



Kickoff-Workshop

Universität Hannover

ISE – Fachgebiet Echtzeitsysteme

Inhalt

- **Installation**
- **Basiskonfiguration**
- **Erste Tests und Diagnosen**
- **Programmentwicklung unter RTnet**
- **Fortgeschrittene Konfiguration**
- **Fragen / Wunschthemen**

Installation

- **Voraussetzung:**
installiertes und lauffähiges RTAI auf Vanilla-Kernel
 - Kernel 2.6.9 (ftp.kernel.org)
 - RTAI 3.1 (www.rtai.org)
- RTAI unbedingt vorher testen (testsuite...)!
- RTnet entpacken (hier: 0.8.1 + rtnet-script.patch)
- im RTnet-Verzeichnis aufrufen:

```
./configure --with-rtai=<RTAI-INSTALLATION>  
--prefix=<ZIEL-VERZEICHNIS>  
--enable-allpci  
[--enable-rtcap] (Analyse-Plug-In)
```
- `make && make install`
- Auf dem Zielsystem: `mknod /dev/rtnet c 10 240`
- Ggf. Pfad auf RTnet-Tools setzen (`<PREFIX>/sbin`)

Konfiguration / Start

- Einstellungen in `rtnet.conf` (`<PREFIX>/etc/rtnet.conf`)
 - `RT_DRIVER:` `rt_eepro100, rt_8139too, rt_via-rhine`
 - `IPADDR:` muss im RT-Netz eindeutig sein
 - `TDMA_MODE:` „master“ oder „slave“
 - `TDMA_SLAVES:` IP-Liste der Slaves (nur für Master notwendig)
- RTAI-Module laden (3.x)
 - `rtai_hal`
 - `rtai_lxrt`
 - `rtai_sem`
 - `rtai_rtdm` (gehört noch zu RTnet: `<PREFIX>/modules/rtai_rtdm`)
- Netzwerkadapter ggf. freigeben, d.h. Standard-Treiber entladen
- `<PREFIX>/sbin/rtnet start`

Erste Tests

- **Echtzeit**
 - `rtping <ZIEL-IP>`
 - `rtping 127.0.0.1`
 - **Beispiele im Quellordner (werden nicht installiert), etwa `addons/examples/round_trip_time`**
- **TCP/IP-Tunnelling**
 - `ping <ZIEL-IP>`
 - `ssh <USER>@<ZIEL-IP>`
- **Beobachter mit Ethereal (ab 0.10.8) anschließen**

