Distributed architectures for high performance and privacy-aware content generation and delivery

Claudia Canali
University of Modena and Reggio Emilia
canali.claudia@unimo.it

Michele Colajanni
University of Modena and Reggio Emilia
colajanni.michele@unimo.it

Riccardo Lancellotti
University of Modena and Reggio Emilia
lancellotti.riccardo@unimo.it

Abstract

The increasing heterogeneity of mobile client devices used to access the Web requires run-time adaptations of the Web contents. A significant trend in these content adaptation services is the growing amount of personalization required by users. Personalized services are and will be a key feature for the success of the next generation Web, but they open two critical issues: performance and profile management. Issues related to the performance of adaptation services are typically addressed by highly distributed architectures with a large number of nodes located closer to user. On the other hand, the management of user profile must take into account the nature of these data that may contain sensitive information, such as geographic position, navigation history and personal preferences that should be kept private.

In this paper, we propose a distributed architecture for the ubiquitous Web access that provides high performance, while addressing the privacy issues related to the management of sensitive user information. The proposed distributed-core architecture splits the adaptation services over multiple nodes distributed over a two-level topology, thus exploiting parallel adaptations to improve the user perceived performance.

1 Introduction

The increasing need for content personalization is driving the Web towards a new dimension of interesting features and related issues that must be addressed by the underlying infrastructure for Web content generation and delivery. The conjunction of *ubiquitous Web access* and *Web personalization* allows the content providers to offer adaptation services that tailor Web resources to the user preferences and to the capabilities of their client devices. On the other hand, providing Web access for a plethora of heterogeneous and mobile client devices requires on-the-fly content adaptation services, because a pre-generation of formats for any combination of devices and network protocols is simply unfeasible. Since most content adaptation services require computationally expensive tasks, much interest of the research community has been focused on high performance architectures that are able of guaranteeing efficient and scalable adaptation services. Different highly distributed architectures [9] have recently emerged as an interesting approach to provide efficient content generation and delivery for heterogeneous and mobile users. To improve the performance of the offered services by diminishing the response time, many intermediate architectures such as CDNs propose the use of server nodes which are located closer to the clients (the so-called *network edge*) with respect to the content provider node.

As the adaptation services must access information about the user preferences, providing personalized value-added services implies an accurate management of user profiles that may contain *sensitive* information. Preserving the confidentiality of the user profile is and will be a key issue for the success of the next generation Web. The need for high performance combined with the necessity of guaranteeing an high security standard for sensitive profile information often results in a trade-off between distributed and centralized approaches. Indeed, in a centralized system that provides content adaptation and delivery, there is no problem about where to store user profile information, while a correct management of the user profiles represents a complex task when we consider highly distributed architectures.

The best choice about the various alternatives is unclear when we consider a distributed infrastructure for personalized Web content generation and delivery that is intermediate between the clients and the content providers.

The study of solutions that combine performance and privacy issues for the delivery of personalized Web contents represents a highly innovative topic. Performance and privacy issues have been addressed separately even by the recent literature (e.g., [4, 3, 7] for performance and [8, 10] for privacy), while we claim that the design of high performance architectures for Web adaptation services should be related to the solutions for a correct management of the user profiles.

The main contribution of this paper is the proposal of an intermediate distributed architecture for Web content generation and delivery, namely *distributed-core* architecture, that provides high performance from the end-user perspective through the distribution of adaptation services over multiple nodes. The *distributed-core* architecture also guarantees an adequate level of privacy in the management of sensitive information contained in the user profiles. The proposed architecture is based on a two-level topology that distinguishes between a limited number of powerful, well-connected and trusted *core* nodes and a large number of simpler *edge* nodes located close to the clients [5]. Since a client request typically refers to multiple resources (the top-level document, called container, and multiple embedded resources that can be multimedia objects) possibly requiring different kind of adaptation, the parallel execution of the onerous adaptation services on multiple nodes belonging to both the *distributed-core* and core levels allows to noticeably improve the user-perceived response time. In this paper, we evaluate the performance of the *distributed-core* architecture through a comparison with an architecture, namely *centralized-core*, that is based on the same topology but does not exploit a high degree of distribution of the adaptation services over the nodes of the architecture.

