Large-Scale Cooperative Caching and Application-Level Multicast in Multimedia Content Delivery Networks

Jian Ni, Yale University and Danny H. K. Tsang, HKUST

ABSTRACT

The Internet and multimedia technologies have greatly changed our lives. With the increasing popularity of multimedia entertainment applications over the Internet, innovative infrastructures and technologies are needed to efficiently distribute the surging amount of multimedia contents. Content delivery networks provide an intermediate layer of infrastructure that helps to deliver the contents from content providers to a large community of geographically distributed users. Cooperative caching and application-level multicast are two technologies that can be implemented in a multimedia content delivery network for delivering on-demand and live multimedia contents respectively. This article introduces the ideas and approaches of implementing cooperative caching and application-level multicast under a hierarchical architecture to achieve large-scale multimedia content delivery.

INTRODUCTION AND THE CDN ARCHITECTURE

Thanks to the development of multimedia technologies and high-speed networks, traditional Web sites are changing into multimedia Web sites, and new multimedia networking applications seem to be emerging everyday. Multimedia entertainment applications like video on demand, online songs and movies, IP telephony, Internet radio and television, and interactive games have now become popular networking applications over the Internet. With the increasing popularity of such applications and the high quality of service (QoS) expected by users, *content delivery networks* (or *content distribution networks*, CDNs) have recently been widely deployed to deliver the contents from content providers to a large community of geographically distributed users.

CDNs may be deployed by a CDN service provider such as Akamai (http://www.akamai.com) that partners with multiple Internet service providers (ISPs). Alternatively, a big ISP like AT&T itself may provide CDN service and deploy CDN servers at the edge of its network. CDNs were originally proposed to achieve scalable Web content delivery over the Internet, and have recently been proposed for multimedia content delivery as well (e.g., SinoCDN, http://www.sinocdn.com). A CDN can achieve scalable content delivery by distributing load among its servers, serving user requests via servers that are close to the users, and bypassing congested network paths.

Basically, a CDN is an *overlay network* constructed from a group of strategically placed and geographically distributed CDN servers. When a user requests some content, say, a movie, the request is redirected to a nearby CDN server (we call it the user's local CDN server) through a certain *user request redirection* mechanism so that content delivery takes place at the edge of the network where bandwidth is abundant. The local CDN server is the entry point for the user to access the CDN. It conducts *admission control* to either accept or block the request. If the request is accepted, the local CDN server serves the user if it has the content; otherwise, it performs *content routing* to locate and then deliver the requested content to the user. Therefore, the main components of the CDN architecture are listed in the subsections below.

CDN SERVER PLACEMENT AND OVERLAY NETWORK FORMATION

In order to reach as many users as possible and to serve those users with satisfactory QoS, the CDN servers of a CDN should be placed strategically on the Internet. These CDN servers are interconnected by the existing network infrastructure (e.g., the public Internet or transmission lines with dedicated bandwidth). They form a logical overlay network and function as overlay routers to deliver the contents from the origin servers to the users.

For a large CDN that consists of hundreds or even thousands of geographically distributed CDN servers, like the hierarchical topology of the IP routers on the Internet, we believe that a hierarchical overlay network topology is required for the CDN servers to perform content routing

0163-6804/05/$20.00 © 2005 IEEE

and content delivery efficiently. In order to form a hierarchical overlay network, the CDN servers are first grouped into clusters, either by manual configuration, or through some self-organizing scheme such as the binning scheme proposed in [1]. Then the CDN servers of the same cluster form a sub-overlay network. Each cluster selects one or more CDN server as the representative of this cluster. Representatives of different clusters form a higher-level overlay network. Thus a hierarchical topology is constructed.

User Request Redirection

There are several ways to redirect the request from a user to its local CDN server. User request redirection can be *nontransparent* (to the user) like explicit user configuration. More preferred it can be *transparent* that relies on some network devices such as layer 4 (L4)/L7 switches, routers, DNS servers, or via URL rewriting by the origin server. DNS-based user request redirection schemes have become popular because of their simplicity and generality [2].

Admission Control

As the entry point, the local CDN server conducts admission control to either accept or block a user's request. The simplest admission control policy, known as complete sharing (CS), is to accept a user's request whenever there are sufficient resources, and to block it otherwise. The CS policy is easy to implement, and performs well if the user traffic demand is light. However, in today's CDNs traffic demands are unpredictable and most often the CDN operates close to its full-capacity regime. The CS policy may lead to poor resource utilization. In addition, *service differentiation* is now becoming a more and more desirable feature in many CDNs. For example, some users may want to pay more to get better QoS. If each accepted user contributes a class-based revenue, the CS policy may lead to a poor revenue rate for the CDN. Therefore, it is desirable to implement admission control policies that may block a user's request even if there are sufficient resources in order to optimize certain performance metrics (blocking probabilities, throughput, revenue rate, etc.) of the CDN [3].

Content Routing and Delivery

If the local CDN server accepts a user's request but does not have the requested content, it will perform content routing to locate and then deliver the content to the user. We classify multimedia contents into two types: *on-demand* multimedia contents, including stored audio and video, and *live* multimedia contents, including Internet radio and television.

