Center for Pervasive Computing
Report Series

Publication
53

A Programming Language Approach to Safety in Home Networks[1]

Author(s): Kjeld H. Mortensen, Kari S. F. Schougaard, Ulrik P. Schultz
Centre for Pervasive Computing/ISIS
Department of Computer Science, University of Aarhus
Denmark
`{ups,khm,kari}@daimi.au.dk`

Version: 1.0

Status: submitted

Abstract

Home networks and the interconnection of home appliances is a classical theme in pervasive computing research. Security is usually addressed through the use of encryption and authentication, but there is a lack of awareness of safety: preventing the computerized house from harming the inhabitants, even in a worst-case scenario where an unauthorized user gains remote control of the facilities. We address this safety issue at the programming language level by restricting the operations that can be performed on devices according to the physical location of the user initiating the request. Operations that pose a potential safety hazard can only be performed within a physical proximity that ensures the safety of the operation.

We define a conceptual model based on capabilities that define the origin of an action, and use a declarative approach integrated with an IDL language to express location-based restrictions on operations. This model has been implemented in a middleware for home AV devices written in Java, using infrared communication and a FireWire network to implement location awareness.

Contents

1 Introduction

The idea of a computerized, “intelligent” home was first introduced long ago in science fiction litterature, but is only now, with the emerging trend of pervasive (ubiquitous) computing, starting to become reality. The advantages of such a home include easing the household chores of the habitants, providing entertainment, and saving energy by intelligently controlling the house temperature. Through the use of the internet and mobile technology, many functions can even be controlled remotely. However, there is an obvious danger in giving computers control of the home, namely that operations that used to be carried out by a person aware of his environment now are invoked through a computer — the user only has limited, if any, awareness of the consequences of his actions. Moreover, if security is compromised, a “virtual intruder” can gain remote control over all the functions of the home, including (perhaps) the central heating, the lock on the front door, or the cooking stove.

The computerized installations of the home can and should be protected by appropriate security measures, e.g. passwords and encryption. However, passwords alone are unlikely to provide the appropriate level of protection, since residents might unwarily reveal their passwords if using them in public (or loosing the slip of paper where the password had been noted). To avoid compromising the safety of the home, an extra safety layer is needed which ensures that a set of basic safety rules are obeyed when controlling functions in the home from afar. This extra safety layer not only applies to remotely controlling the home, but also to safety-critical functions that are controlled from within the home: turning on the cooking stove in the kitchen from the living room could for example be a potential safety hazard.

We propose that each function of every computerized home appliance by design is classified in terms of the maximal distance at which it can be controlled. For example, safety-critical functions of appliances may only be controllable by people who are present in the same room since they would only then be fully aware of the consequences of their actions. Embedding fixed safety constraints in each device ensures that safety is maintained regardless of the complexity of the system as a whole and despite any accidental misconfiguration by the habitants.

Concretely, we are developing a safety-enabled middleware for home networks. Safety concerns are expressed declaratively as access modifiers annotated on each operation in the software components of the system. At run-time, the middleware verifies that safety constraints are obeyed, in our implementation using a combination of infrared signals and a trusted cabling system (FireWire).

Contributions. The primary contributions of this work are as follows. First, we have defined a declarative, language-level approach to expressing safety concerns controlled by physical location. Second, we have implemented a complete solution with Java language bindings based on infrared communication and a FireWire network, which in addition to stationary devices supports both mobile devices and untrusted devices. Last, our work serves to explore an often-ignored aspect of pervasive computing, namely that of ensuring safety (as opposed to security): *safety* is about avoiding potentially dangerous situations from occurring, whereas *security* is about access control and maintaining integrity.

Background: industrial research project with Bang & Olufsen.

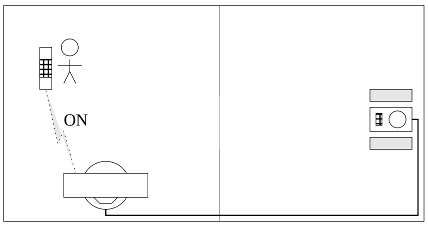
This work has been produced in the context of a research project in collaboration with the Danish AV systems producer Bang & Olufsen (B&O). B&O is a pioneer in home networking for AV devices: B&O products can be connected and share resources using the (proprietary) Masterlink network. This network consists of analog video and audio cables combined with a low-speed, digital serial line which is essentially used for sending remote control commands. Masterlink suffers from a number of limitations: only one audio/video signal can be transmitted on the network at a time, and the low-level serial protocol does not scale to complex interactions between devices.

The goal of the research project is to address the limitations of the Mastlink network, by developing a digital home infrastructure for interconnecting AV devices. The concrete research goal is to connect B&O devices using the IEEE 1394A standard (FireWire, i.Link) and implement appropriate middleware for communication between devices. Although the primary focus is on AV devices and their interaction with other devices such as home computers, telephones, and digital cameras, standard electrical appliances could also be integrated into the network.

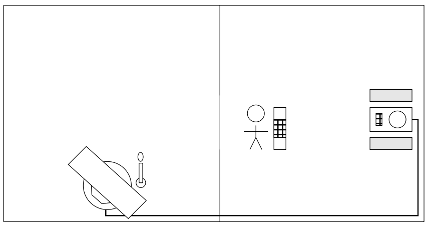
This paper concerns safety issues that can be addressed at the middleware layer of a home network and hence apply globally throughout the entire system.

Overview.

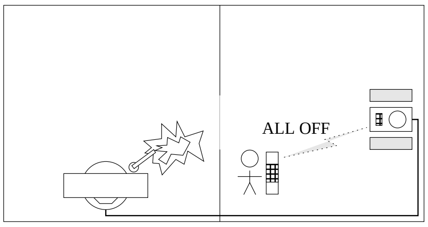
The rest of this paper is organized as follows. First, Section 2 presents two safety-related case studies in the domain of home networking. Then, Section 3 presents the principles behind our solution for ensuring safety, and Section 4 presents our concrete implementation. Last, Section 5 presents related work, and Section 6 concludes.



