2011 31st International Conference on Distributed Computing Systems

Meeting the Digital Rights Requirements of Live Broadcast in a Peer-to-Peer Network

Wenjie Wang*, Hyunseok Chang†, Adam Goodman‡, Eric Wucherer‡, and Sugih Jamin‡
*IBM Research China, Email: wenjwang@cn.ibm.com
†Alcatel-Lucent Bell Labs, Email: hyunseok@ieee.org
‡University of Michigan, Email: {akgood, ewuch, jamin}@eecs.umich.edu

***Abstract*—Live broadcast over a P2P (peer-to-peer) network imposes a unique set of challenges to a digital rights management system. Highly correlated service request arrivals at the start of a live event require peak-load provisioning if clients acquire licenses at playback time. Distributing the license management load across a P2P network requires the digital rights management system to ensure the integrity of both digital rights, the protection of client privacy and, at the same time, system scalability. In this paper we describe the requirements imposed on a digital rights management system in distributing live broadcast over a P2P network and present our design of such a system to meet the above challenges. We discuss the system's operation under a number of threat models and how to extend the system to further improve scalability. We close the paper after presenting some scalability results collected from a production P2P live broadcast network using our DRM design.**

***Keywords*-DRM; live broadcast; P2P; privacy protection.**

I. Introduction

Two basic characteristics of live broadcast fundamentally set its digital rights requirements apart from those of archival content delivery systems such as video-on-demand, web streaming, or progressive download. First, digital rights management (DRM) for archival content is usually provisioned at the unit granularity of a file, where a file could contain a single song, a single episode of a television serial, a single movie, etc. We call DRM systems that discretize contents into files "traditional DRM". In traditional DRM, each client is required to acquire a separate playback license for each file. The acquisition of playback license usually occurs right before the playing back of a file [1], [2], [3], [4]. Second, consumers of archival content are assumed to access the content in an asynchronous manner, such that service request arrivals are not highly correlated.

In contrast, broadcast content, such as live sports, breaking news, interactive talk shows or game shows, etc. is expected to be streamed shortly after capture, without the benefit of being delimited into distinct file units. Further, live broadcast events usually have well-defined start and end times. The "live" nature of broadcast events leads to the licensing of event accesses, instead of file accesses. More generally, both live and archival content may be linearized together into a single broadcast channel, similar to traditional television channels. Several P2P networks have been deployed in the past few years to provide broadcast delivery of linear channels, for example, Joost, Livestation, Octoshape, PPLive, PPStream, RawFlow, SopCast, TVAnts, UUSee, and Zattoo, to name a few. These systems try to provide a user viewing experience as close as possible to watching television and, in some cases, count as their customers' television broadcasters, using the Internet as an alternate broadcast medium.

Live events' having well-defined start and end times leads to highly correlated service request arrivals and departures, which translates into highly bursty peak service load. Provisioning for such high-peak bursty load results in inefficient resource utilization. A common mechanism to smooth out large bursts of arrivals is to introduce service delay, to spread out the requests over time. While this mechanism may be applicable to archival content that has no "liveness" constraint, it could cause a broadcast system with delay-constrained service level agreements to violate its contractual obligations [5]. An advantage of P2P systems is that large bursts of arrivals are naturally handled by the peers. Instead of limiting scalability, highly correlated viewing behavior gives P2P systems their competitive advantage over traditional client-server model. Our challenge is to design a distributed DRM system that takes advantage of P2P distribution without compromising the level of protection provided by traditional DRM.

Our design of such a distributed DRM system has been vetted and accepted by a number of large public broadcasters and cable system operators. It was implemented and deployed on a P2P live streaming network carrying over 200 channels and served live broadcast to over 3 million registered users, with over 60,000 concurrent users at its peak [6].

The two fundamental requirements of our DRM system are: (1) to provide digital rights protection to linearized broadcast channels, and (2) to provide digital rights protection when content is distributed peer-to-peer. The contents

This work was done when authors Chang and Wang were at Zattoo Inc.

1063-6927/11 $26.00 © 2011 IEEE
DOI 10.1109/ICDCS.2011.32

that make up a linear channel often come with different distribution rights similar to that of traditional cable networks. Membership of a P2P network changes as peers join and leave the network. Both of these dynamics give rise to further requirements that we present in the next section. We then describe the architecture and protocols of our DRM system designed to meet these requirements. Following which, we evaluate how our architecture and protocol operate under two common categories of threat models and how the design can be extended to further improve its scaling characteristics. We conclude the paper after presenting some performance results showing that our system does scale well in a production P2P network streaming live broadcast channels.

II. Additional Requirements

Broadcast Restrictions. As with traditional over-the-air television broadcast, each broadcaster usually has the right to broadcast only in certain geographic region(s).[1] Thus a primary design requirement for our DRM system is the ability to restrict distribution to within given geographic bounds. Even within its geographic region, a broadcaster may not have secured the rights to distribute certain content over the Internet, thus necessitating certain programs be "blacked out" during their air times in the Internet distribution A broadcaster may also want to make certain channels subscription-only channels. These could be channels that carry premium content or higher resolution version of free-to-view channels. Our DRM must be able to restrict distribution not only to geographic regions, but also restrict distribution during certain periods of time or only to paying subscribers.

Viewing Experience. To provide a viewing experience close to that of television viewing, we designed our DRM system to be transparent to the users beyond the initial sign-on. Once users are signed on, they can access all channels available free-to-view in their geographic region and all channels for which their subscriptions are current, without further verification challenges from the system. Furthermore, the channel switching delay should be minimal, similar to TV services provided by satellite (around 3 seconds). Subscription to channel packages or individual channels, purchasing of pay-per-view programs, or topping up of user account are all assumed to take place out-of-band, for example at a service provider's web site. We will call such site the *Account Manager*.

