

Middleware Services for DRM

Pramod A. Jamkhedkar
Department of Electrical and
Computer Engineering
University of New Mexico

Albuquerque, New Mexico 87131, USA
Email: pramod54@ece.unm.edu

Gregory L. Heileman
Department of Electrical and
Computer Engineering
University of New Mexico,

Albuquerque, New Mexico 87131, USA
Email: heileman@ece.unm.edu

Ivan Martinez-Ortiz
Centro de Estudios
Superiores Felipe II

Aranjuez, Madrid 28300, Spain
Email: imartinez@cesfelipesecondo.com

Abstract—Lack of generic digital rights management applications has stunted the growth of the media distribution industry. In this paper we point out the need for middleware services required to develop digital rights management (DRM) applications. Many functionalities are common to most DRM applications and by nature are highly distributed. Standalone DRM applications have found it difficult to implement these services in an efficient manner and have led to closed solutions with limited capabilities. This paper categorizes these functions with reference to a layered DRM framework as middleware services. The characteristics and interface of each of these services is defined along with a prototype implementation of an agent-based negotiations service.

I. INTRODUCTION

The growth of the Internet in recent years has provided a cheap and efficient channel for the content distribution industry. However, the ease of reproduction and distribution of content on the Internet necessitates robust and efficient digital rights management (DRM) systems. Over the past decade and a half, there have been numerous attempts to create generic DRM systems. By the late 1990's, IBM had implemented some of these notions in their Cryptolope technology [1], [2], and subsequently InterTrust, in what is perhaps the most ambitious DRM system implemented to date, released their Digibox technology [3]. However, the commercial success of these systems, along with many others, has been limited [4]. Most of these technologies failed because either they did not cover certain aspects of DRM that a customer required, or they crumbled under their own weight while trying to achieve everything.

The digital content and software distribution industry is expected to evolve significantly over the coming years. Most of these applications will eventually require DRM services to manage rights associated with digital content. DRM applications require correct implementations of different kinds of services such as security, trust management, tracking services, and negotiation services etc. Many of these services are common to most DRM applications and need to operate in a highly distributed and networked environment. Standalone DRM applications have found it difficult to implement these services in an efficient manner and have led to closed solutions with limited capabilities. It is therefore necessary to implement these functionalities as middleware services to DRM applications as a part of the network infrastructure. Given

these services, content distributors can then create custom-made DRM applications that cater to the needs of different industries.

The first step in this direction is to identify the services which are common to most DRM applications. The DRM architecture proposed by Jamkhedkar et al. categorizes various DRM functionalities within a layered framework [5]. Each layer represents a set a middleware services which can be used by DRM applications. These middleware DRM services will make it possible to develop interoperable DRM solutions which will facilitate rapid growth in the DRM industry [6], [7].

An overview of the paper is as follows. Section II argues the need for middleware services for DRM systems along with a brief overview of the different types of services that are common to most DRM applications. We provide a brief overview of a layered architecture in section III by categorizing these services among different layers. Section IV explains these services in detail with the functions provided by each service along with the type of interface. Section V provides a brief explanation of a prototype implementation of a rights negotiation service implemented using an agent based architecture. Finally, in section VI we provide some concluding remarks along with a road map for the development of these services.

II. NEED FOR DRM MIDDLEWARE

Traditionally, DRM systems have maintained a very fine balance between the interests of content owners and content users. However, the evolution of computer and the Internet has disturbed this balance significantly in the favor of the consumers. Ease of content reproduction and distribution has given the consumers a significant edge to violate copyright. Due to the fear of rights violations, content distributors have been unable to tap the vast potential of the Internet as a distribution platform. This has in turn worked against the interests of consumers who wish to have unlimited access to authentic digital content at a reasonable price.

A. Background

DRM systems aim to solve this problem by enforcing copyright. There have been numerous efforts over the past decade to create generic DRM systems which cater to the

