Web Proxy Mechanisms for Wireless Networks

- Kedar .B. Mohare
Department Of Computer Science & Engineering
University Of Texas At Arlington
kbm6471@omega.uta.edu

Abstract

The objective behind summarizing the works of several authors in this paper was to brief the audience about the various techniques and approaches to develop efficient and practically feasible web proxy mechanisms for wireless networks/mobile computing environment. The very fact that a host is mobile complicates the standard networking model in many ways and protocols which are commonly used in day to day networks begin to degrade in performance. Mobility has its own set of problems like low bandwidth links, frequent disconnections, high latency, low reliability and high cost. This is further compounded by the fact that the mobile devices have restricted resources in terms of computing power, battery life and display area. This problem can be overcome by introducing an intermediary between the mobile client and its wired peer. The purpose of the intermediary is to process the data flowing between the mobile host and the wired peer in such a way that neither is affected because of the negative characteristics of the wireless environments. The proxy relies on mechanisms like caching, filtering of the data to suit the mobile device, differencing, reducing unwanted headers etc to improve the performance and reduce the latency of information retrieval. This paper describes two architectures proposed to counter the problems in the wireless environment.

1. Introduction:

With the emerging need for "anytime anywhere" information access, access to the web from mobile devices is becoming increasingly significant. But the protocols which are successful in the wired networks have provided very poor performance in the mobile environment. The wireless networks suffer from many shortcomings like low bandwidth, frequent disconnections, high connectivity cost and high retrieval latency.

A substantial amount of work has been done in the area of information access from mobile devices and most of it has been based on the client-proxy server model. Significant work in this area has been done at Daedalus group at Berkeley[7]. In GlopMop[7], the proxy server performs distillation of the document received from the server before sending it to the client. Distillation is defined here as a highly lossy, real time data-type specific compression that preserves most of the semantic content of the document.

The Mowgli system [6] consists of two mediators located on the mobile host and the mobile connection host which cause the Mowgli HTTP protocol to communicate with each other, reducing the number of round trips between the client and the server. Mowgli reduces the data transfer over the wireless link in three ways: data compression, caching and intelligent filtering. In the work of Noble et al [5], the proxy is developed in the context of what the authors term agile, application aware adaptation, they allow an application to register with the operating system its expectation about a resource and the variability it can tolerate. The Odyssey system monitors resources, and informs the application via up calls when the resource value strays outside the bounds decided by the application. The application can then adapt its behavior.

Another study involves providing support to disconnected operation during local caching. Several commercial offline browsers download documents into a local disk for later viewing. The user specifies a set of links which the application downloads and caches. Netscape Netcaster feature allows users to download information from an information broadcast channel, save it in a cache on their system's hard drive and view it at a later date.

Recently the use of personalization techniques and recommender systems are gaining popularity in information retrieval [3], and this can help the wireless web access effort by limiting the information flow on the wireless channel. Grouplens is a system to help people find usenet news articles they will like in the huge stream of available articles.

A lot of research has also been conducted to solve the problem of multimedia content in wireless web access. The proxy mechanism transcode multimedia format, mostly images, using some specific algorithms in some manner that trades quality for bandwidth.

In the following sections we discuss two proxy mechanisms and study their architecture to see how do they offer to solve the problems specific to the mobile environment. This is followed by a discussion of the techniques and how do they perform in the experimental results. At the very end I have proposed a very vague idea to tackle this issue out of my understanding of the subject.

2. General purpose proxy environment [1]:

Proxy based systems are considered commonplace in today's environment. A proxy generally takes the place of an intermediary between the two communicating endpoints such as the mobile client and the server. The whole objective behind deploying an intermediary is to improve the quality of networking services as perceived by the mobile client.

The work involves the development of a mechanism for downloading and executing programs (filters), interposing filters into the middle of client/server connections, and dynamically controlling filter behavior.

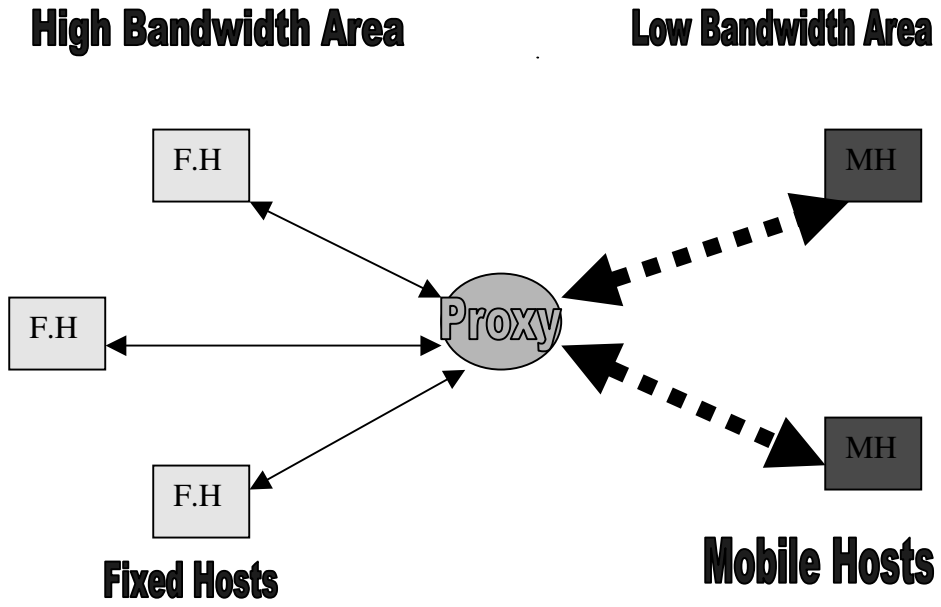


Figure 1. Proxy Computing Environment

The environment for which the model is implemented is shown in the figure. In the above model the **last link** to the mobile host is the **most likely** problem area. The filters that are executing on the proxy can be loaded from another site such as the client or downloaded from any other server. A filter may receive control messages from the application or from anywhere in the surrounding environment, allowing it to dynamically adapt to its environment.

