25/11/2017

Internet of Things

Technical Requirement Document

Giulia Ferri Gianluca Giuffrida

INDEX

1 SCOPE

The goal of the project is built an IoT application able to create an intelligent video-surveillance system. The actual products present in the market are not very intelligent and often they are simple continuous video recorder. Instead, the project system provides some activating options that permit both save memory space and execute a better surveillance using smart turn on.

2 SYSTEM OVERVIEW

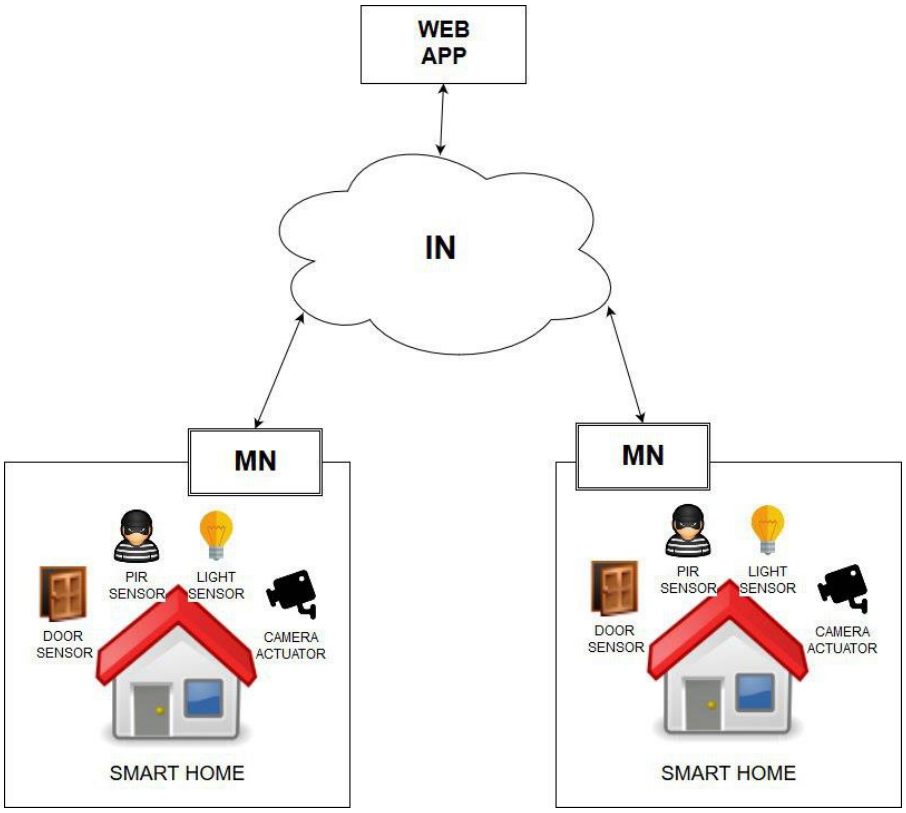
"*The Internet of Things is the interconnection of endpoints (devices and things) which can be uniquely addressed and identified with an IP (Internet Protocol) address. With the Internet of Things, devices can be connected to the Internet, sense, gather, receive and send data and communicate with each other and applications via IP technologies, platforms and connectivity solutions.*"

To help simplify, an IoT system is a group of interconnected smart objects that have the same goal. A smart object may be both sensor and actuator. But, smart objects systems alone are not sufficient for develop an entire IoT infrastructure. Due to the necessity of a big number of sensors and a fast increase of wireless devices, IoT's system developers have choose the use of a low-cost and easy to use standard as the more common IEEE 802.15.4 (Link-Layer). This standard is less power expensive than a typical wireless communication and permits an optimal solution for the limited processing capability devices, but create some constraints for the upper layer (Network-Layer). To use the existing IPv6 Internet layer an adaptation layer named 6LoWPAN is mostly used thanks to its header compression mechanism. Furthermore, some common services layer and Machine-to-Machine interoperability platform are being implemented to help delete the vertical fragmentation and create a common horizontal solution made for heterogeneous devices.

So, a possible formula for a good IoT infrastructure is:

- Horizontal service platform oneM2M for interoperability
- A good message organization for intercommunication
- CoAP application because it is light and LLN oriented
- UDP transportation for avoid multiple re-transmission
- IPv6 as internet layer that is the new standard
- 6LoWPAN for adapt the IPv6 layer to more pour Link-Layer
- IEEE 802.15.4 (Link and physics) for its simplicity and low-cost hardware

These are the principles on which the project is lied.



