Journal of Universal Computer Science, vol. 5, no. 7 (1999), 390-417
submitted: 18/2/99, accepted: 28/6/99, appeared: 28/7/99 © Springer Pub. Co.

A Survey of Web Architectures for Wireless Communication Environments

Stathes Hadjiefthymiades
(Communication Networks Laboratory
Department of Informatics, University of Athens, Athens, Greece
shadj@di.uoa.gr)

Lazaros Merakos
(Communication Networks Laboratory
Department of Informatics, University of Athens, Athens, Greece
merakos@di.uoa.gr)

Abstract: The WWW is currently considered as the most promising and rapidly evolving software platform for the deployment of applications in wide area networks (telematics) as well as enterprise intranets. Another technology that proliferates rapidly in such environments is wireless communication networks (e.g., GSM, wireless LANs). The combination - merging of the two technologies (i.e., mobile computing based on the WWW) is considered of major importance to the computing industry for the forthcoming years. A survey of the research activities undertaken in this area over the previous years is reported. The different approaches that have been proposed to overcome the associated technical difficulties at different layers of the OSI reference model (e.g., the application layer, the transport/network layer) are described and compared. The presented architectures incorporate mechanisms and ideas extensively used in the area of mobile computing (e.g., caching, pre-fetching).

Key Words: WWW, wireless networks, mobile computing, caching, pre-fetching, proxy.

Category: H.5, C.2, D.4

1 Introduction

Currently, the WWW [Ber94a] is considered as a very promising technology that is used extensively for the deployment of applications in the Internet and corporate intranets. This client/server information system, conceived in the early 90's, owes its great success to the standardisation of the communication between clients (browsers) and information servers (WWW servers). The three open standards which are primarily involved in such communication are: the HyperText Transfer Protocol (HTTP), the HyperText Markup Language (HTML) and the Universal Resource Identifiers (URI) addressing scheme.

Apart from the developments in the software area, during the 90's, we also experienced tremendous advances in the area of wireless personal communications. The European cellular system GSM (Global System for Mobile Communications) [Mou95] achieved impressive acceptance and spread rapidly over the globe. Office environments and small industrial installations benefited from the introduction of the

Digital European Cordless Telecommunications standard (DECT, ETS 300 175). More sophisticated applications, enriched with multimedia capabilities, were possible with the HIPERLAN (High Performance Radio Local Area Network) standard [Hal94]. ATM based wireless LAN prototypes emerge constantly [Kal98], [Eng95].

The growth of wireless telecommunications stimulated the interest for the so-called anywhere - anytime computing. This type of computing, also known as “nomadic computing” [LaP96], aims to provide users with access to popular desktop applications, applications specially suited for mobile users and basic communication services. The emergence of nomadic computing was also facilitated by the rapid proliferation of portable computing equipment (portable PCs, portable digital assistants). Wireless mobile-computing is a very challenging area due to:

- the low data rates that are usually available to the wireless equipment. On wide area infrastructures like the GSM/GPRS system rates are usually in the order of a few tens or hundreds of kilobytes (shared between the users of a cell). Moreover, wireless connections suffer a high variability in terms of bandwidth.
- the reduced reliability of wireless connections. Wireless connections are significantly less reliable than wireline connections. Wireless connections, either on the LAN or WAN scale, may be interrupted for various reasons (e.g., handovers, low signal conditions, time-outs in the operation of higher layer protocols, etc.).
- the increased cost for a wireless data connection. The cost per byte transmitted over the wireless interface is considerably higher (orders of magnitude) that in conventional wireline infrastructures.
- the need for the provision of location dependent information. Such requirement may be fulfilled by technologies such as the Global Positioning System (GPS) or by exploiting capabilities of the wireless infrastructure (e.g., the GSM system [Liu98]).

The above mentioned problems and many others (e.g., security, addressing, user interfaces) are discussed extensively in [For94] and in [Duc92].

During the past years a number of efforts were made to consolidate the WWW with wireless communication architectures. The use of the WWW platform in wireless infrastructures extends the challenges discussed above due to the overheads associated with the application/use of HTTP. Specifically,

- HTTP is a ASCII-based, readable protocol which assumes sufficient capacity over the communication channel.
- due to the protocol’s stateless nature, HTTP message headers usually contain the same set of directives (header fields).
- Lastly, the requirement of HTTP/1.0 for a single TCP connection per resource retrieval significantly undermines the performance of the WWW when bandwidth is a scarce resource.

The WWW-wireless consolidation efforts can be classified into two main classes on the basis of the criteria discussed in [Bak94]. The first category of architectures aims at maintaining existing WWW software intact. The peculiarities of the wireless environment are not exposed to the application (mobile-transparent applications) due to the introduction of a proxy/mediator - an application which receives HTTP queries, deals with the unreliable communication service and tries to service the submitted requests. In several architectures, such mediators have their counterparts operating in the wired part of the network (e.g., in the base stations). In those cases, proprietary protocols may be used for the communication of the two proxies properly designed to alleviate the performance handicap of conventional HTTP (i.e., ASCII-based, readable protocol, stateless orientation, requirement for reliable transport like TCP). The application transparent approach for mobile computing has been adopted in the well-known Coda file system [Kis92].

Other efforts shift the mobile environment awareness within the WWW application. Usually, such architectures do not impose changes to the HTTP communication (i.e., no mediators are introduced). Caching and pre-fetching may be applied to improve the quality of WWW service experienced by users. The architectures expand existing WWW software like Netscape and Mosaic browsers and changes are proposed for other components of the WWW platform (e.g., the HTML or the URL addressing scheme). This last category of applications is generally referred to as mobile-aware.

Apart from the work presented in this paper, other modifications are also needed in the protocol stack of mobile terminals in order to expedite the operation of the service in a wireless environment. Such modifications mainly refer to the underlying IP and TCP layers. IP needs to be able to support user roaming (i.e., Mobile IPv4 [Per97], [Per96], Mobile IPv6 [Joh98]) if mobility across different sub-networks is envisaged. TCP, on the other hand, suffers substantial problems when operating in a wireless environment [Cac95].

The paper is structured as follows. At first, we try to set the scene for the survey by brief introductions to the basic components of the WWW platform namely the HTTP, the URL addressing scheme, and the HTML. Section 2 is devoted to this brief introduction. Section 3 comprises the short overviews of the considered architectures for WWW in wireless communication environments. Section 4 contains a comparative analysis of the different approaches and tries to classify the architectures into classes with similar technical characteristics. Section 5 identifies the directions along which relevant research is currently conducted. Section 6 concludes this paper.

2 The building blocks of the WWW technology

This section provides a brief introduction to the basic components of the WWW technology: the HTTP, the URL addressing scheme and the HTML. Additionally, we provide a very brief introduction to the Common Gateway Interface (CGI) specification, the mechanism used for interfacing legacy information systems or services (i.e., mail, RDBMS) to the WWW.

2.1 HyperText Transfer Protocol

HTTP is an "application-level protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems" [Ber96]. The protocol defines the rules for information exchange (e.g., message structures, network connection handling) between the clients and servers of the WWW service. It has been in use since the very early stages of the WWW. The HTTP is based on simple request/response "conversations". Clients request the establishment of network connections with information servers. Using such connections they transmit requests consisting of: a request method, the required information resource in the form of a URL, and the protocol version, followed by a MIME-like message containing request modifiers, client information, and possibly query data. The server responds with a message comprising a status line and a MIME-like message containing server information, resource meta-information, and, lastly, the required resource. In the context of the Internet, HTTP communication generally takes place over TCP connections. This does not preclude HTTP from being implemented on top of any other transport layer protocol. The protocol only presumes a reliable transport so any protocol that provides such guarantees can be applied. The newest version of the protocol [Fie97] has been subject to optimisation for surpassing some of the problems associated with the operation of the 1.0 version on top of existing TCP implementations (i.e., slow start, 3-way handshake) [Pad95].

2.2 Uniform Resource Locators

The specification of Uniform Resource Locators [Ber94b] refers to the syntax needed for addressing resources that are accessible through existing Internet oriented protocols/services (e.g., FTP, HTTP, Gopher). The introduction of the addressing scheme was motivated by the abundance of information systems dispersed over the Internet and their diverse resource addressing syntax. URLs aim at providing a Universal Set of Names and Addresses intended for the uniform handling of this world-wide set of information resources. Practically, a URL comprises the protocol for the communication with the remote server (e.g., HTTP, FTP), the identification of the server, the identification of the wanted resource within the server, and may be augmented by user identification information as well as query information.

