Department of Computing Science and Mathematics
University of Stirling

An Online Environmental Approach to Service Interaction Management in Home Automation

Michael E. J. Wilson

Technical Report CSM-171

ISSN 1460-9673

November 2006

Department of Computing Science and Mathematics
University of Stirling

An Online Environmental Approach to Service Interaction Management in Home Automation

Michael E. J. Wilson

Department of Computing Science and Mathematics
University of Stirling
Stirling FK9 4LA, Scotland

Telephone +44-786-467421, Facsimile +44-786-464551
Email mew@cs.stir.ac.uk

Technical Report CSM-171

ISSN 1460-9673

November 2006

Abstract

Home automation is maturing with the increased deployment of networks and intelligent devices in the home. Along with new protocols and devices, new software services will emerge and work together releasing the full potential of networked consumer devices. Services may include home security, climate control or entertainment. With such extensive interworking the phenomenon known as service interaction, or feature interaction, appears. The problem occurs when services interfere with one another causing unexpected or undesirable outcomes.

The main goal of this work is to detect undesired interactions between devices and services while allowing positive interactions between services and devices. If the interaction is negative, the approach should be able to handle it in an appropriate way.

Being able to carry out interaction detection in the home poses certain challenges. Firstly, the devices and services are provided by a number of vendors and will be using a variety of protocols. Secondly, the configuration will not be fixed, the network will change as devices join and leave. Services may also change and adapt to user needs and to devices available at runtime. The developed approach is able to work with such challenges.

Since the goal of the automated home is to make life simpler for the occupant, the approach should require minimal user intervention.

With the above goals, an approach was developed which tackles the problem. Whereas previous approaches solving service interaction have focused on the service, the technique presented here concentrates on the devices and their surrounds, as some interactions occur through conflicting effects on the environment. The approach introduces the concept of environmental variables. A variable may be room temperature, movement or perhaps light. Drawing inspiration from the Operating Systems domain, locks are used to control access to the devices and environmental variables. Using this technique, undesirable interactions are avoided. The inclusion of the environment is a key element of this approach as many interactions can happen indirectly, through the environment.

Since the configuration of a home's devices and services is continually changing, developing an off-line solution is not practical. Therefore, an on-line approach in the form of an interaction manager has been developed. It is the manager's role to detect interactions.

The approach was shown to work successfuly. The manager was able to successfully detect interactions and prevent negative interactions from occurring. Interactions were detected at both device and service level. The approach is flexible: it is protocol independent, services are unaware of the manager, and the manager can cope with new devices and services joining the network. Further, there is little user intervention required for the approach to operate.

Acknowledgements

There have been so many friends and colleagues through the course of this work who have helped me. Without their suggestions, encouragement and support, it is unlikely I would have successfully completed the thesis. There are, however a few special people who I have to express special thanks to.

Firstly, Professor Evan Magill for introducing me to the field of feature interaction. His help, patience and and encouragement have made my time at Stirling both interesting and thoroughly enjoyable. I also want to thank Dr. Mario Kolberg for his valuable discussions and time spent watching while I drew boxes and arrows on Canadian beer mats!

A big thank you to Alison Martin and John Willison who were there to proof read and provide editorial comments.

I know there are many other people who I have met along the way, given me support, ideas, and who have listened to me looking interested and nodding at suitable moments. There are far too many to even start to list, but you know who you are. To everyone, thank you so much.

Abbreviations

AC	Automatic Callback
AR	Automatic Recall
ARP	Address Resolution Protocol
BCNB	Blocking Calling Number Delivery
CF	Call Forwarding
CFB	Call Forwarding on Busy
CFU	Call Forwarding Unconditional
CNB	Calling Number Blocking
CND	Calling Number Delivery
CNDB	Calling Number Delivery Blocking
CSS	Communications Support Service
CUSY	CUstomer SYstem
CW	Call Waiting
DER	Device and Environmental Representation
DHCP	Dynamic Host Configuration Protocol
DIS	Device Information Service
DLI	Device Location Information
FIM	Feature Interaction Manager
GENA	General Event Notification Architecture
HAVi	Home Video and Audio Interoperability
HES	Home Entertainment Service
HTTP	Hyper Text Transfer Protocol
HVAC	Heating, Ventilation and Air Conditioning
HSS	Home Security Service
IP	Internet Protocol
JES	Java Embedded Server
JPEG	Joint Picture Expert Group
LDAP	Lightweight Directory Access Protocol
LI	Looping Interaction
LTT	Linear Temporal Logic
MAI	Multiple Action Interaction
MUMC	Multiple User, Multiple Component
MUSC	Multiple User, Single Component
MMS	Multimedia Messaging Service
MTI	Missed Trigger Interaction
NA	Networked Appliance
OCS	Originating Call Screening
OSGi	Open Services Gateway Initiative
PCS	Power Control Service
SAI	Shared Action Interaction
SCP	Service Control Point
SIHP	Service Interaction Handling Process
SIM	Service Interaction Manager
SINPC	Service Interaction Network Protocol Converter
SIP	Session Initiation Protocol
SOAP	Simple Object Access Protocol
SMF	Service Management Framework

SMS	Short Messaging Service
SSDP	Simple Service Description Protocol
STI	Shared Trigger Interaction
SUMC	Single User, Multiple Component
SUSC	Single User, Single Component
TCP	Transmission Control Protocol
TLA	Temporal Logic of Actions
UDP	User Datagram Protocol
UPnP	Universal Plug and Play
X.10	A simple powerline protocol for home automation

Contents

List of Figures

List of Tables

Chapter 1

Introduction

People have been intrigued by the home of the future for generations. Such fascinations have been fuelled by science fiction movies and futuristic television programmes. As far back as the early 1960's the popular children's television series, the Jetsons, depicted a futuristic family living in an automated home. This home included regular household appliances: television, washing machine, cooker, etc. along with Rosie (a robot maid), who carried out general household chores. More than three decades later, the vision of an intelligent, self maintaining home is becoming a reality.

Currently, the general consensus of a smart home is relatively conservative. The smart home can be defined as a collection of networked appliances connected to a home LAN and controlled by one or a number of software services. By networked appliances we mean traditional household appliances with additional internal software and a network interface. These devices may include displays, heaters, air-conditioners, fridges or even music stereos. The added value of networking these appliances is that they can be controlled by one or a number of software services remotely. These software services can be purchased by the user from either one or many service providers. These services may include entertainment, security or even climate control services – all of which run from a central point in the home, a residential gateway. It is envisaged that new devices joining the network will automatically configure themselves and become integrated into the home system. This allows the software services to use the most appropriate device available at the time.

Although this is the current view of the smart home, it will change. Already, ubiquitous computing (the invisible computer) is attracting considerable attention. Computers are becoming so small and cheap that they will be embedded into clothing and buttons. This offers the possibilities that items of clothing can communicate with the washing machine, the washing machine can then determine the optimum wash cycle for the load. However, there are many other possibilities which this technology can bring.

A number of trial homes have been created to show how effective home networking can be. These projects include: e2-Home in Stockholm [7]; the Internet Home Alliances OnStar homes in Detroit [8], and the TAHI smart home in Loughborough [9].

These trial homes use devices which are generally not available off the shelf; however, some home networking technologies are available to buy. X.10 [10] is one such technology. X.10 devices have proved popular with home automation enthusiasts as they are relatively cheap, both in terms of cost of equipment and of setting up the X.10 network. Software for controlling devices is also available off the shelf. IBM Home Director [11] is one such product.

Home automation for all is still in its infancy. Trial homes, despite being sophisticated, are not truly dynamic. Software controlling devices in such a house has been carefully engineered for the part. Products such as X.10 and Home Director are relatively simple, and what the user can achieve is limited. The road to fully achieving the automated home for general use is still some way off.

Issues remain to be overcome; however, the area is enjoying considerable growth in both industry and academia. As a result of technological advances and the drop in price of consumer electronics, the automated home for all is beginning to come within reach.

With many software services each controlling a number of devices, it is inevitable that two services may need to control the same device. This leads to serious problems with compatibility. The problem may not even be due to two services trying to control a single device, but perhaps a service controlling a device which clashes with what a separate device (or service) is trying to achieve.

An example of this type of incompatibility is between a heater and air conditioner. Since one device heats and the other cools, clearly they are not compatible with one another. However, there is no direct link between a heater device and an air conditioner device. There is another factor involved – the room temperature. The temperature is part of the room environment, in the same way that movement can be seen as an attribute of the room's environment. This example shows that the room environment is an important factor in the home as compatibility issues arise through it.

Not only can the devices be incompatible with one another, but services can be incompatible too. For example, one service may want to open windows for ventilation, whereas another service wants to keep windows closed to keep the home secure. There is an incompatibility here, as the opening of windows makes the home insecure. However, like the example between the heater and air conditioner, there is no direct link. The link is indirect and is through the environment. It must be noted that this problem is only a problem when security is an issue. If there was no security, opening the window would probably not cause a problem.

As the examples above show, incompatibilities in the home are a problem. Although incompatibilities in the home may be a new area of research, the general topic has been the focus of academic and industrial research for over a decade. The problem is generally known as the *Service Interaction* or *Feature Interaction* problem.

1.1 The Problem

The service interaction problem is where the action of one service has a negative impact on another [12]. The fact that services conflict with one another is not due to badly written services, but simply services with broken assumptions and conflicting goals [13]. Broken assumptions are where the designers of a particular service make assumptions which are then broken by another (unexpected) service. For example, if a security service was developed then the assumption may be that there should be no movement when no one is home. Therefore any movement detected is interpreted as an intruder. The ventilation service may turn on a fan. This causes movement, which is interpreted as an intruder by the security service. The security service made the assumption that there should be no movement, but the ventilation service broke this assumption by turning on the fan and causing movement. No movement in the home is not a *bad* assumption to make by the security service designers; it is the ventilation service which breaks this assumption.

Although some interworking is positive, i.e. allowing devices to work together to reach a common goal, negative interactions are not. These types of interactions must be avoided if the networked home is to succeed [14].

The topic of feature interaction has received much research over the past decade with work widely published on the subject, and indeed a series of workshops held [15–21]. Much of this research has been concentrated in the telephony domain. However, the issue in electronic mail, elevators or web-services, among others, has been studied. Despite good progress, an agreed universal solution to the problem has proved elusive. To date, the home automation domain has received little attention.

There are many causes of interactions in the home. Like all other domains, the primary issue is that some services have conflicting goals and broken assumptions. Solving the problem in the home is not straightforward.

Since most households will have a different make-up of services (and configurations) and devices, it means it is not always clear which device a service may use. For example, a service may have the goal of cooling a room the quickest way possible. If an air conditioner is available, using it would be appropriate. If an air conditioner was not available and it happened to be cooler outside, an alternative would be to open a window. However, by opening a window, the cooling service could interact with a security service. If the owner did not have a security service, no interaction would occur. This highlights the problem of detecting interactions within the home domain.

As previously stated, interactions occur due to broken assumptions and conflicting goals. The first time services meet will be in the network at runtime. If all services were developed by the same vendor, all services and service permutations could be tested against one another. This allows interactions to be fixed before the service is deployed. In the home networking environment this is not possible as there will be many vendors where competition is fierce. Furthermore, vendors are unlikely to exhibit their new service to each other.

Services are one variable in the problem; however, the problem is worsened by the ad hoc nature of devices in the home. New home networking protocols are developed to specifically support leaving and joining networks.

Therefore, it becomes impossible to predict the combination of devices and services (including how they are configured) until runtime.

Although the flexibility with devices and services offers a customised home automation solution for a user, it makes automatically detecting and resolving service interactions difficult. It is worth noting that once an interaction has been found, it is relatively straightforward to fix [13]. However, if a customer is paying for services they will not tolerate these *surprises*, regardless of how easy they are to resolve.

1.2 Aims of this work

As discussed in the previous section, the service interaction problem is an issue in the home. Therefore, the aim of this work will be to develop an approach which will:

1. Avoid negative interactions from occurring in the home network between different devices and services.
2. Consider the environment as many interactions occur though it.
3. Handle new devices and services joining the home network, as well as existing devices and services leaving the network.
4. Handle services from multiple service providers as there will be many companies selling services for the home, each specialising in a particular area.
5. Have limited user intervention as the networked home is supposed to make life easier for the occupant. Therefore, they are not likely be interested in incompatibility issues.

1.3 Contributions of this work

The approach presented in this thesis is a novel approach for service interaction avoidance in the home domain. It is an automatic, runtime approach which is able to avoid negative interactions.

There are few published papers which tackle the feature interaction problem in the home. Nakamura et al. [22] present an approach which uses the environment; however this is an off-line approach. Metzger and Webel [23] present an approach for service interaction in building control (which has many similarities to the home domain) which uses the environment; however, their approach is also off-line. In contrast, the approach presented here is a runtime approach.

The work here is novel as it is the first online approach specifically for the home. The approach is not service specific, and therefore can support a multi-vendor environment. Indeed, the approach here is the first *device centric* approach. This means a device and the device's environment is the focus, rather than traditional approaches in feature interaction which focus on the service.

The approach presented here makes use of the concept of environment to detect and avoid interactions. Within the feature interaction community, there are few avoidance approaches.

As well as being a runtime approach, it does not require a *warm-up* period. That is, it can be deployed straight into a network. It would also work immediately with newly added devices, regardless of their underlying network protocol. This makes it an extremely flexible approach.

However, most importantly the approach fulfils the aims stated above.

1.4 Achievements and limitations of this work

The main achievements of this work include:

- Negative interactions are avoided while positive interactions are allowed. This allows services to work together to achieve a common goal.
- Able to detect and resolve interactions at the device and service level.
- Manager operates at runtime with negligible overhead running cost (less than one second).
- Device protocol independent.
- Services do not need to be aware of the feature interaction manager. Therefore the approach works with services from any vendor.
- Requires no warm-up time. As soon as the manager is deployed onto the network, it is able to start avoiding negative interactions.
- Able to handle devices and services joining and leaving the network.

Although the results from the approach are mainly positive, the approach does have some limitations (these limitations will be discussed in length in Chapter 8). These limitations include:

- The approach is unable to detect interactions caused by the same service (intra-service interactions).
- Devices and services must all be registered and controlled centrally for the approach to work.
- Unable to avoid looping interactions. This is because when a service is finished with a device the *session* has finished and devices and variables are free for others to use them.
- Some *a priori* information is required regarding devices and the environmental variables that the device will affect.
- Rooms in the home are considered to be independent of one another.
- Side effects of devices are not included.

There are some slight limitations of this approach, however these derive from the fact that this is a straightforward and simple technique. The approach presented here is simple and is able to avoid the majority of interactions in the home. In the worst case, a user can always override the manager.

1.5 Structure of the thesis

The thesis is laid out in the following way. Chapter two will discuss the networked home. This will include explaining the types of devices (including protocols) in the home and how devices can be brought together by using a service management platform.

Chapter three will discuss the feature interaction problem, highlighting previous work carried out. This chapter will then outline the requirements for a new approach for the home while explaining why existing approaches are not suitable for this domain.

Chapter four will cover the kinds of services which will be available in the home. The service interaction problem in the home is highlighted in this section, and interaction scenarios are outlined. These scenarios are used in the experimentation section to show the effectiveness of the approach.

Chapter five reveals a new and novel approach for service interaction management in the home. Previous approaches for feature interaction have been service centric – requiring detailed information about the service. Due to services constantly adapting to their surroundings, a service may behave differently each time it is executed. Therefore, the approach presented here focuses on the device and its surrounding environment, as the behaviour of a device does not change. For example, a device may be a heater where the surrounding environment it affects is room temperature. Thus, by controlling access to the device and the device environment, negative interactions can be avoided. For controlling access to the device and environment, inspiration has been drawn from the operating systems domain.

Following the description of the approach, chapter six discusses the architecture of the test-bed. It is important to show that the approach works in practice, as well as theory, therefore a test-bed is required to achieve this. This chapter will detail the devices and services used in the test-bed. Further, the design of the approach will be included here.

Chapter seven presents the results from experimentation. The effectiveness of the approach is shown by using each of the scenarios depicted in Chapter four.

Finally, chapter eight contains the conclusions and further work. The strengths of the approach as well as its limitations are presented here. The section on further work will explain how the approach can be improved and moved forward into other domains.

Chapter 2

The Networked Home

The motivation for automating the home has often been questioned. However, by connecting devices, and allowing them to work together, some exciting possibilities for the home can be delivered [24]. Due to this connectivity, users will have access to their data (e.g. audio or video collections or state of home), regardless of their location, provided they have a network connection.

Figure 2.1 depicts an example home network where a multitude of networked devices[1] are connected using a variety of networking protocols. New possibilities arise when this new technology is used. Three areas which can benefit are health care, entertainment and convenience.

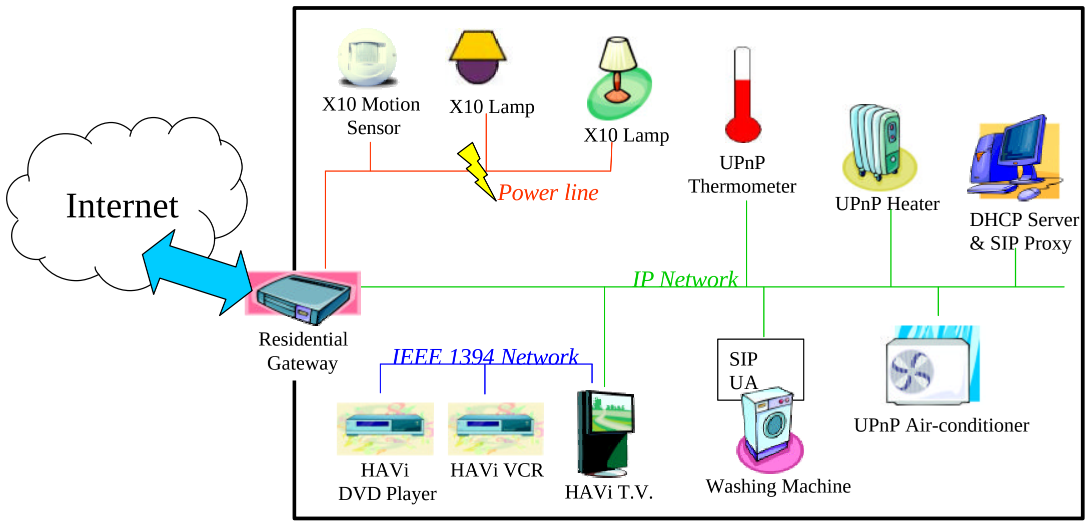


Figure 2.1: The Home Network *(from [1])*

A key area where the automated home is expected to benefit is health care [24, 25]. It is estimated that, currently, cognitive decline and cardiovascular problems currently cost the US economy $600 billion [25]. If these problems are detected earlier, considerable sums of money and lives can be saved. For example, many people do not realise they are ill until the later stages of a disease, by which time it can be more expensive and harder to treat.

Michael J Fox, who starred in the US sitcom 'Spincity', was diagnosed with Parkinson's in 1991. When episodes were analysed, it could be seen that his actions changed over a period of time [25]. It was only by analysing the data which had been collected over a period of time that gradual changes were noticed.

To detect subtle changes in a person, clearly they do not want to be wired to machines, or have to place patches on themselves each day. Tamura et. al [26] suggest techniques which are not intrusive.

[1] A networked device is a normal device with a network adapter. In this thesis, a networked device is considered the same as a networked appliance.

These include ways to measure a person's heart-rate while they are in the bath, their weight through the toilet, and body temperature while the person is in bed. This data is recorded and can then be sent via the home to their GP for analysis [25].

This technology has the potential to save lives, improve people's quality of life, and to save local authorities large sums of money, as pilots in the UK found [27].

Various companies provide smart home solutions for this market [28–30]. One example of the smart home is the Hogar.es Project [31] led by the Spanish telecommunications company Telefonica. These homes allow those who would be placed into a nursing home to stay in their own homes for longer. The same applies for patients recuperating in hospital – they can return home sooner, and be monitored remotely.

These homes can monitor users for falls [27], and notify a carer. Other devices could be used to monitor a patient's heart pattern; again, if the heart pattern goes into a danger zone, someone can be alerted [32]. Also, the home can make sure that the occupant has not left an unlit gas cooker ring on, or left a bath running – both of which are common with people who suffer from dementia. It is the confidence which these technologies bring to the users (elderly or infirm) which allow them to be independent for longer. It leaves them, and their families, safe in the knowledge that if anything were to go wrong, someone would be alerted promptly [27].

As well as health care, these homes can be convenient for a busy family, where any assistance to household tasks is welcome [24]. One such use may be that devices will have the ability to monitor themselves, sending diagnostic data to the manufacturer. If a component in the device is about to fail, an engineer can visit with a replacement part before the appliance fails [33]. Currently, if an engineer has to call to service an appliance within the home, someone has to stay at home. In the connected home, it will be possible that when the engineer calls and rings the door bell, the owner can check it is the engineer, unlock the door, let them in and watch them while they service the appliance. When the engineer leaves, the front door can be locked. All this can be carried out remotely [34].

Climate control services may keep the home at a comfortable temperature, automatically adjusting windows, heaters or air conditioners accordingly. A central locking feature for the home can secure the home and turn devices off (such as an iron) after the front door is locked.

A home described above consists of devices and user services. The devices and the protocols they use are described in the next section, and user services are discussed in Chapter 4.

2.1 Protocols used in the home network

Networked devices are similar to their traditional counterparts but differ in that they have a network interface. These devices are actuators, sensors or displays. This interface allows them to be networked and controlled remotely by services.

In the home there is no single *home networking protocol*; instead, there are a plethora of protocols. Currently there are over 50 protocols developed for the home [35]. The reason for so many is due to the number of consortia and authorised standardisation bodies who are creating their own protocols. Generally, these protocols do not interoperate [36], however with appropriate middleware solutions, they can be made to do so.

Figure 2.1 shows a simple example of a wired home network. The figure shows four different protocols: X.10 [10], UPnP [37], SIP for Appliances [33] and HAVi [38]. These are high level protocols and in the the OSI ISO reference model, these are in the application layer. The underlying protocols, such as IEEE 802.3 (Ethernet) or IEEE 802.11 (Wifi) are in the data link layer.

Of the protocols shown here, two contrasting protocol; X.10 and UPnP. X.10 is a powerline protocol, where no intelligence lies within the device: the X.10 protocol simply controls the voltage to the device through an adapter at the power socket. In contrast, in UPnP devices the intelligence is located on the device itself, potentially giving far more control over the device.

2.1.1 X.10

The X.10 protocol is a simple powerline protocol. X.10 adapters plug into the electrical wall socket, and traditional devices (e.g., lamp or fan) plug into the adapter. The X.10 adapter simply controls the voltage to the normal device, thus there is no *X.10* intelligence in the device. For this reason, the protocol is limited in what it can do, and indeed in the devices it can be used with. Figure 2.2 shows an example X.10 network within a home.

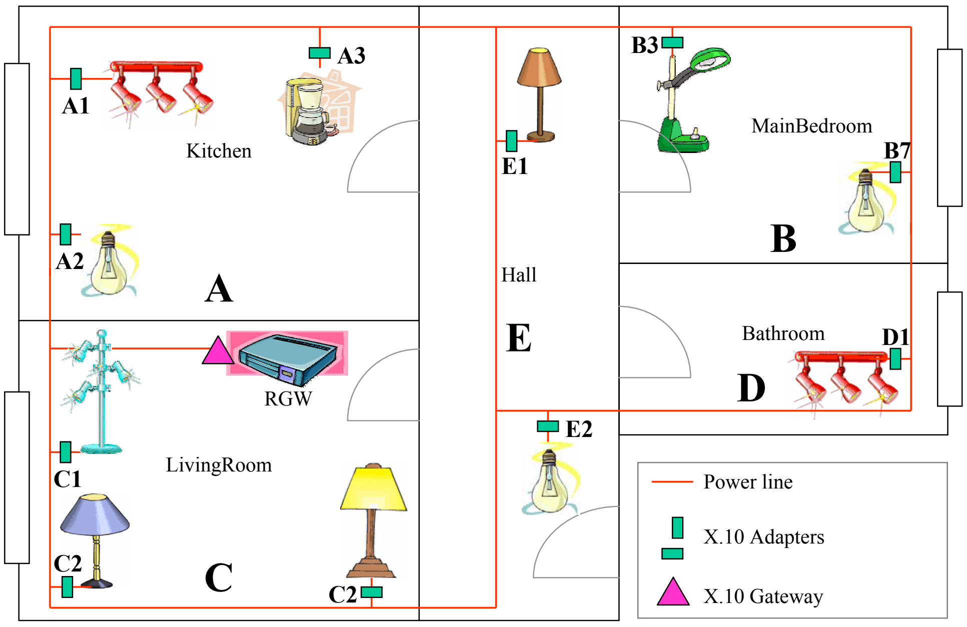


Figure 2.2: Example X.10 Home Network

Its functionality is limited to switching a device either on or off or, in the case of a lamp, dimming and brightening. There are also other features which the protocol supports. These include turning all appliances in one room on or off, all lamps in the home on or off. However, it does no more than control the power a device receives.

The X.10 network works by placing an X.10 gateway (Figure 2.3(a)) into a wall socket. This is the gateway which bridges the powerline network to the personal computer or residential gateway, shown as the triangle in Figure 2.2. The X.10 adapters (Figure 2.3(b)) simply plug into the wall socket and listen for messages sent onto the powerline by the gateway, shown as rectangles in Figure 2.2.

Addressing of devices

Devices and addressing are set up manually. The X.10 protocol supports 256 unique addresses. There are two parts to an X.10 address: the room and the device number. The protocol defines that the room letter runs from A to P (inclusive). Device numbers run from 1 to 16 (inclusive). Figure 2.2 shows an example home with five rooms (A–E).

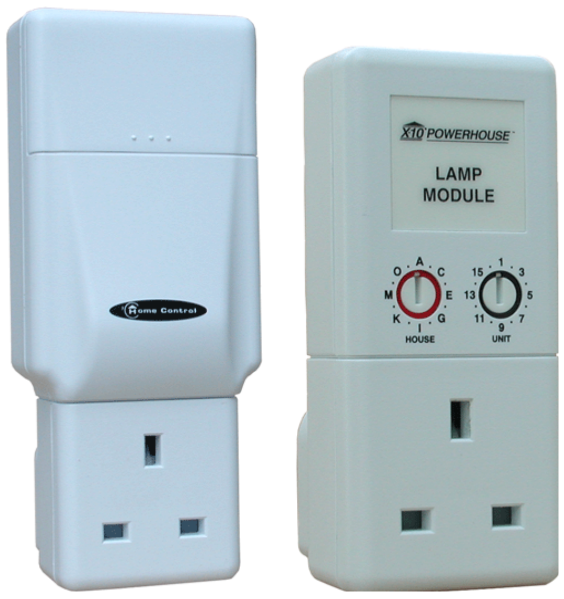
Figure 2.3(b) shows a X.10 lamp adapter with two dials, the first being the room letter and the second being the device number. An `on` command to A1 would turn device 1 on in room A.

The protocol allows many devices to share one address, which can be seen in Figure 2.2. In this example, two adapters have the address `C2` in the living room (room C). This means when a command is sent with address `C2`, both adapters will act upon the request.

It is up to the user to keep addressing consistent and ensure correct room letters are assigned to the appropriate devices. Otherwise commands to turn all devices in room B to an `off` state may turn devices off in other rooms.

The protocol has been popular with home automation enthusiasts. This is partly due to its cost as the adapters are relatively cheap. Also, existing household devices, e.g. lamps or fans, can be used. The purchase of new devices is not required. Software is also freely available for controlling the home using X.10, such as IBM Home Director [11], PowerHome [39] or open source Java APIs [40].

One of the main problems with using X.10 is limited addressing. If a neighbour also has an X.10 network, both neighbours can affect each other's appliances. If neighbours can control each other's appliances, this raises another issue: security. X.10 does not support any form of security.



(a) Gateway adapter (b) Lamp adapter

Figure 2.3: Example X.10 Adapters *(from [2])*

Despite these limitations, X.10 is popular, with a number of web-sites selling equipment in the UK [2]. X.10 functionality is limited due to how it works (controlling the power supply to a device). In contrast, other protocols have been developed which are embedded in the device. These protocols, since they are inside the device, have the potential to be much more powerful, controlling the device in specific ways. UPnP is an example of one such protocol.

2.1.2 Universal Plug and Play (UPnP)

The UPnP Forum [37] was set up in 1999 and is led by Microsoft and Intel. The forum now has over 700 members from a diversity of backgrounds.

As the name suggests, UPnP is *plug and play* – that is, when a new UPnP device is connected to the network, it advertises itself and makes itself available to other network users automatically. Zero configuration is the key to UPnP [35].

Plug and play is not a new concept. The idea has been successfully used in desktop computing for a number of years. Most new keyboards or memory sticks, for example, can be plugged into a desktop computer and the computer has the ability to recognise the device and start using it. No drivers and no fuss to start using them. This is restricted to the desktop computer; broadening it out to be used in the home is where UPnP can be used.

Using a protocol which supports zero configuration for the home is important if home automation is to move from the early adopters to mainstream users. Users will want to buy a new device, take it home, plug it in and for the device to work with existing devices and services. They are unlikely to tolerate the downloading of new drivers and re-configuring existing devices and services to work with the new item. UPnP enabled devices have the potential to be placed into the home and to start working.

UPnP uses a number of established protocols, including: IP, TCP, UDP, XML [41], SSDP [42] and SOAP [43]. As UPnP operates on an IP network it is independent of any specific underlying transport protocol [44], this allows UPnP devices to be wired or wireless.

In a UPnP network there are two components: the *device* and the *control point.* A *device* offers services: a thermometer may offer a temperature service or a lamp may offer a lighting service. The *control point* is the component which is able to search and make use of these services. A device cannot control another device unless it has a control point. Figure 2.4 shows a small UPnP network with a UPnP enabled printer which offers two services, and a UPnP enabled camera offering one service. Since

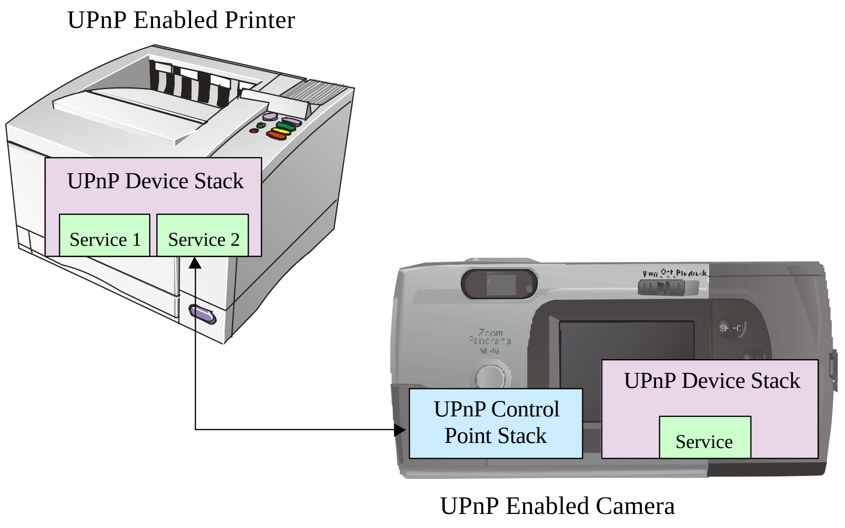


Figure 2.4: Example UPnP configurations

the camera has a control point, it can control the printer. As the printer device does not have a control point, it is not able to control the camera.

Using the example in Figure 2.4, assume the camera has not yet been added to the network. Further, assume the home has a wireless LAN and the UPnP network only has a UPnP enabled printer device, which is active. The home user brings a wireless UPnP enabled digital camera into the home and wishes to print pictures from the camera to the printer.

A number of steps are automatically carried out before the camera is able to use the printer. There are six important stages in the UPnP specification [44, 45]: Addressing, Discovery, Description, Control, Eventing, Presentation.

Addressing

When the new device is introduced to the network (the digital camera shown in Figure 2.4, for example) it must obtain an IP address. If there is a DHCP server on the network, it is able to allocate the new device an IP address.

If a DHCP server is not available the device must create its own address using auto-IP. If this is the case, the device will randomly choose an address in the 169.254.0.0/16 range. This range has been ear-marked by the IETF as the IP range to use for end node auto-configuration when a DHCP server is not found [46]. The device then uses ARP [47, 48] to find out whether another device on the network is using this address, if no reply is returned, the device uses this address. If another device uses this IP address, the selection process is repeated.

At this point, the newly joined camera has an IP address. Next, it must inform the network of its existence.

Discovery

After the device has an IP address it can advertise itself. If it is a control point, it can search for devices of interest on the network.

To broadcast its presence, a UPnP device uses SSDP. An `ssdp:alive` message is broadcast (Figure 2.5) across the network to a well known port (typically port 1900) using a reserved IP broadcast address, `239.255.255.250`, for example. Any control point established on the network will listen on this port for new UPnP devices. When the new device is detected, it is added to a list of devices within the control point. The SSDP alive message also gives a path of where the device description can be found: this is shown in line 4 of Figure 2.5, the location.

```
1.   NOTIFY * HTTP/1.1
2.   SERVER: Windows XP/5.1 UPnP/1.0 CyberLink/1.0
3.   CACHE-CONTROL: max-age=1800
4.   LOCATION: http://139.153.254.43:4004/description.xml
5.   NTS: ssdp:alive
6.   NT: upnp:rootdevice
7.   USN: uuid:mewCamera::upnp:rootdevice
8.   HOST: 239.255.255.250:1900
```

Figure 2.5: Example SSDP alive message

```
1.   M-SEARCH * HTTP/1.1
2.   HOST: 239.255.255.250:1900
3.   MAN: "ssdp:discover"
4.   MX: 3
5.   ST: upnp:rootdevice
```

Figure 2.6: Example SSDP discover message

A device advertises itself to all control points when it joins the network, but when a new control point joins the network, it must find all UPnP devices on the network. To do this, it also broadcasts an SSDP message. The message is slightly different in that it is an `ssdp:discover` message (Figure 2.6). Devices listening (on port 1900) for this message will reply with an HTTP 200/OK message, Figure 2.7. Line 3 of Figure 2.7 shows the IP address of the device.

```
1.   HTTP/1.1 200 OK ST:upnp:rootdevice
2.   USN:uuid:0012172d-7225-0012-172d-72250032011c::upnp:rootdevice
3.   Location: http://192.168.1.1:5431/dyndev/uuid:0012172d
                                        -7225-0012-172d-72250032011c
4.   Server: Custom/1.0 UPnP/1.0 Proc/Ver EXT:
5.   Cache-Control:max-age=1800
6.   DATE: Tue, 04 Jan 2005 12:46:15 GMT
```

Figure 2.7: Example reply message from UPnP router from search

The message shown in Figure 2.6 will get responses from all UPnP devices that want to be found (it is possible for a UPnP device to be on a network, but be ‘hidden’). Rather than getting all devices, it is possible to refine the search and find specific devices or services. This is achieved by changing the ST (Search Target) parameter, line 5 of Figure 2.6. For example, `ST: urn:schemas-upnp-org:-service:SwitchPower:1` which would search for a particular service (SwitchPower), regardless of device.

As well as broadcasting itself when joining the network, a UPnP device should also notify the network when it leaves. Again, an SSDP message is broadcast, `ssdp:byebye` (Figure 2.8).

Any control point with this device on its list will remove the device immediately, and it will only be added when the device rejoins the network. If, however, a control point leaves the network, it unsubscribes from any services it is registered to and leaves. Devices are unaware of control points unless the control point has subscribed to a service which a device offers (this is explained further in section 2.1.2).

Continuing the example of the new camera being added to the UPnP network: currently, the new UPnP camera has joined and has obtained an IP address. By using SSDP, it has advertised itself to other control points. Since the camera has a control point, the camera has found all other UPnP devices, here the UPnP enabled printer. The device component of the camera is now ready to be interrogated and used by other control points on the network.

```
NOTIFY * HTTP/1.1
NTS: ssdp:byebye
NT: upnp:rootdevice
USN: uuid:mewCamera::upnp:rootdevice
HOST: 239.255.255.250:1900
```

Figure 2.8: Example SSDP byebye message

Description

After a device has broadcast its existence to the network, the control point will use an HTTP GET message to get the device description. This address is ascertained when the device replies to a control point's search message (line 3, Figure 2.7). An XML file is returned which the control point analyses to determine details such as device type, manufacturer, serial number, along with services which it offers. Types of devices are defined by the UPnP Forum, and the services certain devices must offer are also defined.

The device description is an XML schema defined by the UPnP Forum. This XML file contains basic device details, including: manufacturer, model number, a serial number and a unique ID for the device. It also gives the type of service and path where the service description files for the device can be found.

The service description files are also XML schemas, defined by the UPnP Forum. These files specify the actions a user can invoke. The file contains *Actions* and *State Variables*.

State variables describe the run-time state of the service. For example, a state variable within the lamp holds current state – on or off. Actions, on the other hand, list the method calls a control point can invoke. The result of an action can either return the actual value of a state variable, or may change the state.

In section 2.1.2, the control point within the camera would have found the UPnP printer device when it searched for all devices on the network. However, it is only after retrieving and parsing the XML device description and service description files that a control point knows exactly the type of device this is and what services it offers. Only then, can the control point control the device; in this case the camera wants to control the printer.

Control

As previously discussed, it is only after a control point discovers the details of the device and service that it can control a device. This is done by invoking actions listed in the XML service description file.

It is worth reiterating that only a control point may control a device. A device on its own can not control another device. If a device, such as the camera in Figure 2.4, wants to control a device, it must be done by a control point.

Control is achieved by sending SOAP messages (Figure 2.9) to a device. The SOAP message will invoke an action, defined in the service description XML file. Figure 2.9 shows a SOAP message which requests the state of a device. In this instance the camera requests the device state by invoking the `GetDeviceState` action (Figure 2.9, line 12). No parameters are required for this call, therefore the tag is closed.

Since SOAP messages are sent over HTTP, HTTP responses are sent back depending on the outcome of the request.

The reply to the message sent in Figure 2.9 is shown in Figure 2.10. Line 1 of Figure 2.10 is the `200/OK`, acknowledging the message was received and processed successfully. Line 11 is the response; the next line shows the device's current state, `ready`.

Since the camera knows the printer is ready, it can then send images to the printer for printing. The UPnP Forum specifies a UPnP printer that prints JPEG format image files [50]. Sending images is achieved through sending another SOAP message which invokes an action with the image as a parameter. Provided the camera sends JPEG format, the printer is able to print the images. Manufacturers may support other image formats, but this is not compulsory.

