A two-level distributed architecture for efficient Web content adaptation and delivery

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

Valeria Cardellini
University of Roma "Tor Vergata"
cardellini@ing.uniroma2.it

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

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

Philip S. Yu
IBM T.J. Watson Research Center
psyu@us.ibm.com

Abstract

The complexity of services provided through the Web is continuously increasing as well as the variety of new devices that are gaining access to the Internet. Tailoring Web and multimedia resources to meet the user and client requirements opens two main novel issues in the research area of content delivery. The working set tends to increase substantially because multiple versions may be generated from the same original resource. Moreover, the content adaptation operations may be computationally expensive. In this paper, we consider third-party infrastructures composed by a geographically distributed system of intermediary and cooperative nodes that provide fast content adaptation and delivery of Web resources. We propose a novel distributed architecture of intermediary nodes which are organized in two levels. The front-end nodes in the first tier are thin edge servers that locate the resources and forward the client requests to the nodes in the second tier. These interior nodes are fat servers that run the most expensive functions such as content adaptation, resource caching and fetching. Through real prototypes we compare the performance of the proposed two-level architecture to that of alternative one-level infrastructures where all nodes are fat peers providing the entire set of functions.

1. Introduction

A new emerging trend of the Web evolution is the growing complexity of Web-based services and multimedia content and the contemporary diffusion of heterogeneous client devices, such as handheld computers, personal digital assistants (PDAs), mobile phones, and other pervasive computing devices, characterized by diverse processing power, storage, display, and connectivity capabilities. Hence, there is an increasing demand for content adaptation and personalization services that enable on-the-fly transformation of (possibly) complex Web content and its delivery to these diverse destination devices.

An approach that adds all adaptation and personalization services to the content provider platform [7] 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 [13]), an infrastructure that uses a geographically distributed system of intermediary nodes seems the most practicable solution among the existing alternatives [1] to improve performance and scalability. The distributed nodes of this intermediate infrastructure, that are interposed in the client-server path, shift the load away from the origin servers of the content providers, simplify their design, and reduce the user-perceived latency when the adapted content can be served from a node closer (i.e., better connected) than the origin server or when it is possible to find an already adapted version of the requested resource.

In this paper we consider an intermediary-based adaptation architecture operated by an independent third party that manages the adaptation and delivery process for the Web resources of any provider. (The solution of an intermediary infrastructure operated directly by the content provider or by a third-party company on behalf of its customer content providers can be easily implemented by the proposed architecture. Indeed, the possibility of strict cooperation between the content provider and the delivery infrastructure would even simplify many design and implementation choices done in this paper.)

Any intermediary-based infrastructure has to provide different functions. First, it has to intercept client request and provide the response that fulfills adaptation requirements without the intervention of any visible component

for the user. Besides this gateway function, the other services that the infrastructure may or must provide include location, adaptation, fetching, and caching of Web resources in both the original and adapted versions.

Any intermediate system can provide an efficient service of adaptation and delivery through the Internet if it is able to:

1. preserve the locality of the requests, because content adaptation causes an augment of one order of magnitude of the Web resource set;
2. share the load among multiple nodes, because content adaptation operations may be computationally expensive.

In this paper we propose an intermediary-based architecture that address the previous goals in an original way. It is based on a two-level distributed scheme, where the the front-end level consists of *thin* edge servers, which are closer to the users and provide light services (at most, gateway, location, and caching operations) from the computational point of view. The computationally expensive functions are concentrated in *fat* interior nodes, which are typically located in the network core that is, in the Autonomous Systems that have many peering points with other Autonomous Systems and are well connected to the Internet backbones. The details of the proposed architecture are described in Section 2. For now, it is important to evidence that the proposed architecture addresses the two main goals that we have identified for achieving fast content adaptation and efficient delivery services through a differentiation of the nodes composing the infrastructure for adaptation and delivery. The architecture preserves the access locality of the requests by using a content partitioning technique, so that each interior node is responsible for a subset of original and adapted resources. As a consequence, the maximization of request locality allows an aggressive reuse of previously adapted resources and has a positive effect on the reduction of the global infrastructure load as it reduces the amount of adaptation operations. Moreover, for each client request a communication must occur only between a node at the edge and one at the interior level, so there is no need of message exchanges among the nodes belonging to the same tier.

