An architecture for the integration of Wireless Actuation Capabilities with IP Multimedia Subsystem

Ru Cheng Hou

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science at
Concordia University
Montréal, Québec, Canada

December 2010

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Ru Cheng Hou

Entitled: An architecture for the Integration of Wireless Actuation Capabilities with IP Multimedia Subsystem

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical and Computer Engineering)

Complies with the regulations of the University and meets the accepted standard with respect to originality and quality.

Signed by the final examining committee:

Dr. Dongyu Qiu Chair

Dr. Anjali Agarwal Examiner

Dr. Nizar Bouguila Examiner

Dr. Ferhat Khendek Co-Supervisor

Dr. Mustafa K. Mehmet Ali Co-Supervisor

Approved by _____

Dr. W. Lynch
Chair, Department of Electrical and Computer Engineering

_____20___ _____

Dr. R. Drew
Dean, Faculty of Engineering and Computer Science

ABSTRACT

An architecture for the integration of Wireless Actuation Capabilities with

IP Multimedia Subsystem

Ru Cheng Hou

The IP Multimedia Subsystem (IMS) is an architecture that aims at seamlessly delivering multimedia services. It enables IP multimedia services for end-user using standard Internet based protocols such as Session Initiation Protocol (SIP). Examples of multimedia services include presence, instant messaging, enhanced voice and video, pervasive gaming and emergency services.

Wireless actuators are small scale devices that can receive/accept instructions and act on their surrounding environment. They are broadly used in automation industry and intelligent control systems. With the rapid development of Internet and mobile telecommunication technologies, more and more actuators are being deployed in applications such as environment monitoring, home automation and health care to improve human beings' living conditions.

Combining actuators' actuation capabilities with IMS will certainly enable novel value added services. However, the actuator networks are application specific and provide proprietary interfaces to the external world. Integrating wireless Actuator Networks (AN) with IMS to enable actuation service to IMS end users through standard protocols and interfaces is the objective of this thesis.

There are several challenges related to this integration: First, there is no ready-to-use

architecture for the integration. New functional entities and suitable protocols for actuation triggering are needed. Second, there are no actuators in the market with open interfaces to the external world, we need to find alternative solutions for the realization of the integrated architecture. Third, there is no information model for abstracting actuation command semantics and this has to be defined.

In this thesis we derive a set of requirements for the integration of AN actuation capabilities with IMS, we review and evaluate related work, and then propose a novel architecture. This architecture includes two new functional entities for IMS: The Actuation Control Function (ACF) and the Wireless Actuator Gateway (WAG). The ACF handles high level actuation requests from other applications. It acts as an intermediate component and hides the low level actuation commands from the applications. The WAG transforms high level actuation commands to low level, proprietary and actual actuation commands that can be understood and executed by actuators.

A detailed survey and evaluation of existing protocols for actuation command carrying is also provided. We define an actuation command information model to abstract the actuation triggering instructions. We implement the key components of the proposed architecture. A proof of concept prototype has been implemented using simulated robots equipped with actuators. The average end-to-end actuation delay of our architecture is evaluated through experiments with the prototype.

ACKNOWLEDGEMENTS

Going back to school to continue studying after more than 10 years of working is not an easy thing. Without the help and support from my supervisors, professors and team members, I could not get this research done.

First, I would like to thank my supervisors Dr. Ferhat Khendek and Dr. Mustafa Mehmet Ali for their ideas, great patience and tolerance, valuable discussions and guidance. They are always supportive and they have helped me out in every step of my research. Words are inadequate to express my thanks. Special thanks to Professor Dr. Roch Glitho. Although he is not my formal thesis supervisor, he has been very involved in this project. Without his constructive ideas, I would not have finished my research. His enthusiasm in research truly inspired me and his warm encouragements helped me through the tough time of my studies. I would like to thank Dr. Fatna Belqasmi who helped me a lot during the last stage of my research and gave me clear guidance on implementation and saved me a lot of time.

I would like to thank my teammates in TSE lab: Majid, Saba, for their ideas, comments and sharing of experiences with me.

The project I have been involved with is partially supported by Natural Sciences and Engineering Research Council of Canada (NSERC) and Ericsson Canada. I acknowledge this financial support.

Finally, thanks to my family, my wife Wang Wei. Without her support I could not focus on the research and cannot go through this adventure.

Table of Content

List of Figures

List of Tables

Acronyms and Abbreviations

3GPP	3rd Generation Partnership Project
A/D	Analogue/Digital
ACF	Actuation Control Function
ACML	Actuation Command Markup Language
AN	Actuator Networks
API	Application Programming Interface
AS	Application Server
B2BUA	Back to Back User Agent
B3G	Beyond 3G
BGCF	Breakout Gateway Control Function
CS	Circuit Switch
CSCF	Call Session Control Function
DSL	Digital Subscriber Line
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications
HSS	Home Subscriber Server
HTTP	Hyper Text Transfer Protocol
I-CSCF	Interrogating CSCF
IETF	Internet Engineering Task Force
iFC	Initial Filter Criteria
IMS	IP Multimedia Subsystem
IP	Internet Protocol
MCCF	Media Control Channel Framework
MGCF	Media Gateway Controller Function
MGW	Media Gateway
MRF	Media Resource Function
MRFC	Media Resource Function Controller
MRFP	Media Resource Function Processor
MSML	Media Server Markup Language

P-CSCF	Proxy CSCF
PIDF	Presence Information Data Format
PLMN	Public Land Mobile Network
PSTN	Public Switch Telephone Network
QoS	Quality of Service
RFC	Request for Comments
RTP	Real-time Transport Protocol
S-CSCF	Serving CSCF
SCTP	Stream Control Transmission Protocol
SDP	Session Description Protocol
SDS	Service Development Studio
SGW	Signaling Gateway
SIP	Session Initiation Protocol
SLF	Subscription Locator Function
SMTP	Simple Mail Transfer Protocol
SPT	service points triggers
TCP	Transport Control Protocol
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
WAG	Wireless Actuator Gateway
WCDMA	Wideband Code Division Multiple Access
WLAN	Wireless Local Area Network
WSAN	Wireless Sensor and Actuator Network
WSN	Wireless Sensor Network
XML	Extensible Mark-up Language

Chapter 1

Introduction

1.1 Research domain

The Third Generation Partnership Project (3GPP) IP Multimedia Subsystem (IMS) [1] is the cornerstone of Third Generation (3G) networks. It is an overlay layer on top of IP based networks and aims at seamless provisioning of multimedia services in an access agnostic way. It is originally designed for the convergence of Internet and mobile networks, later on developed as a service development and delivery platform to support multimedia service generation. The IMS framework specified by 3GPP makes IP multimedia services accessible to mobile end-user using standard Internet based protocols e.g. Session Initiation Protocol (SIP) [2]. Examples of multimedia services include presence, instant messaging, enhanced voice and video, pervasive gaming and emergency services.

An actuator [3] is a mechanical device for controlling and acting on systems. It takes energy, usually transported by air, electric current, or liquid, and converts that into some kind of motion. In mechanical engineering, actuators are usually used for motion, or to clamp an object to prevent motion. In electronics, actuators are seen as a kind of transducers. Actuators are broadly used in automation industry and intelligent control systems. In recent years, with the rapid development of Internet and wireless telecommunication technologies, actuators have been deployed for applications such as home automation and environment monitoring to improve human beings' living

conditions.

Actuators are usually used in combination with sensors. Wireless Sensor and Actuator Networks (WSAN) [4] consist of sensors that can sense the environment and actuators that can act on it in reaction to the sensed events. There exist two categories of WSAN architectures [4]: automated and semi-automated. In the former case, sensors detect events, transmit readings to actuators that process all incoming data and initiate appropriate actions. In the second architecture, sensors send all the sensed data to a central controller that processes the data and issues instructions to actuators.

1.2 Motivations and problem statement

As aforementioned, wireless actuators are small scale devices that can receive instructions and act on their surrounding environment. These capabilities, when integrated to IMS, can enable novel services such as smart home or healthcare applications.

An Actuator Network (AN) [5] is application specific and provides proprietary interfaces to the external world. Our goal is to integrate ANs with IMS such that IMS entities interact with the AN through standard protocols and interfaces. There has been some research work done on the integration of sensor networks and IMS [6] or even on integrating WSAN with Internet [7]. However, none of them have addressed the integration of actuators with IMS.

There are several challenges related to this integration. First, there is no ready-to-use architecture for the integration. The functional entities and protocols for

actuation triggering need to be defined. Second, there are no actuators on the market that have open interfaces to the external, we have to seek for alternative solutions. Third, there is no information model for abstracting actuation command semantics. We need to define this model.

1.3 Contribution of the thesis

The contributions of the thesis are as follows:

- We define application scenarios for the integration of AN and IMS. They are typical scenarios in application domains such as emergency management, health care and home automation. These scenarios are used to generate the requirements for the integrated architecture.
- We derive a set of requirements with respect to the application scenarios. The requirements include those related to the actuation information model and actuation control protocol. The actuation conflict handling and method of actuation triggering are also considered. We evaluate related work according to these requirements.
- We design an architecture for the integration of ANs with IMS with respect to the application scenarios. In the proposed architecture, we define two new functional entities: one is responsible for abstracting away low level actuation information and the other is a gateway which enables communications between the IMS and the heterogeneous actuators. We define a set of criteria for actuation control protocol selection and evaluate a few existing protocols

with built-in command carrying semantics. We define the actuation command information model to abstract the actuation triggering instructions.

- We implement a proof of concept prototype based on the environment monitoring application scenario. This prototype demonstrates how to build new value added services over the integrated architecture. The key components of the proposed architecture have been implemented. The session control and signaling protocols for exchanging actuation commands and transmitting media stream have also been implemented. The end-to-end actuation delay is evaluated through experiments with the prototype.

1.4 Organization of the Thesis

The rest of the thesis is organized as follows:

Chapter 2 provides the necessary background information. It provides an introduction to actuators, actuator networks, wireless sensor and actuator networks including architectures, hardware and applications. The IMS architecture, key functional entities, operations and protocols are also presented before a brief introduction to the SIP protocol.

Chapter 3 provides a detailed review of related research work on WSAN integration and interworking with other networks. We start with application scenarios and derive the requirements for the evaluation of related work before the actual evaluation. After the evaluation, we conclude that none of the existing architectures meet all the requirements and there is a need for a new architecture.

Chapter 4 depicts the proposed architecture for the integration of ANs with IMS and its principles. The functional entities and interfaces are described. The design of the key components is also presented.

Chapter 5 refines further the Actuation Control Protocol of the proposed architecture. First, the criteria for actuation control protocol selection are set. This is followed by the discussion of existing command semantic carrying protocols, their functions and their principles to deliver commands. These protocols have been evaluated with respect to the selection criteria. Based on this evaluation, Internet Engineering Task Force (IETF) Media Control Channel Framework (MCCF) [8] is chosen as the actuation control protocol. An XML [9] format information model is designed to abstract the actuation commands.

In Chapter 6 we present the implementation of the key components of the proposed architecture. The proof of concept of the integrated AN-IMS architecture through the implementation of an environment monitoring prototype is also provided with some preliminary evaluation of the response time.

Finally, Chapter 7 draws conclusions and discusses potential future work.

Chapter 2

Background Information on Actuators, IP Multimedia Subsystem and Session Initiation Protocol

In this chapter, we provide the relevant background information that is useful to understand the content of this thesis. Three areas of information are presented. First, we introduce the definition, classification and technical issues related to actuators and WSAN [4]. Next, the IMS concepts, architecture and operations are provided. Finally, we introduce the SIP, including its entity definitions and protocol messages [2].

2.1 Actuators and Wireless Sensor Actuator Networks

2.1.1 Actuators

An actuator is a mechanical device for moving or controlling a system. It takes energy, usually transported by air, electric current, or liquid, and converts that into some kind of motion [3]. Actuator is not a brand new concept; in fact, they have been widely used in digital control systems and automation industry. In engineering, actuators are often used to introduce motion, or to clamp an object to prevent motion. In electronics, actuators are a subdivision of transducers. They are devices which transform an input signal (mainly an electrical signal) into motion. Specific examples include: electrical motors, pneumatic actuators, hydraulic actuators, linear actuators, comb drive, piezoelectric actuators, thermal bimorphs, micro-mirror devices and electro-active polymers [3].

In [4], actor is differentiated from actuator and is defined as ‘besides being able to act on the environment by means of one or several actuators, actor is also a network entity that performs networking-related functionalities, i.e., receive, transmit, process, and relay data’. In this thesis, our focus is on actuation actions, we will not differentiate between these two concepts. Therefore, we use more general definition – actuator instead of actor.

2.1.2 Wireless Sensor and Actuator Network

WSAN consist of sensors that can sense the environment and actuators that can act on it in reaction to the sensed events. In such a network, sensors gather information about the physical world, while actuators take decisions and then perform appropriate actions upon the environment, which allows remote, automated interaction with the environment.

There exist two categories of WSAN architectures [4]: automated and semi-automated. In the former case, sensors detect events, transmit readings to actuators that process all incoming data and initiate appropriate actions. In the second architecture (as shown in Figure 2.1), sensors send all the sensed data to a controller via a sink. The central controller processes the data and issues instructions to actuators. The research in this thesis is based on the second case.

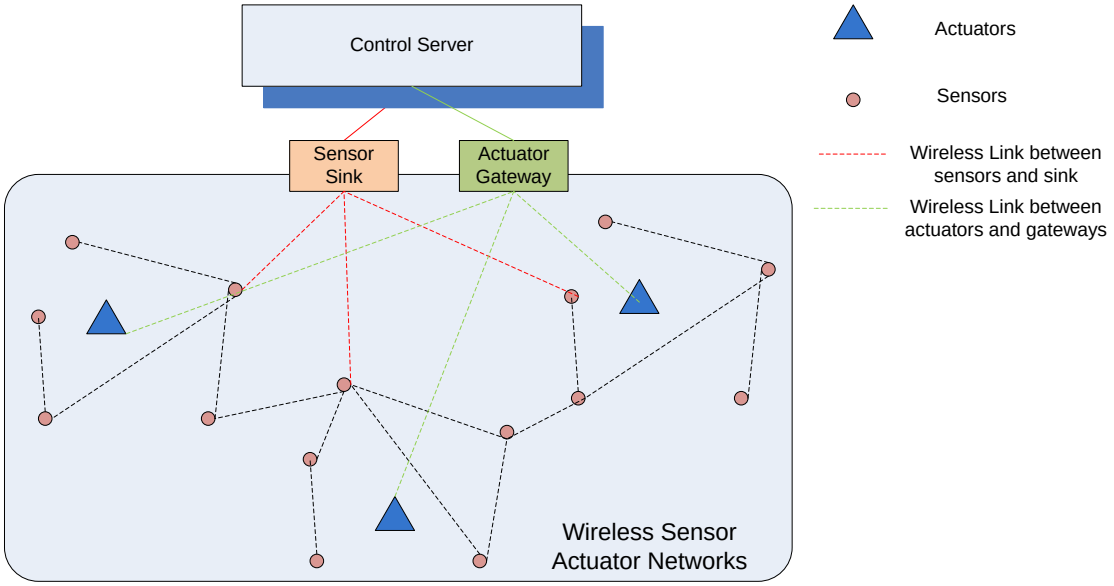


Figure 2.1 Semi-automated WSAN architecture

2.1.3 Wireless Sensor and Actuator Networks vs. Wireless Sensor Networks

WSAN is like Wireless Sensor Network (WSN) but include actuators that bring in new functionalities and issues.

- Sensors are small, cheap devices with limited computation and communication capabilities, actuators are usually resource-rich devices equipped with stronger processing capabilities, and longer battery life.
- In WSAN, the number of actuators is much less than the amount of sensors. The quantity of sensor nodes deployed in one specific application may be on the order of thousands.
- In automated architecture of WSAN, to provide effective actuation, a distributed local coordination mechanism is a must among sensors and

actuators. In some situations, the sensors and actuators are integrated in one physical device. The coordination can be made through internal interactions.

- In semi-automated architecture of WSAN, the physical sensor networks are in fact completely separate from the actuator networks and no coordination is required. The actuation decision is made by the central controller (control server).
- WSAN is a step further. WSN collects the information and status of the environment, WSAN acts upon the collected event which enables wealth of advanced applications and services.

2.1.4 Actuator hardware

In digital control systems, the actuator itself is a mechanical device such as switch, valve, motor and wheels. In our research domain, the actuator is not a single mechanical device. It is compound network equipment with transceiver, A/D transformer and controllable interface as shown in Figure 2.2.

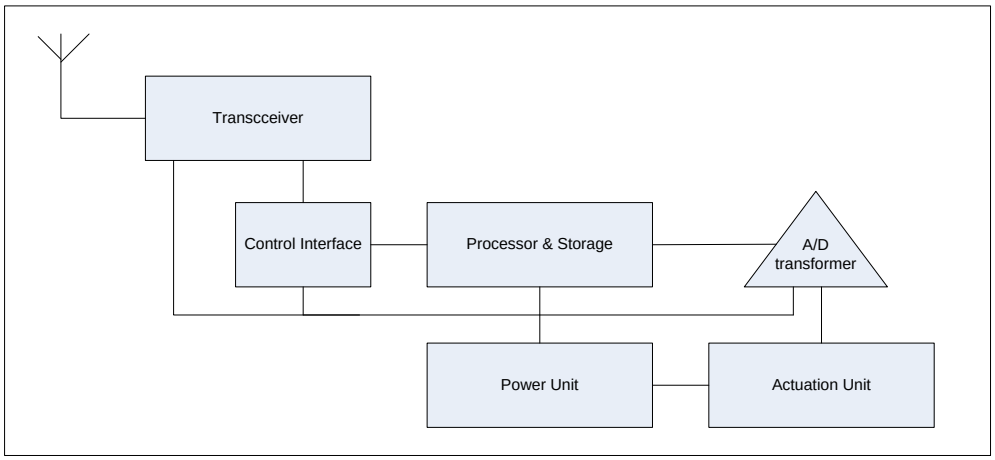


Figure 2.2 Actuator architecture and components

In many real applications, robots are used as actuator nodes. The robots can be

remotely controlled. In adverse circumstances, robots can substitute human beings to react on some event happened on site.

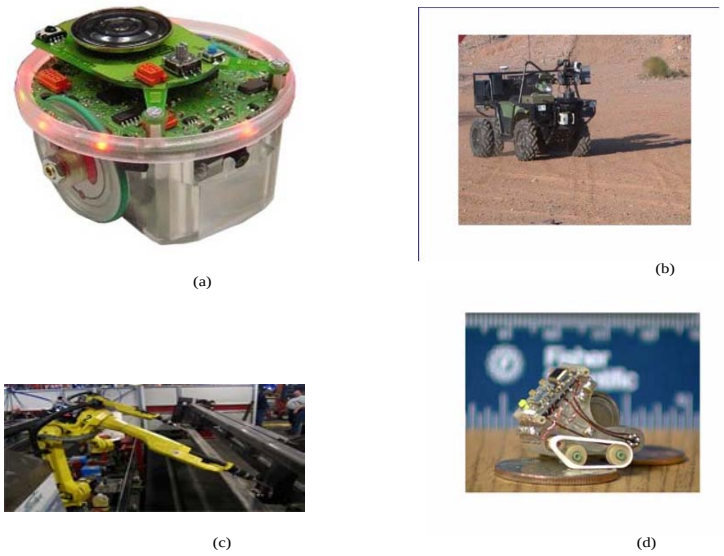


Figure 2.3 Robots developed by different research lab: (a) e-puck, (b) Robotic Mule, (c) robotic arms, (d) mini-robot

The robots designed by several robotics research firms are shown in Figure 2.3. The e-puck robot [10] in Figure 2.3 (a) is developed for the education purposes by F. Mondada and M. Bonani. It is equipped with 8 infra-red sensors measuring ambient light and proximity of obstacles, 2 stepper motors and camera. It can move around and take photos of the scene which can be deployed at emergency rescue scenario and help the central office to instruct the rescue effectively. Possibly the world's smallest autonomous mini-robot [11] (1/4 cubic inch and weighing less than one ounce) has been developed in Sandia National Laboratories is shown in Figure 2.3 (d). Powered by three watch batteries, it rides on track wheels and consists of temperature sensor, and two motors that drive the wheels. According to Ed Heller, one of the researchers, “it may eventually be capable of performing difficult tasks such as locating and disabling land

mines or detecting chemical and biological weapons". An example of Robotic Mule [12] designed for the army is given in Figure 2.3(b). These developed battlefield robots can detect and mark mines, collect information or even detonate explosives. Finally, robotic arms shown in Figure 2.3 (c) are widely used in auto industry.