The proxy system described in this section is aimed at providing a solution to the problems arising in the heterogeneous mobile environment. Mobile hosts exacerbate the heterogeneity problem because they move unpredictably through different networks which have very different speed cost, security and loss rate. Hence the purpose of our system is to allow

dynamic runtime, situation specific, adjustment of design time decisions about the nature and amount of traffic across the **weak** link.

The different actions that may be executed by a proxy include the following:

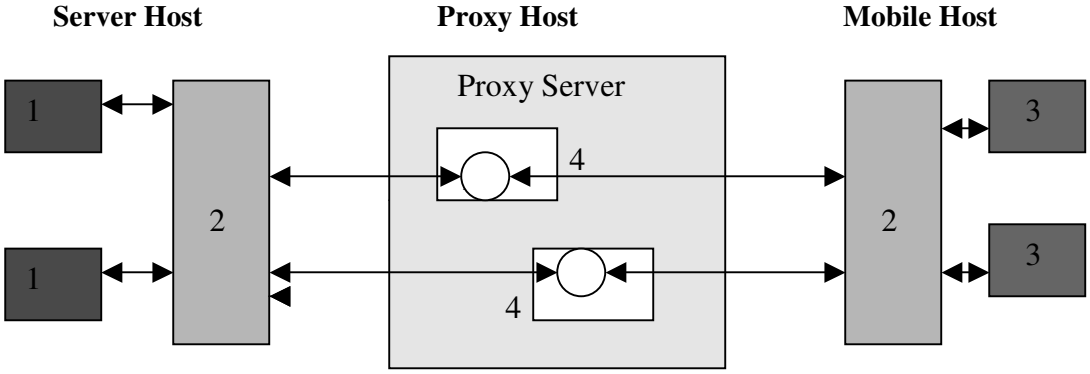
- Running an optimized protocol between itself and the mobile client.
- Selectively drop structured data like frames in a MPEG data stream and superfluous header information in email.
- Compress data instead of dropping it. Note that the compression occurs at the proxy rather than at the end server because the need for compression depends on the link characteristics.
- Delay data instead of dropping or compressing it.
- Act on behalf of the client. For example, reducing frivolous traffic to the mobile host by replying to ICMP echo requests.
- Using different transport protocol or a modified version of standard protocols suited for the wireless networks. It has been proved by studies that small changes in the use of UDP and TCP in wireless networks can result in large performance variations.

The significance of a proxy lies in the fact that with the proxy in place between the client and the server, no modifications need to be done to either of them to improve the performance in a wireless networks.

2.1 Architecture [1]:

The architecture for the proxy mechanism comprises of three major components.

1. **Proxy server** : Filtering unit.
2. **PMICP** : Protocol which moves data to and from the proxy server.
3. **Filter Control** : Enables filters to adapt to their environment.



1 : Server Application
2 : PMICP
3 : Client Application
4 : Filter Execution Environment

Figure 2. Proxy Server/PMICP interaction

The layout of the components and the various data flow paths are shown in the figure above.

2.2 PMICP [1]:

In order for proxy filtering to work, all the traffic to the mobile client must pass through the proxy server. This is accomplished by keeping track of the location of the MH and using the Mobile Support Routers (MSR) to direct the data to its current location. The **PMICP** (Proxy Mobile Internetworking Control Protocol) allows a mobile host to choose a MSR to be its proxy MSR (PMSR). Once chosen, the PMICP protocol guarantees that all the traffic to and from the MH will pass through the PMSR. This helps the data flowing through to be filtered or manipulated to improve the performance in the wireless networks.

Packets destined for the MH are redirected by the MSRs in the network such that the PMSR in charge of the MH is guaranteed to be on the path. Once at the PMSR, data is handled differently depending on its direction:

- Data traveling from the MH : The PMSR sends the data directly to the destination.
- Data traveling to the MH : The PMSR redirects the data to the local PMSR that is in charge of the MH.

2.3 Proxy Server [1]:

This is the heart of the entire architectural model. It provides a dynamic execution environment for filters, which may be permanently resident within the Proxy Server, or downloaded on demand from a mobile host or server on the wired network. The proxy server is considered a process executing on the PMSR. We create a new **Server Process** for each filter as this provides a secure environment in that the processes are protected from each other.

One of the objectives of this system is to provide a mechanism for generalized filtering of protocols ranging from IP up through the application layer. Most modern operating systems make distinction between the network/transport layer protocols and application layer protocols. In the implementation described here, the two types of protocols are each handled by their own separate filtering mechanism. The filtering of application level protocols is handled by a **HIGH** level proxy while the filtering of low level protocols is handled by **LOW** level proxy.