2.3 HyperText Markup Language

The HyperText Markup Language provides the means for the formation of hypertext in the context of the WWW. The language enables the specification of the structural components of a document (e.g., paragraphs, headers). WWW clients, at a later stage, interpret such definitions and render readable documents. The language has been designed on the grounds of the SGML meta-language (ISO 8879:1986 Information Processing - Text and Office Systems). The first versions of the HTML specification enabled, through the use of tags, the structuring of a document (e.g., title, headers, paragraphs), the specification of text enhancements (i.e., bold, italic) as well as the definition of hypertext (i.e., pointers to other HTML documents that contain logically related information and may reside anywhere within the WWW/Internet). In version 2.0 of the language [Ber95], the primitives for supporting user interaction were incorporated (i.e., HTML FORMs). Recently, the World Wide Web Consortium (W3C) has released a working draft presenting a proposal for a new version of HTML (ver. 4) [Rag97]. This new version encompasses many novelties and many of the features introduced by commercial organisations like Microsoft and Netscape during the lifetime of the WWW.

2.4 Common Gateway Interface

The Common Gateway Interface (CGI) is a mechanism initially integrated in the httpd WWW server developed at NCSA, University of Illinois at Urbana Champaign. CGI specifies how executable files (called scripts), providing gateway functionality to legacy information services/systems (i.e., full-text retrieval systems, RDBMSs) or new applications, can be spawned by the WWW server and how information can be exchanged between them and the WWW server. Such gateway functionality has proved very important to the evolution of the WWW since almost none of the conventional information systems were compatible with the standards that the WWW global information initiative has introduced [Ber94a]. Since the early days of the WWW, many companies like Microsoft and Netscape have introduced their own, proprietary alternatives to CGI (e.g., NSAPI). Currently, the 1.2 version of the CGI specification is in the Internet Draft status [Coa98].

3 WWW architectures for wireless communication environments

This section incorporates short overviews of various approaches for the efficient introduction and use of the WWW service in wireless communication environments. The eight architectures presented below include: the IBM WebExpress system, the Mobisaic system, the Mobiscape system, MIT's Rover Toolkit and Dynamic Documents. Apart from those systems, we also discuss the Wireless Application Protocol (WAP) architecture specification.

3.1 The WebExpress system

The WebExpress [Hou96], [Hou98] system has been implemented by IBM and is part of the company's eNetwork Wireless family of wireless/mobile products. The WebExpress architecture adopts the so-called "interception" approach for reducing the data volume and latency for communication over the wireless interface. WebExpress is the first effort to make provision for wireless WWW applications involving transaction processing.

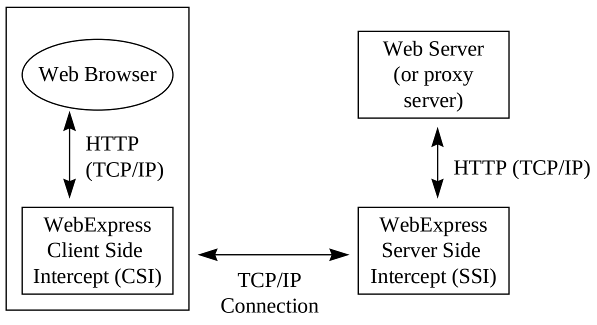

Web Browser
HTTP (TCP/IP)
WebExpress Client Side Intercept (CSI)
Web Server (or proxy server)
HTTP (TCP/IP)
WebExpress Server Side Intercept (SSI)
TCP/IP Connection

Figure 1: WebExpress intercept architecture

The basic architectural model of WebExpress is shown in Figure 1. The Client Side Intercept (CSI) runs in the end-user client mobile device. The Server Side Intercept (SSI) runs in the fixed network. The CSI intercepts HTTP requests posed by a standard WWW browser. Together with the SSI it performs several optimisations with the purpose of reducing the exchange of information over the wireless interface. CSI acts as a local proxy agent co-located with the requesting browser. The CSI communicates with the SSI over a TCP connection using a reduced version of HTTP. The SSI is responsible for reconstituting a URL compatible data request and forwarding it to the designated WWW server. On the reverse direction, the CSI reconstitutes the HTML data stream and delivers it to the local browser. Both the CSI and the SSI incorporate caching mechanisms with LRU discipline. If the requested object is found in one of the two caches, it is immediately returned to the browser. The CSI first checks its local cache and if the document is found stale (decision is based on a coherency interval, CI, value) a protocol is executed between the CSI and the SSI. The latter checks its cache and if the resource is still considered outdated it is retrieved. A new digital signature is calculated and if it is found similar to that of the previous instance, the resource is not transmitted over the radio interface but instead the CSI is notified of the event.

Another optimisation used in the WebExpress architecture is differencing. The application of the differencing technique also has been proposed for the normal operation of WWW software in wired networks [Mog97]. In WebExpress differencing is applied in responses containing dynamic content (i.e., HTML output produced by CGI scripts). A common base object, associated with the resource, is created in both the CSI and the SSI. Subsequent references to the resource will trigger, at the SSI, the computation of differences between its present form and the common base object. Such differences are transmitted over the radio interface. The CSI reconstitutes the referenced object and delivers it to the browser through the proxy interface. HTML byte streams representing query responses usually contain a lot of unchanging formatting information (e.g., graphics, headers, footers). Responses from the same script are usually differentiated only by alphanumeric information.

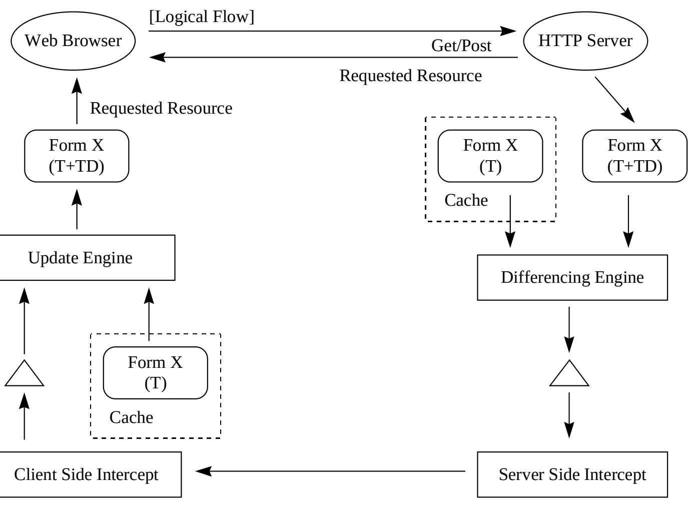
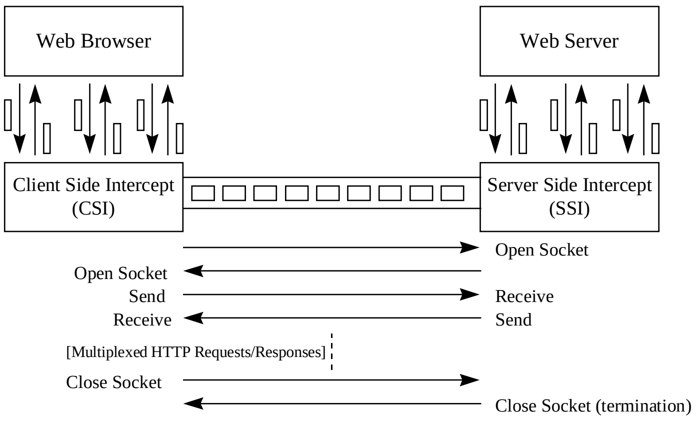


Figure 2: Differencing technique applied to dynamic resources

More specifically, when a request for a CGI script is posted (at time T) to the CSI, the latter detects that and searches its local cache. We assume that the cached copy is not found so the request is relayed to the SSI and subsequently to the designated server. The SSI caches the retrieved resource (and calculates a CRC value for it) and forwards it to the CSI which also performs caching. This cached copy is the common base object. At time T+DT another request for the same CGI script is intercepted at the CSI where a cached copy also exists. The CSI forwards the request to the SSI along with the CRC value of the base object. The HTTP request is relayed to the designated server. When the response is received back at the SSI, a differencing engine is invoked to compute the differences between the recently received response and the cached base object (Figure 2). A report is produced (also termed differencing stream) which consists of a series of copy and insert commands. Such report is sent to the CSI which then performs, on its basis, the required changes on the locally cached base object and delivers the resource to the browser for rendering. Using this approach, the updated resource is not transmitted over the radio interface.

