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

Robert Boczek Dawid Ciepliński

July 14, 2011

Contents

Chapter 1

Introduction

In today's world every successful organisation is based on properly designed communication network. These networks must deal with delay-sensitive data such as video images, real-time voice or mission-critical data. Therefore must provide safe, predicatable and sometimes guaranteed services. Accomplishing the required Quality of Sevice(QoS) by controlling the delay, delay variation(jitter), bandwidth, packet loss parameters is deeply hidden secret of most successful end-to-end business applications. http://www.cisco.com/en/US/products/ps6558/products_ios_technology_home.html

Due to raising concern and importance of these issues in these paper we decided to have more insight into one of possible approaches to this matter which is Solaris OS and the Crossbow technology.

Chapter 2

Context

Constantly growing demand for bandwidth in networks (especially VoD, VoIP, RT) raises the question: 'Whether it is better to increase available bandwidth of networks or to build intelligent systems managing users traffic?'. As high bandwidth is just not enough because there are other equally important transmission parameters (delay, jitter, package missing tolerance) there seems to be just one correct answer for this question. Creating such systems is not an easy task and many groups such as IETF (The Internet Engineering Task Force) are working on this problem. Currently there are three existing models performing QoS in the IP network:

1. best effort,
2. Intserv,
3. Diffserv.

These models are discussed in more details in the following section.

2.1 QoS-aware networking

QoS (Quality of Service) is an issue in all types of networks such as IP, Token Ring, Frame Relay or even ATM. Each of them adapted specific approach towards this matter. IP network for instance is non-deterministic although there is a possiblity of packet classification using CoS (Class of Service) field. Token Ring on the other hand is deterministic and allows using priority to distinct traffic. Last but not least ATM creates virtual path between sender and receiver and sets QoS parameters. Despite the fact that all approaches are very interesting and meaningful this paper focuses mainly on the IP network and its approach to QoS.

Nowadays the IETF (The Internet Engineering Task Force) is working on two approaches beyond the basic best-effort service to provide more advanced handling of packets and providing requested level of bandwidth and delay which are:

1. integrated services - reserves resources necessary to provide the service along path,
2. differentiated services - do not require resource reservation along path

2.1.1 DiffServ

Due to clear need for relatively simple and coarse methods of providing differentiated classes of service for Internet traffic, to support various types of applications, and specific business requirements. The differentiated service approach to providing quality of service in networks employs a small, well-defined set of building blocks from which a variety of aggregate behaviors may be built. [5]

DiffServ works with traffic stream containing many complex microflows which among the same stream have:

- Same QoS requirements,
- Traverses domain towards the same direction.

Microflow is a flow between applications and is identified mainly by:

1. Source and/or Destination Address,
2. Transport protocol,
3. Source and/or Destination Port.

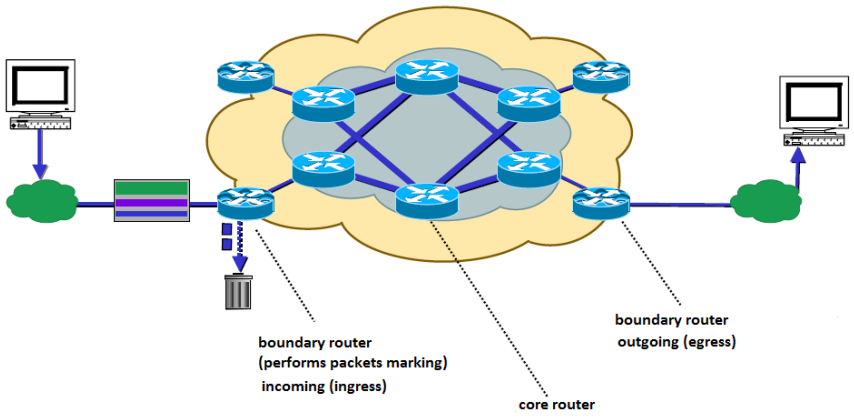

Figure 2.1: DiffServ domain

DiffServ domain is a set of nodes with consistent QoS policy (for example network managed by the same ISP or intranet). Routers within domain are didvided into two groups: boundary routers and core routers. Boundary router performs traffic classification, packet marking, traffic metering, traffic control (policing, shaping) where core routers pass packets according to PHB(Per Hop Behaviour - more about them later) and sometimes changes DSCP labels.

This architecture distinguishes two major components: packet marking using the IPv4 ToS byte or TC in IPv6 and PHBs(Per Hop Behaviour). Redefined ToS field now uses 6 bits for packet classification and is called Differentiated Services Codepoint(DSCP). PHB defines how packet should be treated due to its priority. Accepted policy should be consistent within all domain.

2.1.2 IntServ

Integrated services beyond the basic best-effort service defined in two RFCs provide two levels of service for IP which are called Controlled-Load Service and Guaranteed Service. Both require information about the traffic to be generated.

Guaranteed Service type should be used in applications with real time demands as it provides guaranteed bandwidth and delay, whereas Controlled-Load Service does not give full guarantee and should be used with application less sensitive for packets loss or delay.

IntServ requires resource reservation for certain flows or aggregated data streams. The reservation operation is available thanks to RSVP protocol. Due to this reservation session initialization lasts much longer than in DiffServ.

DiffServ vs IntServ

Table 2.1: DiffServ, IntServ comparison

IntServ	DiffServ
Stateful	Stateless
Not scalable	Scalable
Stream oriented	Processing single packets

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

The IAAS(Infrastructure as a service) sometimes also called Hardware as a Service (HaaS) is one of three cloud computing models, the other two are: Software as a Service (SaaS) and Platform as a Service (PaaS). This service is based on providing by the supplier whole scalable IT infrastructure depending on user demand such as virtualized hardware. The service provider owns the equipment and is responsible for housing, running and maintaining it.

At the beginning the IaaS was just renting dedicated servers services from supplier. Nowadays thanks to virtualization these are most often virtual machines. In the former case user paid for the concrete hardware (box), now client typically pays on a per-use basis.

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 10, Solaris 11 and OpenSolaris

The chapter provides an overview of Oracle Solaris operating system and evaluates it as a platform for resource virtualization. The chapter describes Solaris 11 Express release of the system, as it is the first release (together with OpenSolaris) with Crossbow technology integrated. Special emphasis is put on the networking-related aspects of virtualization. Thus, the Solaris Crossbow technology is described in detail.

Section 4.1 contains introductory information about the system. A short historical note is presented and general description follows. Main components of the system are introduced and described.

Each of the remaining sections describe in more detail these parts of the operating system that are extensively used by the implemented system. Section 4.2 investigates the Solaris Zones technology. After defining the concept of zones, zone lifecycle model is presented, the achieved level of process isolation is described and discussion of Zones advantages in comparison to non-virtualized environments follows.

Section 4.3 introduces Solaris Crossbow - lightweight network virtualization environment. The section starts with general description of the technology. Next, components crucial to efficiency improvement are presented in detail. Etherstubs, VNICs and flows are described. These are building blocks used to create virtualized network elements and apply QoS policies. The section ends with the comparison between Crossbow and DiffServ and a method of integration of these two solutions is presented.

Section 4.4 provides an overview of resource control methods offered by the Solaris OS. The types of resource management mechanisms (constraints, partitioning and scheduling) are identified and defined. Resource control hierarchy used by the system is depicted and explained. Also, the accounting facility is described. The types of resources extended accounting can work with are enumerated and examples of data that can be gathered are listed.

4.1 General information

Oracle Solaris is a *multiuser, multitasking, multithreading UNIX-like operating system* [17]. Since its release in 1992 (as Sun Solaris 1), the system became one of the most popular environments supporting enterprise software. Nowadays, big corporations and companies as well as individual developers use it to do their business and deliver reliable and scalable services.

The Solaris OS provides unique set of tools that support virtualization of practically all types

of resources at various levels. There is Logical Domains (LDOMs) technology for full virtualization and lightweight Zones, when all that is needed is the isolation of processes. Logical domains can be connected with complex virtual networks that are created with virtual switches (vsw) and virtual network devices (vnet) [12] and Crossbow can be used to enable lightweight and efficient networking for zones, exploiting capabilities of underlying hardware layer (network interface cards with virtualization level 1, 2 or 3 [8]).