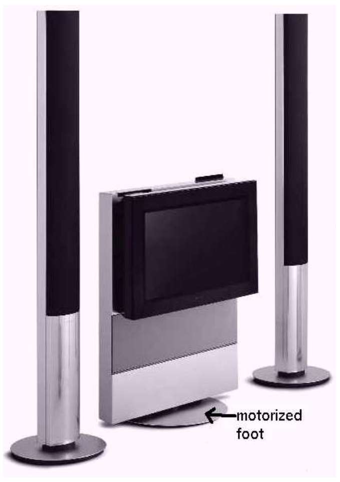
(1) User powers on TV, it turns to default position



(2) User places candlestick and goes into other room



(3) User powers off all devices from other room, television returns to initial position, making candlestick fall over



(a) Bang & Olufsen Avant television with BeoLab 1 speakers

(b) Falling candlestick scenario

Figure 1: Bang & Olufsen scenario: remote operation of television with motorized foot.

2 Case Studies in Home Networking

In this section we will motivate the need for restricting the use of safety-critical operations to only being invoked from specific physical locations.

2.1 Case 1: B&O AV devices

Certain B&O devices are equipped with features that in combination pose a potential (albeit minor) safety risk. B&O televisions are often equipped with a motorized foot, which can turn the entire television 45 degrees to either side; this feature can be used to get a perfect viewing angle or reduce glare from sunlight (see Figure 1(a)). Moreover, as described in the introduction, B&O devices can be connected using the Masterlink network, such that the resources of one device (a satellite tuner or a VCR) can be used from any other device in the house. Through this network, most of the features of each device can be controlled remotely over the network. However, features such as the motorized foot can only be controlled by a device or a remote control present in the same room, such that the user is aware of the consequences of turning the television. If this were not the case, a user controlling the television from a different room could inadvertently cause an accident, for example if a lit candlestick had been placed next to the television such that remotely operating the motorized foot would cause the candlestick to fall over and cause a fire (see Figure 1(b)). The restriction that the motorized foot only can be operated from within the same room is implemented by combining manually configured information about which devices are placed in the same room with information about which device received the infrared remote control command that could cause the foot to move.

The simple restriction on the motorized foot is however non-trivial to implement. First, certain other operations implicitly operate the motorized foot as well. For example, powering on the television causes the motorized foot to turn to the default position (which is configured by the user), and powering the television off causes it to turn back to the neutral position. Any device can be used to turn off all other devices on the network (useful when leaving the house), but the television should only return to neutral position if this command was issued from within the same room. Second, the restrictions are not part of the Masterlink protocol, but are implemented at the application layer of each device. Indeed, there is one known combination of B&O-devices where the aforementioned invariant breaks down due to a programming error in the application layer [2].

2.2 Case 2: Internet-controlled home appliances

Remote control of networked home appliances is by no means specific to B&O. Home network protocols such as X10 [28], UPnP [21], and LonWorks [15] are all intended for controlling home appliances, with applications ranging from controlling the coffee machine through the power lines to controlling the central heating of the house. Networks based on UPnP and LonWorks can be controlled from an IP-based network such as the internet, or through a generic gateway such as OSGI [17] (for example, OSGI can provide internet connectivity for X10-based networks). Thus, in an “appropriately configured” future home, all networked appliances can be controlled remotely over the internet.

Remotely controlling house appliances has numerous useful applications, such as:

- The lights can be turned off remotely if the residents forgot to turn them off before leaving home.
- The VCR can be programmed remotely to record a specific TV show.
- The oven can be turned on remotely to cook the food such that it is ready when the residents return from work.
- The front door (which is monitored by a web cam) can be unlocked remotely to allow a trusted guest to enter the house and e.g. water the plants.
- The house temperature can be reduced while the residents are at work, and turned back up such that the room temperature is comfortable when they return from work.

Safety is of course critical in such a home control scenario: If residents want to turn on the oven to cook some food such that it is ready when they return, it is a potential fire hazard if something unexpected has been left in the oven. Residents turning off the central heating could cause water pipes to break due to frost, for example if they were on holiday and thus unaware of the weather conditions at home. If an intruder was to remotely gain control of the house, the front door could be opened to allow thieves to enter the house.

Security is normally enforced through a protocol-specific encryption scheme. However, a critical question is what kind of remote device can be used to control the house. Suppose a web-based system protected by a password is used. In this case a malicious attacker could easily e.g. monitor the keyboard of the computer that is used to remotely control the house, and obtain the password. A more restricted scheme where a specific device, such as a mobile phone or a PDA, is used to control the house, relies on the physical security of the device, i.e. that it is not stolen, to prevent a malicious attacker performing potentially dangerous remote operations on the house. We consider the safety mechanisms presented to be the last stronghold; even if the security of the system is breached the attacker cannot do anything that compromise the safety.

2.3 Analysis

The key observation in both case studies is that remotely controlling certain home appliances from “too far away” is a potential safety hazard. To ensure the safety of the inhabitants of the house, this remote control needs to be restricted somehow, regardless of what security (as opposed to safety) measures are put in place.

Movement of a part of a device (e.g., ejecting a CD-tray) is only safe when the user is present, as is also the case for operations that can start a fire (e.g., turning on a toaster or an oven). Even though people would want to turn the oven on remotely before they come home from work, they risk an accident in doing so. There are several conceivable operations on a home network which are only safe when the user is present in the room.

Operations such as unlocking the front door or turning off the central heating are a potential hazard to the integrity of the house and are only safe when the user is inside the house (we assume that people who are present in the house are trusted). Similarly, conflicts can arise when multiple users are remotely controlling shared appliances, and overriding the decisions of other users is sometimes necessary; however, the users at home may be aware of circumstances influencing the decision that the remote user does not know about, and hence the remote user should not be able to override the actions of the local users. Again, there are several examples of operations which are not safe unless the user is within the house.

