

CONCA: An Architecture for Consistent Nomadic Content Access

Weisong Shi and Vijay Karamcheti
Department of Computer Science
Courant Institute of Mathematical Sciences
New York University
{*weisong, vijayk*}@cs.nyu.edu

Abstract

Future access to web-based content is likely to be dominated by two trends: (a) increasing amounts of dynamic, personalized content, and (b) a significant growth in “on-the-move” access using various mobile resource-constrained devices. These trends point to a situation where a user would have ubiquitous access to content, but require that content be efficiently delivered to the user irrespective of location, and in a form most suited to the user’s end device. Unfortunately, classical caching and transcoding solutions do not work well together, necessitating a new caching architecture built from the ground-up to handle problems caused by dynamic content, transcoded versions of objects, and the nomadic nature of users. This paper describes the goals and architecture of such a system: CONCA, an architecture for Consistent Nomadic Content Access.

1 Introduction

Future patterns for web content access are likely to be dominated by two growing trends. First, there has been a shift towards dynamically generated content, which is increasingly personalized to users as exemplified by services such as My Yahoo! [29] and MyCiti.com. Recent studies of proxy cache effectiveness have shown that dynamic content accounts for over 50% of all requests [40]. A related study of HTTP traces from AT&T WorldNet Internet service provider has revealed that 30% of all user requests carry cookies, header elements typically indicating that a request be personalized [2]. The second trend has been a significant increase in on-the-move access to web content using a variety of mobile resource-constrained devices. The growing popularity of web-enabled PDAs and cellular phones as well as the explosive growth in Internet startups trying to “mobilize” web content is a testament towards this trend. Together, the two trends raise the potential of ubiquitous (anytime, anywhere) access, but require some way of ensuring that users have access to content in the most efficient fashion,

irrespective of location and the end-device they might be currently using.

Well-understood solutions exist in isolation for improving web access latency and coping with device heterogeneity; however, these solutions do not work well with each other, particularly in light of the above trends. Caching and content distribution are popular solutions for the first problem and have the effect of moving content closer to the client. However, these typically do not work well with dynamically generated and personalized content. In the AT&T study mentioned earlier, caches can only achieve object hit ratios around 35% and byte hit ratios around 30% [2]. Transcoding is a popular solution to resolve server-client mismatches, but is unable to benefit from caching in general, with the result that clients see larger access times. Furthermore, neither technique provides particularly good solutions for coping with user mobility: caches are effective only when there is reuse, which may or may not apply in the user’s new location, particularly when the latter is geographically and culturally different from the origin location.

Although there does not appear to be any way to resolve the above situation, this outlook is more a reflection of current architectures for caching and transcoding as opposed to a fundamental limitation. In particular the following two observations can be made. First, despite the dynamically generated and personalized nature of web content, at the underlying level a relatively large amount of such content can in fact be shared. For example, even on personalized views of the Yahoo portal (my.yahoo.com), different users end up sharing the same news headlines and TV program guides. In fact, a recent study shows that approximately 75% of the bytes in dynamic responses from a set of popular web sites could in fact be reused from a previous retrieval of the page [39]. The second observation pertains to user access patterns. Most users exhibit a relatively static access pattern, often starting from a set of popular documents and following the links contained therein. Modern caching and content-distribution networks already exploit this observation to some extent, biasing resource management decisions in

favor of “hot” documents.

These two observations suggest a solution to our problem: caching architectures should be designed to reuse the shared portions of dynamic content, exploiting knowledge of user content access preferences to efficiently support transcoding and nomadic access (e.g., by prefetching). However, fully exploiting these observations requires fundamental changes not only in caching architectures but also in the nature of client requests and server responses. This paper presents a novel architecture for **CONCA** (Consistent Nomadic Content Access), which attempts to support, from the ground up, caching of dynamic personalized content for mobile users. Research on CONCA has just begun: this paper presents the underlying motivation and goals of the architecture and identifies issues that we intend to focus on as part of this project.

The rest of this paper is organized as follows. We identify the key enablers for CONCA in Section 2. Section 3 presents a high-level description of the CONCA architecture, and Section 4 describes a qualitative analysis of benefits resulting from CONCA-like architectures in the context of two personalized web portals. Section 5 discusses challenges and outstanding issues, Section 6 surveys related efforts, and we conclude in Section 7.

2 Enablers: Additional Server-Side and Client-Side Information

CONCA relies on two key enablers: additional server-side information about content structure, and additional client-side information about user content preferences. Fundamental to the CONCA system is the decoupling between content supplied by a server and that provided a client: CONCA automatically bridges the semantic gap between the two by relying upon these additional pieces of information. Exposing such information, which has traditionally been kept implicit at the server and client, enables the CONCA architecture to enhance the effectiveness of proxy caches.

2.1 Server Information: Document Templates

All content supplied by servers in the CONCA architecture is assumed to be associated with a “document template” which defines both the structure and form of the content. The former refers to layout, the latter to semantics, such as the object type, whether the object is sharable or personalized, its time-to-live (TTL) characteristics, information about server update protocols, etc. Figure 1 shows an example of the kind of information contained in a document template. The original personalized document consists of objects, some of which can be shared among other users (S1 through S5) and some

which are personalized to a specific user (P1 and P2). The document template identifies these different kinds of objects and tags them with additional information such as their time to live. In the example, the news headline and TV program objects are reusable across multiple users, while some of the other objects (stock quotes, regional weather) are less so. Thus, the document template enables efficient construction of a personalized view of the document for other users by fetching only those objects that are unique, reusing to the extent possible the shared objects already present in the cache (taking into consideration their validity).