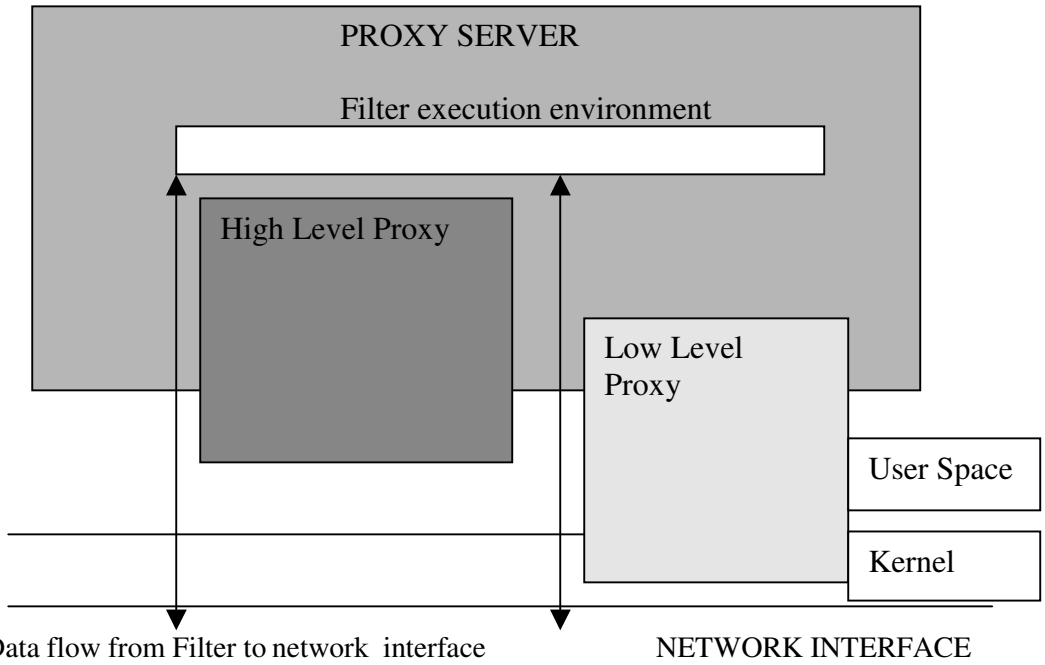


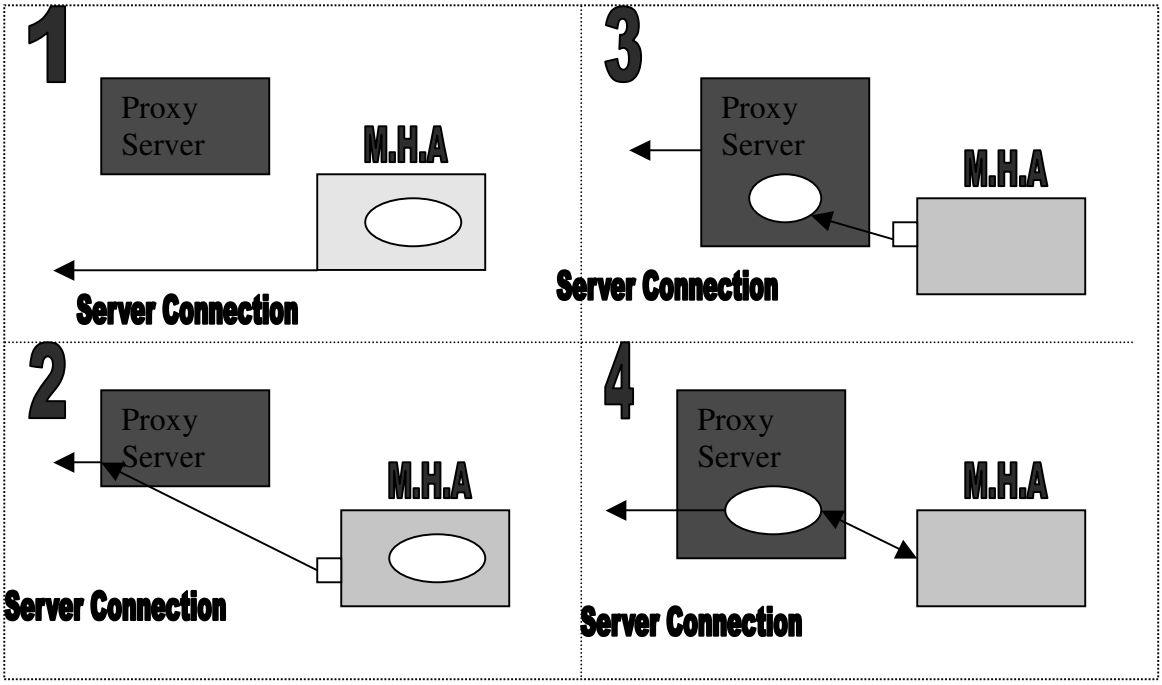
Figure 3 Block Daigram of the Proxy Server

The above figure shows the block diagram of the proxy server showing the locations of the high level proxy and low level proxy within the system. Since the purpose of the High Level

Proxy is to filter application layer protocols, it can exist solely in user space. The Low Level Proxy, on the other hand, filters transport and network layer protocols, so this mechanism spans the user space and kernel space boundary and uses a low level network interface within the kernel.

2.3.1 High Level Proxy:

The High Level Proxy uses the notion of filter insertion. This consists of the Proxy Server functionality and underlying kernel modifications that allow a filter to be inserted into a pre-existing data stream. A single client server connection is broken down into two pieces, one between the client and the filter and other between the filter and the server.



