

QNX

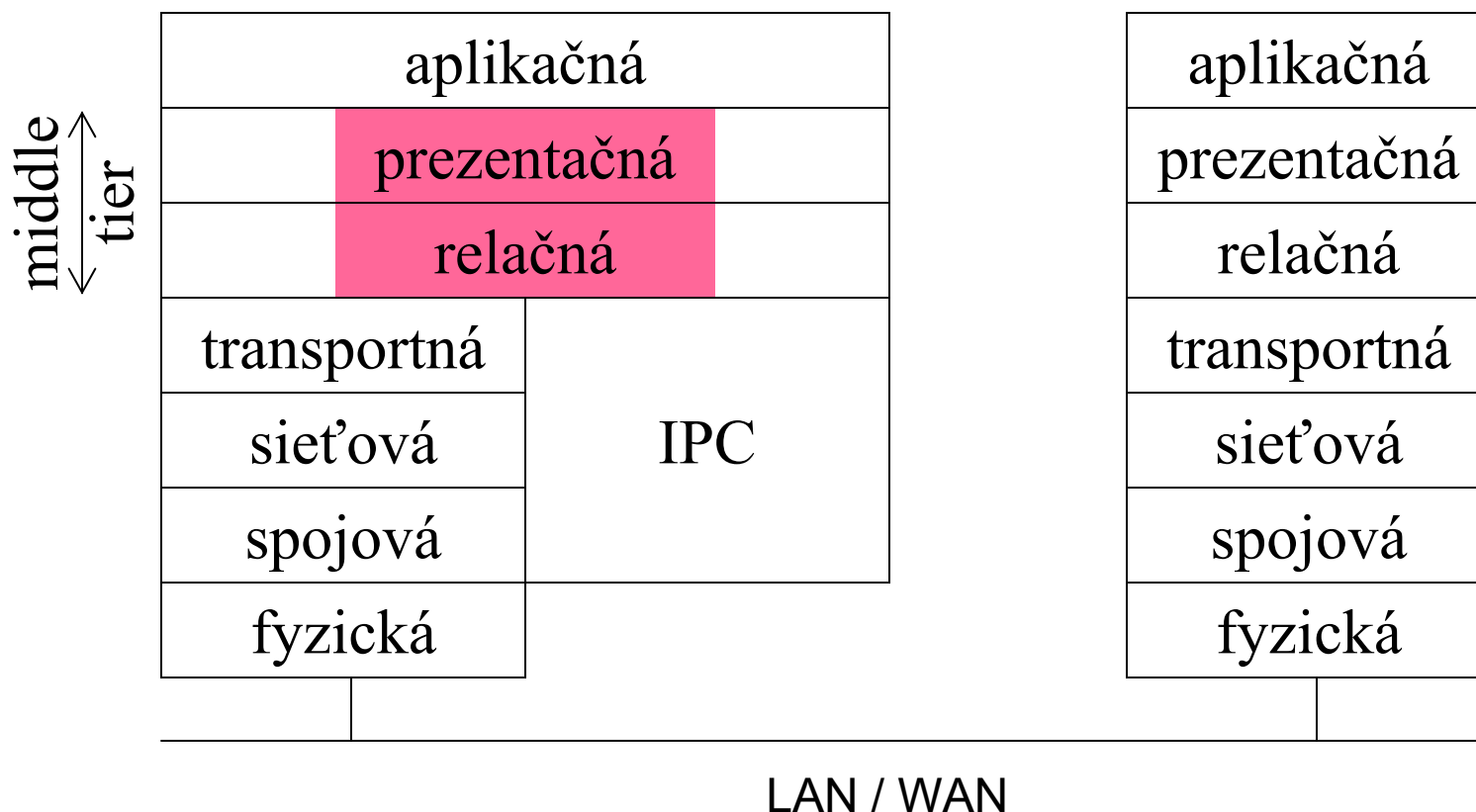
RNDr. Andrej Lúčný

MicroStep-MIS

andy@microstep-mis.com

<http://www.microstep-mis.sk/~andy>

Implementácia MAS

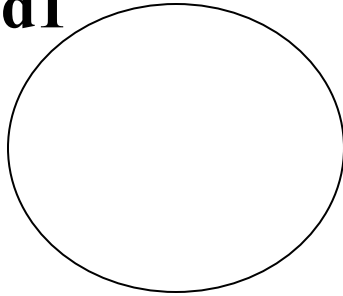


QNX4

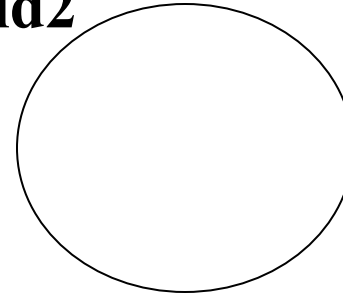
- RTOS pre PC
- Reálny čas = garantovaná odozva (latency=2 μ s)
- IPC založené na message passing-u
- Blokujúci a neblokujúci
- Rovnaký komunikačný model pre IPC a LAN

Procesy: pid

pid1



pid2

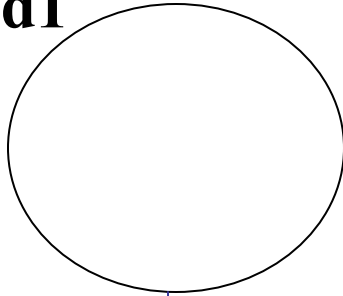


OS

jeden Proces môže komunikovať s
druhým pokiaľ vie jeho pid

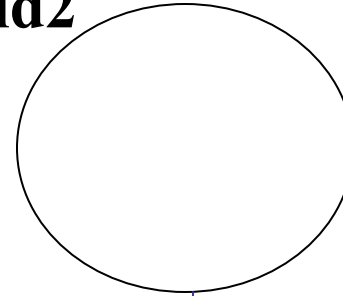
Procesy: names

pid1



Každý vie
svoj vlastný
pid (a pid
otca)

