



**Getting Started With  
OpenSolaris™ Project Crossbow  
or Network Virtualization**

*Nagendra Nagarajayya*

*July 2008*

*Sun Microsystems, Inc.*

Copyright © 2008 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

*U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements. Use is subject to license terms. This distribution may include materials developed by third parties.*

*Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and in other countries, exclusively licensed through X/Open Company, Ltd. X/Open is a registered trademark of X/Open Company, Ltd.*

*All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the United States and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.*

*Sun, Sun Microsystems, the Sun logo, BluePrints, OpenSolaris, Solaris, StarOffice, Sun Fire, and SunSolve are trademarks or registered trademarks of Sun Microsystems, Inc. or its subsidiaries in the United States and other countries.*

*This product is covered and controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.*

*DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.*

# Table of Contents

Introduction to Virtualization.....	4
Network Virtualization.....	4
1. Installing and Configuring Network Virtualization on OpenSolaris.....	4
1.1 Configuring an Etherstub.....	5
1.2 Configuring a VNIC .....	5
1.3 Configuring a Solaris Zone With a VNIC.....	7
2. Managing Bandwidth and Policies.....	9
2.1 Creating a Flow.....	9
2.1.1. Example 1: Creating a Flow by Adding Bandwidth Resource Control .....	9
2.1.2. Example 2: Creating a Flow With Limited Bandwidth.....	10
3. Monitoring VNIC Performance .....	11
3.1 flowadm show-flow Command.....	12
3.2 acctadm Command.....	12
4. Uses of Network Virtualization.....	12
4.1 Building a Network in a Computer.....	12
4.1.1 Steps to Simulate the Network on a Computer.....	15
4.1.2 Testing the "Network in a Computer Configuration" as a Router.....	19
4.1.3 Testing the "Network in a Computer Configuration" as a Firewall/NAT.....	19
5. Conclusion.....	22
6. Acknowledgments.....	22
7. References.....	22
8. For More Information.....	23

## Introduction to Virtualization

Virtualization is the abstraction of a physical device or resource [1]. The physical device could be a computer, a processor, a network card, a memory chip, or storage media. The abstraction allows the virtualized device to be used as the real physical device. Thus the technical complexity of the physical device becomes hidden and a simpler interface is provided.

The abstraction allows multiplexing, in which a single physical device is shared between users or requests, or de-multiplexing, in which a single request is split across multiple physical devices. The abstraction can consist of software and hardware. When the abstraction simulates an entire computer, it is called platform virtualization. When it simulates just a resource, such as a network interface card (NIC) or a storage device (for example, a disk), it is called resource virtualization.

### *Network Virtualization*

Network virtualization [2] refers to the simulation of a network and can consist of hardware and software. Network virtualization combines platform and resource virtualization and can be external or internal.

External virtualization combines many networks into a single network, such as a virtual LAN (VLAN). Internal virtualization simulates a network in a box by using Solaris™ Containers and virtual NICs (VNICs), a technology being developed by the OpenSolaris [3] Crossbow Project for network virtualization [4].

This paper focuses on Crossbow's internal network virtualization. It describes how to build a network in a box, create VNICs and switches, manage bandwidth across VNICs, Solaris Containers, and virtual machines, provide quality of service (QoS), and monitor the performance and status of the virtualized network.

## 1. Installing and Configuring Network Virtualization on OpenSolaris

Crossbow network virtualization is scheduled to be part of OpenSolaris by default, but at the time of writing (July 2008) is not yet integrated into the current release. So you will have to download the Crossbow bits and install them as an add-on from the following URL:

<http://opensolaris.org/os/project/crossbow/snapshots>

Crossbow also requires a NIC that is compliant with the new device driver architecture, Generic LAN Driver version 3 (GLDv3 or "Project Nemo"), that was introduced in the Solaris 10 Operating System [5][6]. A list of GLDv3-compliant NICs [7] can be obtained at the following site:

[http://opensolaris.org/os/project/crossbow/faq/#ipinst\\_which\\_nic](http://opensolaris.org/os/project/crossbow/faq/#ipinst_which_nic).

Crossbow virtualization allows an administrator to create etherstubs and VNICs. An etherstub is a pseudo or dummy device and behaves like a virtual switch. VNICs can be created from an etherstub or a physical NIC.

Unlike a logical interface, a VNIC can be plumbed and remains available even if the physical device is down. Each VNIC behaves just like a physical NIC and has its own hardware resources (Rx/Tx rings, DMA channels), MAC address, kernel threads, and queues.

A VNIC can also be administered just like a physical NIC and can be assigned to a Solaris Container (Zone) or a virtual machine or be part of a VLAN. A VNIC can also have a MAC address that is set manually, automatically, randomly, or at the factory. The MAC address can be monitored like a physical NIC, and it can have QoS properties, such as bandwidth, priority, or CPU, that are set through `dladm` [8].

## 1.1 Configuring an Etherstub

Step 1. An etherstub can be created by using the `dladm` command:

```
# dladm create-etherstub e1

# dladm show-etherstub
LINK
e1
```

## 1.2 Configuring a VNIC

A VNIC can also be created and configured using the `dladm` command.

Step 1. List all links on the system:

```
# dladm show-link
LINK      CLASS      MTU      STATE      OVER
e1000g0   phys       1500     up         --
e1000g2   phys       1500     unknown   --
e1000g3   phys       1500     unknown   --
e1000g1   phys       1500     down      --
```

Step 2. Create a VNIC using a physical device or a dummy etherstub using one of the following methods:

```
# dladm create-vnic -d e1000g0 a1
```

where:

- `create-vnic` is an option to `dladm` to create a VNIC.
- `e1000g` is the physical NIC.
- `a1` is the VNIC link ID.

or

```
# dladm create-vnic -d e1 a1
```

where `e1` is the dummy device created in Section 1.1.