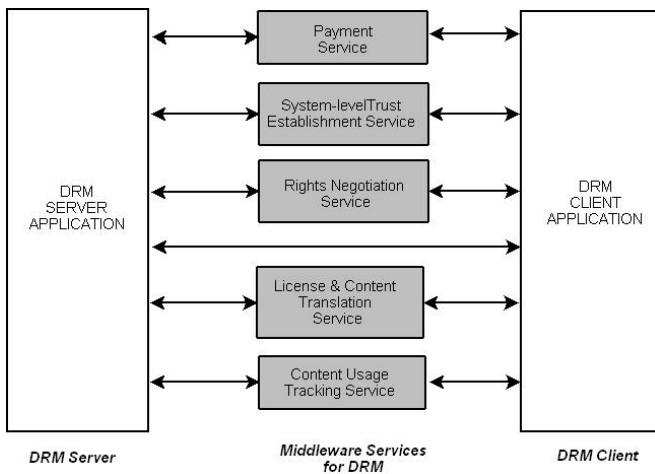


Fig. 1. Middleware services for DRM

needs of different kinds of rights management applications. The commercial success of these products has been extremely limited, which is surprising given the universally recognized importance of the problem [4].

The technical reasons for the failure of earlier DRM systems can be attributed to a number of different factors mostly concerning the architecture and inherent nature of these systems [5], [6]. A DRM system involves a number of players such as content owners, content distributors, license managers, trusted third parties, consumers, etc. The process of rights management involves complex interactions among these players, each with conflicting interests. These conflicting interests mandate a careful implementation of security and trust management protocols. Also, the involvement of a large number of players within DRM systems, make them highly distributed and difficult to manage.

DRM systems are not standalone applications, but generally function as a supporting application for different businesses. This forces additional requirements on generic DRM systems with regards to flexibility and adaptability. Such systems are required to be flexible enough to adapt to different business models in terms of rights expression and enforcement. Also, these systems need to take care of transport and license format compatibilities while dealing with different devices.

Trust management and security considerations coupled with compatibility issues have forced many of the DRM systems to be developed as monolithic systems, despite the distributed nature of DRM processes. Thus we have non-interoperable DRM systems, where vendors are forced to develop all the supporting DRM functionalities leading to ossified and closed DRM systems with limited capabilities. This has resulted in stunted growth in the DRM field. The universal applicability of DRM systems along with their distributed nature, requires the need to rethink certain DRM functions as generic middleware services that will be used by various applications to satisfy their rights management needs. The following discussion makes a case for DRM middleware services.

B. A Case for Middleware DRM Services

Middleware can be viewed as a reusable, expandable set of services and functions that are commonly needed by many applications to function well in a networked environment [8]. Middleware services have seen an evolution from very simple to quite complex. Simpler middleware services created during the the initial stages of network evolution have now become a part of the networking infrastructure itself [8]. As applications evolve in their complexity, the nature of their need for middleware services becomes increasingly complex. We have seen this trend from basic middleware services like DNS to the development of more complex services like PKI.

The digital content distribution industry is expected to evolve significantly over the coming years. Most of these applications will eventually require services to manage a variety of complex rights management scenarios. It is necessary to identify which functions within a DRM process are ideal candidates to be transformed into middleware services. These are the functions which share the same properties as those displayed by typical middleware services. As mentioned earlier, these are commonly used distributed functions needed by different DRM applications. Commonly used functions in the DRM process which display these properties are discussed below.

Certain elements of DRM systems are highly distributed and require a hierarchical delegation and control, similar to the methods required by DNS and PKI services. For example, in most DRM applications, there is a need for content producers to establish not only authentication but also complex levels of trust with the rendering devices before making a decision to transfer the content. Establishing such levels of trust with unknown rendering devices will require a hierarchical and distributed trust management mechanism. Payments also form an inherent part of DRM systems, often requiring complex payment mechanisms to support super-distribution scenarios [9]. In a super-distribution scenario, payments made by the end customer flow back along the value chain, with each player in the chain receiving a share of the end payment.

Many DRM applications require complex negotiations with user devices. It is necessary to carry out these negotiations with different kinds of devices requesting content. Such services can be implemented as middleware acting as a negotiator. Transport and license format compatibility requires translation services to transform licenses from one DRM regime to another. Finally, content usage tracking requires interaction with various devices to track the usage of content. Such a service needs to be highly distributed interacting with various kinds of devices and hence can be effectively implemented as a middleware.

The problems mentioned above are exactly the problems commonly eliminated by middleware solutions. Therefore, DRM middleware must create an expandable, consistent and distributed service representing these functions. Applications can then use this middleware to develop their own DRM services. Figure 1 shows how these services can be

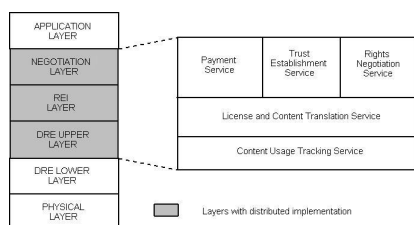


Fig. 2. DRM layers with services to be implemented in each layer.

implemented across a network and used by DRM applications on the server and the client side. In the next section we will provide a brief overview of the layered DRM framework and categorize these functions among different layers.

