Distributed systems to support efficient adaptation for ubiquitous Web

Claudia Canali
University of Parma
claudia@weblab.ing.unimo.it

Sara Casolari, Riccardo Lancellotti
University of Modena and Reggio Emilia
{casolari.sara, lancellotti.riccardo}@unimore.it

Abstract

The ubiquitous Web will require many adaptation and personalization services which will be consumed by an impressive amount of different devices and classes of users. These novel advanced services will stress the content provider platforms in an unprecedented way with respect to the content delivery seen in the last decade. Most services such as multimedia content manipulation (images, audio and video clips) are computationally expensive and no single server will be able to provide all of them, hence scalable distributed architectures will be the common basis for the delivery platform. Moreover these platforms would even address novel content management issues that are related to the replication and to the consistency and privacy requirements of user/client information. In this paper we propose two scalable distributed architectures that are based on a two-level topology. We investigate the pros and cons of such architectures from both a security, consistency and performance points of view.

Keywords: Web content adaptation, High performance architectures, Web content delivery, Ubiquitous Web access, Pervasive computing

1 Introduction

The current trend in the evolution of the Web is towards an ever increasing complexity and heterogeneity. The growth in Web popularity is related to the diffusion of heterogeneous client devices, such as handheld computers, mobile phones, and other pervasive computing devices that enable ubiquitous Web access. Furthermore, the growing complexity of Web services has increased their heterogeneity. In several cases we observe the introduction of so-called personalized Web services which aims to match the user preferences. The need for personalization led to the introduction of content adaptation services that tailor Web resources to the user preferences and to the capabilities of their clients.

Content adaptation is an expensive task from a computational point of view. As a consequence, much interest is focused towards high performance architectures capable of providing efficient and scalable adaptation services. We can define three players in the game of content adaptation: the client which issues requests, the content provider which hosts the Web resources being requested and the adaptation provider which is the intermediary entity that carries out the actual content adaptation.

In the intermediary-based scenario considered in this study, the customers of the Web content adaptation service are the end users. The content adaptation provider is typically an ISP or some network operator which provides its customer with the service of tailoring *any* Web site accessed to their needs and preferences. The adaptation service provider tailors the content delivery service on the end-user requirements. In such a scenario the economical revenue for the adaptation provider comes from the end users which are charged to access adapted and personalized contents. Users enjoy adaptation to their client device(s) as well as some form of personalization such as the creation of resources by merging multiple information sources (e.g., RSS feeds). Such services can be provided to both single users as well as institutions or companies willing to enable ubiquitous Web access for their employees, with the latter scenario seeming the most interesting from an economical point of view. An example of such service is the AvantGo company [12] which enables Web access to PDAs by tailoring the Web page layout and the embedded object size to devices with small displays. The service provided by AvantGo is mainly related to simple adaptation services, hence it is far from exploiting the whole potential of the intermediary-based content adaptation. However, this is a real-world example of such architecture, which demonstrates the feasibility of such approach also from a revenue point of view.

The idea of having a third-party to carry out content adaptation is not the only feasible solution. Some client side solution for advertisement removal or to increase user information privacy are available (e.g. Mozilla Extensions [17], like Bug Me Not [16]), however, such approach ties the user to a specific client platform, hence it lacks the generality required for a truly ubiquitous-access enabling technology. An approach that adds all adaptation services to the content provider platform [15] remains a valid solution when the popularity of the content provider is medium-low. However, with the number of clients and device

profiles continuously increasing (hundreds of different device profiles already exist [22]), an infrastructure that uses a geographically distributed system of intermediary nodes seems the most practicable solution among the existing alternatives [3, 13] to improve performance and scalability. For this reason, in the following of this paper we will focus only on intermediary-based architectures.

A common trend in the development of distributed Web architectures is towards shifting on the network edge as much services as possible. Examples of this approach are provided by the ESI [8] system or by the ACDN architecture [19]. This trend is based on the assumption that the most significant contribution to the system performance is related to the network delays. However, for some computationally-intensive adaptation services, this assumption is no longer true. The trade-off between moving services towards the edge and more centralized solutions is one of the main issues to be addressed in the design and development of intermediary-based distributed infrastructures.

This paper explores the trade-offs related to the placement of adaptation services from both a performance and a data-consistency oriented points of view. The contribution of this paper is twofold.

First, we propose two architectures for efficient content adaptation based on a two-level topology, namely *edge-oriented* and *core-oriented*, that differ for the location of adaptation services close or far from the network edge.

Second, we provide an experimental performance evaluation of the proposed architectures that outlines which choice can provide better performance and under which circumstances the performance gain is more evident.