We implemented the proposed two-level architecture as a working prototype to demonstrate that it is quite compatible with the Web standards. The real test-beds allow us to show the advantages of the two-level architecture over one-level infrastructures. To the best of our knowledge, the proposal is quite original. Other results about intermediate distributed architectures exist in the case of traditional proxy caching and Content Distribution Networks (see [8] for a complete survey on this topic). However, they have not addressed the main issues we are facing in this paper that is, the computational costs of adaptation and the explosion of the working set of the future Web. Other related work is discussed in Section 5.

The rest of this paper is organized as following. Section 2 describes how the two-level content adaptation architecture organizes the server nodes and the Web content. Section 3 outlines alternative one-level distributed architectures for intermediate content adaptation. Section 4 presents the experimental results on the performance of the two-level architecture and other one-level schemes. Section 5 discusses related work in the context of the intermediary-based approach for content adaptation. Section 6 concludes the paper with some final remarks.

2. Two-level content adaptation architecture

In this section we describe the proposed two-level architecture. We first focus on the organization of the servers and resources, and then on client request management. We also outline the main advantages of this new architecture.

2.1. Server organization

Any distributed intermediary-based infrastructure for content adaptation and delivery may or must provide several functions.

Gateway: the infrastructure must intercept client requests and provide the client with the response that fulfills adaptation requirements without the intervention of any visible component for the user.

Location: the infrastructure must identify the remote node(s) that may provide the requested service and/or have a valid copy of the resource.

Adaptation: some node(s) must adapt, when necessary, the original resource to the specific needs of the user, client device and connection.

Fetch: the infrastructure must retrieve the original resource from the origin server, when a valid copy of the requested resource is not found in the nodes of the intermediate infrastructure.

Cache: some node(s) must store original copies and adapted versions of the requested resources.

The common choice of all previous proposals of intermediate architectures for content adaptation and delivery is oriented to fat edge nodes that provide all the previous functions. Two traditional architectures are described in Section 3. Instead, the key idea of this paper is to propose a novel architecture where there are two tiers of nodes that execute, at least in the basic version, two distinct types of functions.

The first level consists of *edge* nodes that are located on the borders of the Internet as described in Fig. 1. These edge nodes are *thin* because they execute *gateway* and *location* servers that are not expensive from the computational and space point of view. In practice, the edge nodes receive the

client requests, identify the interior node that can serve each request, and forward the request to that node.

The second level consists of *interior* nodes that are located in the network core that is, they are in Autonomous Systems of international providers that have a large number of BGP peerings with other Autonomous Systems. The interior nodes are *fat* because they provide the more complex functions of *adaptation*, *caching*, and *fetching*. Only these nodes can interact with the origin servers when the intermediate infrastructure is not able to fulfill the client request by means of previously cached Web resources.

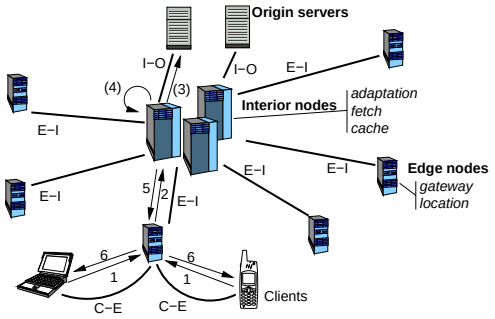

Figure 1. Two-level architecture for intermediary-based adaptation and delivery.

