Tutorial on Wakaama

Internet of Things (2IMN15) 2016-2017, Eindhoven University of Technology
By Leila F. Rahman

Eclipse Wakaama is an open source implementation of the OMA LWM2M protocol in C language. For more information about Wakaama, you can visit its web site at https://projects.eclipse.org/projects/technology.wakaama and it GitHub repository at https://github.com/eclipse/wakaama.

This tutorial will discuss about how to install and execute Wakaama on the Raspberry Pi, how to develop a LWM2M client using Wakaama client example, how to develop a broker or cloud application using the Wakaama server example, and some ideas on the tools you can use for developing a Web application for the broker or the cloud service.

1 Installing and Executing Wakaama on Raspberry Pi

1. Download the Wakaama source code from http://www.win.tue.nl/~lrahman/iot_2016/tutorial/wakaama-master.zip using the following command:
   `wget http://www.win.tue.nl/~lrahman/iot_2016/tutorial/wakaama-master.zip`

2. Extract the zip file using the following command:

   `unzip wakaama-master.zip`

3. Build and run the client and server examples as specified in the `wakaama-master/README.md` file
4. As we are using IPv4 for the assignment, use option -4 when running the client or server (see Figure 1 and Figure 2). Other available options for running the client:

   - -n NAME Set the endpoint name of the Client. Default: testlwm2mclient
   - -l PORT Set the local UDP port of the Client. Default: 56830
   - -h HOST Set the hostname of the LWM2M Server to connect to. Default: localhost
   - -p HOST Set the port of the LWM2M Server to connect to. Default: 5683
   - -4 Use IPv4 connection. Default: IPv6 connection
   - -t TIME Set the lifetime of the Client. Default: 300
   - -b Bootstrap requested.
   - -c Change battery level over time.