The system has three different sensors and an actuator, based on standard IEEE 802.15.4, simulated by COOJA tool. To take data from sensors network, a java framework, knows as Californium with CoAP support, is used. Here, is developed the main part of the application, where the logical part and data storage infrastructure are implemented. The data storage is represented by oneM2M Middle Node. Additionally, an Infrastructure Node that acting as cloud, communicate with the Middle Node, so as to provide an external access to the data through the Web Site.

3 SENSORS NETWORK

In this project, we have been designed two separate COOJA network for simulate two different houses. The two models stand on different machine for better represent the two houses, but have the same kind of sensors. Each sensor must be implemented as a COOJA's mote, using the C language. The code must have at least one resource declared that permit to receive and responds at the request from the extern. This resource uses a library named rest-engine where the REST Engine includes a comprehensive embedded CoAP implementation.

For simulate a real network, the coverage ray used for the sensor and actuator is set to 10m. This is a fundamental information for simulate multi-hop. In this manner is possible cover long distance adding other motes inside the network. For this kind of network based on RPL (Routing Protocol for Low-power and lossy network) the only possibility is use Route-Over protocol (Network Layer).

3.1 LIGHT

This sensor simulates a photo-resistance providing the measurement light value. It exposes two resources, the first is a periodic resource called ***light*** and the second, that is standard, called ***setting***.

The ***light*** resource permit to simulate the virtual light value during a day. It implements GET and POST handler. GET is used to obtain the light current value, instead POST modify the current value for improve the simulation's efficiency. Furthermore, the resource has the observed flag set for permit the CoAP observing. The periodicity of this resource is 50*CLOCK_SECOND after which the ***periodic_handler_value*** function is called (increase the virtual light value).

The ***setting*** resource permit to set the room where the sensor is positioned. Even in this case two handler are provided.

GET furnish the room number already saved and POST insert or modify the actual room. In this case the observing flag is not set, also because for a standard resource the observing method has no sense.

```
RESOURCE(setting, "title=\"setting\";type=\"LIGHT\";rt=\"room_number\"", get_handler, post_handler, NULL, NULL);
PERIODIC_RESOURCE(light,"title=\"light\";obs;rt=\"light_value\";if=\"1\"",get_periodic_handler,post_periodic_handler,
                  NULL,NULL,50*CLOCK_SECOND,periodic_handler_value);
```

3.2 DOOR AND WINDOW

This type of sensor is likely a pressure sensor, when doors or windows do not touch the other side of the sensors they start to signal the movement. The simulation act in the same way e.g. when the doorwin value is 'on' the sensor sends information otherwise it sends just a message of switching off.

This mote exposes two resources, ***doorwin*** and ***setting*** respectively periodic and standard resources. The ***setting*** resource is the same resource used upon, that permit to setting and retrieve the room number. The ***doorwin*** periodic resource send the value in periodic way only when the sensor is activated. Instead, when the sensor is off only one notification message is sent. Even in this case this is an observable resource.

```
RESOURCE(setting, "title=\"setting\";type=\"DOORWIN\";rt=\"room_number\"", get_handler, post_handler, NULL, NULL);
PERIODIC_RESOURCE(doorwin,"title=\"doorwin\";obs;rt=\"doorwin_value\";if=\"2\"",get_periodic_handler,post_periodic_handler,
                  NULL,NULL,50*CLOCK_SECOND,event_post_handler);
```

3.3 PIR

This sensor is the simulation of a normal PIR sensor. When the sensor does not detect the presence of someone send continuously notify message, contrary if the PIR detect movement send a message and wait until the presence is disappear. The simulation reproduces the same behaviour sending message only when the sensor is off.

