SIMULATION OF AN ACCESS CONTROL SYSTEM FOR IMAGE-RELATED SERVICES

Nikos Nikopoulos, Constantinos Dimou, Vassilios Chatzis and Ioannis Pitas
Department of Informatics, University of Thessaloniki
54006 Thessaloniki, GREECE
Tel.: +30-31-996304, FAX: +30-31-996304
e-mail: pitas@zeus.csd.auth.gr

ABSTRACT

This paper presents a simulation of an access control system that deals with the problem of access control and copyright protection for broadcasted image-related services. The services that concentrate our interest include mainly Pay-TV, but also the existing and forthcoming multimedia services which are expected to share the same channels as digital TV for their diffusion in the future. We assume that these services have the basic characteristic of interactivity, which implies the existence of a return channel for the users to the service providers. The simulation is concentrated on the basic operations provided by the system, that make it flexible and, at the same time, make possible or help the access control and the billing of the services.

1. INTRODUCTION

This paper presents a simulation of the access control system proposed in the framework of the European research projects ACCOPI (Access Control and Copyright Protection for Images) and OKAPI (Open Kernel for Access to Protected Interoperable interactive services). The projects deal with the problem of access control and copyright protection for broadcasted image-related services. The main objective is the development of a common, equitable and open access control system, adjusted to the European needs and requirements, that will be proposed as a European standard. The services that concentrate our interest include mainly Pay-TV, but also the existing and forthcoming multimedia services which are expected to share the same channels as digital TV for their diffusion in the future. We assume that these services have the basic characteristic of interactivity, which implies the existence of a return channel for the users to the service providers [1].

After the theoretical study regarding the access control system, there was a need for a software–based simulator that would illustrate the concepts and the capabilities of the kernel of the system in a concise and simple way. Besides, the development of a software–based simulator before a real hardware implementation is very helpful, because several implementation problems and possible theoretical gaps can be found and resolved with lower cost. Also, some parts of the simulation could be used with little or no modifications in the software of the real implementation.

2. DESCRIPTION OF THE COMMON FUNCTIONAL MODEL

The simulation is concentrated on the basic operations provided by the system, that make it flexible and, at the same time, make possible or help the access control and the billing of the services. In fact, what we have simulated is a part of the *Common Functional Model (CFM)* [2] of the access control system, that is the devices and references related with the central functions: the *User*, the *Service Provider (SPv)* and the *Trusted Third Party (TTP)*. The presence of the TTP as a trusted link between the users and the SPv's is necessary to ensure the equitability and the security of the system [3].

The devices related with the user are the *Terminal* and the *Access Control Unit (ACU)*. The devices related with the SPv are the *Access Information Directory (AID)* and the *Entitlements Collection*. The device related with the TTP is the *List of ACUs*. What we have simulated for the user is the combination terminal–ACU and the implementation of the operations provided by the Operating System of the terminal. What we have simulated for the SPv is a non-interactive server that responds to user calls (for ordering/cancelling entitlements) and an interactive operational console. What we have simulated for the TTP is a non-interactive server that responds to user calls (for registration of an ACU, blacklisting of an ACU) and SPv calls (for checking the validity of an ACU, requesting the enciphering key of an ACU, blacklisting of an ACU), and an interactive operational console [4]. The simulations can be downloaded via anonymous ftp from **ftp://poseidon.csd.auth.gr/pub/okapi/**.

3. OPERATIONS

In this section, we shall describe the operations that are provided by the access control system. These operations are the ones that we have so far included in the simulation. Most of the operations depend on, or modify, the references we have described in the previous section. We shall not describe in detail some operations that are obvious, for example several display operations.

3.1. Operations performed by the User

The OS of the terminal provides an interface to the user, enabling him to perform several operations. In fact, what we have simulated for the user is this interface and the implementation of the operations provided:

1. Display terminal settings

2. Reset lockout password
3. Change lockout password
4. Change lockout threshold
5. Insert/Change ACU
6. Registration of a new ACU to TTP
7. Display the SPv index of the ACU in use
8. Display ordered entitlements
9. Order entitlements
10. Cancel entitlements
11. Send blacklist request to TTP

In the following, we shall describe the most important of them.

Reset lockout password

This operation helps in case the user forgets the lockout password he has defined.

- User is asked for the root password (up to three times).
- If User gives the correct root password, the lockout password is reset to the root password.