1. **MH application connects to server**
2. **Socket migrates to the proxy server**
3. **Filter downloaded from MH application**
4. **Sockets are re-connected.**

M.H.A –Mobile Host Application

Figure 4 Filter Download and Insertion

The filter insertion process is shown in the figure above.

1. The client endpoint of a preexisting condition is moved from the Mobile Host to the proxy server.
2. The filter is then downloaded and bound to the connection.
3. A new connection is then made from the filter back to the Mobile Host and provided to the application. Hence the previous client server connection is now split into two connections, client-proxy and proxy-server.

The above process of filter insertion is transparent to the server.

2.3.2 Low Level Proxy:

The architecture behind the low level proxy mechanism is quiet different and requires interface to the network layer itself. To receive packets a matching criterion must be established for that filter, an LLP packet queue must be created and configured. This is analogous to socket creation and binding under the Unix Socket model.

2.4 Execution Environment [1]:

The proxy server run time environment executes the code associated with the filter and allows limited interaction between the filter and the rest of the system. Filter execution environment can be broken down into two groups:

- Native/Binary execution environment:
  These environments consist of filters which have been compiled into binary form. The filter is downloaded in the address space of the Proxy Server and executes as a thread within the process.
- Interpreted Execution Environment:
  These environments use filters that are executed within the Proxy Server using an interpreter of some kind. The filter is down loaded from the mobile host to the proxy server and after verification; it is executed by the interpreter.

In the implementation described in the paper only the binary execution environment has been used.

2.5 Adaptation Through Filter control [1]:

Filter can be adapted to their environment through their filter control. The filters can be controlled in two ways, internally and externally. Internal control is said to be executing when the control is directly from the parent application on the MH. External control refers to an indirect form of control based on environment information provided to the filter.

Event Registry (ER) is an information relay system that is used to exercise both internal and external control techniques [1]. It is an event delivery and notification system where filter and applications may register interest in certain events like (change in network bandwidth, change in MH battery power) and are notified when these events occur. Event notifications delivered directly to filters constitute external control, while notifications delivered to applications lead to internal control.

The Event registry has been realized as a process that receives event messages from local and remote entities and notifies interested parties of the occurrence.

2.5.1 How is the Event Information Used:

Filter adaptation can be generally categorized by the “degrees of freedom” that each filter provides [1]. Most filters allow only two degrees of freedom i.e. ON and OFF [1].

The following is a list of events currently used by the event registry and also states how to use the information for filter control.

- Estimated Network Bandwidth:
  As the amount of bandwidth increases or decreases, the filters turn the filtering ON and OFF [1].
- Network Interface Information:
  As the interfaces are taken up and down, filters can ascertain the maximum bandwidth available. This information along with the estimated available bandwidth is used to compute a bandwidth value [1].

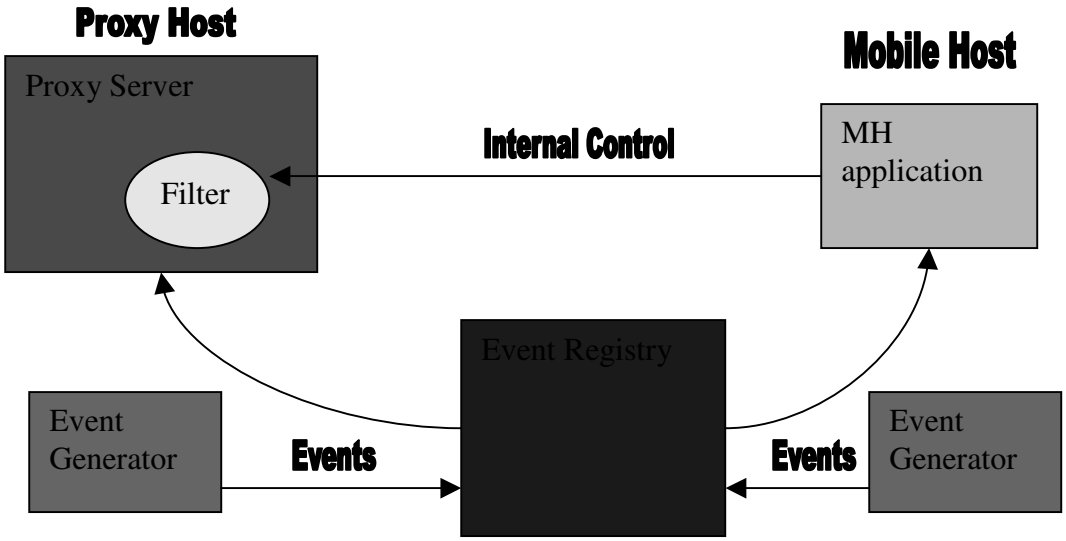


Figure 5. Event Registry Support for Internal and External Support

- Remaining Battery Power.
  The main purpose of the information is to alert the proxy that the battery power is almost exhausted. At this point, the proxy may chose to take drastic steps or to save state and shut down before the MH runs out of power completely [1].
- Mobile Host Location:
  As the mobile host moves from cell to cell, it may experience periods of disconnection. In order to handle this, filters may choose to increase filtering, or may buffer data in the expectation that MH connectivity will be lost [1].