Unique User Count. Traditional DRM systems designed for delivery and play back of archival contents place heavy emphasis on restricting the number of playbacks and the devices on which playback is allowed. In contrast, free-to-view broadcast television generally can be watched on any television that can receive the signal. To comply with regulations concerning payment of television licensing fees and copyright royalties, to enforce per-view payment of paid contents, and to track viewing rate for advertisement purposes, our DRM system is required to ensure that each user is logged in before (s)he can use the system and that an account can be used to join the same channel at most once at any given time.

Rights and Privacy Intermediation. In a P2P network where peers verify each other's credentials, it is particularly important that only absolutely necessary information be exchanged during authorization. For example, when users request access to a channel, they would have to present credentials attesting that they are authorized to access the channel; however, they should not have to reveal all the other channels for which they may also have access.

Assumptions. Both over-the-air and satellite broadcast distributions suffer from some form of signal leakage from their prescribed geographic regions. We similarly assume that some signal leakage due to the use of virtual private network (VPN) is unavoidable. We also assume that there is a secure mechanism by which a user can obtain an account and a copy of client software to access the P2P broadcast network. While we rely on secure user identification and remote attestation against tampering of client software, we do not claim to have new insights or contributions in these areas; both remain active areas of research and development and our DRM system will benefit from progress made in the field [7], [8], [9].

III. Service Description

We have described the architecture of a P2P network used to deliver live broadcast in an earlier paper [6]. Here we describe only salient features of such a network related to our design of a DRM system. We assume that each broadcast channel is carried over its own P2P overlay network. A broadcast service provider may offer its users a multitude of channels, some free-to-view, some by subscription only. The channels may be of differing service quality and may be made available to the users as part of channel bundles or individually, *à la carte*. The service provider may be present (offers its services) in multiple geographic regions. A user may roam from one geographic region to another. When a roaming user enters a geographic region, it sees only the channels offered by its service provider in that geographic region. To receive service from a provider, the user must first register with the provider out of band; for example, by signing up on a web site. Once registered, the user will be given an authentication-protected account that uniquely identifies the user.

A client can logically be a member (or a peer) of only one P2P network, and be receiving data for one channel, at any one time. Every time a client runs, as shown in Fig. 1, the user is authenticated with a *User Manager*. Once

[1] Also known as *designated market area* (DMA) in US television industry parlance.

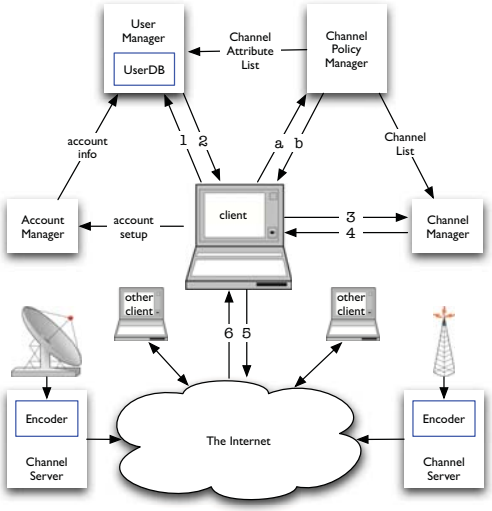


Figure 1. Summary of our DRM system for live broadcast streaming. Further explanation is available in Section IV.

authenticated, the client is granted a *User Ticket* with a limited lifetime (shown as steps 1 and 2 in Fig. 1). When a user wants to watch a channel, the client presents the User Ticket to a *Channel Manager*, which verifies that the user is authorized to watch said channel by inspecting the User Ticket. If verified, the Channel Manager returns a *Channel Ticket* to the client, along with a list of peers from whom the client can obtain a channel signal (steps 3 and 4 in Fig. 1). The client presents the Channel Ticket to peers carrying said channel. If resources at the peers permit and the Channel Ticket is valid the client is joined to the P2P network and the user can start watching the channel (steps 5 and 6 in Fig. 1). Henceforth, when a user switches channels, the client repeats the process of contacting the Channel Manager, obtaining a Channel Ticket, etc. transparent to the user [6].

Logically, a service provider only needs to deploy one User Manager and one Channel Manager. In Section V we explore how a service provider can improve service scalability by partitioning its user space into multiple domains, each managed by one or more User Managers, and by partitioning its set of channels into multiple partitions, each managed by one or more Channel Managers. For ease of exposition, we will first assume single User Manager and single Channel Manager when describing our DRM architecture and protocols.

IV. DRM Architecture and Protocols

Similar to other DRM solutions, digital rights protection on our system relies on encrypting the content and regulating access to the decryption key. Whereas traditional DRM requires clients to access a License Manager to acquire the decryption key, we regulate access to the decryption key by regulating access to the P2P network that acts as the content distribution channel. We rule out traditional DRM solution approach due to scalability and reliability concern as described earlier.

In our design, once a client is verified as having authorization to a channel, it is assumed to have authorization to the channel's content decryption keys. In other words, our DRM system consists of two parts: (1) content encryption and (2) channel access authorization. For content encryption we employ a light-weight rotating symmetric key encryption mechanism, described in Section IV-E. With live content broadcasting to a large number of audience, it is not practical to adopt an asymmetric key based solution due to its computational and management overhead.

For channel access authorization, we assign *Attributes* to both users and channels and *Policies* to channels. Access authorization amounts to securely evaluating the policies of a channel given the attributes of a user and those of the channel. The attribute and policy language is similar to ODRL (Open Digital Rights Language) in concept [10], but heavily tailored in both design and implementation to meet the DRM requirements of live broadcast as stated in previous sections.

We consider our contributions in this paper to be presenting (1) the DRM requirements of live broadcast, (2) an overall architecture and protocol design to meet these requirements, and (3) performance measurements from a successfully deployed real system implementation. The engineering details of the architecture and protocols themselves may be replaced with more advanced, more principled, or more secure ones as they become available.

