PRISM Architecture: Supporting Enhanced Streaming Services in a Content Distribution Network

C. D. Cranor, M. Green, C. Kalmanek, D. Shur, S. Sibal, C. Sreenan[‡] and J.E. van der Merwe[*]

AT&T Labs - Research
Florham Park, NJ, USA

[‡] *University College Cork*
Cork, Ireland

Abstract

PRISM is an architecture for distributing, storing, and delivering high quality streaming content over IP networks. In this paper we present the design of the PRISM architecture and show how it can support an enhanced set of streaming services. In particular, PRISM supports the distribution and delivery of live television content over IP, as well as the storage of such content for subsequent on-demand access. In addition to accommodating the unique aspects of these types of services, PRISM also offers the framework and architectural components necessary to support other streaming content distribution networks (CDN) services such as video-on-demand. We present an overview of the PRISM architecture focusing on components unique to PRISM including content management, content discovery, and content-aware redirection. We also describe the PRISM testbed that we are using for experimental lab service trials.

Keywords: streaming content distribution network (CDN), content management, content discovery, content-aware redirection

1 Introduction

With the emergence of broadband access networks and powerful personal computer systems, the demand for network-delivered full-motion streaming video content is growing. Indeed, a number of companies have already made rudimentary efforts to deliver live (www.icravetv.com) and stored (www.recordtv.com) television over the Internet.[1] While the traditional Web service model can be applied to IP-based streaming content, the user experience does not compare favorably with the quality of cable, satellite, or broadcast television. On the other hand, compared to the Web, current television broadcasting technologies offer high quality but provide limited choice and allow little or no on-demand access to video content.

IP content distribution networks (CDNs) provide scalability by distributing many servers across the Internet "close" to consumers. Consumers obtain content from these edge servers directly rather than from the origin server. Current CDNs support traditional Web content fairly well, however support for streaming content is typically less sophisticated and often limited to the delivery of fairly low quality live streams (so called "webcasting") and distribution of low-to-medium quality video clips ("clip-acceleration"). In order for Internet-based streaming to approach the same level of entertainment value as broadcast media, CDNs must drastically improve the quality of streaming that can be supported. Additionally, streaming CDNs will become more sophisticated in terms of services that can be supported in order to accommodate new business models and new types of service offerings.

In this paper we present PRISM, a Portal Infrastructure for Streaming Media.[2] PRISM is an architecture for distributing, storing, and delivering high quality streaming media over IP networks. PRISM enables services that are more sophisticated than those currently available on the Internet. Our approach has been to consider in detail how such an architecture will support these types of streaming services by considering a TV-like service as an example. In particular, PRISM supports both the distribution and delivery of live television content over IP, as well as the storage of such content for subsequent on-demand access, i.e. stored-TV (STV). We are not aware of any existing streaming CDN architectures with the latter capabilities. The PRISM-based STV service allows users to view content based on the name of the content as well as the time at which it was "aired." For example, users can request programming that aired on CNN at 1PM on January 1, 2001. Content stored inside the network is made accessible throughout the whole PRISM infrastructure. A user based in the U.S. can access European sporting events or other TV content — both live or on-demand. PRISM also

[*]Contact author: Kobus van der Merwe, AT&T Labs - Research, 180 Park Avenue, Florham Park, NJ 07932, USA, kobus@research.att.com.

[1]Both have since been silenced because of legal problems.

[2]An preliminary version of this paper was presented as a work-in-progress report at the NOSSDAV 2000 workshop [1].

allows individual users to specify content to be stored in a "network-VCR" type service. In addition, a managed STV service where certain "bundles" of content are always stored by the service provider for user access is feasible. An example service bundle could include the storage of the three most recent hours of content for a large number of channels, the storage of the peak viewing hours for a smaller set of channels stored for a week, and the archival of a set of popular programs for several months. This type of PRISM service makes available significantly more content to users compared to emerging personal/digital video recorders.

Our goal is to provide content at a quality level comparable to existing broadcast TV. Coupled with the potential on-demand nature of PRISM services, this translates into significant access bandwidth requirements. Technically, broadband access technologies such as cable and xDSL offer enough bandwidth for PRISM services at entertainment quality. An actual service offering would also depend on business considerations both in terms of broadband access economics and business models that are attractive to content providers. In the near future advances in encoding technology might also help to reduce the bandwidth requirements for PRISM-like services.

While we focus on applying the PRISM architecture to services involving live and stored television content, the resulting framework and architectural components are applicable to other content-based streaming services such as video-on-demand. The components of the PRISM architecture are introduced in Section 2. These components include content naming, content management, content discovery, and content-aware redirection. In Section 3 we present a sampling of related work. The PRISM architecture is being realized as a trial within our laboratories. We briefly describe the status of this effort as well as future plans in Section 4.

