



PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 24/05

Proxy-based Adaptation for Mobile Computing

Markus Endler
Hana Rubinsztein
Ricardo C. A. da Rocha
Vagner Sacramento

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900

RIO DE JANEIRO - BRASIL

Proxy-based Adaptation for Mobile Computing*

Markus Endler, Hana Rubinsztein, Ricardo C. A. da Rocha, Vagner Sacramento

{endler, hana, rcarocha,vagner}@inf.puc-rio.br

Abstract. In mobile computing and wireless communication, proxies are mainly used to overcome the three major problems of these networks: throughput and latency differences between the wired and the wireless links, host mobility, and limited resources of the mobile hosts (MH). This report aims to present a general classification of proxy-based approaches, describe the most frequent functions assigned to proxies, discuss some of the underlying techniques used for implementing these functions, and present concrete examples of the most successful and well-known systems.

Keywords: Adaptation, Proxy, Mobile Computing.

Resumo. Em ambientes móveis, proxies são principalmente usados para superar problemas nessas redes, tais como menor vazão, mobilidade e recursos limitados de dispositivos. Este trabalho visa apresentar uma classificação geral de abordagens baseadas em proxy para adaptação, descrever as funções mais usuais atribuídas a proxies, discutir algumas das técnicas usadas para implementar estas funções, e apresentar exemplos concretos de sistemas bem conhecidos.

Palavras-chave: Adaptação, Proxy, Computação Móvel.

* This work has been sponsored by the Ministério de Ciência e Tecnologia da Presidência da República Federativa do Brasil

In charge for publications:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3114-1516 Fax: +55 21 3114-1530
E-mail: bib-di@inf.puc-rio.br
Web site: <http://bib-di.inf.puc-rio.br/techreports/>

1 Introduction

The use of proxies is commonplace in today's networks, where they are used for a huge variety of network services. A proxy is an intermediary placed in the path between a server and its clients. Proxies are used for saving network bandwidth, reducing access latency and coping with network and device heterogeneity.

In the specific case of mobile computing and wireless communication, proxies are mainly used to overcome the three major problems of these networks: throughput and latency differences between the wired and the wireless links, host mobility, and limited resources of the mobile hosts (MH). Although proxies may be used also for implementing specific services in *ad hoc* mobile networks, usually they are used in infra-structured mobile networks, since their functions commonly place high demands on both processing and memory. Thus, in this chapter we will mainly discuss proxy-based architectures for infra-structured mobile networks.

In most cases, proxies act as protocol translators, caches and content adapters for clients with network or device constraints and are placed on, or close to, the border between the wired and the wireless networks, such as at the wireless *Access Points* (AP) (also called *Base Stations* or *Mobility Support Stations*). Besides these canonical functions, however, proxies can perform a wide range of other complex tasks on behalf of the mobile clients, such as handover, session or consistency management, personalization, authentication, checkpointing, service/resource discovery, and others.

The major advantages of using a proxy-based architecture for serving mobile clients, when compared to an end-to-end approach, are the following: (a) all mobility- and wireless-dependent transformations (translation, transcoding) can be assigned to the proxy and need not be handled by the servers, allowing legacy services to be directly used for mobile access; (b) all processing required for protocol and content transformations is distributed to other nodes where they are required, avoiding an overload at the servers; (c) placing proxies at (or close to) a node with the wireless interface enables more agile and accurate monitoring of the wireless link quality, detection of MH disconnections, as well as better selection of the required adaptation; and finally (d) transformations at any communication layer can be implemented, and are more easily adapted/customized according to the specific capabilities of the wireless links.

As expected, there is a huge amount of work on proxy-based middleware for mobile and wireless computing, each solving the problems specific to some sort of service or application, such as Web access, multimedia streaming, database access, etc. Many authors use the terms *gateway*, *intermediary* or *agent* instead of *proxy*, and although there might be some subtle differences in their meanings, we will use these terms interchangeably and use the general definition of a proxy as being *an entity that intercepts communication or performs some service on behalf of some mobile client*.

Road map

In the remainder of this chapter, we first describe our general taxonomy of proxy-based architectures. Then, in section 4 we explain the main categories of tasks assigned to proxies, and give several examples of research and commercial systems that focus on these tasks. Section 5 then presents some tools and frameworks that have been created to support the development, customization and easy deployment of Proxies. Finally, in section 6 we draw our conclusions and point to the major trends in this subject.

2 Classifying Proxy-Based Approaches

Since proxies are primarily used to bridge and smooth the differences between networks and devices, and to perform application-specific adaptations, their functions are designed according to:

- the different characteristics of the wired and wireless networks which are to be bridged, such as throughput, latency, reliability, probability of disconnection, etc.
- the specific characteristics of the mobile host, such as: display size, user input/output mechanisms, processing capacity, size of RAM and persistent memory, limited energy supply, etc.
- the application type and its specific requirements, such as fast response time, low network latency, reliable communication, mobility or disconnection transparency, cache coherence, etc.

These aspects give an idea of the wide range of adaptation and management functions that can possibly be assigned to proxies. They may handle communication protocol issues, data transmission and encoding, device-specific customizations, handover and mobility management, security and authentication, recovery from disconnection, etc.

In spite of the huge diversity of proxy-centered architectures and proposals we have identified two orthogonal forms of classifying and comparing all proxy-based approaches. The first dimension takes into account some general characteristics of the proxy-based architecture, while the second dimension focuses on the tasks, i.e. functionalities, assigned to the proxies. These two classifications will be further detailed in sections 3 and 4, respectively.

Obviously, there are also other possible criteria for classifying proxy-based approaches. In particular, Dikaiakos [18] has written a very interesting survey about proxy-based infrastructures specifically for the Web. He proposes a classification of proxy approaches in three dimensions: *system architecture*, *functionality* and *interactions*. Regarding system architecture, he distinguishes between centralized and distributed architectures, options for proxy placement, and proxy configurability/programmability. Concerning functionality, he proposes six broad categories, which are consistent with our task categorization. Finally, with interactions the author considers whether the proxy supports synchronous or asynchronous communication. In addition, the article also compares eight proxy-based architectures and frameworks for the Web in deep detail. Hence, we recommend it as complementary reading to the interested reader.

3 Architecture-based Classification

In this section we discuss a classification of proxy-based approaches which emphasizes general features of the software architecture, and which is largely independent of the specific task assigned to the proxies. In particular, we have found that proxy-based approaches (and architectures) can be classified according to aspects such as Level, Placement, Single-/Multi-protocol, and Communication and Extensibility, which will be explained and discussed in the following.

3.1 Level

Since proxies may be used for handling adaptation/customization at various software levels, we believe that this is a suitable classification criterion. In our view, proxies can be used at three generic levels:

Communication-level At this level, proxies are in charge of handling all sorts of issues related to the communication protocols and abstractions. The main goal is to make device mobility and use of wireless links transparent to the higher software layers. Typical adaptations at this level are wired-wireless protocol translation or optimization, buffering, handover management, etc. Examples are all the flavors of TCP proposals for wireless networks [20] and Wireless CORBA [7].

Middleware-level At the middleware level, proxies perform general tasks neither tailored to a specific type of application, nor related to a specific communication protocol. Examples are some forms of content adaptation [24, 43], consistency management of cached data [31, 3], service or resource discovery [59, 13], security and authentication, and others.

Application-level Some proxy-based architectures are focused on a specific type of application such as Web-browsing (e.g. [27],[35],[6], etc.), database access [4], P2P data sharing [58], and others. In this case, proxies execute tasks tailored to specific requirements and functions of an application class. For example, when comparing caching in Web and database applications, the former handles heterogeneous objects and essentially aims at reducing response time, while the latter usually handles homogeneous data but requires management of cache consistency.

3.2 Placement and Distribution

Concerning the placement of proxies, we adopt the well-known classification suggested by Pitoura and Samaras [48] for proxy-based architectures, which defines the following main structures: a proxy executing only at a stationary node of the network (server-side); a proxy only at the mobile node (client-side); a pair of proxies, one executing at a stationary host and the other at the mobile host (also called the Interceptor Model); and a proxy that can move between a stationary node and the mobile device (migratory proxy or agent). While most systems use either a server-side proxy or a proxy pair, there are also examples of pure client-side proxies, such as in CODA [53]. As has been discussed elsewhere [48], server-side proxies are suitable for any kind of device, while client-side proxies normally require devices with more computing resources (a.k.a. thick clients). Migratory proxies have been suggested and implemented by several research groups as a means of transferring computing tasks from the MH to the network and “following” the MH while it moves between networks [56].