The remainder of this paper is organized as follows. Section 2 discusses the main issues related to the mapping of adaptation services over the nodes of a two-level topology. Section 3 describes the *distributed-core* architecture, while Section 4 presents a performance evaluation of the proposed architecture through real prototypes. Section 5 discusses some related work. Finally, Section 6 contains some concluding remarks.

2 Adaptation services in a two-level topology

The term *adaptation service* is used to define a plethora of services that range from simple data compression to semantic-aware handling of contents, and simple services can be combined to provide more effective access to the Web contents from herogeneous and mobile clients. The growing amount of *personalization* required by the users leads towards adaptation services that imply that some personal information is added to the user request for the generation of the resource to be delivered. There are several ways to transfer information about user preferences. For example, it may be explicitly communicated by the user through a form, or by the client device together with the request (e.g., by means of cookies or HTTP headers). Alternatively, this information may be inferred by the present and/or past user behavior or protocol. In other instances, the system infrastructure may get it from additional support, as in the case of location-based services. For complex adaptation services, it is considered unpractical to require the client to supply to the system all user profile information with every request. The most common approach is to provide only a small user identifier (e.g., a UserID cookie) and to use this information to retrieve the user profile that has been previously stored in the infrastructure for adaptation and delivery.

The information contained in the user profile may be classified in two parts: we call *sensitive* the information that are considered confidential by the large majority of the users and *impersonal* the information that can be made known. In this paper, we consider two main categories of content adaptation services, depending on whether or not the services require sensitive information contained in the user profile. Examples of adaptation services that typically do not require any sensitive information include virus scanning, that can use just a flag for enabling/disabling the service, and resource transcoding, usually based on the client device capabilities more than on the user preferences. On the other hand, services such as location-based personalization, adaptation to the user navigation style or adaptation to the user interests usually require profile information that should be kept private [9].

The need or not for sensitive information in a content adaptation service limits the possible alternatives when we have to distribute the adaptation services over the nodes of an intermediate architecture for efficient content generation and delivery.

Throughout this paper, we consider a distributed infrastructure for content adaptation that is based on a two-level topology where the server nodes are classified as *edge* and *core* nodes (Figure 1).

The *edge* level is characterized by a large amount of nodes (shown as boxes drawn with a thin line in the figure), that are located close to the network edge. This means that edge nodes are usually placed in the points of presence of ISPs with the goal of limiting the network delays in the response time of a client request. Using a set of front-end

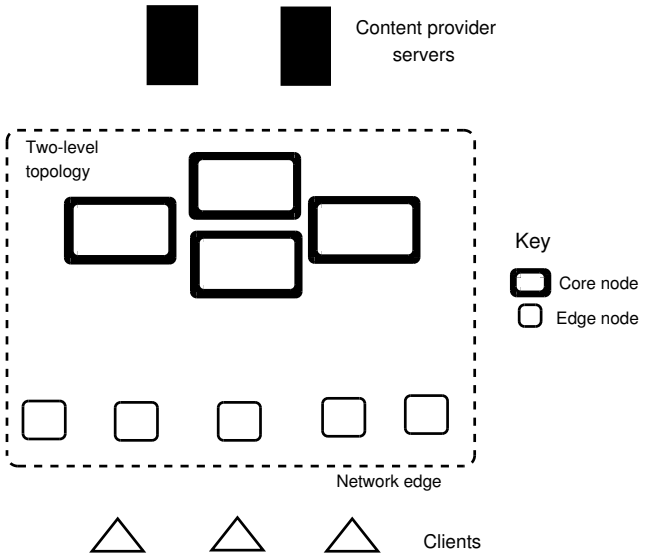

Figure 1. Two-level topology for adaptation and delivery.

nodes closer to the clients is already a common choice for letting mobile clients access Web-based services. It seems then worthy to investigate the convenience of adding novel functionalities to the edge nodes in addition to Internet access and caching.

The *core* level of the distributed infrastructure consists of a smaller number of powerful nodes (thick boxes in Figure 1). To limit communication costs especially with the edge nodes, core nodes are placed in *well connected* locations, which means in Autonomous Systems with a high peering degree. It is worth to note that the reduced number of replicas of core nodes (in the reality, one or two order of magnitude lower than the number of edge nodes) and their placement in strategic locations allows us to simplify many system organization and management aspects.