We must note that document templates need to refer to finer-granularity objects than are currently identified using hyperlinks in top-level documents (e.g., the HTML HREF tag). The latter are already amenable to sharing using conventional caching architectures. More appropriate for our purpose are sub-document entities such as HTML frames, tables, paragraphs, etc. Also relevant are emerging standards such as the XSL Formatting Object specification [20], which could provide an appropriate language in which to express document templates.

2.2 Client Information: Personal Assistants

To support efficient transcoding from cached objects and support nomadic users, the CONCA architecture requires additional information from users. This information, captured in a “personal assistant” identifies the kinds of devices the user has access to, the templates for each of these devices, transcoding information about how the original content must be converted for display on a particular device, and additional information such as (consistency) linkages between transcoded content and original objects, etc. These user preferences allow the CONCA architecture to (a) maintain consistency between an original document and its transcoded versions; (b) decide how best to realize a user request for content access taking into consideration the end-device characteristics; and (c) concisely define the notion of a “per-user state” in the cache, allowing efficient reconstruction of the latter in a different cache to support nomadic users.

Continuing with our example in Figure 1, the information contained in the user’s personal assistant is depicted on the right. The information describes how the content must be transcoded and structured for each of the user’s end devices (in this case, a desktop, a laptop, and a PDA). Note that similar to the document template, user preferences also contain information about both layout and form. For example, the template associated with the PDA device indicates both that the content must consist of four objects (S2’ through S4’ and P1’), and makes explicit the linkage between the original objects (S2 through S4 and P1) as well as how the transcoded objects are gener-

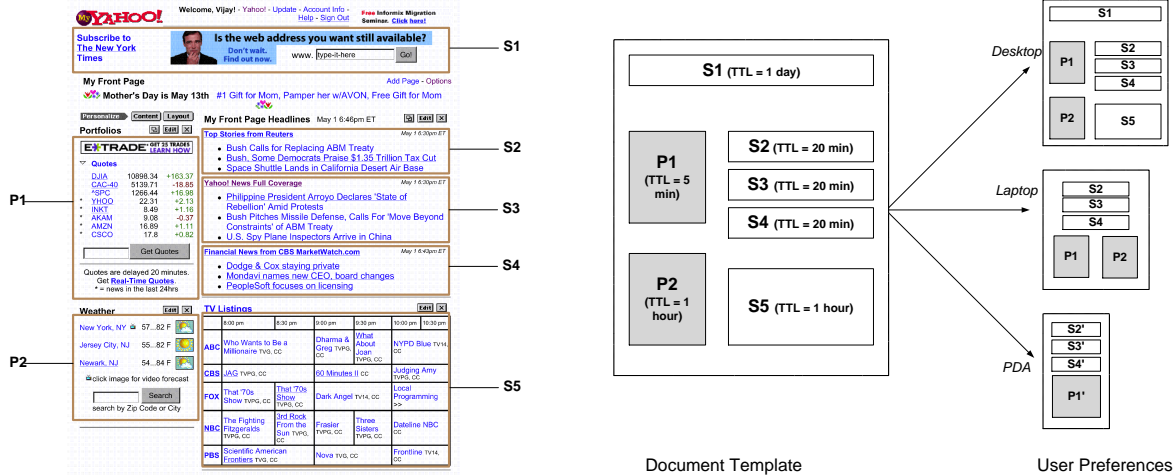


Figure 1. An example of a document template, differentiating between shared and personalized content.

ated (not shown).

We next describe how the CONCA architecture takes advantage of these two pieces of information.

3 CONCA Architecture

Figure 2 shows an overview of the CONCA architecture. CONCA uses a distributed client-side proxy cache architecture, similar to NSF’s IRcache project [14] and other recently proposed projects [43, 26]. Such architectures, which are complementary to server-side solutions such as reverse proxy caches and content delivery networks [23, 8], attempt to reduce network traffic associated with a miss in the local cache; ideally, such schemes would service the miss from a “near” proxy as opposed to requesting the object from the original server. In CONCA, as we shall see below, distributed proxy caches are the key to providing scalability.

Each CONCA node consists of cache storage and three modules—*remote control unit*, *local control unit* and *resource management unit*—which manage the node’s interactions with other cache nodes, with users, and internal storage policies respectively. The information described in Section 2 is leveraged by each CONCA node to provide two broad kinds of support: (a) consistent caching of dynamic personalized content, even when some of the content needs to be transcoded prior to delivery to the client; and (b) support for nomadic users, by enabling efficient recreation of per-user cache state.

