Open access • Proceedings Article • DOI:10.1109/CONTEL.2003.176925

Towards scalable and affordable content distribution services — Source link

Thomas Plagemann, Vera Goebel, Laurent Mathy, Nicholas Race ...+3 more authors

Institutions: University of Liège

Published on: 11 Jun 2003 - International Conference on Telecommunications

Topics: Replication (computing)

Submitted to the 7th International Conference on Telecommunications (ConTEL), Zagreb, Croatia, 2003

TOWARDS SCALABLE AND AFFORDABLE CONTENT DISTRIBUTION SERVICES

Thomas Plagemann[1], Vera Goebel[1], Carsten Griwodz[1],
Pål Halvorsen[1], Laurent Mathy[2], Nicholas Race[2], Michael Zink[3]

[1]University of Oslo, Department of Informatics, {plageman, goebel, griff, paalh}@ifi.uio.no
[2]Lancaster University, Department of Computer Science, {laurent, race}@comp.lancs.ac.uk
[3]Technical University of Darmstadt, KOM, Michael.Zink@KOM.tu-darmstadt.de

ABSTRACT

Content Distribution Networks (CDNs) are based on a static infrastructure, and caching and replication in CDNs are performed in a static manner. We argue that it is necessary to design future CDNs in such a way that they are scalable to reach enough customers, their costs are kept low, and they provide a Quality-of-Service that satisfies the client requirements. To reach these goals, more flexibility is needed in CDNs. Flexibility will enable CDNs to perform faster and better decisions for caching and replication, and it will reduce the amount of manual intervention that is necessary to manage and maintain the CDN. In this paper, we discuss the application of Peer-to-Peer mechanisms in CDNs that support advanced caching and replication strategies and their combination with a dynamic QoS management hierarchy to reach the needed flexibility in CDNs.

1 Introduction

The support of **on-demand services** for entertainment applications, like Video-on-Demand (VoD), over the Internet has been an important research topic for a long time. The first proprietary solutions for content distribution over the Internet for this type of applications are deployed. These solutions are built as a set of servers and caches that are logically connected as an overlay network and are called **Content Distribution Networks (CDNs)**. CDNs operate on the assumption that a better service is achieved when the required multimedia data is stored close to the clients. It is important to note, that all services that are deployed provide a rather low quality to the consumer compared to the quality a TV set in combination with a DVD player can provide. The fact that today's CDNs have partial commercial success must not automatically lead to the conclusion that this technology can also successfully be used for future content distribution services. The reasons for this are as follows: (1) real costs are not reflected because broadband services and content distribution services are still promoted with "special" rates, (2) the costs of storing and managing the content in various competing proprietary encoding formats and different quality levels will dramatically increase with the amount of content, and (3) scaling up CDNs to penetrate all networked consumers will dramatically increase the costs for management and control.

In addition to the prerequisites that will change the competitiveness of content distribution services we must realize that future internet services will (1) use much richer multimedia formats that combine and synchronize (several) video and audio streams with text, graphics, animations, etc. and enable the end-user to navigate interactively in documents, (2) require a much higher level of interaction for these applications, (3) provide a variety of quality levels, e.g., for video this might range from DVD-like quality to low resolution versions for mobile terminals, and (4) Digital Rights Management and Copy Protection need to be supported. To establish and maintain a profitable market for the distribution of multimedia content over the Internet, it is necessary to obey the above listed demands and address the following issues:

- content providers can reach a sufficiently large number of customers, i.e., CDNs must be scalable,
- the costs of CDNs must be low enough such that the services can be provided to the customers at competitive rates (compared to for example traditional broadcasting, pay-TV and video rentals),
- the Quality of Service (QoS) must satisfy customer expectations in terms of response time, availability, etc., but also match their display device capabilities.

An infrastructure for the distribution of interactive multimedia content can be used for traditional VoD, but its strength is that it is suitable for a much larger variety

of application domains, such as education (e.g., Learning-on-Demand), distribution of information in the public sector (e.g., laws and regulations), and distribution of research results to enable tight collaboration between geographically separated research centers and institutions, etc.