A. Channel Policies and Attributes

Channel attributes are centrally managed at a *Channel Policy Manager*. The number and meaning of attributes are determined by the service provider. As they are created, each attribute is assigned a start time (*stime*) and an end/expiration time (*etime*) for which the attribute is valid. Each channel attribute is also assigned a last-update time (*utime*). The last-update time plays a crucial role in accommodating dynamic changes to digital rights, as we will explain later. To iterate, a channel attribute consists of the following fields: $< attribute, value, stime, etime, utime >$. One example is shown in Fig. 2(a).

Channel policies determine how attributes are to be interpreted and enforced. Each channel can have multiple policies attached to it. Each policy is given a priority, with higher priority policies overriding lower priority ones. For example, as shown in Fig. 2(c), the first channel policy in channel A is based on "Region" channel attribute, and it states a user located in "Region" 100 can access channel A. A service provider that re-broadcasts its over-the-air channels on the P2P network can black out the Internet distribution of one of its channels for a period of time by using the channel attribute and policy mechanisms. First it creates a new channel attribute "Region" and assigns a special value

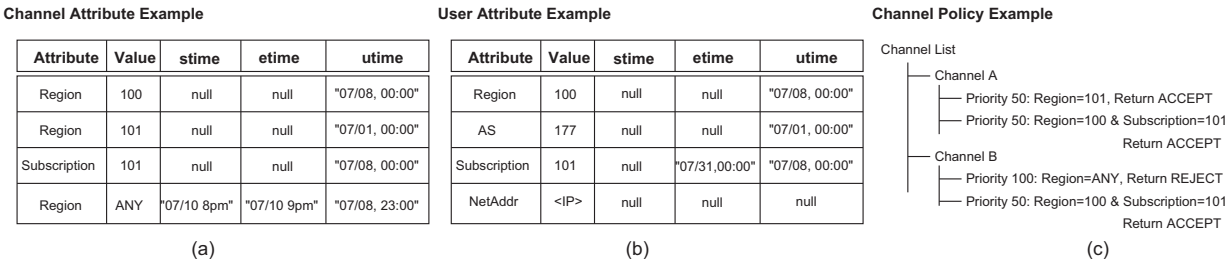


Figure 2. Example of Channel and User Attributes.

signifying "ANY" to it, as shown in Fig. 2(a). This attribute will have start and end times covering the period the channel is to be blacked out. Then the service provider assigns a new policy to the channel requiring users to match this new "Region" attribute. The new policy will be given a higher priority than other existing "Region" attribute(s). Since the User Manager will always assign a user to a "Region" that matches "ANY", no user will be able to match this channel attribute nor access the channel during the given time period, as shown in Fig. 2(c). There are a handful of special attribute values such as ALL, ANY, NONE, NULL, etc. that are globally defined throughout our DRM architecture.

The Channel Policy Manager maintains two lists: (1) the *Channel List* that contains all channels along with their attributes and policies, and (2) the *Channel Attribute List* that contains all the unique attributes collated from all the channels. Whenever there is a change in channel attributes or policies, including the addition and deletion of channels, the Channel Policy Manager updates the Channel List and sends it to the Channel Manager. Whenever a channel is modified, all its attributes' last update times are updated to the current time in the Channel Attribute List. For example, if a channel is added or deleted from the offering of region "X", the "Region" attribute with value "X" will have its last-update time made current. The Channel Policy Manager sends the updated Channel Attribute List to the User Manager. In the next section, we describe how this will prompt clients to retrieve a new Channel List.

B. User Ticket and Attributes

When a user creates an account with the service provider's Account Manager, the Account Manager securely sends the user's identification, subscription, and payment information to the User Manager. The User Manager generates a unique user identification number (UserIN) for this user and creates a new entry in its user database (UserDB) for the user. When a client starts up, it authenticates its user with the User Manager and sends the client's public key to the User Manager for certification. The User Manager authenticates the user and returns a User Ticket to the client (the login protocol is described in Section IV-F1). Fig. 3 shows the structure of a User Ticket. The User Ticket is used by the client to access channels in an authenticate-once, use-often mode of operation, along the line of Kerberos [11]. Similar to Kerberos' ticket, the User Ticket has a limited lifetime. The number and list of User Attributes follow the ticket's start and expiration times. The whole ticket is digitally signed by the User Manager and the signature appended to the end of the ticket. By signing the ticket, the User Manager also certifies the client's public key, which is used in distribution of content decryption keys as described in Section IV-E.

Table I
A NON-EXHAUSTIVE LIST OF COMMON USER ATTRIBUTES

Attribute	Description
NetAddr	The network address of the user
Region	The geographic region the user connects from
AS Number	The network the user connects from
Version	The client version number
Subscription	A package the user has subscribed to

User attributes have the same format as channel attributes (Section IV-A) and are generated by the User Manager based on three sources of data: (1) user account and subscription information from the Account Manager, (2) client connection information, and (3) the Channel Attribute List from the Channel Policy Manager. From the client connection, the User Manager obtains the client's network address (NetAddr), from which it infers the geographic region [12], and the Autonomous System (AS) from where the connection originated [13]. Table I lists a sampling of attributes commonly assigned to a user. Some of these attributes are required to implement certain design requirements.

As with Channel Attributes, User Attributes also have start time (*stime*), end/expiration time (*etime*), and last-update time (*utime*) associated with them, as shown in Fig. 2(b). When they are not used, the User Manager simply assigns the value NULL to these timers. We set the ticket expiration time to be no later than the soonest *etime* of all attributes listed in the ticket, thereby forcing the client to renew its User Ticket before any listed attribute expires. To limit the usefulness of a stolen User Ticket, we recommend that the lifetime of a User Ticket be set to less than the average length of a program in the channel.