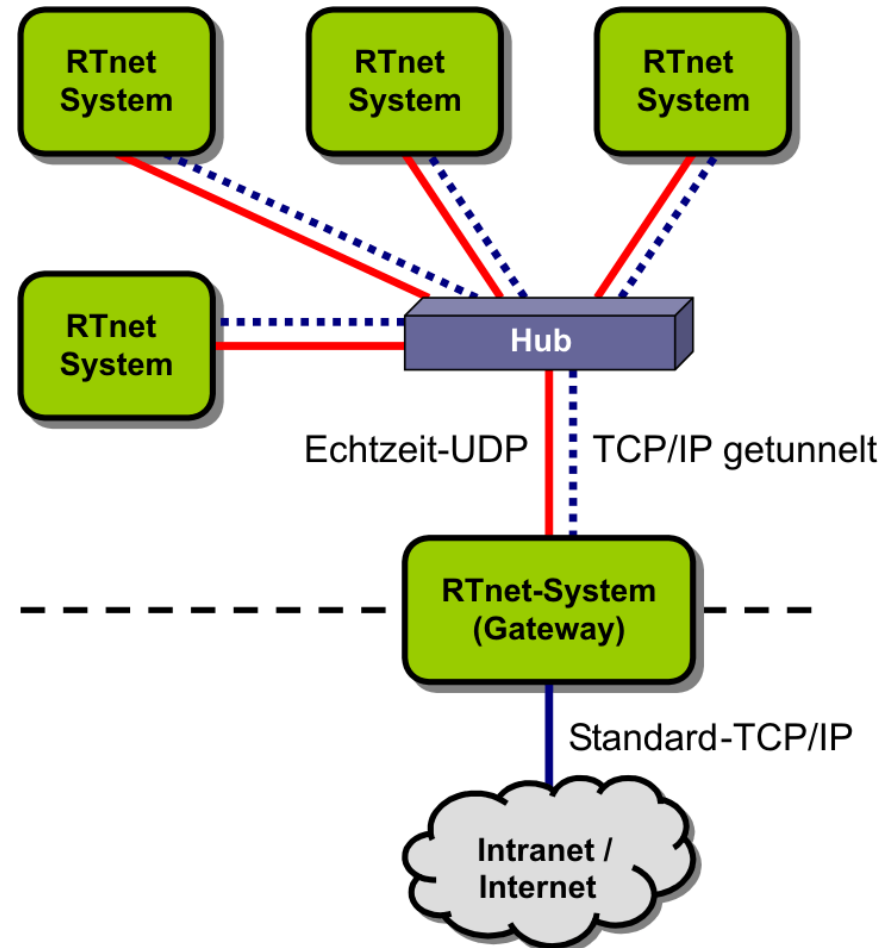
Troubleshooting

Wenn es hakt:

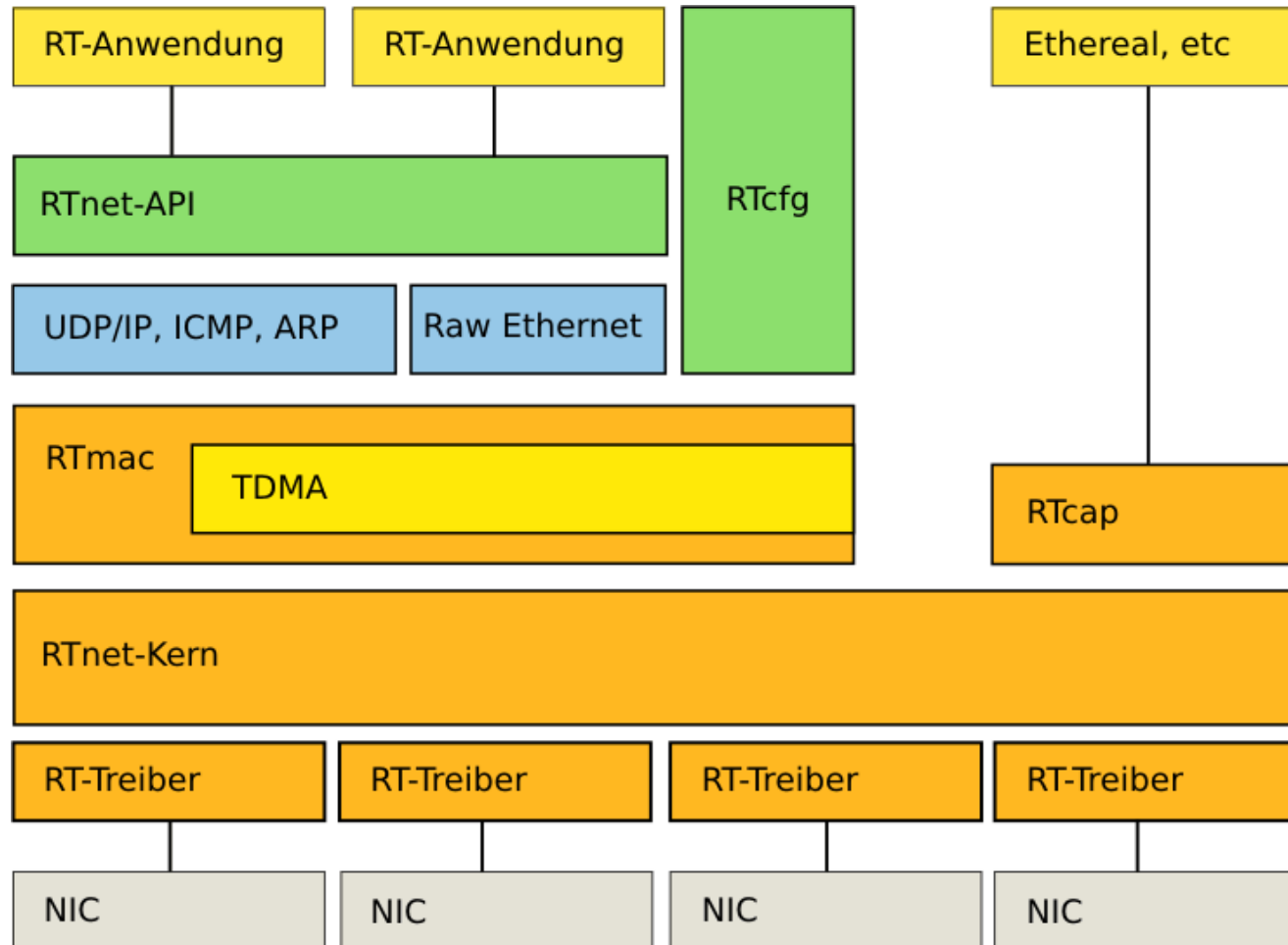
- **Wurden alle Module geladen (lsmod)?**
 - Passen Module wirklich zum Kernel?
 - Wurde die Netzwerkkarte erkannt?
 - Standard-Modulliste RTAI-3.x:
rtai_hal, rtai_lxrt, rtai_sem
 - Standard-Modulliste RTnet-0.8.x:
rtai_rtdm, rtnet, rtcfg, rtmac, tdma, rt_loopback, rt_<Treiber>
- **Existiert /dev/rtnet?**
- **Funktioniert nur Loopback? => Netzwerk-Problem**
- **/proc/rtnet/* untersuchen**
 - RTcfg: /proc/rtnet/rtcfg/<dev>/*
 - RTmac: /proc/rtnet/rtmac/*
- **Beobachtung mit Ethereal&Friends**