2.2. Data organization

The explosion of the working set size caused by content adaptation services is another issue that the two-level architecture addresses in an original way. To improve the cache hit rate in a working set that can easily become one order of magnitude larger than the set of the original resources, the basic idea is to maximize the locality of the requests. The two-level infrastructure achieves this result through a combined strategy. First, it uses a hash-based resource partition scheme, so that each interior node manages a different subset of the global space of resources. Second, all the adapted versions of the same original resource are kept in the same interior node.

The mapping between a Web resource and an interior node is carried out through the hash function $H(x)$. $H(x)$ takes as input a resource ID (e.g., the URL) and returns a value $k = H(\text{resourceID})$, where $k \in [1, \ldots, n]$, and n is the number of interior nodes. An example is $H(\text{resourceID}) = (\text{MD5}(\text{resourceID}) \bmod n) + 1$ that uses the MD5 algorithm to distribute the resources uniformly across n nodes.

It is worth observing that the hash function is computed only on the original resource without taking the object version into account. Through this choice, we address in a simple way all lookup issues related to the need of preserving access locality of the requests. Indeed, if multiple versions of the same resource exist, they are stored on the same interior node and we can avoid to look for Web resources in other interior nodes.

2.3. Management of client requests

When an edge node receives a client request (step 1 in Fig. 1), it extracts the resource ID and applies the hash function to identify the interior node that with some probability holds a valid version of the requested resource (step 2). It is interesting to note that there is no need for cooperative lookup, because if the interior node receiving the request from the edge node does not hold the requested content, no other interior node does. Hence, any request to an interior node has the three possible results. (The numbers between brackets specify steps that are not always executed.)

Exact hit: if the cache of the contacted interior node contains the exact version of the requested object, the resource is immediately sent to the edge node (step 5 in Fig. 1).

Useful hit: if the cache contains a more detailed and adaptable version of the requested object, it can be transformed (step (4)) to meet the client request through a content adaptation process. After that, the interior node sends the exact resource to the edge node (step 5).

Miss: the interior node must fetch the original resource from the origin server (step (3)). If necessary, it adapts the original resource (step (4)), and then sends it back to the edge node (step 5).

The final delivery of the resource to the client (step 6) is always carried out by the gateway server that is located on the edge node.

The two-level architecture requires a twofold location step because each client request must always reach an edge server and then an interior node. Therefore, it is important to analyze the sensitivity of this architecture to the network delays among the nodes of the distributed architecture and the origin servers. We will show in Section 4.2 that the higher access locality achieved by the two-level architecture compensates the twofold location step. For now, we find useful to denote in Fig. 1 the classes of network links that we will consider in our analysis.

Client-to-Edge node links (*C-E* in Fig. 1), connect clients to edge nodes. **Edge-to-Interior** node links (*E-I* in Fig. 1) connect edge nodes to interior nodes. They carry client requests from edge nodes to interior nodes and the exact resources in response to those requests. As we assume that interior nodes are located in well-connected Autonomous Systems, they can be possibly far from the edge nodes, but with large bandwidth connections with them. **Interior-to-Origin server** node

links (*I-O* in Fig. 1) connect the intermediate infrastructure to the origin servers.

2.4. Other benefits of the two-level architecture

We have outlined in Section 2.2 how the proposed two-level architecture addresses both issues caused by the growth of the working set size and by the computational requirements of content adaptation. Here, we discuss other benefits related to the choices of this architecture, including server management, data consistency, and privacy issues.

Edge nodes are gateway servers and request redirectors that receive all requests and deliver all content to the clients. These operations may stress certain system resources (especially network resources and file descriptors), but they are not CPU and memory intensive. Hence, a large number of edge nodes can be easily spread around the Internet borders (local ISP providers) because they do not require complex software and powerful machines. On the other hand, interior nodes execute network-, memory- and CPU-bound operations, such as caching, fetching, and adaptation, that require more powerful platforms. For this reason we find convenient to locate the interior nodes in well-connected and controlled positions. The number of interior nodes may be even one-two orders of magnitude less than that of edge nodes. Strategic location of interior nodes in few well-connected Autonomous Systems provides also the further advantage of reducing response time variability.

