# Networking with Linux® on zSeries
## (Part 2 of 2)

### Session 9268

SHARE
Technology • Connections • Results

**Steffen Thoss (thoss@de.ibm.com)**
**IBM Development Lab, Boeblingen, Germany**

# Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

| | |
|---|---|
| Enterprise Storage Server | ESCON* |
| FICON | FICON Express |
| HiperSockets | IBM* |
| IBM logo* | IBM eServer |
| Netfinity* | S/390* |
| VM/ESA* | WebSphere* |
| z/VM | zSeries |

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Intel is a trademark of the Intel Corporation in the United States and other countries.

Oracle 9i is a trademark of the Oracle Corporation in the United States and other countries.

Java and all Java-related trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc., in the United States and other countries.

Lotus, Notes, and Domino are trademarks or registered trademarks of Lotus Development Corporation.

Linux is a registered trademark of Linus Thorvalds

Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.

Penguin (Tux) compliments of Larry Ewing.

SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

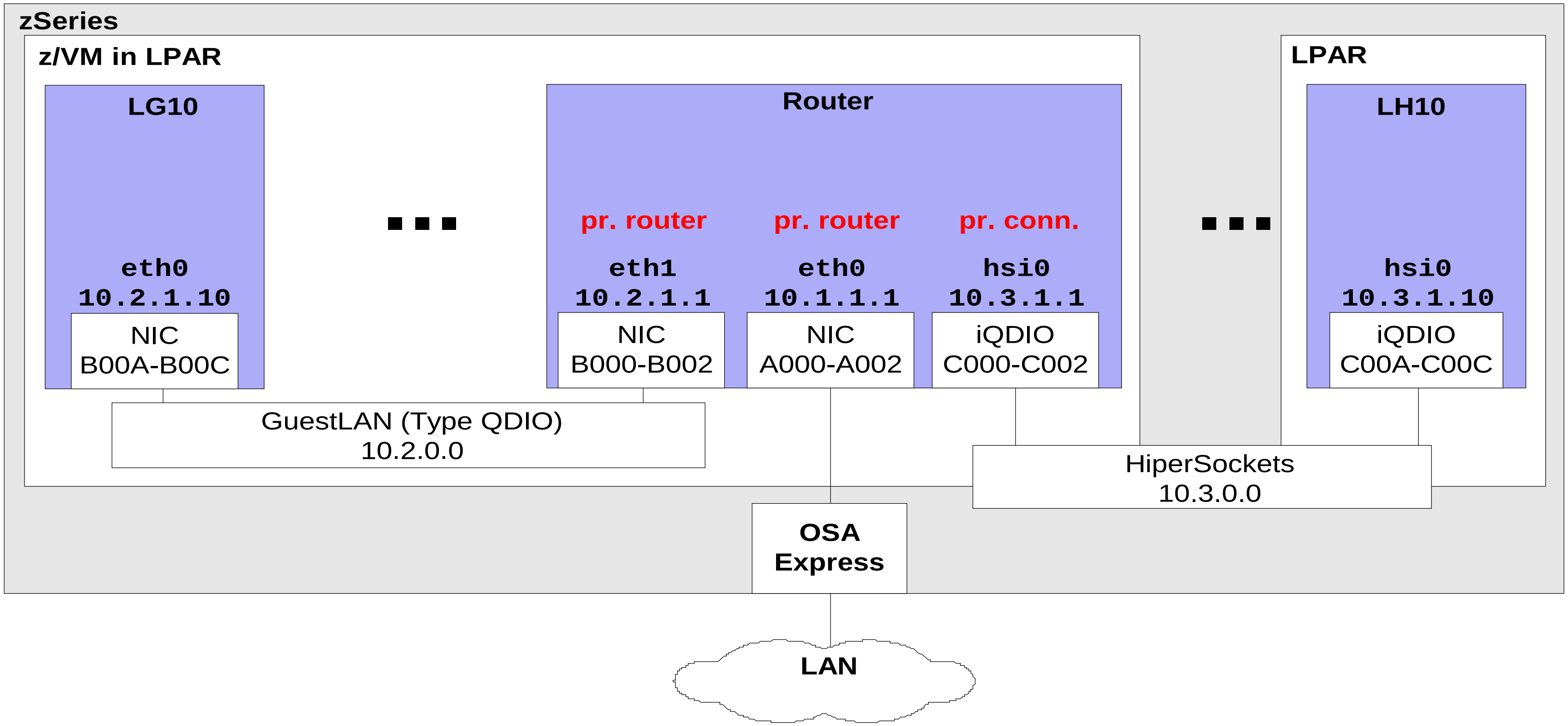
UNIX is a registered trademark of The Open Group in the United States and other countries.

* All other products may be trademarks or registered trademarks of their respective companies.

# Agenda

- Router setup for Linux on zSeries
- Failover and availability solutions:
  - IP Address Takeover
  - Virtual IP Addresses (VIPA)
  - Proxy ARP
- The qethconf tool
- The qetharp tool
- HiperSockets Network Concentrator (HSNC)

# Routing Types




| | OSA Express | VM GuestLAN QDIO | VM GuestLAN Hiper | HiperSockets |
|---|---|---|---|---|
| primary router | X | X (IPv4 only) | | |
| secondary router | X | X (IPv4 only) | | |
| multicast router | X | | | |
| primary connector | | | | X |
| secondary connector | | | | X |

# How OSA handles Router Settings

Ethernet packet

| **Dest. MAC** | Src MAC | prot | **...** | Src IP | **Dest. IP** | payload |
|---|---|---|---|---|---|---|

Ethernet Header | IP Header

Packet reaches OSA Card with destination MAC address

OSA Card

Is the packet's destination IP address set by any client OS image?

YES NO ?

**OSA Address Table (OAT)**

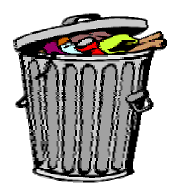
| Device Number | IP Address | Router? |
|---|---|---|
| A000 | 10.1.1.1 | primary |
| A00A | 10.1.1.10 | secondary |
| ... | ... | |
| A0AA | 10.1.1.100 | |

Is there a primary router? ?

YES NO ?

Is there a secondary router?

YES NO



Give packet to OS image

discard packet

**Linux**

IP packet

| **...** | Src IP | **Dest. IP** | payload |
|---|---|---|---|

A **multicast router** also receives **all** multicast packets

# How HiperSockets handles Router Settings

IP packet

| ... | Src IP | **Dest. IP** | payload |
|---|---|---|---|

IP Header

GuestLAN is analogous

OS Image sends IP packet via its iQDIO device

HiperSockets Microcode

Is the packet's destination IP address set by any client OS image?

YES

NO

?

| **Address Table** | | |
|---|---|---|
| **Device Number** | **IP Address** | **Connector?** |
| C000 | 10.3.1.1 | primary |
| C0A0 | 10.3.1.30 | secondary |
| ... | ... | |
| C0AA | 10.1.1.100 | |

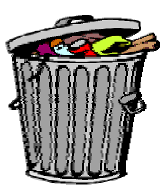
Is there a primary connector? ?

YES

NO

?

Is there a secondary connector?