2.6 Filters:

The following filters were designed and implemented:

- HTTP:
  The HTTP protocol is used by most HTML browsers to transfer data from an HTTP server to a browser. The HTTP protocol is an ideal candidate for filtering since the text in the header and the body is highly compressible [1]. The filter for this protocol compresses the head and the body of the HTTP message destined for MH. The data reaching the MH needs to be uncompressed before being delivered to the browser.
- MPEG:
  Bandwidth reduction can be achieved by discarding intermediate MPEG frames based on the amount of network bandwidth currently available. Depending on the network bandwidth, an estimation of loss incurred on the link, MPEG frames are forwarded or discarded.
- SMTP:
  SMTP is a commonly used protocol to transfer email from one host to another. In this context, an email header consists of all sorts of information, some of which may be superfluous [1]. There are many image and voice standards that may be used to encode voice/image data within an email message. The images may be considered as a waste of

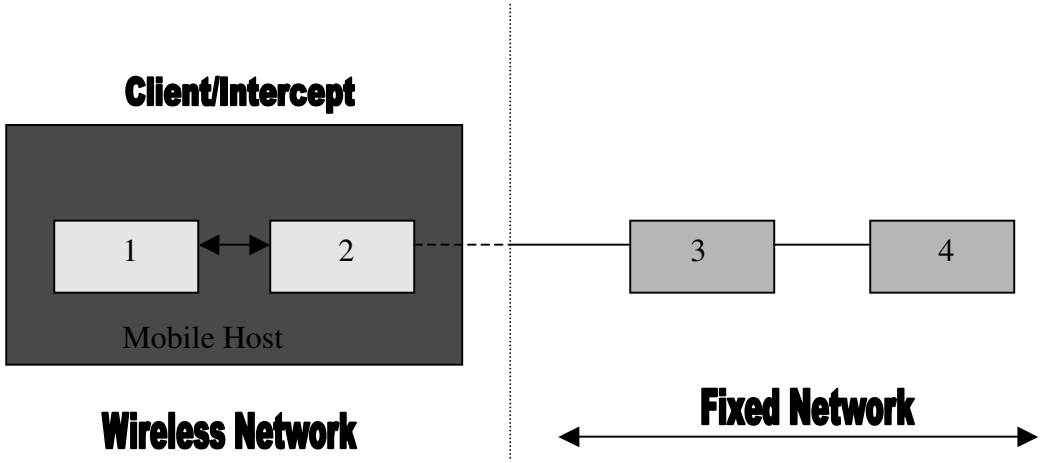
network bandwidth if the image data cannot be rendered by a text based email reader. Hence we can filter these unwanted data from being delivered to the MH

- NFS:
  An NFS client on the MH makes an NFS request to servers within the wired network. Since most of them are NFS read requests, which return data to the MH these are very good candidates for filtering as they will contain compressible text.
- ICMP:
  The ICMP is used by Internet hosts for the purpose of reporting networks errors to fixing bad routes. One example of ICMP control messages is the ICMP_ECHO request. Now, the proxy server has some knowledge of the state of the MH, hence it can use this knowledge to provide replies to queries. Therefore, instead of wasting valuable bandwidth, the filter replies to these requests on its own and thus save valuable network resources.
- TCP:
  A lot of papers have discussed the problems with using the TCP protocol in the wireless environment. Mainly the congestion control algorithm used by TCP is not valid in the weakly connected wireless environment. A solution suggested by a study at Berkeley proposes the use of an intermediary that "snoops" TCP conversations, but does not break the end to end semantics. The main purpose of the intermediary is to cache packets headed towards the MH and to perform local retransmission when packet loss between intermediary and the MH is detected [1]. This has the effect of speeding up the retransmission process and hiding packet loss from the sender side of the connection.

3. WebExpress: A Client /intercept based model [2]:

The limitations of wireless communication results in a form of distributed computing with drastically different connectivity assumptions than in traditional distributed environments [2]. This client/ intercept model aims to alleviate the negative aspects of the wireless links. Another important thing about this model is that it achieves the objective without changes in either the client side or server side application code. It uses a pair of components that run on the client system and within the server's wired network for the interception and optimization of various wireless requests.[2]. WebExpress is a client/intercept system for optimizing Web access. It reduces the traffic over the WAN links in the context of world wide web client server applications. It also reduces the data volume and latency by intercepting the HTTP data streams and performing various optimizations including: file caching, forms differencing, protocol reduction, and the elimination of redundant HTTP header transmission [2].

The client /intercept/server model shown in the figure below, uses an intercept technique that enables the application to intercept and control communication over the wireless links for the purpose of reducing traffic volume and optimizing the communication protocol to reduce latency [2]. The components that implement the intercept technique are inserted in the data path between the server and the client. The Client Side Intercept (CSI) process runs in the end user mobile device and the Server Side Intercept(SSI) process runs in the fixed wired network. The CSI intercepts the client request and together with SSI, performs optimizations to reduce the data transmission over the wireless link, improve data availability to the mobile computation. From the point of view of the client, the CSI appears as the local server proxy that is co-resident with the client. On the same lines, the SSI appears as the local client proxy that resides on the fixed network, typically co-resident with the server. The intercept model is transparent to both the client and the server. The CSI/SSI protocol can facilitate highly effective data reduction and protocol optimization without limiting client functionality or interoperability [2].