It is the goal of this paper to analyze and point out how these high-level requirements for future CDNs, i.e., scalability, low cost, and sufficient quality, can be realized. One important step in this direction are new advanced caching and replication strategies for CDNs. This issue is intensively studied by many research groups worldwide. We believe that these techniques can only provide the full benefit if they can be utilized in a dynamic CDN, in which new servers and proxy caches can be dynamically added and removed. In order to keep the costs of such a flexibility low, it is very important to have the proper tools for (semi-) automatic managemtent of future CDNs. We envision to reach these goals by using the following three ingrediences in a combined fashion: Peer-to-Peer (P2P) mechanisms within the CDN to manage resources and content, advanced caching and replication strategies in the CDN, and a dynamic QoS management hierarchy „on top" of the CDN.

The remainder of the paper is structured as follows: in Section 2, we argue why flexible selforganizing CDNs are needed. Afterwards, we show in Section 3 how advanced caching and replication strategies can function in flexible CDNs. The basic QoS management hierarchy is described in Section 4, and its utilization for flexible CDNs is outlined in Section 5. Sections 6 gives a short conclusion and outlook.

2 The need for flexible selforganizing CDNs

CDNs consist of a number of strategically located servers that deliver content on behalf of content providers. CDNs redirect client requests away from origin servers towards CDN servers, typically using techniques such as DNS redirection or URL rewriting. Whilst these mechanisms provide a way for CDN operators to load-balance between their own servers, the actual CDN infrastructure is often fairly rigid, requiring a significant degree of management (thus increasing the operational costs). Whilst a number of these CDN systems are currently in operation, there is little evidence to suggest that an optimal architecture or design has actually been found [12].

A new research challenge is to pursue a goal towards the architecture and design of autonomic (i.e. self-organizing and self-repairing) CDNs that require minimal management, yet exhibit very good scalability and performance properties. These autonomic features should ideally apply to both the interconnection of components making up the CDN and the content management within the CDN (i.e., what content to put where and for how long). Such features would facilitate the introduction of new content and hardware, and improve the operational characteristics of the CDNs. Autonomic CDNs would exhibit better resilience to failure, offer increased efficiency, whilst simplifying management and keeping operational costs to a minimum.

Furthermore, in a world where mobile access is becoming increasingly commonplace, and emerging technology such as 3G enables broadband capabilities everywhere, the design of specific solutions for the support of mobility within multimedia applications will become increasingly unsustainable. However, we believe that by supporting self-organization at the interconnectivity and content management levels, CDNs could be designed to support any user, mobile or not, without distinction nor special provision. This is because the characteristics of the CDN as perceived by a moving user, and vice versa, appear to be equivalent to those perceived by a fixed user in the presence of failure. Indeed, the loss of connectivity resulting from a server crash, for example, in a wired network produces the same effect to the user application as the apparent loss of connectivity caused by a mobile user moving to a new network domain and re-connecting to a new content cache that does not hold the required content: the flow of media to the user application is interrupted. Investigating the design and use of autonomic CDNs in both wired and wireless contexts is therefore important and timely.

We also claim that the use of ideas from P2P technologies provides an interesting starting point to provide self-organizing and resilient (a.k.a autonomic) overlays [22]. We envisage that such techniques can be extended and combined with other emerging technologies, such as active and programmable networking, to achieve the level of operational performance needed in production CDNs, while retaining the self-organizing, highly resilient and very low management features of existing P2P networks.

3 Advanced caching in flexible CDNs

Originally, caching and replication decisions in CDNs have either been performed manually or based on caching approaches that are also well-known from the web caching world. CDNs that have been designed for real-time streaming media distribution are based on the assumption that required bandwidth would be

sustainable between a CDN node and a client.

The intense discussion of the following two topics showed that this assumption cannot always be made. The first topic is the demand for TCP friendly [25] behavior of all networked application. This consideration requires that media delivery in CDNs conforms to what is considered appropriate behaviour in the Internet, both inside the CDN and between CDN nodes and end-systems. The second topic is the success of multimedia-capable PDAs and mobile phones. It makes it highly desirable that the same content is made available to end-systems with vastly different capabilities in receiving, processing and presenting content.

Due to these two considerations, caching and replication approaches have been developed that make use a scalable content to adapt to network bandwidth and to client demands. All of these techniques make use of subdividing content for more appropriate distribution. They may aim at one or both of the goals of this scalability. The existing work can be broadly classified into two categories of subdivision: *temporal subdivision* and *quality subdivision*. With multimedia content that provides more structure than video that is supposed to be consumed linearly from end to end, a new possibility arises, which is *structural subdivision*.