2.1.5 Wireless Sensor and Actuator Network applications

WSANs are mostly application driven, i.e. they are deployed for specific purposes. Initially, they were focused on specialized applications. However, with the developments in micro-electro-mechanical, electronic systems and wireless communication, it is possible to use WSAN in commercial applications. The typical application areas include:

- Environmental monitoring and controlling,
- Medical/health-care monitoring and emergency care,
- Military surveillance,
- Home automation,
- Intelligent buildings, and
- Pervasive gaming.

For example, a set of robots that sense the environment from distributed monitoring points can turn on watering system when a dry situation is sensed. Based on the data gathered by a sensor network, a smart parking system could redirect drivers to available parking spots, etc.

2.2 IP Multimedia Subsystem

The IMS architectural framework specified by 3GPP enables IP multimedia services accessible to mobile end-user using standard Internet based protocols e.g. SIP. Examples of multimedia services include presence, instant messaging, enhanced voice and video, pervasive gaming and emergency services. Several key issues are addressed by IMS framework: IP Multimedia Sessions, QoS, Interworking, Roaming, Service Control, Rapid Service Creation, and Multiple Access.

In the following subsections, the architecture of IMS, protocols used by IMS and the IMS operations are described.

2.2.1 IP Multimedia Subsystem architecture

In most cases, IMS is modeled in three layers: Service Layer, Control Layer, and Connectivity Layer, as shown in Figure 2.4. From the overlay perspective, IMS consists of two sub-layers over IP based mobile and fix networks: Control Layer and Service Layer [6].

The service layer basically provides value-added services with a set of applications hosted on Application Servers (AS) [13]. It provides various IMS services to users. It is a home of deployed IMS services, e.g. presence, instant messaging.

The control layer provides signaling functions in IMS. It handles the registration, setup and release of calls and sessions. It consists of entities such as the Call Session Control Function (CSCF) and the Home Subscriber Server (HSS). The HSS stores and handles end users and services related information including: authentication and

authorization information, location data, and service profiles. CSCFs are SIP servers in charge of routing and session management. There exist three categories of CSCFs [13]. The detailed information about the IMS functional entities are given in later subsections.

The Connectivity Layer provides IP network access to end users and IMS functional entities. This layer is not only carrying media traffic among end users but also connecting end users to the session control and AS. From the interoperability perspective, the connectivity layer supports different network access technologies like Global System for Mobile communications (GSM), General Packet Radio Service (GPRS), Wideband Code Division Multiple Access (WCDMA), Digital Subscriber Line (DSL), and Wireless Local Area Network (WLAN).

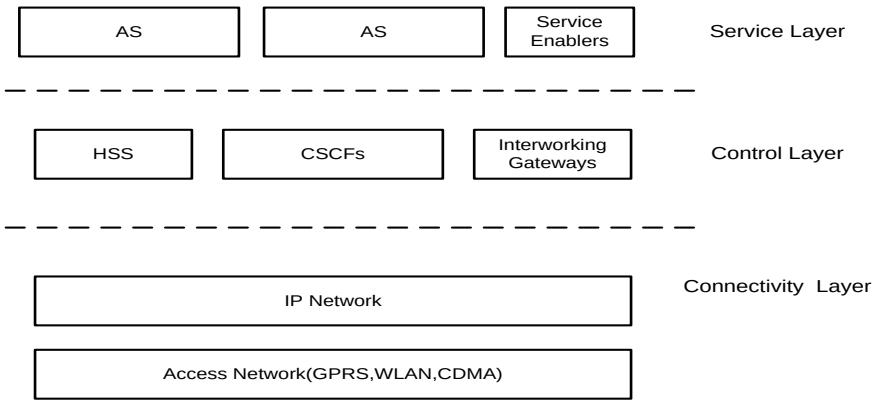


Figure 2.4 Layers of IMS Architecture

2.2.2 IP Multimedia Subsystem architecture entities

IMS architecture is a collection of functional entities linked by standardized interfaces. Figure 2.5 [14] shows the functional entities included in the IP Multimedia Subsystem Core Network. They are: HSS and Subscriber Location Function (SLF), CSCFs, Media Resource Function Controller (MRFC) and Media Resource Function Processor (MRFP), Breakout Gateway Control Function (BGCF) and Public Switch

Telephone Network (PSTN) Gateways. According to the nature of functionality, these entities can be categorized into: signaling (CSCFs); interworking (BGCF; PSTN Gateways- Signaling Gateway (SGW), Media Gateway Controller Function (MGCF), Media Gateway (MGW)); databases (HSS, SLF); media handling (MRFC, MRFP). There are many interfaces defined in 3GPP IMS, most of them are using SIP as the transport protocol, the important ones are: ISC (interface between AS and S-CSCF); Sh (interface between AS and HSS); Cx (interface between S-CSCF and HSS).

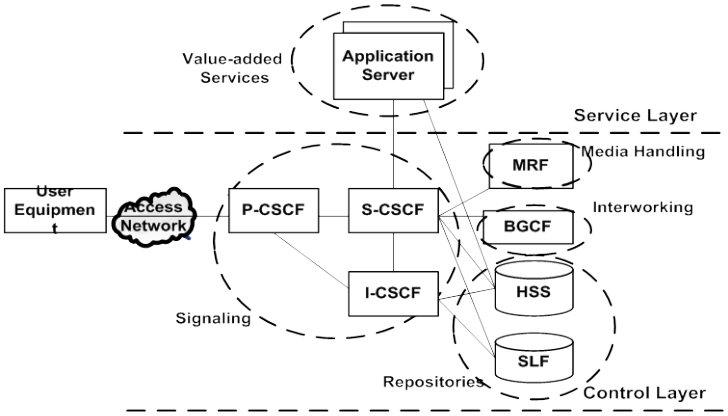


Figure 2.5 Overview of IMS architecture

2.2.2.1 Signaling entities

The IMS signaling functions are managed and maintained by CSCFs. They process SIP signaling in the IMS, generate, route and terminate SIP messages to enable service sessions and manage them. There are three types of CSCF according to their functionality: Proxy-CSCF (P-CSCF), Interrogating-CSCF (I-CSCF) and Serving-CSCF (S-CSCF).

1) Proxy-CSCF

The P-CSCF is the first point of contact between the IMS terminal and IMS Core

from the signaling perspective. It acts as inbound/outbound SIP proxy server. All the SIP messages generated by the user terminals or sent to user terminals must traverse P-CSCF.

The main functions of the P-CSCF are: security management, SIP compression and verification. During the IMS registration period, the P-CSCF will establish few IPSec associations toward IMS user terminal, this can protect the integrity.

The P-CSCF also plays a role in authenticating end users which is part of security association establishment. This will be fulfilled by both P-CSCF and S-CSCF during the registration procedure. Once the authentication has been done, the rest of the IMS network will not repeat authenticating of the same user, and they trust the P-CSCF. The mechanism to realize this function is using “P-Asserted” header. P-CSCF can also verify SIP requests sent by IMS terminals and handle compressing/uncompressing of SIP messages.

There might be several P-CSCFs deployed in the IMS depending on the number of IMS terminals the P-CSCF has to serve. This makes the IMS easy to expand.

2) Interrogating-CSCF

The I-CSCF acts as a location server in IMS Core Network. Its main functions are: selecting registrar for user terminals which try to register, by consulting the HSS, and divert the request to the IMS registrar (normally the S-CSCF). The interface between the I-CSCF and the HSS is the Cx interface using DIAMETER protocol. The HSS will return capabilities of the required S-CSCF in initial registration request situation.

According to these capabilities, the I-CSCF picks up the appropriate S-CSCF. Under re-registering situation, the HSS returns name of the assigned S-CSCF to the I-CSCF, and the request will be forwarded to that S-CSCF directly. The I-CSCF also has an interface to application servers to route requests that are addressed to services instead of end users.

Apart from the above mentioned functions, I-CSCF is located at the edge of an IMS domain and acts as an entry point for SIP messages coming from another domain. It will intercept incoming IMS call session requests from other domains (other IMS networks or legacy networks for example PSTN) and choose appropriate S-CSCF for terminating IMS sessions.

There might be several I-CSCF deployed in the IMS which makes the IMS domain scalable and redundant.

3) Serving-CSCF

The S-CSCF is the main control entity in the signaling plane and the IMS core network. Being a registrar server, it authenticates and authorizes the end users as part of their registration requests. S-CSCF maintains subscriptions to registration state and will send out notifications about any changes to the registration state. Through Cx interface towards to HSS, each user’s IMS service profile can be downloaded from HSS to S-CSCF. Based on these service profiles, S-CSCF decides which application will be triggered and the order of the operations. The S-CSCF interacts with the IMS service plane (AS) over the ISC reference point. The details will be given in IMS operations

subsection.

The S-CSCF acts as a SIP User Agent Server (UAS) to fulfill tasks such as maintaining subscriptions to registration state. As a SIP proxy server, S-CSCF is responsible for forwarding IMS session requests to the next hop SIP entities. The S-CSCF is also responsible for charging the subscriber and collecting usage records.

2.2.2.2 Databases

The HSS is the central database storing user related information. It maintains subscribe information necessary for establishing sessions between subscribers. The HSS exchanges user and service related information with AS through interface Sh. The I-CSCF and S-CSCF can access user and service related information through interface Cx. The HSS contains the following information: location information, security information, user profile information and the S-CSCF assignment.

If more than one HSS is present in IMS, the SLF is needed which is a simple database. It matches users' addresses to HSSs.

2.2.2.3 Media handling

The Media Resource Function (MRF) provides functions such as: announcement playing, adaptation between different codec schemes, mix media streams and perform media analysis and statistics functions. The MRF consists of MRFC which is located in signaling plane and MRFP which is a media plane node. The MRFC, as SIP User Agent (UA), has direct interface towards S-CSCF and it controls the media resources in the MRFP via Mp interface which is using H.248 protocol [15]. The MRFP provides media

processing functions like playing and mixing media streams.

2.2.2.4 Interworking

The BGCF controls the processing of calls to and from the circuit-switched network. It is only used when the IMS user terminals initiate sessions addressed to circuit-switched networks such as Public Land Mobile Network (PLMN).

2.2.3 Important IP Multimedia Subsystem interfaces

3GPP defined interfaces make the communications between the above mentioned entities possible in a standard way. The most important ones are the following:

The Cx interface is located between I-/S-CSCF and HSS. Information exchanged through this interface is: S-CSCF assignment procedure related information, accounting and authorization and routing information. Further details can be found in the 3GPP specifications.

The ISC interface is between the S-CSCF and SIP application servers. The SIP AS hosts services and uses the ISC interface to interact with S-CSCF to influence the SIP session. Through ISC interface, the services can be triggered by S-CSCF according to a set of pre-configured rules and policies in the format of initial filtering criteria (iFC) [16].

The Sh interface allows application servers to talk to HSS to access subscriber and service related information.

2.2.4 IP Multimedia Subsystem operations

2.2.4.1 IP Multimedia Subsystem level registration

IMS-level Registration is a procedure through which the IMS subscriber can be authorized to use the IMS services in the IMS network. This procedure is triggered by a SIP REGISTER request. During the registration, the following tasks will be fulfilled:

- IMS binds the user's public user identity (either a SIP URI [2] or a TEL URI [17]) to a URI containing host name or IP address of the terminal where the user can be reached;
- The home network authenticates the user;
- The home network authorizes the SIP registration and the right to access IMS resources.

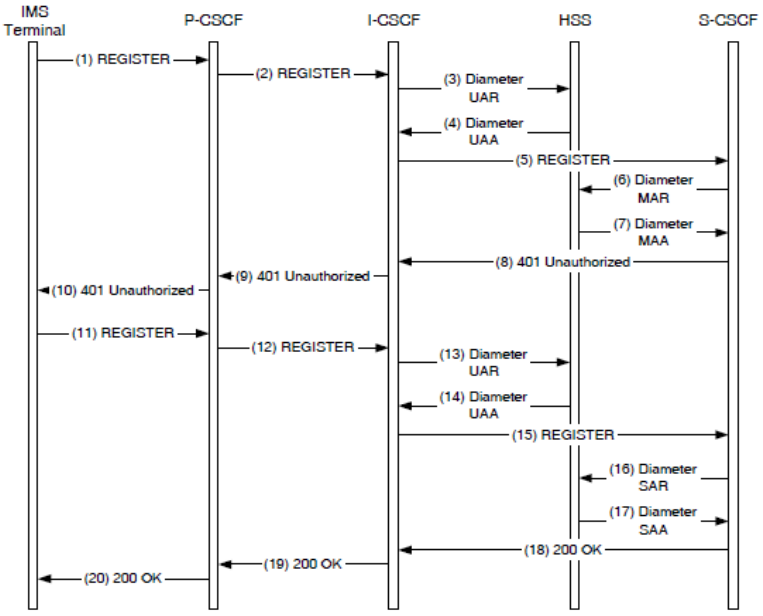


Figure 2.6 IMS-level Registration Signaling Flow

The registration is mandatory before the IMS terminal can use IMS services and

initiate any session. This procedure is shown by Figure 2.6 [13].

2.2.4.2 Service Triggering

IMS uses filter criteria and service points triggers (SPTs) [18] to trigger services. One of the most processing intensive operations undertaken by the S-CSCF is the iFC (Initial Filter Criteria) processing logic. Based on the service profile of a particular user, the S-CSCF needs to evaluate a set of XML fragments (iFCs), that hold the routing information for contacting a SIP application server that is hosting services (such as Presence service [19]) as depicted in Figure 2.7 [18]. This iFC evaluation takes place for all initial SIP requests and standalone transactions, and it takes place uniquely for each subscriber.

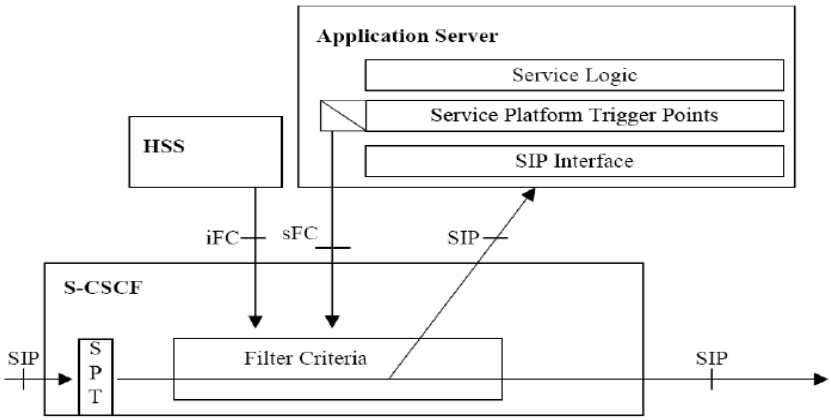


Figure 2.7 Application server triggering architecture

2.3 Session Initiation Protocol

The SIP is an application-level signaling protocol defined by the IETF for the creation and management of sessions over an IP network. The term “session” refers to the media plane communication session. In order to setup a session, SIP messages bear session descriptions that allow the participants to exchange set of parameters of the

media communication channel (session) such as the transport address and media type. In most cases, the session is described by Session Description Protocol (SDP) defined in [20]. For the information transport, the SIP messages could be carried by User Datagram Protocol (UDP), Transmission Control Protocol (TCP) or Stream Control Transmission Protocol (SCTP) [21]. Most SIP stack implementations support UDP.

SIP is a client-server and request-response protocol: client sends a request to server and waits for a reply. Requests can take arbitrarily complex path and responses take the same path in reverse direction. Requests and responses have the same structure: first line contains key information, message headers contain supplementary information, and message body contains application information [22].

2.3.1 Addressing

SIP is in many ways similar to HTTP [23]. As Universal Resource Identifier (URI) is used to identify a resource on a web server, SIP URI is used to identify the users or servers. A SIP URI is a URI. It complies with the general rules for URIs defined in [24], for example: sip:Alice@ece.concordia.ca.

A SIP URI uses the “sip:” scheme, and contains two parts split by the “@” sign. The two parts are:

- An optional user part to identify a particular user or resource at the host where the other part points. In the above example: Alice.
- A host part, which identifies the machine holding the resource. It could be a fully qualified domain name or an IP address plus an optional port value. In the above example: ece.concordia.ca.

SIP URI can be used to represent:

- Public User Identity of a user, the universal ID that anyone can establish multimedia communications with that user;
- A sip server- SIP URIs can also be used to represent SIP servers, for instance: sip:scscf.tse.concordia.ca or sip:192.168.1.3;
- A group of users, for instance, the URI sip:services@abc.com can be used to represent the customer service department in the company ABC. When a request is addressed to this URI, the server will try all the members of the group until someone can accept the request.
- A service-A SIP URI can also represent a service, as described in [25].

2.3.2 Session Initiation Protocol entities

In SIP specifications, SIP applications will assume one of the predefined roles and act on the events (received messages) according to the SIP protocol state machine. As shown in Figure 2.8, most of the SIP applications will be classified into the following roles:

- User Agent Client (UAC)
- UAS
- Proxy
- Back-to-Back User Agents (B2BUAs)

The UAC and UAS are collectively known as User Agents (UA). They are SIP endpoints that exchange messages to establish or terminate sessions. The UAC is

responsible for the generation of new SIP requests and the reception of the associated responses. The UAC acts as a SIP client, and unlike a Proxy, it is an initiator and sender of a SIP request message, instead of being an entity that simply receives and forwards. On the contrary, the UAS generates a response to a SIP request. The response accepts, rejects, or redirects the request.

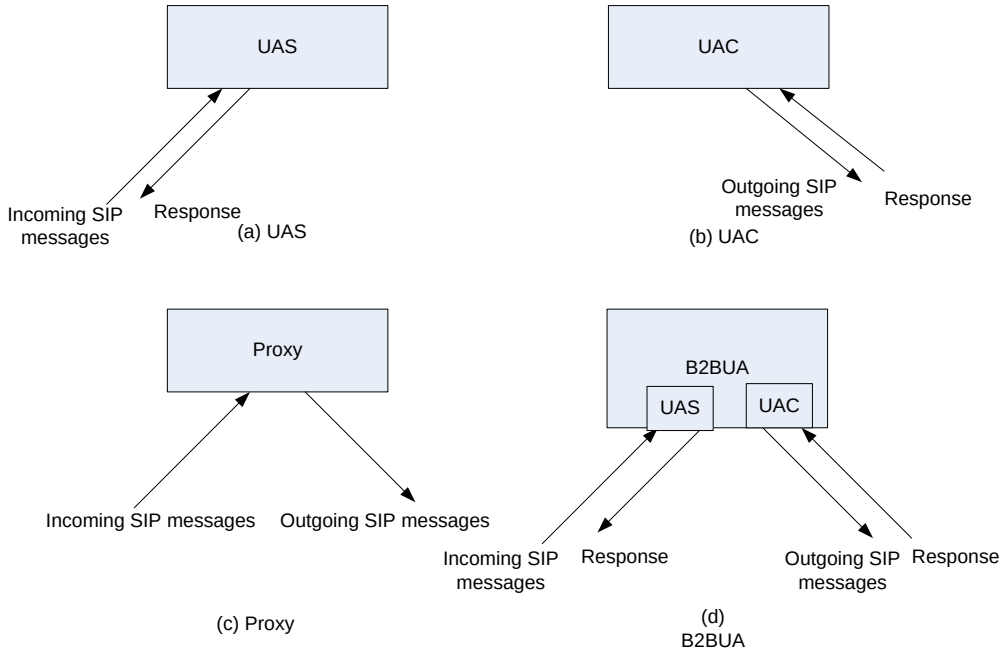


Figure 2.8 SIP Roles: (a) UAS, (b) UAC, (c) Proxy, (d) B2BUA

SIP proxies, an intermediary entity responsible for routing of SIP messages to the appropriate next hop towards their destinations. Normally, the proxy has no impact on the end-to-end interaction. This is enough when the application server hosts service logic that takes the duty of authorization, target selection and ensures the end-to-end SIP signaling proceeds without any interference. It may consult location server in SIP architecture to acquire the contact address of the end user to find out the next hop.

