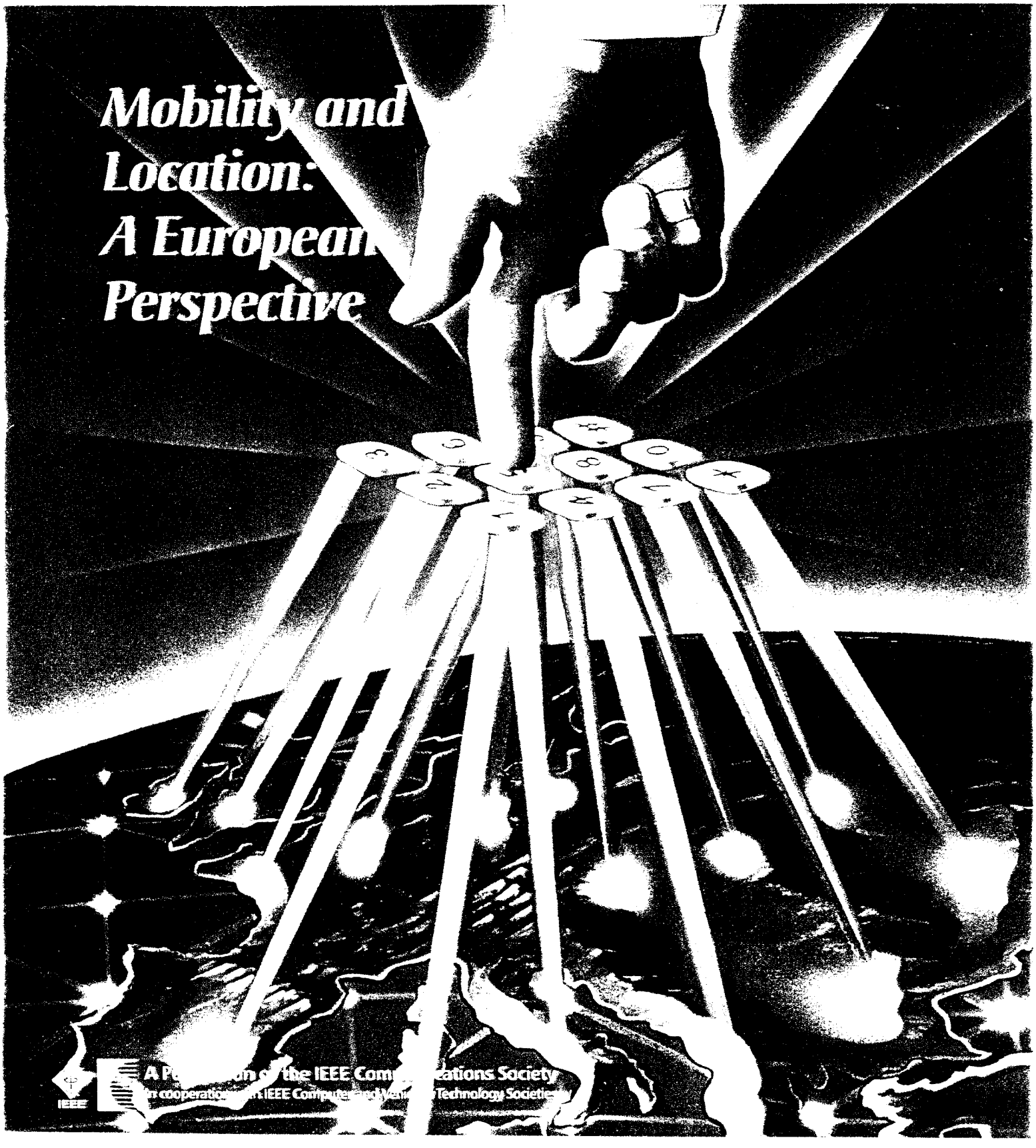
IEEE

October 1997
Vol. 4 No. 5

THE MAGAZINE FOR MOBILE COMMUNICATIONS AND COMPUTING

# Mobility and Location: A European Perspective



A Publication of the IEEE Communications Society
In cooperation with IEEE Computer and Vehicular Technology Societies