Change lockout password

This operation helps the terminal owner to change the lockout password, which is used to restrict the access only to those services with level under a defined lockout threshold (e.g. parental control over their children).

Registration of a new ACU to TTP

After a user has bought a new ACU, he has to register it to TTP. This ACU is supposed to have been certified, so it must contain a certificate. ACU must prove its certification to TTP. The procedure that is followed for this operation is the following:

- Thanks to a random generator, the ACU calculates a couple of enciphering/deciphering keys *(En/Dn)*. The *Dn* should be kept secret inside in the ACU.
- ACU sends its ACU ID, the customer number and the calculated public *En* key to the TTP.
- ACU has to prove that it contains a certificate, ensuring that *Dn* is protected inside the card. This proof is made by a 0-knowledge algorithm between TTP and ACU:
  - TTP chooses a random number r and sends it to the ACU.
  - ACU "signs" *(r)* with the help of the internal certificate and sends it back to the TTP. By checking the signed value, TTP can determine whether the ACU is certified or not.
- If the ACU is proved to be certified, TTP adds an entry for this ACU in the *List of ACUs* (as a valid ACU). TTP notifies the result of the operation to the U.
- If the registration was successful, the ACU updates its registration flag.

Since the ACU is registered to the TTP, it can be used to store entitlements from various SPv's.

Order entitlements

To gain access to a certain service, the user has to request for entitlements. We assume that a service is identified by an ISBN-like number.

- User sends the ACU ID and the ISBN of the requested service to the respective SPv.
- SPv sends the ACU ID to the TTP for validity check.
- TTP sends back to SPv a report on the validity of this ACU.
- SPv notifies this report to the U. If the ACU is valid and the service with the sent ISBN is available from this SPv, it sends also the requested entitlements to the User and adds an entry in the AID.
- If the entitlements are sent, they are stored in the ACU.

Cancel entitlements

Subscriptions that do not contain a fixed ending time or duration can only be terminated by this functionality. If the subscription time is limited, this functionality can be used to cancel the subscription prematurely. In the case of ISBN-like number, the entitlements can be cancelled before the service is consumed.

- User sends the ACU ID and the ISBN of the service that he wants to cancel to the respective SPv.
- SPv sends the ACU ID to the TTP for validity check.
- TTP sends back to SPv a report on the validity of this ACU.
- If the ACU is valid, SPv removes the relative entry from the AID. SPv notifies the User of the result of his request.
- If the entitlements were cancelled, they are removed from the ACU.

Send blacklist request to TTP

A user can ask from the TTP to blacklist an ACU. This should be done if his ACU is lost or stolen. The system should provide a way to find the identity of the person that sends such a request. For further security, this operation is password–protected.

- User is asked for the root password (up to three times).
- If User gives the correct root password, he is prompted to give the ACU ID of the ACU he wants to request blacklisting for from the TTP.
- The ACU ID is sent to the TTP.
- If the ACU is registered and valid, TTP blacklists this ACU. TTP notifies the User of the result of his request.

3.2. Operations performed by the SPv

What we have simulated for the SPv is mainly a non-interactive server that responds to user calls (for opening of a line, ordering/cancelling entitlements). Also, we have simulated an interactive interface, which co-operates with the server, that enables an authorized person at the SPv–end (handler) to perform several operations. The access to this interface is controlled by an authentication password. The operations are the following:

1. Display SPv's data
2. Display ordered entitlements from users
3. Display entitlements for available services
4. Add new entitlements
5. Send blacklist request for an ACU
6. Stop SPv's server
7. Change authentication password

We describe only the most important of them:

Send blacklist request for an ACU

An SPv can ask from the TTP to blacklist an ACU. This could be done, for example, if the owner of this ACU has not paid his depts to the SPv. For further security, this operation is password–protected.

- SPv handler is asked for the authentication password (up to three times).
- If SPv handler gives the correct authentication password, he is prompted to give the ACU ID of the ACU he wants to request blacklisting for from the TTP.
- The ACU ID is sent to the TTP.
- If the ACU is registered and valid, TTP blacklists this ACU. TTP notifies the SPv handler of the result of his request.

