http://www.cs.berkeley.edu/~fox/glomop/glomop-posn.p

September 13, 1995

GloMop: Global Mobile Computing By Proxy

GloMop Group
glomop@full-sail.cs.berkeley.edu

1.0 Overview and Motivation

1.1 Toward a Globile Mobile Information Infrastructure

Mobile computing and the Internet are two of the most rapidly growing areas of the computer industry. However, unlike desktop workstations in a conventional wired WAN environment, today's mobile devices are typically limited to kilobits-per-second connectivity using conven tional (wired) or cellular modems and, in some cases, satellite relays. Existing bandwidths are inadequate for Internet surfing, and will be inadequate for generalized multimedia information retrieval applications. Network bandwidth, especially for wireless networks, is simply not pacing advances in processor speed.

How can applications deal with the bandwidth, memory, computational, and power constraints of a mobile computing device?

1.2 Computation By Proxy

Our central observation is that *access to data is at least as important as local computation and storage.* This assumption is supported by the observation that today's subnotebooks can run at 100 SpecInt and are equipped with up to 1GB of hard disk and a CD-ROM drive, yet without connectivity (to email, to the corporate Lotus Notes server, to colleagues' documents at a remote location) they lose much of their usefulness.

Various solutions exist for connecting mobile devices to the rest of the world, but these solutions almost universally suffer from low bandwidth and high latency.

TABLE 1. Characteristics of overlay communication technologies

Type	Bandwidth	Latency	Mobility	Typical Video Pfmnc	Typical Audio Pfmnc
Thin Ethernet	About 10 Mbps	< 10 ms	Fixed	2-way interactive	16 bits @22 Khz
In-building wireless (WaveLAN, IR)	Commercial RF: 2 Mbps Research IR: 50 Mbps	< 10 ms	Pedestrian	2-way interactive, full frame rate	16 bits @22 KHz
Campus-area packet relay (Metricom)	64 Kbps	100ms	Pedestrian	Medium quality slow scan	Med quality reduced rate
Wide-area (ARDIS, wireline modem)	14.4-19.2 Kbps	> 100ms	Vehicular	Freeze frame	Async (voicemail)
Regional-area (LEO, DBS, VSAT)	4.8Kbps-10+ Mbps (asymmetric)	> 100ms	Vehicular, stationary	Seconds/ frame, freeze frame	Async (voicemail)

We can make mobile devices of all kinds useful to a much wider audience by *isolating the devices from poor networks*. In current usage, a mobile device connects directly to a resource of interest (web page, mail inbox, etc.) using a low-level stream protocol (typically, TCP/IP running over Ethernet or over a modem using SLIP or PPP). These protocols were designed without specific regard for bandwidth; for example, every HTTP request results in a setup and teardown of a new TCP stream, which fails to amortize TCP's considerable overhead, and documents requested by HTTP are transmitted in source form, even though the requesting machine may not have the resources to render the source document (lack of memory, lack of computing power, or a small screen).As a result, Web surfing is extremely painful at 14.4Kbits/sec.

Under the GloMop model, however, the client connects not to a particular service, but to a *proxy process* running at a well-known location. The proxy is running on a server or is distributed among servers that are *well connected* to the desired data, either because the data is at the site or because the site is well-connected (i.e. an order of magnitude better than the client-proxy connection) into the Internet. For UC Berkeley, most data is reachable at 8 Mbps, which is more than 200 times better than a 28.8 modem connection. Although the connection to the proxy is made on a reliable stream such as TCP, the requests and data transmitted on that stream are quite different from what would be transmitted using (e.g.) HTTP directly.

Besides retrieving documents using any desired transport protocol, the proxy provides two key services to the mobile client, which we describe in depth in the next section:

1. **Distillation** of the document, a type of semantic lossy compression we describe in the next section.
2. **Incremental refinement.** Because the proxy caches the original (source) document the client can selectively request refinements of portions of the document.