Another aspect is the distribution of the proxy-specific adaptation and management functionality in the architecture. It may be *centralized*, when all functionality is bundled into each proxy [27, 31]. Or it can be *decentralized*, when the system consists of several cooperating proxies, where each is responsible for some subset of the functions [43, 5].

3.3 Single-/Multi-protocol

Proxy architectures fall into two groups with respect to the number of communication protocols they support. Most systems handle a single protocol, such as TCP or HTTP, and support specific adaptations of these protocols aiming to bridge the wired-wireless gap. However, there are also other proxy-based architectures which adopt a multi-protocol approach, in which the proxy supports wireline-wireless translation using several protocols (e.g. UDP, SMTP, SMS, WSP) and is able to dynamically switch between these protocols for delivering the data to the user independently of which wireless/cellular network the user is currently connected with. Examples of the latter group are iMobileEE [14], TACC [8] and eRACE [16].

3.4 Communication

This aspect characterizes proxy-based architectures with respect to the way a proxy communicates with the client, the server and other proxies.

Essentially, a proxy can communicate with both endpoints, the server and the client, in two modes: in the synchronous mode, the proxy performs the adaptation task and replies to the client in response to an explicit client request. In the asynchronous mode, the proxy does long-term work on behalf of the user (e.g. based on his/her preferences), and sends asynchronous notifications to the client. This asynchronous mode is common when proxies play the role of user agents, searching, collecting and aggregating information on behalf of a user. Examples of architectures supporting both communication modes are WAP[22], WBI [5] and MoCA's ProxyFramework [52].

Some architectures also support communication among proxies, usually for the purpose of session and handover management, checkpointing, multicasting, and others. In this respect, communication can be direct or indirect. In the first mode, a proxy knows – perhaps through its client – which other proxy it needs to interact with [44, 12]. In the second mode, the server (or another proxy) serves as a router of the messages exchanged among the peer proxies.

3.5 Extensibility/Programability

Proxy extensibility, i.e. the possibility to adapt and customize its functions, is also an important criterion to differentiate architectures. In most systems, the proxy has pre-defined adaptive behavior, usually determined by the current state of the execution environment. As a first step towards extensibility, some approaches provide a generic framework in which proxies can be easily tailored to the specific needs of an application or middleware at deployment time, such as in MoCA's ProxyFramework [52]. Yet another group of proxy infra-structures further support the dynamic loading of filters or new modules implementing specific functionality, such as presented in [63].

4 Common Proxy Tasks

In this section we present the other form of classifying proxy-based approaches, which is by the main task, or function, executed by the proxies. One should remark that this classification does not render disjoint sets of systems in independent categories. This is because in most systems proxies handle tasks which pertain to several such categories. But independently of concrete system implementations, several of the task categories discussed in this section are in fact somewhat intertwined. For example, communication protocols play a central role in proxy-centered adaptations, and therefore most of tasks can also be regarded as communication protocol issues. Moreover, the list of tasks discussed in the following is unavoidably incomplete since there certainly are several other roles that can be assigned to proxies. Nevertheless, we believe to have selected the set of most common proxy tasks mentioned in literature.

4.1 Protocol Translation and Optimization

Since most conventional communication protocols for wired networks are usually not suited for wireless links because of the higher error rates, smaller throughput, higher cost and latency, mutual interference, intermittent connectivity, etc., one of the most common tasks of proxies is to deal with protocol translation, as well as optimizations of wireline protocols for wireless links.

Wired-wireless protocol translation is required in many layers of the protocol stack, but in this section we will focus on protocol issues of the transport layer and above, and lower-level transcodings will be considered “below the middleware level”.

In addition to the plain translation between protocol formats (i.e. header transcoding, data alignment and encoding), proxies may also have to deal with an array of other communication-specific issues such as flow control, error detection and recovery, medium multiplexing, and others, which essentially aim at optimizing data transfer over the wireless link and smoothing the wired-wireless gap. This is particularly true for connection-oriented protocols, such as TCP, whose mechanism for flow control does not react properly to disconnections, burst packet losses or fluctuations in round-trip delay. This has motivated the development of several so-called *TCP Split Connection Protocols* [20] (e.g. MTCP, I-TCP, M-TCP, SRP, etc.), where a proxy performs the mapping between the conventional TCP and an optimized transport protocol for the wireless link. Another example is the *Wireless-Profiled TCP* [45], adopted in the WAP 2.0 standard by the name *WTCP*[23], and used in i-Mode [19]. WTCP was developed for wireless MANs and WANs, and essentially uses the ratio of inter-packet separation as the primary metric for rate control, rather than packet loss and timeouts.

There are many other examples where proxies are used for protocol translation also at the session or application layers. For example, the WAP Gateway is responsible for converting between wire-line session, presentation, and application level protocols and the corresponding protocols of the WAP Protocol Stack [22].

A related task commonly assigned to proxies is that of optimizing data transfer of a conventional protocol over the wireless link. Protocol optimization essentially has two goals: to achieve higher bandwidth utilization, and to provide smaller round-trip delay. The usual optimization techniques include caching of data, connection multiplexing, header and payload compression, adaptive flow control, and data volume reduction.

HTTP and TCP are probably the most often cited protocols that have been optimized for wireless networks. Most optimizations done in the *TCP Split Connection* approach are based on the following general principles: using separate error and flow control on each side of the connection (wireless/wire-line); performing faster recovery of wireless errors due to shorter Round-Trip Time (RTT), hiding transmission errors from the sender, and generating selective/spontaneous TCP acks to avoid window resizing.

Concerning HTTP, the main problems with regard to communication over a wireless link are the following: human-readable and verbose headers; transfer of data objects without compression; huge RTT incurred by the use of a connection-oriented transport protocol and frequent DNS lookups; separate HTTP request for each in-line image, such as buttons, icons, bullet marks, etc. One of the earliest work attempting to optimize HTTP over wireless links was Mowgli [35], which employed an HTTP proxy pair using asynchronous messages over long-lived transport-level connections with header and payload compression. In addition, while transferring an HTTP page to the client-side proxy, the server-side proxy would in parallel retrieve the page’s in-line images from the server, even before explicitly requested by the client. IBM’s WebExpress [27] used a similar architecture, where a proxy pair (client-server) was used to optimize HTTP in the following directions: caching, differentiating (i.e. computation and transfer of the difference of an HTTP result with respect to a cached *base* Web object), HTTP header reduction, and multiplexing of several HTTP connections over a single TCP connection reducing the overall RTT. Rodriguez *et al* [51] has proposed a proxy-based solution where the RTT due to DNS lookup was also significantly reduced. Most current services and systems for mobile Web access use proxies and perform some similar HTTP optimizations [17].

Obviously, there are many other communication optimization techniques, such as those related to multimedia transmission and presentation. For multimedia content several coding techniques, scalable and layered coding, have been developed to deal with heterogeneous and variable network conditions, such as done in Mobiware [43] and RAPIDware [38, 37].

4.2 Content Adaptation

While protocol translation deals with protocol-specific adaptations/optimizations, content adaptation is largely protocol-independent and aims at transforming the payload for optimized transmission and presentation at the mobile device. The specific kind of adaptation used is mostly determined by the application requirements which may consider the following issues: quality of the wireless link (broadband, cellular) and the device's characteristics, such as its computational power (CPU, memory), output capabilities (screen size, gray-scale screens) and supported protocols (e.g. HTML, WML).

There is a wide range of proposals for content adaptation for different kinds of data, which include techniques such as data distillation or refinement, summarization, intelligent filtering and transcoding. Although there seem to be no unique and widely accepted definitions of these terms, in the following we will use the most common definitions found in the literature. Since the term *transcoding* is often used to denote any of the previous types of adaptation, we will also use it to discuss general techniques and present architectures supporting a larger spectrum of content adaptations.

Distillation and Refinement

Distillation is a highly lossy, real-time, data-specific compression technique that attempts to eliminate redundant or unnecessary information while preserving most of the *semantic content* of the data. Distillation is thus a general term for several forms of data compression, which may or may not be based on coding standards and representations. For example, JPEG is a lossy compression method where compression rates can be controlled according to desired image quality, and GIF has a fixed compression rate achieved by reducing the color palette available for display.