pid2



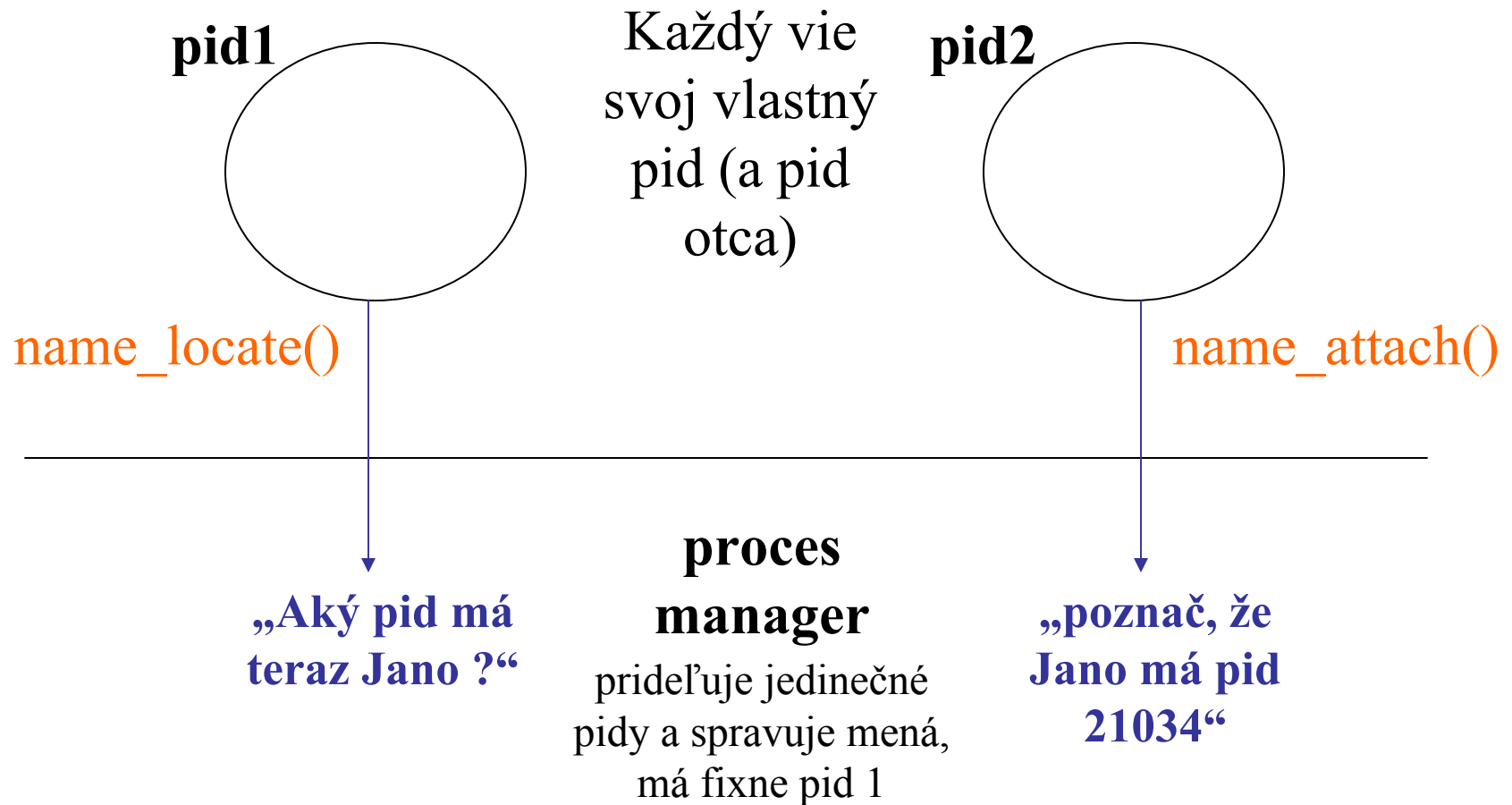
„Aký pid má
teraz Jano ?“

**proces
manager**

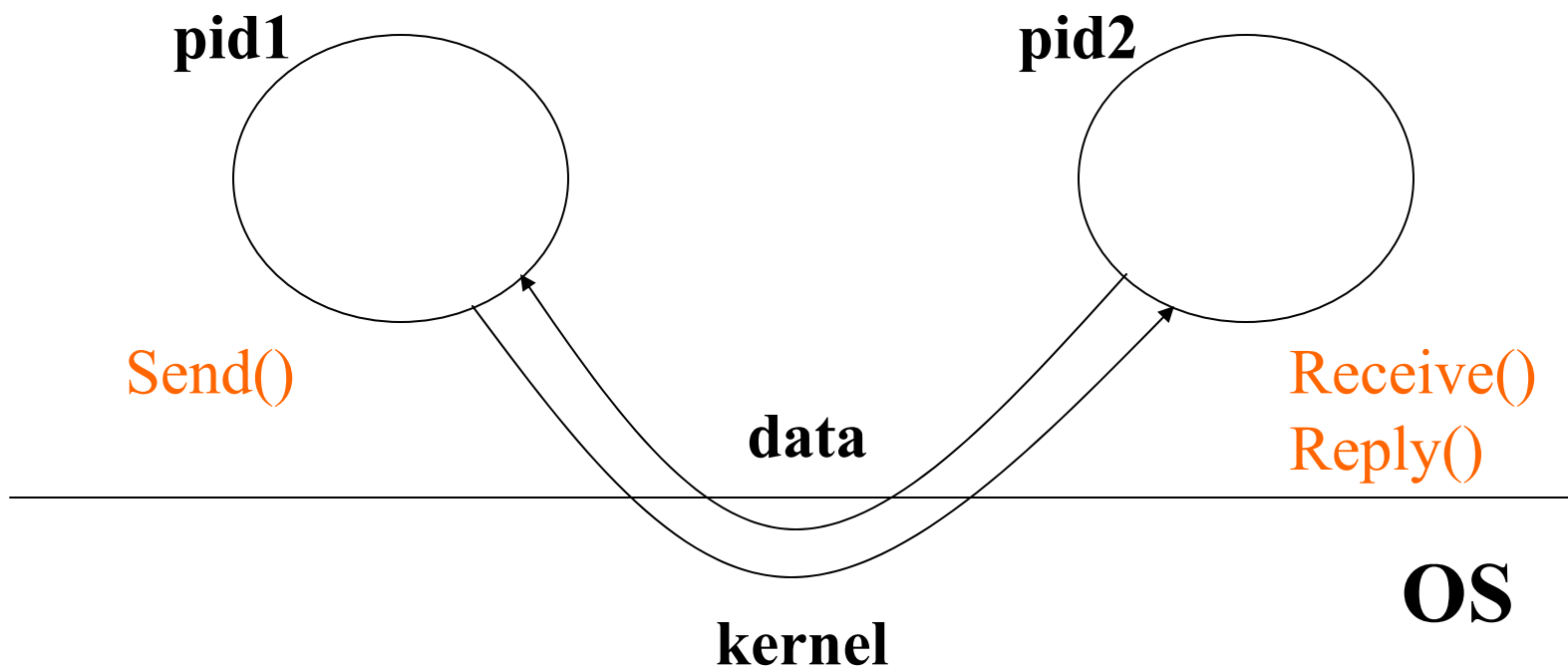
prideluje jedinečné
pidy a spravuje mená,
má fixne pid 1