It is also worth noting that adaptation often requires transcoding operations guided by the type of client device and network connection, but in other cases resource adaptation may be related to the user behavior and preferences. The important consequence is that personalization functions introduce new security and privacy concerns on stored data because they require additional information on the user, such as user profiles or history of user behaviors. Placing critical information only on the interior nodes that are located in few controlled sites instead of spreading data among several edge nodes allows us to address the issues related to security in an easier way. Moreover, reducing the replication of critical and volatile data (such as user profiles on behaviors and preferences) helps in guaranteeing their consistency. However, all these problems and related solutions are beyond the scope of this paper, and in our experiments we do not explore their performance.

3. Alternative intermediate architectures for content adaptation

In a traditional one-level (*flat*) architecture for content adaptation and delivery services, the nodes are all peers and are typically placed at the edge of the content delivery chain. Flat architectures rely on *fat* edge nodes that provide all the functions that is, each node may execute *gateway*, *location*, *cache*, *adaptation*, and *fetch* servers.

An intermediate flat architecture consisting of a geographically distributed set of edge nodes may have two main methods of operation. The difference is whether the nodes cooperate or not in resource lookup, and how to cooperate in the former case.

In the simpler architecture, denoted by **No_Coop**, the edge nodes provide all functions for content adaptation, caching, fetching and delivery, but they do not implement cooperative schemes for location and caching of resources in other peers. Hence, if no valid copy of the requested object is found in the cache of the contacted edge node, this node must fetch the original resource from the origin server, if necessary, adapt that resource, and send it to the client. As there is no attempt to cooperate for locating resources in remote peers, exact and useful hits can only be local.

The alternative architectures implement some form of cooperation among the edge nodes. In particular, the location function allows the edge node to take advantage of cache contents of peers. Different cooperation schemes are available in flat architectures. The first idea is to investigate whether the same previous approach of Web resource partitioning could be apply or not to an intermediate flat architecture consisting of geographically distributed edge peers. We exclude immediately this hypothesis for two main reasons. First, partitioning the resources among a large number of edge nodes increases the variance of the system response time, because a client request that cannot be satisfied by the contacted edge node must be forwarded to a peer that could be near or very far from the first node. In most instances, it would become convenient to fetch the resource from the origin server. Second, the addition/replacement of an edge server would become a complex operation because it would require a complete redistribution of original and adapted contents. (Partition schemes such as consistent hashing [10] can reduce the effects of content redistribution, but do not solve completely the problem.) Indeed, a similar problem exists for the interior nodes of the two-level architecture, but in this paper we can neglect it because the interior nodes are one-two orders of magnitude less than the edge nodes, and they represent a quite stable set.

If we relax the requirement of resource space partition and allow content to be replicated among the edge nodes, the content location function becomes the peculiar element for distinguishing the intermediate flat architectures. Node cooperation may occur according to several schemes, such as query- or summary-based lookup schemes. In previous studies the authors have compared multiple cooperation schemes in flat architectures. In particular, in [2] they show that a query-based cooperation achieves the best performance thanks to the effectiveness of the resource loca-

tion algorithm. Due to space limits, in this paper we consider only flat architectures that use a query-based cooperation scheme, denoted by **Flat**. In Fig. 2 we give an example of the operations of this system by taking into account one edge node that receives client requests from possibly heterogeneous devices. When the gateway server on this node receives a request (step 1 in Fig. 2), it looks for the requested resource in the server cache. If it does not find a useful resource, it activates a cooperative lookup process that is based on a query-based scheme, such as ICP. In this case, the edge node sends a query message to each peer (step (2)). (The solid lines represent HTTP connections, the dashed lines denote query messages, and the brackets identify the steps that are not always executed.) A response from a peer denotes a remote hit on a peer (step (3a)) from which the resource must be explicitly fetched (step (3b)). When no suitable resource is found in any peer, the edge node fetches the resource from the origin server (step (4)).