An example of non-coding based distillation could be a transformation where images are scaled down on each dimension to reduce their total size, thereby also reducing its binary representation. Yet another example of distillation is the reduction of the color depth or the color-map size. The resulting representation, though poorer in color and resolution than the original, is nonetheless still recognizable and therefore useful to the user.

Alternatively, the user may want to see the high-precision content of some part of the original data, for example, by zooming in on a section of a graphic or image, or by rendering a particular PostScript page with figures, without having to render the other pages. *Refinement* is used to refer to the process of selecting some part of a document in the original quality. In fact, one can define a distillation-refinement space for each type of data (text, image, video, etc.), where distillation and refinement can be applied orthogonally to the data in order to reduce binary size.

ActiveProxies [24], developed within the BARWAN project, was a pioneering piece of work focusing on data distillation and refinement. Active proxies are a means to perform on-the-fly content adaptation, in order to support variations on network (i.e. bandwidth), device characteristics (i.e. screen size, color depth, processing power) and software capabilities (e.g. ability to handle specific data encodings). The TACC (Transformation, Aggregation, Caching and Customization) model provides mechanisms for the composition of TACC *workers*, where each worker handles the

distillation or refinement for a specific MIME type. The project built several workers to deal with text, image and video content, such as distillers for GIF and JPEG images, for HTML, for conversion from PostScript to RTF, and for MPEG video streams.

Summarization

Summarization is a sort of lossy compression where *specific* parts of the original data are selected for presentation, aiming at the least possible loss of information. The most common data types summarized for mobile and wireless devices are text and video. Text summarization techniques have been researched for quite a while, but the recent need for displaying Web contents on small screens has given a new push to the field. A video summary (or abstract) is defined as a sequence of still or moving pictures, with or without audio, presenting the content of a video file in such a way that the user is provided with concise information about the content, while the essential message of the original is preserved. It may be a shorter version of a video file assembled by picking important segments from the original, or a series of short clips containing the essence of a longer video file, without a break in the presentation medium [34]. For transmission over a low-throughput connection, video summarization is useful for providing users with a video digest so that they can get the content quickly and comprehensively.

A canonical example of a system that applies video summarization is Mowser [6], a server-side proxy for dynamic context-based modification of HTTP streams, which uses content negotiation as described in the HTTP/1.1 specification. It selects the best representation of a data resource based on the browser-supplied preferences for media type, network connection, available resources, languages and encoding. Mowser allows the user to set viewing/presentation preferences such as: starting point, color capability, video resolution, sound capability, maximum allowed size for text, image, video, audio files, and size restriction for image files. Moreover, techniques to create hierarchical summaries of a video have been developed. The video frames are grouped based on features obtained from visual properties, and the frame closest to the center of each group is chosen as the Representative Frame (Rframe). Rframes are grouped to form a film strip, which is simply a GIF file that can be handled as such by the mobile client.

Intelligent Filtering

Intelligent filtering is usually defined as a mechanism to transform, drop or delay data delivery by applying filters on a data path, according to network or target device conditions.

Mobiware [43, 2] is a QoS-aware middleware platform for mobile multimedia applications. Mobiware introduces the concept of *active filters*, which can be dynamically dispatched during handoff to strategic points in the network (e.g. base stations, mobile devices, etc.) to provide media scaling of audio and video streams when and where needed. Its goal is to support valued-added QoS, with the best utilization of available bandwidth and seamlessly media delivery. There are two styles of filters: *Active Media Filters*, which perform temporal and spatial scaling for multi-resolution video and audio flows, and *Adaptive FEC (Forward Error Correction) Filters*, which protect content against physical radio link impairments. In Mobiware, so-called *QoS Adaptation Proxy (QAP)* objects play a central role in allowing mobile devices to probe resource availability and to adapt to changes in the quality of the wireless link.

Zenel [63] was one of the first to propose a framework for generic filtering. His architecture consists of a Proxy Server, composed by a High-level Proxy and a Low-level Proxy, and a Filter Control (EventManager) component. While the High-level Proxy allows filters for application-layer protocols to be downloaded dynamically from mobile host applications, the Low-level Proxy is used

to create and install filters for the network and transport layers. The EventManager provides a control interface for the instantiated filters. The filters may drop, delay, or transform any sort of data moving to and from the mobile host, in order to improve the perceived quality of the network.

Transcoding

Transcoding is the general process of transforming the format and representation of content: data may be filtered, transformed, converted or reformatted to make it accessible by a variety of devices. Transcoding is commonly used for the conversion of video formats (i.e., VHS to QuickTime, QuickTime to MPEG) or the adjustment of HTML and graphics files to the constraints of mobile devices (e.g. HTML to WML transcoding). It is often used when terminal characteristics prevent the content from being presented in its original format.

Essentially, there are two approaches for transcoding: in the first one the transformation depends only on the type of content, and in the second one the conversion is specified by an external annotation describing specific requirements of the device and the adaptations to be performed. There are far more examples of proxy-based architectures employing the first approach, but we will start describing a system based on the latter approach.

Annotation-based Web content transcoding [26] is an example of the external annotation approach. This system handles HTML documents and focuses on page fragmentation for small-screen devices. Upon receiving a request from a mobile device, the proxy server adapts the document to the capabilities of the particular client, on the basis of associated annotations. An annotation specifies the transformations and contains information to help a transcoding proxy select from several alternative representations the one that best suits the client device. For example, metainformation associated with particular elements may indicate their specific role (proper content, advertisement, decoration, icon) and importance level. Using this metainformation the transcoding proxy can decide not to send an element to a lightweight client, when a decoration role and importance level low are assigned to the element.

In the remainder of this section we summarize some well-known systems adopting the pure content-based transcoding approach.

AT&T Mobile Network (AMN) [49, 14] is a proxy-based mobile platform designed to deliver customized multimedia services to users of mobile devices. The server-side multi-protocol proxy is composed of *devlets*, *infolets* and *applets*. Devlets are protocol adapters that provide protocol interfaces to different mobile devices; infolets are responsible for obtaining information from various data sources; and applets incorporate the application-specific logic. The *proxy engine* arbitrates the communication among devlets, applets and infolets. It also supports user and device profiles for customization, performs content transcoding and adaptation, and invokes proper applets and infolets to answer requests from devlets. The transcoders transform content based on the MIME type specified in the service request. Transcoders have been developed for XML (i.e. transform XML content into the appropriate MIME type), images (i.e. adjust image quality and size according to device profile), video (i.e. convert from MPEG-2 to H.263) and HTML (i.e. remove complex objects such as JavaScripts, replace images with hyperlinks, split long pages into shorter ones).

IBM Internet Transcoding for Universal Access [40] is a transcoding system that adapts video, images, audio and text to the devices with diverse capabilities using a proxy that allows the content to be summarized, translated and converted on-the-fly. This system has two key components: a representation scheme called the *InfoPyramid* that provides a multi-modal, multi-resolution representation hierarchy for the multimedia content; and a *customizer* that selects the best content

representation to meet the client capabilities. The system handles composite multimedia documents and device constraints such as screen size, color, cost, and hardware and software capabilities.

The Mowgli [35] infrastructure consists of two mediators located on the mobile host and the mobile-connection host which use the MowgliHTTP protocol to communicate with each other, reducing the number of round-trips between the client and the server. Mowgli reduces HTTP data transfer over the wireless link by employing three different techniques: data compression, caching and intelligent filtering. It performs only GIF to JPEG conversions, and large images embedded in HTML documents are not transferred to the mobile node.

4.3 Caching and Consistency Management

Caching of data close to (or at) the mobile host is a very common task assigned to proxies. The common and main goals of caching are to reduce traffic to and from the source server, restrict the user-perceived latency, conserve wireless bandwidth and the mobile device's battery power, as well as handle client disconnections, i.e. support some limited functionality of the client application at mobile hosts while disconnected.

While for the first two goals - reducing traffic and latency - it may be sufficient to cache data at a node on the edge of the wired network (i.e. to use a server-side proxy), for the remaining goals caching at a client-side proxy on the mobile host is necessary.