When the application server hosts service that demands more controls on the

end-to-end SIP interaction, it has to support back-to-back user agent (routing B2BUA) behavior. The B2BUA role behaves as an endpoint to both parties in the SIP session. The following examples show the situations where a B2BUA role is necessary: to modify an ongoing SIP request (e.g. in multimedia phone session, turn off the video channel and keep the audio part), to divert a session during its valid period (e.g. to an announcement).

2.3.3 Session Initiation Protocol messages

The core SIP functions are carried by SIP messages. The core specification of SIP defines six basic SIP messages shown in the first six rows in Table 2.1. The rest of the messages in the table are heavily used SIP extensions which are defined in several IETF specifications [26, 27, and 28].

SIP Methods	Description
INVITE	Invites the parties to join a session. It carries the description of the media session to establish.
ACK	Acknowledges the final response of media session
BYE	Terminate an existing media session
OPTIONS	Query capabilities of SIP server, such as methods supported, session description protocol, message encoding, etc.
REGISTER	To register a user to the network
CANCEL	To request a cancellation of a pending transaction
SUBSCRIBE	Subscribe to a resource (event)
NOTIFY	Notifies about subscribed resource (event)
PUBLISH	Publish info about resource (event)
MESSAGE	Send IM messages to other clients

Table 2.1 SIP Request Method and Functions

SIP responses are messages generated by a SIP user agent or SIP server (e.g. proxy) in response to client requests. The request and response together are known as a SIP

transaction. In fact, one request can generate several replies and still be counted as one transaction. There are provisional responses and final response. The response type may vary depending on request type and session context on the server side which is same as a HTTP session. The response messages are categorized into six types as shown in Table 2.2.

Response Code	Description
1xx	Provisional and Informational responses
2xx	Successful responses
3xx	Redirect response
4xx	Client error
5xx	Server error
6xx	Global failure

Table 2.2 SIP Response Code and Description

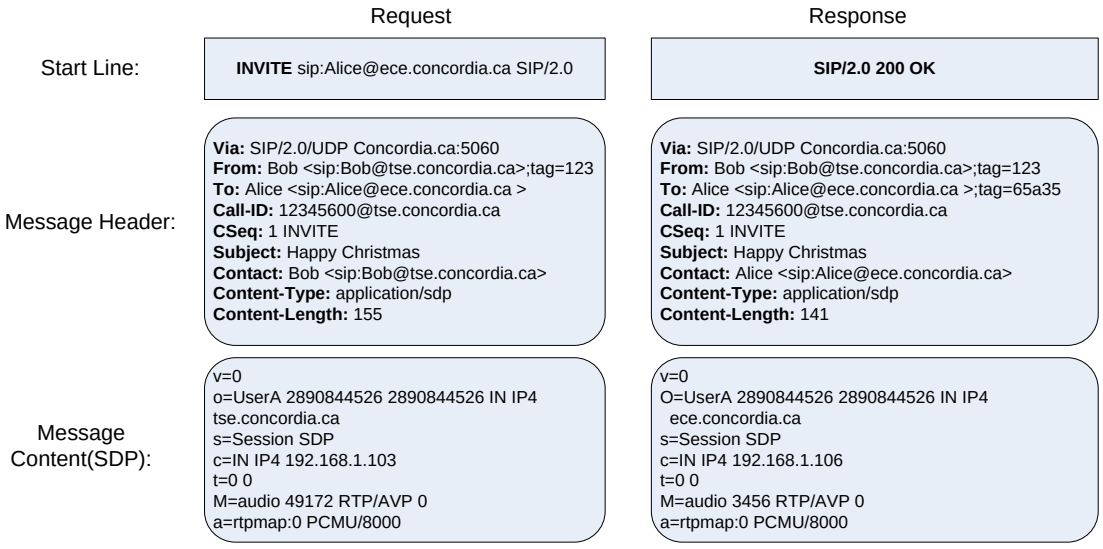


Figure 2.9 Structure of SIP Request and Response

Figure 2.9 depicts the structure of a SIP message. A typical SIP message has the format of: start line with the method name, request URI and SIP protocol versions; SIP headers; message content which is session description (could be SDP descriptions).

2.4 Conclusions

In this chapter we discussed Actuators, WSAN including its architecture and hardware components. We presented an overview of the IMS architecture, entities and its operations. Finally, SIP was described. The next chapter will deal with the state of the art work for the integration of WSAN with existing networks including IMS.

Chapter 3

Integrating actuation capabilities with existing networks: state of the art

This chapter describes WSAN integration with existing networks such as Internet, and IMS using state of the art technologies. First, we describe the application scenarios of interest and then derive a set of requirements related to these application scenarios. Following that, we review and evaluate related work with respect to these requirements.

3.1 Scenarios

This research started by determining a set of application scenarios to abstract the requirements for the integration. Applications may be divided into many domains which are difficult to enumerate all. Among the domains of interest are: Emergency Management, Health Care, and Home Automation. Based on the ambient information sensed by sensors, the action to the surroundings by actuators could make wide-spread services available to end users through IMS network. Some typical scenarios among those domains are:

- Building Fire Control Scenario

This scenario is based on the following assumptions: A huge building equipped with smoke detectors, thermal sensors, humidity sensors and remote-controlled sprinkles, fire alarm controller, camera with motor arms, fire gate motors. Scenario is described as follows: Smoke sensors detect smoke in one of the office rooms in the building. Smoke

sensors trigger alarm system in the building immediately and fire gates are closed. The application deployed over the integrated architecture stores and processes fire location information, and the application notifies fire brigade. The application actuates fire extinguish devices, adjusts Closed Circuit Television (CCTV) camera near the fire scene, turns on the emergency lighting and escape route indication. Thermal, hazardous gas and structure force-sensors are sensing temperature, gas density and pressure and keep the application informed. The application queries intelligent building management system about the structure stress, safety information and updates fire fighters with environmental information.

- Patient Monitoring Scenario

The scenario assumption: Patients with chronic disease equipped with body sensors, such as: physiological sensors (heart beat rate, breath rate, blood glucose, and blood pressure). The patient's house equipped with sensors: humidity sensor, temperature sensor, air pressure sensor and actuators: window with motor, ventilation switch, camera with motor arms. Doctors have Liquid Crystal Display (LCD) monitors, 'patient call' alarm device at their clinics. Scenario is described as follow: Environment information at patient's house is sensed and sent to IMS where the health care application is deployed. The patient's physiological information is sensed and sent to IMS. IMS stores patient's home environment information and physiological information and processes them. In case of an unusual situation, doctor's 'patient call' alarm device will be triggered and doctor's LCD monitor will be switched on. IMS will issue commands to turn on and adjust patient's camera, open windows and switch on

ventilation devices if necessary. IMS queries patient's history health records from patient information management system (external). In emergency, the IMS will notify the hospital emergency department, ambulance will be dispatched and the patient's conditions will be monitored by the doctor during transportation.

- Smart Home Scenario

The scenario assumption: home is equipped with location, thermal, humidity, gas and motion sensors and remotely controlled switches to control TV, light, heating, ventilating (actuators). Scenario is described as below: Family members arrive home (location sensor at entrance captures the info and send to IMS). The IMS instructs to switch on the light. Home environmental information (temperature, humidity) continues to be sensed and sent to IMS. IMS stores and processes the environmental information based on location, switch on/off the heating and dehumidifier based on the calculations. When gas leak is sensed, IMS will notify Gas Company, switch on ventilation, turn off electricity and trigger alarm device. Based on the family members' location (sensed by location sensors) TV/light in relevant rooms will be turned on/off. When anti-theft alarm device is set, motion detectors will be working and any intrusion will be detected and alarm devices will be triggered.

These three scenarios share some commonalities: there are various sensors and actuators involved; in emergency situation, the sensors could interact with actuators directly and make the efficient actuation available; IMS is responsible to deal with actuation coordination to avoid conflict; A lot of interactions among the IMS, WSN and AN, therefore a standardized mechanism for these interactions is a must to enable the

expandability and flexibility.

3.2 Requirements

From the above scenarios, we derived the requirements for the integration architecture:

Requirement 1: The architecture should support a wide range of actuators, e.g. switches, motors and valves.

Requirement 2: Low level actuation details, of specific actuators, should be transparent to high level entities and applications. Developers of applications should not be concerned about low level properties of actuators.

Requirement 3: The architecture should be able to support two actuation models: The automated actuation, i.e. standard actuation service based on context information sensed by sensors, and the semi-automated actuation, i.e. actuation service with intervention of end users or applications.

Requirement 4: The architecture should also support actuation arbitration to avoid collisions when several actuation requests are made in parallel. Furthermore, the framework should provide actuation aggregation and de-aggregation, which will enable simultaneous actuations on multiple actuators from the application perspective.

Requirement 5: The target architecture should rely on standard communication protocols which can easily inter-operate with IMS (e.g. SIP) and enable both synchronous and asynchronous modes of communication.

3.3 Evaluation of related work

This section discusses some of the research on integration wireless sensors and/or actuators with existing networks including IMS and Internet.

3.3.1 SENSEI Project

The SENSEI project is developing an architecture for globally scalable web resources for machines, sensors and actuators - the Real World Internet. This framework offers two fundamental services for the future Internet: context information services and actuation services [29].

The SENSEI architecture provides necessary network and information management services to enable reliable and accurate context information retrieval and interaction with the physical environment. By adding mechanisms for accounting, security, privacy and trust, it enables an open and secure environment for context-awareness and real world interaction through WSAN islands [30].

The SENSEI framework abstracts sensors and actuators as resources [31]. In the SENSEI domain, a resource is a conceptual representation of any information source that enables real world sensing or has the ability to act upon the environment. In addition to resources that have direct access to the physical world, the concept covers also indirect information sources that acquire context information via aggregation, fusion or even inference (composing) from existing SENSEI resources. The SENSEI framework has been designed using fundamental concepts of the World-Wide Web. In order to enable the SENSEI framework even on the most constrained devices like sensors and

networks such as IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) [32], the SENSEI embedded resource concept is used to extend the web resource model to minimal IPv6 [33] nodes with very little overhead. It also maintains end-to-end IP principles and easy interoperability with existing web protocols.

The detailed SENSEI architecture is shown in Figure 3.1 [29]. There are several core components defined in SENSEI:

Resource Directory: Serving as a linking point for resources and resource users, it stores descriptions of all available resources. XML is used to describe the resources. It has direct interface with WSAN islands gateways. The 6LowPAN is selected as the interface protocol for the interactions between the WSAN islands gateways and the Resource Directory.

Semantic Query Resolver: Responsible for analysis of high level user queries and discovery of suitable and available resources capable of providing information required to respond to the queries. It will consult Dynamic Resource Creator (DRC) for resource composition when the requested resource does not exist.

WSAN islands interact with the framework via their respective gateways.

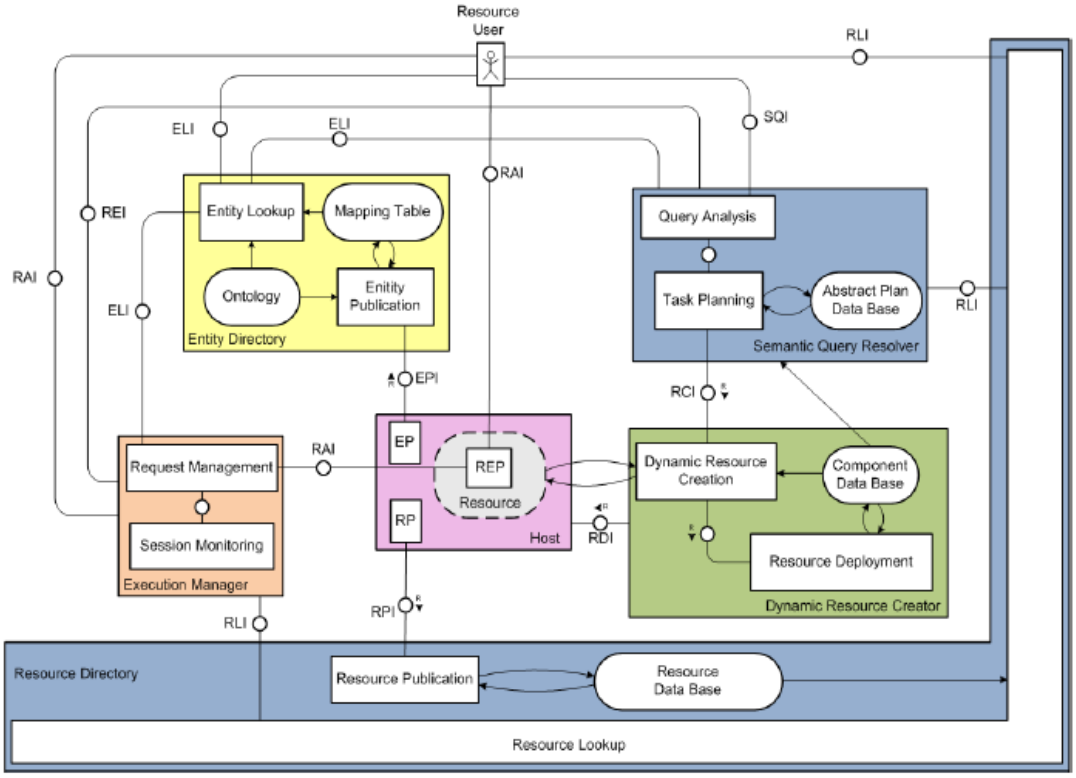


Figure 3.1 Detailed SENSEI Architecture

For the actuation service, SENSEI framework supports the following actuation models:

1) Embedded actuation

Under this model, the actuation task can be started and normally parameterized, but the sources of context information (real-world sensor) as well as the control logic are embedded in the real-world actuation device. Most of the actuators are embedded in the objects without interface to third party applications (see Figure 3.2 [29]). This is the predominant situation on the market.

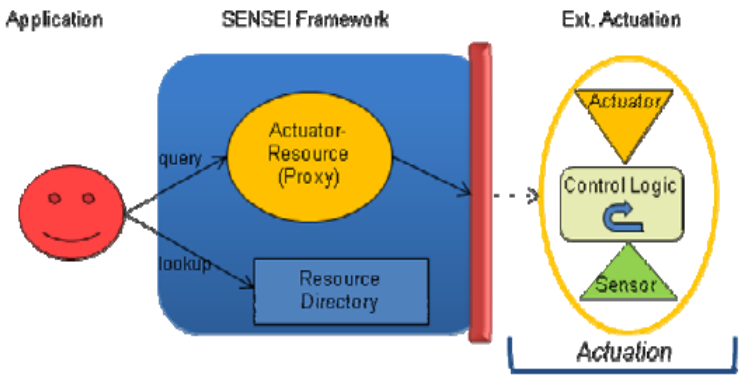


Figure 3.2 SENSEI embedded actuation

2) Application based actuation decision

If the Application is performing the actuation control function, neither the real-world actuator and sensor devices are aware of the control logic, nor is the SENSEI framework (see Figure 3.3 [29]).

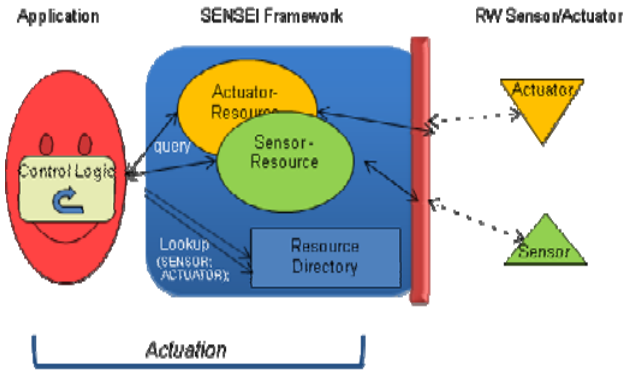


Figure 3.3 SENSEI application based Actuation Decision

The SENSEI framework provides a way of integrating sensors and actuators with Internet. It addresses the interfaces and components for accessing the physical devices through the linking services (resource directory, entity directory). It supports both automated and semi-automated actuation, its main goal is to integrate WSAN with Internet via gateways, but the proposed gateway model and actuation task model is

specific to the Internet and does not consider IMS specific requirements for WSAN/IMS integration. Further it is still ongoing and the protocol for components interactions is yet to be decided.

3.3.2 e-SENSE

The e-SENSE project [34] defines open gateway architecture to facilitate connectivity and integration of information offered by WSN with Beyond Third Generation (B3G) wireless networks. In its later evolution, it proposes architectural extensions for the integration of e-SENSE systems with IMS based service platforms. Within the IMS domain an e-SENSE Service Enabler has been introduced as a new functional element [34].

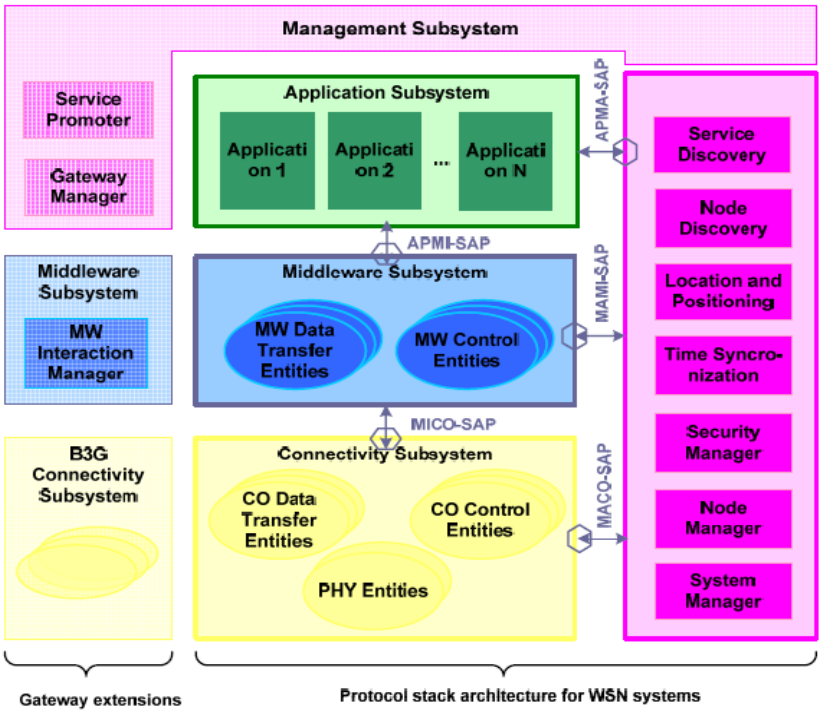


Figure 3.4 Enhanced e-SENSE protocol stack

Figure 3.4 [34] shows the structure of the e-SENSE protocol stack. It is divided

into four logical sub-systems, namely the connectivity (CO), middleware (MI), management (MA), and application (AP) subsystem. Each subsystem comprises various protocol entities, which offer a wide range of services at various service access points (SAPs) to other sub-systems. The services and respective protocol entities can be combined in many ways to configure the protocol stack according to the role of the sensor node and application requirements.

Figure 3.5 shows the relationship of the e-SENSE service enabler with respect to other service enablers in an IMS domain. A context aware application, typically hosted on an IMS AS, is able to use the e-SENSE service enabler as a service building block, similar to other IMS service enablers such as the presence service, instant messaging (IM) service and group management (GM) service.

