Deliverable FI3-D3.1.7

Prototype of a communication layer for Internet of Things running on top of DHT algorithms

Jaime Jiménez, Jouni Mäenpää

Tivit Future Internet Program
(Tivit FI)

Period: 1.4.2011 – 30.9.2011

Tivit, Strategisen huippuosaamisen keskittymän tutkimusohjelma

Rahoituspäätös 1171/10, 30.12.2010, Dnro 2790/31/2010

www.futureinternet.fi

www.tivit.fi

This work was supported by TEKES as part of the Future Internet programme of TIVIT (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT).

Executive summary / Internal release

Title:

Communication Layer for Internet of Things/Machine-to-machine networks.

Content: Contains the Deliverable Information, source code and binaries of the prototype. It also contains three theses that contain extra relevant information.

Impact: Most visions of the Future Internet of Things use centralize approaches to the use of sensors and actuators. We have developed a prototype for the decentralization of this elements.

Contact info: Jaime Jimenez, Jouni Mäenpää, jaime.j.jimenez@ericsson.com, jouni.maenpa@ericsson.com

Link: http://users.piuha.net/jaime/ICTSHOK/

Table of Contents

1. Introduction

Machine-to-machine (M2M) communications is a field of research expected to grow in the following years. New business opportunities arise in this area. Nearly all Machine-to-Machine (M2M) architectures today are based on some types of centralized elements. The centralized elements are often physical servers that provide a set of functionalities for sensors and actuators in M2M networks. The provided functions can be, for example, functions that provide protocol conversion, or monitoring and controlling facilities.

Well known structured P2P networks, for instance by means of a Distributed Hash Table (DHT), present great synergy possibilities with M2M, in particular in the Wireless Sensor Networks (WSN) Area. M2M scenarios in which sensors become more autonomous and self-reliant, independent from a centralized decision-making entity can benefit from the use of DHTs. The goal of this work in ICT-SHOK is to create building blocks for such M2M architectures that reduce network's dependency of centralized elements. The dependency is reduced by shifting functionalities from centralized elements to sensor and actuator devices which can be considered as edge-devices in M2M networks. The resulting M2M network architecture is referred to, in this context, as autonomous M2M network:

Task: Machine-to-machine Communication Enabler

A software module for distributing Sensor and Actuator Information, named Machine-to-machine Communication Enabler (M2MCE), has been designed and implemented. The M2MCE runs on top of a DHT, more specifically Chord. It leverages the distribution offered by the structured P2P topology and the centralized control offered by the M2M Network. The M2MCE offers also a Distributed Rendezvous Service to locate nodes in the overlay in a similar fashion as a Distributed Name Service (DNS) operates. In addition, it also offers a node information database, performing caching of the sensor information in the overlay itself. This is important since it enables energy efficiency in the sensor nodes and provides a common knowledge base of sensor information. The M2MCE bridges two functionalities offered by the M2M network: proxying and monitoring.

The proxying functionality is placed in the border of a Wireless Personal-Area Network (WPAN) and a Wireless Wide-Area Network (WWAN). It is noteworthy, that the proxy itself runs on a moderate edge device, as opposed to being run on a centralized physical server. This is possible since the Proxy itself is part of the DHT and runs the M2MCE. The proxy enables communication between the devices in WPAN (e.g., ZigBee) and the devices in WWAN (e.g., 3G) by means of specific proxying software and the M2MCE.

A key feature of the monitoring and controlling software is that it does not have to be continuously present in a M2M network, like most of the monitoring and controlling applications today. The monitoring and controlling facility provides information about the devices in the M2M network and to allow the manipulation of devices configuration parameters.

The research was carried out in scope of the Master's Thesis "Adapting a DHT (Distributed Hash Table) to a Self-Reliant M2M (Machine-to-Machine) Network" [1]. Two other thesis address the management "Management of Decentralized DHT Based M2M Network" [2] and the proxy "A Proxy for Distributed Hash Table based Machine-to-Machine Networks" [3]. They are all publicly available but included in the deliverable for ease of access.

We provide the procedure for installation and execution of the software in this document, and the thesis should be referred for complete description of the software modules. The network architecture for the autonomous M2M network given below for quick reference.