Resource utilization can be managed with integrated administration tools. Resource access policies can be created with high level of granularity (per-process resource control) as well as in more general way (limiting resource access for LDOMs). Resource consumption can be subject of monitoring and accounting. With extended accounting subsystem enabled, it is possible to capture detailed accounting data even for single processes. Gathered data include CPU usage, number of bytes received or transmitted per DiffServ or Crossbow flow and more.

As far as multiple physical machines are considered, there is also support for VLANs (Virtual Local Area Network). Thanks to VLAN tagging support, it is possible to build systems that guarantee the quality of service from the lowest levels up, even for services belonging to different systems and consolidated within single physical machine.

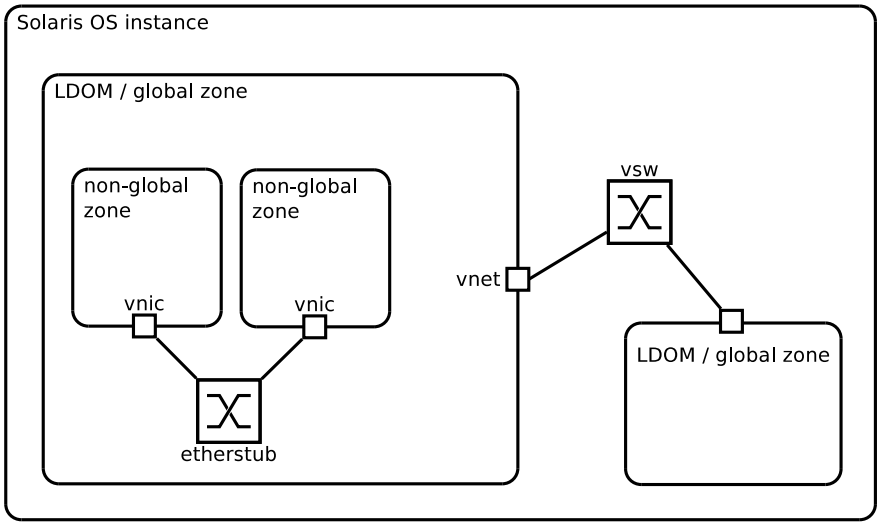

Figure 4.1: The variety of resources that can be virtualized with Solaris OS

As it can be seen, the Solaris operating system is accompanied by vast variety of virtualization-supporting subsystems. This multiplicity and flexibility makes it a promising platform for service provisioning and building even more abstract architectures on top of it. The following sections describe selected aspects of the system in more detail.

4.2 OS-level virtualization with Solaris Containers

The concept of lightweight (OS-level) virtualization is supported by most modern operating systems. The solutions are either integrated into the system's kernel and accessible as soon as it is installed (Solaris Containers, AIX Workload partitions, BSD jails [7]) or are provided by third-party

manufacturers as kernel patches and utility software (OpenVZ and LXC for Linux OS). Because of awareness of other system components and integration with them, it can be expected that Zones have more potential than other virtualization methods.

4.2.1 General information

Zones technology was introduced as of Solaris OS 10. It provides a way of partitioning system resources and allows for isolated and secure application execution environment [14]. Solaris Zones, together with resource management functionality, constitute the Solaris Container environment.

There are two types of zones: global and non-global. Global zone is the default one and is used to execute applications as well as to administer the system. Non-global zones can be created from within the global zone only. A single operating system instance with one global zone can host as many as 8192 non-global zones [14].

Zones can be assigned system resources such as CPU capacity, the amount of random-access memory or even maximum number of lightweight processes that can be running simultaneously. Also, network isolation is supported at two levels: basic, at the IP layer, and network isolation and advanced virtualization with fine grained quality of service control using the Crossbow technology.

Each zone can run a different set of applications, with optional translation of system calls (Branded Zones Technology) thus emulating different operating environments [14]. The user is able to create a branded zone with translation of Linux system calls and run Linux-specific applications without code recompilation.

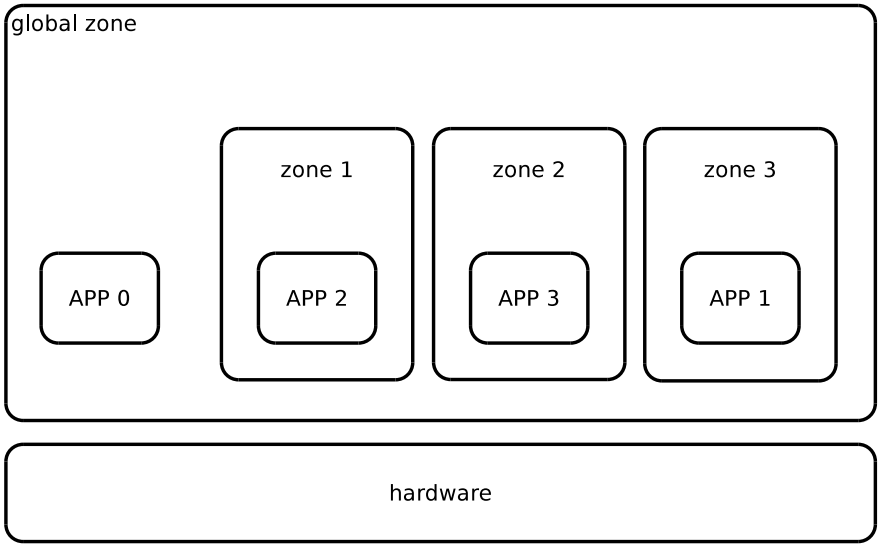

Figure 4.2: Solaris Zones high-level view

4.2.2 Zone lifecycle

A model was created to describe the states in which each zone must exist and its possible transitions. A non-global zone can be in one of six states: *configured*, *incomplete*, *installed*, *ready*, *running*, *shutting down or down* [14]. Figure 4.3 depicts the model.

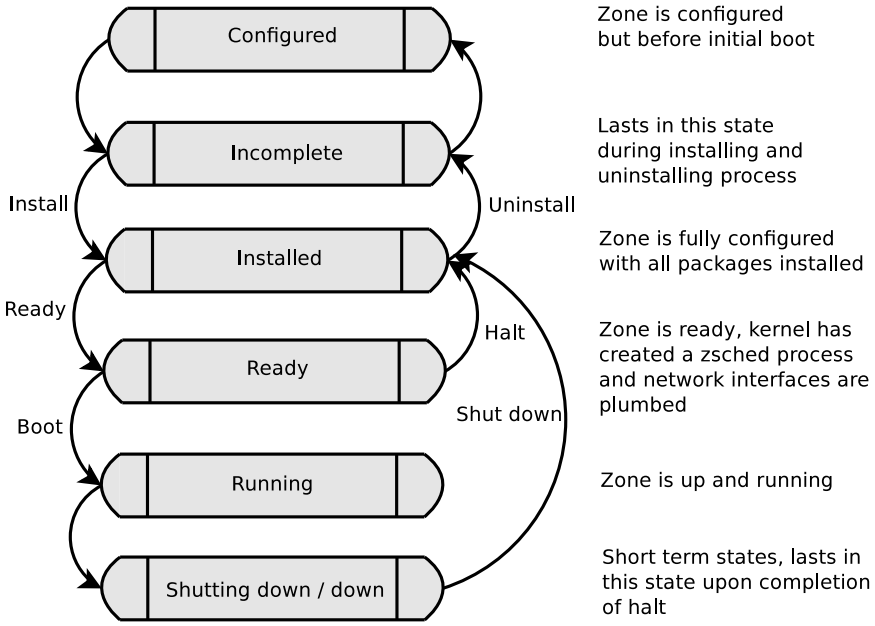

Figure 4.3: Zone states and possible transitions

4.2.3 Isolation of processes

The Containers environment offers a high level of application security and isolation. This is accomplished by imposing software bounds on the resource usage and introduction of additional abstraction layer over hardware.

Every process and its children are bound to concrete zone and the assignment cannot be changed. Moreover, it is impossible for processes in distinct zones to monitor each other operation. They are not visible to each other and no interprocess communication can take place, except for network-based one, if enabled by the administrator.

Because of the isolation, an application failure possibly affects only the processes in the containing zone. Assuming no interaction between processes in separate zones, the rest of the system remains intact and can operate normally.

