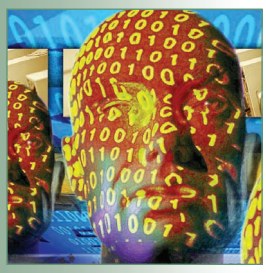

Enhanced Streaming Services in a Content Distribution Network

Prism's content naming, management, discovery, and redirection mechanisms support high-quality streaming media services in an IP-based content distribution network.

Charles D. Cranor, Matthew Green, Chuck Kalmanek, David Shur, Sandeep Sibal, and Jacobus E. Van der Merwe
AT&T Labs–Research

Cormac J. Sreenan
University College Cork

With the emergence of broadband access networks and powerful personal computer systems, the demand for network-delivered full-motion streaming video is growing. While the traditional Web service model can be applied to IP-based streaming content, the user experience of quality does not compare favorably with cable, satellite, or broadcast television. On the other hand, current television broadcasting technologies limit user choices and allow little or no on-demand access to video content.

IP content distribution networks (CDNs) are special-purpose networks that 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, but support for streaming content is typically less sophisticated and often limited to *Webcasting*, the delivery of fairly low-quality live streams, and *clip acceleration*, the distribution of low-to-medium-quality video clips. For Internet-based streaming to approach the same level of entertainment value as broadcast media, CDNs must drastically improve the quality of streaming they can support. Additionally, streaming CDNs will have to support more sophisticated services to accommodate new business models and new types of service offerings.

Prism (Portal Infrastructure for Streaming Media) is a CDN architecture for distributing, storing, and delivering high-quality streaming media over IP networks. The Prism-based stored-TV (STV) service allows users to select content based on the program's name as well as the time it was aired. Content stored inside the network is accessible throughout the whole Prism infrastructure. For example, a user in the U.S. can access European TV content — both live and on-demand. Prism also allows users to specify content to be stored in a "network-VCR" type service.

In this article, we introduce the components of the Prism architecture — content management, content discovery, and

 1089-7801/01/$10.00 ©2001 IEEE

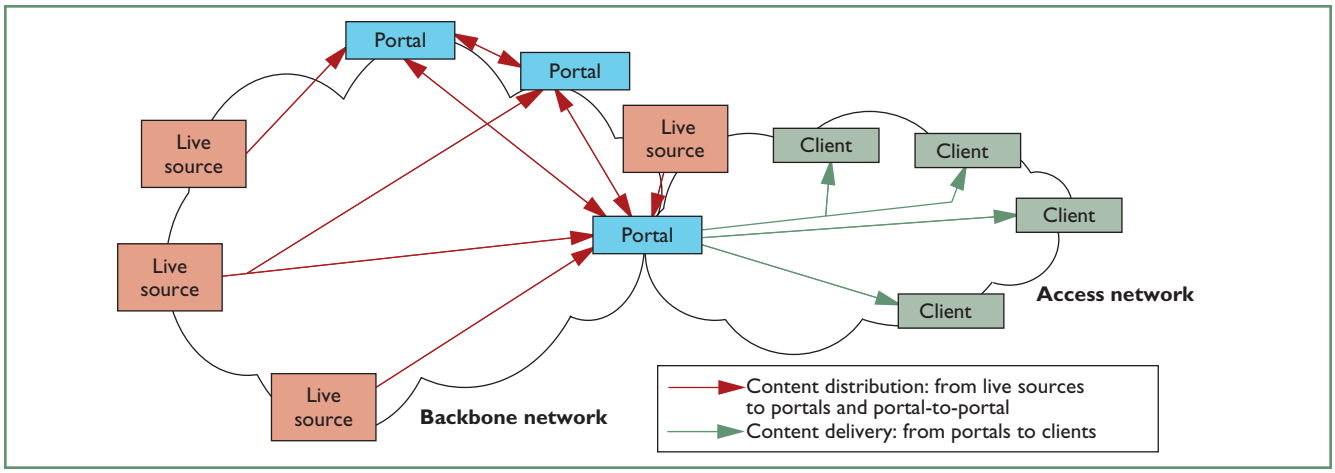

Figure 1. Prism data plane. Content distribution mechanisms transfer content from a live source to one or more portals; content delivery mechanisms pass content from a portal to one or more clients.

content-aware redirection — and briefly describe the status of a trial Prism architecture as well as our future plans for the system. While we focus on applying the Prism architecture to services involving live and stored television content, the framework and architectural components are applicable to other content-based streaming services such as video on demand.