Since joining the network, the UPnP enabled camera has obtained an IP address, advertised itself (camera service) and has used its control point to find other UPnP enabled devices on the network (the

```
POST /service/printer/control HTTP/1.1
HOST: 139.153.254.43:4004
CONTENT-LENGTH: 344
CONTENT-TYPE: text/xml; charset="utf-8"
SOAPACTION:"urn:schemas-upnp-org:service:Printing:1
                                       #GetDeviceState"

<?xml version="1.0"?>
<s:Envelope
    xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
    s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:GetDeviceState xmlns:u="urn:schemas-upnp-org:
                                       service:Printing:1"/>
  </s:Body>
</s:Envelope>
```

Figure 2.9: Example SOAP message to get device state

```
HTTP/1.1 200 OK
CONTENT-TYPE: text/xml; charset="utf-8"
SERVER: Windows XP/5.1 UPnP/1.0 CyberLink/1.0
EXT: CONTENT-LENGTH: 326
DATE: Wed, 15 Sep 2004 12:26:41 GMT

<s:Envelope
    xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
    s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:GetDeviceStateResponse xmlns:u="urn:schemas-upnp-org:
                                              service:Printing:1">
      <CurrentDeviceState>ready</CurrentDeviceState>
    </u:GetDeviceStateResponse>
  </s:Body>
</s:Envelope>
```

Figure 2.10: Example SOAP reply message

printer). After finding the printing service, it requests the printer state and sends JPEG images to the printer for processing. The user is quite oblivious to what has happened, they have simply selected the image and chosen the print option from the camera.

The two remaining components of the UPnP specification are eventing and presentation, both of which are extremely valuable.

Eventing

After a control point has discovered a device and its services, it may subscribe to a service. This means the control point will be notified of any change in the service state. This is particularly useful as it means services do not have to periodically poll a device to obtain its state. In the example of the UPnP enabled printer, if the printer were to run out of paper, a monitoring service may be alerted, or if the digital camera were subscribed, it could be alerted and in turn notify the user.

Subscription and event notification is carried out using the General Event Notification Architecture (GENA), an IETF draft. When a control point wishes to subscribe to a service it will send a subscribe message to the device (Figure 2.11). The device will then send an acknowledgement back to the subscrib-

```
1.  SUBSCRIBE /service/heating/eventSub HTTP/1.1
2.  HOST: 139.153.254.43:4004
3.  CALLBACK: <http://139.153.254.69:81/eventDelivery>
4.  NT: upnp:event
5.  TIMEOUT: Second-1800
```

Figure 2.11: Example GENA subscribe message

```
1.  HTTP/1.1 200 OK
2.  CONTENT-TYPE: text/html; charset="utf-8"
3.  SERVER: Windows XP/5.1 UPnP/1.0 CyberLink/1.0
4.  SID: uuid:36b6-a0ff-2ec5-e0a0
5.  TIMEOUT: Second-1800
6.  DATE: Thu, 16 Sep 2004 10:04:24 GMT
```

Figure 2.12: Example Device accepts subscribe message

ing control point notifying it whether the subscription has been successful (Figure 2.12) or unsuccessful. This message also contains a unique ID for the duration of the subscription, `SID`.

After receiving a subscription request and accepting it, the device will add the control point to a list of control points. These subscribers will be notified of any updates of the service. When the service state changes, the control point will be notified; it is up to the control point what it does with the information received.

Figure 2.11 (line 5) shows the subscription period is set to 1800 seconds. After this time has elapsed, if the control point is still interested in receiving updates from the service, it can re-subscribe. Similarly, if a control point no longer requires state updates, it is able to unsubscribe from a service (Figure 2.13). When the control point unsubscribes, the SID must be included in the header (Figure 2.13, line 3).

```
1.  UNSUBSCRIBE /service/heating/eventSub HTTP/1.1
2.  HOST: 139.153.254.43:4004
3.  SID: uuid:36b6-a0ff-2ec5-e0a0
```

Figure 2.13: Example GENA unsubscribe

Presentation

It is becoming more common for small devices (e.g. routers and print servers) with little or no interface on the device to be configured via a web interface. UPnP devices may support a presentation page, or set of pages. This can be a web server on the device which allows the user to control the device via a web interface. The presentation URL is found in the device description XML file.

In the case of the printer, it may be possible to check the current state of the printer, change settings or perhaps check toner and paper levels. A presentation facility may be extremely useful in the case of the digital camera, allowing users to view their photographs through a web browser, television, PDA or home PC.

2.1.3 Summary of UPnP

The UPnP protocol was developed with the specific aim of being plug and play. It has achieved this goal by using a series of existing, open protocols and standards [44]. Sections 2.1.2 – 2.1.2 have shown how a UPnP enabled camera device has been added to the network, automatically configured, and has advertised itself to control points on the network. The camera, upon finding a UPnP enabled printer, has been able to send its images for printing. Throughout the process, the user has been unaware of this process and simply sees the end product – their printed photographs. This is much easier than today where the user has to install drivers and set up devices manually.

The UPnP Forum continues to grow and publish more device and service specifications. The number of UPnP devices available off the shelf is slowly growing. Routers and residential gateways were among the first UPnP devices, the Linksys WRT54G [51] for example. However, the Nokia N80 comes with a UPnP stack that allows music to be found and played on handsets. Philips produced the Philips SLM5500 which is a UPnP Hi-Fi which is able to work with a UPnP NAS device (e.g. Iomega's Storcenter) and stream music from it. Also, Microsoft Windows XP uses UPnP technology for Internet Connection Sharing (ICS).

There is a host of home networking protocols available. X.10 and UPnP are examples of two contrasting protocols. Figure 2.1 shows a home with X.10 and UPnP. However, it also shows a number of other protocols that will be briefly described in the next section.

2.1.4 Other protocols used for home networking

The previous two sections have described, in detail, the workings of two home networking protocols. As mentioned, there is a plethora of networking protocols developed specifically for the home. Table 2.1 outlines a variety of wired and wireless protocols which may be used in home networking.

Many of the networking technologies listed in Table 2.1 are used by other home networking protocols to transmit data. Other higher level protocols used in home networking are as follows:

SIP for appliances [33] uses any IP based network. SIP for appliances is based on the SIP [52] protocol. Extensions (DO and NOTIFY messages) [34] have been proposed for device control. SIP is useful for controlling devices over a series of networks as the protocol supports encryption and authentication, which UPnP does not support.

Jini [53] is an IP based protocol similar to UPnP led by Sun Microsystems. It also has the notion of devices and service registries (which bear a strong resemblance to control points).

LonWorks [54] is a proprietary protocol developed by Echelon. The protocol uses powerline and twisted pair as the physical medium. It is used for controlling devices such as sensors and switches.

HAVi [38] uses IEEE 1394 as the physical medium. It is primarily used for networking audio and video devices, as the medium offers high data rates.

Name	*Wired or Wireless*	*Maximum raw data rate*	*Application*
Powerline [55]	Wired	10Mbps	Makes use of existing power cabling in the home. Used for electronic device control, low data rate transmission.
HomePNA [56]	Wired	10Mbps	Uses existing telephone lines in the home. Used for electronic device control, phone and low data rate transmission.
IEEE 802.3 (Ethernet) [57]	Wired	100Mbps	New cabling required. Used for PCs, device control, IP data control.
IEEE 802.11 family (802.11a, 802.11b, 802.11g, 802.11n) [58]	Wireless	802.11 (1Mbps), 802.11b (11Mbps), 802.11a & 802.11g (54Mbps), 802.11n(100Mbps)	A range of protocols with varying speed which are used for IP data transmission in various devices, for example, routers, PCs, WiFi enabled mobile phones.
IEEE 802.15.4 (Zigbee) [59]	Wireless	250Kbps	Wireless protocol for Personal Area Networks (PANs). Used in devices with small batteries, typically sensors.
IEEE 1394 (FireWire) [60]	Wired	400Mbps	Audio visual devices, for example, camcorder, music player, television, DVD/VCR player. A newer version of the protocol (IEEE 1394b) offers speeds of up to 3.2Gbps.
Wireless 1394 [61]	Wireless	70Mbps	Typically, home audio/visual devices.
HomeRF [61]	Wireless	1.6Mbps	Small devices, PC peripherals.
Bluetooth [62]	Wireless	720Kbps	Mobile phone accessories, PC peripherals (Printer, Mouse, Keyboard).

Table 2.1: Networking protocols used in home networking

As discussed, there is a variety of protocols used in the home. Each is suited to a different application. For example, Zigbee (IEEE 802.15.4) is ideal to send sensor data from small sensor devices as it uses very little battery power. At the other extreme, IEEE 1394 is used to send high quality digital audio and video to television displays. Since these protocols are suited to particular applications, they generally do not inter-operate. Therefore, since it is unlikely one protocol will emerge as the standard protocol for home networking, a middleware solution is required. This solution should allow devices and services to cooperate and work together. The OSGi Alliance has proposed one such solution.

2.2 A middleware solution from the OSGi Alliance

The OSGi Alliance is a non-profit making organisation which was set up in 1999. The companies who formed the alliance are from a range of backgrounds including utility companies (such as Electricit de France, Deutsche Telekom), automobile manufacturers (BMW), OEMs (Panasonic, Philips, Siemens) and software houses (IBM, Gatespace), to name a few. The main aim of the Alliance is (*from [3]*):

> "... to define and foster rapid adoption of open specifications for the delivery of managed broadband services to networks in homes, cars and other environments."

To achieve this objective, the alliance had to consider all parties involved in the service chain. This ranges from the service providers (those who deliver home services), through to the end devices in the home [6]. This fact is reflected in the range of companies who have been involved in developing the specifications.

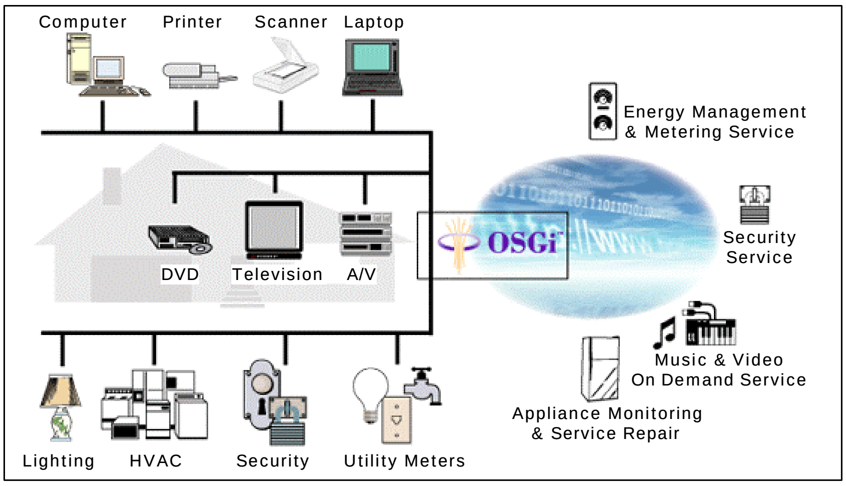

Computer
Printer
Scanner
Laptop
Energy Management & Metering Service
OSGi
Security Service
DVD
Television
A/V
Music & Video On Demand Service
Appliance Monitoring & Service Repair
Lighting
HVAC
Security
Utility Meters

Figure 2.14: OSGi home network (*from [3]*)

As the scenarios in Figure 2.1 and Figure 2.14 show, a home is likely to include a variety of devices using a range of protocols. A user in the home will want their devices to work with new services, and vice versa. They will not care whether a device uses UPnP or X.10 or Jini, or who has developed specific services – they will just want it to work. For the automated home to be a success, services must be able to make use of any suitable devices in the home. In a market where competition between hardware manufacturers and service developers is fierce, it is unlikely a home would be fitted and equipped with devices from one manufacturer and services from one service provider, using a single protocol. The key advantage is lost where a new networked device can be added and controlled by any service.

Therefore, a middleware solution is crucial, which has the ability to join devices and services from different backgrounds (whether it be different manufacturers or different protocol standards), as shown in Figure 2.14. Being able to *glue* different devices and services together is fundamental for the networked home to succeed [44]. The solution offered by the OSGi alliance offers this.

2.2.1 The OSGi framework

The solution developed became the OSGi Service Platform specification. The specification is currently on its fourth release [3] and is used commercially. The 5 Series of cars from BMW have an OSGi gateway installed [64]. The Hogar.es and TAHI [9] Connected Home both run services from an OSGi gateway.

The service specification consists of two parts: the OSGi *framework* and a set of *standard service definitions*.

The framework

The *framework* is core to the OSGi specification. The framework provides a safe and managed execution environment for services. Since the framework is Java based, it runs within a Java virtual machine. Applications are distributed in the form of *bundles*. These are uploaded onto the framework and can be managed. Since the framework is Java based, bundles can access standard Java libraries, as well as the OS for bundles containing native code (Figure 2.15).

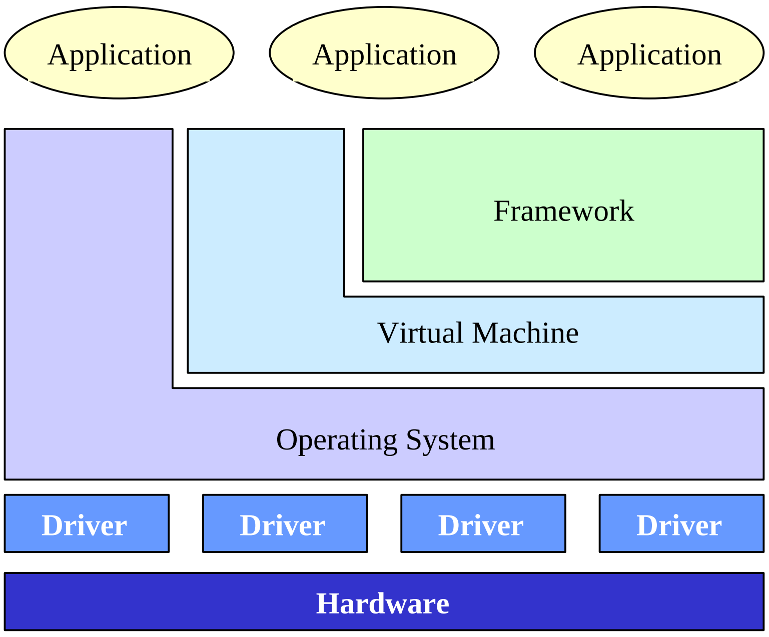


Figure 2.15: OSGi protocol stack (*from [4]*)

The bundle

A bundle is a Java ARchive (JAR) file containing the code (Java classes) and resources the service may require (images and perhaps configuration or data files). A bundle can support two tasks. A bundle may be a simple collection of Java classes in a package. This package can be exported to the framework for other bundles to use. Although exporting these packages makes them available to other bundles, they are not advertised in the service registry (section 2.2.2). An example of this may be a proprietary XML parsing class which is used by several bundles from one vendor. The XML parsing class can then be used as normal in Java code within other bundles, similar to standard Java libraries.

The second, more common use of a bundle is to offer *services*. These services are registered within a service registry (section 2.2.2). Other services can query the registry for a service it requires. This allows services to work together, allowing what starts as a small bundle to become quite large and complex. After the bundles have been downloaded to the framework they can be managed.

The bundle life-cycle

The bundle life-cycle is shown in Figure 2.16. When a bundle is installed onto the framework the bundle moves to the 'installed' state. The framework will then try to resolve the bundle, meaning any required (imported) Java packages defined in the bundles header file (manifest file) should be available in the framework. Figure 2.17 shows an example Manifest file. There are three packages required for this to function. Unless another bundle exports these packages, this bundle will be unable to start and will remain in the installed state.

When a bundle has moved into the resolved state it can either be removed from the framework (uninstalled) or can be started. It can also be updated, which uninstalls the current version of the

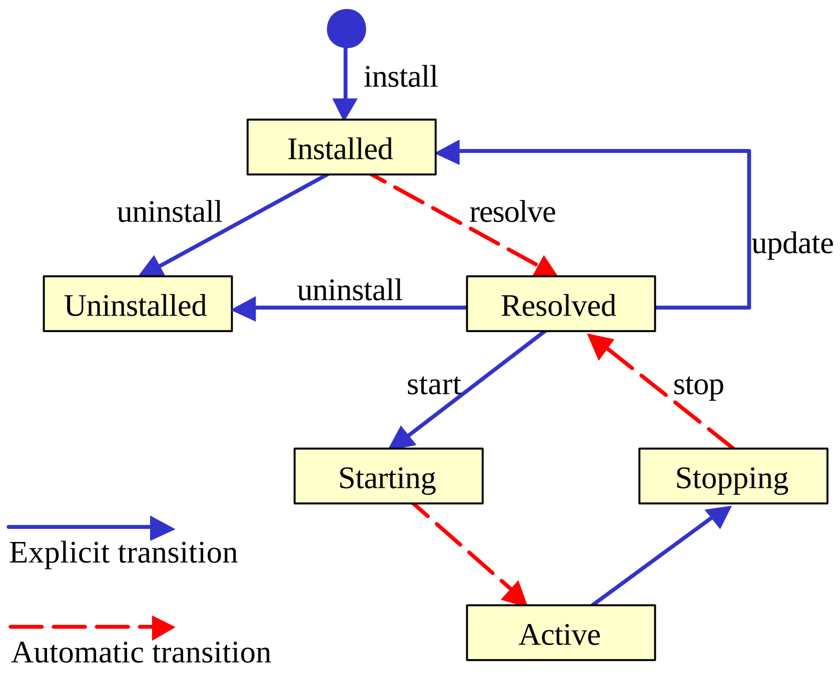


Figure 2.16: OSGi bundle states (*from [5]*)

```
1.   Manifest-Version: 1.0
2.   Bundle-ContactAddress: mew@cs.stir.ac.uk
3.   Bundle-Description: UPnP Driver
4.   Bundle-Name: UPnPBaseDriver
5.   Bundle-Vendor: Michael Wilson
6.   Bundle-Activator: uk.ac.stir.cs.osgi.
                             upnpBaseDriver.Activator
7.   Export-Package: org.osgi.service.upnp
8.   Import-Package: org.osgi.framework,
                       org.osgi.service.device, org.cybergarage.xml
```

Figure 2.17: Example bundle manifest file

bundle and retrieves a new copy of the bundle from the original source. Therefore, if a vendor updates their bundle at the source, when the update is executed, the new version would be retrieved and installed.

When a bundle is started, the bundle moves into the active state. The bundle will remain active until it is stopped, updated, uninstalled or the framework is shut down. In the main class of each bundle there is a `start` and `stop` method. When a bundle is started, the start method is called by the framework. Similarly, when updating or uninstalling, the stop method is called to allow the bundle to close gracefully. If a bundle is stopped, it will remain in the resolved state until it is started or removed.

The manifest file (Figure 2.17) is an important part of every bundle. This file holds specific details of the bundle. The format and fields of this file are defined in the OSGi specifications, see [63] section 4.3 for full details. Figure 2.17 shows an example manifest from the UPnP driver bundle.

Sun specifies that all Java manifest files start with the manifest version, Line 1 of Figure 2.17. Lines 2–5 are optional in this example. Line 6 shows the path of the bundle activator (the main class of a bundle), this is the class which contains the start and stop methods, used for starting and stopping the bundles.

Lastly, line 8–9, lists the packages this bundle requires to import before it can move into the resolved state. Line 7 declares any packages this bundle is to offer the framework.

2.2.2 The Service Registry in OSGi

The service registry is an important component within the OSGi framework. This is the component which allows services to advertise their services to other bundles.

In the bundle life-cycle (Figure 2.16), when the bundle is in its `active` state, it can start to offer services by registering them in the Service Registry. It is worth noting that a bundle may offer zero or many services.

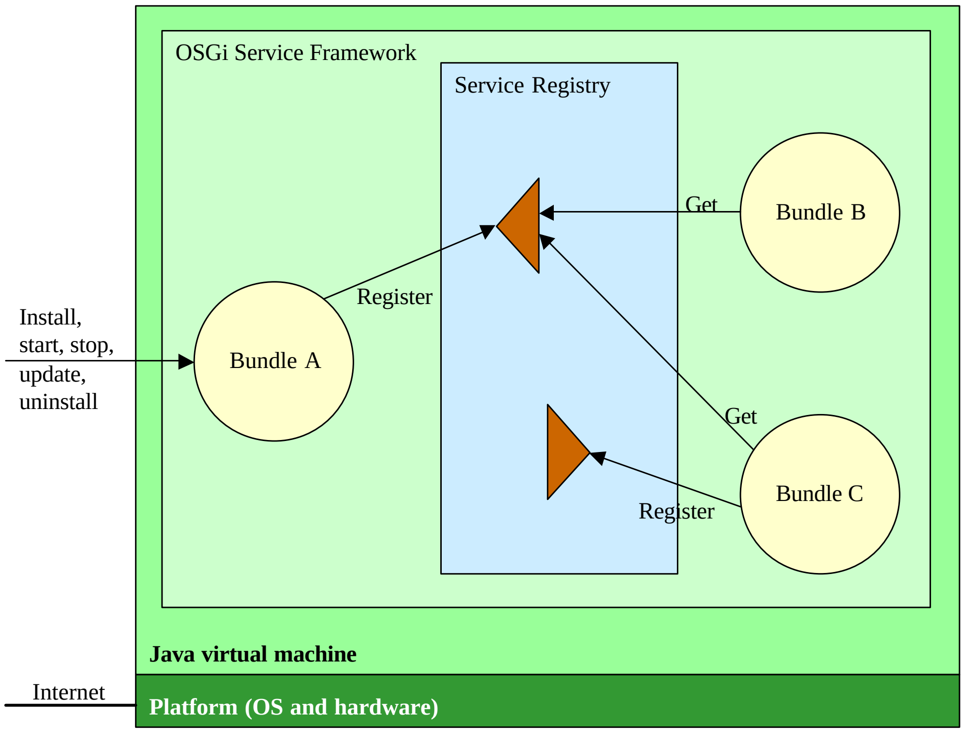


Figure 2.18: OSGi service registration (*from [6]*)

The role of the service registry is shown by an example in Figure 2.18. The figure shows three bundles (Bundle A, Bundle B and Bundle C), where Bundle A and Bundle C have each registered their service in the Service Registry. It can be observed in the figure that Bundle B and Bundle C make use of the service which Bundle A offers. Bundle C, as well as offering a service, makes use of the service which Bundle A offers.

When registering a service in the service registry, a bundle will add properties to the service entry. These properties, can include service version, generic description or other details. Figure 2.19 shows an example of the properties from a UPnP lamp device which has been added to the service registry. In this example, it would be a UPnP driver bundle that would listen for the new UPnP devices and automatically add them to the service registry.

The OSGi service specification does specify some properties which must be registered with a service, however this is only the case if OSGi has defined the service. For example, in Release 3 [63] of the specification, the OSGi alliance included service specifications for Jini and UPnP. Thus, when a UPnP device is added as a UPnP Device service to the gateway, there are certain properties which must be included when added to the service registry.

These properties are included with registered services in the service registry, and can be queried by other bundles to find the desired service.

Searching is achieved by looking for the service name (e.g., `org.osgi.device .x10.appliance`) and/or an optional LDAP query selection filter [65] (e.g. `(ROOM="kitchen",DEVICE="fan")`). Upon finding the service, the service registry passes a reference back to the calling bundle. If a matching service is not found, the service registry will return a `null` reference.

```
1.  ID=uuid:siemensTestDevice,
2.  UPC=123456789012,
3.  MODEL_NUMBER=1.0,
4.  UDN=uuid:siemensTestDevice,
5.  SERIAL_NUMBER=1234567890,
6.  MODEL_NAME=Vanilla, DEVICE_CATEGORY=UPnP,
7.  MODEL_DESCRIPTION=A Test Device for the UPnP Stack
                                              Implementation,
8.  PRESENTATION_URL=http://192.168.1.13:81/siemensTestDevice
                                          /presentation.html,
9.  MODEL_URL=/model.html,
10. DEVICE_IP_ADDRESS=192.168.1.13:81/siemensTestDevice,
11. TYPE=urn:schemas-upnp-org:device:binarylight:1,
12. FRIENDLY_NAME=Siemens Test Device,
13. MANUFACTURER=Siemens ZT SE 2
```

Figure 2.19: Example service registry properties for UPnP lamp device

2.2.3 Standard services

As well as defining the framework specification, the OSGi Alliance has also defined several standard services. A number of services have been defined in Release 4 of the specification which include; logging, security, user administration, XML parser and service tracking. An HTTP server specification has also been defined. This allows servlets to be used as a method of accessing bundles externally. Protocols such as Jini and UPnP specifications have also been added to the specification.

Creating specifications for UPnP and Jini means UPnP and Jini enabled devices can be accessed by services within the gateway. Specific Jini or UPnP drivers could be created which would add Jini or UPnP devices to the service registry for use by other services.

2.2.4 Services working together

The OSGi platform specification allows devices to be added and controlled by services. Since some bundles (normally drivers) add devices to the framework as they become available on the network, other services are able to select which devices they use at runtime. Therefore, the behaviour of a service can vary depending what devices, or services, it has available to it.

Devices and their protocols have been discussed. Being able to connect devices is useful, however the full power is not realised until new services which manipulate these devices are introduced. The notion that OSGi is the *glue* is emphasised here and can be clearly shown in the security service (alarm service) in Figure 2.20.

In this scenario, an alarm has been triggered. By using the service registry, the alarm service has been able to find an SMS service to send the owner a message that an intruder has been detected. The service also finds a USB web camera and UPnP VCR, and streams the data from the camera to the VCR. The service also finds a UPnP alarm bell which it rings. Finally, by using SIP instant messaging, the owner is sent a message to notify them. This example shows how one service is able to make use of a multitude of device protocols. It also shows how the alarm service uses other services (e.g. SMS service) to enhance the basic service.

The framework also has the ability to export devices to different networks by using bridging bundles [66]. For example, an X.10 lamp which is registered as an X.10 Lamp Device service in the service registry could be exported (using a bridging bundle) to a UPnP network. This means the device which was X.10 would appear in UPnP control points as a UPnP device. OSGi becomes the *glue* which allows services to use different devices regardless of the underlying device protocol [66].

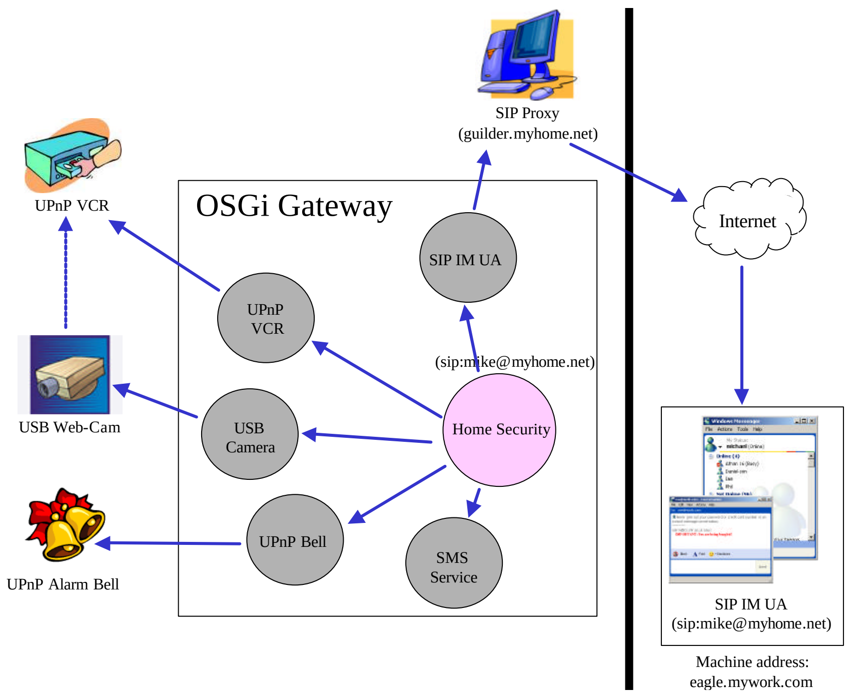


Figure 2.20: An example alarm service using many services

2.3 Summary

This chapter has discussed the motivation behind automating the home. There is an array of protocols used for automating the home. Two contrasting protocols have been reviewed in depth. Also, the chapter has discussed a selection of other home networking protocols. A middleware solution which brings together the devices which facilitate intelligent, smart, services has been outlined. Using the OSGi platform, user services are deployed which carry out tasks for the home user. These services are deployed on an OSGi gateway and make use of the devices available. The actual services which enable home automation are discussed in Chapter 4.

Chapter 3

The Feature Interaction Problem

A feature can be described as a component which is added to a software system to provide additional functionality. For example, in telephony, the basic service is being able to make a call. A feature can be introduced to enhance this basic functionality, perhaps to forward a call while the user is taking another call. As features are added to a system the phenomenon known as the *feature interaction* problem can occur.

The feature interaction problem was first highlighted by Bowen et al. in [67] at the Seventh International Conference on Software Engineering for Telecommunication Switching Systems. Since this publication, like minded academics and industrialists have held a series of Feature Interaction Workshops (FIW) [15–21, 68], with the first being held in Florida in 1992. The aim of these workshops has been to discuss the feature interaction problem and possible solutions.

Many interactions occur because features are developed and tested in isolation. When features operate on their own, they execute normally with no interference. However, problems can occur when certain features interact with one another, causing undesirable and unexpected outcomes [12]. Zave in [69] emphasises the point that interactions are an inevitable by-product of feature modularity. Zave also makes the point that not all interactions are bad, indeed some interactions are welcome.

Although the majority of the research has been within the telephony domain, some limited work has been carried out in other domains, for example: elevators [70], e-mail [71], web-services [72] and home networks [22].

One of the most influential papers to help understand the feature interaction problem was by Cameron et al. [12], which was later extended in [73]. These two papers were the first to present a taxonomy to help understand the feature interaction problem. These early papers provide a good taxonomy to show the *nature* and *causes* of interactions. However, there are two other taxonomies which are useful. Marples [74, 75] presents a taxonomy which is useful to classify features and interactions. Kolberg et al. [1, 76] then uses Marples taxonomy as a basis for a taxonomy for feature interactions (or service interactions) in networked appliances. In telephony there is only one service and many features, whereas in the home networking domain (which is the focus of this document) there are many services which may have some features.

Each of the three taxonomies will be discussed in the next section. Following this discussion, the chapter will then discuss previous approaches to the feature interaction problem. General limitations of these approaches will then be highlighted. The chapter will finish by examining current feature interaction work for the home domain.

3.1 Taxonomies for feature interaction

Section 3.1.1 and section 3.1.2 will discuss the taxonomies presented by Cameron and Marples, respectively. A discussion will follow in section 3.1.3 which gives a critical analysis of both taxonomies. Since the focus of this thesis is the service interaction problem in home networks, the taxonomy presented by Kolberg et al. will be presented in section 3.1.4.

3.1.1 Cameron's taxonomy

Cameron et al. highlight the problem of feature interactions in telecommunication networks and discus the importance of developing approaches to tackle the problem. The authors present ways of categorising the problem: the *nature* of the interaction and the *cause* of the interaction.

Nature of the interaction

Three dimensions of the nature of interactions were identified:

1. *Kind of features* – customer features vs. system features. Customer features are features the customer can use; these include call waiting (CW) or call forwarding (CF). System features are concerned with the administration of the system and include operations, billing and other system administration features.

2. *Number of users* – single user vs. multiple users. Single user interactions occur when different features are simultaneously triggered by a single user. Multiple user interactions occur when one user's feature interferes with another user's features.

3. *Number of network components* – single component vs. multiple component. Single component interactions only occur when there is one network component (switch, network node). Multiple component interactions occur when there is more than one network component.

Using these three dimensions, five categories of interaction were defined. These were:

SUSC (Single User, Single Component) are interactions between customer features when incompatible features are subscribed to on the same network node, or service control point (SCP). The interaction occurs because two (or more features) are designed to deal with the same trigger, but in different ways. An example of this type of interaction is between Call Waiting (CW) and Call Forwarding on Busy (CFB), where both try to handle the call, but in conflicting ways.

SUMC (Single User, Multiple Component) are interactions which occur when features available to one customer are deployed on two or more network nodes. An interaction arises here when one of the features on one node is not aware of a feature on the other node, and therefore some features are *missed*. Operator Services and Originating Call Screening (OCS) is an example of this type of interaction: as the operator makes the call, the subscriber's OCS feature is bypassed. It is worth noting that this kind of interaction is not common.

MUSC (Multiple User, Single Component) are interactions that occur when multiple users share the one physical phone line, hence they will be forced to share features. Some of these problems are caused when different households (e.g. those living in remote, rural areas) have to share a phone line. However, due to advances in technology, this is no longer such a problem. A problem can still exist between users within households. For example, parents may have OCS set to block calls to a premium rate number. A teenager may use CFB (on the same line as OCS) to forward all calls to a premium rate number. Therefore, when teenagers call their own number, the call will be forwarded to the premium rate number.

MUMC (Multiple User, Multiple Component) are interactions that can occur when two or more users access features on multiple network components. An example of this type of interaction is between one user's OCS and another user's CF. User A subscribes to CF and forwards calls to number X. User B subscribes to OCS and has number X on their OCS list. Therefore, if user B calls user A, their call will be forwarded to number X by A's CF.

CUSY (CUstomer SYstem) are interactions between a customer feature and a system, operations or administrative feature. The main focus with these types of interactions is billing.

Causes of interactions

Cameron et al. identified three main causes of interactions. These are:

Violation of feature assumptions: When features are developed, certain assumptions are made. These assumptions include: naming, data availability and signalling. When one of these assumptions is broken, an interaction can occur. For example, an interaction can occur between Calling Number Delivery (CND) and Calling Number Delivery Blocking (CNDB). CND delivers the directory number of the calling party, whereas CNDB makes the number private. Therefore, if CNDB is used, CND is unable to work as it does not have the available data.

Limited network support: The set of signals that can be sent from most telephones is limited to *, #, flashhook, disconnect and the ten digits (0–9). This is caused partly by the user interface on the telephone device and also by the signalling within the telephony network. Ambiguities can arise when two features use the same keypress. For example, an interaction can occur between a credit card call and a voice mail service. The credit card call allows users to make a call, paying with a credit card. When they want to make a new call, they press the # key. The voice mail service allows users to call a number to check voice mail by entering their PIN followed by the # key. If users check their voice mail with the pay by credit card feature, when they press the # key, do they mean a new call or to signal they have typed their PIN?

Intrinsic problems in distributed systems: This group of interactions relates to the general issues when large, complex, real-time systems are used. These problems include race conditions, resource contention and distributed support of features. An example of a race condition is between Automatic Call Back (AC) and Automatic Recall (AR). If user X dials user Y, and Y is busy, X's AC feature will automatically dial Y when they are idle. However, Y's AR feature will automatically call back the last person to call them when they were busy, in this case X. Therefore, when Y becomes idle, X's AC will try calling Y and Y's AR will try calling X. Thus, there is a race between the features.

The taxonomy presented by Cameron et al. shows there is more than one cause of an interaction, whether it is limited network support, or basic problems with distributed systems. However, the main point is that there is no simple solution to the problem.

3.1.2 Marples' taxonomy

Marples [74, 75] presents a different taxonomy for feature interaction in telephony. His classification is based on using a stimulus to trigger features and observing the response in a telephony environment. Through the results from experimentation, Marples devised a new taxonomy to classify interaction types.

Four categories were identified by Marples in this work:

Shared Trigger Interactions (STI) occur when more than one feature tries to respond to an event. For example, if a person subscribes to both CW and CFB, when a call is received, both features will react to the stimulus.

Sequential Action Interactions (SAI) occur when the action of one feature triggers a second feature. For example, this may happen between CFU and CW. Suppose person B forwards calls to person C. Person C has call waiting, therefore, when person A calls person B, the call will be forwarded to person C and get call waiting, which is not what person A expects.

Looping Interactions (LI) are a special case of SAI. Interactions can occur when a feature triggers another feature, which then triggers another feature, which triggers another, until a loop begins. An example of this may be between CFU and CFU, shown in Figure 3.1. User A tries calling user X. However, X's CFU forwards the call to Y, Y's CFU forwards the call to Z and Z's CFU forwards the call to X, and the loop starts again.

Missed Trigger Interactions (MTI) are types of interaction that occur when the presence of one feature prevents another feature from being triggered.

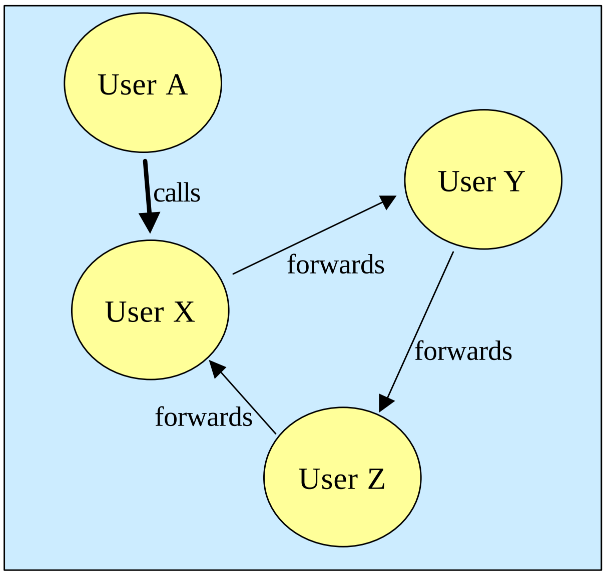


Figure 3.1: Looping interactions within CFU

3.1.3 Discussion: Cameron and Marples

The taxonomies presented by Cameron et al. and Marples show that interactions are complex with no single cause. Both taxonomies are valuable in helping to understand the feature interaction problem, but do not offer a solution. The list of causes of interactions Cameron presents is not complete. Since the work by Marples is at a more technical level (the signalling level) in call control, the taxonomy presented for his model in a particular environment is complete.

Although the single user and multiple user categories identified by Cameron are useful, the use of the single component and multiple component categories is doubtful. The reason is that features do not communicate directly with one other. Therefore, it makes no difference if the features are on one node, or placed on many [77]. Magill also states that the proximity of features is rarely exploited. If the proximity of features was explicitly used, SC and MC would be valuable.

Marples' taxonomy gives a better sense of interaction types than Cameron. If single component and multiple component (SC and MC) interaction types are removed from their taxonomy, it only leaves single user and multiple user (SU and MU) interactions.

The deregulation of the telecommunications market has exacerbated the problem. Nowadays, many vendors have the opportunity to deploy their features onto the network. Detecting interactions here is made harder. Reiff-Marganiec [78] discusses *intra portfolio* and *inter portfolio* interactions. Intra portfolio interactions are interactions which occur between features from the same vendor. Inter portfolio interactions are interactions between different vendor's features. Therefore, intra portfolio interactions (whether SU or MU) should be detected at design time. It is harder to detect SU or MU interactions if the features are from different vendors.

The two classifications, SU and MU, do not give a sense of *what* has caused the interaction. The taxonomy by Marples gives a better description of the cause of the interaction, for example it was caused due to a missed trigger. Since these interaction types are more focused, it allows the success of an approach to be measured. It also allows specific approaches to be developed for the type of interaction. For example, the approach presented by Marples is able to detect and resolve SAI, LL, STI interactions, but is not able to detect MTI. This then leaves MTI to be investigated further.

Both of these taxonomies help understand the problem in the telephony domain. Since this document focuses on the service interaction problem in the home networking domain, some themes can be used from the telephony work. However, there are many differences between the two domains.

In the telephony world the number of services and users involved in an interaction is small (typically no more than two or three users). However, in the networked appliance domain the number of services and appliances involved can be much larger. For example, a service could try to turn *all* devices off.

Like the telephony market, the home networking market will be competitive. Different vendors will sell their solution, which will be more complex and *clever* in a bid to stand out from the crowd. This complexity and interworking will exacerbate the service interaction problem in the home.