In previous papers, we have demonstrated that a two-level topology allows to better address the privacy issues arising from the management of user profiles in a highly personalized context with respect to traditional flat or hierarchical architectures [5, 9]. Indeed, it is practically unfeasible to provide a high level of security for every node in a distributed system that may contain up to thousands of nodes, while we need to guarantee high security standards for the server nodes that store user profiles containing sensitive information. If we refer to the two-level topology, we can observe that the core nodes are suitable for the management of user profiles due to their limited number and protected location. On the other hand, the edge nodes are not suitable for handling sensitive profile information for two reasons: first, their high degree of replication may cause data inconsistency problems; second, the user profile privacy cannot be guaranteed when the related files are spread among the nodes of thousands of ISP points of presence around the world.

From the previous considerations, we can conclude that the nature of profile information necessary to the personalization services and the security level of the server nodes offer important limits to the mapping of the adaptation services among the nodes of the two-level topology, because personalization services requiring access to sensitive information cannot reside on nodes that do not guarantee high security standards. Hence, to preserve the privacy of sensitive profile information we need to place personalization services that require sensitive data only on the core nodes. On the other hand, content adaptation services not requiring sensitive information, such as resource transcoding or virus scanning, can be carried out on the edge nodes.

After having identified the existing constraints about the possible mapping of adaptation services over the nodes of the two-level topology, we have to analyze how to provide high performance in the service of client requests involving personalization services. Since a client request typically refers to multiple Web objects (the base HTML container and the embedded resources) that may require different adaptation services, the key issue to achieve high performance from the perspective is to exploit concurrent adaptation tasks distributed on multiple nodes. In the next section we present the *distributed-core* architecture that provides a solution for distributing adaptation tasks on the nodes of a two-level topology.

3 *Distributed-core* architecture

The *distributed-core* architecture aims to distribute the adaptation services over multiple nodes belonging to both the core and the edge levels. The goal is to provide high user-perceived performance through a twofold action. First, the distribution of adaptation services over multiple nodes allows to exploit parallel adaptations of the different objects composing a Web resources (the HTML container and the embedded resources). Second, the use of the nodes of the edge level to perform adaptation allows to move services close to the clients.

Since adaptation services require access to user profile information, the *distributed-core* architecture needs a mechanism for distributing the profile among all the nodes responsible for the adaptation of the resources referring to a client request.

Let us now describe our solution to the problem of dis-

tributing the adaptation services and the user profile information necessary to perform the adaptation over the nodes of the *distributed-core* architecture.

The user profiles necessary for personalization services are stored on the trusted core nodes, since the edge nodes cannot guarantee adequate security standards as discussed in Section 2; in particular, we assume that each user is managed by one core node, that is called *authoritative node* for that user. The user profiles are mapped on the core nodes through a hash function $H(x)$, that receives a (unique) user ID and returns a core node identifier $k = H(\mathrm{ID})$, where $k \in [1, \ldots, n]$, and n is the number of core nodes of the two-level topology. Our experience with MD5-like hash functions demonstrates that $H(x)$ provides a fair load sharing of the users among the core nodes. Any node of the *distributed-core* architecture implements the same $H(x)$ function, hence, any node is able to identify the core node that is authoritative for a certain user. Since the core nodes need to access user profile information to perform content adaptation, we introduce a caching service on the core nodes that allows to store a local copy of the user profiles, even if the primary copy is maintained on the authoritative core nodes.

We let the core nodes to take all decisions about the distribution of the adaptation services over the nodes of the *distributed-core* architecture, because each core node manages a set of complete user profiles and, consequently, is aware of the differentiation between sensitive and impersonal profile information. The decision is driven by the classification of the user profile information: the adaptation services requiring sensitive user information are split among the core nodes, while the adaptation services requiring only impersonal information are split among the edge node contacted by the client and the core nodes. We chose to distribute part of the adaptation services not requiring sensitive information on the core nodes to avoid the risk of system bottlenecks at the edge node, since some of such services may be computationally very expensive (e.g., device-oriented transcoding) [4]. It is worth to note that the edge nodes need to be provided with the impersonal part of the profile to carry out the adaptation services they are responsible for.