A single TCP connection between the CSI and the SSI is used for all HTTP communication (i.e., requests and responses are multiplexed over this connection). The single TCP connection is persistent between different transactions. It alleviates the deficiency of HTTP's operation on top of TCP [Pad95] which would have much more negative effects over a wireless interface. WebExpress adopts a technique called *virtual sockets* to provide multiplexing support over the unique TCP connection. Virtual sockets enable the CSI to exchange, through the TCP connection with the SSI, HTTP messages pertaining to multiple Web interactions. Data sent for a given request is prefixed by a small header that contains a virtual socket id, a command byte and a length field. At the CSI the virtual id is associated with a normal socket to the browser (over the loopback interface). On the fixed network side, the virtual socket is associated to a normal socket on the designated HTTP server. Some of the established virtual sockets are used for conveying commands related to normal socket calls (e.g., open, close, etc.) as a form of inband signalling.



Web Browser
Web Server
Client Side Intercept (CSI)
Server Side Intercept (SSI)
Open Socket
Open Socket
Send
Receive
Receive
Send
[Multiplexed HTTP Requests/Responses]
Close Socket
Close Socket (termination)

Figure 3: Multiplexing of HTTP connections over one TCP connection

Additionally, some of the HTTP header fields (e.g., Accept lists), which are usually constant for a specific browser, are cached once in the SSI and not transmitted continuously by the CSI. The CSI allows the header information to flow towards the SSI only in the first request. Both the CSI and the SSI maintain the Accept list as part of the connection state information. When a new request reaches the CSI, the Accept list is checked to verify that it is identical to the CSI stored copy. If yes, it is removed from the request but subsequently inserted by SSI. This technique is referred to as HTTP Header reduction.

3.2 The Mobisaic system

Mobisaic [Voe94] is an extended WWW client/server architecture capable of supporting mobility in two ways. It allows WWW authors to reference/retrieve dynamic information, such as the user's location, in hypertext links called *Dynamic URLs*. When a dynamic URL is invoked, the client resolves any references to dynamic information it may contain and communicates the result to the server. Additionally, Mobisaic supports Active Documents. Active documents present and automatically update information for the user as the information they contain changes over time or becomes invalid. Such update is performed by the client software on the basis of notifications (callbacks) received by specialised server software when the designated dynamic information changes.

To facilitate the use of dynamic information, Mobisaic extends the conventional Web infrastructure to include:

- **a network server that maintains mobile computing contexts within a client-specific domain**. The basic unit in a dynamic environment is the dynamic environment variable, which is quite similar to standard UNIX shell variables. Variables can be used to customise applications to the user's mobile computing environment just as shell environment variables customise applications.
- **an asynchronous call-back mechanism to notify Web clients when a user's dynamic computing environment changes**. For example, the wireless communication system that monitors the user's location can publish the new location by updating the Location variable in the user's dynamic environment. If the client was displaying a document that was sensitive to location, the client would have subscribed to the Location variable and would receive a notification informing it about the change.
- **a syntax for referencing dynamic information in URLs and documents**. The extended syntax is of the form $(*environment.variable*) where environment and variable denote a user's environment and a dynamic variable respectively (e.g., $(*stathes.Location*) would reference Stathes' current location within the wireless access network).

Dynamic URLs allow a single URL to access different WWW resources (e.g., static pages, CGI scripts), depending upon the state of the user's dynamic environment at the time the query is posed. An example of a dynamic URL is: *http://www.my.server/places/$(Location).html*. An invocation of this URL will retrieve the document 433.html if the user is currently in room 433 (value of the Location dynamic variable). Similarly, the reference to the dynamic variable can be passed as a parameter to a CGI executable.

Another novelty introduced in the Mobisaic architecture is that of Active Documents. Active Documents are HTML resources that enable the WWW client to automatically react to changes in the user's mobile context. The client is capable of updating the information being displayed without the need for additional user interaction. Active Documents should contain the *subscribe to* command embedded in HTML comments[1] (i.e., `<!- (subscribe to $(Location)) ->`). Such commands instruct the client to subscribe to specific variables contained in the users' dynamic environment. When such variables change values, clients receive notifications with the new values. When the client receives a notification for changes in a subscribed variable, the new value determines the kind of action the client should undertake. The new value can be an explicit directive to the client:

- reload: forces re-execution of the URL that loaded the current document.
- load URL: forces retrieval of a new URL
- spawn URL: forces retrieval of a new URL in a new client window
- close: closes the current window

[1] Commands embedded in HTML comments allow documents to be backward compatible, interpretable by any browser.

In terms of implementation, Mobisaic has been based on the Mosaic client which was slightly extended to deal with the peculiarities of the proposed architecture. Within the client software, filters have been programmed to deal with dynamic variable references and subscriptions. Filters in the output data stream aim at resolving references to dynamic variables. Filters in the input stream recognise subscription commands within the HTML code and proceed with the required interaction with the network server.

The Zephyr notification system [Del88] has been used as subscription server which holds the information on mobile user dynamic contexts. Software of the communication subsystem is assumed to monitor changes pertaining to mobile terminals (e.g., handovers) and "publish" such information in the Zephyr system.

3.3 The MobiScape system

The MobiScape architecture [Baq95] adopts the interception approach similarly to the WebExpress work. The architecture is shown in Figure 4.

The MobiScape project capitalises on the proxy interface that most modern navigation tools (e.g., Communicator, Internet Explorer, Mosaic) support. The Mobile Host (MH) uses the Support Station (SS) as a gateway to the WWW. A caching mechanism is provided in both the MH and the SS in order to minimise the periods of connection between the two machines. The cache in the MH facilitates disconnected operation. The cache in the SS reduces the wait periods experienced when fetching remote documents. Thus, the HTTP data flow between the browser (MH) and the HTTP server is intercepted twice (in the MH and the SS). Both the MH and SS cache servers operate as proxies [Luo94] with simple LRU disciplines. The data flow between the SS proxy and the MH proxy is compressed (CL: Compression Layer). The profilers, which support the operation of both proxies, consult user-defined scripts and trigger the pre-fetching of a series of "should-always-be-in-cache" documents which are very likely to be requested during the user's session. Profiling capabilities are provided in both the SS and the MH. In the SS (which is part of the fixed access network), user profiles can be extended while in the MH (where storage and bandwidth are definitely more expensive) profiles need to be considerably smaller. The profiler in the SS is invoked periodically while the profiler of the MH can only work during periods of full connectivity. Cache servers, shown in Figure 4, manage a multi-level cache buffer. They are capable of distinguishing whether some document was fetched dynamically or by the profiler.

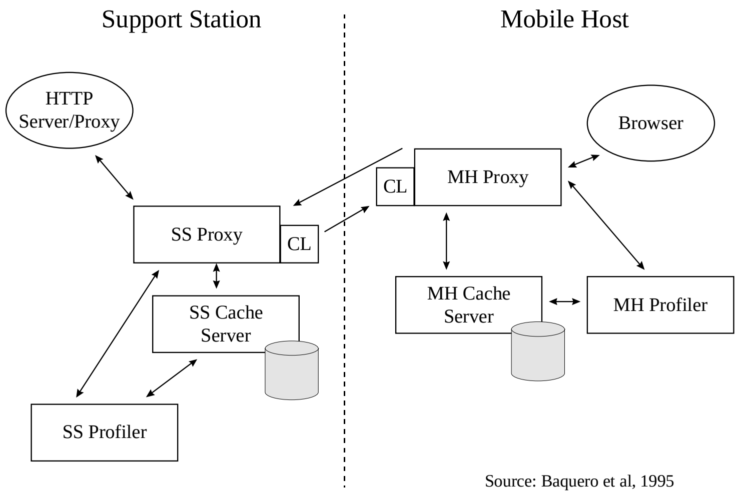


Figure 4: MobiScape Model

Mobiscape components were developed in C, as conventional UNIX processes. The system has been operated in Linux portables having PPP dial-up connections to a campus wide Ethernet infrastructure.