„poznač, že
Jano má pid
21034“

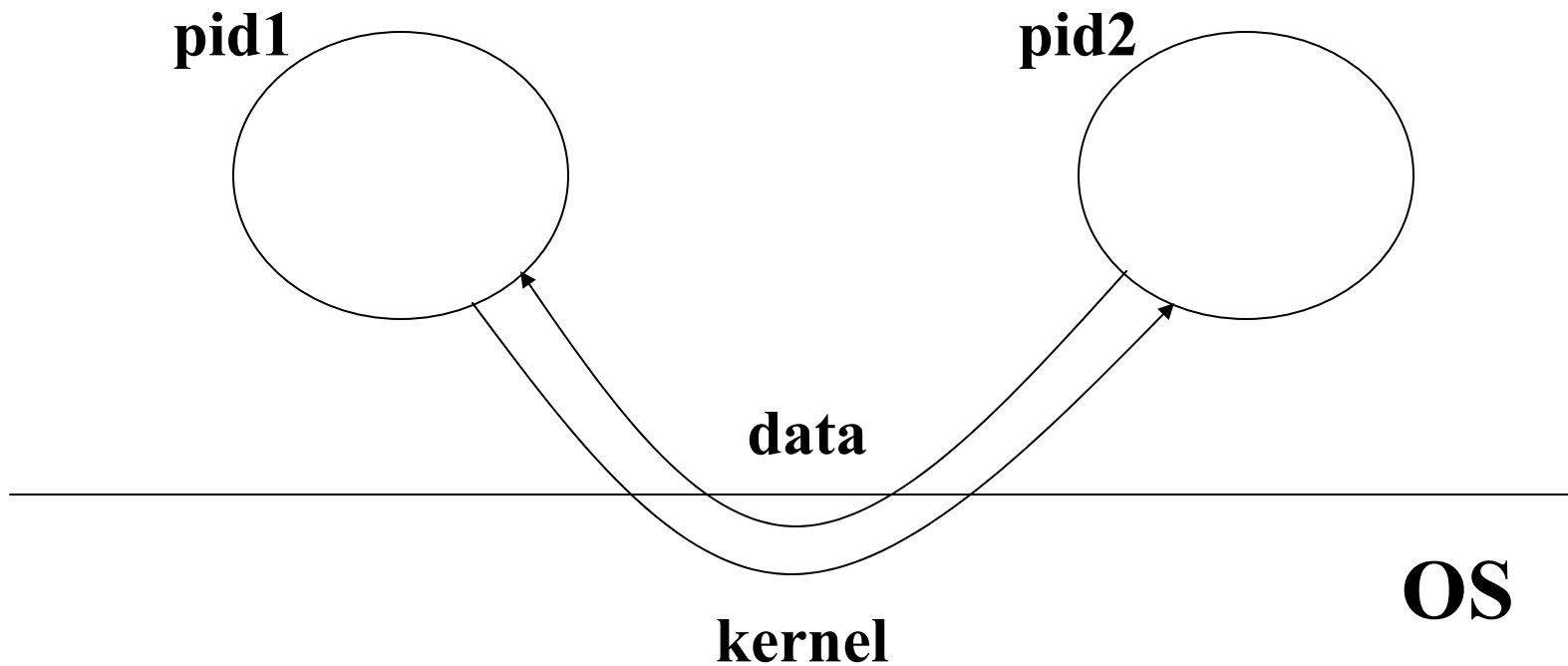
Procesy: names



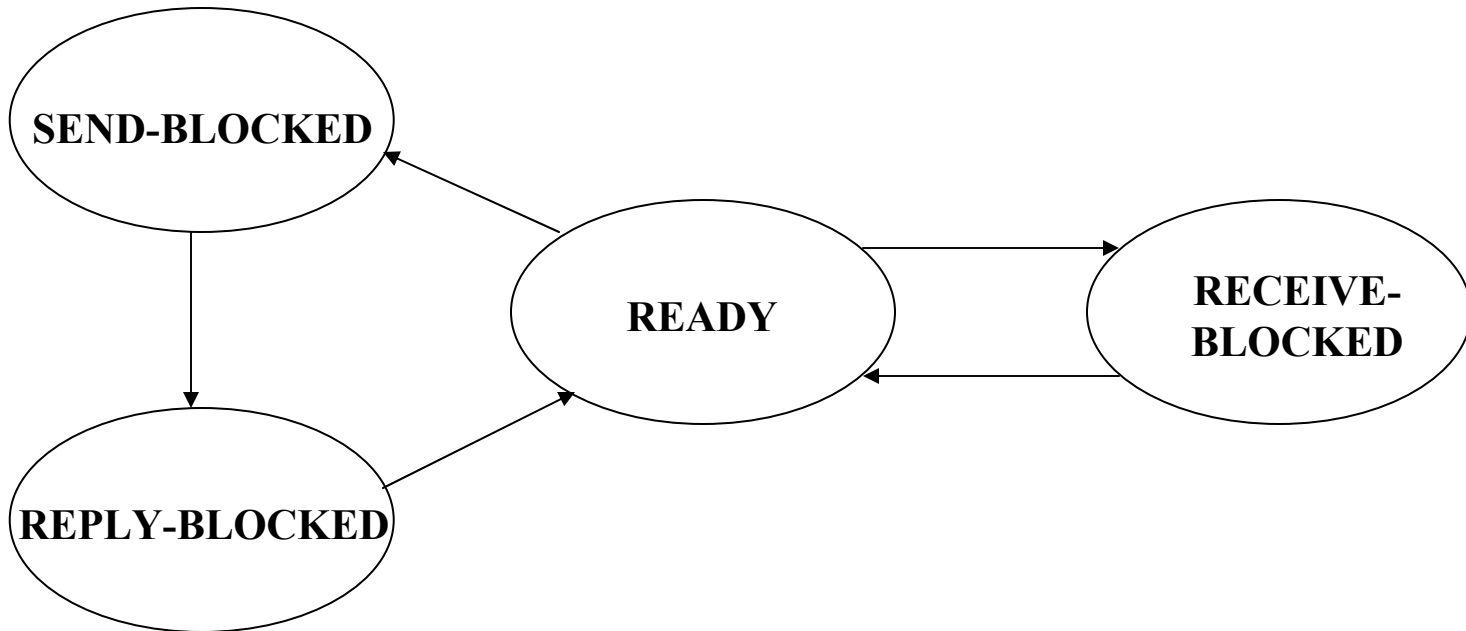
Procesy: komunikácia

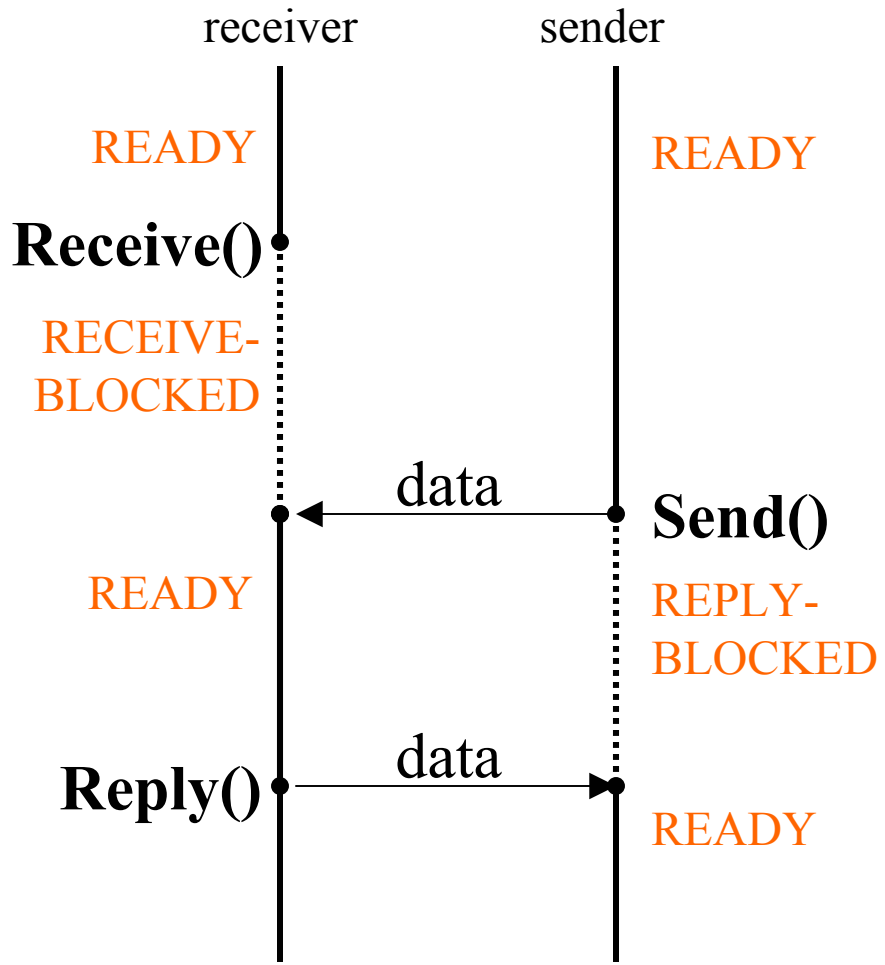


Procesy: komunikácia

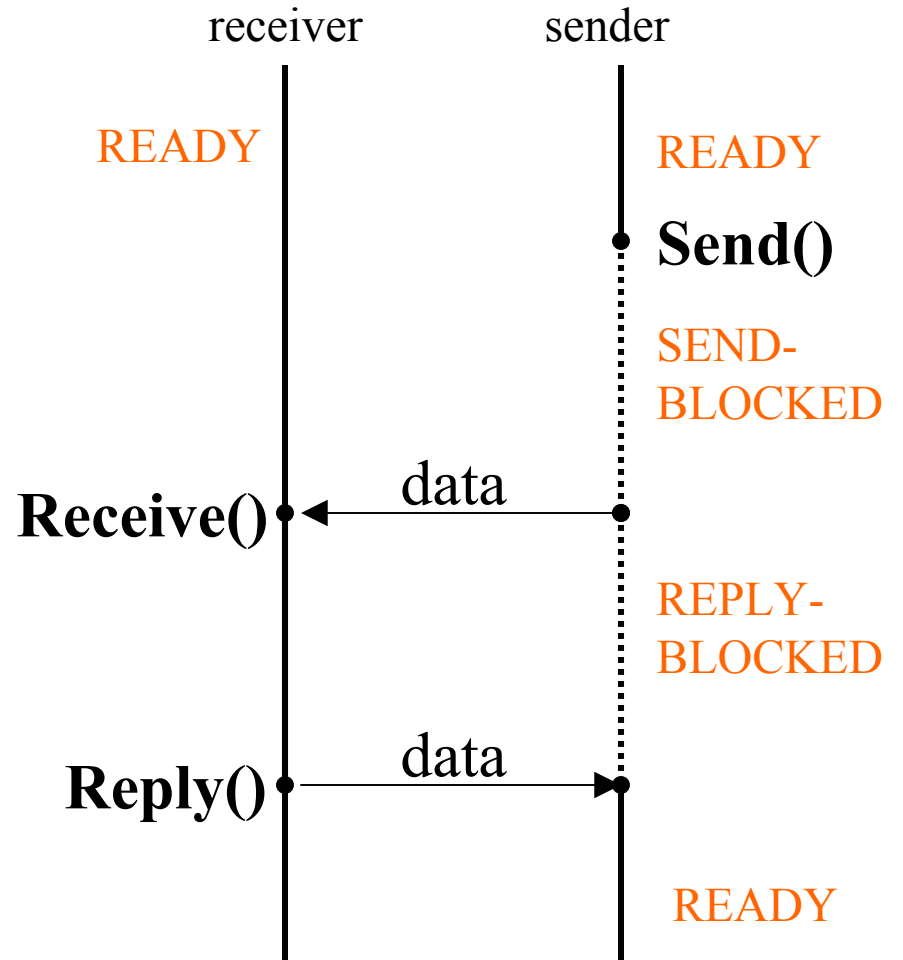


Processy: stavy

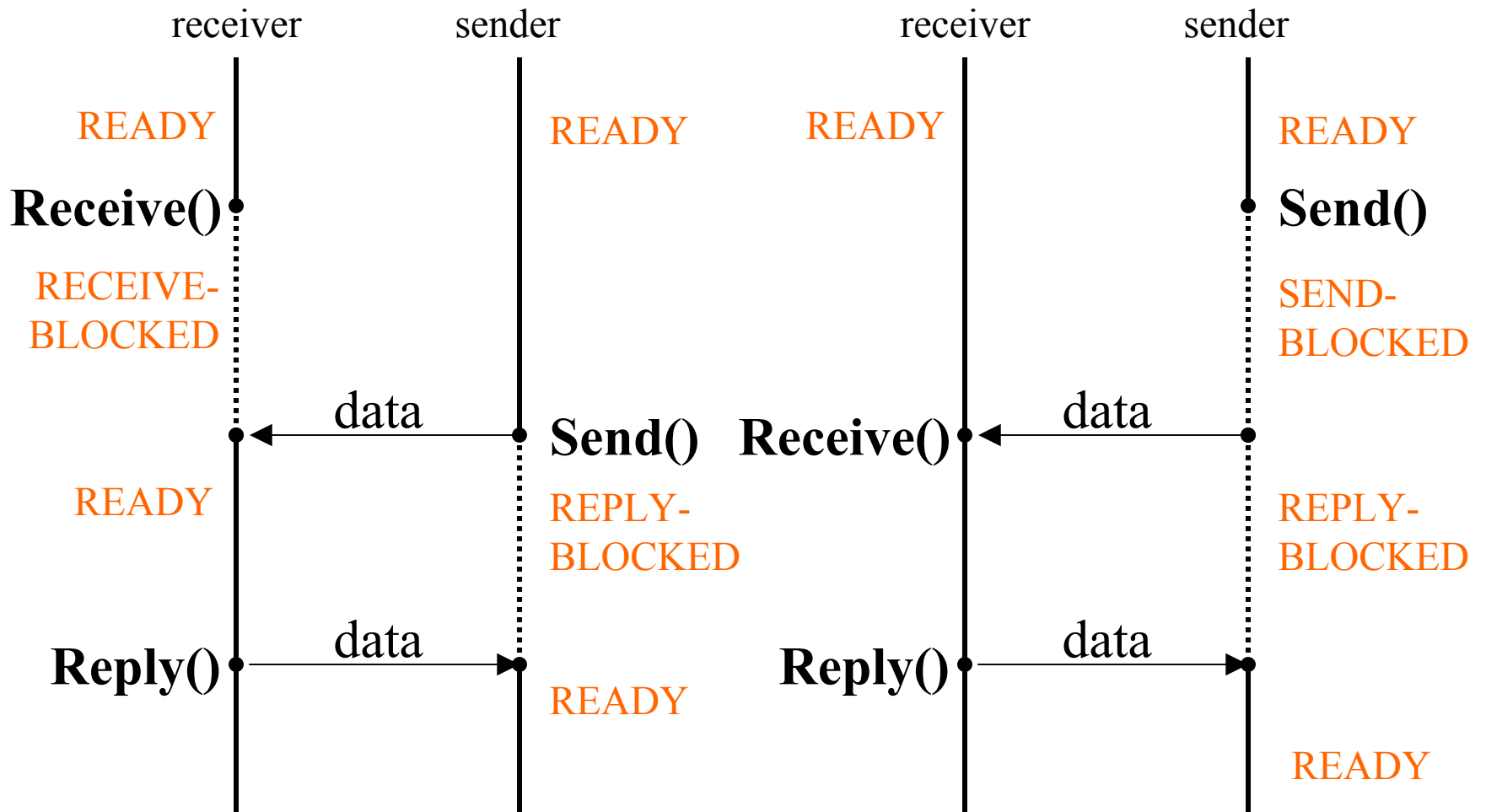




SRR model



SRR model