2 PRISM Architecture

PRISM facilitates the creation of services that make both live and on-demand TV content available over an IP network. The PRISM architecture is built around three types of basic elements, as shown in Figure 1:

Live sources: receive content from a content provider, encode and packetize it, and then stream it into the PRISM IP network infrastructure.

Portals: receive multimedia content from live sources and other portals and transmit it to PRISM clients. Portals can store and archive live content, thus allowing content to be viewed on-demand. Portals also provide VCR-like functions such as fast-forward and rewind to clients. Portals are positioned between clients and live sources, typically at or upstream of a bandwidth discontinuity such as a cable head-end.

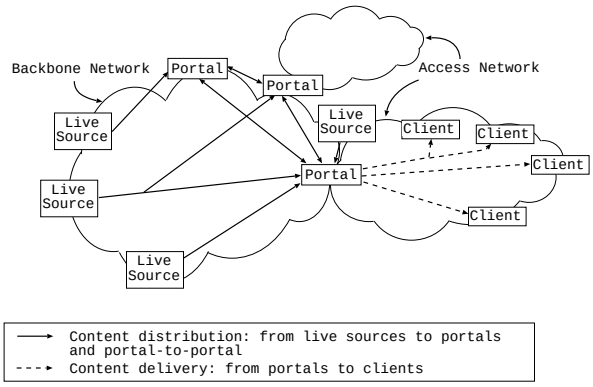

Figure 1: PRISM data plane

Clients: receive content from a portal and display it to end-users. Clients are networked set-top boxes or PCs connected to the backbone using broadband access. A client normally interacts with a *local* portal that is close to it in the network. Note that a client's local portal may act as a proxy when it receives a request for content that it does not have locally stored. This allows the local portal to provide VCR-like controls to the client even when the content being viewed is coming from a remote portal, and allows the local portal to cache the recently viewed content.

The three types of PRISM elements communicate through network interactions. There are two classes of PRISM network interactions: those that occur on the data plane and those that occur on the control plane.

The data path functionality is required by any streaming CDN. Since it is not unique to PRISM it is only briefly considered below. Figure 1 shows PRISM's data plane components:

Content distribution: transferring content from a live source to one or more portals, or transferring content between portals. Content should be transfered efficiently while maintaining an appropriate balance between timeliness and reliability.

Content delivery: streaming content from a portal to one or more clients. In order to provide acceptable perceived performance for latency-sensitive operations such as VCR-like functions it is important that portals be topologically close to clients.

The remainder of the architectural discussion focuses on the PRISM control plane. A unique aspect of the PRISM control plane comes from storing live content inside the network for subsequent on-demand access. This differs from on-demand access in conventional streaming CDNs in that there is no well-known origin server where content is known to reside. This has a major impact on the control plane architecture. The role of PRISM's control

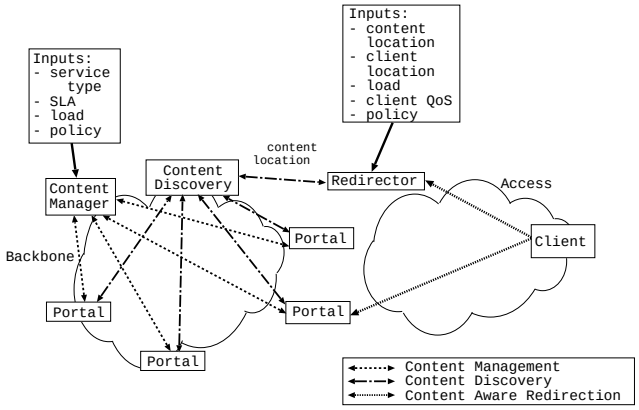

Figure 2: PRISM control plane

plane is to control how content is located and flows through the PRISM infrastructure. Figure 2 shows the three main PRISM control plane components:

Content management: coordinating and managing the storage of content at portals. Input to the content management process includes information on type of service, service level agreements (SLAs) with content providers, portal capacity and load caused by user access. Section 2.2 describes a content management system that can accomodate service-specific input to support a stored-TV service.

Content discovery: determining the existence and location of streaming content within the PRISM infrastructure. When users request specific content, PRISM uses a content discovery mechanism to determine from which portal or live source the content can be obtained. Section 2.3 presents our content discovery solution. Note that while content discovery is triggered by a user request, the actual discovery process, as shown in Figure 2, happens between PRISM entities and is not visible to the user other than through the redirection process discussed next.