On-demand multimedia contents are normally requested by the users asynchronously. After locating a content, the local CDN server should preferably cache the content in order to serve future user requests. It is not feasible for a single CDN server to cache all the contents existing in the Internet because of its limited disk space; therefore, the *cooperative caching scheme* among the CDN servers and content routing are closely related.

Live multimedia contents cannot be cached in advance. But since live contents are synchronous, a CDN server can aggregate the user requests for the same live content that are redirected to it, and become a multicast group member for that content. The multicast group members for the same content form an overlay multicast tree (or mesh) that delivers the content from the origin server to all multicast group members. If a CDN server that has not yet been a multicast group member for certain content receives a user's request for that content, it performs multicast content routing to add itself to the multicast tree. This is known as *application-level multicast* (ALM) or *overlay multicast*.

Cooperative caching and ALM are two important technologies to efficiently distribute the ever increasing multimedia contents. In the first part of this article we discuss how to implement large-scale cooperative caching in a CDN for delivering on-demand multimedia contents. In particular, we propose a hierarchical architecture in which the CDN servers cooperate through intracluster and intercluster cooperative caching schemes. We also compare the performance of different intracluster cooperative caching schemes. In the second part of this article we discuss how to implement large-scale ALM in a CDN for delivering live multimedia contents. We introduce different approaches to constructing overlay multicast trees by employing different routing metrics and routing strategies. We also demonstrate that a hierarchical ALM approach scales much better than a flat approach to achieve large-scale live streaming.

For a lage CDN that consists of hundreds or even thousands of geographically distributed CDN servers, like the hierarchical topology of the IP routers on the Internet, we believe that a hierarchical overlay network topology is required for the CDN servers to perform content routing and content delivery efficiently.

Large-Scale Cooperative Caching for Delivering On-Demand Multimedia Content

Suppose a two-level hierarchical overlay network is formed as shown in Fig. 1, either by manual configuration or through self-organization. The CDN servers in the same cluster are close to each other and we assume that they are logically fully connected. Clusters are interconnected by the representative CDN servers of different clusters. The steps the CDN takes to serve a user's request are:

1 Try to satisfy the user's request using the local CDN server.
2 If step 1 fails, try to satisfy the user's request using a CDN server inside the cluster including the local CDN server.
3 If step 2 fails, try to satisfy the user's request using a CDN server inside a nearby cluster.
4 If step 3 fails, try to satisfy the user's request using the origin server.

If the local CDN server has the requested content and serves the user, we call it a *local hit*. If not, the local CDN server performs intracluster content routing. If the request is satisfied by a CDN server inside this cluster (including the local CDN server), we call it a *cluster hit*. If not, intercluster content routing is performed. If the request is satisfied by any CDN server of this CDN, we call it a *global hit*. In order to avoid retrieving the content from a remote CDN server and causing a long delay, steps 3 and 4 can be performed simultaneously so that the origin

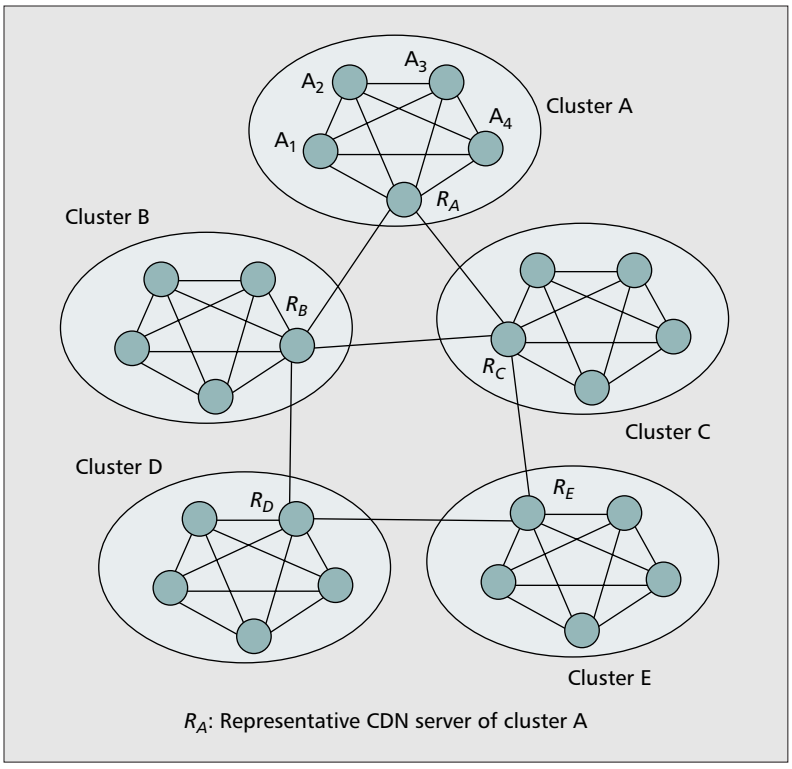

Figure 1. *A two-level hierarchical overlay network formed by the CDN servers of a CDN.*

server is contacted earlier.

The local hit rate H_{local} of a CDN is defined as the ratio between total local hits and total user requests arriving at the CDN. Similarly, the cluster hit rate $H_{cluster}$ and the global hit rate H_{global} of a CDN are defined accordingly. These are important metrics to evaluate the performance of a CDN. The main objectives of a CDN are to save network resources, reduce the origin server's load, and provide better QoS to users.