**Note:** The name of a device or VNIC needs to be of the form `a-z|A-Z|_ [a-z|A-Z|0-9|_] 0-9`. See more information about vanity naming in an OpenSolaris document [9], page 22.

Step 3. The created VNIC can be listed using the `dladm` command:

```
# dladm show-vnic
```

LINK	OVER	SPEED	MACADDRESS	MACADDRTYPE	VI
<b>a1</b>	<b>e1000g0</b>	<b>1000 Mbps</b>	<b>2:8:20:8e:63:2b</b>	<b>random</b>	<b>0</b>

The previous output shows a1 as a link over e1000g0 physical link with 1 Gb/sec speed and a randomly assigned MAC address.

The a1 link can also be listed with the show-link command, as in Step 1:

```
# dladm show-link
LINK          CLASS      MTU      STATE    OVER
e1000g0      phys      1500    up      --
e1000g2      phys      1500    unknown --
e1000g3      phys      1500    unknown --
e1000g1      phys      1500    down    --
a1          vnic      1500    up      e1000g0
```

The previous output shows a1 as a link with class vnic over the e1000g0 physical link.

#### Step 4. Plumb and bring up the the VNIC:

```
bash-3.2# ifconfig a1 plumb

bash-3.2# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index
  1
    inet 127.0.0.1 netmask ff000000
e1000g0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index
  2
    inet 192.168.16.158 netmask ffffffff broadcast 192.168.16.255
    ether 0:14:4f:1e:1a:30
e1000g1: flags=201000803<UP,BROADCAST,MULTICAST,IPv4,CoS> mtu 1500 index 3
    inet 10.0.1.1 netmask ffffffff broadcast 10.0.1.255
    ether 0:14:4f:1e:1a:31
a1: flags=201000842<BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 5
    inet 0.0.0.0 netmask 0
    ether 2:8:20:8e:63:2b
lo0: flags=2002000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6,VIRTUAL> mtu 8252 index
  1
    inet6 ::1/128
e1000g0: flags=202004841<UP,RUNNING,MULTICAST,DHCP,IPv6,CoS> mtu 1500 index 2
    inet6 fe80::214:4fff:fe1e:1a30/10
    ether 0:14:4f:1e:1a:30

# ifconfig a1 192.168.15.10 up
```

The a1 VNIC behaves just like a real physical NIC, and it has its own hardware and software resources and can be administered with ifconfig, monitored with netstat, snooped, and so on.

The a1 VNIC with IP address 192.168.15.10 should now be pingable.

#### Step 5. Ping the IP address on the VNIC:

```
# ping 192.168.15.10
192.168.15.10 is alive
```

Pinging 192.168.15.10 should return a valid response.

## 1.3 Configuring a Solaris Zone With a VNIC

A VNIC can be assigned to a zone just like a physical NIC by setting the `ip-type` to `exclusive` when you configure the zone. By setting `ip-type` to `exclusive`, the zone can have its own IP instance, and consequently, IP routing tables, ARP tables, IPsec policies, and IP Filter rules can be defined for the zone. You can also run the `ndd` command to set TCP/IP tunables within the zone.

The VNIC IP address can also be configured after a zone is installed and booted. This step is similar to configuring a physical NIC in the global zone either after you install the operating system or after you issue the `sys-unconfig` command.

Perform the following steps to configure a VNIC in a zone.

Step 1. Unplumb the `a1` interface that you created in Section 1.2:

```
# ifconfig a1 unplumb
```

Step 2. Create and install the zone, making sure that `ip-type` is set to `exclusive`:

```
# zonecfg -z a1-zone
a1-zone: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:a1-zone> create
zonecfg:a1-zone> set zonepath=/export/a1-zone
zonecfg:a1-zone> set autoboot=false
zonecfg:a1-zone> set ip-type=exclusive
zonecfg:a1-zone> add net
zonecfg:a1-zone:net> set physical=a1
zonecfg:a1-zone:net> end
zonecfg:a1-zone> verify
zonecfg:a1-zone> commit
zonecfg:a1-zone> exit

# zoneadm -z a1-zone install
Preparing to install zone <a1-zone>.
Creating list of files to copy from the global zone.
Copying <8992> files to the zone.
Initializing zone product registry.
Determining zone package initialization order.
Preparing to initialize <1194> packages on the zone.
Initialized <1194> packages on zone.
Zone <a1-zone> is initialized.
Installation of these packages generated warnings: <SUNWstaroffice-core02 SUNWcr
yptpoint>
The file </export/a1-zone/root/var/sadm/system/logs/install_log> contains a log
of the zone installation.
```

Step 3. Boot the zone:

```
# zoneadm -z a1-zone boot
```

Step 4. Log in to the console by using `zlogin -C`:

```
# zlogin -C a1-zone
Enter the Internet Protocol (IP) address for this network interface. It
must be unique and follow your site's address conventions, or a
system/network failure could result.
```

IP addresses contain four sets of numbers separated by periods (for example 129.200.9.1).

Step 5. Configure the host name and IP address as prompted by the system configuration screens; for the IP address, use 192.168.1.100:

...  
...

IP address for a1: 192.168.1.100

...  
...

Confirm the following information. If it is correct, press F2; to change any information, press F4.

Host name: a1zone  
IP address: 192.168.1.100  
System part of a subnet: Yes  
Netmask: 255.255.255.0  
Enable IPv6: No  
Default Route: Specify one  
Router IP Address: 192.168.1.100

...  
...

After you have configured the zone, the system reboots.