Content-aware redirection: redirecting user requests to the "appropriate" portal from where their request can best be satisfied. The location of the requested content along with other inputs, can be used to determine the appropriate portal from which content should be served to the customer. Basic redirection capabilities are part of all edge servers including portals. Specialized redirection servers, the redirector(s) in Figure 2, will however typically be deployed to perform this control plane function. The details of our request redirection solution are presented in Section 2.4.

Note that any TV content that was "aired" could be available via PRISM. Thus, in addition to these three control plane components, PRISM also requires a content naming

```
name      := "stv:" channel_name
           | "stv:" channel_name "?" spec
spec      := "start=" utctime
           | "start=" utctime ";end=" utctime
           | "program=" name
           | "program=" name ";offset=" time

channel_name     := "<" brand ";" channel ";"
                    distributor ";" location ">"
```

Figure 3: Naming scheme URI syntax

scheme to allow us to uniquely identify all or parts of a broadcast stream. We describe the naming scheme we have adopted in the next section.

2.1 Content Naming

In PRISM, content is referenced by name, that is by Uniform Resource Name (URN), rather than by its location or Uniform Resource Location (URL).[3] Identifying content by name rather than by its location allows content to be accessed in a variety of ways (e.g. schedule based or via content-aware search engines) without having to make the portal structure of PRISM externally visible. The naming scheme should also be compatible with the Web. Recognizing that existing TV broadcasts are a rich source of content, our naming scheme allows such content to be identified with reference to the existing channel names, TV network distributor, and original viewing location. To properly capture localization of content from a cable system it may be necessary to specify which cable head-end the content was sent over. The naming scheme should therefore also allow content to be identifiable to a fine level of granularity while at the same time not requiring such detail when it is unnecessary. Furthermore, since content is being accumulated and stored over time, it must be easy to reference content by date and time (e.g., "show me ABC TV aired on June 30th, from 9AM to 10AM"). PRISM gains access to content by digitizing TV channels for distribution and delivery over IP.

The syntax of our naming scheme is shown in Figure 3. The channel name (described in detail below) identifies a unique stream of content. The start and stop times are expressed as UTC times [3] (e.g. "utc:20010215T2200"). The program name is a text string identifying content within a particular channel, and the time offset is relative to the start of a program in seconds (e.g. "program=nypd_blue" or "program=nypd_blue;offset=60"). A

[3] URNs name content without providing location information, while URLs specify the location of content without necessarily providing a meaningful name. For example, in PRISM, names of TV shows are encoded in URNs, while file and portal host names are encoded in URLs. More generally, identifiers such as URNs and URLs are called Uniform Resource Identifiers (URIs) [2].

Channel name	Meaning (for listings)	Meaning (for requests)
<abc;;;>	list available ABC stations	the default ABC station
<abc;wabc;;>	list available source for ABC station WABC (broadcast, cable, satellite, etc.)	ABC station WABC from the default source
<abc.net.au;;;>	list available Australian Broadcasting Co. sources	the default source for Australian Broadcasting Co.
<abc;;directv;>	list available ABC stations on DirectTV	the default ABC station on DirectTV
<abc;wabc;comcast;orange_nj>	test for ABC station WABC on Comcast's Orange, NJ system	ABC station WABC on Comcast's Orange, NJ system

Table 1: Example channel names

URN with no time offset implies the current live version of that content.

The *channel name* consists of four elements, as shown in the bottom of Figure 3. The *brand* is the channel name users typically know, e.g. a simple identifier such as "ABC," or a fully qualified domain name for the brand. The *channel* is the call letters or channel number associated with the content. In some cases this field may be null. The *distributor* indicates the entity that is distributing the content, e.g. the owner of a broadcast station, a cable company, a satellite company, or content distribution system on the Internet. The *location* is the source of the specified version of the content. This can be used to indicate a specific broadcast tower or cable head-end. All these elements are optional. The meaning of unspecified elements in a channel name depends on the context in which the name was used. If the channel name is used in a channel listing query, then unspecified elements match all the available values for the given user. Otherwise, unspecified values take the default values from the user profile. Examples of channel names are shown in Table 1. Note that end users need not understand or be aware of the channel naming syntax as it can easily be hidden behind a user friendly Web interface.

Industry adoption of work being done within the IETF's URN working group allows for the use of URN schemes such as ours. In the absence of such a support infrastructure in the current Internet, we follow common practice of encoding our URNs within a URL. For example, `stv:<abc;wabc;;>` can be encoded in a Real Time Streaming Protocol (RTSP) [3] URL as: `rtsp://server/prismurn/abc/wabc/*/*/`.