3.4 Pro-active proxy cache relocation in cellular environments

One of the missing elements in the WebExpress and MobiScape architectures, which adopt the HTTP interception technique, is that both schemes do not make provision for the roaming of the mobile user within a cellular environment. If the wireless infrastructure provides for handovers between base stations the SS cache (the terminology of the MobiScape work is adopted) has to be reconstructed each time the mobile terminal crosses the boundaries of a cell. A technical solution has been proposed in [Had98]. In that paper, the SS cache follows the movement of the mobile station. The relocation of the SS cache is performed prior to the realisation of handovers depending on the outcome of movement/path prediction algorithms [Liu95], [Liu97]. Such algorithms provide the means for a pro-active management of resources. The algorithms take into consideration the randomness in the movement of the user as well as the already identified movement patterns. They reach decisions together. The architecture assumes the existence of user profiles stored in specialised nodes within the user's home network (the part of the network to which the user administratively belongs). When the mobile terminal migrates to a sub-network different from its home, the user profile database (home registry) is queried and the relevant information is forwarded to the visited network by means of specialised signalling. It is assumed that the home registry of a mobile terminal also incorporates path prediction algorithms (Figure 5), which provide indications on which cells the terminal is likely to be handed-over. The invocation of the path prediction algorithm can be performed some time after the entrance of the mobile terminal into the current cell. From that point, the base station which the mobile terminal is likely to attach to in the near future (target base station) requests the relocation of the SS cache, which has been gradually accumulated in the current base station.

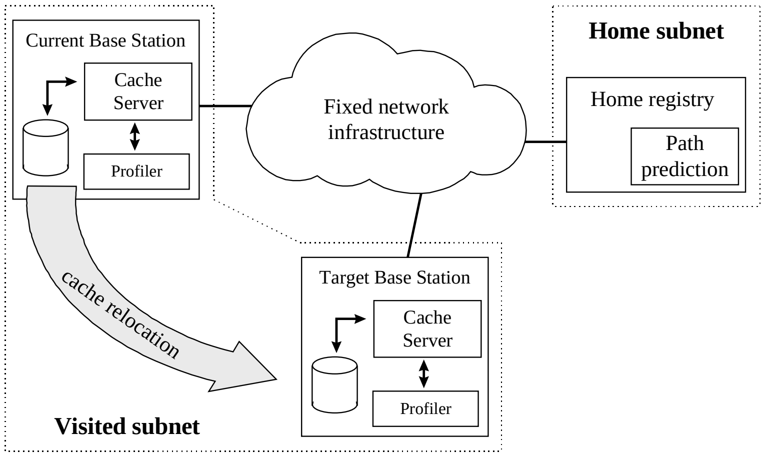


Figure 5: Network components involved in the management of the "moving" cache

The sequence of actions undertaken by various network entities as well as the signalling required for their completion are depicted in the Message Sequence Chart (MSC) of Figure 6.

In Figure 6, the *Determine_Target(MT_ID, BS_ID)* signal is used for triggering the path prediction algorithm in the home registry. Its parameters denote the ID of the terminal for which the algorithm should be executed as well as the identification of the base station. The home registry invokes the path prediction algorithm and notifies, through the *Relocate_Cache(MT_ID, BS_ID, UserProfile)* signal, the base station, which has been selected as candidate (Target BS), on the possibility of terminal's entrance in its control area. The *BS_ID* parameter of the signal indicates the current base station the involved mobile terminal is attached to. The *UserProfile* is intended for the operation of the local profiler. An analogous signal is forwarded by the target base station to the current base station to trigger the relocation of the accumulated cache. As the current base station may handle more than one terminal, the signal conveys the identification number of the involved terminal (*MT_ID*). Upon reception of this last signal, the current base station packs and compresses the part of its cache which has been associated with the terminal. The information is de-compressed by the target base station and fed into its local cache. The whole procedure is executed in parallel to terminal's roaming, without obstructing its communication through the current base station.

At some time after the relocation of its cache, the terminal executes a handoff operation. This operation may result in terminal's attachment to the base station which has been designated by the Path Prediction algorithm or, alternatively to another base station.

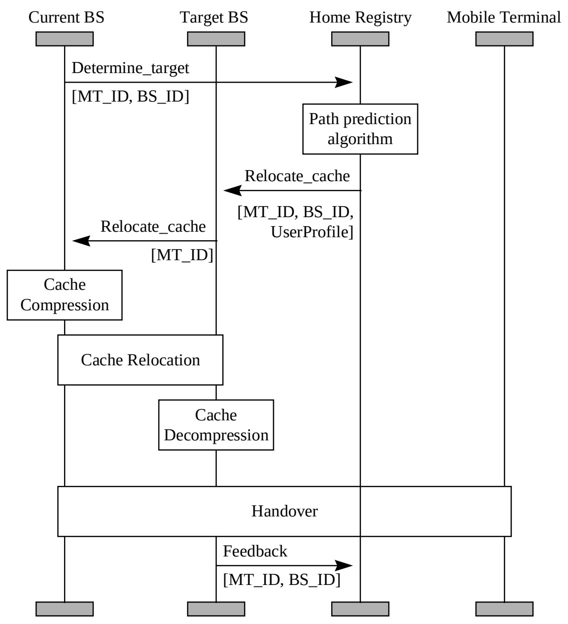


Figure 6: Message Sequence Chart for cache relocation and handover

Simulations of the above architecture, using Pareto-based models for the self-similar WWW traffic [Cro97], have shown that:

- When the efficiency of the prediction algorithm drops to 0% the user perceived average delay is about 5 sec. When the efficiency of the algorithm reaches 95% the delay drops to 3-3.5 sec.
- Lowering the prediction efficiency from 95% to 0% causes the percentage of handoff interrupted connections to double (3.5% $\rightarrow$ 7%). When the prediction algorithm is applied, the base station proxy may shorten the HTTP chain and, thus, cause the WWW query to be rapidly dispatched. As a result, more connections can be dispatched within the time spent by the user in the current cell. Consequently, the percentage of handover interrupted connections drops. Furthermore, reducing the prediction efficiency also causes a significant increase in the absolute number of interrupted connections. This is due to the fact that longer connection duration increases the probability for connection interruption by a handover. The above observations are also very crucial since the HTTP uses TCP as a reliable transport protocol. Apart from the well-known problems in the interaction of the stateless HTTP (ver.1.0) with TCP [Pad95], Caceres and Iftode in [Cac95] have shown how TCP's congestion control mechanism undermines throughput during handovers. The above researchers have quantified the performance degradation in TCP con-

nections caused by MT movement across cell-boundaries. In an overlapping cell scenario, handovers cause the throughput of the TCP connection to decrease by 6%, while in the non-overlapping cell scenario with 0-sec rendezvous delay[2] throughout drops by 12%. TCP connections conveying HTTP messages are generally short-lived (their duration equals the time of the resource transmission time) due to the stateless character of the protocol. Limiting the probability of interruption, by a handover, of a WWW connection will, thus, be beneficial for the performance of the underlying TCP layer.

A newer version of the discussed architecture which takes provision for the misses of the path prediction mechanism is presented in [Had99] along with simulation results.

3.5 Dynamic documents

In the Dynamic Documents [Kaa94] proposal, extensions to conventional WWW browser software are introduced. Dynamic Documents are programs, written in the portable Tcl scripting language, which are transmitted to the browser and interpreted there. Provision is taken for Documents' secure interpretation. The browser software which was customised to deal with Dynamic Documents was NCSA's Mosaic. The resulting software is referred to as *adaptable* Mosaic client. The use of Dynamic Documents allows adaptable clients to adjust to a variety of mobile platforms and network environments.

Practically, Dynamic Documents allow the client to apply **fetching**, **displaying** and **interface processing policies** different from the standard ones, in order to adapt to environmental changes. In a wireless infrastructure, standard policies may cause problems like the unjustified consumption of bandwidth and network overload.

A Dynamic Document may contain both the data of the document as well as a series of scripts to control information presentation. Execution of a Dynamic Document may force the client to perform a number of actions in order to achieve the final presentation; other documents may be fetched, new HTML code may be generated, local information may be accessed or an entirely new interface may be generated. This last feature is supported through the Tk toolkit which enables the building of arbitrary, adhoc interfaces.

In terms of retrieval policies, the Dynamic Documents framework allows the client to apply different strategies (e.g., for pre-fetching or caching) on a per document basis. Similarly, the client may apply different presentation policies on a per document basis (i.e., if the network bandwidth is limited, fetching large multimedia resources may be delayed or executed in the background). Interface processing tasks also may be affected by Dynamic Documents. For example, the processing of fill-out Form [Rag97] contents may be restricted to the client instead of relaying the request to servers and thus consuming precious network bandwidth. Additionally, the client may trigger the generation of new interfaces.

[2] The MT receives control information from the adjacent cell as soon as it leaves the current cell.