Figure 1 Running Wakaama lwm2mserver with option -4

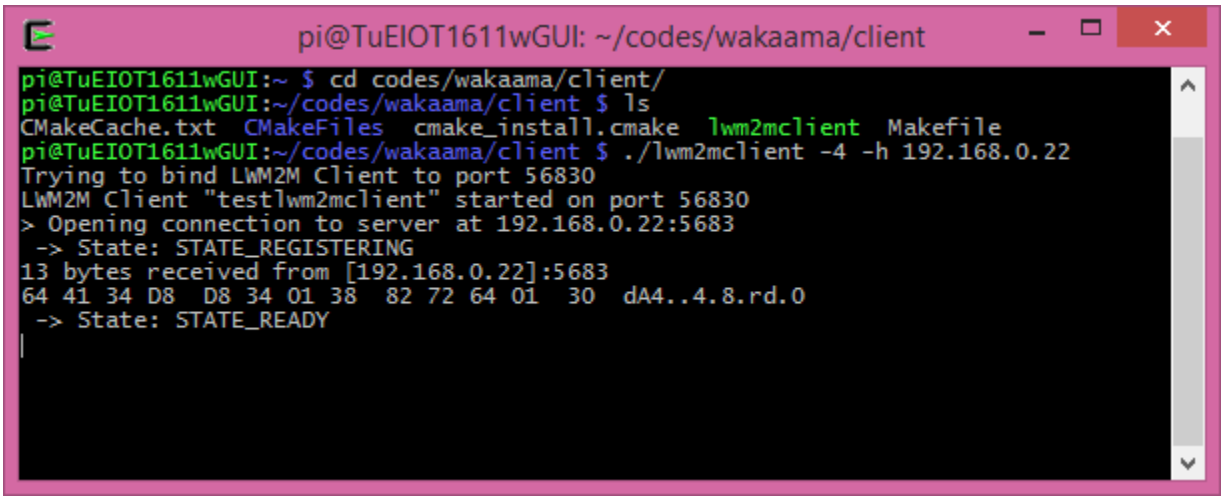

Figure 2 Running Wakaama lwm2mclient with option -4 and –h

5. Figure 3 and Figure 4 show the runtime operation of lwm2mserver.

```
pi@TuEIOT1611wGUI: ~/codes/wakaama/server
pi@TuEIOT1611wGUI:~/codes/wakaama/server $ ./lwm2mserver -4
> 153 bytes received from [192.168.0.22]:56830
44 02 34 D8  D8 34 01 38  B2 72 64 11  28 3D 05 65   D.4..4.8.rd.(=.e
70 3D 74 65  73 74 6C 77  6D 32 6D 63  6C 69 65 6E   p=testlwm2mclien
74 03 62 3D  55 06 6C 74  3D 33 30 30  FF 3C 2F 3E   t.b=U.lt=300.</>
3B 72 74 3D  22 6F 6D 61  2E 6C 77 6D  32 6D 22 3B   ;rt="oma.lwm2m";
63 74 3D 31  35 34 33 2C  3C 2F 31 2F  30 3E 2C 3C   ct=1543,</1/0>,<
2F 32 2F 30  3E 2C 3C 2F  33 2F 30 3E  2C 3C 2F 34   /2/0>,</3/0>,<4
2F 30 3E 2C  3C 2F 35 2F  30 3E 2C 3C  2F 36 2F 30   /0>,</5/0>,</6/0
3E 2C 3C 2F  37 2F 30 3E  2C 3C 2F 31  30 32 34 2F   >,</7/0>,</1024/
31 30 3E 2C  3C 2F 31 30  32 34 2F 31  31 3E 2C 3C   10>,</1024/11>,<
2F 31 30 32  34 2F 31 32  3E                         /1024/12>

New client #0 registered.
Client #0:
        name: "testlwm2mclient"
        binding: "UDP"
        lifetime: 300 sec
        objects: /1/0, /2/0, /3/0, /4/0, /5/0, /6/0, /7/0, /1024/10, /1024/11, /
1024/12,

> help
help    Type 'help [COMMAND]' for more details on a command.
list    List registered clients.
read    Read from a client.
disc    Discover resources of a client.
write   Write to a client.
time    Write time-related attributes to a client.
attr    Write value-related attributes to a client.
clear   Clear attributes of a client.
exec    Execute a client resource.
del     Delete a client Object instance.
create  create an Object instance.
observe Observe from a client.
cancel  Cancel an observe.
q       Quit the server.

> help read
 read CLIENT# URI
   CLIENT#: client number as returned by command 'list'
   URI: uri to read such as /3, /3/0/2, /1024/11, /1024/0/1
Result will be displayed asynchronously.
```