YES

NO

Copy packet to OS image storage

discard packet

**Linux**

IP packet

| ... | Src IP | **Dest. IP** | payload |
|---|---|---|---|

# SUSE SLES 9 Network Configuration Basics

Hardware **devices** — Logical **interfaces**

Configuration files:

**`/etc/sysconfig/hardware`**          **`/etc/sysconfig/network`**

**1:1 relationship**

--> A hardware device always gets the right IP address

Naming convention:

**hw/ifcfg-<device type>-bus-<bus type>-<bus location>** or

**hw/ifcfg-<device type>-id-<identifier>** (e.g. for IUCV)

e.g. **hwcfg-qeth-bus-ccw-0.0.a000**

**ifcfg-qeth-bus-ccw-0.0.a000**

# Setting up a Router

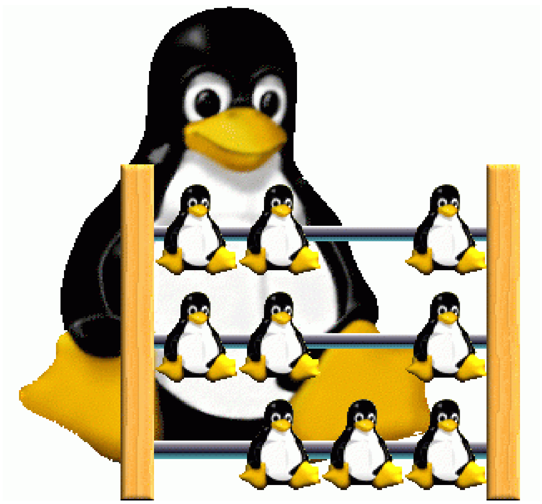
1. Enable IP forwarding:

```
#> sysctl -w net.ipv4.conf.all.forwarding=1
#> sysctl -w net.ipv6.conf.all.forwarding=1
```

or enter the following lines in `/etc/sysconfig/sysctl`:

```
IP_FORWARD=”yes”
IPV6_FORWARD=”yes”
```

to keep the setting persistent.



# Setting up a Router (cont.)

2. Set the router status for your device, e.g. eth0 of the Router (see above):

```
                                                                  *
#> echo primary_router > /sys/class/net/eth0/device/route4
#> echo primary_router > /sys/class/net/eth0/device/route6
```

or enter the following line in the appropriate hwcfg-file to keep the setting persistent:

```
QETH_OPTIONS='route4=primary_router *
              route6=primary_router'
```

*) Other possible values:

```
secondary_router
multicast_router
primary_connector
secondary_connector
no_router
```

(to reset)

# Querying the Router Status

Router status is displayed in `/proc/qeth`:

```
#> cat /proc/qeth                                                            *
devices                    CHPID interface  cardtype       ... rtr4 rtr6
-------------------------- ----- ---------- --------------     ---- ---- ...
0.0.a000/0.0.a001/0.0.a002 xA0   eth0       OSD_1000           pri  pri
0.0.b000/0.0.b001/0.0.b002 x01   eth1       GuestLAN QDIO      pri  pri
0.0.c000/0.0.c001/0.0.c002 xC0   hsi0       HiperSockets       p+c  no
```

or can be retrieved from sysfs:

```
#> cat /sys/class/net/eth0/device/route4*
primary router
#> cat /sys/devices/qeth/0.0.a000/route6
primary router
```

note the alternative ways to your device

*) All possible values:

| procfs | sysfs | Description |
|---|---|---|
| pri | primary router | Primary Router |
| sec | secondary router | Secondary Router |
| mc | multicast router | Multicast Router |
| mc+ | multicast router+ | Multicast Router with broadcast filtering |
| p.c | primary connector | Primary Connector |
| p+c | primary connector+ | Primary Connector with broadcast filtering |
| s.c | secondary connector | Secondary Connector |
| s+c | secondary connector+ | Secondary Connector with broadcast filtering |

# Setting up a Router on Linux 2.4

Set the router status for your device via `/proc/qeth`:

```
#> echo primary_router4 > /proc/qeth*
#> echo primary_router6 > /proc/qeth
```

or enter the following lines in `/etc/chandev.conf`:

```
qeth0,0xa000,0xa001,0xa002                    *
add_parms,0x10,0xa000,0xa002,primary_router4
add_parms,0x10,0xa000,0xa002,primary_router6
```

*) Other possible values: `secondary_router[4|6]`
`multicast_router`
`primary_connector`
`secondary_connector`
`no_router[4|6]` (to reset)