One of the main differences between the home and the telephony problem is *how* the interaction occurs. The home has the possibility of many more interactions occurring indirectly. For example, a service starts a device which causes movement in a room. The movement is then detected by another service which monitors movement. It is possible that the movement in a room causes no harm; however, if it were a burglar alarm, unnecessary movement is clearly not welcome.

To develop a taxonomy for interaction types in the home, there are subtle differences between the two domains which have to be considered. In the home domain, assume there are multiple services controlling multiple devices. Here, who is the user? Is the user the occupant of the home, or is the user the service which control the devices? Also, what is the component? Is it the service, or is it the device? This would depend on who the user was.

If Cameron's taxonomy were used for the home, the taxonomy would need to be extended to include more relationships. A relationship for occupant and service would have to be created, and also another relationship for service and device. However, would there then be a need for a relationship between occupant and device?

Neither Marples or Cameron discuss indirect interactions. This may be fair as few interactions have been identified that occur indirectly in telephony. However, they are important in the home domain.

Clearly, modifying Cameron's taxonomy for the home is unworkable. Since their taxonomy was developed explicitly for telephony, it is hard to adopt it for the home domain.

However, the taxonomy presented by Marples moves away from the concept of users and looks at the interaction types from a different angle. Work has been carried out to adapt this taxonomy for use with networked appliances.

3.1.4 A taxonomy for networked appliances

One taxonomy for feature interaction in networked appliances currently exists. It was first presented by Kolberg et al. in [76]. The taxonomy was later updated in [1] to explicitly include the working environment of devices. The environment here refers to factors such as room temperature, lighting levels, etc.

Multiple Action Interaction (MAI)

These types of interaction occur when two services try to control the same device, shown in Figure 3.2. It could be argued that these interactions are not necessarily bad as two services may, simultaneously, try to turn the same device on. The problem arises when one service requires one setting and the other service requires another setting. Or, when one service has finished with the device and wants to turn it off but the other service still requires it. In general, however, having different services control the same device is not desirable as it can lead to negative interactions and ambiguity.

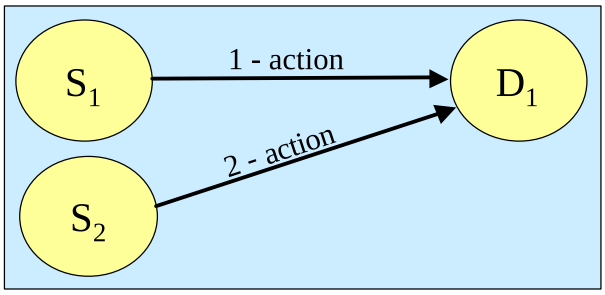


Figure 3.2: Multiple Action Interaction *(adapted from [1])*

Shared Trigger Interaction (STI)

The second type of interaction is when a device (typically a sensor) informs two services of a change, and these services carry out conflicting actions. This is shown in Figure 3.3. Although Kolberg et al. argue that all these types of interactions are *bad*, this may not be the case. If a sensor does alert two services, the services may not necessarily carry out conflicting actions. If they did, clearly there would be a negative interaction.

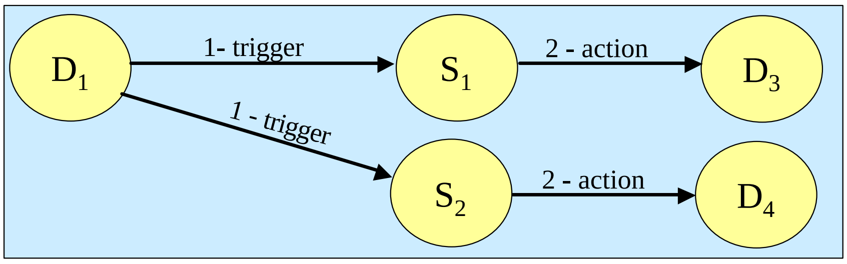
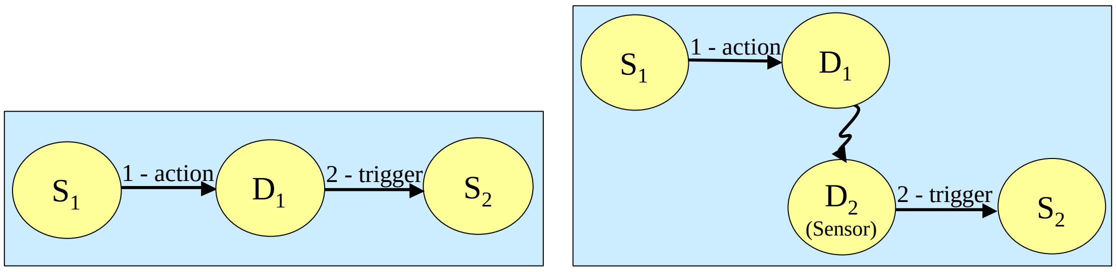


Figure 3.3: Shared Trigger Interaction *(adapted from [1])*

Sequential Action Interaction (SAI)

This type of interaction occurs when a service (or indeed another device) sends a command to a device, and the device in turn notifies another service, shown in Figure 3.4(a). However, these types of interaction can also occur through the environment (Figure 3.4(b)). This can happen when a service sends a command to a device which, in turn, affects a sensor. The sensor can then notify another service.



(a) Direct trigger (b) Trigger via environment

Figure 3.4: Sequential Action Interaction *(adapted from [1])*

The work by Marples identified looping as a separate type of interaction. However, for the work in appliances, looping is considered to be a special case of SAI.

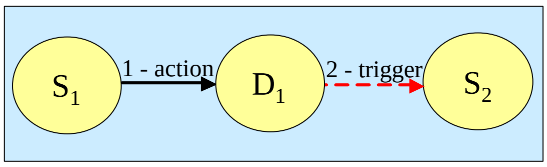
SAIs can be either positive or negative types of interactions, except looping which is never positive.

Missed Trigger Interaction (MTI)

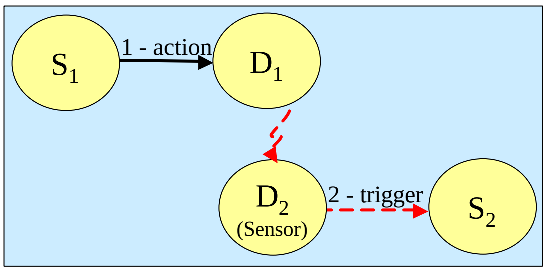
The last category of interaction identified was Missed Trigger Interaction (MTI). These interactions can occur when one service prevents the other from operating, shown in Figure 3.5(a). This occurs when a service sends an action to a device which stops it sending triggers to another service, illustrated by the dashed line in Figure 3.5(a). Again, like SAIs, these types of interactions can also occur via the environment, Figure 3.5(b).

3.1.5 Discussion: taxonomy for networked appliances

This taxonomy, is valuable as it allows us to understand the feature interaction problem in networked appliances (NA). It shows that the interactions can occur either directly between devices and services



(a) Direct trigger



(b) Trigger via environment

Figure 3.5: Missed Trigger Interaction *(adapted from [1])*

or indirectly, through the environment. This taxonomy highlights the importance of the environment in networked appliances. The taxonomy is also useful for measuring the effectiveness of future approaches for feature interaction in the home.

3.2 Approaches to the problem in telephony

Three taxonomies have been presented to help understand the feature interaction problem; however, none offer a solution. Over the past decade considerable progress has been made to easing the feature interaction problem, yet an agreed solution has still to be found.

Current approaches to the problem can fall into two categories: off-line approaches and runtime approaches (often referred to as on-line approaches). Each type of approach will be discussed in the following subsections.

3.2.1 Off-line approaches

There are two main areas within off-line approaches: software engineering approaches and formal methods. Software engineering approaches are carried out during the service creation process (design time). Formal methods can be used to analyse existing features, or to analyse features at design time to detect feature interactions.

Software engineering approaches

This approach uses general software engineering techniques to help the feature interaction problem, normally at the requirements or specification stage of the software process.

These techniques can aid the feature interaction problem indirectly or directly. Indirect techniques are where good software engineering techniques are applied. It is through the introduction of these approaches, and the rigour they bring, that interactions are eliminated. Direct techniques are achieved through explicitly testing and developing methods to detect and resolve feature interactions.

Finding feature interactions at design time means the problem can be fixed relatively early. Generally, the feature interactions are solved by hard coding the solution [75].

Within software engineering there are two main approaches:

Focused techniques are where a particular technique which is used in software engineering has been introduced into the service creation process, with the aim of eliminating feature interactions. Examples of this type of work include Use Case Maps (or models) [79–81]. This category also includes filtering [82], which involves removing unlikely combinations of features and applying a more complex approach to likely feature combinations. This is useful when there are a large number of features in a system.

Process models are where existing software engineering techniques are taken and adapted for the service creation domain. Their aim is to eliminate feature interactions. The emphasis with this approach is detecting feature interactions at an early stage in the life cycle. An approach was developed, SIHP (Service Interaction Handling Process), as part of the EURESCOM (European Institute for Research and Strategic Studies in telecom) Project P509, discussed by Kimbler [83]. The approach which came from this project was to develop a phase in the software development process explicitly for feature interaction detection.

The main advantage of software engineering approaches is that they provide a strong, industrial scale approach to the feature interaction problem. One of the problems with process models is that they do not offer a solution to the feature interaction problem. They only offer a framework for another feature interaction approach to be placed in.

Formal methods

Formal methods are techniques which involve the system being modelled and features being analysed using formal reasoning techniques. There are a number of reasons for using formal methods for feature interaction detection. Using a formal description (or model) forces assumptions to be made explicit. Also, automated analysis and reasoning tools are available to detect interactions.

There are three types of formal methods approaches (defined in [13]):

Properties is where theorem proving or model checking is used to identify inconsistencies of properties once features are combined. Property languages include TLA (temporal logic of actions) [84] and LTL (linear temporal logic) [85].

Behaviour is where the behavioural description of features and the basic service is defined. These specifications can use specially developed tools, or use standard languages such as LOTOS or CSP. In this approach, if the results of analysis show deadlock, unreachability or abnormal termination for example, then an interaction has occurred. Examples of these approaches include Plath and Ryan [86] who use CSP to detect deadlocks.

Properties and behaviour model is where the feature and basic service is defined by properties and behaviour. In this approach, interactions occur when features plus their basic service satisfy their respective properties individually, but when they are combined, the conjoined properties are not satisfied. Examples of this work include [87] where TLT is the property language and Promela is the behavioural language.

Although considerable research has been carried out using formal methods in feature interaction, they do have some problems. When creating the models, they have to be complete. If they are under specified, the frame problem is introduced (what is not changed by the feature) and over-implementation generates false positives. Getting the balance correct is not straightforward.

Despite their problems, formal methods do have advantages. Using formal methods forces assumptions to be made explicit. These assumptions cause ambiguities, which can cause interactions.

3.2.2 On-line approaches

On-line approaches are approaches which are carried out at runtime. These approaches have many advantages over off-line approaches which include:

- They can be applied in an operational system.
- They support a quick time-to-market for new features as the feature does not have to be tested against all other features in the network.
- On-line techniques are future proof, since they are running in a live network. As new features are added, the techniques can deal with them. However, the fact that they do run on a live system can be seen as a slight processing overhead. An approach which is future proof is important in the multi-vendor market since new features are introduced frequently, and therefore will not know about one another.
- In larger, legacy systems, on-line approaches have the advantage that there is no need to change the existing code.

- On-line techniques do not require a specification of the system and its features.

Although on-line approaches can be very powerful to help avoid or detect and resolve interactions, there are very few approaches available. The main reason for this is that they are hard to develop. Firstly, an on-line approach may have little or no knowledge of a service. Without the knowledge, how does the approach know what service it is looking at, therefore how does it prioritise them?

Within on-line approaches there are two classes of approach regarding the location of control:

Feature manager based approaches are where a manager on the network observes the call control process (in telephony). The managers will observe features and either avoid the interaction, or detect the interaction and apply a resolution technique. One of the drawbacks of using managers is that they generally have a central point of control. If managers are distributed, they can communicate with one another, however the communication is an unwelcome overhead.

Negotiation based approaches are where each feature has an agent which has the ability to communicate with one another and negotiate a resolution. Within negotiation there are three different approaches:

1. *Direct negotiation* where agents talk to one another.
2. *Indirect negotiation* where an entity carries out the negotiation between the agents and routes the messages. The negotiator will also ensure the process is carried out correctly and that there are no deadlocks. If the agents can not resolve their problem, the negotiator can decide the outcome. The outcome could be based on information it has gathered during runtime.
3. *Arbitrated negotiation* which takes the scripts from each agent and finds a resolution for the interaction.

Whether the on-line approach is in the form of a manager, or a negotiation approach, they will require some information. The information these approaches require to run can be gathered in one of three ways.

A priori information is information which is gathered at design time. The information can then be presented as per-service information or as a per-service pair matrix. The disadvantage with a per-service pair matrix is that it starts to get extremely large as the number of services grows.

Captive environment is where information for detection and resolution is gathered at runtime in a closed system. The information gathered from the closed system is then used in a live system. Using a closed system means testing can be kept simple without the complexities of a live system with live calls.

During runtime is where no special information is required. These approaches gather the information they need at runtime in a live system.

Griffeth and Velthuijsen [88] developed a negotiation approach which employed a negotiator. They did not use an arbitrator as this requires all information from the agents, and they deemed this overhead to be too much. The issue of privacy also becomes an issue here. However, one of the main drawbacks of their approach, and negotiation approaches in general, is that the solution must be known before negotiation starts. The example given by Griffeth and Velthuijsen in [88] highlights this issue. They give the example of an interaction where Blocking Call Number Delivery (BCND) prevents Calling Number Delivery (CND) from working. Therefore, when a user with BCND calls a user with CND, the user is unable to see the number, as delivery of the number has been blocked by BCND. Using the approach presented, the result of negotiation is to allow the person's name to be delivered, rather than their number. Therefore, to arrive at this solution, both the interaction and answer must be known.

An example of one on-line approach which included a feature interaction manager using a priori information was by Cain [89]. The approach here used a matrix to find interactions between features. Although this approach did work, when a new feature was added, the new feature had to be checked against all other features in the matrix off-line. Clearly, this is not scalable as testing one feature against all others is very time consuming.

Marples and Magill [74] developed a feature interaction manager which required no information. This approach assumes that an interaction occurs when two or more features respond to a stimuli. The approach uses a roll-back and commit algorithm to find a suitable resolution. This work was extended by Reiff-Marganiec [78] to include a more sophisticated resolution technique.

For most on-line techniques, after the interaction has been detected, essentially priorities are used to resolve the conflict.

On-line approaches offer the best way forward in a changing network where new services are being added. However, both off-line and on-line approaches do have their limitations.

3.3 General limitations of previous approaches

Although off-line approaches do have advantages, their main problem is that all features in a system have to be known. This is not such a problem in single vendor environments. However nowadays, telecommunications networks are large, multi-vendor environments, which hampers the use of off-line approaches. The first time features meet are when they are deployed onto a live network. This is the only point where interactions between different features from vendors can be detected.

Another problem with formal methods is that all features have to be modelled and analysed. These models are abstractions so there is a danger they are not a true representation of the system. Further, the problem of creating accurate models means all features and their attributes are required. In legacy systems where features have changed, and documentation is often poor, acquiring this information is difficult. Further, since there are many vendors developing software for telecommunication networks, vendors are unlikely to share detailed information with their competitors. This makes formal methods more suitable for detecting intra-portfolio interactions.

Scalability becomes an issue when there are hundreds of features. Modelling all features, and the system, introduces state space explosion [78].

On-line approaches have their limitations too. These problems include processing overhead, the decision must be made at runtime. Also, scalability becomes an issue. In telephony, networks are extremely large and complex. Developing a manager which can handle this scale is challenging. Since the processing is being handled by a manager, this can cause a bottleneck.

Negotiation approaches have their own problems. As previously discussed, the solution needs to be known before resolution can start. Drawbacks of using an arbitrator include a higher communication overhead. Privacy is also an issue here as the agents have to send their scripts to the arbitrator for it to analyse.

Manager based approaches have limitations, mainly due to the information they require. Also the fact they are centralised becomes an issue. However, from all the approaches, a manager which gathers its information at runtime is the most powerful and flexible.

In the home networking and networked appliances domain, some feature interaction work has been carried out. The next section discusses these approaches.

3.4 Feature interaction in home networks

Feature interaction in home networks is a relatively new area, with the first paper being published four years ago [76]. This paper highlighted the problem of feature interactions with networked appliances, but did not offer any solution. Since then only two different approaches have been presented which explicitly tackle the home: one off-line [22] and one on-line [1, 90]. An approach developed for service creation for SIP in internet telephony has also been presented by Wu and Schulzrinne [14], which they also use for feature interaction detection and resolution only within multimedia systems in the home. Although their approach is not a general solution for the feature interaction problem in the home, it is worthwhile including.

An off-line approach which discusses feature interactions in office buildings by Metzger [23] has been included in this section, as many of the issues for feature interaction in the office are similar to those in the home.

These approaches will be discussed in more detail in the next section.

3.4.1 Current approaches for feature interaction in the home

Nakamura et al. [22] present an off-line approach for the feature interaction problem in home networks. They developed a tool which was able to take the specifications of the home network, including devices and services, and model them. Each device was regarded as an object with properties and methods. The methods used in their model relate to actual methods within the device. For their approach to work, the authors assume the APIs to devices would be published and be the same for all device types. This assumption is valid as protocols, such as UPnP, publish interfaces and method calls for each device type.

The authors of this paper also recognise the environment as being important to detect interactions. Therefore, the environment is modelled as a global object within their model.

Using their detection tool, the authors claim to detect 43 appliance interactions, with 24 of these interactions being at the environment level.

Although the approach developed does detect interactions within the home, it has the major drawback of being rigid. The way in which services were defined suggests that services in the home do not change, and exhibit the same behaviour every time they are executed. This may not be the case as home services should be designed in such a way as to make use of service platform technologies which allow them to find other devices if their first choice is not possible. Also, there are likely to be many companies developing services for the home. Using the OSGi gateway as an example, this has an open specification that allows other companies to quickly and easily develop services for the home.

As well as services being tightly defined, the devices used are also defined. All new home networking protocols (including the service deployment platforms) support devices joining and leaving the network. It is possible that the home network changes on a daily basis. Coupled with the fact that users can configure services differently makes an off-line approach unworkable as a general solution for the home.

Wu and Schulzrinne [14] present an approach which is specifically aimed at service creation. They developed a markup language called LESS (Language for End System Services) [91]. Although the language was developed for call control in SIP, its use was extended for multimedia devices in the home. Their approach only detects interactions at the device level. It does not take into account the environment. This is one of the main drawbacks of their approach.

Although the approach by Nakamura et al. is the only paper for feature interactions in home networks, work by Metzger et al. [23] looks at the problem within the office domain. The reason for including this paper is that these authors discuss the importance of the environment for detecting interactions. However, since this is another off-line approach, its general applicability is limited within the home networking domain. Interestingly, their approach was extended in [92] for feature interaction detection within automobiles. Since the network and services in the car are less likely to change, using an off-line approach in this domain is valuable; however, this is outside the scope of this work.

Chapter 4

Services Enabling Home Automation

In Chapter 2, devices and protocols were discussed as well as a framework for executing and managing services for the home (OSGi framework). This chapter will discuss the kind of services (home security, entertainment or safety) which are anticipated to be on a home gateway.

Simply connecting smart devices in a network adds little value to the home. In fact, it does not offer much more value than the conventional stand-alone devices of today. The real benefit of connected devices is when user services interact with, and control, a series of devices. For example, when the burglar alarm is triggered, rather than the home security service simply sounding an alarm bell, if a camera and VCR were available, the service may use the VCR to record video from the camera. Further, if an SMS messaging service were available, the owner could be informed via SMS that an intruder was detected in their home. The user may have another service which allows them to connect to the home via their mobile telephone and remotely view the home through various cameras placed around the it.

A middleware solution (such as that defined by the OSGi Alliance), allows services to use a range of devices without having to worry about the underlying protocol. This gives services the ability to dynamically select which devices they use, depending on what is available. Therefore, a service may behave slightly differently over time as the home network can change with devices joining and leaving. However, the consequence of a service automatically selecting devices, and subtle changes in the service's configuration, can lead to incompatibilities between services in the home.

To show these incompatibilities, examples will be used. The next section of this chapter will describe some of the services which may be found in the networked home. Using these services as examples, the chapter will then discuss negative interactions which can occur under certain circumstances between these services.

4.1 User Services

As this section will illustrate, services for the home cover a wide spectrum in terms of functionality. Here, emphasis is given to the aspects concerned with services which control devices.

The services used in this section have been inspired from the literature [6, 22, 66, 93–95]. The examples used cover a broad range of areas – from security, through climate control and entertainment, to energy conservation. While these services are likely to be found in the home, more importantly for this thesis, they also have potential to interact with one another. The interactions can be both positive and negative. The five services clearly show the advantages and potential problems of the networked home as described below.

4.1.1 Home Ventilation and Air Conditioning (HVAC)

HVAC is the climate control service which integrates the control of the heating and air-conditioning. A thermometer is polled by the service for current temperature or, in the case of a UPnP thermometer, listening parties are notified of a change in temperature.

If the temperature in the house rises to a certain level, the service will start the air-conditioning. If the temperature drops below a certain level the service starts the heating. The service also offers the option of energy efficient climate control. Here, the service includes the control of windows (open or close), an air fan, and also window blinds. The service is aware of the outside temperature, therefore it can determine how the inside temperature will be affected if a window is opened.

4.1.2 Home Security Service (HSS)

This service integrates various security and safety aspects. The service has two features: the basic alarm feature and an away from home feature.

The alarm feature works together with movement sensors in the home. If any of the sensors detect any movement, the service is notified and the alarm is triggered. Upon being triggered, the service records the picture from a security camera onto the VCR and then calls the police.

The security service also has an *away from home* feature. The aim of this feature is to give the impression that the home is occupied during the absence of its owners. The feature achieves this by turning on lights, drawing curtains, as well as turning the television and stereo on. These actions are carried out in certain sequences. For example, the feature can turn the living room lamps on and close the curtains at dusk. Perhaps another sequence is room lights being turned on and off to simulate the user passing through rooms.

4.1.3 Power Control Service (PCS)

The power control service can enable the home owner to obtain cheaper electricity by giving up some control to the power service through using a separate on-switch. With this separate power switch, the user specifies that the appliance is ready to run. However, it is the power service which actually switches the appliance on. The times for determining when appliances are to be turned on can be created by the service. The power control service may work in conjunction with a service provided by the energy supplier, which sends energy costs. Based on these costs and user preferences, the service can determine when to turn devices on. This allows the power service to determine the exact time the appliance is used, and hence to manage power consumption during peak demand. For example, the user may turn the washing machine on at 8am before going to work, but (after checking with the electricity supplier) the power service knows electricity is cheaper after 2pm. Therefore, although the owner has turned on the washing machine at 8am, it will not actually start until 2pm when electricity is cheaper. However, this does assume the energy supplier differentiates between peak and off-peak usage.

The service also offers an option of running the house efficiently. When the home is vacant, energy can be saved by switching off unnecessary appliances, such as lamps and televisions.

The energy provider Electricité de France (EDF) are one company to have piloted such a project. Since it costs energy companies substantial sums of money to put extra power into the electricity grid during peak hours, they would like to reduce this peak.

The scheme EDF propose involves giving users energy quotas during peak hours. If users stay below their quota, they receive cheap electricity. If they go over their quota they are penalised and charged more for their electricity. Although this is a pricing mechanism by the energy supplier, the PCS can be used to control when energy is consumed to ensure quotas are used efficiently. This could be used to start the washing machine mid-morning, rather than peak times. Therefore, the service is able to help reduce costs.

4.1.4 Home Entertainment Service (HES)

This service controls entertainment devices in the home, such as the television, stereo system, and VCR/DVD players. A key feature of this service is its ability to monitor the owner's viewing habits. These viewing trends and patterns can then be used to automatically record certain popular television shows. The service can also check the owner's diary and, if they are out, will automatically record their programmes.

4.1.5 Communications Support Service (CSS)

This service supports the use of email and the telephone. The user can be notified by a message displayed on the television when a phone call or email arrives. The message will contain the caller for a phone, call and the sender and subject line for an email. Optionally, users may subscribe to a feature which turns down the volume of the television and the stereo when a phone call arrives.

4.2 Conflicts between services (Interactions)

The five services and their capabilities have been discussed. Each service is able to use a number of devices to achieve its goal. However, since services are automatically controlling several devices, each carrying out their own role, it is inevitable some conflicts will occur. This section highlights the issue of negative interactions.

Feature interactions can occur for a number of reasons. The most common are conflicting goals and broken assumptions [73]. Services pursue specific goals, e.g. to switch off unnecessary appliances to save energy. However, an extra feature of the security service is to switch on lights to pretend the home is occupied. Clearly, the goals of the features conflict.

Similarly, services need to make certain assumptions about their environment. For instance, the security service assumes that when activated, nobody is at home, therefore appliances will not be used and there should be no movement. However, the climate control service may control the blinds to prevent the sun from unnecessarily heating up the home. Here the assumption of the first service (security) that no appliances will be used, is violated by the second (climate control).

The two interactions described above are examples where the interaction occurs between two independent services. However, interactions can occur within the same service. Assume the home security service has two features: monitor the home for intruders by detecting motion, and the away from home feature. The owner is away from home and the alarm is set, but they have also set the away from home feature to close curtains and turn lights on in the evening. When the curtains close, the alarm is unnecessarily triggered. Thus, two types of interactions have been identified: interactions between two different services, and interactions within the one service, *inter-service* and *intra-service* interactions [1], respectively.

4.2.1 Inter-service interaction

These are interactions between two different services. Since services are developed in isolation by different vendors, it is impractical to test all services against each other. Even if all services are tested against one another, it is impossible to know the devices a user utilises and how they configure each service. Therefore (as discussed in Chapter 3), a runtime approach is the only viable solution to detect interactions between services in the home.

4.2.2 Intra-service interaction

In contrast, intra-service interactions are interactions which occur within a service. These types of interactions should be discovered at design time. However, like inter-service interactions, it is difficult to know the devices a service will use and how the user configures the service. Therefore, finding *all* interactions at design time may be difficult. As these are valid interactions, the approach developed should be designed to handle these types of interaction.

4.3 Interaction examples

From the services described in section 4.1, a number of potential interactions have been identified.

4.3.1 Security vs. Power Control Service

Imagine the situation where the owner of the house is absent, and the *away from home* (AFH) feature of the security service is active. Also, the efficient energy usage option of the power control service is active. The power control service will switch off lamps and televisions. However the away from home feature will switch lamps and the stereo on to give the impression that somebody is at home. The problem is caused by two overlapping requests from the two services trying to control appliances in conflicting ways (one service turns devices on, while the other turns them off). Since the interaction is caused by two different services, this is an inter-service interaction. The type of interaction is a Multiple Action Interaction (MAI) (section 3.1.4), as two services are trying to access the same devices.

4.3.2 Security vs. Entertainment

The security and the entertainment services both try to control the VCR. Once triggered, the security service records the picture from a camera to the VCR. However, the two services overlap when the entertainment service tries to record a television show at the same time – disabling part of the security service functionality. Unlike the former example where the first action (switching off lamps) was finished before the second occurred, the problem here occurs because the first action has not yet completed (recording the camera picture) when the second is triggered. Again, as this interaction is caused by two independent services, this is an inter-service interaction. The type of interaction is also an MAI, as two services try to control a single device.

4.3.3 Security vs. Climate Control

The away from home feature of the security service turns devices on and draws curtains (or blinds) following a pre-defined sequence to make it look as if the owner is at home. Since the owner is away the climate control service has been set to keep the home at a comfortable temperature, but by doing so as cheaply as possible. The climate control makes use of blinds and opening windows as these are cheap alternatives to heaters and air conditioners.

Suppose the away from home feature is active, and turns lights on and closes the blinds to make it look as if the owner is at home. To heat the home, the climate control service needs to raise the blinds to let some sunlight (heat) in. There is a conflict here as one service wants the blinds open, whereas the other wants them closed. This is an inter-service interaction and is of type MAI.

4.3.4 Climate control vs. Security

An interaction can occur through a number of different events between climate control and the security service. If movement in the house is detected, the security service is notified. It interprets movement as an intruder, triggering the alarm. The climate control service may lower the blinds, start the ventilation fan or even open windows. All of these actions create movement, which the motion sensor detects and notifies to listening parties. This consequently triggers the security alarm. This is an inter-service interaction. They are all of type Sequential Action Interaction (SAI) (section 3.1.4), as the action of service one triggers another (service) causing a negative interaction.

4.3.5 Power Control Service vs. Climate Control

During cold spells the climate control service keeps the home at a stable and warm temperature. If the owner is not at home the power control service will turn off all appliances to save energy. In doing this, it interacts with the climate control service by disabling devices (including the thermometer). This means the climate control service is not notified of the dropping temperature in the home. As the home gets colder, the water pipes may freeze. This is an example of a missed trigger interaction (MTI) (section 3.1.4), because the power control service turns off the thermometer, which means the climate control service is not aware of the low room temperature.

4.3.6 Within climate control

It is possible that there may be interactions within the climate control (intra-service interactions). The following scenarios should be discovered at design time as they are interactions within a single service. As discussed, inter-service interactions are virtually impossible to detect until runtime as a vendor will not know what other services a user will have in their home. Further, the problem is worsened by how these services are configured. The examples are included in this section for completeness and to show that any solution developed should be able to handle both types.

Action conflicts: energy waste

An interaction may occur within the climate control service. A thermometer notifies interested services of any change in temperature. If the temperature rises above a certain level and if the user has activated the energy efficient climate control option, the service may start the air conditioning (for normal cooling) and open the windows (for energy efficiency). Clearly, these two actions are not compatible as the open windows compromise the correct and efficient functioning of the air conditioner. This interaction is caused by the energy efficient component of this service, which will in fact waste energy if not correctly integrated. Clearly, this is an intra-service interaction. This is a Shared Trigger Interaction (STI) (section 3.1.4) as the change in temperature triggers the service to use two devices which, although they both aim to cool the room, may conflict in doing so.

Action conflicts: energy waste through looping

An infinite loop can be caused within the climate control service. Assuming the service consists of a heating component controlling the heating and a cooling component controlling the air-conditioning, the following interaction may occur. Reaching a certain temperature will trigger the climate control service, resulting in the air conditioning being activated. However, this may cause the temperature to drop below the pre-set temperature for the heating to be activated. The heating will increase the temperature. However, this again may result in the temperature being too high and the air conditioning being started again! This is an intra-service interaction, but is a special case of type SAI as the action of the heater (to heat the room) causes the air-conditioner to come on.

4.3.7 Within Security

An interaction can occur between the away from home feature and the alarm feature. The alarm is armed and monitoring the home for movement. At dusk, the away from home feature will follow a sequence where it turns lights on and closes some curtains (or blinds). If not configured properly, it is possible that when the away from home feature tries to draw curtains (or blinds) this causes movement, which in turn triggers the alarm. This is an intra-service interaction and is of type SAI.

4.4 Summary of the problem

Table 4.1 shows a summary of the interactions manually identified in the previous section. As these have been identified manually, testing may show interactions which have been missed.

The table shows the interaction (as a '•'), the type of interaction and the section where the interaction was described. The lightly shaded areas show intra-service interactions, with the white background being inter-service interactions. The abbreviations for services in the table are as follows:

- HSS:Alarm – Home Security Service – basic alarm feature
- HSS:AFH – Home Security Service – away from home feature
- PCS – Power Control Service
- HES – Home Entertainment Service
- HVAC – Heating, Ventilation, Air-Conditioning (Climate Control Service)

Generally, on their own these services operate in a coherent and consistent manner however, there are occasions where services clash. The clash, or interaction, may be caused by two different services (inter-service) or a problem within the service (intra-service). Many intra-service interactions should be discovered at design time. However, the *harder* interactions to solve are those between two different services.

	HSS:Alarm	HSS:AFH	PCS	HES	HVAC
HSS:Alarm		● SAI (4.3.6)		● MAI (4.3.2)	● SAI (4.3.4)
HSS:AFH			● MAI (4.3.1)		● MAI (4.3.3)
PCS					● MTI (4.3.5)
HES					
HVAC					● STI (4.3.6.1) ● SAI (4.3.6.2)

Table 4.1: Interactions identified from examples

4.5 A new approach to the problem in home networks

This chapter has discussed several scenarios where interactions can occur in the home. Although the service interaction problem in the home is similar to the feature interaction problem in telephony, there are some differences.

The main difference is that many more interactions happen indirectly. They happen through an additional level – the environment. Here, the environment can be room temperature or movement in the room, for example.

Previous approaches in telephony (such as those discussed in Chapter 3) do not use the environment to detect interactions. A-priori and captive environment approaches are not suitable because in the home a service can behave differently depending on the devices available and how a service is configured. Further, the configuration of services and devices in the home can easily change as home networking protocols have been designed to specifically support ad hoc networking (UPnP for example).

Another issue in the home is that all homes are likely to be different: no two homes is likely to have the same service and device configuration. The services and devices a user will purchase are unlikely to be from the the same supplier. Cost is the likely reason for this as replacing all electrical appliances at once will be costly. Further, since the home electronics market is competitive, there will be many manufacturers and service vendors. However, even if all homes were to have the same service and device configuration, owners will generally want to personalise and configure services to suit their lifestyle. This may cause the behaviour of the service to change and, in turn, the devices it may use.

Therefore, the aims for a new approach for the home should:

- Include the environment as interactions do happen here as well as at the device level.
- Avoid negative interactions while allowing devices and services to cooperate to achieve a common goal.
- Accommodate devices joining and leaving the network.
- Accommodate home service change, including updates to existing services, or services being added or removed.
- Be flexible to change, regarding new networking protocols and new types of device.
- Be independent of service, as it is impossible to know all services a user will have and how they are configured.
- Need little user configuration and intervention; a home owner will not be interested in feature interactions, so the automated home will be expected to *work*.

- Must make a decision quickly on whether a negative interaction will occur.
- Be scalable to handle all devices and services a person may have in their home.

4.6 Summary

This chapter has outlined services which are expected to be in an intelligent home. Interaction scenarios have also shown that service interaction in the home is a real problem.

The chapter also introduced the concept of the environment and has demonstrated why it is crucial in the home. By using the environment, it is possible to see how services can interact with one another.

The chapter finished by describing the problem and outlining the aims for a new approach. Using these aims, a new approach has been developed for service interaction avoidance in home networks. This is discussed in detail in the next chapter.

Chapter 5

An Environmental Approach

5.1 Introduction

Chapter 4 outlined several aims an approach for the home should meet. This chapter presents a new approach, which achieves these aims, for the service interaction problem in the networks.

Traditional approaches to feature interaction have been service centric, concentrating completely on this aspect. Further, these approaches are typically off-line. Some work by [22, 23] does concentrate on the device and environment, rather than just the service. However, this approach is off-line. As previously discussed, although off-line approaches are useful for detecting some interactions, they only work in a system where all services and devices are known. In the home this is unlikely. A home is likely to have different services from many vendors. Even if all services are known, the configuration of the devices in use will be unclear as devices will join and leave the network. This is shown in the networking protocols for the home where automatic configuration and setup is crucial – UPnP or Jini, for example.

Also, the services which control devices may behave differently depending on the devices available at runtime. This makes an off-line approach unworkable for general use within the home.

Therefore, since a new approach has to be flexible to cope with change, both in terms of devices and services, the new approach has to be an on-line approach.

There are two approaches within on-line work: negotiation and feature interaction managers. As discussed in Chapter 3, negotiation approaches have the drawback that the solution to the interaction has to be known before negotiation can begin. Although feature manager approaches have disadvantages such as being centralised and can have scalability issues, these are not such a problem in the home. The residential gateway is centralised where all devices and services register on the same platform. Also, scalability is not such an issue in the home as the number of devices and services a user can have will be relatively small (in comparison to a telecommunications network).

Since the automated home is to make life easier for occupants, users will not be interested in managing interactions in the home. Therefore, this approach has to be invisible to users. By using an on-line manager, this is achievable, as it makes decisions in the background whether a negative interaction will occur or not. If a negative interaction occurs, it should be avoided. However, if the interaction is positive, it should be allowed, as this is desirable.

These are some of the important goals which this new approach has to achieve. The remainder of this chapter will discuss the role of the environment and how it can be used to manage interactions. To understand how the environment is affected, the devices which affect it are included in the model. Since services affect devices, they are included too. These three components form the three layered model.

The chapter then discusses how to control access to these components. It is through controlling access that undesirable interactions can be avoided. The technique the approach uses to control access will also be explained, as well as how services can be ranked, so safety services can take priority over and override less critical services.

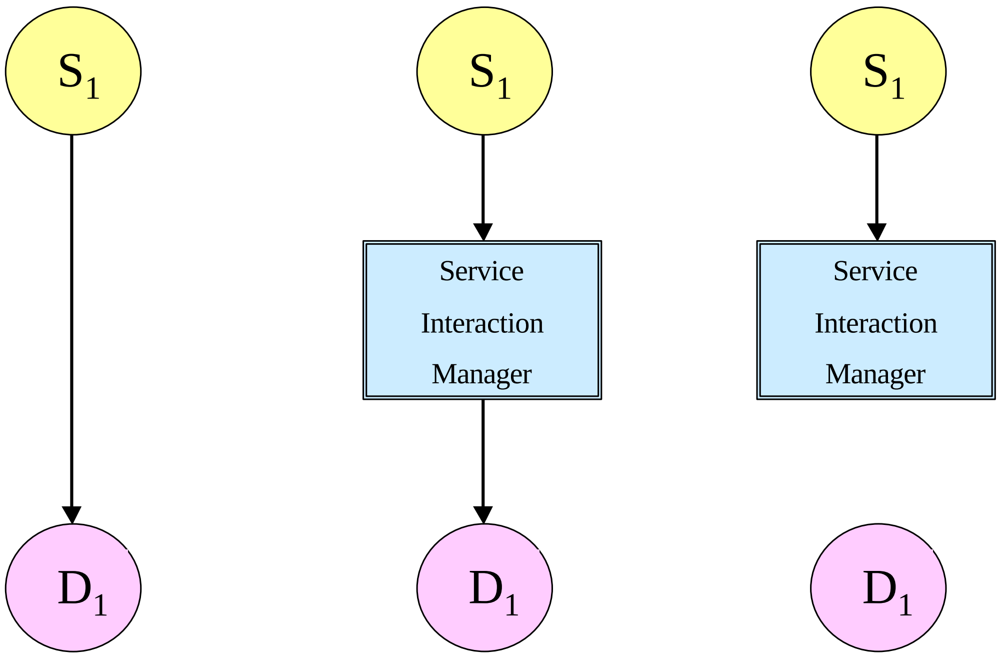
The chapter will also present a Feature Interaction Manager (simply 'manager') and explain how it builds a picture of all services and devices in the home. The manager also keeps its view of the home consistent by listening for devices joining and leaving the network. Further, it will keep track of the device's current state and which service is using the device.

Finally, the chapter will finish with a complete worked example of how the manager works, controlling access to the environment layer to avoid interactions.