2. Software and Hardware Specification

The complete specification of the hardware and software used in the testbed are given in Section 4 of [1].

3. Deliverables

The deliverables consist of the following items:

- ✓ A document describing the project and the deliverables (i.e. this document)
- ✓ Source code for M2MCE module
  - ❖ p2p-ce : M2MCE module
    - p2p-ce/m2m: M2MCE application.
    - p2p-ce/CoAP: CoAP module (jCoAP source, not Ericsson's).
    - p2p-ce/BouncyCastle: Cryptographic module (Bouncy Castle source, not Ericsson's).
    - p2p-ce/P2pSipSE: DHT binaries. Other DHT's might be used.
- ✓ Pre-built binaries and configuration files for the above software modules are also provided for connivence (since the software uses Java, the binaries are readily usable). We also include pre-built binaries for monitoring and controlling module:
  - M2mManager.jar
  - snmp4j-1.11.3.jar

- m2m.jar - this is for simulated WNs at bootstrap server - it automatically joins DHT P2P on launch.
- M2mCeRmiInstance_noJoin.jar - this for all other devices - wn/pn/mcn - it does not automatically joins DHT P2P, but awaits for snmp-agent/shell to invoke join().

✓ Some *Expect* scripts for automation of execution of testbed-software

All the items packaged in a zip file.

4. Installation

As mentioned in [1], there are 4 different types of devices in the network - *WideArea Node* (WN), *Proxy Node* (PN), *Local Node* (LN), *Monitoring and Controlling Node* (MCN). In addition, the testbed has a P2P Bootstrap-Peer devices which runs simulated WNs (the number of simulated WNs is configurable via command line). Bootstrap peer machine also hosts a Smart-M3 Knowledge Process necessary for providing command line interface for humans (henceforth we will refer to this as *Smart-M3 Client)*. Hence, there are 6 different type of devices in the network. The software to be installed (directly copied from the zip-file) onto these devices depends upon the type of device:

➡ **Bootstrap Peer device**

✓ This device also hosts Smart-M3 Client

✓ Under ${INSTALL_DIR}/

- m2m.jar
- M2mManager.jar
- mcnlaunch (expect script to automatically launch MCN)
- p2pm2m_demo.sh (expect script to automatically trigger the whole demo)
- pnlaunch (expect script to automatically launch PN)
- start_m3.sh (expect script to automatically launch Smart-M3 client)
- wnlaunch (expect script to automatically launch WN)
- client.policy (for Java socket access permissions)
- server.policy (for Java socket access permissions)

✓ ${INSTALL_DIR}/platform : install the Smart-M3 platform [6] here using the procedure described in the setup document in [6]

➡ **WideArea Node (WN)**

- ✓ Under ${INSTALL_DIR}/
  - M2mCeRmiInstance_noJoin.jar
  - M2mManager.jar
  - TestAgentM2mMIBConfig.properties (file to configure default SNMP MIB values)
  - log4j.properties (to control log4j trace level)
  - start_m2m.sh (expect script to automatically launch WN)
  - wn101.txt (file to configure P2P attributes)
  - client.policy (for Java socket access permissions)
  - server.policy (for Java socket access permissions)
- ✓ ${INSTALL_DIR}/../platform : install the Smart-M3 platform [6] here using the procedure described in the setup document in [6]

➡ **Proxy Node (PN)**

- ✓ Under ${INSTALL_DIR}/
  - M2mCeRmiInstance_noJoin.jar
  - M2mManager.jar
  - TestAgentM2mMIBConfig.properties (file to configure default SNMP MIB values)
  - log4j.properties (to control log4j trace level)
  - start_m2m.sh
  - wn100.txt (file to configure P2P attributes)
  - client.policy (for Java socket access permissions)
  - server.policy (for Java socket access permissions)
- ✓ ${INSTALL_DIR}/../platform : install the Smart-M3 platform [6] here using the procedure described in the setup document in [6]

➡ **Local Node (LN)**

- ✓ Upload DemoXbee.pde using the Ardurino IDE (provided in labsetup/pn/)