A qualitative analysis [4] reveals that higher H_{local} and $H_{cluster}$ indicate that more network bandwidth is saved. In addition, with higher H_{local}, $H_{cluster}$, and H_{global}, the origin server has less load. Because of the current best effort Internet service model, it is difficult to guarantee the QoS to users quantitatively if the content is delivered via the default Internet path. For multimedia content delivery that consumes a large amount of bandwidth, a CDN server close to a user (so that content delivery takes place at the edge of the network where bandwidth is abundant) provides better QoS to the user than a remote server. Higher H_{local} and $H_{cluster}$ imply that more user requests are satisfied by nearby CDN servers, and thus better QoS is perceived by users.

There are two approaches used to distribute on-demand multimedia contents to CDN servers. One is the *proactive* approach as in replicated server systems. Contents are replicated in the CDN servers in advance. Here content routing is easy because content locations are known beforehand. The other is the *reactive* approach as in cooperative caching systems. Contents are cached in different CDN servers based on the requests of users. Here content routing is closely related to the cooperative caching scheme among the CDN servers. The caching approach is more adaptive and often has higher hit rates, at the cost of employing more complicated content routing. Sometimes both approaches need to be combined in order to achieve high performance.

Intracluster Cooperative Caching and Content Routing

The most straightforward scheme is the *query*-based scheme [5], in which a CDN server broadcasts a query for the requested content to other CDN servers inside the same cluster if it does not have the content. The implementation overhead of this scheme includes significant query traffic, and sometimes long delay because a CDN server needs to wait for the last "miss" reply before concluding that none of the cooperating CDN servers has the content.

In order to avoid flooding queries, the *digest*-based scheme was proposed [6]. Each CDN server maintains a content digest that includes the content information of other CDN servers inside the same cluster. Once a CDN server has cached/deleted some contents, it notifies other CDN servers to update their content digests. Hence, a CDN server knows where to locate the content by checking its content digest. The major overhead is update traffic because update messages need to be exchanged frequently to ensure that all cooperating CDN servers have the correct information about each other.

A centralized version of the digest-based scheme is the *directory*-based scheme [7], in which a directory server maintains the content information of the CDN servers inside the cluster. A CDN server only needs to notify the directory server when local updates occur, and queries the directory server when there is a local miss. Compared to the digest-based scheme the update traffic is greatly reduced, but the directory server is a single point of failure because it needs to handle the update and query messages from all the cooperating CDN servers.

A more efficient scheme is the *hashing*-based scheme [8, 9]. The CDN servers inside a cluster maintain the same hashing function. Each content is assigned to a designated CDN server based on the content's URL (or other unique identification), unique IDs (e.g., IP addresses) of the CDN servers, and the hashing function. All requests for the same content are redirected to the designated CDN server for that content. Compared to other schemes, the hashing-based scheme has the smallest implementation overhead. In addition, under pure hashing (in which each content is cached only in its designated CDN server) the contents are distributed among the CDN servers without any duplication, so it has the highest content sharing efficiency. However, pure hashing does not distinguish a local CDN server from other CDN servers. Local requests are often redirected to and served by other designated CDN servers, so the local hit rate of the CDN is quite low. For multimedia content delivery that consumes a large amount of bandwidth, a high local hit rate is crucial to save network bandwidth and provide satisfactory QoS to users.

We were therefore motivated to propose the *semi-hashing-based* scheme [4]. Under the semi-hashing-based scheme, a local CDN server allocates a certain portion, P_{local}, of its disk space to cache the most popular contents for its local users, and the remaining portion to cooperate with other CDN servers via a hashing function. Like pure hashing, semi-hashing has small implementation overhead and high content sharing efficiency. In addition, it significantly increases the local hit rate of the CDN.

We built a discrete-event simulation model that consists of six CDN servers in a cluster to evaluate the performance of different intracluster cooperative caching schemes [4]. Without considering the implementation overhead, the performance of the query, digest, and directory-based schemes are the same, because by either querying peer CDN servers, checking the content digest, or querying the directory server, a local CDN server knows where to locate the content.

The local hit rate H_{local} and cluster hit rate $H_{cluster}$ under different schemes are shown in Fig. 2. As expected, the pure hashing scheme has the highest content sharing efficiency and thus the highest $H_{cluster}$, but the lowest H_{local}. Under the query, digest, and directory-based schemes duplicate contents are cached in different CDN servers, so their $H_{cluster}$ is low. $H_{cluster}$ of the semi-hashing scheme (with $P_{local} = 0.2$) is slightly less than that of the pure hashing scheme, but its H_{local} improves a lot as a result of caching popular contents for local users in the local CDN server. We also found that under hashing-based schemes the load was more evenly distributed among the CDN servers, and the blocking probability experienced by users was lower than with the other schemes.

We conducted more experiments to study the effect of the adjustable parameter P_{local} (the portion of a CDN server's disk space allocated for local popular contents) on the performance of the semi-hashing scheme under different user request patterns. The results are shown in Fig. 3. When $P_{local} = 0$, it is in fact the pure hashing scheme. More disk space for hashing cooperation (i.e., smaller P_{local}) results in higher content sharing efficiency and hence higher $H_{cluster}$. More disk space for local popular contents (i.e., larger P_{local}) results in higher H_{local}. When $P_{local} = 1$, a CDN server reserves all its disk space for local users and there is no cooperation among the CDN servers, so $H_{local} = H_{cluster}$. By adjusting P_{local}, we can make a trade-off between full hashing cooperation (high $H_{cluster}$) and standalone proxies (high H_{local}).