As discussed before, Dynamic Documents are written in the Tcl language. Dynamic Documents are based on the callback technique. Callbacks are procedures provided by the Dynamic Document and registered in the Web client software. They are invoked by the client when some special condition occurs (i.e., the decision on what information the user should obtain).

A prototype has been developed for the Dynamic Documents architecture. Such prototype was first tested in a wired network but subsequently ported to a WaveLAN infrastructure (http://www.wavelan.com) with portable computers. In the prototype, similarly to other wireless WWW architectures, the use of a cache memory in the mobile terminal is adopted. This cache is made persistent across HTTP sessions. To maintain the correctness of the cached information, documents are cached only for a specific period of time. Such period is calculated per document on the basis of the previous modification time and the time of document retrieval from the server. A cache hit rate of 40% is reported. Each Dynamic Document may bring its own caching strategy into effect and thus, override the global caching policy. Additionally to the caching technique, prefetching of WWW resources also is applied by the Mosaic client in the Dynamic Documents prototype. Prefetching is temporarily suspended when an HTTP request is in progress so that the full network bandwidth is made available to the client. Prefetching is based on historical access patterns (the client prefetches those documents that have been accessed from the current page in the past).

3.6 The Mowgli architecture

The Mowgli (Mobile Office Workstations using GSM Links) architecture [Lil95], [Koj97] has been designed to increase the usability, reliability and efficiency of client/server communication between fixed and mobile nodes. The basic idea behind the architecture is to split a channel with end-to-end control into two segments by using a store-and-forward interceptor. Such component is termed Mobile-Connection Host (MCH). Practically, two co-operating communication subsystems are deployed: the wireline oriented sub-system and the wireless. The Mowgli architecture is shown in Figure 7.

Reliable communication between the MCH and the Mobile Node (MN) is provided by a specialised transport service, the Mowgli Data Channel Service (MDCS)[3]. Unmodified TCP/IP protocols are used between the MCH and fixed hosts. The communication between the WWW client and a WWW server is intercepted twice; once at the Mowgli Agent in the Mobile Node and once in the Mowgli Proxy, positioned within the MCH. Applications in the Mobile Node connect to the Agent through the Mowgli Socket Interface which is downwards compatible with the classical BSD socket API. The Agent and the Proxy, apart from relaying traffic to and from other layers, incorporate advanced logic for optimising the operation of the supported protocols.

The Mowgli Agent and the HTTP proxy communicate by exchanging messages asynchronously over a long-lived Mowgli data connection. The agent intercepts re-

[3] The MDCS takes care of link disconnections and reconnections.

quests from the local WWW client and forwards them over the wireless interface to the proxy. The proxy, in turn, connects to the appropriate server for each request and asks it to dispatch the posed request. Since an end-to-end TCP connection is no longer established for each HTTP request, the round-trip delays over the wireless link cease to be a performance factor.

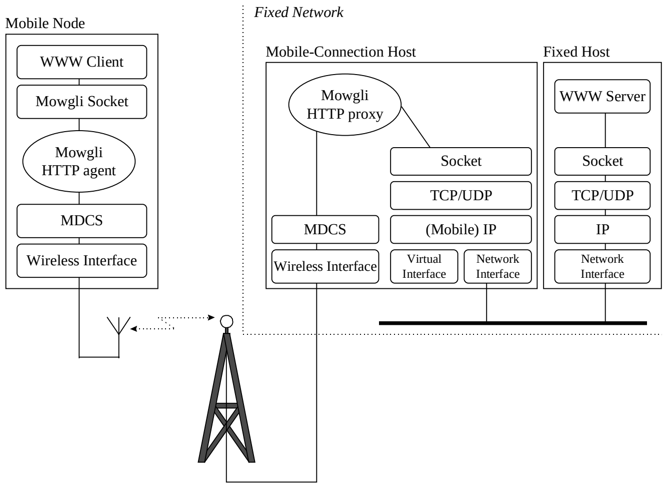


Figure 7: Mowgli Communication Architecture

The proxy also performs some optimisation in the retrieval of documents containing inline images. The proxy parses the document, identifies references to images and starts retrieving them, without explicit request from the agent, in parallel to the document retrieval. Subsequently, the proxy is capable of immediately dispatching requests pertaining to the specific inline images. To reduce the DNS queries, which are also another source of delay for the WWW, a special Mowgli DNS Agent is also introduced. Such agent, in collaboration with the proxy, may pre-fetch addresses that are likely to be needed in the near future.

The volume of information exchanged between the mobile node and the fixed network is reduced by using data compression and caching. A reduced version of the protocol headers is used in the communication between the agent and the proxy. Additionally, transferred data can be compressed either in a lossless or in a lossy way. The proxy also acts as an “intelligent” filter; if the size of images exceeds a user defined threshold, the proxy replaces it with a default small-size image. To further optimise the operation of the HTTP protocol, the agent stores all incoming objects into its local cache to restrict the need for transfers over the wireless interface. If the requested resource is not found in the cache, the request propagates to the HTTP proxy as with the WebExpress and MobiScape architectures.

Some asynchronous operations can be performed by the agent. Prefetching of documents in the cache is one technique which entails drastic improvements in the response times experienced by users. Pre-fetching is performed as a background task.

Users are not obliged to explicitly identify which documents are to be prefetched. This is a decision taken by the agent and the proxy. A prototype consisting of Mowgli HTTP Agent and Proxy has been built on the Linux operating system.

3.7 The Rover Toolkit

The Rover toolkit [Jos97] has been implemented in M.I.T. (Laboratory for Computer Science) for the facilitation of mobile application development. The toolkit is capable of supporting the two main orientations in mobile computing (i.e., proxies/mediators for mobile transparent applications or mobile-aware applications [Bak94]). The toolkit architecture follows the client/server model. Clients are Rover applications in mobile nodes. Servers execute on the fixed network and maintain system state information. Rover is based on two basic components: the **Queued Remote Procedure Calls** (QRPC) and the **Relocatable Dynamic Objects** (RDO). A relocatable object is an object-like combination of data and code (currently implemented in Tcl/Tk) that can be loaded from a server node and executed on a client node (or vice versa) with the purpose of reducing the communication requirements over the unreliable radio interface. RDOs can be subject to caching to further improve the perceived service quality. QRPC is a enhanced communication mechanism, based on the RPC model, which is properly structured to deal with network disconnections - failures. QRPCs may be scheduled as needed, compressed and manipulated in batch mode.

RDOs provide increased flexibility since they enable the system to dynamically re-distribute the computational burden (i.e., the client may request the server node to perform some processing on its behalf) depending on the workload of the mobile node or the available bandwidth in the wireless network. This is facilitated by the co-packaging of data and code in RDOs.

In terms of implementation, the Rover toolkit is based on four components (Figure 8) namely the access manager, the object cache, the operation log and the network scheduler. Apart from the object cache, all other entities can be found in both the clients and the servers of the architecture. Failure recovery is also the responsibility of the access manager. The access manager handles all client-server or client-client interactions. RDOs are retained in the object cache. QRPCs are registered in the operation log. Finally, the network scheduler manipulates the entries of the operation log, groups and re-schedules the involved QRPCs, and performs all network transactions on the behalf of other components.

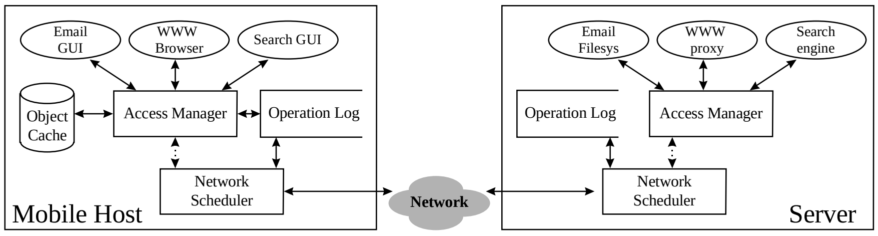


Figure 8: Rover Architecture

Rover has been used for the development of Web application proxies[4] which are capable of acting as mediators between an unmodified client application (e.g., Netscape Navigator) and the unreliable network interface. The Rover HTTP proxy is compatible with almost all WWW browsers. It intercepts all HTTP requests, consults the local cache and if the resources are not found there, denies the document to the browser (null response) and registers the request in the operation log (Figure 8). When network communication resumes the requested resources are automatically retrieved. In the meantime, the mobile user may continue browsing the contents of the local cache. The client and server modules collectively perform pre-fetching of WWW resources.