An attribute's *utime* is set based on the corresponding Channel Attribute's *utime* in the Channel Attribute List from the Channel Policy Manager. It is used to communicate

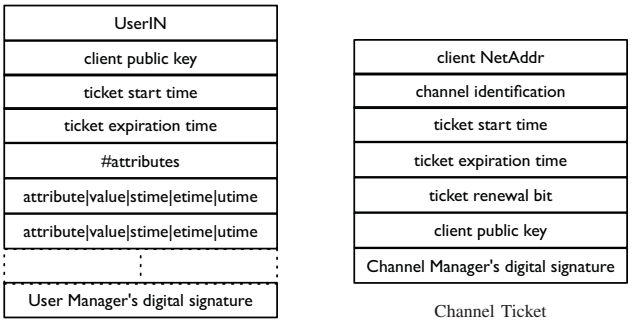


Figure 3. Structure of a User Ticket and Channel Ticket (field width not to scale with actual field size).

changes in the channel lineup to the client. Upon receiving a User Ticket, a client compares the *utime* of every attribute in the new User Ticket against its counterpart in the previous User Ticket. If any of the *utimes* is more recent than its counterpart in the previous User Ticket, the client will contact the Channel Policy Manager with a list of attributes with more recent *utimes* to obtain an updated Channel List.

C. Channel Ticket

After a client obtains a User Ticket from the User Manager and, if necessary, an updated Channel List from the Channel Policy Manager, it presents the list of available channels for user selection.

Once the user has made its selection of which channel to watch, the client contacts the Channel Manager to obtain a Channel Ticket and a list of peers from which it can receive the channel signal. Introducing Channel Ticket into the architecture serves several purposes: (1) to simplify channel access authorization at target peering clients, (2) to reduce the amount of user information disclosed to target peers, and (3) to log viewing activities for license payment, royalty payment, and billing purposes.

To obtain a Channel Ticket, the client presents its User Ticket along with the channel of interest to the Channel Manager. The Channel Manager verifies the User Ticket against the User Manager's digital signature, checks that the User Ticket has not expired, matches the value of the NetAddr attribute in the User Ticket against that of the client's current connection to the Channel Manager, and evaluates the policies of the channel using the attributes present in the User Ticket. If all these checks pass, the Channel Manager issues a Channel Ticket to the client. Fig. 3 shows the structure of a Channel Ticket. The Channel Manager digitally signs the whole Channel Ticket, in the process certifying again the client's public key. The use of the "ticket renewal bit" to renew Channel Ticket and prevent simultaneous use of a user account will be described in Section IV-D. By filtering out all user attributes other than the client's network address, the Channel Manager serves to intermediate between the protection of user privacy and protection of content owner's digital rights. The user's identity and its subscription to other channels are visible only to the service provider, and not to peers in the distribution network.

The lifetime of a Channel Ticket can be no longer than the remaining lifetime of the client's User Ticket. This puts an effective lower bound on the amount of lead time the service provider must give the DRM system when deploying any new channel viewing policy. If a program in a channel must be blacked out, for example, the policy must be put in place at least one User Ticket lifetime prior to the start of the black out period. Otherwise, a user's Channel Ticket could be valid into the blackout period, and they would not be kicked from the channel. To avoid service interruption, Channel and User Tickets must be renewed in time.

To join a P2P distribution channel, a client contacts the peers returned by the Channel Manager along with the Channel Ticket. The process by which a client joins a P2P distribution network is part of the P2P protocol and is outside the scope of this paper. We describe such a protocol in an earlier publication [6]. Relevant to our DRM system is that as part of the peer join procedure, the client presents its Channel Ticket to the target peers. Since the possession of a valid Channel Ticket signifies that the Channel Manager has authorized the client to access the channel, authorization verification at target peers is greatly simplified. A target peer only needs to verify the Channel Ticket against the Channel Manager's digital signature, check that the Channel Ticket has not expired, match the NetAddr in the Channel Ticket against that of the client's current connection, and check that the channel of interest is indeed the channel it is currently carrying. It does not need to evaluate channel viewing policies and it does not have access to any other user attributes. If the client is accepted at the target peer, a peering relationship is formed between them, and the target peer creates a symmetric session key that it sends to the client, encrypted using the client's public key. This session key is used for distribution of content decryption keys, described in Section IV-E.

D. Ticket Renewal

To renew a User Ticket, the client can simply repeat the process of obtaining a new User Ticket. To renew a Channel Ticket, however, requires a different process. To enforce the requirement that an account cannot be used to join the same channel from multiple locations at the same time, a peer will terminate a peering relationship whose Channel Ticket has expired if a renewal ticket is not presented. A renewal ticket is a Channel Ticket with the "ticket renewal bit" set. To issue a renewal ticket, a Channel Manager must be presented with the expiring Channel Ticket, in lieu of the "channel identification," within a small window of the ticket expiration time.

Every time the Channel Manager issues a new Channel Ticket, it logs the UserIN, channel watched, and client NetAddr. During Channel Ticket renewal, it looks up the log for the latest entry of the joint field of UserIN from the User Ticket and channel identification from the expiring Channel Ticket. If the resulting log entry shows a NetAddr that does not match the NetAddr in both the User Ticket and the expiring Channel Ticket, the Channel Manager will not issue a renewal ticket. Otherwise, the Channel Manager performs the same check as it would when issuing a new Channel Ticket and, if all checks pass, updates the expiration time of the expiring Channel Ticket, sets the ticket renewal bit, and sends it back to the client with a new digital signature.