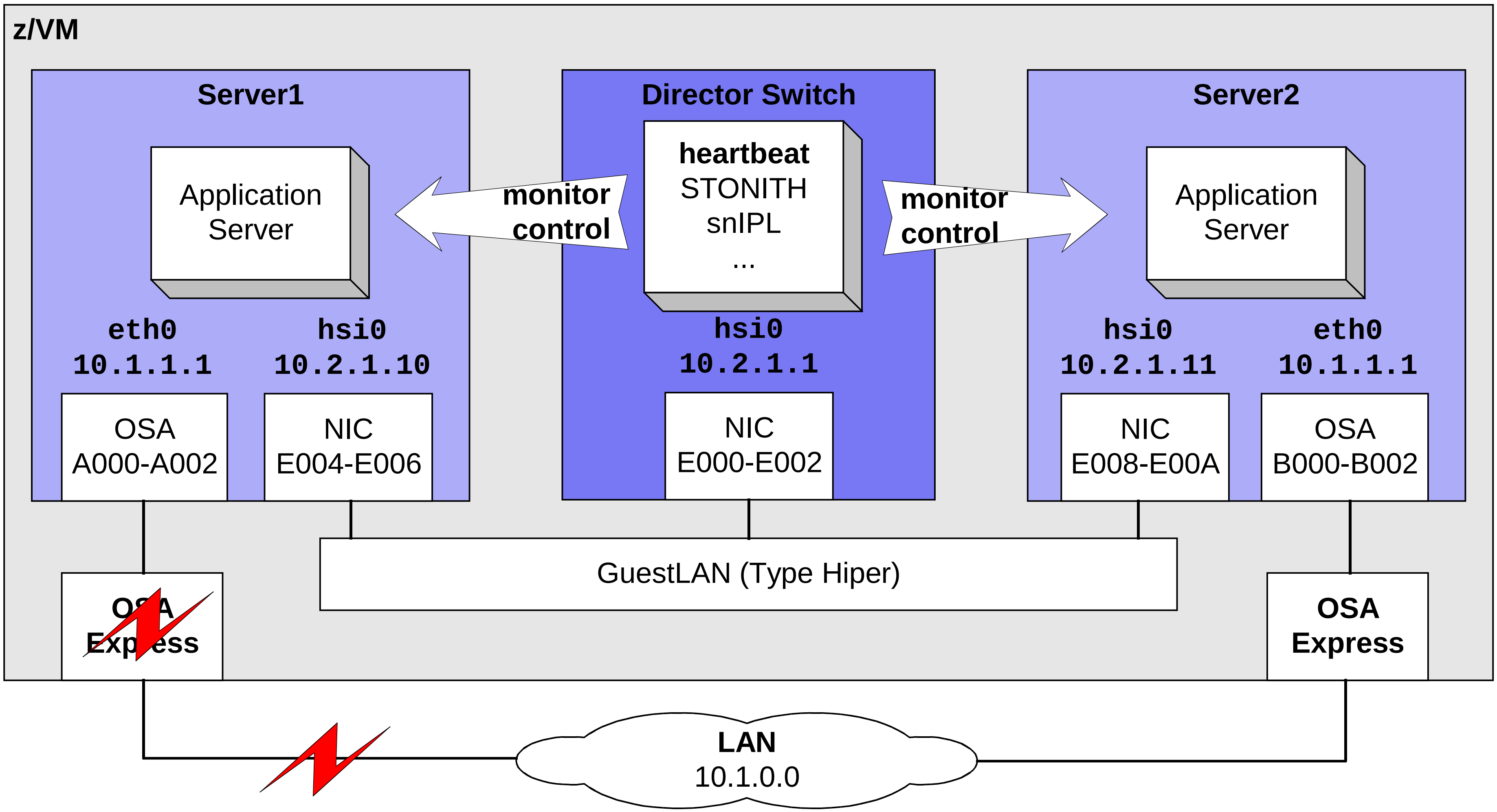
Query router status: `cat /proc/qeth`

# IP Address Takeover

- Idea of IP Address Takeover:
  - Setting an IP address on Linux always succeeds
  - No failure due to ARP (Address Resolution Protocol) conflicts
  - On shared OSA cards, HiperSockets or GuestLAN IP addresses are really taken away from previous owner (in OSA Address Table)
- Takeover must be enabled on both systems
- Takeover of IP addresses is secured by keys:
  Only systems with same key can take over their IP addresses

# IP Address Takeover

Enables implementation of failover strategies




Server2 takes over role of Server1 in case of connectivity problems

# IP Address Takeover Details

- Server images are monitored and controlled by a switch image
- In case of problems the switch initiates takeover
- Management of server nodes done, e.g. via open source **Heartbeat** framework of the High-Availability Linux Project http:/linux-ha.org
- Part of Heartbeat: **STONITH** (Shoot The Other Node In The Head) (http:/linux-ha.org/stonith.html)
- Adaptation of STONITH to fit actual target systems through plugins

# IP Address Takeover Details (cont.)

- STONITH plug in for Linux on zSeries: **snIPL** (simple network IPL)
- snIPL is a Linux image control tool for LPAR and z/VM
  - Can boot, stop, reset Linux images, send and receive OS messages
- On LPAR
  - Uses management application programming interfaces (APIs) of HMC/SE (Hardware Management Console/Support Element)
  - communicates via SNMP (Simple Network Management Protocol)
- On z/VM
  - Utilizes system managements APIs of z/VM 4.4 (or higher)
  - Communicates with z/VM host via RPC (Remote Procedure Call) over internal network connections

# IP Address Takeover Device Attributes

**IP Address Takeover configuration is done per device via sysfs attributes**

```
/sys
|--devices
    |--qeth
        |--0.0.<devno>
            |--ipa_takeover
                |--add4        add/display IPv4 takeover ranges
                |--add6        add/display IPv6 takeover ranges
                |--del4        delete IPv4 takeover ranges
                |--del6        delete IPv6 takeover ranges
                |--enable      enable takeover
                |--invert4     invert IPv4 takeover ranges
                |--invert6     invert IPv6 takeover ranges
```

# IP Address Takeover Setup

1. Enable IP Address Takeover for your device by adding

```
QETH_IPA_TAKEOVER=1
```

to the appropriate hwcfg-file.

- Takeover will be enabled the next time the Linux Image is booted
- To activate the setting on a running Linux:

Stop the device: `#> hwdown qeth-bus-ccw-0.0.a000`

Re-initialize device with new settings:

```
#> hwup qeth-bus-ccw-0.0.a000
```

# IP Address Takeover Setup (cont.)

2. Specify address ranges for IP Address Takeover

```
#> echo <IP address>/<mask> >
   /sys/class/net/eth0/device/ipa_takeover/add4
```

For example, to handle all IP addresses in the range 10.1.1.1 to 10.1.1.254 in takeover mode, issue:

```
#> echo 10.1.1.0/24 >
   /sys/class/net/eth0/device/ipa_takeover/add4
```

Ranges can be inverted, i.e. all addresses except those in a specified range are handled in takeover mode:

```
#> echo 1 >
   /sys/class/net/eth0/device/ipa_takeover/invert4
```

# IP Address Takeover Setup (cont.)

3. To check current IP Address Takeover ranges, issue

```
#> cat /sys/class/net/eth0/device/ipa_takeover/add4
10.1.1.0/24
10.1.2.0/25
```

--> All IP addresses from 10.1.1.1 to 10.1.1.254 and from 10.1.2.1 to 10.1.2.127 are handled in takeover mode when being set with '**`ifconfig`**' or '**`ip addr add`**'

4. Takeover ranges can be deleted by writing to the del4 attribute:

```
#> echo 10.1.2.0/25 >
   /sys/class/net/eth0/device/ipa_takeover/del4
```

IPv6 is analogous, i.e. add/delete ranges via `add6/del6`

# IP Address Takeover Setup on Linux 2.4

1. To enable takeover on a device, add this line to `/etc/chandev.conf`:

```
qeth0,0xa000,0xa001,0xa002
add_parms,0x10,0xa000,0xa002,enable_takeover
```

2. Takeover ranges are set via `/proc/qeth_ipa_takeover`:

```
#> echo add4 <addr in hex>/<mask>[:interface] >*
   /proc/qeth_ipa_takeover
```

e.g.

```
#> echo add4 0a010100/24:eth0 > /proc/qeth_ipa_takeover
```

*) Other possible commands:

```
del4 <addr>/<mask>[:interface]
add6 <addr>/<mask>[:interface]
del6 <addr>/<mask>[:interface]
inv4
inv6
```

If the interface parameter is omitted a range is applied to all interfaces.

