Contiki OS - Networking

Giacomo Tanganelli
PhD student @ University of Pisa
g.tanganelli@iet.unipi.it

Networking

- uIP: world's smallest, fully compliant TCP/IP stack
  - IPv6, 6LoWPAN, RPL, TCP/UDP, CoAP/HTTP
  - MAC layers:
    - Carrier Sense Multiple Access (CSMA)
    - NullMAC
  - Radio Duty-Cycling (RDC) layers:
    - ContikiMAC
    - NullRDC
    - Others
  - UIP_CONF_IPV6=1 inside Makefile

Networking UDP primitives


```
struct uip_udp_conn * udp_new (const uip_ipaddr_t
*ripaddr, uint16_t port, void *appstate)

udp_bind(conn, UIP_HTONS(port));

uip_udp_packet_send(conn, buffer, sizeof(buffer));
```

TOO DIFFICULT!

SIMPLE-UDP


```
#include "simple-udp.h“
#include "net/ip/uip.h"

int simple_udp_register (struct simple_udp_connection *c, uint16_t local_port,
uip_ipaddr_t *remote_addr, uint16_t remote_port, simple_udp_callback
receive_callback)

static void
receiver(struct simple_udp_connection *c,
        const uip_ipaddr_t *sender_addr,
        uint16_t sender_port,
        const uip_ipaddr_t *receiver_addr,
        uint16_t receiver_port,
        const uint8_t *data,
        uint16_t datalen);

simple_udp_register(&connection, UDP_PORT, NULL, UDP_PORT, receiver);
```

SIMPLE-UDP

int simple_udp_sendto (struct simple_udp_connection *c, const void *data, uint16_t datalen, const uip_ipaddr_t *to)

uip_ipaddr_t addr;

- Unicast:

  uip_ip6addr(&addr,0xfe80,0,0,0,0x212,0x7401,0x0001,0x0101);

- Broadcast:

  uip_create_linklocal_allnodes_mcast(&addr);
  simple_udp_sendto(&connection, "Test", 4, &addr);

- Also add UIP_CONF_IPV6=1 in the Makefile

See example/ipv6/simple-udp-rpl/broadcast-example.c

Exercise 1 & 2

- Write a program that loops indefinitely, check if the timer has expired, and if so, sends a broadcast message using simple-udp.
- NOTE: try to avoid congestion by introduce a random delay before the send command.

- Try to modify the previous program in order to send unicast messages. Write also a client which displays received messages.

Process interactions

- More processes in a single program
- Define events

  static process_event_t myevent;
- Allocate event struct

  myevent=process_alloc_event();
- Exchange values between processes

  process_post(&process2, myevent, &value);

  PROCESS_WAIT_EVENT_UNTIL(ev == myevent);
  value=*(int*)data;

Exercise 3

- Write a program with two processes. The first one simply check if the button has been pressed, and if so, post a user-defined event with a random number. The second process waits for the user-defined event and prints the output.