5.2 The approach

Using the examples from Chapter 4, some interactions can be detected at the device level – two services try to use one device. However, some interactions occur which seem unconnected. Take the example between the security service and the climate control service (section 4.3.3). Assume the security service is armed and monitoring the home for movement. Also, the climate control service wants to circulate air in the home by turning on the fan. When the fan is turned on, it creates movement which triggers the alarm. The conflict does not happen at the device level, it happens elsewhere – in the environment. The alarm service is monitoring room movement and the fan is switched on, which affects movement. For this reason, the environment layer is included and is central to this approach.

Since some interactions occur in the environment layer, access to this layer must be controlled. For this reason, the approach makes use of concepts from the operating systems domain where controlling access to resources (e.g. files) is achieved through locking. Drawing inspiration from this domain and adapting the locking technique to control access to devices and the environment, an online approach has been developed.



(a) Direct to device (b) Via SIM – success (c) Via SIM – reject

Figure 5.1: Service issues request to device

It is assumed that there will be a residential gateway in the home where services are managed and executed [96] (discussed in section 2.2). The services which run on the gateway will send commands directly to the device (Figure 5.1(a)). Since these messages are sent at runtime, a *live* manager is required. This manager must intercept messages which are sent from the service to the device and determine whether a particular command will cause an interaction. The manager must decide what will happen if the device executes an instruction. There are three possible outcomes: no interaction, positive interaction or a negative interaction.

It is important for the manager to be able to distinguish between positive and negative interactions. Positive interactions are where two or more services or devices can work together to achieve a common goal. In contrast, a negative interaction is where the outcome is undesirable or unexpected. The approach presented here is able to distinguish between the two types.

After analysing an instruction sent to the device, if the manager decides the action will cause either no interaction or a positive interaction, the message will be forwarded to the device (Figure 5.1(b)). However, if the manager detects a negative interaction, the message will not be allowed to proceed to the device (Figure 5.1(c)). Instead, the manager adds an entry to its log and discards the message.

As mentioned, the environment is central to this approach; however, the services and devices must be included. The flow of information means a service will affect a device, and through the action of the device, the device will affect the environment. A 3-layered model shows this flow.

5.2.1 The 3-layered model

The approach developed requires three layers, as there are three main parts to the approach: the home services, the devices and the environment (Figure 5.2). The top layer is the service layer which contains the services that automates the home. These will include climate control, entertainment or home security services (as discussed in chapter 4). These services may use one or a combination of home appliances (devices) which are located in the second layer. Devices may include a heater, a television or perhaps a thermometer. The protocols which these devices use will vary as no one protocol will be used for home networking. The approach developed has been designed in such a way that the underlying protocol is not relevant. This makes the approach extremely flexible as it does not have to change as new protocols are developed.

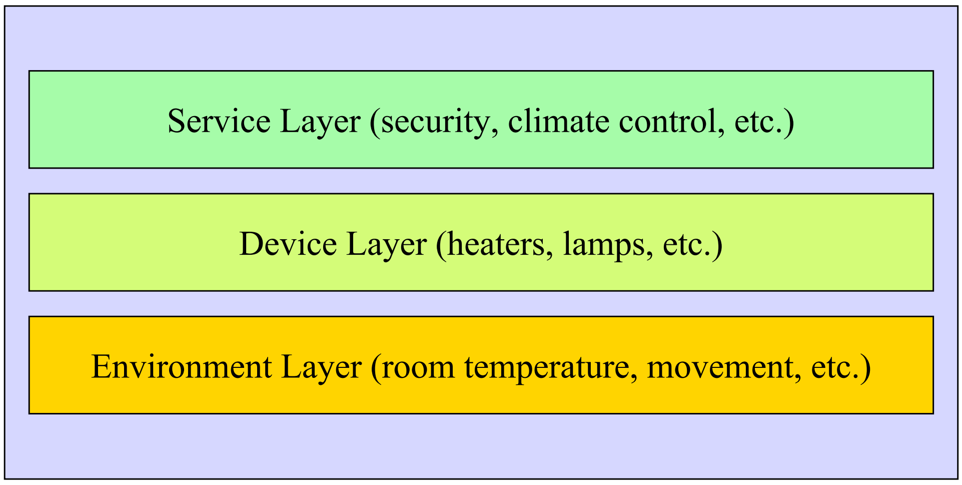

Service Layer (security, climate control, etc.)
Device Layer (heaters, lamps, etc.)
Environment Layer (room temperature, movement, etc.)

Figure 5.2: Three Layered Model

In the device layer, two types of device have been identified: *input devices* and *output devices*. An input device will only monitor an aspect of the environment (e.g. room temperature). An output device, on the other hand, will alter the environment in some way. For example, a heater is an output device as it wants to control the room temperature by increasing it. In this example, the heater is explicitly increasing the temperature. A thermometer only reads the temperature and as it is an input device, can only return what it reads. A device may be both an input device and output device, for example a lamp may be a light, but also have a light sensor attached. In this instance, it would be seen as two devices: a lamp and a light sensor. UPnP splits devices in this way, physically they are one, but logically seen as two.

Finally, the bottom layer is the environmental layer containing environmental variables. These variables are a representation of a room's environment. Examples of environment variables include: room movement, room temperature, room lighting levels, humidity, smoke, carbon-monoxide levels and pollen levels. An example could be an active air fan, which affects the room movement variable. Similarly, a lamp would affect the room light variable. The relationship between variables is interesting. However, for the purpose of this work, the variables are adequate on their own [1].

As previously stated, the reason for including the environment in this model is to show conflicts between devices which only occur through the environment. For example, it is not obvious that blinds and a home alarm service may be linked – but as blinds open, or close, they cause movement. The alarm service monitors room movement. Thus, both are linked through room movement.

Similarly, a heater device would affect room temperature, so if a heater is active room temperature would increase. An air conditioner, when active, would decrease room temperature, therefore it would be undesirable to have it and a heater active at the same time as they have conflicting goals. By using the environmental variables, links between devices become clear. Moreover, by controlling access to the variables it becomes possible to avoid negative interactions.

5.2.2 Controlling access to components

Within the device and environment layer, controlling access to devices and environmental variables is achieved through locking. Controlling access is central to how this approach works. The use of locks has been inspired from the operating systems domain. Access to environmental variables and devices can be likened to files. When an operating system process opens a file, it can generally be opened with one of three properties: append, read-only and write [97]. The idea of *reading* and *writing* can be used when a device requires access to an environmental variable or a service wishes to access a device.

When the issue of locking is introduced, deadlock becomes a concern. For deadlock to occur, there are four necessary conditions [98]:

- *Mutual exclusion* – one resource held locked, not shared.
- *Hold and wait* – where one process is holding at least one resource, while waiting for an answer.
- *No preemption* – a resource can only be released, voluntarily, by the process holding it.
- *Circular wait* – there must exist a set, e.g. {Process A, Process B, Process C} where Process A waits for Process B, Process B waits for Process C, and Process C waits for Process A.

The technique here violates two of these necessary conditions – point 2 and point 4. A service cannot wait. Also the manager (SIM) can unlock a device, if required.

Simply locking a variable is too crude for this approach. It is not adequate as it does not allow two devices with the same goal to work together. For example, when a heater is active the room temperature variable would be locked. Since the variable is locked, another heater would not be able to heat the room. Although there is an interaction here, it is a positive interaction and should be allowed. However, it would not be correct to share the temperature variable between a heater and an air-conditioner as the two devices have conflicting goals.

Concepts were used from the *Biased Protocol* [98], where locks may be either *Shared Locks* or *Exclusive Locks*. This locking protocol is useful for this work as it allows two devices to work together to achieve a common goal. Using these two locking types, a refined locking technique for this approach has been developed.

Since input devices do not affect their environment in any way, they do not need to be locked. On the other hand, because output devices do have an impact on their environment, controlling access is necessary.

After consideration, a new locking technique was developed.

If a service wishes to lock a device, or a device wants to lock a variable, they must be locked with one of four options:

- *NS* : Not Shared. The variable or device is locked and may not be altered by any other device or service. This lock is similar to the exclusive locks in the biased protocols.
- *S+* : Shared, but increase only. The variable is shared on the condition that anyone wishing to use the variable must increase it. Therefore, two devices may lock a variable with S+ if they both increase value. This allows two heaters to operate.

[1]This issue will be discussed further in Chapter 8.

- *S–* : Shared, but decrease only. Like the previous setting, the variable is shared on the condition that anyone wishing to alter the variable must decrease it.
- *S±* : Shared. The variable or device is shared and it is unknown whether the variable will be decreased or increased in value. This can also be used for binary values, e.g. whether there is movement or not. This lock is not compatible with S– or S+ because S± could go either way (increase or decrease). Therefore, S+ and S± could allow one device to increase while the other decreased. Also, S+ and S± could result in both increasing. However, this can not be guaranteed, therefore S± is not compatible with S+ or S–. This lock type can be likened to the shared lock type from the biased protocol.

By using these four locks, devices are able to cooperate and work to a common goal, whereas devices with conflicting goals would be avoided. Table 5.1 summarises the list of locks above and shows the combination of locks which are allowed (•).

	NS	S+	S–	S±
NS				
S+		•		
S–			•	
S±				•

Table 5.1: Locking – allowed pairs

Many services or devices may lock a device or variable with matching *S+*, *S–* or *S±*. However, only one service or one device has access when a device or variable is set with *NS*. Once a lock has been set, it is sometimes not clear when the task is complete and the lock can be lifted. In other domains, telephony, for example, this is clearer.

In telephony a session is clear, the session starts when the receiver is lifted off-hook, and finished when the handset is placed on-hook. In the home domain, the notion of a session is less clear [99]. This approach assumes that a session begins when a service starts using a device, e.g. opening a window or turning a heater on, and finishes when the service closes or switches the device off. Therefore, when a lock is placed on a device, the lock is valid until the service turns the device off. Although this seems a rather simplistic approach to the problem, this is how operating systems lock files. When a file is in use it is locked, when it is not, it is generally unlocked. The problem is that there *is* no simple way of determining when a session starts and ends in the home.

5.2.3 Locality

As well as knowing the duration of a session, it is also important to include the locality of the actions of a device. This is to include devices which, by operating, may affect the whole home, or just one room. For example, a heater may only affect the temperature in one room, whereas a security alarm may monitor one room or the whole home. The approach must also be able to handle this, which it successfully does by creating a hierarchy of rooms within the house. It is important to note that each room is treated independent of others, in other words, one room cannot affect another in this model.

When a device is active, locks are placed on the appropriate variables in whichever room is required (Figure 5.4 later).

5.2.4 Service priorities

As an example, suppose the home is being burgled and the VCR is in use by the entertainment service to record the owner's favourite show. Since the VCR is in use, the home security service is unable to access the device to record the intruder. As presented so far, access to devices and variables operate on a first come first served basis which, as will be shown, is not adequate.

The entertainment service sets the VCR device with an *NS* lock, which does not allow any other service access to the device. A mechanism of overriding this is required. A mechanism which allows important services to override convenience services is required. Thus, service priorities have been introduced to the approach. This allows safety services to override convenience services.

Priorities are widely used for feature interaction resolution (Chapter 3). Priorities have been used here by giving each service in the gateway a priority number. A priority value may change as new services are added to the home, or a user's preferences change.

The priorities of services will range from 1 to n, where n is the total number of services in the gateway. Priority 1 is the lowest and n is the highest.

There are three other service priorities: *-1* (meaning no priority) for services that have not been assigned a priority, *0* which is used by the service interaction bundles only as the highest priority. It can be used by the manager when a device's state changes (e.g. a lamp being switched on). Having *0* as the highest priority means that these updates will definitely be included in the manager's view of the home. It is only right that these updates are made as they reflect the actual state of a device.

5.2.5 The remote device database

If a heater is turned on, it will produce heat, which in turn affects its environment and increases the room temperature. For this approach to work, the manager needs to understand these details and know *what* devices do, and how the device actions affect the surrounding environment.

Therefore some sort of database which holds device details is required. The database of devices is likely to be constantly growing as new devices come onto the market. Where this database is stored does pose a problem. If the database were to be stored in the home, not only would it be very large (since it has details of all devices), but managing it and keeping it up to date may cause problems. In contrast, if the device were to be hosted remotely, all homes would be able to share the same data. However, the disadvantage of this is that the home would need a constant connection to the internet, which is a fair assumption. If newer smart devices were able to store information about themselves on the device, this would be the better solution. However it cannot be assumed that devices would come with this data and, politically, adding such data to all new devices may not be straightforward.

For the purpose of the thesis, a remote database is used. The modular way in which the manager has been developed means that it is not important where this data comes from. As long as the information can be made available to the service interaction manager.

After receiving the device description from the remote service, a local copy could be cached, however, this is purely a performance issue.

Describing devices

If the manager is to operate successfully, it requires some basic information about a device and how its actions affect the environment. The approach needs this information; using a remote database with device details is a convenient way to achieve this. It also has potential to be used by other services, if they require information about devices. This system could potentially help facilitate adaptive services.

Although this may be seen as a constraint on the system, it is a reliable, and scalable solution to the problem. A self learning system may be implemented where the manager sends commands to devices and records how the environment has been affected by the device's action. However, this would require that the manager knew all APIs for the device. Further, the manager would require a training period where it conducts tests under strict conditions. This is to ensure the manager understands all the variables a device affects. If the tests are not done carefully, other factors (external to the device) may affect the environment, which causes the manager to record incorrect data.

There is a danger that if these tests are not conducted properly that the manager would hold false information about the device. Having incorrect information on each device would have a serious impact on the effectiveness of this approach. Therefore, since these devices will be deployed in homes, the simplest and most reliable solution is to use a remote database. This means there is no warm-up time for the approach and a user does not have to worry about training the manager for new devices.

The device type is essential. This describes the type of device, such as heater, air-fan or television. When querying the remote site, the manager should supply a device type. The type supplied by the caller is matched with the description within the database. The remote device database does nothing more than return device specifications in an XML format, given a device type. Example XML details are shown in Figure 5.3. The example XML shows the Lamp is an output device, also the default usage for a service is *NS*. The XML also shows the environmental variables this device uses, `RoomLight`. The default value for this variable is *S+* as room light will be increased, and the locality is set as 0 (Figure 5.3), which means it will only affect the current room.

```
<Device>
    <DeviceType>Lamp</DeviceType>
    <DeviceIO>Output</DeviceIO>
    <DefaultDeviceUsage>NS</DefaultDeviceUsage>
    <Action>
        <Name arg="on">deviceOn</Name>
        <SuggestedDeviceUsage use="NS" />
        <EnvironmentalVariable name="RoomLight"
              defaultValue="S+" duration="3" locality="0" />
    </Action>
    <Action>
        <Name arg="off">deviceOff</Name>
        <SuggestedDeviceUsage use="" />
        <EnvironmentalVariable name="RoomLight"
               defaultValue="" duration="0" locality="0" />
    </Action>
</Device>
```

Figure 5.3: Example lamp description

Various other attributes are included in the device description, such as the default lock values for the device. This makes it possible for this approach to work without having any input from the service – fully operational on its own.

As well as knowing what type the device is, it is important to understand the actions of a device and how each action affects the environment. The alarm control panel device is a good example as it may have different functions. An alarm control panel device can be used with a security service running on the gateway. The alarm control panel may have two functions. One function is fully armed where any movement in the home is interpreted as an intruder. The second function may be used as a notifier. When the front door is opened, a 'beep' is sounded to make the owner aware the front door has been opened. These two functions will want to control the environment in different ways. One will detect movement, whereas the other detects but allows movement. Thus, depending on the way which a device is used, different locks for the variables will be required.

The fully armed function would lock the movement variable with *NS* as it does not want to share it. The notify function would share movement with $S\pm$ as it does not need to have the variable fully locked. However, it does not want another service fully locking the variable.

A device's environmental variables and locking information are crucial for the manager to operate. Another important piece of information is locality.

Consider the alarm control panel device as an example. Depending on the action, the alarm panel may be set to monitor the whole home or only one room. Again, depending on the action, the variables a device affects (and how the variables are affected), will be different.

5.3 The service interaction manager

The service interaction manager (SIM) is central in this approach as it implements the issues covered in section 5.2. As discussed previously, the manager would be a service within the service gateway within the home. This is because all devices will be registered within the gateway, and all services will be managed and executed from the same gateway.

If the SIM is not in use, when a service issues a command to a device, the message is sent directly to the device (Figure 5.1(a)). When the manager is active, messages are intercepted and authorised.

There are two possible outcomes: either forward the message to the device (Figure 5.1(b)) or reject the message (Figure 5.1(c)).

As stated in earlier sections, to avoid interactions the manager restricts access to devices and environmental variables. To make the decision the manager analyses the state of required devices and associated environmental variables. For this to work, the manager must keep an internal image of the state of all devices (Figure 5.4).

The manager generates the view of the home, which is of a hierarchical form. This view is logical rather than physical as a device may be in one room but may control a device in another – for example a Hi-Fi with speakers in two rooms. The manager generates the view by searching the gateway for all device objects registered. Once it has a list of all devices, it consults the remote device database (section 5.2.5) and obtains all associated environmental variables required for this device (this process is explained fully in section 5.3.1).

If a device joins or leaves the network this change will be registered in the home gateway, in turn notifying the manager. If a device changes room, unless this change is captured by the gateway, the manager's view will not change. This is not unreasonable, as services in the framework may search the framework's registry for a device in a specific room.

Figure 5.4 shows the hierarchy with the *home* at the top. Within the home there are *rooms* and within rooms there are *devices* and the *environment* (variables).

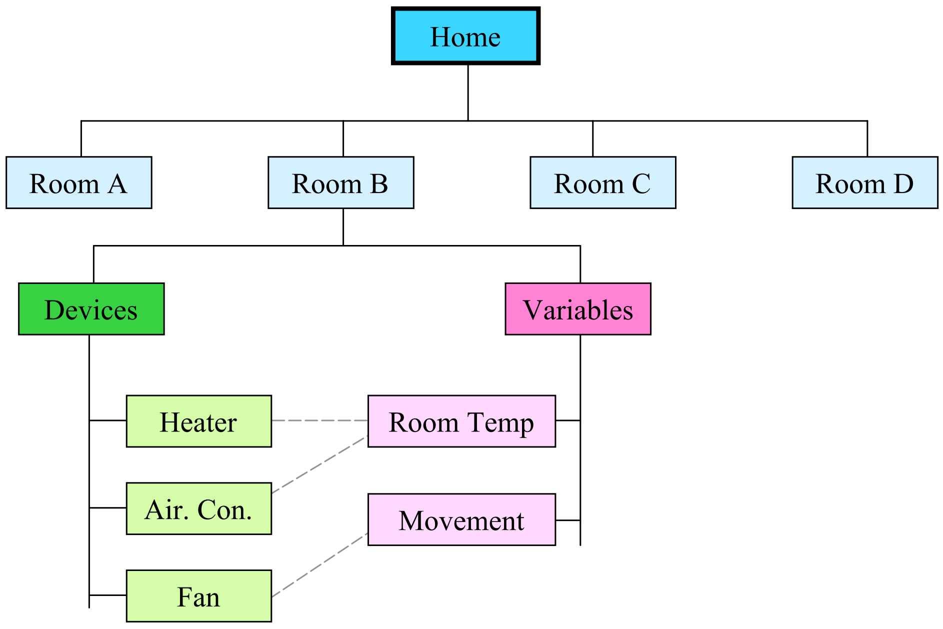


Figure 5.4: Internal representation

5.3.1 Keeping the manager up to date and consistent

Keeping the manager's view of the home correct is not straightforward. In a home network, devices will join and leave the network in an ad hoc fashion. The joining and leaving of devices from the network must be monitored, and these changes must be reflected in the manager's internal view (Figure 5.4). The manager also needs to record the devices a service is using. The manager's internal view is kept up-to-date and consistent in three ways:

- Monitor the gateway for devices joining and leaving the network.
- Note service commands which are authorised by the manager.
- Observe direct device control by the user.

These three ways are discussed further in sections 5.3.1 to section 5.3.1.

Monitor the gateway for devices joining and leaving the network

When a device is added to the gateway, the manager determines the type of device which has been added. It then consults the remote database that returns, among other properties, the variables which a device will affect (Section 5.2.5). After obtaining these properties, the manager tries to determine which room the device is located in. This information about the device is then added to the manager's internal view (Figure 5.4). If the manager can not determine the type of device or location, the user must supply this information. Informing the system of the location of a device is the most a user would be expected to undertake.[2] However, some protocols, such as X.10, provide room location in the address, assuming the setup is correct.

As well as listening for new devices being added to the framework, the manager must also remove devices which have been unregistered, or removed from the framework. Protocols such as UPnP advertise when they leave the network, whereas X.10 devices do not. It would be up to the user to remove the devices from the framework. This is simply a limitation of the X.10 protocol. When devices are removed, any locks a device has on an environmental variable are removed.

Service commands which are authorised by the manager

When the service issues a command to the device, it is intercepted by the SIM. If authorised by the SIM, the command is forwarded on to the device (Figure 5.5). After the command has been authorised by the SIM, the manager must record the new device state. It does this by updating to its internal view of the home, thus keeping itself up to date and consistent.

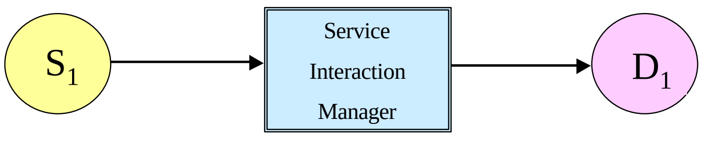


Figure 5.5: Service to device via SIM

Direct device control by the user

If a device is controlled directly by the user, perhaps the user has pressed *play* or *stop* buttons on the VCR, for example, this new state must be recorded in the model. Depending on the protocol, some may broadcast a change in state. UPnP, for example, does this. If the device state is changed, a message is sent to all subscribed parties. If a device changes to a state which causes an interaction, the state still has to be recorded as this *is* the current state of the device, Figure 5.6. If this change is not recorded, the manager's view is not a true reflection of what is happening in the home.

5.4 Operation of the approach

5.4.1 Model

A new model has been developed to show how interactions are avoided using this approach. The model is based on the three-layered model (section 5.2.1), which shows the home services as well as devices and their relationship with the environmental variables. With this in mind, a model has been developed and is shown in Figure 5.7. All services, devices and environmental variables, from all rooms in the home,

[2]Location Aware Computing is beyond the scope of this thesis, however it is an active area of research and more information can be found in [100, 101]

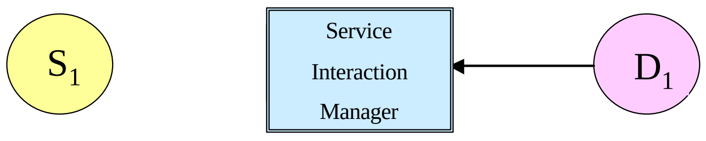


Figure 5.6: Device to SIM

would normally be shown here. However, for clarity, only one service, two devices (an input device and output device) and one environmental variable has been included.

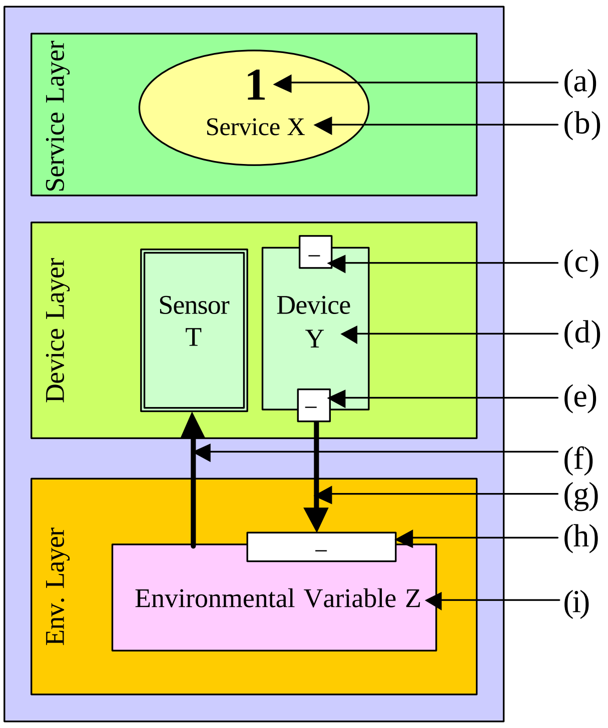


Figure 5.7: 3-layered model populated

The first layer in the three-layered model is the service layer, and one service is shown in Figure 5.7. The service name is shown (Figure 5.7(b)), along with the priority of the service (Figure 5.7(a)). In this instance, the service has a priority of '1'.

The middle layer of the three-layered model is the device layer. All devices registered in the framework are represented here. Figure 5.7 shows two devices. 'Sensor T' is surrounded by a double rectangle whereas 'Device Y' has a single rectangle. The double rectangle represents an input device. An input device does not affect its environment, it only monitors it. In contrast, the single rectangle represents an output device. Output devices will affect their environment. Only the primary effect on the environment variable is captured here. If a device has a side effect, this may impact other variables. To keep the model simple, side effects have not been captured here.[3]

The fact this is an output device is shown by the direction of the arrows Figure 5.7(f) and (g). Figure 5.7(f) shows that it is the environment (variable) that will have an effect on the sensor device. On the other hand, the arrow in Figure 5.7(g) shows that the device (output device) will affect the environment (variable).

[3]Implications of not including side effects are discussed in depth in Chapter 8.

Central to this approach is the concept of controlling access to devices and the environment to avoid interactions. Therefore, the model has to show the access control mechanisms – the locks.

Figure 5.7(c) shows the lock for controlling access to the device itself. This lock is set by the service. Generally, a service will use *NS*, as it is unlikely to want another service using the device while it is in use. If a service does not understand how to set the lock for the device, a default lock from the device description database (section 5.2.5), which is normally *NS*, is used.

It is only output devices which can be locked by services; input devices can not be locked, they simply report details back to services.

Once access to the device has been gained, the manager must ensure the device is able to gain access to all required environmental variables. It is only the variables which the device will affect by carrying out this particular action that are consulted. The variable a device affects when it carries out specific actions are obtained from the remote device database. Figure 5.7(e) shows the proposed lock for the environmental variable. The arrow, Figure 5.7(g), points to the variable where the lock is to be set.

Figure 5.7(h) shows the current lock of the environmental variable. The reason for showing locks on both the bottom of a device and on the environmental variable is that the environmental variable may already be locked by another device. Figure 5.7(i) is the name of the environmental variable.

Using this approach, the interaction discussed in section 4.3.4, is used to show how the approach operates.

5.4.2 A worked example: Interaction between climate control and security

Under certain circumstances a negative interaction can occur between the home security service and the climate control service (section 4.3.3).

As previously discussed, if the alarm is armed, it would not be appropriate for the climate control to open windows in the home. Not only would opening the windows cause movement, triggering the alarm, but the two services have conflicting goals here. The goal of the security service is to keep the home secure, while the goal of the climate control is to cool the home by opening the window which makes the home insecure. These goals conflict which causes a negative interaction.

The approach described in this chapter can be used to avoid this interaction.

Setting the scene

To model this interaction, two services are required – security service and climate control service. For the services to operate they require a number of devices. For the sake of clarity and simplicity, a home with only one room and minimal devices is assumed.

The climate control service requires a temperature sensor (input device) to get the room temperature. In this home, there is also an external thermometer, so the outside temperature can be obtained (for clarity, the external thermometer has not been included in any of the Figures, Figure 5.8 – Figure 5.12). To control the temperature the service has three output devices available for use: a heater, an air-conditioner and a window. Each of the output devices changes its environment in some way. The heater will heat the room, and therefore increase room temperature. The air-conditioner will cool the room, reducing room temperature. The window, when open, will change room temperature either up or down, depending on the outside temperature. When opening and closing, the window will create movement within the room. Therefore, among the three devices, two environmental variables are required: temperature and movement.

The security service requires an input device, a motion sensor to detect movement within the room. The service also requires two output devices: an alarm control panel, which is used to set or disable the alarm, and alarm bell. On the surface, the alarm control panel may seem to be an input device as a user uses it to arm the alarm. However, although there is no direct output to the environment, this device does control a variable in the environment, movement. Only output devices can lock variables; input devices cannot control the environment, they simply monitor (read) the variable. Therefore, the alarm control panel appears as an output device.

The alarm panel wants to control movement within the room, it does not want any movement created when the alarm is active. When the alarm is triggered, the bell device is used to draw attention by making a noise. These two devices each require an environmental variable: the alarm control panel requires movement and the bell requires the sound variable.

The static setup

The static relationships between devices and the environment are defined in the device description database (Figure 5.3, line 8 & 9).

Assuming these were the only services and devices in the home, the manager would generate an internal image shown in Figure 5.8. This representation is stored in memory, so if the gateway is restarted, the internal image would simply be rebuilt.

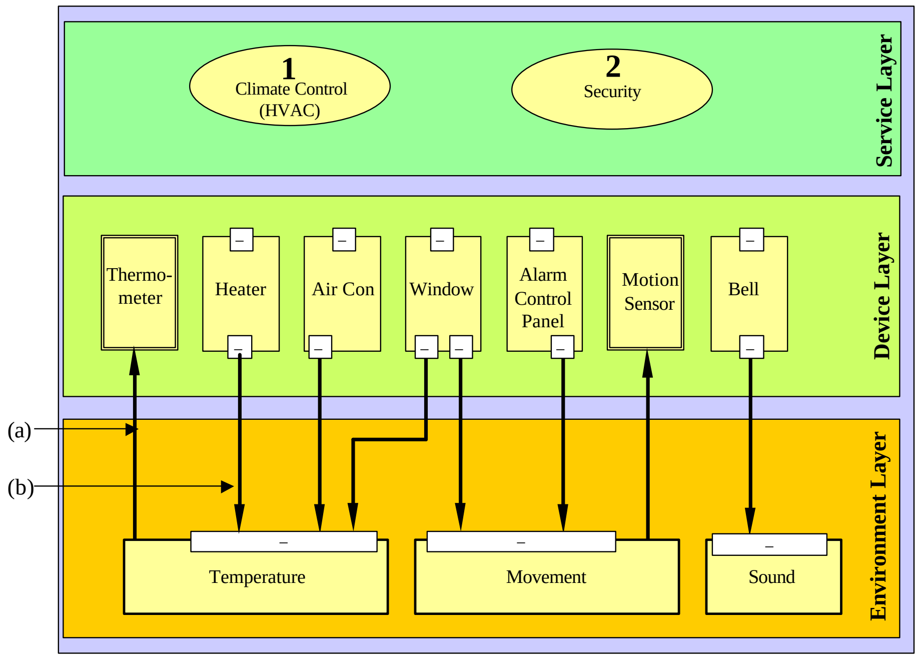


Figure 5.8: Static model of security and climate control service setup

In this scenario there are two services, both of which are in the service layer. Assume the service priorities have been set by the user (or a service provider on their behalf). The climate control has been set with the lowest priority, '1', and the security service with '2', which is the highest in this scenario. Therefore, the security has overall control, if it needs to gain access to any device which the climate control is using.

The device layer contains the seven devices (two input and five output) and the environment layer contains three environmental variables. The black arrows between the devices and variables show the static links between a device and its environment. The direction of the arrows also show the flow of information. Figure 5.8(a) shows the temperature variable will affect the thermometer, and Figure 5.8(b) shows that the heater will affect room temperature.

This static model is generated automatically by the manager at runtime. When each of the seven devices is added to the gateway, the manager would automatically register each in its own internal view. The manager would then consult the remote device database (section 5.2.5) to obtain device details such as default values and associated environmental variables.

Arming the security service

Assume the climate control service is active but only monitoring the room temperature (not controlling any devices) and the security service is off.

If the owner leaves the home, they use their mobile phone to communicate with the security service which turns the alarm control panel to active.

The security service first needs to get access to the alarm device. Since this device is not in use, it is able to gain access and lock it with *NS* as it does not want anyone else using the device (Figure 5.9(a)). Next, using the default values from the device description database, the manager knows that for the 'arm' command, the movement variable should be locked using *NS*. Figure 5.9(b) shows the lock the device wants to place on the environmental variable, *NS*. Since the environmental variable is not locked, this value is set in the variable (Figure 5.9(c)).

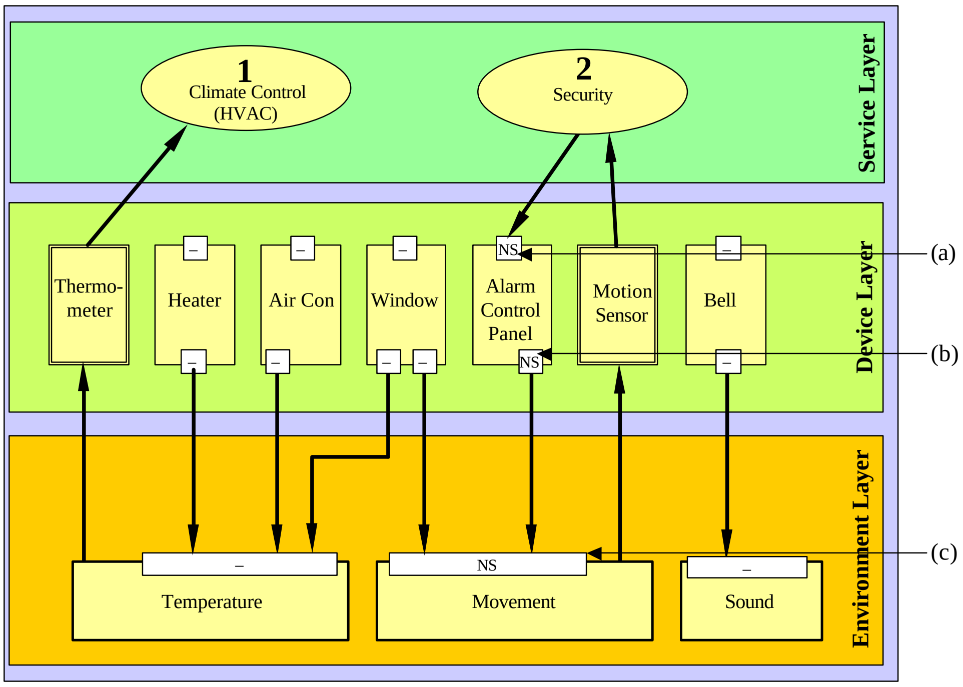


Figure 5.9: Security service armed

The security service is now fully armed and monitoring the home. The climate control service is also active, but it does not require any devices, other than the thermometer notifying the service of a change in room temperature.

Avoiding the interaction

The security service in the home is armed. Assume that the temperature within the home starts to rise and the climate control service (knowing that it is cooler outside) needs to open a window to allow the cool air in. If the service interaction manager were not in place, the climate control service would open the window making the home insecure and also triggering the alarm.

If the manager were active and the climate control were to issue a command to open a window, the command would have to be authorised by the manager. Assume the climate control service issues a command to open the window. First, the climate control service must be able to access the window. Since the window is unused, the service is granted access and sets the device lock with *NS*, Figure 5.10(b). The dashed lines (Figure 5.10(a) and (e)) are to show temporary links. Until authorisation has been granted for both device and all related variables, they remain dashed.

Access to the window device has been granted. The manager can now try and set the device's proposed locks for the environmental variables. Since opening the window will impact two environmental variables, two proposed lock boxes are shown (Figure 5.10(c) and (d)). Since it has been determined that the temperature outside is colder than inside, the device would like to lock the temperature variable with $S-$ (Figure 5.10(c)). Also, since opening the window will cause movement, the device would like to set movement to $S\pm$ (Figure 5.10(d)). When the manager tries to place a lock for the device on the environmental variables, the manager can set the temperature variable with $S-$ (Figure 5.10(g)) as this is free. It can not set $S\pm$ to the movement variable. This is because movement is already set with *NS*

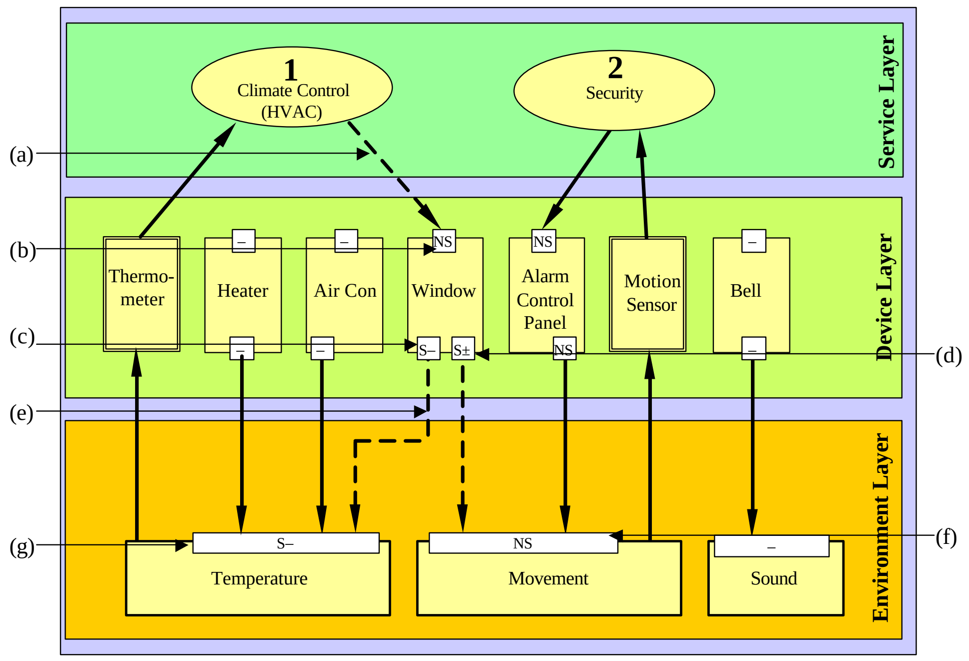


Figure 5.10: Avoiding Interaction between Climate and Security service

(Figure 5.10(f)), and *NS* and *S*± are not compatible (Table 5.1). The lock on the movement variable can not be overwritten as it has been set by a service with a higher priority.

Since the lock can not be placed on the movement variable, the window device is unable to open and the interaction has been successfully avoided.

If the climate control service truly were a *smart* service, it would search for an alternative way of cooling the room. It should realise an air-conditioner is available. Since the air-conditioner is available and the temperature variable is also available, the climate control service would be able to gain access and control the air-conditioner device and turn the device on, Figure 5.12. Both services are now active and work in harmony.

Avoiding the interaction under different circumstances

The example described above assumes the home security service is set first. Assume the home security service is inactive and the climate control service is currently active and has the window open. Further, assume that the security service has extra functionality that makes it check that all the windows are closed.

This is one example of an instance where priorities must be used. Before arming itself, the security service checks that all windows within the home are closed. Whilst checking, it notices that one window is open, and has been opened by the climate control service (Figure 5.11).

Figure 5.11 shows the two services. Climate control is active and has opened the window. To make the home secure, the security service is trying to gain access to the window device to close it. However, when the security service tries to gain access to the window device it finds it is locked with NS. Normally access would be denied, but the security service checks the priorities and finds that the window has been locked by the climate control service which has a priority of 1. The security has a priority of 2, which is higher than the climate control priority, so the security can gain access and control the window.