For the distribution of the adaptation services on the nodes of the architecture we simply use a round robin algorithm. The use of the round robin algorithm is well known in the field of Web clusters distributed over a local area network; it has been also proposed for geographically distributed Web systems [6], but its use is typically limited to Web servers systems instead of intermediary-based architectures. Due to the limited number and the well-connected locations of the core nodes, it is feasible to consider a load-aware context where the core nodes asynchronously communicate to each other information about their current load. In this case, it is possible to use a weighted round robin algorithm based on the server loads to distribute the adaptation services on the core nodes. It is interesting to note that the use of a weighted round robin algorithm may counterbalance potential hot spots due to a not perfect load balancing of the hash function $H(x)$ that partitions the user space among the core nodes. Indeed, an authoritative node that is overloaded due to operations related to adaptation service distribution may be associated with a minor number of adaptation tasks with respect to other core nodes. However, the study of a weighted round robin algorithm is not directly addressed in this paper.

We describe the details of this mechanisms through the Figure 2 that represents the service of a client request in the *distributed-core* architecture. The operations related to the service of the base HTML container and the embedded resources are shown in separate images. The first request issued by a client for accessing a Web resource is the request for the HTML container, which is described in Figure 2(a). When the edge node receives the client request (step 1), it extracts the user ID and applies the hash function $H(x)$ previously described to identify the authoritative core node that owns the user profile. The request is forwarded to the authoritative node (step 2) that fetches the HTML page from the content provider server (steps 3 and 4) and, if required, carries out content adaptation on the HTML document (step (5) - numbers within brackets are optional steps). Then, the authoritative core node considers the Web objects embedded within that page and the classification of the user profile information. From this analysis, the authoritative core node can decide how to distribute the adaptation of the embedded objects among the core nodes of the architecture, including itself, and the edge node contacted by the client (step 6). The information about the decision is piggybacked in the response for the HTML container, that is sent back to the edge node (step 7) and then to the client (step 8). In this way, the edge node knows which nodes are responsible for the adaptation of the different object embedded within that page and will be able to forward to the appropriate node the subsequent requests coming from the client. The user profile information that is required for content adaptation at the edge node is also communicated through the response for the first file.

The sequence of the steps for managing requests for resources that have to be adapted on the edge node is described in Figure 2(b): the client request is received by the edge node (step 1) which fetches the resource directly from the content provider server (steps 2 and 3). Since the user profile information has been already copied on the edge