In principle, server-side caching for mobile hosts does not significantly differ from conventional proxy-based caching for wired network access (e.g. Web proxies). However, in a mobile setting the main difference is that access happens through a wider span of mobile devices (ranging from laptops to cell phones) and wireless links with very different capabilities. In order to cope with such diversity, information providers have begun to store their contents in different formats and fidelities, and to use Web servers enabled with HTTP's Content Negotiation feature to select the most suitable content format for each client, according to browser-supplied preferences. This practice has the following major implication on caching, since each request is treated independently, popular items (e.g. Web objects) might be cached in different formats at the proxy at the same time, wasting much storage also at the proxies. In order to handle this problem several studies have proposed the combination of active, i.e. dynamic, transcoding (cf. section 4.2) and adaptive caching at the proxy, so as to transcode contents into the various formats closer to the client. An example is the Transcoding-enabled Caching Proxy (TeC) proposed in [54], which is able to dynamically transcode video objects from a higher bit rate to a lower bit rate, depending on client and link capabilities.

For many protocols, caching is also a key feature used to optimize data transfer over links with smaller throughput, such as wireless links. For example, when accessing dynamic Web pages a client-side cache can hold a base page, and only the differences to this base page need to be transmitted. This technique has been exploited in Web Express [27].

Client-side caching, on the other hand, aims at enabling some limited form of data access by the user during the time in which the mobile host is disconnected. The main problem is to handle involuntary disconnections and to guarantee consistency of the cached objects (e.g. files, database records, etc.), specially when cached objects can be modified by clients, and more than one client can cache the same data object, or the original copy of the object at the server can be modified by other means.

Several approaches for handling cache consistency in those networks have been proposed in the context of databases [4], but there is also significant work from other areas, such as distributed file systems and other data-sharing applications.

Due to the high probability of disconnections and the limited wireless bandwidth, neither a pure detection-based approach (client detects inconsistencies), nor a pure avoidance-based approach (i.e. server sends Invalidation Reports to the cache holder whenever the original object is modified) can be used for guaranteeing cache coherence. However, several other strategies for cache coherence, which are either based on stateful, stateless or hybrid servers, or on incremental approaches, have been proposed [3, 10]

In fact, many recent studies suggest that Invalidation Report-based caching management is better suited for mobile networks. But a major problem with Invalidation Reports is that disconnected clients may miss some of these reports. In order to overcome this problem and also avoid stateful servers that must track which clients have received (and acknowledged) which reports, Gupta and Srimani [31] have proposed the Asynchronous Stateful (AS) caching scheme, where server-side proxies (called Home Location Caches (HLC)) buffer the invalidation reports from servers while the MH is disconnected, and deliver these reports to the MH when it reconnects to the network. Furthermore, each time a MH migrates, this buffer of invalidation reports is transferred to the HLC close to next Access Point. More recently, other cache invalidation schemes based on intermediates, which claim to be more efficient [60], have been proposed.

In order to ensure operation in spite of intermittent connectivity, two main approaches have been explored. The first is to support eager prefetching of data objects, and perform conflict resolution on demand. The second considers each mobile host as being an autonomous entity, and regards the disconnected mode, rather than the connected mode, as the norm and not the exception. Here, hosts synchronize their data objects upon sporadic connections.

The Coda file system [32] is the canonical example for the first approach. As long as the mobile host is connected, Coda's client-side proxy, Venus, automatically caches all the files being used by the user's applications, allowing the user to work on these files during the disconnection phase. The set of files to be cached is partially predicted by the proxy, and partially indicated explicitly by the user. Coda's predictive caching (called hoarding) involves monitoring the user activity to discover his profile (e.g. the common applications and the corresponding files). When the mobile host reconnects, Venus reintegrates the files in the cache with the corresponding files on the servers. Any conflicts during reintegration due to updates on cached objects are resolved based on an optimistic concurrency control scheme [32] using logs, application-specific resolvers, and possibly with manual intervention of the user.

Another example of the first approach is the general-purpose OSMOSE Mobility framework [21], which aims at supporting general service continuity in spite of disconnections. The main responsibilities of its client-side caching proxy include the detection of changes of the MH's connectivity, transparent servicing of application requests from a local cache shared by all client applications, and cache reconciliation whenever the mobile host becomes reconnected.

A well-known example of the other approach (asynchronous operation) is the Bayou [58] system for Peer-to-Peer file sharing based on pair-wise communication, propagation of write operations, and constraints on the propagation of the writes. Bayou's replicated data manager (i.e. client-side proxy) executes an anti-entropy protocol for conflict resolution with data managers on other connected hosts and offers a session guarantee to mobile users. In Bayou, a session is defined as an abstraction for a sequence of read and write operations performed during the execution of an application. Instead of ensuring atomicity and serializability as with atomic transactions, sessions with guarantees enable a client to observe a replicated database that is consistent with its own actions even if it reads and writes data from various potentially inconsistent servers. Sessions also support controlling the scope and selection of the guarantees.

4.4 Session Management

Many applications are based upon the notion of a *session*, which in general consists of a set, or sequence, of coherent actions performed by a user. Although a session may differ very much from one type of application or service to another, all of them have the notion of a *session state*. In a mobile and wireless computing environment, session management is thus concerned with maintaining an application's or service's session state in spite of disconnections and migrations of the user. One should remark that in this context, migration can have several meanings. In the simplest form, a user keeps her device and just reconnects to a different AP within the same network, or a different network, which we call *network migration*. A more complex kind of migration happens when the user switches devices, but wants to seamlessly continue using the same service from the new device (device-migration). In this case, the session state must not only be transferred to the new device, but probably must also be adapted to the new communication/transport protocol (e.g. HTTP to WAP). Finally, there is yet a more sophisticated migration, where the user switches between different, albeit related applications (application-migration). For example, the user may switch from synchronous to asynchronous communication when she notices that her device is connected to a wireless network with higher latency and smaller throughput. In this case, the session initiated with the first service must be transformed into the session of the new service.

Session management essentially deals with following issues: how to represent, encapsulate, and adapt the session state? How to transfer and install the session state at the new device? How to implement mechanisms for controlling on-line sessions?

Gardner and Shahi [36] have proposed a middleware-level proxy architecture which persists voice and Web data sessions, and allows user to seamlessly transfer session states between different devices, or to share them with other users. It consists of two parts: a server-side proxy that intercepts application-level command, handles user authentication, authorization and session storage and synchronization. On the client side, it consists of a GUI for session administration (e.g. the user may keep several ongoing sessions) and application plug-ins for capturing the state of the associated applications, managing the transfer and synchronization of session state between multiple clients. Central to this work is the definition of a session schema for capturing state information of web browsing sessions.

4.5 Handover Management

Among the several advantages offered by the wireless network, user mobility is perhaps the most appealing benefit, since it enables users to access information from different locations and even while they are moving. However, in order to support this, mobile networks must provide mechanisms for mobility management, a.k.a. handover management. A handover, or handoff, occurs when a user previously connected to some network reconnects to the same or to a new network. In both cases, handover should be seamless, i.e. no data should be lost. Handover management is mainly responsible for two tasks: updating the MH's location/address to ensure that it can be reached, and transferring the MH's session state from the old to the new network. Thus, essentially handover management is concerned with offering mobility transparency to the applications.

Wireless CORBA [44] and Mobile IP [47], are two canonical examples of proxy-based infrastructure that support handover management. Both define a very similar architecture composed by three basic elements: Home Location Agents (HLA), proxies (in mobile IP terminology Foreign Agents) and the MHs. The HLA contains records of which proxy is serving which MH, and the proxies do the handover management and intermediate all communication between the MH and

servers at the wired network. Due to the similarity between the Wireless CORBA and Mobile IP approaches, in the following we will only describe Wireless CORBA in some more detail.

In 2001 the Object Management Group issued the Wireless Access and Terminal Mobility in CORBA specification [44], a.k.a Wireless CORBA (wCORBA) [7]. This specification supports mobility transparency of objects through a mobile Interoperable Object Reference (Mobile IOR) and a GIOP tunneling protocol, which handles handovers between Access Bridges in a technology-independent way. The Access Bridge plays the role of an object proxy, through which clients on the wired network can request a method of an object on a MH, and vice-versa. Among other tasks, Access Bridges are also in charge of synchronizing the session state transferred between them when the MH performs a handover. This is implemented through notification messages about the MH's mobility events, which are exchanged among the Access Bridges and the MH's Home Location Agent.

For providing mobility transparency to CORBA applications the Wireless CORBA's special Interoperable Object Reference, Mobile IOR, is used to hide the device's mobility from clients invoking operations on target objects on the device. Instead of informing the concrete address of the target object, the host and port fields of an IIOP Profile in a Mobile IOR indicate the address of either the HLA of the MH with the target object, or the Access Bridge currently associated with the MH. This way, all information addressed to the MH is routed through its HLA, if it has one, which in turn forwards the received information to the MH's current Access Bridge (through a LOCATION_FORWARD message) which forwards the information directly to the MH. If the MH is homeless the information should be sent directly to the MH's current Access Bridge.