# IP Address Takeover with OSA Layer2

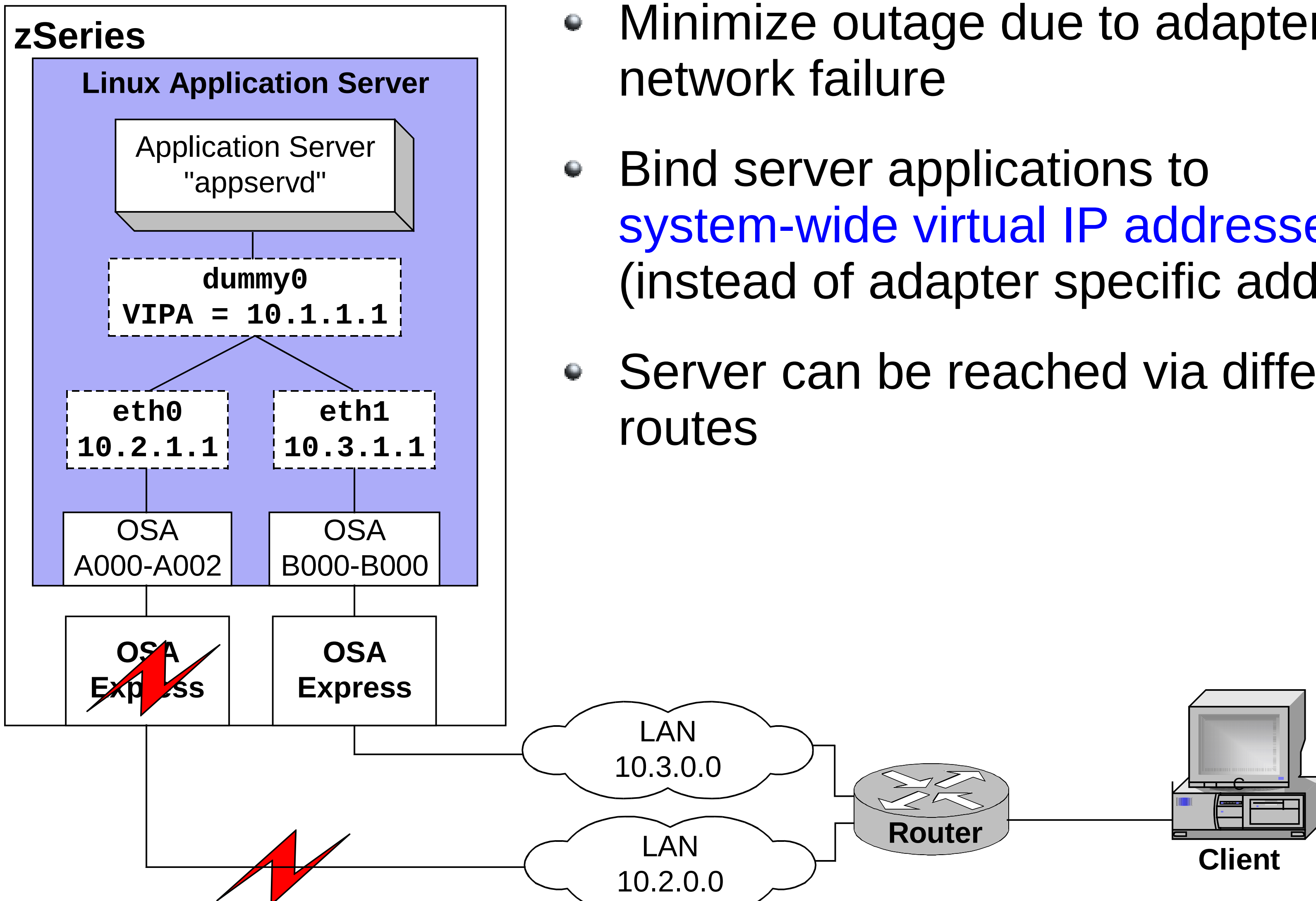
- Echo of IP Address ranges to /sysfs does not work

```
#> cat /sys/class/net/eth0/device/ipa_takeover/add4
10.1.1.0/24
10.1.2.0/25
```

```
QETH_IPA_TAKEOVER=1
```

- Director switch must guarantee that failing system is down
- Director switch must send gratuitous ARP request to refresh ARP tables of connected hosts
- Attention: Some networking hardware does not allow gratuitous ARP !

# Virtual IP Addresses




- Minimize outage due to adapter or network failure
- Bind server applications to system-wide virtual IP addresses (instead of adapter specific addresses)
- Server can be reached via different routes

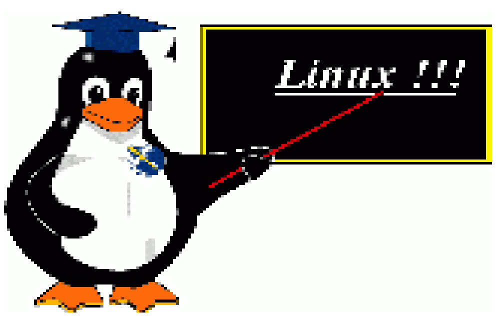
# Virtual IP Addresses (cont.)

- VIPAs are registered with each physical adapter via which a system should be reachable
- Setting a VIPA on an OSA card ensures that packets are handed to Linux image (and are not discarded)
- Linux network stack forwards received packets to configured virtual device (e.g. dummy)
- For outbound use of VIPA the SOURCE VIPA package is required (available at DeveloperWorks)
  - Linux user space layer between application and kernel network stack
  - Sets virtual IP address as source address for sent packets (normally packets get source IP of the interface via which they leave the system)

# Virtual IP Address Device Attributes

**VIPA configuration is done per device via sysfs attributes**

```
/sys
|--devices
    |--qeth
        |--0.0.<devno>
            |--vipa
                |--add4      add/display IPv4 VIPAs
                |--add6      add/display IPv6 VIPAs
                |--del4      delete IPv4 VIPAs
                |--del6      delete IPv6 VIPAs
```



# Virtual IP Address Setup

1. Create a virtual interface and assign the VIPA using a dummy interface:

```
#> modprobe dummy
#> ifconfig dummy0 10.1.1.1 netmask 255.255.0.0
```

or using an interface alias:

```
#> ifconfig eth0:1 10.1.1.1 netmask 255.255.0.0
```

2. Register the virtual IP address with physical devices:

```
#> echo 10.1.1.1 > /sys/class/net/eth0/device/vipa/add4
#> echo 10.1.1.1 > /sys/class/net/eth1/device/vipa/add4
```

3. On the router add a route to the routing table:

`#> route add -host 10.1.1.1 gw 10.2.1.1` if LAN1 works

`#> route add -host 10.1.1.1 gw 10.3.1.1` if LAN2 works

or, better, configure the routes with a dynamic routing daemon (e.g. zebra: http://www.zebra.org).

# Virtual IP Address Setup (cont.)

- VIPA settings can be checked by reading the `add4` attribute:

```
#> cat /sys/class/net/eth0/device/vipa/add4
10.1.1.1
#> cat /sys/class/net/eth1/device/vipa/add4
10.1.1.1
```

- VIPAs are deleted by writing to the `del4` attribute:

```
#> echo 10.1.1.1 > /sys/class/net/eth0/device/vipa/del4
#> echo 10.1.1.1 > /sys/class/net/eth1/device/vipa/del4
```

- IPv6 is analogous, using the `add6` and `del6` attributes.

# Virtual IP Address Setup on Linux 2.4

1. Creation of a virtual interface is done like in Linux 2.6
2. Register the virtual IP address via `/proc/qeth_ipa_takeover`:

```
#> echo add_vipa4 <addr in hex>:<interface> >*
   /proc/qeth_ipa_takeover
```

e.g.