Several operations are always safe to perform, this includes turning off devices (unless they move when they are turned off) and turning up and down the heat (within a normal temperature range). Considering only safety issues these operations may be performed whereever you are, as long as they do not violate other protection in the form of standard security measures such as encryption (which can still applied).

We have thus identified three safety categories for operations on a home network. For the first category the user has to be present to perform the operation safely, for the second category the user has to be in the house, and for the last category no safety considerations apply. The reification of these three location categories as language-level constructs in a middleware for home networking is the subject of the next section.

3 Location-Based Capabilities

We now present the principles behind our approach to safety in home networks, location-based capabilities. We first present a few considerations, and then describe the conceptual model. Then, we define simple language constructs for expressing safety restrictions, and show how to map them to our Java-based implementation which enforces the safety.

3.1 Considerations

We consider a distributed system of devices programmed using software components. The software component model is language-independent, so the interfaces of the components are described using an IDL language similar to CORBA IDL [16]. The operations (methods) implemented by the software components are activated by the users who operate the devices.[2] Executing an operation may involve invoking operations defined by software components located in other devices of the system. Safety-critical operations should only be executed if the user who initiated the operation is located physically close to the device — how close depends on the specification of the operation.

Determining the location of a user cannot in general be done in a completely reliable way, and hence the degree of safety offered by the system is limited by the technology used for determining the user location. This issue is explored in detail in Section 4.

Last, we note that security mechanisms are needed in addition to the safety mechanisms presented in this paper; we expect that standard techniques for security can be used, but consider the introduction of security mechanisms into our system to be future work.

3.2 Conceptual model

Conceptually, our system consists of a number of devices that are organized into hierarchically nested zones. Each device is a member of a zone (and the zones that enclose this zone). The zones are named according to their nesting level, e.g. all zones nested n levels deep from the root have the same name. Restricted operations on devices are marked with the zone level that can call them: to call an operation marked with a given zone name, the call must originate from within the same zone of that name. The *origin* of a call is an operation which has been marked as generating an origin in terms of the calling context (for example, the mapping of user input to action could be such an operation).

When an origin operation is invoked at run-time, the call is annotated with a capability that denotes the physical location of the origin of the call, as illustrated in Figure 2. This capability is propagated to callee operations as designated by the caller. Before executing a restricted operation, the capability is verified by querying the origin device to verify both that the call did indeed originate from it and that the device is currently at a location in the allowed area. How the origin is queried is implementation-specific (and is discussed in Section 4), but it must be done in a

[2]Depending on the scenario, users could allow software agents to act on their behalf. See future work for a discussion of agents.

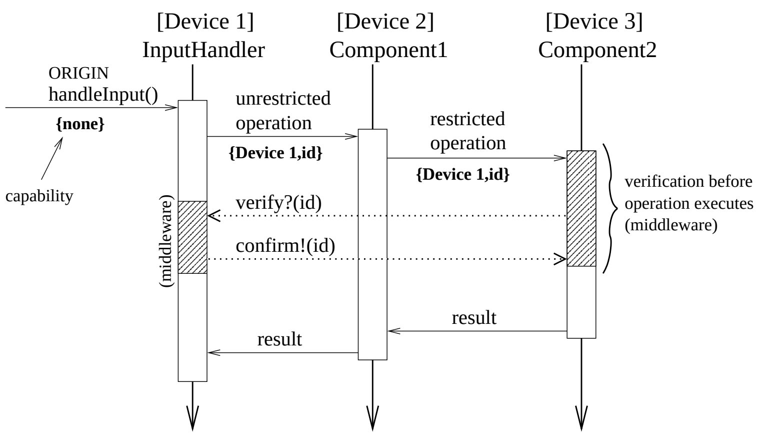


Figure 2: Capabilities and verification in a call sequence

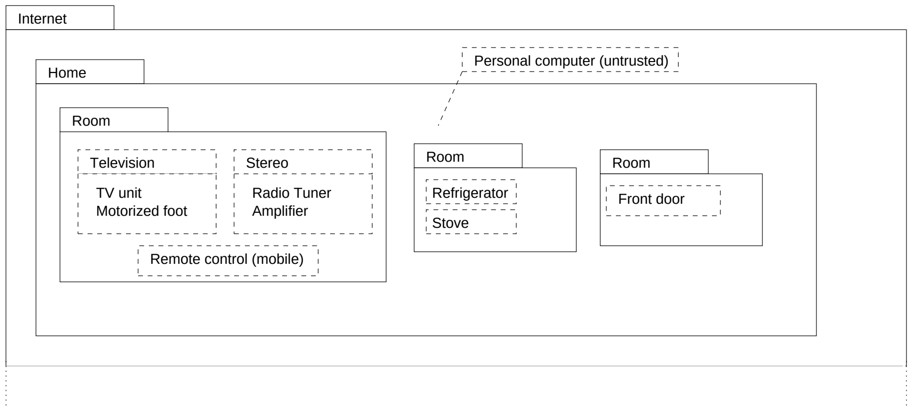


Figure 3: Location zones for the home network scenario

way that does not compromise the safety guarantees provided by the implementation.

Each device of the network is either considered trusted or untrusted. Untrusted devices can only be members of the root zone, and hence cannot invoke restricted operations. Trusted devices are assumed not to lie, so that the verification of the origin call can be trusted.

Home network scenario.

In our home network scenario, we operate with three location zones:

present Close enough that the user is aware of the consequences of his action, which we define to be that the operation was initiated within the same room.

local Close enough that the user can be trusted not to do damage to the home, which we define to be that the operation was initiated within the home.

global Anywhere, which in practice means anywhere on the internet.

The location zones are illustrated with a concrete example in Figure 3. Note that some devices are composed from multiple components, such as a stereo unit made from a tuner, and an amplifier (shown in the example).