4.2.4 Advantages of Containers technology when compared to non-virtualized environments

The architecture of Solaris Containers makes it a competitive solution as far as systems administration and operation efficiency is concerned. The technology, imposing negligible overhead [10], allows to perform tasks that would be impossible or very hard to accomplish if traditional setup is used. Examples of such tasks include dynamic resource assignment, instantaneous cloning and migration of systems between physical nodes.

The technology allows for running a number of isolated instances of operating system sharing CPU time, physical network bandwidth, filesystem contents and binary code. Sharing of these resources can greatly improve overall system efficiency and reduce the amount of occupied memory.

The speed of network communication between different zones can also be improved thanks to „short-circuited" traffic (i.e. omitting the layers below IP in the OSI/ISO stack). The instances are able to execute applications with minimum overhead introduced mainly due to accessing commands and libraries through the lofs (loopback filesystem) [10, 13].

When using file system that supports snapshots (as, for example, ZFS), zones can be serialized (a snapshot of the file system can be taken) and sent over the network connection or other means of data transfer to another machine. There, the zone can be restored and operate as a part of the host system.

Another important aspect of building the infrastructure with containers is resource control. The Solaris system makes it possible to define resource controls (rctls) at various levels, also on per-zone basis. CPU shares, maximum number of lightweight processes and maximum swap size are examples of resource control properties that can be set for a zone. This can be further extended by providing fine-grained properties at project, task and process levels [14]. The resource control process is dynamic - assignments can be changed as the system is running, without interrupting the container's normal operation. This can be of extreme importance as far as high-availability systems are considered.

Containers facilitate service consolidation - all components of a system can be executed in a single machine with network-based communication handled entirely by the host operating system, thus eliminating the need for additional networking hardware and its management. The consolidated infrastructure becomes more flexible as the majority of administration tasks can be performed by issuing a series of terminal commands. All these factors make total cost of ownership lower [10].

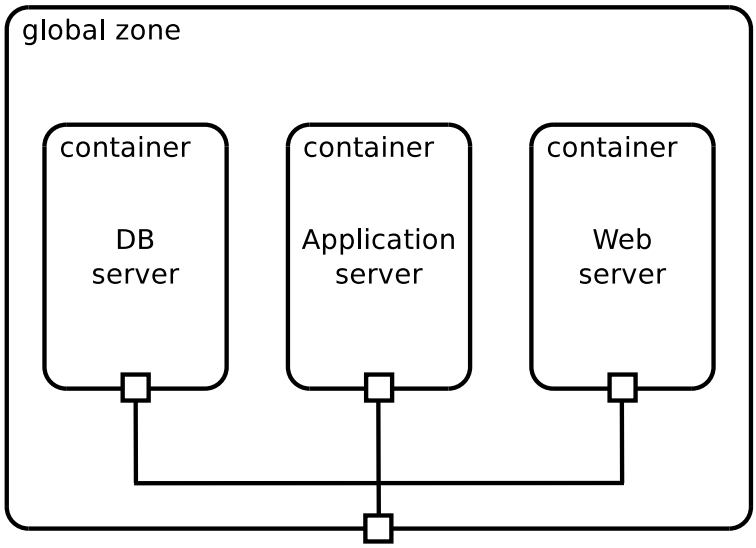

Figure 4.4: Service consolidation within a Solaris OS instance with internal network connectivity

4.2.5 Virtual appliances

Virtual appliance is a *pre-built, pre-configured, ready-to-run (enterprise) application packaged along with an optimized operating system inside a virtual machine* [11]. Solaris Zones, together with other components of the Solaris OS, constitute a complete framework that implements virtual appliance

approach to systems management.

The main problem virtual appliances can solve is the complexity and duration of application deployment process. In general, a service deployment can be described as comprising the following stages: preparation (learning the dependencies), pre-installation, installation and post-installation. With traditional (non-virtualized) approach, these stages have to be repeated every time a service is deployed on different machines.

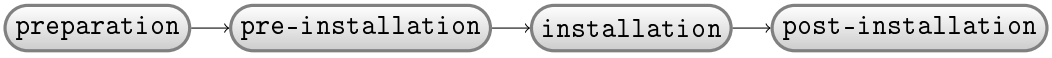

Figure 4.5: Traditional application deployment stages.

Virtual appliance approach makes it possible to reduce deployment time significantly [11]. This is achieved by performing most of the deployment stages once and storing the configured environment in a virtual appliance. The appliance can then be moved to publicly-available repository for actual deployment on host systems.

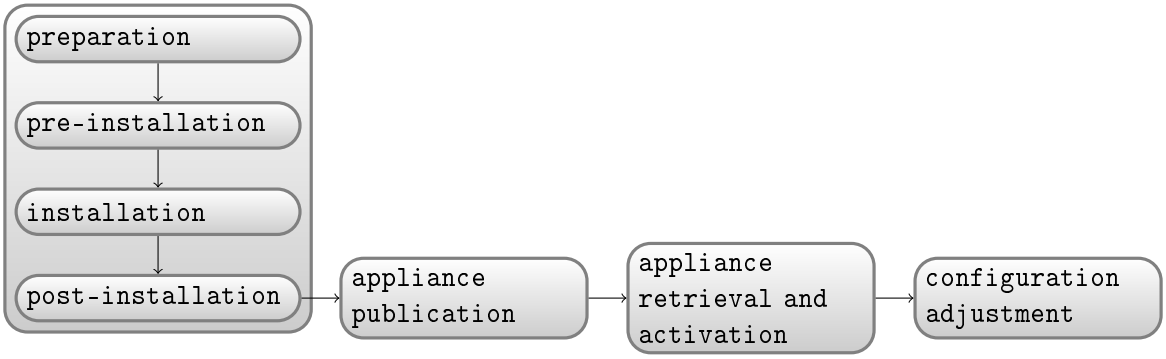

Figure 4.6: Deployment process with virtual appliances. Stage 1 is executed once.

It is possible to prepare sets of virtual appliances containing traditional services (such as application servers, database servers or media servers) as well as highly specialized networking-focused appliances that can act as routers, firewalls or load balancers. These Virtual (Network) Appliances, together with other components provided by Solaris OS, can be leveraged to build fully virtual network topologies.

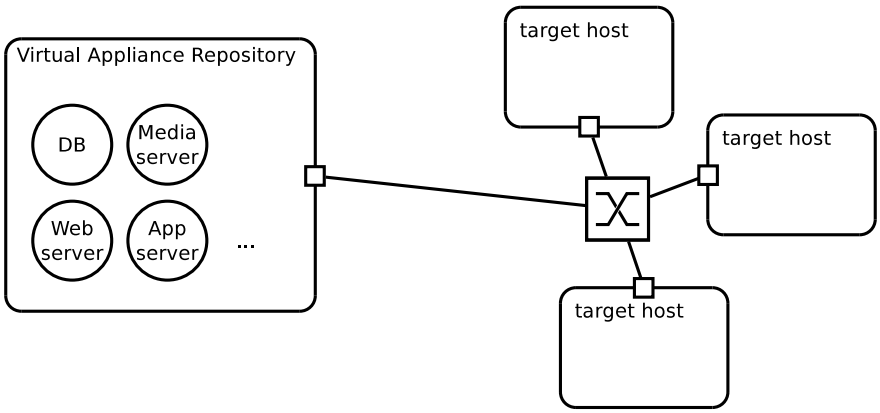

Figure 4.7: An example of infrastructure utilizing virtual appliances with appliance repository.

4.3 Crossbow - network virtualization technology

It is generally acknowledged that Crossbow was invented in China in 341 B.C but it was in middle ages when it earned its recognition. Very easy in use and simultaneously very effective. The Solaris Crossbow mechanism for QoS are just like real crossbows, very efficient in comparison to other existing QoS mechanisms and this similarity indicates the project name origin.

4.3.1 Crossbow architecture

One of the most important conditions 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. Otherwise, virtual machines will definitely interfere with each other [9].

In a particular case when a physical NIC has to be shared 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 interference will be avoided. These approach was adapted in the Crossbow architecture in the Solaris OS [9].