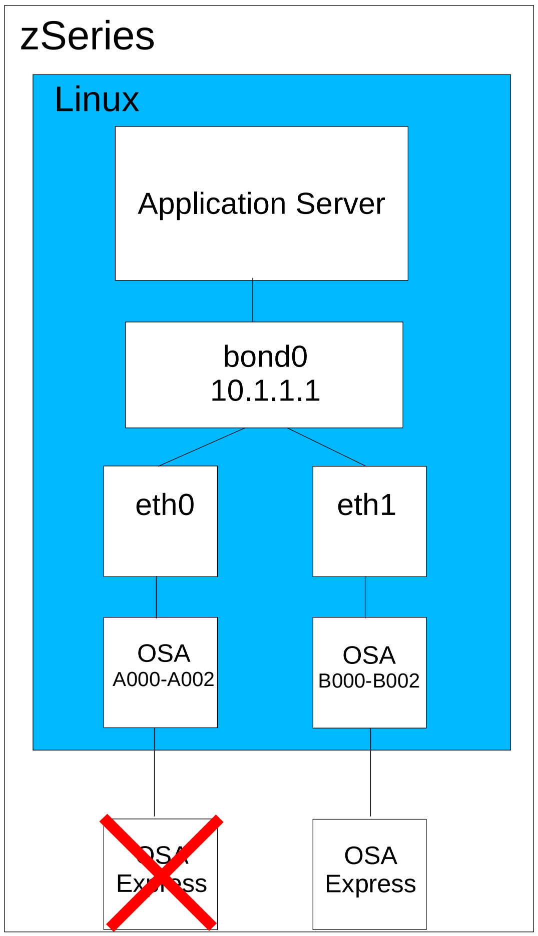
```
#> echo add_vipa4 0a010101:eth0 > /proc/qeth_ipa_takeover
```

*) Other possible commands:

```
del_vipa4 <addr>:<interface>
add_vipa6 <addr>:<interface>
del_vipa6 <addr>:<interface>
```

# Channel Bonding

## Virtual IP Addresses with OSA Layer2




- VIPA functionality with OSA Layer2
- provides failover functionality
- better performance depending on bonding mode
- transparent for LAN infrastructure
- latest setup description:

http://sourceforge.net/projects/bonding/

- Both adapter must be connected to the same LAN !
- Otherwise use Layer 3 approach with routing daemon (e.g. zebra, quagga)

# Channel bonding setup

- Add MAC address to eth0 eth1 (not necessary for GuestLAN)

```
#> ifconifg eth0 hw ether 00:06:29:55:2A:01
#> ifconifg eth1 hw ether 00:05:27:54:21:04
```

- Load bonding module with miimon option (otherwise bonding will will not detect link failures)

```
#> modprobe bonding miimon=1000
```

- Bring up bonding device bond0

```
#> ifconfig bond0 10.1.1.1 netmask 255.255.255.0
```

- connect eth0 & eth1 to bond0

```
#> ifenslave bond0 eth0 eth1
```

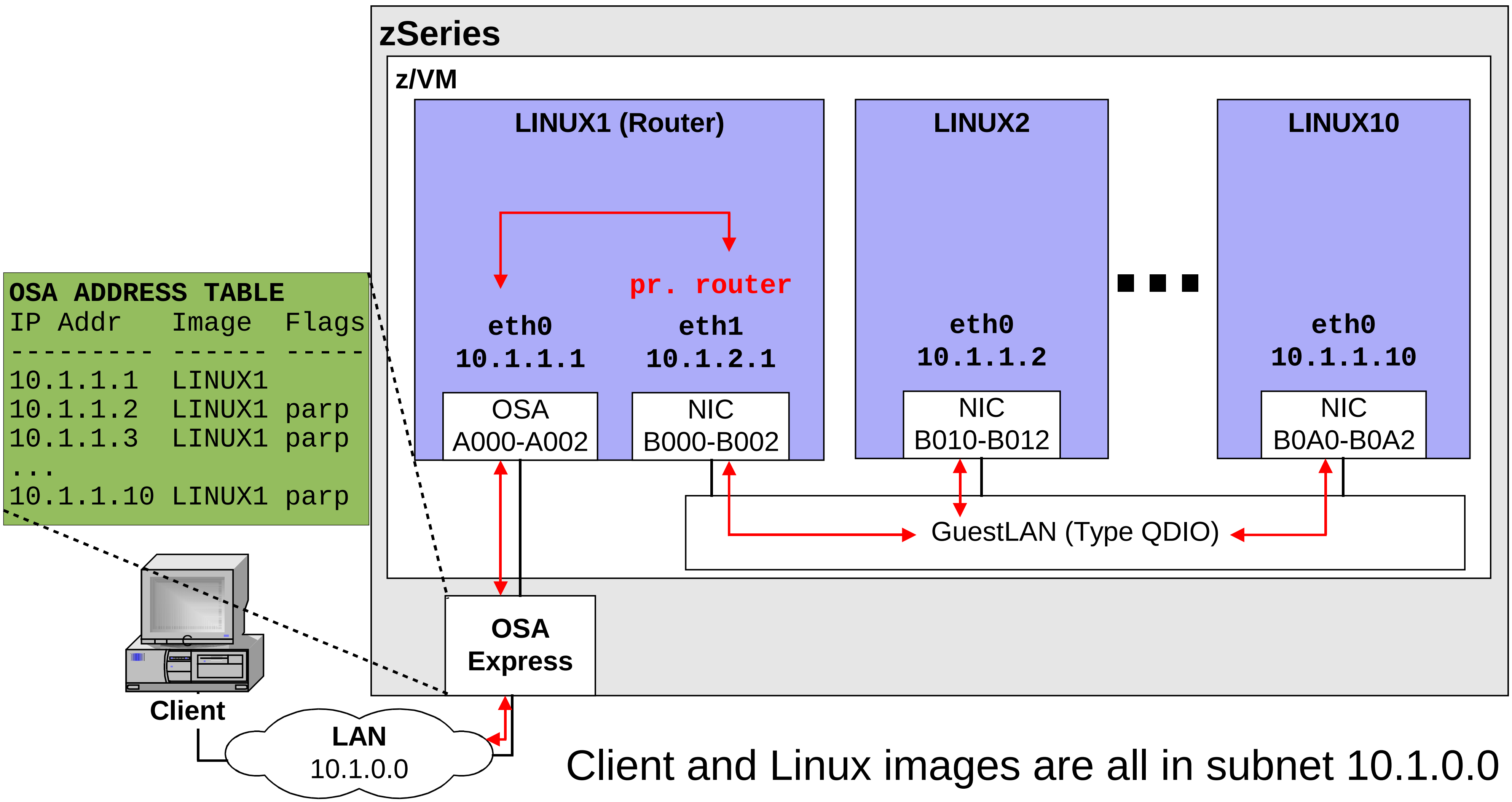
# Channel bonding setup

```
#> ifconifg
bond0     Link encap:Ethernet  HWaddr 00:06:29:55:2A:01
          inet addr:10.1.1.1  Bcast:10.255.255.255 ...

eth0      Link encap:Ethernet  HWaddr 00:06:29:55:2A:01
          UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500...

eth1      Link encap:Ethernet  HWaddr 00:06:29:55:2A:01
          UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500 ...
```

```
#> cat /proc/net/bonding/bond0

Bonding Mode: load balancing (round-robin)
MII Status: up
MII Polling Interval (ms): 1000

Slave Interface: eth0
MII Status: up
Permanent HW addr: 00:06:29:55:2A:01

Slave Interface: eth1
MII Status: up
Permanent HW addr: 00:05:27:54:21:04
```

# Proxy ARP

Example: Integration of a virtual LAN into a real LAN




Client and Linux images are all in subnet 10.1.0.0

# Proxy ARP Details

- OSA card handles ARP requests for all configured Proxy ARP addresses
- Gratuitous ARP packets are sent out to advertise Proxy ARP addresses
- Completely transparent to outside clients
- Seamless integration of arbitrary virtual networks (HiperSockets, GuestLAN, IUCV, virtual CTC) into a real LAN