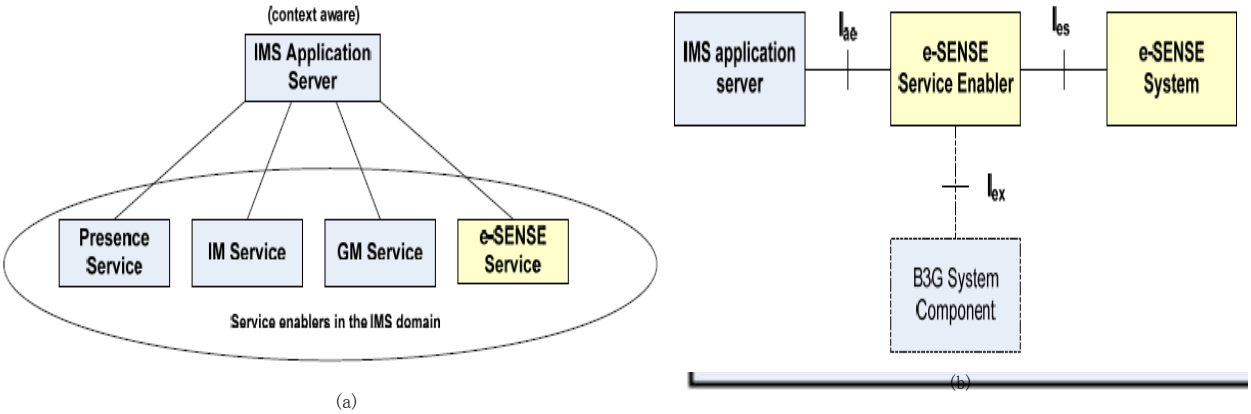


Figure 3.5 e-SENSE enabler in IMS: (a) e-SENSE service in IMS environment, (b) interfaces between e-SENSE entities and other IMS network components

Although e-SENSE project extends its initial architecture by introducing open gateways towards IMS enable its integration with IMS as a service enabler, it does not offer any support for actuation. Further, wrapping everything in a gateway is not an efficient way of information access, for the context information, this may incur repeat

adaptation when it has to go through two levels of gateways: WSN level (to e-SENSE) and e-SENSE level (to IMS).

3.3.3 Presence based integration of Wireless Sensor Network and IP Multimedia Subsystem

This project proposes a presence based architecture for the integration of WSN and IMS, focusing on how the information is conveyed from the WSN to the presence infrastructure in IMS.

To enable the integration of WSNs in the IMS, this architecture assigns the role of Presence External Agent (PEA) to the WSN gateway, which publishes information provided by the WSN about different entities (user and non-user entities) to an extended Presence Server (PS). The PS manages the different types of context information provided by the sensors. Other entities such as other IMS ASs, IMS core network entities (e.g. CSCFs), and IMS user applications can act as watchers to the information published in the presence server, and use this information to provide value-added services to end users. Figure 3.6 [6] depicts the proposed architecture.

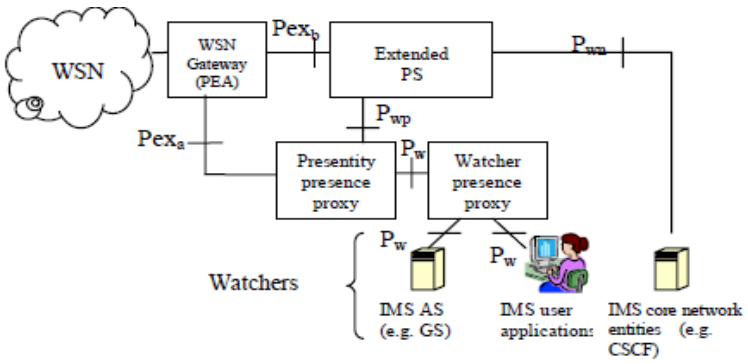


Figure 3.6 WSN/IMS integrated architecture

The architecture divides the interface between the WSN gateway and the PS into

two sub-interfaces: Pexa, and Pexb. Pexa is used for the exchange of contextual information between the gateway and the PS, via a trusted node (a presence proxy). PS directly interacts with the gateway through Pexb interface. The proposed architecture also defines interfaces between the PS and watchers: the Pw and the Pwn interfaces. Pw is an enhancement of the existing 3GPP interface which enables end user applications and IMS AS to access presence information on the PS, via presence proxies. Pwn is a new interface that enables network entities acting as watchers to get direct access to presence information from the PS.

However, the sensing is focus on the context information acquiring and providing while the actuation is command based and there should be input from the third party to trigger it. The architecture designed by this project intends to integrate sensing capabilities and enrich the context aware services in IMS, but it does not address issues relevant to actuation.

3.4 Evaluation summary

In the previous sections, we discussed three projects most relevant to our targeted architecture. These projects aim to integrate sensors with either IMS or Internet. But none of them directly addresses the protocols and information models needed for integration of actuators with IMS. In fact, as mentioned in the introduction chapter, our target is to find a solution for enabling actuation service for the end users or applications through a ubiquitous mechanism. The evaluation of the three approaches with regard to the requirements set in the beginning of this chapter is shown in Table 3.1. As we can see from the table, none of these approaches come close to meeting all the requirements.

Thus, we have to develop a new architecture dealing with the stated objectives.

Related Work / Criteria	SENSEI	e-SENSE	Presence Based integration of WSN and IMS
Criterion 1: support a wide range of actuators	Not mentioned	No	No
Criterion 2: two levels of actuation abstraction	No	No	No
Criterion 3: two actuation models	Yes	No	No
Criterion 4: support actuation arbitration	Yes	No	No
Criterion 5: support standard session control protocol-SIP	No	Yes	Yes

Table 3.1 Evaluation summary of related work

3.5 Conclusions

This chapter considered related research projects on the integration of actuation capability with existing networks or integration of sensing ability with IMS. According to the evaluation, none of the existing projects meet for the targeted architecture requirements. In the rest of the thesis, we present a new architecture, demonstrate that it meets the requirements and give its implementation.

Chapter 4

Integration of Wireless Actuator Networks with IP Multimedia Subsystem

The evaluation of related work in the previous chapter showed that there is no existing architecture that meets the requirements of the integration. Thus, we need to develop a novel architecture. This chapter presents the proposed architecture and its principles and the key architecture components.

4.1 Overall architecture and principles

Figure 4.1 depicts the integrated architecture with two new entities, the Actuation Control Function (ACF) and the IMS/Wireless Actuator Gateway (WAG), as well as actuation command interfaces-Aa and Ag.

According to the Requirement 5 mentioned in the previous chapter, we try to leverage existing components of IMS as much as possible in our architecture to ensure the compatibility. To meet the Requirement 2, we introduce the ACF to IMS to provide two levels of abstraction: the low level for the proprietary and actual commands used to control an actuator via the WAG and the high level which enables requests from AS. The applications do not need to know how to control an actuator directly and which actuator the command should be sent to. The applications see the ANs at a high level of abstraction and from a logical point of view. With regards to the Requirement 1 and 2, we use gateway based solution to hide heterogeneity of actuator network structures and

technologies. The rest of the architecture remains the same as in the IMS.

The ACF acts as an application server from the IMS perspective and handles high level actuation requests from other applications. It decomposes the requests into actuation commands targeting specific ANs. The ACF receives actuation responses from the WAGs, aggregates them and feeds them back to the appropriate applications.

The WAG transforms the actuation commands sent by the ACF to low level, proprietary and actual actuation commands that can be understood and executed by actuators. It dispatches these commands to individual actuators and collects the responses. The WAG is not necessarily an entity in IMS. It could be part of the actuation infrastructure or even belong to a third party. The actuators act on the environment in reaction to the commands. The main reason for introducing the WAG into our architecture lies in two aspects: providing support functions, such as actuation arbitration; and taking care of the low level actuation commands to lighten the load of other IMS entities. These entities do not need to know or handle the physical commands for controlling specific actuators.

Like other application deployed in the IMS, the ACF can be invoked by the S-CSCF through the ISC interface based on SIP protocol. Other applications interact with the ACF via the Aa interface through which actuation requests and responses are carried. The ACF can also interact with HSS via the Sh interface which carries actuation service related control information, such as subscription, authorization and accounting. Through the Ag interface, the ACF interacts directly with the WAG exchanging decomposed actuation commands and actuation feedbacks. Ac, the interface between

P-CSCF and the WAG, carries registration information and facilitates the interaction between the ACF and the WAG using standard SIP protocol. The other entities in the architecture remain the same as in the existing IMS framework.

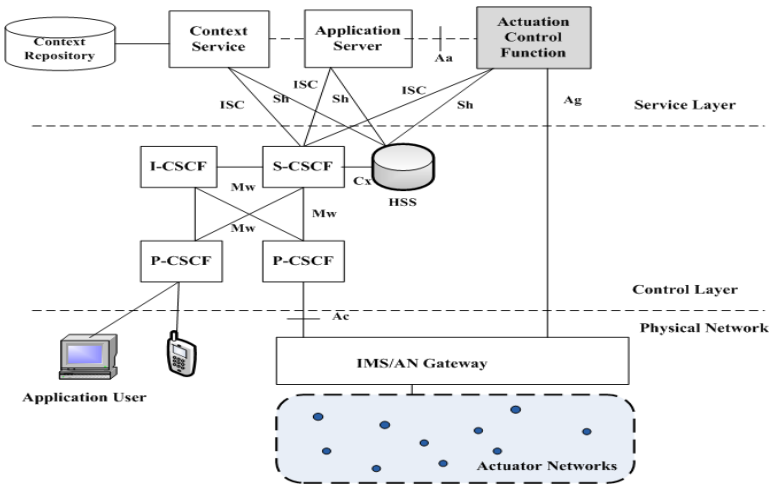


Figure 4.1 Architecure Overview

4.2 Design of Actuation Control Function

ACF is an intermediate entity between other applications deployed on an IMS AS and WAG. It is an application level entity that provides standard interfaces (Ag, Aa) for actuation command transporting. Figure 4.2 shows the proposed ACF structure which consists of two layers, namely connectivity layer and actuation control layer.

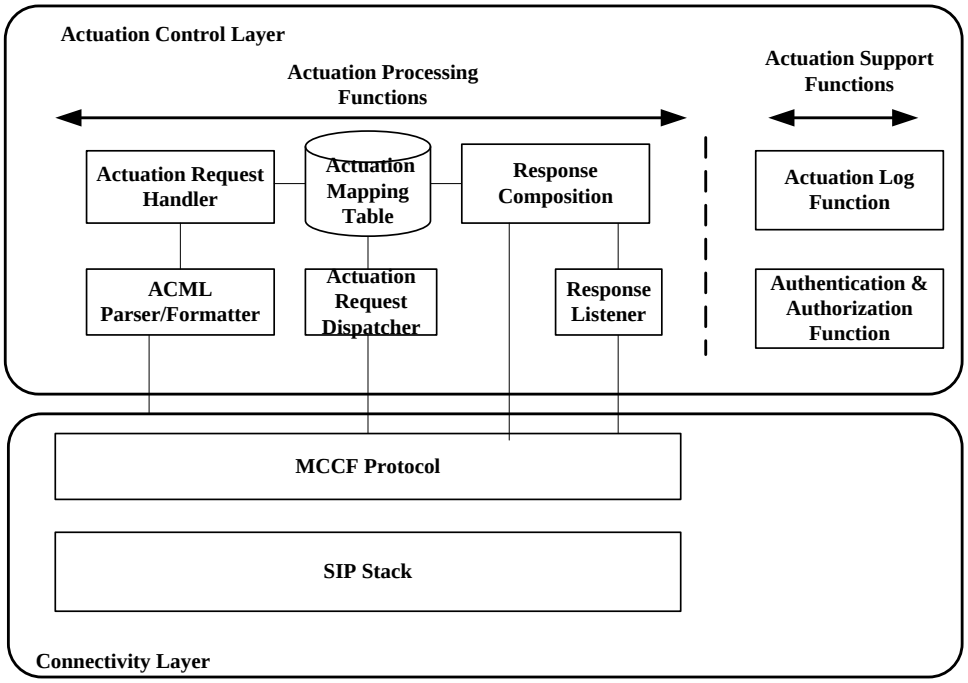


Figure 4.2 ACF software structure

4.2.1 Actuation control layer

The actuation control layer is the core functional layer of the ACF. It is responsible for handling and dispatching actuation requests from other applications or services deployed on a SIP AS. It consists of two categories of functions: actuation processing functions and actuation support functions.

4.2.1.1 Actuation processing functions

The actuation processing functions are realized by the following modules: actuation request handler, actuation request dispatcher, response composition via response listener function and an Actuation Command Markup Language (ACML) parser/formatter.

When the ACF receives a request from an application, it decomposes it into actuation requests according to its knowledge of the actuators distribution, i.e. the

mapping table of the targeted objects versus actuator gateways; the actuation request dispatcher will then send the decomposed requests to the appropriate gateways. Upon reception of the responses from the gateways, the response listener forwards them to the response composition module, which will combine them and send a response to the originator of the request via the MCCF. The detailed process is explained in the following paragraphs.

ACML Parser/Formatter Function: This module performs message content abstraction, ACML parsing or ACML formatting depending on whether it is receiving or sending a message. When an actuation request carried by CONTROL message reaches ACF, the control package in the content part of the message will be analyzed and the values of the elements will be put into an object as attributes. According to the ACML definition, the elements are: Object Name, Civic Address, Index, etc. The ACML will be explained in more detail in the next chapter. Once the Actuation Request Handler module finishes its processing, a new request will be generated. This module will then be used to build the control package and embed it into the new message.

Actuation Request Handler Function: Based on the result of the ACML parser, this module will perform the following tasks: first, it will do mapping. According to the element values, it will look into the preconfigured mapping table and try to find out which WAG is involved and its IMS identity (SIP URI). At the same time, if more than one WAG entries are found, it has to keep records of the request for each entry for reverse mapping. The reverse mapping is used for response composition. This module will built the actuation request content in ACML with the SIP URI of WAG as its

destination and hand over to Actuation Request Dispatcher for sending.

Actuation Request Dispatcher Function: Acting as SIP UA and later as control client, the ACF through this function module sends out a request message addressed to specific WAGs. It will act as control client towards WAG and start a new SIP INVITE session to establish an MCC. It stores the unique MCCF Dialog ID (from the CONTROL message header) together with the ID of the new MCCF Dialog established towards WAG. The main function of this module is to liaise with Connectivity layer functions and trigger new SIP and CONTROL sessions. It also manages a request queue when there is more than one request towards the same WAG.

Response Listener and Composition Functions: The response listener monitors the response from WAG. Once the response arrives to ACF from WAG, this module will get the unique dialog id from the response, then hand over the id and the status code to the Composition module. The composition module will build the overall response to the original request according to the reverse mapping table established by Actuation Request Handler Function. Then it will trigger the CONTROL session to send the response to the original applications which sent out the request.

Actuation Mapping Tables: There are two types of mapping tables: WAG mapping table, and request matching table (reverse mapping). WAG mapping table is used to locate the specific WAG to which a request from an AS addresses, it is preconfigured and static. Request matching table is used during response composition process. Through the entries of this table, a response message from WAG will be related to an existing request from AS which involves multiple requests to WAG. This table is

dynamically built by Actuation Request Handler Function.

WAG mapping table entries will have the following format: (civic location, object name, WAG IMS identity). A sample WAG mapping table is shown in table 4.1.

Civic location	Object name	WAG IMS identity
A.11.08	Camera	wag_a@ericsson.com
B.12	Sprinkler	wag_b_sprinkler@ericsson.com

Table 4.1 Sample WAG mapping table

Request matching table entries will have the following format: (id of original MCCF Dialog from AS to ACF, id of generated MCCF Dialog from ACF to WAG).

The id is generated by the message initiator and negotiated by SDP during the establishment of MCC. It was carried by “cfw-id” attribute. It is key information for the MCCF dialogs which will be referred to by both control client and control server. A sample request matching table is shown in table 4.2.

Original MCCF Dialog ID	Generated MCCF Dialog ID
6e5e86f95609	518ba6047880
6e5e86f95609	5feb6486792a
4hrn7490012c	2b4dd8724f27

Table 4.2 Sample request matching table

4.2.1.2 Support functions

The support functions include: Actuation Log module, Actuation Authorization and Authentication module.

Actuation Log function: This module keeps records of actuation service usage for accounting purpose and tracks the problem or request failure and events, similar to the server log on application server.

Authorization and Authentication function: This function performs actuation service authorization and authentication. It interacts with HSS to update service access profile which is used to judge whether an application has the right access authorization. It also performs the actuation request verification to ensure the request has the right format and integrity of necessary parameters.

4.2.2 Connectivity Layer

The connectivity layer implements IMS communication stack (SIP) which is used to establish connectivity to IMS network. It also implements MCCF protocol, which is used to interact with application servers and WAG.

4.3 Design of Wireless Actuator Gateway

WAG masks the details of the heterogeneous ANs and makes standard access to actuation service from applications possible through the integration with IMS. It is an application layer gateway and performs translation of actuation request to proprietary commands that actuators can accept. The software architecture of the WAG is depicted in Figure 4.3. It has similar structure to the ACF and also consists of two layers: connectivity layer and actuation control layer.

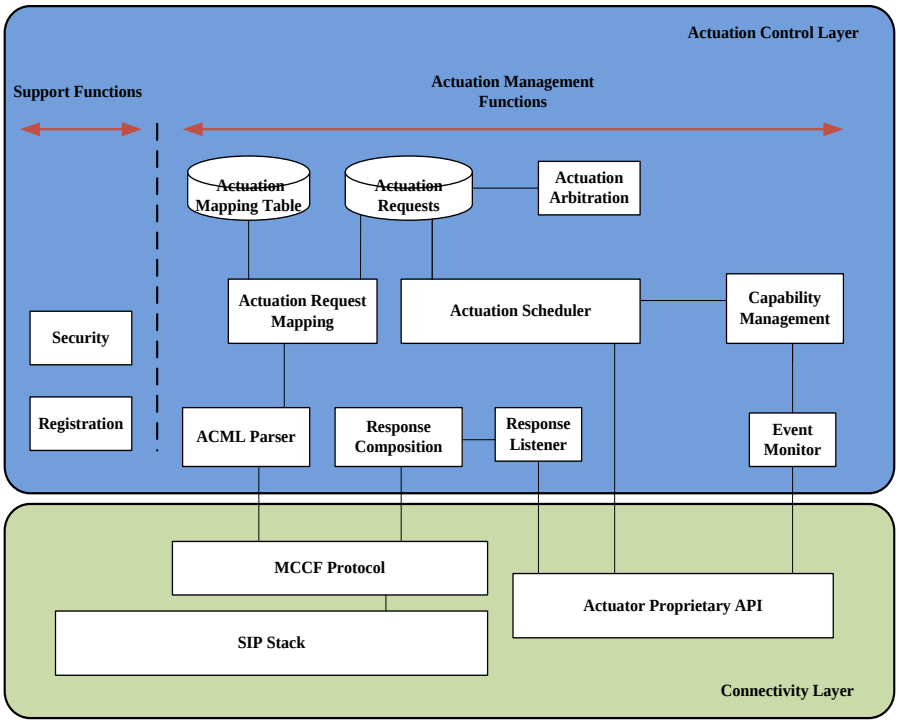


Figure 4.3 WAG software structure

4.3.1 Actuation control layer

The core of the WAG is the actuation control layer functions, which consists of two groups of functions: actuation management functions and actuation support functions.

4.3.1.1 The actuation management functions

The actuation management functions are performed by the following modules: a set of actuation processing modules (request mapping, actuation arbitration, and actuation scheduler), a response composer, a set of monitor modules, capability management and an ACML parser. The ACML parser has the same function as the one in ACF.

