Streaming and Broadcasting over the Internet

Markus Hofmann and Krishan Sabnani, Bell Labs Research, Holmdel, NJ, USA

Abstract

As the number of people accessing the Internet grows and commerce becomes an important part of the Internet, a desperate need for deployment of scalable and reliable networking technologies arises. The need for scalability is further exacerbated by bandwidth intensive applications using video and audio streaming services. Adding more bandwidth and QoS support to the Internet is one potential remedy for performance problems, but large scale deployment is costly and will take a long time. Recently, a new breed of service providers emerged offering complementary solutions encompassing technologies such as caching, enhanced routing, and load balancing. These providers are mostly called *Content Delivery Service Providers (CDSPs).* This paper discusses techniques for building enhanced content delivery services over best-effort networks. While the described mechanisms do not assume any specific support from the network, they are also useful in QoS enabled networks to improve network efficiency and user experience. Particular focus is on support for streaming and broadcast services, including architectural design, protocol related issues, and mechanisms for traffic redirection.

1 Introduction

Mass acceptance of the Internet as a tool for retrieving and delivering information has placed a serious strain on the infrastructure upon which it is built. Emerging broadband technologies such as cable modem and DSL will aggravate the problem, by forcing servers, routers, and backbone links to perform even faster. In addition, a significant increase of commercial products for resource intensive playback of streamed video and audio has occurred over the past several years, which makes things even worse. As a result, consumers typically experience low quality, due to high delay, unstable throughput, and loss of packets in the best-effort model of the Internet. Adding more bandwidth and QoS support to the Internet is one potential remedy for performance problems. But while this may provide some relief, it does not address the fundamental problem of overloaded servers and data traversing multiple domains and many networks. In addition, deployment of quality-of-service enabled systems is costly, difficult and lengthy. Even when high quality network services are available, people might prefer to use best-effort services if the cost is lower. Network providers are also concerned with scaling up to meet spikes in data traffic. It is difficult to handle unpredicted spikes, such as breaking news stories, which overwhelm web sites with unexpected demand.

A promising approach in overcoming the Internet's notorious bottlenecks and slowdowns is distributing and moving content to the edge of the network where it becomes faster to retrieve. User requests are then redirected to and served from these edge devices. Web caching is an example of such a service. A web cache, preferably in close proximity to the client, stores requested web objects (e.g. HTML pages or images) in an intermediate location between the object's origin server and the client. Subsequent requests can be served from the cache, thus shortening access time and conserving significant network resources – namely bandwidth and processing resources. Caching of web objects has been studied extensively starting with simple *proxy caching* [1], followed by improvements in *hierarchical caching* and *cooperative caching* under the Harvest project [2] and the Squid project [3], respectively. For web caching to be effective on a scale required by ISPs and enterprises, it must integrate methods for active page management, content freshness, load balancing and replication. Lucent's web caching solution, IPWorX [4], is an example of a next generation caching system.

Only recently, there has been work done to extend web caching systems to support streaming media [5] [6] [7]. Compared with caching for traditional web content, streaming media presents new challenges. It is delay-intolerant and requires start-to-finish transmission without interruptions. The real time requirements of multimedia streaming contribute to the complexity of a streaming cache. Many of the server functionalities such as scheduling and resource management have to be incorporated in the cache design. Furthermore, multimedia files tend to be quite large, requiring long transmission times and making it infeasible to always cache them in their entirety. A single, two hour-long MPEG movie, for example, requires about 1.4 Gbytes of disk space. Given a finite buffer space, only a few streams could be stored at a cache, thus, decreasing the hit probability and the efficiency of the caching

system. Streaming caches also have to support additional protocols besides HTTP, such as the *Real Time Transport Protocol (RTP)* [8] and the *Real Time Streaming Protocol (RTSP)* [9].

We have explored and assessed several techniques addressing these challenges. We considered them in an integrated fashion and developed a unified architecture to better support streaming media over the Internet [5]. The work resulted in the implementation of an RTP/RTSP based caching system for streaming media [10].

This paper discusses some issues to be considered in the design of a caching architecture for streaming and broadcasting over the Internet. Section 2 examines design alternatives for streaming caches, followed by Section 3 that talks about protocol related issues. Section 4 discusses different mechanisms for redirecting user requests to an appropriate cache. Section 5 explores future directions and develops a vision for more general and open content delivery platforms.