# Proxy ARP Device Attributes

```
/sys
|--devices
   |--qeth
      |--0.0.<devno>
         |--rxip
            |--add4
            |--add6
            |--del4
            |--del6
```

**Proxy ARP configuration is done per device via sysfs attributes**

- add4: add/display IPv4 Proxy ARP entries
- add6: add/display IPv6 Proxy ARP entries
- del4: delete IPv4 Proxy ARP entries
- del6: delete IPv6 Proxy ARP entries

# Proxy ARP Setup

1. Create Proxy ARP entries for leaf nodes (i.e. with no own connection to outside LAN):

```
#> echo 10.1.1.2 > /sys/class/net/eth0/device/rxip/add4
#> echo 10.1.1.3 > /sys/class/net/eth0/device/rxip/add4
#> echo 10.1.1.4 > /sys/class/net/eth0/device/rxip/add4
...
```

2. Proxy ARP settings can be checked by reading `add4` attribute:

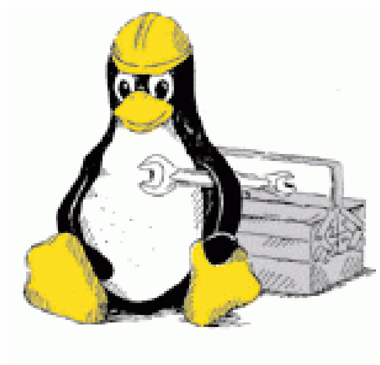
```
#> cat /sys/class/net/eth0/device/rxip/add4
10.1.1.2
10.1.1.3
10.1.1.4
...
```

# Proxy ARP Setup (cont.)

- Proxy ARP entries are deleted by writing to the `del4` attribute:

```
#> echo 10.1.1.2 > /sys/class/net/eth0/device/rxip/del4
#> echo 10.1.1.3 > /sys/class/net/eth0/device/rxip/del4
#> echo 10.1.1.4 > /sys/class/net/eth0/device/rxip/del4
...
```

- IPv6 is analogous, using the `add6` and `del6` attributes.



# Proxy ARP Setup on Linux 2.4

1. Register Proxy ARP entries via `/proc/qeth_ipa_takeover`:

```
#> echo add_rxip4 <addr in hex>:<interface> >*
   /proc/qeth_ipa_takeover
```

e.g.

```
#> echo add_rxip4 0a010102:eth0 > /proc/qeth_ipa_takeover
```

*) Other possible commands:

```
del_rxip4 <addr>:<interface>
add_rxip6 <addr>:<interface>
del_rxip6 <addr>:<interface>
```

# Proxy ARP with OSA layer2

- Echo of Proxy ARP Address ranges to /sysfs does not work

```
#> echo 10.1.2.2 > /sys/class/net/eth0/device/rxip/add4
```

- Enable IP forwarding and Proxy ARP at Linux 2

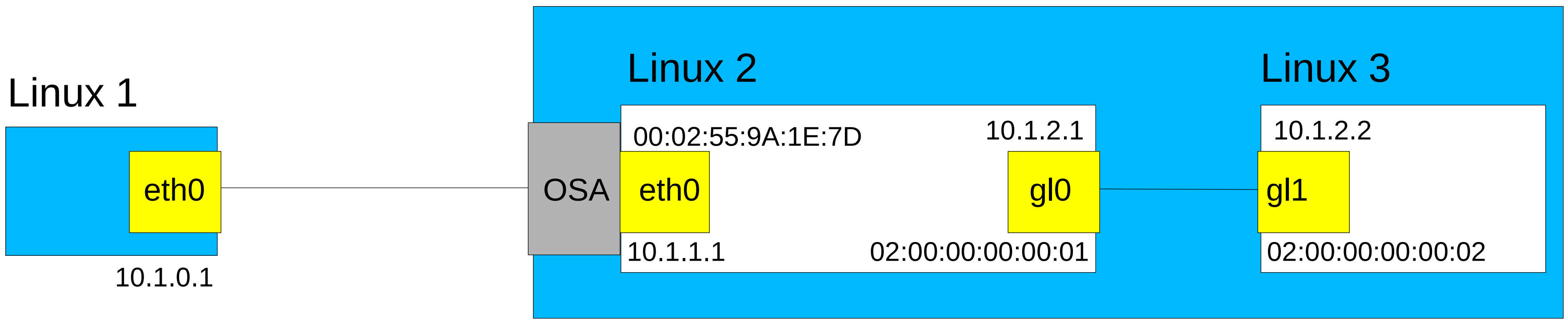
```
#> sysctl -w net.ipv4.ip_forward=1
#> sysctl -w net.ipv4.conf.eth0.proxy_arp=1
```

- Add MAC Address of gl1 to ARP cache of eth0 on Linux 2

```
#> arp -s 10.1.2.2 02:00:00:00:00:02
```

- ARP table of Linux2

```
#> arp -a
? (10.1.2.2) at 02:00:00:00:00:02 [ether] PERM on gl0
```



# The qethconf Tool

- Command line utility to configure IP Address Takeover, VIPA and Proxy ARP
- Part of the **s390-tools** package available at DeveloperWorks
- Consistent user interface on Linux 2.4 and 2.6

| Linux 2.4 | Linux 2.6 |
| --- | --- |
| `qethconf` script | |
| `/proc/qeth_ipa_takeover` | `/sys/.../qeth/0.0.*/ipa_takeover` |
| | `/sys/.../qeth/0.0.*/rxip` |
| | `/sys/.../qeth/0.0.*/vipa` |

- Especially useful: list commands iterate over all devices found in Linux 2.6 sysfs

# qethconf – Examples

IP Address Takeover configuration:

```
#> qethconf ipa add 10.1.1.0/24 eth0
```

Proxy ARP configuration:

```
#> qethconf rxip add 10.1.1.2 eth0
```

VIPA configuration:

```
#> qethconf vipa add 10.1.1.1 eth0
#> qethconf vipa add 10.1.1.1 eth1
```

```
#> qethconf vipa list
vipa add 10.1.1.1 eth0
vipa add 10.1.1.1 eth1
```

Display all IP Address Takeover, VIPA and Proxy ARP settings:

```
#> qethconf list_all
ipa add 10.1.1.0/24 eth0
ipa add 10.1.1.0/24 eth1
rxip add 10.1.1.2 eth0
vipa add 10.1.1.1 eth0
vipa add 10.1.1.1 eth1
```

# The qetharp Tool

- Command line utility to query the ARP cache of OSA and HiperSockets devices
- Part of **s390-tools** package available on DeveloperWorks
- Query of OSA devices returns real ARP cache and entries of local OSA Address Table (OAT)
- Query of HiperSockets devices returns entries of address table associated to a HiperSockets CHPID, i.e. all IP addresses that are currently set on that CHPID
- Currently not supported on VM GuestLAN
- Not working on hosts running with layer2

