The *DiMaS* System for Authoring Communities: Distributing Semantic Multimedia Content on Peer-to-Peer File Sharing Networks

Tommo Reti and Risto Sarvas

Helsinki Institute for Information Technology HIIT
P.O. Box 9800, FIN-02015 HUT, Finland
{tommo.reti, risto.sarvas}@hiit.fi
http://www.hiit.fi/

Abstract. This paper presents the Digital Content Distribution Management System (*DiMaS*). *DiMaS* proves as a concept that it is possible to make a system for multimedia producers to publish their work on highly popular P2P file sharing networks, and importantly, the system enables producers to insert content metadata, to manage intellectual property and usage rights, and to charge for the consumption. All this can be done *without* introducing another new content or metadata file format and a dedicated client application to read the format. This example technical implementation of *DiMaS* is specified for a micromovie producing community that creates short movies for handheld devices, and in this particular case for a PDA with network access and a Java environment.

1 Introduction

As media recording devices and editing tools have become widely popular[8], multimedia is produced more and more by amateurs and home users. Also, the traditional distribution channels, even the client-server architecture of the World Wide Web, have lost popularity to the peer-to-peer (P2P) file sharing networks.[4][11] However, the file sharing networks are notorious for not supporting user rights, pricing, or rich content description.[7]

This paper describes the multimedia distribution system *DiMaS* for amateur multimedia authoring communities who want to publish their creations to a wide audience on P2P file sharing networks and still hold on to certain rights and charge for the consumption. The system also takes into account the requirements for describing the multimedia content to help the audience finding the kind of multimedia they would enjoy. The example multimedia producing community in this research is a micromovie producing community that creates short movies for handheld devices.[3]

1.1 Peer-to-Peer File Sharing Networks

Highly popular P2P file sharing networks offer an alternative approach to distribute information products such as multimedia content. Unlike distributing information products in a central server architecture, decentralized P2P networks offer high availability, better bandwidth through many users' wideband connections, and better scalability without central servers as bottlenecks. However, while being mainly a channel for distributing files, P2P file sharing networks lack traditional web pages' way of describing the content alongside the downloadable file itself. On many file sharing networks the content search is still limited to file names that can be very misleading or otherwise inefficient.[7][19] Nevertheless, the impact of P2P in file sharing is significant (see, *e.g.*, [17]): hundreds of millions of copies are already in use making file sharing the most popular application on the World Wide Web.[4] Therefore, P2P file sharing networks create a huge content sharing base and a complex value network, which offers new business opportunities and models to various actors, such as content right owners and network operators, and also, to novel actors like multimedia producing individuals and communities. This potential could be utilized better, if there was a way to include rights, pricing, and content descriptions to enrich searchability and controllability.[12][9][20]

1.2 Multimedia Authoring Communities

The popularity of digital media recording devices and media editing tools has enabled amateurs to generate professional looking multimedia. Examples of these are computer game modifications (*i.e.*, "mods") (see, *e.g.*, [16]), digital image manipulation competitions, and home made movies that have acquired large audiences through "word of mouth" on the Internet. In other words, the availability of multimedia authoring tools and the rather unconstrained distribution on the Internet has enabled non-professionals to produce high-quality digital content for others to enjoy. These people have formed meeting places on the Internet by creating various virtual communities that share knowledge and authoring tools.

These multimedia producing communities are looking for alternatives to the traditional media publishing and distribution channels which are hindered by cumbersome intellectual property, organizational, and marketing practices. On the other hand, the community members do not want to lose control over all of their rights to their own works (*e.g.*, author attribution, or right to use the work commercially). Also, some of the communities and individual creators are interested in getting a compensation for their work through charging mechanisms.

The example community we use in this paper and with the *DiMaS* system is a micromovie producing community that creates short movies for handheld devices such as PDAs or mobile phones.[3]

1.3 Multimedia Content and Metadata File Formats

To distribute multimedia content on any digital channel there is the issues of file formats and compatibility – for both the content itself and the associated metadata. Multimedia authoring communities want to distribute different kinds of multimedia in all kinds of file formats to reach as wide as audience as possible and also the authoring tools they use vary a lot. However, the lack of common standards, digital rights management systems, and software companies' race for the dominant market leader position have created a cluttered selection of multimedia file and metadata formats. Even among the same media, like movies, there are numerous different file formats and within the file formats several different codecs. In addition, many of the most popular file formats, such as MP3, are not the most effective ones in compression or content descriptions. Thirdly, hardly any of the multimedia file formats support rich rights, pricing, and content descriptions. Although there are standardized and popular de facto formats for describing user rights and licenses (*e.g.*, Creative Commons[6]), pricing (*e.g.*, ODRL[14]), and multimedia content (*e.g.*, MPEG-7[10]), these languages address only certain issues and there is no widely accepted standard that addresses all of the issues. Therefore, in addition to the content file, there are often several metadata description files associated with it.