The remainder of this paper is organized as follows. Section 2 describes the idea of a two-level topology, while Section 3 describes the two architectures that will be studied in the paper. Section 4 describes the experimental setup used to evaluate the performance of the proposed architectures. Section 5 provides the results of the performance comparison of the architectures described in the paper. Section 6 presents some related work. Finally, Section 7 provides some concluding remarks.

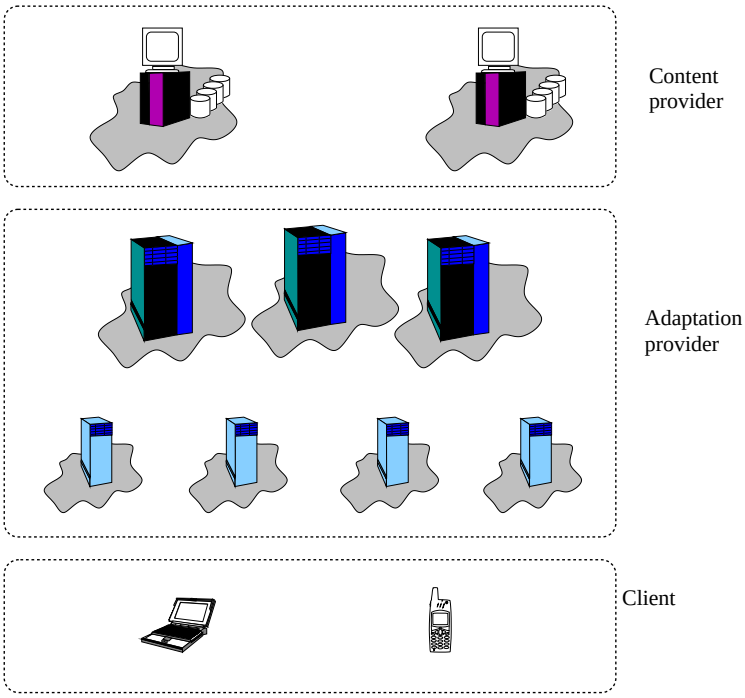

Figure 1: Two-level architecture

2 Content adaptation in a two-level topology

In this section we describe a topology for the deployment of distributed infrastructures for content adaptation, namely *two-level topology* and we discuss the actual content adaptation services that can be provided by such infrastructure. A two-level topology is based on a subdivision of nodes in two levels namely *edge* and *internal*, as shown in Figure 1.

The edge level is characterized by a large amount of nodes, each located close to the network edge. Such nodes are usually placed in the points of presence of ISPs to be as close as possible to clients, as shown in the figure. The internal level of the content adaptation architecture is composed by a smaller number of powerful nodes. Such nodes can be placed in *well connected* locations, which means in Autonomous Systems with an high peering degree, to reduce communication costs, especially with respect to the edge nodes. Figure 1 also shows the origin servers belonging to the content provider which are not part of the content adaptation infrastructure and host the repository of the Web resources (shown as the white data storage attached to the origin server nodes).

To better understand the available architectural options, we recall that the possible content adaptation services are extremely heterogeneous and each type of adaptation service can introduce different bounds on the available options, especially depending on the type of information required for the content adaptation. We can define three categories of content adaptation services [7]: *transcoding*, *state-less personalization* and *state-aware personalization*. Transcoding is a content adaptation to the client device used to access the Web. Examples of transcoding are data compression or other Web resource manipulation aiming to save bandwidth. State-less personalization is a content adaptation aiming to adapt Web resources to the user preferences. A typical example of state-less personalization is content translation into different languages or the insertion of random banners. Being state-less, this service does not rely on user profile stored in the content adaptation infrastructure. Both transcoding and state-less personalization require no special handling of (possibly) sensitive information because every information needed for the adaptation is contained in the user request (for example, information on the client device, and on the preferred language(s) can be extracted from the HTTP headers of each request). As a consequence, transcoding and state-less personalization do not place any constraint on the architecture. On the other hand, when content adaptation is carried out on the base of a stored user profile, as in the case of state-aware personalization, we must handle personal information possibly containing sensitive information. For example in the case of sophisticated ubiquitous location aware services, the position of the user introduces both privacy and consistency issues because this information is sensitive and highly dynamic in nature.

