M3: Machine-to-Machine Management Framework

Heikki Mahkonen[1], Tony Jokikyyny[1], Jaime Jimenéz[1] and Sławomir Kukliński[2]
[1]*Ericsson Research, Kirkkonummi, Finland*
[2]*Orange Labs, Warsaw, Poland*

Keywords: Machine-to-Machine, Internet of Things, Cognitive Network Management, Distributed Hash Tables, Publish Subscribe Networking.

Abstract: The number of deployed sensor devices with Internet connection is expected to exceed 50 billion units. Many of these devices spend most of their time in sleep mode to conserve energy. This sets new kinds of requirements for network management, and creates the need of redesigning conventional network management. Hence, most of the manual deployment, configuration and operation tasks need to be automated in a scalable fashion, using protocols that can deal with the uncertainty caused by the intermittent nature of the devices. For scalability reasons, the network management logic needs to be distributable in the network management architecture. In this document we describe our management framework for M2M networks. It is also shown, how the framework has been implemented as a prototype testbed. We have used the testbed to study centralized and de-centralized M2M network management logic for different management scenarios.

1 INTRODUCTION

In future networks the amount of users and M2M devices are growing. This sets new requirements for network management. Typical users do not have the expertise to deploy and configure M2M networks by themselves. Neither ISPs have the manpower to manually configure all the expected Machine-to-Machine (M2M) network deployments. For this reason, new automated and scalable ways of deploying and managing network equipment are required.

This work is related to a CELTIC project, called COgnitive network ManageMent under UNcErtainty (COMMUNE) (COMMUNE, 2013), which studies cognitive network management under uncertainty. The main goal of the project is to design a network management architecture that can distribute programmable network management algorithms to different parts of the network. In a typical case, these algorithms are cognitive in nature, allowing them to learn and adapt to changes in the environment where they are running. We have implemented a M2M management testbed as part of COMMUNE work. The purpose of the testbed is to study different network management algorithms in a real M2M networking environment.

In this paper, we describe the current state of our testbed and evaluate some of the implemented network management functionality. The paper is structured as follows. In Section 2 we give background on M2M networks, and current work on network management. Section 3 discusses our cognitive M2M network management framework. Section 4 describes the testbed implementation and experimented scenarios. Finally, Section 5 concludes the work.

2 BACKGROUND

2.1 M2M Networks

European Telecommunications Standards Institute (ETSI) describes a high level architecture (TS-102.690, 2011) for M2M networking and for Internet of Things (IoT) services and applications. In this architecture, a functional split can be made between the constrained M2M devices, a middleware layer with more logic and processing power, and an IoT service and the application layer.

M2M devices may use multiple different communications protocols e.g. 6LoWPAN (Montenegro, 2007), CoAP (Z. Shelby K. H., 2013).

The main differentiator between M2M device and any other device is that M2M device is a battery, memory, and CPU constrained device, serving a certain predefined purpose. Device management for such a device is required to enable automated configuration and management of the service.

To connect M2M devices to Internet services different kinds of M2M middleware services can be used e.g. CoAP, HTTP REST (Fielding, 2000), or CoRE Resource Directory (Z. Shelby S. K., 2013). Management functions for the M2M middleware need to deal with configuration of the Personal Area Network (PAN), device identification and lookup as well as providing semantic descriptions of devices.

Internets of Things (IoT) service platforms are handling service and application capabilities. Typically, it provides users a web service interface, through which users can view M2M device data and use M2M devices. For scalability reasons, IoT services are often implemented into a cloud platform, e.g. OpenStack (OpenStack, 2013).

2.2 Network Management

The Telecommunication Standardization Sector of the International Telecommunications Union (ITU-T) (M.3400, 2002) has defined a model and framework for network management, called FCAPS. This model has been widely used as a basis, when designing network management frameworks and protocols.

SNMP is an Internet Engineering Task Force (IETF) standard for managing devices on IP networks (Rose, 1991) (Case, 1990). The main problem of using FCAPS and SNMP in M2M, is the scalability when the volume of devices and gateways can reach up to 50 billion units.

2.3 Self-Organizing Networks

The problems with classical network management have already been identified about 10 years ago, generating the concept of autonomic or self-organized networks. We went through some of the most interesting work; however, as we noticed in these activities the problem of M2M and IoT management has been ignored.