Therefore, the security service is able to override the lock and gain access. The security service can close the window. When the window is closed (or turned off), the locks it has on the temperature and movement variables are released. When the security service tries to arm the alarm it succeeds and the alarm is armed. Thus, the internal image would look like Figure 5.9.

If the climate control service does try and open the window again, since the environmental variables are locked, the request will be denied as security has a higher priority, just like Figure 5.10. There is

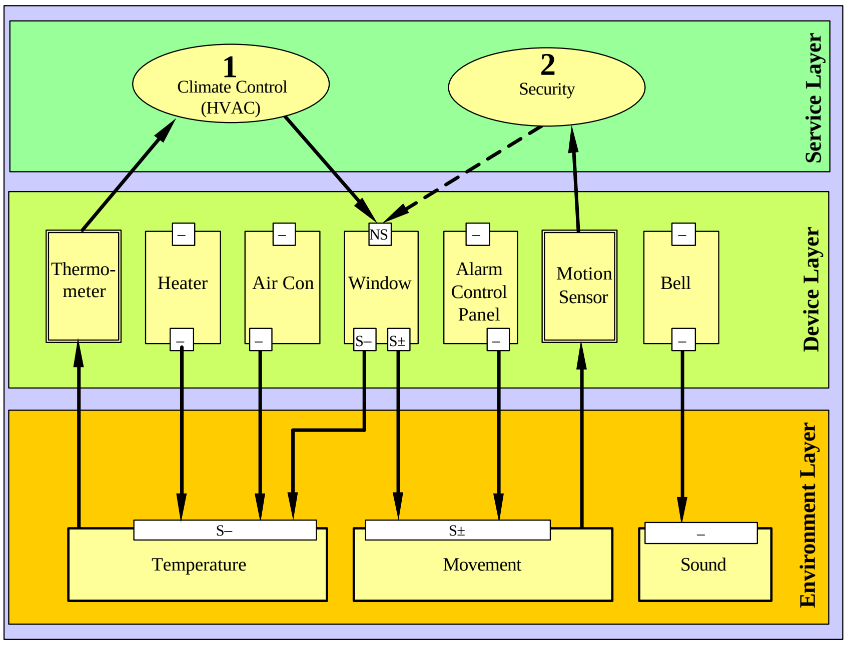


Figure 5.11: Avoiding interaction when climate control is active first

an issue that the service and device may become out of sync. If a protocol, such as UPnP is used, once a command is issued to a device, a service would normally expect a `200/OK` message to be returned. If it does not receive this, the message will be resent. Since the manager is blocking the messages, the resent message will be blocked, like the first. Devices like X.10 offer no acknowledgement of commands. Therefore there is a possibility a service thinks it has turned a device on when it has not. An argument could be made that the manager notifies the calling service with the reason why the window is unable to open. Currently, the approach does not support this as it would require services having knowledge of the service interaction manager, this is an unwise assumption, though technically it is relatively straightforward to add.

This is not necessarily a problem. Services where it is vital a command is carried out by a device should expect a response from the device. In a home security alarm, for example, an alarm will not set properly if some doors or windows are left open. Services where it is important that a command is received and processed by a device, should require an acknowledgement from the device. Therefore, if the manager blocks all commands to a device, the service should not assume it was successful, unless it receives a response. However, this is an implementation issue for the service vendor.

5.5 Summary

This chapter has described how this technique is able to avoid interactions with little, or no, user intervention. An online service interaction manager has been presented. The manager runs on the services gateway where home services and devices reside.

When messages (commands) are sent from a service to devices, the manager will intercept the message, analyse it, and decide whether a negative interaction will occur. The technique draws inspiration from other domains, such as Operating Systems, for locking. By refining locking algorithms, services and devices with a common goal can work together, whereas services and devices with conflicting goals are not allowed.

The manager has an internal view of the home network. The internal view is automatically generated by searching for all devices on the framework. Since the manager consults an external database to get device descriptions, as long as the database is kept up to date, the manager can handle new types of devices.

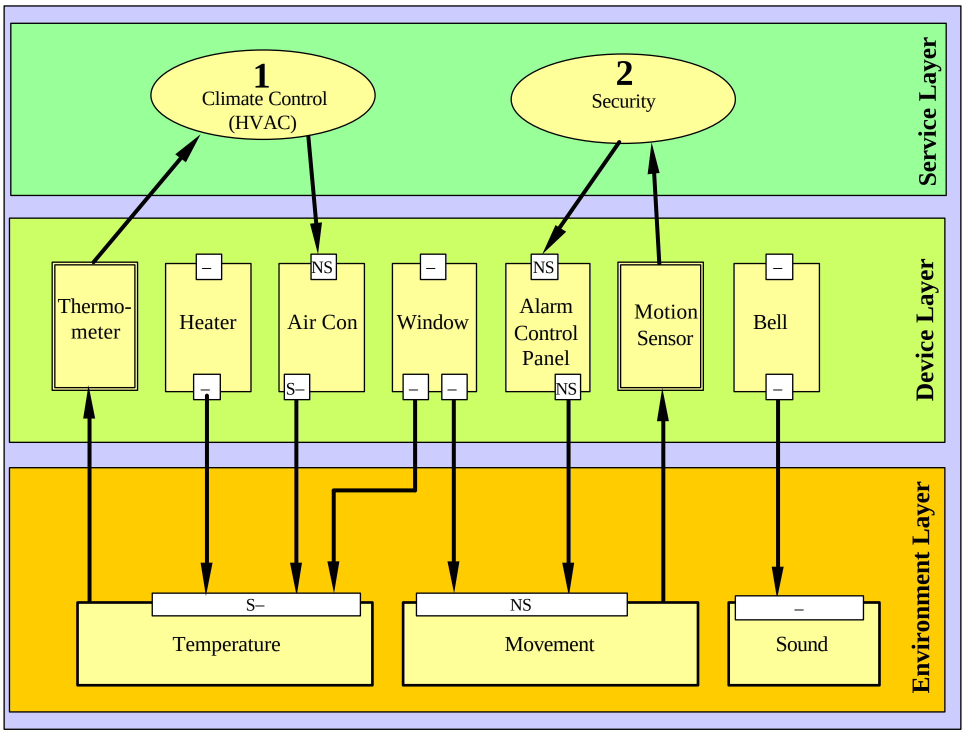


Figure 5.12: Security service and climate control service both active

The internal model is then kept up to date by listening for devices, and services, joining and leaving the residential gateway.

An interaction example between the security service and climate control has been presented. The example shows how the technique is successfully able to avoid negative interactions. Importantly, it shows how negative interactions can be avoided using the environment.

This approach is independent of the service, as the approach described focuses on devices and the environment. However, to avoid some interactions, it is necessary to know the priority of a service. This priority does not come from the service itself, but instead comes from another service which is part of the manager. The example presented earlier shows how priorities were used to avoid the interaction.

An example of how the interaction outlined in section 4.3.4 has been successfully avoided has been presented.

The next chapter will describe, in detail, the test-bed used to test this approach. The subsequent chapter will discuss the experimental results.

Chapter 6

Architecture of the Test-bed

6.1 Introduction

To show service interactions do happen in the home network, and that the approach detailed in the previous chapter does work, an experimental home network (test-bed) was created.

The test-bed has to reflect what a real home network may look like. This includes user services and the kind of devices one would expect to find in such a home. Following a review of the literature [9, 102], a test bed was developed.

There were two phases in the development of the test-bed. The first phase was to develop and test a test-bed which contained devices and home services. It is important that the test-bed is stable and working correctly before including the approach, as this ensures results from the service interaction manager are reliable. This is explained in the next section. The second phase was the design and implementation of the approach. A discussion will follow regarding the testing of the approach.

6.2 Design of the basic test-bed

Here, by the term '*basic test-bed*', we mean a test-bed which includes a selection of devices (UPnP and X.10), the service management framework (OSGi) and the home services (Security, Climate Control, etc.). These three parts form the basic test-bed. Once this is stable and reliable, the approach can be added. This section, however, concentrates on three components: devices, the service management platform and user services.

6.2.1 Devices

As there is a plethora of protocols used in the home, and no single protocol is likely to emerge as the de facto standard. It would be impractical and unnecessary to include devices which use each protocol in the test-bed. Therefore, two protocols have been selected, X.10 and UPnP.

X.10 devices

X.10 was chosen as a protocol because it is currently used in homes. The protocol is popular with home automation enthusiasts because of the availability [2] and cost of the components. Also, typical household devices (e.g. lamps, fans, heaters) can be used. Moreover no additional wiring is involved as it uses the power lines as the transport medium. This makes it a quick, cheap and easy way of automating the home.

For experimentation, the following X.10 modules were used:

- CM11 X.10 gateway module.
- An X.10 lamp module.
- An X.10 appliance module.

- An X.10 motion sensor with receiver.

The X.10 gateway was connected to the computer which was to host the service platform. This can be seen in Figure 6.1. The figure shows the complete test bed, with the X.10 portion shown.
The following general appliances were connected to X.10 modules:

- A lamp was connected to the lamp module.
- A fan was connected to the appliance module.

Since the number of X.10 modules available was limited for the project, a virtual X.10 device was used to enhance the network. The only virtual device used was the home window. The window device was a virtual X.10 device that would open and close. Thus, when an `on` command was sent, it would open. When an `off` command was sent, it would close.

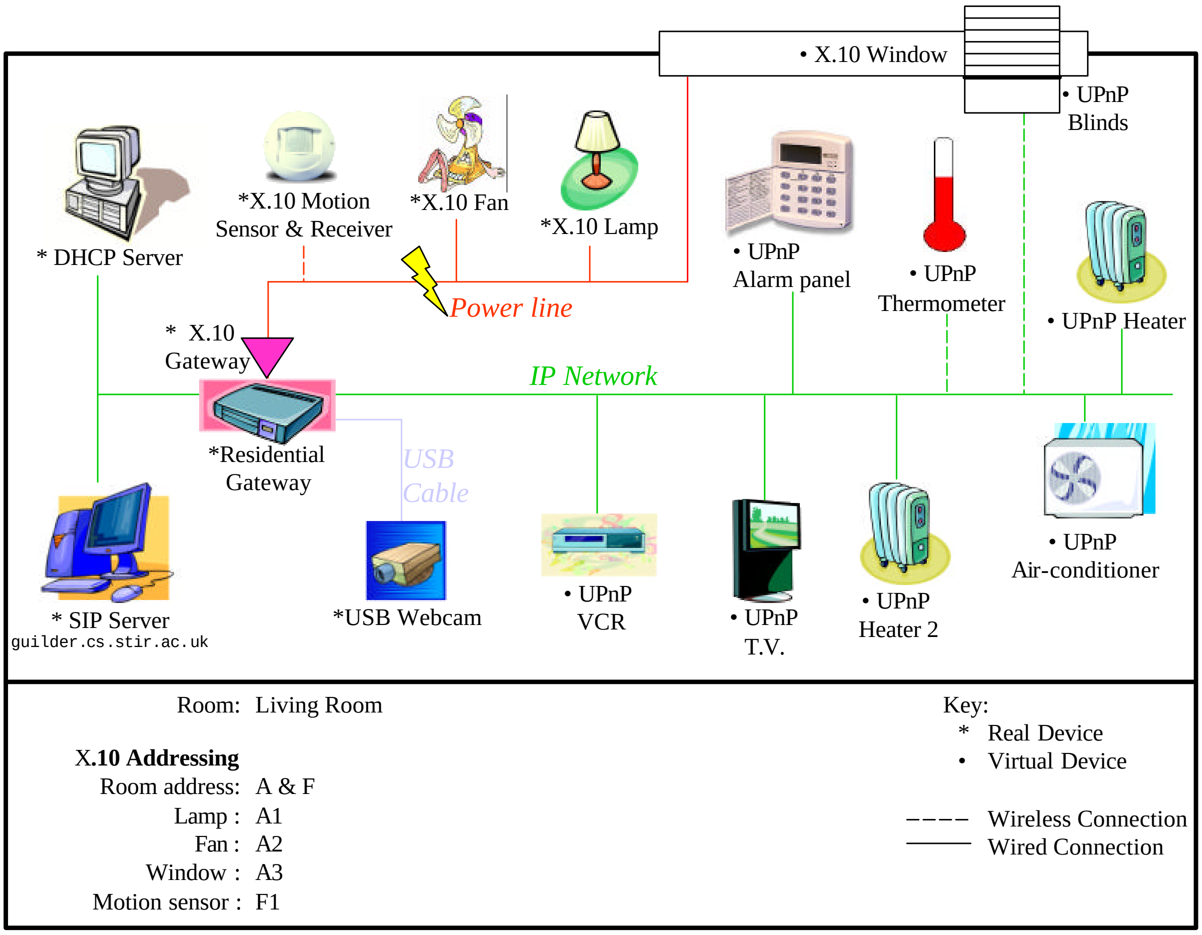


Figure 6.1: The test bed used for experimentation

The X.10 motion sensors are used to detect movement. Upon detecting movement, they send an `on` command to a pre-set X.10 address. Since they send commands to other addresses, it means they do not have an address themselves.
This is not an issue if the home only uses X.10 as the networking protocol. However, it is unlikely a home will only use X.10 because of the limited capabilities of the protocol and the requirements of users. Therefore, some way of notifying other parties that movement has been detected is required. Although crude, the only solution was to change how addressing is interpreted; this is caused by the limitations of the X.10 protocol.
The solution was to reorganise the addressing, in that output devices should have one room address (`A` in Figure 6.1) and a second room address should be used for input devices (sensors) (`F` in Figure 6.1).

This means that when the motion sensor is triggered it sends an `on` command to address `F1` (assuming the sensor was given the address F1 to trigger). Although this would not actually turn any device on, any listening party who knew that room F was set for sensor information in the living room, would know this was to be interpreted as movement in that room.

As the X.10 gateway can listen to X.10 messages on the X.10 network, an interface between the X.10 gateway and the service management framework was created. This allowed the framework, and services on the framework, to be notified of any X.10 messages.

UPnP devices

UPnP is a new home networking protocol and is growing with more OEM companies becoming members. The increased membership and increased attention has meant many new UPnP standards have been defined by the UPnP Forum. At present, only UPnP routers and internet gateways are available to buy off the shelf. Devices required for this test-bed, such as heaters or air conditioners are not available yet. Therefore, virtual devices were used. Using virtual devices offers flexibility for both creating and controlling the device. To create a virtual UPnP device, a UPnP SDK, including UPnP stack, was used.

There is a selection of UPnP SDKs available to third party developers [103]. Many of these however are commercial and the source code for the stack is not available. For this reason, an open source stack was used – CyberLink [104]. There are two implementations of the stack available: Java and C++. As the Java stack was more mature, it was chosen over the C++ implementation which was an early beta version. As well as forming the base for the UPnP devices, it was also used to create the UPnP driver for the service management framework.

Each UPnP device developed had a simple GUI for user input (e.g. set device on or off, or set channel, etc.). These are the kind of controls one would expect on simpler devices. The XML device description and service definitions followed those published by the UPnP Forum. Where definitions for devices were not available, the basic device specification was used [105].

The CyberLink SDK was used to create the following UPnP devices (each with a different GUI, XML service and XML device description):

- *Thermometer* – a device which reads room temperature. When queried it returns the room temperature. Also when the temperature changes, subscribed parties are notified. As this is not a real device, a slider-bar is used to manually change the temperature.
- *Heater* – a simple heater device with two options: on or off. Since this is a virtual device and to mirror what would happen in reality, the heater finds all thermometers in the room and increases their temperature. In reality this is not required as the heater would increase room temperature which the thermometer would detect. This addition did not change the functioning of the device. For experimental purposes, the device service was extended to include location. This is a text field which allows a user to enter the location of the device. This allows other services to query the device to determine location. Again, this extra functionality does not change the operation of the device. This does get around the issue of automatically detecting the location of devices, which is a separate area of research [100].
- *Air-conditioner* – like the heater, this is a simple device which has two options: on or off. Similar to the heater, to simulate reality the device finds thermometers and decreases the temperature.
- *Television* – a device which tunes into a channel and displays the picture sent by a TV station. The TV has a number of functions: on and off, change the channel, and volume up and down. For experimental purposes, a TV station had to be created. This is a Java server which devices can connect to. Images are then sent by the server at regular intervals (1 per second). This was enough to simulate the role of a TV station.
- *VCR* – like the TV device, the VCR records to file any images it was sent. The user can then tune the TV into the VCR and play back the recorded images. The options available in this device are: on and off, play, record and set channel to record. The setting of channel also allows the VCR to record from another source, e.g. a web-camera.
- *Window blinds* – This device simulates a small motor which can open and close the blinds accordingly.

Although the majority of devices in the test bed were either X.10 or UPnP, there were some other important auxiliary devices.

Auxiliary devices

Three support devices were used in the test-bed. These were: a web camera, a SIP Server and a DHCP server.

The DHCP server is used by most IP devices in a network for the allocation of IP addresses. In a home, many home gateways offer a DHCP service (Linksys WRT54G [51] is an example device). In the home, it should not make a difference how IP addresses are allocated; this has simply been included in Figure 6.1 for completeness. If a DHCP server was not on the network, IP based devices could allocate their own IP address using auto-IP.

The SIP protocol is mainly used for VoIP. However, another use of the protocol is instant messaging and presence. For this reason, a SIP server was required for use. The server used was SER [106]. This is a stable server available for use in the Computing Science Department within the University. The SIP server and setup is out of the scope of this project as it is unlikely many home owners would go to the trouble of installing and setting up their own. This is due to the complexity and time required. A more likely situation is that a home owner would be offered a SIP service by their service provider. For example, Microsoft Windows Messenger is based on SIP [107]. This is an instant messaging service that allows users to log in and send messages to *buddies*. The SIP server, shown in Figure 6.1, has been included in the diagram for completeness.

The web camera device is a USB device and is connected directly to the computer which runs the service platform. More sophisticated web cameras will soon be available which are wireless and even UPnP based. This means they can be anywhere in the home and do not have to be directly connected to the computer which controls it. However, the web camera used here was adequate for experimentation, and shows how a service management platform can support devices of different protocols. [1]

The devices form only part of the test bed. A service platform which is able to gather devices and make them available to user services is required. The discussion in Chapter 2 highlights the importance of this, as the full potential of networked devices is not released until middleware, a *glue*, is available to join services and devices.

6.2.2 The service management platform

A service management platform is a platform which can manage and execute services. In the home environment, these kinds of services would include entertainment, security or climate control.

When selecting a service management platform for the test bed, some requirements had to be met. These requirements included:

- The platform is stable and reliable – since the gateway is running in the home, it has to run for long periods between restarts, typically months or even years.
- Robust – if one service fails, the rest of the platform should function as normal and not crash.
- Portable – be easy for third party developers to deploy their services.
- The ability to suppose a range of networking protocols and have the ability to accommodate new protocols.
- Give all services the opportunity to find and use devices.
- Dynamic – as services and devices change, the platform should be able to accommodate these changes without having to redeploy a new platform.

Currently, the OSGi framework is the only suitable candidate which meets these requirements.

At the time of development there were only three OSGi frameworks publicly available: the Sun Java Embedded Server [108], the IBM Service Management Framework [109] and an open source implementation, Oscar [110].

[1]Provided they have drivers for the service platform.

The offering from Sun, the Java Embedded Server (JES), was an implementation of version 2 of the OSGi specification. However, the implementation was not OSGi compliant. Further, version 2 has been superseded by version 4, therefore it would be advantageous to use the newer specification. Version 3 of the OSGi specifications include UPnP, which was not included in version 2 of the OSGi specification. Since many of the devices used in the test bed were UPnP based, the Sun implementation was unsuitable.

The open source implementation, Oscar, was an implementation of version 1 of the OSGi specification. Further, this implementation was not reliable, or mature for full scale use. Therefore, it was decided not to use this framework.

The IBM implementation, the Service Management Framework (SMF) version 3.5 was an implementation of version 3 of the specification. It was also an OSGi compliant platform. This platform was mature and reliable, but lacked a UPnP driver.

Since this platform was the most reliable and mature, it was chosen as the service platform used for experimentation. The framework was run on a desktop PC. The specification of this computer was modest – Intel Pentium III 500MHz processor with 128Mb of RAM and a 10Gb hard drive. The machine was running Microsoft Windows XP SP1, with the Sun Java runtime environment, version 1.4.2. As a UPnP driver was not distributed with the IBM OSGi framework, one had to be created.

The UPnP OSGi driver

Although the UPnP driver had not been implemented by IBM, the specifications were included in [63]. Following these specifications, a driver was developed.

Effectively, the behaviour of the UPnP driver was similar to that of a UPnP control point. The role of the UPnP driver was to listen for new UPnP devices leaving and joining the network. When a device joins the network, it had to be registered in the framework's service registry as a UPnP Device service. When the device left the network, the device service entry had to be removed from the registry.

The driver itself was created by modifying the UPnP control point component of the CyberLink SDK. This is because the OSGi UPnP driver is simply a modified UPnP control point. As UPnP devices are self configuring, no user interface was required for the driver itself. It simply adds devices when they are connected to the network and removes them when they leave.

When a new device does join the network, the driver has to read the device XML description to get the device details. These details are added to the service description in the service registry. Figure 6.2 shows the UPnP driver service along with the service registry entry for a device it has added. In this example, the device registered was a UPnP thermometer. The properties from the thermometer device description XML have been parsed and used for the device description. Services can search these properties (line 5–21) in the service registry, as described in section 2.2.2.

Registering the UPnP devices in the framework allows other bundles to use the devices. However, the driver also has to allow other bundles to be notified of changes to a device state. For example, it would be useful for a climate control service to be notified of a change in temperature. To facilitate this, the OSGi specifications stated that the UPnP base driver would subscribe to all new devices. This allows other bundles within the gateway to register with the driver to be notified of changes in the device.

Although Release 3 of the OSGi specification includes UPnP it does not include specifications for X.10. Drawing inspiration from how the UPnP driver was constructed, and using themes from [5] to create device drivers, a similar driver was created for X.10.

The X.10 OSGi driver

The role of the X.10 driver is allow X.10 devices to be added and removed from the service registry in the framework. Since X.10 is not self configurable, details of new X.10 devices have to be manually entered.

The X.10 driver was developed used an existing open source Java X.10 API [40]. This code was taken and adapted for use within an OSGi environment. Communications port drivers from Sun were also required since the X.10. controller was connected to the serial port of the gateway. The Java Communications API [111] was used to access the communications (comm) port.

Like the UPnP driver, the X.10 driver had to support the addition and removal of devices from the gateway. Also, it had to offer a mechanism for listening to X.10 messages on the network. Adding devices was carried out manually via a web page, shown in Figure 6.3(a).

```
file:/h:/www/bundle/upnp-basedriver.jar [28]
id=28, Status=ACTIVE      Data Root=H:\smf\jarbundles\28\data
Registered services:
  {org.osgi.service.upnp.UPnPDevice, org.osgi.service.device.Device}=
  {service.id=27,
    UPnP.device.UDN=uuid:mewUPnPThermometerLaptop,
    UPnP.device.UPC=1234567890,
    UPnP.device.modelNumber=1.0,
    UPnP.device.parentUDN=uuid:mewUPnPThermometerLaptop,
    UPnP.device.serialNumber=000001,
    UPnP.device.modelName=Thermometer,
    DEVICE_CATEGORY=UPnP,
    UPnP.device.modelDescription=UPnP Temperature Sensor v1.0,
    UPnP.presentationURL=http://www.cs.stir.ac.uk/,
    UPnP.device.modelURL=http://www.cs.stir.ac.uk,
    DEVICE_IP_ADDRESS=192.168.1.14:4004,
    UPnP.device.manufacturerURL=,
    UPnP.export=,
    UPnP.device.type=urn:schemas-upnp-org:device:TemperatureSensor:1,
    UPnP.device.friendlyName=UPnP Temperature Sensor,
    UPnP.device.manufacturer=mew
  }
No services in use.
Exported packages
  org.osgi.service.upnp[exported]
Imported packages
uk.ac.stir.cs.fi.manager.service<file:/h:/www/bundle/fi/manager.jar[30]>
  org.osgi.service.device; specification-version="1.1"<file:
                                     bundlefiles/osgi-services.jar [1]>
  org.osgi.service.upnp<file:/h:/www/bundle/upnp-basedriver.jar[28]>
  org.cybergarage.xml<file:/h:/www/bundle/xmlNode.jar[29]>
```

Figure 6.2: Service details for UPnP bundle, including service registry entry.

Removing an X.10 device from the gateway is shown in Figure 6.3(b). The list of X.10 devices is compiled by the servlet, searching the service registry for X.10 device services. This is then displayed on the page so the user can select the device they wish to remove. When the remove command is executed the device is removed from the service registry in the gateway.

In addition to adding and removing a device, the driver offers services an option to listen to X.10 messages on the network. The X.10 gateway can listen for X.10 commands on the power line. The X.10 gateway sends these messages to the serial port of the gateway, which the X.10 driver receives and distributes to interested parties.

The final stage of development of the test bed is the user services. These are the services which make use of the devices. The services were made into OSGi bundles which were deployed onto the framework.

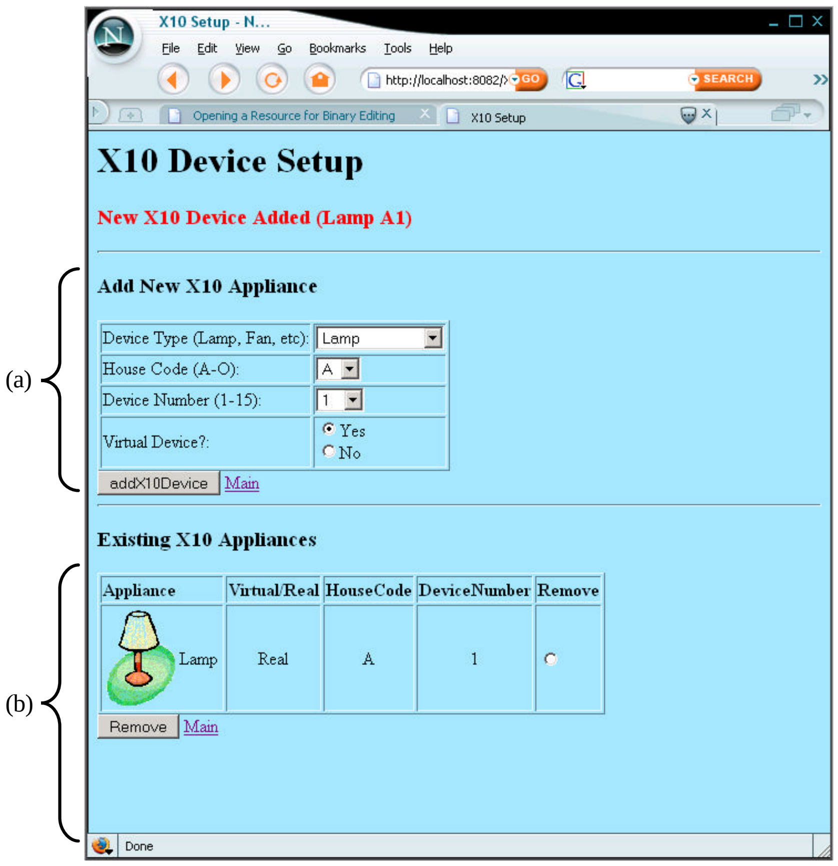
6.2.3 Services

The services implemented for use in the test bed are those discussed in Chapter 4. To recapitulate, there were five services:

- *Heating, Ventilation and Ai-conditioning (HVAC)* – The climate control service monitors the room temperature throughout the home and keeps it at a comfortable temperature, defined by the user. The service was developed, and a web interface was created for a user to control the service. This allows values to be entered for testing. The main screen of the service is shown in Figure 6.4.

- *Home Security Service (HSS)* – This service is used in conjunction with an alarm device. The service can be set, and configured, through the alarm device. The service can also be configured through a servlet, shown in Figure 6.5. The role of this service is to monitor the home for intruders and alert the owner of such an event. This service also has the away from home feature which makes the home look occupied when the home is empty.

- *Power Control Service (PCS)* – The aim of this service is to reduce the amount of power a home consumes. It does this by turning off devices when no one is home. The service can be set to


X10 Setup - N...
File Edit View Go Bookmarks Tools Help
http://localhost:8082/
GO
SEARCH
Opening a Resource for Binary Editing
X10 Setup
(a)
(b)
Done

X10 Device Setup

New X10 Device Added (Lamp A1)

Add New X10 Appliance

Device Type (Lamp, Fan, etc):	Lamp
House Code (A-O):	A
Device Number (1-15):	1
Virtual Device?:	[✓ Yes] No

addX10Device Main

Existing X10 Appliances

Appliance	Virtual/Real	HouseCode	DeviceNumber	Remove
Lamp	Real	A	1	○

Remove Main

Figure 6.3: X.10 setup servlet

turn on devices which consume a lot of energy when electricity is cheaper – a washing machine, for example.

- *Home Entertainment Service (HES)* – This service controls entertainment devices in the home, such as the television, stereo, and VCR/DVD devices. One of the main features of this service is that it can in principle monitor viewing habits and automatically record certain television shows. The service does allow an option to manually set the time and date to record a television programme, shown in Figure 6.6.

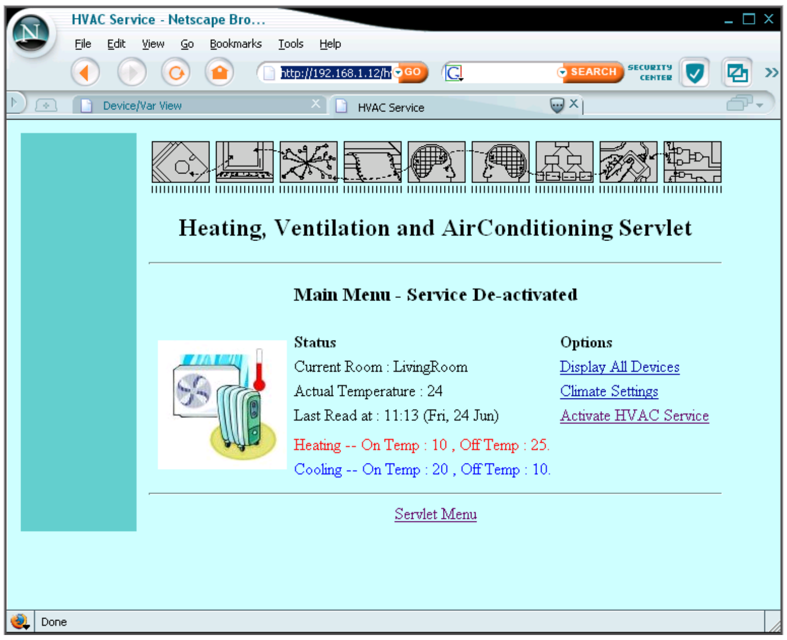
- *Communications Support Service (CSS)* – This service supports the use of email and telephone. The user can be notified of email arriving through their television. Also, any incoming telephone calls can be displayed on the screen; the user can decide whether to accept the call or not.

Each of the above services was implemented as an OSGi bundle.

Each service had a specific goal, e.g. the HVAC service was to keep room temperature at a comfortable level, the security service was to keep the home secure. To achieve these goals, services have to use devices. All services were developed to query the service registry for the required devices.

The HVAC service will be used as an example here. This service had to keep the home at a comfortable temperature. The temperature levels are set manually, but until then default values are used. For this service to function effectively, a minimum of three devices is required: a thermometer, a heater and an air conditioner.

When the HVAC service starts, it searches the service registry for a thermometer device to ascertain the room temperature. If one is found, the service registers with the thermometer service as the HVAC service needs to be notified of changes in temperature. If no thermometer is found, the service would remain in an idle state until a thermometer is introduced into the network. The service could be designed


HVAC Service - Netscape Bro...

Heating, Ventilation and AirConditioning Servlet

Main Menu - Service De-activated

Status

Current Room : LivingRoom

Actual Temperature : 24

Last Read at : 11:13 (Fri, 24 Jun)

Heating -- On Temp : 10 , Off Temp : 25.

Cooling -- On Temp : 20 , Off Temp : 10.

Options

Display All Devices

Climate Settings

Activate HVAC Service

Servlet Menu

Figure 6.4: HVAC setup servlet

with default times when it starts. However, this is a service design issue and is therefore not relevant here.

When the temperature in the room drops below the minimum temperature, the HVAC service will search the service registry on the gateway for a heater. The registry returns an array of devices which match the search criteria. The service will then turn all on. This means that if two heaters were returned, both would be turned on. The advantage of searching for devices only when the service needs them is that a newly introduced device can be used without having to restart the service.

As the heater heats the room and the room temperature increases, the HVAC service is notified of the increase in temperature by the thermometer. When the temperature reaches the desired level, the service will turn the heating devices off.

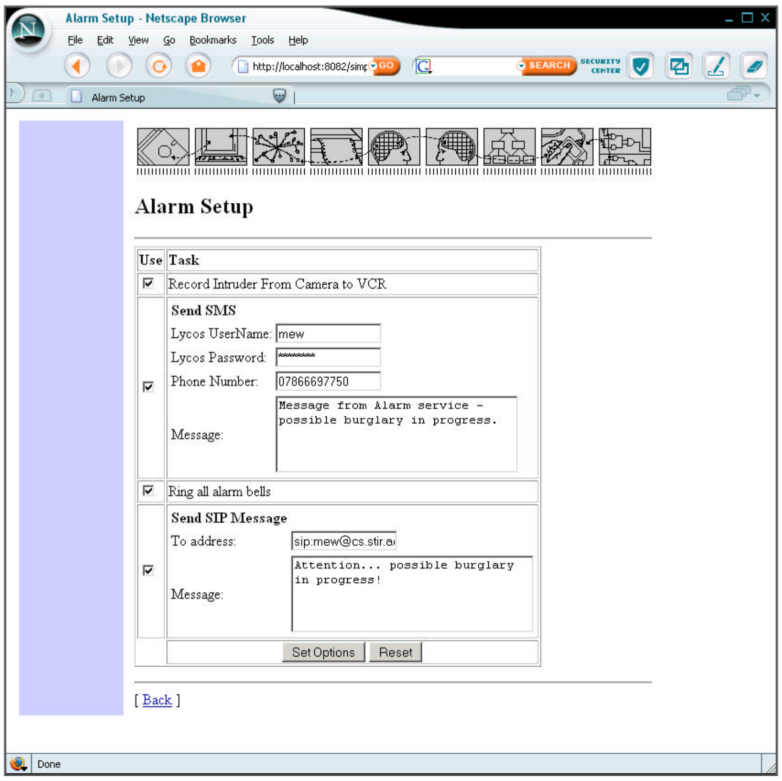
As stated previously, each of the services has been implemented as a series of OSGi bundles. Each is designed to search the service registry in the gateway for the type of device they require. If a service cannot find a device, it remains idle until a suitable device is added to the network. A service which requires more than one device, the alarm service for example, will use the devices available. This means that if one of the devices is unavailable, perhaps the alarm bell for example, other functions would be carried out as normal, e.g. sending a SMS.

In addition to the services detailed in Chapter 4, three other services were implemented: an SMS service, a SIP Instant Messaging (IM) User Agent (UA) service, and a web camera service.

The OSGi SMS bundle service works in conjunction with the Lycos SMS service [112]. This service requires a Lycos username and password along with the mobile phone number and message to be sent. A connection was then made to the Lycos service and the SMS message sent. This is particularly useful if a home service requires the owner to be notified urgently, e.g. a suspected burglary.

The SIP IM UA bundle was developed using the SIP stack from NIST [113]. This is an open source Java based SIP IM stack which was adapted for use in an OSGi gateway. The service was set to register with the SIP proxy. When registered, the service could then be called by other services to send instant messages, given an email address and message.

A web camera service had to be created to allow a connection from the gateway to the USB web camera which is on a USB port. An OSGi service bundle was developed which made a connection to the USB camera, using the Java Media Framework API (JMF) [114]. When the connection is active between the gateway and the device, images are streamed from the camera to the service on the gateway. Other services can then use this service to get images from the web camera.


Alarm Setup - Netscape Browser

http://localhost:8082/simp

Alarm Setup

Use	Task
☑	Record Intruder From Camera to VCR
☑	**Send SMS** Lycos UserName: mew Lycos Password: ******* Phone Number: 07866697750 Message: Message from Alarm service - possible burglary in progress.
☑	Ring all alarm bells
☑	**Send SIP Message** To address: sip:mew@cs.stir.a Message: Attention... possible burglary in progress!
	Set Options Reset

[Back]

Done

Figure 6.5: Home alarm setup servlet

The basic test-bed was then complete. The home network has a selection of devices: real X.10 devices, some virtual UPnP and X.10 devices, and a real USB webcam. The user services automate the home in a number of ways, from automatically controlling the living temperatures, to recording the owner's favourite show. The component which makes this connectivity possible is the service management platform, the OSGi gateway.

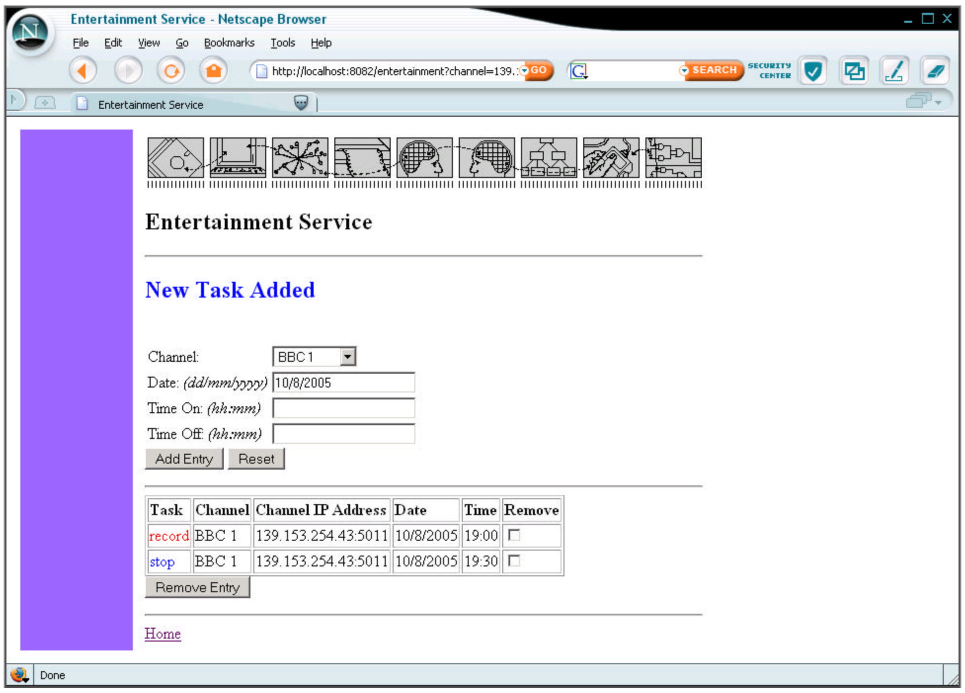
At this point the devices can be started and services installed. However, before they can be used, both devices and services have to be configured.

