Networking with Linux®
on System z

SHARE Orlando – Session 9267
02/28/08

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.
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 Torvalds.
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

- Linux 2.6 device model
- Linux on System z network device drivers
- Configuration of network devices
  - SUSE SLES10
  - RedHat RHEL5
- Channel Bonding
- VLAN
Linux 2.6 Device Model

- Integrated uniform device model that reflects a system's hardware structure
- Simplified device reference counting and locking
- Unified user interface via sysfs
  - Hierarchical, tree-like representation of system's hardware
  - Several subsystems provide different views of the hardware
  - Configuration of devices via attribute files
  - Dynamic attach/detach of devices possible
Block subsystem (view):
Block devices and partitions (dasda, ram0)

Bus subsystem (view):
Device drivers and devices sorted by bus (ccw)

Class subsystem (view):
Logical devices sorted by type, i.e. to which class they belong;
Logical devices have link to hardware device

Devices subsystem (view):
All the devices of a system
**Linux 2.6 Device Model – System z Examples**

/sys
|--block
 ||--dasda
 ||...
|--bus
 ||--ccw
 ||--ccwgroup
 | ||--devices
 | | ||--0.0.a000
 | ||--drivers
 | | ||--lcs
 | | ||--qeth
 | | | ||--0.0.a000
|--css
|--class
 ||--net
 | ||--eth0
 | |||--device
|--devices
 ||--qeth
 | ||--0.0.a000

**Block Devices:**
DASD, RAM-Disk, Minidisk
SCSI, Loopback

**CCW Group Devices:**
QETH, LCS

Example: a QETH device

Many ways to find a device
Linux for System z Network Device Drivers

- QETH
- LCS
- CTC (stabilized)
- NETIUCV (stabilized)
QETH Device Driver

Includes support for:

- OSA Express(2) - Fast/Giga/10GBit Ethernet
- 1000Base-T Ethernet
- Highspeed Tokenring
- ATM (running Ethernet LAN Emulation)
- z/VM GuestLAN  Type QDIO (layer2 / layer3), Type Hiper
- z/VM VSWITCH  (layer2 / layer3)
- System z HiperSockets
- IPv4, IPv6, VLAN, VIPA, Proxy ARP, IP Address Takeover, Channel Bonding

Primary network driver for Linux on System z

Main focus in current and future development
Primary Network Device: OSA Express

- 'Integrated Power computer' with network daughter card
- Shared between up to 640 / 1920 TCP/IP stacks
- OSA Address Table: which OS image has which IP address
- Three devices (I/O subchannels) per stack:
  - Read device (control data <-- OSA)
  - Write device (control data --> OSA)
  - Data device (network traffic)
- Network traffic Linux <---> OSA at IP or ARP level
- One MAC address for all stacks
- OSA handles ARP (Address Resolution Protocol)
The Queued Direct I/O (QDIO) Architecture

Control Data exchange using CCWs (channel command words)

OSA Express

NIC

MAC: 00:06:29:55:2A:09

OSA ADDRESS TABLE

<table>
<thead>
<tr>
<th>IP Addr</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1.1.1</td>
<td>LINUX1</td>
</tr>
<tr>
<td>10.10.1.5</td>
<td>LINUX2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10.11.1.1</td>
<td>z/OS</td>
</tr>
</tbody>
</table>

QDIO as “long-running” channel program

Pages in Linux Memory

E - Empty Buffer
P -Primed Buffer
LCS Device Driver

- LCS – LAN Channel Station
- Supports:
  - OSA-2 Ethernet and Tokenring
  - OSA-Express Fast Ethernet, 1000Base-T Ethernet (z890 and z990) and Highspeed Tokenring (in non-QDIO mode)
  - Since z990: OSA-Express 2 Gigabit Ethernet (in non-QDIO mode)
- May be preferred instead of QETH for security reasons
  - Administrator defines OSA Address Table, whereas with QETH each Linux registers its own IP address --> restricted access

But: performance is inferior to QETH's performance!!!
Message to CTC and IUCV users

- CTC = Channel-to-Channel connection
- IUCV = Inter User Communication Vehicle
- CTC and NETIUCV device drivers are deprecated (LINUX 2.6+)
- Device drivers still available for backward compatibility
- Migrate
  - Virtual CTC and IUCV to guest LAN Hipersockets or guest LAN type QDIO
  - CTC inside a CEC to Hipersockets
  - CTC to OSA-Express (QDIO)
SUSE SLES 10 Network Configuration

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>

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

ifcfg-qeth-bus-ccw-0.0.a000

see /etc/sysconfig/hardware/skel/hwcfg-<device type>
SUSE SLES 10 Network Configuration (cont.)