Traffic separation is achieved with fundamental blocks of new architecture which are Virtual NICs (VNICs) created by partitioning physical NIC. A VNIC can be created over NIC or Etherstub and be dynamically controlled by the bandwidth and CPU resources assigned to it [9, 4]. New architecture after introducing new networking features combined with existing features like Solaris Containers, resource control can be presented as following:

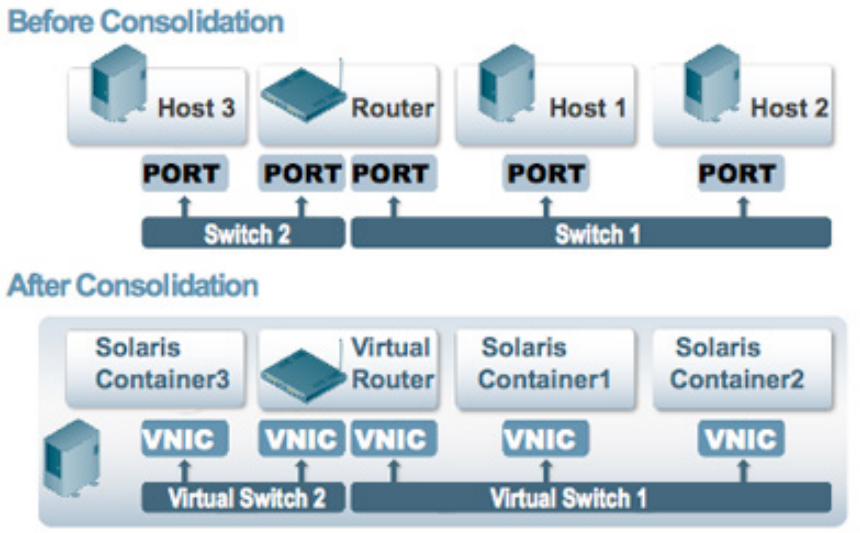

Figure 4.8: The Solaris Crossbow network virtualization enhancement, source: http://www.net-security.org/images/articles/crossbow.jpg

The crossbow architecture has introduced fully parallel network stack structure. Each stack could be seen as an 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 than to expend OS CPU resources [9].

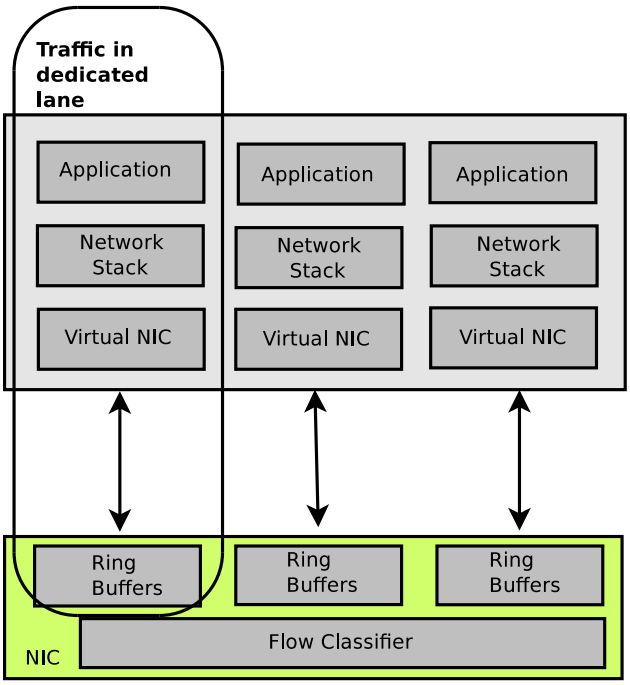

Figure 4.9: Dedicated lanes in the Crossbow architecture

4.3.2 Virtualization lanes

Virtualization lane is the most key component in the Crossbow architecture. Each lane consists of some dedicated hardware and software that might be used to handle specific 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 [9].

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 [9].

4.3.3 Dynamic polling

The Crossbow architecture proposed two types of working modes. Currently used mode is determined by traffic and load. Under low load, where the rate of arriving packets is lower than time of packet processing, a 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 [9]:

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

2. Worker 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 retrieving transmit descriptor. Application or virtual machine can transmit any packets without performing queuing because of flow control or context switching.

4.3.4 Virtual switching

Virtual switches are always created implicitly when the first VNIC is defined under existing NIC and could never be accessed directly nor be visible by any user (even administrator) [2].

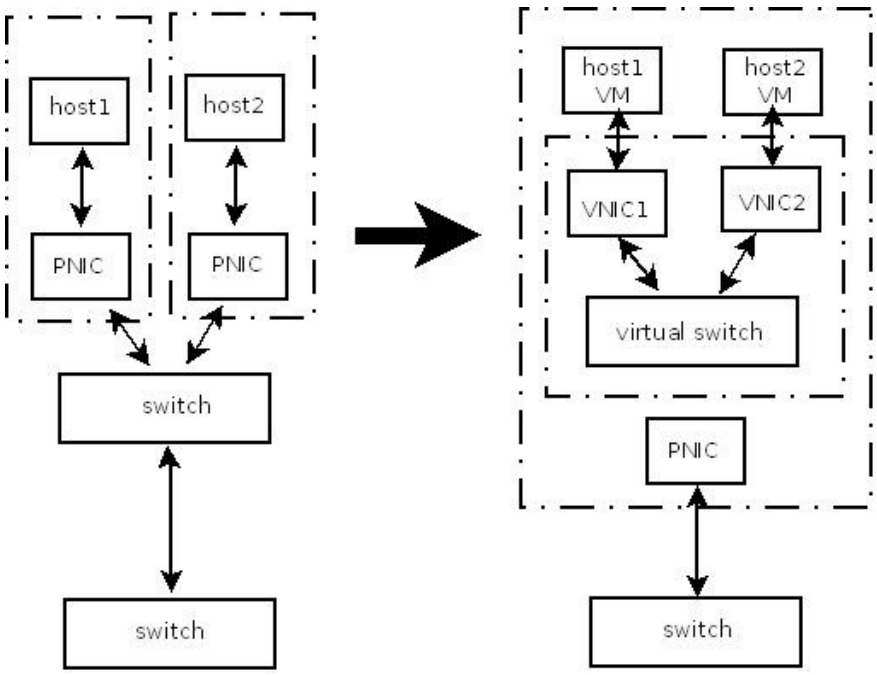

Figure 4.10: Mapping between physical and virtual network building elements

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

1. VNICs created on top of the same NIC can send 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.

Data Link Layer connectivity between VNICs is available only when they were defined on top of the same NIC.

4.3.5 Crossbow components

The Crossbow specification describes three major components: VNICs, Etherstubs and Flows. This section gives an insight into their application and usage.

VNICs

Virtual NICs (VNICs) each containing their own lane are the key element in crossbow architecture. There is no difference between NIC and VNIC in administration, as 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 [9].

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 [9].

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 [9].

Flows

Flows are additional instruments created to allow easier network traffic administration. They might be used in order to provide bandwidth resource control and priority for protocols, services, containers. Virtual networks can be described to maintain isolation and different network properties, and define flows to manage quality of service [4].

Defined flow is a set of attributes based on Layer 3 and Layer 4 headers of the OSI/ISO model which are then used to identify protocol, service or virtual machine. Flows assigned to a link must be independent therefore before adding new one its correctness is checked. Input and output packets are matched to flows in very efficient manner with minimal performance impact.

Crossbow flows can be created with one of the following sets of attributes:

- Services (protocol + remote/local ports),
- Transport (TCP, UDP, SCTP, iSCSI, etc),
- IP addresses and IP subnets,
- DSCP field.

For each flow the following properties can be set [15]:

- bandwidth,
- priority.

flowadm is the console command used to create, modify, remove or to display network bandwidth and priority limits assigned to a particular link.

4.3.6 Running examples of flowadm and dladm command

dladm and **flowadm** are two basic administrative commands for dealing with the Crossbow's components. Below a few general examples of their usage are presented.

dladm is the admin command for crossbow datalinks elements management. Below a few examples of VNICs, Etherstubs management commands are presented and how bandwidth and priority values might be assigned to these 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.

More examples can be found in **man dladm**.

flowadm is the admin command for flow management. It might be used as follows:

1. # flowadm show-flow -l e1000g0 - displays all flows assigned to link **e1000g0**,
2. # flowadm add-flow -l e1000g0 -a transport=udp udpflow - creates new flow assigned to link **e1000g0** for all udp packets.