If a user moves from one computer to another, the request for a new Channel Ticket will append to the log an entry for UserIN and channel watched with the NetAddr in the new Channel Ticket. When the client at the old location renews, the request will be not processed since the Channel Ticket renewal process always tries to match the latest entry for UserIN and channel watched. This way, a user moving from one computer to another doesn't have to wait for the old Channel Ticket to expire. The old Channel Ticket will not be renewed and the client at the old location will have its peering relationship severed by its peers upon the expiration of its Channel Ticket.

E. Content Encryption

As shown in Fig. 1, live content is ingested and encoded at the Channel Server. If the service provider wishes to encrypt the content for distribution,[2] encryption can be done at the Channel Server using symmetric key encryption, e.g., with 128-bit AES. By re-keying the channel frequently, e.g., at one-minute interval, the service provider can provide forward secrecy such that if a symmetric key (henceforth "content key") is lost, it can only be used to decrypt contents generated during its corresponding one-minute period. Each iteration of the evolving content key can be marked with an 8-bit serial number. By the Channel Server's pre-pending this serial number to each content packet, the client would know which content key to use to decrypt a packet. As previously mentioned, once a client is verified as having authorization to access a channel, it is assumed to have authorization to access the channel's content keys. We can therefore distribute the content keys using the same P2P distribution network, alongside the encrypted channel signal.

The purpose of encrypting channel signal is two fold: (1) to prevent eavesdroppers who have not been authorized to access a channel from siphoning off channel contents, this includes previously authorized clients who have not renewed their authorizations, and (2) to detect when the channel has been hijacked, whereby rogue contents are accidentally or maliciously injected into the P2P network to masquerade as legitimate contents. Our DRM system is designed specifically to protect digital rights for the broadcast of live content in a linearized channel. The issues of remote client attestation, watermarking of content, and digital rights management of *recorded* contents are active areas of research and development and are outside the scope of this paper. Results from these related research could be adopted to complement our DRM system where the protections they provide are needed.

[2]Some service providers with a public mandate to distribute content do not want to have their contents encrypted, even if they want to control distribution, and therefore access, to within their geographic region.

Once a client accepts a new peer into its P2P network, it generates a symmetric session key and sends it to the new peer, encrypted with the new peer's public key. At the same time, it sends a copy of the current content key to the new peer, encrypted with the session key. Content keys are generated by the Channel Server and are distributed to all peers in a pair-wise manner using the session keys. Consider an example distribution tree consisting of peer A at the root: peer A has two children, B and C, and peer B further has two children, D and E. Each link in the tree ($A-B, A-C, B-D, B-E$) is associated with a different pair-wise session-key, known only to the two peers incident to the link. When A generates (or receives) a new content key, for each of its children, it encrypts the content key with the appropriate session-key for that child, and sends the encrypted content key to that child. When B receives the encrypted content key from A, B first decrypts the content key using the session key it shares with A, then B re-encrypts the content key twice, once with the session-key it shares with D, and once again with the session key it shares with E. The resulting encrypted content keys are sent to the respective children. The process continues until the new content key is distributed through the entire tree.

New instances of the evolving content key are sent some amount of time in advance of their use, to ensure that all clients would have received the new content key before they need it. The underlying P2P protocol ensures reliable distribution of content key. If the P2P network allows a peer to connect to multiple parents, for example when the stream is sent as sub-streams through multiple parents [6], a peer may receive multiple copies of the same content key from its parents. Since each iteration of the evolving content key is tagged with a serial number, a peer can simply discard the duplicated keys.

F. DRM Protocols

We now provide more specific details on the protocols used in our DRM system.

1) Obtaining User Ticket: When a client starts up, it runs the login protocol with the following goals: (1) to authenticate the user with the User Manager, (2) to obtain a User Ticket from the User Manager, (3) to provide the User Manager with remote attestation that the client has not been modified, and (4) to have the User Manager

certify the client's public key.[3] The client initiates the login process by sending the user's email address and its public key to the User Manager. The User Manager locates the user in the UserDB by its email address and sends back to the client a nonce and some parameters for the checksum computation over client application program.[4] Both the nonce and checksum parameters are symmetrically encrypted using the secure hash of the user's password (*shp*) as the encryption key. The client then returns the nonce and computed checksum, encrypted using its private key (*privk*). The fact that the client can decrypt the nonce and checksum parameters and that the User Manager can decrypt the nonce and checksum returned by the client gave assurances to the User Manager that it is indeed in live communication (as opposed to captured packets play back) with a user who knows the user password and a client who has in its possession the private key corresponding to the public key sent earlier. The User Manager then issues the User Ticket, as described in Section IV-B and sends it to the client, completing the login process. Fig. 4(a) summarizes the login protocol. The two rounds of message exchanges in this protocol are labeled `LOGIN1` and `LOGIN2` respectively. The "client's version number" is used to enforce minimum version requirement of client application, for example when a new DRM architecture or protocol is deployed. The "timing information" is used to synchronize client clock with that of the User Manager's.

2) Obtaining Channel Ticket: When a user switches channels, the client must obtain a Channel Ticket for the new channel by contacting the Channel Manager. The client starts the channel switching process by sending the target channel identification and the User Ticket to the Channel Manager. The Channel Manager challenges the client with a nonce, which the client responds to by sending the nonce back, encrypted with its private key. Upon decrypting the nonce, the Channel Manager issues a Channel Ticket, as described in Sections IV-C and IV-D, and sends it to the client, along with a list of peers on the channel's distribution network. A similar protocol is used for renewal of Channel Ticket. However, instead of the target channel identification, the expiring Channel Ticket is sent along with a new User Ticket in the first message from the client. Fig. 4(b) summarizes the channels switching protocol. The two rounds of message exchanges in this protocol are labeled `SWITCH1` and `SWITCH2` respectively.

