RTNET
for the SPB and LiRE

Patrick Hohmann
www.L3S.de/spb
MoRob – Modular Educational Robotic Toolbox

A project to develop a framework for teaching/research with Robotics

Project Partners:

- L3S Hannover / RTS  
  (Principal Investigator: Bernardo Wagner)
- KTH Stockholm, Centre for Autonomous Systems  
  (Principal Investigator: Henrik Christensen)
- Stanford University, Robotics Laboratory  
  (Principal Investigator: Oussama Khatib)

Funding: WGLN – Wallenberg Global Learning Network
What is MoRob about?

Universities need robot platforms which
• are flexible and modular, yet powerful
• can easily be customized to the requirements of different subjects and research goals

MoRob project aims:
• develop such a platform
• provide standard set of robot control modules + teaching units

This is aimed at teaching undergraduate and graduate students as well as for PhD research.
Scalable Processing Box

• Scalable Processing Box (SPB):

  standardized, scalable, modular, easily combinable

  processing unit for applications in educational robotics.

• SPB provides basic infrastructures to run software for research and education with real-time requirements

• Two main aspects of development:
  design of the hardware modules
  development of a software architecture (LiRE)
LiRE Setup I

- Base system can be booted from a CF-card
- Suitable Linux distribution
- Pre-compiled packages
- Easy to use installation scripts
- Installation in modular steps
  - Setup of basic Linux structures
  - Development environment
  - Web support
  - Real-time structures (RTAI),
  - Real-time communication layer
- API structures + device drivers can be implemented in add. packages
LiRE Setup II

Kernel
• Linux 2.4.21
• RTAI 24.1.12

Linux
• busybox common unix tools (like ls, rm, cp, vi ...)
• tinylogin common unix tools (like login, getty, passwd, su...)
• ssh client/server
• web server configuration with browser
• ...

Additional Packages
• Atmel USB wlan package
• RTnet
• ...
Pre-compiled RTnet package for LiRE

- /lire/package/rtnet location of the rtnet package in LiRE
- pck.init.sh script to start required drivers for rtnet
- pck.start.sh setup client or server
- pck.stop.sh stop rtnet and unload drivers
- pck.values.sh configuration values
- pck.cfg.sh startup behaviour

pck <list init start stop val cfg> rtnet

is a service script for packages in a LiRE system.
RTNet tools

rtifconfig
basic functions like ifconfig for normal networking (eg. assigning IP) and RTmac control

rtping
alive ping for RTnet structures

rtroute
static routing for RTnet structures (see README.routing)

rtcfg
main configuration tool to setup RTnet networking (see RTcfg.spec and README.rtcfg)

rtnet
shell script to handle start-up
RTcfg

Master

```
rtcfg rteth0 server  I am server
rtcfg rteth0 add <IP> -stage1 <stage1_file> -stage2 <stage2_file>
  Add a client to the network and distribute configuration or execution files for client handling (stage1 and stage2)

rtcfg rteth0 wait  wait for the client
rtcfg rteth0 ready  server is ready and waiting
```

Client

```
rtcfg rteth0 client -f <stage1_file>
  activate rtetht0 client and write stage1 config. file to <stage1_file>

rtcfg rteth0 announce -f <stage2_file>
  client is available (stage1 is done), save stage2 file to <stage2_file>

rtcfg rteth0 ready  client is ready and waiting
```
Example

Master
- insmod ...
- rtifconfig rteth0 up 10.0.0.1
- rtcfg rteth0 server
- rtcfg rteth0 add 10.0.0.123
- rtcfg rteth0 wait
- rtifconfig rteth0 mac master 5000
- rtifconfig rteth0 mac add 10.0.0.123 200
- rtifconfig rteth0 mac up
- ifconfig vnic0 up 10.0.0.1
- rtcfg rteth0 ready

Client
- Insmod ...
- rtifconfig rteth0 up 10.0.0.123
- rtcfg rteth0 client
- rtcfg rteth0 announce
- rtifconfig rteth0 mac client
- ifconfig vnic0 up 10.0.0.123
- rtcfg rteth0 ready
Routing

Master

echo "### route del vnic0"
route del -net `route | grep vnic0 | cut -c -15` netmask `route | grep vnic0 | cut -c 33-47` dev vnic0
for client in $PCK_RTNET_TDMA_CLIENTS ; do
    echo "### add client route for $client"
    route add $client dev vnic0
    arp -Ds $client eth0 pub
done
echo 1 >/proc/sys/net/ipv4/ip_forward

Client

echo "### route del vnic0"
route del -net `route | grep vnic0 | cut -c -15` netmask `route | grep vnic0 | cut -c 33-47` dev vnic0
echo "### route add $PCK_RTNET_ROUTING_GW"
route add $PCK_RTNET_ROUTING_GW dev vnic0
echo "### route add default gw"
route add default gw $PCK_RTNET_ROUTING_GW