Prism Architectural Elements

The Prism architecture comprises three basic elements.

- *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 it to be viewed on demand, and provide VCR-like functions such as fast-forward and rewind. Portals are positioned between clients and live sources, typically at or upstream of a bandwidth discontinuity such as a cable head end.
- *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.

These three elements, shown in Figure 1, communicate through the network on either the data plane or the control plane. Because the data plane is not unique to Prism (all streaming CDNs require this functionality), we consider it only briefly.

Figure 1 shows Prism's data plane components.

- *Content distribution* mechanisms transfer content from a live source to one or more portals, or transfer content between portals.
- *Content delivery* mechanisms stream content from a portal to one or more clients. Portals should be topologically close to clients in order to provide acceptable perceived performance for latency-sensitive operations such as VCR-like control functions.

A unique aspect of Prism is that it stores live content inside the network for subsequent on-demand access. This differs from on-demand access in conventional CDNs in that there is no well-known origin server where content is known to reside. Prism's control plane regulates the location and flow of content through the Prism infrastructure. Figure 2 (next page) shows the three main Prism control plane components.

- The *content manager* coordinates content storage 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.
- *Content discovery* mechanisms determine the existence and location of streaming content within the Prism infrastructure. Note that while content discovery is triggered by a user request, the actual discovery process, as shown in Figure 2, occurs between Prism entities and is not visible to the user other than through the redirection process.
- *Content-aware redirection* mechanisms forward user requests to the portal that can best satisfy them. The location of the requested content, along with other input, can be used to deter-

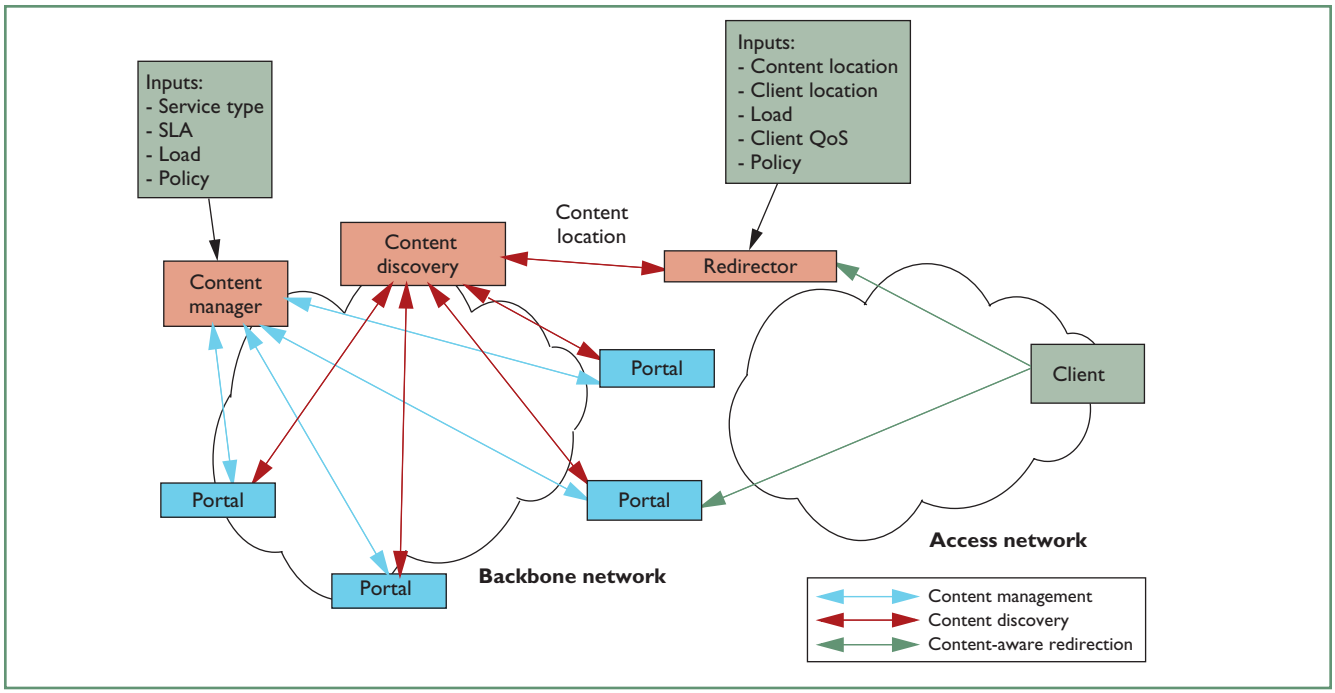

Figure 2. Prism control plane. The control plane regulates the location and flow of content through the Prism infrastructure using content management, content discovery, and content-aware redirection mechanisms.

```
name := "stv:" channel_name
      | "stv:" channel_name "?" specification

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

specification:= "start=" utctime
              | "start=" utctime ":end=" utctime
              | "program=" name
              | "program=" name ";offset=" time
```