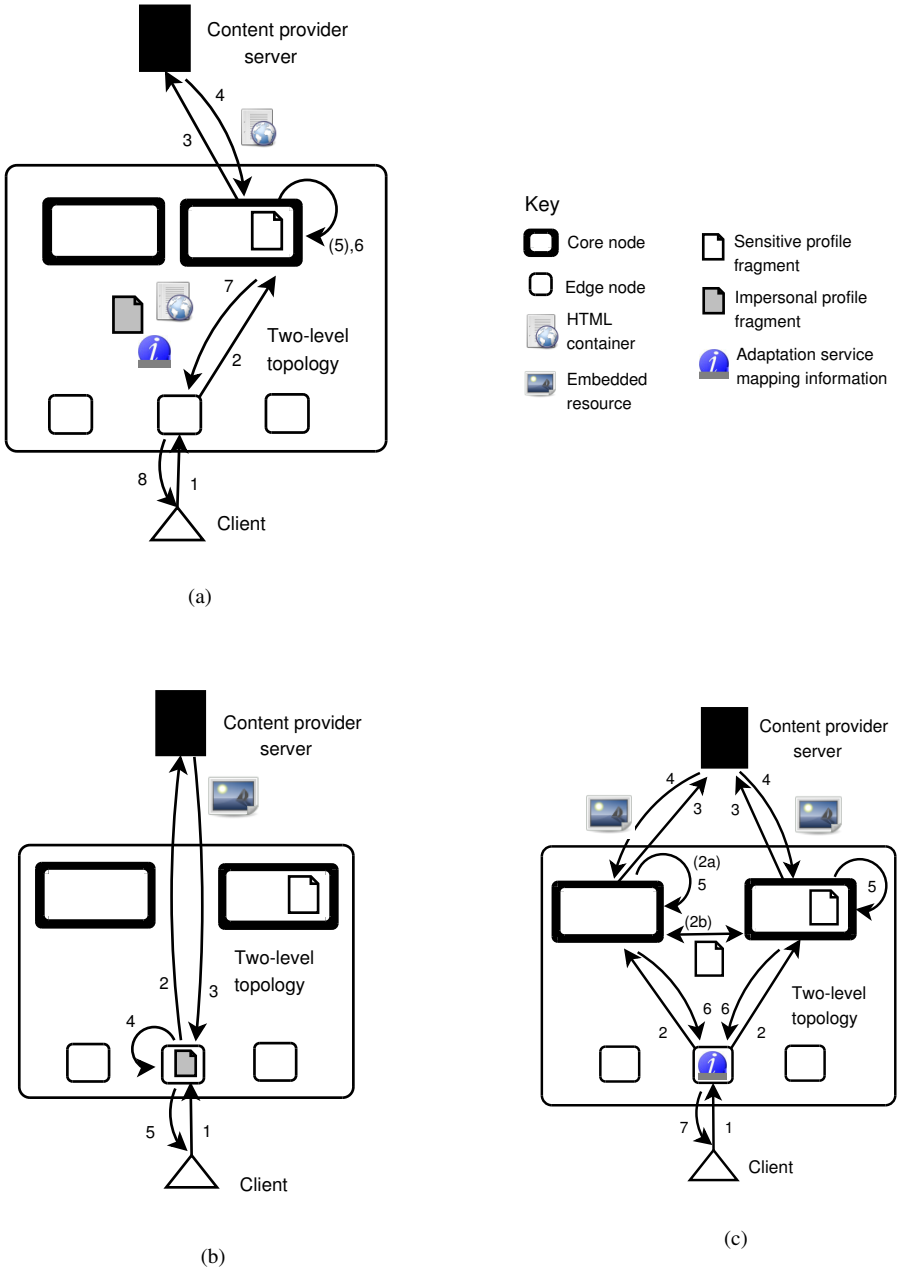

Figure 2. Request service in the *distributed-core* architecture

node, the content adaptation is carried out immediately on the edge node (step 4) before sending the adapted resource back to the client (step 5).

The case of requests involving adaptation on the core nodes is described in Figure 2(c). After receiving the client request for an embedded resource (step 1), the edge node contacts the core node associated to that resource on the basis of the information received with the response for the HTML container (step 2). At this point we may have two possible cases. If the contacted core node is the same node

that manages the profile for the user, then it certainly owns the profile information necessary to perform adaptation. Otherwise, the node has to check if a copy of the profile is present in the local cache (step (2a)); if the node does not own the profile, it retrieves the user information through a direct request to the core node responsible for that user (step (2b)), then caches a copy of the profile. Now the core node can fetch the Web resource from the content provider server (steps 3 and 4) and carry out the adaptation (step 5). Finally, the adapted resource is sent back to the edge node (step 6) and then to the client (step 7).

4 Experimental results

In this section we present a performance evaluation of the *distributed-core* architecture for ubiquitous Web access. To measure the performance gain achievable by exploiting parallel adaptation tasks in the service of a client request, we compare the *distributed-core* architecture with a simplified variant, namely *centralized-core* architecture, based on the same two-level topology.

The *centralized-core* architecture distributes the adaptation services among two only nodes: the edge node contacted by the client and the core node responsible for the user profile that issued the request. To preserve the privacy of sensitive data, only the adaptation services requiring impersonal user information can be carried out on the edge node.

To perform a fair comparison between the *distributed-core* and the *centralized-core* architectures, we use the same mechanism to distribute the adaptation tasks among edge and core nodes: all the adaptation services requiring sensitive information are carried out on the core node, while the services based on impersonal data are divided between the edge and the core nodes to avoid bottlenecks at the edge node. It is worth to note that in the *distributed-core* architecture the edge node contacted by the client is responsible for the adaptation of $1/n$ of the embedded resources not requiring sensitive information, where n is the number of the core nodes. Similarly, in the *centralized-core* architecture the edge node is responsible for carrying out the half of the adaptation tasks not involving sensitive information.