See also: /usr/share/doc/packages/sysconfig/README and README.s390
Networking Example

System z

LINUX 1

eth0
10.1.1.1

OSA
A000-A002

NIC
B000-B002

evb0
10.2.1.1

LINUX 2

eth0
10.2.1.2

NIC
B003-B005

GuestLAN (Type QDIO)
10.2.0.0

MY_VS
C001-C003

OSA Express

LAN
10.1.0.0

OSA Express

LAN
10.2.0.0
Static QETH Device Setup (SUSE SLES10)

For LINUX 1 eth0

1. Create a hardware device configuration file:

```
/etc/sysconfig/hardware/hwcfg-qeth-bus-ccw-0.0.a000:
  CCW_CHAN_IDS='0.0.a000 0.0.a001 0.0.a002'
  CCW_CHAN_MODE='OSAPORT'
  CCW_CHAN_NUM='3'
  MODULE='qeth'
  MODULE_OPTIONS=''
  MODULE_UNLOAD='yes'
  SCRIPTDOWN='hwdown-ccw'
  SCRIPTUP='hwup-ccw'
  SCRIPTUP_ccw='hwup-ccw'
  SCRIPTUP_ccwgroup='hwup-qeth'
  STARTMODE='auto'
  QETH_LAYER2_SUPPORT='0'
  QETH_OPTIONS='fake_ll=1'
```

further attributes
Static QETH Device Setup (SUSE SLES10) (cont.)

- **CCW_CHAN_IDS** are Read, Write, Data subchannels
  - Hexadecimal characters must be lowercase
- **STARTMODE**  'auto' --> started by hotplug agents
  'manual' --> manual startup
- **QETH_OPTIONS** allows to set optional attributes

  e.g. QETH_OPTIONS=’fake_ll=1’

- A sample hwcfg-file for QETH can be found at
  /etc/sysconfig/hardware/skel/hwcfg-qeth
2. Create an interface configuration file:

```
/etc/sysconfig/network/ifcfg-qeth-bus-ccw-0.0.a000
  BOOTPROTO='static'
  BROADCAST='10.1.255.255'
  IPADDR='10.1.1.1'
  NETMASK='255.255.0.0'
  NETWORK='10.1.0.0'
  STARTMODE='onboot'
```

Explanations are found in

```
/etc/sysconfig/network/ifcfg-qeth-bus-ccw-0.0.a000.template
```

3. Before reboot: test your config files:

```
-> hwup qeth-bus-ccw-0.0.a000
```
RedHat RHEL5 Network Configuration

• Configuration files:

  /etc/modprobe.conf
  alias eth0 qeth
  alias eth1 qeth
  alias hsi0 qeth
  alias eth2 lcs

  /etc/sysconfig/network-scripts/ifcfng-<ifname>
  NETTYPE  qeth | lcs | ctc | iucv
  TYPE     Ethernet | CTC | IUCV
  SUBCHANNELS  0.0.b003, 0.0.b004, 0.0.b005
  PORTNAME
  OPTIONS
  MACADDR

• ifup/ifdown scripts contain mainframe-specifics
Static QETH Device Setup (RedHat RHEL5)

For LINUX 1 eth0

1. Create the configuration file:

/etc/sysconfig/network-scripts/ifcfg-eth0:

DEVICE=eth0
SUBCHANNELS='0.0.a000,0.0.a001,0.0.a002'
PORTNAME=OSAPORT
NETTYPE=qeth
TYPE=Ethernet
BOOTPROTO=static
ONBOOT=yes
BROADCAST=10.1.255.255
IPADDR=10.1.1.1
NETMASK=255.255.0.0
MACADDR='00:09:6B:1A:9A:89'
OPTIONS='layer2=1'

further attributes
2. Add / verify alias in /etc/modprobe.conf:

/etc/modprobe.conf:

... 
alias eth0 qeth 
...