Two are the resources exposes, ***PIR*** and ***setting***. ***Setting*** works exactly how described before. ***PIR*** is the periodic resource, that offer the possibility to observe it. As already say it send message in periodic way only when the sensor is inactive, however it sends only one notification message when the sensor passes in an active state.

```
RESOURCE(setting, "title=\"setting\";type=\"PIR\";rt=\"room_number\"", get_handler, post_handler, NULL, NULL);
PERIODIC_RESOURCE(PIR,"title=\"PIR\";obs;rt=\"PIR_value\";if=\"3\"",get_periodic_handler,post_periodic_handler,
                  NULL,NULL,50*CLOCK_SECOND,event_post_handler);
```

3.4 CAMERA

Camera represent the simulation of actuator for the system. Rather then the other sensors, the actuator do not execute any kind of work. Simulate real video-camera is very difficult, so the simulation is just a switch enables by the others sensor through the application.
For this mote, as in the other sensors two resources are be declared. In this case, neither of those is a periodic resource, since there is not necessity of periodic control. One, as in the other sensors, is a resource called ***setting*** for the room number management and the other, named ***camera*** that permit to change and get camera value.

```
RESOURCE(setting, "title=\"setting\";type=\"CAMERA\";rt=\"room_number\"", get_handler, post_handler, NULL, NULL);
RESOURCE(camera,"title=\"camera\";rt=\"camera_value\";if=\"0\"",get_camera_handler,post_camera_handler,NULL,NULL);
```

3.5 BORDER ROUTER

This mote is a basilar component in an 6LoWPAN network, otherwise all the packet remains inside a link. Without it the topology is Ad-Hoc 6LoWPAN that is not right for the project scope. Basically, this mote takes all the packets direct to an external route, translate it in an IPv6 standard packet and send it to the originally destination.

4 CALIFORNIUM AND ONEM2M

Californium (Cf) is an open source implementation of the Constrained Application Protocol (CoAP) and it is used for obtaining the value of the sensors from the COOJA tool and communicate with oneM2M.
The aim of COOJA tool, is developing virtual motes that using the Contiki operating system with the entire IPv6 and 6LoWPAN support. It is implemented in C and uses a make build environment for cross-compilation on most platforms. A wide variety of microcontroller and device platform ports exist, along with examples and reusable applications.
For the application layer, as already say, a good choice is CoAP. Using some part of the web architecture Representational State Transfer (REST), CoAP permit to Discovery, Subscribe, and perform all the HTTP request (GET, POST, PUT, DELETE) on resources.

4.1 JAVA PROJECT

Here are described the general functionality of the developed project with it main classes.

The project is written in java and based on Californium framework. It is divided in two main java packages formed by Common Services Entity (CSE) node of oneM2M, Infrastructure Node (IN) and Middle Node (MN).

4.1.1 ManagerOneM2M

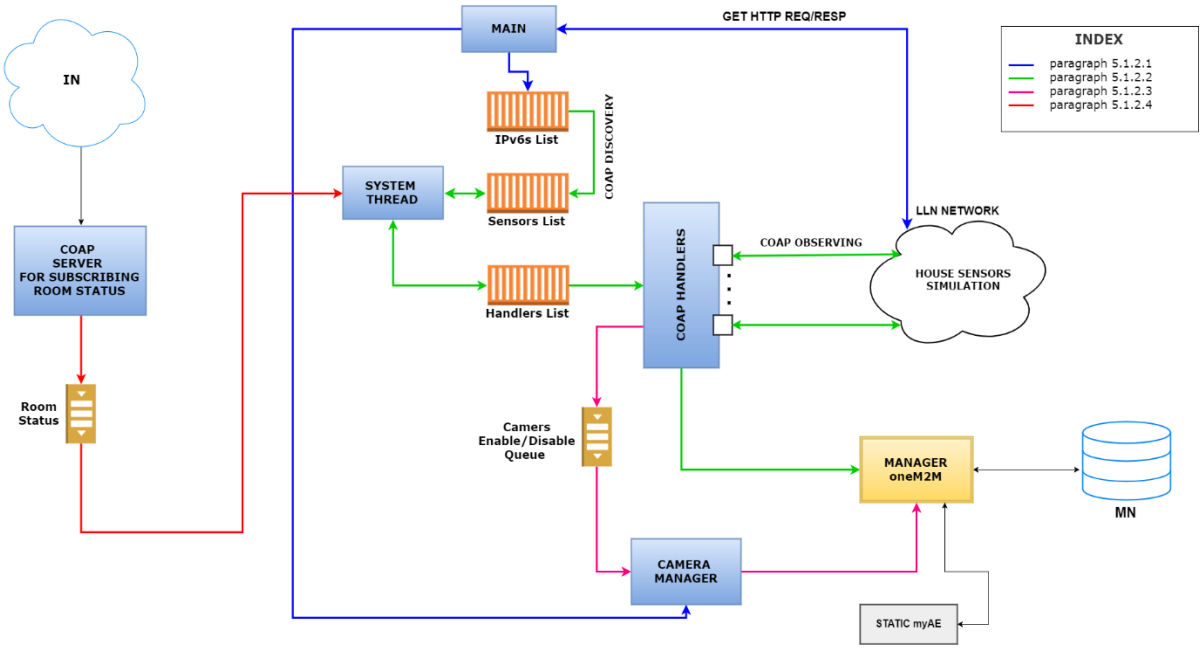
The IN node is considered as a cloud i.e. where the external entity (Web Site, mobile devices) may take the data. The MN, instead, is linked to the IN and can exchange data with it. In the MN there are all the data collected from the sensors and are saved in local (in our case the House). The most important entity in a oneM2M application are Application Entity (AE), Common Service Entity (CSE), and Network Service Entity (NSE).
AE contain the logical of the solution of oneM2M. CSE contains a set of Common Services Function (CSF) that are functions used for offering a service e.g. subscription. NSE are the network services in which the link with low network layer is made.