Step 6. After the system reboots, log back into the zone or use the same console to continue the configuration:

```
# zlogin a1-zone
[Connected to zone 'a1-zone' pts/3]
Last login: Wed Jun  4 10:04:35 on console
Sun Microsystems Inc. SunOS 5.11 net-virt xb_27 snv_84 051208 Near Future
```

Step 7. To check the VNIC's interface configuration, use the `ifconfig` command:

```
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
a1: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
    inet 192.168.1.100 netmask fffffff0 broadcast 192.168.1.255
    ether 2:8:20:8e:63:2b
lo0: flags=2002000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6,VIRTUAL> mtu 8252 index 1
    inet6 ::1/128
```

Step 8. To monitor the virtual interface, use the `netstat` command:

```
# netstat -a -I a1 10
      input   a1      output      input (Total)      output
packets errs  packets errs  colls  packets errs  packets errs  colls
```

```
2029    0    1141    0    0    2042    0    1154    0    0
0        0        0        0        0        0        0        0        0        0
0        0        0        0        0        0        0        0        0        0
```

Step 9. To set any IP, TCP, or UDP tunable, use the `ndd` command:

```
# ndd /dev/tcp
name to get/set ? tcp_time_wait_interval
value ? 1000
name to get/set ? tcp_time_wait_interval
value ?
length ?
1000
name to get/set ?
```

Step 10. Ping the IP address:

```
# ping 192.168.1.100
192.168.1.100 is alive
```

## 2. Managing Bandwidth and Policies

Crossbow network virtualization allows bandwidth control and policies to be established so that network traffic can be managed more efficiently to provide QoS [10][11][12]. The bandwidth control and policy can be set at layer 2, 3, or 4 headers that can be used to identify a protocol, service, or virtual machine.

A bandwidth control policy uses its own kernel resources like layer 2, 3, or 4 queues, and so on, so other traffic will not be affected by the traffic that is being managed by the policy or vice versa. Traditional QoS mechanisms are very complex to set up. Moreover, because they are mostly interrupt-driven and implemented as a layer between the NIC driver and the IP stack [13], these mechanisms also affect performance.

Crossbow can avoid this problem by implementing the bandwidth management as part of the stack architecture without a performance penalty, thus simplifying the setup of a policy.

A policy or a flow is created by using the `flowadm` command [11]. A flow can be created for a Solaris Container or a virtual machine by assigning the flow its own IP address or MAC address. Similarly, a flow can be created for a service by making use of the transport and port number. Thirdly, a flow can be created just for the transport. A flow can also be restricted to a set of CPUs or just a CPU. A policy or flow that is created for a physical NIC will not affect the policy or traffic flowing through a VNIC, and vice versa.

### 2.1 Creating a Flow

#### 2.1.1. Example 1: Creating a Flow by Adding Bandwidth Resource Control

A flow can be created around HTTPS traffic as follows.

Step 1. Create a flow policy to prioritize HTTPS traffic:

```
# flowadm add-flow -l e1000g0 -a transport=TCP,local_port=443 https-1
```

Step 2. List the newly created policy:

```
# flowadm show-flow https-1
NAME          LINK          ATTR          VALUE
https-1      e1000g0
              ip_version    4
              transport     tcp
              local_port    443
```

Step 3. Modify the flow policy, for example, by adding a bandwidth resource control:

```
# flowadm set-flowprop -p maxbw=200Mbps https-1
```

Step 4. List the flow properties to see bandwidth limits:

```
# flowadm show-flowprop -l e1000g0
FLOW          PROPERTY      VALUE          DEFAULT      POSSIBLE
https-1      maxbw         200           --           --
https-1      cpus          ?             --           --
https-1      priority      ?             --           --
https-1      fanout        ?             --           --
```

Step 5. Add priority and processing control or modify assigned bandwidth limits:

```
# flowadm set-flowprop -p maxbw=500Mbps,priority=high,cpus=9 https-1
```

```
# flowadm show-flowprop -l e1000g0
```

```
FLOW          PROPERTY      VALUE          DEFAULT      POSSIBLE
https-1      maxbw         500           --           --
https-1      cpus          9             --           --
https-1      priority      High          --           --
https-1      fanout        9             --           --
```

### 2.1.2. Example 2: Creating a Flow With Limited Bandwidth

You can create a policy to limit bandwidth as well as lower the priority for FTP transfers on a VNIC.

Step 1. Create a policy around the ftp-data port for ACTIVE FTP transfers:

```
# flowadm add-flow -l v1 -a transport=TCP,local_port=20 ftp-policy
```

Step 2. Set a flow property to limit bandwidth:

```
# flowadm set-flowprop -p maxbw=10Mbps ftp-policy
```

Step 3. List flow properties of VNIC v1:

```
# flowadm show-flowprop -l v1
```

FLOW	PROPERTY	VALUE	DEFAULT	POSSIBLE
ftp-policy	maxbw	10	--	--
ftp-policy	cpus	?	--	--
ftp-policy	priority	?	--	--
ftp-policy	fanout	?	--	--

Step 4. List flow policy on link v1:

```
# flowadm show-flow -l v1
```

NAME	LINK	ATTR	VALUE
ftp-policy	v1	ip_version	4
		transport	tcp
		local_port	21

### 3. Monitoring VNIC Performance

You can obtain different types of information about VNICs by using regular monitoring tools as well as new commands that are VNIC-specific. The following examples show the different syntaxes to use to monitor VNICs.