# qetharp – Examples

Query ARP cache of an OSA Express card:

```
#> qetharp -nq eth1
Address                    HWaddress            HWType    Iface
10.30.151.31               02:00:00:00:29:29    ether     eth1
10.30.30.13                00:09:6b:1a:0c:ed    ether     eth1
10.30.30.7                 00:09:6b:1a:0c:ed    ether     eth1
10.30.130.17               00:09:6b:1a:0c:ed    ether     eth1
10.30.30.9                 00:09:6b:1a:0c:ed    ether     eth1
10.30.130.111              00:09:6b:1a:0c:ed    ether     eth1
```

remote

local OAT

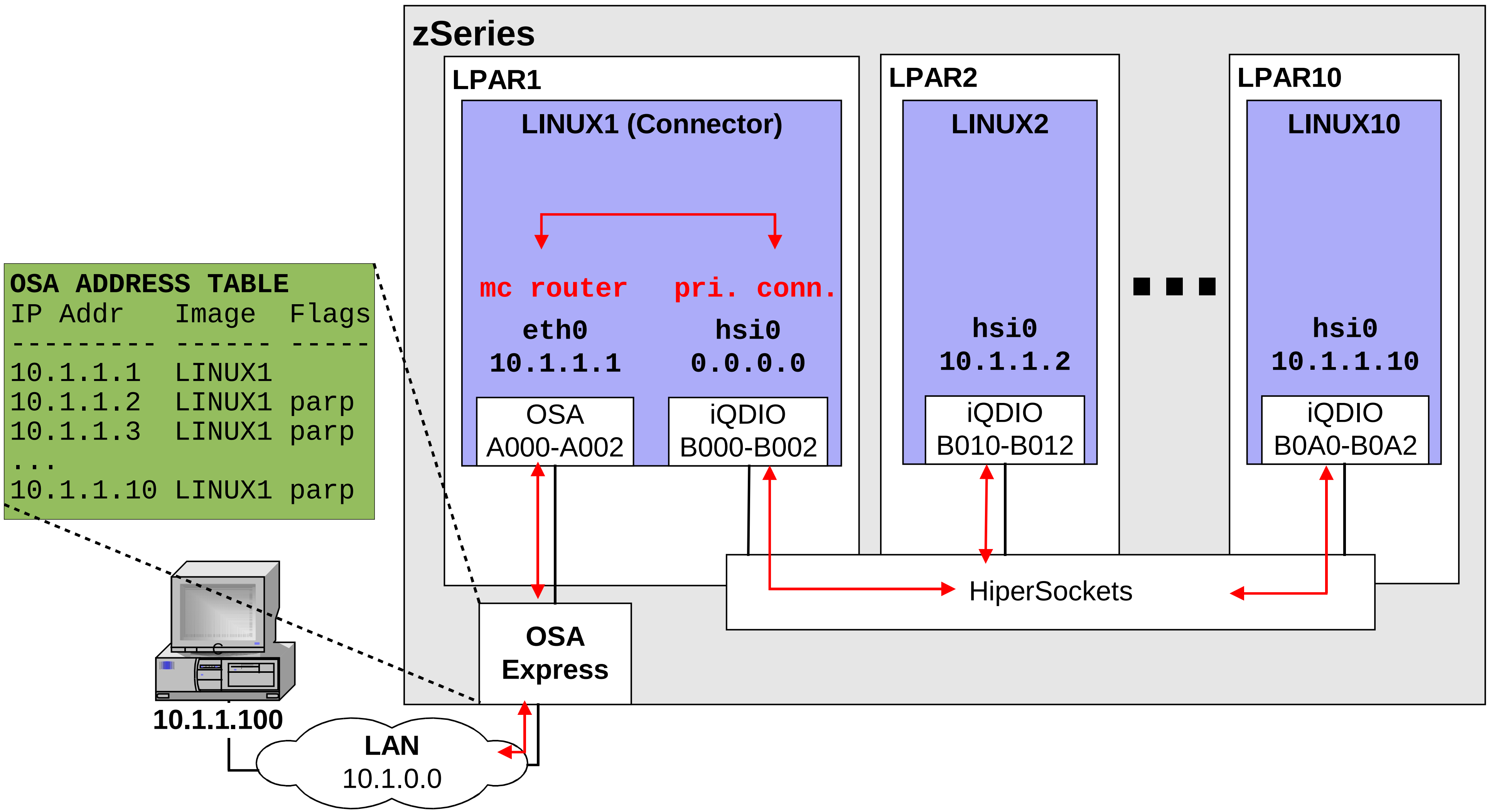
Query address table of a HiperSockets CHPID:

```
#> qetharp -nq hsi0
Address                    HWaddress            HWType    Iface
10.1.3.1                                        hiper     hsi0
10.1.3.2                                        hiper     hsi0
10.1.3.8                                        hiper     hsi0
10.1.3.10                                       hiper     hsi0
```

# HiperSockets Network Concentrator

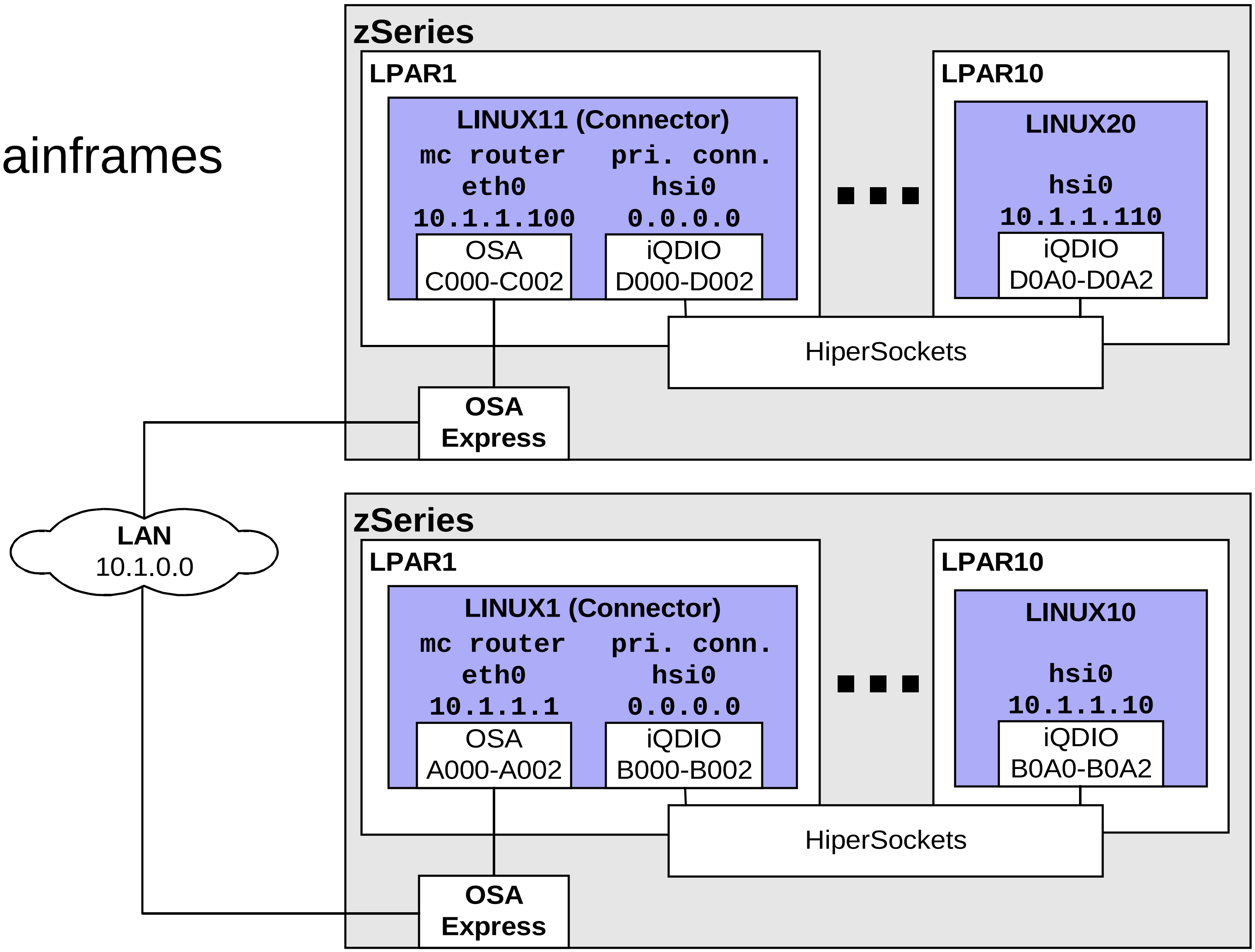
- What is HSNC:
  - A combination of tools for enhanced HiperSockets connectivity
  - Part of **s390-tools** package available at DeveloperWorks
- Enables integration of nodes in a HiperSockets network into an external LAN (i.e. one IP subnet)
- Enables creation of IP subnets across multiple HiperSockets networks on different CECs (Central Electronic Complex) “XCEC HiperSockets”
- Except for connector nodes, completely transparent for attached operating system images
- **Simplification of network topologies** and server consolidation efforts
- Does not work with OSA layer2 at the moment