3.1 Temporal subdivision

Temporal subdivision requires no special encoding formats and relies on the distribution of content in temporal segments. The one example that is currently in commercial use is prefix caching [19]. It works by storing the first few minutes of videos on a proxy cache server close to clients, while the rest of the video is retrieved from a central server. The use of the terms prefix and suffix for the initial temporal segment of video and the remainder, respectively, has become common. A form of temporal subdivision that works with asymmetrically organized hierarchies of servers, variances in customer interest, and an arbitrary number of caches was introduced by periodic multicast with pre-storage [5].

Temporally segmented techniques cannot adapt to network problems beyond a predefined level. They rely on the delivery of data from a node that holds one segment to the client in just-in-time, because they are generally designed under the assumption that buffer space is valuable and not abundant at end-systems. Thus, optimized versions of all protocols are vulnerable to dynamic changes in network conditions. In general, temporal segmentation ideas have no support for different end-system capabilities because they are mainly designed for high quality CBR movie delivery. Some limitations in the bandwidth of access networks could be overcome, though Paris et al. demonstrated that the combination of broadcasting protocols with the pre-storage of all movies' prefixes at the receiver side would considerably reduce or even eliminate startup latency [18]. Other authors noticed that this technique could just as well be applied to prefix caching.

It is generally true that the results of the broadcast family of protocols that was spawned by Aggarwal et al.'s paper [1] and was continued for example by Paris' Pagoda [17] can be transformed into temporally segmented on-demand ideas of spawned by patching [10], all of which can be combined with caching as shown in gleaning [9] and mpatch [24].

3.2 Quality subdivision

Quality subdivision requires a special encoding format that allows the delivery of lower quality versions of the content that requires less storage space and network bandwidth for delivery than the full version. This kind of subdivision became known by receiver-driven layered multicast (RLM) [15], an approach for scalable live video broadcasting. This approach assumes a video that consists of several layers, where the lowest layer provides a low quality version of the video, and higher layers can be added to increase this quality. It has subsequently been exploited in conjunction with caching. TCP-friendly delivery of layered video from central servers to cache servers is the objective of [27]. Assigning different values to the layers of videos, the value of caches' content has been optimized in [13]. [28] uses layering to transport in an adaptable manner from a cache server to an end-system. Recently, multiple description coding has attracted attention. In which the usage of at least two independent base layers increases error resilience.

3.3 Structural subdivision

Like quality subdivision, structural subdivision requires content semantics to be well-defined and known by the nodes of a CDN. Structural subdivision relies on semantic knowledge of the content, such as which part of the information contained in a file is relevant, or frequently retrieved.

One multimedia format that has been developed with video compression in mind but that does not loose the structural abilities of other multimedia formats is the ISO-standardized MPEG-4. MPEG-4 can model scenes that are interactively explored. This ranges from user selection of branches of a presentation but may extend to

virtual worlds. Due to this support for navigation, parts of the presentation will be more frequently demanded than others. Knowledge of such conditions allows the selection of a structural part of an MPEG-4 presentation for caching. Alternatively, MPEG-4 can support the controlled degradation of the presentation quality of scenes to adapt to external conditions such as network load of end-system performance limits. It specifies video codecs that support fine-grained scalability and by this means, supports quality subdivision. However, the ability to construct scenes from several objects makes it also possible to exclude individual objects from delivery. For example, it would be possible to exclude semantically irrelevant eye-catchers, news anchors, or background scenes when a delivery of the complete scenes is not possible.

3.4 Next steps

In this overview of techniques for the distribution of streaming media in CDNs, it becomes clear that approaches based on caching and streaming are in the focus of research so far. A frequent complaint about the mentioned streaming techniques is that they ignore the reality of downloads at rates lower or higher than the playback speed. In fact, this distinction is just an implementation issue at the end-system. Temporal subdivision approaches can make excellent use of higher bandwidth and perform just as well as download techniques. Quality subdivision approaches, on the other hand, are meant to enable concurrent playback during download, even if the available bandwidth can not support full quality streaming.

Alternatives to the use of caching have not been explicitly discussed above. However, by applying techniques such as prefix caching, pre-storage and pre-distribution, an active replication approach is actually implied. The current schemes do explain the amount of content that needs to be replicated in detail for theoretical CDN topologies. For realistic topologies, or even for dynamically changing topologies, the appropriate use of replication is an important research issue. Existing P2P approaches can provide insight into such dynamically developing systems, even though they are typically aimed at even more dynamic systems.