6.3 Configuration of the basic test-bed

6.3.1 Setting up the X.10 devices

Since the X.10 protocol does not support automatic configuration, the new X.10 devices have to be added to the framework manually. A servlet was created to facilitate this (shown in Figure 6.3(a)). Adding a service is straightforward. The type of device is selected, e.g. fan, lamp, window, then the address is entered. A user then selects *Add X10 Device*. The virtual radio buttons were only used for testing. This did not affect the way in which the gateway dealt with the device. When devices are added, changes to the X.10 devices are saved to the file system. This means when the gateway restarts, the X.10 configuration data is loaded and the user does not have to re-enter the data.

When the 'add' button is selected, a new X.10 device service is created in the framework. The details entered into the service registry are similar to those used for UPnP. Figure 6.7 shows an entry for an X.10 lamp in the service registry. This shows the basic details of the device: the type (Lamp) and address (A1).


Entertainment Service - Netscape Browser
File Edit View Go Bookmarks Tools Help
http://localhost:8082/entertainment?channel=139.:
GO
SEARCH
SECURITY CENTER
Entertainment Service
Done

Entertainment Service

New Task Added

Channel:	BBC 1
Date: *(dd/mm/yyyy)*	10/8/2005
Time On: *(hh:mm)*	
Time Off: *(hh:mm)*	

Add Entry Reset

Task	Channel	Channel IP Address	Date	Time	Remove
record	BBC 1	139.153.254.43:5011	10/8/2005	19:00	☐
stop	BBC 1	139.153.254.43:5011	10/8/2005	19:30	☐

Remove Entry

Home

Figure 6.6: Television recording setup of Entertainment Service

6.3.2 Setting up the UPnP devices

UPnP devices, by their nature, require little work to set up. A room value was set in the service description of each device. The default value used was *living room*. This could be changed via a control point if a device was to be put elsewhere. However, since the majority of testing was concentrated in one room, this was kept as the default.

When each UPnP device was started, it was automatically registered with the OSGi gateway. When the UPnP devices were registered in the service registry, they were ready to be used by other services.

6.3.3 Configuration of the services

The five services listed above each had to be configured for the home. Using the servlets, the services were configured in the following way:

Heating, Ventilation and Air-conditioning (HVAC): The climate control service was set with two types of values: those that conflicted and those which did not. The conflicting values dealt with where the maximum heating value overlaps with the triggering value for cooling.

Home Security Service (HSS): In the event of an intruder being detected, this service was set to send an SMS message, send an instant message using the SIP IM client, record the intruder from the web camera to the VCR, and then ring the alarm bell.

Power Control Service (PCS): This was set to turn all devices off once the owner left the home. There were no exceptions in the devices that could stay on.

Home Entertainment Service (HES): This service was set to automatically record a television channel at a certain time and date. Monitoring an owner's viewing habits was not included.

Communications Support Service (CSS): This service was set to display email only; the telephone option was not included for testing. An email account was configured for the email aspect. Since push email is not supported with the email server used, the service has to poll at regular intervals to check for new mail.

```
file:/h:/www/bundle/x10driver.jar [24]
id=24, Status=ACTIVE      Data Root=H:\smf\jarbundles\24\data
Registered services:
  {uk.ac.stir.cs.service.x10.X10LampModule, org.osgi.service.device.Device}=
  {service.id=34,
    DEVICE_CATEGORY=X10Module,
    DEVICE_TYPE=Lamp,
    DEVICE_ROOM=A,
    DEVICE_ROOM_FRIENDLY=LivingRoom,
    DEVICE_NUMBER=1
  }
No services in use.
Exported packages
  uk.ac.stir.cs.service.x10[exported]
Imported packages
  uk.ac.stir.cs.fi.manager.service<file:/h:/www/bundle/fi/manager.jar[30]>
  org.osgi.service.device; specification-version="1.1"<file:
                                     bundlefiles/osgi-services.jar [1]>
```

Figure 6.7: Example service entry for an X.10 lamp

Now that services had been configured and devices set up, testing was required to show that the basic test-bed worked.

6.4 Testing of the basic test-bed

To ensure the test-bed was stable and did what it was meant to do, some basic testing was carried out. This included testing the devices and also the services which controlled the devices.

6.4.1 Testing the devices

Testing UPnP devices

As previously stated, the UPnP devices were created using CyberLink SDK (version 1.1). Although the devices appeared to work and function correctly, they were tested using the Intel Device Validator Tool [115] (Figure 6.8). This tool tests the discovery of the device, control of the device and eventing (subscription). All devices passed the tests.

The devices could also be seen in control points from Siemens [116], Intel and CyberLink. All devices appeared in these and functioned correctly, i.e. returned correct values and allowed users to control the device. Therefore, this confirmed that these UPnP virtual devices were correctly implemented. If they were not, they would not have appeared correctly in the control points, or have passed the Intel UPnP device validator.

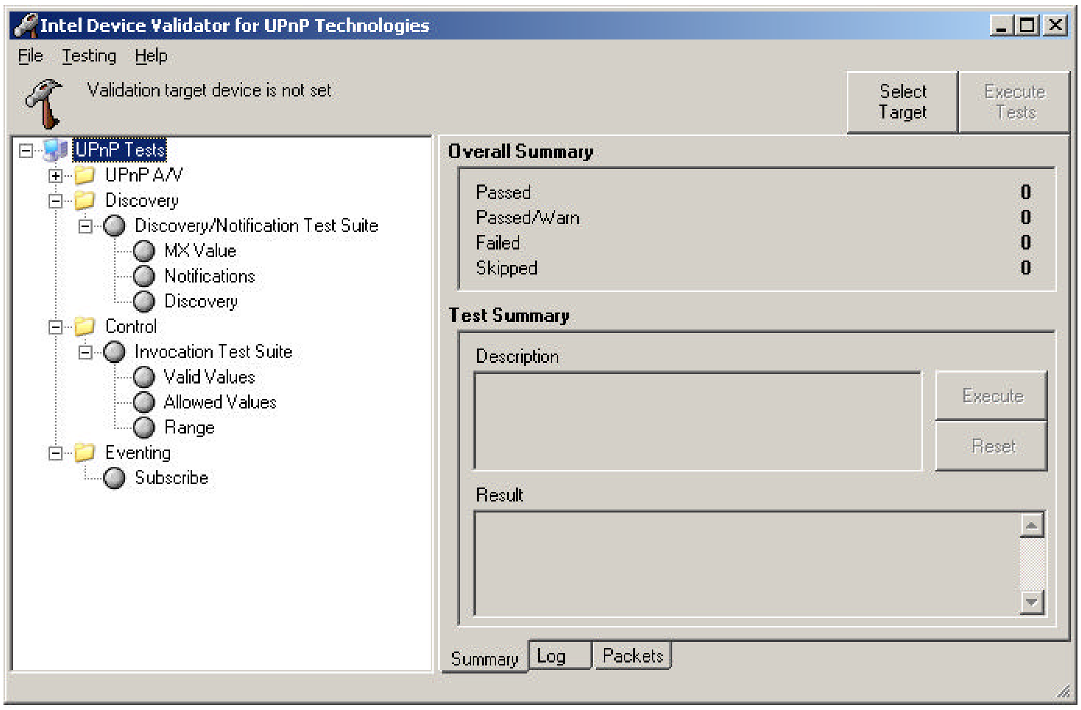
Testing of the X.10 devices

After the X.10 devices were connected to the network and manually entered into the framework, a test servlet was used to control the devices. The X.10 servlet offered basic functions: turn a device on or off. For lamp modules, dimming was supported. Since the X.10 devices used were real, the tests would either work or not work. These tests were carried out successfully and devices worked as expected.

6.4.2 Testing the services

Each of the five services had to be tested to make sure they worked in the OSGi gateway, and that they found the correct devices.

Heating, Ventilation and Air conditioning (HVAC): When the service started it found the thermometer and received the current room temperature. Since the service had registered with the thermometer, when the temperature changed, the service was notified. If the room temperature dropped below the set minimum temperature the service found a heater and turned it on. This meant the room temperature increased and the service kept the heater active until the room



Validation target device is not set

Passed	0
Passed/Warn	0
Failed	0
Skipped	0

Figure 6.8: Intel UPnP device validator tool

temperature reached the set upper value. Similarly, if the room exceeded the maximum room temperature, the service would find an air conditioner and turn it on, thus cooling the room.

Home Security Service (HSS): When this service started, it registered with the motion sensor in the framework. When movement was detected, the service was notified causing the service to send an SMS, send an instant message, start ringing the alarm bell, and use the web-camera to record the room on the VCR. If one or more of the devices were not available, the service would continue to the next action.

Power Control Service (PCS): This behaved as expected. When the service reached a set time, it turned all devices off.

Home Entertainment Service (HES): This service was used to set the VCR to record certain channels at certain times. Testing found that this service behaved as expected – it used the VCR to record shows at specific times.

Communications Support Service (CSS): This service polled the configured email account every five minutes for new mail. When new mail was received, the service would find the television device and send an image to it. The television would then display the image, along with the current picture.

Through basic tests it was found that the services, on their own, behaved as expected. They all found the relevant devices and sent the correct commands to them. The devices executed these commands successfully.

With a stable test-bed, the approach was implemented as a series of OSGi bundles. The next section will discuss the design and testing of the approach.

6.5 Design of the approach

As stated, the approach described in Chapter 5 has been implemented as a series of bundles which run on the gateway. The approach is able to avoid interactions by carefully controlling access to devices and environmental variables. Control is achieved through a locking algorithm. To turn the approach from theory into a working prototype, five key components were developed. These were:

- *The Service Interaction Manager (SIM)*[2] – This is the main component of the approach. This component has to intercept commands before they leave the framework to the device. The manager has to analyse the commands and determine whether they would cause a negative interaction. To do this, the manager uses a series of other services.
- *Protocol converters* – The approach has to handle a number of protocols, and even be expanded to handle new, or proprietary protocols. Essentially, these converters translate a protocol specific message to a format the manager can use.
- *Device information* – For this approach to work, it is important that information regarding *how* the device works is required. This is used to work out whether an interaction may occur. The manager needs to know *what* a device does and *what* variables it will affect when it carries out an action.
- *View of the network* – It is important that the manager has an overall representation of the home network. This view includes devices, their location, current state and variables they affect.
- *Service priorities* – As highlighted in Chapter 5, safety services need to be given priority over less important services. This component is used for setting priorities and for other services to get a priority of another service.

These five key points were the main development areas for the implementation of the approach. They are shown in Figure 6.9. Since the requirements for each component are well defined, each was implemented as a separate OSGi service bundle. The advantage of building the approach in a modular fashion is that it becomes flexible and can easily be upgraded as protocols and devices change.

Each of these bundles will now be discussed in more detail.

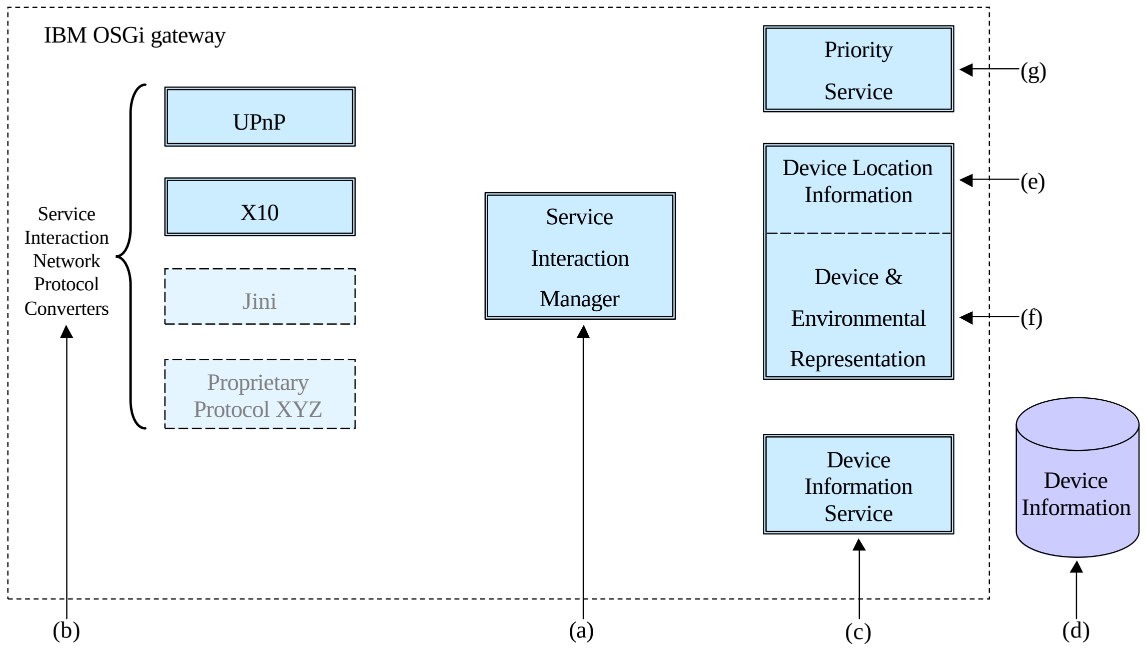

IBM OSGi gateway
Service Interaction Network Protocol Converters
UPnP
X10
Jini
Proprietary Protocol XYZ
Service Interaction Manager
Priority Service
Device Location Information
Device & Environmental Representation
Device Information Service
Device Information
(a)
(b)
(c)
(d)
(e)
(f)
(g)

Figure 6.9: Bundles for the approach

[2]The words SIM and manager are used interchangeably in the text

6.5.1 Service Interaction Manager (SIM)

The central bundle in this approach is the Service Interaction Manager (SIM), or manager, shown in Figure 6.9(a).

The manager itself does little processing work. Instead, it has a list of tasks and uses services from other bundles to get the answers.

When a new message is received by the manager, it has to work out whether the message will cause a negative interaction if left to proceed to the device. To decide if there will be a problem with the message, the manager has to convert it from a protocol specific message to one it can understand.

Since the manager is independent of networking protocols, it used the Service Interaction Network Protocol Converters (SINPC) to translate the message. The SINPC services return the message in a format which is understood by the manager. When the manager has the translated command, it sends this to the Device and Environmental Representation (DER) component. It is this component which decides whether there is a conflict. If there is a conflict, the manager uses the priority service to get the priority of the service which sent the message. If the calling service has a lower priority than the service currently holding the locks, the message is rejected and the interaction is avoided. On the other hand, if the priority is higher for the calling service, the command will be sent to the device. If the priority is the same, the manager works on a first come, first served basis. It must be noted that a user can change priorities at any time, if their preferences change.

Services are not notified if they lose control of a device to a service with a higher priority. This approach presented here has no direct communication with the services. This is because when the manager starts communicating with services, for the manager to avoid interactions properly, all services must be aware of the manager. This should be avoided as it cannot be guaranteed that service vendors will develop their services to work with the manager.

As well as intercepting messages being sent from service to device, the manager also has to handle notification messages being sent from the devices. These notification messages do not get sent directly to the manager; instead, they are processed by the SINPC. The SINPC translates the messages into a format the manager understands. These messages are then sent directly to the DER to ensure the internal model of the network is maintained.

The manager provides a log which can be used to show whether interaction has been detected as shown in Figure 6.10. In the figure, it can be seen that no interaction was detected in Figure 6.10(a) and (b). However, in Figure 6.10(c) an interaction was successfully avoided.

6.5.2 Service Interaction Network Protocol Converters (SINPC)

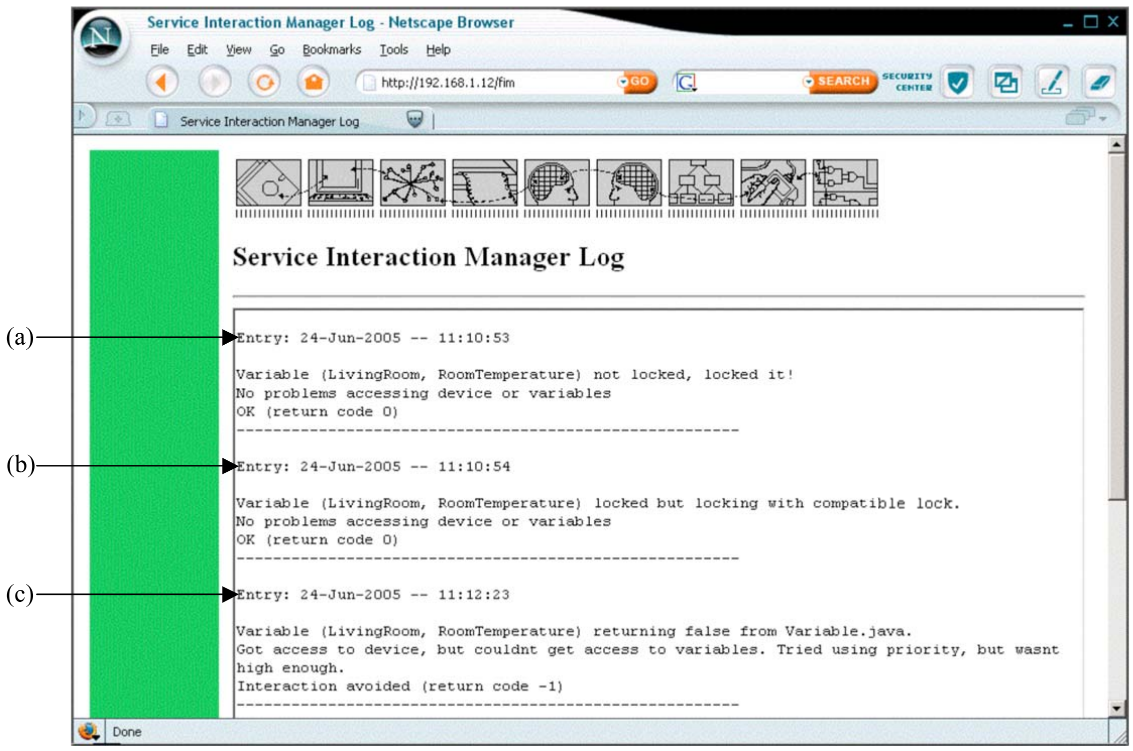
The manager relies on the SINPC (Figure 6.9(b)) to carry out message translation. The SINPC are a series of individual bundles which are different implementations of the `uk.ac.stir.cs.fi.protocol.parser.FIProtocolPa` service. This service was created as part of the project for parsing different protocols.

By implementing the protocol parser service as a series of bundles which implement the service, it means that a wide selection of protocols can be translated. Figure 6.9(b) indicates two protocol converters were implemented (UPnP and X.10) by surrounding them in dark lines. The dashed lines shown in the Figure give an example of other protocols which could be implemented.

The manager makes use of these services by searching the framework service registry for services of type `uk.ac.stir.cs.fi.protocol.parser.FIProtocolParser`. From the search, the manager will get an array of all services found. Figure 6.11 is an extract from the framework service registry for the UPnP protocol parser service. Since the manager does not know what protocol the message uses, it sends the message to each protocol parser service individually. The manager could be enhanced to improve efficiency so it learns to recognise messages. The manager would then know what the message means so as to avoid sending it to each of the converters.

If the protocol parser does not understand the message (e.g. an X.10 message is sent to the UPnP component) a value of `null` is returned, otherwise a `String` is returned which contains the message in a format the manager can understand.

The format of the returned message is in the following format:
`Calling-service-id; device-type; device-id; command; parameters`


Service Interaction Manager Log - Netscape Browser
File Edit View Go Bookmarks Tools Help
http://192.168.1.12/fim
GO
SEARCH
SECURITY CENTER
Service Interaction Manager Log
Done
(a)
(b)
(c)

Service Interaction Manager Log

```
Entry: 24-Jun-2005 -- 11:10:53

Variable (LivingRoom, RoomTemperature) not locked, locked it!
No problems accessing device or variables
OK (return code 0)
----------------------------------------------------------

Entry: 24-Jun-2005 -- 11:10:54

Variable (LivingRoom, RoomTemperature) locked but locking with compatible lock.
No problems accessing device or variables
OK (return code 0)
----------------------------------------------------------

Entry: 24-Jun-2005 -- 11:12:23

Variable (LivingRoom, RoomTemperature) returning false from Variable.java.
Got access to device, but couldnt get access to variables. Tried using priority, but wasnt
high enough.
Interaction avoided (return code -1)
----------------------------------------------------------
```

Figure 6.10: Service interaction manager log

```
Registered services:
  {uk.ac.stir.cs.fi.protocol.parser.FIProtocolParser,
   uk.ac.stir.cs.fi.upnp.service.FIUPNPService} =
    { Description=Feature Interaction UPnP Component,
      service.id=18 }
  }
```

Figure 6.11: Extract from OSGi service registry for UPnP protocol converter

Therefore, an example returned value is:
`47;AirConditioner;23;SetDeviceState;off`
Where the calling service was 47 (the climate control service), and the device type was the air conditioner. The unique identification number of the air conditioner device in the framework was 23. The service wanted to turn the air conditioner off, by setting the device state to off. The numeric values of devices and services are only used by the system, therefore they do not need to be human readable.

A secondary role of the SINPC is to inform the manager of any change in their devices. Each protocol parser should register to be notified of updates for their device type. For example, the X.10 protocol parser should subscribe to all X.10 devices.

Therefore, when a new state message is received, the message is parsed into the same format and sent to the manager. New states can be triggered when a person in the home manually turns on or off a heater, for example. Since these new states are not instigated by a service, the first parameter in the manager's message, `calling-service-id`, has to be set to '1', which means manual change.

The advantage of translating protocol messages this way is that as new protocols are developed and introduced, the approach can accommodate them with little hassle. An implementation of the `uk.ac.stir.cs.fi.protocol.parser.FIProtocolParser` service is all that is required. New protocol parser bundles can then be installed onto the gateway. The manager will then find them the next time it searches for the protocol parser services to process a message.

The other advantage of implementing this in a modular fashion is that since there are so many home networking protocols, the home owner only needs protocol converter bundles for the protocols they have in their home.

6.5.3 Device Information

The Device Information Service (DIS) is shown in Figure 6.9(c). Given a device type and command, this service will return the environmental variables the device with this command will affect, and how. There are two parts to this service. The first part is the OSGi service which runs on the gateway (Figure 6.9(c)). The second part is the remote database which contains the information about the device (Figure 6.9(d))

Device information database

The device information database is a database which holds all device details including their commands and the environment variables these commands affect. To implement this, a database server was required. The database used was MySQL Server version 4.0.18-nt [117]. MySQL is a popular database which is freely available. It is reliable, fast and scalable (with the largest table it can support being 64TB [117]).

For the purpose of experimentation, the database server was installed onto the same computer as the gateway. In reality, this database would be hosted with a service provider. Having a database with all possible devices in a person's home would be impractical. Not only would this database be extremely large, but keeping it up to date with new devices would be difficult. For these experiments it did not make a difference where the SQL database resided.

The database designed consists of four tables. Figure 6.12 shows the relationship between the four tables. The 'Devices' table holds details such as the device name, whether it is an input or output device and what the default lock for the device is. The 'Actions' table holds the names of the actions, including the name of the action argument. The 'DeviceAction' contains only the suggested device usage. This value will override the device default value as the default device usage is dependent on the action. The 'EnvironmentalVariable' table has details of each environmental variable; this depends on the device and what the action is. Details for the variable, such as the name of the environment variable, default usage, duration and locality are all held here.

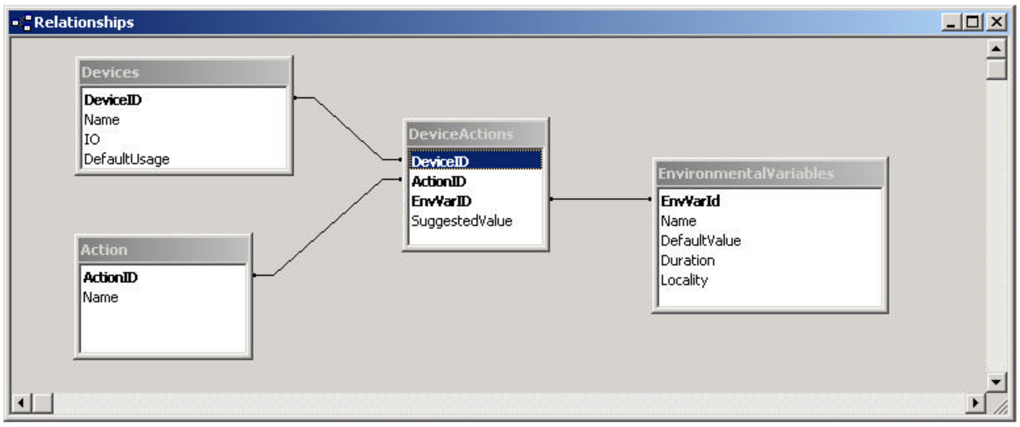


Figure 6.12: Relationship diagram of device information database

Once the database was set up, records were manually entered. The devices used for experimentation were all included and all details were filled in accordingly. Keeping the database up to date is outside the scope of this project, however getting data into databases through clients and web interfaces is well understood.

The Device Information Service (DIS)

The DIS (Figure 6.9(c)) works closely with the database (described above) to get information about devices. A service on the gateway will call the DIS with certain parameters, and will receive an XML result.

The DIS has one method which takes three parameters: the device type, the command and the argument. This is converted into an SQL query and sent to the database. If the database is unable to find any matches, the DIS will return a `null` value to the calling service.

```
<Device>
    <DeviceType>Window</DeviceType>
    <DeviceIO>Output</DeviceIO>
    <DefaultDeviceUsage>NS</DefaultDeviceUsage>
    <Action>
        <Name arg="on">deviceOn</Name>
        <SuggestedDeviceUsage use="NS" />
        <EnvironmentalVariable name="RoomTemperature" defaultValue="S"
                                          duration="3" locality="0" />
        <EnvironmentalVariable name="RoomMovement" defaultValue="S"
                                          duration="1" locality="0" />
    </Action>
</Device>
```

Figure 6.13: Example return value from the device information service (output)

However, if the device is found in the database, the DIS processes the results from the database and puts them into an XML format, which is returned to the calling service. The XML schema used depends on the type of the device: input device or output device.

Figure 6.13 shows sample XML data for the heater device, which is an output device. This XML data shows all the variables this device can potentially affect.

```
<Device>
  <DeviceType>MotionSensor</DeviceType>
  <DeviceIO>Input</DeviceIO>
  <DefaultDeviceUsage>S</DefaultDeviceUsage>
  <SensorEnvironment variable="RoomMovement" />
</Device>
```

Figure 6.14: Example return value from the device information service (input)

Figure 6.14 shows sample XML data returned from the device information service for an input device. This XML data only shows the data that this device can monitor.

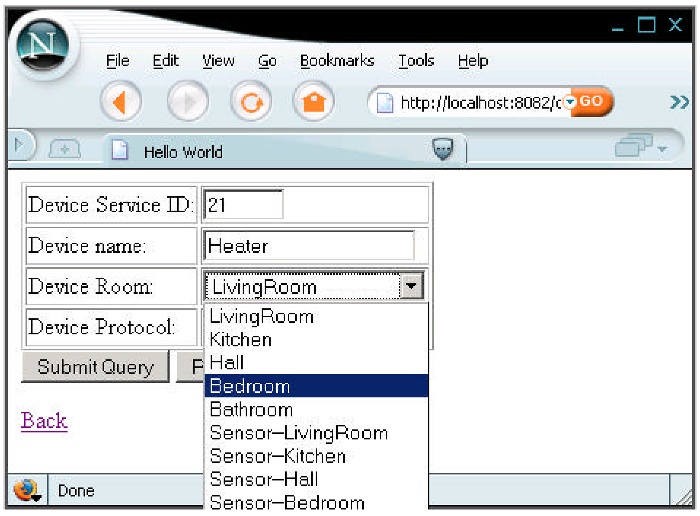
This service is only an interface between the gateway and the device database. The DIS only searches the database based on a device type with command. Processing and interpretation of the XML data is handled by the device and environment bundle.

6.5.4 Device Location Information and the Device & Environment Representation component

Figure 6.9(e) and (f) show the Device Location Information (DLI) and Device and Environmental Representation (DER), respectively. Essentially, it is the DER component which authorises a message and works out whether an interaction will occur. The feature interaction manager bases its decision on the result from the DER component.

The component has two important roles. Firstly, the DER component must handle the addition and removal of locks depending on the translated message sent by the manager. Secondly, the DER component listens for new devices registering in the gateway's service registry. When this occurs, the component will try and discover which room the device is in. This is achieved by interrogating the device. To do this, the DER must have some knowledge of the device protocol. For example, with X.10 the DER can work out the room from the address of the device. In the experimental UPnP devices, an extra field was included for this purpose. However, the DER must understand the protocol for this. If the DER cannot determine the location of the device, this can be input via a web page, shown in Figure 6.15.

To determine the type of device, the DER will search the properties in the service registry. If the device finds the device type, it will consult the DIS (Figure 6.9(c)). The device information service will then return device details and all variables the device could affect. If the device information service is


File Edit View Go Bookmarks Tools Help

http://localhost:8082/c

Hello World

Device Service ID:	21
Device name:	Heater
Device Room:	LivingRoom
Device Protocol:	

LivingRoom
Kitchen
Hall
Bedroom
Bathroom
Sensor–LivingRoom
Sensor–Kitchen
Sensor–Hall
Sensor–Bedroom

Figure 6.15: Set location of device

unable to return a value for the device, or the DER was unable to determine the type of the device, the DER offers a service to allow a user to input the details, device type, variables it affects, commands, etc. This form can be obtained by selecting any devices in part Figure 6.16(a). In the Figure, there are no *unknown* items – all are known Figure 6.16(b).

As this is a detailed form, which must be entered correctly, a user would not be expected to complete it. It is envisioned that having the user enter details of device and environmental variables would be a rare event as the remote device database will be comprehensive and contain most household devices.

At this stage, room, device type and variables are known and this information is placed in the internal representation, shown in Figure. 5.4. Figure 6.16(c) shows the DER internal view with some devices added: two heaters, an air conditioner and a thermometer. All devices are in the living room, and the only environmental variable is room temperature. Although there are potentially many variables in a room, to keep the table clear, only the variables used by devices are shown here. Therefore, although room light is a valid variable, it would not be included until a device was registered which affected this variable.

The DER representation adds devices to its internal view. Similarly, when a device is removed from the gateway, the device is removed from the internal representation and any locks the device may hold on environmental variables are released.

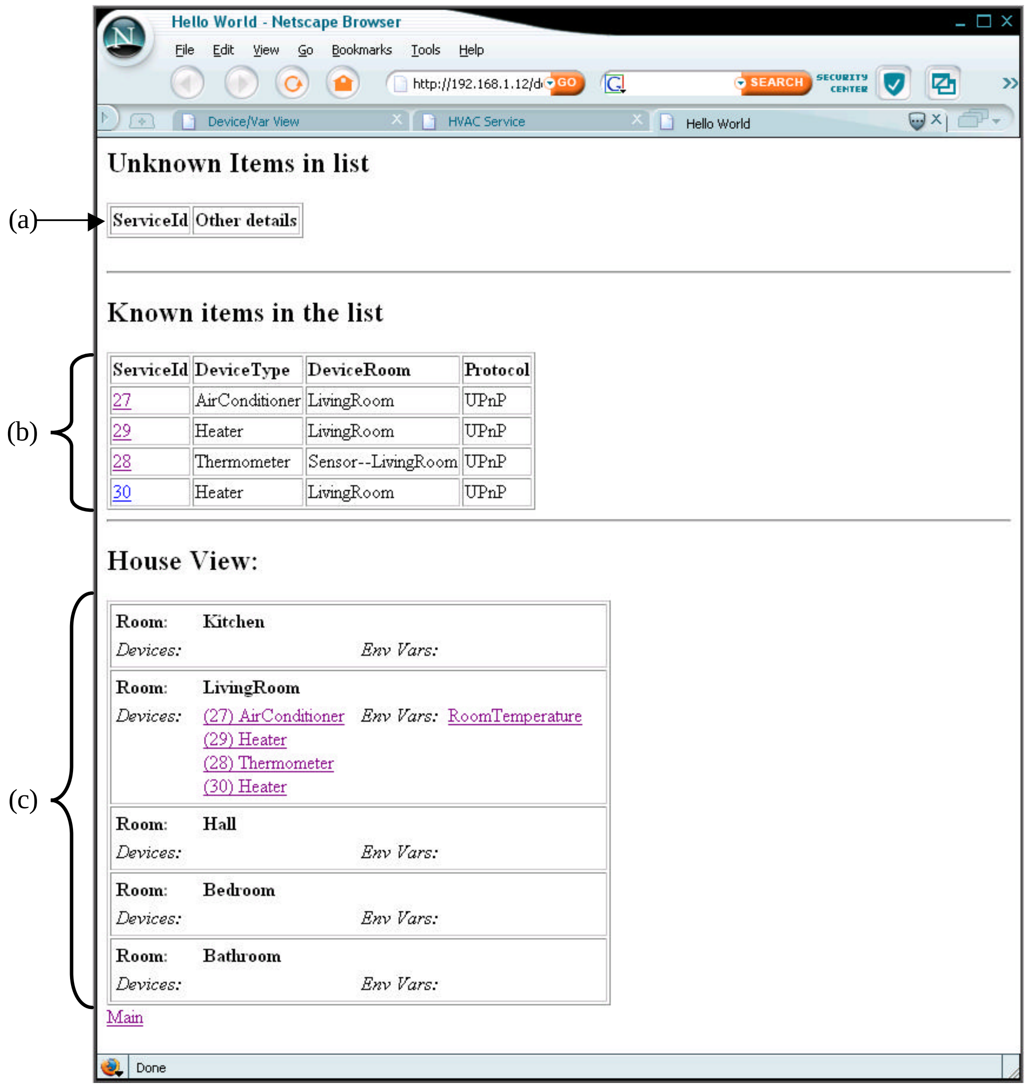
The internal representation held by the DER component is of a hierarchical structure (see Figure 5.4). At the top level is the *home*. Within a home there are *Rooms*, and within rooms there are *Devices* and *Environmental Variables*.

Part of the DER component is the Device Location Information (DLI). This information can potentially play an important role for other services. For example, if an entertainment service required a television in the kitchen, it could consult the DLI and a device object would be returned which may be used. The DLI and the DER work closely, and most of the work for determining location is done through the DER. This is why the DLI and the DER are included in the same service.

When the devices, real and virtual, were added to the framework, they were found by the DER. The location of UPnP devices which did not implement the location field did have to be entered manually. However, the remaining UPnP devices and all X.10 devices were added successfully. Adding and removing of devices is one task of the DER. The other role is to help avoid interactions.

Managing devices and locks

Along with the location of devices, the DER maintains a list of locks on devices and variables. A user can check the lock of a device or variable by selecting the item from the lists in Figure 6.16(c) (selecting a link in (a) or (b) will allow the user to enter room or device details, if the DER has incorrectly identified the device).



Unknown Items in list

ServiceId	Other details

Known items in the list

ServiceId	DeviceType	DeviceRoom	Protocol
27	AirConditioner	LivingRoom	UPnP
29	Heater	LivingRoom	UPnP
28	Thermometer	Sensor--LivingRoom	UPnP
30	Heater	LivingRoom	UPnP

House View:

Room:	Kitchen		
Devices:		*Env Vars:*	
Room:	LivingRoom		
Devices:	(27) AirConditioner (29) Heater (28) Thermometer (30) Heater	*Env Vars:*	RoomTemperature
Room:	Hall		
Devices:		*Env Vars:*	
Room:	Bedroom		
Devices:		*Env Vars:*	
Room:	Bathroom		
Devices:		*Env Vars:*	

Main

Figure 6.16: DER servlet to show internal view

To set a lock, the DER uses the rules and locks detailed in section 5.2.2. Therefore, when it receives the message from the manager to try and set the lock, it tries. If it fails, it will return `false` to the manager. The manager will then try using priorities. The message is resent to the DER and the DER will try applying the locks with the new priority. If it succeeds, a value of `true` will be returned, otherwise it will be `false`.

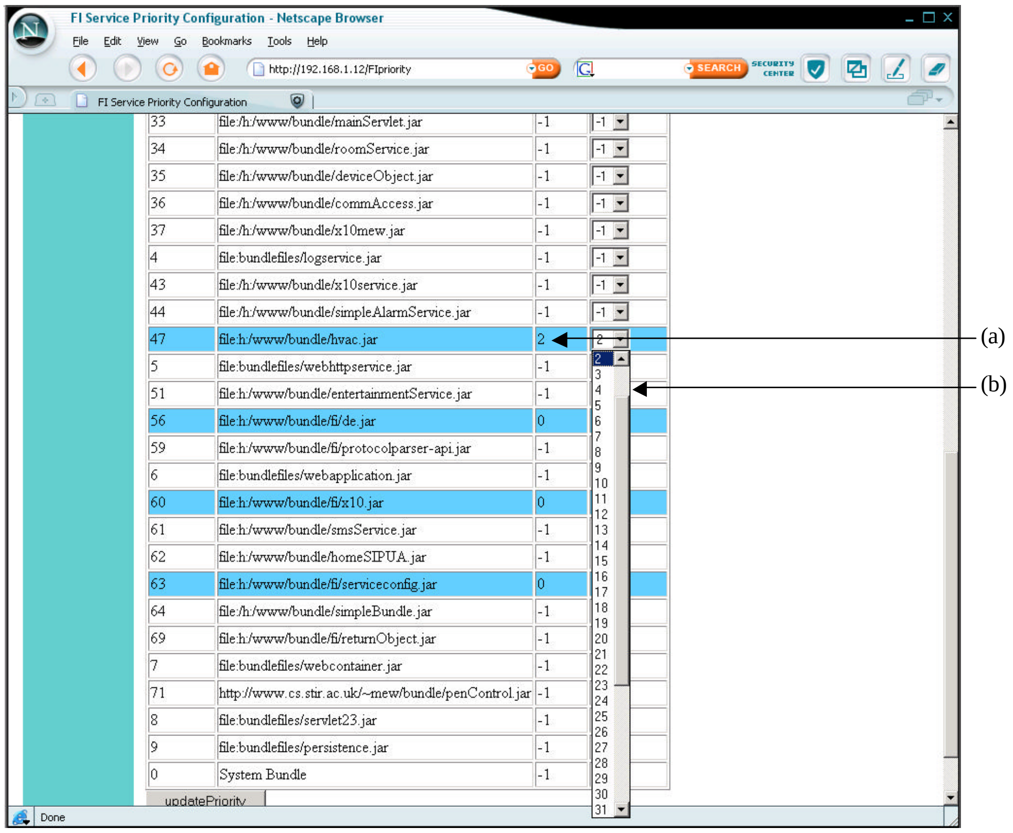
The same technique is applied when device state updates come through. However, when the manager sends them, they are automatically sent with a priority high enough to ensure the placing of locks is successful. As this *is* the state of the device, the manager has no choice. These updates from devices generally happen because of users interacting directly with the device. These commands *must* be accepted, even if it will cause an interaction.

6.5.5 Priority service

This bundle is shown in Figure 6.9(g). The task for this service is to maintain a list of priorities for all services in the gateway. The priority service can be called by other services with a service ID as a parameter. The priority service will then return the priority of the desired service.

When the service starts, all services in the gateway are gathered and assigned a priority of '-1'. This means that no priority has been set; all are treated equally. However, the exception to this is the FIM suite of bundles. They are all set to ‘0’ which is the highest, meaning they have the ability to override any decision.