For both java package there is a class called ManagerOneM2M. This class contains all method regarding the communication between java and oneM2M. Other to the usually functions creation AE, Container (CNT) and Content Instance (CI), the get onem2m state is one that permit to avoid the duplication of Container and AE. Another functionality is the subscribe function that is like add a new CI when the request is performed, but permit to establish an asynchronous communication with some CNT, or AE. In fact, every time that a value is added, removed or updated a notification is rised to the listener subscribe server.
The delete function permit to erase a CI or a subscription, but in this project, it is never used.
All the CoAP messages that are sent to oneM2M use JSON encoding and are processed with the JSON parser of java.

4.1.2 MN

MN contains the main application that is ran inside the "local "space, or in this case, in every house that execute the systems. The most important functions are:

1. Acquire IPv6 addresses from COOJA motes
2. Get all the resources declared for every motes and for each if they are observable, a new coap_handler is declared for obtain relative values
3. A subscription thread is activated for retrieve which rooms are be enabled by the user
4. Build oneM2M_structure for the management and communication of the oneM2M platform using manager_oneM2M described upon
5. The camera manager executes all the starting and stopping for all the camera



4.1.2.1 GET SENSOR IP

First of all, in Main Class, a GET HTTP request is sent to Border Router for retrieve all the IPv6 addresses present inside COOJA simulation. The default border router of COOJA offer a Web server in which information about all motes of the network are displayed, as is possible see in figure below. From this page the addresses are took and inserted inside the IPv6s' list.

Neighbors

```
fe80::212:7407:7:707
fe80::212:7408:8:808
fe80::212:7406:6:606
```

Routes

```
aaaa::212:740a:a:a0a/128 (via fe80::212:7407:7:707) 16711369s
aaaa::212:7405:5:505/128 (via fe80::212:7407:7:707) 16711344s
aaaa::212:740c:c:c0c/128 (via fe80::212:7408:8:808) 16711306s
aaaa::212:7404:4:404/128 (via fe80::212:7406:6:606) 16711187s
aaaa::212:7402:2:202/128 (via fe80::212:7406:6:606) 16711165s
aaaa::212:7407:7:707/128 (via fe80::212:7407:7:707) 16711086s
aaaa::212:7408:8:808/128 (via fe80::212:7408:8:808) 16711084s
aaaa::212:7406:6:606/128 (via fe80::212:7406:6:606) 16711082s
aaaa::212:7403:3:303/128 (via fe80::212:7407:7:707) 16710953s
aaaa::212:740e:e:e0e/128 (via fe80::212:7408:8:808) 16710905s
aaaa::212:7409:9:909/128 (via fe80::212:7408:8:808) 16710760s
aaaa::212:740d:d:d0d/128 (via fe80::212:7408:8:808) 16710471s
aaaa::212:740b:b:b0b/128 (via fe80::212:7408:8:808) 16709559s
```

Main thread, thus, create the Camera Manager that handle the actuators.

4.1.2.2 SYSTEM THREAD e COAP RESOURCE

COAP provide a resources discovery facility through a GET request to resource well-known/core.