- To monitor VNIC performance with regular monitoring tools, such as `netstat` or `nicstat` [14], use the following:

```
# netstat -a -I a1 10
```

input		a1			output		input (Total)		output	
packets	errs	packets	errs	colls	packets	errs	packets	errs	colls	
110259	0	79	0	0	110287	0	107	0	0	
14	0	0	0	0	14	0	0	0	0	

- To list VNICs in the system, use the `show-vnic` subcommand with options:

```
# dladm show-vnic -s -i 1
```

	ipackets	rbytes	opackets	obytes		
a1	2146	105420	1256	67597	175.5	100.6
a2	1965	82530	928	39036	160.7	74.3
v1	1249	67303	1229	66942	102.1	98.4
v2	2178	106345	4421	432690	178.1	354.0
v3	1223	66654	1249	67303	100.0	100.0

- To list physical links in the system and also monitor performance, use the `show-link` subcommand:

```
# dladm show-link -s -i 1
LINK          IPACKETS  RBYTES    IERRORS  OPACKETS  OBYTES    OERRORS
e1000g0       476532    38728602  0         17783     2183512   0
```

- To monitor a flow policy, use either the `flowadm show-flow` command or the `acctadm` command, as follows.

### 3.1 *flowadm show-flow* Command

```
# flowadm show-flow -i 10 -s
FLOW          IPACKETS  RBYTES    IERRORS  OPACKETS  OBYTES    OERRORS
https-1       0         0         0         0         0         0
```

### 3.2 *acctadm* Command

```
# acctadm -e extended -f /var/log/net.log net
```

```
# flowadm show-usage -f /var/log/net.log
```

Bytes	Packets	Errors	Duration	Bandwidth	Link/Flow
0	0	0	920	0.00 bps	https-1
221820420	216148	0	920	1.93 Mbps	ftp-policy

```
# flowadm show-usage -f /var/log/net.log ftp-policy
```

Start Time	End Time	In Bytes	Out Bytes	Bandwidth	Device
13:37:19	13:37:39	615846	31035764	12.66 Mbps	ftp-policy
13:37:39	13:37:59	1544580	77710536	31.70 Mbps	ftp-policy
13:37:59	13:38:19	0	0	0.00 bps	ftp-policy
13:38:19	13:38:39	0	0	0.00 bps	ftp-policy
13:38:39	13:38:59	0	0	0.00 bps	ftp-policy
13:38:59	13:39:19	0	0	0.00 bps	ftp-policy
13:39:19	13:39:39	2167026	108746668	44.37 Mbps	ftp-policy

## 4. Uses of Network Virtualization

### 4.1 *Building a Network in a Computer*

An entire network can be simulated by using network virtualization [15][16] on a single host. Further, the simulated network can actually be used to consolidate a cluster of servers onto a single host.

A network consists of hosts, switches, and routers, as illustrated in Figure 1. In this section, a sample use case is presented that shows how network virtualization can use routers, virtual switches, a firewall

or Network Address Translation (NAT), and Solaris Zones to simulate the real network shown in Figure 1. The simulated network is illustrated in Figure 2.

The network in Figure 1 is made of three networks: 192.168.1.x, 20.10.10.x, and 192.168.16.x. Switch `s1` is used to connect hosts on the 192.168.1.x network, while Router `r1` routes traffic between 192.168.1.x and 20.10.10.x, and Router `r2` routes traffic between the 20.10.10.x and 192.168.16.x network.

In the virtualized network, the Switch `s1` is simulated by using etherstub `e1`. An etherstub is a dummy device that simulates a physical NIC and switches between VNICs connected to it.

The Router `r1` is simulated by using a Solaris Zone, `v1-zone`, and two etherstubs, `e2` and `e1`. One etherstub faces the external world, global zone (20.10.10.20, VNIC `v3`), while the other is behind a network (192.168.1.1, VNIC `v1`). The router can also work as a firewall with NAT capability by using IP Filter [15][17] to make the 192.168.1.x network invisible to the external world.

The hosts `a` and `b` in the Fig 1 network are simulated by consolidating with Solaris Zones as `a1-zone` (192.168.1.100, VNIC `a1`) and `a2-zone` (192.168.1.101, VNIC `a2`).

Etherstub `e2` has another VNIC `v1` with an IP address 20.10.10.25 to which the external world (global zone) is connected. The global zone also has a physical NIC with an IP address 192.168.16.158 and has a default route through 192.168.16.1 as the gateway.