Experiments have shown that, if Netscape was used in conjunction with Rover HTTP proxy, the time needed to retrieve and display 10 HTML pages was less than in the standalone Netscape scenario, in network environments like 128/64 Kbps ISDN or 9.6 Kbps over a cellular link. This behaviour can be attributed to the scheduling (i.e., batch handling) and compression techniques applied to QRPCs. In faster networks like Ethernet or AT&T WaveLan, the performance of the brower/proxy system was slightly worse that in the browser standalone scenario.

3.8 The Mobile Application Support Environment (MASE)

MASE [Cha98] has been defined and implemented in the context of the **OnTheMove** project which, in turn, is realised in the context of the EU Advanced Communications Technologies and Services (ACTS) program. MASE practically is a mobile middleware architecture for the provision of multimedia services over a range of network technologies. MASE adopts the mobile transparent approach discussed before, by masking the problems encountered in the wireless infrastructure (e.g., link outages, disconnection, time-outs) from the application. MASE introduces an independent Session Manager capable of performing typical session management functions like session initiation, maintenance, termination, recovery and disconnected operation. The presence of MASE is required in the mobile terminal as well as the **Mobility Gateway**, a node of the fixed infrastructure with a special role. The architecture implemented by OnTheMove is shown in Figure 9.

[4] All application code and data manipulated by the application are in the form of RDOs.

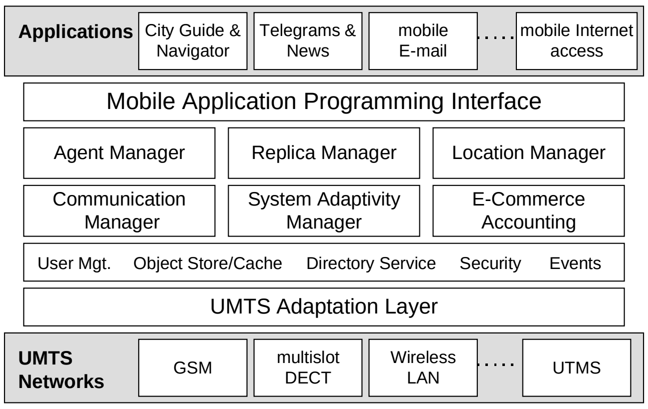


Figure 9: MASE Architecture

The Session Manager is embedded in the Communication Manager module shown in Figure 9. The same module incorporates proxies for E-mail and HTTP services. The Session Manager accomplishes the isolation of application sessions from transport protocol (i.e., TCP) connections. The Session manager implements a proprietary protocol for the provision of its services. Messages from applications are segmented, marked, transmitted and acknowledged by communicating entities (checkpointing procedure). Specialised messages are introduced for communication resynchronisation in the event of link outages.

Applications use a Session Sockets API (quite similar to the BSD Socket API) in order to take advantage of the session management features. Session Sockets terminate at the Mobility Gateway. There an interworking function (IWU) associates the Session Sockets with conventional transport connections in the wired part of the infrastructure. Thus, an approach similar to I-TCP [Bak95] is adopted but on a higher layer within the protocol stack. An application connection to/from a fixed node is broken into two connections; one from the mobile client to the gateway and one from the gateway to the designated host (if different from the gateway).

3.9 Wireless Application Protocol (WAP)

The main objective of the WAP Forum[5] is the efficient introduction of Internet/Web services into the constantly developing wireless telecommunication domain. In this respect, the Forum has drafted a set of specifications collectively known as WAP 1.1. WAP 1.1 refers to a micro-browser architecture; a stack of content transfer protocols, application framework and content formats. Notable among the components of the WAP 1.0 specification are the Wireless Markup Language[6] (WML), the Wireless Session Protocol (WSP), the Wireless Transaction Protocol (WTP) and the Wireless Datagram Protocol (WDP). The work of WAP Forum is independent of wireless technologies and thus, can be applied in a wide spectrum of platforms like GSM or the upcoming CDMA technology.

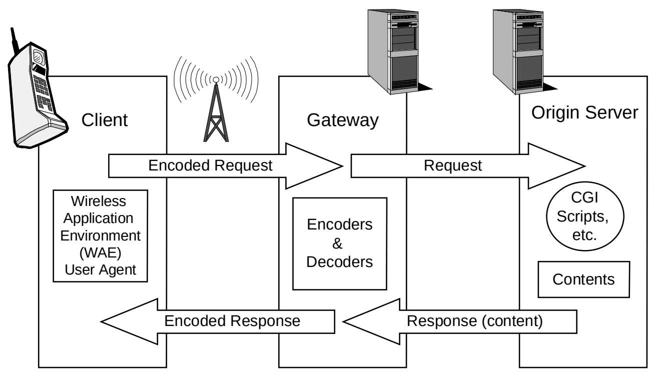


Figure 10: WAP application model

WAP's application model (Figure 10) is quite similar to that of the WWW. Information resources can be retrieved in miscellaneous formats using the URL addressing scheme (named data objects). WAP is largely based on a proxy entity operating within the fixed part of the network. Such proxy incorporates functionality for:

- **Protocol Conversions**. Requests coming from the WAP protocol stack (i.e., the WSP, the WTP, the WDP) are translated into protocols of the WWW stack (i.e., HTTP, TCP/IP) and vice versa.
- **Content coding/encoding**: WWW content is translated into "compact" forms with the purpose of reducing the data volume exchanged over the wireless interface. This resource "distillation" technique has been also studied in [Fox96] and [Sch96].

[5] The WAP Forum was founded in 1997 by Ericsson, Motorola, Nokia and Unwired Planet.
[6] WML is based on W3C's XML specification.

The adoption of the WAP proxy entity allows the reuse of existing applications that have been developed using technologies like CGI and are, presently, hosted by conventional WWW servers.

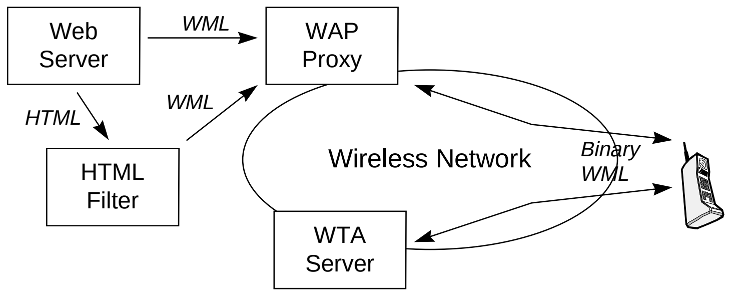


Figure 11: Delivery of compact media forms to the wireless device

Figure 11 demonstrates the use of the WAP proxy for the delivery of a resource (either HTML or WML page) to the wireless device. The HTML resource is translated into WML and then fed to the WAP proxy. The proxy converts the WML input (retrieved either by a server or by a format converting filter) into a binary representation (binary WML) which can be interpreted by the micro-browser of the wireless device.

A WAP micro-browser has already been implemented in Nokia's 7110 dual-band cellular phone [http://www.wapforum.org/new/Nokia6_30_99.htm].

4 Synopsis of architectures

In this section, we compare the different architectures presented in the previous section of the paper. We have identified the main features incorporated in the presented works, and have listed them in Table 1, which assists in the comparative analysis of the different systems. In Table 1, the first row lists the considered architectures while the first column includes the most important characteristics encountered in the course of this survey.

Two of the presented architectures, namely the MASE and the Rover Toolkit are not confined to the Web technology but are more general architectures for mobile computing. The interception technique is very popular among the considered systems. It accomplishes substantial performance gains and also allows modified (optimised) versions of protocols to be used over the wireless medium, while maintaining compatibility with their wireline counterparts. The verbose character of HTTP, which often has been blamed for reduced service performance, has forced protocol reduction in some of the architectures (i.e., WebExpress). Additionally, some of the architectures adopt compression as a method of reducing the data volume exchanged over the wireless interface. The application of both techniques is assessed under the criterion titled "Standard HTTP over the radio interface". The introduction of proxies in the mobile node as well as the access network provides the means for the HTTP communication interception. The MASE architecture specifies the interception of the data stream at the Sockets layer in the Mobility Gateway, a specialised node of the fixed network.

Differencing is an innovation of the WebExpress architecture, which is not found in other systems. The applicability of the technique is studied for the entire Web platform [Mog97]. The multiplexing of HTTP “conversations” over the same transport layer connection also is assessed since it may considerably ameliorate the provision of the Web service to nomadic users. In general, the interaction of HTTP with the underlying transport service has been largely criticised.