Differently from Wireless Corba and Mobile IP, the Home-proxy based wireless Internet framework [12] provides mobility support through an application level proxy. Besides supporting handover management, this framework aims to facilitate the integration of mobility support with QoS management mechanisms.

Like in Mobile IP, all packets addressed to the MH are first routed to its home network. Then the Home Proxy (HP) intercepts the packets and redirects them to the current subnet of the MH. Unlike the Home Agent of Mobile IP, the HP uses a Split-Connection Approach (cf. Section 4.1), based on Session Layer Mobility (SLM) [33], to relay packets to the MH. Using this approach the HP creates two separate connections, one with the MH and the other with the peer host so that it can route packets between them. This work extends the SLM for recovering and managing the TCP connections state when the MH performs a handover.

Furthermore, the HP supports a dual-level mobility scheme: macro-mobility, which uses the SLM, and micro-mobility, which uses a dynamic per-host routing scheme. The micro-mobility management technique can be used to supplement the split-connection approach that handles macro-mobility. The HP also supports advance macro and micro resource reservation to meet QoS requirements. For guaranteeing compatibility with other protocols (e.g. DiffServ, RSVP) the HP does not require IP-level tunnelling or use of mobility agents at the visited subnet. This way, the roaming traffic of an MH is indistinguishable from other traffic generated by fixed hosts. This allows mobility management to be fully decoupled from the particular QoS management architecture on the Internet backbone.

4.6 Discovery and Auto-configuration

The dynamicity of mobile computing environments imposes stronger requirements for service discovery mechanisms than traditional distributed environments. For example, service discovery

should handle changes in availability of devices or services, and be able to choose the most suitable service for each client, according to its current context.

Proxy-based system architectures can help to overcome such challenges by hiding network heterogeneity and dynamicity, as well as reducing complexity of control mechanisms to manage such dynamism. Accessing a service through a proxy, instead of interacting directly with a particular instance of the service, can remove from the client the need for choosing the best service, and reconfigure it when the execution context changes. Some distributed systems, such as Jini, use proxy-based approaches for service discovery.

Jini [59] is a distributed system middleware based on the idea of federating groups of users, resources and services. A client can obtain references to services using the *Jini Lookup Service* (JLS). A new service must register by uploading a service proxy in the JLS. When a client obtains a reference to a service, it downloads such proxy to access the service. A service proxy is a mediator between a client and the service and encapsulates the protocol required for communication. Jini's join/leave protocol provides a flexible solution for handling service dynamicity, such as location change or unavailability. In order to address mobile computing requirements, some Jini-based service discovery mechanisms have been developed allowing middleware deployment in limited resource devices [29], supporting a more flexible discovery mechanism [13], or offering richer service query capability [11].

Other proxy-based architectures focus on dynamic service reconfiguration. In WebPADS [15], clients and servers over wireless links communicate to each other using a WebPADS proxy instance that provides a service discovery mechanism. When some adaptation is required (e.g. bandwidth decrease), the proxy loads the suitable service from the network and installs it in the WebPADS proxy. A service is developed in *mobilelets*, using the WebPADS API, and can migrate from the service directory to a proxy. Such reconfiguration is transparent to clients and servers.

PARM [41] uses proxy servers at a wired network responsible for intermediate negotiations among mobile clients and servers. A power broker assigns each mobile device to a proxy server, depending on its location and resource availability. In order to decrease energy consumption, the mobile device can migrate code to its proxy server or be assigned to another proxy server. A single actual server can be associated to several proxy servers, in order to achieve load balance.

4.7 Security and Privacy

Services for secure mobile communications may also employ a proxy-based approach. In a mobile environment, proxies can be used to decentralize the authentication process and allow the application to use a public-key security model on the wired network that may require a high computational effort, while keeping the computed functions at the device as simple as possible. Furthermore, by acting as an intermediary between clients and servers, proxies may provide a natural and efficient anonymity for the mobile application hiding the real identity of the requester whenever necessary. In this case, the proxies are used to represent mobile clients and may be responsible for handling privacy, authorization, user authentication, or data encryption.

In the proxy-based security architecture proposed by [9], the proxies implement a public-key security model to control the access over shared resources (file, printer, etc.). For guaranteeing security and privacy this work uses two separate protocols: a protocol for secure device-to-proxy communication and a protocol for secure proxy-to-proxy communication.

The protocol for device-to-proxy communication sets up a secure channel that encrypts and authenticates all the messages in the wireless link using symmetric keys. The HMAC-MD5 algorithm

is used for authentication and the RC5 algorithm for encryption. On the other hand, the proxy-to-proxy protocol uses the SPKI/SDSI(Simple Public Key Infrastructure/Simple Distributed Security Infrastructure) [50] to implement the access control through ACLs (Access Control Lists) on the public or protected resources. If a requested resource is protected by an ACL, the request must be accompanied by a “proof of authenticity” which shows that it is authentic, and a “proof of authorization” which shows that the requester is authorized to perform the particular request on the particular resource. The proof of authenticity is typically a signed request, and the proof of authorization is typically a chain of certificates.

With respect to user authentication, there is plenty of work employing proxy-based signature generation. For instance, Park and Lee [46] have proposed a nominative proxy signature scheme, a method in which the proxy generates a nominative signature and transmits it to the Certification Authority, instead of sending it to the original client. The advantages are that it preserves the users’ anonymity and decreases the mobile client’s computational cost of computing the signature. A refinement of their approach has recently been proposed by Seo and Lee [55], which guarantees non-repudiation by the proxy, and which does not assume a secure channel between the client and the proxy.

Furthermore, there are several other studies that provide algorithms and approaches for ensuring privacy. For example, Gruteser and Grunwald’s work on spatial and temporal cloaking [25] uses a trusted proxy to adjust the resolution of location reported to services based on the density of users in a region. Since many users report their location through the proxy, user density is known. Thus, the proxy can provide anonymity [57], informing the number but not the identity of users in a given area.

4.8 Checkpointing and Recovery

In distributed systems, recovery is typically based on checkpointing, a mechanism to create snapshots of distributed computations. Checkpointing algorithms may be expensive in mobile computing environments because they generate a control message overhead in the network. Moreover, a checkpointing algorithm must handle handover and disconnected operation of all MH participating in the recoverable environment.

Proxy-based recovery is an interesting approach to address such requirements. Using a proxy at the wired network as a representative of a MH, an algorithm can transfer to the fixed network the task of managing a client checkpoint view and then reduce the impact of managing mobile participants.

Proxy-based checkpointing approaches are typically used in coordinated checkpointing. For example, the checkpointing algorithm proposed in [42] uses a proxy coordinator that acts as a proxy for processes running in a mobile host. Proxy coordinators are typically placed at APs in order to reduce the communication overhead imposed by wireless links. This approach reduces the overhead caused by checkpoint messages in wireless medium, because these messages are exchanged only among hosts in the wired network. Checkpoint control messages are handled by the AP, whereas application-specific ones are buffered at the AP.

Some studies propose a more transparent use of proxy-based approaches for recovery. The work in [62] adopts a three tier model, composed by a client interacting with a server, using a proxy as intermediary. A proxy monitors interaction between client and server and transparently maintains the entire state of interaction between the two peers. When the mobile client disconnects or fails, the proxy sustains the client-side state of the connection so as to hide from the server the non-availability of the client. The proxy also logs all messages sent to the client that can affect its state. A client

application recovers from failures by requesting its latest state and message logs from the proxy. This approach avoids expensive protocols for reconnection with the server and is useful for transaction management for m-commerce applications.

4.9 Other tasks

In this section we will briefly discuss some other proxy tasks commonly found in the literature. There are probably many other application- or protocol-specific tasks that could be mentioned, but due to space limitation, unfortunately we cannot discuss them in depth here.

Personalization refers to the function of tailoring the information content in response to a request, learning the user's profile or current preferences, and possibly performing some complex task on behalf of the user. For example, when searching for some information on the Web, the proxy can select the most suitable URLs for a given user request, or infer a semantic match between the user's query terms and the objects referred by the URLs. For example, *eRACE* [16] supports personalization in form of user-specific differentiated services (filtering, data aggregation, personalized dissemination), which are determined by XML-encoded eRACE profiles.