2 Caching of Streaming Media

Several fundamental differences between streaming media and conventional web objects such as text and images have to be considered when designing a streaming cache. The large size of most streaming objects, for example, makes conventional caching techniques inappropriate. Rather than storing large streaming objects in an all-or-nothing fashion, a streaming cache could store a subset of the most frequently requested frames of the most popular clips. Previously, *prefix caching* [6] has been proposed to address this challenge. In prefix caching, a proxy caches a set of frames from the beginning of the clip (i.e. the prefix of the clip). This allows the cache to immediately serve an incoming user request from disk, while simultaneously requesting the remainder of the clip from the origin server. The prefix should be large enough to hide round-trip delays and to absorb jitter in the client-cache communication. Larger prefixes allow for more time in fetching the remainder of the clip, but they reduce the number of prefixes that can be stored at the cache. These trade-offs have to be considered when defining the prefix size.

Prefix caching assumes that users will request an audio or video clip from the beginning. This might not always be the case. News stations, for example, typically record an entire news broadcast into a single video file. A corresponding link on the station’s web page lets interested users access and watch the recording. Given that users are not very likely to be equally interested in all parts of the broadcast, some stations provide index links to each of the subsections of the broadcast. For example, there might be links to headline news, sports, or weather. Typically, index links point to the same video clip, but they use different time ranges to request only the portion of a clip in which the user is interested [11]. As a result, it might very well be the case that the most frequently accessed frames are *not* at the beginning of the clip. If most users are interested in the middle part of a news broadcast (e.g. in the latest baseball reports), strict prefix caching does not provide much benefit. We have therefore designed a more flexible caching scheme that automatically divides streaming objects into smaller segments based on the user request patterns. By doing so, the segments can be cached independently and, therefore, disk space can be most efficiently used to cache popular portions of large clips. A detailed description and evaluation of the caching scheme can be found in [5].

A drawback of such independent segment caching is the added complexity for composing the outgoing data stream. A cache uses locally stored segments to satisfy a client request, but it has to fetch the missing parts from the server or from another cache. The different portions of the clip have to be linked together to create a single, consistent data stream that will be forwarded to the client. Details on how to compose a single stream from multiple segments are given in [10].

3 Protocol Issues

Caching systems can play an important role in improving user experience and network performance for streaming multimedia over the Internet. For ease of deployment, a caching service should not require modification of existing client/server applications or communication protocols. Although HTTP can be used for rudimentary streaming services, the RTP/RTSP protocols have been widely accepted and are supported by all major streaming applications. We have therefore chosen to base our work on the RTP/RTSP protocol suite.

RTP provides the basic functionality for transferring real-time data over the Internet. It supports time-stamping, identification of different payload types, source identification, and sequence numbering. RTP can run over TCP or UDP, the latter one typically being used to stream across firewalls that discard UDP packets.

RTSP is used to control the delivery of multimedia streams. It implements all the functions of a "VCR control", allowing a user to start, stop, play, forward, or rewind a data stream. The protocol typically runs over TCP, though UDP can also be used. Unlike HTTP, RTSP is a stateful protocol, meaning that it maintains per session information. RTSP defines different methods for session establishment and control.

First, the OPTIONS method is used for getting information about server capabilities (e.g. supported protocol version). The DESCRIBE method inquires about the properties of a particular stream (e.g. streaming rate). Information about the data transport method (e.g. port numbers and use of UDP or TCP) is exchanged via the SETUP method. The PLAY method initiates data transfer and allows the specification of a particular time range to control the playback interval. A session is terminated with a TEARDOWN method. More methods are defined by RTSP, but we will not consider them in the reminder of the paper.

The usage of RTSP and RTP in the caching context is illustrated in Figure 1. The example shows a case where the cache has stored the beginning of a stream, but it has to fetch the remainder of the clip from the original server (or from another cache).