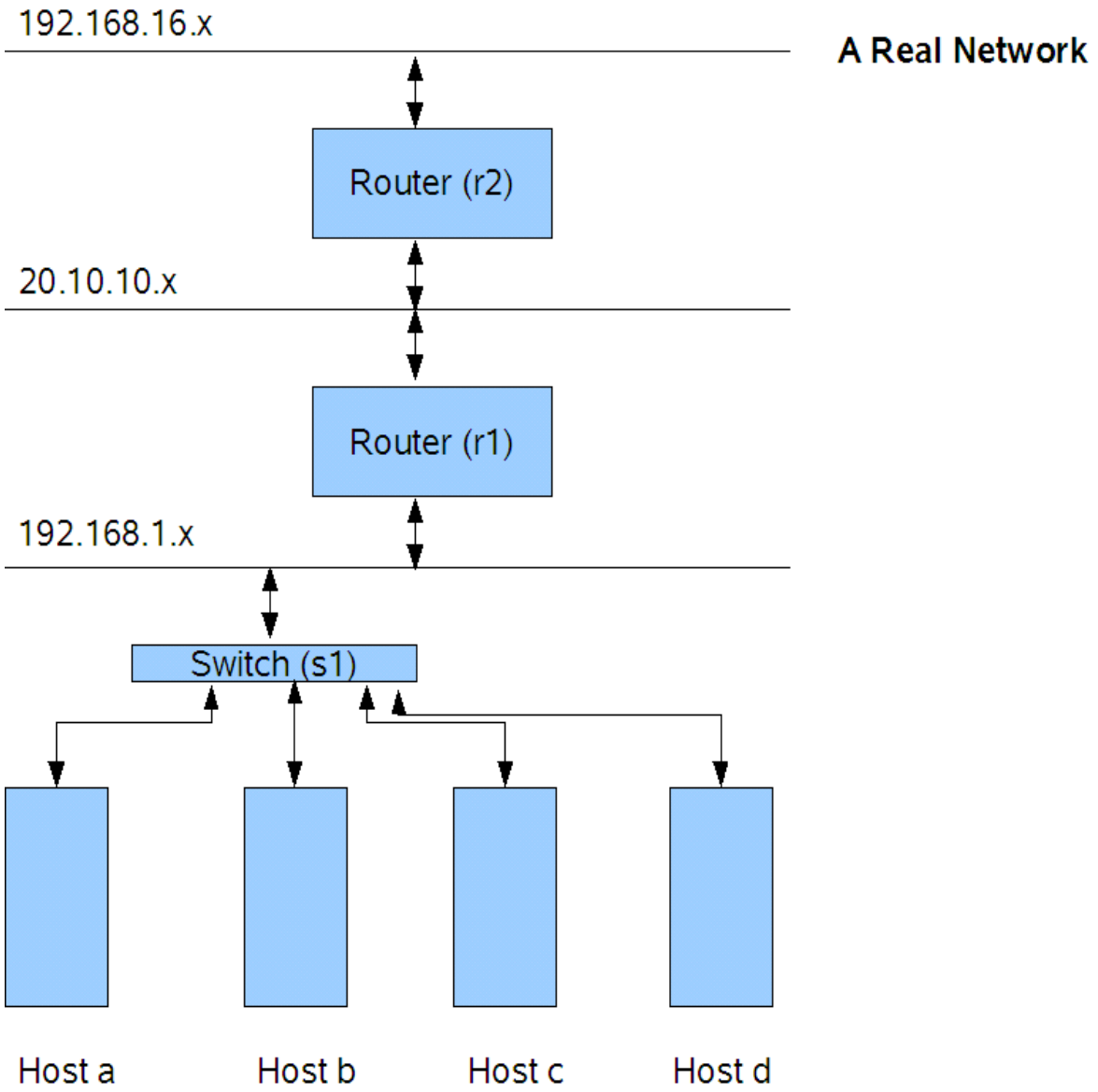


Fig 1

Figure 2 shows the real network from Figure 1 simulated within a Sun Fire™ T2000 system.

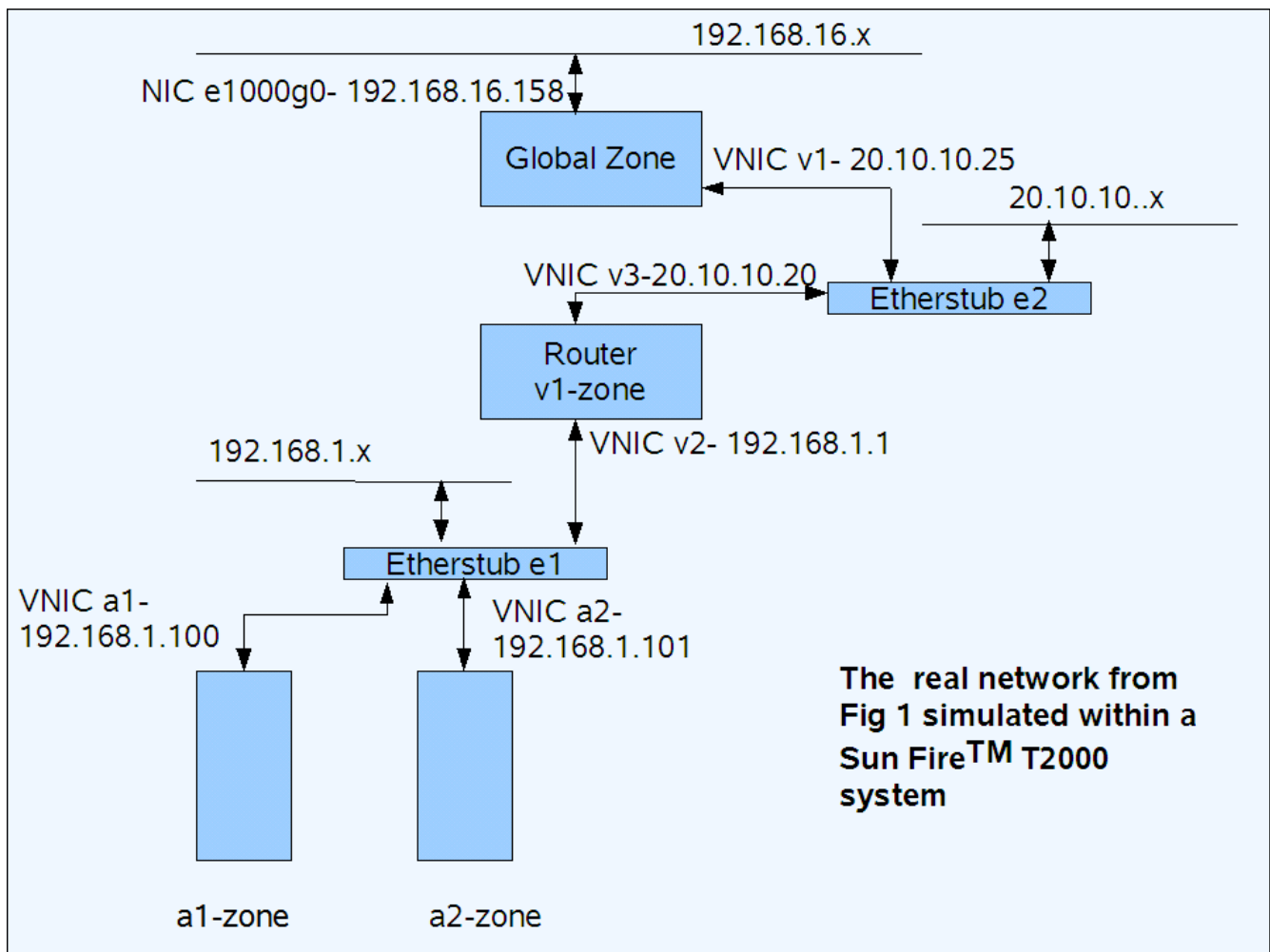


Fig 2

#### 4.1.1 Steps to Simulate the Network on a Computer

Step 1. Delete the VNIC that was created in Section 1.2 on e1000g0, the physical NIC, and which was assigned to a zone in Section 1.3.