An important observation from Fig. 3 is that $H_{cluster}$ decreases almost linearly with P_{local}, but a small P_{local} like 10 or 20 percent allocated for local popular contents results in a significant increase of H_{local}. This is due to the Zipfian popularity assumption we made about the contents: a small number of the most popular contents account for a large portion of user requests. While the Zipfian popularity assumption may not be true for multimedia contents, recent research [10] shows that the requests for videos on the Web are even more biased toward popular titles than a Zipfian distribution: the top 10 percent ranked titles account for 50 percent of all the requests. This implies that the semi-hashing scheme will perform even better in real situations: a CDN server only needs to allocate a small portion of its disk space for local popular contents to ensure both satisfactorily high H_{local} and $H_{cluster}$.

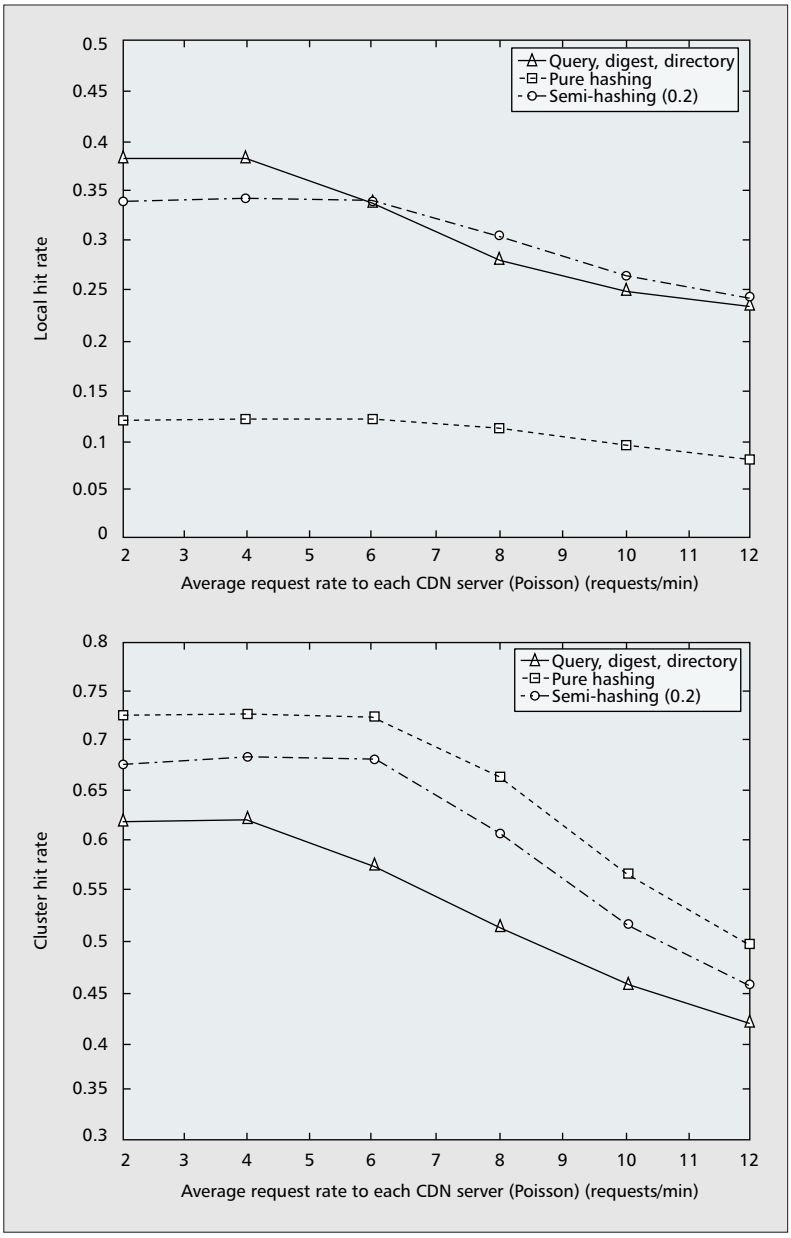

Figure 2. *Comparison of different intracluster cooperative caching schemes.*

Intercluster Cooperative Caching and Content Routing

If intracluster content routing fails, intercluster content routing is performed based on the intercluster cooperative caching scheme. For intercluster cooperative caching a hashing-based scheme is not appropriate because representative CDN servers of different clusters are normally geographically distributed. The digest-based scheme is also not suitable because it is very difficult for a representative CDN serv-

Multicast over the Internet was originally proposed at the network layer, referred to as IP multicast. However, after a decade of research, there are still many hurdles in the deployment of IP multicast, such as the lack of higher layer functionalities and scalable inter-domain multicast routing protocols.

er to maintain a huge and correct content digest including the content information of CDN servers of other clusters.