## Abstract

Wireless communication links are expensive and slow, and are therefore a scarce resource. Their usage should be subject to special scrutiny, especially when used by general-purpose application programs. We present the services of the "Mobile Application Framework" which supports conventional applications while using these links in three aspects: first, by optimizing outgoing communication calls via call interception and the spoofing of replies; second, by allowing disconnected operation through file logging and automatic modification reconciliation; and third, by scheduling and communication calls to allow the user to specify a monetary communications budget. As an example application, we show how the system can be used to mobilize a conventional e-mail system. Similar functionality to that provided by dedicated mobile e-mail clients and the ability to adapt to changing network environments is achieved without requiring any changes to the e-mail system itself or its user interface.


# Mobilizing Applications


**STEFAN G. HILD**
**UNIVERSITY OF CAMBRIDGE COMPUTER LABORATORY**
**AND IBM UK LABORATORIES**
**PETER ROBINSON**
**UNIVERSITY OF CAMBRIDGE COMPUTER LABORATORY**


Recent developments in data communications have provided the basis for new advances in mobile computing. While the potential of mobile computing is great and there is widespread interest in the subject, remarkably few applications are currently available. The reasons for this are numerous and diverse, cost being one of the major obstacles: data connections are expensive to set up and maintain, and offer limited throughput and disappointing line characteristics.

Current networking applications and their protocols, however, are written for comparatively high-speed and low-cost wired data links and often perform poorly across mobile data channels. Research is currently focusing on several areas. The first is the improvement of the underlying protocol stacks and communication protocols. Work is concentrating on adding functionality to the widely used Transmission Control Protocol/Internet Protocol (TCP/IP) suite of protocols through extensions and new TCP/IP options.

The second area of research concerns the applications themselves. "Mobile-enabled" versions of popular applications such as e-mail readers or Web browsers are now available. These programs are either heavily adapted versions of conventional wired applications or newly engineered programs, to allow them to function efficiently across mobile data links.

We briefly illustrate both these approaches in the introductory sections of this article.

A third approach are the so-called mobile enablers, small software products that allow conventional "wired" applications to function across data links with little or no modification to the application itself. The prime advantage of these enablers is that the user does not need to invest in new applications, and that a generic "enabler" can tap into an existing huge market of network applications.

Our work follows this third school of thought: in this article we describe our system of background services which support wired applications on mobile hosts. Further details of the system can be found in [1]; here, we show how the services can be used to "mobilize" wired applications, that is, to make them run efficiently across a broad range of wired and wireless networks. As an example, we apply our system to a conventional e-mail system.

## Protocol Adaptation

Although it has competition, TCP/IP is by far the most commonly used communications protocol. Most researchers have therefore concentrated their efforts in adapting protocols on this particular suite.

TCP/IP performs poorly across wireless links, mainly because it was developed for the Internet, a topologically static, wired network of comparatively high-speed links with constant transmission characteristics, used by a large number of users at the same time.

Wireless links, on the other hand, are often point-to-point, with much less favorable speed and transmission char-