Most devices in the home are embedded systems which we consider to be trusted devices. The PC is however considered an untrusted device: it is likely that it could be compromised by a hacker or infected by a virus. Other devices could nonetheless still use the PC to display a control panel, if doing so does not violate the safety of the device.

3.3 Language Constructs

The concept of operations that are limited to being invoked from certain locations gives rise to the issue of determining the location of a caller. As it is a dynamic property, the location of the caller should be checked each time a restricted method is called. Programmers could easily forget to make such verifications, so we provide them with a middleware that automate them.

To enforce restrictions on who should be able to call an operation, we use a combination of a protection modifier for the operations and a specification of the operation call origins. The protection modifier is an annotation indicating the location restriction for the operation resembling modifiers such as private and public known from object-oriented languages. The annotation is incorporated in our IDL, which raises the safety considerations to the interface design level. Thus, safety issues automatically become an important part of the design of the devices which participate in a home network, as they must be considered as part of the interface design.

Methods that handle user input can typically be designated as origin methods. However, since this concerns concrete methods from the implementation, it is not described in the IDL definition, but rather in an implementation-specific way. We return to this issue in Section 3.4.

It turns out that in practice an additional modifier is needed to restrict certain operations to be used only internally in a device constructed from several components. The modifier is named **internal** and operations annotated with this modifier are executable exclusively when originating inside the device. Internal is not a zone enclosed in the present zone. A component of a compound device is always allowed to call internal operations in one of the other components of the device, even if it could not invoke any other kinds of restricted operations.

Home network scenario.

The IDL is extended with one modifier for each zone, and thus operations can be annotated as **present**, **local** or **global** (as most operations are safe to invoke from within the house, *local* could be default). The annotation means the operation is executable only if the call originates in the specified zone.

An example IDL for a television with safety modifiers is shown in Figure 3.3. The Avant television is composed from the main television unit itself, a motorized foot, and a power supply. Almost all the operations in the television have the modifier *global*, as there are no safety issues related to the use of them. The exception is `powerOn` which is limited to local callers because there is a small risk that a device which is turned on can overheat. Besides, there is the possible risk of

```
module com.beo.avant {

  interface Television {
    local void powerOn();
    global void powerOff();
    global boolean getPowerStatus();
    global void mute();
    local void deMute();
  };

  interface MotorizedFoot {
    present void turnToPosition(in short degrees);
    present void turnRelative(in short degrees);
    global short getPosition();
  };

  interface PowerSupply{
    internal void setPowerLevel(in short level);
    global short getPowerLevel();
  };

};
```

Figure 4: Interface for the Avant television in the safety modifier extended IDL

chock or confusion to residents at home if the television is suddenly turned on.[3] Another method to consider is `powerOff` which calls `turnToPosition` in the motorized foot to turn the television into the neutral position; `turnToPosition` can only be invoked by a present caller because the movement is a safety issue, so the success of the call should depend on the location of the caller of `powerOff` (how the origin of the call is determined is discussed in section 3.4). The *present* modifier is not needed on the television's `powerOff` because the operation in the motorized foot defines its safety restrictions itself. `turnRelative` in the motorized is also connected to turning the screen, and thus limited to present users, while `getPosition` is accessible from anywhere and thus has the modifier *global*. For the power supply there is global access to see the power level, but only internal access to set the power level. This is because the power level internally in a device is safety critical, since if more power is produced than consumed the power supply can malfunction.

3.4 Java Language Mapping

Each safety-extended IDL interface is mapped to a Java interface and two Java classes: A public interface for the device, a public proxy class and a dispatcher. Our language mapping is non-intrusive as no language extensions (as for example aspect-oriented programming) are used. Methods in the proxy and the dispatcher wrap the actual method call and are integrated into our middleware. When the generated proxy is used for remote method calls, the details of the network communication and the verification of the capability used to call the method are taken care

[3]Ultimately, balancing safety guarantees with usability must be done by the manufacturer; we here describe how we would design the interface of the television.

of by the middleware. To be able to check that the safety restrictions are obeyed, we require the components to use our middleware. Direct interaction between devices, for example through a TCP/IP connection, would naturally circumvent the safety regulations enforced by the middleware.

All methods generated have as an extra parameter a *safety capability*. The safety capability contains an identification of the caller. The dispatcher takes care of the verification that the caller actually is in the required zone by sending a challenge to the caller, that can only be responded to by a device located in the right area. The safety capability identifying the caller can be passed along in several method calls. If safety capabilities were generated at every call, giving a method a safety restriction does not lead to better safety, so the generation of safety capabilities must be controlled.

If verification fails, a `SafetyException` is thrown, which can be handled by the caller. Depending on the situation, the caller may choose to silently ignore the failure (e.g., the motorized foot does not turn when powering off all devices in the house), or may choose to propagate the error condition all the way up to the user interface level (e.g., by informing the user that the operation was not permitted).

The safety capability designates which method should count as the origin of a call. This ability to act as origin is limited by the origin specification. We use a static verifier to ensure that only the methods listed in the origin specification act as origin of a call. Our middleware uses an implementation specification (IMS) file in which the programmer specifies what methods of the implementation are allowed to act as origin. This enables verification of the generation and use of safety capabilities. IMS is currently very simple: it consists of one reserved word **origin** which should be followed by a method name. The **origin verifier** ensures that only methods pointed out in the origin specification make safety capabilities.

The generated code which takes care of the network communication and the verification of the safety capability is exempted from this restriction.

Home network scenario.

The origin specification for the Avant television (`Avant.ims`) looks like this:

```
origin com.beo.Avant.concreteTelevesion.inputHandler;
origin com.beo.Avant.concreteTelevision.legacyIRremoteHandler;
```

4 Implementation

We now describe our concrete implementation of a home network middleware that supports restricting safety-critical operations. We first survey location awareness technologies with regards to their appropriateness for our home network, and then present our concrete choice of technologies. We then describe the software (middleware) that runs on the home network, and last we perform an experimental verification.