More information about **flowadm** and **dladm** tools can be found in manual.

4.3.7 Crossbow and Differentiated Services - interoperability

The Crossbow technology is designed to work inside single operating system instance, there are no mechanisms meant to cope with problems that arise when dealing with installations spanning multiple physical machines connected with traditional (non-virtual) network. Crossbow's flows are, by design, relatively simple (when compared to DiffServ) but more efficient as far as receive performance is considered [16]. Crossbow, unlike DiffServ, does not require special hardware, although if it is present it can boost overall operation performance [16].

DiffServ, on the other hand, provides sophisticated QoS mechanisms that require proper hardware (DiffServ-aware routers) to be present for it to work. DiffServ is standardized (RFC 2475) and offers a multiplicity of classification, marking, policing and traffic shaping alternatives [1]. Special fields (called DSCP) contained in IP packet's header are used to carry processing-related information with packets. The approach can be used with complex networks, comprising a number of routers with QoS awareness.

These two environments complement one another rather than compete. Crossbow supports flow matching based on the DSCP field value. DSCP field generation is planned but not yet supported. It is possible (although, at the moment, only partially) to integrate these and build a comprehensive end-to-end networking solution with QoS support and virtualized components.

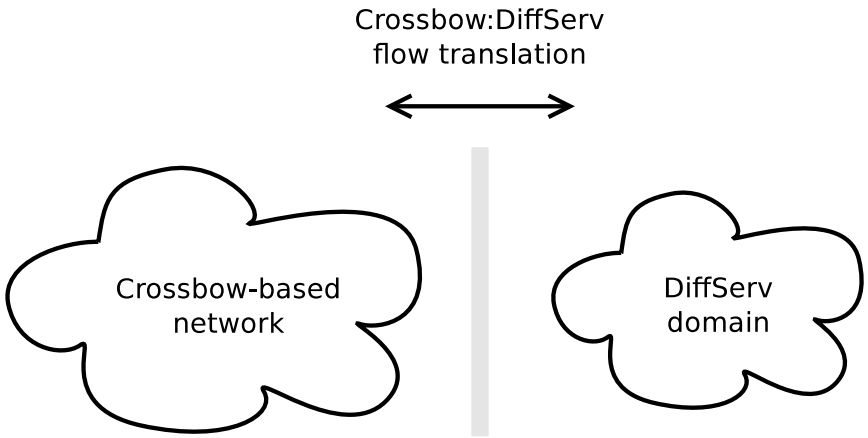

Figure 4.11: DiffServ integration using Crossbow-provided mechanisms

4.4 Resource control

Nowadays existing operating systems must provide mechanisms for response to the varying resource demands per workload which is an aggregation of processes of an application. By default resource management features are not used and system gives equal access to resources. When necessary, it is possible to modify this default behaviour with respect to different workloads. It is allowed to:

1. Restrict access to specific resource,
2. Offer resources to workloads on a preferential basis,
3. Isolate workloads from each another.

Resource is any part of computing system that may be modified in order to change application behaviour. Resource management enables more effective resource utilization and avoid wasting available ones due to load variability. Reserving additional capability during off-peak periods and effective sharing resources definitely increases application performance.

Most of the operating systems limited the resource control just to per-process control, whereas Oracle Solaris has extended this concept to the task, project and zone. Due to introducing granularity levels processes, tasks, and zones are efficiently controlled from excessive resource consumption. All these enhacements are available thanks to resource controls (rctls) facility [3].

Solaris Operating System introduced three types of resource management control mechanisms:

1. constraints - allows defining set of bounds on used resources for a workload,
2. partitioning - enables binding subset of system's available resources to specific workload,
3. scheduling - involves predictable algorithm making sequence of allocation decisions at specific intervals.

Hierarchical architecture allows defining set of resource control sets on each level. However, if more than one is assigned to a resource, the smallest container's control level is enforced [3].

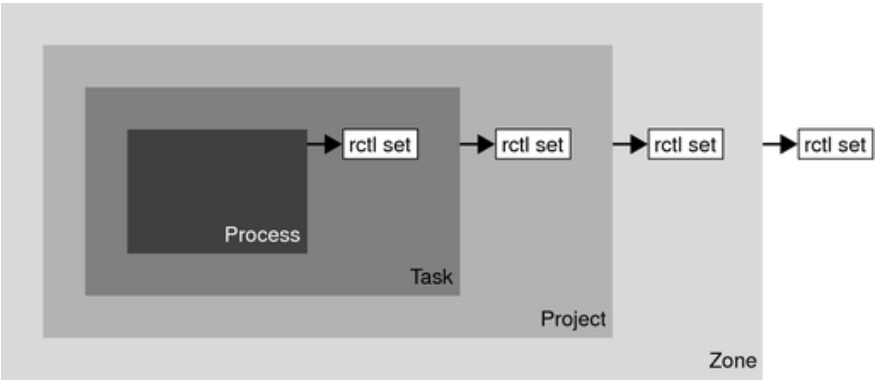

Figure 4.12: Solaris system multilevel architecture and its resource control sets (source: http://oracle.com)

4.4.1 Accounting

Highly configurable accounting facility is provided as part of the system. Its role is to gather historical usage records of system and network resources. There are two levels accounting can work on in Solaris OS - basic and extended. Basic accounting allows for per-zone and per-project statistics gathering while extended accounting facility makes it possible to collect the data for tasks and processes. Statistics gathered by the extended accounting can be examined using C or Perl interface of the libexacct library [14].

The extended accounting facility can gather data for:

- system resources usage (per-task and per-process),
- flows defined with the IPQoS tools,
- links and flows created with Crossbow.

Summary

The chapter presented Solaris operating system with regard to resource virtualization. The stack of tools integrated into the system provides extensive support for virtualization techniques: Containers facilitate OS-level resource virtualization and Crossbow, shipped with Solaris 11, makes virtualization of networking resources possible. Resource control subsystem gives the administrator even more fine-grained control over resource utilization. Last, but not least, accounting functionality provides detailed view of resource usage history.

The features mentioned above make realization of flexible, scalable and efficient systems possible. With these foundations, it is possible to build and consolidate complex network-oriented infrastructures that prove to be reliable, relatively easy to manage and adjust to changing requirements.

Solaris 10 OS seems to be ideal cross-platform choice for customers dealing with management of high level services, complex system administration and high costs. It is the only open operating system which has proven results running from every critical enterprise databases to high performance Web farms that is why Solaris OS is becoming strategic platform for today's constantly growing demands towards operating systems [6].

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.

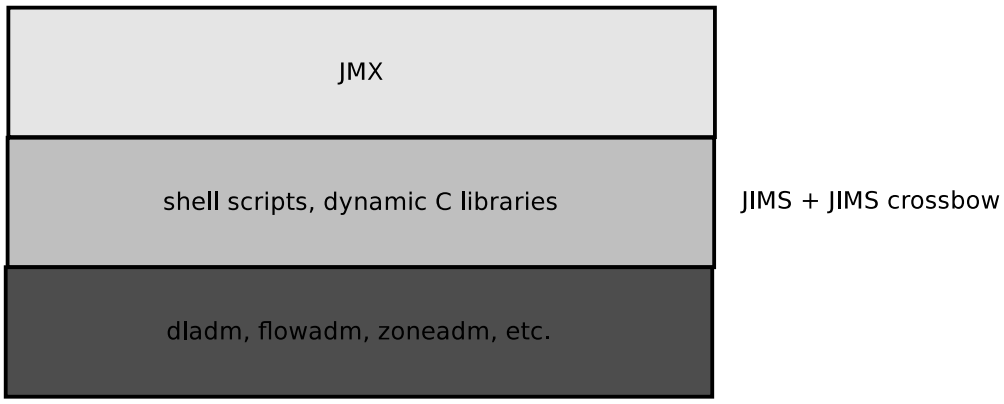

Figure 5.1: Layered system architecture

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

Created object model allows to describe network structure contaning of: switches, resources with attached addressable interfaces and policies. Figure below presents concrete classes and their dependencies.

