NetRPC

Walter Fetter Lages

w.fetter@ieee.org

Universidade Federal do Rio Grande do Sul

Escola de Engenharia

Departamento de Engenharia Elétrica

ENG04008 Sistemas de Tempo Real

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Introdução

- RPC = passagem de mensagens síncronas entre tarefas
- NetRPC = tarefas distribuídas ou não
- Através do NetRPC o RTAI suporta sistemas distribuídos
- Depende de uma camada de suporte a mensagens
  - Por *default* utiliza os serviços de rede do Linux
  - RTnet

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

API

- Extensão das funções de RPC, com as seguintes modificações:
  - A função tem o mesmo nome das funções de RPC, substituindo o `rt_` por `RT_`
  - Existem mais dois argumentos iniciais, o nodo e a porta na máquina remota que executarão as funções que tiveram o `rt_` substituído por `RT_`
- Exemplo

```
rt_mbx_send(mbx,msg,msglen);
```

torna-se

```
RT_mbx_send(node,port,mbx,msg,msglen);
```

API

- Argumentos de tempo devem ser sempre em nanosegundos
  - Não é necessário conhecer a freqüência do *timer* do nodo remoto
  - É independente de qual nodo remoto será utilizado
- O tamanho máximo da mensagem é `MAX_MSG_SIZE` = 1500 bytes

Nodo Local

- O endereço local pode ser definido:
  - Na carga do módulo através do parâmetro `ThisNode="xxx.xxx.xxx.xxx"`
  - Através de uma chamada à

```
rt_set_this_node(const char *ddn,
        unsigned long node,int hard);
```

- Se o RTnet estiver habilitado devem ser definidos nodos para chamadas *hard* e para chamadas *soft*
  - `rt_set_this_node()` chamadas duas vezes
  - `ThisHardNode="xxx.xxx.xxx.xxx"` e `ThisSoftNode="xxx.xxx.xxx.xxx"`, na carga do módulo

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Nodos

- `node=0` força a execução da função `rt_` local
  - Facilita chavear entre aplicações locais e distribuídas
  - `port` não é utilizado, mas deve ter um valor válido
  - As únicas penalidades são dois argumentos e um `if` a mais
- Também pode-se fazer o nodo ser `localhost`, mas a penalidade é maior

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Nodos

- `node` deve ser o endereço IP do nodo remoto, em binário
  - No espaço do *kernel* pode-se converter de DDN para binário com

    ```
    unsigned long ddn2nl(const char *ddn);
    ```

  - No espaço do usuário podem ser utilizadas as funções padrão da `libc`

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Portas

- Utilizando `-port` ao invés de `port` resulta uma chamada assíncrona
- O resultado é enviado pelo nodo remoto, mas a chamada de função não espera o retorno
- No entanto, existe um mecanismo para recuperar o resultado de uma chamada assíncrona
- Uma chamada assíncrona subseqüente não será enviada antes que a anterior tenha sido respondida
  - A chamada retornará imediatamente, sem ser enviada

Portas

- Verificação da resposta de uma chamada assíncrona

```
void *rt_waiting_return(unsigned long node,int port
```

- Antes chamar um serviço remoto, a tarefa deve solicitar uma `port` no nodo remoto

```
myport=rt_request_soft_port(node);
myport=rt_request_hard_port(node);
rt_release(node, myport);
```

- Cada tarefa pode solicitar apenas uma `port`, múltiplas solicitações retornarão o mesmo valor

Portas

- Tarefas podem criar e acessar mais portas através de:

```
anotherport=rt_request_soft_port_id(node,id);
anotherport=rt_request_hard_port_id(node,id);
rt_release(node,anotherport);
```

- `id` é um `unsigned long` "combinado" entre as aplicações
- Ao solicitar uma porta, também pode-se fornecer um *mailbox* para recuperar os resultados de chamadas assíncronas

```
myport=rt_request_soft_port_mbx(node,mbx);
myport=rt_request_hard_port_mbx(node,mbx);
myport=rt_request_soft_port_id_mbx(node,id,mbx);
myport=rt_request_hard_port_id_mbx(node,id,mbx);
```

Portas

- Quando uma nova rpc é feita o resultado pendente de uma chamada assíncrona anterior é enviado para o *mailbox*
- De forma mais direta, pode-se forçar a sincronização com:

```
int rt_sync_net_rpc(unsigned long node,int port);
```

- Solicitação e liberação de portas **não** são executadas em tempo real

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

Time out

- Por *default* as chamadas não possuem *time out*
- Pode ser configurado um *time out*

```
rt_set_netrpc_timeout(port,timeout)
rt_set_netrpc_timeout(port, 0);
```