4.1 Location awareness technologies

We now survey location awareness technologies, and evaluate their usefulness with regards to safety in home networks. Specifically, we survey technologies that can realistically be used in a home scenario to determine the location of the target device relative to the origin device, to be able to verify if they can interact via restricted operations. For a detailed survey of location awareness technologies, we refer to Hightower and Borriello [12].

Infrared. Infrared (IR) signals tend to be diffused throughout an entire room, but do not traverse walls, which makes them useful for determining location at room granularity (given an IR receiver/sender in each room) [24, 25, 23, 22]. IR signals can be transmitted through a window or from one room to another using a reflective surface, but requiring two-way IR communication limits the degree to which this can be done. Evaluation: medium degree of safety at room granularity.

Ultrasound. Ultrasound signals can give location information with sub-room precision, down to roughly 10cm, but require an extensive grid of precisely placed, ceiling-mounted sensors [11]. However, each grid is linked to a room, and hence the distance between devices in different rooms cannot be computed. Evaluation: high degree of safety at sub-room granularity.

Bluetooth. Bluetooth uses a wireless communication protocol with a communication range of approximately 10m. Thus, simply detecting whether a device is within range provides location information, but triangulation using multiple devices is also possible [29, 19]. Nonetheless, since it is a wireless medium that uses radio communication, signals can traverse walls. Evaluation: low degree of safety at room granularity.

WaveLAN. Standard IEEE 802.11b wireless networking can provide location information similarly to Bluetooth, very crudely but at long distances in terms of which basestation is used, and more precisely using multiple basestations [1]. Evaluation: low degree of safety at room granularity, medium degree of safety at home granularity.

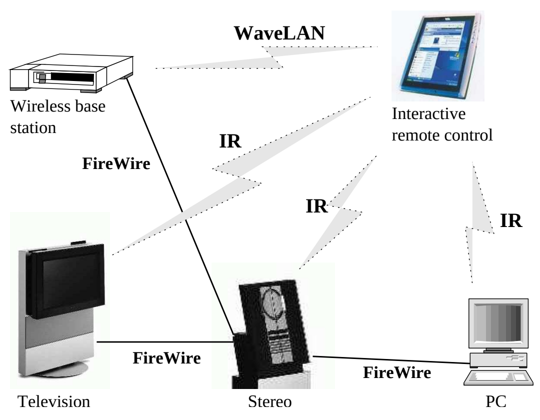


Figure 5: Hardware in the home network (note: currently simulated on PCs)

Local network. A local home network, for example based on IP, FireWire, or other home networking protocols, can trivially be used to verify physical presence (if it is connected to the network, it is in the home). However, untrusted devices that have unrestricted access to the network can pose as other devices, and could therefore constitute a safety hazard. We note that unlike standard ethernet, FireWire devices cannot normally perform packet snooping [14], which facilitates constructing a safe communication mechanism. Evaluation: high degree of safety at home granularity, depending on the network type.

Physical contact. This is a very old-fashioned and secure way of establishing the presence of a person. Evaluation: high degree of safety at room and home granularity.

Efficient communication with a mobile device normally implies using wireless radio communication such as Bluetooth or WaveLAN (IR is not reliable enough as a primary communication media, as it is too easily influenced by other factors, such as sunlight). However, since wireless signals traverse walls, an intruder could easily send wireless signals to devices in the home from the outside. Thus, for the wireless communication to be safe for verifying that safety critical operations can be executed, it must either be secured by using strong encryption (if it is deemed safe enough), or it can simply rely on a secondary means of communication to verify locality, such as IR.

4.2 Home network hardware: IR and FireWire

The concrete setup that we choose for our implementation of a home network is shown in Figure 5. Stationary devices (including PCs) are connected by a home-wide FireWire network which cannot be physically accessed from outside the house (e.g, an external surveillance camera would be connected by other means).[4] Mobile devices use WaveLAN communication to a base station connected to the FireWire network. All devices are equipped with IR senders and receivers based

[4] All devices are currently simulated using stationary and portable PCs; porting the system to an embedded platform is future work.

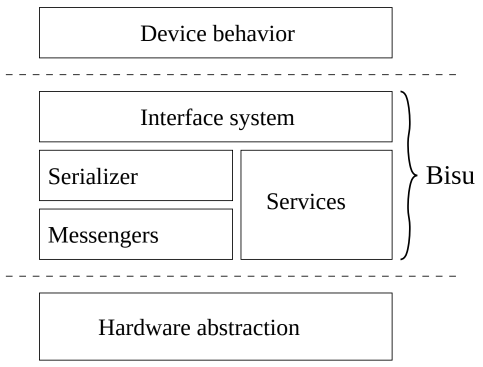


Figure 6: Overview of Bisu architecture

on the IrDA standard. A gateway device bridges the FireWire network to the internet. All devices are trusted except for the PCs; these are equipped with standard OHCI FireWire adapters, which cannot perform packet sniffing.

In this setup, an origin device is *present* with regards to a target device if the devices can communicate over IR. To verify a safety capability, an origin device must connect to the target device using an IrDA connection, which must echo a randomly generated 64-bit integer to verify that it is present.

Similarly, an origin device is *local* with regards to a target device if the devices can communicate over FireWire or they can communicate over IR with trusted devices that can communicate over FireWire. To verify a safety capability between two devices that are connected by FireWire, a randomly generated 64-bit integer must be echoed by the origin device. If the devices are mobile, they attempt to locate a stationary device using IrDA, and communicate the randomly generated integer with the device using IrDA.

In both cases, a physical means of communication which is not radio-based is used. Hence, an intruder must to some degree have physical access to the home to be able to take control over devices in the home (again, assuming security has failed, for example because the intruder is in possession of the system password).