Some of the architectures involve the use of new and proprietary HTML/URL syntax for providing mobile dependent functionality. This is an important criterion in our comparison since it categorises the relevant architecture in the mobile-aware systems (see last two rows in Table 1). One additional criterion, which affects this categorisation, is the “Modified browser software” criterion. In certain cases, mobile-specific behaviour is accomplished by the browser software not through HTML-embedded commands but through specialised scripts, executed as “satellite entities”.

The call-back technique is used for signalling environmental changes to the browser software. In the Mobisaic cases, this task is performed centrally through the introduction of a mobile context server.

Lastly, we assessed the application of pre-fetching techniques. Various clever approaches have been proposed for assuring the presence, in caches, of a series of resources that are very likely to be requested in the near future. Such pre-fetching mechanisms may operate stand-alone or be triggered by user specific profilers. Notable among the pre-fetching facilities provided in the various architectures is the HTML document parsing in the proxy server and the immediate pre-fetching of referenced resources such as inline images.

In the last two lines of Table 1, we try to classify the various architectures is accordance to the categorisation suggested in [Bak94]. Specifically, we see that WebExpess, MobiScape, the Proactive Cache Relocation, Mowgli and MASE are compatible with existing browser software and hence, belong to the “Mobile transparent application” category. On the other hand, WAP, Dynamic Documents and Mobisaic require proprietary browser software. Rover is compatible with both approaches. The Mobisaic work is not an effort to meet the bandwidth, cost and communication reliability challenges that wireless computing applications introduce (see Section 1). Instead, Mobisaic aims to deliver location dependent information to WWW users, which is also one of the important challenges introduced it this new computing paradigm.

Architecture / Criterion	WebExpress	Mobisaic	MobiScape	Proactive Cache Reloc.	Dynamic Documents	Rover Toolkit	Mowgli	MASE	WAP
WWW specific	Yes	Yes	Yes	Yes	Yes	No	Yes	No	No
Provision for transaction processing	Yes	No	No	No	No	No	No	-	No
Interception technique	Yes	No	Yes	Yes	No	Yes	Yes	Yes[7]	Yes
Standard HTTP over the radio interface	No	Yes	No	-[8]	Yes	No	No	Yes	No
Proxy on the mobile node	Yes	No	Yes	-	Yes[9]	Yes	Yes	No	No
Proxy on the access network	Yes	No	Yes	Yes	No	No	Yes	No	Yes
Differencing	Yes	No	No	No	No	No	No	No	No
Multiplexing of HTTP connections in the radio interface[10]	Yes	No	No	-	No	-	Yes	No	No
Standard HTML/URL	Yes	No[11]	Yes	Yes	Yes	Yes	Yes	Yes	No
Callbacks for mobile environment information	No	Yes	No	No	Yes	No	No	No	No
Proprietary browser	No	Yes	No	No	Yes	Yes/ No[12]	No	No	Yes
Automatic reload of location specific resources	No	Yes	No	No	Yes	No	No	No	No
Mobile context server	No	Yes	No	No	No	No	No	No	No
Pre-fetching	No	No	Yes	Yes	Yes	Yes	Yes	No	No
User profilers	No	No	Yes	Yes	No	No	No	No	No
DNS provision	No	No	No	No	No	No	Yes	No	No
Mobile aware application		●			●	●			●
Mobile transparent application	●		●	●		●	●	●	

Table 1: Comparison of WWW architectures for wireless environments

[7] The TCP connection is intercepted at the Mobility Gateway.
[8] The "-" sign implies that the feature is not applicable or not relevant to the architecture.
[9] Persistent cache
[10] Use of a single, persistent TCP connection (or other kind of transport service).
[11] Active Documents
[12] Depending on the kind of application developed (mobile aware vs. mobile transparent).

Looking at the reviewed systems from the network protocols' perspective, the need for a wireless aware, optimised TCP layer is more important than Mobile IP. Almost none of the architectures (except for the Proactive Cache Relocation and Mowgli) support roaming of the nomadic user, and thus require the routing/addressing mechanisms specified in the latter protocol specification.

The performance of some of the presented architectures is compared against conventional WWW operation over wireless infrastructures in works such as [Hou98], [Koj97], [Had98], [Had99], [Jos97].

5 Directions of research

Nowadays, the area of WWW-based mobile/nomadic computing experiences an unprecedented growth. The introduction of the WAP framework and its adoption on a commercial basis have acted as a catalyst in this growth. Moved by the developments, the W3C has issued the Mobile Access Activity Statement to "ensure that the protocols and data formats of the Web provide an effective fit for mobile devices". Moreover, the W3C has formed the Mobile Access Interest Group as part of its User Interface Domain. A WAP Forum - W3C co-operation white paper has been also published (October 1998) to outline possible areas of co-operation between the two organisations.

There are several issues in the Web-based mobile computing paradigm that attract scientific research. Pre-fetching of WWW resources is one of them. Pre-fetching is being addressed in [Pad96] and in [Jia98] where simulations of the proposed schemes have shown considerable benefits for the WWW user.

The WAP framework has adopted the real-time conversion of resources into formats suitable for transmission over narrow-band wireless links. As already mentioned, this technique has also been addressed, under the terms "real-time distillation", in older works like the [Fox96] and [Sch96].

The study on the use of differencing for HTTP responses is also a very important topic in the area of Web-based mobile computing. This topic has been addressed in [Mog97] where considerable savings (both in terms of data volume and resource transfer times) have been demonstrated. The same topic is also studied in [Ban97].

The W3C is also working on the specification of the HTTP-NG (Next Generation HTTP). The design of this new protocol tries to avoid the deficiencies of its predecessors (e.g., by adopting binary-, instead of ASCII-, encoded messages and caching techniques) [Jan99].

Considerable work is being also performed in the area of transport protocols intended for wireless communication infrastructures [Bal97]. As already discussed, TCP's operation in the context of a wireless interface has proven troublesome. Four approaches have been proposed for improving its performance: Selective Acknowledgements, Indirect TCP (I-TCP), the Snoop protocol and Link-layer protocols. The suitability of these mechanisms for new technologies like UMTS and GPRS is still an issue under study. Lastly, we should also mention the ongoing research on IP-based protocols for mobility management (e.g., Mobile IPv6, HAWAII, Cellular IP).

6 Conclusions

In this paper, we have briefly reviewed some of the WWW architectures intended for wireless network environments. We have seen that the design of such architectures has been largely based on techniques such as data compression, protocol reduction, caching and pre-fetching, which are considered of major importance in the emerging area of mobile/nomadic computing.

Almost half of the presented architectures focus on the optimisation of the HTTP/Web communication through the introduction of proxies placed in both the mobile node and the nodes of the access network of the wireless infrastructure. The remaining architectures focus on other components of the Web technology like the HTML. Web client software is modified to react to environmental changes according to pre-programmed instructions embedded in extended HTML syntax or fetched by Web servers in the form of self-contained, portable and executable scripts.

We also have presented generic software frameworks for mobile computing environments. Such architectures include the Rover Toolkit as well as the Mobile Application Support Environment. The Rover Toolkit enables the development of all kinds of mobile computing applications, namely mobile-aware applications or mobile-transparent applications. Finally, we have discussed the architecture and protocols of the emerging Wireless Application Protocol framework, which has been adopted as an application deployment environment by manufacturers and operators in the wireless telecommunications domain.

References

[Bak94] Baker, M.G., “*Changing Communication Environments in MosquitoNet*”, Proceedings of the 1994 IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, California, December 1994.

[Bak95] Bakre, A. and Badrinath, B.R., “*Handoff and System Support for Indirect TCP/IP*”, Proceedings of the Second USENIX Symposium on Mobile and Location Independent Computing, April 1995.

[Bal97] Balakrishnan, H., Padmanabhan, V.N., Seshan, S. and Katz, R.H., “*A comparison of mechanisms for improving TCP performance over wireless links*”, IEEE/ACM Transactions on Networking, Vol.5, No.6, 1997.

[Ban97] Banga, G., Douglis, F. and Rabinovich, M., “*Optimistic Deltas for WWW Latency Reduction*”, Proceedings of the 1997 USENIX Technical Conference, 1997.

[Baq95] Baquero, C., Fonte, V., Moura, F. and Oliveira, R., “*MobiScape: WWW Browsing under Disconnected and Semi-Connected Operation*”, Proceedings of the 1st Portuguese WWW National Conference, Braga, Portugal, July 1995.

[Ber94a] Berners - Lee, T., Cailliau, R., Luotonen, A., Frystyk Nielsen H. and Secret, A., “*The World-Wide Web*”, Communications of the ACM, Vol.37 No.8, 1994.