III. A LAYERED DRM FRAMEWORK

Most of the current DRM systems are process-based. They are developed on the basis of certain common use cases. A typical DRM use-case can be explained as follows:

The digital content is packaged and stored in the content servers protected by certain encryption mechanisms. The media is then transferred to the rendering device which obtains the license from the license server. The license contains the decryption key which is used to decrypt the content. Finally, the DRM controller on the rendering device ensures that the terms of license are not violated.

It is difficult to separate out various DRM functions by using a process-based approach. Another approach would be to divide the DRM system in layers as explained by Jamkhedkar et al. [5]. Each layer in a layered DRM system categorizes DRM functions according to the role played by these functions. Figure 2 shows the six layers that make up a complete DRM system. The Application Layer represents the DRM application which supports a particular business model. The Negotiation Layer represents the layer where rights negotiations and system level trust establishment takes place. The Rights Expression and Interpretation (REI) layer forms the core service of any DRM system, involving expression and interpretation of rights. The Digital Rights Enforcement (DRE) Upper layer includes all the license management mechanisms and content usage tracking. DRE lower layer involves all the operating system level rights enforcement mechanisms. And finally, the physical layer represent the actual hardware of the rendering device.

The middle layers, namely, the Negotiation Layer, REI Layer and the DRE Upper layer, are distributed layers and thus most of the middleware services would be concentrated in these layers. Figure 2 shows the categorization of the middleware services according to each of these middle layers.

The Negotiation Layer has three sub-services: rights negotiation, system-level trust establishment and payment. All three of these services provide a way to help carryout negotiations between the content provider and the user device. Hence, these three services collaborate with each other to provide a high-level complex negotiation service at the Negotiation Layer.

The REI Layer, representing rights expression, carries out the role of content and license creation. For devices operating in a different DRM regime than that of the license issuer, it is necessary to transform the content and the license to the formats supported by the device [10]. Hence, license and content translation services would operate at the REI Layer. Finally, the tracking service needs to coordinate with the license management mechanisms of the devices on which content is being rendered. Therefore this service would operate at the DRE Upper Layer, which represents high level rights enforcement mechanisms.

In the next section, we will discuss these services in greater detail with respect to nature of their services, their interfaces and coordination with other services in the same layer.

IV. CHARACTERISTICS OF DRM MIDDLEWARE SERVICES

As mentioned in the previous section, DRM middleware services can be categorized in different layers. The characteristics and interfaces of these services are defined below.

A. Negotiation Layer Services

Every DRM application needs to carry out negotiations with the rendering device before actually passing over the content. Two of the crucial functions in the Negotiation Layer are the establishment of system-level trust with the rendering device and performing rights negotiations with the device in accordance with the policies defined by the content provider.

In an interoperable DRM system, the enforcement mechanism running on the rendering device and the device itself are not developed by the content provider. For example, in the case of iTunes, the device, i.e. the iPod, and the software running on it, are both developed by the license issuer. Hence in this case there is no question of trust establishment since the capabilities and authenticity of the device and the rights enforcement mechanism on the device are known to the license issuer. However, in case of an interoperable system, the license issuer must first be assured of the capabilities and the authenticity of the device and the rights enforcement software. For example, before handing over a time-based license to a device, the license issuer must be assured that the device includes a secure clock. Also, before handing over the license, the license issuer must know about the types of licenses handled by the license management mechanism on the device. It is only after these considerations are addressed that a license issuer can actually issue a license.

A middleware providing trust establishment will have to provide two basic kinds of services:

- 1) **System-level trust establishment:** This includes a service which authenticates the devices and its rights enforcement capabilities. This includes, authentication of the hardware capabilities of the device and the license management and rights enforcement mechanism of the device. It is necessary to define certain standards for the types and levels of security that a particular device can support. The license issuer can then categorize the rights that can be associated with a particular type and