For obtain all resources from the motes, this request is sent individually to all IPs in the list. Using response, a new Sensor Object is instantiated were inside there are all information about sensor and his resources, with attention for link attribute such as 'if', 'obs', etc. A flag discriminates if it is an actuator.

Now the system known all sensors/actuators involved. Using another facility called Observing is possible retrieve, from all of them that have the observing flag set, periodically notification when data are updated. For exploit this functionality and improve it, three new class are being developed. Each one extended Thread Class for execute computing and implement Coap Handler for the notifications.

The class are:

- LightResource
- PirResources
- DoorResource

In all resources class there is a common LinkedBlockingQueue, created in the main and shared with actuators, used for notifying it of activation/deactivation of each sensor. Furthermore, for avoid data losing, a LinkedBlockingQueue is being implemented in each class with the scope of pausing the thread when there are no new notifications.

Thanks to ManagerOm2m, all instances of this three classes create an AE referred to the belonging room (if it not exists) and inside it a Container relative of its sensor type. Also, every time a new notification is received a CI is written inside Container of the corresponding sensor.

The whole of resources Thread are handled by System Thread that is responsible for lock and unlock the notifications arrives for a certain room. This is given by a chose of the user, though web page that send an asynchronous notification to MN (Subscribed to IN).

4.1.2.3 SUBSCRIPTION THREAD

The Subscription Thread is the entity that receive all the messages relatives to the managed house from the web site. Since the user may communicate only with web site through IN, a subscription is made for obtain the notification in MN without make polling on IN.

4.1.2.4 CAMERA MANAGER

This project foresees a multi sensor thread to one actuator thread (analogous N-to-One). This choose was due to the fact that there are many more sensors than actuators.

In fact, the system holds a shared LinkedBlockingQueue that exchange MessageObject in which are described the sensor that throws the message, the room, and the value.

When a message is delivered, the logical of the project is executed, in order to activate the right camera for the right time.

The logic covers the follow cases:

- When the Ligth sensor switches on (decide by threshold) camera switch on and if:

  - Door sensor was active a 60 second timer start;
  - Door sensor was disable or is not present in the same room a 10 second timer start;

  if during this period also Pir sensor is activated a camera stay on otherwise at the end of timer camera go back to being off.

- When the Door sensor switches on (decide by threshold) camera switch on and if:
  - Light sensor was active a 60 second timer start;
  - Light sensor was disable or is not present in the same room a 10 second timer start;
  - a smart procedure is performed by reading a file with JSON encoding in which are indicated the adjacent room, and also in this room camera is switched on.

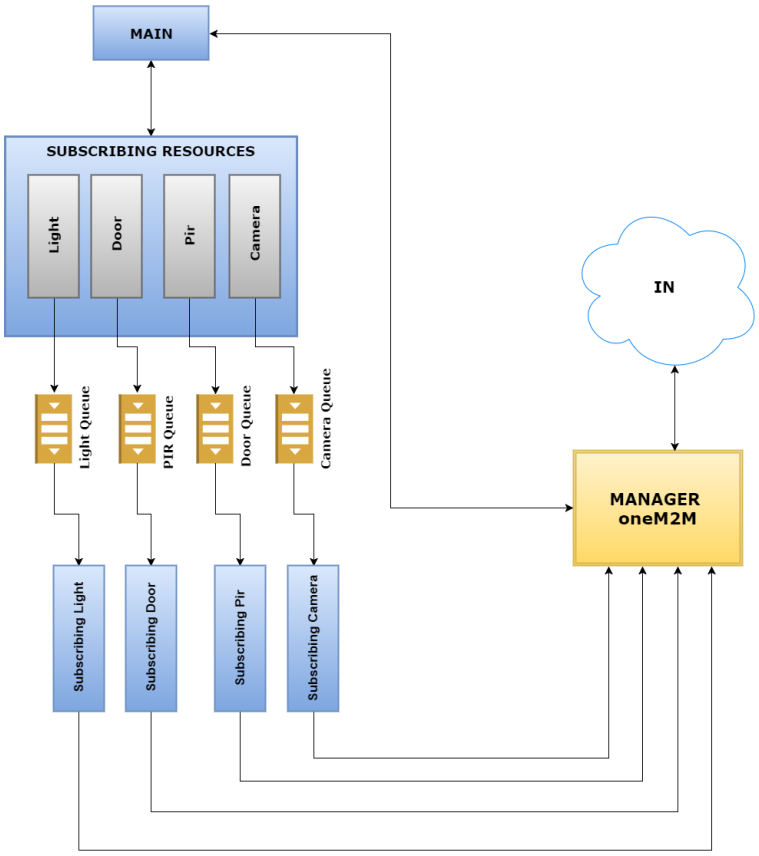
  if during this period also Pir sensor is activated a camera stay on otherwise at the end of timer camera go back to being off.