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Recepção

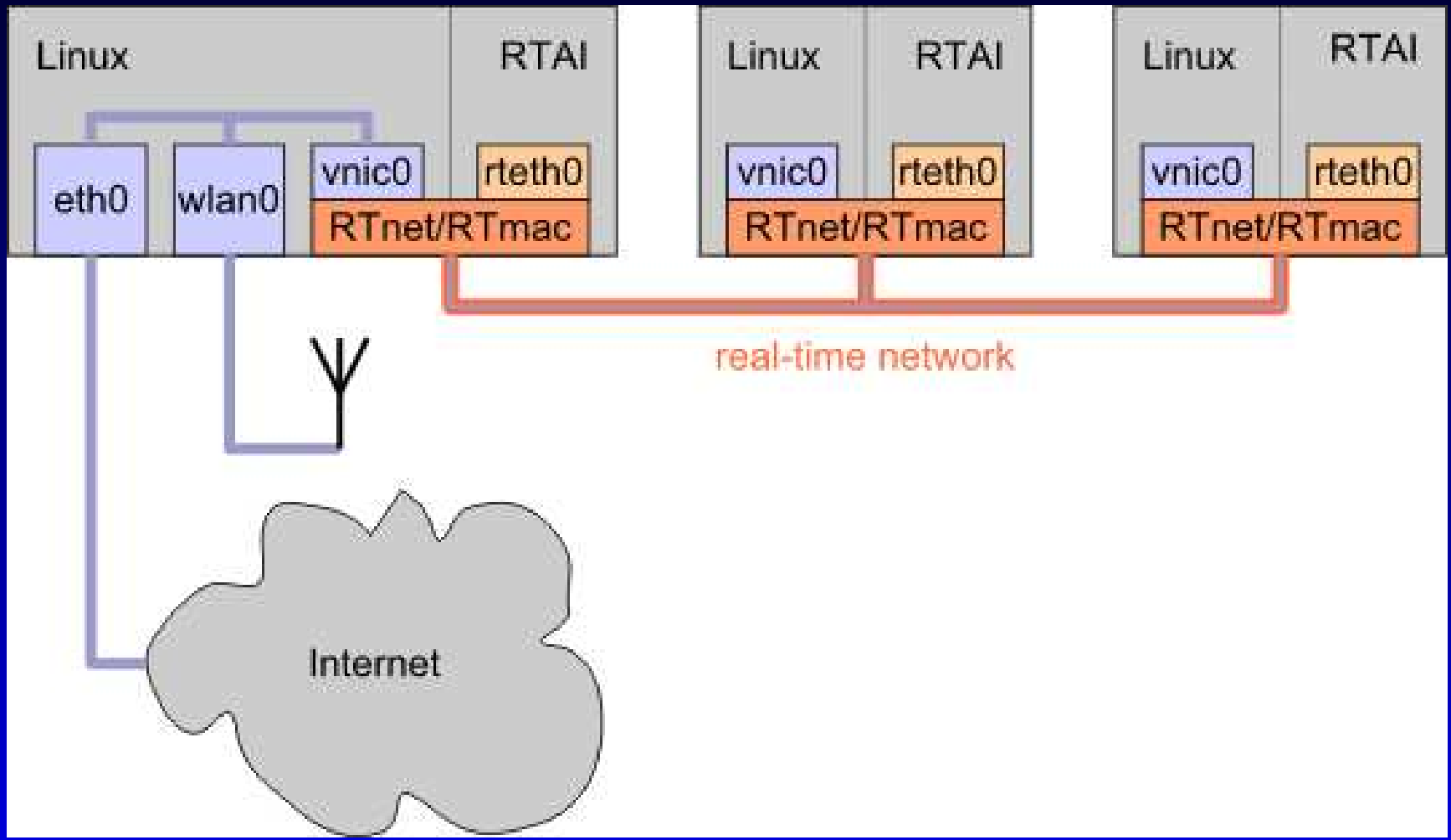
- Para receber de qualquer tarefa utiliza-se `rt_receive()`
  - Recebe também de tarefas remotas
- Para receber de uma tarefa específica utiliza-se `RT_receive()`
- Deve-se utilizar `rt_return()` para receber com `rt_receive()` e `RT_return()` para receber com `RT_return()`