The actuation request mapping module: Performs mapping of logical entities to

physical actuators. It looks up the preconfigured mapping table and find out the relevant actuators. Later on, the mapping table can be updated dynamically based on the capability management events. An example of the mapping table is shown in Table 4.3:

Civic location	Object name	index	Actuator identity	Status
A.11.08	Camera	02	Camera02	Active
B.12	Sprinkler	03	Sprinkler03	Inactive

Table 4.3 sample of mapping table at WAG

The index element is optional. If no index appears, it means all the actuators at the location. The actuator identity format may vary in different actuator networks. In an IP-enabled actuator network, WAG will talk to the actuators via IP connections. The IP address can be found through the actuator ID. Then the communication will be established based on the IP address.

Capability Management: Monitors status of actuators in real time through an event monitor module. If some actuators fail, the actuators' status will be changed from "active" to "inactive" in the mapping tables.

Actuation scheduler: It interacts with capability management module to acquire the current availability of specific actuators. When the actuation request mapping completes, one or more actuation commands related to the request will be dispatched to specified actuators via the actuation scheduler.

Actuation Arbitration:

In the situation of multiple requests addressed to the same actuators in parallel, this module will put all the requests into a queue and resolve the conflict using ‘First Come First Service’ policy.

Response Listener and Composition: The Response Listener and Composition modules have similar functionality with their namesakes in ACF. There are though some differences: The Response Listener receives the response message through the proprietary interface, and the mapping table is different, it is using actuator ID to match with the MCCF dialog ID. The mapping table entries are in the following format:

(ID of MCCF Dialog from ACF to WAG, actuator ID).

When the results and feedbacks from the actuators come in, an overall response will be sent by the response composition module to the ACF.

4.3.1.2 Support functions

Security functions: Besides authorization and authentication functions, this module acts as firewall to actuator networks. It will force the security policy preconfigured in the WAG to avoid being attacked.

Registration Function: This module is for future expansion. Currently this architecture implements static (preconfigured) WAG mapping that means all the WAGs are configured into the system before being put into service. In the future, we will implement dynamic joining or leaving of WAG by using IMS registration operation procedure.

4.3.2 Connectivity layer

In the connectivity layer, the standardized actuation control interface (composed of SIP stack and MCCF) and proprietary actuator control interface are combined together to enable communication between IMS entities and individual actuators.

SIP and MCCF: This interface implements IMS communication stack (SIP) and establishes connectivity to the IMS network. The IMS interface interacts with IMS network entities, CSCFs (Proxies). It also implements MCCF, established by SIP INVITE sessions [8], which will be used to interact with ACF.

Actuator proprietary interface: This interface implements communication stack of ANs. The communication interfaces between actuator nodes are proprietary. It should support different proprietary interfaces to communicate with heterogeneous actuator platforms such as Zigbee actuators, 6LowPAN actuators, Robots. In our implementation, we implement an interface to a simulated robot.

4.4 Actuation application server

In order to use the actuation service, the application has to implement MCCF and send the request via CONTROL messages. The rest part of the application deployed in the IMS remains unchanged.

4.5 Conclusions

In this chapter, we have proposed an architecture for the integration of actuation capabilities with IMS. The architecture introduces two new functional entities into the IMS framework: the ACF and the WAG. Then the design of the key architecture

components is presented. The software functions of these components are depicted thoroughly.

This architecture is designed to meet the requirements we set in chapter 3. First, it can support wide range of actuators via gateways (the WAGs). Second, it introduces two levels of actuation command abstraction through the ACF and the WAG, which masks the physical details of the actuators. This provides service developers flexibility to develop actuation enabled applications. Third, it supports both actuation models: automated and semi-automated. The actuation arbitration is also supported at the WAG level. Finally, this architecture is based on the IMS, it supports standard session control protocol-SIP.

The implementation of the architecture is described in chapter 6. The next chapter gives detailed explanation of the Actuation Control Protocol and the information model we developed.

Chapter 5

Actuation Control Protocol and Actuation Control Information Model

In this chapter, we will determine actuation control protocol and information model for the architecture described in the previous chapter. In order to select the appropriate protocol to carry the actuation commands, we first set criteria for evaluating potential solutions. Then a few of existing protocols that deal with command semantics are evaluated with regards to the selection criteria. Later, the actuation information model is presented. Finally, the data flow of the actuation task delivery and processing is described.

5.1 Criteria for actuation control protocol selection

Based on the functionality of Aa and Ag interfaces specified in chapter 4, we set the following criteria for the protocol evaluation:

Criterion 1: The protocol should be stateful. The stateful requirement comes from the fact that application may cancel actuation requests or trigger a sequence of actuations with some kind of dependency.

Criterion 2: The protocol is supposed to leverage IMS capabilities as much as possible. IMS has already defined a number of supporting facilities to ease the service delivery, such as authentication, authorization functions. Instead of choosing protocols alien to the IMS scheme, try protocols compatible with IMS networks which will ease

the implementation and improve expandability.

Criterion 3: The protocol should support conditional actuation requests. Under certain circumstances, the user could request an actuation to be triggered at a later time or under certain conditions to increase service flexibility.

Criterion 4: The protocol should support complex actuation requests. It should allow conditional actuation and compound actuation. For instance, a user could request actuation A or actuation B executed under condition 1, both of them to be executed under condition 2. This should rely on the information model. However, the protocol should be able to support carrying of such information model.

Criterion 5: Actuators should be transparent to the integrated architecture entities (IMS AS, ACF) as much as possible.

Criterion 6: In order to minimize processing time and ease its implementation, the actuation control protocol should be as simple as possible.

5.2 Evaluation of command semantic carrying protocols

A few existing protocols can be considered for actuation controlling: Megaco [35], Simple Mail Transfer Protocol (SMTP) [36], Standard SIP, and MCCF [8]. All of them except SIP can carry command semantics. Standard SIP plus command semantic carrying information model could also be an option. Next we evaluate each of these protocols with regard to the set criteria in the above.

5.2.1 Simple Mail Transfer Protocol

SMTP is a relatively simple, text-based, application level protocol. Figure 5.1

shows the SMTP structure. It uses a limited set of commands (as shown in table 5.1) and reply codes for its communication. The objective of SMTP is to transfer mail reliably, efficiently and easily. Under SMTP, a mail sender communicates with a mail receiver by issuing command strings and supplying necessary data over a reliable channel, typically a TCP connection.

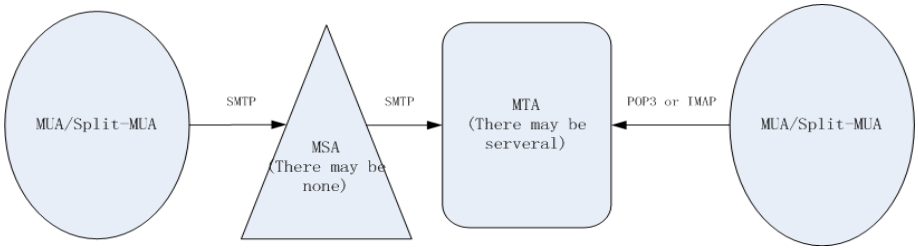


Figure 5.1 SMTP structure

An SMTP session contains commands initiated by an SMTP client and corresponding responses from the SMTP server. A session may include zero or more SMTP transactions. It is stateful and contains three states: session initiation, mail transactions and session termination. An SMTP mail transaction consists of three command/reply sequences. They are:

1) **MAIL** command, to establish the return address.
2) **RCPT** command, to establish a recipient of this message.
3) **DATA** to send the text content of the message. It consists of a message header and a message body separated by an empty line.

The SMTP has limitations: It does not support for the server proactively notifying client about special events, and does not have any delayed and periodical command execution mechanism. It is not compatible with standard protocols used in the IMS.

SMTP Commands	Description
HELO / EHLO	This command is used to identify the sender (client) to the SMTP server.
MAIL FROM:	Specifies the sender's e-mail address (and name, if used).
RCPT TO:	Specifies the recipient's e-mail address (and name, if used).
DATA	Starts the transfer of the actual data (body text, attachments etc).
RSET (RESET)	Specifies that the current mail transaction will be aborted.
VRFY (VERIFY)	Asks the receiver to confirm that the argument identifies a user or mailbox.
HELP	This command causes the server to send helpful information to the client.
QUIT	Quits the session.

Table 5.1 SMTP commands

SMTP is specifically tailor made for mail transfer, the commands it defines serves that goal well but does not offer any support to other complex control semantics. If we consider inserting command in the message content, the plain text message is not able to carry any command semantics and provides no help to our problem domain.

5.2.2 Session Initiation Protocol with Media Server Markup Language

The Media Server Markup Language (MSML) [37] provides a way to control media servers independent of transport protocols, although it is normally carried by a SIP INFO [38] message or an MCCF message [39]. It can be used to control and invoke many different types of services on IP media servers. The Clients can use it to define how multimedia sessions interact on a media server. MSML can be used, for example, to control media server conferencing features such as video layout and audio mixing, create sidebar conferences or personal mixes, and set the properties of media streams [37].

It abstracts the media server as Media Server Object Model. This model assumes that there exists one single control context within a media server. This control context is aware of the state of all media objects and media streams within the media server. The objects are endpoints of one or more media streams. There are four types of such objects: network connections, conferences, dialogs and operators. The single control context receives and processes all MSML requests and events generated internally by media objects and sends them to the appropriate SIP dialog.

The IETF draft presents two alternative ways to transport MSML: One is by SIP INFO messages and the other is using the MCCF. MSML commands are sent from a client to the media server via SIP messages (most notably the INFO message). The body of the SIP message contains the XML control syntax. The MSML request may carry several actions (elements) to be processed or a single command.

The language structure of MSML is based on a package and a profile scheme as shown below:

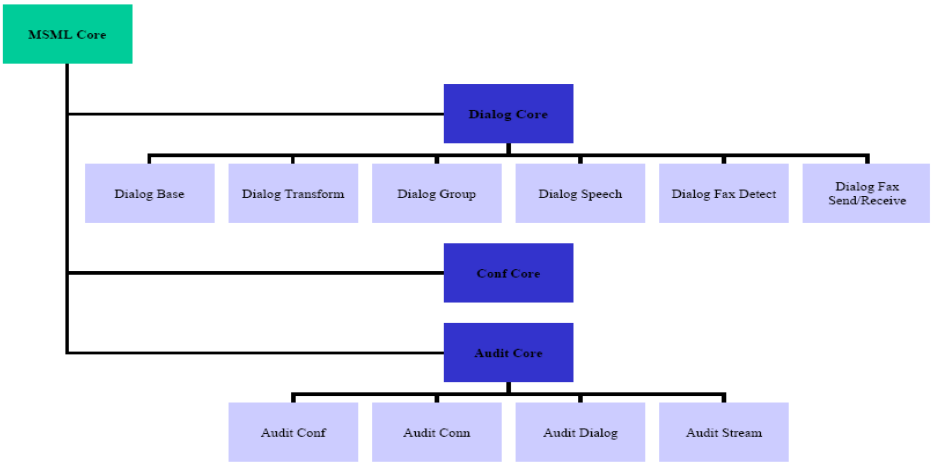


Figure 5.2 MSML core package scheme

Not all devices and applications using MSML need to support the entire MSML

schema. For example, a media processing device might support only audio announcements, or only multimedia IVR. It is highly desirable to have a system for describing what portion of MSML a particular media processing device or control agent supports. The MSML profile scheme is designed for this purpose and it defines subset of a given MSML package with specific definitions of elements and attributes.

From the functional perspective, the MSML is more like definition of an information model. It was designed specifically for the media server controlling, to support actuation service, we have to define a new package. And the SIP INFO method is not suitable for carrying control messages for the following reasons: SIP INFO is an opaque request with no specific semantics. A SIP endpoint that receives an INFO request does not know what to do with it if only based on SIP state machine. It was not created to carry generic session control information along the signaling path, and it should only really be used for optional application information. It traverses the signaling path, which is an inefficient use for control messages that can be routed directly between the controller and the controlled.

5.2.3 Megaco

Megaco, officially called H.248, is a mature and complex protocol and specifically designed for a media server to control media gateways. It specifies the relationship between the Media Gateway Controller (MGC) and the Media Gateway (MG), which initially defined with function of converting circuit-switched voice to packet-based traffic, and later on it was also assigned the responsibility of mixing streams for multimedia conferencing.

Megaco/H.248 defines two basic components: terminations and contexts [35]. Terminations represent streams entering or leaving the MG (for example, analog telephone lines, and RTP streams [40]). Terminations have properties, which can be inspected and modified by the MGC.

Megaco/H.248 uses a series of commands to manipulate terminations, contexts, events, and signals. Table 5.2 shows a list of the commands.

Megaco commands	Function
Add	Add a termination to a context
Modify	Modify the properties, events and signals of a termination
Subtract	Disconnect a termination from its context
Move	Atomically move a termination to another context
AuditValue	Return the current status of properties, events, signals and statistics of Terminations
AuditCapabilities	Return all the possible values for termination properties, events and signals
Notify	Inform the MGC of the events happened in the MG
ServiceChange	Notify the MGC of any service related status change, e.g. a Termination is about to leave service or has returned to service; MG is available or restarted. The MGC may use ServiceChange to announce a handover to the MG or instruct the MG to take a Termination in or out of service.

Table 5.2 Megaco commands and functions

Megaco message structure is depicted in Figure 5.3, multiple transactions could be assembled in one single Megaco message and each transaction may contain several actions. One or more commands embedded with multiple descriptors could be inserted into a single action. The termination properties, event descriptors and other controllable parameters are modeled as descriptors, which can be defined in a package. This structure made Megaco a complicated protocol to operate and under which the efficiency is affected.

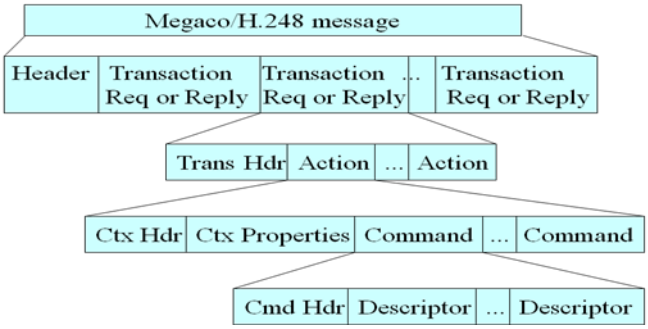


Figure 5.3 Megaco Message Structure

The Megaco/H.248 protocol is ideally designed as a media gateway control protocol. The defined packages are specifically focused on telephony applications. To support actuation, existing packages have to be extended and redefined. This is a complicated process and the complex structure of the protocol also limits the performance of high-efficiency demand applications. Again, like SMTP, it is not easy to interact with SIP.

5.2.4 Media Control Channel Framework

MCCF is specified in [8]. It defines mechanism of using SIP/SDP for establishing, using, terminating reliable connection (channel) to control an external server. The initial objective of this protocol was to replace Megaco in certain media control situation to simplify the procedure, however, it is not limited to that objective. It can be easily expanded to support the control of a general external server [8]. Three entities are defined in this framework: Control Client, Control Server and Control Channel as depicted in Figure 5.4.

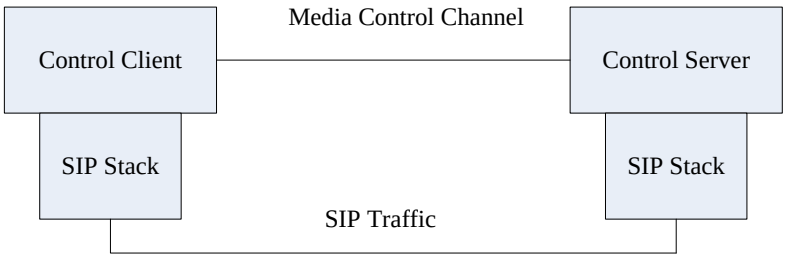


Figure 5.4 MCCF Overview

5.2.4.1 Entities and concepts

Few entities and concepts are introduced in MCCF which are described below,

Control Server: A Control Server is an entity that performs a service, such as media processing, on behalf of a Control Client. For example, a media server offers mixing; announcement; tone detection and generation; play and record services. The Control Server has a direct RTP relationship with the source or sink of the media flow.

Control Client: A Control Client is an entity that requests processing from a Control Server. The Control Client might not have any processing capabilities. For example, the Control Client may be a SIP Application Server (B2BUA) or other endpoint requesting manipulation of a third-party's media stream, which terminates on a media server acting in the role of a Control Server.

Control Channel: A Control Channel is a reliable connection between a Control Client and Control Server that is used to exchange framework messages.

Framework Message: A Framework Message is a message on a Control Channel that has a type corresponding to one of the Methods defined in [8]. A Framework message is often referred to by its method, such as a "CONTROL message".

Control Command: A Control Command is an application level request from a

Client to a Server. Control Commands are carried in the body of CONTROL messages. Control Commands are defined in separate specifications known as "Control Packages".

5.2.4.2 Framework messages

Four methods are defined in this framework: SYNC, CONTROL, REPORT, and K-ALIVE as listed in Table 5.3. They are defined in separate specifications known as "Control Packages". Currently, there are few control packages defined in [39, 41, 42].

MCCF Message	Function
SYNC	used to negotiate the timeout period for the control-channel keep alive mechanism, to allow clients and servers to learn the Control Packages that each supports and most important, to associate SIP dialog with control channel
CONTROL	used by the Control Client to pass control related information to a Control Server; also used as the event reporting mechanism
K-ALIVE	enables the control channel to be kept active during time of inactivity; also provides the ability for application level failure detection
REPORT	used by a Control Server when processing of a CONTROL Command extends beyond the Transaction-Timeout, as measured from the Client

Table 5.3 MCCF Messages

5.2.4.3 Media Control Channel establishment

SIP provides the ideal mechanism for establishing and maintaining control connections to external server components. The control connections can then be used to exchange explicit command/response interactions that allow for media control and associated command response results.

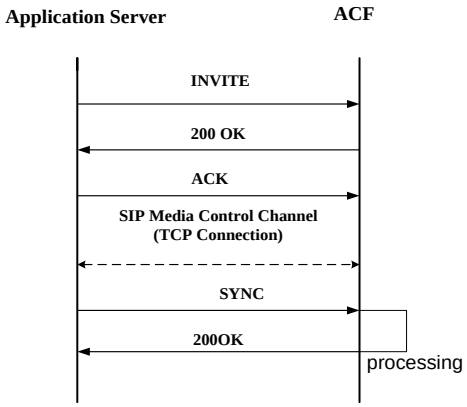


Figure 5.5 MCC establishment

As shown in Figure 5.5, the control client (AS) and the control server (ACF) establish a Media Control Channel (MCC) through a SIP dialog which usually originates in the AS. The AS generates a SIP INVITE message which contains in its SDP body information about the MCC that it wants to establish with the ACF. In the provided example (see Figure 5.6), the AS wants to actively open a new TCP connection, which on his side will be bound to port 5757. If the request reaches the ACF successfully (in time and no error), the ACF responds with its own offer by communicating to the AS the transport address to connect to in order to establish the TCP connection. In the provided example, the ACF will listen on port 7575. Once this negotiation is over, the AS can effectively connect to the ACF. The negotiation includes additional attributes, the most important is the ’cfw-id’ attribute, since it specifies the dialog id which will be subsequently referred to by both the AS and ACF, as specified in the core framework draft [8].

```
1. AS -> ACF (SIP INVITE)
-------------------------
INVITE sip:acf@ericsson.com:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.15:5060;\
branch=w3ef4qb-c6308z-4g06d3412h8sd224-1---c6308z-
;rport=5060
Max-Forwards: 70
Contact: sip:ApplicationServer@192.168.1.10:5060

To: <sip:acf@ericsson.com:5060>
From:
<sip:ApplicationServer@ericsson.com:5060>;tag=3456rt24
Call-ID: SGf3LKU1HDT3UnKkZjgzYTQwYmJlNjE5NTA4ZDQ1OGY.
CSeq: 1 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, UPDATE,
REGISTER
Content-Type: application/sdp
Content-Length: 263
v=0
o=lminiero 2890844526 2890842807 IN IP4 ericsson.com
s=MediaCtrl
c=IN IP4 ericsson.com
t=0 0
m=application 5757 TCP cfw
a=connection:new
a=setup:active
a=cfw-id:5feb6486792a
a=ctrl-package:actuation-camera/1.0
```

```
2. AS <- ACF (SIP 200 OK)
-------------------------
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.10:5060; \
branch=z9hG4bK-d8754z-9b07c8201c3aa510-1---d8754z-
;rport=5060
Contact: <sip:acf@ericsson.com:5060>
To: <sip:acf@ericsson.com:5060>;tag=499a5b74
From:
<sip:ApplicationServer@ericsson.com:5060>;tag=4354ec63
Call-ID: MDk2YTk1MDU3YmVkZjgzYTQwYmJlNjE5NTA4ZDQ1OGY.
CSeq: 1 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, UPDATE,
REGISTER
Content-Type: application/sdp
Content-Length: 296
v=0

o=lminiero 2890844526 2890842808 IN IP4 ericsson.com
s=MediaCtrl
c=IN IP4 ericsson.com
t=0 0
m=application 7575 TCP cfw
a=connection:new
a=setup:passive
a=cfw-id:5feb6486792a
a=ctrl-package:actuation-camera/1.0
```

```
3. AS -> ACF (CFW SYNC)
-----------------------
CFW 6e5e86f95609 SYNC
Dialog-ID: 5feb6486792a
Keep-Alive: 100
Packages: actuation-camera/1.0
```

```
4. AS <- ACF (CFW 200)
----------------------
CFW 6e5e86f95609 200
Keep-Alive: 100
Packages: actuation-camera/1.0
Supported: actuation-camera/1.0
```

Figure 5.6 Example of SIP dialog messages and MCCF SYNC message

5.2.4.4 CONTROL transactions

A CONTROL message is used by the Control Client to pass control related

information to a Control Server. It is also used as the event reporting mechanism in the control framework. Reporting events is simply another usage of the CONTROL message which is permitted to be sent in either direction between two participants in a session, carrying the appropriate payload for an event. The message is constructed in the same way as any standard framework CONTROL message, as shown in Figure 5.7. In most cases, a CONTROL message contains a message body. The explicit control command(s) contained in the message payload of a CONTROL message should be specified in a separate Control Package.