3.1 Caching of Dynamic, Transcoded Content

CONCA cache nodes service client requests for dynamic content, transcoding the latter to suit the client’s end device. Their performance advantage stems from (a) reuse of shared portions of the dynamically generated content;

and (b) elimination of some of the transcoding operations. In more detail, the cache performs the following six operations in response to a client request: (1) looks up the personal assistant associated with the user, (2) according to the description in the personal assistant, figures out what “device” to transcode content for and how to perform the transcoding on the content, (3) examines what shared/personalized objects need to be fetched (based on what is already present in the cache), (4) fetches the necessary objects,¹(5) transcodes as required, while storing the intermediate objects, and (6) constructs the response appropriate for user’s device according to the related templates. These operations depend upon the cache having access to both the document templates and user personal assistants. The personal assistant associated with the user can be either preloaded (e.g., when the user registers with its *home cache*) or obtained on demand (e.g., from the home cache when the user moves to a new location). We defer discussion of the *home cache* notion to Section 3.2. Similarly, the cache can have the document templates preloaded or obtain them on demand from either the server itself or from a third party. Note that dynamically obtaining the template from the server needs modifications in the protocol between servers and caches.

Figure 3 shows the logical organization of cache storage, which consists of two separate portions: *shared* and *personalized*. The shared portion contains static content (e.g., objects associated with the URL `www.nyu.edu`) and the sharable subset of dynamic content (e.g., the objects S1 through S5 in our previous example associated with the URL `my.yahoo.com`). The personalized portion stores the per-user state, both for “home” users as well as other (nomadic) users who are temporarily us-

¹Note that the cache can (and should) prefetch contents that a user is likely to access, guided by the user’s personal assistant.

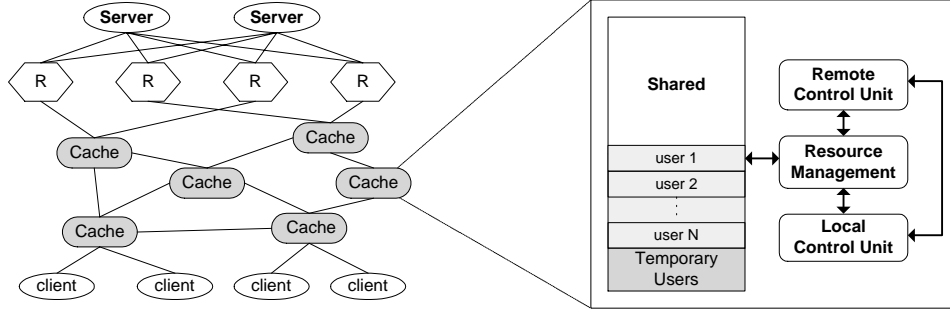


Figure 2. CONCA architecture: distributed proxy caches receive data from multiple server replicas (R).

ing this cache. This consists of (a) the user’s personal assistant, which contains information about the user’s profile, devices, and transcoding preferences, (b) downloaded personalized objects (e.g., P1 and P2 from our earlier example), and (c) (intermediate) transcoded versions of these objects. Note that although the figure shows transcoded versions of all objects as being stored in per-user storage, in general we may be able to share intermediate transcoded versions of the shared objects. In response to a user request, the cache first looks up the personal assistant associated with the user to determine the objects of interest, then acquires the missing objects, transcodes them as required, and finally delivers to the user a document assembled from the various pieces. Each of these operations—content fetching, transcoding, and assembly—can be done in an anticipatory fashion using the information in the document templates and the personal assistants, further reducing client-perceived latency.

We must note that the reuse of shared objects in CONCA can save more bandwidth than the *delta encoding* technique proposed in HTTP/1.1 [31]. For example, if a shared object is part of many pages at a site, delta encoding will require a separate retrieval for each changed page, while only one retrieval is required in CONCA.

The CONCA architecture relies on transcoding to cope with device heterogeneity. However, transcoding raises a consistency problem in having multiple representations of the same object. The CONCA architecture ensures consistency among the various representations by associating a TTL with each transcoded version (which depends upon its creation time and the TTL and creation time of its root object), and maintaining linkages between an object and its transcoded versions. While this scheme works, it has the shortcoming that the granularity of consistency is the entire object. Thus, a transcoded version of an object might get invalidated even when updates to the object do not affect the transcoded portion. To capture such finer-grained and object-specific relationships, we plan to leverage the notion of object views [28] developed by the authors in prior work.

Object views permit explicit specification of conflicts among multiple object representations, permitting efficient maintenance of consistency.

3.2 Support for Nomadic Users

CONCA cache nodes support nomadic users by exploiting the fact that the cache storage conveniently identifies per-user state. This allows the state to be efficiently recreated on another proxy node that the user is currently close to. More specifically, each user associates a fixed proxy cache as his *home cache* in CONCA, defined as the cache that maintains his per-user state persistently. The user has the freedom to any cache as his home cache, although for performance reasons one might choose the one closest to the user’s residence. All of the users that associate a particular cache as their home cache are collectively referred to as *home users*.

When a user travels away from his home cache, requests from client applications are routed to whichever cache is nearest to the user’s current location. We plan to leverage the Web Proxy Auto-Discovery (WPAD) protocol [7] recently reviewed by IETF to help nomadic users locate near proxy caches. This new cache contacts the user’s home cache to obtain information about the state associated with the user. It can then satisfy the user’s requests more efficiently by reusing any locally cached content and prefetching personalized content from the user’s home cache. Information about the home cache can be provided either explicitly, as part of the protocol employed when connecting to a new cache, or implicitly by looking up a directory.