Struktur eines RTnet-Netzwerks

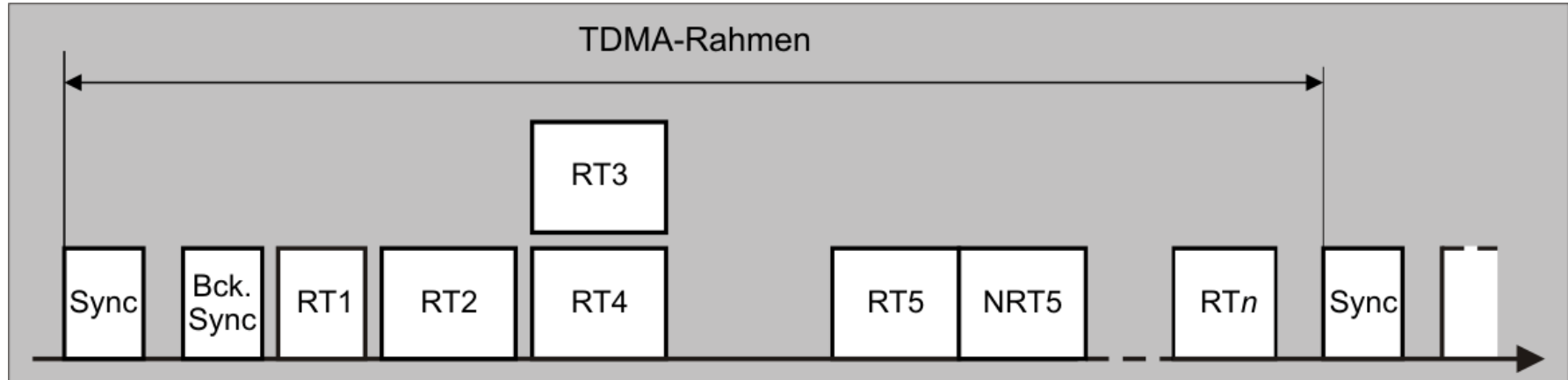
- **Dediziertes Netzwerk**
- **Keine Einschränkung der Netzwerkstruktur**
- **Passive Verteiler (Hubs)**
- **Switches verwendbar** (höhere Latenz, keine Vorteile)
- **Protokolltunnel zur Weiterleitung von TCP/IP**
- **Gateway stellt Verbindung zum Intra-/Internet her**