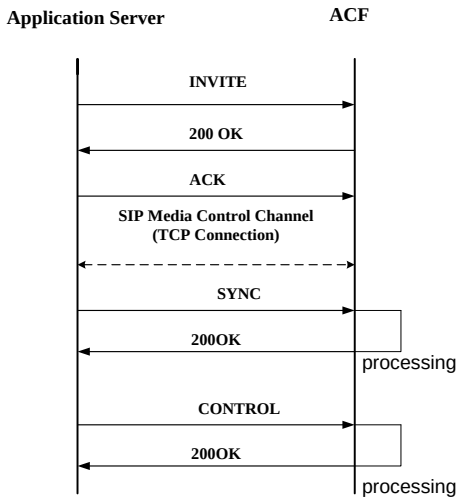


Figure 5.7 CONTROL transaction

A sample CONTROL message is shown in Figure 5.8.

```
1. AS -> ACF (CFW CONTROL)
----------------------
CFW 101fbbd62c35 CONTROL
Control-Package: actuation-camera/1.0
Content-Type: application/acml+xml
Content-Length: 178

<actuation-camera version=” 1.0”
xmlns=” urn:ericsson:xml:ns:actuation-camera”
<commandTuple>
    <action>
       <name>switch</name>
       <attribute>on</attribute>
    </action>
    <Objeccts>
       <object>
           <name>camera</name>
           <location>a.11.08</location>
       </object>
    </objects>
</commandTuple>
</actuation-camera>
```

```
2. AS <- ACF (CFW 200)
--------------------
CFW 101fbbd62c35 200
```

Figure 5.8 CONTROL message and response

5.2.4.5 Control packages

As aforementioned, the MCCF requires specific control packages to be designed to support expected service. In fact according to the definition in [8], the control packages are in the form of XML. In MCCF, multiple control packages can be embedded in one CONTROL message payload. Through this mechanism, MCCF can easily be extended to support new control services and logics.

5.2.5 Evaluation summary

The evaluation summary of the control protocols candidate is given in Table 5.4. From the evaluation, the MCCF seems the closest option in meeting the criteria of the actuation control protocol selection. It uses SIP/SDP for establishing, maintaining reliable sessions for controlling actuators. By using SIP as session control protocol,

Protocols / Criteria	SMTP	SIP+MSML	Megaco/H.248	MCCF
Criterion 1: Stateful	Yes	Yes	Yes	Yes
Criterion 2: Leverage IMS capabilities as much as possible	No	Yes	No	Yes
Criterion 3: Support conditional actuation requests	No	Extension required	Extension required	Extension required
Criterion 4: Support complex actuation requests	No	Extension required	Extension required	Extension required
Criterion 5: Transparency	No	No	No	Yes
Criterion 6: Simplicity	Yes	Yes	No	Yes

Table 5.4 Evaluation summary

it has inherent compatibility with IMS networks. Through properly designed information model (control packages), it can support actuation command provisioning well. It is simple and with some extensions it can meet all the criteria we set. Therefore we select it as the actuation control protocol. The remainder of this chapter will explain the information model and the actuation command delivery operations.

5.3 Actuation Control Information Model

In order to model actuation command properly, we have done some research on Presence Information Data Format (PIDF) [43] and extended PIDF [6]. Then we design our information model -ACML, which can be embedded into the content of the CONTROL message in MCCF. The structure of ACML is shown in Figure 5.9.

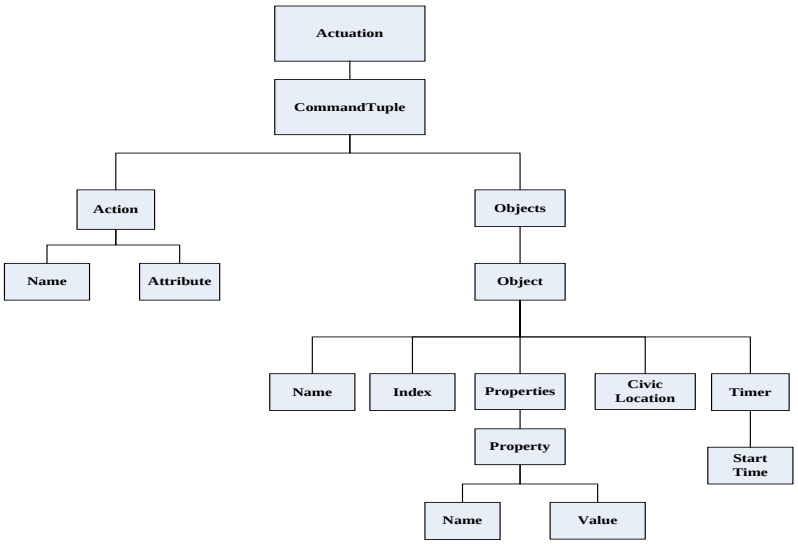


Figure 5.9 ACML Scheme

In ACML, each actuation command is organized as a tuple like the situation in PIDF. The structure elements are:

- Action-actuation command type, e.g. switch (on/off) for entity with switch, set (speed) (mandatory);
- Object-targets on which the actuation will be applied (mandatory);
- Location-specify particular place where the objects are located , e.g. place names(mandatory);
- Time-whether this command should be executed ‘immediately’, ‘some time later’, etc. (optional). When no time is specified the command is executed immediately.

A simple example of actuation control package is given in Figure 5.10. The actuation control package is carried by the CONTROL messages.

```
<?xml version="1.0"?>
<actuation  xmlns='http://encs.concordia.ca/tse/acim'>
<commandtuple>
      <action>
          <name>switch</name>
        <attribute>on</attribute>
      </action>
      <objects>
          <object>
                <name>camera</name>
                <index>3rd</index>
                <properties>
                      <property>
                            <name>power</name>
                            <value></value>
                      </property>
                 </properties>
                 <location>building A room118</location>
                 <timer>
                      <starttime> Nov 25, 2009,10:00</starttime>
                 </timer>
           </object>
       </objects>
< /commandtuple>
</actuation>
```

Figure 5.10 An example of a control package in ACML

In the proposed architecture, interfaces Aa and Ag use SIP MCCF. Aa will carry aggregated actuation commands which will be decomposed by ACF into commands targeting a specific AN. For instance, an AS may request to switch on lights in all rooms in a given building, and this will be done in a single request through the Aa interface; when ACF receives the message it will decompose the request into commands targeting a specific AN according to pre-established mapping rules.

5.4 Actuation control command delivery and processing

The delivery and processing of an actuation control command based on the selected protocol is as follows:

Upon reception of a request, the application server will initiate the establishment of the MCC with the ACF using an INVITE message (shown in Figure 5.11). When the channel is established, the AS sends the SYNC message before sending actuation control command (simply command, hereafter) to confirm the control packages it supports. When the CONTROL message reaches the ACF, it extracts the command, processes it

and initiates the establishment of a separate channel with a specific WAG. Similarly, once the channel is established, after sending out the SYNC message, a CONTROL message carrying the command is sent to the WAG. The WAG extracts the command and sends it to a specific actuator. The results or actuation feedbacks will be returned in the opposite direction via response messages. A channel is set for the lifetime of the application.

Our architecture offers the possibility of delivering continuously captured data to applications. During the lifetime of the media control channel, the applications could send sequences of commands and request real-time data to be collected by actuators from the scene. The collected data belongs to the media plane and we use RTP as the transport protocol to transfer the captured data.

At the end, the application sends BYE message to the ACF to terminate the media control session. The ACF does the same as the WAG. Finally, after receiving confirmation from actuators, the WAG responds to the ACF with 200 (OK) message and the ACF does the same with the application. The application session ends.

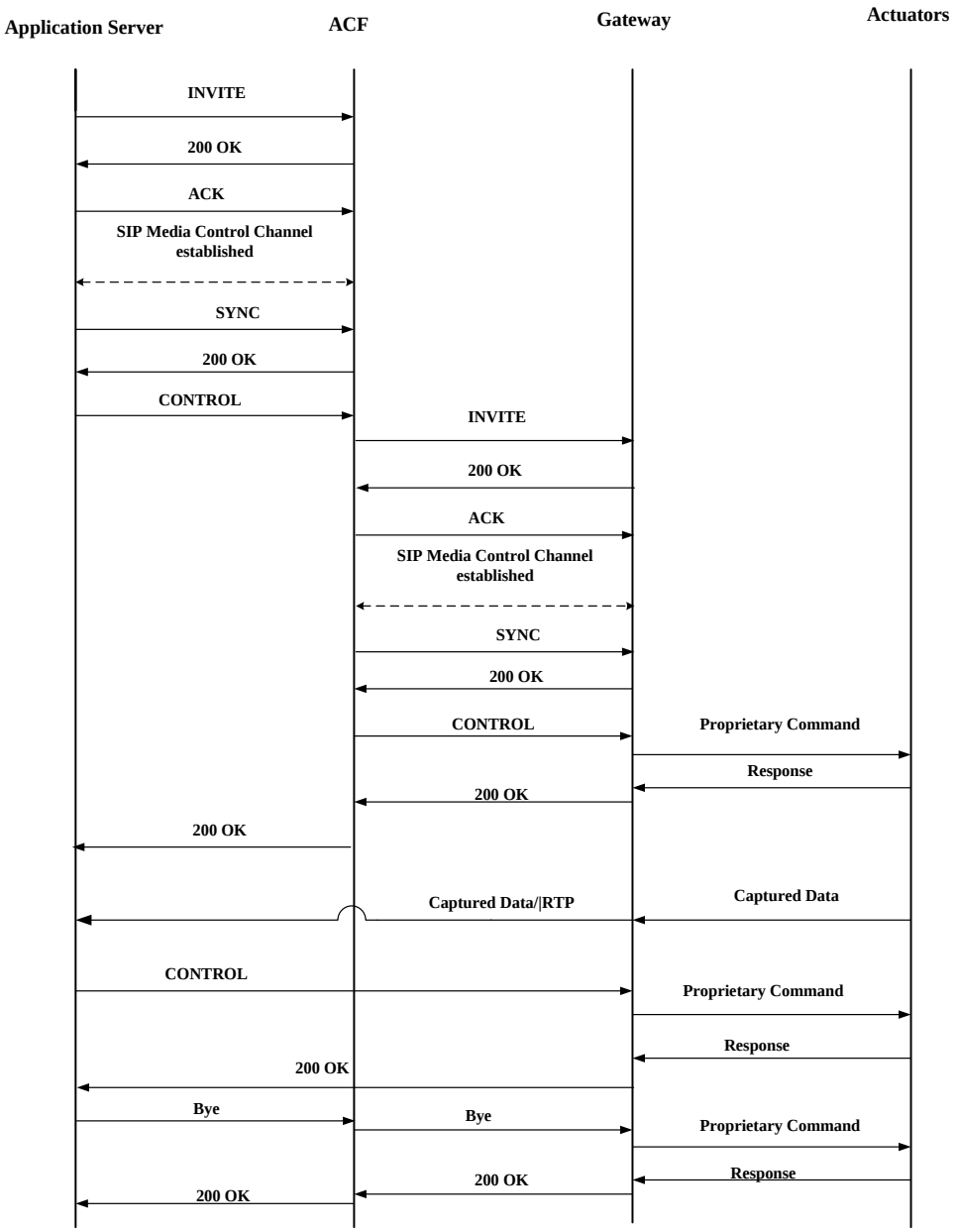


Figure 5.11 Call flow of actuation command delivering and processing

5.5 Conclusions

In this chapter, a set of criteria for actuation control protocol selection is carefully chosen. A number of potential actuation control protocols have been described and evaluated with regard to the selection criteria. Based on the evaluation, MCCF protocol is selected as our actuation control protocol. The information model is discussed and finally the detailed data flow of actuation command delivery and processing is

determined. The next chapter will present a proof of concept prototype implementation and experiments to verify the proposed architecture.

Chapter 6

Prototype Implementation and Experiments

This chapter describes the implementation of key architecture components. We also describe the proof of concept prototype application demonstrating the usage of the architecture. Experiment results are briefly discussed at the end.

6.1 Architecture implementation

In this section we discuss the implementation of ACF, WAG and the implementation environment.

6.1.1 Implementation of the Actuation Control Function

For the implementation, we chose Java SIP Servlet API [44] and Java Socket API [45] toolkits to realize the function blocks of ACF. All the functions of ACF have been implemented except actuation support functions. In this implementation, the function blocks are structured into Java classes. Figure 6.1 (generated by an Eclipse plug-in UML tool - ObjectAID [46]) shows the implemented Java classes.

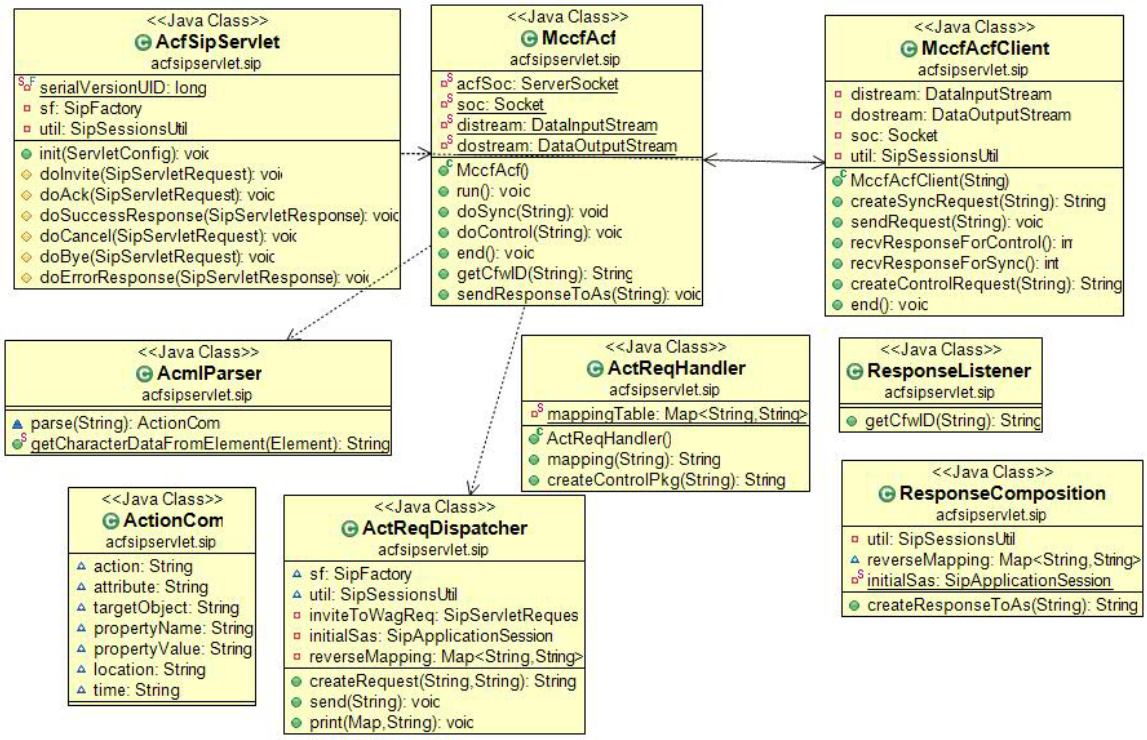


Figure 6.1 Class diagram for the implemented ACF

The main application logic of ACF is implemented by the 'AcfSipServlet' class. It deals with SIP signaling related session control. The 'ActReqHandler' and 'ActReqDispatcher' classes will be triggered respectively by the 'AcfSipServlet' class according to the message received.

The MCCF is implemented by a separate class 'MccfAcf'. It deals with all the MCCF interactions. After the MCC is established, the 'MccfAcf' is ready to receive actuation request from application servers. The message content will be analyzed by 'AcmlParser' class and the result will be delivered to class 'ActReqHandler' in the form of 'ActionCom' class which is a data class. After being processed by the 'ActReqHandler', new requests addressed towards specific WAG will be generated and sent by the 'ActReqDispatcher' class. The class 'MccfAcfClient' will be subsequently triggered to deal with the MCCF interactions with WAGs. In this case, the ACF acts as a

control client and sends actuation requests to specific WAGs.

The remaining ‘ResponseListener’ and ‘ResponseComposition’ classes implement the functions exactly the same as in the design described in section 4.2. The ‘ResponseListener’ class receives individual responses and the ‘ResponseComposition’ class gathers all relevant responses to an existing actuation request by an application and composes them into a complete response according to a reverse mapping mechanism.

6.1.2 Implementation of Wireless Actuator Gateway

Similarly for the implementation of the WAG, we also chose Java SIP Servlet API to realize the SIP interactions between the ACF and the WAG. All the functions of WAG gateway have been implemented except capability management and support functions. The actuation arbitration function is simplified and implemented by the ‘ActScheduler’ class using First come-First service policy.

All the function blocks of WAG gateway are structured into Java classes. Figure 6.2 shows the implemented Java classes.

In class ‘ActScheduler’, we implemented one proprietary interface to an actuator network (Webots Simulated Robots) based on the Controller API provided by Webots [47]. The interface is identical to the one on real robot-e-puk (shown in Figure 2.3(a)).

The ‘JpegImagesToMovie’ class transforms a series of static images captured by camera actuator on a simulated robot into a QuickTime movie. This is because the simulation environment can only store a series of jpeg images instead of media streams.