We introduce a query-based scheme for inter-cluster cooperative caching using Fig. 1. The representative CDN server of cluster A, R_A, queries its neighbors R_B and R_C for the missing content. R_B replies with a hit message if it has the content; if not, it forwards the query message to its own neighbors R_C (R_C will ignore this duplicate query) and R_D. At the same time, based on the intracluster cooperative caching scheme used in cluster B (say, a hashing-based scheme), R_B queries the designated CDN server for that content in cluster B. Here R_B only queries one CDN server in its cluster, but the chance of getting the content is high because actually all the CDN servers inside cluster B are serving this request via the hashing-based scheme.

A *TIMEOUT* value and time to live (*TTL*) number are set for each query message. If timeout occurs, the origin server is contacted in order to avoid retrieving the content from a remote CDN server. The *TTL* number is decreased by 1 every time the query message is forwarded to another CDN server. When the *TTL* number decreases to 0, the query message is not forwarded in order to avoid flooding the request in the CDN.

Application-Level Multicast for Delivering Live Multimedia Contents

Delivery of live multimedia contents, known as *live streaming*, is similar to traditional radio and television broadcast, except that transmission takes place over the Internet. This class of applications often has many users who are receiving the same live content synchronously, which can be efficiently accomplished via *multicast* techniques. Multicast over the Internet was originally proposed at the network layer, referred to as *IP multicast*. However, after a decade of research, there are still many hurdles in the deployment of IP multicast, such as the lack of higher-layer functionalities (e.g., reliable transport; flow, congestion, and admission control; multicast security) and scalable interdomain multicast routing protocols. Therefore, ALM, or overlay multicast, in which multicast functionality is pushed up to the application layer, has recently been proposed for a number of Internet multicast applications [11–13].

There are two different ALM architectures: *peer-to-peer* (P2P)-*based* and *proxy-based*. In the P2P-based architecture, the users are considered to be equivalent peers and are organized into an overlay network for multicast content delivery. This architecture is suitable for applications like video conferences and interactive games. In the proxy-based architecture, a group of proxy servers (or multicast service nodes) are organized into an overlay network and provide content delivery service to users. This architecture is suitable for applications like live streaming. In this article we concentrate on the proxy-based architecture for implementing ALM in CDNs.

We first give a brief comparison between IP multicast and ALM as deployed in CDNs. A *multicast group* for a multicast content is a group established for the distribution of that content.

Multicast Group Identifier — In IP multicast, a class D address is used to identify a multicast group. In ALM, a URL or other application-related key can be used to identify a multicast group.

Multicast Group Members — In IP multicast, in order to receive certain multicast content, a user explicitly registers to its directly attached router via the Internet Group Management Protocol (IGMP). The IP routers of the same multicast group form a mutlicast tree and take responsibility to deliver the multicast content to their registered users. In ALM, the request from a user is redirected to the user's local CDN server. The CDN servers of the same multicast group form an overlay multicast tree (or mesh) and take responsibility to deliver the multicast content to their users.

Network Topology — In IP multicast, the topology of the IP routers exactly reflects the physical network topology. While in ALM, the CDN servers form a logical overlay network on top of the underlying network infrastructure. This overlay network may or may not accomplish a good match with the real physical network.

Multicast Routing — Both IP multicast and ALM require a certain multicast routing protocol to construct a multicast tree for delivering the multicast content. IP multicast routing normally relies on the underlying unicast routing protocols that are quasi-static and employ simple routing metrics like *number of hops* or *delay*. On the contrary, as discussed in the next subsection, the implementation of ALM routing is much more flexible.

Application-Level Multicast Routing

The CDN servers of the same multicast group form an overlay multicast tree which delivers the multicast content from the origin server to all multicast group members. The key issue is how to construct such overlay multicast trees.

When constructing overlay multicast trees for live streaming, first we can choose different *routing metrics*. One commonly employed routing metric is *delay*. Note that ALM for live streaming applications is *unidirectional* transmission: a CDN server who is transmitting a live content to another CDN server will not receive the same content back from that CDN server. Hence, for a multicast group, a single-source spanning tree is formed where the source is, for example, the origin server. In this case we want to construct a *minimum-delay-path spanning tree* that minimizes the delay from the source to each of the nodes on the tree. This is different from P2P-based ALM for applications like videoconferences and interactive games, in which the transmission is *bidirectional*: a pair of overlay nodes of the same multicast group may exchange contents in both directions. For these applications we want to construct a *minimum-delay spanning tree* that minimizes the sum of the tree link delays.

For live streaming applications that consume large amounts of bandwidth, the *available bandwidth* of the network path between the sender and the receiver is crucial for maintaining streaming quality. In addition, end-to-end delay normally consists of propagation delay, transmission delay, and queuing delay. Once the end-to-end bandwidth is guaranteed, transmission delay and queuing delay can be bounded; only the propagation delay plays an important role. For streaming applications, a user can tolerate some startup (propagation) delay to receive the content, but will be annoyed if streaming is interrupted because of insufficient available bandwidth. Therefore, available bandwidth may be a more suitable routing metric. In this case we want to construct a *widest-path spanning tree* that maximizes the minimum available bandwidth of the tree links. Here accurately measuring the available bandwidth between the CDN servers without causing intrusive traffic is a challenging task.