From the overview of these existing options for partial distribution of content in a CDN, we derive a series of demands that are addressed by our research work.

- The exploitation of encoding formats that include such semantic information is the next step towards more efficient CDNs, which can deliver the same content over the Internet to a variety of end-systems.
- Beyond the work that exists for the delivery on scalable media to heterogenous end-systems, we will consider end-systems that roam during content reception, such that a change of servers would be appropriate.
- We will integrate quality- and structural-subdivision approaches with existing P2P ideas that make use of temporal subdivision to deliver content from several sources to overcome network bottleneck.
- Subdivided content can, and structurally subdivided content in particular must, be amended with meta-information about the relevance of its parts and relation among parts. Some of the information can be generated along with the content, other is dynamically generated by observing end-user behavior. We will investigate how this information can help a better distribution of content.
- To a lesser extent than P2P systems, CDNs have to deal with service failures and the arrival and departure of nodes. This requires an investigation of distribution approaches for their adaptivity to changes and stability when they occur. Our distribution approaches will be evaluated for their performance in this respect.

4 The QoS management hierarchy

The QoS management hierarchy has been developed in the OMODIS project for QoS support in distributed multimedia applications, with a focus on distance learning applications that use Multimedia Database Management Systems (MMDBMS) and other media services in a distributed environment [6], [7]. These applications consist of long-lived sessions where users submit requests for multimedia presentations and might specify QoS requirements per session, per request, or per multimedia object. It is important to note that the distributed application components that participate in providing service to the session are dynamically determined based on the multimedia data to be retrieved and the multimedia processing to be performed.

With respect to our environment and goals, we reviewed the state-of-the-art in QoS management services and observed the following shortcomings in previous works:

- *Static, pre-configured multi-layer QoS solutions*: Several static, three-tiered QoS management structures have been proposed [14], [20]. These management structures limit the adaptations that can be applied. For example, when replacing a component, the new service component must be within the scope of one of the previously known QoS managers or this type of adaptation cannot be

used, i.e., new QoS managers cannot be dynamically added to the configuration.

- *Component-embedded QoS management*: Early works have integrated configurable QoS services into communication protocols and end-systems in order to meet new application requirements and to support end-to-end QoS guarantees, e.g., [4], [16], [26]. Others have proposed that all QoS service issues be managed by the application [23] or by end service systems [2], [3], [8], and [26]. QoS mechanisms that handle concrete resources, like CPU time or disk bandwidth, must consider the particular properties of the resource, resulting in type-specific QoS management techniques and mechanisms that are not easily reusable.

The key concept of our QoS management hierarchy is the dynamic (re-)configuration of a hierarchy of QoS managers per session [6]. The hierarchy is of arbitrary depth and consists of application service components and two types of QoS managers: *strategic QoS managers* and *tactical QoS managers*. Basically, a strategic manager is responsible to enforce the policies or higher-order goals of a service provider or policy domain owner, and a tactical manager performs concrete control actions on *Managed Components* (MC). There is a 1:1 correspondence between strategic managers and policy domains. A policy domain contains a set of application services that are governed by a common QoS policy. A QoS policy is represented by a set of policy statements. These statements define resource limits within the domain and specify procedures that control how QoS contracts are negotiated and how QoS-based adaptation can be performed within the domain. A policy domain corresponds to an authority realm. Each policy can be independently updated depending on the implementation and the requirements of the domain.

4.1 Managed components

A MC is any system, service, or resource that presents itself as an atomic entity to a QoS manager. This means a MC can be a single server, a set of servers providing an encapsulated service, or a subsystem of services with an existing QoS management system. MCs execute on platforms. A platform may host multiple components concurrently, but for simplicity we assume that each component executes on a separate platform.

MCs may be *QoS-aware* or *QoS-unaware*. Examples of QoS-aware MCs are *Self-adapting components* [21] and *QoS-mechanistic components* [16]. *QoS mechanisms* contained in QoS-aware MCs are service-specific algorithms for maintaining QoS contracts held by that component. For example, a QoS-aware network service can implement flow filtering for multimedia streams, and channel sharing for specific types of network traffic. These mechanisms are not generic QoS management services, but a QoS manager can selectively invoke these specific mechanisms to achieve QoS goals as part of its QoS management responsibilities. We differentiate between three basic types of MCs [7]: (1) QoS-aware MCs that implement all messages defined by our QoS management middleware; (2) QoS-unaware MCs that are wrapped with component-specific software that implements messages sent by QoS management middleware; and (3) QoS-aware, multi-component service or subsystem that encapsulate MCs and its own QoS management solution, such as a QuO-managed [23] subsystem for communications, command, and control applications.