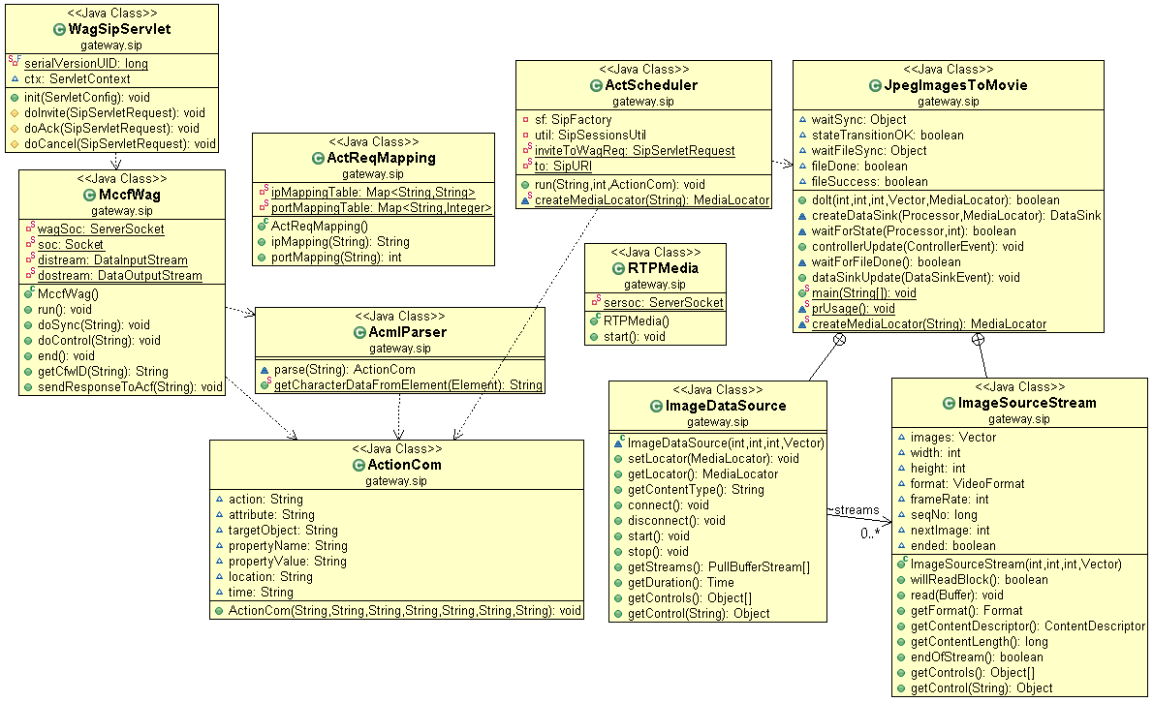


Figure 6.2 WAG Java Classes diagram

6.1.3 Implementation of actuation application

In order to use actuation service provided by the integrated architecture, the only change to the existing IMS application structure is to add MCCF support. We use Java SIP Servlet API to implement the SIP signaling interactions. Same as in the ACF implementation, Java Socket API is used to implement the MCCF. Figure 6.3 shows the simplified class diagram of actuation application.

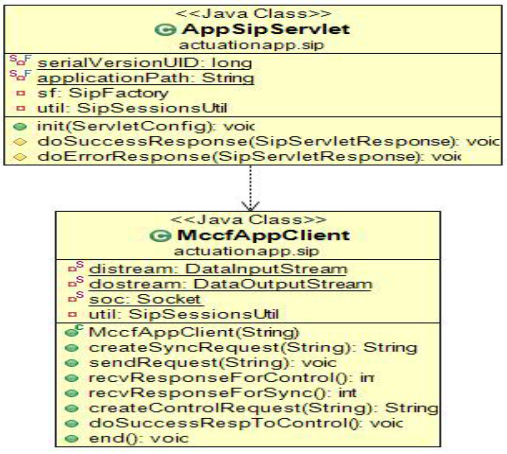


Figure 6.3 Simplified actuation application Java class

6.1.4 The implementation environment

In this section, several software tools used in the implementation will be described.

6.1.4.1 SIP Servlet API

A SIP servlet [44] is a Java-based application component which is managed by a SIP servlet container that performs SIP signaling. Like other Java-based components, servlets are platform independent Java classes that are compiled to platform neutral byte code that can be loaded dynamically into and run by a Java-enabled SIP application server. Containers, sometimes called servlet engines, are server extensions that provide servlet functionality. Servlets interact with (SIP) clients by exchanging request and response messages through the servlet container. Figure 6.4 provides a high-level illustration of the API layer's role. In our implementation, we use SailFin server [48] as the SIP Servlet Container.

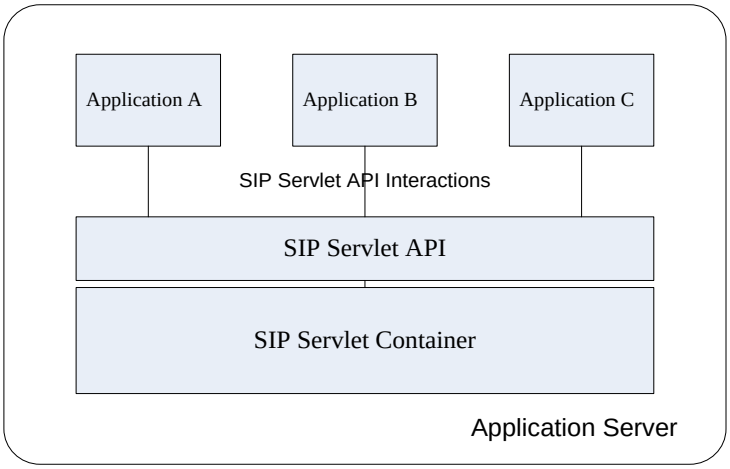


Figure 6.4 SIP Servlet API role

The new version of SIP Servlet API standard is defined in JSR289, which supports

the RFC specifications listed in Table 6.1.

SIP Specification	SIP Feature
RFC 3261	SIP Session Initiation protocol core features
RFC 2976	INFO method
RFC 3262	Support reliability of provisional (1xx) responses
RFC 3265	SIP Event Notification Framework
RFC 3428	MESSAGE method
RFC 3311	UPDATE method
RFC 3515	REFER method
RFC 3903	PUBLISH method

Table 6.1 Supported SIP specifications in SIP Servlet API

6.1.4.2 Ericsson Service Development Studio

The Ericsson service development studio (SDS) [49] offers a comprehensive developing environment for design, implementation and end-to-end testing of new convergent all-IP (IMS) value added services.

SDS runs in a PC Windows environment and supports the creation of both client and server sides IMS applications using built-in IMS emulators. SDS provides high-level APIs to hide device and network complexity and includes a multitude of templates and wizards to help the developer shorten project lead times.

The SDS consists of two client side components, the IMS Client Platform (ICP) or IMS JME Client Utility (IJCU) [50] and the developed IMS Device Client. On the server side, applications can be built on an open architecture based on Java with support for SIP/HTTP Servlets using SailFin as the default container. In addition, with the SDS, developers can use the high-level APIs to control and access advanced capabilities such as Presence and Group Management (PGM), Push-to-Talk (PTT), IMS Messaging

(IMS-M), Internet Protocol Television (IPTV) [51].

Figure 6.5 [49] illustrates a high-level view of the SDS functionality, components, and steps in the design, debugging, testing, and deployment.

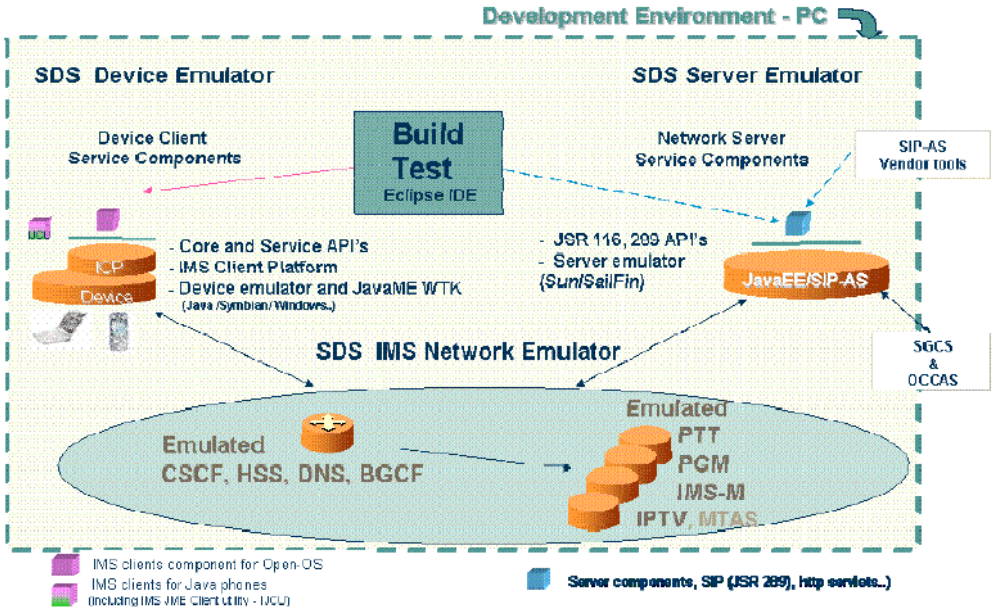


Figure 6.5 SDS development and emulation environment

6.1.4.3 Webots Controller API

Webots [52] is a professional mobile robot simulation software package developed by Cyberbotics Ltd. It offers a prototyping environment that allows the user to create mobile robots equipped with a number of sensor and actuator devices, such as distance sensors, drive wheels, cameras, servos, touch sensors, grippers, emitters and receivers. That is why we choose this environment to implement our proof of concept prototype. The camera actuator is simulated by Webots. The WAG communicates with the actuators via Webots controller API. One example of the controller API is shown in Figure 6.6. Webots contains a large number of robot models and a number of interfaces to real

mobile robots, once the simulated robot behaves as expected, control programs can be downloaded to a real robot like e-puck and Aibo. We use its camera actuator simulations in this research.

A Webots simulation is composed of three components:

1) A Webots world file that define one or more 3D robot and their environment.

2) Controller programs for the above robots.

3) An optional Supervisor.

We use the first two features to develop our application. In fact, the WAG is utilizing the controller API to send actuation commands to the simulated robots, like switch on a camera, capture images, move robots forward and backward or turn around.

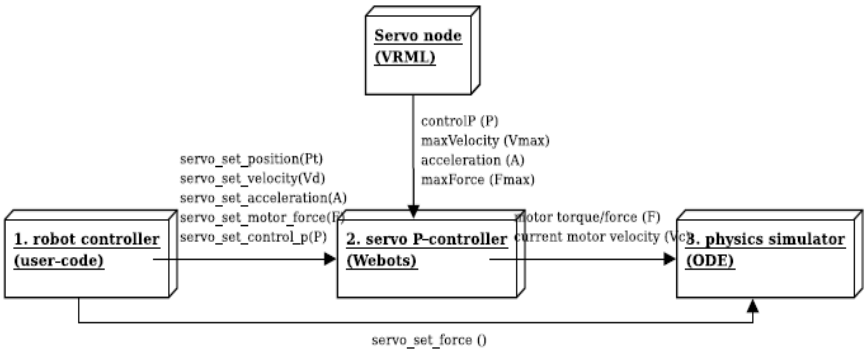


Figure 6.6 Programming model of controller in Webots simulation environment

6.2 Prototype application

The integrated architecture provides a framework for quick development of actuation services to IMS service developers. Through the gateway (WAG) and the ACF, this architecture abstracts away all the device specific and lower level details and physical complexities for interacting with actuators. Through this architecture, the

service developers no longer need to deal with the proprietary factors of heterogeneous actuators. They can implement multimedia services with actuation capabilities in a standard and persistent way. Figure 6.7 shows the deployment model of the proposed architecture.

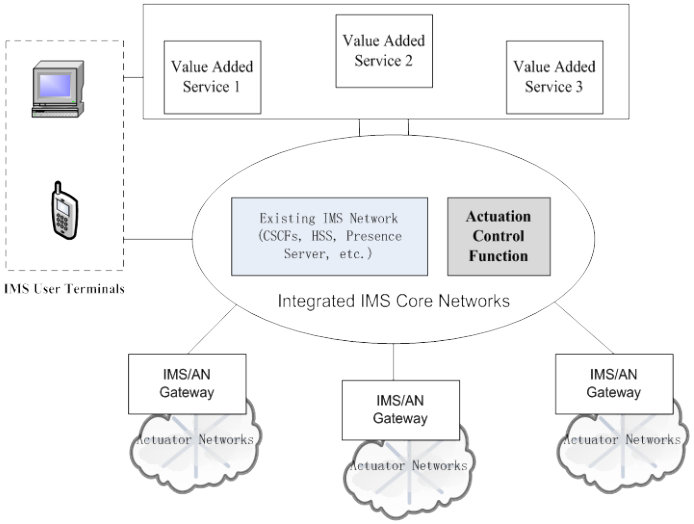


Figure 6.7 Prototype application deployment structure

The design and implementation of the prototype application will demonstrate how this architecture can be used to create attractive services to customers in a convenient and standard way. In the following sections, a motivating scenario will be developed and implemented using the integrated architecture to demonstrate proof of concept.

6.2.1 Environment monitoring prototype scenario

Environment monitoring cannot always be performed with humans on the scene. This is the case for volcano surveillance or earthquakes, for instance. Robots equipped with sensors and actuators can help in such cases. An application scenario can be as follows: robots with camera, light sensors and differential wheel devices, are remotely deployed on an earthquake zone. Upon the receiving an end user request, the application

will instruct the robot to turn on cameras through switch actuator, move robot using motor actuators and take pictures of the scene. The robot can then send back the pictures to the application. The pictures can be sent to an end user and displayed. The end-user can initiate a conference call via IMS where the pictures are shared and discussed and rescue staff could be dispatched to the appropriate locations. Robots can be instructed to move around and take pictures of targeted zones.

6.2.2 Prototype design and implementation

We leveraged Cyberbotics' Webots and Ericsson's Service Development Studio (SDS) to develop the proof of concept prototype of the proposed architecture. The camera actuator in the scenario is simulated by Webots. The WAG communicates with the actuators via Webots controller API as mentioned in 6.1. Figure 6.8 shows the prototype components.

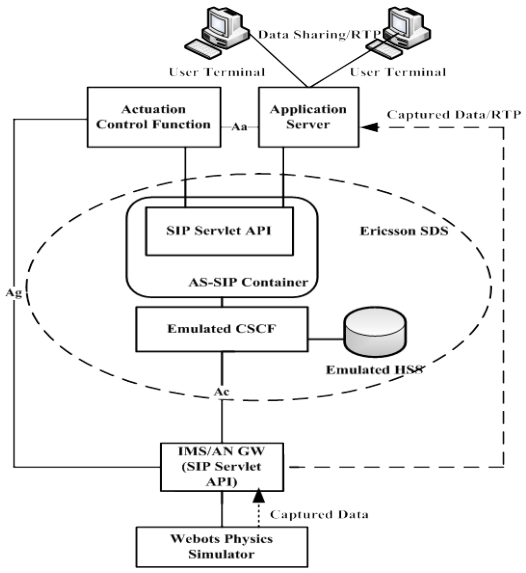


Figure 6.8 Implementation of the prototype components

At the user side, there are two parts: One part is a web application which presents user with a web page showing the parameters related to the actuation requests. The user

can fill in the fields and send the request to actuation application triggering the actuations. The other part is the X-Lite [53] which simulates IMS user terminals and is able to register with IMS, interacts with the application using SIP. The server side consists of a conference application and an actuation application which is an HTTP server and SIP server. The conference application was developed by a PhD student in our lab. It is triggered by the actuation application. The prototype actuation application was implemented as a set of java classes which is shown in Figure 6.9.

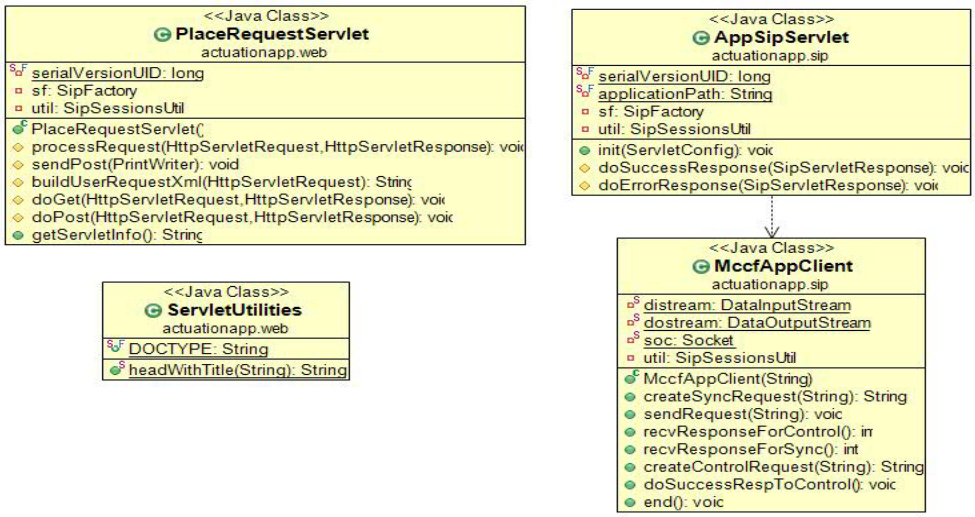


Figure 6.9 Java class diagram of the actuation application prototype

6.2.3 Setup and work flow

The setup consists of three laptops and one desktop. One laptop runs X-Lite simulation and the second one hosts another X-Lite simulation, the SDS, the conference and actuation applications in parallel. The ACF is deployed on the third laptop, while the robots simulation and the WAG are deployed on the desktop. The following interactions are successfully tested:

1) two end users registered with IMS,

2) one user initiated a request to the application requesting to turn on the simulated camera,

3) the application established a dedicated control channel for carrying the actuation requests to the ACF,

4) a control channel between the ACF and the WAG was then established and an actuation request was delivered to the latter,

5) the WAG issued an actuation command to the camera actuator through the controller interface method Camera.enable(),

6) the camera captured the image and sent it to the WAG,

7) the image file was sent back to the application using RTP,

8) The application acted as B2BUA and triggered the establishment of a conference call using SIP INVITE message between two end users. The subsequent captured images were continuously transferred to end users until the termination of the conference call.

The detailed information flow diagram is shown in Figure 6.10.