**Note:** The zone configured and booted in Section 1.3, Step 2 needs to be halted first before deleting the VNIC a1.

```
# zoneadm -z a1-zone halt
# dladm delete-vnic a1
```

Step 2. Create two etherstubs:

```
# dladm create-etherstub e1
# dladm create-etherstub e2
```

Step 3. Create three VNICs on etherstub e1:

```
# dladm create-vnic -d e1 a1
# dladm create-vnic -d e1 a2
# dladm create-vnic -d e1 v2
```

Step 4. Create two etherstubs on etherstub e2:

```
# dladm create-vnic -d e2 v1
# dladm create-vnic -d e2 v3
```

Step 5. Configure the external interface v1 and assign it the IP address 20.10.10.25:

```
# ifconfig v1 plumb
# ifconfig v1 20.10.10.25 up
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
e1000g0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
    inet 192.168.16.158 netmask ffffffff broadcast 192.168.16.255
    ether 0:14:4f:1e:1a:30
v1: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 9000 index 3
    inet 20.10.10.25 netmask ff000000 broadcast 20.255.255.255
    ether 2:8:20:8f:80:a1
lo0: flags=2002000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6,VIRTUAL> mtu 8252 index 1
    inet6 ::1/128
e1000g0: flags=202004841<UP,RUNNING,MULTICAST,DHCP,IPv6,CoS> mtu 1500 index 2
    inet6 fe80::214:4fff:fe1e:1a30/10
    ether 0:14:4f:1e:1a:30
```

Step 6. Create two more zones, one to consolidate the host from Fig 1 and the second for use as a router/firewall/NAT host:

**Note:** Make sure a1-zone is halted. Refer to Step 1. The a1-zone is the equivalent of host "a" in Fig 1.

Step 7. Export a1-zone so that it can be cloned:

```
# zonecfg -z a1-zone export -f /export/master
```

Step 8. Use a text editor to make the following changes (see lines in bold) to the /export/master file:

```
# vi /export/master
create -b
set zonepath=/export/a2-zone
set autoboot=false
set ip-type=exclusive
add inherit-pkg-dir
set dir=/lib
end
add inherit-pkg-dir
set dir=/platform
```

```
end
add inherit-pkg-dir
set dir=/sbin
end
add inherit-pkg-dir
set dir=/usr
end
add net
set physical=a2
end
```

**Note:** Make sure you have enough disk space under `/export`. Otherwise, change `zonepath` to `/export/home/a2-zone` or something similar on a different slice or disk.

Step 9. Create a new zone with the name `a2-zone`:

```
# zonecfg -z a2-zone -f /export/master
# zoneadm -z a2-zone clone a1-zone
```

Step 10. Edit the `export/master` file again with the following changes:

- Change `a2-zone` to `v1-zone`.
- Change `physical=a2` to `physical=v2`.

Step 11. Create a new zone named `v1-zone`:

```
# zonecfg -z v1-zone -f /export/master
# zoneadm -z v1-zone clone a1-zone
```

Step 12. Boot the three zones:

```
# zoneadm -z a1-zone boot
# zoneadm -z a2-zone boot
# zoneadm -z v1-zone boot
```

Step 13. Configure the zones by performing the following steps:

- a. Set the `a2-zone` IP address to `192.168.1.101`.
- b. Ensure that `a1-zone` is configured with an IP address of `192.168.1.100`, as shown in Section 1.3.
- c. Set the `v1-zone` IP address to `192.168.1.1`.

Step 14. After the system reboots, log in to the zone and ensure the following by using the `ifconfig` command:

- `v1-zone` IP on interface `v2` has the IP address `192.168.1.1`.
- `a2-zone` IP on interface `a2` has the IP address `192.168.1.101`.

Step 15. Open a terminal window and log in to the v1-zone to configure the outside world interface:

a. First add the v3 interface to 1-zone.

```
# zonecfg -z v1-zone
zonecfg:v1-zone> add net
zonecfg:v1-zone:net> set physical=v3
zonecfg:v1-zone:net> end
zonecfg:v1-zone> verify
zonecfg:v1-zone> commit
zonecfg:v1-zone> exit

# zoneadm -z v1-zone halt
# zoneadm -z v1-zone boot
```

b. Plumb the v3 interface after logging in to the zone.

```
# zlogin v1-zone
# ifconfig v3 plumb
# ifconfig v3 20.10.10.20
# ifconfig v3 up
# ifconfig v2 router
# ifconfig v3 router
```

c. Ping the 20.10.10.25 interface to see if the external network is reachable:

```
# ping 20.10.10.25
20.10.10.25 is alive
```

Step 16. Add the following changes to the /etc/hostname.v3 file to make the interface address persistent across reboots:

```
# vi /etc/hostname.v3
20.10.10.20
```

Step 17. Make 20.10.10.25 the gateway:

```
# route add default 20.10.10.25
# vi /etc/defaultrouter
20.10.10.25
```

Step 18. Open a new terminal window and log in to a1-zone:

```
# zlogin a1-zone
```

Step 19. Remove the default route and create a new default route, 192.168.1.1, to make v1-zone the gateway:

```
# route delete default 192.168.1.100
# route add default 192.168.1.1

# vi /etc/defaultrouter
192.168.1.1
```

## 4.1.2 Testing the "Network in a Computer Configuration" as a Router

Step 1. Open a new terminal window and verify that you are in the global zone:

```
# zonename
global
```