2.2 Content Management

PRISM content management is depicted in Figure 4. A content manager(s) communicates with a set of portals. The main purpose of content management in PRISM is to coordinate the distribution and storage of content within the network infrastructure. This is done in a way that balances the resources required and those available against the timely delivery of requested content to users. A content manager receives external input, for example by means of a user interface or a configuration file, as well as feedback

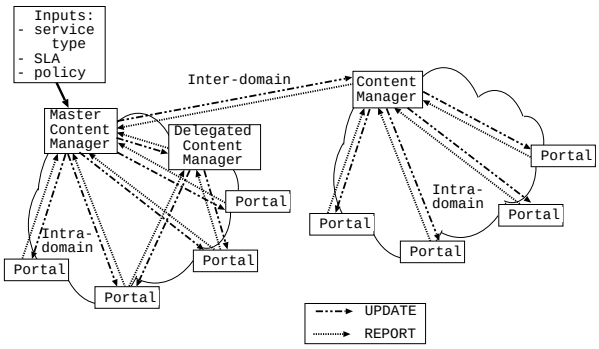

Figure 4: Content Management

from the portals it is managing.

Unlike conventional video-on-demand (VoD) solutions where the content is fairly static, the STV service we are considering has the additional property that new content is being added continuously. Therefore, based on the service bundle being offered to users, the infrastructure needs to be instructed on which TV streams are of interest and which parts of those streams should be stored for what period of time. This adds an aspect to PRISM content management beyond the efficient management of large media objects required by all streaming CDNs.

Figure 4 shows the two main types of messages involved in PRISM content management. (There is also an OPTIONS message which allows discovery of capabilities and mode of operation.)

UPDATE messages: originate from the content manager to instruct the portal regarding policies, resource usage and content manipulation. A content manager can instantiate storage and eviction policies on a portal. The content manager tells portals what content should be obtained when and which storage and eviction policies should be bound to such content. UPDATE messages allow the content manager to tell the portal how its resources should be partitioned and which entity is responsible for managing it. The main resource of concern is portal storage space, but other resources could also be managed in this way.

```
.
.
<Binding>
  <Store>
    rtsp://staging/customerY/firstclip.mov
  </Store>
  <Delete at-utc-time="20011001T000001Z">
    rtsp://staging/customerY/firstclip.mov
  </Delete>
</Binding>
.
.
```

Figure 5: UPDATE message contents: instructions for storing and evicting a file

REPORT messages: carry information from portals to a content manager(s). Messages can be triggered by a request from a content manager or can be sent from a portal as a result of some other trigger e.g. a timer expiring or a threshold being reached. A portal uses this message to inform the content manager about its available resources. This is primarily used at portal startup, allowing the content manager to know what resources it has available. A portal also uses the REPORT message to tell the content manager about content usage statistics and load. The content manager uses this information in dynamic content management decisions. For example, a content manager could instruct a portal to retrieve and store a popular new movie if the set of portals that currently have the movie are becoming loaded.

As indicated in Figure 4, we allow multiple content managers to manage the same portals. This is useful if different content managers are specialized for different services or types of content. For example, one content manager could exclusively deal with a movie-on-demand service, while another could realise STV. Figure 4 also shows interaction between content managers in different administrative domains, facilitating inter-CDN content management. For example in a content brokering [4] arrangement, one CDN might want to push content to the peering CDN before redirecting requests for that content to that CDN.

The essence of the content management exchanges is best conveyed by means of a few simple examples. In concert with widespread industry acceptance for the use of XML [5] technology for machine-to-machine communication, we use XML to encode the exchange of information between content managers and portals. Figure 5 shows an UPDATE message fragment indicating to the portal that it should immediately download the clip indicated. This clip will then be stored at the portal and be deleted at 1 second after midnight on October 1st 2001 (according to the indicated UTC time).

Figure 6 shows part of a slightly more involved UPDATE message. This message creates policies on the portal, manages resource usage on the portal as a whole, and also performs content manipulation.

```
.
.
<Policy>
  <Create> my_stv_policy
    <Type duration="5h">
       DeleteOlderThan
    </Type>
  </Create>
</Policy>
<Associate>
  <Storage allocation="50%"
     manager="cm.att.com" />
  <Storage allocation="50%"
     manager="local" />
</Associate>
<Binding>
  <Store>
    rtsp://prism/prismurn/pbs/*/*/*
  </Store>
  <Delete eviction-policy="my_stv_policy">
    rtsp://prism/prismurn/pbs/*/*/*
  </Delete>
</Binding>
.
.
```

Figure 6: UPDATE message contents: instructions for policy creation, resource usage and file handling

- The first part instructs the portal to create a named policy (`my_stv_policy`) which will evict any content bound to it after it has been stored for five hours.
- The second part of the message tells the portal that 50% of its storage resources will be managed by the specified content manager (`cm.att.com`), while the remaining 50% is available for local caching by the portal (indicated by the fact that this space is delegated to the "`local`" content manager).
- Finally, the portal is instructed to store live content associated with the indicated "PBS" URL. This content will be stored continuously for 5 hours, at which time the oldest part will be evicted.