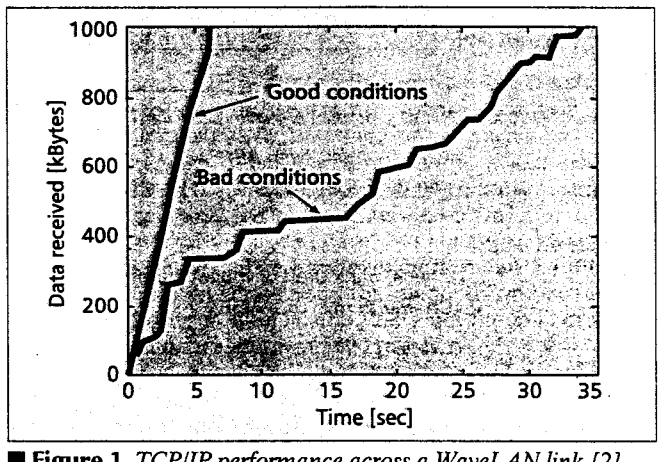



**Figure 1.** *TCP/IP performance across a WaveLAN link [2] under good and bad conditions.*



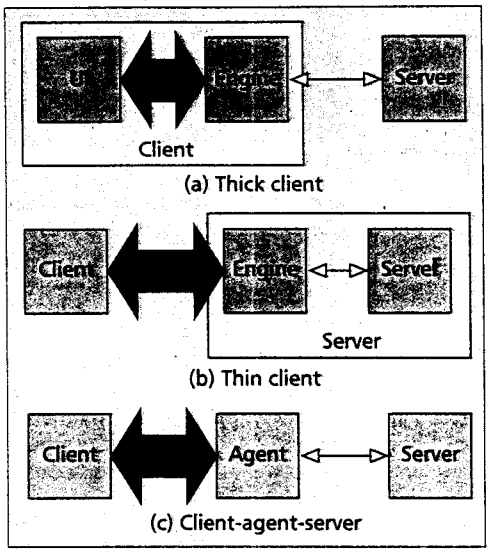
1070-9916/97/$10.00 © 1997 IEEE

acteristics. Many of the attributes of wired networks are hard-coded into the TCP/IP protocol; TCP/IP does not distinguish between the loss of the packet and a transmission error. Both are interpreted as indicating network congestion, and TCP/IP reacts by backing off and reducing throughput. In the wireless world, on the other hand, such packet losses are much more likely to be due to transmission errors. Forward error correction could correct these, but TCP/IP backs off, which does not solve the problem and leaves the otherwise unused expensive data link idle for significant periods of time. In Fig. 1 this effect is evidenced by prolonged flat segments in the transmission progression graph under bad conditions (i.e. signal strength 10% of maximum). These segments appear to become shorter as the transmission proceeds due to TCP/IP adjusting to the link characteristics and reducing the transmission window size. Unfortunately these adjustments are late and are insufficient.

Furthermore, TCP/IP is unable to take advantage of the point-to-point nature of mobile data links such as GSM channels. In the multi-user environment for which TCP/IP was designed, each packet requires extensive headers to allow routing, assignment to individual logical connection, and reassembly of the packets at the remote side. On point-to-point links most of these headers are redundant and simply waste scarce bandwidth.

Routing in TCP/IP is based on static and hierarchical IP addresses; this causes problems if mobile hosts connect to the network at different access points and do not want to change their IP addresses.

Extensions and modifications to TCP/IP have been proposed by several researchers; a good overview is presented in [3]. Briefly, Mobile IP addresses TCP/IP's routing problems by allowing "IP tunnels" between "mobile support stations" which ensure the forwarding of IP traffic to a mobile's current location [4]. "Indirect TCP" [5–7], "Semi-connected TCP/IP" [8], and "Daedalus TCP" [9] are examples of TCP/IP extensions which address the inadequacy of the transmission algorithms. All intercept conventional TCP/IP traffic at a gateway and modify the data stream. Of these, Indirect TCP is the most efficient: it actually terminates the TCP/IP connection at the gateway and forwards data to the mobile across a second connection using a modified and mobile-adapted TCP/IP stack. Although very effective, the scheme has problems preserving TCP/IP's end-to-end semantics.




■ **Figure 2.** *Possible client/server application architectures.*

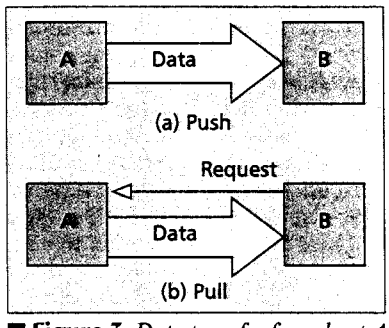
## Application Adaptation

Applications usually regard network bandwidth as free and plentiful. When adapting an application to expensive wireless links, programmers carefully use data compression wherever possible and minimize the connectivity required during a transaction. For such application reengineering, two aspects must be considered: thick and thin client architectures, and push and pull transaction modes.