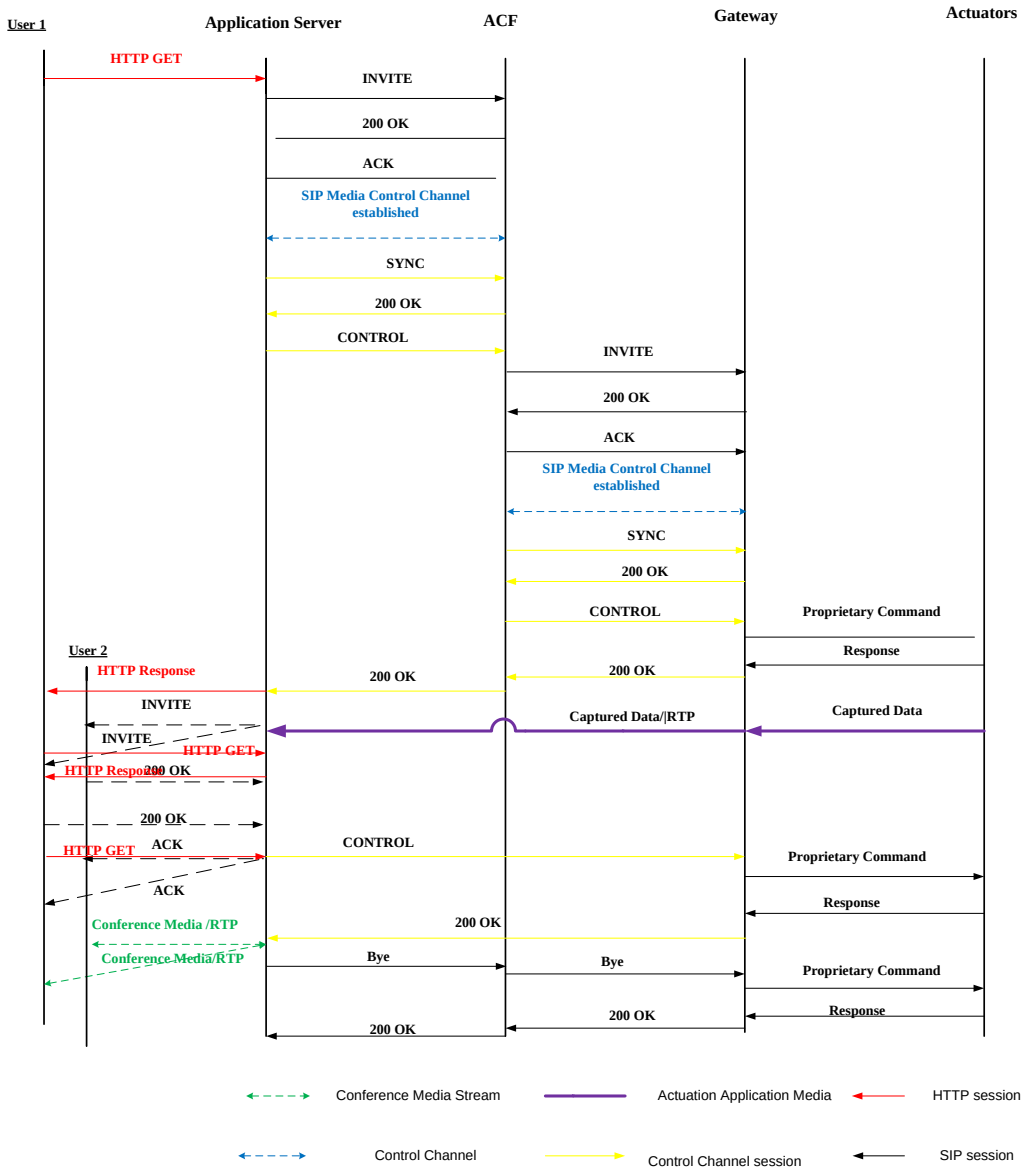
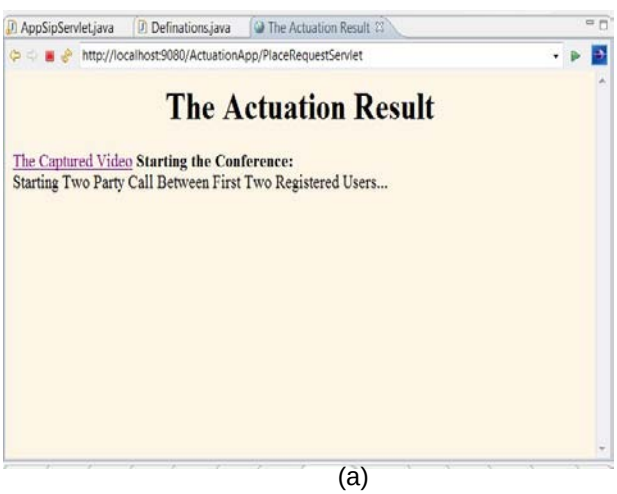


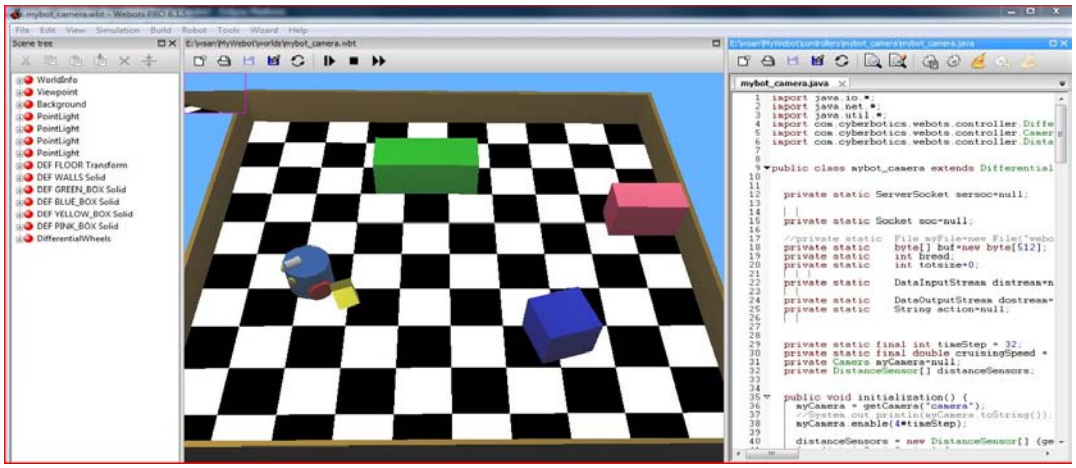
Figure 6.10 The prototype information flow

Few snapshots of the prototype running results are shown in Figure 6.11. The small square window frame in Figure 6.11 (c) is the camera lens scope which set the image size, this is the weak point of the simulation environment. Figure 6.11 (a) shows the webpage when the captured video arrives at the application and immediately before the conference begins and Figure 6.11 (b) shows that the conference has been established.



(a)

(b)



(c)

Figure 6.11 Snapshots of prototype running result: (a) webpage shown on end user terminal after the image is ready; (b) the simulated mobile phone joins the conference; (c) the simulated robot capturing images

6.3 Experiments and results

In this section, we conduct experiments based on the prototype application. The purpose of the experiments is to collect data on end-to-end actuation delays (actuation delay hereafter) and calculate the average actuation delay.

6.3.1 Actuation delay

The QoS requirements for the actuation service depend on the nature of the

application. In most cases, applications requesting actuation services are time sensitive and may impose constraints on the response time. The end-to-end actuation delay is probably one of the key performance metrics for the integrated architecture. It is defined as the span of time from the moment an actuation request is sent by an application to the moment an actuation response carrying the actuation results is received.

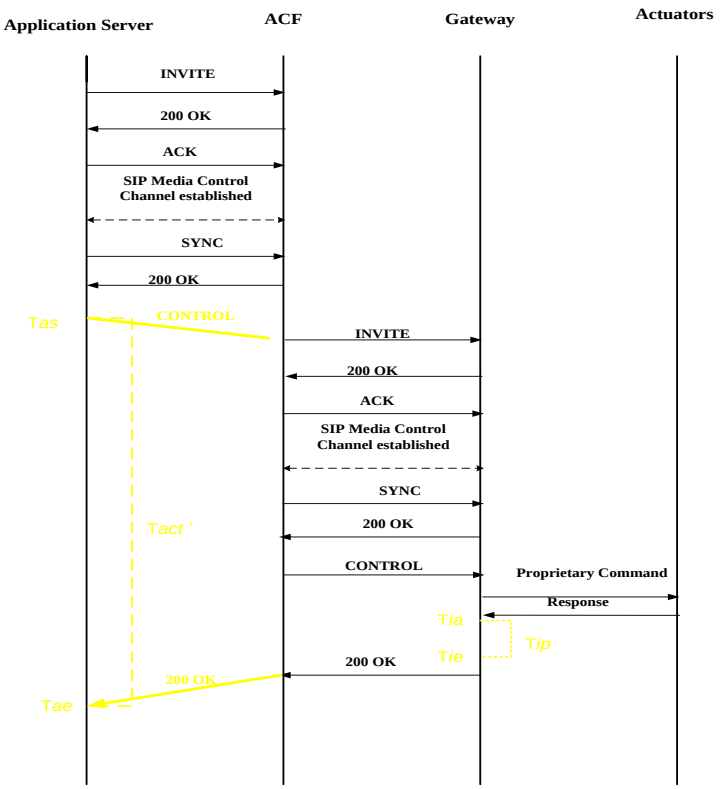


Figure 6.12 End-to-end actuation delay calculations

As shown in Figure 6.12, the actuation delay is calculated in the following way: We denote the actuation delay by *Tact*. It is calculated as the time between the moment the application send out the MCCF CONTROL messages toward the ACF (*Tas*) and the moment a relevant response is received from the ACF (*Tae*). It includes the delay between the ACF and the WAG.

6.3.1.2 Setup of the experiment environment

The experimental setup consists of: Webots 6.5 robot simulation which acts as the actuator network, Ericsson SDS 4.1 simulates the IMS environment including the HSS and CSCFs and the integrated SailFin server as the SIP container. The actuation application, ACF and WAG are deployed over SDS. The hardware specifications are shown in Table 6.2.

Computers	Hardware configuration	Software
Laptop 1	Athlon Dual-Core, 1.9GHz/4G RAM/ 802.11a/g Wi-Fi	Application Server: Actuation Application, Conferencing; IMS simulation: Ericsson SDS4.1, NetBeans 6.5, Windows Vista 64
Laptop2	Intel Dual-Core, 1.8GHz/3G RAM/ 802.11a/g Wi-Fi	ACF, X-Lite cell phone simulator Ericsson SDS4.1, Windows XP
Desktop	Intel Quad-Core, 1.9GHz/4G RAM/ 802.11a/g Wi-Fi	WAG, Simulated Robots with camera device, X-Lite cell phone simulator Ericsson SDS4.1, Webots 6.5, Windows Vista

Table 6.2 Test setup hardware configuration

6.3.1.3 Experiment results

Fifteen experiments have been performed for the prototype application. Table 6.3 shows the results: the actuation delays (Tact). The Java Date object is used to acquire the time moment in milliseconds.

Due to the limitations of the Webots simulation environment, we can only capture

static JPEG image. We have to transform the static JPEG image into Quicktime video stream which will be transmitted to the application via RTP. This took a lot of time. In our experiments, the image processing time is around 27s.

Experiment Runs	Tas(ms)	Tae(ms)	Tact(s)
1	1286250955969	1286250986850	30.88
2	1286504278943	1286504309552	30.61
3	1286547253595	1286547284549	30.95
4	1286556268967	1286556299662	30.70
5	1286569174402	1286569205264	30.86
6	1286589114094	1286589145719	31.63
7	1286572272947	1286572303842	30.90
8	1286572706334	1286572737122	30.79
9	1286590318666	1286590348954	30.29
10	1286591113068	1286591143849	30.78
11	1286591548192	1286591578462	30.27
12	1286592055698	1286592085736	30.04
13	1286592826389	1286592856690	30.30
14	1286593314441	1286593344770	30.33
15	1286598113202	1286598143288	30.09

Table 6.3 Actuation delay results

From the experiments above, we can see that for the prototype scenario, the average actuation delay is about 30 seconds. If we remove the impact of the image processing, the average actuation delay will be about 3 seconds. This result shows the performance of the prototype alike application over the integrated architecture is acceptable.

6.4 Conclusions

In this chapter we presented the implementation of the architecture components and discussed the implementation environment. This integrated architecture provides service developers with a convenient and efficient way of building new actuation

enabled value added services. A proof of concept prototype application has been implemented for the environment monitoring application scenario.

As a key factor affecting the performance, the actuation delay is evaluated experimentally. The result shows that the performance of the prototype alike application over the integrated architecture is acceptable. However, a more realistic set up or more formal analysis study of the architecture needs to be carried in the future.

Chapter 7

Conclusions and Future Work

In this chapter, we summarize the contribution of the thesis and discuss potential future work.

7.1 Summary of contributions

Research on WSANs has been very active in recent years. The main motivation is the possibility of novel applications. IMS alone enables attractive multimedia applications. The integration of the heterogeneous actuators with the IMS will enable more. However, this integration is a challenging task. This thesis focuses on the integration of actuation capabilities with the IMS. The main contributions of this thesis are as follows.

- We examined existing solutions related to integration of WSAN and other networks including IMS and Internet. In order to conduct the evaluation, a set of requirements has been derived for the integration of AN with IMS based on the application scenarios we are interested in. This evaluation concluded that none of the existing solutions fulfills all the requirements. We therefore decided to develop a new architecture based as much as possible on existing standards.
- We proposed an architecture for the integration. The architecture includes two key components: the ACF and the WAG. In this architecture, two levels of abstraction are required based on the derived requirements. The introduction

of the ACF is motivated by the requirement of abstracting away the lower level details of actuation from the applications and other IMS entities. We introduced the WAG to make ANs accessible to the external world.

- Existing command carrying protocols and information models have been evaluated with respect to a set of derived requirements. Based on this evaluation, MCCF has been selected for actuation command delivery. We designed a new information model - ACML to abstract the actuation triggering instructions.
- Finally, as a proof of concept, a prototype has been developed for the environment monitoring scenario. This prototype demonstrates the possibility of new value added services with the integrated architecture. The key components of the proposed architecture have been implemented. The session control and signaling protocols for exchanging actuation commands and transmitting media stream have also been implemented. The end-to-end actuation delay is evaluated through experiments with the prototype. The result shows that the performance of the prototype alike application over the integrated architecture is acceptable.

7.2 Future work

The research on integrating actuators with existing networks is still evolving. The future work would be to implement the remaining features of the integrated architecture and perform an overall performance evaluation and a thorough analysis.

In our prototype, the actuation is triggered by the end user intervention. In the

future we plan to introduce sensor based triggering mechanism into the architecture to increase the efficiency and enrich the application domain.

We also assumed a simplified business model which is: one operator owns the IMS and the actuator networks, the WAGs are preconfigured and known to the ACF. As future work, it will be interesting to relax this constraint and extend the architecture.

It will also be interesting to work with “real open” actuators when they become available.

REFERENCES

[1] 3GPP. TS23.228. “IP Multimedia Subsystem (IMS); Stage 2 (release 8)”, Sep. 2009

[2] Rosenberg et al., “SIP: Session Initiation Protocol”, IETF RFC 3261, June 2002

[3] Wikipedia, “Actuator”, available at http://en.wikipedia.org/wiki/Actuator

[4] I.F. Akyildiz, I.H. Kasimoglu, “Wireless Sensor and Actor Networks: Research Challenges”, Ad Hoc Networks 2 (2004), pp. 351–367

[5] J. Schiff, et al., “Actuator Networks for Navigating an Unmonitored Mobile Robot”, IEEE Conference on Automation Science and Engineering (CASE), August 2008

[6] M. El Barachi, A. Kadiwal, R. Glitho, F. Khendek, and R. Dssouli, “A Presence-based Architecture for the Integration of the Sensing Capabilities of Wireless Sensor Networks in the IP Multimedia Subsystem”, Proceedings of IEEE WCNC, April 2008, pp. 3116-3121

[7] SENSEI Consortium, “The SENSEI Real World Internet Architecture”, March 2010

[8] C. Boulton, et al., “Media Control Channel Framework”, IETF Internet Draft < draft-ietf-mediactrl-sip-control-framework-11>, October 2009

[9] W3C, “Extensible Markup Language (XML) 1.0 (Third Edition)”, W3C Recommendation, Feb. 2004

[10] E-puck project, available online at http://www.e-puck.org/

[11] Mini-robot Research, available online at http://sandia.gov/media/NewsRel/NR2001/minirobot.htm

[12] Military robot, available online at http://en.wikippedia.org/wiki/Military_robot

[13] G. Camarillo, M. A. Garcia-Martin, “The 3G IP Multimedia Subsystem (IMS) Merging the Internet and the Cellular Worlds”, John Wiley & Sons, 2008

[14] R. Hou, R. Glitho, F. Khendek, and M. Ali, “Integrating Wireless Actuation Capabilities with the 3GPP IP Multimedia Subsystem for Enhanced Multimedia Services”, IEEE PIMR 2010, Sep. 2010

[15] H.248.1, “Gateway control protocol: Version 3”, ITU-T Recommendation, Sep. 2002

[16] Syed A.Ahson, and Mohammad Ilyas, “IP Multimedia Subsystem (IMS) Handbook”, CPC Press, 2009

[17] H. Schulzrinne, et al., “The tel URI for Telephone Numbers”, IETF RFC 3966, Dec. 2004

[18] 3GPP TS 23.218, “IP multimedia session handling: IM call model; Stage 2 (Release 9)”, June 2010

[19] 3GPP TS 24.141, “Presence service using the IP multimedia (IM) core network (CN) subsystem; Stage 3 (Release 9)”, Dec. 2009

[20] M. Handley, et al., “SDP: Session Description Protocol”, IETF RFC 4566, July 2006

[21] R. Stewart, et al., “Stream Control Transmission Protocol”, IETF RFC 2960, October 2000

[22] J.Kuthan, D. Sisalem, “SIP: more than you ever wanted to know about”, March 2007

[23] R. Fielding, et al., “Hypertext Transfer Protocol - HTTP/1.1”, IETF RFC 2616, June 1999

[24] T. Berners-Lee, et al., “Uniform Resource Identifier (URI): Generic Syntax”, IETF RFC 3986, Jan. 2005

[25] B. Campbell, et al., “Control of Service Context using SIP Request-URI”, IETF RFC 3087, April 2001

[26] A. B. Roach, “Session Initiation Protocol (SIP)-Specific Event Notification”, IETF RFC 3265, June 2002

[27] B. Campbell, Ed. , “Session Initiation Protocol (SIP) Extension for Instant Messaging”, IETF RFC 3428, Dec. 2002

[28] A. Niemi, Ed., “Session Initiation Protocol (SIP) Extension for Event State Publication”, IETF RFC 3903, October 2004

[29] F. Carrez, Ed., “D.3.2 –Reference Architecture”. SENSEI, Public Deliverable D.3.2, 2009, available online at http://www.sensei-project.eu/

[30] R. Gold, Ed, “D.3.1-State of the art – Sensor frameworks and future Internet”, SENSEI Public Deliverable D.3.1, 2008, available online at http://www.sensei-project.eu/

[31] Mirco Rossi, et al., “D.2.3-Components for Context Modeling and Interfaces”, SENSEI Public Deliverable D.2.3, 2009, available online at http://www.sensei-project.eu/

[32] N. Kushalnagar, et al, “IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statements and Goals”, IETF RFC 4919, Aug. 2007

[33] S. Deering et al, “Internet Protocol, Version 6 (IPv6) Specification”, RFC 2460, IETF, Dec. 1998

[34] A. Gluhak et al. “e-SENSE Reference Model for Sensor Networks in B3G Mobile Communication Systems”, Information society technologies (IST), 2006.

[35] C. Groves, et al., “Gateway Control Protocol Version 1”, IETF RFC 3525, June 2003

[36] J. Klensin, “Simple Mail Transfer Protocol”, IETF RFC 5321 October 2008

[37] A. Saleem, et al., “Media Server Markup Language (MSML)”, IETF RFC5707, Feb. 2010

[38] S. Donovan, “The SIP INFO Method”, IETF RFC2796, October 2000

[39] S. McGlashan, et al., “An Interactive Voice Response (IVR) Control Package for the Media Control Channel Framework”, IETF Internet Draft< draft-ietf-mediactrl-ivr-control-package- 09>, Nov. 2010

[40] H. Schulzrinne, et al., “RTP: A Transport Protocol for Real-Time Applications”, IETF RFC3550, July 2003

[41] S. McGlashan, et al., “A Mixer Control Package for the Media Control Channel Framework”, IETF Internet Draft < draft-ietf-mediactrl-mixer-control-package-11>, February 2010

[42] A. Amirante, et al., “Media Control Channel Framework (CFW) Call Flow Examples”, IETF Internet Draft< draft-ietf-mediactrl-call-flows-03>, Feb. 2010

[43] H. Schulzrinne, et al., “Extensions to the Presence Information Data Format (PIDF)”, IETF RFC 4480, July 2006

[44] Java Community Process, “SIP Servlet API Specification, Version 1.1”, JSR 289, August 2008

[45] Elliotte Rusty Harold, “JAVA Network Programming, Third Edition”, O'Reilly, November 2004.

[46] The ObjectAid UML Explorer for Eclipse, online at http://www.objectaid. com/

[47] Cyberbotics Ltd., “Webots Reference Manual”, March 2010

[48] SailFin Project, available online at: https://sailfin.dev.java.net/

[49] Ericsson AB., “Service Development Studio (SDS) 4.1 Tutorial”, February2009

[50] Ericsson AB., “Service Development Studio (SDS) 4.1 Technical Description”, February, 2009

[51] Ericsson AB., “Service Development Studio (SDS) 4.1 Developer’s Guide”, February, 2009

[52] Cyberbotics Ltd., “Webots User Guide”, June 2009

[53] X-Lite, available online at http://www.counterpath.com/x-lite.html