# HSNC – Topology Example 1

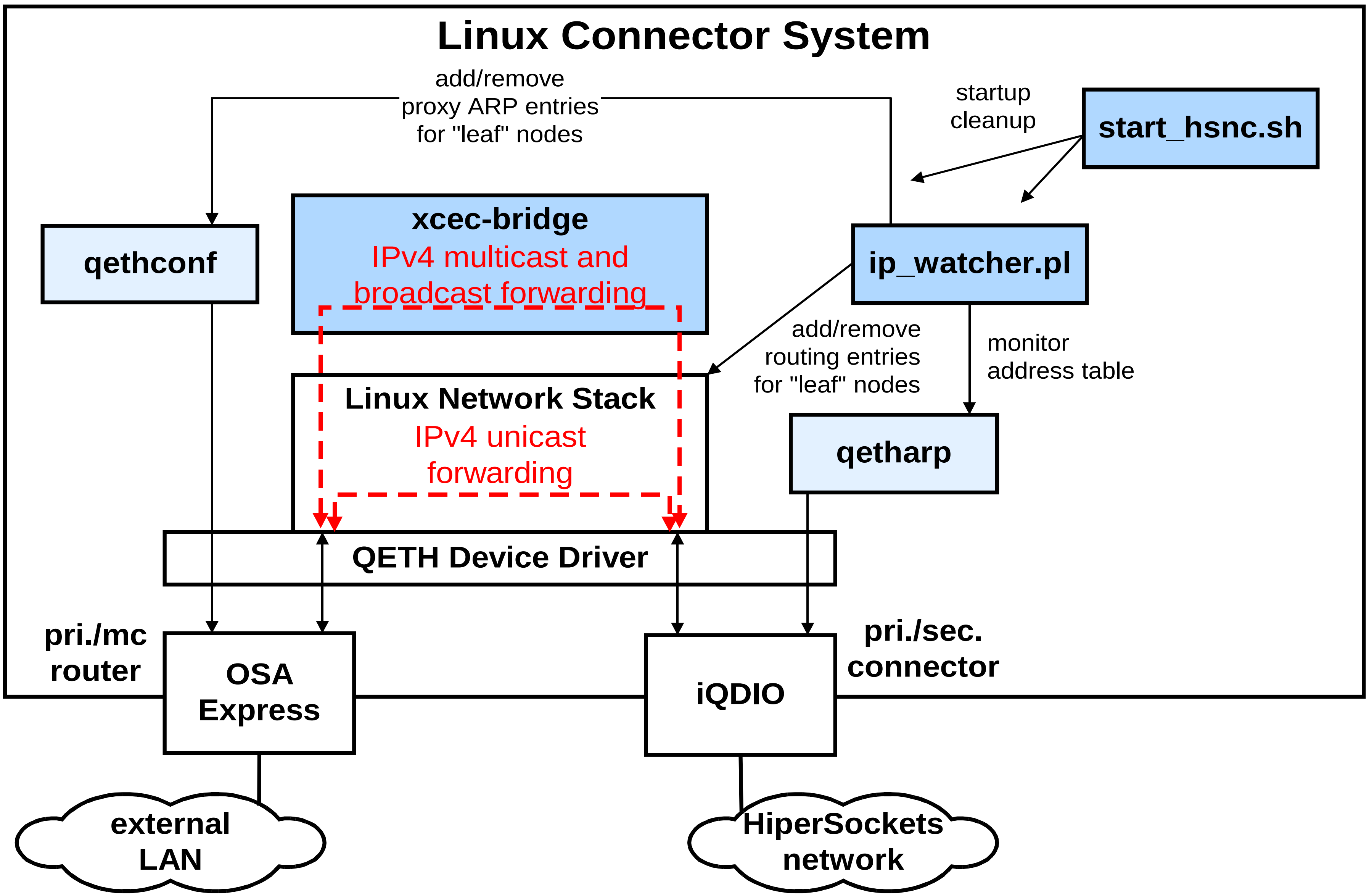


# HSNC – Topology Example 2

Different mainframes or XCEC



# HSNC – How it works



# HSNC – Setup of Connector System

1. Configure your OSA device as multicast router: *

```
#> echo multicast_router > /sys/class/net/eth0/device/route4
```

If no multicast forwarding is desired, use `primary_router`

2. Configure your HiperSockets device as primary connector: *

```
#> echo primary_connector > /sys/class/net/hsi0/device/route4
```

*Optional: You can configure a backup Connector System for failover strategies. Use `secondary_router` for the OSA device and `secondary_connector` for the HiperSockets device on the backup system.

# HSNC – Setup of Connector System (cont.)

3. Check the routing configuration:

```
#> cat /proc/qeth
devices                    CHPID interface  cardtype       ... rtr4 rtr6
-------------------------- ----- ---------- --------------     ---- ---- ...
0.0.a000/0.0.a001/0.0.a002 xA0   eth0       OSD_1000           mc+  no
0.0.b000/0.0.b001/0.0.b002 xB0   hsi0       HiperSockets       p+c  no
```

The '+' sign indicates broadcast filtering capability. This is required for broadcast forwarding.

4. Enable IP forwarding:

```
#> sysctl -w net.ipv4.ip_forward=1
```

5. Start HSNC:

```
#> start_hsnc.sh
```

The OSA device can be specified as start option. This enables unicast forwarding only.

# References

- Linux for zSeries and S/390 on DeveloperWorks

  http://www-128.ibm.com/developerworks/linux/linux390/index.html

- Linux for zSeries and S/390 Documentation

  http://www-128.ibm.com/developerworks/linux/linux390/april2004_documentation.html

- Linux for zSeries and S/390, useful add-ons

  http://www-128.ibm.com/developerworks/linux/linux390/useful_add-ons.html