Figure 3. Naming scheme syntax. The channel name indicates a unique source of content. The specification indicates a time segment content from a source.

mine the appropriate portal from which content should be served to the customer. All edge servers, including portals, have basic redirection capabilities. However, separate redirection servers, or *redirectors*, typically perform this specialized control plane function.

Another important architectural component is content naming. Before we address the control plane components in more detail, we will describe the content naming scheme Prism uses to uniquely identify all or parts of a broadcast stream.

Content Naming

In Prism, content is referenced by name (uniform resource name, or URN), rather than location (uniform resource locator, or URL). Identifying content by name allows users to access content in a variety of ways without revealing Prism's internal structure. Users can identify content according to a TV schedule or via content-aware search engines. To properly capture localized content from a cable system, the naming scheme must be able to specify which cable head end the content is sent over. Besides identifying content to a fine level of granularity where necessary, the naming scheme should also be compatible with Web protocols. Prism content is generated by digitizing TV channels that are distributed and delivered in real-time by unicast or multicast IP. Because content is accumulated and stored over time, it must be easy to reference by date and time (for example, "show me ABC-TV aired on 30 June, from 9 a.m. to 10 a.m.").

Figure 3 shows the naming scheme syntax. The content-naming URN consists of two parts: *channel name* and *specification*. The channel name consists of four elements, which collectively identify a unique source of content:

- The *brand* is the channel name users typically know. It may be a simple identifier like "ABC" or the brand's fully qualified domain name.
- The *channel* is the call letters or channel number associated with the content. In some cases this field might be null.
- The *distributor* indicates the entity responsible

for distributing the content such as the broadcast station owner, a cable or satellite company, or an Internet-based content distribution system.
- 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.

Table 1. Examples of channel names.

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, and so on)	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, N.J., system	ABC station WABC on Comcast's Orange, N.J., system

The channel name can be further qualified with start and end times or a program name (the specification). Start and end times are expressed as Coordinated Universal Time (UTC) times (for example, "utc:20010215T2200"), as defined in the real-time streaming protocol (RTSP).[1] The program name with optional time offset is a text string identifying content within a particular channel, and the time offset is relative to the start of a program in seconds (such as "program=nypd_blue" or "program=nypd_blue; offset=60"). A URN with no time offset implies the current live version of that content.

All of the channel name elements are optional. The meaning of unspecified elements in a channel name depends on the context in which the name is used. If a channel listing query includes the channel name, then unspecified elements match all the available values in the system. Otherwise, unspecified values take the default values from the profile of the user making the query. Table 1 lists example channel names. Note that end users need not understand or be aware of the channel-naming syntax, which can easily be hidden behind a user-friendly Web interface.

Industry adoption of work being done within the Internet Engineering Task Force's URN working group will allow for the use of URN schemes like ours (see the sidebar, "Related Work on Streaming Media," page 74). In the absence of such a support infrastructure in the current Internet, we follow the common practice of encoding URNs within a URL. For example, `stv:<abc;wabc;;>` can be encoded in an RTSP URL as `rtsp://server/prismurn/abc/wabc/*/*/`.

Content Management

Content management in Prism coordinates content distribution and storage within the network infrastructure. This is done in a way that balances the resources required and those available against the timely delivery of user-requested content. A content manager receives external input, for example from an administrative interface or a configuration file, as well as feedback from the portals it is managing (see Figure 4 on the next page).

Unlike conventional video-on-demand solutions where content is fairly static, our STV service allows new content to be added continuously. The content manager needs to tell the infrastructure which TV streams are of interest and which parts of those streams should be stored for how long.

Figure 4 shows the two main types of messages involved in Prism content management. (There is also an options message that allows content managers and portals to discover capabilities and operation modes of portals.)

