CoAP

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

CoAP

- CoAP is an application protocol similar to HTTP.
- Specifically designed for constrained environment.
- Works over UDP by default.
- In Contiki the CoAP app is called Erbium.

Erbium

- Erbium create a CoAP server on a mote:
  - A server statically defines its resources
  - Each resource has its allowed methods
  - Each resource must be implemented statically
- Erbium can be used also to deploy a CoAP client.

Copper

- Copper is a Firefox extension.
- It is a CoAP client.
- Useful to debug CoAP servers
- Can work with different CoAP version.

https://addons.mozilla.org/it/firefox/addon/copper-270430/

Define a resource

RESOURCE(**resource_example**, "title=\“Resource\";rt=\"Text\"”, get_handler, put_handler, post_handler, delete_handler);

void

get_handler(void* request, void* response, uint8_t *buffer, uint16_t preferred_size, int32_t *offset){

/* Populat the buffer with the response payload*/

REST.set_header_content_type(response, REST.type.TEXT_PLAIN);
REST.set_header_etag(response, (uint8_t *) &length, 1);

REST.set_response_payload(response, buffer, length);

}

Define a CoAP Server

```
#include “contiki.h”
#include “contiki-net.h”
#include ”rest-engine.h“
PROCESS(server, ”CoAP Server");
AUTOSTART_PROCESSES(&server);
PROCESS_THREAD(server, ev, data){
        PROCESS_BEGIN();
        rest_init_engine();
        rest_activate_resource(&resource_example, “resource_path”);
        while(1) {
                PROCESS_WAIT_EVENT();
        }
        PROCESS_END();
}
```

Makefile

```
UIP_CONF_IPV6=1
SMALL=1
CFLAGS += -DUIP_CONF_IPV6=1
CONTIKI=/home/user/contiki
CFLAGS += -DPROJECT_CONF_H=\"project-conf.h\”
CFLAGS += -DUIP_CONF_TCP=0
APPS += er-coap
APPS += rest-engine
```

Project-conf

```
#undef IEEE802154_CONF_PANID
#undef NETSTACK_CONF_RDC
#define NETSTACK_CONF_RDC     nullrdc_driver
#undef NETSTACK_CONF_MAC
#define NETSTACK_CONF_MAC     nullmac_driver
#undef REST_MAX_CHUNK_SIZE
#define REST_MAX_CHUNK_SIZE    64
#undef COAP_MAX_OPEN_TRANSACTIONS
#define COAP_MAX_OPEN_TRANSACTIONS   4

/* Save some memory for the sky platform. */
#undef NBR_TABLE_CONF_MAX_NEIGHBORS
#define NBR_TABLE_CONF_MAX_NEIGHBORS     10
#undef UIP_CONF_MAX_ROUTES
#define UIP_CONF_MAX_ROUTES   10
#undef UIP_CONF_BUFFER_SIZE
#define UIP_CONF_BUFFER_SIZE  280
```

Exercise 1

- Deploy a CoAP server with only one resource.
- The resource must allow the GET method.
- Use Copper to interact with the CoAP server. Try CON and NON messages.

- NOTE: in order to interact between Copper (running on the host) and the CoAP server (running in Cooja) a border router is needed.

Change a resource

- Retrieve method:
  - uint8_t method = REST.get_method_type(request);
- Check method:
  - if (method & METHOD_POST)
- Set response:
  - REST.set_response_status(response, REST.status.CREATED);

Exercise 2

- Modify the last exercise in order to have the resource which accept GET and POST.
- If the client sends a POST request the server must use the request payload to update the resource value.
- The GET behaviour must be the same as before.

Parameters

- Get a query parameter:
  - REST.get_query_variable(request, "color", &color)
- Get a post parameter:
  - REST.get_post_variable(request, "mode", &mode)
- Analyze parameter:
  - strncmp(mode, "on", len)

Exercise 3

- Write a CoAP server with a resource which change the status of the leds depending on query and post parameters.
- Query parameter:
  - color=r|g|b
- Post parameter:
  - mode=on|off