1. **Application Client**
2. **Client Intercept**
3. **Server Intercept**
4. **Application Server**

Figure 6. Client/Intercept/Server Model

3.1 WebExpress [2]:

The objective of WebExpress is to facilitate the use of web technology to run typical commercial processing applications over wireless networks [2]. The predictability of this type of usage makes it possible to employ optimizations that make wireless web access practical both from a usability and cost perspective.

3.1.1 The Web Express Intercept Model :

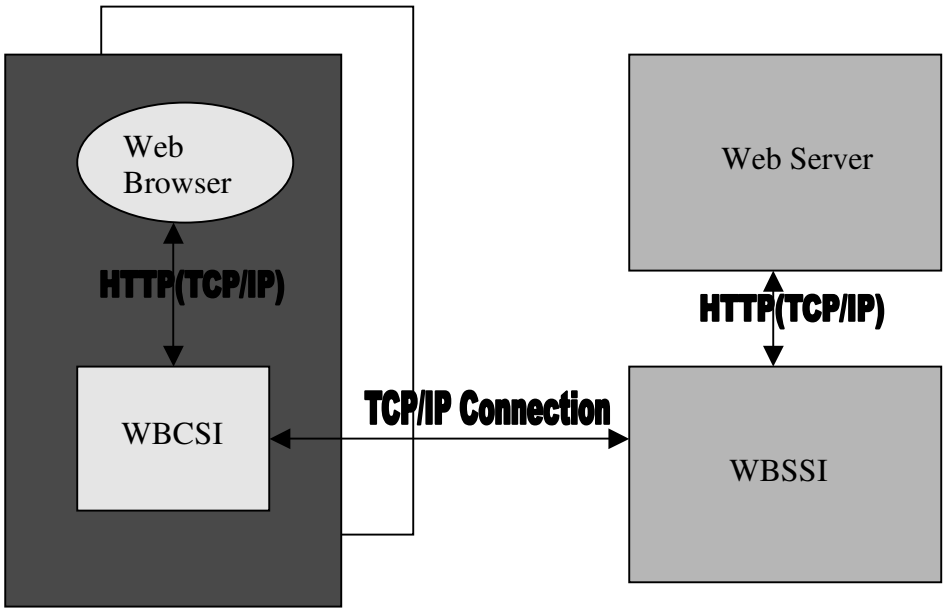


Figure 7. WebExpress Intercept Model

An important objective of the WebExpress is to be able to run with any web browser and any web server without imposing any changes to either. The Web Express model with all of its components is shown in the above figure. The two components that are inserted between the client and the server are the WebExpress Client Side Intercept(WBCSI) and the WebExpress Server Side Intercept(WBSSI). The CSI process runs in the client mobile device and the SSI runs on a host on the wired network.

The CSI appears as a local web proxy that is co-resident with the web browser, The CSI communicates with the browser over a local TCP/IP connection. The CSI communicates with the SSI process over a TCP connection using a reduced version of the HTTP The SSI reconstitutes the HTML data stream and forwards it to the designated web proxy server. Similarly for responses returned by the web servers, the CSI reconstitutes an HTML steam received from the SSI and sends it to the web browser over the local TCP connection.

3.1.2 Caching [2]:

The WebExpress caching methods significantly reduce the volume of data application transmitted over the wireless link [2]. These cache methods that are described in this section have been designed for browsing of stored data that changes relatively infrequently. WebExpress supports client and server caching. It used the LRU algorithm on both the client side as well as the server side. The model intends to maximize the client cache efficiency through cache methods that allow a user to declare frequently accessed or important information for object persistence and to enable the system to adapt to the browsing patterns of the user. The server cache on the other hand is designed to adapt to the browsing patterns of a set of users.

In the WebExpress model, the client or server cache does persist across browser sessions. To solve the cache coherency problems the criteria was based on the age of the information. To provide a cache coherency model, WebExpress associates a digital signature (CRC) computed via a cyclic redundancy check algorithm and a coherency interval (CI) with each cached object [2]. The coherency interval is used to ascertain when the object is to be checked for changes. When a cached object is to be referenced, the CSI carries out a check to find out whether the coherency interval has been exceeded. If it has been exceeded, the CSI and SSI execute protocol to determine if a fresh copy of the page has to be fetched. The CSI requests that the SSI **ve**rify that the object in question has not changed**.** The CSI provides to the SSI the URL of the object, a CRC of the object and the coherency interval associated with the object. If the object within the SSI is too old based on the coherency interval or the object is not in the SSI cache, the SSI will obtain a new copy of the object from the web server and enter a **Store Date Time** (SDT) indicating the current time the object is being accessed as well as the CRC for the object into its cache directory. Now if the CRC match between the CSI's copy and the SSI'*s* copy, SSI indicates to the CSI that the object in question is up to date and CSI then updates the stored date time (SDT) of the object to reflect the SSI's SDT of the same object. If the object has changed, the SSI indicates to the CSI that the object is out of date and sends the updated object to the CSI. The CSI then updates its cache and directory appropriately. The use of a coherency interval allows users to specify how frequently to update cached objects, trading off wireless traffic for cache coherency.

3.1.3 Differencing [2]:

Web browsers and HTML are used to perform transaction processing. The HTML definition allows the specification of HTML streams that enable users to enter data and then submit data for processing by an executable program located elsewhere in the Web. The rules for invoking programs and enabling them to read parameter data and generate replies is the responsibility of the Common Gateway Interface (CGI).