*Thin vs. Thick Clients* – In client/server architecture, the functionality of the overall system is split between server and client, connected through some communications channel. Although there is no universally agreed definition for "thin" and "thick" clients, it is generally related to the exact division of this functionality. Our definition is based on the amount of communications required to complete a transaction: thin clients require more communication and interaction with the server, and consequently more connection time. On the other hand, they can be simpler and less powerful than thick clients, which process data locally and submit requests to the server as a batch. Figure 2 shows common client/server architectures and the required communications bandwidth between the functional components (i.e., the user interface, data processing engine, and server back-end).

Thus, thick clients minimize required connectivity. However, there are compelling reasons for retaining the predominantly thin-client style architectures of conventional wired applications. In particular, it removes the requirement for specially adapted hardware and software, and preserves the client's full functionality; "thick" clients tend to have less functionality than their "thin" counterparts, which rely on servers with adequate resources to support them.

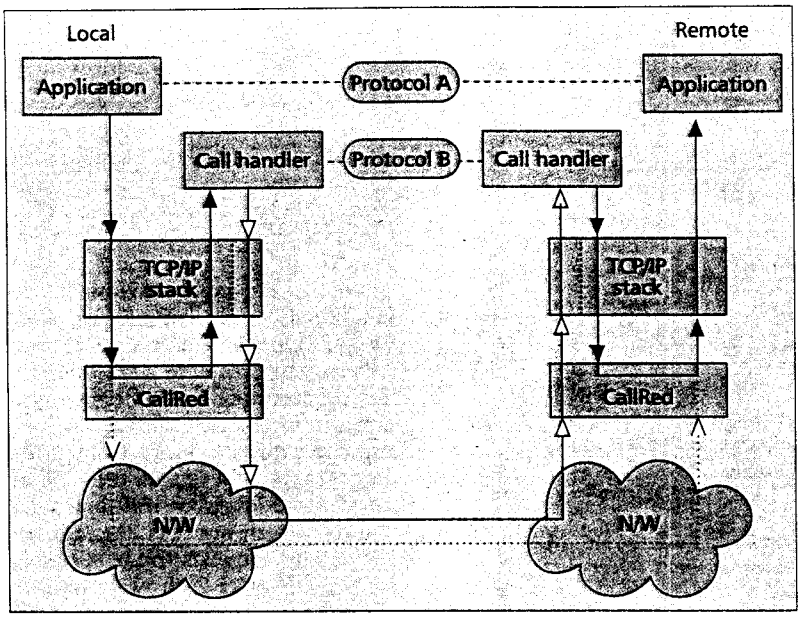
A third approach is to introduce "agents." These allow clients to remain thin and share some of the communications efficiency of their thick versions by separating the communications-intensive part of the application into the software agent which migrates through the network to the most favorable location for the communications to follow; the thin client communicates the task to the agent and "disconnects." The agent acts for and on behalf of the thin client and attempts to accomplish the task by communicating to servers while moving through the network. Once the task is achieved the agent moves back to the client to return the results. "Notus" [10] is a recent implementation of such a system; "Telescript" [11] is one of several industry standards for such agent software. "Proxy" servers as used in the World Wide Web are essentially nonmoving agents.




■ **Figure 3.** *Data transfer from host A to host B with push and pull architectures.*

*Push vs. Pull Architecture* – Passing data between host computers can be simple: the sending host connects to the receiver and transmits the data across the network connection. If the connection fails, the sender retries after a suitable period of time. This "push" model of data transfer operates satisfactorily as long as network downtimes are rare. However, a mobile host may only be connected to the network intermittently, and the sender may have to retry several times to establish a connection. If an exponential backoff algorithm is used, the chances of actually establishing a connection diminish with each failed attempt.

In such situations the "pull" model is more appropriate; here, the sender buffers the data and waits for the receiver to connect and request the transfer (Fig. 3).