For lack of space, example REPORT messages are not shown.

2.3 Content Discovery

When a client requests content from PRISM, the redirector first uses content discovery to determine the location of the content. Usually the redirector then redirects the client to a portal that has the requested content, but a client can also be redirected to a portal that does not currently have the content. In this latter case the actual location of the content may be encoded in the redirection URL, or the portal may itself query the content discovery service. (Client redirection is described in Section 2.4.)

PRISM decouples content discovery and content management functions. An alternative would be an integrated approach, where the content management system maintains all of the information used for content discovery. However, the integrated approach has several limitations. First, it does not readily support local storage policies/decisions by portals, e.g. to allow discovery of content that is cached by portals. Second, it is conceivable that different content management solutions with different levels of sophistication or supporting different services will be used within the same infrastructure. In this case, having a single content discovery mechanism is more efficient than having to interface with several content management systems to do content discovery.

PRISM's content discovery architecture is built on a *Mapping Service* that maintains a mapping between content, identified by a URN, and a set of URLs for the content's location. Each portal manages its own content in accordance with content management policies. Each portal dynamically updates information about content stored in the Mapping Service using the PRISM *URI Mapping Protocol* (UMP),[4] and queries the Mapping Service using UMP for the location of a piece of content. The Mapping Service maintains meta-data describing content, such as its type, encoding, and descriptive information. This meta-data can be used to limit responses to queries. UMP reuses both the syntax and semantics of existing protocols, such as HTTP and SIP, and is described below.

UMP was designed to allow several specific architectural models for implementing the Mapping Service, offering different tradeoffs in terms of scalability, reliability, and supporting different administrative models.

2.3.1 Mapping Service Architecture Models

Figure 7 illustrates a portal infrastructure in which portals are organized in neighborhoods, which in practice might be located in geographically separated data centers. The Mapping Service consists of a Local Mapping Server in each neighborhood, and a Global Mapping Server. Local Mapping Servers are updated by local portals using UMP with information about content that they store, and resolve all queries for local content. The Global Mapping Server is updated by every Local Mapping Server, and resolves queries that can't be resolved locally. The reliability and scalability of this approach can be improved by replicating the Mapping Servers. If the load on the Mapping Service is dominated by queries rather than updates (this is likely since queries are triggered by client requests, while content is updated infrequently), then the simple approach of replicating the servers and distributing queries amongst the replicas is attractive, provided that the database size is reasonable. Since the size of the meta-data is orders of magnitude smaller than the size of the (video) objects, it appears that this approach is feasible even for a large streaming CDN.

[4]Note that UMP maps from URNs to URLs. In general, however, it might map between any type of URI and we therefore use the more general term.

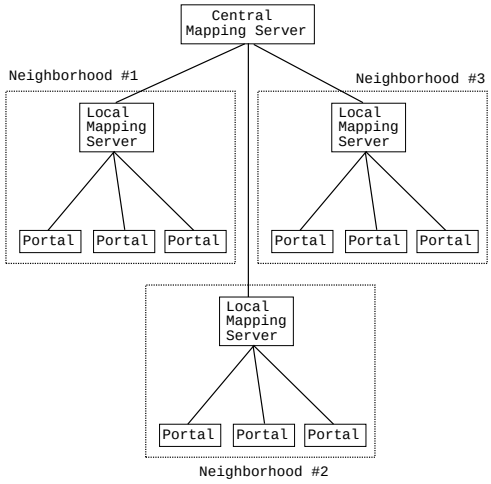

Figure 7: Hierarchical Mapping Service

UMP also allows more distributed architectures for the Mapping Service. In one approach, each portal implements a Local Mapping Server responsible for its content. Queries are multicast to all portals; portals containing the content respond directly to the node initiating the query. In a variation of this approach, portals update a Local Mapping Server in their neighborhood, and Local Mapping Servers multicast queries to remote Mapping Servers for non-local content.

These schemes can be combined and extended to support multiple levels of hierarchy and multiple administrative domains. For example, a CDN can use a hierarchical approach where the Global Mapping Server acts as a gateway to another CDN that uses a different implementation approach (e.g., distributed). Mapping Servers acting as gateways between administrative domains can be configured with policies to control UMP information flow. For example, two CDNs might act as peers so that each CDN is willing to serve specific content to the other CDN. However for other content, the CDNs are not peers. In this case, the gateway nodes would be configured with policies to filter UMP updates crossing the CDN boundary.