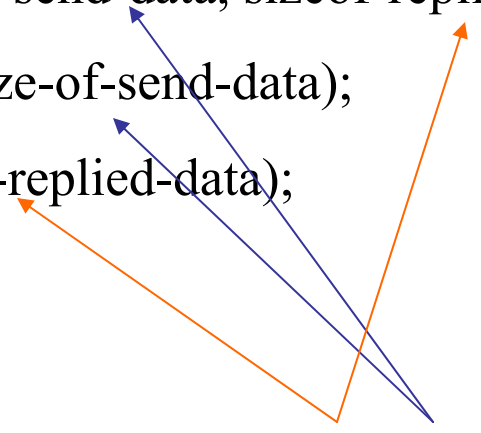


SRR model

Ktorú možnosť si programátor želá ?

Primitívy

```
Send (pid, send-data, replied-data, sizeof-send-data, sizeof-replied-data);  
pid-of-sender = Receive (0, send-data, size-of-send-data);  
Reply(pid-of-sender, replied-data, sizeof-replied-data);
```

The diagram consists of three arrows originating from a single point below the text. Two blue arrows point to the 'sizeof-send-data' and 'sizeof-replied-data' parameters in the 'Send' function call. One orange arrow points to the 'size-of-send-data' parameter in the 'Receive' function call. Another orange arrow points to the 'sizeof-replied-data' parameter in the 'Reply' function call.

Kto zaručí, že tieto veľkosti si budú zodpovedať ?