The aim of EFIPSANS FP7 project (EFIPSANS, 2009) was to expose the features in IPv6 protocols that could be exploited or extended for the purposes of creating autonomic networks and services. The project implemented autonomic networks and services through a Generic Autonomic Networking Architecture (GANA). This approach seems to be too complex to use it for M2M management.

The 4WARD FP7 (Ghader, 2009) project has similar objectives as EFIPSANS, although it is not focused on IPv6. The approach is also less hierarchical. The idea of this In-Network Management (INM) system is to execute management functions on its own.

The Self-Organizing Networks (SON) solutions for cellular networks are currently being defined in the 3rd Generation Partnership Project (3GPP) standardization (TS-32.500, 2013). The problem with SON is that it is focused on plug-and-play deployment of new 3GPP radio base stations and therefore cannot be used in other networks.

2.4 Distributed Management

The nature of M2M calls for distributed network management. There are already several distributed management examples, such as (G. Goldszmidt, 1995) allowing the distribution of management, and (Waldbusser, 2006) extending the functionality of SNMP's MIB. More recent work provides a management framework for a distributed Machine-to-machine network, using Chord, (Y. Peng, 2012) (I. Stoica, 2001). Chord provides very efficient lookup with a key-based routing system (P. Pietzuch, 2007). A similar example is the SNMP Usage for RELOAD (Gupta, 2012) that uses RELOAD as lookup mechanism for SNMP.

One of the advantages of having decentralization in management logic is that it enables management while some devices are offline. Another advantage is that distribution can provide autonomous monitoring by delegating some of the monitoring tasks into the, often very powerful, monitored devices themselves. Finally, in scenarios with high churn a distributed approach would ensure the scalability of network.

The current management systems are not completely distributed. For example, in order to authenticate the devices and their operations on a distributed network, there is usually a central enrolment server serving as a trust anchor and Certification Authority (CA) for the whole overlay.

In this work we propose a new kind of network management framework for M2M and IoT service management that is autonomous and distributed. The autonomous features minimize the required human intervention. The distribution of the management logic and signalling enable the framework to be scalable even in most demanding network scenarios.

3 M3: FRAMEWORK

In this section we describe our M3 framework, based on the GARSON model (S. Kukliński, 2012). GARSON is a generic paradigm that allows for centralized and distributed management operations. It divides the network into smaller management domains. Moreover, it suggests a way in which the management functions should be decomposed, what provides profits related to reusability of components and more systematic and simpler network management programmability.

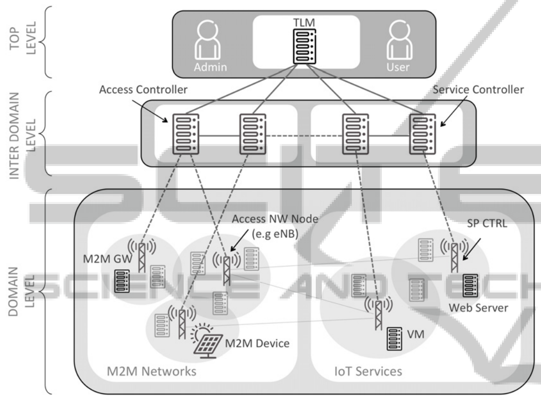


Figure 1: GARSON architecture for M2M and IoT Network Management.

Figure 1 shows how the hierarchy proposed by GARSON is applied to M2M management. We have defined two management domains for M2M networking and IoT services. Common Inter Domain Level (IDL) can handle bot domains. In the Top Level Management (TLM), the proxy server provides interfaces for system administrator and users to control the M2M management using policy updates.

3.1 Distribution in M3 Framework

The M3 framework allows both centralized and distributed operations simultaneously. Centralized network management is required in some use cases e.g. authentication, security, accounting, and bootstrapping of distributed features. Most of the network management should, however, be run in distributed and autonomous fashion.

In GARSON, distribution of the management decisions can be implemented in the intra-domain and inter-domain levels of the architecture. Distribution of management data and commands between these autonomous management logics can be done either with DHT (H. Balakrishnan, 2003) or publish/subscribe (P. Jokela, 2009) networking.

