9267 - Networking with Linux® on System z

Klaus-Dieter Wacker (kdwacker@de.ibm.com)
IBM Development Lab, Boeblingen, Germany

Schedule: Thursday 11:00 AM
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 on System z network device drivers
- Configuration of network devices
  - SUSE SLES10
  - RedHat RHEL5
  - Generic (manual)
- Further networking driver aspects
- Advanced aspects
  - Channel Bonding
  - VLAN
  - Virtual IP Addresses
Linux for System z Network Device Drivers

- QETH
- LCS
- CTC (stabilized)
- NETIUCV (stabilized)
Network Example

System z

z/VM in LPAR

- LINUX 1
  - iucv0 10.5.1.1
  - ctc0 10.6.1.1
  - CTC E000,E001
  - eth0 10.1.1.1
  - NIC B000-B002
  - OSA A000-A002

- IUCV

- CTC/A

- GuestLAN (Type QDIO) 10.2.0.0

- OSA Express

- LAN 10.1.0.0

LPAR

- LINUX 2
  - iucv0 10.5.1.2
  - ctc0 10.6.1.2
  - CTC E000,E001
  - eth0 10.2.1.1
  - NIC B003-B005
  - iQDIO C000-C002

- iQDIO

- HiperSockets 10.3.0.0

- LCS Card

- LAN 10.4.0.0

- hsi0 10.3.1.3

-eth0 10.4.1.3

LINUX 3

- iQDIO C003-C005

- LCS D000-D001

- LAN 10.4.0.0

- hsi0 10.3.1.3

-eth0 10.4.1.3
Linux 2.6 Device Model – System z Examples

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

CCW Group Devices:
QETH, LCS

Example: a QETH device

Many ways to find a device
LCS Device Driver

- LCS – LAN Channel Station
- Supports:
  - OSA Express(2) (in non-QDIO mode)
    - Fast Ethernet
    - 1000Base-T Ethernet
    - HighSpeed TokenRing (<= z990)
    - ATM (running Ethernet LAN Emulation) (<= z990)
- 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 (under z/VM) to guest LAN HiperSockets or guest LAN type QDIO
  - CTC inside a CEC to Hipersockets
  - CTC to OSA-Express (QDIO)
QETH Device Driver

- Supports:
  - OSA Express(2)
    - Fast/Giga/10GBit Ethernet
    - 1000Base-T Ethernet
    - HighSpeed TokenRing (<= z990)
    - ATM (running Ethernet LAN Emulation) (<= z990)
  - System Z HiperSockets
  - z/VM
    - GuestLAN Type QDIO (layer2 / layer3), Type Hiper
    - z/VM VSWITCH (layer2 / layer3)
- 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 (layer 3) or ethernet (layer 2) level
- One MAC address for all stacks (layer 3)
- OSA handles ARP (layer 3) (Address Resolution Protocol)
The Queued Direct I/O (QDIO) Architecture

- **Control Data exchange using CCWs (channel command words)**
- **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>
```

- **OSA Express**

---

**Legend:**
- E - Empty Buffer
- P - Primed Buffer

**QDIO as “long-running” channel program**
SUSE SLES 10 Network Configuration

Hardware **devices** ↔ Logical **interfaces**

Configuration files:

- `/etc/sysconfig/hardware`
- `/etc/sysconfig/network`

1:1 relationship

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`

Scripts: `hwup / hwdown` and `ifup / ifdown`

see `/etc/sysconfig/hardware/skel/hwcfg-<device type>`
/new/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

eth1
10.2.1.1

LINUX 2

eth0
10.2.1.2

NIC
B003-B005

GuestLAN (Type QDIO)
10.2.0.0

OSA Express

LAN
10.1.0.0
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='checksumming=hw_checksumming'
```
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='checksumming=hw_checksumming'

- A sample hwcfg-file for QETH can be found at
  
  /etc/sysconfig/hardware/skel/hwcfg-qeth
Static QETH Device Setup (SUSE SLES10) (cont.)

2. Create an interface configuration file:

```bash
/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.template
```

3. Before reboot: test your config files:

```bash
#> 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/ifcfg-<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:

```bash
/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
  OPTIONS='checksumming=hw_checksumming'
```

further attributes
Static QETH Device Setup (RedHat RHEL5) (cont.)

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

/etc/modprobe.conf:

```bash
... 
alias eth0 qeth
... 
```

3. For details see:

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

System z

LINUX 1
- eth0
  - 10.1.1.1
- OSA A000-A002
- NIC B000-B002

LINUX 2
- eth1
  - 10.2.1.1

NIC B003-B005
- 10.2.1.2

GuestLAN (Type QDIO)
- 10.2.0.0

VMTCPIP
- NIC B097-B099
- OSA C001-C003

OSA Express

LAN 10.1.0.0

MY_VS

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

   #CP SET VSWITCH MY_VS GRANT LINUX2

   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_VS
Dynamic QETH Device Setup (cont.)

2. Load the QETH device driver module:

   `#> modprobe qeth`

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

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

4. Set optional attributes:

   `#> echo 64 > /sys/bus/ccwgroup/drivers/qeth/0.0.b003/buffer_count`

   `#> echo 1 > /sys/devices/qeth/0.0.b003/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.b003/online
```