3. For details see:

http://www.redhat.com/docs/manuals/enterprise/
Networking Example

System z

z/VM

LINUX 1

eth0
10.1.1.1

OSA
A000-A002

NIC
B000-B002

LINUX 2

eth1
10.2.1.1

NIC
B003-B005

GuestLAN (Type QDIO)
10.2.0.0

MY_VS
C001-C003

OSA Express

OSA Express

LAN
10.1.0.0

LAN
10.2.0.0
Dynamic QETH Device Setup

For LINUX 2 eth0

1. In your z/VM console (if not already defined in user directory) do

1.1. Create a GuestLAN or VSWITCH

```
#CP DEFINE LAN MY_LAN TYPE QDIO
#CP DEFINE VSWITCH MY_VS RDEV C001 CONTROLLER * IP
```

1.2. Create a virtual NIC

```
#CP DEFINE NIC B003 TYPE QDIO
```

1.3. Couple virtual NIC to GuestLAN/VSWITCH

```
#CP COUPLE B003 TO * MY_LAN
```
Dynamic QETH Device Setup (cont.)

2. Load the QETH device driver module:

```bash
#> modprobe qeth
```

3. Create a new QETH device by grouping its CCW devices:

```bash
#> echo 0.0.b003,0.0.b004,0.0.b005 > /sys/bus/ccwgroup/drivers/qeth/group
```

4. Set optional attributes:

```bash
#> echo 64 > /sys/bus/ccwgroup/drivers/qeth/0.0.b004/buffer_count

#> echo 1 > /sys/devices/qeth/0.0.b004/fake_ll
```

Note the alternative ways to your device
Dynamic QETH Device Setup (cont.)

5. Set the new device online:

```bash
#> echo 1 > /sys/devices/qeth/0.0.b004/online
```

6. Check your QETH devices:

```bash
#> cat /proc/qeth
```

```
devices                    CHPID interface  cardtype
-------------------------- ----- ---------- -------------- ...
0.0.c000/0.0.c001/0.0.c002 xC0   hsi0       HiperSockets
0.0.b003/0.0.b004/0.0.b005 x01   eth0       GuestLAN QDIO ...
```

7. Configure your new eth0 interface:

```bash
#> ifconfig eth0 10.2.1.2 netmask 255.255.0.0
```
**QETH Device sysfs Attribute `fake_ll`**

- Build **fake ethernet headers** before handing packets to the network stack.

- Required by some network applications, e.g. **DHCP** or **TCPDUMP**

```
Ethernet Header:
  src. MAC addr
dest. MAC addr

IP Header:
  src. IP addr
dest. IP addr

IP Header
```

```
Fake Ethernet Hdr
```

```
Payload
```

```
OSA Express
```
**QETH Device sysfs Attribute `large_send`**

- Offload TCP segmentation from Linux network stack to OSA-card
  
  ```
  QETH_OPTIONS='large_send=TSO'
  
  #> echo TSO > /sys/devices/qeth/0.0.0.004/large_send
  
  ===> move workload from Linux to OSA-Express adapter
  ```

- Offload TCP segmentation from Linux network stack to device driver

  ```
  QETH_OPTIONS='large_send=EDDP'
  
  #> echo EDDP > /sys/devices/qeth/0.0.0.004/large_send
  ```

---

**Diagram:**

- **Linux TCP stack**
  - TCP frame up to 64 kbytes

- **QETH device driver**
  - MTU sized TCP frame
  - MTU sized TCP frame
  - MTU sized TCP frame

- **OSA Express card**
  - MTU sized TCP frame
  - MTU sized TCP frame
  - MTU sized TCP frame

---

===> performance advantage with large outgoing packets
QETH Device sysfs Attribute **check_summing**

- Offload checksumming for incoming IP packages from Linux network stack to OSA-card
  
  ```
  QETH_OPTIONS='checksumming=hw_checksumming'
  ```

  ```
  #> echo hw_checksumming > /sys/devices/qeth/0.0.b004/checksumming
  ```

- move workload from Linux to OSA-Express adapter

QETH Device sysfs Attribute **recover**

- enforce recovery of a qeth device

  ```
  #> echo 1 > /sys/devices/qeth/0.0.b004/recover
  ```
The number of allocated buffers for inbound QDIO traffic --> Memory usage.