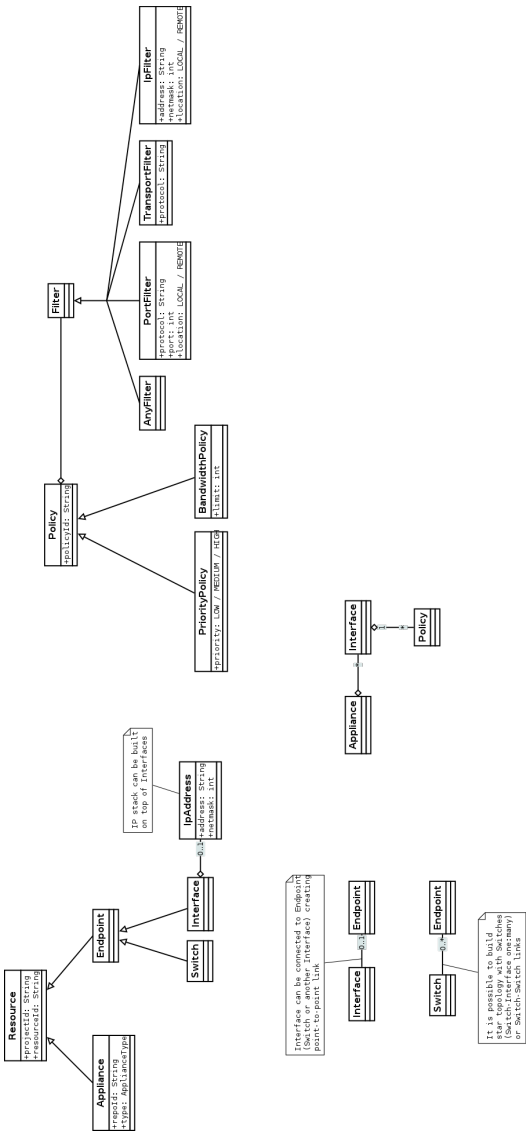

Figure 5.2: Resource object model and their relationships

Summary

Chapter 6

Implementation

Chapter overview

6.1 Implementation environment

6.2 Crossbow components implementational details

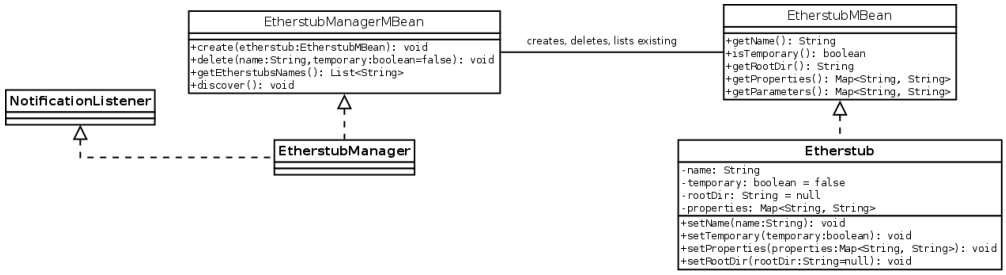

Figure 6.1: Etherstub class diagram

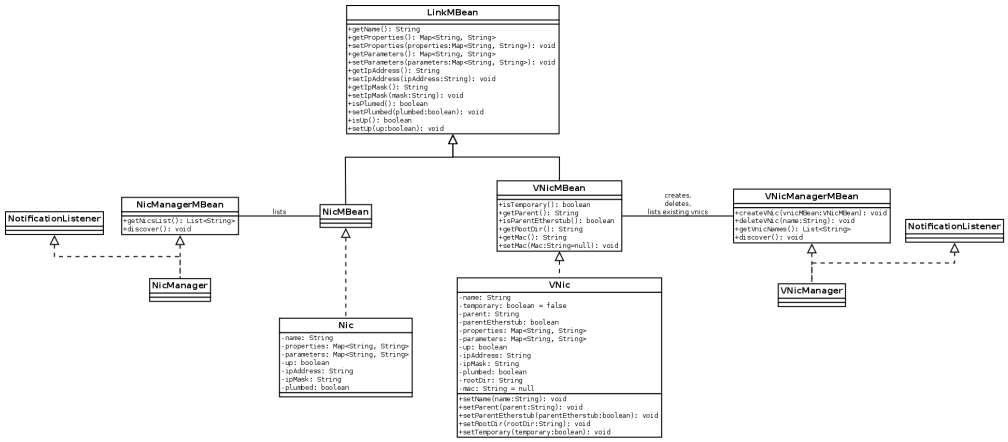

Figure 6.2: Link (VNic, Nic) class diagram

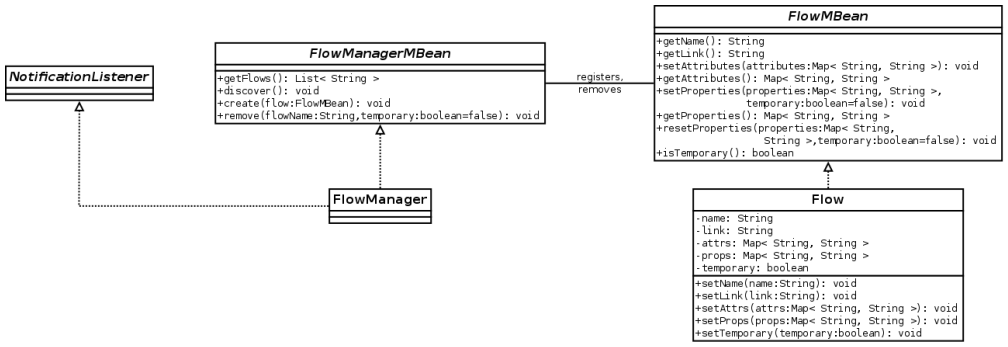

Figure 6.3: Flow class diagram

6.3 Domain model transformation details

6.4 Low-level functions access

6.5 Building and running the platform

To build and run the platform the following prerequisits are required:

- Java SE 1.6, maven,
- jims sources downloaded,
- jims-crossbow module downloaded from https://github.com/robertboczek/solariscrossbow/tree/master/code.

Afterwards jims project must be built. Detailed description of how to build jims is in **README** file located in the main catalog of jims sources. One of the most common problems are missing jars that unfortunately must be manually downloaded and installed in the local repository.

Subsequently jims-crossbow module should be copied to the main folder containing jims and then build. The script **inst-crossbow-lib.sh** must be later executed to copy shared libraries *.so files to /usr/lib folder.

If everything went well another step is running jims :

- jims-gateway: .../jims-gateway/bin/jims-agent.sh [-b host_address] start|stop|restart,

- jims-agent: .../jims-agent/bin/jims-agent.sh [-b host_address] start|stop|restart.

We should run just single jims-gateway and on the rest nodes as many jims-agents as we require. For more information about **jims** and its architecture please refer to:

If jims-agent or jims-gateway did not started it is worth to see logs files, located respectively at target/.../jims-agent/var/jims/log/agent.log and target/.../jims-gateway/var/jims/log/agent.log

It also recommended to have logs opened during jims start to see whether any exception was thrown (**tail -f target/.../jimsgateway/var/jims/log/agent.log**)

Jims has jmx-based architecture so each jims-agent and jims-gateway can be accessed through jconsole. In order to do that start jconsole, select remote process and enter type:
service:jmx:rmi:///jndi/rmi://address:port/jims where **address** and **port** is concrete address and port under which jims was started. **JConsole** allows browsing registered mbeans and performing CRUD operations. Especially in case of the crossbow module it allows these operations as the figure below presents:

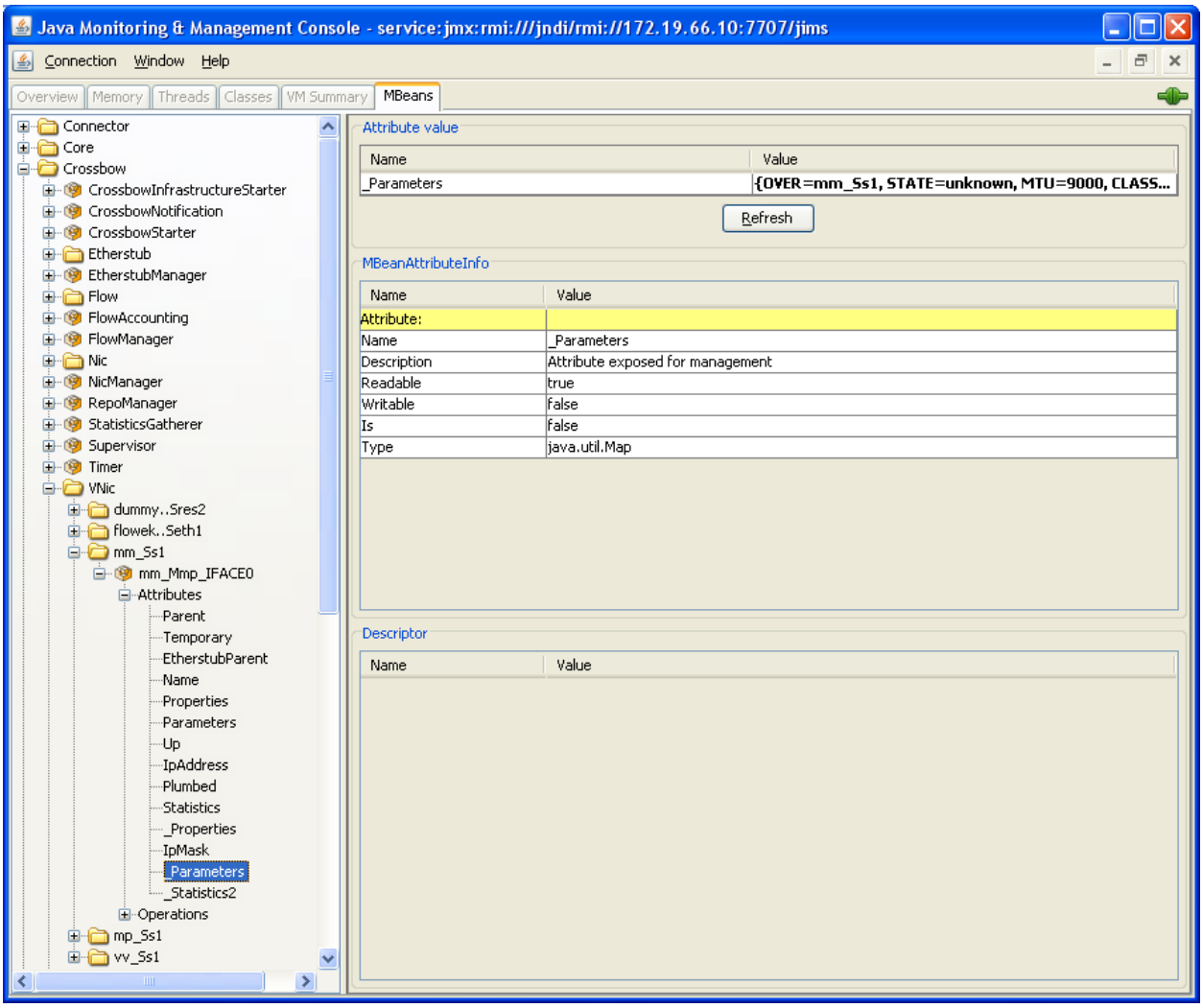

Figure 6.4: The Crossbow module registered MBeans example

To simplify working with our system gui application was created. It is located in jims/jims-crossbow/jims-crossbow-gui catalog. Application may be imported to eclipse and then build and run or build using maven (**mvn assembly:assembly**) and run **java -cp target/jims-crossbow-gui-3.0.0-exe.jar org.jims.modules.crossbow.gui.Gui**

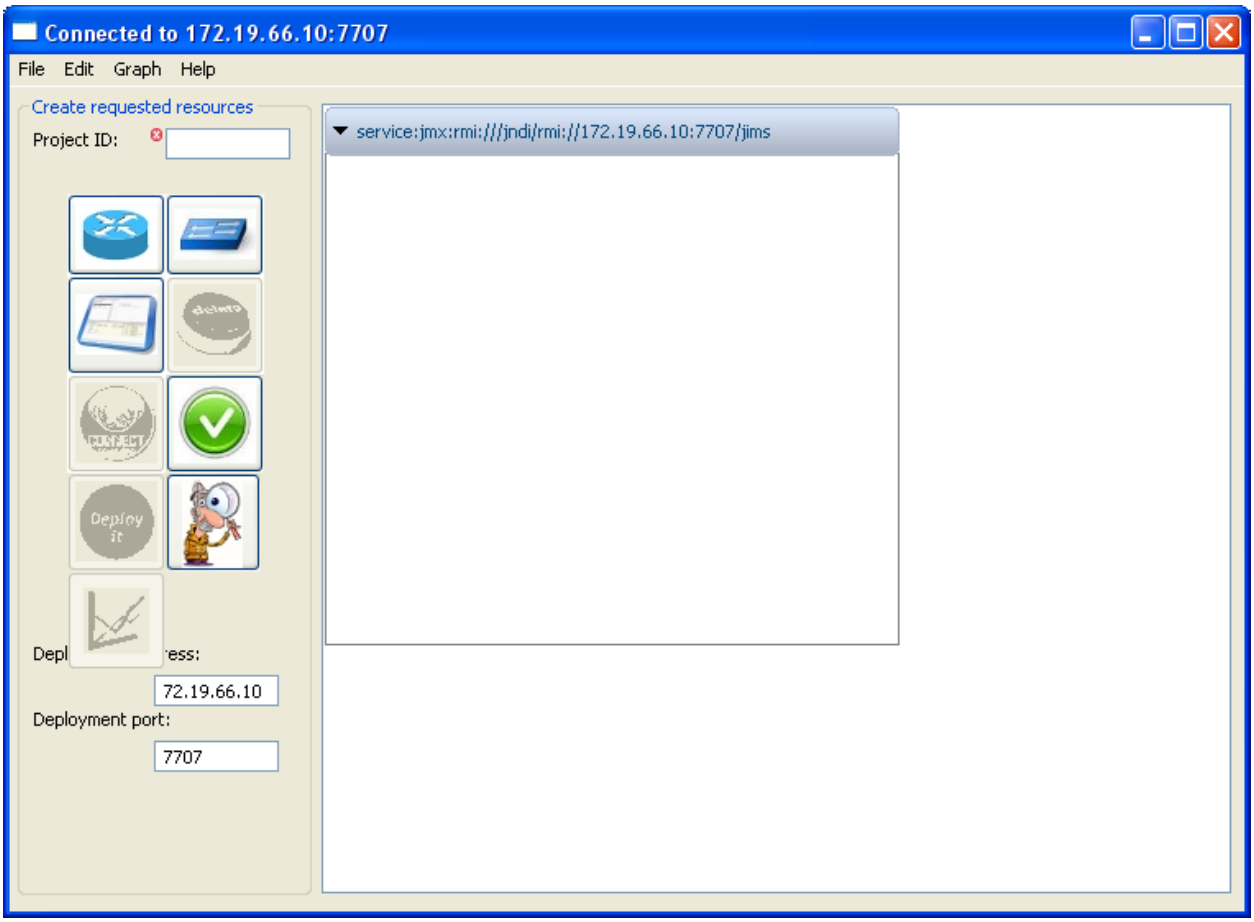

Figure 6.5: Gui application

Implemented gui application allows:

- Connecting to jims-gateway,
- Creating, modifying and removing desired network structure with requested virtual appliances,
- Discovering already created projects,
- Detailed information about links and flows like bandwidth load presented in charts from requested time periods or just simple numbers,
- Automatic logging using ssh to selected (already deployed) nodes and opening gui-type terminal.

Summary

Chapter 7

Case Study

In order to present capabilities of created system, case with video streaming server and clients with differential QoS demands was prepared.

7.1 Multimedia server

7.1.1 Scenario description

Network traffic in this case was differentiated among users through priorities and maximum bandwidth filters created over the flow paths. The figure below presents precisely network structure and client configuration.