To summarize, the content producers face the problem of having several different file formats to support and the same problem becomes worse as the multimedia is described with content, rights, or pricing metadata. In this paper and in the *DiMaS* system we focus on the latter issue of having several metadata descriptions.

1.4 Research Problem and Solution

The multimedia authoring communities want to distribute their creations to a large audience which P2P file sharing networks can provide. However, the P2P networks do not support the use of metadata for digital rights management, charging mechanisms, nor content descriptions. Having rights, pricing, and content describing functionalities in multimedia is often solved by introducing a new file format and a respective multimedia player that supports the new format.[1] This would add another file and metadata format to the already large pool of different formats for multimedia, and would require the user to install another dedicated player to view the multimedia. Also, different authoring communities might have different requirements for the metadata descriptions, for example, due to local legal regulations or the nature of the multimedia produced.

Therefore, before the P2P file sharing networks can really be harnessed and leveraged by multimedia authoring communities there are several problems to be addressed: How to facilitate the user input of content descriptions as well as rights and charging data into standardized metadata? How to make these descriptions compatible and comparable? How to bundle the metadata and the actual content into one package rather than having them in separate files? How to make this package easily accessible by the end-user without creating a new file format and another player application?

The *DiMaS* system presented in this paper provides one solution to these issues. It has a configurable user interface for inserting rights, pricing, and content information, which are automatically converted into user-selected metadata standards. The metadata and the multimedia content itself are combined into one package that is shared on P2P file sharing networks to gain as large audience as possible. The package is a Java file that can be executed on any device that has a standard Java Runtime Environment, therefore, the package does not require any dedicated player application. The rights and pricing requirements are taken into account by encrypting the actual multimedia content and control files of the package, and implementing an authentication and charging infrastructure into the consumption of the package.

2 System Design

The *DiMaS* system relies on an extendible and modular architecture. As the system is further developed, various modules are added which increase the functionality of the system. To support maintainability, the system's module interfaces are carefully defined and documented. By dividing the system into smaller parts, they can be independently designed, and most of all, independently implemented.

There are still some designed features which have not been implemented and open issues which have not been solved. Therefore, it is important to isolate the different modules and problems from each other, so that the basic framework of the system remains functional. This modularity makes also possible to use and utilize third party code, such as open source software. The objective of the implementation is to use open-source software so that the *DiMaS* system itself can be distributed as open-source. *DiMaS* has also been produced using open-source development tools.

Figure 1 shows the main functional parts of *DiMaS*:

1) Content Import User Interface,
2) Publishing System,
3) Distribution Package, and
4) Verification Server.

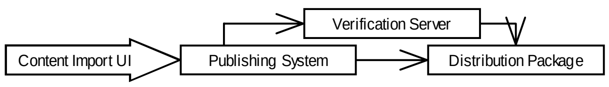

Fig. 1. The main functional parts of the *DiMaS* system.

The multimedia author uses the Content Import User Interface to import the actual multimedia content (*e.g.*, a micromovie file) with an associated optional preview file (*e.g.*, a movie trailer or a still image), content, rights, and pricing information, and an optional feedback question. The same user interface enables the user to modify and publish the imported content. Once the user chooses to publish the content (the imported content needs to be explicitly published before distribution), the inserted information is processed by the Publishing System. The Publishing System is responsible for converting the inserted information into standardized metadata, and it also does the content encryption, hash key generation, and finally the creation of the

Distribution Package that is made available on the P2P file sharing networks. Also, the Publishing System informs the Verification Server of the newly created Distribution Package so that the Verification Server can handle the authentication, billing, metadata updates, user statistics, and feedback gathering once the Distribution Package is executed. The Distribution Package contains the content file, preview file, inserted metadata, authentication information, and the user interface graphics. The Distribution Package also contains the execution logic for the user interface, *e.g.*, contacting the Verification Server, and decrypting the content file. The actual compatible player that shows the included multimedia content file is called by the Distribution Package from the local computer's available players.