➡ **Monitoring and Controlling Node (MCN)**

- ✓ Under ${INSTALL_DIR}/

- M2mCeRmiInstance_noJoin.jar
- snmp4j-1.11.3.jar
- start_mcn.sh (expect script to automatically launch MCN)
- wn102.txt (file to configure P2P attributes)
- client.policy (for Java socket access permissions)
- server.policy (for Java socket access permissions)

The following files need to be updated with the IP address, interfaces, user id and passwords of the respective devices - p2pm2m_demo.sh, wn100.txt, wn101.txt, , wnlaunch, pnlaunch, mcnlaunch (self evident in the last 3 scripts). For files p2pm2m_demo.sh, wn100.txt, wn101.txt, the following two fields need updation:

•interface; <interface of local machine for communication e.g. eth0>

•bootstrapPeerIP; <IP address of bootstrap peer>

5. Execution Procedure

The execution of test setup is flexible, and the startup is typically done as shown in FIgure 3.

➡ The demo is started by launching p2pm2m_demo.sh on the bootstrap peer device (shown in Figure 3).

➡ The MCN interface can be used to check and enable/disable SmartM3 on various devices like WNs and PNs (shown in Figure 5).

➡ The command *lsn* can be used to use the bookkeeping feature and list available devices. The information is updated automatically as devices join/ leave. (shown in Figure 4,5,6).

➡ The Smart-M3 client interface can be used to interact with the resources on WN/PN/LN via Smart-M3 SIB (shown in Figure 6).

The snapshot's shown below illustrate the system working. Please note the title on terminal to identify the device type - PN / WN / MCN / Bootstrap / SmartM3-Client.