A web page is provided to allow a user to set priorities, shown in Figure 6.17.


FI Service Priority Configuration - Netscape Browser
File Edit View Go Bookmarks Tools Help
http://192.168.1.12/FIpriority
GO
SEARCH
SECURITY CENTER
FI Service Priority Configuration
(a)
(b)
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
updatePriority
Done

33	file:/h:/www/bundle/mainServlet.jar	-1	-1
34	file:/h:/www/bundle/roomService.jar	-1	-1
35	file:/h:/www/bundle/deviceObject.jar	-1	-1
36	file:/h:/www/bundle/commAccess.jar	-1	-1
37	file:/h:/www/bundle/x10mew.jar	-1	-1
4	file:bundlefiles/logservice.jar	-1	-1
43	file:/h:/www/bundle/x10service.jar	-1	-1
44	file:/h:/www/bundle/simpleAlarmService.jar	-1	-1
47	file:h:/www/bundle/hvac.jar	2	2
5	file:bundlefiles/webhttpservice.jar	-1	
51	file:h:/www/bundle/entertainmentService.jar	-1	
56	file:h:/www/bundle/fi/de.jar	0	
59	file:h:/www/bundle/fi/protocolparser-api.jar	-1	
6	file:bundlefiles/webapplication.jar	-1	
60	file:h:/www/bundle/fi/x10.jar	0	
61	file:h:/www/bundle/smsService.jar	-1	
62	file:h:/www/bundle/homeSIPUA.jar	-1	
63	file:h:/www/bundle/fi/serviceconfig.jar	0	
64	file:/h:/www/bundle/simpleBundle.jar	-1	
69	file:h:/www/bundle/fi/returnObject.jar	-1	
7	file:bundlefiles/webcontainer.jar	-1	
71	http://www.cs.stir.ac.uk/~mew/bundle/penControl.jar	-1	
8	file:bundlefiles/servlet23.jar	-1	
9	file:bundlefiles/persistence.jar	-1	
0	System Bundle	-1	

Figure 6.17: Service priority configuration servlet

The servlet shows the current priority in Figure 6.17(a) and a selection of possible priorities in Figure 6.17(b). The dropdown list is created at runtime. The list runs from two to the total number of services plus two.

All five components were implemented as a series of OSGi bundles and installed onto the gateway. The manager could be turned on and off during testing. The results from testing are included later in this chapter. To summarise, the flow of information between the five bundles is displayed below.

6.5.6 Summary of the flow data

There are two scenarios when the FIM suite of modules is used: when a service sends a command to a device, and when a device sends an update on its state.

Flow of data when service sends to device

Figure 6.18 shows the order in which the components are called after a service (S_1) tries to send a command to a device (D_2). The numbers in brackets in the figure is the order of the flow of data.

First, the message is sent into the manager. The manager then routes the message to the protocol parser. In this example, the message was of type UPnP, so the manager does not have to try other parsers. The message is then sent to the DER where the DER tries to set the locks. Before the DER can set the locks, it must get the variables from the DIS. In turn, the device information service has to query the device database.

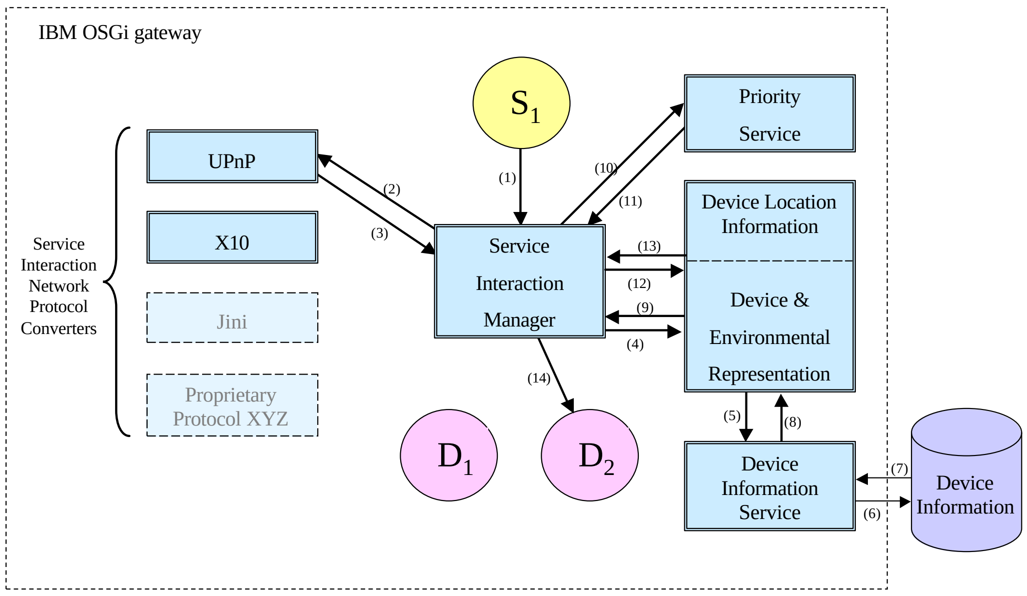

IBM OSGi gateway
S1
Priority Service
UPnP
X10
Jini
Proprietary Protocol XYZ
Service Interaction Network Protocol Converters
Service Interaction Manager
Device Location Information
Device & Environmental Representation
Device Information Service
Device Information
D1
D2
(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)
(9)
(10)
(11)
(12)
(13)
(14)

Figure 6.18: Flow of information sent from service to device

The results are then sent back, in XML form, from the DIS to the DER. The DER can then try to set the locks. In this instance, the setting of the locks fails. The manager then gets the priority from the priority service. Obtaining the priority, the message is then sent back into the DER. In this example, the priority was able to get the lock set and a value of `true` is returned to the manager. The manager then allows the message to be sent to the device, D_2.

Although there seems a lot of work here to send one instruction, no optimisation techniques have been employed in this implementation. To determine the performance of the manager, a test was carried out to turn on an X.10 lamp with, and without the manager running. Without the manager, the lamp took 1.5 seconds. If the manager was used, this figure increased to 2.7 seconds. This is the figure for the manager to determine the message, call the SQL database (over the network) and process the results to authorise the instruction.

Optimisation can be applied to the manager. Messages results from previous calls to the SQL database could be cached. Calls to the various protocol converters could be avoided if the manager learns to recognise messages. These are simple steps which could be applied to the manager. These will be discussed further in Chapter 8.

As well as commands being sent from the service to the device, there may be times where the device's state has changed. If a device is able to notify listening parties of its change in state, the manager should take advantage of this to keep its internal model up to date.

Flow of data when device sends update

Figure 6.19 shows the flow of data when a device sends a message for its change in state. Like the previous figure, the numbers by the arrows represent the order in which the data flows. Instead of the manager registering with the devices, it is the protocol converter that registers to receive updates from its type of devices. For example, the UPnP protocol converter would receive event notifications from all UPnP devices. Implementing it in this way is more efficient. If the manager did receive updates, it would have to send them to the SINPC anyway. This is simply more efficient and further avoids the manager being aware of specific protocols.

On receiving a message, the SINPC translates the message into the appropriate format for the manager. After the SINPC sends the message to the manager, the manager simply forwards the message to the DER.

Before the DER can update the internal model, it must call the DIS to determine the variables this device and action will affect and how. The DIS returns the appropriate values, and the DER is then able to update the internal model accordingly.

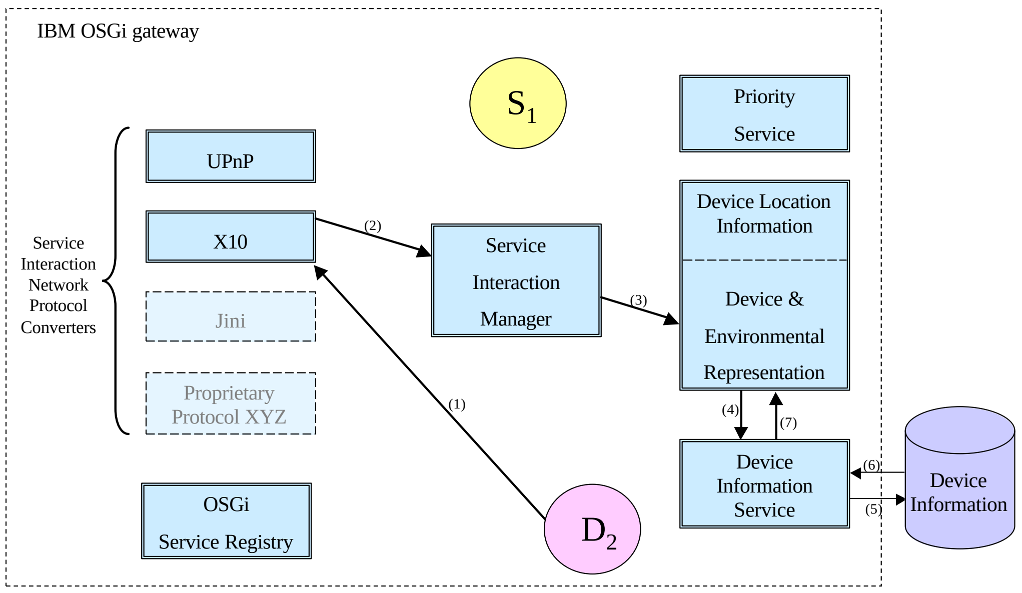

IBM OSGi gateway
S1
Priority Service
UPnP
X10
Jini
Proprietary Protocol XYZ
Service Interaction Network Protocol Converters
Service Interaction Manager
Device Location Information
Device & Environmental Representation
Device Information Service
Device Information
OSGi Service Registry
D2
(1)
(2)
(3)
(4)
(5)
(6)
(7)

Figure 6.19: Flow of information sent by device update

At this stage the test-bed is complete. All devices are set up and running and all services have been deployed. The FIM suite of bundles have been installed. Also, the SQL database is running and has been populated with test data. However, it is important to show that this has been implemented correctly and does work as expected. Some testing is required to show this works.

6.6 Testing of the approach

Rather than testing to demonstrate that the approach avoids interactions, the testing here is to show that the locks are placed and released correctly.

To show this, the HVAC service with a heater and thermometer was used. When the manager was started, the DER searched for all devices. It found both the thermometer and the heater. The DIS service was used to obtain information about both devices. The information was successfully retrieved from the database and the XML was returned to the DER.

The devices were now registered in the DER and both were unlocked. When the HVAC service was triggered to start the heater, it sent an on command to the heater device. This message was intercepted by the manager and processed. The message was parsed by the SINPC UPnP module and was passed to the DER. The DER then checked which variables this action would affect – in this instance room temperature.

Both the device and room temperature variable were unlocked, therefore no interaction would occur and the message was allowed to be forwarded to the device. The DER placed the locks on the device and variables. The locked variable is shown in Figure 6.20.

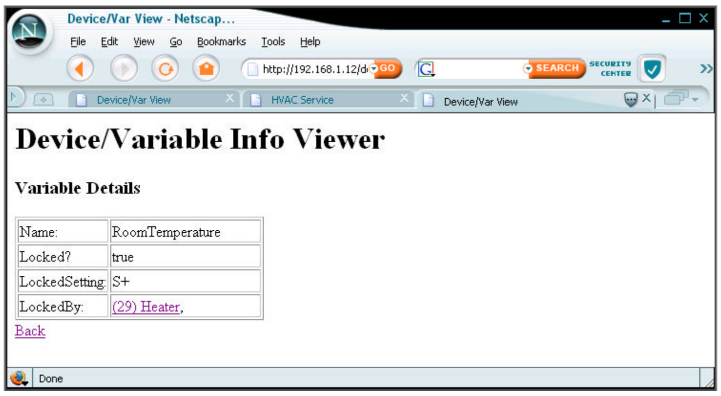
When the HVAC reaches the desired temperature, it turns off the device. The off command follows the same path through the manager, this message is approved and the lock on the device and the variable are released.

This test showed that the approach does work as expected.

6.7 Summary

This chapter has discussed the implementation of the approach detailed in Chapter 5. To implement the approach, a test-bed was required. The test-bed included services, devices and an OSGi framework.

For the services to be usable, devices were made available. The array of devices included real X.10 devices, as well as virtual UPnP and X.10 devices.



Device/Variable Info Viewer

Variable Details

Name:	RoomTemperature
Locked?	true
LockedSetting:	S+
LockedBy:	(29) Heater,

Back

Figure 6.20: Locked room temperature variable

Both services and devices were brought together using the IBM OSGi implementation. After the basic test-bed had been tested, the suite of Service Interaction Manager bundles was installed.

The test-bed with the feature interaction manager was tested and was found to be reliable and stable. Testing found that the manager behaved as expected.

The next chapter shows how effective the manager is in avoiding interactions.

Chapter 7

Experimentation and Results

To show the effectiveness of the approach, experimentation using the test-bed (described in Chapter 6) was carried out. A total of eleven scenarios was tested in the test environment. Most of the scenarios are taken from Chapter 4; however, these are only negative interactions. It is important to show that while the approach avoids negative interactions, it does allow services and devices to work together. The next section will list the eleven scenarios along with the results of testing. The results are summarised at the end of this chapter.

7.1 The test cases

For testing, all the devices and services were used. The devices used were: four input devices (thermometer, motion sensor, humidistat and carbon monoxide detector) and thirteen input devices (two heaters, an air conditioner, window, blinds, fan, TV, VCR, camera, lamp, an alarm control panel, dehumidifier and humidifier). For simplicity, all devices were placed in the same room, the living room.

The services used were as follows:

- HSS:Alarm – Home Security Service – basic alarm feature
- HSS:AFH – Home Security Service – away from home feature
- PCS – Power Control Service
- HES – Home Entertainment Service
- HVAC – Heating, Ventilation, Air-Conditioning (Climate Control Service)
- HCS – Humidity Control Service
- CMSS – Carbon Monoxide Safety Service

These services and devices, along with their environmental variables, were registered on the gateway. The static model, similar to that used in Chapter 5, is shown in Figure 7.1. Essentially, this is the internal representation the DER builds, however the DER does not include the services in its view.

The representation includes all the devices and all the services. It can be seen that the security service and carbon monoxide safety service have priorities set. Through experimentation, it was found that a 'first come, first served' approach was adequate for all but safety services. However, it must be noted that this does depend on user preferences. Therefore, a user may wish to set all services to have a priority.

For clarity, the diagrams used to explain each interaction will only include devices being used for the particular scenario.

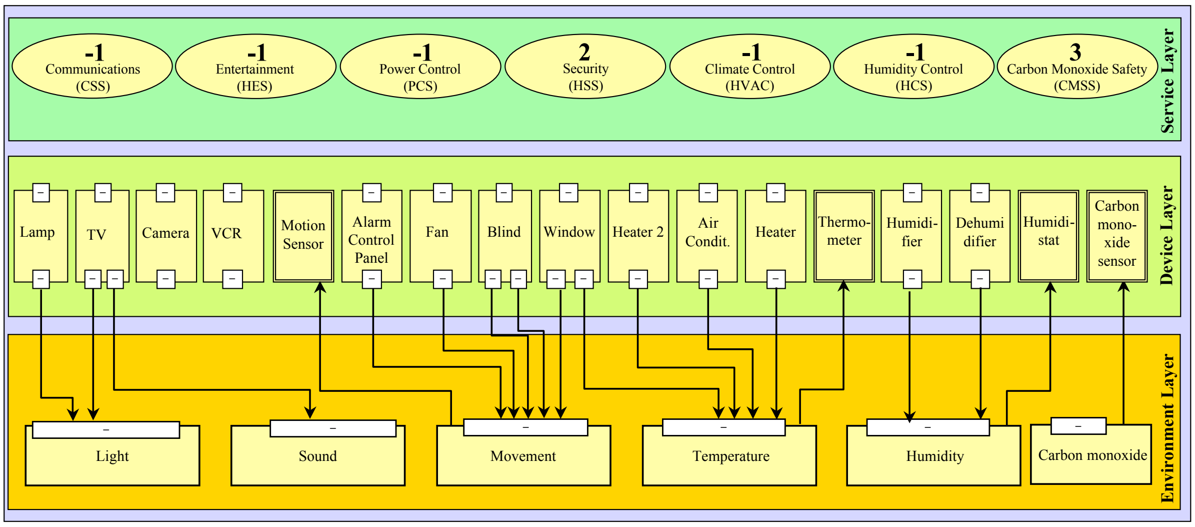


Figure 7.1: The representation created by the DER at runtime

7.1.1 Scenario 1: HSS:AFH vs PCS

The negative interaction discussed in section 4.3.1 is between the security service and the power control service.

The security service's away from home feature turns appliances on to give the impression that someone is at home. However, the power control service turns appliances off to save energy.

Assuming the security service is running first, it has locked the TV and lamp with *NS*, therefore no other service can access them. Consequently, when the power control service tries to gain access to turn the devices off, its request is denied (shown by the dashed lines in Figure 7.2). If, however, the power control service accesses and locks the devices first, since the security service has a higher priority, the power control service will be overridden by the security service. Therefore the security service will be allowed access to both the lamp and the TV.

Without using the manager, the interaction did occur – the devices were turned on by the away from home feature, and the power control service then turned them off. However, by using the service interaction manager, the interaction was successfully avoided.

7.1.2 Scenario 2: HSS:Alarm vs HES

The second negative interaction is taken from section 4.3.2. The interaction occurs between the security service alarm feature (HSS:Alarm) and the home entertainment services (HES).

The home entertainment service is recording a TV program on the VCR. However, the security service is triggered and wants to record pictures from the security camera on the VCR. Although the VCR device has already been locked (NS) by the home entertainment service (Figure 7.3(a)), due to the higher priority of the security alarm service, the home entertainment service has to give up control of the VCR. The VCR can then be reassigned to the security service. Thus, the picture of the burglar can be recorded on tape.

Suppose the HSS:Alarm is active first and the HES tries to record a programme. Here, the HSS:Alarm feature has acquired the VCR and locked it with *NS*. The home entertainment service tries to gain access, but it is refused as it has a lower priority than the security service, this is shown in Figure 7.3(b). Thus this scenario is successfully avoided.

Without using this approach, if the entertainment service was recording first, the security alarm feature would be able to access the VCR device. However, if the security service was recording first and the entertainment service then tried to record a show, it would be able to gain access to the device and record its programme. Clearly, it would be advantageous to record the intruder rather than a television programme.

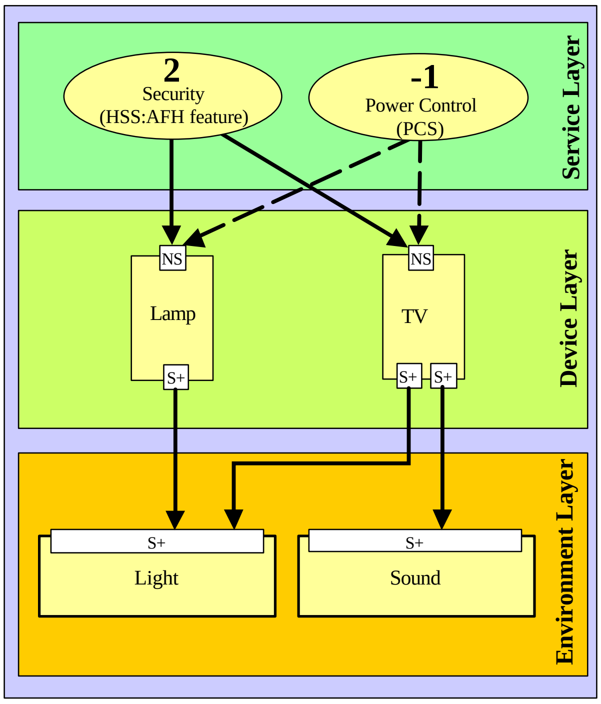


Figure 7.2: Interaction avoided between HSS:AFH and PCS

The results from experimentation for this scenario were as expected. The approach was successfully able to avoid this interaction.

7.1.3 Scenario 3: HSS:AFH vs HVAC

This negative interaction is from section 4.3.3. In this scenario, the away from home feature of the security service has to follow a pattern to make it look as if the owners are home. This includes turning on lights and closing the blinds. The away from home feature of the security service has turned a lamp on and closed the blinds. However, the climate control service notes that the temperature is dropping in the home and wants to increase the room temperature in the cheapest way possible. This can be achieved by opening the blinds.

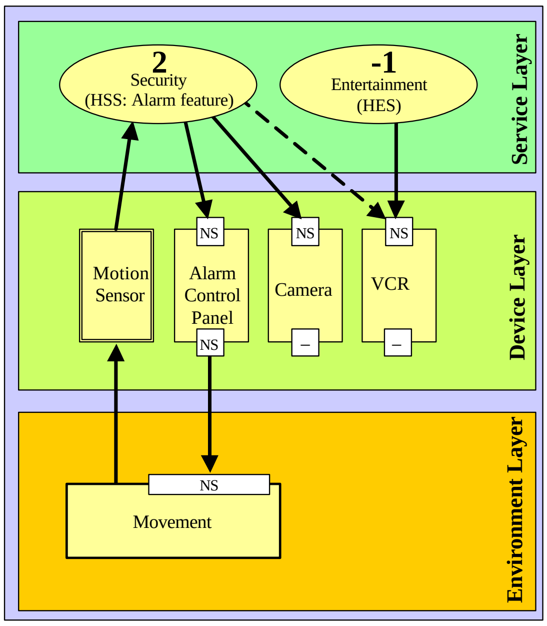
If the blinds are opened, this violates the goal of the away from home feature. However, the climate control tries to open the blinds. But since they are locked (NS), it cannot get access. As the climate control has a priority lower than the security service, it is unable to gain access. The blinds remain closed.

Testing found that this interaction was successfully avoided. If the away from home service was not active, the climate control service could gain access and open the blinds. If this was the case and the away from home feature wanted to close the blinds, it was able to as it had a higher priority. Again, through priorities the negative interaction was avoided.

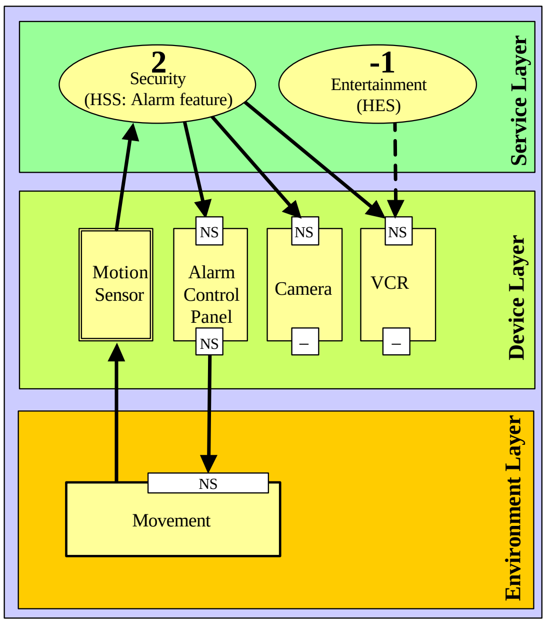
7.1.4 Scenario 4: HVAC vs HSS:Alarm

This negative interaction is from section 4.3.4, and is between the climate control service and the security service alarm feature. The issue here is that when active, the alarm feature interprets all movement as a potential intruder. The climate control is set to cool the home. Realising it is cooler outside, the climate control wants to open the window to allow cool air in. Clearly, this violates the alarm feature by making the home insecure, further, the movement triggers the alarm. This interaction was discussed in detail in section 5.4.2.

During testing, it was found that if the service interaction manager was not active, this did cause an interaction. If the manager was active, the interaction was successfully avoided.



(a) HES using VCR first



(b) HSS:Alarm using VCR first

Figure 7.3: Interaction between HSS:Alarm and HES

7.1.5 Scenario 5: Within HVAC – Issue 1

This negative interaction is from section 4.3.6. The problem here is that the climate control service has been configured incorrectly and can potentially allow both the air conditioner and heater on in the same room simultaneously. Clearly, this is not efficient as while one device heats, the other cools.

Figure 7.5 shows that the climate control service has activated the heater first. Since the heater increases room temperature, the variable is locked with S+. The climate control then tries to turn on the air conditioner. Since the air conditioner device is not in use it gains access. However, before it can be turned on, it must access the environmental variable. For the air conditioner this is room temperature. Since this device wants to lower room temperature, it needs to lock temperature with S–. The room temperature variable is already locked by the heater with S+. As S– and S+ are not compatible, the air conditioner is unable to gain access and the interaction is avoided.

During testing it was found that this interaction was an issue. If both the heater and air conditioner were active, the room temperature remained constant, because as one heated, the other cooled. This meant both devices would be active indefinitely. When the service interaction manager was active, the interaction was successfully detected and avoided.

7.1.6 Scenario 6: Within HVAC – Issue 2

This is a negative interaction from section 4.3.6. This interaction is similar to the interaction above, where the air conditioner tries to turn on but cannot because the heater is in use. However, depending on the settings, it is possible that when the heater stops, the variables are released. This allows the air conditioner to start and lower the room temperature.

If the maximum and minimum temperature settings have been entered incorrectly, then it is possible that the air conditioner lowers the room temperature to one below the minimum heating temperature. When the air conditioner stops, the heater starts, and the loop starts again.

Unfortunately, this is one type of interaction which this approach cannot detect. This is because after the heater, or air conditioner, has completed its task, it releases locks on all its variables. These variables are then free to be locked by any other device. This is one limitation of the approach which will be discussed further in Chapter 8.

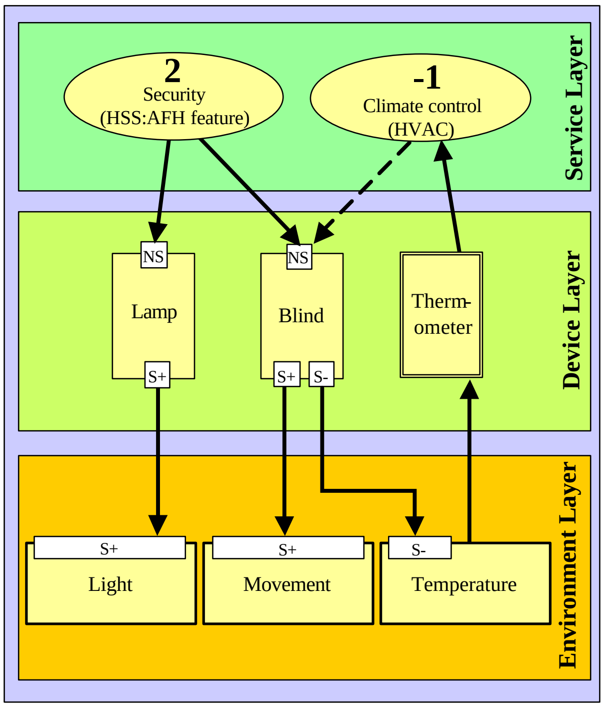


Figure 7.4: Interaction avoided between AFH and HVAC

7.1.7 Scenario 7: within HSS

This negative interaction is from section 4.3.7, and is an interaction within the home security service. This occurs when the alarm feature is monitoring the home for intruders and the away from home feature wants to lower the blinds. When the blinds are lowered this creates movement triggering the alarm.

However, this interaction can be avoided using the approach. In this example, the alarm is armed and the movement variable is locked with NS, which means it cannot be changed. When the away from home feature tries to lower the blinds, it is able to get access to the blind device. However, this device requires two environmental variables: room temperature (as lowered blinds can cool the home) and the movement (as the blinds lower this causes movement).

Figure 7.6 shows the device is able to get access to the room temperature variable and sets it with S–. However, when the device tries to set the movement variable with S+, it is rejected as S+ and NS are not compatible. Therefore, the blinds cannot be lowered and the interaction is avoided. Although the security service has a high priority, it cannot override itself. If it *had* to lower the blinds, it would have had to explicitly remove the lock from the movement variable (by turning the alarm off). The blinds would then be free to be lowered.

During testing, the results were consistent with the theory. If the service interaction manager was disabled, when the blinds were lowered this caused the alarm to be triggered. However, if the service interaction manager was active, the interaction was avoided.

7.1.8 Scenario 8: within HVAC

The first seven scenarios have shown how the approach is able to avoid negative interactions. Just as important, the approach must allow devices to cooperate and work together to achieve a common goal.

This example shows how two heaters can operate together to heat the room quickly. Assume the HVAC service has to heat the home as quickly as possible.

When the service starts it finds all heating devices. The service finds two heaters. If the service wants to control them, access must be granted. Since no other service is using the devices, access is granted. As the heater heats a room, the heater has to lock the environment variable, temperature, with S+. The first heater is able to lock with this value successfully. When the second heater tries to lock with S+, the

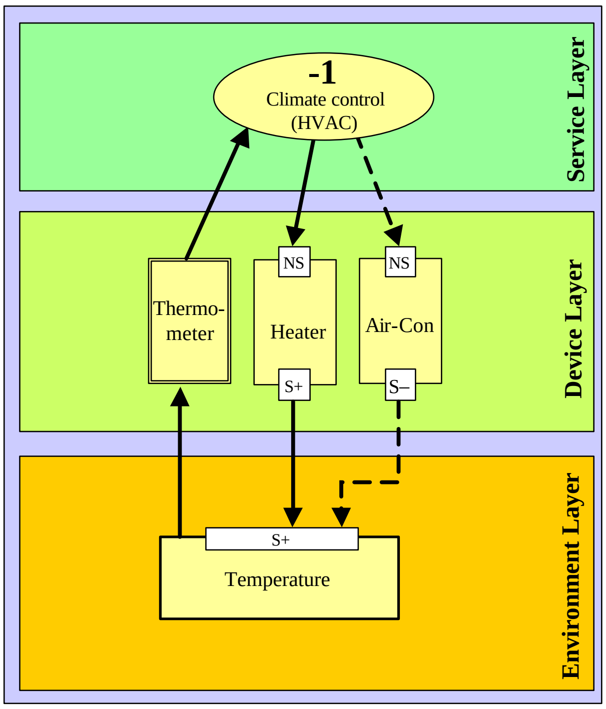


Figure 7.5: Interaction within HVAC

variable is already in use. However, this is allowed because its value of S+ is compatible with the locked value of S+. As no interactions were detected both devices are able to operate, shown in Figure 7.7.

This was tried within the test-bed and worked successfully. The result was the same when the feature interaction manager was enabled and when it was disabled. This is one, simple example which shows that devices can work together.

7.1.9 Scenario 9: HES and HVAC

A second interaction which shows the positive interworking of devices is between the home entertainment service and the climate control service.

Assume the owner is at home watching a movie through the entertainment service. As it is a hot day, the climate control is using the air conditioner to keep the home cool.

As the owner is watching the television, the glare on the screen is irritating, so the entertainment service tries to close the blinds.

As Figure 7.8 shows, the climate control service is using the air conditioner. Since the air conditioner is active, it is cooling the room and has locked the temperature variable with S–.

The entertainment service is using the television which affects light and sound. The entertainment service wants to close the blind. Access is granted to the device as it is not in use. When the blind closes, it causes movement, further it will help cool the room. As the movement variable is not in use, it is able to lock it with S+. The temperature variable is in use. However, as the blinds cool, the S– should be used. Since the temperature variable is already locked S–, this lock is allowed as S– and S– are compatible.

Therefore, the blind is allowed to close. This means that the air conditioner is cooling the room, and the blinds are helping keep the room cool as well as stopping the sunlight glare on the television screen.

This scenario was tested in the test-bed with the feature interaction manager, where it succeeded. However, it is worth noting that it worked in the same way when the manager was disabled.

Although this scenario worked with and without the manager, it shows that the manager is able to avoid negative interactions while allowing positive interactions to happen unhindered.

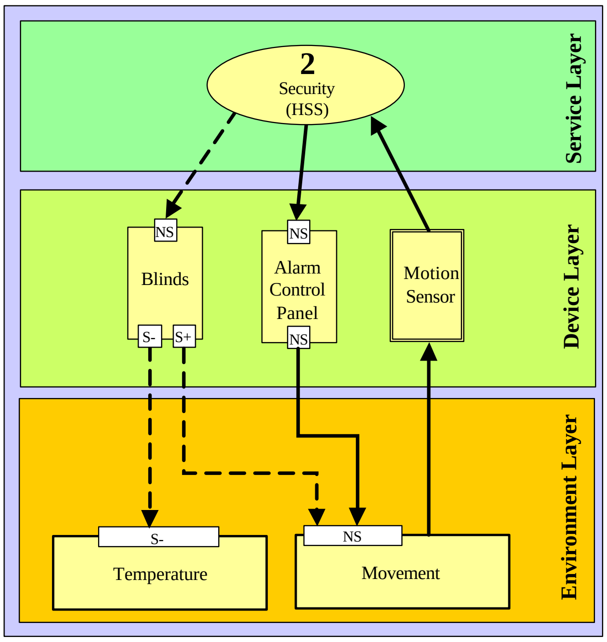


Figure 7.6: Interaction within HSS

7.1.10 Scenario 10: CMSS vs HSS

The Carbon Monoxide Safety Service (CMSS) monitors carbon monoxide levels in the home. If these reach a dangerous level, the service will open windows to let fresh air in and alert the owner (perhaps by sending a message or ringing a bell). An interaction could occur between this service and the security service.

If the CMSS tries to open a window while the security service is armed, the security service may try to close the window, since the aim of the security service is to keep the home secure. Carbon monoxide fumes should not be kept in the home and the window should be allowed to stay open.

The approach presented here does avoid this interaction. The security service has been set with a priority of 2 and the CMSS has been set with priority 3. Assume the security service is armed and the CMSS is monitoring the home. Suppose levels of carbon monoxide exceed a safe limit, the CMSS must alert the ower and open a window.

Since there are no services using the window, the CMSS is able to get access to it. However, to open the window the CMSS has to get access to the movement variable, which has been locked by the security service. However, since the CMSS has a higher priority than security service, CMSS is able to get access to the variable, shown Figure 7.9.

The window can be opened, and will stay open until the CMSS closes it. The security service may try to close the window, but since it has a lower priority, access will be denied.

An argument can be made that the window should stay closed if the alarm is armed. This is because there may be no one in the home and the home should remain secure. However, what if the owner has pets inside? If the owner did want the home to remain secure, they could adjust the priorities and ensure security has a higher priority than CMSS. If this were the case the window would not be opened (keeping the home secure), however other functions of the service would still work, like alerting the owner to the problem.

A similar interaction may occur between the climate control service and CMSS. The climate control service may be trying to heat the home (using heaters). By opening a window this would let cold air in. The climate control service may then try to close the window, which would be undesirable. Whereas the security service is likely to be active when the owners are away, there is a more likely chance that the climate control service would be active when the owners are in the home.

The conflict is the same – the CMSS wants to open the window, whereas the other services (climate control or security) want to close it.

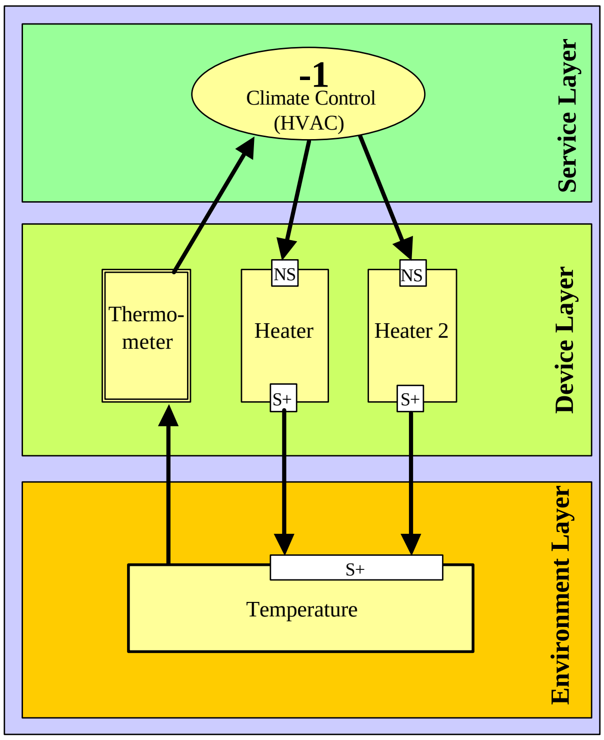


Figure 7.7: Interaction within HVAC

7.1.11 Scenario 11: HCS and HVAC

The Humidity Control Service (HCS) aims to keep the humidity levels of a room (or indeed a home) at a comfortable level. The service uses three devices: a humidistat to ascertain humidity in the room, a dehumidifier and a humidifier.

In previous examples the heater device has only affected room temperature, however it could be argued that the heater affects humidity too. In this particular scenario, let us assume that the heater's side effect on humidity should be captured in the model. Therefore, as well as affecting the room temperature variable, the heater should be shown to affect the humidity variable.

In this case, an interaction would occur between the climate control service and the humidity control service. If the climate control service is using a heater, it will heat a room as well as cause humidity to decrease. If the humidity control service is also active it may be wanting to increase humidity. Clearly there is potentially an issue between the heater and humidifier through the humidity variable.

Technically there is an interaction through the humidity variable. However, it is likely that a home owner may want both the heater and humidifier on at the same time, in the middle of winter for example.

Figure 7.10 shows what would happen if the climate control service was active first and heating the room. The climate control service would have access to the heater which would cause room temperature to increase and the humidity in the room to decrease.

If the humidity in the room drops below a certain level, the HCS will try to increase it by turning the humidifier on. The HCS will gain access to the humidifier device, but when it tries to get access to the humidity variable and lock it with S+, access will be denied as the variable is already locked with S-.

In this instance the humidifier would not be able to turn on. This would not be the expected outcome for the user as they want both devices on at the same time. This example opens the discussion on how side effects should be treated. If the heater's side effect on humidity was ignored (as it currently is in this approach), the interaction would not be detected. However, although this is the correct result, we are getting it because the model is incomplete. However if it is included we get a false positive. Therefore, side effects can not be treated in the same way as primary effects. This issue will be discussed further in Chapter 8.

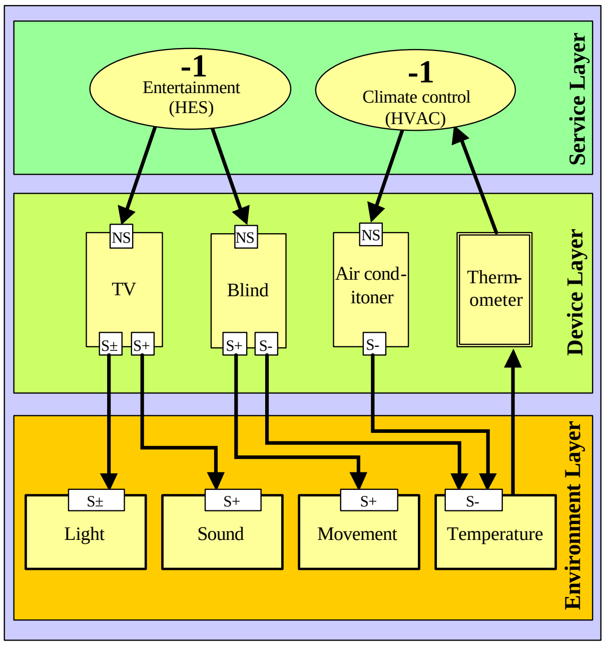


Figure 7.8: Interaction between HES and HVAC

7.2 Summary of results

The results from section 7.1 have been summarised in Table 7.1. As the table shows, the approach was successful. However, the approach was not able to avoid the looping interaction.