3) Obtaining Channel Signal: To join a peer, a client sends a join request along with the Channel Ticket to a number of peers listed in the peer list returned by the Channel Manager. If a peer has enough resources to accommodate the client, it sends back a join accept, a session key encrypted with the client's public key, and the current content key encrypted with the session key, as described in Section IV-E. Fig. 4(c) summarizes the peer join protocol. The single round of message exchanges in this protocol is labeled `JOIN`.

G. Threat Discussions

1) Ticket Capture and Replay: Both User and Channel Tickets are digitally signed by their respective issuers. The peer list returned by the Channel Manager in the `SWITCH2` message exchange round as part of the channel switching protocol, however, is not signed. An attacker can replace the peer list with itself and its accomplice peers to cause the victim to peer with them. Aside from denying service to the victim, the attacker(s) can also capture the victim's Channel Ticket in this way. We decided not to digitally sign the peer list because an attacker who has access to the victim's traffic, and can modify it, would be able to deny service to the victim regardless. If the attacker had access to the victim's network, it can also eavesdrop on the network and obtain the victim's User Ticket. Encrypting the User Ticket with a symmetric key generated by the victim, per the secure socket layer (SSL) protocol, would prevent the User Ticket being captured, but not so the Channel Ticket as the victim must send its Channel Ticket to its peers during the peer join procedure. Additionally, using a SSL-like protocol between peers would not prevent a Channel Ticket being captured by a malicious peer who forges the initial peer communication protocol. We evaluate the risk of having either ticket captured by an attacker next.

Since both tickets are digitally signed, they cannot be forged or tampered with. A stolen ticket is useful to an attacker for its contents and for replay attack. The only use of the User Ticket is for obtaining Channel Ticket from the Channel Manager and Channel List from the Channel Policy Manager. In both cases, the client would be required to send a nonce encrypted with its private key, hence an attacker that has a client's User Ticket but not the client's private key cannot do much with the ticket. A Channel Ticket is used to join a P2P distribution network carrying a channel. Since the channel content key is protected by a session key generated and distributed by the target peer, encrypted with the client's public key, the Channel Ticket is again not useful to an attacker without the attacker's having possession of the client's private key.

Should the contents of the User Ticket or other information exchanged with the infrastructure servers be considered sensitive enough to be protected from eavesdropper, we can easily enforce an SSL-like protocol for all communications with infrastructure servers, as the client already must obtain the public keys of all our infrastructure servers in the current design.

[3]Of these four goals, user identification, which is required for authentication, and remote attestation are outside the scope of this paper only the most rudimentary technologies to accomplish these are presented here for illustrative purposes.

[4]Using checksum of client application is insufficient for remote attestation as a modified client can simply keep a copy of the original client around for the sole purpose of computing checksum.

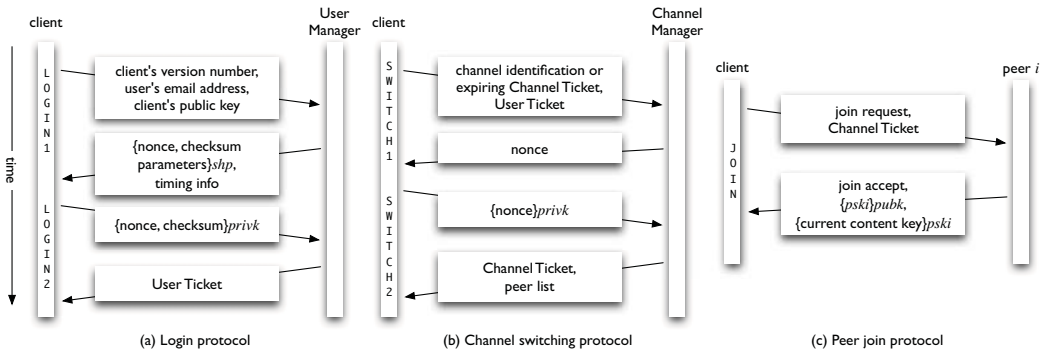


Figure 4. Protocol details among client, User Manager, Channel Manager, and peers.

2) Denial of Service and Compromised Client: Distributed Denial of Service (DDoS) or other forms of DoS attacks on our infrastructure or peers are not threats unique to this DRM system, thus they can be addressed in similar fashion as protecting general network services on the Internet.

We review some of the threats to our design if a client application is compromised. A compromised client can decrypt channel signal and send it outside the channel's broadcast region. A compromised client can also record the decrypted signal. Unfortunately, unless the signal is hardware protected all the way to the display device, a determined attacker can always capture the signal at any point along the path, e.g., by capturing unencrypted signal between video memory and display device. This vulnerability applies to all DRM systems, including traditional DRM systems. Even if the full path is protected, a good recording from the display can be made with cameras that have advanced stabilization technology. A compromised client can also allow a user account to be used to watch the same channel at multiple locations. However, this threat is no worse than rebroadcasting a decrypted channel signal.

V. Scalability

The design of our DRM architecture takes advantage of the P2P network by using it to distribute the rotating content encryption key in a push-based manner, and by delegating authorization verification to the individual peer. By its nature, granting of access authorization needs to be administratively centralized. In this section we briefly discuss the scalability of the authorization granting parts of the architecture.