Step 2. Add a route to the internal 192.168.1.x network:

```
# route add 192.168.1.0 20.10.10.20
```

```
# netstat -rn
```

```
Routing Table: IPv4
Destination          Gateway              Flags  Ref    Use    Interface
-----
default              192.168.16.1        UG      1      1    e1000g0
20.0.0.0             20.10.10.25         U        1      2    v1
192.168.1.0        20.10.10.20        UG     1     0 v1
192.168.16.0        192.168.16.158     U        1     15    e1000g0
127.0.0.1           127.0.0.1          UH       1     33    lo0
```

Step 3. Ping the 20.10.10.20 address, which is facing externally and functions as the gateway to the internal 192.168.1.x network:

```
# ping 20.10.10.20
20.10.10.20 is alive
```

Step 4. Ping 192.168.1.100 or a1-zone:

```
# ping 192.168.1.100
192.168.1.100 is alive
```

Step 5. Log in to a1-zone by using the ssh command:

```
# ssh 192.168.1.100 -l root
Password:
Last login: Fri Jun  6 09:43:38 2008
Sun Microsystems Inc.  SunOS 5.11      /disk1/crossbow/crossbow-clone xb_28 snv
_84 daily/daily.060308  Jun. 03, 2008
SunOS Internal Development:  crossbow 2008-06-03 [crossbow-clone]
bfu'ed from /net/npt/export/crossbow/crossbow-gate/daily/daily.060308/archives/s
parc/nightly on 2008-06-05
Sun Microsystems Inc.  SunOS 5.11      net-virt xb_27 snv_84 051208    Near Future

# zonename
a1-zone
```

## 4.1.3 Testing the "Network in a Computer Configuration" as a Firewall/NAT

Step 1. Delete the route to the 192.168.1.x network:

```
# route delete 192.168.1.0 20.10.10.20
delete net 192.168.1.0: gateway 20.10.10.20
```

As a consequence of this deletion, pinging or logging in to the zone will fail.

```
# ping 192.168.1.100  
(will not have any response)
```

```
# ssh 192.168.1.100 -l root  
(will not connect and just time out)
```

Step 2. Open a new terminal window and log in to the v1-zone (the router):

```
# zlogin v1-zone
```

Step 3. Set up the IP Filter configuration so that all traffic from the 192.168.1.0 network will go through the external facing interface, 20.10.10.20:

```
# vi /etc/ipf/ipf_set.conf  
  
# ipf.conf  
#  
# IP Filter rules to be loaded during startup  
#  
# See ipf(4) manpage for more information on  
# IP Filter rules syntax.  
map v3 192.168.1.0/24 -> 20.10.10.20 /32 portmap tcp/udp auto  
map v3 192.168.1.0/24 -> 20.10.10.20/32
```

**Note:** The `ipf_set.conf` file can be created as `/etc/ipf/ipf.conf`. However, the `ipfilter` service does not start and generates a syntax error, so `ipnat` needs to be run manually.

Step 4. Start the IP Filter service:

```
# svcadm enable network/ipfilter
```

Step 5. Run `ipnat` manually by using the `-f` option to read from the `ipf_set.conf` file:

```
# ipnat -v -f /etc/ipf/ipf_set.conf  
map v3 192.168.1.0/24 -> 20.10.10.20/32 portmap tcp/udp auto  
map v3 192.168.1.0/24 -> 20.10.10.20/32
```

Step 6. Run `ipnat -l` to make sure the mapping is set in the kernel and to ensure there are no active sessions:

```
# ipnat -l  
List of active MAP/Redirect filters:  
map v3 192.168.1.0/24 -> 20.10.10.20/32 portmap tcp/udp auto  
map v3 192.168.1.0/24 -> 20.10.10.20/32
```

List of active sessions:

Step 7. Open a new terminal window and log in to a1-zone:

```
# zlogin a1-zone
# zonename
a1-zone
```

Step 8. Log in to an external network:

```
# ssh 192.168.16.158 -l root
Password:
Last login: Fri Jun 6 09:52:12 2008 from v280r-240-01
Sun Microsystems Inc. SunOS 5.11 /disk1/crossbow/crossbow-clone xb_28 snv_84
daily/daily.060308 Jun. 03, 2008
SunOS Internal Development: crossbow 2008-06-03 [crossbow-clone]
bfu'ed from /net/npt/export/crossbow/crossbow-
gate/daily/daily.060308/archives/sparc/nightly on 2008-06-05
Sun Microsystems Inc. SunOS 5.11 net-virt xb_27 snv_84 051208 Near Future
```

Step 9. Issue the netstat -a command to show the connection from the external NAT interface:

```
netstat -a | grep EST
ns-t2000-11.ssh v280r-240-01.33953 49640 0 49640 0 ESTABLISHED
ns-t2000-11.ssh 20.10.10.20.26467 53760 47 53760 0 ESTABLISHED
ns-t2000-11.ssh v280r-240-01.34086 49640 0 49640 0 ESTABLISHED
```

Step 10. Issue the zonename command to verify that you are in the global zone:

```
# zonename
global
```

Step 11. Open a new terminal window and log in to the v1-zone or use the terminal window from Step 2:

```
# zlogin v1-zone
```

Step 12. Run ipnat -l:

```
# ipnat -l
List of active MAP/Redirect filters:
map v3 192.168.1.0/24 -> 20.10.10.20/32 portmap tcp/udp auto
map v3 192.168.1.0/24 -> 20.10.10.20/32

List of active sessions:
MAP 192.168.1.100 43587 <- -> 20.10.10.20 26467 [192.168.16.158 22]
```

The output shows the connections that go through the IP Filter.