1.3 Distillation and Refinement

Simply put, *distillation* can be thought of as lossy compression that preserves enough semantic information to make a document useful while making the document drastically smaller and easier to render on the mobile client, and possibly exploiting the document's semantic structure. Illustration by example is best:

Example: Graphic image. We can reduce the area or the color palette or both for a large full-color graphic. For example, we reduced an 8-bit-color, full-screen (VGA) GIF image to a thumbnail-sized 4-gray image for display on a Sony MagicLink PDA. In so doing, we reduced the image's size from 503 Kbytes to 560 bytes, and the transmission time (including real-time distillation) from 2,000 seconds to 13 seconds at 1200 baud. Furthermore, because our image processing included optimal quantization to the PDA's (static) graymap, the resulting image was much sharper on the PDA screen (and therefore more useful to the human viewer) than the dithered original, so we actually increased the perceived image quality while reducing the size a thousand fold.

Example: Formatted text. The default format for rich text today is PostScript, which is highly verbose, requires much memory and computation to render, and reproduces well only on a high quality display. Today's PDA's have none of these features. The proxy can convert PostScript to formatted ASCII plaintext; we have even encoded the resulting plaintext as a GIF (to exploit LZW and run-length compression) and achieved an overall reduction of 11x to 13x. For documents whose "source code" is available (e.g. FrameMaker or MS Word), the proxy can even transmit the document's table of contents and allow the client to request only specific chapters, with or without included graphic images (which are distilled separately using the techniques described previously).

Example: Continuous video. Intraframe video compression can be achieved using a variant of the techniques used for stills, and frame dropping and interframe compression can further reduce the bandwidth requirements of the video stream. Berkeley's Daedalus Group has demonstrated real-time transcoding of motion JPEG to subsampled H.261, which is transmitted at 128Kbps to a WaveLAN-equipped IBM ThinkPad. Half of this bandwidth can be achieved using two parallel 28.8Kbps wireline modems (in fact, the PPP specification already allows this), enabling the possibility of-useful videoconferencing on a PDA.

In summary, distillation exploits large amounts of computing power on the proxy side to maximize the client's limited computing and network resources, by performing intelligent *semantic compression* in a datatype-dependent way.

1.4 Why An Application-Level API

To date, most work on mobile computing protocols has either centered on the adaptation of existing protocols to mobile computing [1] or has focused on the design of link-level or transport-level protocols for mobile computing. Much less work has been done on application-level views of mobile computing protocols; a notable exception to this is the Wit project at the Univ. of Washington [2].

We believe that an application-level view is critical in designing mobile computing protocols, for the following reasons.

1. Mobile computing is a multi-platform world. To enable multiple platform versions of an application requires a common cross-platform API. TCP/IP provides such an API, but it is too low-level: higher-level protocol functionality built on top of TCP/IP (e.g. FTP, HTTP) must be replicated in each different version of an application (e.g. every Web browser must include an HTTP client, a GIF renderer, etc.). Applications should be shielded from having to replicate this functionality for common data types.
2. Applications need a well-defined way of dealing with the severe resource constraints faced by mobile applications, *and* for delegating partial responsibility for dealing with them to the underlying protocol handlers.
3. The datatype-specific distillation we have described is most useful if performed at the document level, but current commodity protocols generally have no notion of document structure. (An exception to this is MIME [3], and we do indeed have a MIME email client that uses our techniques for distillation of each MIME inclusion.)

Thus, to capitalize on the growing momentum of mobile computing across multiple platforms, we must provide an application-level, document-centric, and platform-independent API.

1.5 Proxy Platforms

Our initial proxy will run under POSIX, but because it will be written in ANSI-compliant C, we expect that it will not be difficult to port to other platforms.

The proxy can and will take advantage of two technologies that maximize exploitation of server-side compute power:

1. **Multithreading.** Windows NT, the Power Macintosh OS, and many commercial versions of Unix now support multithreading capable of exploiting multiple processors in a single box.
2. **Networks of Workstations.** The Berkeley NOW project [4] is investigating how to harvest the idle cycles from an ATM-connected network of workstations using a "glue layer" that sits on top of a commodity Unix and provides cross-machine resource allocation and process control.