In our intermediary-based content-adaptation scenario there is no preferential access from the adaptation provider to the origin Web servers. As a consequence the adaptation provider has no way to know the semantic of the Web applications if we exclude what can be inferred by analyzing the user interactions. The lack of this knowledge hinders the development of sophisticated adaptation services which could take advantage from the knowledge about the Web page content. A typical example is provided by the insertion of context-sensitive advertisement banners: if the personalization system has no idea about the page being sent to the user, the banners can be only related to a previously stored user interests list, but cannot be tailored on the user context. Furthermore, user preferences cannot be easily

extracted from the user behavior, and cannot be automatically tuned without recurring to computational expensive data mining to be carried out off-line. Nevertheless, the deployment of interesting adaptation services is still possible. State-less personalization and transcoding tasks can be easily carried out. In a similar way, state-aware personalization tasks which are not context sensitive can be easily provided. An example would be the so-called social-navigation where multiple users can share a single annotated bookmark.

A separate note should be devoted to the impact of caching in an Web content adaptation system. While Web caching has proven to be useful in the case of traditional applications, caching of personalized resources obtained by means of dynamic content generation is usually scarcely effective because the intersection between user preferences is small. In this case caching on the client device is usually more effective than caching on intermediary nodes. For this reason we can conclude that, although useful, caching is not the most critical task in the context of advanced content adaptation services because the personalization of resources hinders the effectiveness of most caching algorithms. Hence, in the following of this paper we will not delve into the details related to the impact of caching on the proposed solutions.

3 Architectures for content adaptation

The two-level topology allows the deployment of different architectures depending on the mapping between the personalization service and the level of the topological structure. In particular in this section we define and describe two different architectures, namely *edge-oriented* and *core-oriented* that allow an evaluation of the performance differences obtained by moving content adaptation services close or far from the edge of the network.

3.1 Edge-oriented architecture

The edge-oriented architecture tries to carry out content adaptation on the edge nodes whenever this is possible, also in the case of state-aware personalization, when no sensitive information is required. To this aim the portion of the user profile not containing critical information is moved to the edge nodes, while the internal nodes are only used for the most critical personalization involving sensitive user data and as a backup repository for user

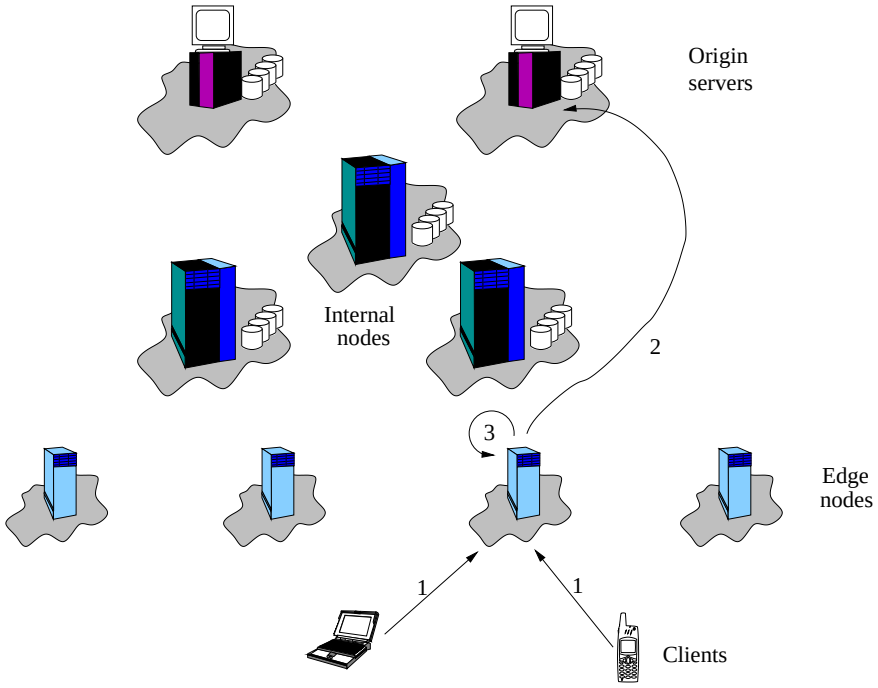

Figure 2: Edge-oriented architecture

profiles.

The choice of placing most of the adaptation services on the edge nodes rises multiple issues that must be taken into account in the design and deployment of the architecture. First, since the number of edge nodes is high, such nodes must be as simple as possible to reduce management issues. Furthermore, special care should be devoted to consistency in the information stored on the edge nodes, which means that data replication should be avoided whenever possible [11]. Finally, we must face an issue related to security. Since edge nodes are not under the strict control of the adaptation provider, sensitive information related to the user profile should not be stored in this level of the adaptation infrastructure.

Due to the limited replication of the internal nodes, we can guarantee higher security standards for them. Hence, these nodes are more suitable for adaptation tasks requiring sensitive information stored in the user profile.