- A content manager sends *update* messages to the portal regarding policies, resource usage, and content manipulation. The content manager tells portals what content to obtain when and which storage and eviction policies to bind to that content. Update messages let the content manager tell the portal how its resources should be partitioned and which entity is responsible for managing which partition. Portal storage space is the main resource of concern, but other resources could also be managed in this way.
- *Report* messages carry information from portals to a content manager. Messages can be triggered by a content manager request, a timer expiring, or a threshold being reached. A portal uses this message (primarily at startup) to inform the content manager about its available resources. Report messages also convey content usage statistics and load. The content manager uses this information in dynamic content management decisions. For example, if the set of portals that currently have a particular movie are becoming loaded with

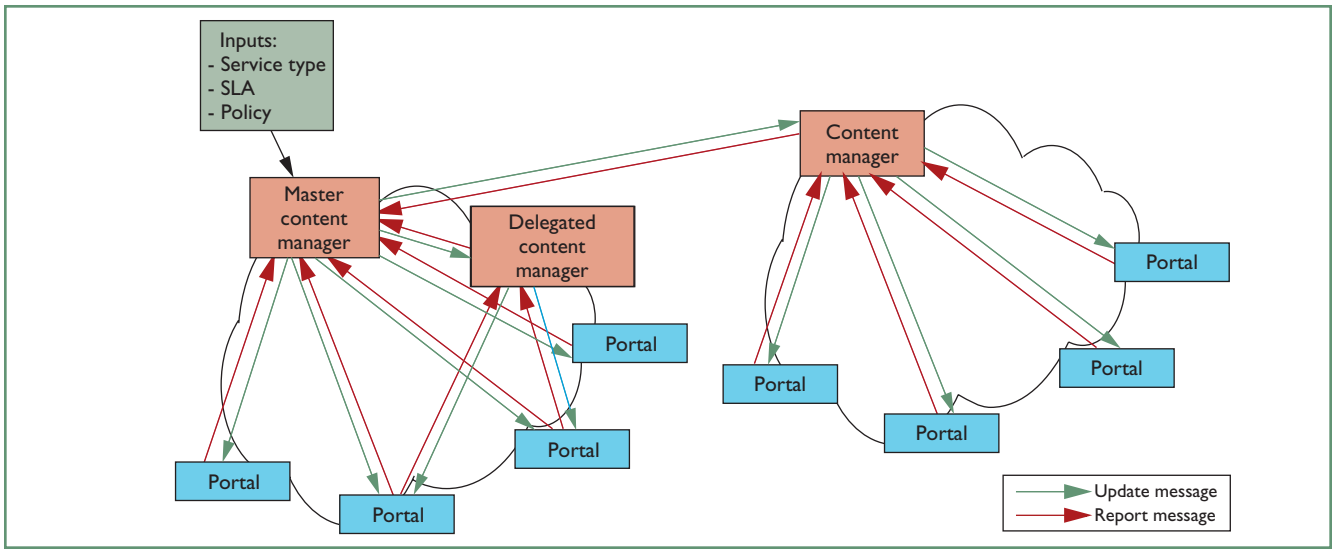

Figure 4. Content management in Prism. A content manager coordinates media storage on a set of distributed portals. A master content manager can delegate some of the portal resources to be controlled by another content manager. Content managers also signal to peer content managers in other administrative domains.

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

Figure 5. An example update message. The message instructs the portal to download a file, store it, and evict it at a specified time.

requests, a content manager could instruct other portals to retrieve and store that movie.

As indicated in Figure 4, multiple content managers can manage the same portals. This is useful if different content managers are specialized for different services or content types. For example, one content manager could exclusively deal with a movie-on-demand service, while another could realize STV. Figure 4 also shows interaction between content managers in different administrative domains, facilitating inter-CDN content management. For example, in a content brokering arrangement,[2] a CDN might want to push content to the peering CDN before redirecting requests for that content to it.

The essence of content management exchange is best conveyed through a few simple examples. We use XML to encode the information exchanged between content managers and portals. Figure 5 shows an update message fragment indicating to the portal that it should immediately download and store the clip indicated by the RTSP URL, and delete it at one second after midnight on 1 October 2001.