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

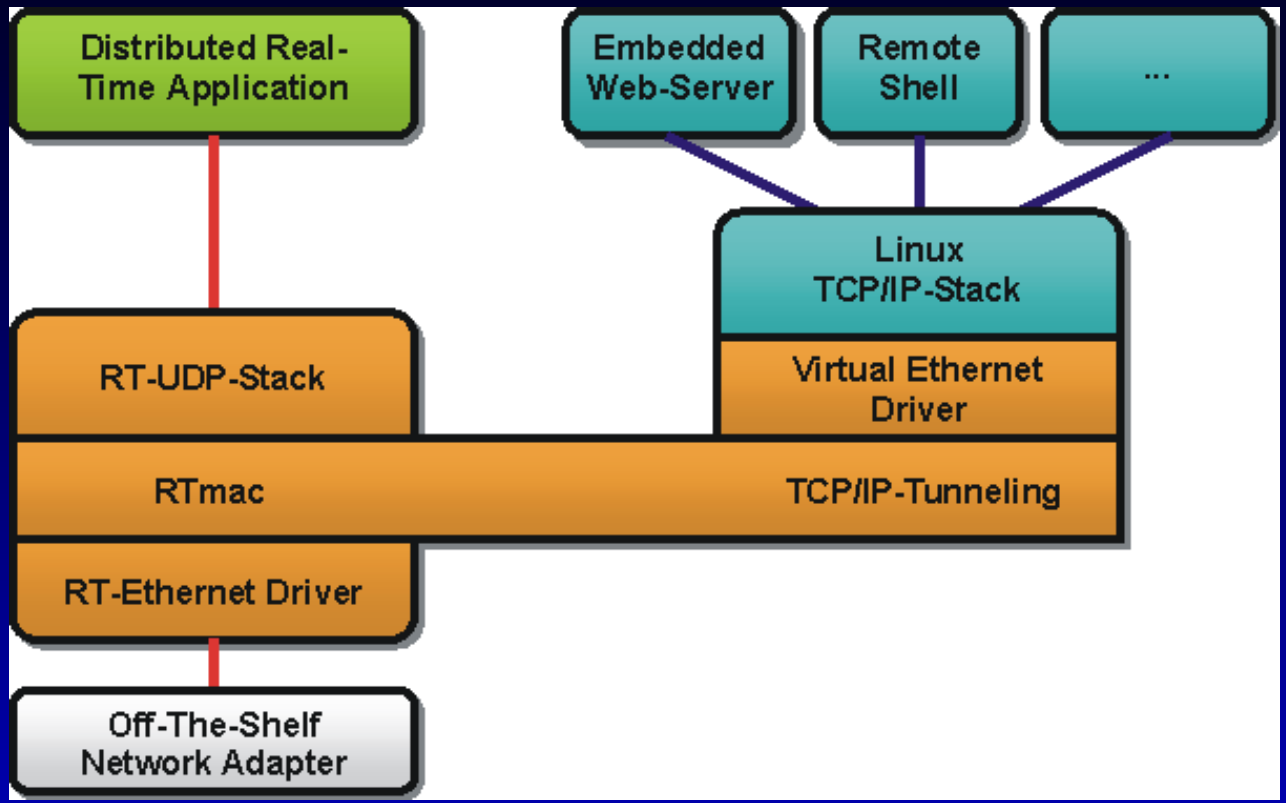
RTnet

- Normalmente o Netrpc utiliza os serviços de rede do Linux
  - Não tempo real
- RTnet implementa UDP/IP, ICMP e ARP de forma determinística
  - É utilizado o RTmac para controlar o acesso ao meio
- Se configurado, tarefas *hard real time* utilizam RTnet automaticamente
  - Deve ser utilizado o nodo *soft*, a conversão é feita internamente
- Tarefas *soft real time* utilizam os serviços de rede do Linux