Supporting nomadic users introduces two consistency problems: *inter-proxy cache consistency for shared objects* (a consequence of the distributed cache architecture), and *consistency of personalized objects*. Our initial solution to the first problem is to maintain a directory structure for each shared cache item using the *coarse bit-vector method* [10]: a bit represents a group of neighboring caches. When an invalidation message is received from the original server or higher-level caches, the invalidation message is sent out to each group of sharers.

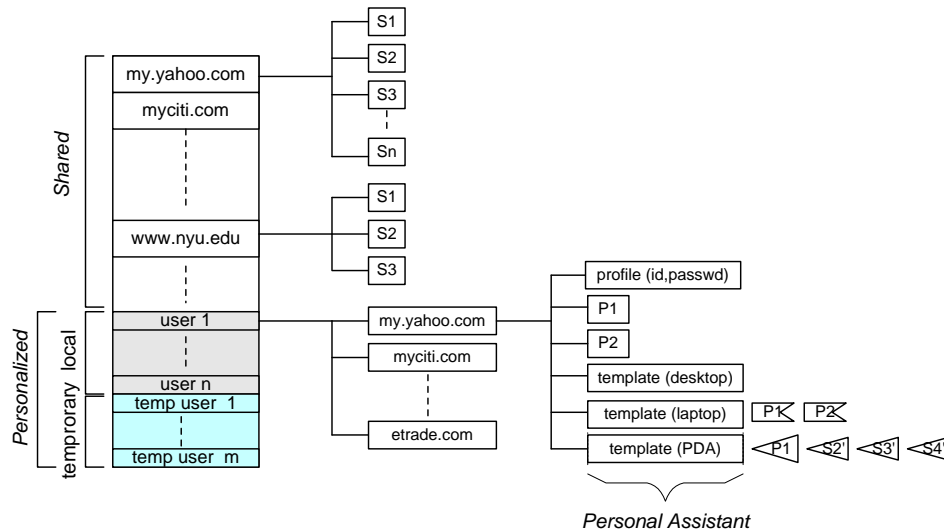


Figure 3. Detailed structure of a CONCA cache node, showing shared and personalized portions of the cache.

We intend employing a home-based eager update protocol to maintain consistency of personalized data between a user's temporary cache and his home cache. Eager update means that the latest version of per-user content stored in the remote proxy's temporary cache is sent back to the user's home cache either periodically or when the user explicitly disconnects from the cache. As such, a strong consistency model is provided for personal content. The underlying assumption with this scheme is that network transmission is faster than people moving, which is reasonable.

4 Case Studies

To assess the benefits likely from CONCA-like architectures when accessing dynamic and personalized content from web sites, we qualitatively examined two representative portal sites: NYUHome and My Yahoo!. The first is used by the relatively localized New York University community, while the second caters to geographically distributed clients who exhibit larger diversity. We are currently in the process of quantifying the qualitative observations discussed here.

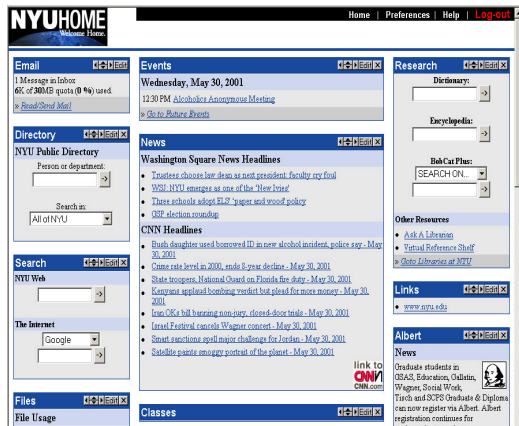
4.1 NYUHome Portal

NYUHome is a portal for information, collaboration, interaction and communication for all students, faculty and staff of New York University (NYU), and is being widely used by the NYU community. Using any web browser, NYUHome account holders are able to obtain news and stock information, access their e-mail, register for courses, participate in web forums, access class pages, research tools, and more. The NYUHome screen is customizable, and can be personalized by different

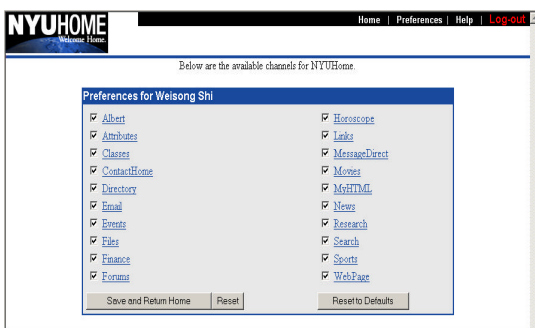
users to fit different needs and interests. Figure 4(a) shows a screen snapshot of the dynamically generated and personalized NYUHome page. Different resources, such as news, courses management, reference searches and/or a schedule of campus events, are provided through channels, and the user can choose which channels he wants to display on his screen. Currently NYUHome supports 20 channels as shown in Figure 4(b).

Among these 20 channels, 9 of the channels are for objects that are sharable by everyone that has selected a particular channel. These include event listings for the NYU campus, news headlines, research tools linking to the NYU library pages, movie listings, sports headlines, web search interface, and form-based interfaces to the NYU directory and an instant messaging system. Note that users can still choose to locate these channels at different positions within their page; however, exposing this layout information via server-supplied document templates would enable an intermediate CONCA-like cache architecture to construct responses to user requests by reusing cached data. Two (2) other channels are made up of a common sharable part and a personalized part. For example, the finance channel comprises a portion that shows headlines from financial newspapers and a second part that provides information about user-selected stocks. The document template would identify for this channel, the part that was shared and the part that was personalized. Two (2) more channels exhibit more diversity in their object contents but can still be thought of as being sharable in a large user population. For example, the horoscope channel can be shared among the subset of the user pool that shares the same zodiac sign.