References [12, 13] choose the *access* (*interface*) *bandwidth* of the CDN servers as the routing metric. This is based on the assumption that links in the core networks are overprovisioned and hence not bottlenecks, while the access bandwidth of a CDN server limits the number of simultaneous multicast sessions it can support. In order to avoid overusing a CDN server's access bandwidth, its outgoing degree on the multicast tree should be bounded. In addition to access bandwidth, [12, 13] also consider delay. For example, [12] considers the minimum-diameter degree-limited spanning tree problem (MDDL), in which the *diameter* of a tree is defined as the delay of the longest tree path. Reference [13] considers the *minimum-average-latency degree-bounded directed spanning tree problem*, in which the *average latency* of a tree is defined as the weighted summation of the latencies of all nodes on the tree. The weight of a node is determined by its user population. In general, when multiple routing metrics are considered simultaneously, the idea is similar to that of QoS routing, but implemented at the application layer.

Second, we can employ different *routing strategies*.

Static vs. Dynamic — In the *static* or *quasi-static* approach, a static multicast tree that spans all the CDN servers of a CDN is built beforehand based on some long-term estimation of the network conditions. Every time a new multicast group is formed, a subtree of the static multicast tree that spans all the multicast group members is used to deliver the content. In the *dynamic* approach, every time a new multicast group is formed, a multicast tree is constructed on demand among the multicast group members based on current network conditions. Compared to the static approach, the dynamic approach is more adaptive and can achieve better resource utilization, but requires heavier communication and control overhead.

Centralized vs. Distributed — In the *centralized* approach, a central manager/server constructs the multicast tree for every multicast group. It collects the link information (available bandwidth, delay, etc.) and server information (CPU power, residual access bandwidth, etc.) of the entire CDN, and makes decisions based on this global information. In the *distributed* approach, each CDN server collects the local information of its directly attached links and neighboring CDN servers. The multicast tree is constructed distributively among the multicast group members. The centralized approach can achieve global optimality and works well for small CDNs, while the distributed approach is more robust and scales much better for large CDNs.

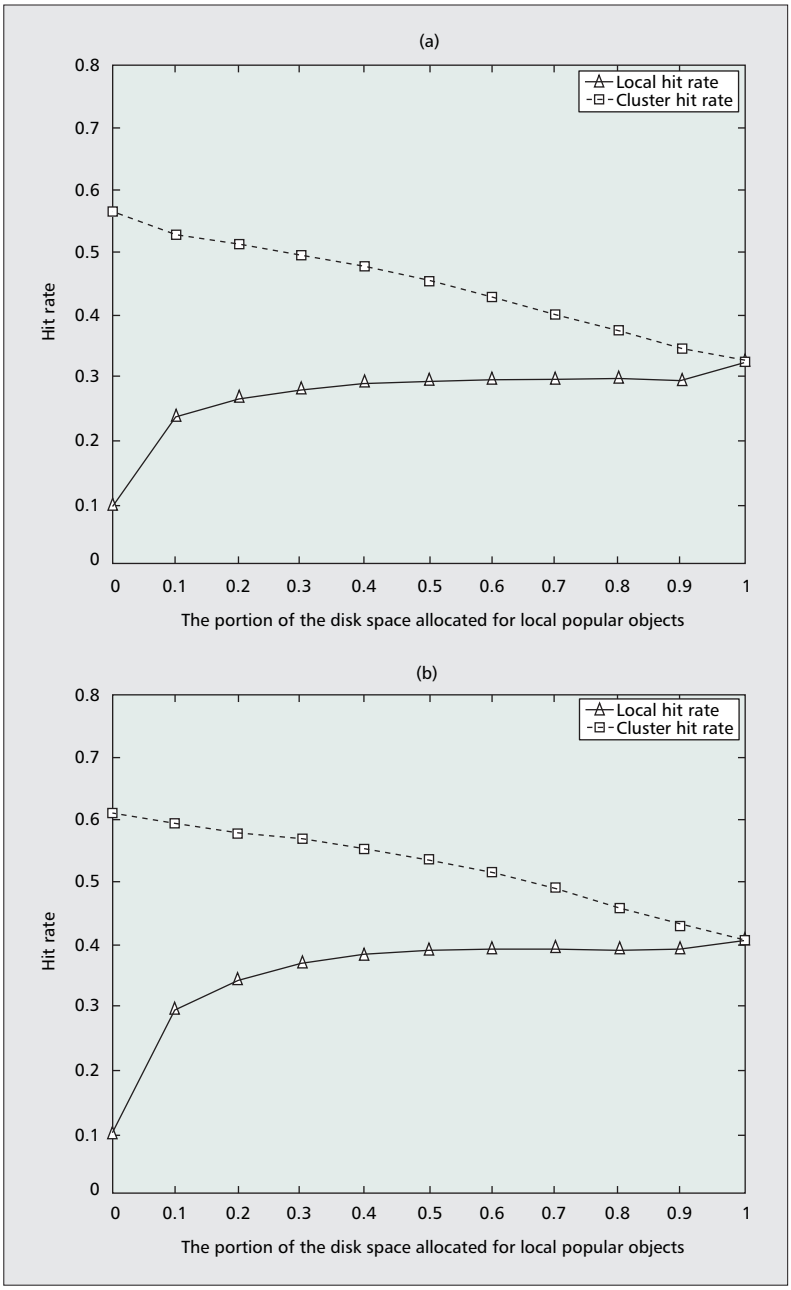

Figure 3. *The effect of* P_{local} *on the performance of the semi-hashing-based cooperative caching scheme: a) fixed average request rate (Poisson, 10 requests/min); b) variable average request rate (nonstationary Poisson, 2–10–50 requests/min).*