1.6 Client Platforms

Although some laptops are powerful enough to run Unix, it is one of the least used ope: ating systems in the mobile computing world because it lacks user-friendliness, power management (until recently), and access to the huge installed base of productivity appl: cations. Furthermore, although we distinguish PDA-class devices (such as the Apple Newton or Sony MagicLink) from conventional laptops, we wish to enable application: across the entire range of devices. Besides lacking the computing resources to run Unix or Windows, PDA's have an entirely different interface and programming model, whicb are often tightly integrated.

These assessments suggest the following partial list of target platforms, sorted according to the sizes of their user populations:

1. Intel x86-based portables running Microsoft Windows and its derivatives (and in a few cases, Linux or BSD/386)
2. Macintosh PowerBooks running MacOS
3. DOS-derivative palmtops such as the HP 95LX
4. Self-contained PDA's (Apple Newton, Sony MagicLink). We have a prototype MIME email client that uses the GloMop API running on the MagicLink.
5. Java-aware browsers, to exploit platform independence and code migration. We have a prototype text email client written as a Java applet that uses the GloMop API.

1.7 Related Work at Berkeley

The Daedalus project [5] is exploring seamless migration among overlaid wireless networks. Most of the time, applications can be insulated from handoffs between networks (at least if the handoff is successful). However, the application may want to be informed if the network characteristics (bandwidth, latency, cost) have changed more than some threshold amount, so it can adjust its policies and/or user interface accordingly. GloMop provides the insulating layer between Daedalus' migration functionality and client applications.

The Berkeley InfoPad project [6] uses an in-building high-speed wireless network to provide connections between multimedia I/O terminals and fixed workstations that provide computation and connectivity to the rest of the Internet. The InfoPad is an extreme example of a proxy-based mobile device in that it has no general computation power onboard, relying exclusively on the fixed hosts for *all* computation and connectivity. While InfoPad has done a good job of exposing important issues in low-power picocell system design, we believe that the future of mobile computing lies with a more general model in which the mobile client provides at least some functionality while disconnected.

1.8 Relation to Other Mobile Computing Research

A mature project that is closer in spirit to our approach is Wit [2] at the University of Washington. The focus of that project is the partitioning of application functionality between the mobile client (HP 100LX palmtop running MS-DOS, communicating via the Xerox IRNet) and a proxy process on a workstation. Wit provides a tightly-integrated application-level environment on HP palmtops, including simple windowing of a text display, Tcl with threads, RPC, and a custom stream protocol. However, Wit does not provide a common layer that handles datatype-specific distillation, nor does it specify an API for applications to deal with changes in network characteristics or specify how quality-of-service parameters could be mapped onto network characteristics.

The Odyssey project [7] provides an application-level API for document retrieval that has a notion of datatypes, can deliver one of several levels of document quality, and allows the client to adapt to network changes by requesting "policy changes" from the server delivering the document. The difference between Odyssey and GloMop is best characterized as static vs. dynamic. In Odyssey, the server must explicitly store multiple representations of a document, and deliver a specific version requested by the client; in GloMop, the server can retrieve a source document from anywhere on the Internet, negotiate an appropriate representation with the client, and perform the required distillation, all in real time. A document's type in Odyssey is determined by which directory volume (subtree) it resides in, with each volume having a static type; a document's type in GloMop is determined on a per-document basis. Adding new datatypes to the Odyssey system is difficult because the type-specific calls must be interfaced to the Unix filesystem calls; in contrast GloMop provides an API for adding new types on the fly, and in fact the Java implementation of the GloMop client has this feature by default, since it exploits Java's code migration capabilities.

The Xerox ParcTab [8] is a pager-size device designed to operate in a highly-connected local environment (one of the Xerox PARC buildings), using IRNet to communicate with workstations. In this sense it is a closer relative of the InfoPad than of GloMop. The ParcTab's custom applications generally exploit location knowledge and their interfaces are suited to its small size (three inch display, stylus input).

The LittleWork system [9] primarily addresses disconnected filesystem operation for mobile devices. Our approach is more general: a filesystem-consistency mechanism could be (and probably will be) built on top of GloMop.

The Mobile IP project [1] explores how to hide the relocation of IP connections from mobile applications. GloMop, in contrast, assumes an underlying reliable stream model (e.g. TCP); mechanisms such as Mobile IP are in some sense orthogonal to our goals, although the Daedalus project is investigating connection-oriented services for mobile hosts [10].