4.2 Management hierarchy and policy domains

A tactical QoS manager provides direct QoS management to a MC, using a QoS policy that is specific to that MC. Each component (or wrapped legacy component) is bound with its tactical QoS manager at compile time or at load time (by dynamic binding). Tactical QoS management protocols are uni-directional (from the manager to the MC). Tactical QoS managers are coordinated and guided by higher-level strategic QoS managers. The number of strategic managers and the depth of the QoS management hierarchy are determined by the structure of the end-to-end application and the sets of nested *QoS policy domains* over which the application is distributed. A QoS policy domain is an authority realm containing a set of application services governed by a common QoS policy. Each domain contains one strategic QoS manager that provides QoS management over all tactical and strategic managers within that domain. The strategic management hierarchy is rooted at a strategic *session manager*. The session manager enforces the QoS policy in the *session domain*, which is created when a user initiates a session and it exists only for the duration of the session.

A QoS policy is represented by a set of policy statements that define resource limits within the domain, define domain-wide QoS goals, and specify procedures that control QoS negotiation and QoS motivated adaptation within the domain. For example, QoS policy statements can specify that clients outside of the local domain can negotiate only best-effort service between the hours of 8:00 and 12:00, and that clients may never be moved between MCs during primary operating hours. Policy domains are defined, for example, within corporations, between corporations, by governments,

and by international bodies. They are long-term authority realms that exist for indefinite time periods. Each domain policy is independently updateable, online or offline, depending on the implementation and the requirements of the domain.

Figure 1 illustrates a sample end-to-end multimedia application. We use this example to further describe the essential aspects of our QoS management middleware. All MCs are represented by gray shaded icons in Figure 1. The MCs work together to provide a service to a distinguished MC called the *end-client*. The end-client initiates an application session, submits an application request, and is the final sink point for responses to the application request. The *end-user* is a human interacting with an application component executing on the end-client system. When directed by the end-user, the end-client sends a request over Net1 to the application server. The application server parses the request and determines which backend servers could be used to service the request (perhaps using a broker or trader, e.g., [11] to locate appropriate services). In this example, the application server requests data from the MMDBMS over Net2, and additional data from a web-server over Net3. The application server works as a client of the MMDBMS and the web-server. Upon receiving the multimedia request, the web-server determines that it must retrieve some archive data from another media server over Net3. Each service component has a corresponding tactical QoS manager that manages QoS for all clients of the local service. The initial service configuration is determined during QoS negotiation, based on the functional and QoS capabilities of each MC.

Figure 1 shows a hypothetical set of six policy domains, with strategic managers, overlaying the services that support the client request. The top-level, session domain is created dynamically when the client session is initiated. Three corporate policy domains (Corp A, Corp B, and Corp C) and their strategic managers are shown. Corporation C allows company departments to define QoS policies over their resources. In this example, two departments control their own resources (Dept 1 and Dept 2) and define their own departmental policy domains.

When a client session is initiated, a session-specific set of *QoS management connections* among the existing strategic managers are dynamically configured. The tactical managers and their respective MCs are the terminal leaves of the QoS management structure. For each tactical manager, a unique *QoS management chain* is formed by the set of strategic managers responsible for the nested policy domains containing the tactical manager's MC. The management chain includes all strategic managers between the tactical manager and the session manager. Each QoS manager, except the session manager, has a unique *parent manager* defined by the management chain from the tactical manager to the session manager. A *QoS management path* is a subset of a QoS management chain. A path begins at a leaf and ends before reaching the session root. Thus, the management structure forms a tree, rooted at the strategic session manager. Each level in the management hierarchy monitors and manages QoS within its scope.