- When Pir sensor switches on, camera switch on in turn.

If a camera in a specific room is turned on a CI is write in the camera CNT relative to the room (AE).

4.1.3 IN

The IN represents the union of the all data inside the MN. It structure is made up by all the MNs connect to it as AE. Inside each AE there are the rooms, and the sensors for each. Furthermore, there is an additional container where is indicated the room status. The IN logical part is simpler than MN. It is composed principally by a COAP Monitor, used for retrieves the asynchronous messages written on the MNs. When a data is received it is put in a dedicated LinkedBlockingQueue for process it.



4.1.3.1 COAP MONITOR

A CoapServer is a server that is used for retrieve the asynchronous informations coming from MN. How is knows, a server needs a port where listen to request and an address. So, for this case the loopback address is chose and the port number in this case is 5687. This function offers also another functionality, the possibility to add Resource. The project foresees four SubscribingResources based on the typology of sensor that they are listening to.

- Light
- Door
- Pir
- Camera

Every resource runs a correspondent kind of thread, where some decisions and information adapting are made. At the end of each information processing the result is written on oneM2M through the managerOneM2M, ready for external uses. All threads have their personal queue where the dedicated SubscribingResources write the information.

4.1.4 Overview

Obviously, the entire system may change during the time, so each section, both IN and MN, perform periodically scan on the network for find and automatically include new sensors in the already existent set. The checks are executed every 90 second each in IN and MN.

Another problem for the system is how describe the interconnection of the room's house in way to realize a smart system?

The solution is use a configuration file where are described the adjacent rooms. This file is encoded in JSON and is reading by using the java parser for JSON.

```
{
  "home": "house2",
  "doors": [
    {
      "ip": "aaaa::212:7405:5:505",
      "neighbors": "room1"
    },
    {
      "ip": "aaaa::212:7406:6:606",
      "neighbors": "room1"
    }
  ]
}
```

4.1.5 SETTING

Setting Class is composed by a single thread that run once. Within setting class there is a JSON file where a mapping IP sensor-room is stored. This configuration file is read and the right room number are assigned for all sensors inside their setting resource. It should be executed every time that the COOJA system is restarted or launched for the first time.

5 WEB SITE

The web site is the user interface of our system. For each house there is a dedicated page that is dynamically loaded using the information taking from IN.

Through house's page is possible set in which room the security system is enable or not. Furthermore, is possible disable the entire system.

For each room are showed the values of all sensors and the actuators that are in it. Inside a room, a camera animation starts not only when a room sensor turned on it but also when is caused by another room (smart activation). For the last in the information text of relative camera there is "SMART ON" text.

All the showed information talked about above is obtained through PHP file that build an GET HTML request within the right header for communicate with oneM2M. The choose of use PHP with cURL support is to avoid cross-origin problem.

There is a similar PHP file in which a POST HTML request to oneM2M is executed. It is used when the room status is changed in order to inform oneM2m.

Another case is when a sensor POST is done. This function is just for simulation purpose and is the only that need the support of HTML-to-COAP proxy. Thanks to it is possible change values of the all sensors, by buttons in the web page, for simulate intrusion case in the house.

6 CONCLUSION

The possibility offered by this project are:

- ✓ Adding new house in every moment as MN (clearly a new house web page is not automatically created);
- ✓ Adding new sensor in every moment even if two sensors of same type are seen by the web site like one;
- ✓ MN service interruption are supported from IN (web page stays in the state where is left);
- ✓ Reduced traffic overhead for oneM2M through maintain a local structure that reflect the three structures of it.

Obviously, there are also limitations. The most important are:

- o It's no possible adding directly a new room in MN without modify configuration file.
- o There is no security access control but is possible added it with oneM2M ACP.