2.0 Services Provided by a Proxy

This section describes some of our ideas for other services that a proxy architecture is uniquely positioned to provide. We do not currently have prototypes of any of these services.

2.1 Local File Caching

We can use PC Card hard drives (formerly called PCMCIA), up to 540MB, as portable file caches. The user synchronizes the file cache before leaving and the proxy would keep track of the contents. Mobile applications can check the file cache before asking for information from the proxy. The client can ask the proxy to verify that the local ver sion of a given file is current. As an example, we would store all of the users' email and web cache on the drive. We might include all of the user's home directory as well. Of course missing items will be added to the cache as they are brought to the client.

Our current API allows this functionality to be implemented virtually transparently, in that the application would initially request a document and the GloMop protocol layer would transparently try to satisfy the request from the PC Card drive before notifying the proxy.

Issues such as how to resynchronize the cache upon reconnect have been addressed by the Coda disconnected file system [11].

2.2 Urgent Notification

The proxy can notify the user of an "important" event via a paging service or through RadioMail (ARDIS). Notification is a primitive operation and can be used in any application or user script. Typical uses might include email from a particular person or with a particular subject line, or the completion of some background process run on the proxy.

2.3 Fax Back

The user can request that the proxy retrieve a document and fax it to a particular phone number, performing some intermediate image processing to maximize the quality of graphics rendering on the fax. The *tpc.int* service already routes faxes from the Internet to many metropolitan areas for free, and adding a *tpc.int* transport module to the proxy would be trivial.

2.4 Telephone Access

It should be possible to interact with the proxy via a touch-tone phone. Input would be a comination of touch-tone dialing (like voice-mail menus) and voice recognition. Each user can build a menu structure and associate menu commands with user procedures. A standalone PC running the proxy would handle menu selections and send output to the user via text-to-speech conversion, for which free, high quality code is available. Example applications include listening to your email and checking the status of a running agent.

We can also use the phone line for notification: the proxy calls you for an urgent notification if it cannot reach your mobile device (e.g. because the mobile device is offline, or because you are not carrying it with you).

2.5 Agents

An agent can be thought of as a generalization of a long-lived document request that carries very low priority, i.e. one whose semantics are "do whatever is necessary to get the information, and I'll check back with you later." We still need to define the environment in which agents would run, but since they run on the proxy side, they have the run of the Internet.

The General Magic Telescript environment, which is currently implemented in the AT&T PersonaLink, allows PDA applications to launch and communicate with remote autonomous agents that inhabit a Telescript network. Unfortunately, Telescript has not realized widespread use becuase it is currently a proprietary technology. In contrast, GloMop agents will use and enhance existing non-proprietary Internet services.

3.0 GloMop Functional Description

3.1 Abstractions

Since the goal is to make application writers' lives easier (and thereby encourage development of truly mobile client applications), the application's view of the network must provide easy-to-manage high-level abstractions for a number of important properties of the virtual connections. This has been observed by [12]. Furthermore, it should be possible for the client application to delegate management of any property to the network software, rather than managing it explicitly. For example, in the case of changes in bandwidth, the client application might choose to let the network software decide how to perform distillation to utilize available bandwidth most efficiently, rather than making explicit distillation decisions for each document accessed remotely.

GloMop's API provides abstractions for the following properties of interest:

1. **Bandwidth and latency.** These parameters span a broad range from in-building wired networks to satellite relays.
2. **Priority.** Different data types by nature have different priorities, e.g. motion video should have high priority while email and FTP can use the lowest priority.
3. **Price.** A cost model for each wireless service provider (and perhaps for the actual documents retrieved) can help the user stay within a preset budget, or specify cost constraints for document retrieval.
4. **Security/encryption.** Security is already recognized as a major issue for wireless networks. Some networks provide link-level encryption, which may be enough for transmission of nonsensitive data, but users should have the option of enabling end-to-end encryption (at increased computational cost) for sensitive data.
5. **Power consumption.** GloMop's network management module will develop statistical power consumption profiles and use them to decide when it is worth activating multiple network interfaces, or when bandwidth requirements can be safely reduced in order to conserve power.