The *DiMaS* system has been tuned for high portability. It is independent of multimedia content file format, it allows different kinds of payment systems and the viewer is independent of the end-terminal. Security requirements are taken into account by having the decrypted content only in the main memory of the device. Also, the user is not able to modify the encrypted verification information, and the user always receives the correct verification information intended to the content file of the Distribution Package.

In the following subsections we go into more detail in the actual implementation and rationale for each of the four main functional parts mentioned above.

2.1 Use Case

A content producer browses via PC onto the main page of the content production community's *DiMaS* system, where he registers into the system. After registration he logs on and wants to import a micromovie that he has created on his PC. He goes through the steps of the content description import pages (*e.g.*, he inserts his name as the director of the movie, a text description of the movie, he selects that it is suitable for children, he inputs the information that it is free of charge, and the right to make derivative works of the movie), and at the end approves the inserted information and media files. Looking at his personal page he can see the names of the imported content. He decides to publish the newest content with the descriptions he just inserted and the preview file he uploaded. The system announces that the publishing is successful and a Distribution Package is created successfully. The content producer logs off.

After this, another user, a consumer, browses via PC the content selection on the Content Search page. She tries several different search categories (*e.g.*, action and video), browses descriptions of the contents, checks one micromovie preview, and finally decides to download an actual micromovie (*i.e.*, Distribution Package). When she has finished downloading, she moves the content to her attached PDA. Later, she opens the micromovie on her PDA with one click. The user interface, which is similar to a DVD menu (see Fig. 4), opens and it asks if the user wants to update the content metadata. This means that the Distribution Package has found a network connection and offers to update the content description. The user browses through the content description, including the pricing information. Before a purchase decision she wants to check the preview again, and after that she buys the content by pressing the "Buy" button on the user interface. The PDA retrieves a decryption key from the Verification Server and informs of the successful purchase. The user notices how the price page shows the new licenses she just bought, *i.e.*, the number of times the user can view the micromovie, and the play button activates from gray to green. The user presses the "Play" button and waits for the content to decrypt. After watching the micromovie, the user fills in a feedback form, which is submitted to the Verification Server at once, and the metadata on the PDA is updated considering the new feedback, (*e.g.*, the rating can be changed due to the user's feedback). The user closes the Distribution Packet on her PDA.

2.2 Content Import User Interface

The Content Import User Interface of *DiMaS* has been targeted for the multimedia authoring community that wants to use the system to distribute their creations on P2P file sharing networks. Our example case is a micromovie producing community. An actual micromovie community *Blauereiter.net*[3] was consulted in the design and the specification of the system.

As the user logs into the system he is presented with a list of all content he has either imported or published. The user can modify imported content and its information as long as it is not published, in other words, created into a Distribution Package. Once the content is published, only the metadata can be updated as a part of the authentication procedure.

The main feature of the Content Import User Interface is to provide a dynamic web page interface for the multimedia authors to import their multimedia content into the system and to insert descriptive information about the content as well as information about user rights and pricing. The word *dynamic* here means that the import interface adapts according to the type of content (*e.g.*, a movie, a picture, a song), user profile of the importer (*e.g.*, a specific group, power users, administrators), the general requirements of the specific community (*e.g.*, community pricing or rights policy), and information inserted (*e.g.*, if the user selects that a movie is unsuitable for children, the user needs to specify why).

The content importing is divided into several pages to enable the dynamic adaptation of the importing procedure (see Fig. 2). The user can move back and forth between the pages as in several web store shopping cart implementations. The pages are generally categorized so that the content and preview files are uploaded first. Based on the type of content, the user is then presented with the content description pages, where the most relevant information is presented first. After the content descriptions the user is asked to describe the user rights for the content, and then in a similar fashion the pricing policy for the content. All of these pages can be configured to fit the requirements of the multimedia authoring community. Especially, the pricing information can be a selection of choices or even a default charging policy of the whole community. Also, the rights can be, for example, Creative Commons licenses[6] which can be chosen by the user or automatically follow a community rights policy.