Figure 2 shows the edge-oriented architecture we observe a significant amount of distributed edge nodes and few internal nodes as in the typical two-level topology. When a client request is received by an edge node (Step 1 in Figure 2), the edge node accept the

request and carries out the necessary steps for its service. In particular the edge node retrieves the Web resource(s) from the origin Web server (step 2), and carries out the content adaptation locally (Step 3).

In the case where critical user profile information is required for the content adaptation, the internal nodes are responsible for this task. In this latter case the behavior of the edge-oriented architecture is similar to the core-oriented architecture described in the following of this section. However, in our architectural comparison we prefer to take into account the most common case, hence we omit to describe in detail the case where state-aware personalization involving critical information is to be carried out.

3.2 Core-oriented architecture

The opposite approach, namely *core-oriented* architecture, forces most of adaptation services to be deployed on internal nodes. In such an architecture the edge nodes are extremely lightweight and are only responsible for forwarding client requests to the appropriate internal node.

Since most operations in the core-oriented architecture are carried out by the internal nodes, computational power may seem an issue. However, sophisticated local replication strategies can provide the amount of computation power required. An example of local replication is clustering [4] in which a Web switch is placed in front of the system to transparently distribute requests evenly among the nodes of the cluster. With such an approach the only interface to the system is the switch, which means that computational power can be seamlessly improved by adding nodes to the cluster. This complex local architecture does not introduce management problems thanks to the reduced degree of replication of the internal nodes that avoids the constraints related to the high number of edge nodes.

The core-oriented architecture is also more suitable when it is critical to guarantee adequate data consistency. The most common approach in this case is to reduce data replication. Literature shows that data consistency can hardly be provided over more than 10 nodes and the bound of synchronous updates is to be relaxed if more replication is needed [11]. The core-oriented architecture can reduce data replication based on hashing mechanisms that

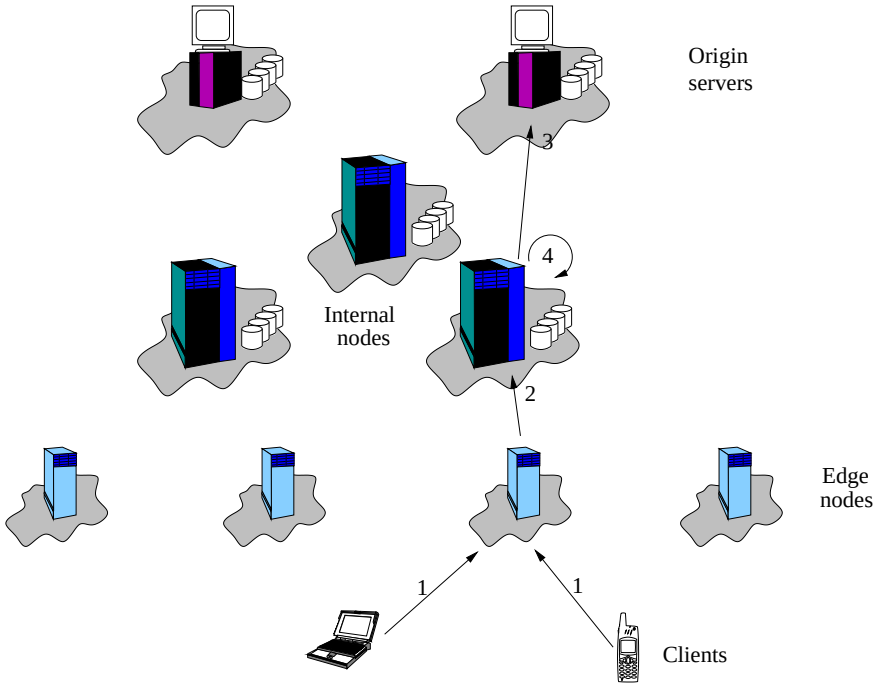

Figure 3: Core-oriented architecture

partition the space of user profiles. The user profile is replicated only on one internal node (or few nodes, depending on the hash algorithm used and on whether backups are kept) to avoid replica consistency issues.

Figure 3 shows the core-oriented architecture. When a client request is received by an edge node (Step 1 in Figure 3), the edge node decides (based on hashing algorithms applied to the information contained in the user request) to which internal node the request is to be forwarded (Step 2) for processing (Step 4). Since we assume that no caching is adopted, a fetch operation from the origin server (Step 3) has to be carried out prior of the content adaptation.