4.3 Home network middleware: Bisu

As described in the introduction, we are developing a middleware for home networks in collaboration with B&O. This middleware, named Bisu[5], constitutes the middle layer of a complete system architecture for networked AV devices implemented in Java. An overview of the architecture of Bisu is shown in Figure 6. The lower layer abstracts the underlying hardware whereas the upper layer contains device-specific code (mainly implementing the behavior of the unit). Bisu itself contains a messaging system which supports FireWire and IP-based networks, a serialization system which allows object structures to be exchanged, an interface system which defines the semantics of distributed calls and includes a component repository, and last a large number of services specific to the AV device domain.[6]

[5]Bisu is named after the Egyptian household deity (also called Bes) believed to guard against evil spirits and misfortune. Bisu is based on joint work developed in collaboration with B&O

[6]Bisu is modularized in the sense that there are multiple coexisting implementations of some services (e.g., the messaging system) and multiple (but not coexisting) implementations of other services such as the interface system.

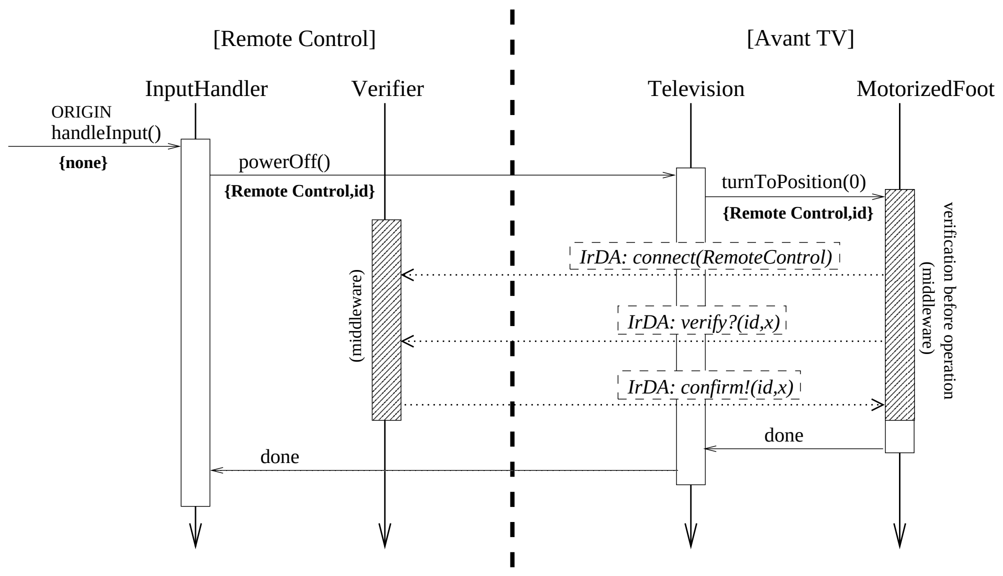


Figure 7: Run-time verification of capabilities: *present* modifier

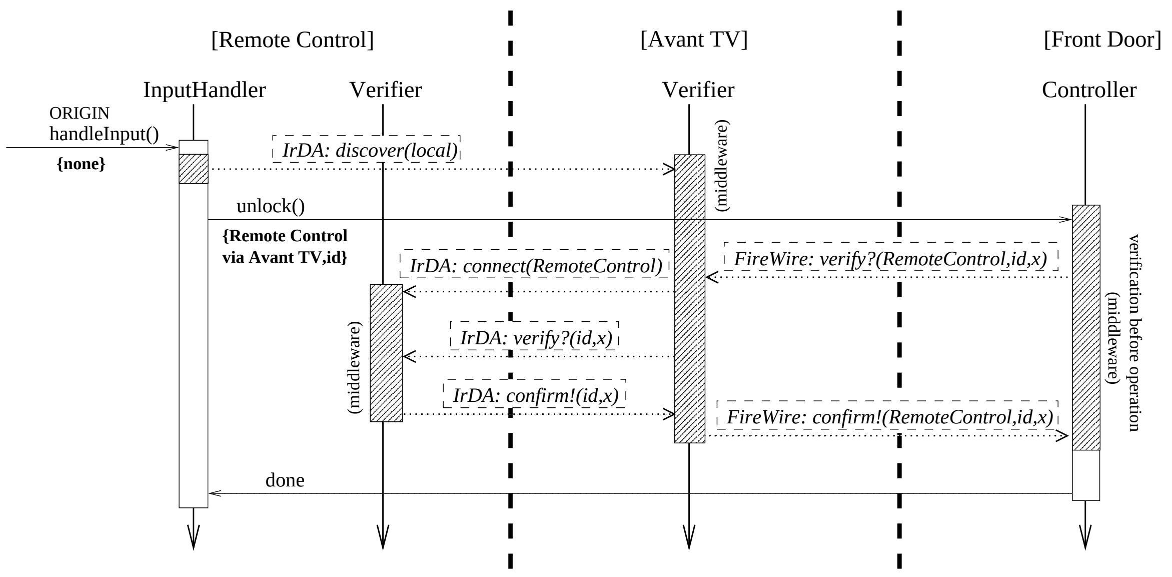


Figure 8: Run-time verification of capabilities: *local* modifier

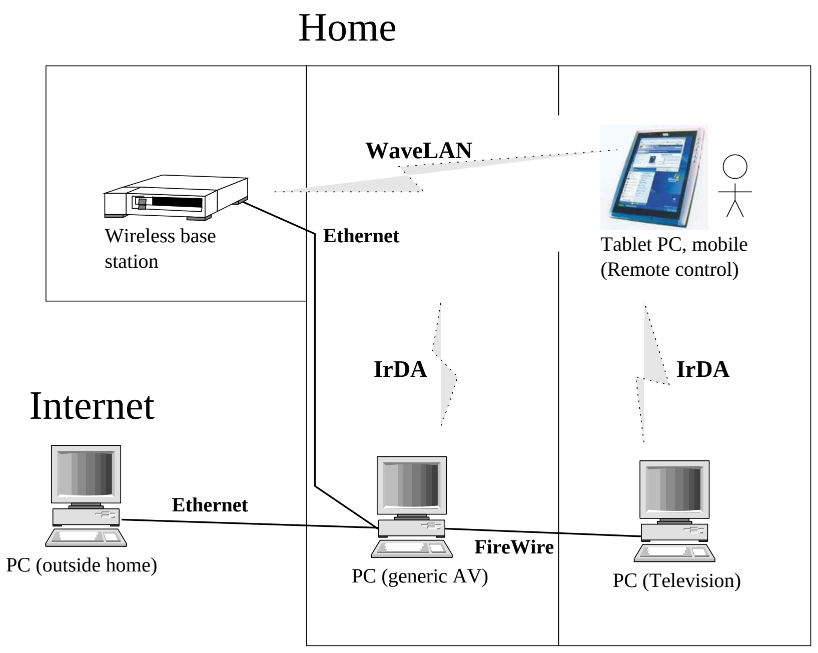


Figure 9: Experimental setup

Bisu interfaces are declared using the IDL language described in Section 3.3. The IDL compiler (described in Section 3.4) generates appropriate interface, stub and skeleton code for each component. The code verifier is implemented using the IBM *Jikes Bytecode Toolkit* [13] to check that the class files of the program do not inappropriately instantiate capabilities, as described in Section 3.4. The *internal* modifier is implemented by using a non-distributed dispatcher algorithm for such methods, making it impossible to [directly] call them in a distributed fashion.

Capabilities are simply passed as additional arguments when invoking remote operations. Capabilities are verified when restricted operations are invoked, as illustrated in Figures 7 (*present* operations) and 8 (*local* operations). A *present* operation is verified by the target by creating an IrDA connection to the origin and exchanging a key (as described earlier). Before a *local* operation can be verified, if the target or origin devices are mobile, they must each select a stationary proxy device connected to the FireWire network. This selection is done by looking up an arbitrary present device using IrDA. Afterwards, the target and origin devices can exchange a key over FireWire, either directly or through IrDA communication with their proxies.

4.4 Experimental verification

To experimentally verify the operation of our system, we have used the setup illustrated in Figure 9. A tablet PC acts as an advanced (mobile) remote control that interacts with two stationary PCs connected using FireWire. One PC acts as a television with a motorized foot, implemented using the IDL declaration described in the example of Section 3.3, the other PC as a generic AV device.

As expected, local calls can be made only when there is IR contact betwen the Tablet PC and one of the PCs, and present calls can only be made when there is IR contact between the Tablet PC and the PC representing the television. The time to make a *present* or a *local* call ranges between 400ms and 1500ms. We observe that a significant amount of time can be spent establishing the IrDA connection. The effective range of remote control is limited to roughly 2.5m with our hardware (although different IrDA hardware exhibits different range characteristics). To overcome the limitations of the IrDA procotol, we are currently building our own hardware based on standard remote control technology, which should give both shorter call-times and a much longer range.

5 Related Work

