CoAP

Carlo Vallati
Assistant Professor @ University of Pisa
c.vallati@iet.unipi.it

CoAP

- CoAP is an application protocol similar to HTTP.
- Specifically designed for constrained environment.
- Works over UDP by default.

- It exposes functionalities provided by things as resources that can be discovered and accessed in the same way browsers access HTTP resources

CoAP

- In a typical CoAP deployment a sensor behaves as a server, offering resources to applications (CoAP clients) to gather data
- Other configurations are possible, e.g. a sensor behaving as CoAP client to interact with other sensors or external CoAP servers (to register)

Erbium

- Erbium is the Contiki implementation of CoAP
- It implements both server and client functionalities
- For servers the following limitations hold:
  - A server statically defines its resources
  - Each resource has its allowed methods
  - Each resource must be implemented statically

Copper

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

coap://vs0.inf.ethz.ch/

Makefile

- Enable IPv6 and RPL:

```
UIP_CONF_IPV6=1
CFLAGS += -DUIP_CONF_IPV6=1
CFLAGS += -DUIP_CONF_IPV6_RPL=1
```

- Include a project-conf.h file and disable TCP (to reduce OS footprint):

```
CFLAGS += -DPROJECT_CONF_H=\"project-conf.h\"
CFLAGS += -DUIP_CONF_TCP=0
```

- Enable CoAP and the REST engine:

```
APPS += er-coap
APPS += rest-engine
```

Configure the project-conf.h

- Set the CoAP parameters:

```
// Set the max response payload before enable fragmentation:
#undef REST_MAX_CHUNK_SIZE
#define REST_MAX_CHUNK_SIZE    64
// Set the maximum number of CoAP concurrent transactions:
#undef COAP_MAX_OPEN_TRANSACTIONS
#define COAP_MAX_OPEN_TRANSACTIONS   4
```

- Set some IPv6 parameters to reduce OS size (yes, CoAP is large):

```
/* 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
```

Define a CoAP Server


```
#include “contiki.h”
#include “contiki-net.h”
#include "rest-engine.h”

PROCESS_THREAD(server, ev, data){
      PROCESS_BEGIN();
      rest_init_engine();
      rest_activate_resource(&res_hello, “hello”);
      while(1) {
            PROCESS_WAIT_EVENT();
      }
      PROCESS_END();
}
```

Define a resource

```
RESOURCE(name_resource, "attributes", get_handler,
post_handler, put_handler, delete_handler);


void
get_handler(void* request, void* response, uint8_t
*buffer, uint16_t preferred_size, int32_t *offset){
  …
  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);
}
```

Exercise 1

- Deploy a CoAP server with a 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 a border router is needed.

- er-server-only-get.c

Change a resource (PUT–PUSH)

- 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);
  - REST.set_response_status(response, REST.status.BAD_REQUEST);

Parameters

- Get a query parameter (**URL?value=10**):
  - REST.get_query_variable(request, "color", &color)
- Get a post parameter (**value=10 in the post or put payload**):
  - REST.get_post_variable(request, "mode", &mode)
- Analyze parameter:
  - if(strncmp(mode, "on", len) == 0)

Exercise 2

- Add a resource that accepts both GET and PUT requests to retrieve and set an integer value, respectively.
- If the client sends a PUT request the server must update the value. Post parameter “value=20”
- The GET must return the stored value.

- er-server-put.c

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

- er-leds.c

Californium

Carlo Vallati
Assistant Professor@ University of Pisa
c.vallati@iet.unipi.it

Californium

- Californium is a Java CoAP library
- Easy and simple interface to deploy CoAP enabled applications
- It allows the implementation of CoAP server and clients

Include Californium using Maven

- Californium can be easily included in any Java project, e.g. a Tomcat servlet or an Android application, using Maven
- Apache Maven is a build automation tool for Java project that can help in managing “dependencies”, i.e. external libraries
- Maven exploits a configuration file (pom.xml) to download and install all the required dependencies

Use maven

- Install maven:
  - sudo apt-get install maven
- Make the pom.xml file of your program
- Compile your project using Maven:
  - mvn install
- During this process Maven will download and install californium from a repository
- Download with:
  - git clone https://github.com/eclipse/californium.git

Modify californium

- In case modifications to the californium library are requried the library can be installed from its sources using maven as well
- Download the library source code:
  - git clone https://github.com/eclipse/californium.git
- Compile the library:
  - mvn install
- Maven will override the binaries downloaded from external repositories

GET Example

```
URI uri = null;

try{
		uri = new URI("coap://[aaaa::c30c:0:0:9e]:5683/hello");
} catch (Exception e) {
		System.err.println("Caught Exception: " + e.getMessage());
}

CoapClient client = new CoapClient(uri);

CoapResponse response = client.get();

if (response!=null) {

		System.out.println(response.getCode());
		System.out.println(response.getOptions());
		System.out.println(response.getResponseText());

		System.out.println("\nADVANCED\n");
		// access advanced API with access to more details through .advanced()
		System.out.println(Utils.prettyPrint(response));

} else {
		System.out.println("No response received.");
}
```