Figure 3 Runtime operation of Wakaama lwm2mserver

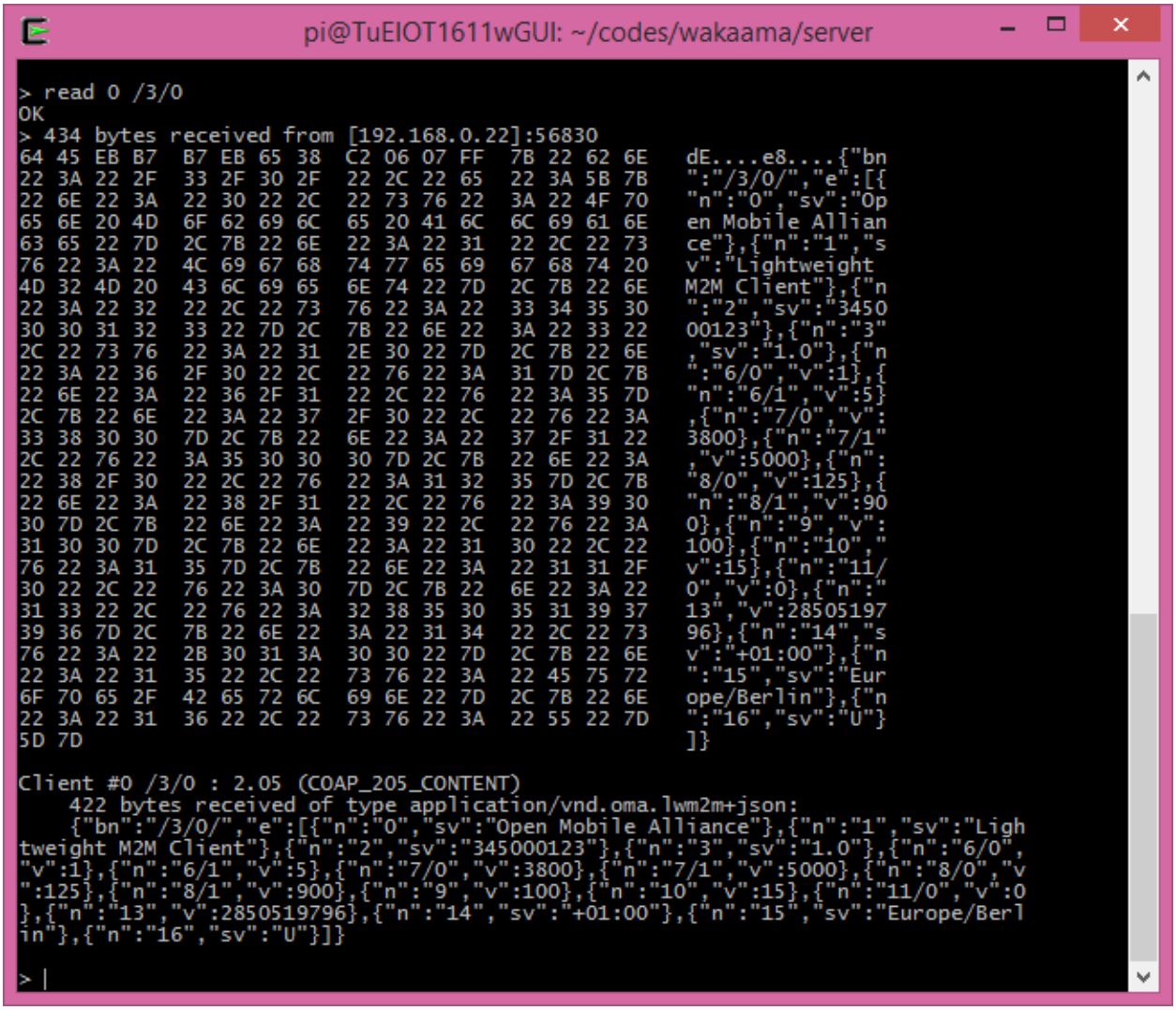

Figure 4 Read operation of Wakaama lwm2mserver

2 Developing LWM2M Client on Raspberry Pi

1. Developing your LWM2M client (for the Light Device and the Sensor device) is done by modifying the client example source code in the `../wakaama-master/examples/client` folder.
2. Create new C file for every new object in your LWM2M client. For example, for the LWM2M client in the Light Device, you need to create a new `object_light_profile.c` file for the Light Profile object by modifying the `object_device.c` file in the `../wakaama-master/examples/client` folder. Adjust the `write` and `read` functions for the Light Profile object in `object_light_profile.c`. Adjust as well the `get_object` and `free_object` functions in `object_light_profile.c`.

3. Adjust the `lwm2mclient.c` and `lwm2mclient.h` files in `../wakaama-master/examples/client` folder accordingly. In `lwm2mclient.c,` you need to initialize the Light Profile object in the beginning of the main function, and free the object memory allocation in the end of the main function. In `lwm2mclient.c,` you also need to set the right number of objects in your LWM2M client.
4. Adjust file `../wakaama-master/core/liblwm2m.h` by adding the ID of the new object under line 167.
5. Adjust file `../wakaama-master/examples/client/CMakeLists.txt` to include the new object file and any new libraries used in the modified code.
6. Build and run the new lwm2mclient.

3 Developing a Broker or Cloud Application Using Wakaama Server

1. Open file `../wakaama-master/core/liblwm2m.h` and look at lines 703 – 722. Those lines provide a list of APIs that you can use in your broker or cloud service implementation.
2. Another option is to create Inter Process Communication (IPC) between your broker or cloud application and the Wakaama server. The IPC is used to send command data accepted by Wakaama server during its runtime (see Figure 3 and Figure 4 for the list of commands) by your broker or cloud application to the Wakaama server.

4 Developing Web Application for your Broker or Cloud Service

Some possible (not limiting) options:

1. Use Wt, a C++ Web Toolkit (https://www.webtoolkit.eu/wt).
2. Use Django, a Python Web Framework (https://www.djangoproject.com/), or any other Python Web framework (https://wiki.python.org/moin/WebFrameworks).
3. Use Spark, A micro framework for creating web applications in Java 8 with minimal effort (http://sparkjava.com/).
4. Use node.js (https://nodejs.org/en/) as backend and AngularJS (for example) as front end.
5. Use Go programming language (https://golang.org/).
6. Use any other framework, tools or libraries that you wish or that you are familiar with.