4 Workload model

We now introduce the experimental testbed and the setup used in our performance evaluation. In Section 3 we pointed out that content adaptation involves a plethora of possible services that can be deployed through a distributed architecture. In this performance evaluation we will consider three different content adaptation services based on a fixed stored user

profile. Hence, these services are of state-aware type, even if they do not involve complex personalization tasks. On the other hand, they are particularly challenging from a performance point of view since they require computationally expensive adaptations. We define three workloads, each used to exercise a different service.

The first workload, namely *banners*, aims to exercise an adaptation service that reduces bandwidth consumptions related to advertisement banners. This workload is based on a real-world trace obtained from the IRCache cooperative Web caching infrastructure. The trace is rich of small images (mostly codified according to the GIF format) representing banner files. Such images are processed through aggressive compression and transcoding algorithms to reduce image size and color depth to save bandwidth.

The second workload, namely *photo album*, aims to test a content adaptation system that enables ubiquitous access to a Web album application. In this case the traces used in the experiments contains several photos in the form of large, high-resolution JPG files. To enable ubiquitous access we must reduce bandwidth consumptions. However, data compression must not be as aggressive as in the case of the banner adaptation. Indeed, photographic images must conserve enough quality to be enjoyable, while with banner files we want to preserve just minimal information (e.g. the name of the sponsored product). Photographic images must be processed by reducing image size to adapt the photo to the small displays typical of mobile clients. Furthermore, to reduce download time, we can reduce the JPEG quality factor.

The third workload is more focused towards *multimedia* resources. In this case content adaptation aims to enable ubiquitous access to audio clips stored as MP3 files. Content adaptation in this case is mostly related to save bandwidth by means of recoding the audio streams at lower bit rates. The core of such adaptation (the MP3 recoding algorithm) is an example that suits other more complex systems, as in the case of the text-to-speech application proposed in [1].

For the performance evaluation of the proposed architectures, we set up a Web delivery infrastructure with a client node, a two-level content adaptation infrastructure composed of four edge and four inner nodes and a Web server. Each node is a equipped with a 3.0 GHz AMD Sempron processor, 512 MB of RAM and an EIDE 80GB hard drive. The

CPU frequency is the most significant configuration information since content adaptation is an heavy task for the CPU. For the Web server, we use Plone [18], a popular content management system based on the Zope middleware system as the repository of original Web resources.

In our experiments, we also take into account the effect of geographic links. Each node is connected to the others through a Fast Ethernet LAN, however, we simulate such effects by means of a WAN emulator based on special packet schedulers that are part of the 2.6 Linux kernel. We simulate two WAN effects: packet delay and bandwidth limitation. Packet delay is provided by the *netem* packet scheduler, while bandwidth limitation is obtained through the *token bucket filter* traffic shaper. Delay is modeled through a linear combination of Pareto and Gaussian distribution, as suggested in [23]. Since we are interested in the performance differences of the architectural choices, in this study we neglect the delay due to the last mile (that is, the link between the client and the edge nodes). On this link we disable WAN effect emulation.

We use the WAN emulation to create two different network scenarios. The first scenario, namely *good connection*, represents a best case for the network and is characterized by network delays of 10 ms and bandwidth of 50 Mbit/s, while for the second scenario, namely *poor connection*, we consider high latency (100 ms) and low bandwidth (1 Mbit/s).

5 Performance evaluation

We now present the main results of our experimental performance evaluation of the various architectures sketched in Sections 3. We are interested in understanding which architectural choice provides better performance and under which circumstances the difference is more evident.

We focus our analysis on the different location of the content adaptation functions. From Section 3 we recall that in the core-oriented architecture most personalization tasks are carried out on the internal nodes, while edge nodes are less used for content adaptation. On the other hand the edge-oriented architecture tries to shift as much content adaptation operations as possible on the edge level.

Table 1: Comparison of core-oriented and edge-oriented architectures (Good connection)

Workload	**90-percentile of response time [s]**		**Perf. gain [%]**
	Edge oriented	**Core oriented**	
Banners	0.29	0.34	12%
Photo album	0.60	0.64	6.3%
Multimedia	9.12	9.29	1.8%

Table 2: Comparison of core-oriented and edge-oriented architectures (Poor connection)

Workload	**90-percentile of response time [s]**		**Perf. gain [%]**
	Edge oriented	**Core oriented**	
Banners	0.46	0.80	42%
Photo album	1.47	2.17	32%
Multimedia	17.05	19.15	10%

We configured our experimental testbed to provide content adaptation on the edge and on the internal nodes according to the two architectures and we measured the response time in the two cases.