Fig. 2. The chain of the Content Import User Interface web forms for inserting the content, rights and pricing information, and an optional feedback question.

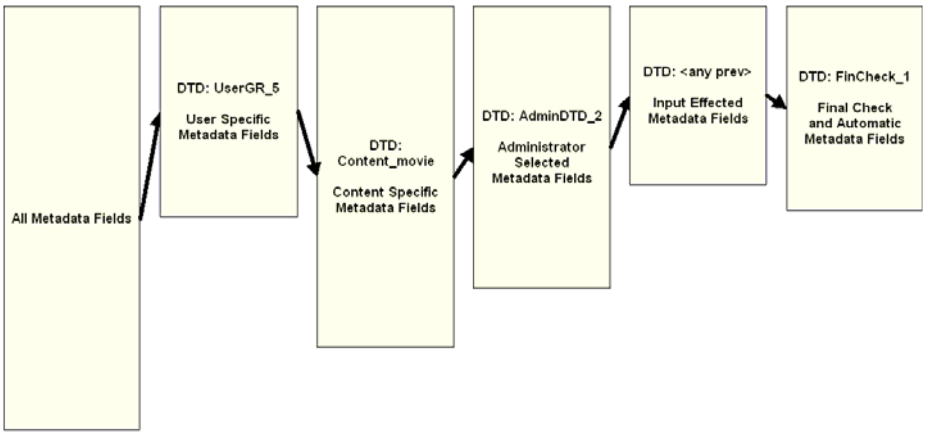
All Metadata Fields
DTD: UserGR_5
User Specific Metadata Fields
DTD: Content_movie
Content Specific Metadata Fields
DTD: AdminDTD_2
Administrator Selected Metadata Fields
DTD: <any prev>
Input Effected Metadata Fields
DTD: FinCheck_1
Final Check and Automatic Metadata Fields

Fig. 3. Visualization of how the length of the presented metadata fields changes on the way of the content import process.

The implementation of the dynamic user interface is based on a chain of DTD like XML files used to configure the interface for each import (see Fig. 3). The chain of configuration files acts as a filter that determines which metadata fields out of all possible ones are actually presented to the user for insertion. The first configure file defines user specific metadata fields, the second file defines content specific metadata fields, the third file defines metadata fields and values required by the administrator, the fourth file defines the metadata fields that are affected by previously input information, and the last file defines automatic metadata fields. After processing through all these files, the system presents the user the insertion page that has the appropriate metadata fields.

The insertion page layout takes also into account the nature of the information asked from the user. The information (*i.e.*, metadata) has been arranged according to the following classification, which has been taken into account in the parsing and validation of the information:

- Automatic information, not imported by the user, such as file size.
- Necessary information, especially highly visible information that is used often to describe the content.

- Additional information that is not necessary.
- Clear and interpret-free information, such as length of a movie.
- Author’s opinions.
- Quantified information, can be used for indexing and statistical calculations.
- Free-text information.

As the implementation suggests, the administrator of the system acts on behalf of the community, especially in the case of default or forced charging or rights policies which can be configured in the third configuration file.

After filling in the fields (*i.e.*, inserting the information), the user is redirected to a verification page that verifies information and summarizes all imported data for the user to confirm. Before the summarization of the inserted information is shown to the user, it is parsed, validated, and stored into a special datamap. The datamap is the metadata format used inside the system. Once the user confirms that all information is correct the content files and the datamap are stored into a database.

Now the user can see the newly imported content in his list of all imported content. From the list the user can select a content file which has not been published and upload a new version of it, change the preview file, and modify the content, pricing and rights information. From the list the user can also select imported content for publishing. The publishing activates the Publishing System described in the next sub-section.

2.3 Publishing System

The Content Import User Interface has received the inserted information, parsed it into a datamap, and validated that the datamap against the system’s policies. When the inserted content - and the associated datamap - is published by the importer, the datamap is forwarded to the Publishing System. The Publishing System executes a process invisible to the user. It processes the datamap, including the actual content file, and the optional preview file into a Distribution Package and feeds the Verification Server with appropriate data. In the order of execution, the Publishing System part of *DiMaS* has the following functions:

1. Encrypt the actual multimedia content file.
2. Create a hash of the encrypted multimedia content file.
3. Transcribe the metadata in the datamap into a standardized XML document or to a XML metadata description.
4. Upload 1) the XML metadata description, 2) the hash, and 3) the decryption key to the Verification Server.
5. Produce the Distribution Package.
6. Deliver or upload the produced Distribution Package to administrator listed locations on the server and on file sharing networks.
7. Add 1) the new XML metadata description to the Content Index database with 2) the preview and 3) the location where the Distribution was uploaded to.