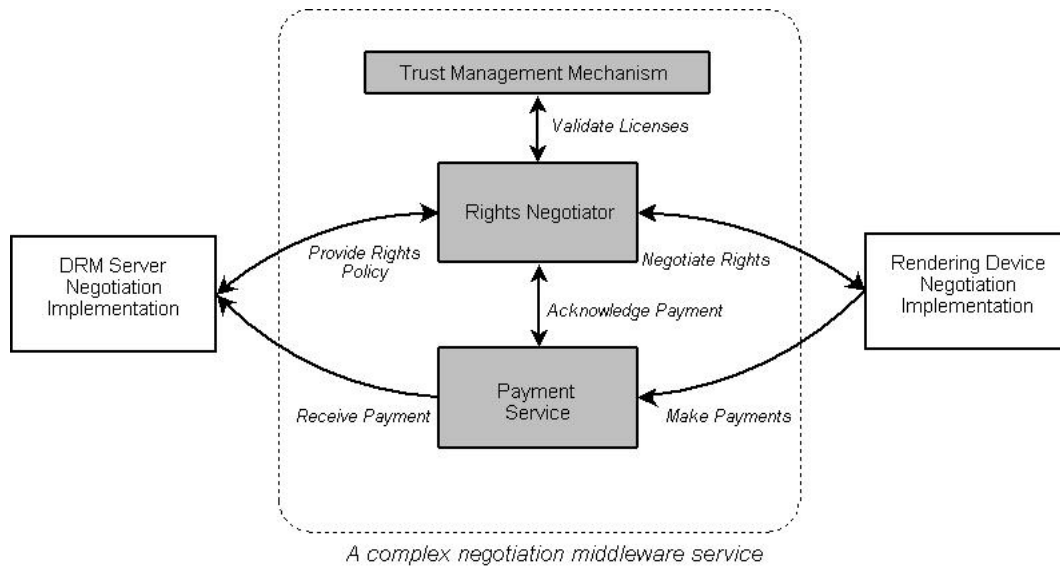


Fig. 3. A complex negotiations middleware composed of simpler middleware services at the Negotiations Layer

level of security that is needed in order to enforce these rights. The license issuer also needs to authenticate the rights enforcement software running on the device and check whether it has been breached and/or updated accordingly. Once these capabilities are authenticated, then the license issuer can be assured that the rights will be duly enforced on the device.

- 2) **Compatibility:** At present, there are various kinds of rights expression languages and the licenses are created in different formats. In addition, there are various structures used for the licenses themselves, for example hierarchical licenses. For different devices running different license management softwares, how does the license issuer know what kind of a license to issue to a particular device. As explained earlier, this is not an issue in monolithic DRM systems where the license management software for the device is provided by the license issuer. Hence the license issuer must obtain information about the the kind of licenses managed by the license management software and create the correct license type for that particular device.

Both of these functionalities can be provided by the current PKI middleware. It is necessary to create the required digital certificate formats which will represent the above information. These certificates will carry the necessary information regarding the capabilities of these devices and license management softwares. These certificates will have to be assigned by their corresponding device and software manufacturers. License issuers can then authenticate these certificates.

The second type of service that needs to operate at the Negotiation Layer is the rights negotiations service. Many content providers have a set of policies for offering rights to users. These policies are defined in terms of the following:

- 1) **Content Properties:** These generally include properties of content that decide the policy. For example, the

quality of the content or value of the content in terms of user demand.

- 2) **Rights:** These represent the actual actions that can be performed on content including reproduction, transfer and/or usage rights.
- 3) **Constraints:** Constraints define the conditions under which the rights should be exercised. Constraints can be defined in terms of range, time period, location, etc.
- 4) **Environment:** Environment defines the properties of the device on which content will be rendered. For example, environment would be defined by the security mechanisms present in the rendering device.

It is essential that any offer made by the license issuer, does not violate the policies defined by the content provider. In fact, the license issuer has to negotiate with the buyer in the form of a bargain in order to provide attractive offers which adhere to the policy provided by the content provider. A potential DRM middleware for the negotiations layer would be a service which conducts negotiations on the behalf of the content provider. The rights negotiation service is distributed in nature. The rights negotiator has to provide offers to different requests and authenticate and validate the capabilities of different devices before giving out these offers. The rights negotiation service can be seen as a negotiator working on behalf of the content provider. We implemented a prototype negotiator using an agent based framework. Details on the workings of such a negotiator will be discussed in the next section.