2.3.2 URI Mapping Protocol

UMP is a request-response protocol, in that a node that receives a request message always generates a response. Pipelining of requests is allowed so that the order of responses does not need to match the order of requests. On receipt of a UMP request, a node can generate a response or can forward the request to other nodes. In the latter case, this (intermediate) node is responsible for collecting the responses, possibly filtering or aggregating them, and

```
UPDATE rtsp://redirector/prismurn/abc/wabc/comcast/*/ UMP/1.0
Via: UMP/1.0/UDP portal2.att.net
Trans-ID: 87654321@portal2.att.net
TTL: 2
UpdateMode: update
ToURI: rtsp://portal2.att.net/abc.m2t

UMP/1.0 200 OK
Via: UMP/1.0/UDP portal2.att.net
Trans-ID: 87654321@portal2.att.net
```

Figure 8: UMP UPDATE message exchange

```
QUERY rtsp://redirector/prismurn/abc/wabc/comcast/*/ UMP/1.0
Via: UMP/1.0/UDP portal5.att.net
Trans-ID: 12345678@portal5.att.net
TTL: 4
MaxMappings: 5

UMP/1.0 200 OK
Via: UMP/1.0/UDP portal5.att.net
Trans-ID: 12345678@portal5.att.net
FromURI: rtsp://redirector/prismurn/abc/wabc/comcast/*/
ToURI: rtsp://portal2.att.net/abc.m2t
ToURI: rtsp://portal10.att.net/abc.sdp
```

Figure 9: UMP QUERY message exchange

forwarding one or more responses to the request originator. Thus, UMP is extremely flexible: UMP can be used in a simple client-server model, or messages can be forwarded along a mesh of nodes.

There are currently three UMP request methods defined: UPDATE, QUERY, and OPTIONS.

- An UPDATE request is sent to indicate that content identified by one or more From-URIs (or URNs in the PRISM case) is stored at a particular To-URI (or URL for PRISM).
- A QUERY request is sent to request a mapping from a particular From-URI (URN) to one or more To-URIs (URLs).
- An OPTIONS request is sent to request parameters related to UMP operation (e.g., centralized versus distributed, multicast or unicast).

Figure 8 shows a sample UMP update exchange with a request from a portal to a mapping server, and a response from a mapping server to a portal. The UMP UPDATE request message updates the URN mapping for WABC broadcast via Comcast, and indicates that this content is stored on portal2.att.net. The Via header contains the identity of the request originator, portal2, and a transaction id uniquely identifies the transaction. The TTL limits the scope, the mode indicates that this is an update (as opposed to a deletion), and the ToURI gives the URL for accessing the object. The response indicates that the update is successful.

Figure 9 shows a sample UMP query exchange for the default ABC channel. The figure illustrates a request from

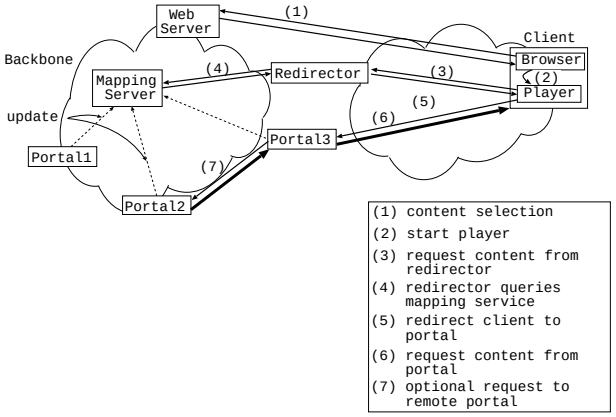

Figure 10: Content-aware redirection

portal to mapping server, and a response from mapping server to portal. The query indicates that the mapping server should return at most five URI mappings. The response indicates that this object is stored on portal2 (as updated above) and at portal10.

An analysis of the scalability and performance of UMP is beyond the scope of this paper. Note however that the number of UPDATE messages exchanged relates to the rate at which live content is being *stored* in and *evicted* from portals. This is a function of the content management policies employed and largely under control of the service provider. QUERY messages are triggered by user requests and are therefore a function of user behavior.

2.4 Content-Aware Redirection

The default PRISM redirection policy is to have the client directed to its local portal in order to allow VCR-like functions on streams to be performed in a fast and responsive fashion. This policy is used even in situations where the content may not currently be stored at the local portal. However, there are situations where streaming content through the local portal is inefficient. This is especially true in cases where the local portal is overloaded. In such situations a client may be directed to other portals to access the content. These aspects of PRISM redirection are also found in existing CDN redirection systems. The PRISM redirection system extends this basic functionality to also take the content location into account during redirection.