Capabilities originated within the operating systems community, as a uniform means of controlling access to (shared) system resources [9, 26, 27]. In this context, a capability typically is self-contained, in that it does not need to be verified (an operation can be executed when an appropriate capability is presented). Due to the dynamic nature of home networks equipped with mobile devices, capabilities cannot be acquired in advance and used later. Furthermore, explicitly verifying the capability provides protection against attempts by untrusted callers to execute restricted operations. Other recent uses of capabilities include controlling pointers to shared data [5] and automatic memory management using regions [8].

Context awareness is a key topic in pervasive (ubiquitous) computing, as it is what allows computers to take the current usage context into account [18, 11, 10]; it thus has a huge variety of different uses. Proximity-based login is probably what resembles our work the most. Here, the basic approach is that the user is automatically logged into a computer when physically located within a given perimeter [3, 22, 4, 11, 6, 20]. Similarly, we also use context awareness to restrict the operations that can be executed on a computer according to the physical location of the user who initiated the operation. However, unlike these other approaches where restrictions are essentially implemented manually, we reify the restrictions at the programming language level to be part of the interface each device, and we allow a fine-grained (per-operation) approach to controlling access.

Home networks technologies such as X10, UPnP, LonWorks and OSGI were briefly discussed in Section 2. In all cases safety is enforced through security — there is no backup mechanism in case an intruder gains access to the home network or if residents are not aware of unwanted consequences of their actions. Our approach on the other hand is designed to prevent unsafe actions from being executed when the user has access to the home network — rightfully or not.

6 Conclusion and Future Work

Pervasive computing is an emerging trend not only in computer science but also in everyday life. Nonetheless, programming language support for central issues such as context awareness have not been widely explored, and little attention has thus far been devoted to basic concerns such as safety. We have presented a novel approach that expresses location awareness at the programming language level, and applied this approach to improving the safety of a home network for AV devices currently being developed in collaboration with an industrial partner. This approach forces safety concerns to be considered as part of the design of the software interface of each device. Our current implementation, which is based on IR communication and FireWire, enforces basic safety concerns, in principle making it impossible for an intruder to compromise safety without physically entering the home.

In terms of future work, we are primarily interested in further exploring software aspects of location awareness. Languages such as C# and the forthcoming version of Java integrate metadata, which could be used to annotate location-based access modifiers directly onto methods without the use of a separate IDL (language independence would of course be sacrificed when moving from an IDL to a specific language). Another language-related issue is that it is often the case that two different variants of an operation exists, a more and a less privileged version, to be used depending on the origin of the caller. To make such cases explicit, overloading by access modifiers could perhaps be allowed (resolution of what method to call would take place at run-time, depending on the origin of the call). More generally, an approach similar to predicate dispatching could be used to declaratively differentiate between operations based on distance and the safety of the concrete arguments supplied to the operation [7]. In addition to these programming language concerns, we believe that mobile agents which perform actions on behalf of the user would be useful for implementing certain automated features in AV systems. However, mobile agents pose a difficult problem in terms of enforcing location-based access to a device, since they can migrate to the device and then execute operations locally; we expect that a kind of “relative” location-based access modifiers that couple the origin of an agent to what operations it can execute can be used to solve the problem.