The current encryption module of the *DiMaS*'s Publishing System encrypts the content file using the AES encryption with 128 bit long key. The encryption key strength can be chosen separately for each different content type, if needed. By using the Strategy design pattern the system is able to change the encryption algorithm even at run-time.

The Hash Generator module calculates the unique identifier for the encrypted content file by which it will be identified later. The identifier is needed for distinguishing the particular Distribution Package among others, for example, when fetching a decryption for it. Also, the hash algorithm can be changed at run-time. Currently, *DiMaS* uses the SHA-256 hash algorithm.

The Metadata Generator module is responsible for extracting the information concerning the content from the datamap and collecting this into a structure that can be easily used by the other parts the system. The processing of the metadata is divided into two parts: the content description metadata, and rights and pricing information which will later be presented in rights expression language ODRL[14]. The deliverable of the Metadata Generator module is a special data-container class Content Description, which holds all the information of the content in an accessible form. Since the Content Description object contains all the relevant information referring to a specific package, including encryption key and hash identification, it can alone represent the content.

The Metadata Generator module does not process or validate the content in any way; it only extracts the values from the datamap to the Content Description object. Validation has occurred already in the Content Import UI part, so that only meaningful and useful data is stored to the system. The structure of the metadata in XML format is a set of name-value pairs with an optional description. This makes possible to keep Metadata Generator modules intact when changing the format of the metadata.

The ODRL information (*i.e.*, the rights and pricing information) is contained in a tree-like structure of special classes corresponding to the different supported elements of the ODRL language. This approach is chosen to ease the task of parsing ODRL information in XML format into a structure that can be handled by the system. The most important object within the ODRL structure is the Offer object which describes the rights the user can be given with one purchase transaction and the price of these rights. This Offer object contains one or several Permission objects, which describe the actions allowed on the content (*e.g.*, play, save, print) and also the price for these actions. Permission object can also contain objects called Restraints, which may limit usage to a certain number of times or to a certain timeframe.

2.4 Distribution Package

The multimedia content authors want to make the viewing or consumption of the multimedia as easy as possible for the consumers. As mentioned previously, one way to do this is to produce multimedia that can be consumed without a dedicated client application.

At the same time, the content authors want to improve the searchability of the content by adding descriptive metadata, and to retain control over their content in the form of licenses, user rights, and usage charging. This requires interpretations skills from the application utilizing the metadata. In this sense, an application may differ in many various ways:

If an application understands only one description language, the handicap is that there is not a single description language, which would satisfy all the needs. If an application supports several description languages, which languages would they be? Some applications support only reading of the selected file properties, but not the separate metadata files associated with the content file.

Traditionally these challenges have been met by creating an end-to-end channel from a publishing system to an end-user application that is downloaded from the publisher's web site before downloading the multimedia content. Through the publisher's own end-user application many of the metadata compatibility issues can be tackled by encoding the needed information inside the application.

Distribution Package is a concept of *DiMaS* that tackles this issue. It features a user-friendly DVD-like graphical interface for consumers to browse the content metadata, for example, regular descriptive fields as content name and author, but also information about content pricing and the usage rights to consume the content. The information in the metadata is meant to encourage the consumer to make the decision to purchase the actual content, so it is possible to include even a separate preview file like a movie trailer. Distribution Package doesn't require a separate client program. It uses the Java-enabled run-time environment. The system has been successfully ported to a standard wireless HP iPAQ Pocket PC model H4150 (see Fig. 4) with the Java Standard Edition[18] Virtual Machine by NSIcom CrEme[13].

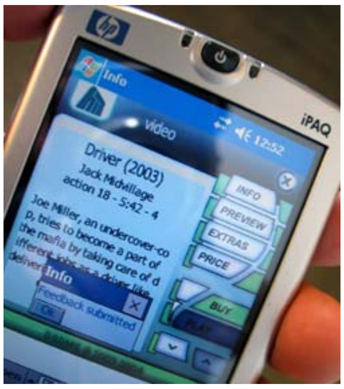

Fig. 4. Distribution Package on a PDA.