Figure 10 shows the steps of the redirection process. First, the client uses a Web-based program guide or some other service to obtain the URN of content to be viewed. Second, this URN, encoded within a URL that points to the redirector, is passed to a running PRISM client, e.g. `rtsp://redirector/prismurn/cnn/*/*/*`. This client may be PRISM-specific, or it may be a legacy RTSP-based media player that is unaware that it is communicating with the PRISM infrastructure. Third,

the client connects to the redirector (as specified in the URL) and requests the content using the RTSP protocol. Fourth, the redirector queries the Mapping Service as to the availability and location of the requested content using the UMP protocol. Fifth, the redirector takes into account the results of the Mapping Service query, the current load, and the proximity of the client and redirects the client to the "best" portal to serve the request. This is typically accomplished with an RTSP redirect, though alternative techniques may be used. Finally, on receipt of the response, the client issues RTSP requests to the portal specified by the redirector to start streaming.

If the portal serving the redirected client does not have the content locally, it must first obtain it before it can start streaming. To prevent another request to the Mapping Service in this case, the URI the client receives from the redirector contains both the original URN as well as the URL where the content can be retrieved from (i.e. the result of the query to the Mapping Service). For example, `rtsp://portal3/prismurn/cnn/*/*/*?url=rtsp://portal2/33.m2t`, means that the client asked for the local live version of CNN and that it can be obtained from the indicated URL. Having both the URN and the URL encoded in the URI presented to the local portal, allows the portal to use the original URN to query the Mapping Service if, for example, it transpires that the supplied URL is no longer valid.

3 Related Work

PRISM's primary focus is on the unique mechanisms required to support the delivery of TV-related services over IP. For example the PRISM naming scheme is designed to easily address arbitrary programs and time-ranges among the entire set of broadcast TV channels worldwide. PRISM requires large amounts of distributed storage capable of archiving all of this content, thus requiring mechanisms to discover where particular content is located. Because of the highly distributed nature and quantity of the stored content, PRISM-specific management, mapping and discovery protocols have been devised. The PRISM content discovery mechanism has some similarity with directory based cooperative caching systems [6]. In cooperative caching, however, finding a copy of an object in a cache is an *optimization* issue because the object can always be retrieved from the origin. This is not the case in PRISM where there is no origin after the content has been aired, and finding a stored version in the infrastructure is crucial for correct operation. Other required PRISM mechanisms have a high degree of commonality with existing and proposed CDNs. For example, the distribution of content between PRISM Live sources and Portals could take place using the application layer multicasting mechanisms of Overcast [7], Scattercast [8] or Yoid [9], instead of IP Multicast. All of the latter three systems focus primarily on the distribution aspects, detailing the mechanisms by which CDN nodes become aware of each other, and form distribution trees. While Overcast can be used to store and distribute broadcast quality TV clips on demand, the details to support large scale TV distribution have not been addressed. PRISM shares some features with Scattercast, such as the use of strategically placed agents/proxies and the use of a distribution architecture on which a range of services can be deployed, but their aims are fundamentally different. Scattercast focuses on providing a well known communication service (multipoint distribution) without the shortcomings of IP Multicast.

There is a considerable body of existing work related to content naming and request routing. In PRISM we employ a general location-independent content naming scheme based on the use of naming URIs to identify content. Locating services by name rather than location was also the basis on which the Intentional Naming System was designed [10]. INS routes client requests to the services of their choice based on a hierarchical naming scheme consisting of name-value pairs. Unlike PRISM, the INS system is not aimed at wide area networks and office automation type applications are described. This part of our work is also related to work-in-progress in the URN working group within the IETF. Specifically the working group has defined how the Dynamic Delegation Discovery System (DDDS) might be realized using extensions to the domain name system (DNS) [11]. Such a system would allow a DNS lookup to point to a resolver other than another DNS server, based on a set of rules applied to the query. For example, a PRISM STV-URN based DNS query could be resolved to an interface associated with a PRISM mapping server or redirector to map the URN to a URL.

In [12] the problem of referencing television broadcasts via Web browsers running on consumer TV appliances such as set-top boxes was considered. That proposal defines a URI for TV broadcast channels, based on the use of DNS-style domain names to uniquely identify TV channels. It addresses a less general problem than our naming scheme, which allows a finer level of detail to be used in referencing content.

From a service point of view, various services are emerging that utilize home based consumer equipment to manage the recording and time-shifting of TV content (e.g., ReplayTV and TiVo). Our work differs from this in that it considers the primary storage medium to be located in the network. This has a number of advantages. In particular, the storage is shared by multiple users and the library of shared content is potentially vast. A similar service concept is described in [13] where sophisticated analysis and indexing techniques are employed to allow users to browse previously aired TV content.