In closing, we note that there is a tension in pervasive (ubiquitous) computing between usability. For example, the safety mechanisms integrated in Bisu could be used to automate security procedures, like proximity-based login. We are interested in exploring to what extent incorporating *safety awareness* into applications can free the user from annoying security procedures.

Acknowledgments.

This work is based on software developed in collaboration with software engineers from B&O, namely Carsten Bjerring, Niels-Ole Gram, Anders H. Poder, and Henrik Rasmussen. We also wish to thank Erik Ernst for useful comments during the initial phase of our research and Mads Torgersen for insightful commenting towards the end. Last, we wish to thank Kristian Høgsbjerg for his assistance with FireWire at a critical moment.

References

[1] Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *INFOCOM (2)*, pages 775–784, 2000.

[2] Communication at internal meeting between B&O and the authors.

[3] J.E. Bardram, R.E. Kjær, and M.Ø. Pedersen. Context-aware user authentication: Supporting proximity-based login in pervasive computing. In *Proceedings of Ubicomp 2003: Ubiquitous Computing*, volume 2864 of *Lecture Notes in Computer Science*, pages 107–123, Heidelberg, October 2003. Springer-Verlag.

[4] F. Bennett, T. Richardson, and A. Harter. Teleporting — making applications mobile. In *Proceedings of the IEEE Workshop on Mobile Computer Systems and Applications*, pages 82–84, Los Alamitos, CA, USA, 1994. IEEE CS Press.

[5] J. Boyland, J. Noble, and W. Retert. Capabilities for sharing. In J.L. Knudsen, editor, *Proceedings of the European Conference on Object-Oriented Programming (ECOOP'01)*, volume 2072 of *Lecture Notes in Computer Science*, pages 2–25, Budapest, Hungary, 2001.

[6] B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer. EasyLiving: Technologies for intelligent environments. In *Proceedings of Handheld and Ubiquitous Computing, HUC 2000*, volume 1927 of *Lecture Notes in Computer Science*, pages 12–29, Bristol, UK, 2000. Springer Verlag.

[7] C. Chambers. Predicate classes. In *Proceedings of the European Conference on Object-oriented Programming (ECOOP'93)*, volume 707 of *Lecture Notes in Computer Science*, pages 268–296, Kaiserstautern, Germany, July 1993. Springer-Verlag.

[8] K. Crary, D. Walker, and G. Morrisett. Typed memory management in a calculus of capabilities. In ACM, editor, *POPL '99. Proceedings of the 26th ACM SIGPLAN-SIGACT on Principles of programming languages*, ACM SIGPLAN Notices, pages 262–275, Austin, Texas, USA, January 1999. ACM Press.

[9] J.B. Dennis and E.C. Van Horn. Programming semantics for multiprogrammed computations. *Communications of the ACM*, 9:143–154, March 1966.

[10] Anind Dey, Gregory D. Abowd, and Daniel Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16:97–166, 2001.

[11] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster. The anatomy of a context-aware application. *Wireless Networks*, 8(2/3):187–197, 2002.

[12] Jeffrey Hightower and Gaetano Borriella. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, 2001.

[13] IBM. Jikes Bytecode Toolkit. URL: `http://www.alphaworks.ibm.com/tech/jikesbt`.

[14] IEEE Computer Society. Ieee standard for a high performance serial bus, ieee std 1394-1995.

[15] LonWorks. `http://www.echelon.com`.

[16] Object Mgmt. Group, OMG. The Common Object Request Broker, architecture and specification. Ver. 2.4.2, OMG Doc. formal/01-02-33. Framingham, Mass., Feb. 2001.

[17] OSGI. `http://www.osgi.org`.

[18] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, US, 1994.

[19] Kiran Thapa and Steven Case. An indoor positioning service for bluetooth ad hoc networks. Midwest Instruction and Computing Symposium, MICS, 2003.

[20] J. Trevor, D.M. Hilbert, and B.N. Schilit. Issues in personalizing shared ubiquitous devices. In *Proceedings of Ubicomp 2002: Ubiquitous Computing*, volume 2498 of *Lecture Notes in Computer Science*, pages 56–72, Göteborg, Sweden, September 2002. Springer Verlag.

[21] UPnP. `http://www.upnp.org`.

[22] Roy Want, Andy Hopper, Veronica Falcão, and Jonathan Gibbons. The active badge location system. *ACM Transactions on Information Systems (TOIS)*, 10(1):91–102, 1992.

[23] Roy Want, Bill N. Schilit, Norman I. Adams, Rich Gold, Karin Petersen, David Goldberg, John R. Ellis, and Mark Weiser. The parctab ubiquitous computing experiment. Technical report, 1995.

[24] Marc Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, September 1991.

[25] Mark Weiser. Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7):75–84, 1993.

[26] M.V. Wilkes and R.M. Needham. *The Cambridge CAP Computer and its operating system*. Elsevier, London, 1978.

[27] W.A. Wulf, R. Levin, and S.P. Harbison. *HYDRA/C.mmp: An Experimental Computer System*. McGraw-Hill, New York, 1981.

[28] X10. `http://www.x10.org`.

[29] Ana Zapater, Kyandoghere Kyamakya, Silke Feldmann, Marc Kruger, and Isaac Adusei. Development and implementation of a bluetooth networking infrastructure for the a notebook-university scenario. In *Proceedings of the International Conference on Wireless Networks, ICWN '03*, pages 383–390, Las Vegas, Nevada, USA, June 2003.