Reconstructive vs. Cumulative — During the session time of live streaming, a CDN server may join or leave the multicast group dynamical-

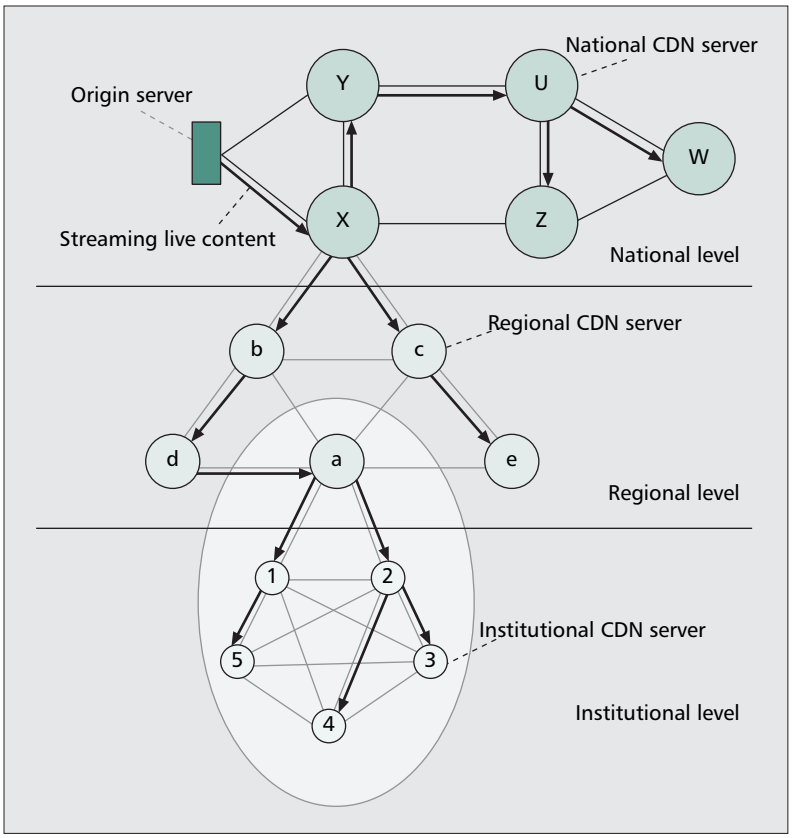

Figure 4. *Hierarchical application-level multicast in a CDN.*

ly, depending on whether there are some users receiving the live content from that CDN server. In the *reconstructive* approach, every time a CDN server joins or leaves the multicast group, a new multicast tree is reconstructed. Note that this may change some of the ongoing transmissions between the CDN servers of the multicast group. In the *cumulative* approach, when a CDN server joins the multicast group, a unicast path that connects a node on the multicast tree to this CDN server is added to the multicast tree. When a CDN server leaves the multicast group, if it has no child node on the multicast tree, it just detaches itself from the tree; otherwise, it remains on the multicast tree until all its child nodes leave the multicast group. Note that in the cumulative approach no ongoing transmission between a pair of CDN servers of the multicast group will be interrupted. The reconstructive approach can maintain optimality when changes occur, but it requires smooth switchover techniques to change ongoing transmissions.

Hierarchical Application-Level Multicast

It is difficult to construct an optimal or even efficient overlay multicast tree for a multicast group consisting of hundreds or even thousands of CDN servers. This may happen, for example, when Internet broadcasting the World Cup or Olympic Games to users all around the world. Like the hierarchical architecture for large-scale cooperative caching in a CDN, we believe that a hierarchical ALM approach is needed for such large-scale live streaming as well. Figure 4 gives an example of a three-level overlay multicast tree formed by the CDN servers of a CDN. At the national level, an *international* multicast tree is built to deliver the live content from the origin server to all national CDN servers (X, Y, Z, etc.) interested in the live content. At the regional level, all the regional CDN servers (a, b, c, etc.) in the same nation and of the same multicast group form an *interregional* multicast tree in which the national CDN server of that nation is the source. It is similar at the institutional level.

We give an example to illustrate that a hierarchical ALM approach scales much better than the flat approach (in which all CDN servers are at the same level). Suppose there are $N = 1000$ CDN servers participating in large-scale live streaming. Assume the computational complexity of constructing an N-node multicast tree that optimizes certain routing metrics is $O(N^3)$ (e.g., the computational complexity of constructing a minimum-diameter spanning tree on a dense graph is $O(N^3)$). In the flat approach with the reconstructive routing strategy, every time a CDN server joins or leaves the multicast group, a new optimal multicast tree is reconstructed, with a computational complexity of $O(10^9)$. In a three-level hierarchical approach as shown in Fig. 4, a cluster at each level has

$$O(\sqrt[3]{1000}) = O(10)$$

CDN servers. With the reconstructive routing strategy, when a CDN server in region a of nation X joins or leaves the multicast group, under the worst case, a new optimal interinstitutional multicast tree in region a, a new optimal interregional multicast tree in nation X, and a new optimal international multicast tree are reconstructed. This only requires a computational complexity of $O(3 \times 10^3)$. Note that the hierarchical approach may not achieve global optimality, but scales much better.