4 Status and Conclusions

The PRISM architecture is being realized as a testbed across a number of geographically dispersed laboratories. With this trial we are verifying the overall architecture through prototype implementation of all of the protocols, systems, and mechanisms that make up a stored-TV service.

The current PRISM testbed consists of a network of portals running prototype software on FreeBSD (www.freebsd.org) servers. Our portal network is distributed among several sites with varying levels of connectivity, including a fairly high capacity intranet in the United States and a public Internet trans-oceanic connection to Cork, Ireland. In addition one of the facilities is equipped with an operational cable plant, allowing experiments over a real broadband access network. The portal software receives and transmits data using standard RTSP/RTP and can therefore deal with any standard compliant streaming software including commercial streaming software. This allows us to make use of the software-only QuickTime client as a low-quality PRISM client. We also have our own PC-based MPEG-2 PRISM client which in addition to providing higher quality allows us to take full advantage of the VCR-like capabilities of the system. In addition we have a self-contained MPEG-2 video endpoint or set-top-box, which allows PRISM-generated content to be viewed on a TV. Finally, the testbed also has a number of live sources which continually encode television and stream it into the network. Content is encoded both in MPEG-2 streams as well as lower quality streams suitable for the QuickTime clients.

In the current operational system, content management is done in static fashion by having each portal read a configuration file at startup telling it where to receive content from and the storage/eviction policy that should be applied to that content. Experience with this simple implementation served as input into the more sophisticated content management system discussed in this paper which is currently being implemented to replace the static solution. Portals inform a centralized implementation of the mapping server of the content they are storing via the PRISM UMP protocol. The current redirector implementation only takes into account the location of the requested content and the location of the requesting client. Users can access this service through two Web based interfaces. One Web interface provides a simple TV-guide like listing of the available content. The second Web interface employs the indexing technology described in [13] to offer search capability of content stored in the PRISM infrastructure.

The stored-TV service we have designed is an example of the more sophisticated services that future CDNs will have to support. We are continuing our work in this domain by extending the capabilities and reach of our testbed and by considering the applicability of our architecture and protocols to current industry activities.

Acknowledgments

The self-contained video-endpoint described in Section 4 was developed by Larry Ruedisueli of AT&T Labs Research. David Gibbon, also from AT&T Labs Research, adapted the SBTV system [13] to allow its use with the PRISM infrastructure.

References

[1] A. Basso, C. D. Cranor, R. Gopalakrishnan, M. Green, C. Kalmanek, D. Shur, S. Sibal, C. Sreenan, and J. van der Merwe, "PRISM, an IP-based architecture for broadband access to TV and other streaming media." Proceedings of NOSSDAV 2000, June 2000.

[2] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifiers (URI) : Generic Syntax." IETF RFC 2396, August 1998.

[3] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)." IETF RFC 2326, April 1998.

[4] A. Biliris, C. Cranor, F. Douglis, M. Rabinovich, S. Sibal, O. Spatscheck, and W. Sturm, "CDN Brokering," Dec 2000. Submitted for publication.

[5] W3C, "Extensible Markup Language (XML)." `http://www.w3.org/XML/`.

[6] S. Gadde, M. Rabinovich, and J. Chase, "Reduce, reuse, recycle: An approach to building large internet caches." 6th Workshop on Hot Topics in Operating Systems, May 1997.

[7] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. James W. O'Toole, "Overcast: Reliable Multicasting with an Overlay Network." OSDI 2000 Conference, October 2000.

[8] Y. Chawathe, *Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service*. PhD thesis, University of California at Berkeley, Dec 2000.

[9] P. Francis, "Yoid: Extending the Internet Multicast Architecture." Unrefereed Report, April 2000. Available from: `http://www.isi.edu/div7/yoid/docs/index.html`.

[10] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley, "The design and implementation of an intentional naming system," *Operating Systems Review*, vol. 34, pp. 186–201, December 1999. 17th ACM Symposium on Operating Systems Principles (SOSP'99).

[11] M. Mealling and R. Daniel, "The Naming Authority Pointer (NAPTR) DNS Resource Record." IETF RFC 2915, Sept 2000.

[12] D. Zigmond and M. Vickers, "Uniform Resource Identifiers for Television Broadcasts." IETF RFC 2838, May 2000.

[13] D. Gibbon, A. Basso, R. Civanlar, Q. Huang, E. Levin, and R. Pieraccini, "Browsing and Retrieval of Full Broadcast-Quality Video." Packet Video Workshop, April 1999. Available from: `http://mediabank.research.att.com/sbtv/`.