The distribution package is a JAR file (Java Archive) which contains the following parts (see Fig. 5):

- Encrypted multimedia content file. The content file is included unmodified after encryption. It is identified by having the name `content.file`.
- The Unencrypted content preview is included unmodified and stored under the name `preview.file` in the archive.

- Hash ID of the actual content file. It is used especially for fetching the right decryption key.
- Content description metadata. The content description is included as a simple text file in XML format describing the content. The file is identified by having the name `description.file`.
- Feedback questions to be presented to the end-user.
- There is a reserved option to add also various “extras” into the Distribution Package, for example, links, still images, or advertisements.
- Client viewer program or logic. The client consists of Java class files and resources needed in the user interface. Compared to average content files, the logic does not occupy much space. Currently it is under 100 kB.

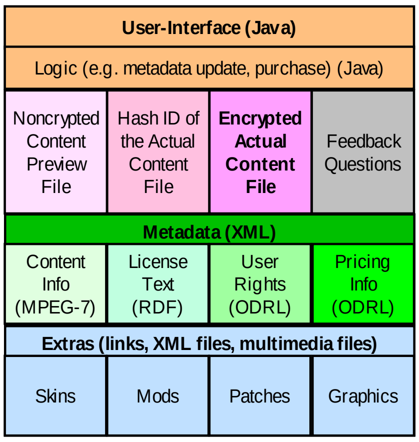
User-Interface (Java)
Logic (e.g. metadata update, purchase) (Java)
Noncrypted Content Preview File
Hash ID of the Actual Content File
Encrypted Actual Content File
Feedback Questions
Metadata (XML)
Content Info (MPEG-7)
License Text (RDF)
User Rights (ODRL)
Pricing Info (ODRL)
Extras (links, XML files, multimedia files)
Skins
Mods
Patches
Graphics

Fig. 5. The structure of Distribution package.

As the JAR file (*i.e.*, the Distribution Package) is executed by the user it opens up the viewer program embedded in the Distribution Package. The viewer user interface resembles the menus on DVD movie discs. Once the viewer is launched it prompts the user to update the metadata of the package. This is done only if there is a network connection available. On the user interface the user can browse through the content information, as well as pricing and rights information. The user can also watch the included preview file. If the user wants to purchase viewing rights to the content (unless the content can be viewed free of charge), the user selects the “Buy” button. The Distribution Package retrieves a decryption key from the Verification Server and informs the user of the successful purchase. The user also notices how the price information shows the new viewing rights that were bought (*e.g.*, the number of times the user can view the multimedia content). Now the “Play” button changes from gray color to green indicating the possibility to view the content.

The user presses the “Play” button and waits for the content to decrypt. After watching the micromovie, the user fills in a feedback form, which is submitted to the Verification Server at once, and the metadata in the Distribution Package is once again updated taking into account the new feedback (*e.g.*, the rating can be changed due to the user's feedback). If the network connection is not available for some reason, the feedback is transferred to the Verification Server next time the user executes the same Distribution Package.

The overall goal in the user interface design has been to make it as simple to use as possible. The objective is that the user doesn't need to have anything related to *DiMaS* on his device before downloading and using a Distribution Package, except the above-mentioned Java environment, and the application needed to view the actual content (*e.g.*, a Windows Media Player or a Quicktime Player). Fetching a decryption key from the Verification Server, the actual decryption of the content and launching the execution of the content is invisible to the user and happens right way after the purchase event.

2.5 Verification Server

When a Distribution Package is executed on an end-terminal, it checks for a network connection and if a network connection is available it connects to the Verification Server part of the *DiMaS* system. The user is asked whether he wants to utilize this network connection and update the metadata. This means that the Distribution Package offers to update the content description of the Distribution Package. This way even if the Distribution Package had been stored on various locations on a network for a long time, the content description can be updated to reflect the newest possible modifications the content author or the publisher have made. These modifications can include, for example, a notification of a newer version of the content, a sequel, a discount in pricing, or any other important matters which may affect the consumer's behavior and willingness to buy the content.