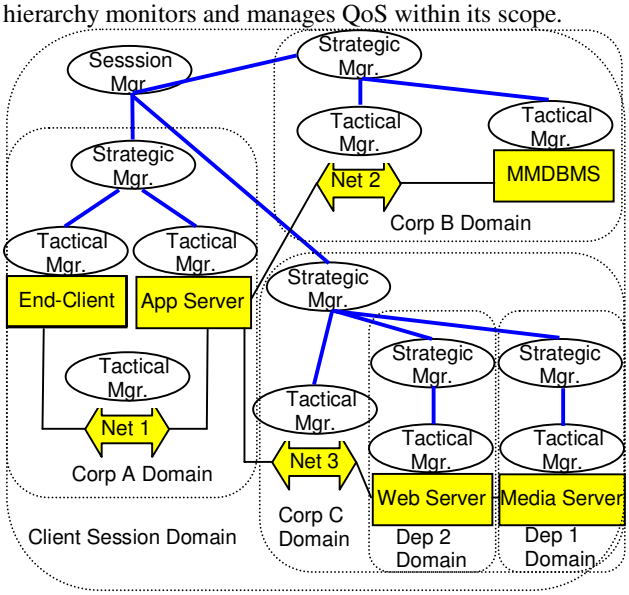

Figure 1: Policy domains and strategic QoS managers overlay components in a distributed system.

One important property of our approach is that the management hierarchy can be dynamically reconfigured, because the QoS management structure must adapt as the application structure adapts. Consider for example the following scenario: the media server in our sample application becomes overloaded and begins missing deadlines for delivering video to the web server. This QoS violation is detected by the tactical QoS manager for the media server. The manager attempts, without success, to remedy the problem through local component adaptation, i.e., increases a client's priority, taking disk bandwidth from best effort clients and giving it to the troubled client. Having exhausted the local adaptation schemes, the tactical manager escalates the problem to the strategic QoS manager for policy domain Dept 2. Based on adaptation policy and the current status of clients and MCs within the Dept 2 domain, the strategic manager creates an adaptation plan. For example, the strategic manager may move some users from the overloaded media server to another server storing partially replicated media data. If adaptation is still not effective, then the strategic manager informs its parent

manager for domain Corp C. This manager may use a service replacement adaptation to solve the problem and uses the replacement video server in a new policy domain, Corp D. In order to successfully perform a cross-domain replacement, the QoS management hierarchy must dynamically adapt as the structure of the distributed application adapts. The strategic QoS manager for the domain Corp D must be added to the QoS management hierarchy and the strategic and tactical QoS managers in the domain Dept 2 must be removed from the instantiated QoS management hierarchy.

5 Managing flexible CDNs

In this section, we discuss the combination of the three elements that have been discussed previously, i.e., P2P mechnisms for CDNs, caching and replication in CDNs, and the QoS management hierarchy. P2P mechnisms, like distributed hash tables, are used to handle in a scalable manner the dynamic aspects in a CDN, which include dynamic adding and removing of computing resources as well as dynamic creation and removing of multimedia object copies. Furthermore, lookup services from P2P solutions can be used to efficiently identify a CDN node that is close to the client, i.e., to identify those components that together provide the requested service.

The QoS management hierachy is designed for dynamic environments and will be used in the context of this research for two purposes: (1) QoS management, i.e., QoS contract negotiation, monitoring of sessions, renegotiation, adaptation, etc., and (2) providing information to the CDN such that it can make decisions concerning the dynamic creation, placement, and removal of data copies in the CDN.

It is important to note, that the QoS management hierarchy is designed to be independent of any particular application. This seeems to be in conflict with the goal of managing a flexible CDN, because many management decision must be guided by application semantics. We will discuss this potential conflict and its resolution by a typical situation CDNs have to cope with, i.e., changing popularity of data elements. The popularity of data elements changes over time and is often also depending on certain user groups. For example, a cross country skiing event that takes place in North America is popular in Norway early in the morning before people go to work, and it is popular in Germany after people come home from work. A flexible CDN could adapt to this popularity changes by placing a copy of the cross country event early in the morning on a CDN node in Oslo, and one copy in the afternoon on a CDN node in Frankfurt. In case the popularity is even that high that it leads to an overload situation for these nodes, additional replicas could help to satisfy all user requests with sufficient quality.

The decision when and where a replica of a certain multimedia object should be generated is clearly driven by application semantics. The QoS hierarchy must not understand the semantics of the multimedia object and its meta data descriptions. On the other hand, it is the QoS management hierarchy that performs QoS negotiations and knows whether they are succesful or not. Furthermore, it monitors all sessions and controls whether the QoS contracts are fulfilled or not. The application is entirely relieved from these tasks.