The discovery process must cope with the possibility of having different versions for the same resource. When an exact version of the requested resource is not found in any peer, an adaptation operation is required (step (5)). Therefore, the caching semantic is more complex with respect to the two-level architecture: a local lookup may result in a *miss*, in a *local exact hit* or in a *local useful hit*; a remote lookup may result in a *miss*, in a *remote exact hit* or in a *remote useful hit*. Once an exact version of the requested object is found or generated by the adaptation operations, the gateway server on the edge node transfers the resource to the client (step 6 in Fig. 2).

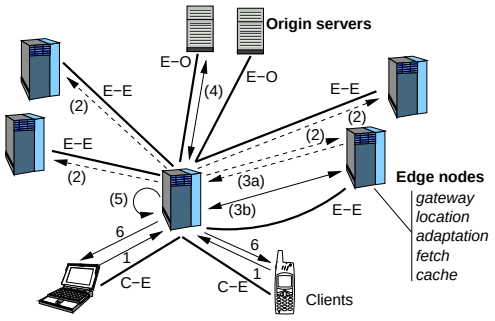

Figure 2. Flat architecture for intermediary-based adaptation and delivery.

In Fig. 2 we also show the classes of network links that we consider for the sensitivity analysis to the networks parameters carried out in Section 4.2. Besides client-to-edge node links (*C-E* in Fig. 2) that connect clients to edge nodes as in the two-level architecture, there are the following links. **Edge-to-Edge** links (*E-E* in Fig. 2) interconnect edge nodes and are used whenever some interaction among the peers required; for example for cooperative lookup in the case of local misses, and for resource retrieval in the case of remote hits. **Edge-to-Origin server** links (*E-O* in Fig. 2) connect the intermediate infrastructure to the origin servers.

4. Experimental results

The main goal of our experiments is to compare the performance of the flat, two-level and non cooperative architectures. As performance indexes we consider the *cache hit rates* (local, remote, global, exact, useful), and the *response time* that represents the interval elapsed between the instant in which the client sends a request to the edge node and the instant in which the client receives the response.

A fair comparison between the architectures should be based on the same total amount of CPU power and cache capacity for the servers of the intermediate infrastructures that provide caching, fetching, and adaptation functions. However, we decided to be a little bit unfair with the proposed two-level scheme to demonstrate its good performance. Hence, for our experiments we set up an intermediate system consisting of 16 nodes for the flat architecture and 14 interior nodes for the two-level architecture. As the nodes have same CPU power, in the experimental results we should consider that the two-level architecture may count on 12.5% less power for fetching and adaptation services. In the flat scheme all nodes have sibling relationships for cooperation.

In our experiments we set up the intermediary-based infrastructure consisting of servers that are equipped with our prototypes and configured according to the previously presented schemes; the nodes are connected through a fast Ethernet network (that provides E-E and E-I links, according to Figs. 1 and 2). In the first set of experiments, the origin servers are placed in a remote location that is connected with the nodes of the intermediate architecture through a geographic link (E-O or I-O link) with 14 hops in between, a mean round-trip time of 60 ms, and a bandwidth of 4 Mb/s.

The workload model aims at denoting a scenario where the adaptation operations have a high computational cost. As the trend of the Web is towards a growing demand for non-textual resources, this workload can represent a situation with a high amount of large multimedia objects. In our experiments we define six client profiles, where five require some form of content adaptation. The service time for adaptation operations is equal to 270 ms and 1720 ms for the median and 90-percentile, respectively. The workload and the client profiles are detailed in [2].

4.1. Cache hit rate and response time

In this section we analyze the performance of the two-level architecture. Tab. 1 reports the cache hit rates, that are divided in local, remote, exact, and useful. The last column of this table shows the global cache hit rate, which is the sum of the previous hit rates. We do not report the local hits of the two-level architecture, because the edge nodes act only as gateway servers and do not cache any resource. For the No_Coop architecture, there are not remote hits because the lookup among the peers is not activated.