■ **Figure 4.** *The call management system allows calls to be redirected to "call handlers" which transparently replace the application's native protocol with an arbitrary protocol.*

An interesting example of an application that has recently been moved from a push to a pull architecture is electronic mail. The original protocol for mail delivery, the "Simple Mail Transfer Protocol" (SMTP) [12], is implemented in the form of "mail delivery agents," which pass mail from host to host in push fashion: incoming e-mail is pushed to the next hop toward its destination. For delivery to the recipient SMTP has recently been replaced by the pull-oriented "Post Office Protocol" (POP) [13], and more recently with the "Internet Message Access Protocol" (IMAP) [14]. Both protocols buffer e-mail at the user's mail server, and wait for the user to log on and retrieve the buffered mail messages. IMAP also allows the user to selectively retrieve messages or parts of messages, for example, to avoid large appends if connected over a slow data link.

# The Mobile Application Framework

Of the two previously described approaches, adapting the application is far more efficient in improving the system's performance over wireless links than restricting improvements to the protocol level. However, it involves very specific modifications which can rarely be abstracted and reused in other applications. Thus, much of the effort has to be repeated for every application.

The "Mobile Application Framework" is a system of background services which allows conventional unmodified systems to function efficiently across wireless links. The services are transparent to the application, but not to the user, who is given constant information regarding the state of connectivity and the current cost of any communication calls.

We have designed and implemented three distinct services for the Mobile Application Framework, which we will briefly outline in the following sections.

## Call Management

Traditional call managers for mobile environments are usually bundled with mobile middleware. Middleware interfaces the application programming interfaces (APIs) provided by mobile networks, which are often proprietary to the TCP/IP APIs used by most application programs. The functions provided include simple connection-time accounting, and sometimes allow the user to specify a maximum idle period after which the link is taken down automatically. An example of such a call manager is "ARTour" [15].

The Mobile Application Framework's call manager supports all these functions, but has added facilities which allow networking applications to use specially adapted, highly efficient point-to-point protocols if they require.

In essence, the system intercepts the communication calls between the two communicating applications (similar to the interception done by Indirect TCP), and the call is terminated by a local "call handler." An identical call handler establishes a similarly short connection to the "target" server on the remote side of the connection (Fig. 4).

Call handlers are written specifically for each network service; for example, an "ftp"-call handler intercepts and optimizes communications calls from ftp clients; a "Web" call handler does the same for Web browsers. The fallback handler for services for which no specialized call handler is available sends TCP/IP traffic onto the network without any modifications.

The call handlers spoof the requests and responses from the server, while maintaining communications with each other through an arbitrary protocol.

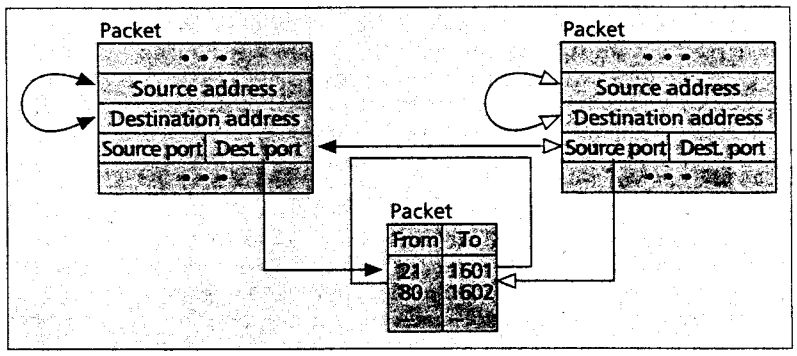
***Call Interception and Redirection*** – Indirect TCP intercepts outgoing communication calls by modifying the binary library containing the TCP/IP API calls. This relies on the library being linked to the client binary dynamically. We have implemented the call interception as part of the network interface for the serial line device. This automatically bypasses the redirection system if a fast network is available through an alternative (unmodified) network interface and simplifies its addition to a running system.

By examining the destination port number of outgoing packets the required network service is determined. For example, "ftp" clients connect to servers on port 21. We have to assume that the clients use these standard port numbers.