SRR model

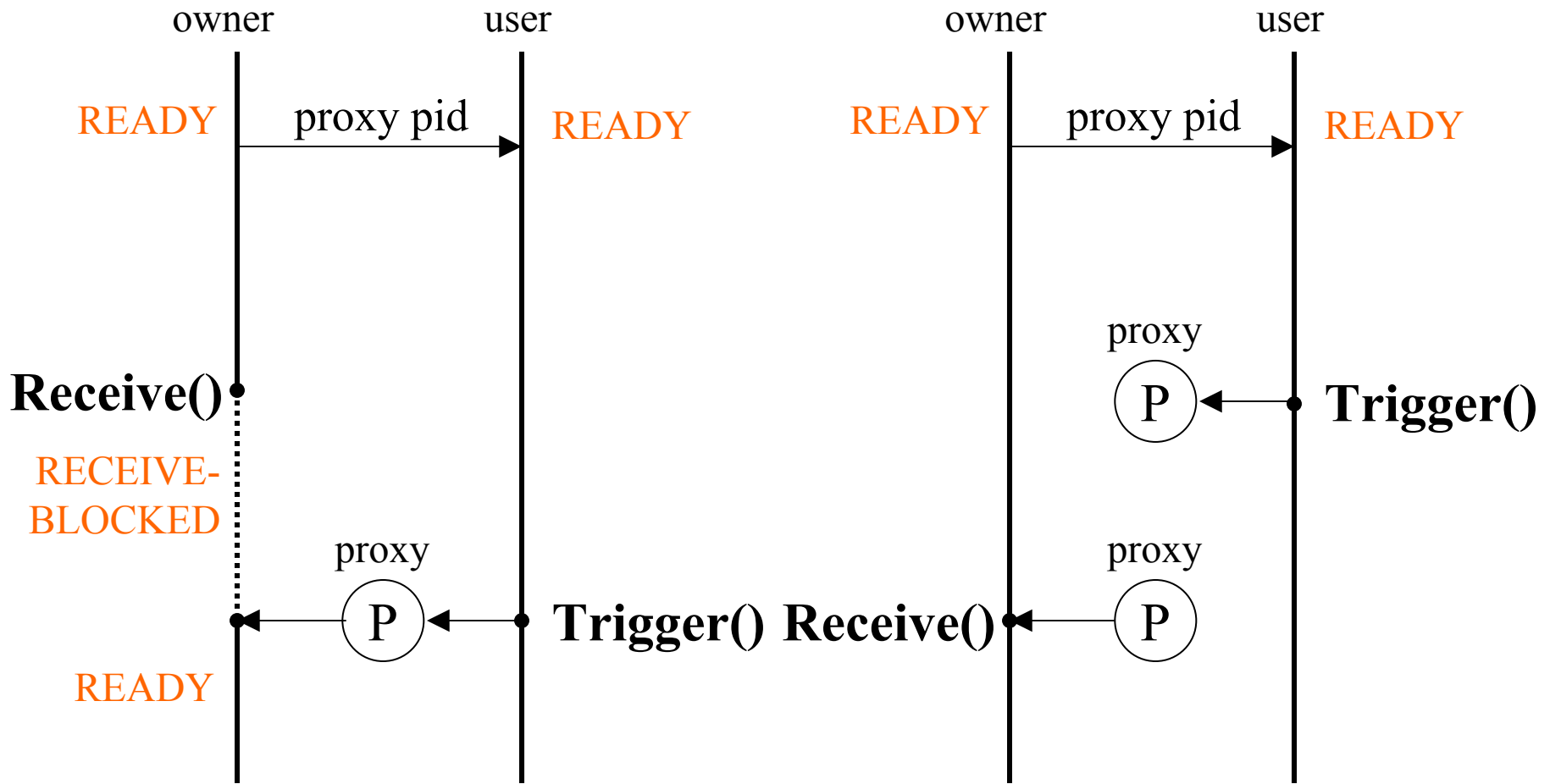
neblokujúce posielanie správ

- virtuálny proces proxy
- virtuálny proces timer

komunikácia cez LAN

- virtuálny circuit

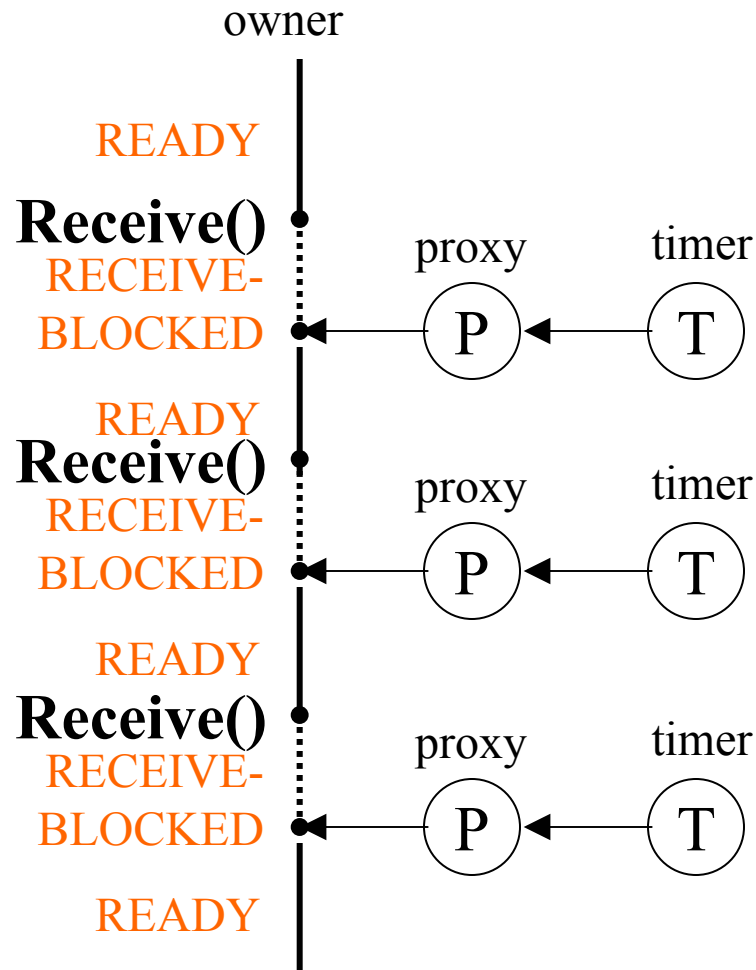
SRR model



`pidp = proxy_attach(0,0,0,-1)`

`Trigger(pidp)`

SRR model



TIMER

`pidt = timer_create(-pidp)`

SRR model

`timer_set(pidt, typ, sec0, nsec0, sec, nsec)`

Server

```
typedef struct server_msg {
    short header;
    short action;
    union {
        ...
    } data;
};
#define SERVER_HEADER 'SH'
#define SERVER_ACTION1 'A1'
...
#define SERVER_ACTIONx 'Ax'
main ()
{
    struct server_msg msg;
    // inicializacia
    for (;;) {
        pid = Receive(0,&msg,sizeof(msg));
        if (msg.header != SERVER_HEADER)
            continue;
        switch (msg.action) {
            case SERVER_ACTION1:
                // spracuj msg
                break;
            ...
            case SERVER_ACTIONx:
                ...
                break;
        }
        Reply(pid,&msg,sizeof(msg));
    }
}
```

Client-utilita

```
main ()
{
    struct server_msg msg;
    // inicializacia
    pid = name_locate("...");
    msg.header = SERVER_HEADER;
    msg.action = SERVER_ACTIONy;
    // naplni msg.data;
    Send(pid,&msg,&msg,
        sizeof(msg),sizeof(msg));
    // spracuje msg.data
}
```

Client-data collector

```
main ()
{
    // inicializacia
    pids = name_locate("...");
    pidp = proxy_attach();
    pidt = timer_create(-pidp)
    timer_set(pidt,RELATIVE,0,0,1,0);
    for (;;) {
        pid = Receive(0,NULL,0);
        if (pid == pidp) {
            // vytvor msg
            Send(pids,&msg,&msg,
                sizeof(msg),sizeof(msg));
            // spracuj msg
        }
    }
}
```