Content Creation by a proxy is possible when the infra-structure supports the off-loading and execution of application- and client-specific code at the proxy. On behalf of the client, and based on user preferences, the proxy might be able to autonomously access network services, discover new data sources, retrieve information from several sources, in order to produce new content which is a composition, a summary or a selection of the different pieces of retrieved data. IBM's WBI [5] and TACC [24] are examples of infra-structures supporting the dynamic instantiation of content aggregation modules into general-purpose proxies.

Name Resolution is a common task performed by Web proxies (e.g. the WAP Gateway [22]). The main reason is that current Web pages usually consist of several objects from different sources, requiring several DNS lookups before the full page can be displayed. In fact, it has been identified that DNS queries are the major overhead of Web traffic, and several groups have worked on techniques to reduce it. For example, DNS rewriting mechanisms have been proposed [51], where a proxy intercepts DNS responses and caches IP addresses for the mobile client.

5 Proxy Frameworks

As proxies have been used as a general approach for handling dynamic adaptation, several efforts have been made to develop generic proxy architectures, or *proxy frameworks*, that can be customized or extended to solve a particular problem. An example of such an effort is IETF's Open Pluggable Edge Services [61], which proposes a reference architecture for web proxies, addressing issues as security, distribution and dynamic configuration.

In this section we describe common mechanisms used in proxy frameworks and compare well-known systems, such as TACC, RAPIDware, Mobeware, MARCH, Web Intermediaries, and MOCA ProxyFramework. The RAPIDware [37, 39] project has proposed adaptive proxy services for multimedia streams. Mobeware [2, 43] is a QoS-aware middleware platform for multimedia applications which also provides support for handoff control. Web Intermediaries (WBI) [5, 28] have been developed at IBM, for HTTP-based adaptations, such as personalizing contents, transcoding, or caching. MARCH [1], TACC [8] and MOCA's ProxyFramework [52] are general-purpose content adaptation frameworks.

Most proxy frameworks provide general-purpose solutions for the following four main issues: (a) implementation and composition of adaptation modules, called *adapters*; (b) description of the conditions in which the adapters should be applied; (c) monitoring of the context, such as the mobile device's profile, the application's state and the communication bandwidth; and (d) the loading of adapters. In the remainder of this section we will discuss these features in more detail. A complementary discussion about proxy frameworks can be found in [18].

Adapter Development

The main customization point of a proxy framework is the *adapter*¹, a module responsible for implementing a transcoding function of a message or its content. A proxy (i.e. an instance derived from the framework) may use several adapter instances for implementing specific adaptations required for different clients or contexts. Taking into account the client's current context, a proxy determines at runtime which adapter should be used for a message or data content. In some situations (e.g. contexts), more than one adapter can be selected for transcoding a message. Therefore, some frameworks support the definition of priorities, ordering, and/or composition of adapters.

Most proxy frameworks are designed using extensibility mechanisms and component-based approaches to support the development and composition of adapters, as well as their loading into a proxy. In some frameworks, such as WBI, RAPIDware and MARCH, adapters can be developed as independent and composable components that are stored in adapter repositories or libraries and deployed in proxies. Some frameworks provide classes of special-purpose adapters. For example, Mobiware supports two kinds of adapters: Active Media Filters, for media content adaptation, and Adaptive FEC Filters, for error correction. RAPIDware also provides some FEC filters, in order to improve the ability of the audio/video stream to tolerate errors in a wireless environment. The TACC model supports adapters for transformation (content adaptation), aggregation (information collecting), caching and customization.

Adapter Selection

The decision of which adapters to use and when to use them is an extensible characteristic of proxy frameworks, which can be defined in two ways: via programmable interfaces or via rule-based configuration. An example of the first way can be found in Mobiware, where the application requirements (utility function) and the adaptations to be applied (adaptation policy) must be programmed using a framework-provided API. When a rule-based configuration is supported, the developer must define rules which contain the trigger conditions, described in terms of the client and network states (i.e. context); the adaptations to be executed; and sometimes also a priority of the rule. Usually, the rules are described manually via an XML (or RuleML) file. In MARCH the selection process evaluates the rule set during session setup, and produces as the result a set of adapters to use (chain of adapters). In MoCA's ProxyFramework and WBI, rules are evaluated just before each message is sent to the client.

Rule-based systems are easily configured and less error prone (defining a model) than the ones based on programmable interfaces; besides there is no need to deal with intrinsic details of the framework. Furthermore, only the content provider can decide which adaptation is acceptable under different contexts, and thus, by using rules, may define the sequence of adaptations to apply to data, better controlling their composition, which is a very complex task to automate.

¹Some publications use different names for the adapter, such as *filter* [2, 39], *transcoder* [5] and *worker* [8].

Context Monitoring

The monitoring and gathering of context information (i.e. the client's profile, and conditions of the execution environment, such as available resources, load and energy at the mobile host and the network) are part of the desired functionality of proxy frameworks. The collection of the network state, such as available bandwidth or connectivity, is generally done via a monitoring function or service, as in TACC, MARCH and MoCA ProxyFramework. Information related to the client may be obtained at their startup connection request [1], via a customization database containing profiles [8], or through monitoring of the device's resources [52]. In most frameworks, context changes are notified through asynchronous events, which must be interpreted and processed by the proxy in order to execute the appropriate action.

Adapter Loading and Execution

According to how adapters are loaded and activated, proxy frameworks can be classified as *configurable* or *dynamic* proxies. In a *configurable proxy*, adapters are defined statically at proxy deployment time. The developer can change the proxy's behavior by using trigger rules that define the order and the context in which an adapter should be executed. A *dynamic proxy* supports dynamic and on-demand loading of adapters from an adapter repository, according to the current context.

Two examples of dynamic proxies are RAPIDware and MARCH. RAPIDware provides a composable proxy framework to support the dynamic composition of services by fetching adapters (called filters) from a repository, and instantiating and reconfiguring them dynamically on the proxy in response to the changing needs of mobile clients. MARCH provides a dynamic execution environment for adapters, which facilitates the uploading of proxies on a per-session basis, which may be placed on the server or on mobile devices. In MARCH, the MAS (Mobile Aware Server) component is in charge of making the decision of which adapters, chosen from the proxy repositories, are to be used and where to execute them.

An example of framework for configurable deployment of proxies is Web Intermediaries (WBI). At proxy startup, the registered adapters (or plug-ins) are instantiated with the corresponding firing conditions in rules with an associated priority. WBI supports the aggregation of adapters, and the proxy can be placed either on the server or on the client side. Another example is MoCA ProxyFramework, where the adapters are instantiated during proxy initialization, according to the trigger rules (described in an XML configuration file) specifying the context in which the adaptation (or set of adaptations) should be applied. This framework also supports chaining of adapters, use of priorities, and mechanisms for specifying caching policies.

Comparing the two approaches, the dynamic loading of adapters provides more flexibility to the system. However, configurable proxies support verification of a consistent combination/configuration of adapters. In addition, dynamic (down)loading of adapters can be time consuming. Therefore, it is more suited for systems where context changes are not very frequent.

Table 1 presents the cited frameworks, summarizing their main characteristics according to the aspects discussed in this section and in section 3.

	RAPIDware	Mobiware	MARCH	TACC	MoCA Framework	WBI
Purpose	Multimedia	Multimedia, QoS	General	General	General	Web Apps.
Level	Middleware	Middleware	Application	Middleware	Middleware	Application
Proxy Placement	server-side	client-side and server-side	server-side and proxy-pair	server-side	server-side	client-side and server-side.
Dynamic Adapter Loading	Yes	Yes	Yes	No	No	No
Adaptation Selection	Programmable	Programmable	Trigger-Rules Configuration	Programmable	Trigger-Rules Configuration	Trigger-Rules Configuration
Functionality	Content Adaptation	Content Adaptation, Handover Mngt	Content Adaptation	Caching, Content Adaptation	Caching, Content Adaptation	Caching, Content Adaptation
Communication	Synchronous	Synchronous	Synchronous	Synchronous	Synchronous, Asynchronous	Synchronous, Asynchronous
Context Awareness	wireless link	wireless link	device & wireless link	wireless link	device & wireless link	-

Table 1: Comparison table of extensible proxy approaches

Comparing the presented systems, one should notice that all of them support content adaptation, while caching management appears as the second most frequent functionality, and handover management is provided only by Mobiware. Furthermore, there are equal numbers of systems concerning the level (middleware *versus* application), the capability of dynamic adapter loading, and the form of adaptation selection (programmable *versus* trigger-rule configuration). Concerning communication capabilities, only MoCA Framework and WBI support asynchronous (publish/subscribe) communication, which has been recognized as best suited for mobile computing. Context awareness is also supported by most of the frameworks (i.e. except WBI), but only MARCH and MoCA Framework consider also the state of the client’s devices. Although it is quite difficult to compare the frameworks, Mobiware seems to be one of the most complete systems in terms of supported functionality, extensibility and architecture.