Per QETH card memory usage:

- control data structures: ~ 200 KB
- memory for one buffer: 64 KB

- buffer_count = 8 --> ~ 712 KB
- buffer_count = 128 --> ~ 8.4 MB

8 buffers 16 buffers (default, recommended) 128 buffers

Save memory Boost performance
QETH Layer 2 mode

- OSA works with MAC addresses ==> no longer stripped from packets.

<table>
<thead>
<tr>
<th>MAC Address</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:06:29:55:2A:01</td>
<td>LINUX1</td>
</tr>
<tr>
<td>00:06:29:55:2A:02</td>
<td>LINUX2</td>
</tr>
<tr>
<td>00:02:02:02:02:02</td>
<td>LINUX3</td>
</tr>
<tr>
<td>00:06:29:55:2A:03</td>
<td>z/VM</td>
</tr>
</tbody>
</table>

- hwcfg-qeth... file (SLES10):
  QETH_LAYER2_SUPPORT=1

- ifcfg-qeth... file (SLES10):
  LLADDR='<MAC Address>'

- ifcfg-... file (RHEL5):
  MACADDR='<MAC Address>'
  OPTIONS='layer2=1'

- Direct attached OSA:
  MAC address must be defined with ifconfig manually
  ifconfig eth0 hw ether 00:06:29:55:2A:01

- with VSWITCH or GuestLAN under z/VM
  MAC address created by z/VM
QETH Layer 2 mode (cont.)

- activating Layer 2 is done per device via sysfs attributes
- possible layer2 values:
  - 0: use device in Layer 3 mode
  - 1: use device in Layer 2 mode

- setting of layer2 attribute is only permitted when device is offline!
- DHCP, tcpdump working without option fake_ll
- channel bonding possible

/sys
|--devices
 ||--qeth
  | |--0.0.<devno>
   | |--layer2
QETH Layer 2 mode (cont.)

- Direct attached OSA
- GuestLAN type QDIO supported
  
  GuestLAN definition for layer2:
  
  ```
  define lan <lanname> ... type QDIO ETHERNET
  define nic <vdev> QDIO
  couple <vdev> <ownerid> <lanname>
  ```

- VSWITCH
  
  ```
  define vswitch <vswname> ... ETHERNET ...
  define nic <vdev> QDIO
  couple <vdev> <ownerid> <lanname>
  ```

- Restrictions:
  
  Layer2 and Layer3 traffic can be transmitted over the same OSA CHPID, but not between two hosts sharing the same CHPID!
Channel Bonding

- The Linux bonding driver provides a method for aggregating multiple network interfaces into a single logical "bonded" interface
- provides failover and / or load-balancing functionality
- better performance depending on bonding mode
- transparent for LAN infrastructure
- latest setup description:

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

1. Add MAC address to eth0 & eth1 (not necessary for GuestLAN or Vswitch)
   ```
   => ifconfig eth0 hw ether 00:06:29:55:2A:01
   => ifconfig eth1 hw ether 00:05:27:54:21:04
   ```

2. Load bonding module with miimon option
   (otherwise bonding will not detect link failures)
   ```
   => modprobe bonding miimon=100 mode=balance-rr
   ```

3. Bring up bonding device bond0
   ```
   => ifconfig bond0 10.1.1.1 netmask 255.255.255.0
   ```

4. Connect eth0 & eth1 to bond0
   ```
   => ifenslave bond0 eth0
   => ifenslave bond0 eth1
   ```
Channel bonding setup (SLES10 – config files)

• interface configuration file for a slave

/etc/sysconfig/network/ifcfg-qeth-bus-ccw-0.0.a000
   BOOTPROTO='static'
   IPADDR=''
   SLAVE='yes'
   STARTMODE='onboot'

• interface configuration file for a master

/etc/sysconfig/network/ifcfg-bond0
   BOOTPROTO='static'
   BROADCAST='10.1.255.255'
   IPADDR='10.1.1.1'
   NETMASK='255.255.0.0'
   NETWORK='10.1.0.0'
   STARTMODE='onboot'
   BONDING_MASTER='yes'
   BONDING_MODULE_OPTS='mode=1 miimon=1'
   BONDING_SLAVE0='qeth-bus-ccw-0.0.a000'
   BONDING_SLAVE1='qeth-bus-ccw-0.0.b000'
Channel bonding setup (cont.)

```bash
#> ifconfig
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): 100

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
```
Virtual LAN (VLAN) support

- Risk of big switched LANs: flooded with broadcast traffic
- Devide LANs logically into subnets
  
  ==> fewer waste of bandwidth
- IEEE Standard 802.1Q
Virtual LAN (VLAN) support (cont.)