3.2 Key Features of Proxy Architecture

Built on reliable streams. GloMop will assume that the OS can provide an abstraction for a reliable stream, such as TCP or X.25. The details of the abstraction are unimportant, and it is not necessary that the OS allow more than one reliable stream connection to be open simultaneously.

Type-specific, adaptive "pipes". The most important abstraction provided by GloMop is a data pipe that has intimate knowledge of the data being sent and received through it. A hierarchy of typed pipes is possible, e.g. there may be a "generic" IMAGE pipe that provides minimal support for transmitting any type of graphic image, as well as specific pipes that have detailed knowledge about different formats such as GIF and JPEG. These pipes can make decisions about how to distill, compress or otherwise manipulate document data to make the best use of the underlying network. The typed pipe architecture is extensible: new type modules register themselves with the system, enabling an aftermarket for document-specific extensions. An example might be a document type that provides support for transmitting different semantic layers of a map graphic: first the major arteries only, then the mountain ranges, then the smaller roads, etc.

One reliable stream per NI. For our purposes, a "network interface" is loosely defined as "any I/O port that transmits data between the mobile client and the external world." Thus, a conventional modem, cellular/radio modem, IR port, or Ethernet card all qualify as network interfaces. The rationale is to minimize the effect of the overhead associated with a reliable stream; experience with the World Wide Web has shown [13] that associating a separate reliable stream connection with each virtual connection (e.g. one TCP connection per HTTP request) exacts a high price in performance and data overhead. GloMop will explicitly allow multiple, possibly asymmetric NI's to be in use simultaneously; for example, a pair of conventional 28.8 modems (e.g. a PC-card modem and an internal modem on a laptop) could be used in parallel to achieve 56 Kbits/sec using two different telephone connections. Each NI will have different properties with respect to bandwidth, latency, cost of operation, etc., but for the most part knowledge of these details will be confined to the measurement/scheduling layer described below.

Measurement and scheduling. This layer, which lives between the API seen by the application and the OS interface to a reliable stream service, will be responsible for multiplexing virtual connections onto the reliable streams associated with the NI's. Automated statistical modelling techniques [14] will be used to collect information about the performance of the physical connection. This information will be used in scheduling the competing virtual streams as well as providing the user with predictions of the cost of retrieving a given document (in both connect minutes and dollars). one reliable-stream connection *per hardware network interface.*

3.3 Document-Centric Request Model

We loosely define a *document* as a collection of bytes that can be taken to represent a single macroscopic object when the appropriate interpretation is imposed on them. By this definition, a code module is as much a document as is a word processor file, a graphical image, a sound sample, or a motion video clip.

When the client connects to the proxy, it registers a list of data types it is prepared to accept. Note that since the datatype-specific modules reside in the GloMop client layer and not in the individual applications, this single list can apply to all requests regardless of the originating application.

To request a document, the client passes the proxy a *document locator*, specifying how the source document may be retrieved, and a set of *QOS parameters*, specifying which constraints are most important to the client. The proxy retrieves the source document, determines its type, and locates a type-specific module (TSM) consistent with the registered type list to handle the request. The TSM negotiates with the network management layer (NM) to determine how to distill the document for this client given the QOS parameters, which may encode a constraint such as "Deliver the best representation possible while keeping the transmission time under 30 seconds" or "...while keeping the transmission cost under $5.00".

3.4 Document Structure

Documents are *structured* in two respects. First, each document is divided into an optional *document table of contents (TOC)* and a required *body*. The TOC, if present, describes the sizes and contents of each *chunk*, which collectively make up the body. Simple documents, such as plain text, may consist of only a single chunk and have no TOC. Rich documents, such as layered maps or MIME-encoded email messages, may consist of several chunks plus a TOC describing the chunks. The semantic interpretation of chunks is left to each document type; the intention is that a chunk be a logical semantic subdivision of the document's content.

Second, each chunk can contain a *document reference table*—a list of other documents referred to by this one. For example, an HTML page consists of one or more chunks of HTML and a DRT with an entry for each of the inlined graphics (and perhaps an entry for each hyperlink). Note that this implies a fair amount of intelligence at the proxy: the proxy must examine the document, replace the inline directives with DRT entries (appropriately annotated so that the human using the client application knows what inline images were substituted), and (ideally) prefetch and distill the referenced documents, so they will be ready and waiting should the client request them.