The Verification Server is the system part that is responsible for the communication with the Distribution Package. In a way, it glues together the user-end and the Content Import User Interface, even if the package has been traveling uncontrolled on file sharing networks. It also facilitates payment system, if available, and controls the integrity and validity of the distributed content.

The communication is done through the public Internet and it uses the hypertext transfer protocol (HTTP) with SSL encryption and a XML based documents for information exchange. The Verification Server's tasks include answering to update requests of the Distribution Package's metadata, receiving and storing user feedback, and handling purchases by providing decryption keys for payment. Since it distributes decryption keys, it is critical from the commercial and security aspects of the system. However, currently the Verification Server is not designed to handle the billing logic of the system. Therefore, billing would require another component which is not within the scope of the *DiMaS* system for now.

The Verification Server's database provides persistence for the system. It is implemented as a standard relational database and accessed through SQL queries over the JDBC protocol. It may therefore reside on another host and accessed through a network. As the Distribution Packages are identified within the *DiMaS* system by their hash value, the information in the Verification Server is also structured according to it.

The structure of the database divides into four parts:

- The content information. This includes the metadata, encryption and pricing information.
- A transaction log. All purchases (both unconfirmed and confirmed) are recorded here.
- An event log. All events (updates, purchases) are recorded here.
- A feedback storage. Stores the received feedback.

The transaction and event logs offer an opportunity to make statistical profiling and analysis of the consumer behavior. These calculations could be directed back to the system to advise multimedia seeking users to make more pleasing content selections based on a recommendation engine for example.

3 Future Work

Awareness of the need for richer semantic descriptions for content and services is growing in the business world and within commercial Web services. The cross-organizational, collaborative, and internetworked nature of business today increasingly calls for an instant access to data on other organizations. This requires that organizations have more transparent view of data, information, and processes of their partners.

Web services have emerged as an interesting novel technology for designing collaborating information systems on the Internet.[2][5] Rightly used they offer interoperability across platforms and systems.[15] They are neutral to description languages offering standardized messages over system interfaces. This makes them suitable for accessing various connecting systems over heterogeneous environments.

However, this also implies for a more reliable and immediate licenses or contracts on how the transactions of the two parties are to be conducted. Contracts are still treated mostly as legal documents, not relating to user rights or enterprise information systems, even if they are a primary mechanism for defining interactions and policies for co-operation. The modern mechanism for business collaborations may be lacking both support for user rights and licenses, and other contract information, and a system to manage these cross-organizational collaboration contracts. The capabilities of new technologies such as bundled standard content descriptions and common Web services interfaces may enable new solutions for governing contracts, but also as catalysts for more internetworked collaboration they require novel approaches to metadata managing. The future versions of *DiMaS* will consider in more detail various aspects of electronic contracts as a part of metadata, such as enterprise modeling, e-business, and current legal aspects, extending the use of the system to cross-organizational collaboration.

Furthermore, Web services may offer an interesting solution for binding numerous system input sources to the same interface and with the services the system provides. When extending the data insertion beyond the introduced web form interface, multimedia content can be expected to come from various devices, such as PCs, PDAs, digital cameras and camcorders, and camera phones, still in various media formats, with or without metadata that can also be in any format. Multimedia content can be forwarded from various systems, like proprietary content management and multimedia archiving systems, which may offer their own Web services interfaces. Also, metadata insertion may have already happened at a number of different phases, even before the actual content has been created, or it could be situated in a different place than the content, *e.g.*, in the case of a PC storing images from a digital camera.

In the future, the overall security of any multimedia distribution system may be enhanced with a separate integrated memory chip in the user's end-terminal that can be used, *e.g.*, for storing authentification information and decryption keys. Since the *DiMaS* system uses external programs to execute the actual content file, this naturally creates potential vulnerability. From this part the system could be improved by a closer integration of the actual content file and the player used on the end-terminal. If there was a media player with a streaming interface, *e.g.*, a server socket, to which the content could be fed, this would make possible to keep the major part of the actual content file encrypted, only some of it in the main memory of the device while playing the stream, and also it would enable to start playing the content quicker.

4 Conclusions

This paper presented the technical implementation of the Digital Content Distribution Management System (*DiMaS*). *DiMaS* proves as a concept that it is possible to make a system for multimedia producing communities to publish their work on highly popular P2P file sharing networks and the system enables them to charge for the consumption and manage their intellectual property rights, as well as have content describing information available. In the technical implementation of *DiMaS* a micromovie producing community was used and consulted as an example. The client device that was used for testing was a PDA (HP iPAQ) with network access and a Java SE environment.