The caching techniques described earlier do not help in CGI processing because no two replies to requests to the same URL are likely to be the same as different users enter different data for different requests and expect to receive different results. To minimize responses from CGI

programs, a differencing technology approach is used based on the observation that different replies from the same program are usually very similar. HTML byte responses representing query responses contain lots of unchanging formatting data which include images as well. The CSI determines that the HTTP request is a CGI request if the method is a POST or if the URL is followed by a name/value parameter list. In the very first instance there is no record of a cached response for the URL at the CSI, and the request is sent to the SSI and forwarded to the server as normal. When the response is received by the SSI, it is cached and its CRC is computed before forwarding it to the CSI. Similarly it is also cached at the CSI before it is sent to the browser. At this point a base object has been established for the CGI URL. The cache coherency methods described previously are not used for base objects since CGI responses are never coherent. Now suppose that another request is made to program X using the same URL at time T+DT. When the request for CGI processing is detected, the CSI checks to see if the URL is cached. In our example since we had cached the object at time T, a cached version is found. The CSI forwards the request to the SSI along with the CRC value of the base object, this CRC is maintained as part of the request state. The HTTP data stream is forwarded to the HTTP server to execute the request.

When an object is returned from the server, it is received at the SSI. Here SSI determines that differencing is possible because a base object for the URL exists in the cache and its CRC matches that received with the request from the CSI. The differencing engine computes the "differencing stream" between the received report and the base object of the URL. A "differencing stream" consisting of a sequence of copy and insert commands, is sent to the CSI. The CSI update engine uses the difference stream and the requests base object to reconstruct a new report. The copy command tells which byte sequences of the base object to be copied to the new report during reconstruction. The CSI then sends the reconstructed report to the browser for display.

3.1.4 Protocol Reduction [2]:

The use of caching and differencing techniques helps reducing the volume of application data to a large extend. However these provisions do not address the overhead of repeated TCP/IP connections and redundant header transmissions. The Web Express system employs techniques to reduce the overhead of both these categories.

3.1.4.1 Reduction of TCP/IP connection overhead [2]:

The Web Express system eliminates most of the overhead of opening and closing connections across the wireless links by establishing a single TCP/IP connection between the CSI and the SSI. The CSI intercepts connection requests and documents requests from the browser and sends the document requests over the single TCP/IP connection to the SSI. For each request received from the CSI, the SSI establishes a connection with the destination server and forwards the request. When the SSI receives the response from the server, it closes the connection with the server, sends the documents to the CSI via the single TCP/IP connection but does not close this single TCP/IP connection. The CSI then forwards the document to the browser. The connection setup and take down overhead is incurred between the browser and the CSI and the SSI and the Web server but not over the wireless link between the CSI and the SSI.

Web Express uses a mechanism called **virtual sockets** to provide this multiplexing support [2]. Virtual sockets enable a CSI to establish a single TCP/IP connection with an SSI and use the connection for many HTTP requests. Data send for a given request is prefixed by a small header that contains a virtual socket id and a length field. At the CSI, the virtual ID is associated with a socket to the browser; likewise, at the SSI the virtual socket is mapped to a socket connection to an HTTP server. A real TCP/IP connection is established with the first virtual socket open, and is closed when a preset time interval expires after the last virtual socket is closed. Hence, this mechanism permits efficient transport of HTTP requests and responses while maintaining correct HTTP protocol and transparency with respect to Web browsers and servers.

3.1.4.2 Reduction of HTTP headers [2]:

HTTP requests and responses are prefixed with headers. HTTP request headers contains list of MIME content types that tell the server various document formats the browser can handle. Since it is usually the same each time, it is unnecessary to send it across the wireless link in every request. Instead the CSI allows the information to flow in the first request after the CSI-to-SSI connection has been established. Both the CSI and the SSI save this list as part of the connection state information. For each request received from the browser, the CSI compares the list received with its saved version; if they match, the list is deleted from the request before it is forwarded to the SSI. When the SSI receives a request from the CSI with no access lists, it inserts its saved copy into the request header. If an access list is present in the received request, it replaces the saved version at the SSI if one exists.

HTTP response headers unlike the request headers, may be different each time. Typically only a few bytes vary from one response to another and hence encoding the data can reduce the response to just a few bytes. The reduction techniques have proved to achieve an overall reduction in HTTP header of up to 90%.

4. Discussion :

The objective behind the work presented in the section 2 is to provide a mechanism for downloading and executing proxy programs, interposing filters into the middle of client-server connections, and dynamically controlling filter behavior. The advantage of deploying a proxy between the client and server is that either little or no modifications need to be done to either the server or the client This system also provides the mechanism for generalized filtering of protocols ranging from IP up through the application layer. In the tests conducted, the quantitative analysis was directed towards the overall performance of the system including the filters. The tests were based upon the HTTP, NFS and the TCP protocols. These were chosen because these are the most commonly used protocols. HTTP and NFS filters use lossless compression, so over all data throughput measurements can be made for transfer of various sizes. The TCP filter improves performance over a lossy link, which can be measured for varying packet loss rate.

The overall conclusions that can be drawn form the HTTP filter tests are mixed. If the transfer sizes are small (6K), then using the HTTP filter will not help if latency is the primary performance metric. But in cases where the file sizes are large, the system clearly performs extremely well. The NFS filter performs somewhat poorly in environments with little heterogeneity, but performs well when the level of heterogeneity is high. In case of the TCP filter, the TCP proxy performs better than the standard TCP as the error rate increases.