Stack-Aufbau



TDMA Revision 2



- **Master sendet periodische Sync-Frames**
- **Bei Ausfall übernimmt Backup-Master verzögerungsfrei**
- **Uhrensynchronisation auf den Master**
- **Dezentrale Konfiguration**
- **Hot-Plugging: Stationsaustausch im laufenden Betrieb**

RTnet-API

- **POSIX Socket-Funktionen mit „_rt“-Suffix**
 - `socket_rt, close_rt`
 - `sendmsg_rt, recvmsg_rt, ...`
 - `bind_rt, connect_rt`
 - `get/setsockopt_rt, ioctl_rt, ...`
- **Kernel- und Userspace (LXRT, Fusion in Arbeit)**
- **Gültiger Kontext der Funktionen**
 - **Erzeugen und Konfigurieren von Sockets:**
Non-RT und RT (mit Einschränkungen)
 - **Senden/Empfangen:**
Nur RT

Beispiel

```
void *bouncer(void *arg)
{
    [Variablendeklaration]
    /* RT-UDP-Socket oeffnen */
    recv_sock = socket_rt(AF_INET, SOCK_DGRAM, 0);

    /* Socket an den lokal Empfangsport binden */
    local.sin_family = AF_INET;
    local.sin_port    = htons(RECEPTION_PORT);
    local.sin_addr.s_addr = INADDR_ANY;
    bind_rt(recv_sock, (struct sockaddr *)&local, sizeof(local));

    /* Empfangs-Timeout setzen */
    timeout = 100000000;
    ioctl_rt(recv_sock, RTNET_RTIOC_TIMEOUT, &timeout);

    /* Echtzeit-Task anlegen */
    bnc_task = rt_task_init_schmod(nam2num("BOUNCER"), 11, 0, 0,
                                   SCHED_FIFO, 0xF);

    rt_make_hard_real_time();
    ...
}
```