The key benefits of the *DiMaS* system for content provider were as follows:

– It is not necessary to limit or to restrict the distribution of the content, but encourage distribution even in the uncontrolled file sharing networks.
– When the content is independent of distribution channels, it can be brought closer to consumers.
– It is possible to utilize different kinds of payment systems, and not be dependable on a single system that may introduce technical constraints.

The key benefits of the *DiMaS* system for the content consumer were as follows:

– After downloading the content, a network connection is necessary only for paying. If the paying is done in advance there is no requirement for a connection.
– The consumer can view the multimedia content without installing another new dedicated client application that understands the content descriptions.
– Content consumer does not need any special skills to view the content.
– The content import process creates informative and browsable metadata enriched content files which enable more efficient searching and browsing.

5 Acknowledgements

We wish to thank H. Hallantie, A. Illukka, T. Kiuru, J. Koskela, H. Koskinen, T. Perälä, M. Rosendahl, I. Rämö, and J. Välikylä for the technical implementation aid, and the Finnish National Technology Agency in line with participating companies who have generously funded our research work. We also wish to thank M. Kailu and P. Keränen from the *Blauereiter.net* community for their expertise. This research is part of the Digital Economy Core (DE Core) project and the Mobile Content Communities (MC2) project at the Helsinki Institute for Information Technology HIIT.

References

1. Apple iTunes Music Service, http://www.apple.com/itunes/
2. Barry, D.K.: Web Services and Service-Oriented Architectures: The Savvy Manager's Guide, Morgan Kaufmann Publishers (2003)
3. Blauereiter.net, available at http://www.blauereiter.net/
4. Borland, J.: File swapping shifts up a gear, News.com http://news.com.com/File+swapping +shifts+up+a+gear/2100-1026_3-1009742.html?tag=mainstry
5. Clark, M., Fletcher, P., Hanson, J.J., Irani, R., Waterhouse, M., Thelin, J.: Web Services Business Strategies and Architectures, Expert Press Chicago (2002)
6. Creative Commons Web Site, http://www.creativecommons.org/
7. Good, N., Krekelberg, A.: Usability and Privacy: A Study of Kazaa P2P File Sharing, Proceedings of the CHI 2003 ACM Press (2003)
8. Henning, T.: The camera-phone phenomenon, The Future Image Report (2003)
9. Loser, A., Naumann, F., Siberski, W., Nejdl, W., Thaden. U.: Semantic Overlay Clusters within Super-Peer Networks, Proceedings of the International Workshop on Databases, Information Systems and Peer-to-Peer Computing in Conjunction with the VLDB (2003)
10. Martínez, J.M., Koenen, R., Pereira, F.: MPEG-7: The Generic Multimedia Content Description Standard, Part 1. IEEE Multimedia 9(2) (2002)
11. MP3 Search Tools, download.com http://www.download.com/3150-2166-0-1-5.html
12. Nejdl, W. et al.: EDUTELLA: A P2P Networking Infrastructure based on RDF, Proceedings of the 11th World Wide Web Conference (2002)
13. NSIcom Web Site, http://www.nsicom.com/products/creme.asp
14. ODRL Web Site, http://www.odrl.org/
15. Papazoglou, M., Yang, J., Kramer, B.: Leveraging Web-Services and Peer-to-Peer Networks. Technical Report DIT-02-088 University of Trento (2002)
16. Sotamaa, O.: Computer Game Modding, Intermediality and Participatory Culture, http://www.imv.au.dk/eng/academic/pdf_files/Sotamaa.pdf
17. Strumpf K., Oberholzer F.: The Effect of File Sharing on Record Sales: An Empirical Analysis, working paper at www.unc.edu/~cigar/papers/FileSharing_March2004.pdf
18. SUN Java J2SE Web Site, http://java.sun.com/j2se/
19. Yang, B., Molina, H.G.: Improving Search in Peer-to-Peer Networks, Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS) Vienna Austria (2002)
20. Yee, K-P., Swearingen, K., Li, K., Hearst, M.: Faceted Metadata for Image Search and Browsing, Proceedings of the CHI 2003 ACM Press (2003)