A hierarchical ALM approach is also flexible to implement. At each level of the hierarchical overlay network, we can employ the most appropriate routing metrics and strategy to construct the overlay multicast trees based on the properties of that level. For example, at the lowest level, for CDN servers in the same cluster that are close to each other, we can employ the centralized and reconstructive routing strategy to achieve optimality within that cluster. At the higher level, for CDN servers that are geographically distributed, we can employ the distributed and cumulative routing strategy in which each CDN server only shares information with its neighbors, and overlay multicast routing is implemented in an ad hoc distributive way.

Conclusions

Cooperative caching and application-level multicast are two technologies that can be implemented in a multimedia content delivery network for delivering on-demand and live multimedia contents, respectively. This article introduces the ideas and approaches of implementing large-scale cooperative caching and application-level multicast under a hierarchical architecture. CDN servers in the same cluster that are close to each other can achieve full cooperation due to the

small size of a cluster. CDN servers in different clusters cooperate via higher-level cooperation. Such higher-level cooperation can be limited to achieve scalability,which will not reduce cooperation efficiency much because it is much less beneficial for a CDN server to cooperate with a remote CDN server.

ACKNOWLEDGMENT

The authors would like to thank Xiaojun Hei for his help on the work presented in this article.

REFERENCES

[1] S. Ratnasamy *et al.*, "Topologically Aware Overlay Construction and Server Selection," *Proc. IEEE INFOCOM 2002*, June 2002.
[2] A. Shaikh, R. Tewari, and M. Agrawal, "On the Effectiveness of DNS-Based Server Selection," *Proc. IEEE INFOCOM 2001*, Apr. 2001.
[3] J. Ni *et al.*, "Threshold and Reservation Based Call Admission Control Policies for Multiservice Resource-Sharing Systems," *Proc. IEEE INFOCOM 2005*, Mar. 2005.
[4] J. Ni *et al.*, "Hierarchical Content Routing in Large-Scale Multimedia Content Delivery Network," *Proc. IEEE ICC 2003*, May 2003.
[5] D. Wessels and K. Claffy, "Internet Cache Protocol (ICP), Version 2," IETF RFC 2186, Sept. 1997.
[6] A. Rousskov and D. Wessels, "Cache Digests," *Comp. Net. and ISDN Sys.*, vol. 30, no. 22–3, Nov. 1998, pp. 2155–68.
[7] 5. Gadde, M. Rabinovich, and J. Chase, "Reduce, Reuse, Recycle: An Approach to Building Large Internet Caches," *Proc. Wksp. Hot Topics in Op. Sys.*, Apr. 1997, pp. 93–98.
[8] V. Valloppillil and K. W. Ross, "Cache Array Routing Protocol vl.0," Internet draft, Feb. 1998.
[9] D. Karger *et al.*, "Web Caching with Consistent Hashing," *Proc. 8th Int'l. WWW Conf.*, May 1999.
[10] 5. Acharya, B. Smith, and P. Parnes, "Characterizing User Access to Videos on the World Wide Web," *Proc. SPIE/ACM MMCN 2000*, San Jose, CA, Jan. 2000.
[11] D. Pendarakis *et al.*, "ALMI: An Application Level Multicast Infrastructure," *Proc. 3rd Usenix Symp. Internet Tech. and Sys.*, Mar. 2001.
[12] 5. Shi and J. S. Turner, "Routing in Overlay Multicast Networks," *Proc. IEEE INFOCOM 2002*, June 2002.
[13] 5. Banerjee *et al.*, "Construction of an Efficient Overlay Multicast Infrastructure for Real-Time Applications," *Proc. IEEE INFOCOM 2003*, Apr. 2003.

BIOGRAPHIES

JIAN NI (jian.ni@yale.edu) received his B.Eng. degree in automation from Tsinghua University, Beijing, China, in 2001, and his M.Phil. degree in electrical and electronic engineering from the Hong Kong University of Science and Technology (HKUST) in 2003. He is currently working toward his Ph.D. degree in electrical engineering at Yale University, New Haven, Connecticut. His research interests include analysis, design, control, and optimization of communications and networking systems.

DANNY TSANG [SM'00] (eetsang@yale.edu) received a Ph.D. degree in electrical engineering from the University of Pennsylvania, Philadelphia, in 1989. He is currently an associate professor in the Electrical and Electronics Engineering Department at HKUST. His current research interests include QoS issues in wireless LANs, P2P networks, and optical networks. He is currently an Associate Editor for *OSA Journal of Optical Networking* and an Associate Technical Editor for *IEEE Communications Magazine*. During his leave from HKUST in 2000–2001, he assumed the role of principal architect at Sycamore Networks and was responsible for the network architecture design of Ethernet MAN/WAN over SONET/DWDM networks. He invented 64B/65B encoding and also contributed to the 64B/65B encoding proposal for Transparent GFP in the ITU T1X1.5 standard.

Cooperative caching and application-level multicast are two technologies that can be implemented in a multimedia content delivery network for delivering on-demand and live multimedia contents, respectively.