In order to understand the cooperation of CDN and QoS management hierarchy, it is worthwile to go step-by-step through the main interactions between a CDN and the QoS hierarchy for a single session: First, a user is requesting a multimedia object from the CDN. The CDN identifies the components that are needed to establish a session for the user, e.g., the CDN node that stores the requested multimedia object, the network between client and node, and the users end-system. If the CDN has multiple copies of the multimedia object, it returns a list of all components that hold a copy. This list is ordered according a predefined criteria, like the distance function in the P2P lookup service. The client passes the component list and its QoS requirements to the QoS management hierarchy. The QoS management hierarchy requests every involved component to initiate the establishment of a QoS management chain from itself to the session manager that roots the hierarchy. For each encapsulated policy domain there will be the strategical manager of this domain included in the hierarchy. When there is a chain from all components up to the central session manager, the QoS hierarchy is established. The task of the QoS management hierarchy is to govern the negotiation of a QoS contract. Each MC has to find out whether it can accept the QoS contract or not. This process is than performed the next higher level in the hierarchy etc. until it reaches the root of the hierarchy, i.e., the session manager. SMs that have more than one child must combine the incoming QoS contract terms and conditions. The same is valid for the session manager that performs the final decission. Finally, all MCs are informed from the session manager about the outcome of the negotiation and in case of a succesful negiotiation, they can start streaming. In case the negotiation is not sucessful, the QoS management hierarchy informs the CDN about the reason for it. By this, the CDN can collect data that can be considered in later decissions on the creation and removal of replicas. For example, if a MC is often refusing QoS contracts, it indicates that the MC is in an overload situation and a

replica should be created to improve the CDNs QoS. If a SM is often refusing a contract, it indicates that a replica in a different policy domain will improve the CDN's QoS.

The QoS managament hierarchy is monitoring all sessions and initiates a QoS renegotiation if QoS contracts are violated. The information about this could also be passed to the CDN to support later decissions on replicas. If there are already existing replicas, the QoS hierarchy must only be changed in order to include a MC that holds another copy (the list where given initially). Thus, a session can automaticlly adapt to maintain its QoS by using another copy.

6 Conclusions and outlook

In this paper, we have argued that future CDNs must be more flexible to reach scalability with low costs and can provide sufficient QoS. We have shown how P2P mechanisms and advanced caching and replication strategies can be used in CDNs in combination with the QoS management hierarchy to realize such CDNs. Our ongoing and future research is concerned with the detailed design and implementation of such a flexible CDN with QoS.

REFERENCES

[1] Charu C. Aggarwal, Joel L. Wolf, and Philip S. Yu. A permutation-based pyramid broadcasting scheme for video-on-demand systems. In Proceedings of the International Conference on Multimedia Computing and Systems, Hiroshima, Japan, pages 118--126. SPIE, June 1996.

[2] Blake, S., Black, D., Carlson, M., Davies, E., Wang, W., Weiss, W., An Architecture for Differentiated Services, IETF Internet RFC 2475, December 1998

[3] Braden, R., Clark, D., Shenker, S., Integrated Services in the Internet Architecture: An Overview, IETF Internet RFC 1633, June 1994

[4] Campbell, A., Coulson, G., Hutchinson, D., A Quality of Service Architecture, ACM Computer Communications Review, Vol. 24, No. 2, April 1994, pp. 6-27

[5] S.-H. Gary Chan and Fourad Tobagi. Distributed Server Architectures for Networked Video Services. IEEE/ACM Transactions on Networking, 9(2):125--136, April 2001.

[6] D. J. Ecklund, V. Goebel, T. Plagemann, E. F. Ecklund, Dynamic End-to-End QoS Management Middleware for Distributed Multimedia Systems, ACM/Springer Multimedia Systems Journal, Volume 8, Number 5, December 2002, pp. 431-442

[7] D. J. Ecklund, V. Goebel, T. Plagemann, E. F. Ecklund, C. Griwodz, J. Ø. Aagedal, K. Lund, A. Berre, QoS Management Middleware: A sperable, Reusable Solution, 8th International Workshop on Interactive Distributed Multimedia Systems (IDMS'01), Lancaster (UK), September 2001

[8] Gopalakrishna, G., Parulkar, G., A Real-time Upcall Facility for Protocol Processing with QoS Guarantees, 15th ACM Symp. on Operating Systems Principles (SOSP), December 1995

[9] Carsten Griwodz. Wide-area True Video-on-Demand by a Decentralized Cache-based Distribution Infrastructure. PhD thesis, Darmstadt University of Technology, Darmstadt, Germany, April 2000.