Step 13. Open a new terminal window and log in to a1-zone:

```
# zlogin a1-zone
```

```
# zonename
a1-zone
```

Step 14. Issue the `netstat -a` command to show the connection from `a1-zone` to the external `192.168.16.x` network:

```
# netstat -a | grep EST
a.37396          192.168.16.158.ssh    53760           0 53760           0 ESTABLISHED
```

## 5. Conclusion

This paper introduces the OpenSolaris Crossbow Network Virtualization project and also demonstrates a use case, building a network in a computer. Crossbow network virtualization is an extremely powerful technology that enables a NIC or an entire network to be virtualized and can also provide QoS without performance degradation.

## 6. Acknowledgments

First, I would like to acknowledge Venu Iyer, who provided the lab system, suggestions for content, and engineering information. I would also like to acknowledge Steffen Weiberle and Ralf Weber for taking the time to review the paper and provide valuable feedback. I would also like to acknowledge the different Crossbow team blogs and their authors, such as Sunay Tripathi, Nicolas Droux, Venu Iyer, Kais Belgaied and others, for allowing me to derive content for the paper. Finally, I would like to acknowledge the editorial assistance of Raoul Carag from the Crossbow team and the editors at the BigAdmin portal for their contribution in changing this paper into easily readable technical content.

## 7. References

1. Wikipedia definition of virtualization: <http://en.wikipedia.org/wiki/Virtualization>
2. Wikipedia definition of network virtualization: [http://en.wikipedia.org/wiki/Network\\_virtualization](http://en.wikipedia.org/wiki/Network_virtualization)
3. OpenSolaris web site: <http://www.opensolaris.org>
4. Crossbow web page: <http://www.opensolaris.org/os/project/crossbow>
5. GLDv3 architecture blog: [http://blogs.sun.com/sunay/entry/the\\_solaris\\_networking\\_the\\_magic#mozTocId767708](http://blogs.sun.com/sunay/entry/the_solaris_networking_the_magic#mozTocId767708)
6. GLDv3 porting guide: <http://wikis.sun.com/display/WDD/GLDv3>
7. List of GLDv3-compliant NICs: [http://opensolaris.org/os/project/crossbow/faq/#ipinst\\_which\\_nic](http://opensolaris.org/os/project/crossbow/faq/#ipinst_which_nic)
8. `dladm` man page: <http://dlc.sun.com/osol/netvirt/downloads/20080310/dladm.1m.txt>
9. Link ID format information: <http://www.opensolaris.org/os/project/clearview/docs/vnameoverview.pdf>

10. "Crossbow Flows" blog: [http://blogs.sun.com/iyer/entry/opensolaris\\_project\\_crossbow\\_pre\\_beta](http://blogs.sun.com/iyer/entry/opensolaris_project_crossbow_pre_beta)
11. flowadm man page: <http://dlc.sun.com/osol/netvirt/downloads/20080310/flowadm.1m.txt>
12. "Dynamic Bandwidth Allocation" blog: <http://blogs.sun.com/kais/>
13. "CrossBow: Solaris Network Virtualization & Resource Control" blog: <http://blogs.sun.com/sunay/date/20060824>
14. nicstat command: [http://www.brendangregg.com/K9Toolkit/nicstat\\_example.txt](http://www.brendangregg.com/K9Toolkit/nicstat_example.txt)
15. "Private virtual networks for Solaris xVM and Zones with Crossbow" blog: <http://blogs.sun.com/droux/date/20080214>
16. "Network in a Box (Creating a real Network on your Laptop)" blog: <http://blogs.sun.com/sunay/date/20080229>
17. Setting Up NAT on Solaris Using IP Filter: [http://www.rite-group.com/rich/solaris\\_nat.html](http://www.rite-group.com/rich/solaris_nat.html)
18. Download page for Crossbow bits: <http://opensolaris.org/os/project/crossbow/snapshots>

## 8. For More Information

Here are additional Sun resources related to virtualization:

- Sun training courses at <http://www.sun.com/training/>, for example:
  - Solaris 10 Containers (SA-230-S10, WS-2260-S10, and VC-SA-230-S10)
  - Sun Virtualization: Solaris 10 Containers Administration (SA-355-S10)
- Documentation at <http://docs.sun.com>, such as:
  - *System Administration Guide: Virtualization Using the Solaris Operating System*
  - *System Administration Guide: Solaris Containers-Resource Management and Solaris Zones*
  - *Solaris Containers: Resource Management and Solaris Zones Developer's Guide*
- Wikis:
  - Sun xVM wiki: <http://wikis.sun.com/display/xVM/Sun+xVM>
  - Sun BluePrints™ wiki: <http://wikis.sun.com/display/BluePrints/Main>
  - BigAdmin wiki's Virtualization page: <http://wikis.sun.com/display/BigAdmin/Virtualization>
- Related web sites and articles:
  - Virtualization Resources for System Administrators on the BigAdmin web site: <http://www.sun.com/bigadmin/topics/virtualization/>
  - Main Sun web page on virtualization and consolidation: <http://www.sun.com/datacenter/consolidation/index.jsp>
  - Virtualization Learning Center: [http://www.sun.com/software/solaris/virtual\\_learning\\_center.jsp](http://www.sun.com/software/solaris/virtual_learning_center.jsp)
- Events of interest to users of Sun products:
  - Worldwide developer events: <http://developers.sun.com/events/>
  - Current events: <http://www.sun.com/events/index.jsp>

- Support:
  - Register your Sun gear: <https://inventory.sun.com/inventory/>
  - Services: <http://www.sun.com/service/>
  - SunSolve<sup>SM</sup> Online: <http://sunsolve.sun.com>

## Licensing Information

Unless otherwise specified, the use of this software is authorized pursuant to the terms of the license found at [http://www.sun.com/bigadmin/common/berkeley\\_license.html](http://www.sun.com/bigadmin/common/berkeley_license.html).