RTnet

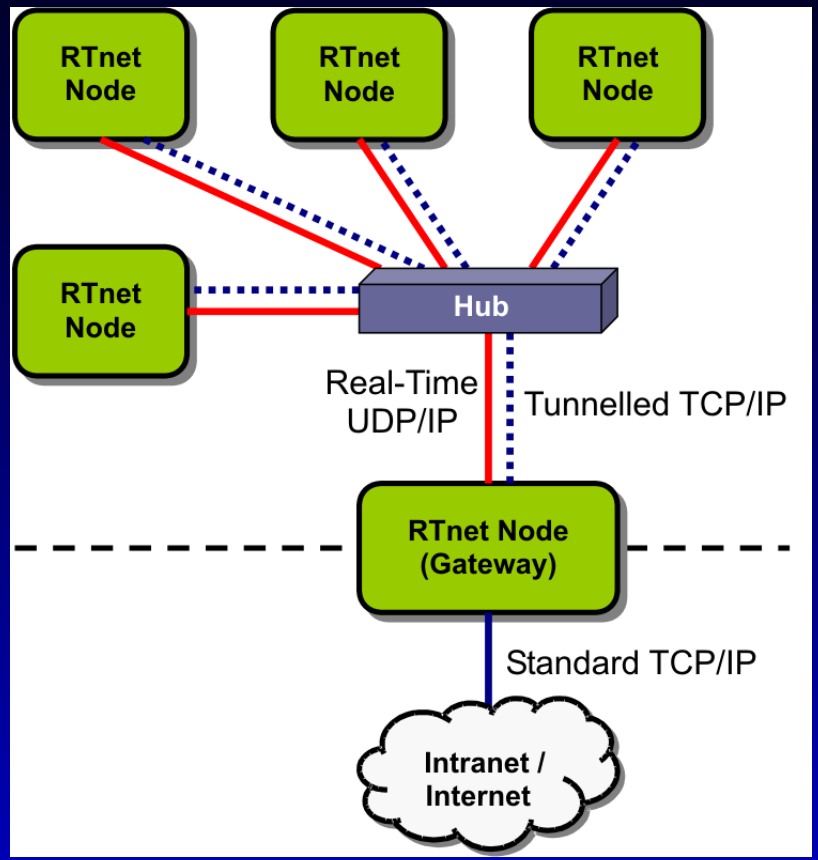

Linux
RTAI
eth0
wlan0
vnic0
rteth0
RTnet/RTmac
Linux
RTAI
vnic0
rteth0
RTnet/RTmac
Linux
RTAI
vnic0
rteth0
RTnet/RTmac
real-time network
Internet

RTnet



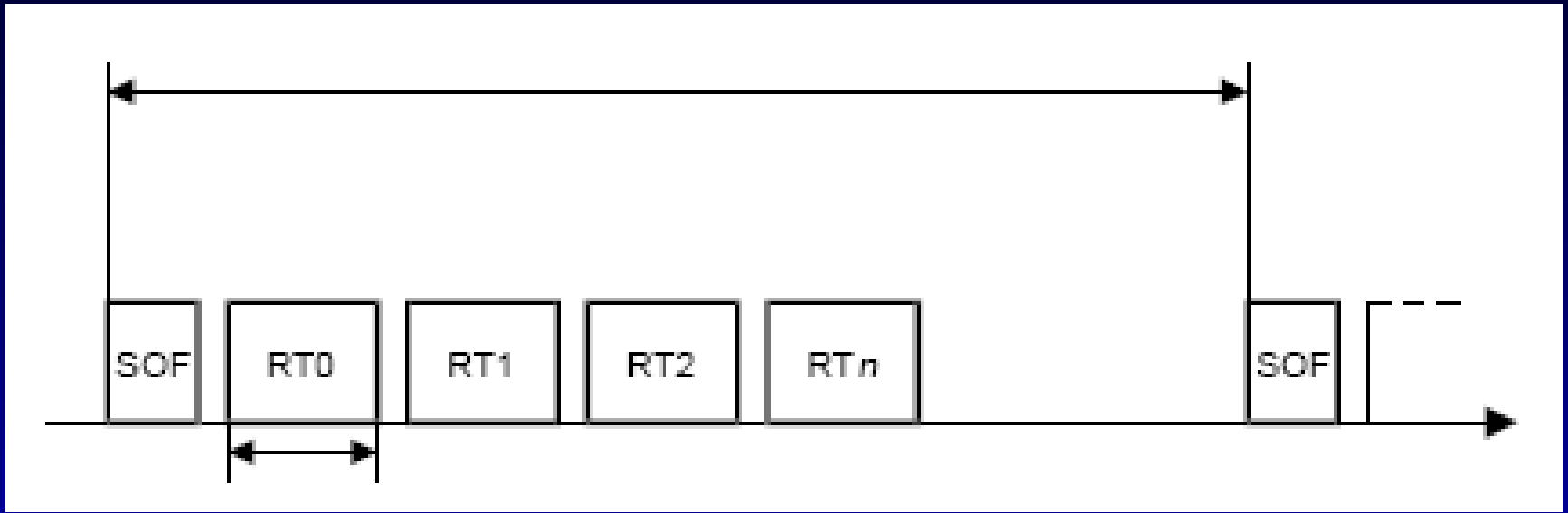
- Driver Ethernet semelhante ao do Linux
- Controle de acesso ao meio opcional (RTmac)
- Tráfego não tempo real "tunelado"através de VNIC

RTmac



- Requer rede dedicada
- Controla o acesso ao meio
- Time Division Medium Access (TDMA)

TDMA



- Mestre transmite pacote de sincronismo periódico (SOF)
- Cada cliente transmite em um slot dedicado relativo ao SOF

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

NICs Suportadas

- Intel 8255x EtherExpress Pro100
- DEC 21x4x Tulip
- RealTek RTL8139
- AMD PCnet32/PCnetPCI
- VIA Rhine
- NatSemi DP8381x
- MPC8xx
- MPC8260
- SMSC LAN91C111