Beispiel (Fortsetzung)

```
...
/* Empfaengerschleife:
 * Liest ankommende Pakete und schickt sie zurueck. */
while (!terminate) {
    if (recvfrom_rt(recv_sock, &buffer, sizeof(buffer),
                   0, (struct sockaddr *)&remote,
                   &remote_sz) == sizeof(buffer)) {
        sendto_rt(recv_sock, &buffer, sizeof(buffer), 0,
                  (struct sockaddr *)&remote, sizeof(remote));
    }
}

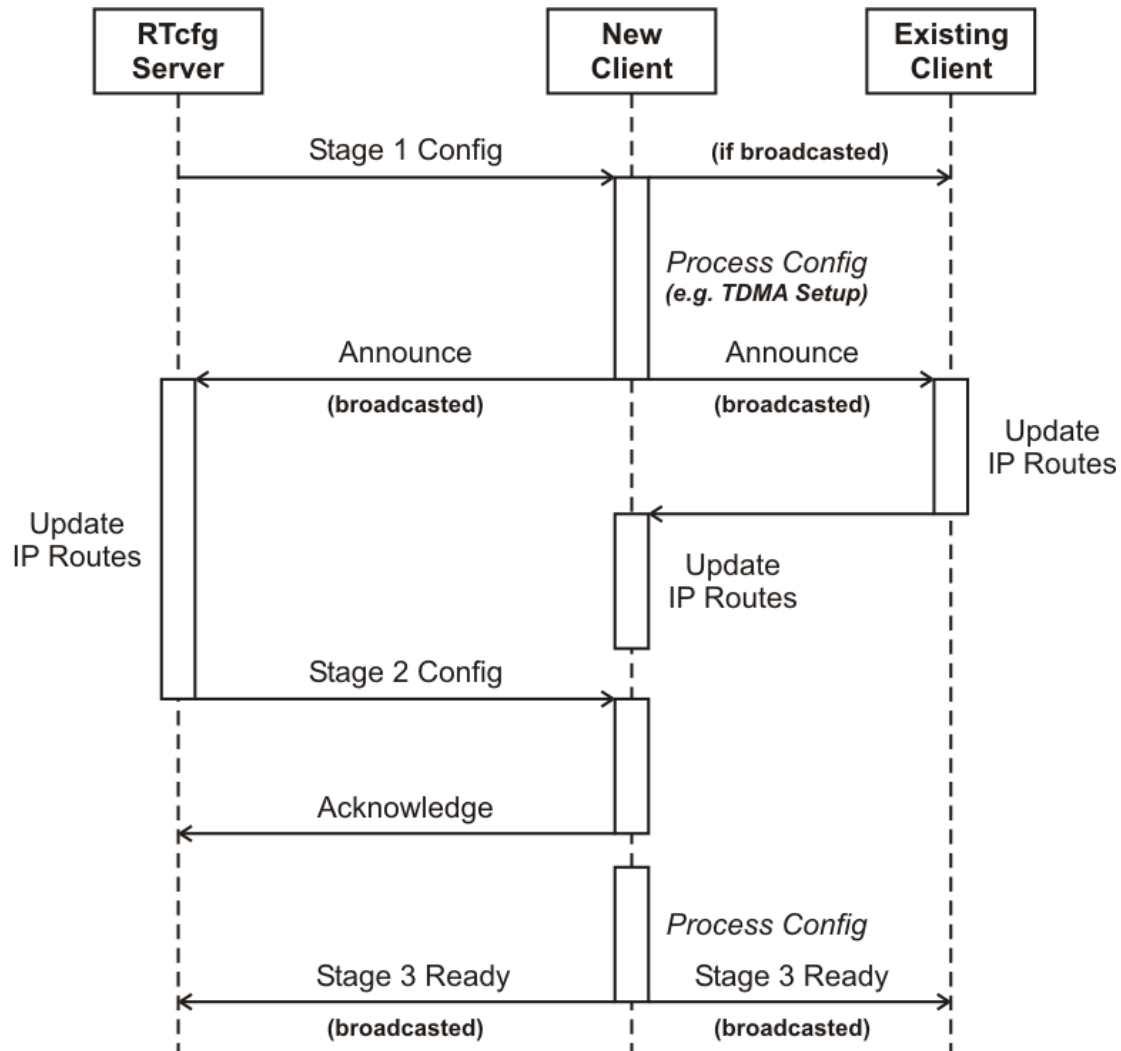
/* Aufraeumen */
rt_make_soft_real_time();
rt_task_delete(bnc_task);
close_rt(recv_sock);

return 0;
}
```