3.5 Document Uploading

The client can send a document to the proxy using a similar protocol. Sending actually performs an enqueue operation, which allows documents to be queued for sending while disconnected and allows the NM layer to decide when is a good time to perform a waiting send (e.g. it may decide to do a send when the application is temporarily quiescent). Examples of document uploading include sending of new email, sending a log file to resynchronize the user's inbox on the proxy with the one on the client, and annotating (marking up) a downloaded source document.

3.6 Type-Specific Modules

TSM's encapsulate knowledge about distillation: how to exploit specific properties of the document for compression, and how to present a document to the client in a structured way. A TSM is composed of two parts: a client side (decoder) and proxy side (distiller). The interface between the client side TSM and the client side of GloMop is well defined, as is the interface between the proxy side TSM, the "core" of the proxy, and the NM layer. Thus adding new TSM's is straightforward. The Java [15] implementation of the GloMop client (already underway) will use Java's built-in ability to dynamically link architecture-neutral code to add data types on the fly: a document server will provide not only a document, but if necessary the code for decoding that document type.

4.0 References

[1] J. Ioannidis, D. Duchamp, and G. M. Jr. Ip-based protocols for mobile internetworking. In *SIGCOMM 91*, 1991.

[2] T. Watson. Application design for wireless computing. In *Mobile Computing Systems and Applications Workshop*, August 1994.

[3] K. Moore and N. Borenstein. Mime (multipurpose internet mail extensions) parts 1 and 2. RFC 1521 and 1522, September 1993.

[4] T. E. Anderson, D. E. Culler, and D. A. P. et al. The case for now (networks of workstations). *IEEE Micro*, (To appear). Also see http://now.cs.berkeley.edu.

[5] R. H. K. et al. The daedalus project (home page). http://daedalus.cs.berkeley.edu.

[6] B. Barringer, T. Burd, F. Burghardt, A. Burstein, A. Chandrakasan, R. Doering, S. Narayanaswamy, T. Pering, B. Richards, T. Truman, J. Rabaey, and R. Brodersen. Infopad: A system design for portable multimedia access. In *Calgary Wireless 94 Conference*, July 1994.

[7] B. D. Noble, M. Price, and M. Satyanarayanan. A programming interface for application-aware adaptation in mobile computing. In *USENIX Symposium on Mobile and Location-Independent Computing*, 1995.

[8] N. Adams, R. Gold, B. Schilit, M. Tso, and R. Want. An infrared network for mobile computers. In *First Usenix Symposium on Mobile and Location-Independent Computing*, pages 41–51, August 1994.

[9] L. Huston and P. Honeyman. Disconnected operation for afs. In *First Usenix Symposium on Mobile and Location-Independent Computing*, pages 1–10, August 1994.

[10] K. Keeton, B. Mah, S. Seshan, R. Katz, and D. Ferrari. Providing connection-oriented network services to mobile hosts. In *USENIX Symposium on Mobile and Location-Independent Computing*, August 1993. Also at http://www.cs.berkeley.edu/ kkeeton/Papers/usenix-paper-release.ps.

[11] J. J. Kistler and M. Satyanarayanan. Disconnected operation in the coda file system. *ACM Transactions on Computer Systems*, 10(1):3–25, February 1992.

[12] G. H. Forman and J. Zahorjan. The challenges of mobile computing. Technical Report 93-11-03, University of Washington Computer Science & Engineering, December 1993. Available at ftp://ftp.cs.washington.edu.

[13] V. Padmanabhan. Improving world wide web latency. Technical Report UCB/CSD-95-875, UC Berkeley Computer Science Division, May 1995. Available at http://cs-tr.cs.berkeley.edu/Document/UCB:CSD-95-875.

[14] E. A. Brewer. High-level optimization via automated statistical modeling. In *Principles and Practice of Parallel Programming*, 1995. Also at http://www.cs.berkeley.edu/ brewer/hll.ps.gz.

[15] S. Labs. The java language: A white paper. Available at http://java.sun.com.