To support the programmability and functionality update, a suitable execution environment is needed that can handle distribution of execution modules. In our framework we used e.g. Java OSGi (OSGi, 2013). Due to this approach new management functions can be easily added or existing updated on fly.

3.2 GARSON System Model

Figure 2 shows the general system model for autonomic and cognitive management that we used in our work.

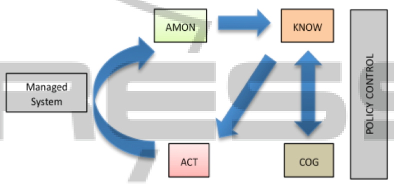


Figure 2: System model of autonomic and cognitive network management.

This model provides management system decomposition into internal layers to provide monitoring (AMON), knowledge based autonomic reasoning (KNOW), cognitive reasoning (COG) logic, and actuation (ACT) in different network elements. In addition, the functionality of these processes can be controlled through policies. Components of each layer can be implemented in different networks nodes and intra-layer communication mechanisms are provided in order to achieve the assumed goals:

- AMON: A programmable monitoring functionality, needed to produce network management input for management decision logic.
- KNOW: Knowledge base reasoning algorithms using input from AMON and producing decisions ACT.
- COG: Cognitive functions e.g. machine learning required to autonomously learn from previous actions.
- ACT: An actuation layer is required to format reasoning decisions into management commands.
- POLICIES: Administrators can control the management through policies.

3.3 Communication Models

Figure 3 a) shows a traditional manager-agent communication pattern used for centralized network management. In this model, agent is the managed entity and the manager implements the management function. To enable distribution of the management reasoning logic and functions communication without the need of direct connection a publish/subscribe overlay can be used. This option enables agent discovery via the overlay, still having the MIB within the manager entity Figure 3 b).

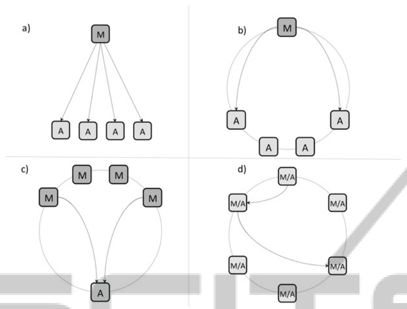


Figure 3: Some possible network topologies: (a) Centralized approach, (b) Multiple Distributed Agents, (c) Multiple Distributed Managers, (d) Fully decentralized.

In a similar way, multiple managers can manage one or more agents via a P2P overlay, without the need for establishing and maintaining direct connections between them, and having the MIB distributed only among the managers Figure 3 c).

Finally, to solve the fully distributed management scenario, an overlay of peers with the functionality of both managers and agents can be used, see Figure 3 d). In this scenario, peers will forward and route their messages via each other. An extra logic is required on the peers to establish the preference, when sending commands to each node.

4 M3: TESTBED

A testbed was created to enable experimentation of M2M network and IoT service management algorithms. Figure 4 shows overview of the testbed architecture.

The M2M devices in our testbed are Arduino (Arduino, 2013) devices with various kinds of sensors and actuators. Each device is connected to the M2M network wirelessly.

The M2M gateway is running a Java OSGi framework. The gateway handles M2M communication over Digi XBee (Digi, 2013). In addition, it connects to publish/subscribe and DHT management frameworks.

We implemented the DHT and publish/subscribe overlay using Hazelcast (Hazelcast, 2013) in the

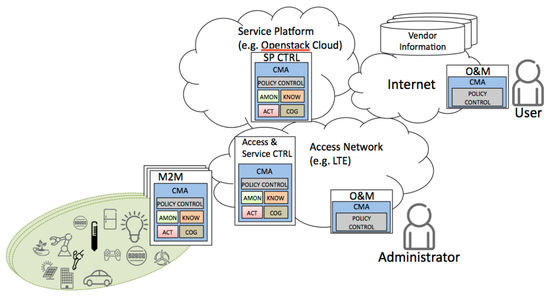


Figure 4: Overview of the cognitive network management testbed.

testbed. The Hazelcast API provides easy to use API for DHT implementations. In addition, it provides a Topic posting feature that was used to implement the publish/subscribe mechanism.

The IoT service platform in the testbed is an OpenStack cloud platform (OpenStack, 2013). In this environment, we have a capability to instantiate virtual machines in real time depending on our needs.