The remaining 7 channels provided by NYUHome refer to truly personalized objects, such as a user's Email,



(a)



(b)

Figure 4. NYUHome portal: (a) a screen snapshot of a personalized page, (b) a listing of all 20 channels whose contents can be personalized by users.

enrolled classes, personal files, personal web page(s), interesting links and forums, messages, and a personal My-HTML channel. Although these personalized channels cannot be shared among different users, note that their identification as part of the document template and the user's personal assistant permits their prefetching.

The benefits of using a CONCA-like architecture in the NYUHome context are clear. In addition to the fact that there is significant reuse of traditionally uncacheable content, CONCA would also enable efficient consistency maintenance between the server and the cache. In contrast to treating the entire page as a single consistency unit, which has the consequence that the page must be updated every time any of its objects change, CONCA permits different objects to be updated according to their individual TTL settings. This has the potential of providing a big performance advantage over the traditional scheme because objects with longer TTLs (e.g., links to a user's courses) need not be refetched everytime an object with a shorter TTL (e.g., stock information) is updated.

4.2 My Yahoo! Portal

Yahoo! was one of the first sites on the Web to use personalization on a large scale, most notably with its My Yahoo! (my.yahoo.com) application introduced in July 1996. My Yahoo! is a customized personal copy of Yahoo!. Users can select from hundreds of modules, such as news, stock prices, weather, and entertainment, and place them on one or more Web pages. Figure 1(a) shows an example of a personalized My Yahoo! page of one of the authors.

Like the NYUHome portal, My Yahoo! users can benefit from module-level sharing because several user pages contain general modules sharable in their entirety. For example, every user (at least those within geographical proximity) see the same content for channels such as the "top health news", "new movie releases", etc. Unlike the small-scale NYUHome portal, the My Yahoo! portal sometimes customizes the context of these general modules based on a user's location. Such customized content still lends itself to caching as long as a significant fraction of users using the same proxy cache are subject to similar customization. As with the NYUHome portal, truly personalized My Yahoo! modules supporting personal information management such as Email, calendar, and weather information can benefit from prefetching.

However, an important difference between NYUHome and My Yahoo! is that the latter provides many more content modules to satisfy the diverse needs of a larger user base, each of which is associated with a relatively large number of alternatives. For example, the news headlines of My Yahoo! can be customized to user-selected news sources, such as Reuters and New York Times, and the channels in the TV listings can choose from different cable companies, etc. Although it is unlikely that a significant subset of the user population would have the same *module-level* personalization for such channels, there is a high likelihood of being able to reuse individual *items* (e.g., the New York Times headlines) among a moderately-large user base. Thus, unlike channel-level sharing in the NYUHome portal, the My Yahoo! pages benefit from item-level sharing, requiring that the document templates supplied by the server identify this hierarchical structure. Note that effective exploitation of such reuse requires the cache to store and manage individual items on the page. Whether or not the overheads of such fine-grained management outweigh potential benefits is one of the issues we intend to investigate in this project.

5 Challenges and Outstanding Issues

CONCA is a novel architecture which attempts to support, from the ground up, caching of dynamic personal-

ized content for nomadic users. Whether or not an architecture such as CONCA is successful depends on solutions to several concerns, specifically how to obtain document templates and personal assistant information and how to manage cache node resources so as to achieve an overall performance benefit. Our research is investigating these issues, discussed in additional detail below.

5.1 Obtaining Document Templates

As described in Section 2, document templates are essential to caching dynamic and personalized content by facilitating the reuse of shared objects. We anticipate that document templates will be obtained in one of two ways. The first, relies on servers making available information about document structure that they already possess. Such CONCA-friendly servers, for example along the lines of fragment-based publishing systems proposed in [5], may be willing to share information that they already maintain internally (at either the XML/XSL level or other similar forms) to ensure that they continue to exert control over the content seen by the user. Since the CONCA architecture is primarily concerned with presentation of the final content to the user, servers do not need to expose any application logic. The second scheme relies on a third party to infer the document template either via automatic analysis or through human assistance. Several services such as Avant Go, Spyglass, Yodlee, and EveryPath are already employing a similar strategy to “mobilize” legacy web content for various devices.

5.2 Capturing User Content Access Profiles In Personal Assistants

Logically, CONCA personal assistants capture the following three pieces of information: (1) the information about the home cache of this user, (2) information about the kinds of devices the user has access to, the templates for each of these devices, transcoding information about how the original content must be converted for display on a particular device, and (3) additional information such as (consistency) linkages between transcoded content and original objects, etc.

An important question here is what is the “right” level at which such information must be expressed? Should the personal assistants be mobile code that on behalf of the user explicitly handle sensing the user’s context, (pre)fetching, cache lookups, transcoding, and constructing the response content, or should they just be high-level specifications? The latter for instance could specify transcoded forms for various devices at the level of MIME types (e.g., `Text+Image` for a laptop versus `Text` for a PDA), relying upon a preexisting set of transcoding components (e.g. different ICAP services [19]) to convert one type into another. The cache

node takes responsibility for determining how best to obtain a required type trading off concerns of quality against resource usage. Dynamic creation of such transcoding paths has recently attracted a fair bit of attention, including in the context of the CANS system [16] developed by the authors.