Agent-Space

Architektúra založená na nepriamej komunikácii medzi agentami

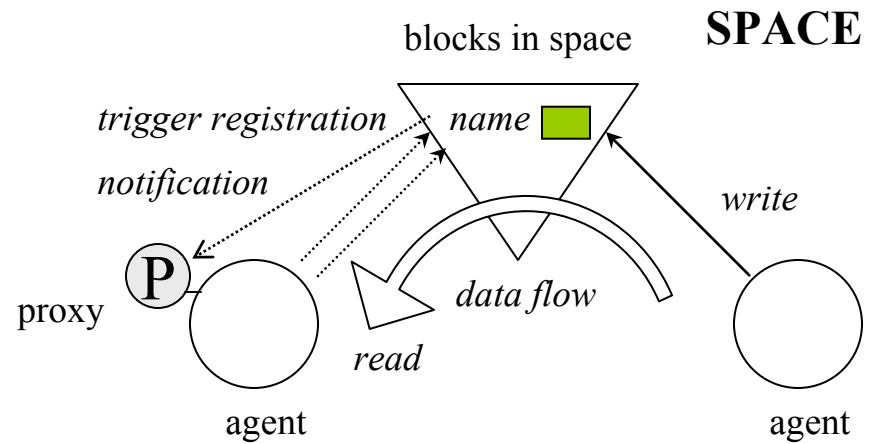
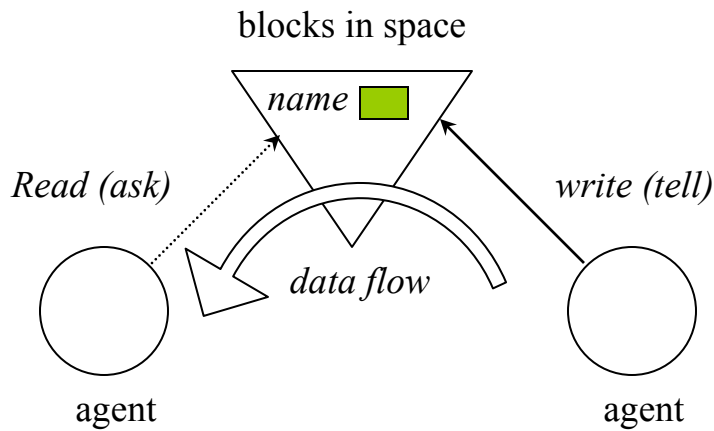
Každý proces je buď

- agent

alebo

- space

Implementácia Agent - Space pod QNX



Space

```
typedef struct server_port {  
    pid_t pid;  
    ...  
};
```

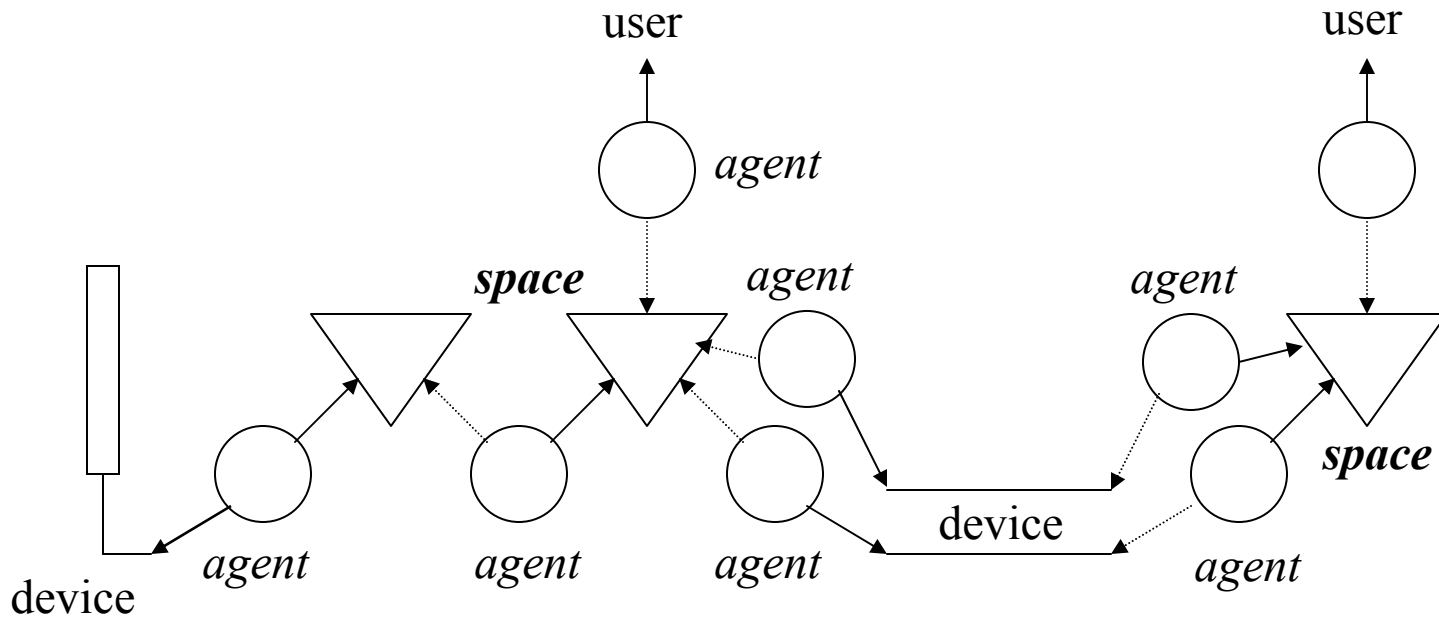
```
main ()  
{  
    struct server_msg msg;  
    struct server_port *port;  
    // inicializacia  
    ports_init();  
    for (;;) {  
        pid = Receive(0,&msg,sizeof(msg));  
        if (msg.header != SERVER_HEADER)  
            continue;  
        ports_reinit();  
        if ((port = port_get(pid)) == -1) {  
            port = port_new();  
            port_setdefaults(port);  
        }  
    }  
}
```

```
switch (msg.action) {  
    case WRITE:  
        // spracuj *port a msg  
        break;  
    case READ:  
        ...  
        break;  
    case TRIGGER:  
        ...  
        break;  
    ...  
}  
Reply(pid,&msg,sizeof(msg));  
}
```

Agent

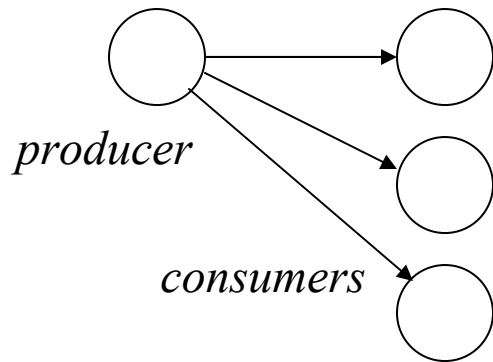
```
void main ()
{
    // initialization
    pids = qnx_name_locate(0,"SPACE");
    pidp = qnx_proxy_attach(0,0,0,-1);
    pidt = timer_create(-pidp);
    timer_set(pidt,RELATIVE,0,0,...);
    for (;;) {
        pid = Receive(0,NULL,0);           // Receive(trigger_proxy,NULL,0);
        if (pid == pidp) {
            // sense
            Send(pids,...);
            Send(pids,...);
            Send(pids,...); ...
            // select
            ...
            // act
            Send(pids,...);
            Send(pids,...);
            Send(pids,...); ...
        }
    }
}
```