6. Check your QETH devices:

```bash
#> lsqeth -p
```

<table>
<thead>
<tr>
<th>devices</th>
<th>CHPID</th>
<th>interface</th>
<th>cardtype</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.c000/0.0.c001/0.0.c002 xC0 hsi0</td>
<td></td>
<td></td>
<td>HiperSockets</td>
</tr>
<tr>
<td>0.0.b003/0.0.b004/0.0.b005 x01 eth0</td>
<td></td>
<td></td>
<td>GuestLAN QDIO</td>
</tr>
</tbody>
</table>

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** (for layer3 devices only)

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

IP Header:
  src. IP addr
  dest. IP addr

Fake Ethernet Hdr
  IP Header

fake_ll=1
```

```
Payload

OSA Express

QETH Driver

Network Stack

Application

Linux
```
QETH Device sysfs Attribute **large_send**

- Offload TCP segmentation from Linux network stack to OSA-card
  
  `QETH_OPTIONS='large_send=TSO'`
  
  or

  ```
  #> echo TSO > /sys/devices/qeth/0.0.b004/large_send
  ```

- Offload TCP segmentation from Linux network stack to device driver
  
  `QETH_OPTIONS='large_send=EDDP'`
  
  or

  ```
  #> echo EDDP > /sys/devices/qeth/0.0.b004/large_send
  ```

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

- additional redundancy check to protect data integrity
- 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
- Available for OSA-devices in layer3 mode only

---

QETH Device sysfs Attribute **recover**

- enforce recovery of a qeth device

```
#> echo 1 > /sys/devices/qeth/0.0.b004/recover
```
QETH Device sysfs Attribute `buffer_count`

- 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

Boost performance

Save memory
QETH Layer 2 mode

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

**OSA Express**

<table>
<thead>
<tr>
<th>MAC Addr</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

- HiperSocket: **new** layer2 support starting with z10
  - MAC address automatically generated

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

```
/sys
|-- devices
    |-- qeth
       |-- 0.0.<devno>
          |-- layer2
```

- 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!

- Advantages:
  - Independent of IP-protocol
  - DHCP, tcpdump working without option fake ll
  - channel bonding possible
  - No OSA-specific setup necessary for
    - Routing, IP Address Takeover, Proxy ARP
QETH Layer 2 mode (cont.)

- Direct attached OSA
  - Restrictions:
    - Older OSA-generation (<= z990):
      Layer2 and Layer3 traffic can be transmitted over the same OSA CHPID, but not between two hosts sharing the same CHPID!

- HiperSocket (new with z10)
  - Layer2 and Layer3 traffic separated

- 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>
  ```
Commands / tools for qeth-driven devices

- List of known qeth devices: `cat /proc/qeth` or `lsqeth -p`

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

<table>
<thead>
<tr>
<th>Devices</th>
<th>CHPID</th>
<th>Interface</th>
<th>Cardtype</th>
<th>Port</th>
<th>Chksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.a000/0.0.a001/0.0.a002 xA0</td>
<td></td>
<td>eth0</td>
<td>OSD_1000</td>
<td>0</td>
<td>sw</td>
</tr>
<tr>
<td>0.0.b000/0.0.b001/0.0.b002 xB0</td>
<td></td>
<td>hsi0</td>
<td>HiperSockets</td>
<td>0</td>
<td>sw</td>
</tr>
</tbody>
</table>

- Attributes of qeth device: `lsqeth` or `lsqeth <interface>`

```bash
#> lsqeth eth0
```

Device name : eth0

Card type : OSD_1000
Cdev0 : 0.0.f5f0
Cdev1 : 0.0.f5f1
Cdev2 : 0.0.f5f2
Chpid : 76
Online : 1
Checksumming : sw checksumming
State : UP (LAN ONLINE)
Buffer count : 16
Layer2 : 0
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
- applies to layer2-devices only
- latest setup description:
  
  http://sourceforge.net/projects/bonding/
Channel bonding setup

- 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
  ```

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

  ```
  #> modprobe bonding miimon=100 mode=balance-rr
  ```

- Bring up bonding device bond0

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

- 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  ...
```

```bash
# > 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, HiperSockets, z/VM Guest LAN, z/VM VSWITCH
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 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. Layer3 only: register 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. quagga: [http://quagga.net](http://quagga.net)).
## 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 lcs lcs</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>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 lcs lcs</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>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</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</th>
<th>GuestLAN</th>
<th>LCS</th>
<th>CTC</th>
<th>IUCV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adapters</td>
<td>100 Mbps, 1Gbps, 1000 Base-T, HSTR</td>
<td></td>
<td>IPv4, IPv6</td>
<td>IPv4, IPv6</td>
<td>100 Mbps, 1000 Base-T, HSTR</td>
<td>ESCON, FICON, Virtual CTC/A</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>point-to-point</td>
<td>point-to-point</td>
</tr>
<tr>
<td>Layer</td>
<td>Layer2 / 3</td>
<td>Layer2 / 3</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></td>
<td>restricted access (admin defines OSA Address Table)</td>
<td>Deprecated</td>
<td>Deprecated</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

- IBM System z Connectivity Handbook
  http://www.redbooks.ibm.com/redpieces/abstracts/sg245444.html