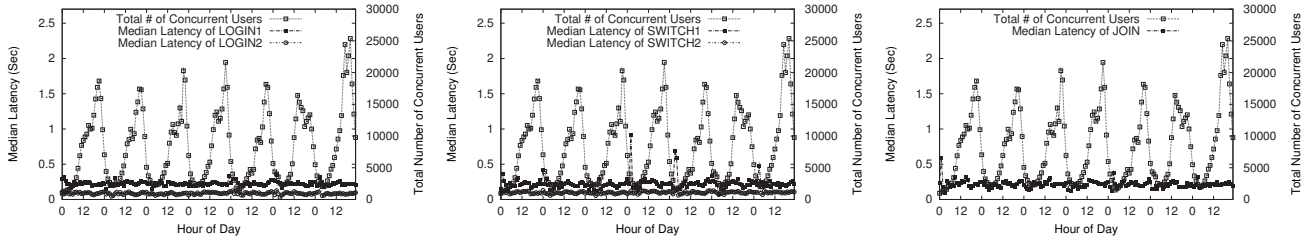
Both the User Ticket and Channel Ticket acquisition processes are atomic and neither the User Manager nor the Channel Manager keeps per-client states in memory, which helps with the scalability of their designs. Service providers with a large customer base or with offerings in different geographic regions may nevertheless want to deploy multiple User Managers and/or Channel Managers for performance or administrative reasons. We call the set of users assigned to a User Manager an *Authentication Domain* (or just *domain*), and the set of channels assigned to a Channel Manager a *Channel Listing Partition* (or just *partition*).

The coverages of a broadcast geographic region, an Authentication Domain, and a Channel Listing Partition can be set independently of each other. Furthermore, the User Manager managing a domain is singular only logically. In deployment, the single User Manager can be implemented across a server farm of User Managers. In which case, the multiple instantiations must all share the same network name/address and public/private key pair to maintain the logical view of a single User Manager per domain. In our design, we ensure the authentication with User Manager to be stateless, so that a client can finish the authentication process with different User Managers at each step. This greatly improves the simplicity and efficiency of User Managers, and makes it resilient to churn and flash crowd.

The same applies to the deployment of a Channel Manager in a partition. The logical view of a single Channel Manager per partition is maintained by having multiple instantiations of the Channel Manager share a single network name/address, public/private key pair, and user viewing activity log. In the extreme case, a very popular channel can be put in a partition of its own and served by a farm of Channel Managers sharing a single network address/name and public/private key pair.

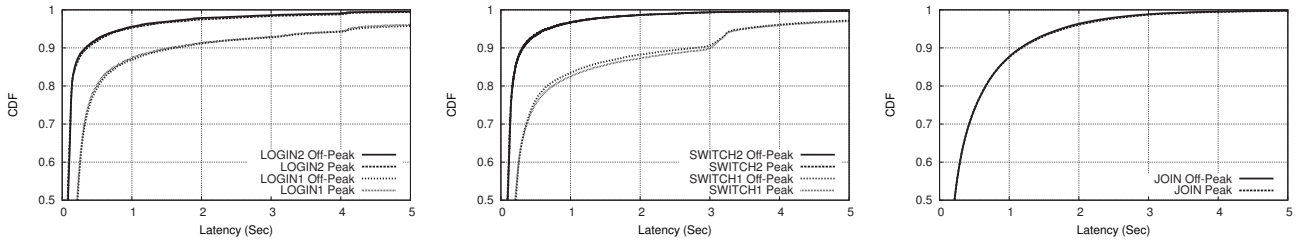
To direct client to the right User Manager, we introduce a new backend service called the *Redirection Manager*. The job of the Redirection Manager is simply to look up the User Manager a user has been assigned to. For future extensibility, we have the Redirection Manager also return the network name/address and public key of the Channel Policy Manager. Since the load of this service is very light (a single hash table lookup), a single Redirection Manager per service provider network is sufficient. The network name/address and public key of the Redirection Manager is built-in to the client application. Existing approaches to secure webservers from DoS attacks can be used against the Redirection Manager.

While the load imposed on a User Manager and/or Channel Manager depends on the size and activities of the user



(a) Login protocol (b) Channel switching protocol (c) Join protocol

Figure 5. Median latency vs. total number of concurrent users.



(a) Login protocol (b) Channel switching protocol (c) Join protocol

Figure 6. CDF distribution of latencies for peak hours vs. off-peak hours.

population, the load imposed on the Channel Policy Manager can be managed by the service provider by timing the implementation of its administrative decisions; for example, by not adding a new channel right before the start time of a popular live event. We thus do not foresee the need for more than one Channel Policy Manager in a service provider's distribution network. When there are multiple Channel Listing Partitions in a service provider's network, each channel is assigned to one, and only one, partition. The network name/address and public key of the Channel Manager a channel is assigned to becomes part of the channel description. Thus when a client obtains a Channel List from the Channel Policy Manager, it also retrieves the network name/address and public key of the Channel Manager assigned to each channel.

VI. Performance Study

Our DRM system design has been incorporated into a production P2P live streaming network [6]. This network and our DRM system have served live broadcast of over 200 channels to over 3 million registered users, with over 60,000 concurrent users at its peak. We examine the impact of highly bursty traffic on the scalability of our DRM architecture by measuring the latencies of the various message exchange rounds that are part of our protocols, labeled as `LOGIN1`, `LOGIN2`, `SWITCH1`, `SWITCH2`, and `JOIN` in Section IV-F and Fig. 4.

To estimate individual components of latencies incurred by our DRM system, we leverage the "user feedback" logs collected by the live streaming system. The feedback logs are generated in an encrypted form by the client. The logs record detailed behavior of the client software, for debugging purposes. When users experience an error with their client software, they can choose to submit the feedback log by clicking a "Submit Feedback" button on the client user interface. Since the submitted feedbacks include logs from all channel watching sessions at the client prior to the one with error, these feedbacks also include sessions without errors. From the collected user feedback logs, we can measure the latency of the five message exchange rounds listed above. The feedback logs used in our analysis were collected for a period of one week from June 23rd to June 29th, 2008. A total of 60,669 feedback logs were received during this period. In an earlier publication [6], we analyzed and validated the representativeness of such asynchronously submitted user feedback logs. We use two User Managers and four Channel Managers in total to serve two partitions. The physical machines are 1U duo Intel Xeon Servers.