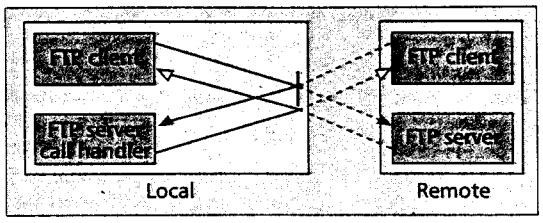
A "redirection" table is maintained as part of the modified network interface. Call handlers are user-space application programs similar to conventional network daemons, which "listen" on particular TCP/IP ports.

As shown in Fig. 5, redirecting a packet to the appropriate call handler is done by:

- Swapping the source and destination IP address




■ **Figure 5.** *Redirection of packets that originate from applications (dark arrows) and replies from the call handlers (light arrows).*




■ **Figure 6.** *Virtual and actual connections after redirection.*

- Replacing the destination port number with the call handler port number

After updating the checksums the redirected packet is returned to the local TCP/IP stack for delivery to the call handler. For reply packets the process is reversed and the packets are delivered to the original sender application.

Due to the redirection, the original sender and the call handler appear to be connected to a "virtual" sender and receiver: the sender's packets appear to go to, and be answered by, the originally targeted destination, whereas the call handler sees a connection from the sender at the original target's destination (Fig. 6).

*Call Handlers* – Call handlers improve communication efficiency in two ways: first, they avoid unnecessary connections by spoofing and batching; second, they use optimized link layer protocols instead of TCP/IP. For example, consider the ftp protocol: ftp login procedures are static, and replies can be spoofed effectively. File transfers through ftp are, for the reasons detailed earlier, inefficient over TCP/IP links when compared with thin-wire protocols such as "Z-Modem." A call handler will therefore use such a thin-wire protocol for the actual data transfer.

The amount of spoofing implemented with each call handler can vary, depending on its expected usage:

- An aggressive call handler for ftp may spoof the entire ftp login procedure and establishes connectivity only when files or directories not already held in a local cache are being requested.
- A Web call handler can scan pages for inline images and other appendages prior to transmission, and bundle these with the requested page. Idle lines can be used to prefetch pages, as proposed in [16].

In effect, call handlers provide platforms for improvements to individual protocols such as ftp or http, and make these available to all applications using these protocols.

## File Logger

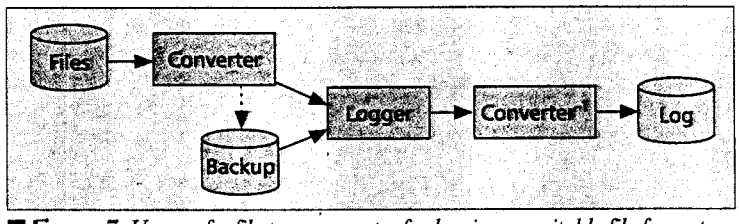
Data sharing between the stationary desktop machine and the user's mobile notebook is accomplished by the "File Logger," in a manner similar to network file systems such as Sun's NFS [18], Burrows' MFS [17], or the Andrew File System (AFS) [19]. Our system uses the same paradigm to allow conventional applications to manipulate shared data, but differs from wired file systems in various aspects.

Network file systems rely on fast and readily available data links to redirect read/write operations across the network. Caching is used to reduce traffic load on the network. Our system is predominantly disconnected: expensive data links are only established occasionally — in the meantime, a copy of the file has to serve all read/write calls. This allows data inconsistencies to develop due to simultaneous editing of the same file at disconnected locations. File locking (a summary of the most important locking policies is given in [20]) is usually employed to prevent inconsistencies, but in a mobile environment with prolonged disconnected periods this is rarely acceptable. Most mobile file systems, such as Tait's Mobile FS [21], Coda [22], or Microsoft's "Briefcase" system [23], therefore, allow inconsistencies temporarily, and rely on either the user or the application program to resolve conflicts once reconnected. "Mobile-aware" applications now frequently come with their own replication engine (e.g., "Lotus Notes" [24]).