To evaluate the performance of the *distributed-core* and *centralized-core* architectures we implemented two prototypes based on the popular Apache Web server version 2.0 [1]. The profile management software is written in Perl and is processed by the Web server using the *mod_perl* module [13]. This choice allows us to combine the flexibility of an high level programming language with the efficiency of an Apache module that can interact with the Web server internals. The user profiles are stored as XML files that are accessed through the profile management software layer.

To exercise our prototypes, we use a workload model based on traces collected from a real Web site. HTML resources contain embedded objects ranging from a minimum of 2 to a maximum of 18, with a mean value of 10. The mean size of a single embedded object is 8.5 KB. The service times of the content adaptation services are characterized by heavy-tailed distributions, with 140 and 350 ms for median and 90-percentile, respectively. We assume that the time of an adaptation service does not change if the service is performed on an edge or a core node.

We set up a system with client nodes running the load generator *httperf* [11]. The two-level architecture consists of 8 edge nodes and 5 core nodes that are equipped with our prototypes. Another node hosts the Web server with the original Web resources. The client and the nodes of the two-level architecture are connected through a fast Ethernet network, while the Web server is placed in a remote location. The clients issue requests for 3000 pages (and related embedded objects) at the rate of 5 pages per second. The client requests are evenly distributed among all edge nodes.

For our performance evaluation, we compare the *centralized-core* architecture with the *distributed-core* architecture. For the *distributed-core* architecture we consider various scenarios that differ for the number of core nodes used to distribute the adaptation services related to a client request for a page (and related embedded objects). Figure 3 shows the 90-percentile of the page response time for the *centralized-core* and *distributed-core* architectures under the different scenarios. In particular, the x-axis reports the increasing number of core nodes used by the core architecture for the distribution of the adaptation tasks.

From Figure 3 we observe that the *distributed-core* architecture allows to achieve a significant performance gain over the *centralized-core* architecture under all the considered scenarios, except for the case where only one core node is used: in this case the adaptation tasks are divided in an identical way between the nodes of the two architectures leading to similar page response times. When a major number of core nodes is considered, the 90-percentile of the response time for the *distributed-core* architecture decreases as the number of the core nodes grows. The *distributed-core* architecture allows to reduce the page response time by exploiting parallel executions of the required adaptation tasks, thus achieving a performance gain up to the 74% of the 90-percentile of the response time in the case of 5 core nodes.

These results confirm our intuition that the distribution of the adaptation services over multiple nodes of the intermediate architecture allows to achieve a significant performance gain, especially in a high personalized context where

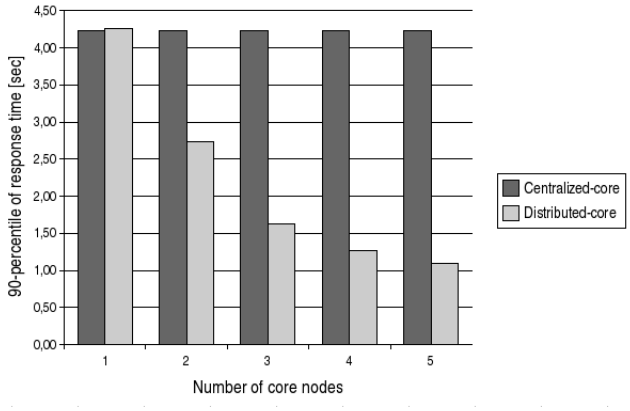

Figure 3. 90-percentile of the page response time

adaptation typically requires computationally expensive operations.

It is worth to note that, passing from one to two core nodes, the *distributed-core* architecture provides a performance gain of 35% over the *centralized-core* approach, as shown in Figure 3. If we consider further increasings in the number of the core nodes used to distribute the adaptation services, we observe more limited performance improvements (13% passing from 4 to 5 core nodes). This result is motivated by a twofold reason. First, the positive impact of an additional node on the load sharing is less effective if we already distribute the adaptation services over several nodes. Second, the overhead introduced by the mechanism for distributing adaptation services over multiple nodes increases as the number of nodes growths, thus reducing the performance gain achievable over the *centralized-core* approach.