The third type of service to operate in the Negotiation Layer is the Payment Service. DRM systems require much more complex payment service than traditional business models. Many DRM applications require super-distribution of content in which the buyers sell the content on behalf of sellers and get a share of the price paid. In this type of business model, the payments made by the consumers must flow back to the content owner with each intermediary getting his/her share

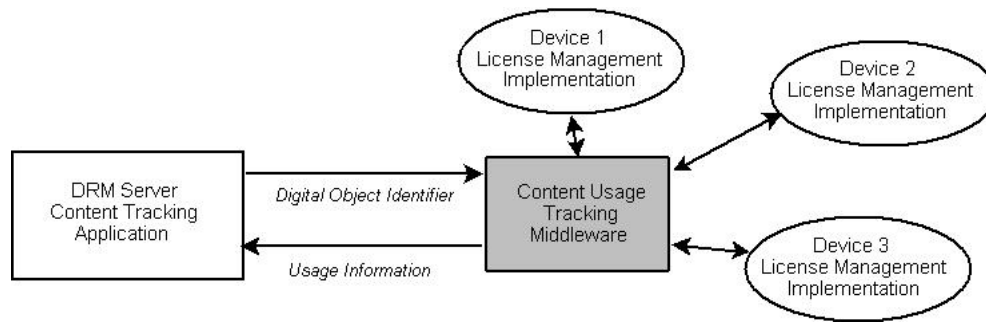


Fig. 4. A middleware providing content usage tracking service.

of profit. DRM applications will require payment services to manage such complex transactions.

These three services defined in the Negotiation Layer must collaborate to provide sophisticated negotiation services to DRM applications. The interaction of these services is shown in Figure 3. Such a service will allow application developers to support a range of complex negotiation scenarios.

B. Rights Expression and Interpretation Layer Services

Licenses are expressed in Rights Expression Languages (RELS). ODRL and XrML are the two most common RELs currently used [11], [12]. The expressive power of RELs is not the same. The process to convert a license in one format to another is very complex. Before the license is converted, the rights need to be cast in terms of the new environment. The factors that need to be taken into consideration during license conversion is an area of active research [7], [10].

In interoperable DRM systems, the license issuer does not manage the license management software running on the rendering devices. Hence, the license format used by the license issuer and the one used by the device are different. A DRM vendor application which intends to sell content to such devices must provide a license in the format used by the rendering device. This will be a common requirement in an interoperable DRM environment, and DRM vendors will need to implement the license conversion technologies within their own DRM solutions.

One approach is to implement this required service as a middleware which will ensure the compatibility of license formats between DRM vendors and rendering devices. This service could then be used by different DRM vendors to sell their content to a wider range of consumer devices. The workings of such a service can be explained as follows:

- 1) The DRM vendor creates the license in its own format, and the license, along with the device id is passed on to the translation service.
- 2) The translation service obtains the information required for license translation from the device.
- 3) The license is converted to the new format and handed over to the device.

C. Digital Rights Enforcement Upper Layer Services

In addition to rights enforcement, content usage tracking forms one of the crucial components of many DRM applications. The DRE Upper layer implements license management mechanisms. It is the license management mechanism that keeps track of content usage by the user. Different devices can have different license management mechanisms which can keep tracking data in different forms. A DRM vendor application which wants to keep track of the usage of particular content must have the ability to query these different license management mechanisms.

Instead of having individual DRM applications contain such capabilities, it is more convenient to create a service which gathers usage information about a given digital content from different devices, and provides the result back to the DRM vendor application. Such a service can process the information received from different devices to convert it to a form understood by the DRM vendor application. The workings of such a tracking service are shown in Figure 4 and can be explained as follows:

- 1) The DRM vendor application provides the digital object identifier (a globally unique identifier for digital content) to the tracking service along with the information about the nature of service required.
- 2) The tracking service connects to the license management application interface of the devices and queries the usage record for that particular digital object.
- 3) The tracking service processes the information obtained and provides the result back to the DRM vendor application.

These are the middleware services that need to be implemented for DRM applications. These middleware services will facilitate the development of customizable DRM applications. In the next section, we explain the interaction diagram showing the collaboration among various services in the Negotiations Layer to provide a complex negotiation mechanism for the DRM vendors. This will be followed by a detailed explanation of the implementation of an agent based rights negotiation implementation.