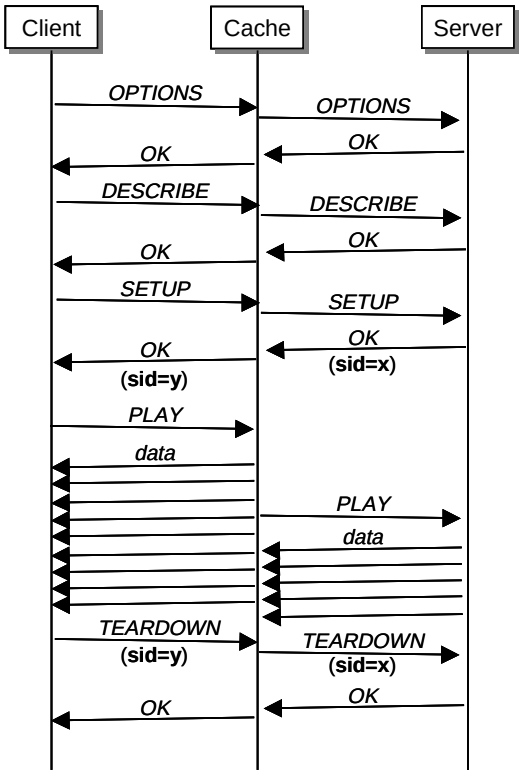

Figure 1: RTSP Message Exchange

The cache passes the OPTIONS, DESCRIBE, and SETUP messages from the client to the server and vice versa. Note that two *separate* sessions between client and cache and between cache and server are established. Therefore, it is necessary for the cache to translate some of the message fields, such as sequence number, session identifier, and transport headers. In Figure 1, for example, the session between client and cache was assigned the session identifier SID=y, while the session between cache and server has SID=x. On receiving a PLAY message, the cache starts transmitting the requested data to the client using RTP. In the example, the cache has already stored the beginning of the requested clip. Before the cache runs out of stored data, it sends a PLAY message with an adjusted *range header* to the server in order to request the remainder of the clip. TEARDOWN messages from the client are passed on to the server and both sessions are terminated.

One would expect that the cache could use previously received information to respond directly to a client's OPTIONS, DESCRIBE, or SETUP request, avoiding the delay in contacting the server. However, the RTSP implementation in media players and servers from Real Networks [12], for example, uses proprietary headers with encrypted header fields. Without knowing the interpretation of these fields, a cache is not able to generate the expected answer. As a result, a session cannot be established and no data can be transferred. Our cache implementation avoids this problem by forwarding OPTIONS, DESCRIBE, and SETUP messages to the server - even if the entire data was available at the cache. The benefit of complete transparency and independence of proprietary mechanisms comes at the cost of slightly increased setup delays. Note, however, that the additional delay for contacting the server is almost negligible. Due to client side data buffering, the observed playback delay is typically in the range of 5-10 seconds, while the round-trip time for an RTSP message is typically less than half a second. Forwarding RTSP messages also allows for authentication and authorization by the origin server, which is useful for access control, billing, and charging. It also allows verifying the freshness of cached objects, before forwarding them to the client.

The described mechanism uses existing RTSP/RTP protocol mechanisms and does not require any modifications to the client/server software. Alternative approaches might use different mechanisms for data transfer between server and cache. The Real Networks Proxy [12], for example, uses a rate controlled file transfer to ship data from a server to a cache. This solution requires installation of additional software plug-ins at both sides, the server and the cache.

4 Traffic Redirection

Web caching and streaming caching require that client requests be redirected to a nearby cache, rather than being forwarded to the origin server. To accomplish this, the user can elect to use a caching service and manually configure the cache to which the request will be redirected. Common web browsers, for example, allow the user to specify proxies for various services. However, explicit *client configuration* has several drawbacks:

- It puts administrative burden on the user, who must be capable to configure the software properly (i.e. she needs to have the knowledge, the permission, and the information necessary to configure the software).
- User based redirection is static. Once a user has properly configured its software, requests will always be redirected to the specified location. There is no possibility for dynamic redirection using load-balancing methods.
- Proper operation and usage of redirection rely on the user himself. Network service providers can hardly control whether and how a user configures service redirection.

For these reasons, explicit client-side proxy configuration may not really be an option. When this is the case, the service provider needs to be able to reroute traffic transparently without the need for reconfiguration of client software. Transparent redirection involves the interception and redirection of user requests to one or more service boxes, transparently to the client.

There are several ways to perform transparent redirection, depending on the kind of control a service provider has on the client's access network. If the service provider has control over the client's access network, he can deploy boxes handling the transparent redirect. In this case, we talk about *network-integrated redirection*, which is provided either by routers (*Router-assisted Redirection*) or by dedicated boxes (*L4/L7 Redirection*).

With router-assisted redirection, routers in the access path of the clients are configured to handle the transparent redirection. When such a router receives an IP packet, it determines if the packet is a client request that should be redirected to a network storage device. In the case of HTTP redirection, the router looks for TCP as the protocol field in the IP header and for 80 as the destination port in the TCP header. If the packet meets these criteria, it is redirected to a web-cache. Other combinations of protocol and port number are possible to support services different from HTTP redirection (e.g. for streaming using RTSP). A router is aware of available network storage devices either through manual configuration or through running the *Web Cache Coordination Protocol (WCCP)* [13]. WCCP allows a router enabled for transparent redirection to discover, verify, and advertise connectivity to one or more web-caches. WCCP is a relative simple protocol specified by Cisco Systems. It has to be implemented on both sides, the caches and the router, i.e. existing routers in the client's access network might have to be updated or replaced. Currently, Cisco is providing WCCP support in their router products.