5.3 Communication versus Computation Tradeoffs

Transcoding operations are often time consuming, therefore the cache node may sometimes have to decide between (re)transcoding an object locally and fetching the transcoded object from a neighbor cache, even when the original version is available locally. Similar issues crop up in trying to decide between fetching content from a cache node connected with a slow link that already has the content in transcoded form, versus fetching it from a cache node reachable via a faster link but where one has to spend time transcoding the content. More generally, it is possible to treat the fetch and transcode operations as components along the path, each with their cost and effect on response time or throughput. Therefore, the tradeoff between computation and communication becomes an optimization problem, with the objective being minimization of response time or maximization of throughput. This problem is very similar to our earlier work on automatic path creation and reconfiguration in the context of the CANS system [17], where the goal was to select and map appropriate components along network-aware access paths.

5.4 Resource Management Issues

Several resource management aspects of the cache node will end up determining overall performance benefits. These include questions such as how should the cache space be divided up between shared and personalized content, and between home and temporary content, is it possible to further reuse personalized objects by migrating some of the server-side processing to the caches, what are the best cache placement/replacement algorithms, and how to efficiently support multiple home caches.

6 Related Work and Discussion

Our work on CONCA builds on a large body of related work in the general area of web caching. The interested reader is referred elsewhere [1, 38] for good surveys of different technologies. Instead of describing each separately, we group related efforts into four broad categories: commercial, dynamic content generation and delivery, transcoding, and prefetching.

Commercial efforts Commercial caching and edge

server products, most notably IBM's WebSphere [9] and Akamai's Edgesuite [22] are beginning to incorporate functionality that can be viewed as necessary infrastructure for CONCA-like architectures. Specifically, both of these products support use of XML as a means to encode document semantics in a presentation-independent fashion, combined with tools that convert XML-based documents to formats such as HTML and WML to cope with device heterogeneity. More recently, these products have also been extended to enable execution of server application logic on edge servers within a content-distribution network. Doing so lowers client-perceived latency for dynamic and personalized content since user requests are now intercepted by edge servers, which dynamically assemble appropriate responses instead of being forwarded to origin servers.

CONCA builds on the infrastructure provided by such efforts but differs from them in focusing on orthogonal issues that are currently beyond their scope. First, unlike the server-oriented focus of the above solutions (e.g., permitting application logic to be shifted to edge servers), CONCA focuses on intermediary cache-level support. This focus is complementary to that adopted by content distribution networks and is worthwhile pursuing because it provides the advantage that cache policies can be better tailored to the supported user base. The encoding of user preferences for prefetching and document layout envisioned in CONCA personal assistants is an example of this. However, supporting caching of dynamic and personalized content in intermediate caches requires a well-defined interface for exchanging information between servers and caches that continues to maintain content rights with the server, which is something that the above efforts have not needed to look at. Second, CONCA emphasizes the notion of per-user cache state, using it as the basis for managing sharing and prefetching of personalized content as well as coping with nomadic users. Current commercial products provide little support for this category of users. Finally, CONCA makes explicit the linkages between multiple transcoded versions of the same logical object in a cache. This permits us to efficiently maintain consistency among these versions, and is something that as far as we know is not supported in either of the products mentioned above.

Dynamic content generation and delivery The CONCA notion of splitting up a dynamically generated and personalized document into sharable and personalized components to improve cache effectiveness builds upon previous work done both at the server-side [4, 5, 45] and at the cache [3, 11, 30, 39, 33] to enable efficient generation and delivery of dynamic content.

Representative examples of server-side mechanisms include data update propagation (DUP) [4], fragment-

based page generation [5], and the class-based page classification scheme used in the Cachuma page caching system [45]. The common thread uniting each of these mechanisms is that they all maintain dependence information between dynamically generated pages and the underlying data (typically fields in a database). This explicit dependence information enables servers to cache or incrementally update previously generated dynamic pages, reducing the cost of generating dynamic content, maintaining its consistency (at caches). CONCA extends such server-side work by proposing similar functionality in cache nodes (away from the server) where dependence information not only reduces the cost of maintaining consistency for dynamic content [42] but also facilitates reuse of fragment objects among multiple users to reduce access costs of personalized content as well.

Researchers have also suggested extending cache-side functionality to better handle dynamic content generation and caching. Active Cache proposed by Cao et al. [3] and more recently the Gemini system proposed by Myers et al. [33] require web servers to provide both content and specialized code (applets in Active Cache, Java classes in Gemini) that can be executed by proxy caches to produce a new version of a cached document. This mechanism in effect requires content providers to relinquish control over part of the application logic, leading one to question whether such schemes will see widespread use. CONCA is closer in spirit to two other efforts, which retain control of application logic with the content provider instead relying upon additional information about the document structure. Douglis et al. in their HPP work [11] propose separating a document into a cacheable static portion and a dynamic portion that must be obtained on every access. Mikhailov and Wills recently proposed content assembly techniques [30], which enable construction of pages from individual components cached in proxy caches closer to users with future access to similar pages being able to share components from previous ones. In both cases, the primary goal is to reduce the amount of dynamic content that must be retrieved from the origin server. CONCA takes a more general view of such mechanisms, relying upon richer document templates and user-supplied personal assistants to additionally support personalization, transcoded content delivery for multiple devices, and nomadic access.