The results are outlined in Tables 1 and 2 for the good and poor network scenarios, respectively. Columns 2 and 3 of Table 1 show the 90-percentile of response time for the two architectures as a function of the workload. In particular column 2 is referred to the edge-oriented architecture, while column 3 refers to the core-oriented architecture. We also report the performance gain provided by the edge-oriented architecture over the core-oriented option. Table 2 has a similar structure, with column 2 and 3 reporting the 90-percentile of the response time of the edge-oriented and core-oriented architectures, respectively, and column 4 showing the performance gain.

If we compare the two tables, we have a confirmation of the intuitive effect that good connectivity reduces the impact of having two hops to serve the resource. Indeed, besides the absolute values, the performance gain is nearly 8 times higher in the case of poor network connection.

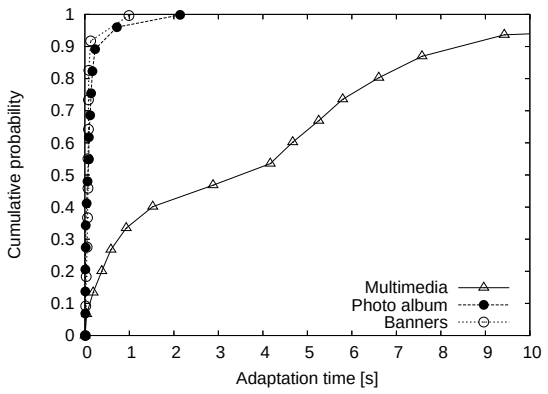

Figure 4: Content adaptation time

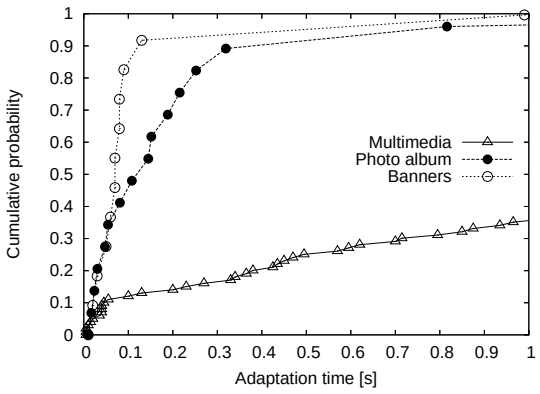

Figure 5: Content adaptation time (Zoom)

In order to better understand the performance of the proposed architectures, we also evaluate the breakdown of the user response time into its main components for the three different workloads. Such times are similar for both architectures. Hence, we report only one set of results. Figure 4 shows the cumulative distribution of content adaptation time under the different workloads, while Figure 5 presents a zoom of the curves between 0 and 1 second. We can see from Figure 5 that the 90-percentile of adaptation time for the banners scenario is 0.13 seconds, while for the photo album workload, that requires processing of larger files (pictures), the 90-percentile is almost three times higher. Figure 5 clearly shows that adaptation is more expensive in the multimedia workload, with a 90-percentile that is one order of magnitude higher than that of the other two workloads. Since multimedia content adaptation is a computationally intensive task (much heavier than the other considered services), a more sophisticated delivery architecture could rely on streaming technologies or use chuncked encoding to reduce response latency. However, this evolution is out of the scope of this paper and is left as an open issue that we plan to address in future work.

We also measure the contribution the Web server in the global response time. Figure 6 shows the cumulative distribution of the time taken by the server to satisfy client requests for the three workloads. We see that the banner workload is the less intensive for the Web server. Indeed page generation is straightforward and requires only the insertion of a random banner. From the curve in the figure we see that the 90-percentile of the server response

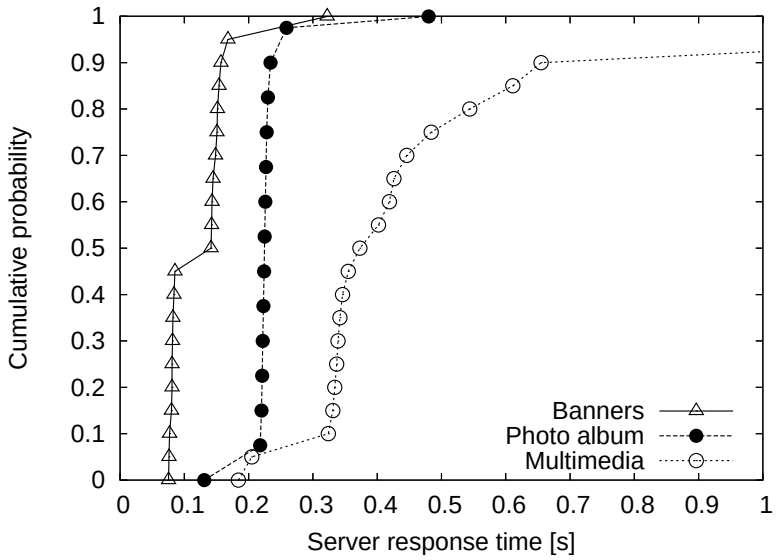

Figure 6: Server response time

Table 3: Contributions to response times

Workload	**90-percentile of response time [s]**	
	Content adaptation	**Web server**
Banners	0.13	0.15
Photo album	0.32	0.23
Multimedia	8.30	0.65