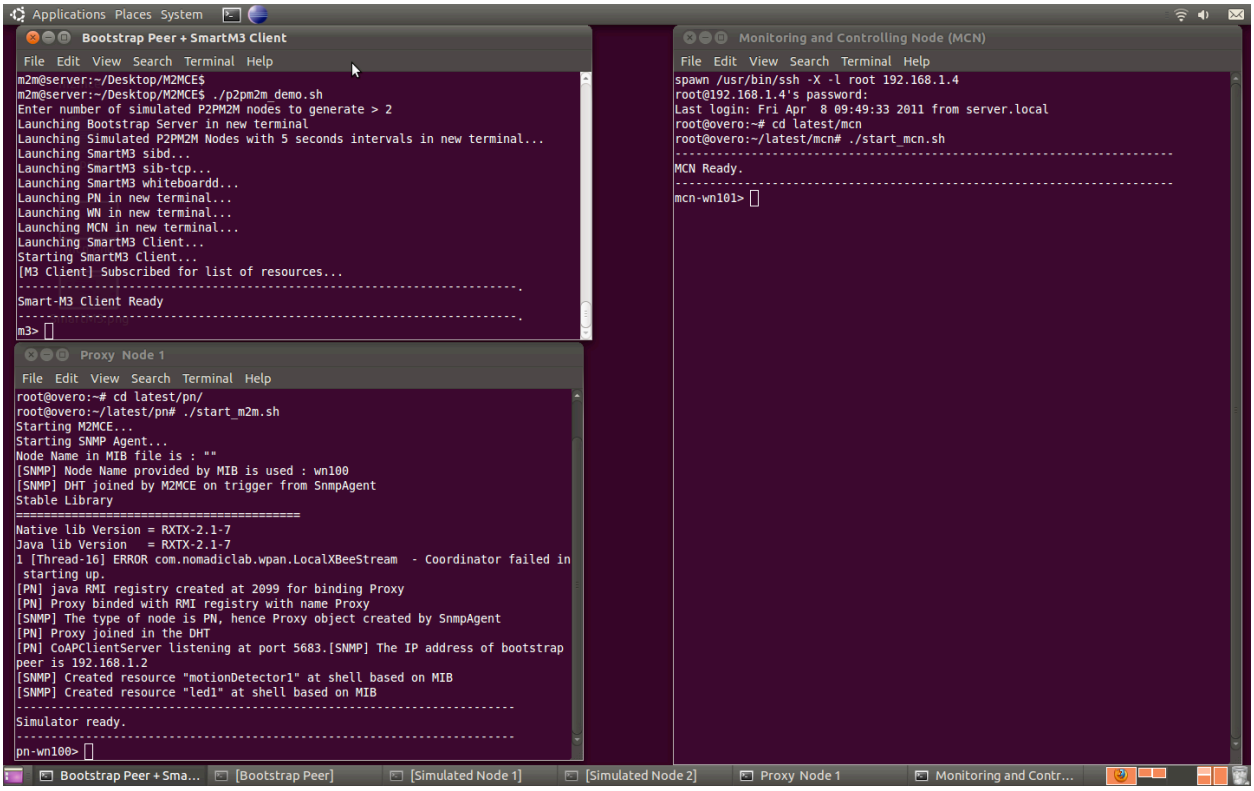

Figure 3 : System Startup using the automation scripts

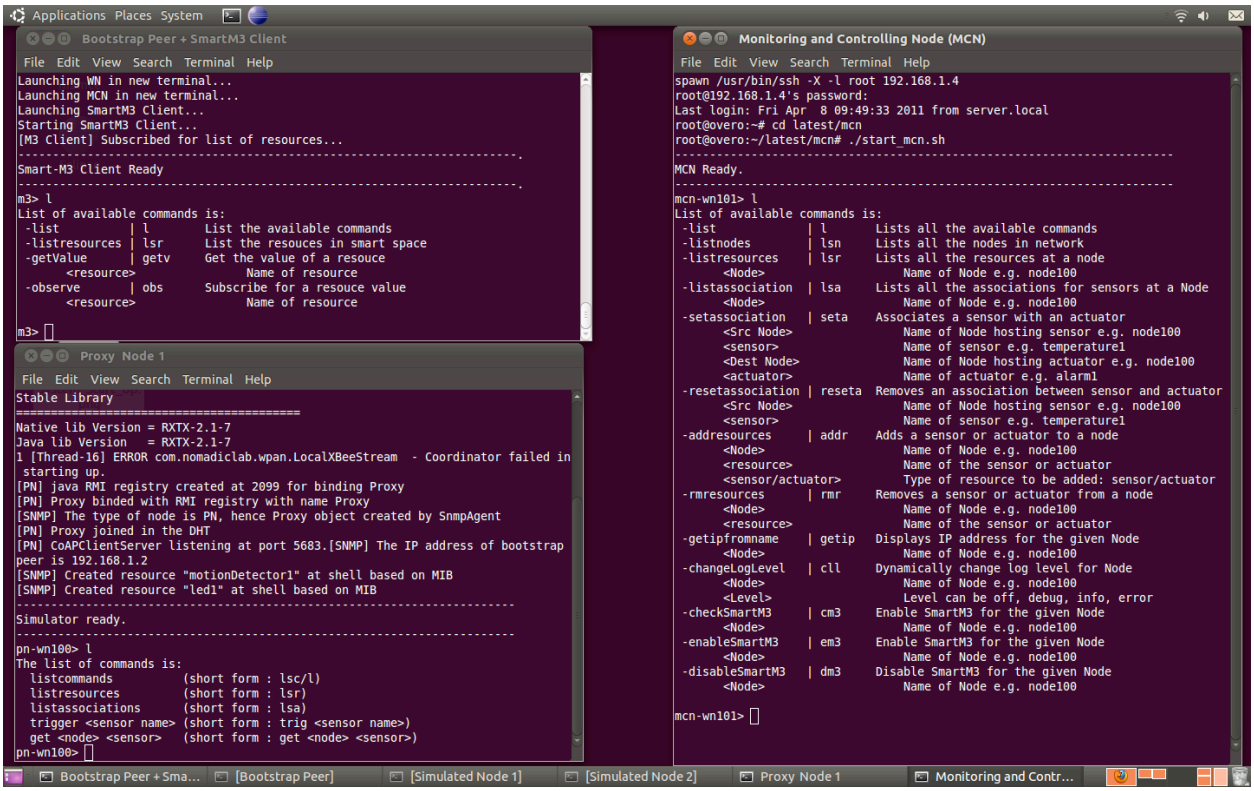

Figure 4 : Displaying the available commands on different devices

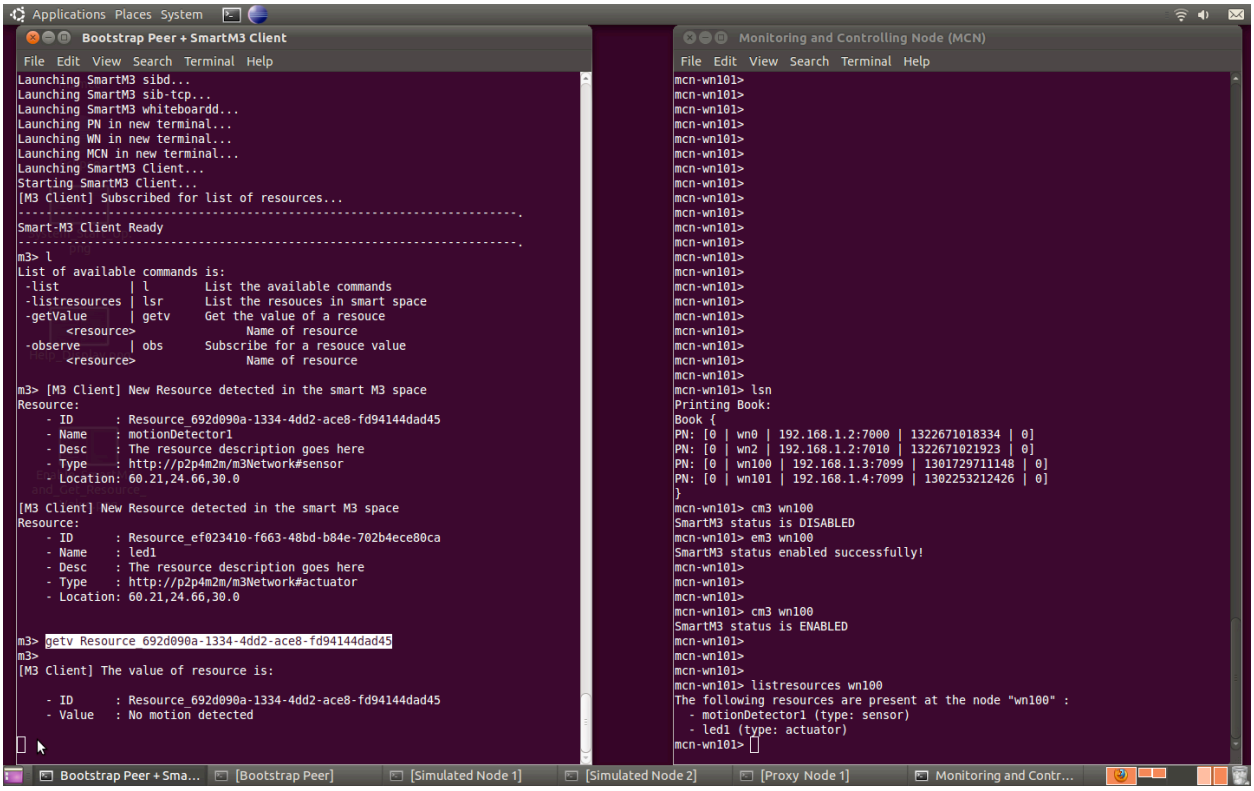

Figure 5: Using M2MCE to get value of a resource on WN device

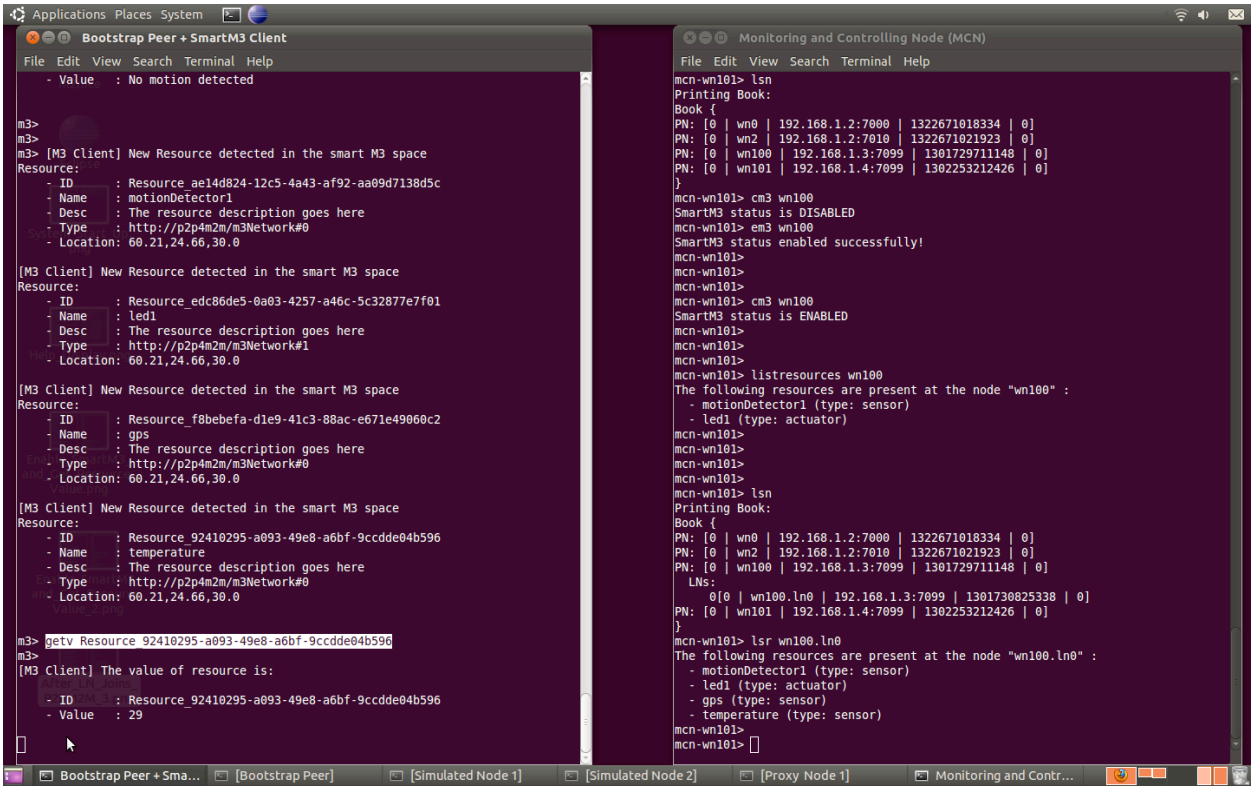

Figure 6: When LN joins the system, PN is responsible for representing LNs resource in Smart-M3 SIB, and, using Smart-M3 Client to get value of a resource on LN device (which is handled by PN)

6. Common problems in installation and execution

3 problems seen: sofia Java lib on Cacao JVM, Armstrong linux X11 issues, SmartM3 platform build issue.