- Setup:

  
  ```
  ifconfig eth1 9.164.160.23 netmask 255.255.224.0
  vconfig add eth1 3
  ifconfig eth1.3 1.2.3.4 netmask 255.255.0.0
  ```

- Displaying info:

  ```
  cat /proc/net/vlan/config
  VLAN Dev name | VLAN_ID
  Name-Type: VLAN_NAME_TYPE_RAW_PLUS_VID_NO_PAD
  eth1.3 | 3 | eth1
  ```

- Implemented:
  VLAN tag, added to packets transmitted

- Supported by:
  real OSA-card, z/VM Guest LAN, z/VM VSWITCH
## Interface names

<table>
<thead>
<tr>
<th>Interface Name</th>
<th>Device Driver</th>
<th>Interface / Link Type</th>
<th>Model / Submodel</th>
<th>Used for</th>
</tr>
</thead>
<tbody>
<tr>
<td>eth&lt;x&gt;</td>
<td>qeth</td>
<td>Ethernet</td>
<td>1731/01, 3088/01, 3088/60</td>
<td>OSA-card / type OSD, P390-LCS-card, OSA-card / type OSE</td>
</tr>
<tr>
<td></td>
<td>lcs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>lcs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hsi&lt;x&gt;</td>
<td>qeth</td>
<td>Ethernet</td>
<td>1731/05</td>
<td>HiperSockets / type IQD</td>
</tr>
<tr>
<td>tr&lt;x&gt;</td>
<td>qeth</td>
<td>Token Ring</td>
<td>1731/01, 3088/01, 3088/60</td>
<td>OSA-card / type OSD, P390-LCS-card, OSA-card / type OSE</td>
</tr>
<tr>
<td></td>
<td>lcs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>osn&lt;x&gt;</td>
<td>qeth</td>
<td>SNA&lt;-&gt;Ethernet</td>
<td>1731/06</td>
<td>OSA-card / type OSN</td>
</tr>
<tr>
<td>ctc&lt;x&gt;</td>
<td>ctc</td>
<td>Point-to-Point</td>
<td>3088/08, 3088/1e, 3088/1f virtual</td>
<td>Channel-To-Channel adapter, FICON adapter, ESCON adapter, VM-guest communication</td>
</tr>
<tr>
<td>iucv&lt;x&gt;</td>
<td>netiucv</td>
<td>Point-to-Point</td>
<td>virtual</td>
<td>VM-guest communication</td>
</tr>
</tbody>
</table>
# Summary of Linux Network Device Drivers

<table>
<thead>
<tr>
<th>QETH</th>
<th>OSA</th>
<th>HiperSockets</th>
<th>GuestLAN QDIO</th>
<th>GuestLAN Hiper</th>
<th>LCS</th>
<th>CTC</th>
<th>IUCV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adapters</td>
<td>100 Mbps, 1Gbps, 10Gbps</td>
<td>1000 Base-T, HSTR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connection type</td>
<td>LAN</td>
<td>LAN</td>
<td>LAN</td>
<td>LAN</td>
<td>LAN</td>
<td>*</td>
<td>point-to-point</td>
</tr>
<tr>
<td>Layer</td>
<td>Layer2 / 3</td>
<td>Layer3</td>
<td>Layer2 / 3</td>
<td>Layer3</td>
<td>Layer3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remarks</td>
<td>Primary network device driver for Linux on System z</td>
<td></td>
<td></td>
<td>restricted access (admin defines OSA Address Table)</td>
<td></td>
<td></td>
<td>Deprecated in LINUX 2.6</td>
</tr>
</tbody>
</table>
References

- Linux on System z on developerWorks

- Linux on System z Documentation

- Linux on System z, useful add-ons

- Linux on System z – Tuning Hints & Tips