Agent-Space

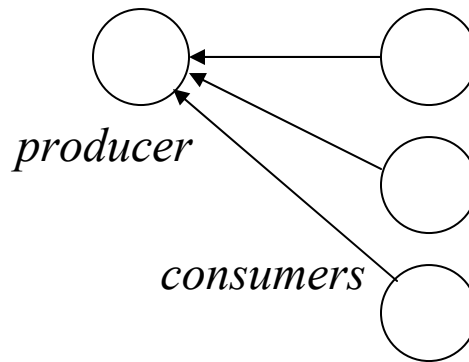


Dátový tok

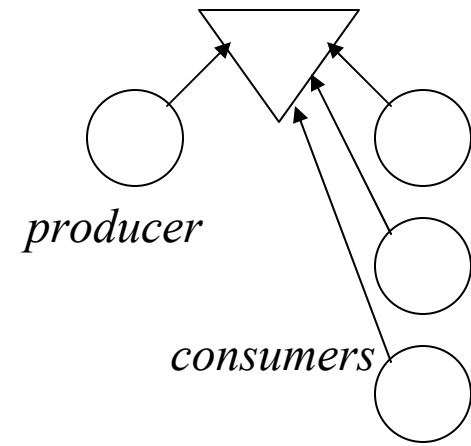
- Client-server je jednou z realizací datového toku



traditional



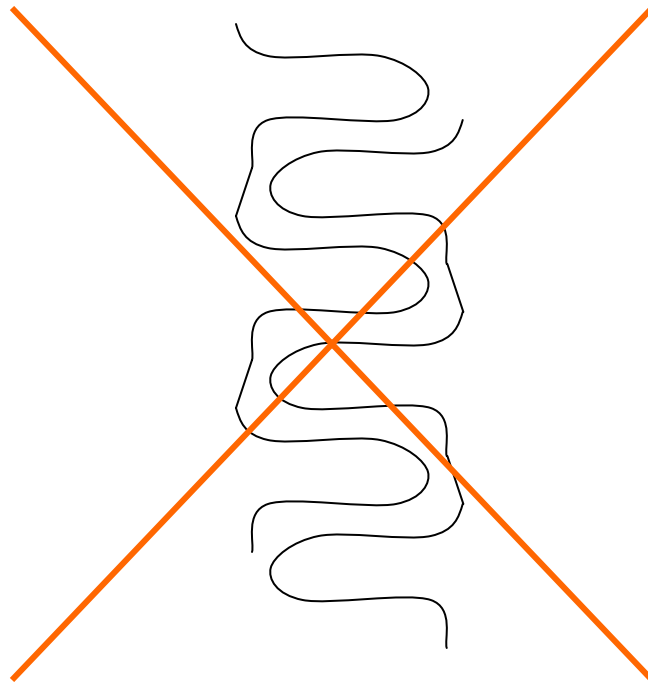
client-server



agent-space

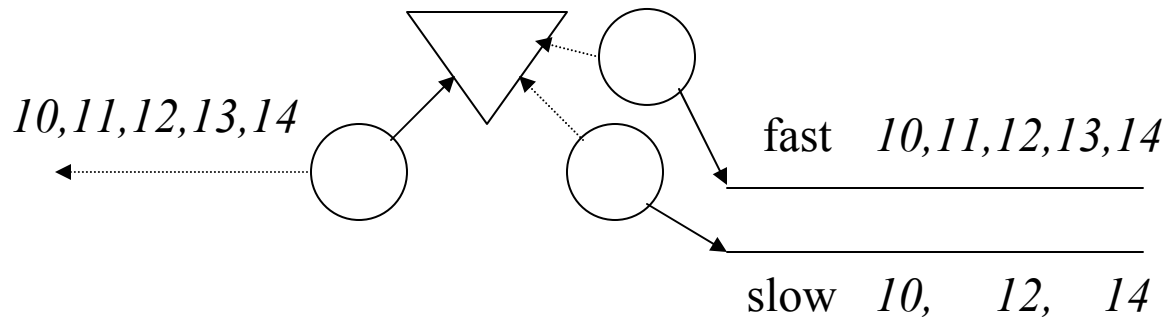
Štruktúra kódu

- Space(y) obsahuje len komunikačný kód
- Agenti obsahujú len aplikačný kód



Vlastnosti

- dead lock nemôže nastať
- live lock – každý proces dobrovolne opúšťa procesor
- podpora reálneho času – vyplýva z povahy informácie v Space – natívne vzorkovanie



Vzt'ah ku Client-Server

Agent je špeciálnejší client

Space je špeciálnejší server

Vývojové prostriedky

- File manager: nc
- Editor: vp
- Compiler: Watcom C: cc
- Toolkit: Space: MsEnv -N <name>
- Toolkit: MsAgent.h, MsAgent.Lib
- Project ALLEN

MsAgent toolkit

MsEnv -N ALLEN &

joridrv &

agwi forward 1

agwi forward 0

agent &

MsAgent library

MsAgentInsTrigger (proxy-pid, block-name, 0, 0, timeout-s, timeout-ms);

MsAgentInsTrigger (proxy-pid, NULL, period-s, period-ms, 0, 0);

MsAgentRead (block-name, read-data, sizeof-read-data, return-code);

MsAgentWrite (block-name, written-data, sizeof-written-data, return-code);

MsAgent (space-name, NULL);

Agent

```
void main ()
{
    // initialization
    ...
    pidp = qnx_proxy_attach(0,0,0,-1);
    pidt = MsAgentInsTrigger(pidp,NULL,0,500000000,0,0);
    MsAgent("ALLEN",NULL);
    for (;;) {
        pid = Receive(pidp,NULL,0);
        // sense
        MsAgentRead("block1",&block1,sizeof(block1),&ok1);
        block2 = def_value;
        MsAgentRead("block2",&block2,sizeof(block2),NULL);
        MsAgent("ALLEN",NULL);
        // select

        ...
        // act
        MsAgentWrite("block3",&block3,sizeof(block3),NULL);
        MsAgent("ALLEN",NULL);
    }
}
```