[Ber94b] Berners - Lee, T., Masinter, L. and McCahill, M., “*Uniform Resource Locators (URL)*”, RFC 1738, December, 1994.

[Ber95] Berners - Lee, T. and Connolly, D, “*HyperText Markup Language Specification - 2.0*”, Internet Draft, 1995.

[Ber96] Berners - Lee, T., Fielding, R. and Frystyk, H., “*Hypertext Transfer Protocol - HTTP/1.0*”, RFC 1945, Network Working Group, 1996.

[Cac95] Caceres, R. and Iftode, L., “*Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments*”, IEEE JSAC, Vol.13, No.5, June 1995.

[Cha98] Chandrasekaran, V. and Lo, A., “*The Design and Implementation of a Session Manager for Mobile Applications*”, Proceedings of the 3rd ACTS Mobile Summit, Rhodes, Greece, June 1998.

[Coa98] Coar, K.A.L. and Robinson, D., “*The WWW Common Gateway Interface Version 1.2*”, Internet Draft, February, 1998.

[Cro97] Crovella, M.E. and Bestavros, A., “*Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes*”, IEEE/ACM Transactions on Networking, Vol.5, No.6, December 1997.

[Del88] DellaFera, C.A., Eichen, M.W., French, R.S., Jedinsky, D.C., Kohl, J.T. and Sommerfeld W.E., “*The Zephyr Notification Service*”, Proceedings of the USENIX 1988 Winter Conference, 1988.

[Duc92] Duchamp, D., “*Issues in Wireless Mobile Computing*”, Proceedings of the 3rd IEEE workshop on Workstation Operating Systems, Florida, April 1992.

[Eng95] Eng, K.Y., Karol, M., Veeraraghavan, M., Ayanoglu, E., Woodworth, C. and Valenzuela, R.A., “*A Wireless Broadband Ad-Hoc ATM Local-Area Network*”, ACM/Baltzer Wireless Networks Journal, Vol. 1, May 1995.

[Fie97] Fielding, R., Gettys, J., Mogul, J.C., Frystyk, H. and Berners - Lee, T., “*Hypertext Transfer Protocol - HTTP/1.1*”, RFC 2068, Network Working Group, 1997.

[For94] Forman, G.H. and Zahorjan, J., “*The Challenges of Mobile Computing*”, IEEE Computer, Vol.27, No.4, April 1994.

[Fox96] Fox, A. and Brewer, E., “*Reducing WWW Latency and Bandwidth Requirements via Real - Time Distillation*”, Computer Networks and ISDN Systems, Vol.28, No.7-11, 1996.

[Had98] Hadjiefthymiades, S. and Merakos, L., “*Improving the Performance of the World Wide Web in Cellular CPN Environments*”, Proceedings of the 5th International Workshop on Mobile Multimedia Communications (MoMuC '98), Berlin, Germany, 1998.

[Had99] Hadjieftymiades, S. and Merakos, L., “*A resource management for efficient WWW computing in wireless communications*”, Proceedings of the IEEE VTC ’99 Fall, Amsterdam, The Netherlands, 1999.

[Hal94] Halls, G.A., “*HIPERLAN: the high performance radio local area network standard*”, Electronics and Communication Engineering Journal, December 1994.

[Hou96] Housel, B.C. and Lindquist, D.B., “*WebExpress: A system for Optimising Web Browsing in a Wireless Environment*”, Proceedings of ACM/IEEE MobiCom ’96, New York, October 1996.

[Hou98] Housel, B.C., Samaras, G. and Lindquist D.B., “*WebExpress: A client/intercept based system for optimizing Web Browsing in a wireless envi-*

ronment", ACM/Baltzer Mobile Networks and Applications, Vol.3, No.4, 1998.

[Jan99] Janssen, W.C.Jr., "*A 'Next Generation' Architecture for HTTP*", IEEE Internet Computing, Vol.3, No.1, January/February 1999.

[Jia98] Jiang, Z. and Kleinrock, L., "*An Adaptive Network Prefetch Scheme*", IEEE Journal on Selected Areas in Communications, Vol.16, No.3, April 1998.

[Joh98] Johnson, D. and Perkins, C., "*Mobility Support in IPv6*", Internet Draft, Mobile IP Working Group, March 1998.

[Jos97] Joseph, A.D., Tauber, J.A. and Kaashoek, M.F., "*Mobile Computing with the Rover Toolkit*", IEEE Transactions on Computers, Special issue on Mobile Computing, February 1997.

[Kaa94] Kaashoek, M.F., Pinckney, T. and Tauber, J.A., "*Dynamic Documents: Extensibility and Adaptability in the WWW*", Proceedings of the 2nd International WWW Conference, October 1994.

[Kal98] Kaloxylos, A., Hadjiefthymiades, S. and Merakos, L., "*Mobility Management and Control Protocol for Wireless ATM Networks*", IEEE Network (special issue on PCS Network Management), July/August 1998.

[Kis92] Kistler, J.J. and Satyanarayanan, M., "*Disconnected operation in the Coda file system*", ACM Transactions on Computer Systems, Vol.10, February 1992.

[Koj97] Kojo, M., Raatikainen, K., Liljeberg, M., Kiiskisen, J. and Alanko, T., "*An Efficient Transport Service for Slow Wireless Telephone Links*", IEEE JSAC, Vol.15, No.7, September 1997.

[LaP96] La Porta, T.F., Sabnani, K.K. and Gitlin, R.D., "*Challenges for Nomadic Computing: Mobility Management and Wireless Communications*", ACM Journal of Nomadic Computing, Vol.1, No.1, 1996.

[Lil95] Liljeberg, M., Alanko, T., Kojo, M., Laamanen, H. and Raatikainen, K., "*Optimizing World-Wide Web for Weakly Connected Mobile Workstations: An Indirect Approach*", Proceedings of the 2nd International Workshop on Services in Distributed and Networked Environments (SDNE '95), Canada, 1995.

[Liu95] Liu, G.Y. and Maguire, G.Q., "*A Predictive Mobility Management Scheme for Supporting Wireless Mobile Computing*", Technical Report TRITA-IT R95-04, Royal Institute of Technology, February 1995.

[Liu97] Liu, T., Bahl, P. and Chlamtac, I., "*An Optimal Self-Learning Estimator for Predicting Inter-Cell User Trajectory in Wireless Radio Networks*", Proceedings of ICUPC '97, San Diego, California, October 1997.

[Liu98] Liu, T., Bahl, P. and Chlamtac, I., "*Mobility Modeling, Location Tracking, and Trajectory Prediction in Wireless ATM Networks*", IEEE JSAC, Vol.16, No.6, August 1998.

[Luo94] Luotonen, A. and Altis, K., "*World-Wide Web Proxies*", Proceedings of the 1st International WWW Conference, Geneva, Switzerland, May 1994.

[Mog97] Mogul, J., Douglis, F., Feldmann, A. and Krishnamurthy, B., "*Potential benefits of delta encoding and data compression for HTTP*", Proceedings of ACM SIGCOMM '97, Cannes, France, September 1997 (also in ACM SIGCOMM CCR Vol.27, No.4).

[Mou95] Mouly, M. and Pantet, M.B., "*The GSM System for Mobile Communications*", ISBN: 2-9507190-0-7

[Pad95] Padmanabhan, V.N. and Mogul, J.C., "*Improving HTTP Latency*", Computer Networks and ISDN Systems Vol.28, 1995.

[Pad96] Padmanabhan, V.N. and Mogul, J.C., "*Using predictive prefetching to improve World Wide Web latency*", ACM SIGCOMM Computer Communication Review, July 1996.

[Per96] Perkins, C., "*IP Mobility Support*", RFC2002, October 1996.

[Per97] Perkins, C., "*Mobile IP*", IEEE Communications Magazine, May 1997.

[Rag97] Raggett, D., Le Hors, A. and Jacobs, I., "*HTML 4.0 Specification*", W3C Working Draft, 1997.

[Sch96] Schilit, B.N., Douglis, F., Kristol, D.M., Krzyzanowski, P., Sienicki, J., and Trotter, J.A., "*TeleWeb: Loosely connected access to the World Wide Web*", Computer Networks and ISDN Systems, Vol.28, No.7-11, 1996.

[Voe94] Voelker, G.M. and Bershad, B.N., "*Mobisaic: An Information System for a Mobile Wireless Computing Environment*", Proceedings of the 1994 IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, California, 1994.