Figure 6 shows part of a slightly more involved update message (for lack of space, we do not show example report messages). This message creates policies on the portal, manages resource usage on the portal as a whole, and manipulates content. The first part of the message tells the portal to create a named policy (`my_stv_policy`), which will evict any content bound to it after five hours. The second part tells the portal that 50 percent of its storage resources will be managed by the specified content manager (`cm.att.com`), while the remaining 50 percent is available for local caching (indicated by the delegation of this space 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 five hours, at which time the oldest part will be evicted.

Content Discovery

When a client requests content from Prism, the redirector first uses content discovery to determine the content's location. The redirector then typically sends the client to a portal that has the requested content. When portals that have the content are heavily utilized, the redirector can dynamically increase the number of portals serving the content. In the latter case, the content's actual location

might be encoded in the redirection URL, or the portal might query the content discovery service.

Prism decouples content discovery and content management functions. An alternative is an integrated approach, where the content management system maintains all the information used for content discovery. This approach, however, has several limitations:

- First, it does not readily support local storage policies or decisions by portals (for example, to allow discovery of content cached by portals).
- Second, the same infrastructure can include different content management solutions that vary in their levels of sophistication or service support. An integrated approach would require portals to interface with several content management systems, rather than a single content discovery system, to do content discovery.

Prism's content discovery architecture is built on a *mapping service*, which links content, identified by a URN, to a set of URLs. Each portal manages its own content according to content management policies. The mapping service maintains content metadata, such as its type, encoding, and descriptive information, which can be used to limit responses to queries.

Uniform resource identifier mapping protocol. Portals use the Prism URI mapping protocol (UMP) to dynamically update content metadata stored in the mapping service and to query the mapping service for content location. (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.) UMP reuses both the syntax and semantics of existing protocols, such as the hypertext transfer protocol (HTTP)[3] and the session initiation protocol (SIP).[4]

UMP is a request-response protocol, where a node receiving a request message always generates a response. UMP allows the issuance of concurrent requests to improve performance. Each UMP message carries a transaction identifier so that the order of responses does not have to match the order of requests. When a node receives a UMP request, it can immediately generate a response or it can first forward the request to other nodes. In the latter case, the intermediate node collects the responses from the other nodes, possibly filtering or aggregating them, and forwards one or more responses to the request originator. Thus, UMP is extremely flexible: it can be used in a simple client-server model, or it can be forwarded along a mesh of nodes.

```
.
.
<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. A complex update message. The message gives instructions for policy creation, resource usage, and file handling.

There are currently three UMP request methods:

- An *update* message indicates that content identified by one or more FromURIs (or URNs) is stored at a particular ToURI (or URL).
- A *query* requests a mapping from a particular FromURI to one or more ToURIs.
- An *options* message requests parameters related to UMP operation (for example, centralized versus distributed, multicast, or unicast).

Figure 7 (next page) shows a sample UMP update exchange between a portal and a mapping server. The UMP update request message indicates that the content for a WABC broadcast via Comcast is stored on portal2.att.net. The *Via* header identifies the request originator, portal2; and a transaction ID uniquely identifies the transaction. The time-to-live (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 was successful.

Figure 8 (next page) shows a sample UMP query exchange for the default ABC channel between a portal and a mapping server. 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.