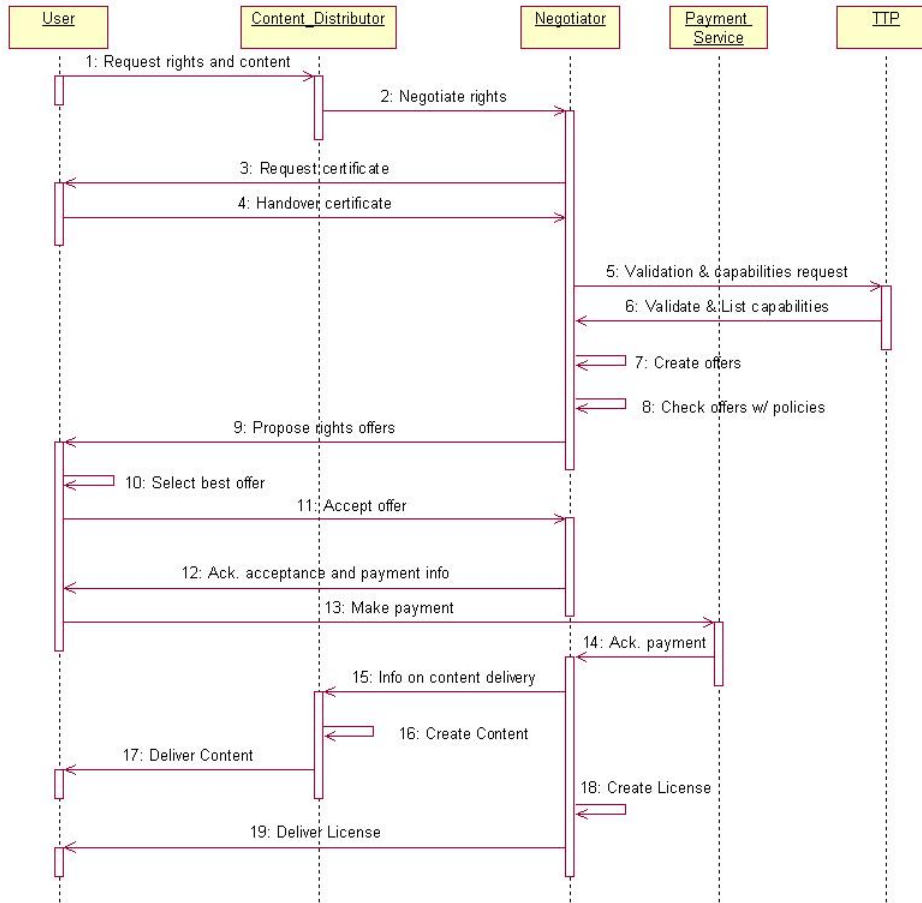


Fig. 5. Interaction diagram showing rights negotiation middleware

V. NEGOTIATION MIDDLEWARE PROTOTYPE IMPLEMENTATION

As introduced in section IV one of the layers representing DRM services is the Negotiation Layer. The Negotiation Layer represents the processes that lead to the finalization of the deal between the content provider and the consumer. As mentioned earlier, these processes involve trust establishment between the parties involved, negotiations leading to rights agreements, and finally leading to payment agreements. These three services involve complex interactions among themselves to achieve a common goal of finalizing the deal.

A. Negotiation Layer Middleware

The implementation of the above mentioned processes as middleware services aims to reduce the complexity of the server and client DRM applications, making them lightweight and easy to develop with a well-defined API. The responsibility of the content distributor is reduced to provide the content and policies (rules to be adhered to while distributing rights

offers, or alternatively, the rules of negotiation). The negotiator then negotiates on behalf of the content provider.

The negotiation is initiated by the user through a client application, connecting to an application that belongs to a content distributor to search for specific content (message 1). If the content distributor has the requested content, then a negotiation over rights can be initiated. The content distributor application delegates the negotiation to the DRM negotiation service (message 2). The Negotiator starts the negotiations with trust establishment protocols which include validation of the capabilities of the device in terms of rights enforcement (messages 3, 4, 5 and 6). Once the capabilities are available and validated, along with the policies provided by the content distributor, the negotiation service creates a set of offers that will be sent to the user application (message 9). The user then selects the best offer and accordingly informs the Negotiator (messages 10 and 11). Once the negotiation service receives the acceptance notification, it sends the payment voucher for the user and waits to receive confirmation of the payment (messages 12, 13 and 14). Next, the negotiation service notifies

the content distributor application that the sale has been concluded, including the technical details for the content creation (file formats, encryption options, etc.). Thereafter, the content distributor creates the appropriate content and delivers it to the user (messages 15, 16 and 17). Meanwhile the negotiation service generates the license that will be delivered to the client application upon request (messages 18 and 19).