6 Conclusion

In this chapter we have presented two classifications of proxy-based architectures for mobile computing, identified and discussed broad categories of responsibilities assigned to these intermediaries, and presented the most representative examples of such systems. Despite the widespread adoption of proxy-based architectures for mobile computing, there is a number of open challenges which need to be addressed in order to make proxy-based systems more flexible, scalable and shaped to the specific requirements of current and future mobile networks and applications.

More precisely, there exist some justified concerns about the scalability of the proxy-based approaches. As the number of mobile users connecting through wireless links increases, server-side proxies may not be able to cope with the increasing computational demands of the mobile clients they represent. This is particularly true if the adaptation/transcoding performed at the proxies is specific for each mobile client (e.g. takes into account the particular device characteristics and limitations) and is “resource-hungry”, such as for example the transcoding of multimedia streams.

The obvious alternative to the use of proxies is the adoption of an end-to-end approach where information servers pre-transcode contents (e.g. multimedia streams) to a pre-defined array of

different formats and resolutions, and deliver data to each client in the most suitable form and fidelity, based on the information of the corresponding device capabilities and the current throughput of the wireless link being used. Because disk storage is increasingly inexpensive, in fact several content providers are pursuing such an end-to-end approach.

However, a pure end-to-end approach has the following major drawbacks: (a) it will only work with servers that are capable of interpreting the client capabilities and the current network conditions; (b) all content would have to be transcoded independently of whether it will be or not accessed by mobile clients, which is not feasible on a large scale; (c) since the quality of wireless connections can vary significantly and dynamically, specially in heterogeneous wireless networks, dynamic and seamless switching from one format/fidelity to another during a transmission is likely to be a problem, and (d) disconnection management, caching and protocol translation are tasks that are not suited to be handled by servers, since they concern specific problems related to the wireless link and device limitations. Instead, they should be handled by an independent entity (e.g. a proxy) which is only concerned with the choice and execution of suitable adaptations, rather than with the application logic. Hence, for many adaptations required by mobile applications, proxy based approaches seem to be the most natural and best solution.

Nevertheless, a major challenge remains: to design proxy-based architectures which are scalable. One possibility is to combine the end-to-end and proxy approaches, such as discussed in [30]. Another promising approach is to develop infra-structures that support the deployment of distributed and cooperative networks of intermediaries that collectively perform adaptations for a huge variety of devices and protocols, such as in IETF's proposals of *Open Pluggable Edge Services* [61].

As the number of applications for mobile networks increases, and their services become more complex and personalized, proxies will be used for an increasing number of specialized functions. Although each (type of) application will have specific demands for proxy based functions, we have identified a common and recurrent set of functions, structures and architectural patterns in proxy implementations which shall be used as the basis for developing proxies for specific needs. In our opinion, there is an increasing demand for flexible and extensible tools and frameworks for the rapid development and customization of proxy-based architectures, both at the application and the middleware levels.

The other trend we envisage in proxy-based architectures is that of dynamic proxy configuration, which allows shaping the proxy's functionality according to dynamic demand by the clients, server load, or the current mobile network conditions. Ideally, we should have a library of standardized modules for content adaptation, protocol translation, caching management, etc., which could be automatically loaded, instantiated and interconnected in a proxy framework, according to the specific needs and network conditions.

References

- [1] S. Ardon, P. Gunningberg, B. Landfeldt, M. Portmann Y. Ismailov, and A. Seneviratne. March: a distributed content adaptation architecture. *International Journal of Communication Systems, Special Issue: Wireless Access to the Global Internet: Mobile Radio Networks and Satellite Systems.*, 16(1), 2003.
- [2] A. Balachandran, A.T. Campbell, and M.E. Kounavis. Active filters: Delivering scalable media to mobile devices. In *Seventh Intl. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, St Louis, May 1997.

- [3] D. Barbara and T. Imielinski. Sleepers and Workaholics: Caching Strategies in Mobile Environments. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, May 1994*.
- [4] Daniel Barbara. Mobile Computing and Databases - A Survey. *Trans. on Knowledge and Data Engineering*, 11(1):108–117, 1999.
- [5] R. Barrett and P. P. Maglio. Intermediaries: An approach to manipulating information streams. *IBM Systems Journal* 38, 1999.
- [6] Harini Bharadvaj, Anupam Joshi, and Sansanee Auephanwiriyakul. An active transcoding proxy to support mobile web access. In *Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems*, 1998.
- [7] Kenneth Black, Jon Currey, Jaakko Kangasharju, Jari Lämsiö, and Kimmo Raatikainen. Wireless access and terminal mobility in CORBA, 2001. White Paper.
- [8] E. Brewer and et al. A network architecture for heterogeneous mobile computing. *IEEE Personal Communications Magazine*, oct 1998.
- [9] M. Burnside, D. Clarke, T. Mills, A. Maywah, S. Devadas, and R. Rivest. Proxy-based security protocols in networked mobile devices. In *SAC 02: Proceedings of the 2002 ACM symposium on Applied computing*, pages 265–272. ACM Press, 2002.
- [10] J. Cai, Kian-Lee Tan, and Beng Chin Ooi. On incremental cache coherency schemes in mobile computing environments. In *ICDE '97: Proceedings of the Thirteenth International Conference on Data Engineering*, pages 114–123. IEEE Computer Society, 1997.
- [11] Dipanjan Chakraborty, Filip Perich, Sasikanth Avancha, and Anupam Joshi. Dreggie: A smart service discovery technique for e-commerce applications. In *20th Symposium on Reliable Distributed Systems*, October 2001.
- [12] J. Chan, B. Landfeldt, R. Liu, and A. Seneviratne. A home-proxy based wireless internet framework in supporting mobility and roaming of real-time services. *IEICE Transactions on Communications, Special Issue on Mobile Multimedia Communications*, E84-B(4), April 2001.
- [13] Harry Chen, Anupam Joshi, and Timothy W. Finin. Dynamic service discovery for mobile computing: Intelligent agents meet jini in the aether. *Cluster Computing*, 4(4):343–354, 2001.
- [14] Yih-Farn Chen, Huale Huang, Rittwik Jana, Trevor Jim, Matti Hiltunen, Sam John, Serban Jora, Radhakrishnan Muthumanickam, and Bin Wei. immobile ee: an enterprise mobile service platform. *Wireless Networks*, 9(4):283–297, 2003.
- [15] Siu-Nam Chuang, Alvin T.S. Chan, Jiannong Cao, and Ronnie Cheung. Dynamic service reconfiguration for wireless web access. In *WWW '03: Proceedings of the twelfth international conference on World Wide Web*, pages 58–67. ACM Press, 2003.
- [16] M. Dikaiakos and D. Zeinalipour-Yazti. A Distributed Middleware Infrastructure for Personalized Services. Technical Report TR-2001-4, Dept. of Computer Science, University of Cyprus, December 2001.
- [17] Marios Dikaiakos. Intermediaries for the world-wide web: Overview and classification. In *Seventh International Symposium on Computers and Communications (ISCC'02)*, pages 231–238, Taormina, Italy, July 2002.
- [18] Marios D. Dikaiakos. Intermediary infrastructures for the world wide web. *Computer Networks*, 45(4):421–447, February 2004.