From the last column of Tab. 1 we observe that the global hit rates differ significantly. As expected, the two-level architecture achieves a global hit rate considerably higher than that of the flat and No_Coop schemes. Its best cache hit rates are due to the hash function that maximizes the locality by avoiding intersections in the resource ID subspaces and consequent presence of duplicated resources in the servers. The same table indicates that cooperation can increase the cache hit rate: the flat architecture takes advantage of remote hits, thus increasing its global cache hit rate with respect to the No_Coop scheme.

We observe an important difference between the flat and two-level architectures. The percentages of exact and useful hits for the flat architecture are similar, while the exact hits of the two-level architecture are much higher than the useful hits. This result confirms that the choices of partitioning the resource ID space among the interior servers and keeping all the existing versions of a resource on the same interior nodes allows the two-level architecture to maximize locality and to achieve the highest rates for the most precious (exact) cache hits.

On the other hand, the flat architecture has 15% of local hits that can be immediately served, thus avoiding the twofold step of the two-level architecture. Hence, to evaluate the pros/cons of the three architectures it is not sufficient to analyze only the cache hit rates.

	Local		Remote		Global
Architecture	**exact**	**useful**	**exact**	**useful**	
No_Coop	8.3%	6.4%	n/a	n/a	**14.7%**
Flat	8.0%	7.0%	25.1%	26.8%	**66.9%**
Two-level	n/a	n/a	60.2%	21.0%	**81.2%**

Table 1. Cache hit rates.

We now pass to consider the response time. Fig. 3 shows the cumulative distribution of the response times for the three architectures.

From this figure we can conclude that the performance of the two-level scheme is clearly superior: about 80% of the requests are served in less than 1.2 s by the two-level architecture, while for the flat architecture this percentage is below 60%. This figure also shows that the two-level infrastructure is effective in reducing response time variance: for the flat architecture the lookup process can be very fast (requiring only a local cache lookup in the case of local hit), but it can also be very time consuming especially in the case of remote miss. As already pointed out in [4], a miss is detected only when the last miss message has been received, hence the system must wait for the slowest peer to respond or for a timeout to expire.

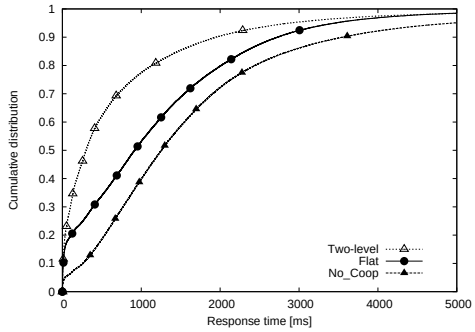

Figure 3. Cumulative distributions of response times.

We conclude that the higher cache hit rate (especially for the exact hits that avoid adaptation) of the two-level architecture has a positive impact on the response time. Moreover, the two-level architecture serves misses in a faster way with respect to the flat scheme that has to carry out an unsuccessful cooperative lookup before detecting a global miss.

4.2. Impact of the network costs

The most critical issue of the two-level architecture is that the edge nodes must contact the interior nodes for each client request, while the flat architecture needs a cooperative lookup only in the case of local miss. Therefore, it is important to evaluate under which network conditions the two-level architecture guarantees the best performance. To this purpose we analyze the sensitivity of the flat and two-level architectures to the network delays among the nodes of the intermediate infrastructures, and among the intermediary nodes and the origin servers. In this set of experiments we do not consider the No_Coop scheme that the previous section has demonstrated to be not competitive.

Since our experiments focus on the comparison of the intermediate architectures, the analysis of the last mile (that is, C-E links in Figs. 1 and 2) is out the scope of this paper, because is has the same impact on any organization of the intermediate infrastructure.