Konfigurationsprotokoll RTcfg

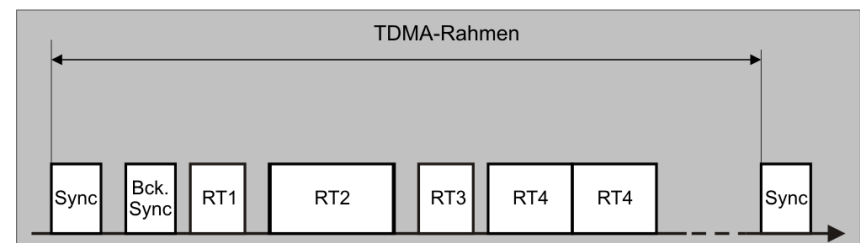
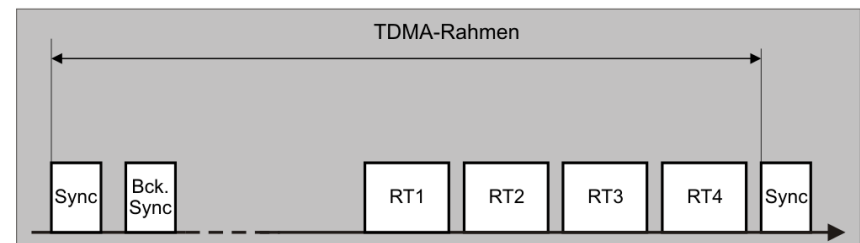
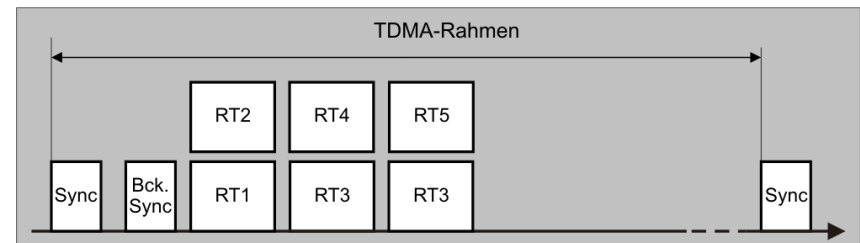
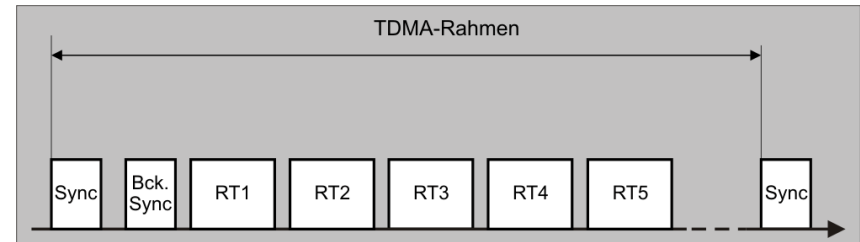
- **Generisches Protokoll**
 - Unabhängig vom Transportprotokoll (IP, ...)
 - Unabhängig vom Transportmedium (Ethernet, ...)
- **Ermöglicht zentrale Konfiguration von RTnet-Stationen**
- **Ablauf zum Hot-Plugging einer Station:**
 1. **Station ist per IP- oder MAC-Adresse bekannt**
 2. **Erhält grundlegende Konfiguration zugesandt**
(z.B. TDMA-Slots)
 3. **Übernimmt Konfiguration und kann aktiv am Netz teilnehmen**
- **Verteilung zusätzlicher, anwendungsspezifischer Konfigurationsdaten möglich**