```
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 7. UMP update message exchange. The exchange indicates the request originator, the transaction ID, the message type, and where the requested content is stored.

```
QUERY rtsp://redirector/prismurn/abc/wabc/comcast/*/ UMP/1.0
Via: UMP/1.0/UDP portal5.att.net
Trans-ID: 12345678@portal2.att.net
TTL: 4
MaxMappings: 5

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

Figure 8. UMP query message exchange. The query requests a mapping from a URN to no more than five URLs. Two mappings are returned in the successful response.

Mapping service architectural models. UMP allows several specific architectural models for implementing the mapping service, offering different trade-offs in terms of scalability, reliability, and support for different administrative models.

Figure 9 illustrates an example portal infrastructure in which portals are organized in neighborhoods, which are located in geographically separated data centers. The mapping service consists of local mapping servers in each neighborhood and a global, or central, mapping server. Using UMP, local portals send updates about their stored content to local mapping servers, and local mapping servers update the global mapping server. Local mapping servers resolve all queries for local content, and the global mapping server resolves queries that cannot be resolved locally.

Replicating the mapping servers improves the reliability and scalability of this approach. If queries other than updates dominate the mapping service load (which is likely, since queries are triggered by client requests, while content is updated infrequently), then simply replicating the servers and distributing queries among the replicas is an attractive approach, provided that the database size is reasonable. Since the metadata is orders of magnitude smaller than video objects, this approach is feasible even for a large streaming CDN.

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 nonlocal content.

We can combine these schemes 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 (such as a distributed approach). We can configure these mapping servers with policies to control UMP information flow. For example, two CDNs might act as peers so that each CDN serves specific content to the other. For other content, however, the CDNs are not peers, so the gateway nodes would be configured with policies to filter UMP updates crossing the CDN boundary.

An analysis of UMP's scalability and performance is beyond the scope of this article. 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 portal's content management policies and is largely under the service provider's control. Query messages are triggered by user requests and are therefore a function of user behavior.

Content-Aware Redirection

The default Prism redirection policy is to send the client to its local portal to allow quick and responsive VCR-like functions on streams, even when the content is not currently stored at the local portal. There are situations, however, where streaming content through the local portal is inefficient. This is especially true when the local portal is overloaded. In such situations, the redirector might forward a client to other portals to access the content. These aspects of Prism redirection are similar to those in existing CDN redirection systems; however, Prism extends redirection to take into account content location.

The seven steps in Prism's redirection process are:

1. Using a Web-based program guide or some other service, a user's browser obtains the URN of content to be viewed.
2. The Web browser passes this URN, encoded within a URL that points to the redirector, to a running Prism media player. This media player may be Prism-specific, or it may be a standard RTSP-based media player that is unaware that it is communicating with the Prism infrastructure.
3. The Prism client connects to the redirector specified in the URL and uses RTSP to request the content.
4. The redirector sends a UMP query to the mapping service for the availability and location of the requested content.
5. 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 portal best able to serve the request. This is typically accomplished with an RTSP redirect, but alternative techniques can be used.
6. On receiving the response, the client issues RTSP requests to the portal specified by the redirector to start streaming.
7. If the portal serving the redirected client does not have the content, it must obtain it from a remote portal before it can start streaming.

To avoid additional requests to the mapping service, the URI sent by the redirector to the client contains both the original URN and the URL where the content can be retrieved (that is, the result of the mapping service query). 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.

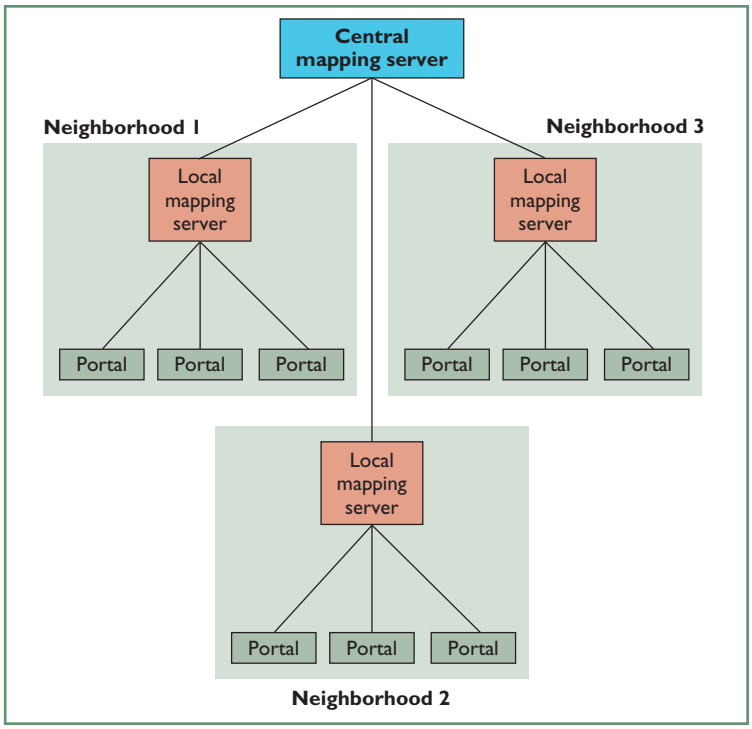

Figure 9. Hierarchical mapping service. Portals report detailed content information to a local mapping server. Local mapping servers aggregate this information and report it to a central mapping server.

System Status and Ongoing Work