The access controller functionality is in charge of connecting M2M devices to management framework by configuring them with publish/subscribe information and bootstrapping DHT. Communication to it is secured with Generic Bootstrap Architecture (GBA) (TS-33.220, 2013).

4.1 Device Description Management

The device description management is needed to identify and configure M2M. We use the publish/subscribe channel to connect attaching M2M devices to different management entities. The management entities can be added and removed by subscribing and unsubscribing them. This architecture provides an easy way to introduce new management functionality without updating the software of a management server. Different users may have different requirements for the management of M2M devices, e.g. ISP, device vendors, users, etc.

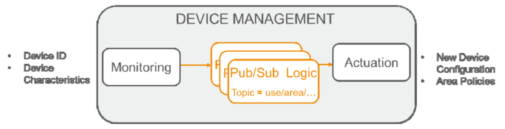


Figure 5: Device Description Management.

Figure 5 shows that the AMON functionality is distributed into M2M gateways that collect device information. The device identity that is a 64-bit serial number is published to a well-known management channel that connects the information to KNOW logic built into management entities. The management entity maps the device identity to a device description and publishes the information to the channel as ACT command. The M2M gateway that is responsible for managing the device will receive the device description and actuate the management commands based on it.

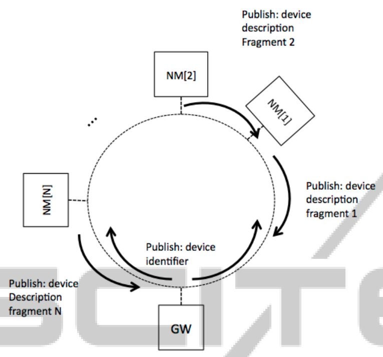

Publish: device description Fragment 2
NM[2]
NM[1]
...
NM[N]
Publish: device description fragment 1
Publish: device identifier
Publish: device Description fragment N
GW

Figure 6: Network management publish subscribe channel for device management.

As shown in Figure 6, the publish/subscribe interface supports subscription from multiple network management entities. These entities can have a single network management function that they are responsible for. If there are multiple management functions, the gateway gets a number of device description fragments as a reply to a device description request. The device description is then constructed by combining the fragments.

4.2 WPAN Coordination

We used the DHT implementation to cluster the management information from M2M gateways that reside in particular geographic area. This offers us a way to distribute some management tasks as the data required is available in the DHT e.g. WPAN coordination.

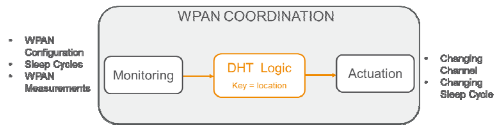
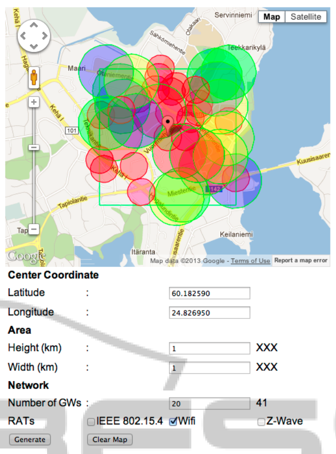


Figure 7: Wireless Private Area Network Management.

Figure 7 shows the monitoring information we can store into the DHT. The AMON layer is shared by each gateway through the DHT based on geographical location. Each gateway implements the KNOW layer and has a capability to calculate local optimum for their WPAN configuration as they know the network configurations around them. The ACT layer is implemented inside each gateway and acts according to the output from KNOW layer.



Center Coordinate

Latitude :	60.182590	
Longitude :	24.826950	

Area

Height (km) :	1	XXX
Width (km) :	1	XXX

Network

Number of GWs :	20	41

RATs ☐ IEEE 802.15.4 [✓ Wifi] ☐ Z-Wave

Generate Clear Map

Figure 8: M2M network emulation GUI.

In our testbed each gateway stores the WPAN configuration into the DHT with a key that has geographical significance. This way the stored information also maps to a location or area. By using the same key the gateways can collect all WPAN information in their vicinity. We have implemented a simple algorithm in all gateways to optimize WPAN channel usage based on the stored information. The algorithm constructs a coverage map based on the information in DHT and selects the channel based on the minimum overlap on other WPANs.

