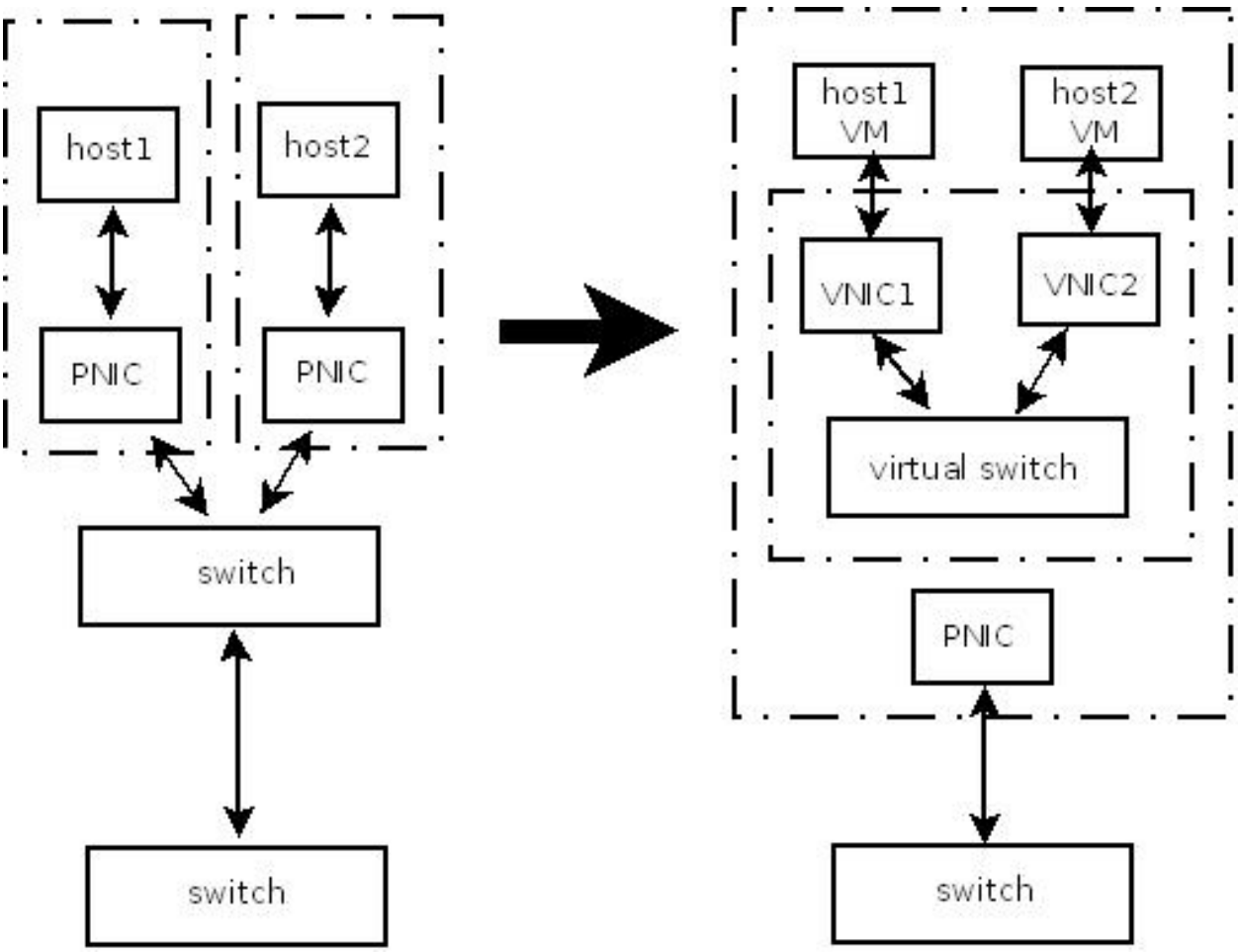

Figure 4.2: Mapping between physical and virtual network building elements

Semantics assured by virtual switches are the same as provided by physical switches:

1. VNICs created on top of the same NIC can send ehterstub packets to each other
2. Broadcast packets received by the underlying NIC are distributed to every single VNIC that was defined on the top of this NIC
3. Broadcast packets sent by one of the VNICs is distributed to all VNICs defined on the top of the same NIC and to the NIC for further transmission as well
4. In terms of multicast network traffic multicast group membership is monitored and used for distributing packets to appropriate VNIC

Connectivity between VNICs is available only when they were defined on the top of the same NIC.

Etherstubs

As it was mentioned before, 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.

Examples

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 - assignes 1Mbps bandwith limit to **vnic11** link
5. dladm set-linkprop -ppriority=low vnic11 - assignes 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

The Domain model and data flows section describes the transformations performed by the system's components in order to instantiate/deploy an object model. These include simple one-node instantiation as well as more complex multi-node instantiations.

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]

[2]

[3]

[4]