time is 0.15 seconds. The same curve also shows an interesting step close to the median value. This step is related to the Plone caching function that reduces Web page generation time in the case of cache hit. The Photo album workload is heavier also for the Web server because page generation requires the interaction with a database storing the index of the photo album. In this case the 90-percentile of response time is 0.25 seconds. Finally, the multimedia workload is the heaviest of all for the Web server. In this case ID3 tags lookup in the MP3 files leads to a 90-percentile of response time of 0.65 seconds.

The breakdown of the response time allows a more sophisticated performance comparison of the two architectures. We can infer the impact of the network-related delay from the difference between the response time in Tables 1 and 2 and the contributions in Table 3

It is interesting to note that the additional hop in the core-oriented architecture represents a performance issue mainly in the case where the adaptation time is low. If we read the column of the performance gain in Tables 1 and 2 from the top to the bottom, we see that the performance gain is reduced as the workload computational requirements grow. When adaptation time is low, the edge-oriented architecture performance clearly outperforms the core-oriented solution, as shown for the banner workload. In this case even for the best scenario (good network connection) the performance of the edge-oriented architecture over the core-oriented architecture gain is higher than 10%. On the other hand, most computationally-expensive adaptations require a time that far overweight the cost of the additional step, as shown for the multimedia workload, where adaptation time is in the order of seconds. In this latter case both the core-oriented and the edge-oriented architectures are a viable solution, even in the case of poor network connectivity.

6 Related work

Content adaptation has been an interesting topic in Web-related literature of the last years. We can ascribe most contribution to two large groups of research topics: proposal of adaptation services and proposal of efficient and scalable architectures. Most studies on content adaptation services propose novel sophisticated adaptation systems. Examples include text-to-speech conversion [1] directed to single users as well as collaborative Web browsing [2]. Even more interest has been devoted to the proposal of transcoding services aiming to enable ubiquitous Web access from heterogeneous client devices [6, 3]. However, most of these studies does not take into account performance issues and the only proposals to improve system scalability are related to the introduction of caching [21] or locally distributed clusters [9].

Studies proposing scalable content adaptation and delivery systems evaluate distributed architectures of collaborative intermediary nodes. Different architectures have been evaluated ranging from flat and hierarchical schemes [5]. However, most of these studies only focus on transcoding services, which do not introduce significant security and consistency problems. A noteworthy proposal is a peer-to-peer content adaptation system [20] called Tuxedo which allows ubiquitous Web access providing both personalization and transcoding

services. However, the study does not evaluate in deep detail security issues arising from the distribution of sensitive information among untrustworthy nodes. The importance of providing security and consistency by controlling information replication has been pointed out recently in the field of distributed Web systems [10]. Although the study does not focus on Web content adaptation but is more directed towards generation of dynamic Web content, this study confirms our concern for security and consistency guarantee which imposes bounds that must be taken into account in the design of scalable distributed architectures. Other studies [25, 24] propose algorithms and protocols to ensure Web cache consistency. However, such solutions, while reducing the cost of cache update, do not address the issues related to data privacy and cannot be easily adapted to the case of personalized Web content.

7 Conclusions

In this paper we proposed two novel distributed architectures for the deployment of content adaptation services that enable Ubiquitous Web access.

The two architectures, namely edge-oriented and core-oriented, share the same topological structure but differ for the location of the adaptation service which is close or far from the network edge for the edge-oriented and core-oriented architectures, respectively.

For each architecture we discuss the security and consistency issues that must be addressed to provide sophisticated adaptation services and we show how these issues can be addressed.

The paper also provides a performance evaluation of the performance trade-offs related to the location of adaptation services. Our experiments show that the performance gain derived from locating adaptation functions on the network edge is clearly influenced by both the network status and the computational cost of the provided adaptation services. In particular we found that this performance gain is limited (below 10%) for heavy adaptation functions (as for our multimedia workload) even in the case of poor network connection among the nodes of the intermediary infrastructure. Furthermore, this performance gain is almost negligible in the case of good network connections. On the other hand, light content adaptation services can increase the performance gain of the edge-oriented architecture up

to nearly 50% over the core-oriented architecture in the case of poor network connection.