The Prism architecture is being implemented as a testbed across a number of geographically dispersed laboratories. With this trial, we are verifying the overall architecture through prototype implementation of all the protocols, systems, and mechanisms that make up an STV service.

The current Prism testbed consists of a network of portals running prototype software on FreeBSD servers (http://www.freebsd.org). 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 transoceanic connection to Cork, Ireland. One of the facilities also 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 for control and the real-time protocol (RTP)[5] for data transfer. It can therefore deal with any standard-compliant streaming software including commercial streaming software. This lets us use 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 video, lets us take full advantage of the system's VCR-like capabilities. 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 has a number of live sources that continually encode television and stream it into the network. Content is encoded both in MPEG-2 streams and lower-quality streams suitable for the QuickTime clients.

In the current operational system, content management is static. At startup, each portal reads a configuration file, which tells it where to obtain content and what storage/eviction policy to apply

Related Work on Streaming Media

Prism focuses on the mechanisms required to support delivery of TV-related services over IP. Because of the highly distributed nature and quantity of the stored content, we have devised Prism-specific management, mapping, and discovery protocols. Our work is related to several efforts in the general area of content distribution.

Cooperative Caching

The Prism content discovery mechanism is similar to directory-based cooperative caching systems.[1] 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 server. This is not the case in Prism where there is no origin server after the content has been aired, and finding a stored version in the infrastructure is crucial for correct operation.

Content Distribution

Other required Prism mechanisms are similar to mechanisms in existing and proposed CDNs. For example, content could be distributed between Prism live sources and portals using the application-layer multicasting mechanisms of Overcast,[2] Scattercast,[3] or Yoid,[4] or native IP multicast. The three aforementioned systems detail 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, it does not address large-scale TV distribution.

Like Scattercast, Prism uses strategically placed agents and 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.

Naming and Routing

Prism's general location-independent content-naming scheme uses URNs to identify content. Locating services by name rather than network address was also the basis for the Intentional Naming System.[5] 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 its focus is office automation applications.

Our work on naming is also related to work in progress in the IETF's URN working group. Specifically, the working group has defined how the Dynamic Delegation Discovery System (DDDS) might be realized using extensions to the domain name system.[6] This system would allow a DNS lookup to point to a resolver other than another DNS server, based on a set of

continued on p. 75

to that content. Portals use the Prism UMP protocol to inform a centralized mapping server of their stored content. The current redirector implementation takes into account only the locations of the requested content and the requesting client. Users can access this service through two Web-based interfaces. One interface provides a simple TV-guide-like listing of the available content; the other employs the indexing technology described in Gibbon et al.,[6] which lets users search content stored in the Prism infrastructure.

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 to deliver 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.

Acknowledgments

A preliminary version of this article, "Prism, an IP-Based Architecture for Broadband Access to TV and Other Streaming Media," was presented as a work-in-progress report at the 10th International Workshop on Network and Operating System Support for Digital Audio and Video.[7] Larry Ruedisueli of AT&T Labs–Research developed the self-contained video endpoint described in the section, "System Status and Ongoing Work." David Gibbon, also from AT&T Labs–Research, adapted the SBTV system to allow its use with the Prism infrastructure.

References

1. H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)," IETF RFC 2326 (proposed standard), Apr. 1998, available at http://www.ietf.org/rfc/rfc2326.txt.
2. A. Biliris et al., "CDN Brokering," *Proc. Sixth Int'l Workshop Web Caching and Content Distribution*, Elsevier, Boston, June 2001.
3. T. Berners-Lee, R. Fielding, and H. Frystyk, "Hypertext Transfer Protocol — HTTP/1.0," IETF RFC 1945 (informational), May 1996, available at http://www.ietf.org/rfc/rfc1945.txt.
4. M. Handley et al., "SIP: Session Initiation Protocol," IETF RFC 2543 (proposed standard), Mar. 1999, available at http://www.ietf.org/rfc/rfc2543.txt.
5. H. Schulzrinne et al., "RTP: A Transport Protocol for Real-Time Applications," IETF RFC 1889 (proposed standard), Jan. 1996, available at http://www.ietf.org/rfc/rfc1889.txt.