B. Agent Based Rights Negotiation Prototype

Agreements in currently deployed DRM systems are inflexible and operate in a single direction dictated by the content provider. Since a rights agreement is a deal between two parties, namely the content provider and the consumer; there is a need for systems where both parties play an equally active role in negotiating the deal [13]. The automatic negotiation process is not a new topic in computer science, particularly, software agents have been successfully applied in different negotiation scenarios [14], [15], [16]. The rights negotiation prototype is created using a Multi Agent System (MAS) implemented over the Java Agent DEvelopment (JADE) framework [17].

The prototype includes a set of seller agents representing the negotiator and buyer agents that represent the user. The seller agents negotiate on behalf of the content provider and employ the services of the policy agents who are given policies by the content providers. The initial goal of the buyer agent is to publicize a request to buy particular content with certain specifications provided by the user. On the other hand, the seller agents try to provide offers to buyer agents and adhere to the policies provided by content providers. The buyer agent then selects the best offer from those obtained from various seller agents. Both the seller agents and the policy agents register their services to other agents by means of a common directory service which functions as the yellow pages. In this directory each agent registers its services identified by a “type”. In this prototype implementation, the policies are represented in the form of a MySQL database, which can be replaced by rule-based languages such as Jess or Prolog in commercial systems. Rights are used as a payload for negotiations and are expressed in terms of a rights ontology which we defined using the Protege software [18]. The agents negotiate using the Foundation for Intelligent Physical Agents(FIPA) contract net interaction protocol [19]

Figure 6 shows the messages exchanged by agents during a typical run of a rights negotiations transaction among a set of agents using Java Sniffer. First the Buyer agent requests a list of agents selling rights, from the Directory and the Directory service replies with the list of seller agents (messages 1 and 2). In message 3, the Buyer agent sends a Call For Proposal (CFP), requesting the rights from the seller agent. In the CFP, the Buyer agent provides the Seller agent with the rights requested, the digital object over which the rights are requested and the environment within which the content will be used. In message 4 the Seller agent requests the Directory service for the list of policy agents and receives the list in message 5. In messages 6, 7 and 8, the Seller agent queries the policy agent and receives a list of rights offers. Here, the Policy agent

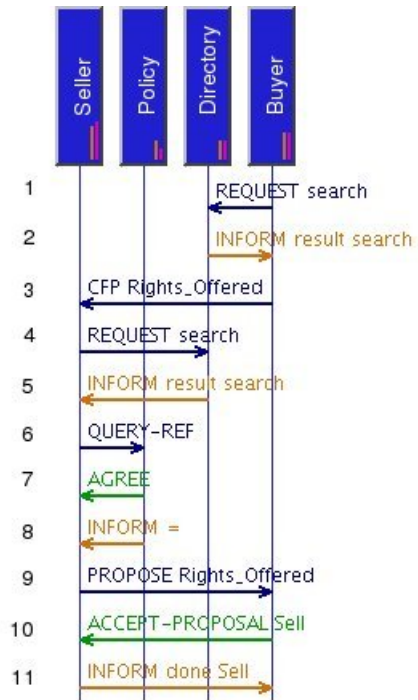


Fig. 6. Agent interactions during rights negotiations.

has access to the policies provided by the content provider, creates a set of rights offers which are closest to the Buyer agent’s request, while adhering to the content providers’ policy statements. The seller agent then proposes different sets of available rights offers to the Buyer agent. The Buyer agent then selects the best offer and sends an ACCEPT PROPOSAL message to the Seller agent. Finally, in message 11, the Seller agent informs the Buyer agent that the deal is finalized and completes the negotiation.

Although this layer has been implemented using agents to ease the development, commonly, companies do not provide agent interfaces for their business process. They usually provide a web service interface [20]. In fact, from the content distributor point of view, it need not understand the details of the implementation of the negotiation service. Hence this service can have a web service interface for seamless integration with the content distributor application. Also, from the client application point of view, the DRM process is not the main task of the application so it is an appropriate candidate for a middleware implementation.

The definition of a clear interface for the Negotiation Layer allows us to change the implementation of this layer. In particular, most of the logic of the user negotiation can be pushed to the server for devices with restricted computation power. For more powerful devices it is possible to use a personal software agent to minimize the user interaction during the acquisition of content licenses once the user provided a set of preferences to the agent.