An emulation interface was used to evaluate our self-optimization functionality for WPANs. The interface depicted in Figure 8, shows our physical gateways and their coverage areas, as well as emulated gateways and their expected coverage areas. Channels are colour coordinated. The physical gateways are separated in the map by the marker. This setup allows us to study scalability issues in M2M management.

5 CONCLUSION AND FUTURE WORK

In this paper we described a new kind of network management framework, designed for M2M network and IoT service management. We also described our current implementation of the framework. The capability of our testbed was shown in two example scenarios. Two ways of distributing the monitoring and reasoning logic for network management were described.

We showed, that the management functions can be distributed into different nodes and interconnected using a publish/subscribe interface. This enables multiple separated network managers to correspond to a single signal that is published. One future work item is to look into how SNMP could use the publish/subscribe interface as its data transport mechanism.

To enable independently running distributed management algorithms, the input data needs to be distributed. We can achieve distribution by using DHT to store the monitored input data. Each managed node can run the reasoning logic locally and use monitoring information collected globally, or from locations near to the entity. The coordination of the distributed management logic, COG layer support and policy control is left for future work.

REFERENCES

Arduino. (2013). *http://www.arduino.cc.* Arduino.

Case, J. F. (1990). *Simple NetworkManagement Protocol (SNMP).* RFC 1157.

COMMUNE. (2013). *http://projects.celticinitiative.org/ commune.* CELTIC EU Project.

Digi. (2013). *http://www.digi.com/xbee.* Digi XBee.

EFIPSANS. (2009). *Third draft of autonomic behaviours specifications (abs) for selected diverse networking environments.* EFiPSANS EU Project.

Fielding, R. (2000). *Architectural styles and the design of network-based software architectures. http:// www.ics.uci.edu/fielding/pubs/dissertation/top.htm.*

G. Goldszmidt, Y. Y. (1995). *Distributed management by delegation, in proceedings of the 15th international conference on distributed computing, pp. 333-340.* IEEE.

Ghader, M. (2009). *Network management architecture in the future internet.* Spain: ICT MobileSummit.

Gupta, N. (2012). *Management of decentralized dht based m2m network.*

H. Balakrishnan, M. F. (2003). *Looking up data in p2p systems.* In Communications of the ACM.

Hazelcast. (2013). *http://www.hazelcast.com.* Hazelcast.

I. Stoica, R. M. (2001). *Chord: A scalable peer-to-peer lookup service for internet appli-cations.* in Proceedings of SIGCOMM'01 Conference.

M.3400. (2002). *M.3400 (02/200): Tmn management functions.* ITU-T.

Montenegro, G. K. (2007). *Transmission of IPv6 Packets over IEEE 802.15.4 Networks.* RFC 4944 (Proposed Standard). Updated by RFCs 6282, 6775.

OpenStack. (2013). *http://www.openstack.org.* OpenStack.

OSGi. (2013). *Open services gateway initiative. http://www.osgi.org.*

P. Jokela, A. Z. (2009). *Lipsin: Line-speed publish/subscribe inter-networking.* Barcelona, Spain: in proceedings of SIGCOMM '09.

P. Pietzuch, D. E. (2007). *Towards a common api for publish/subscribe.* Canada: in Proceedings of DEBS'07.

Rose, M. T. (1991). *The simple book: an introduction to management of TCP/IP-based internets.* Englewood Cliffs, NJ;: Prentice Hall, 2nd edition.

S. Kukliński, M. S. (2012). *Garson: Management performance aware approach to autonomic and cognitive networks.* IEEE MENS.

TS-102.690. (2011). *Machine-to-machine communications (m2m) functional architecture.* ETSI.

TS-32.500. (2013). *Telecommunication management self-organizing networks (son) concepts and requirements.* 3GPP.

TS-33.220. (2013). *Generic bootstrapping architecture (gba).* 3GPP SON standardization.

Waldbusser, S. (2006). *Remote Network Monitoring Management Information Base Version 2.* RFC 4502.

Y. Peng, W. H. (2012). *An snmp usage for reload.* IETF draft.

Z. Shelby, K. H. (2013). *Constrained application protocol (coap).* IETF draft.

Z. Shelby, S. K. (2013). *Core resource directory.* IETF draft.