- [19] NTT DoCoMo. i-Mode, 2004. www.nttdocomo.com/corebiz/services/imode (Last visited: February 2005).
- [20] Hala Elaarg. Improving TCP Performance over Mobile Networks. *ACM Computing Surveys*, 34, September 2002.
- [21] Denis Conan et al. Wp2 frameworks: Architecture and api of the mobility framework. Technical Report WP2-040108-1, ITEA (Information Technology for European Advancement), INT, INRIA/OASIS, and Thales, December 2004.
- [22] WAP Forum. Wireless Application Protocol Architecture Specification. Wireless Application Protocol Technical Specification WAP-210-WAPArch-20010712-a, 2001. www.wapforum.org.
- [23] WAP Forum. Wireless Profiled TCP, Version 31. Wireless Application Protocol Technical Specification WAP-225-TCP-20010331-a, March 2001.
- [24] A. Fox, S. Gribble, Y. Chawathe, and E. Brewer. Adapting to network and client variation using active proxies: Lessons and perspectives. *IEEE Personal Communications*, 1998.
- [25] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of The First International Conference on Mobile Systems, Applications, and Services*, (Mobisys 2002) 2002.
- [26] Masahiro Hori, Goh Kondoh, Kouichi Ono, Shin ichi Hirose, and Sandeep Singhal. Annotation-based web content transcoding. *Computer Networks*, 33(1-6):197–211, 2000.
- [27] Barron C. Housel and David B. Lindquist. Webexpress: a system for optimizing web browsing in a wireless environment. In *MobiCom '96: Proceedings of the 2nd annual international conference on Mobile computing and networking*, pages 108–116. ACM Press, 1996.
- [28] Steven C. Ihde, Paul P. Maglio, Jörg Meyer, and Rob Barrett. Intermediary-based transcoding framework. In *Ninth International World Wide Web Conference*, Amsterdam, The Netherlands, 2000.
- [29] PsiNaptic Inc. Jmatos. Home page, 2004. <http://www.psinaptic.com/> (Last visited February 2005).
- [30] Anupam Joshi. On proxy agents, mobility, and web access. *Mobile Networks and Applications*, 5(4):233–241, 2000.
- [31] Anurag Kahol, Sumit Khurana, Sandeep K.S. Gupta, and Pradip K. Srimani. A strategy to manage cache consistency in a disconnected distributed environment. *IEEE Trans. Parallel Distrib. Syst.*, 12(7):686–700, 2001.
- [32] J.J. Kistler and M. Satyanarayanan. Disconnected Operation in the Coda File System. *ACM Transactions on Computer Systems*, 10(1):3–25, February 1992.
- [33] B. Landfeldt, T. Larsson, Y. Ismailov, and A. Seneviratne. Slm, a framework for session layer mobility management. *IEEE Computer Communications and Networks*, pages 452–456, 1999.
- [34] R. Lienhart, S. Pfeiffer, and W. Effelsberg. Video abstracting. *Communications of the ACM*, 40(12), December 1997.
- [35] M. Liljeberg, T. Alanko, M. Kojo, H. Laamanen, and K Raatikainen. Optimizing world-wide web for weakly connected mobile workstations: An indirect approach. In *Proc. 2nd International Workshop on Services in Distributed and Networked Environments (SDNE'95)*, Whistler, Canada, June 1996.

- [36] Gardner M and Shahi A. Mobile Web sessions for mobile computing. Chimera Working Paper 2004-01, University of Essex, 2004.
- [37] P. K. McKinley, U. Padmanabhan, and N. Ancha. Experiments in Composing Proxy Audio Services for Mobile Users. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, Heidelberg, pages 99–120, November 2001.
- [38] P. K. McKinley and Udiyan I. Padmanabhan. Design of composable proxy filters for wireless collaborative computing. Technical Report MSU-CSE-00-26, Department of Computer Science, Michigan State University, East Lansing, Michigan, November 2000.
- [39] Philip K. McKinley, Udiyan I. Padmanabhan, Nandagopal Ancha, and Seyed Masoud Sadjadi. Composable proxy services to support collaboration on the mobile internet. *IEEE Transactions on Computers*, 52(6):713–726, June 2003.
- [40] Rakesh Mohan, John R. Smith, and Chung-Sheng Li. Adapting multimedia internet content for universal access. *IEEE Transactions on Multimedia*, 1(1):104–114, 1999.
- [41] Shivajit Mohapatra and Nalini Venkatasubramanian. Parm: Power aware reconfigurable middleware. In *IEEE International Conference on Distributed Computer Systems - (ICDCS-23)*, Rhode Island, May 2003.
- [42] Weigang Ni, Susan V. Vrbsky, and Sibabrata Ray. Low-cost nonblocking coordinated checkpointing in mobile computing systems. In *Proceedings of the 8th IEEE Symposium on Computers and Communications (ISCC '03)*, pages 1427–1434. IEEE, June 2003.
- [43] Angin O., Campbell A.T., Kounavis M.E., and Liao R.R.-F. The Mobiware Toolkit: Programmable Support for Adaptive Mobile Networking. *IEEE Personal Communications Magazine, Special Issue on Adapting to Network and Client Variability*, August 1998.
- [44] OMG. Wireless access and terminal mobility in corba, 2002. http://www.omg.org/technology/documents/formal/telecom_wireless.htm (Last visited February 2005).
- [45] Sinha P., Venkitaraman N., Sivakumar R., and Bhargavan V. WTCP: A Reliable Transport Protocol for Wide Area Networks. In *ACM Mobicom'99, Seattle*, 1999.
- [46] H.-U. Park and I.-Y. Lee. A digital nominative proxy signature scheme for mobile communication. In *Proc. of Information and Communications Security (ICICS'01)*, volume 2229 of LNCS, pages 451–455, 2001.
- [47] Charles E. Perkins and David B. Johnson. Mobility support in ip. In *Mobile Computing and Networking*, pages 27–37, 1996.
- [48] E. Pitoura and G. Samaras. *Data Management for Mobile Computing*. Kluwer Academic Press, 1998.
- [49] Herman Chung-Hwa Rao, Di-Fa Chang, Yih-Farn Chen, and Ming-Feng Chen. iMobile: a proxy-based platform for mobile services. In *ACM Workshop on Wireless Mobile Internet (WMI01)*, Rome, pages 3–10, July 2001.
- [50] Ronald L. Rivest and Butler Lampson. SDSI - A simple distributed security infrastructure. Presented at CRYPTO'96 Rumpsession, 1996.
- [51] P. Rodriguez, S. Mukherjee, and S. Rangarajan. Session level techniques for improving web browsing performance on wireless links. In *Proc. of the World Wide Web (WWW) Conference, New York*, May 2004.

- [52] H.K. Rubinsztein, M. Endler, and N. Rodrigues. A Framework for Building Customized Adaptation Proxies for Mobile Computing. Monografias da Ciência da Computação 22/05, PUC-Rio, Departamento de Informática, 2005.
- [53] M. Satyanarayanan, J. J. Kistler, P. Kumar, M. E. Okasaki, E. H. Siegel, and D. C. Steere. Coda: A highly available file system for a distributed workstation environment. *IEEE Transactions on Computers*, 39:447–459, 90.
- [54] B. Schen, S.-J. Lee, and S. Basu. Streaming media caching with transcoding-enabled proxies. In *Proc. of the 6th Int. Conference on Internet and Multimedia Systems and Applications (IASTED), Kauai*. (also available as HP Labs Technical Report HPL-2002-210R1), August 2002.
- [55] Seung-Hyun Seo and Sang-Ho Lee. New nominative proxy signature scheme for mobile communications. In *International Conference in Security and Protection of Information, Brno, Czech Republic*, pages 149–156, April 2003.
- [56] C. Spyrou, G. Samaras, E. Pitiura, and P. Evripidou. Mobile agents for wireless computing: the convergence of wireless computational models with mobile-agent technologies. *Mobile Network and Applications*, 9(5):517–528, 2004.
- [57] Latanya Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, 2002.
- [58] Douglas B. Terry, Karin Petersen, Mike J. Spreitzer, and Marvin M. Theimer. The case for non-transparent replication: Examples from Bayou. *IEEE Data Engineering*, pages 12–20, December 1998.
- [59] Jim Waldo and the Jini Technology Team. *The Jini(TM) Specifications*. The Jini Technology Series. Addison-Wesley, 2nd edition, 1999.
- [60] Z. Wang, S. Das, H. Che, and M. Kumar. A Scalable Asynchronous Cache Consistency Scheme (SACCS) for Mobile Environments. *IEEE Transactions on Parallel and Distributed Systems*, 15(11), November 2004.
- [61] OPES WG. Open Pluggable Edge Services. Technical report, IETF, 2003. www.ietf.org/html.charters/opes-charter.html (Last visited: March 2005).
- [62] Bin Yao and W. Kent Fuchs. Proxy-based recovery for applications on wireless hand-held devices. In *SRDS '00: Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems (SRDS'00)*, page 2. IEEE Computer Society, 2000.
- [63] Bruce Zenel. A general purpose proxy filtering mechanism applied to the mobile environment. *Wireless Networks*, 5(5):391–409, 1999.