Scenario	*-/+ Interaction*	*Interaction description*	*Avoided by approach*
Scenario 1	−ve	Security (Away from home feature) with Power control service	√
Scenario 2	−ve	Security (Alarm feature) with Entertainment service	√
Scenario 3	−ve	Security (Away from home feature) with Climate control service	√
Scenario 4	−ve	Climate control service with Security (Alarm feature)	√
Scenario 5	−ve	Within HVAC (issue 1): wasting energy	√
Scenario 6	−ve	Within HVAC (issue 2): Looping	×
Scenario 7	−ve	Within Security (between away from home and alarm features)	√
Scenario 8	+ve	Within HVAC	√
Scenario 9	+ve	Entertainment service and Climate control	√
Scenario 10	−ve	Carbon monoxide safety service and security	√
Scenario 11	−ve	Humidity control and Climate control	×

Table 7.1: Summary of results

Table 7.2 shows the types of interactions which the approach avoids. Kolberg et al. identified four types of interaction, so looping has been included here for completeness. The approach is not able to handle the looping interaction type because the locked variables are released, and are then free to be locked by another device. The approach does not check the new lock with the previous lock. Indeed, it may be valid for a device which previously held the lock to hold the lock again.

Interaction type	*Handled by approach*
Multiple action interaction (MAI)	√
Sequential action interaction (SAI)	√
Looping (Special case of SAI)	×
Shared trigger interaction (STI)	√
Missed trigger interaction (MTI)	√

Table 7.2: Interaction types handled by the approach

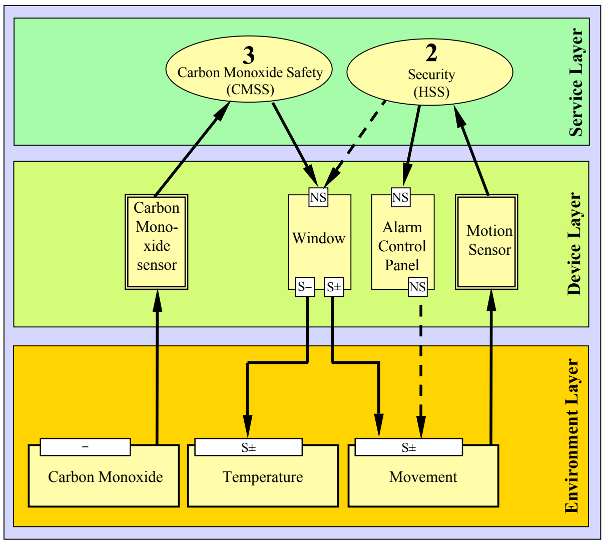


Figure 7.9: Interaction between CMSS and HSS

7.3 Support for multiple rooms

Although all testing was carried out in one room in the home, some testing was done by placing different devices throughout the home.

For these tests, it was assumed that rooms would not affect one another. It is difficult to ascertain how rooms would affect one another because this will depend on the type of doors, window types and the type of walls. This issue will be discussed further in the next chapter.

However, to show that the approach did work for different rooms, a heater and an air conditioner were placed in separate rooms. When the command was issued, they both started. The heater was able to lock the room temperature variable in its room, and similarly the air conditioner was able to lock the temperature variable in its room. Since rooms do not have an impact on one another, there was no problem in setting the locks.

Additional tests were carried out using the alarm service and setting it to protect all rooms in the home. When the alarm was armed, no device which created movement could be turned on as the movement variables in each room was locked. These tests were extended to alarm certain rooms. In a room which was not *alarmed*, devices which created movement were allowed to operate.

7.4 Summary

This chapter has shown how the approach was used to avoid negative interactions as depicted in Chapter 4. Experimentation also showed how positive interactions were allowed by the approach.

If the service interaction manager were not activated, negative interactions could happen. However, when the manager was active, all but two of the interactions was detected successfully and avoided. Scenario 11 was the only false positive to be generated by the manager. It suggests that we need to differentiate between a primary effect and side (secondary) effect.

Overall, the implementation and testing of this approach was a success, the results in Table 7.1 show this. Furthermore, the four types of interaction defined by Kolberg et al. are avoided by the approach. However, the approach was not able to detect looping interactions.

There are some limitations of the approach and some further work which could be carried out. These issues will be discussed in the next chapter.

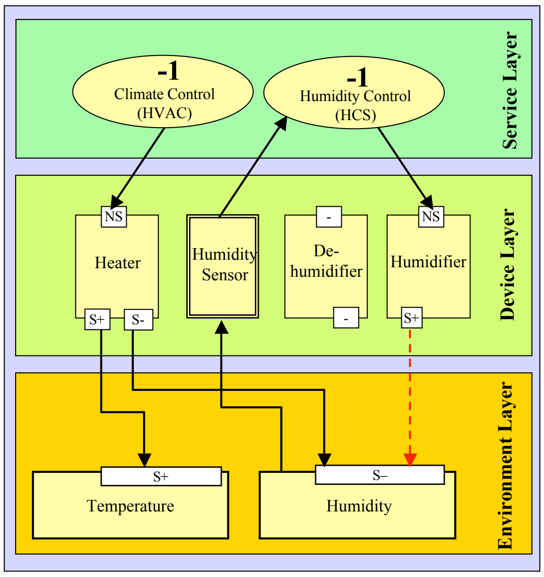


Figure 7.10: Interaction between HCS and HVAC

Chapter 8

Conclusions and Further Work

This thesis has presented a novel approach to the service interaction problem in home networks. In this chapter the achievements of the presented approach will be discussed, followed by how well it met the aims stated in Chapter 1. Afterwards, limitations of the approach will be discussed and how they can be resolved by further work.

8.1 Achievements of the approach

This thesis presented a novel approach to the service interaction problem in the home. There are a small number of approaches available for service interaction problem in the home domain. However all of these are off-line. This means that they are unable to accommodate the rapid change in the home with devices and services leaving and joining the network. They are also unable to cope with a multi-vendor environment. Off-line approaches are not suitable for this kind of environment. Therefore, an on-line approach is the only viable approach for the home domain.

In the telephony domain, where most feature interaction work has been carried out, on-line approaches are relatively rare. This thesis has presented a new on-line approach for the home network.

The approach is device centric as it focuses on the device and its surrounding environment. This differs from all previous feature interaction approaches which focus on the service.

The environment is a factor which has not been taken into consideration in feature interaction work before. The main reason for this is that there is no obvious environment (like there is in a home) in the telephony domain. However looking at the *environment* in a different way may prove useful in telephony.

The use of the environment is crucial in the home domain for detecting interactions as some occur through it. Furthermore, interactions also occur at the device level too. This justified including both device and environment in this approach.

By combining the flexibility of on-line approaches and the extra information which is obtained by including the environment, a successful solution to the service interaction problem in home networks was developed. The approach has been shown to avoid negative interactions while allowing positive interactions at runtime.

The approach achieves this by using a service interaction manager. The manager keeps an internal representation of the home with all devices and services. The internal representation is automatically created at runtime by the manager making use of a remote device database. This means there is very little user intervention is required.

One of the main problems with using an on-line approach is scalability. This may be an issue in telephony where systems are extremely large, complex and distributed; in the home it is not such a problem. In the home all devices and services are centralised by using the residential gateway. Also, the number of services and devices in a home is limited.

8.2 Strengths of the approach

The key strength of this approach is that it is online and uses a manager which requires no *warm-up* period. This means it is ready to be deployed directly into the home. The manager does not need to know of the services in the home. Further, the manager is able to determine the location of many of the devices in the home. For unrecognised devices, an straightforward web interface is available to input details.

Other strengths of the approach are covered by the requirements (which were met) listed in section 1.2. These are:

1. **Avoid negative interactions in a home network.**
   The approach successfully avoided negative interactions in the home. This is achieved by controlling access to devices and the environment. Through preliminary experimentation it was found that a way of prioritising services was required. Therefore, priorities were introduced. This allowed safety services to override less important services.

2. **Consider the environment**
   The environment was successfully used to avoid interactions, by characterising it through variables. These variables included temperature, lighting and movement. Access to the variables was controlled by using locks. It was found that using either *locked* or *unlocked* was not adequate as this did not allow devices or services to work together. Therefore, a refined locking technique was developed which allowed devices with the same goal to lock with the same lock.

3. **Manage new devices and services joining the home network as well as existing devices and services leaving**
   The approach presented is able to listen for devices joining and leaving. If a device joins the network the approach is able to automatically determine the device type and then the variables it affects using the remote device database. This worked as expected. However, there is the issue of who hosts the device and keeps it up to date; this is discussed below.

4. **Handle services from multiple service providers**
   Since the approach concentrates on the device, rather than the service, this aim has been achieved. By using the device and its surrounding environment, this has meant there has been no need to analyse services.

5. **Limited user intervention**
   Since the approach presented operates automatically, then there is very little user intervention required. If the manager cannot determine the type of device, a user may have to tell the manager the type. However, if a device is added to the framework, it will need to give enough of a description of itself so other services can search and find it.

As well as satisfying the aims set out in Chapter 1, the approach also avoids almost all the types of interactions identified by Kolberg et al.

After experimentation, a second use was found for the manager. The manager can be used by services to find a device to achieve a particular aim. For example, if a service has to cool a room, then the service can query the manager to find devices which affect room temperature. This can also be used if a service requires a device in a particular room. Although these possibilities were not explored fully in this work, the worked detailed in [118] does use the manager to find the required devices.

8.3 Limitations of the approach

Although the approach presented here has many advantages and does go some way to solving the service interaction in the home, it does have some limitations.

Device database
One of the main limitations of this approach is the device database. There is an issue of who keeps this database up to date and who populates it. If this approach is to work it is vital that this database is kept consistent. If devices are entered incorrectly into the database then the manager will not operate correctly.

Devices must be in the framework
The approach assumes that all devices and services are registered within the residential gateway. Although this will be the case for most devices, there is a possibility of device-to-device control and control

which bypasses the gateway. Although there are techniques available to solve this issue in IP based networks (discussed in the further work section), there is no known workaround for X.10. This means interactions can occur if devices (or services on devices) start directly controlling other devices.

Issue of looping interactions

Table 7.2 shows that the approach is able to avoid most types of interactions, except looping. Due to the way in which this approach works, looping is one type which can not be detected. This is because the locks are released on a device (or the environment). When these locks are released, the device is free to be used by another service. For example, where there is a loop with climate control service: when the heater turns off, then the air conditioner comes on, then when the air conditioner is turned off the heater comes back on. This is an endless loop. However, since the room temperature variable is released when either the heater or air conditioner is turned off, there is no restriction on who can lock it next. In this case, since the interaction is within the climate control service, this interaction should be detected when the service is designed.

8.4 Further work

The approach presented does help ease the service interaction problem in the home. However, there is further work that could be carried out.

Side effects of devices

In the context of this work, a side effect from a device can be defined as any effect other than an intended primary effect. Scenario 11 in section section 7.1.11 highlights the shortcomings of not including side effects in this approach.

If side effects are to be included, it is not enough simply treating them in the same way as primary effects. This can result in the approach producing false negatives. Therefore, further work is required to investigate how to best capture side effects. Preliminary work has been carried out to determine how they can be handled. One approach is to include a flag called SE (side effect) which is only used for side effects. This flag is placed on an environment variable by a device and can have the value: SE+ (when the side effect increases), SE– (when the side effect decreases or SE± (when there is change, but it is unknown). Like the locks, SE must be placed on a variable before the device is allowed to run.

The SE flag would not lock an environment variable, rather it would be used for information. A device can place the SE flag on any environment variable, provided the variable is not already locked with NS. If a variable is locked with S+, S– or S±, SE can be added to the variable.

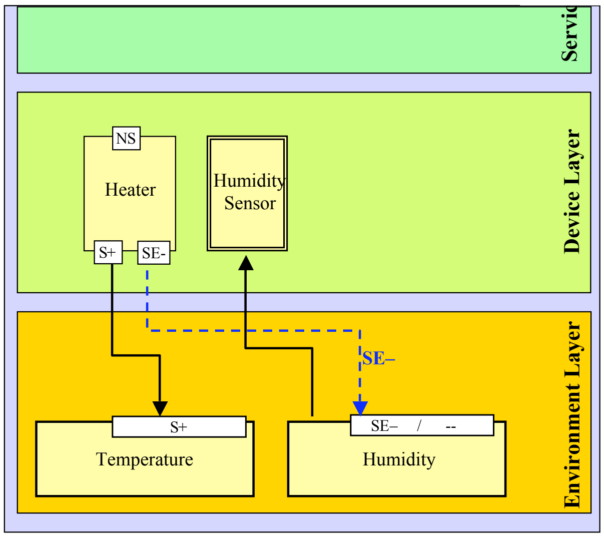


Figure 8.1: Heater's relationship with the environment when active

Using a heater as an example, when it is heating a room it will be increasing room temperature (primary effect), however will also be reducing humidity (side effect). Therefore, when the heater is switched on, it will lock room temperature with S+ (as it wants to increase temperature) and will place SE– on the humidity variable, as a side effect of the heater running is reduction in humidity. This is shown in Figure 8.1.

This is a more accurate and complete view of the device and how it affects its environment. In the previous chapter, scenario 11 (section 7.1.11) resulted in a false positive. Using the SE flag, the approach produces the desired outcome. Figure 8.2 shows the updated view.

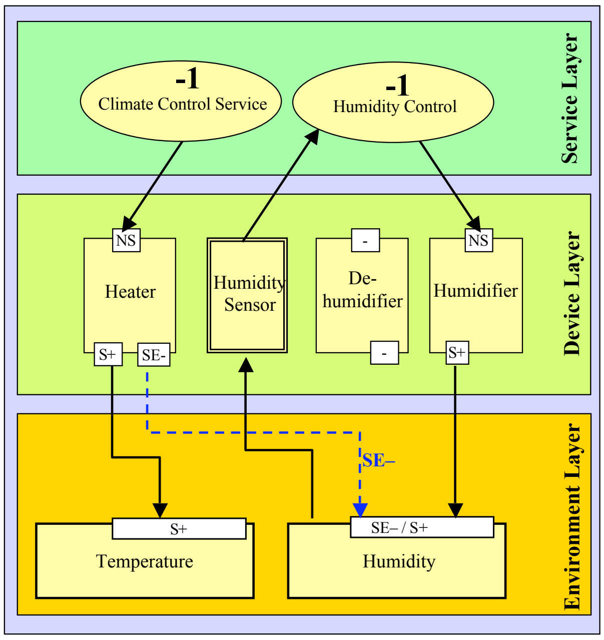


Figure 8.2: Interaction between climate control service and humidity service

In scenario 11, when the heater was active it locked both the temperature and humidity variables (Figure 7.10). Therefore, the humidifier was unable to turn on because it could not get access to the humidity environment variable. Now, with the introduction of SE, the heater locks the temperature variable, but sets the SE flag on the humidity variable to show that the heater is affecting it. When the humidifier is turned on, it is able to get access to the variable and lock it with S+.

Using the SE flag, if a previous scenario is taken, the approach still works. Consider the interaction between climate control and the security service, where the climate control wants to open a window while the alarm is armed.

This interaction is described in scenario 4 (section 7.1.4). Rather than having movement as a primary effect of window, it could be considered to be a side effect. Figure 8.3 show the updated view – the window having a primary effect of changing room temperature and a side effect which affects movement. Since the movement variable is locked with NS, the window device is unable to get access to it and place the SE flag on it, thus avoiding the interaction.

Initial investigations into using the SE flag look promising, however further work and implementation would have to be carried out before its value can be properly seen. Also, deciding what the primary effect of a device is and what the side effect is. Depending on the circumstance, a side effect of a device may be considered by some users to be primary effect. This issue requires further work.

Advanced relationships

Another issue to be investigated is to determine whether there are relations between environment variables. Currently, this is not considered in this approach. However, rather than looking at the relationship between variables, a more comprehensive study of how devices affect their environment and the inclusion of side effects may be of more use.

Consider the humidity variable: when room temperature increases humidity may change. The change in humidity will depend on *how* the room temperature is being increased. If room temperature is being increased by a heater, humidity will decrease. If room temperature is being increased by the outside air, humidity may increase. Therefore, it is not the temperature variable which is causing the change in

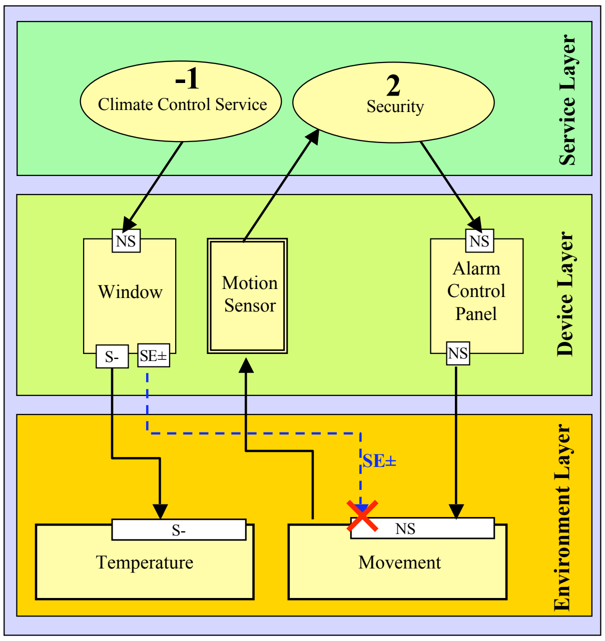


Figure 8.3: Interaction between climate control and security using SE

humidity, it is the device which is causing both room temperature and humidity to change.

Further work to investigate whether this is the case for all variables, or whether indeed there are some relationships between variables would be valuable.

As well as investigating a further relationship between environmental variables, it may also be worth investigating the relationship between rooms. This approach assumes there is no relationship between rooms and each room is a separate entity. If a relationship was created between rooms it could potentially be hard to define. Factors such as thickness of walls or whether the door was open or closed would have to be included. Whether including this to build a complete relationship between rooms would add extra value to the approach is unknown. This is also an issue which is worthy of further research.

Device database

Currently there is no formal documentation detailing how a device should appear in the device database. A formal specification should be created to define a device and state how one decides how a device affects the environment. For this work we have not found it necessary. However, if this approach were to be deployed in reality then this would need to be carried out.

There is also the issue of who maintains the device database. It is not practical for all homes to have their own database with all devices, as this would be too large. Therefore, a service provider may consider hosting the database.

The database would also need to be kept up to date, so when a new device is released onto the market then the new details are entered into the database. It could be the manufacturer of the device who does this or another organisation could take responsibility. Keeping the device database is similar to web services which provide users with music CD track listings.

Granularity of device

Another area in which this approach may benefit from extra research is granularity of device control. The approach detailed here focuses on the action of the device, whether it is on or off. Nakamura et al. argue that this is insufficient and more detail should be included. Therefore, rather than simply on or off, it should be passed a parameter value. For example, a heater may be turned on, but temperature set to twenty five degrees celsius. Through experimentation carried out in this project, it was found that this level of detail was not important. However, it is worth further attention. If the approach was to be improved it should be included as part of the locking. Rather than simply lock with S+, perhaps lock with *Shared* and a value. More work would need to be carried out to determine whether this would be useful. However, this may affect the adaptability of the approach as it becomes specific to one environmental variable. For example, temperature, sound, etc., would each require their specialised unit (degrees celsius, decibels, etc.). This makes it quite specific, rather than having a generic variable object.

8.5 How this approach compares to others for the home domain

Although the approach presented in this thesis does have some limitations and weaknesses, in comparison to other approaches for service interaction in home networks, it does help to provide a good and flexible solution to the problem.

Wu and Schulzrinne [14] present a language called LESS which can be used to avoid interactions at the service creation stage. Their studies only included multimedia services, not the general home services. While this is useful for avoiding intra-service interactions, it can not be used to detect interactions between independent services.

An approach presented by Metzger [23] is an off-line approach which can be used to detect interactions in building control. This approach uses the environment to detect interactions, however it is an off-line approach, so it does not lend itself to an environment which changes (like the home, for example). The approach was later used for feature interaction detection in [92], an environment which may not change once deployed.

Other than the approach presented in this thesis, the only other approach to be specifically aimed at the service interaction problem in the home was by Nakamura [22]. Both the approach here and Nakamura's approach detect interactions at the service level and environment level. The approach in this thesis focuses on the interactions at the device and environment level.

Although Nakamura's approach can be used to detect interactions between different vendors, the services have to be known beforehand. Further, detailed information about the service composition must be known. The approach presented in this thesis does not require any special knowledge of services to detect interactions. For Nakamura's approach to work, not only do the services have to be known, but also their configuration as this may affect the devices they use. Different homes will have different devices, services and configurations. Testing all possible permutations on a home by home basis would not be practical. Further, when any services change, or a new device is introduced, testing would have to be carried out again.

In contrast, the approach presented in this thesis is able to gather the relevant information at runtime about devices and services. A user does have to provide some input to begin with, however this is minimal. With device information, the manager can automatically detect and avoid negative interactions. If a service or device does change, the manager automatically handles the approach and is ready to avoid interactions with these changes. Off-line approaches can not support this functionality.

Nakamura briefly discusses the possibility of converting their approach into an online approach. However, their approach is service centric and requires detailed information of all services. If this information is made available, it is possible their approach may work. However, it is unlikely service vendors will disclose detailed descriptions of their services and methods within the services. However, it is not enough detecting interactions, the interaction must be handled. This is not discussed in their work.

The issue of looping interactions is not discussed by Nakarmura. Like the approach presented here, Nakamura looks at messages from devices to services independently of one another. Since Nakamura's approach does not look at previous instructions from services to devices, they will not be able to detect looping interactions.

Although Nakamura's approach is able to detect interactions, the detection is carried out off-line. This has the major drawback that testing needs to be carried out everytime a new service, or even device is added to the network. Although homes may have similar devices and services, the configurations may vary considerably. The approach presented in this thesis is an online manager. It used at runtime and is independent of services meaning it will work with different service vendors. Services require no knowledge of the manager for this approach to work. Also, the fact that the manager can change (by having new protocol converters added) does mean it can extended as time progresses and protocols change.

8.6 Summary

This thesis has presented a new approach to the service interaction problem in home networking. The approach is novel in that it uses the environment to avoid negative interactions. Inspiration has been drawn from the operating systems domain to develop a locking technique. With the adapted locking technique, access to devices and environmental variables is controlled. By carefully controlling access to devices and the environment, negative interactions are avoided while positive interactions are allowed.

Using a testbed, the approach has been shown to work. The approach has successfully fulfilled the aims laid out at the start of the thesis. Although the approach has worked well, there are some improvements which could be made to make this new approach even stronger.

References

[1] M. Kolberg, E. Magill, and M. Wilson. Compatibility issues between services supporting networked appliances. *IEEE Communications Magazine*, 41(11), 2003.

[2] LetsAutomate.co.uk. `http://www.letsautomate.co.uk/`, viewed: 18/05/2004.

[3] OSGi: The Open Services Gateway Initiative. `http://www.osgi.org`.

[4] Th. Zahariadis and K. Pramataris. Multimedia home networks: standards and interfaces. *Computer Standards & Interfaces*, 24(5):425, 2002.

[5] K. Chen and L. Gong. *Programming Open Service Gateways with Java Embedded Server(TM) Technology.* Aiddison-Wesley, 2002.

[6] D. Marples and P. Kriens. The open services gateway initiative: An introductory overview. *IEEE Communications Magazine*, 39(12), December 2001.

[7] e2 Home. `http://www.e2-home.com`, viewed: 12/08/2004.

[8] OnStar at Home Pilot. `http://www.internethomealliance.com-/pilots_projects/family/onstar_at_home/`, viewed: 05/07/2004.

[9] TAHI Connected Home. `http://www.theapplicationhome.com`, viewed: 09/08/2004.

[10] X10 Technology and Resource Forum. `http://www.x10.org`.

[11] IBM Home Director. `http://www.homedirector.com/`, viewed: 12/08/2004.

[12] E. J. Cameron, N. Griffeth, Y.-J. Lin, M. E. Nilson, W. Shnure, and H. Velthuijsen. Towards a Feature Interaction Benchmark for IN and Beyond. *IEEE Communications Magazine*, 31(3):64–69, March 1993.

[13] M. Calder, M. Kolberg, E. H. Magill, and S. Reiff-Marganiec. Feature interaction: A critical review and considered forecast. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 41(1):115–141, 2003.

[14] X. Wu and H. Schulzrinne. Feature interactions in internet telephony end systems. *Department of Computer Science, University of Columbia Technical Report*, January 2004.

[15] H. Velthuijsen, N. Griffeth, and Y.-J. Lin, editors. *International Workshop on Feature Interactions in Telecommunications Software Systems*, December 1992.

[16] L. G. Bouma and H. Velthuijsen, editors. *Feature Interactions in Telecommunications Systems.* IOS Press (Amsterdam), May 1994.

[17] K. E. Cheng and T. Ohta, editors. *Feature Interactions in Telecommunications Systems III.* IOS Press (Amsterdam), October 1995.

[18] P. Dini, R. Boutaba, and L. Logrippo, editors. *Feature Interactions in Telecommunication Networks IV.* IOS Press (Amsterdam), June 1997.

[19] K. Kimbler and L. G. Bouma, editors. *Feature Interactions in Telecommunications and Software Systems V.* IOS Press (Amsterdam), September 1998.

[20] M. Calder and E. Magill, editors. *Feature Interactions in Telecommunications and Software Systems VI.* IOS Press (Amsterdam), May 2000.

[21] D. Amyot and L. Logrippo, editors. *Feature Interactions in Telecommunications and Software Systems VII.* IOS Press (Amsterdam), June 2003.

[22] M. Nakamura, H. Igaki, and K. Matsumoto. Feature interactions in integrated services of networked home appliances. In *[68]*, pages 236–251, June 2005.

[23] A. Metzger and C. Webel. Feature interaction detection in building control systems by means of a formal product model. In *[21]*, pages 105–122, June 2003.

[24] K. Wacks. Home systems standards: Achievements and challenges. *IEEE Communications Magazine*, 40(4), 2002.

[25] P.E. Ross. Managing care through the air. *IEEE Spectrum Magazine (INT)*, page 14, 2004.

[26] T. Tamura, T. Togawa, M. Ogawa, and M. Yoda. Fully automated health monitoring system in the home. *Medical Engineering and Physics*, page 573, 1998.

[27] P. Dinsdale. Broad band. *The Guardian*, 7th, May 2003.

[28] Ark Housing. `http://www.arkhousing.co.uk` viewed: 04/01/05.

[29] Tunstall. `http://www.tunstall.co.uk` viewed: 04/01/05.

[30] Rehab Tool. `http://www.rehabtool.com` viewed: 04/01/05.

[31] Hogar.es. `http://www.hogardigital.net`.

[32] B. Sridharan, A.P. Mathur, and S.G. Ungar. Digital device manuals for the management of connectedspaces. *IEEE Communications Magazine*, 40(8), 2002.

[33] S. Moyer, D. Marples, and S. Tsang. A protocol for wide area, secure networked appliances communication. *IEEE Communications Magazine*, 38(10), October 2001.

[34] S. Moyer, D. Marples, S. Tsang, J. Katz, P. Gurung, T. Cheng, A. Dutta, H. Schulzrinne, and A. Roychowdhury. *Framework Draft for Networked Appliances using the Session Initiation Protocol.* IETF Internet Draft draft-moyer-sip-appliances-framework-02.txt, June 2001. work in progress.

[35] B. Rose. Home networks: A standards perspective. *IEEE Communications Magazine*, 39(12), 2001.

[36] T. B. Zahariadis. *Home Networking Technologies and Standards.* Artech House, 2003.

[37] UPnP: Universal Plug and Play Forum. `http://www.upnp.org`.

[38] HAVi: Home Audio Video Interoperability. `http://www.havi.org`.

[39] PowerHome Home Automation Software for X.10. `http://www.myx10.com/`, viewed: 31/08/2004.

[40] Jesse Peterson X.10 API. `http://www.jpeterson.com/rnd/x10`, viewed: 10/01/2002.

[41] W3C recommendation Extensible Markup Language. `http://www.w3c.org/XML` viewed: 10/9/04.

[42] SSDP: Simple Service Discovery Protocol. `http://www.upnp.org/download-/draft_cai_ssdp_v1_03.txt`.

[43] SOAP: Simple Object Access Protocol SOAP. `http://www.w3.org/TR/soap`.

[44] B.A. Miller, T. Nixon, C. Tai, and M.D. Wood. Home networking with universal plug and play. *IEEE Communications Magazine*, 39(12), 2001.

[45] Microsoft Corporation. Universal plug and play device architecture. 2000.

[46] B. Manning. *Documenting Special Use IPv4 Address Blocks that have been registered with IANA.* IETF Internet Draft draft-manning-dsua-06.txt, February 2001. work in progress.

[47] D.C. Plummer. *An Ethernet Address Resolution Protocol, RFC 826.* Internet Engineering Task Force, 1982.

[48] E.A. Hall. *Internet Core Protocols: The Definitive Guide.* O'Reilly, 1st edition, 2000.

[49] D. Meyer. *Administratively Scoped IP Multicast, RFC 2365.* Internet Engineering Task Force, 1998.

[50] UPnP Imaging Working Committee Chair S. Albright. Imaging working committee, `http://www.upnp.org/newsletters/newsletter_09_2001/committee.asp` viewed: 15/9/04.

[51] Linksys WRT54G. `http://www.linksys.com/products/product.asp?prid=508&scid=35` viewed: 04/01/05.

[52] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session initiation protocol. *Request for Comments (Standards Track) 3261*, 2002. Internet Engineering Task Force.

[53] JINI Network Technology. `http://www.jini.org`.

[54] Echelon Corporation LonWorks. `http://www.echelon.com/products/lonworks/default.htm` viewed: 15/9/04.

[55] Home plug Powerline Alliance. `http://homeplug.org` viewed: 04/02/05.

[56] Home PNA Network Technology. `http://www.homepna.org`.

[57] IEEE 802.3 Wikipedia. `http://en.wikipedia.org/wiki/IEEE_802.3` viewed: 04/02/05.

[58] IEEE 802.11 Wikipedia. `http://en.wikipedia.org/wiki/IEEE_802.11a` viewed: 04/02/05.

[59] Zigbee Wikipedia. `http://en.wikipedia.org/wiki/IEEE_802.15.4` viewed: 04/02/05.

[60] FireWire Wikipedia. `http://en.wikipedia.org/wiki/IEEE_1394` viewed: 04/02/05.

[61] M. Nakagawa, H. Zhang, and H. Sato. Ubiquitous homelinks based on ieee 1394 and ultra wideband solutions. *IEEE Communications Magazine*, page 74, 2003.

[62] Bluetooth. `http://www.bluetooth.com` viewed: 16/9/04.

[63] The Open Services Gateway Initiative. *OSGi Service Platform, Release 3.* IOS Press, 2003.

[64] The Open Services Gateway Initiative. *About the OSGi Service Platform: Technical Whitepaper.* OSGi Alliance, 2003.

[65] R.S. Hall and H. Cervantes. Challenges in building service-oriented applications for osgi. *IEEE Communications Magazine*, 5(42):144 – 149, 2004.

[66] P. Dobrev, D. Famolari, C. Kurzke, and B.A. Miller. Device and service discovery in home networks with OSGi. *IEEE Communications Magazine*, 40(8), 2002.

[67] T.F. Bowen, F.S. Dworack, C.H. Chow, N.Griffeth, G.E. Herman, and Y.-J Lin. The feature interaction problem in telecommunications systems. In *Seventh International Conference on Software Engineering for Telecommunication Switching Systems*, page 59, 1989.

[68] S. Reiff-Marganiec and M. Ryan, editors. *Eigth International Conference on Feature Interactions in Telecommunications and Software Systems.* IOS Press, 2005.

[69] P. Zave. An experiment in feature engineering. pages 353–377, 2003.

[70] M. Plath and M. Ryan. Plug-and-play features. In *[19]*, pages 150–164, September 1998.

[71] R. Hall. Feature interactions in electronic mail. In *[20]*, pages 67–82, May 2000.

[72] M. Weiss. Feature interactions in web services. In *[21]*, pages 149–156, June 2003.

[73] E. J. Cameron, N. Griffeth, Y.-J. Lin, M. E. Nilson, and W. K. Schnure. A feature interaction benchmark for IN and beyond. In *[16]*, pages 1–23, May 1994.

[74] D. Marples and E. H. Magill. The use of rollback to prevent incorrect operation of features in intelligent network based systems. In *[19]*, pages 115–134, September 1998.

[75] D. Marples. *Detection and Resolution in of Feature Interactions in Telecommunications Systems at Runtime.* PhD Thesis, Communications Division, Department of Electrical and Electronic Engineering, University of Strathclyde, 2000.

[76] M. Kolberg, E. Magill, D. Marples, and S. Tsang. Feature interactions in services for networked appliances. In *IEEE International Conference on Communications (ICC-2002), New York, USA.*, April 2002.

[77] E. H. Magill, K. J. Turner, and D. J. Marples, editors. *Service Provision: Technologies for Next Generation Communications*. John Wiley and Sons, 2004.

[78] S. Reiff-Marganiec. *Runtime Resolution of Feature Interactions in Evolving Telecommunications Systems.* PhD Thesis, University of Glasgow, Glasgow (UK), 2002.

[79] D. Amyot, L. Charfi, N. Corse, T. Gray, L. Logrippo, J. Sincennes, B. Stepien, and T. Ware. Feature description and feature interaction analysis with use case maps and lotos. In *[20]*, pages 274–289, May 2000.

[80] M. Nakamura, T. Kikuno, J. Hassine, and L. Logrippo. Feature interaction filtering with use case maps at requirements stage. In *[20]*, pages 163–178, May 2000.

[81] K. Kimbler and D. Sobirk. Use case driven analysis of feature interactions. In *[16]*, pages 167–177, May 1994.

[82] K. Kimbler, E. Kuisch, and J. Muller. Feature interactions among pan-european services. In *[16]*, pages 73–85, May 1994.

[83] K. Kimbler. Addressing the interaction problem at the enterprise level. In *[18]*, pages 13–22, June 1997.

[84] J. Blom, B. Jonsson, and L. Kempe. Using temporal logic for modular specification of telephone services. In L. G. Bouma and H. Velthuijsen, editors, *[16]*, pages 197–216, May 1994.

[85] A. Felty and K. Namjoshi. Feature specification and automatic conflict detection. In *[20]*, pages 179–192, May 2000.

[86] M. Plath and M. Ryan. The feature construct for SMV: Semantics. In *[20]*, pages 129–144, May 2000.

[87] M. Calder and A. Miller. Using SPIN for feature interaction analysis - a case study. *Proceedings* SPIN *2001. Lecture Notes in Computer Science*, 2057:143–162, 2001.

[88] N. D. Griffeth and H. Velthuijsen. The negotiating agents approach to runtime feature interaction resolution. In *[16]*, pages 217–236, May 1994.

[89] M. Cain. Managing run-time interactions between call processing features. In *IEEE Communications Magazine*, pages 44–50, February 1992.

[90] M. Wilson and E.H. Magill. An environmental model for service interaction in home networks. In *Proceedings of Prep 2004*, 2002.

[91] X. Wu and H. Schulzrinne. Programmable end system services using sip. In *ICC 2003 - IEEE International Conference on Communications*, number 1, pages 789–793, December 1992.

[92] T. Metzger. Feature interactions in embedded control systems. *Computer Networks*, 45:625, 2004.

[93] W.K. Edwards and R.E. Grinter. At home with ubiquitous computing: Seven challenges. In *Proceedings of the 3rd international conference on Ubiquitous Computing*, pages 256–272. Springer, 2001.

[94] P. Durman. Future isn't all bright for orange's high-tech home.

[95] D. Valtchev and I. Frankov. Service gateway architecture for a smart home. *IEEE Communications Magazine*, page 126, 2002.

[96] F. T. H. den Hartog, M. Balm, C. M. de Jong, and J. J. B. Kwaaitaal. Convergence of residential gateway technology. *IEEE Communications Magazine*, page 138, May 2004.

[97] P.D. Smith and G.M. Barnes. *Files & Databases: an introduction.* Addison Wesley, 1987.

[98] A. Silberschatz and P.B. Galvin. *Operating System Concepts.* 5th edition, 1998.

[99] M. Wilson, E.H. Magill, and M. Kolberg. An online approach for the service interaction problem in home networks. In *IEEE Consumer Communications and Networking Conference (CCNC-2005), Las Vegas, USA.*, January 2005.

[100] Intel: Location Aware Computing website. `http://www.intel.com/labs/wireless/lac/`.

[101] I.A. Essa. Ubiquitous sensing for smart and aware environmentss. *IEEE Personal Communications*, 7(5):47, 2000.

[102] B. Horowitz, N. Magnusson, and N. Klack. Telia's service delivery solution for the home. *IEEE Communications Magazine*, 40(4), 2002.

[103] UPnP Software Developer Kits. `http://www.upnp.org/resources/sdks.asp` viewed: 22/06/05.

[104] CyberLink development package for UPnP devices. `http://www.cybergarage.org/net-/upnp/java/index.html` viewed: 22/06/05.

[105] UPnP Basic Device Specification. `http://www.upnp.org/standardizeddcps/basic.asp` viewed: 14/08/05.

[106] SIP Express Router (SER). `http://www.iptel.org/ser/` viewed: 22/06/05.

[107] Microsoft Windows Messenger. `http://www.microsoft.com/windowsxp/using-/windowsmessenger/default.mspx` viewed: 09/08/05.

[108] Sun Microsystems Java Embedded Server. `http://java.sun.com/products-/consumer-embedded/` viewed: 22/06/05.

[109] IBM Service Management Framework (SMF) 3.5.1. `http://www-306.ibm.com/software/wireless/smf`.

[110] Open Service Container Architecture (OSCAR). `http://oscar.objectweb.org` viewed: 24/06/05.

[111] Java Communications API. `http://java.sun.com/products/javacomm/index.jsp` viewed: 09/08/05.

[112] Lycos SMS service. `http://mobile.lycos.co.uk/` viewed: 10/08/05.

[113] National Institute of Standards and Technology (NIST): SIP Stack. `http://dns.antd.nist.gov/proj/iptel/` viewed: 10/08/05.

[114] Java Media Framework (JMF) API. `http://java.sun.com/products/java-media-/jmf/index.jsp` viewed: 10/08/05.

[115] Intel UPnP Stack and Tools. `http://intel.com/cd/ids/developer/asmo-na-/eng/downloads/upnp/index.htm` viewed: 22/06/05.

[116] Siemens UPnP Stack. `http://www.plug-n-play-technologies.com/` viewed: 22/06/05.

[117] mySQL Database. `http://www.mysql.org`.

[118] M. Wilson, E.H. Magill, M. Kolberg, P. Burtwistle, and O. Ohlstenius. Conrolling appliances using pen and paper. In *IEEE Consumer Communications and Networking Conference (CCNC-2005), Las Vegas, USA.*, January 2005.

[119] UPnP Forum: XML Files for Testing. `http://www.upnp.org/standardizeddcps-/documents/XMLFilesforTesting.zip` viewed: 10/9/04.