VI. CONCLUSION

In this paper, we pointed out the need for generic middleware services, which are required for the growth of the DRM industry. To create such services we first identified processes within DRM which are common to most applications and are distributed in nature. Rather than having each DRM application implement these functions from scratch, common middleware services for these functions will provide a major boost to the growth of the DRM industry. These functions are then categorized according to the layered DRM framework. Each layer, thus provides a set of generic rights management services which can be reused by different DRM applications operating at that particular layer. Finally we implement a complex negotiation service using an agent based framework, where software agents negotiate on behalf of DRM vendors and DRM users, trying to achieve specific goals. The next step will be to integrate this service with the existing PKI and payment services to create a complex reusable negotiation system. License translation services and content usage tracking services also need to be developed along similar lines for DRM applications.

ACKNOWLEDGMENT

This research was partially supported by the National Science Foundation under grant CNS 0626380 provided through the FIND initiative.

REFERENCES

- [1] T. Clark, "IBM closes cryptolopes unit," 2002, <http://news.com.com/2100-1001-206465.html?legacy=cnet>.
- [2] M. A. Kaplan, "IBM cryptolopes, superdistribution and digital rights management," 1996, <http://researchweb.watson.ibm.com/people/k/kaplan/cryptolope-docs/crypap.html>.
- [3] O. Sibert, D. Bernstein, and D. Van Wie, "Securing the content, not the wire, for information commerce," InterTrust Technologies Corp., Tech. Rep., 1996.
- [4] "The DRM vendor graveyard," http://www.info-mech.com/drm_vendor_graveyard.html.
- [5] P. A. Jamkhedkar and G. L. Heileman, "DRM as a layered system," in *Proceedings of the Fourth ACM Workshop on Digital Rights Management*, Washington, DC, Oct. 2004, pp. 11–21.
- [6] G. L. Heileman and P. A. Jamkhedkar, "DRM interoperability analysis from the perspective of a layered framework," in *Proceedings of the Fifth ACM Workshop on Digital Rights Management*, Alexandria, VA, Nov. 2005, pp. 17–26.
- [7] R. H. Koenen, J. Lacy, M. MacKay, and S. Mitchell, "The long march to interoperable digital rights management," *Proceedings of the IEEE*, vol. 92, no. 6, pp. 883–897, 2004.
- [8] B. Aiken, J. Strassner, and B. Carpenter, "Network policy and services: A report of a workshop on middleware," Network Working Group, Request for Comments: 2768, Tech. Rep., 2000.
- [9] B. Rosenblatt, B. Trippe, and S. Mooney, *Digital Rights Management: Business and Technology*. New York, NY: M&T Books, 2002.
- [10] R. Safavi-Naini, N. P. Sheppard, and T. Uehara, "Import/export in digital rights management," in *Proceedings of the Fourth ACM Workshop on Digital Rights Management*, Washington, DC, Oct. 2004, pp. 99–110.
- [11] R. Iannella, "Open digital rights language (ODRL), Version 1.1," <http://odrl.net/1.1/ODRL-11.pdf>, Aug. 2002.
- [12] "XrML 2.0 technical overview, version 1.0," www.xml.org/reference/XrMLTechnicalOverviewV1.pdf, Contentguard, March 2002.
- [13] A. Arnab and A. Hutchison, "Fairer usage contracts for DRM," in *Proceedings of the Fifth ACM Workshop on Digital Rights Management*, Alexandria, VA, Nov. 2005, pp. 1–7.
- [14] N. R. Jennings and M. Wooldridge, *Agent Technology: Foundations, Applications and Markets*. Springer, 1998.
- [15] C. Bartolini, C. Preist, and N. R. Jennings, "Architecting for reuse: A software framework for automated negotiation," in *Agent-Oriented Software Engineering*, Bolonga, Italy, 2002.
- [16] A. Chavez and P. Maes, "Kasbah: An agent marketplace for buying and selling goods," in *First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Systems*, London, UK, 1996.
- [17] F. Bellifemmine, A. Poggi, and G. Rimassa, "Jade: A fipa compliant agent framework," in *Proceedings of the 1st Joint International Conference on Autonomous Agents and Multi-Agent Systems*, Bolonga, Italy, 2002.
- [18] "Protege," <http://protege.stanford.edu/>.
- [19] "FIPA contract net interaction protocol specification," <http://www.fipa.org>.
- [20] E. Cerami, *Web Services Essentials*. O'Reilly, 2002.