3.3. Operations performed by the TTP

What we have simulated for the TTP is mainly a non-interactive server that responds to user calls (for registration of an ACU, blacklisting of an ACU) and SPv calls (for checking the validity of an ACU, requesting the enciphering key of an ACU, blacklisting of an ACU). Also, we have simulated an interactive interface, which co-operates with the server, that enables an authorized person at the TTP–end (handler) to perform several operations. The access to this interface is controlled by an authentication password. The operations are the following:

1. Display data for registered ACUs
2. Send blacklist request for an ACU
3. Stop TTP's server
4. Change authentication password

We describe only the most important of them:

Send blacklist request for an ACU

The TTP handler can ask from the TTP server to blacklist an ACU (this is because the server is responsible for updating the *List of ACUs*). This could be done, for example, if the owner of an ACU goes in person to TTP and asks for its blacklisting for some reason. For further security, this operation is password–protected.

- TTP handler is asked for the authentication password (up to three times).
- If TTP handler gives the correct authentication password, he is prompted to give the ACU ID of the ACU he wants to request blacklisting for from the TTP server.
- The ACU ID is sent to the TTP server.
- If the ACU is registered and valid, TTP blacklists this ACU. TTP notifies the TTP handler of the result of his request.

4. COMMENTS ON THE USE OF C++, PVM, TCP/IP, MOTIF

The simulation was implemented in UNIX environment, using the C++ programming language together with the PVM programming system. The requirement of using a language which is close to the problem to be solved, so that the concepts of a solution can be expressed directly and concisely, is met by using C++. Therefore, the concepts that have been introduced in the CFM (functions and functionalities) can easily be expressed and managed by using C++. Besides, the relationships and interactions between the different concepts in the CFM can be more naturally expressed by C++. Thus, C++ can help us in the creation of a program which will be more readable, more comprehensive and more reliable than that which can be implemented by using C. These characteristics are very important in our application due to the fact that our program will be large enough, and a variety of persons will cope with it.

Besides the use of C++ for the programming of the processes related to the different functions of the CFM, we used PVM (Parallel Virtual Machine) as the tool for implementing the communication between these processes. PVM is a programming system that enables message-passing between processes that can run on the same or different machines. It also enables processes to join groups and makes possible the broadcasting of messages to all the processes of a group [5]. Without going into many details, we mention that the above characteristics are implemented by employing a daemon process, called *pvmd*, on each machine that participates in the system. PVM can interconnect a network of UNIX machines, which are referred to as *hosts* and can have different architectures, and see them as one system referred to as *Virtual Machine*. It should be noted that the primary motivation for the development of PVM was the implementation of a tool that would enable the parallel exploitation of the computational power of a number of available UNIX machines for the quicker solution of computationally demanding problems. This is achieved by splitting each problem into a number of parts that can run in parallel and co-operate by exchanging intermediate results. Thus, the above mentioned characteristics of PVM make it a good choice for applications that need to simulate communication networks only with software. Also, it is very important that the capabilities of PVM are provided as library routines that can be called inside C, C++ or Fortran applications. Finally, PVM is free software and it is portable to a wide variety of architectures. The role of PVM in the simulation is depicted in Figure 1.

Although PVM was very useful at the first stages of the simulation, it was later substituted by the use of TCP/IP Sockets. Sockets are a well known means of communication between processes used widely by many applications. They provide a more low-level control over the communication medium and are available for almost all the hardware platforms. In order to standardize the exchange of messages between the different processes of the simulation environment, the Interface Management Document (IMD) protocol was developed by the ADETTI group. The protocol describes the exchange of messages between the Functional Modules (FM) and the Graphical User Interface (GUI) modules of the actors as well as between the different actors.

In order to provide friendliness and easy-to-use access to the end user of the system, a GUI for X-Windows was added to the simulation. In particular the GUI was applied in the case of the User menu, the SPv console and the TTP console.

X-Windows System is a network-based system which provides a way of writing device independent graphical and windowing software that can easily be ported from one machine to another. X-Windows communicate with the user through a set of routines known as *Xlib*, which translate C data structures and procedures in a special form of X protocol messages and handle the interface between the application and the network [6]. Xlib provides only low-level routines, therefore special higher level routines have been developed in order to provide tools for creating attractive and easy-to-use graphical user interfaces. A popular versatile GUI creation tool is *OSF/Motif* [7]. Motif applications are based on event driven programming and on the use of *widgets*, such as push buttons, menus and *callback routines* which connect the graphical interface with the kernel procedures.