It remains an open question whether the integration of transparent redirection has a negative impact on a router's performance. Besides packet forwarding, a router now has to look at the protocol ID and the port number of all incoming IP packets. Splitting the tasks and having a separate box doing transparent redirection might achieve better performance. This approach has the additional advantage of being independent from the router technology in the access network. The idea of separate boxes leads to the design of layer four (L4) and layer seven (L7) switches.

A L4 switch either replaces the local hub/switch or sits between the client's access point and the first router. It looks at the protocol ID and the port number of all incoming packets to determine if the packet should be passed on or redirected to a network storage device. Most L4 switches support redundant storage devices, automatic failover, quality of service, and other features. Load balancing can be performed in several ways, including least number of connections, round robin, least busy (derived by the L4 switch), or hash algorithms. When the simple load balancing methods used by L4 switches are inadequate, L7 switches can be used. As the name implies, L7 switches work at the application layer of the OSI stack, taking advantage of application specific information (e.g. the URL name included in an HTTP request).

Network integrated redirection requires control over the client's access network for deployment of redirection enable routers or L4/L7 switches. This approach is appropriate for ISPs or enterprises offering a value-added service to their customers. Provisioning of content-provider oriented services, however, would require deployment of network integrated redirection devices in all access networks worldwide, which is not a feasible solution. Traffic redirection in a global environment requires different mechanisms, for example, DNS-based redirection.

In the Internet, the *Domain Name System (DNS)* [14] is used to map host names into actual IP addresses. A client sends a request with a host name to a DNS server, which returns the corresponding numerical IP address. Normally, the mapping between host names and IP addresses is relative static and based on manual configuration. It is possible, however, to modify the mapping mechanism to allow a dynamic name resolution based on current network and server statistics. Such an approach can be used for transparent traffic redirection. The example shown in Figure 2 illustrates the procedure.

Assume a client wants to access the CNN web page at http://www.cnn.com/. The client's browser first has to resolve the host name "www.cnn.com" to a numerical host IP address. If the mapping information is not available at the client, it sends a DNS request to its local DNS server (see (1) in Figure 2). If the local DNS server is not able to resolve the name, it will contact the authority name server for the domain

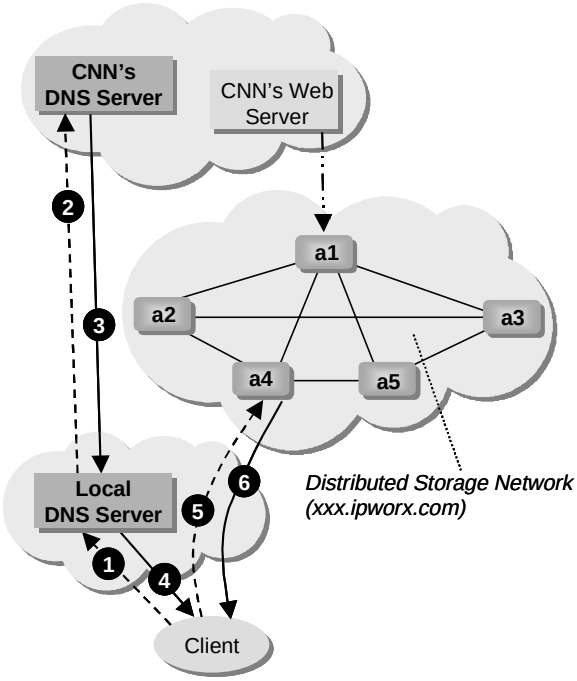

Figure 2: Example for DNS-based Redirection

"cnn.com" (see (2) in Figure 2). Rather than return a host machine in its own local network, the CNN authority server returns a storage host that is close to the user. The problem is that the name server does not know the identity of the client itself, but only of the local DNS server sending the request on behalf of the client. It will therefore use the IP address of the local DNS server as an approximation for the client's locality. Finding the best storage device depends on various metrics: content availability, network status, storage load, etc. In our example, the storage device "a4.ipworx.com" is the most promising candidate currently caching the requested document. Therefore, CNN's name server returns the IP address of "a4.ipworx.com" (see (3) in Figure 2). Finally, the answer will be forwarded to the client (see (4) in Figure 2), which will now use the IP address of "a4.ipworx.com" to request the web page (see (5) and (6) in Figure 2). The local DNS server will cache the mapping information according to the returned TTL value. Therefore, subsequent name resolution requests from the same client area might be answered locally, requiring only steps (1), (4), (5), and (6).