As far as the performance statistics of the WebExpress project is concerned, the authors have shown that a 60% to 99% reduction in wireless network traffic and 36% to 97% improvements in application response. To demonstrate the effectiveness of WebExpress, a series of transactions were run against two applications on active web sites in the Internet: a DB2 world wide web connection demo application and a popular quote server application. The bytes transferred and the elapsed response time were recorded for each transaction. The WebExpress model is different from other models in the use of differencing technologies with forms processing, its distributed cache model with its coherency algorithms and the virtual socket design and implementation that minimizes the number of network connections needed for HTTP requests. The Web Express technology supports these technologies via the intercept model and thus transparently supports today's browsers, servers and transport stack.

5. Conclusion:

The general proxy approach discussed in section (2) does not preserve the TCP end to end semantics. This is because a TCP acknowledgement confirms that data was delivered to the destination. But whenever a server receives an acknowledgement on its TCP connection to the proxy, it is not necessary that data has been delivered to the mobile host. There is no simple

solution to the problem. The security issue that needs to be addressed is the runtime execution environment of the filters. Since the filters are downloaded to the proxy server, there might be a serious design flaw if authentication issues are not dealt with. Proxy mobility is another issue that has to be given serious consideration. Both approaches of having a static proxy and a moving proxy have their own pros and cons. So a thorough analysis needs to be done to weight each alternative and find the optimum solution.

Based on my study of the various techniques, I would like to propose a vague idea for this topic. I have thought about this concept from two point of views. One is that we have to live with the inherent disadvantages that come with the wireless computing environments and that we have to find ways to adapt to it, second that we also have to keep in mind the capabilities and limitations of the mobile devices that we are be using and the specific requirements behind the people using these devices. My idea is based on the proxy mechanism concept. On the first contact with the proxy, the mobile device furnishes information about its own system capabilities and limitations to the proxy server. This helps the server to cater to the specific requirements of the device. For example, if the mobile device has only a text based browser, the there is no use in downloading images which the browser will not be able to render and thus waste network bandwidth.

Also most of the time, the mobile users are looking for specific information, like stock quotes, weather reports, traffic conditions, corporate email etc. There will be software on the mobile device which keeps track of what information the user accesses and at what time of the day. This will help to build up a profile of the user which can be passed on to the proxy server whenever the device connects to it. Also this software module can issue the requests to that information on behalf of the user without any sort of initiation from the user. Thus it can cache this either at the proxy server or if possible on the mobile device. Thus when the user looks for this information, there will be less latency in retrieving the information. This work could be build up on the work that has already been carried out in the past.

References:

1. B.Zenel, A general purpose proxy filtering mechanism applied to the mobile environment, J.C. ACM, Wireless Networks 5 (1999) 391-409
2. B. C.Housel, G. Samaras, D.B.Lindquist, WebExpress : A client/intercept based system for optimizing web browsing in a wireless environment, ACM , Mobile Networks and Applications 3(1998) 419-431
3. A,.Joshi, On proxy agents, mobility and web access, ACM , Mobile Networks and Applications, Dec 2000 vol 5 issue 4
4. J. Steinberg, J.Pasquale, Mobility and Wireless Access: A web middleware architecture for dynamic customization of content for wireless clients, ACM Proceedings of the 11th international conference on WWW, May 2002.
5. B.Noble, system Support for mobile, adaptive applications, IEEE Personal Computing systems, vol 7, 44-49, Feb 2000.
6. M.Liljeberg, M.Kojo, K. Raatikainen, Enhanced Services for the world wide web in mobile WAN environment (1996), http://www.cs.Helsinki.FI/research/mowgli/mowgli-papers.html.
7. A.Fox and E.A.Brewer, Reducing www latency and bandwidth requirements by realtime distillations, in Proc. Of the 5th International www conference (May1996).
8. R.Katz, Adaptation and mobility in wireless information systems, IEEE personal communications1(1) (1994).6-17
9. GLoMop: global mobile computing by proxy, GloMop Group (March 1995).
10. G.H. Forman and J.Zahorjan, The challenges of mobile computing, IEEE Computer 27(6) April 1994.
11. C.Brooks, M.Mazer, S.Meeks and J.Miller, Application specific proxy servers as HTTP stream transducers, World Wide Web Journal(Dec 1995),539-548.
12. J.Ioannidis, D.Duchamp, G.Q.Maquire.Jr and S.Deering , Protocols for supporting mobile IP Hosts, Interent Draft(June 1992).
13. B.Badrinath,P. Sudame, To send or not to send: Implementing deferred transmissions ina mobile host, in Proc of the 16th International Conference on Distributed Computing Systems, IEEE(May 1996).
14. D.Andersen, D.Bansal, D.Curtis, S.Seshan, and H.Balakrishnan, System Support for bandwidth management and content adaptation in Internet applications, Proc. Of the 4th symposium on Operating Systems Design and Implementation, 213-226, Oct 2000.
15. T.Knutz and J.P.Black ,An architecture for adaptive mobile applications, Proc. Of the 11th international conference on Wireless Communications, 27-38, July 1999.
16. M.Lijeberg, T.Alanko, M.Kojo, H.Laamanen and K.Raatikainen, Optimizing world wide web for weakly connected mobile workstations: An Indirect Approach, Proc of the 2nd International Workshop on Services in Distributed and Networked Environments, June 1995.