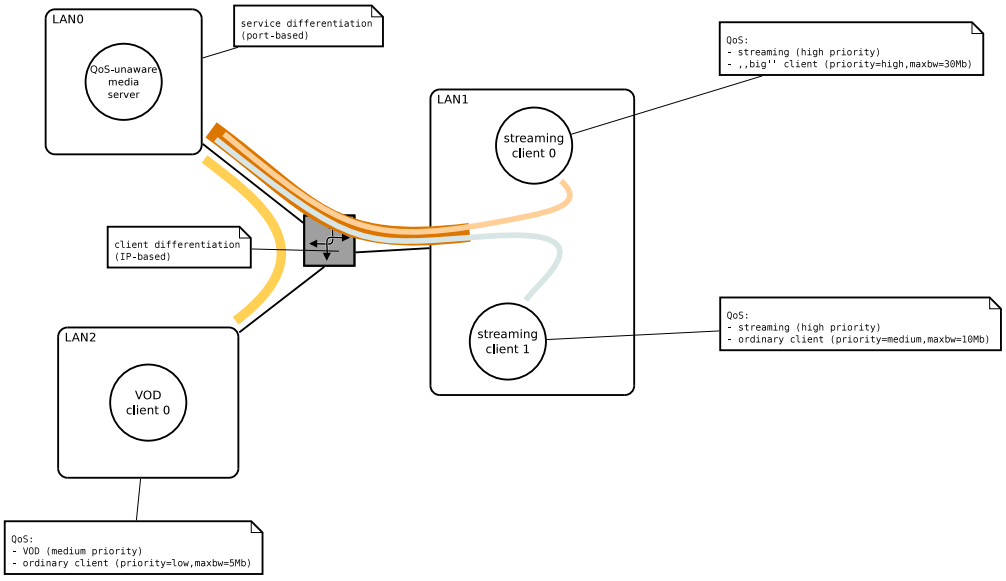

Figure 7.1: VOD + streaming clients test case example

Our network distinguishes three node types:

- http server - containing thttpd server,
- media server - containing Darwin Streaming Server,
- client player - containing MPlayer.

The section **'Preparation of the environment'** describes an example of how a new zone and then snapshot may be created, which then would be used as virtual appliance.

- similar to DiffServ (traffic classes, selectors, filters, priority, queuing)
- DiffServ doesn't specify anything virtual
- DSS and adaptive codecs
- 2 classes: VOD + streaming
- _ unicast _ vs multicast streaming
- access rules for resources of different quality
- enabling QoS for defined classes
- priorities + limiting the bandwidth (per user)!
- 3 users: 2 streaming, 1 VOD

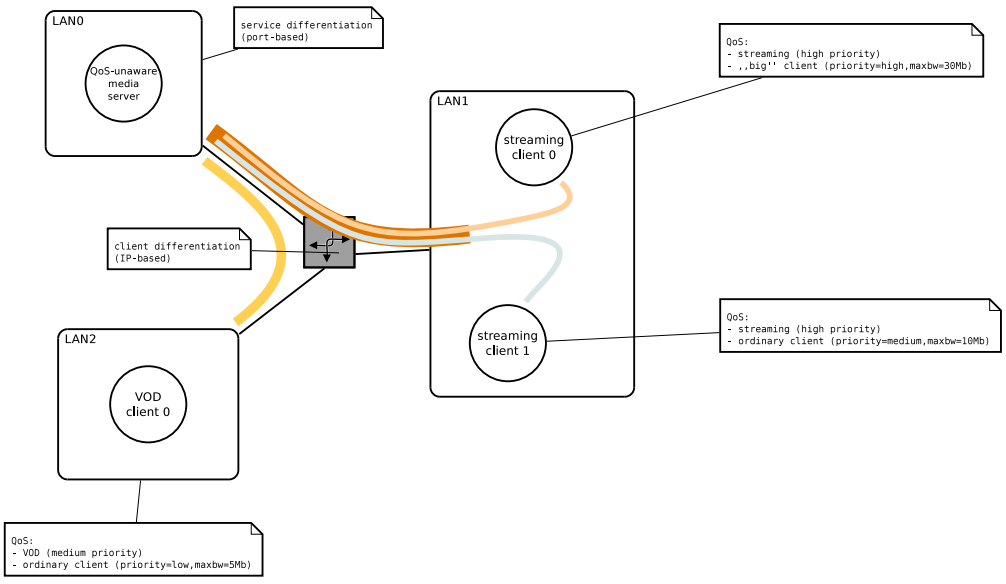

Figure 7.2: VOD + streaming clients test case example

7.1.2 Resource access requirements

7.1.3 Preparation of the environment

Virtual appliances

In order to create new zone (called for example 'mplayer') following steps should be taken:

zonecfg -z mplayer

create

zonecfg:mplayer> set zonepath=/rpool/Appliances/mplayer

zonecfg:mplayer> set autoboot=true

zonecfg:mplayer> set ip-type=exclusive

zonecfg:mplayer> verify

zonecfg:mplayer> commit

zonecfg:mplayer> exit

chmod 700 rpool/Appliances/mplayer

zoneadm -z mplayer install

After installation zone can be booted and used after logging.

zoneadm -z mplayer boot

zlogin -S mplayer

When zone is prepared, it can be transferred to the repository.

zfs snapshot -r <path>@SNAPNAME

zfs send <path>@SNAPNAME > /appliance/<appname>.SNAP

Example:
zfs snapshot -r rpool/Appliances/mplayer@SNAP
zfs send rpool/Appliances/mplayer@SNAP > /appliance/mplayer.SNAP

Afterwards these exported snaphots may be used by our gui application as desired virtual appliance to be restored.

7.1.4 Providing tunable and scalable virtual infrastructure

7.1.5 Enhancements provided by the solution

Summary

Chapter 8

Summary

Chapter overview

Bibliography [?] test.

8.1 Conclusions

8.2 Achieved goals

8.3 Further work

In terms of the future work there are many improvements that might be implemented. Probably the largest component, which was initially plannned was automatic resource assigner, that would run and perform automatic resource assignments to nodes that run under lowest load. This assigner with attached rule based system would gather data about the load on each node and based on that decide what and where instantiate. Presented and discussed system in this thesis lacks that functionality. Instead, it offers manual assignments, where user selects on which node his virtual resources should be created.

Bibliography

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated service. RFC 2475 (Informational), dec 1998. Updated by RFC 3260.

[2] Nicolas Droux. Virtual switching in solaris. 2007.

[3] http://download.oracle.com/docs/cd/E19963-01/html/821 1460/toc.html. *System Administration Guide: Oracle Solaris Zones, Oracle Solaris 10 Containers, and Resource Management.* 2010.

[4] http://itnewscast.com/servers-storage/flow-control-solaris-11-express-network virtualization. Flow control in solaris 11 express network virtualization. 2010.

[5] http://www.cisco.com. Diffserv – the scalable end-to-end qos model. August 2005.

[6] http://www.sun.com/software/solaris/ds/solaris10os.jsp. Solaris operating system. 2009.

[7] Poul-Henning Kamp and Robert N. M. Watson. Jails: Confining the omnipotent root. In *In Proc. 2nd Intl. SANE Conference*, 2000.

[8] Jose Renato Santos Yoshio Turner Jayaram Mudigonda. Taming heterogeneous nic capabilities for i/o virtualization. 2008.

[9] Thirumalai Srinivasan Nicolas Droux, Sunay Tripathi. Crossbow:from hardware virtualized nics to virtualized networks. 2009.

[10] Daniel Price and Andrew Tucker. Solaris zones: Operating system support for consolidating commercial workloads. 2004.

[11] Changhua Sun, Le He, Qingbo Wang, and Ruth Willenborg. Simplifying service deployment with virtual appliances. 2008.

[12] Inc. Sun Microsystems. *Logical Domains 1.3 Administration Guide.* 2010.

[13] Inc. Sun Microsystems. *System Administration Guide: Devices and File Systems.* 2010.

[14] Inc. Sun Microsystems. *System Administration Guide: Solaris Containers—Resource Management and Solaris Zones.* 2010.

[15] Kais Belgaied Sunay Tripathi, Nicolas Droux. Crossbow virtual wire: Network in a box. 2009.

[16] Sunay Tripathi, Nicolas Droux, Thirumalai Srinivasan, Kais Belgaied, and Venu Iyer. Crossbow: a vertically integrated qos stack. In *Proceedings of the 1st ACM workshop on Research on enterprise networking*, WREN '09, pages 45–54, New York, NY, USA, 2009. ACM.

[17] Paul A. Watters. *Solaris 10: The Complete Reference*. 1st edition, 2005.