6. D. Gibbon et al., "Browsing and Retrieval of Full Broadcast-Quality Video," *Packet Video Workshop*, Apr. 1999, New York, available at http://www.research.att.com/~mrc/pv99/CONTENTS/PAPERS/GIBBON/sbtv_pv99last.htm.

Related Work on Streaming Media continued

continued from p. 74

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.

Zigmond and Vickers[7] propose a solution for referencing television broadcasts via Web browsers running on consumer TV appliances such as set-top boxes. Their 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 Prism to use a finer level of detail in referencing content.

Content Storage

Various emerging services utilize home-based consumer equipment to manage the recording and time-shifting of TV content (for example, ReplayTV and TiVo). Our work differs from these systems in that Prism 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. Gibbon et al.[8] describe a similar service that employs sophisticated analysis and indexing techniques to allow users to browse previously aired TV content.

References

1. S. Gadde, M. Rabinovich, and J. Chase, "Reduce, Reuse, Recycle: An Approach to Building Large Internet Caches," *Proc. Sixth Workshop Hot Topics in Operating Systems*, IEEE Computer Soc. Press, Los Alamitos, Calif., May 1997, pp. 93-98.
2. J. Jannotti et al., "Overcast: Reliable Multicasting with an Overlay Network," *Proc. Fourth Symp. Operating Systems Design and Implementation*, Usenix Assoc., Berkeley, Calif., Oct. 2000, pp. 197-212.
3. Y. Chawathe, "Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service," doctoral dissertation, Univ. of California at Berkeley, Dec. 2000.
4. P. Francis, "Yoid: Extending the Internet Multicast Architecture," unrefereed report, Apr. 2000, available at http://www.isi.edu/div7/yoid/docs/index.html.
5. W. Adjie-Winoto et al., "The Design and Implementation of an Intentional Naming System," *Operating Systems Rev.*, vol. 34, Dec. 1999, pp. 186-201.
6. M. Mealling and R. Daniel, "The Naming Authority Pointer (NAPTR) DNS Resource Record," IETF RFC 2915 (proposed standard), Sept. 2000, available at http://www.ietf.org/rfc/rfc2915.txt.
7. D. Zigmond and M. Vickers, "Uniform Resource Identifiers for Television Broadcasts," IETF RFC 2838 (informational), May 2000, available at http://www.ietf.org/rfc/rfc2838.txt.
8. D. Gibbon et al., "Browsing and Retrieval of Full Broadcast-Quality Video," Packet Video Workshop, New York, Apr. 1999, available at http://www.research.att.com/~mrc/pv99/CONTENTS/PAPERS/GIBBON/sbtv_pv99last.htm.

7. A. Basso et al., "Prism, an IP-Based Architecture for Broadband Access to TV and Other Streaming Media," *Proc. 10th Int'l Workshop Network and Operating System Support for Digital Audio and Video*, Univ. of North Carolina at Chapel Hill, June 2000, available at http://www.nossdav.org/2000/abstracts/10.html.

Charles D. Cranor is a senior technical staff member at AT&T Labs–Research, Florham Park, New Jersey. His interests include networking, operating systems, and computer architecture. Cranor received an MS and a DSc in computer science from Washington University, St. Louis, Missouri. He is a member of the IEEE, the ACM, and Usenix.

Matthew Green is a developer with AT&T Labs–Research. His research interests include streaming audio and digital rights management research. He holds a BA in computer science from Oberlin College and a BMus from Oberlin Conservatory, with a specialization in electronic music.

Chuck Kalmanek is manager of the Networking Research Division at AT&T Labs–Research. His research interests include multimedia systems and IP transport over optical networks. He has a BS in applied physics from Cornell University, an MS in electrical engineering from Columbia University, and an MS in computer science from New York University.

David Shur is a technology consultant at AT&T Labs–Research, where he works on multimedia streaming and multicasting. He received a PhD in electrical engineering from Stanford University. Shur was appointed Distinguished Member of Technical Staff at AT&T Bell Labs in 1991. He is a senior member of the IEEE.

Sandeep Sibal is a senior technical staff member at the Internet and Networking Systems Research Center, AT&T Labs–Research. Sibal has an MS in electrical and computer engineering from Rice University and a PhD in electrical, computer, and systems engineering from Rensselaer Polytechnic Institute. He is a member of the IEEE and the ACM.

Cormac J. Sreenan is professor and head of the Department of Computer Science at University College Cork, Ireland. His research interests are in the areas of mobile and Internet systems, including streaming media, packet telephony, and mobile/wireless computing. He has a PhD from the University of Cambridge and is a member of the IEEE.

Jacobus (Kobus) E. Van der Merwe is a principal technical staff member at AT&T Labs–Research. He has BEng and MEng degrees in electronic engineering from the University of Pretoria in South Africa, and a PhD in computer science from the University of Cambridge in England. He is a member of the IEEE.

Readers can contact Van der Merwe at kobus@research.att.com.