The described mechanisms for traffic redirection each have their own advantages and disadvantages. While DNS-based redirection might not reveal as good performance as network integrated redirection, it is more suitable for global services as it does not require the installation of switching devices in access networks. Each mechanism has its place in different scenarios meeting different requirements.

5 Future Directions

So far, we have focused our efforts on caching services. Caching has the advantage that requests for content do not have to travel across the Internet to the origin server. Instead, the requests can be served from a nearby cache, often located in an ISP's point-of-presence (POP) or data center. Caching benefits the ISPs by reducing their upstream bandwidth costs. It also enables a better experience for their customers, because their requests can be served faster from a local cache. In general, caching services support the ISP and its customers in obtaining data from potentially all content providers. We therefore refer to caching as a *content-consumer oriented* service, because it offers a value-add to the consumers. Recently, a new breed of service providers emerged, which are mostly called *Content Delivery Service Providers (CDSPs)*. CDSPs inverse the business model of caching. They offer to help particular content providers delivering their content to potentially all consumers on the Internet. It is the content provider and not the consumer or its ISP who is paying for the service. We therefore refer to this kind of service as *content-provider oriented*. CDSPs enhance the user experience for certain web content by delivering content and applications using the previously discussed technologies. They place the equipment that implements the technologies as close to as many users as possible, typically inside multiple ISPs. In most cases, the CDSP does not own the networking infrastructure, which is why they normally use network-independent traffic redirection.

The success of current content-delivery services shows the enormous potential of content-oriented solutions to the Internet slowdown. In the future, we anticipate the emergence of a variety of content-oriented solutions. We believe that there is a huge potential for different application-specific services that could be built on top of a basic content-distribution infrastructure. Examples include mailing systems for multimedia email, radio broadcasting, content adaptation, dynamic web content, eCommerce, or distributed search services. Quick and easy deployment of these new services will be essential to their success. We are therefore working on an open platform and a set of building blocks that allow the creation of custom-tailored content storage and delivery networks. The platform will be built on our existing cache system, taking advantage of its superior performance through the networking of caches and active content management. A DNS-based module will provide network-independent traffic redirection, which is required for provisioning of content-provider oriented services. An open and standardized interface will enable third party developers to quickly implement advanced services beyond web

caching. We are currently implementing enhanced mailing services for rich content mails and a system for radio broadcasting. Our goal is the establishment of an open and flexible platform for all kinds of content-delivery services and the creation of novel application-specific services on top of it.

6 Literature

[1] A. Luotonen, H.F. Nielsen, T. Berners-Lee: Cern httpd; Homepage at http://www.w3.org/Daemon/, 1999.

[2] A. Chankhunthod, P.B. Danzig, C. Neerdaels, M.F. Sewarts, K.J. Worrell: A Hierarchical Internet Object Cache; Usenix Technical Conference, 1996.

[3] D. Wessels: Squid Web Proxy Cache; Homepage at http://www.squid-cache.org/, 2000.

[4] IPWorX Web Performance Solutions: Homepage at http://www.lucentipworx.com/, 2000.

[5] M. Hofmann, T.S. Ng, K. Guo, S. Paul, H. Zhang: Caching Techniques for Streaming Multimedia over the Internet; Bell Labs Technical Memorandum, April 1999.

[6] S. Sen, J. Rexford, D. Towsley: Proxy prefix caching for multimedia streams; IEEE Infocom, New York, USA, March 1999.

[7] R. Rejaie, H, Yu, M. Handley, D. Estrin: Multimedia Proxy Caching Mechanism for Quality Adaptive Streaming Applications in the Internet; IEEE Infocom, Tel Aviv, Israel, March 2000.

[8] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson: RTP: A Transport Protocol for Real-Time Applications; Internet Request for comments 1889, January 1996.

[9] H. Schulzrinne, A. Rao, R. Lanphier: Real Time Streaming Protocol (RTSP); Internet Request for comments 2326, April 1998.

[10] E. Bommaiha, K. Guo, M. Hofmann, S. Paul: Design and Implementation of a Caching System for Streaming Media over the Internet; IEEE Real-Time Technology and Applications Symposium (RTAS), Washington, VA, USA, May 2000.

[11] ARD Tagesschau (German News Station): Homepage at http://www.tagesschau.de/, 2000.

[12] Real Networks: Homepage at http://www.real.com/, 2000.

[13] M. Cieslak, D. Forster: Web Cache Coordination Protocol V1.0, Work in Progress, Internet Draft draft-ietf-wrec-web-pro-00.txt, June 1999.

[14] P. Albitz, C. Liu, M. Loukides: Dns and Bind; 3rd Edition, O'Reilly & Associates, September 1998.