We use our File Logging system as a convenient method of propagating file updates across file systems, and model the system for a user who has a stationary, fixed computer system ("desktop machine") and a mobile notebook computer ("mobile"), and uses the same applications on both. The service allows the user to replicate data files from the desktop machine onto the mobile, for example, the document the user is currently editing: while in the office, the document is updated on the desktop machine; while on the move, the same document is edited on the mobile system. Our File Logger tracks file updates and propagates these to the desktop machine (Coda calls this process "hoarding"). However, our system is unlike Coda and other mobile file systems in that:

- The File Logger does not replicate the entire file space automatically. The process of replicating a file and maintaining the necessary logs to ensure eventual resynchronization is expensive and unsuitable for large numbers of files. The system is built as a "private" user replication facility for a small number of important data files which are indicated by the user manually.
- It is built to function with conventional wired applications on running systems without requiring modifications to existing setups. The File Logger provides facilities for automatic conflict detection and reconciliation. The algorithms used can be augmented by the user to suit any application or usage environment.
- Files are shared "consciously": a group of users might edit a document jointly and share replicas of the original file. Here conflicts are relatively rare, since users arrange their access rights informally: they will edit different parts of the same document and so on.
- Reconciling replicas at different locations requires only a minimum amount of connectivity during log exchange. Total reconciliation of all replicas can be achieved through a series of partial reconciliation steps among subsets of all replicas. This is useful if some mobiles holding copies of the file cannot be reached because they are down, damaged, or simply outside the coverage area of the mobile network.

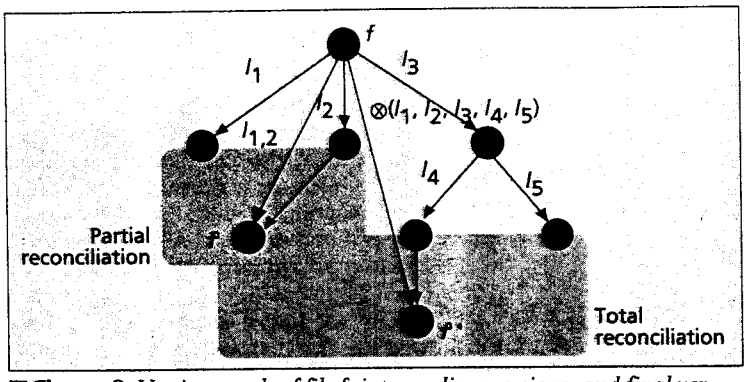
The system is user-configurable; in the following sections we outline the default policies and settings.




■ **Figure 7.** *Usage of a file type converter for logging unsuitable file formats.*

*Detecting File Modifications* – Detecting file modifications is nontrivial since no semantic meaning can readily be associated with the content of a file: all methods need to be file-type-independent. Hence, we adopt the most basic view imaginable, and read all files as a simple sequence of symbols.

A log of all modifications executed on the local copy of the file is accumulated by retaining a backup copy of the file and comparing the actual file with that backup at regular intervals. A derivative of the edit distance is used to model the file modification history in terms of *insertion* and *deletion* operations. We accommodate structurally more complicated file




■ **Figure 8.** *Version graph of file f, intermediary versions, and final version f''.*

formats by allowing conversion programs to be executed on the cached file prior to file logging (Fig. 7).