All basic program functions were implemented in the form of push buttons, and appropriate functions were attached to them. The front panel of the applications comprised of a basic window with a number of push buttons justified in rows and columns, as shown in Figures 2, 3 for the USER and SPv program respectively. In turn, every callback routine opens appropriate windows, dialogue boxes or other widgets specific to the needs of the corresponding function. When a function ends, it returns to the main application window.

All the work of the application is therefore done in each button's callback routines. They are responsible for creating the appropriate widgets for getting input and displaying output, and also they call the underlying kernel routines for communicating through PVM with the servers. Effort was made in order for the GUI implementation to be quite independent from the kernel routines. Therefore, only in a few cases Motif routines were inserted in critical points of the kernel. In most cases the Motif user interface was restricted to getting user input and providing output. Finally, the design philosophy of the application was towards event driven programming, as imposed by the Motif and X system structure in general.

ACKNOWLEDGEMENTS

The work reported in this paper was performed for the European ACTS projects ACCOPI and OKAPI.

REFERENCES

[1] Jean-Yves Mertes, Jean-Jacques Quisquater, *"List of Requirements"*, Project ACCOPI internal report WP1/I/UCL/3.3, June 1994.

[2] Jochen Rindfrey, Eckhard Koch, *"The Common Functional Model V6.2"*, Project ACCOPI internal report WP2/I/IGD/6.2, Sept. 1994.

[3] J.-J. Quisquater, J.-M. Boucqueau, S. Lacroix, B. Macq, *"Conditional Access and Copyright Protection based on the use of Trusted Third Parties"*, Project ACCOPI internal report, May 1995.

[4] Nikos Nikopoulos, Ioannis Pitas, *"Kernel Simulation V1.0"*, Project ACCOPI internal report WP4/I/AUT/1.0, June 1995.

[5] Al Geist *et al.*, *"PVM 3 User's Guide and Reference Manual"*, Oak Ridge National Laboratory, TM-12187, Sept. 1994.

[6] Adrian Nye, *"Xlib Programming Manual"*, O'Reilly & Associates, 1992.

[7] SunSoft (Sun Microsystems, Inc.), *"Motif 1.2 Programmer's Guide"*, Open Software Foundation, 1992.

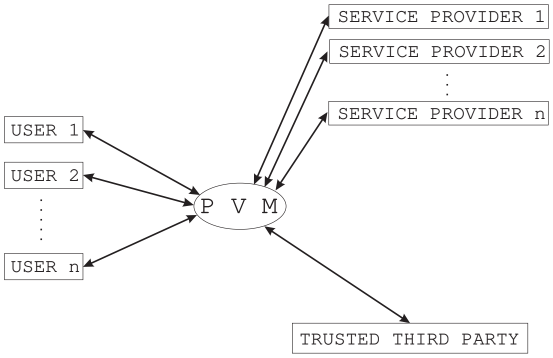

Figure 1. The role of PVM in the Kernel Simulation.

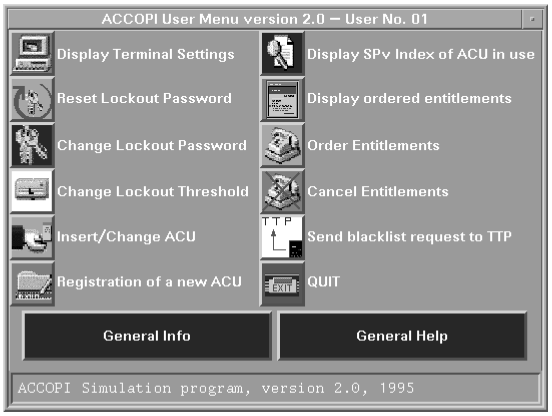

Figure 2. The front panel of the USER program.

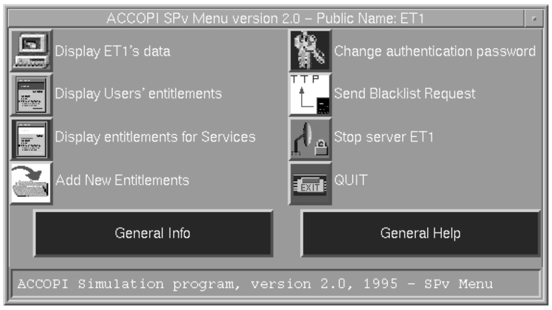

Figure 3. The front panel of the SPv program.