Transcoding There is a large amount of prior work on transcoding architectures [15, 18, 32, 35, 41], infrastructures for their effective deployment [16, 24, 34, 36, 37], and lightweight protocols, such as IETF's ICAP [19] and W3C consortium's SOAP [6]. CONCA intends to leverage several of the underlying ideas in these works. However, a notable difference in CONCA is the attempt to reuse intermediate objects along the transcoding path

and integrate them into caching architectures. The latter is also what distinguishes our work from previous work on consistency protocols for web caches [21, 27, 42], and on object location and request forwarding services in the context of distributed web caching architectures [12, 14, 26, 43, 44]. CONCA extends current support for transcoded versions in commercial server platforms such as IBM WebSphere by allowing caching and reuse of transcoded versions as well as providing more flexible transcoding paths (e.g., those that span multiple cache nodes).

Prefetching The CONCA architecture also makes extensive use of prefetching, which has been demonstrated to be an efficient mechanism for reducing web access latency. As noted by several researchers [13], there are three distinct prefetching scenarios: prefetching between clients and servers, between clients and proxies, and between proxies and servers [25]. Prefetching in CONCA can also happen between two caches (proxies) and differs from its traditional use in being driven by the user context and information about the latter's content access preferences as captured in the personal assistant.

7 Conclusions and Future Work

This paper has presented a distributed proxy cache architecture called CONCA that provides support for consistent caching of transcoded content to address problems arising from two growing trends: (a) dynamic, personalized nature of content; and (b) web access "on-the-move" using mobile resource-constrained devices. CONCA reflects our belief that caching architectures need to be closely integrated with additional information from servers and users to effectively support future requirements.

To better quantify the benefits resulting from reusing sharable objects in dynamically generated and personalized content, we intend to instrument active web sites to collect usage statistics. This information will help drive the development of a prototype of the CONCA system.

Acknowledgments

We thank the anonymous reviewers for helping us improve this paper. This research was sponsored by DARPA agreements F30602-99-1-0157 and N66001-00-1-8920; by NSF grants CAREER:CCR-9876128 and CCR-9988176; and Microsoft. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of DARPA, Rome

Labs, SPAWAR SYSCEN, or the U.S. Government.

References

- [1] G. Barish and K. Obraczka. World wide web caching: Trends and techniques. *IEEE Communications Magazine Internet Technology Series*, May 2000.
- [2] R. Caceres, F. Douglis, A. Feldmann, G. Glass, and M. Rabinovich. Web proxy caching: The devil is in the details. In *Proceedings of ACM SIGMETRICS Internet Server Performance Workshop*, June 1998.
- [3] P. Cao, J. Zhang, and K. Beach. Active cache: Caching dynamic contents on the web. In *Proc. of IFIP Int'l Conf. Dist. Sys. Platforms and Open Dist. Processing*, 1998.
- [4] J. Challenger, A. Iyengar, and P. Dantzig. A scalable system for consistently caching dynamic web data. In *Proceedings of Infocom'99*, March 1999.
- [5] J. Challenger, A. Iyengar, K. Witting, C. Ferstat, and P. Reed. A publishing system for efficiently creating dynamic web content. In *Proceedings of INFOCOM'00*, March 2000.
- [6] W3C Consortium. Simple object access protocol (soap) 1.1. In <http://www.w3.org/TR/SOAP/>, 2000.
- [7] I. Cooper, P. Gauthier, J. Cohen, M. Dunsmuir, and C. Perkins. The web proxy auto-discovery protocol. In <http://www.wrec.org/Drafts/draft-cooper-webi-wpad-00.txt>, November 2000.
- [8] Digital Island Corp. <http://www.digitalisland.com/>.
- [9] IBM Corp. Websphere platform. In <http://www.ibm.com/websphere>.
- [10] D. E. Culler and J. P. Singh. *Parallel Computer Architecture: A Hardware/Software Approach*, chapter 12. Morgan Kaufmann, Inc., 1998.
- [11] F. Douglis, A. Hario, and M. Rabinovich. HPP:HTML macro-pre-processing to support dynamic document caching. In *Proceedings of USITS'97*, 1997.
- [12] L. Fan, P. Cao, J. Almeida, and A. Border. Summary cache: A scalable wide-area web cache sharing protocol. In *Proceedings of ACM SIGCOMM'98*, 1998.
- [13] L. Fan, P. Cao, W. Lin, and Q. Jacobson. Web prefetching between low-bandwidth clients and proxies: Potential and performance. In *Proceedings of ACM SIGMETRICS'99*, 1999.
- [14] A Distributed Testbed for National Information Provisioning. <http://www.ircache.net/cache/>.
- [15] A. Fox, S. Gribble, Y. Chawathe, and E. A. Brewer. Adapting to Network and Client Variation Using Infrastructural Proxies: Lessons and Prespectives. *IEEE Personal Communication*, August 1998.
- [16] X. Fu, W. Shi, A. Akkerman, and V. Karamcheti. CANS:Composable, Adaptive Network Services Infrastructure. In *Proc. of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS)*, March 2001.
- [17] X. Fu, W. Shi, and V. Karamcheti. Automatic deployment of transcoding components for ubiquitous, network-aware access to internet services. Technical Report TR2001-814, Computer Science Department, New York University, March 2001.
- [18] S. D. Gribble and et al. The Ninja Architecture for Robust Internet-Scale Systems and Services. *Special Issue of IEEE Computer Networks on Pervasive Computing*, 2000.
- [19] ICAP Protocol Group. ICAP: the internet content adaptation protocol. In <http://www.i-cap.org/icap/media/draft-elsson-opes-icap-01.txt>, February 2001.