We consider two network scenarios (*intermediate* and *server* described in Tab. 2), where bandwidth limitation and

Connection	Link bandwidth	Link delay	
		intermediate	server
Client-to-Edge (C-E)	n/a	n/a	n/a
Edge-to-Edge (E-E) = Edge-to-Interior (E-I)	16 Mb/s	0-20 ms	50 ms
Edge-to-Origin Server (E-O) = Interior-to-Origin Server (I-O)	2 Mb/s	10 ms	20-100 ms

Table 2. Network scenarios.

delays have been emulated through a WAN emulator that we implemented as a patch to the operating system kernel on every node. Each connection between two nodes is assumed as composed of a number of links varying from 1 to 5. For each link, network delays are modeled taking into account the link bandwidth and the link delay. Bandwidth is reported for every connection in the second column of Tab. 2. However, it is important to note that the conclusions on performance are not dependent on the reported bandwidth values. A link delay modeled through a Pareto distribution occurs for every packet. Columns 3 and 4 of Tab. 2 show the mean link delay values for the two scenarios.

The *intermediate* scenario is used to study the performance sensitivity to network delays among the nodes of the intermediate infrastructure (E-E and E-I links), while the mean delays in E-O/I-O links are kept constant. The *server* scenario is used to study the sensitivity to the delay in the links connecting the intermediate infrastructure to the origin servers (E-O/I-O links), while the E-E and E-I delays are fixed to 10 ms.

As a final remark, we observe that the assumption of having equal delays on E-I and E-E links favors the flat architecture. Indeed, in a real scenario we expect E-I links (connecting edge nodes to the network core) to have a lower delay than E-E links (connecting nodes on the network edge).

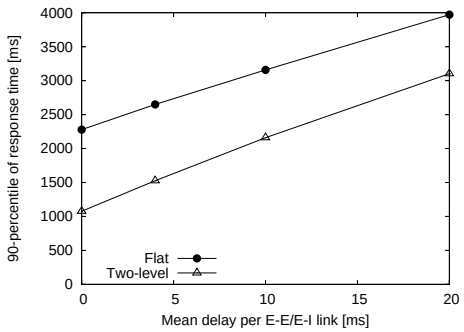

Figure 4. 90-percentile of response time for increasing network delays.

Intermediate scenario. The first set of experiments focuses on the sensitivity of the delays on the E-E and E-I links. The two-level architecture implies a twofold step for the lookup of the resources because edge nodes have always to contact the interior nodes for each request. Hence, we expect that higher delays in E-I links should have an impact on response time. On the other hand, in the flat architecture the edge nodes may serve directly client requests resulting in local exact and useful hits.

Fig. 4 shows the 90-percentile of the system response time as a function of increasing E-E and E-I delays. The two-level architecture gets a response time lower than that of the flat architecture for every value. However, as the network delay increases, performance gain of the two-level architecture over the flat architecture is reduced because the network delay contribution to the response time, which is common to both architectures, becomes more significant.

A disaggregated analysis of the contributions to the response time (not reported for space limits) shows that the local hits in the flat architecture are more than twice faster than the hits in the two-level architecture. However, the cost of the twofold step of the two-level scheme is highly compensated when the flat architecture must activate a cooperative lookup, which is both time and CPU consuming, especially for remote useful hits (and global misses).

Server scenario. In this set of experiments we focus on the impact of delays to the origin servers. Connections to the origin servers are used only in the case of global miss, hence the global cache hit rate plays a fundamental role here. Fig. 5 shows the 90-percentile of the response time as a function of delays in the link from the intermediate infrastructure to the origin servers. The two-level architecture provides best performance for every delay value. The motivation should be clear: the higher cache hit rate of the two-level architecture determines a lower number of fetch operations from the origin servers. As a consequence, the two-level architecture is less sensitive to E-O/I-O delays than the flat architecture.