Fig. 5 shows how the latencies incurred by our DRM protocols are affected by total system usage variations during the one week period. The y-axis of the three plots shows the median latency incurred in the login, channel switching, and join protocols, respectively. The login and channel switching protocols are further broken down into two steps, and the median latency of each step is plotted separately. The median latencies are then compared against the total number of concurrent users at the time. It can be easily seen that for all three protocols, the incurred latencies are hardly affected by the total number of concurrent users. Sporadic spikes in median latencies are due to statistically insignificant samples, all occurring between 0AM-6AM. Pearson product-moment correlation coefficient between median latency and number of users ranges from -0.03 to 0.08 for login and channel switching protocols, and is 0.13 for join protocol. Although join protocol overhead exhibits slightly higher

dependence on total system usage, its correlation can still be considered weak.

Fig. 6 is another representation of the weak correlation between DRM protocol overhead and total system usage. In this figure, we compare the CDF distribution of latencies experienced during peak hours (from 6PM to 0AM) and off-peak hours (from 0AM to 6PM). For all three protocols, the CDF distribution curves from the two separate time periods are virtually identical, confirming the weak correlation of latencies experienced by the protocols with system load.

VII. Related Work

Several research projects studied the integration of DRM into P2P based distribution of archival content discretized into files. Depending on the need for the centralized authority server in the system, these can be realized as either a semi-distributed (e.g., [14]) or purely distributed system (e.g., [15], [16]). The authors of [17] presented a prototype secure streaming media system in a centralized client-server environment. The DRM proposal for P2P based IPTV presented in [18] employs a semi-distributed architecture where critical DRM functions such as license and key distributions and user authentication and authorization are centralized. The network described in the paper is actually also closer to an overlay network serving as a virtual backbone than a P2P network where each peer forwards data to another peer on an equal footing. As far as we know, there does not exist any other study presenting digital rights management architecture in an operational P2P streaming system deployed with significant real usage. Our paper is the very first in presenting a successfully deployed real system with performance measurements.

VIII. Conclusion

We have presented a DRM system that provides digital rights protection to content channels broadcast live over a P2P network. In the live broadcast environment, a DRM system must be able to handle highly bursty traffic in a scalable fashion, without introducing significant delay or resorting to inefficient resource allocation. In addition, a DRM system must be versatile enough to support time-limited, location-restricted and audience-dependent distribution rights, and at the same time, be flexible enough to accommodate dynamic rights and access policy changes in ongoing live broadcast. Our system meets these requirements by employing an access control mechanism distributed over a P2P network, and centralized rule-based authorization techniques. We have shown by real-world measurements that our system can be made scalable for large-scale deployments.

References

[1] Adobe Systems Inc., Adobe Flash Access 2.0. [Online]. Available: http://www.adobe.com/products/flashmediaserver/pdfs/flashaccess2_0_whitepaper.pdf

[2] Apple Inc., FairPlay. http://en.wikipedia.org/wiki/FairPlay.

[3] Microsoft Corp., Architecture of Windows Media Rights Manager. [Online]. Available: http://www.microsoft.com/windows/windowsmedia/howto/articles/drmarchitecture.aspx

[4] RealNetworks, Helix Technology. http://www.realnetworks.com/products-services/helix-technology.aspx.

[5] R. auf der Maur, "Die Weiterverbreitung von TV- und Radioprogrammen über IP-basierte Netze," in *Entertainment Law (f. d. Schweiz)*, 1st ed. Stämpfli Verlag, 2006.

[6] H. Chang, S. Jamin, and W. Wang, "Live Streaming with Receiver-based Peer-division Multiplexing," vol. 19, no. 1, February 2011, a shorter version of this paper appeared in *Proc. of the ACM Internet Measurement Conference*, Oct. 2009.

[7] A. Seshadri *et al.*, "Pioneer: Verifying Code Integrity and Enforcing Untampered Code Execution on Legacy Systems," in *Proc. of SOSP*, October 2005.

[8] E. Shi, A. Perrig, and L. van Doorn, "BIND: A Fine-grained Attestation Service for Secure Distributed Systems," in *Proc. of IEEE Symp. on Security and Privacy*, 2005.

[9] T. C. Group, "Trusted platform module main specification," Oct. 2003, http://www.trustedcomputinggroup.org.

[10] S. Guth, G. Neumann, and M. Strembeck, "Experiences with the enforcement of access rights extracted from odrl-based digital contracts," in *the 3rd ACM workshop on Digital rights management*, October 2003.

[11] B. Neuman and T. Ts'o, "Kerberos: An Authentication Service for Computer Networks," *IEEE Communications*, vol. 32, no. 9, pp. 33–38, September 1994.

[12] MaxMind, "GeoIP IP Address Location Technology," http://www.maxmind.com.

[13] H. Chang, S. Jamin, and W. Willinger, "Inferring AS-Level Internet Topology from Router-Level Path Traces," in *SPIE ITCom*, August 2001.

[14] T. Iwata *et al.*, "A DRM system suitable for P2P content delivery and the study on its implementation," in *The 9th Asia-Pacific Conference on Communications*, September 2003.

[15] C.-C. Chu *et al.*, "Mobile DRM for Multimedia Content Commerce in P2P Networks," in *IEEE Consumer Communications and Networking Conference*, January 2006.

[16] J. Y. Sung, J. Y. Jeong, and K. S. Yoon, "DRM Enabled P2P Architecture," in *The 8th Internation Conference on Advanced Communication Technology*, February 2006.

[17] D. Holankar and M. Stamp, "Secure Streaming Media and Digital Rights Management," in *Hawaii International Conference on Computer Science*, January 2004.

[18] X. Liu *et al.*, " A DRM Architecture for Manageable P2P Based IPTV System," in *IEEE International Conference on Multimedia and Expo*, July 2007.