5 Related work

The ubiquitous Web access has been recognized in literature as a fundamental challenge by multiple authors [17, 15]. Multiple proposals for content adaptation services are present in literature, both in the field of simple Web content transcoding [7] and in the case of more complex pervasive computing applications [14, 2]. However, such studies focus mainly on the heterogeneity of client devices, without taking into account the issues related to the privacy of sensitive information contained in the user profiles. A noteworthy example of these proposals is a peer-to-peer content adaptation system [16], called Tuxedo, which allows ubiquitous Web access providing both personalization and transcoding services, but the study does not evaluate in deep detail the issues arising from the distribution of sensitive information among untrustworthy nodes.

The importance of providing privacy in the user profile management has been pointed out recently in literature [8, 10]. The need of taking into account privacy guarantees in the design of scalable distributed architectures is confirmed by P3P (Platform for Privacy Preferences) [12], a W3C proposal that suggests a mechanism for Web sites to encode their privacy policies in a standardized format that can be easily retrieved and interpreted by user agents. However, these studies are more tailored to a server-side approach for the generation of personalized Web content rather than to the intermediary-based model for Web content adaptation considered in this paper.

6 Conclusions and future work

The design of architectures for the delivery of highly personalized Web contents has to address two critical issues: the user-perceived performance and a correct management of the sensitive information that may be contained in the user profiles.

In this paper we proposed a distributed architecture, namely *distributed-core* architecture, that distributes adaptation services over multiple nodes of a two-level topology. The *distributed-core* architecture provides high performance by exploiting parallel executions of the adaptation tasks, addressing at the same time the privacy issues related to the management of sensitive information. We evaluate the performance of the *distributed-core* architecture trough the comparison with a simplified variant, namely *centralized-core* architecture, that is based on the same two-level topology, but does not exploit a high degree of distribution of the adaptation services over the nodes of the architecture. Our experimental evaluation confirms that the *distributed-core* architecture allows to achieve a performance gain up to the 74% of the 90-percentile of the response time over a less distributed approach.

The results of this study suggests future research directions. We plan to study in deeper details the overhead introduced in our architecture by the management of user profiles. In particular, the overhead due to profile management can be critical in a system that updates user profile information in a dynamic way. Another issue that deserves further investigation is the study of load sharing mechanisms for the distribution of the adaptation services over the distributed nodes of the architecture.

References

[1] The apache software foundation: Apache, 2005. http://www.apache.org/.

[2] P. Bellavista, A. Corradi, and C. Stefanelli. The ubiquitous provisioning of internet services to portable devices. *IEEE Pervasive computing*, 2002.

[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] C. Canali, V. Cardellini, and R. Lancellotti. Content adaptation architectures based on squid proxy server. *Word Wide Web Journal*, Mar. 2006.

[5] C. Canali, S. Casolari, and R. Lancellotti. Distributed systems to support efficient adaptation for ubiquitous web. In *Proc. of the 2005 International Conference on High Performance Computing and Communications (HPCC 2005)*, Sorrento, Italy, September 2005.

[6] 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.

[7] 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.

[8] R. K. Chellappa and R. G. Sin. Personalization versus privacy: An empirical examination of the online consumer's dilemma. *Information Technology and Management*, 6(2-3), April 2005.

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

[10] R. Hull, B. Kumar, D. Lieuwen, P. F. Patel-Schneider, A. Sahuguet, S. Varadarajan, and A. Vyas. Enabling context-aware and privacy-conscious user data sharing. In *Proc. of 2004 IEEE International Conference on Mobile Data Management (MDM'04)*, 2004.

[11] D. Mosberger and T. Jin. httperf - a tool for measuring web server performance. In *Proc. of Workshop on Internet Server Performance*, Wisconsin, Jun 1998.

[12] Platform for privacy preferences project, 2005. http://www.w3.org/P3P.

[13] The apache software foundation: mod_perl, 2005. http://perl.apache.org/.

[14] H. Rao, Y. Chen, D. Chang, and M. Chen. imobile: a proxy-based platform for mobile services. In *Proc. of the 1st workshop on wireless mobile Internet (WMI2001)*, 2001.

[15] D. Saha and A. Mukherjee. Pervasive computing: A paradigm for the 21st century. *IEEE Computer*, 36(3), Mar. 2003.

[16] 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.

[17] G. C. Vanderheiden. Anywhere, anytime (+ anyone) access to the next-generation www. *Computer Networks and ISDN Systems*, 8(13), Sep. 1997.