5. Related work

In this section we review some recent research proposals regarding content adaptation when the latter is carried out by intermediary nodes enhanced with functions to adapt the content on-the-fly and to cache it. Although some of the latest studies on content distribution networks [9] focus on coordination among resource-intensive applications over geo-

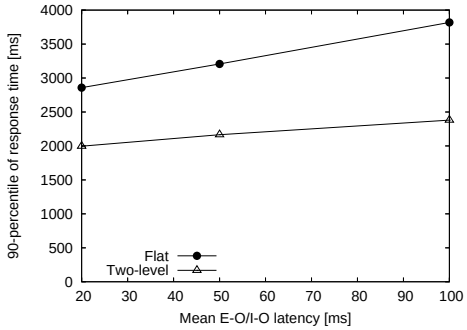

Figure 5. 90-percentile of response time for increasing delays to the origin servers.

graphically distributes systems, their results are not directly applicable to the case of content adaptation.

The main research efforts have been devoted to the investigation of solutions that are similar to the No_Coop architecture that is, a flat scheme in which the adaptation and caching functions are provided on stand-alone not cooperative servers. Some cooperative architecture have been recently proposed by the authors [2], while fully distributed P2P networks mainly focused for content personalization have been considered in [11].

Other research efforts aimed to handle the variations in client bandwidth and display capabilities [5, 6] without focusing on caching aspects, while considering object compression techniques [6].

A limited number of recent results have also combined both adaptation and caching to reduce the resource usage at the edge server [3, 12]. However, all of them are limited to the enhancement of a single server.

The main motivation that leaded us to study distributed architectures for intermediate services of caching and adaptation is the limited scalability of a single intermediary-based approach because of significant computational costs of adaptation operations. Fox *et al.* [5] have addressed this scalability issue through a cluster of locally distributed edge servers. This approach may solve the CPU-resource constraint, but it tends to move the system bottleneck from the node CPU to the Internet connection of the cluster. On the other hand, the proposed two-level infrastructure is designed to be distributed over a wide area Network, thus preventing network bottlenecks.

6. Conclusions

In this paper we propose a novel distributed architecture for efficient content adaptation and delivery of Web resources. The architecture is based on a two-level organization of the nodes where the edge nodes are simple gateways, and the interior nodes, located in the network core, execute the main services such as content adaptation, caching, and fetching. We have evaluated through real prototypes the performance of the two-level architecture against two flat architectures: in the former, all nodes provide the same functions and cooperate in content adaptation and delivery; in the latter, the nodes do not cooperate. The entire set of experiments, that have been reported only in part, allows us to conclude that the two-level architecture outperforms the other flat architectures for any considered network condition and cache size.

References

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

[2] C. Canali, V. Cardellini, M. Colajanni, R. Lancellotti, and P. S. Yu. Cooperative architectures and algorithms for discovery and transcoding of multi-version content. In *Proc. of 8th Int'l Workshop on Web Content Caching and Distribution*, Sept./Oct. 2003.

[3] C.-Y. Chang and M.-S. Chen. On exploring aggregate effect for effi cient cache replacement in transcoding proxies. *IEEE Trans. on Parallel and Distributed Systems*, 14:611–624, June 2003.

[4] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary cache: A scalable wide-area Web cache sharing protocol. *IEEE/ACM Trans. on Networking*, 8(3):281–293, June 2000.

[5] 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.

[6] R. Han, P. Bhagwat, R. LaMaire, T. Mummert, V. Perret, and J. Rubas. Dynamic adaptation in an image transcoding proxy for mobile Web browsing. *IEEE Personal Communications*, 5(6):8–17, Dec. 1998.

[7] 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.

[8] M. Rabinovich and O. Spatscheck. *Web Caching and Replication*. Addison Wesley, 2002.

[9] 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.

[10] K. Ross. Hash-routing for collections of shared Web caches. *IEEE Network*, 11(6):37–44, Nov./Dec. 1997.

[11] 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.

[12] 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.

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