RTcfg: Ablauf beim Start

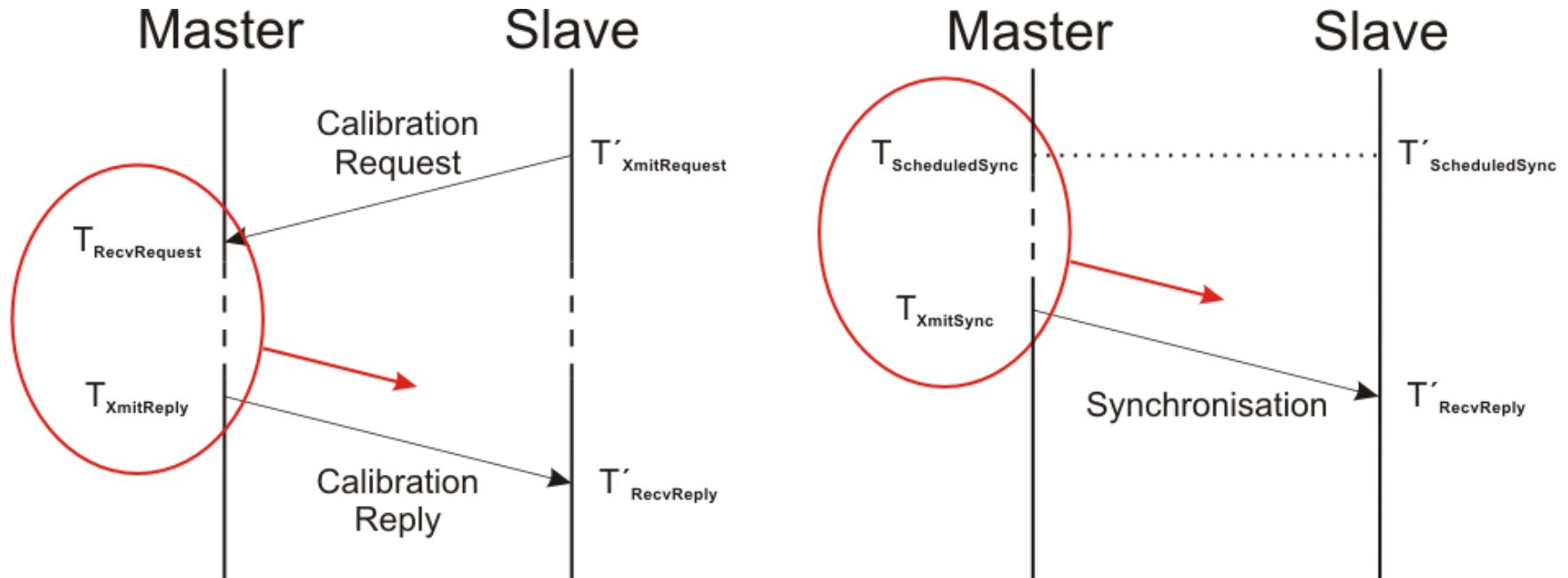


TDMA 2 – Flexible Konfiguration

- Zyklusdauer frei einstellbar
- Freie Sende-Slot-Konfiguration
 - Stationszuordnung
 - Offset zum Sync-Frame
 - Periodizität und Phase
=> alternierende Slot-Nutzung
- Explizite Slot-Zuweisung für Anwendungen möglich
- Nachrichtenpriorisierung innerhalb des Slots
(31 Stufen + 1 für TCP/IP)
- ▶ Individuelle Anpassung an Leistungsfähigkeit und Bandbreitenbedarf pro Station



TDMA 2 – Uhrensynchronisation



- **Kalibrierungsphase ermittelt Übertragungslatenz von Hard- und Software**
- **Genauigkeit abhängig von Interruptlatenz**

- **Zyklische Synchronisation überträgt auch Globalzeit**
- **Kompensation von Schwankungen des Sendezeitpunkts**

TDMA-Konfiguration

```
# tdma.conf
master:
[ip 1.2.3.4]
cycle <cycle_in_us>
slot <id> <offset_in_us> [<phasing>/<period> [<size>]]
[slot ...]

slave:
ip 1.2.3.4
[mac AA:BB:CC:DD:EE:FF]
[stage2 <file>]
slot ...

slave:
mac AA:BB:CC:DD:EE:FF
...

backup-master:
ip 1.2.3.4          (oder IP+MAC oder nur MAC)
backup-slot <offset_in_us>
[stage2 <file>]
slot ...
```

Wichtige RTnet-Tools

- `rtifconfig`
Konfiguration eines Netzwerk-Interfaces
- `rtroute`
Setzen und Abfragen statischer Routen (auch arp-Ersatz)
- `rtcfg`
Konfiguration von RTcfg
- `tdmacfg`
Konfiguration der TDMA-Medienzugriffskontrolle
- `rtnet`
Skript zum Starten und Stoppen des Netzwerks
- `rtping`
Netzwerktest

Fragen / Wünsche





www.rts.uni-hannover.de/rtnet

kiszka@rts.uni-hannover.de