References

[1] M. Barra, R. Grieco, D. Malandrino, A. Negro, and V. Scarano. Texttospeech: a heavy-weight edge service. In *Poster Proc. of 12th WWW Conference*, Budapest, HU, 2003.

[2] M. Bonnet. Personalization of Web services: opportunities and changes. *Ariadne*, (28), Jun. 2001.

[3] M. Butler, F. Giannetti, R. Gimson, and T. Wiley. Device independence and the Web. *IEEE Internet Computing*, 6(5):81–86, Sept./Oct. 2002.

[4] V. Cardellini, E. Casalicchio, M. Colajanni, and P. S. Yu. The state of the art in locally distributed web-server systems. *ACM Comput. Surv.*, 34(2):263–311, 2002.

[5] V. Cardellini, M. Colajanni, R. Lancellotti, and P. S. Yu. A distributed architecture of edge proxy servers for cooperative transcoding. In *Proc. of 3rd IEEE Workshop on Internet Applications*, June 2003.

[6] C. S. Chandra, S. Ellis and A. Vahdat. Application-level differentiated multimedia Web services using quality aware transcoding. *IEEE J. on Selected Areas in Communication*, 18(12):2544–2465, Dec. 2000.

[7] M. Colajanni and R. Lancellotti. System architectures for web content adaptation services. *IEEE Distributed Systems online*, 2004.

[8] Edge Side Includes, 2002. `http://www.esi.org/`.

[9] A. Fox, S. D. Gribble, Y. Chawathe, E. A. Brewer, and P. Gauthier. Cluster-based scalable network services. In *Proc. of 16th ACM SOSP*, pages 78–91, Oct. 1997.

[10] L. Gao, M. Dahlin, A. Nayate, J. Zheng, and A. Iyengar. Application specific data replication for edge services. In *Proc. of 12th WWW Conference*, Budapest, HU, 2003.

[11] J. Gray, P. Helland, P. E. O'Neil, and D. Shasha. The dangers of replication and a solution. In *Proc. of the 1996 ACM SIGMOD International Conference on Management of Data*, Jun. 1996.

[12] "iAnywhere Inc.". AvantGo, 2005. `http://www.avantgo.com/`.

[13] A. Joshi. On proxy agents, mobility, and Web access. *Mobile Networks and Applications*, 5(4):233–241, 2000.

[14] B. Knutsson, H. Lu, and J. Mogul. Architecture and performance of server-directed transcoding. *ACM Trans. on Internet Technology*, 3(4):392–424, Nov. 2003.

[15] R. Mohan, J. R. Smith, and C.-S. Li. Adapting multimedia Internet content for universal access. *IEEE Trans. on Multimedia*, 1(1):104–114, Mar. 1999.

[16] Mozilla. Bug Me Not, 2005. `http://http://www.bugmenot.com//`.

[17] Mozilla. Mozilla extensions., 2005. `http://extensionroom.mozdev.org/`.

[18] Plone. Plone: A user-friendly and powerful open source content management system document actions., 2005. `http://plone.org/`.

[19] M. Rabinovich, Z. Xiao, and A. Aggarwal. Computing on the edge: A platform for replicating internet applications. In *Proc. of 8th Int'l Workshop on Web Content and Distribution*, Hawthorne, NY, Sept. 2003.

[20] W. Shi, K. Shah, Y. Mao, and V. Chaudhary. Tuxedo: a peer-to-peer caching system. In *Proc. of PDPTA03*, Las Vegas, NV, June 2003.

[21] A. Singh, A. Trivedi, K. Ramamritham, and P. Shenoy. PTC: Proxies that transcode and cache in heterogeneous Web client environments. *World Wide Web*, 7(1):7–28, Jan./Mar. 2004.

[22] G. Singh. Guest editor's introduction: Content repurposing. *IEEE Multimedia*, 11(1):20–21, Mar. 2004.

[23] Stephen Hemminger. Netem home page. `http://developer.osdl.org/shemminger/netem/`.

[24] R. Tewari, T. Niranjan, and S. Ramamurthy. WCDP: a protocol for Web cache consistency. In *Proc. of 7th Int'l Workshop on Web Content Caching and Distribution (WCW)*, Boulder, CO, aug 2002.

[25] J. Yin, L. Alvisi, M. Dahlin, and A. Iyengar. Engineering web cache consistency. *ACM Trans. on Internet Technology*, 2(3):224–259, Aug. 2002.