- [20] W3C XSL Working Group. <http://www.w3.org/style/xsl/>.
- [21] J. Gwertzman and M. Seltzer. World-wide web cache consistency. In *Proceedings of the 1996 USENIX Technical Conference*, January 1996.
- [22] Akamai Technologies Inc. Edgesuite services. In http://www.akamai.com/html/en/sv/edgesuite_over.html.
- [23] Akamai Technologies Inc. <http://www.akamai.com/>.
- [24] E. Kiciman and A. Fox. Using Dynamic Mediation to Intergrate COTS Entities in a Ubiquitous Computing Environment. In *Proc. of the 2nd Handheld and Ubiquitous Computing Conference (HUC'00)*, March 2000.
- [25] T. M. Kroegeer, D. E. Long, and J. C. Mogul. Exploring the bounds of web latency reduction from caching and prefetching. In *Proceedings of the USITS'97*, 1997.
- [26] J. Kubiawicz and etc. OceanStore: An architecture for global-scale persistent storage. In *Proc. of the ASPLOS'00*, November 2000.
- [27] D. Li and D. R. Cheriton. Scalable web caching of frequently updated objects using reliable multicast. In *Proceedings of the USITS'99*, October 1999.
- [28] I. Lipkind, I. Pechtchanski, and V. Karamcheti. Object views: Language support for intelligent object caching in parallel and distributed computations. In *Proc. of OOPSLA'99*, November 1999.
- [29] U. Manber, A. Patel, and J. Robison. Experience with personalization on Yahoo! *Communications of ACM*, 43(8):35–39, August 2000.
- [30] M. Mikhailov and C. E. Wills. Change and relationship-driven content caching, distribution and assembly. Technical Report WPI-CS-TR-01-03, Computer Science Department, WPI, March 2001.
- [31] J. C. Mogul, F. Douglis, a. Feldmann, and B. Krishnamurthy. Potential Benefits of Delta-Encoding and Data Compression for HTTP. In *Proc. of the 13rd ACM SIGCOMM'97*, September 1997.
- [32] R. Mohan, J. R. Simth, and C.S. Li. Adapting Multimedia Internet Content for Universal Access. *IEEE Transactions on Multimedia*, 1(1):104–114, March 1999.
- [33] A. Myers, J. Chuang, U. Hengartner, Y. Xie, W. Zhang, and H. Zhang. A secure and publisher-centric web caching infrastructure. In *Proc. of the IEEE INFOCOM'01*, April 2001.
- [34] A. Nakao, L. Peterson, and A. Bavier. Constructing End-to-End Paths for Playing Media Objects. In *Proc. of the OpenArch'2001*, March 2001.
- [35] Brian D. Noble. *Mobile Data Access*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1998.
- [36] B. Raman, R.H. Katz, and A. d. Joseph. Universal Inbox: Providing Extensible Personal Mobility and Service Mobility in an Integrated Communication Network. In *Proc. of the Workshop on Mobile Computing Systems and Applications (WMSCA'00)*, December 2000.
- [37] P. Reiher, R. Guy, M. Yavis, and A. Rudenko. Automated Planning for Open Architectures. In *Proc. of OpenArch'2000*, March 2000.
- [38] J. Wang. A survey of web caching schemes for the internet. *ACM Computer Communication Review (CCR)*, 29(5), October 1999.
- [39] C. E. Wills and M. Mikhailov. Studying the impact of more complete server information on web caching. In *Proc. of the 5th International Workshop on Web Caching and Content Distribution (WCW'00)*, 2000.
- [40] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy. On the scale and performance of cooperative web proxy caching. In *Proc. of 17th ACM Symposium on Operating Systems Principles (SOSP)*, 1999.
- [41] M. Yavis, A. Wang, A. Rudenko, P. Reiher, and G. J. Popek. Conductor: Distributed Adaptation for complex Networks. In *Proc. of the Seventh Workshop on Hot Topics in Operating Systems*, March 1999.
- [42] J. Yin, L. Alvisi, M. Dahlin, C. Lin, and A. Iyengar. Engineering server driven consistency for large scale dynamic web services. In *Proceedings of the WWW10*, 2001.
- [43] L. Zhang, S. Michel, K. Nguyen, A. Rosenstein, S. Floyd, and V. Jacobson. Adaptive web caching: Towards a new global caching architecture. In *Proc. of the 3rd International WWW Caching Workshop*, June 1998.
- [44] B. Zhao, J. Kubiawicz, and A. Joseph. Tapestry: an infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, Computer Science Division, UC Berkeley, April 2001.
- [45] H. Zhu and T. Yang. Class-based cache management for dynamic web content. In *Proceedings of INFOCOM'01*, April 2001.