[10] Kien A. Hua, Yin Cai, and Simon Sheu. Patching: A Multicast Technique for True Video-on-Demand Services. In Proceedings of the ACM Multimedia Conference, Bristol, UK, pages 191--200, September 1998.

[11] ISO/IEC JTC1/SC21, 1997, ODP Trading Function, Report: ITU-T X.950 – ISO/IEC 13235

[12] B. Krishnamurthy, C. Wills, Y. Zang, "On the Use and Performance of Content Distribution Networks", *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, pp. 169-182, November 2001.

[13] Jussi Kangasharju, Felix Hartanto, Martin Reisslein, and Keith W. Ross. Distributing Layered Encoded Video through Caches. IEEE Transactions on Computers, 51(6):622--636, 2002.

[14] Li, B., Agilos: A Middleware Control Architecture for Application-Aware Quality of Services Adaptations, Ph.D. Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 2000

[15] Steven McCanne, Van Jacobson, and Martin Vetterli. Receiver-driven layered multicast. In {ACM {SIGCOMM}}, volume 26, pages 117--130, New York, August 1996. ACM Press.

[16] Nahrstedt, K., Smith, J.M., Design, Implementation, and Experiences of the OMEGA End-Point Architecture, IEEE Journal on Selected Areas in Communications, Vol. 14, No. 7, September 1996, pp. 1263-1279

[17] Jehan-Francois Paris, Steven W. Carter, and Darrell D. E. Long. A hybrid broadcasting protocol for video on demand. In Proceedings of the Multimedia Computing and Networking Conference (MMCN), San Jose, CA, USA, pages 317--326. SPIE, January 1999.

[18] Jehan-Francois Paris, Darrell D. E. Long, and Patrick E. Mantey. Zero-Delay Broadcasting Protocols for Video-on-Demand. In Proceedings of the ACM Multimedia Conference, Orlando, FL, USA, pages 189--197, November 1999.

[19] Subhabrata Sen, Jennifer Rexford, and Don Towsley. Proxy Prefix Caxching for Multimedia Streams. In Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies 1999, New York, NY, USA, pages 1310 --1319, March 1999.

[20] Siqueira, F. Quartz: A QoS Architecture for Open Systems, Trinity College Dublin, TCD-CS-2000-05, Ph.D. Thesis, February 2000

[21] Thimm H., Klas W., Walpole J., Pu C., Cowan C., Managing Adaptive Presentation Executions in Distributed Multimedia Database Systems, Proc. Int. Workshop on Multimedia Database Management Systems, 1996

[22] D. Tran, K. Hua, T. Do, "ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming", *Proceedings of IEEE INFOCOM*, March 30th-April 3rd 2003, San Francisco, CA, USA.

[23] Vanegas, R., Zinky, J., Loyall, J., Karr, D., Schantz, R., Bakken, D., QuO's Runtime Support for Quality of Service in Distributed Objects, Proc. IFIP Int. Conf. on Distributed Systems Platforms and Open Distributed Processing (Middleware'98), The Lake District, England, September 1998

[24] Bing Wang, Subhabrata Sen, Micah Adler, and Don Towsley. Proxy-based Distribution of Streaming Video over Unicast/Multicast Connections. In Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies 2002, New York, NY, USA. IEEE Computer Society Press, June 2002.

[25] Jörg Widmer, Robert Denda, and Martin Mauve. A Survey on TCP-Friendly Congestion Control. Special Issue of the IEEE Network Magazine "Control of Best Effort Traffic", 15:28--37, February 2001.

[26] Wolf, L.C., Resource Management for Distributed Multimedia Systems, Kluwer Academic Publishers, 1996

[27] Michael Zink, Carsten Griwodz, Jens Schmitt, and Ralf Steinmetz. Exploiting the Fair Share to Smoothly Transport Layered Encoded Video into Proxy Caches. In Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN), San Jose, CA, USA, pages 61--72. SPIE, January 2002.

[28] Michael Zink, Carsten Griwodz, Jens Schmitt, and Ralf Steinmetz. Scalable TCP-friendly Video Distribution for Heterogeneous Clients. In Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN), San Jose, CA, USA, pages 102--113. SPIE, January 2003.