- ✓ Cacao JVM has some issue due to which the wrapper Java libraries for Smart-M3 platform, smartm3-javakpi [4], developed under the European Project SOFIA. Hence, a modification is required in the file KPICore.java, function send() - we need to remove half-duplex-closing of TCP socket using API shutDownOutput(). Due to some issue with the JVM, the complete TCP socket was closing on invoking this API.
- ✓ The WN/PN devices uses Ångström Linux Distribution [5],and it has an issue with X11 initialization during execution of “sibd” process (even though X11 is not actually used). To fix this, we need to set DISPLAY environment variable to a value (exact “value” is not important, just setting of the env variable). Currently, we are doing this in Expect scripts.

7. References

1. Bolonio, J. A. J. “Adapting a DHT (Distributed Hash Table) to a Self-Reliant M2M (Machine-to-Machine) Network.” Master's thesis, Aalto School of Science and Technology, Finland, 2011
2. Gupta, N. “Management of Decentralized DHT Based M2M Network.” Master's thesis (Draft version), Aalto School of Science and Technology, Finland, 2011.
3. Li, D. “A Proxy for Distributed Hash Table based Machine-to-Machine Networks.” Master's thesis, Aalto School of Science and Technology, Finland, 2011.
4. "Smart-M3 Java KPI Library," Accessed 30 Jan 2012, http://smartm3-javakpi.sourceforge.net/
5. "Ångström Distribution," Accessed 30 Jan 2012, http://www.angstrom-distribution.org/
6. "Smart-M3," Accessed 30 Jan 2012, http://smart-m3.sourceforge.net/