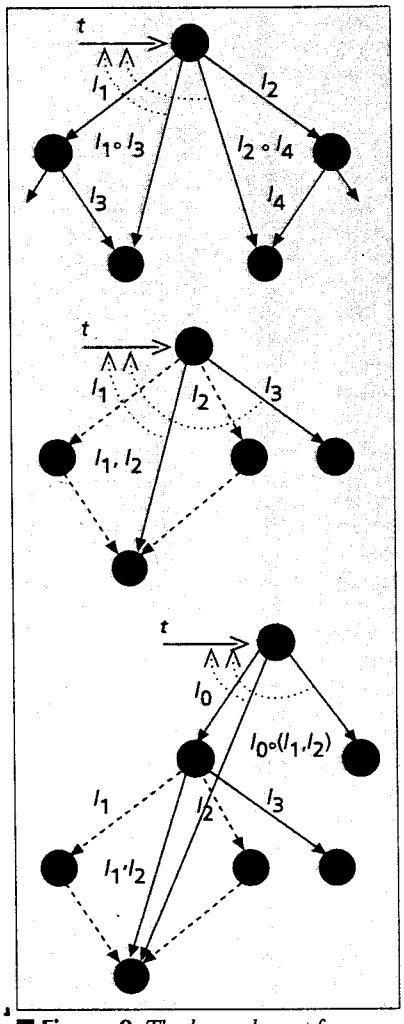
All modifications in the log are time-stamped to allow automatic conflict resolution. Interdependent modifications (i.e., modifications which extend or augment earlier modifications in the same log which are not separated by synchronization events) are collapsed into single modifications, thus removing the interdependency. This reduces log length and simplifies conflict detection which requires the order of the modifications in the log to be changeable. This is only possible if there are no interdependencies between individual modifications and the system merely needs the location information to be updated.

***Reconciliation*** – If all replicas can be contacted, a "total reconciliation" is executed which will lead to a new synchronized file copy to be generated on all sites. If one or more sites cannot be reached, a "partial reconciliation" is executed that generates a semi-synchronized version in the connected cluster which is still pending reconciliation with the remaining replicas. Figure 8 shows a number of versions of the same file as nodes in a directed graph; disconnected operation leads to divergence, represented by nodes appearing as children of the same version. Reconciliation steps lead to two or more different versions being united into a single node.

Once connected, all file copies determine the last point in time $t$ when they had identical file copies (i.e., the last total reconciliation). The relevant logs are generated from the recorded logs by removing all modifications recorded prior to $t$, and moving all modifications to a time frame relative to $t$ (Fig. 9).

The relevant logs are exchanged, and the hosts disconnect. Note that we do not require the clocks on all machines participating in the process to be synchronized, since we compare clocks at connection time and compensate for any differences in the received logs.

Since all modifications are relative to the same time frame, we can directly compare the location information to identify conflicts between individual modification from different users:




■ **Figure 9.** *The logs relevant for a reconciliation of the two marked (blue) replicas.*

- Two deletions conflict if the range of file positions deleted overlap.
- Two insertions conflict if both insert at the same file position.
- An insertion and a deletion conflict if the insertion inserts at a file position deleted by the deletion.

Conflict resolution proceeds by assigning priorities to the conflicting modifications. We have opted not to adopt the concept of a "master copy" which automatically receives higher priority than other copies (e.g., as in Coda), but we believe that during disconnected operation each host has an equal right to read and modify the file; upon reconnection, all modifications are equally valid.

We have implemented two priority assignment policies. The first is based on the time of modification —earlier modifications take preference over later ones. Taking the reverse approach would be counterintuitive in that modifications have a greater chance of committing successfully the later they are executed. Second, we envisage that the owner of the file can receive a certain "owner bonus." In contrast to the "master copy," the bonus is bound to the user executing the modification, not to the location of the replica.

We provide two default resolution policies. The "Total Invalidation" policy simply removes the lower-priority resolution. This preserves the file format (assuming that each individual modification preserves the file format), but leads to the conflict resolution step being nonassociative and may lose a large amount of information.

The "Partial Invalidation" policy removes only the conflicting area of the lower-priority modification. This is associative and removes only the minimum amount of information needed to resolve the conflict, but may not preserve the file's format.

Once the logs are merged and conflicts reconciled, the locally cached file is brought in line with the new log.

***User Interface*** – The File Logger provides a user interface which supplies constant health information regarding the state of the locally cached files. The health is indicated by giving a rough estimate of the file's divergence from other replicas, measured as a function of the amount of locally recorded modifications and the length of the disconnection. Figure 10 shows such a "file divergence meter" with annotations.

Also indicated is the level of divergence at which a reconnection is triggered automatically and the maximum divergence specified by the user for each individual file.

The user can demand immediate resynchronization by clicking the "SNC" button provided as part of the user interface.

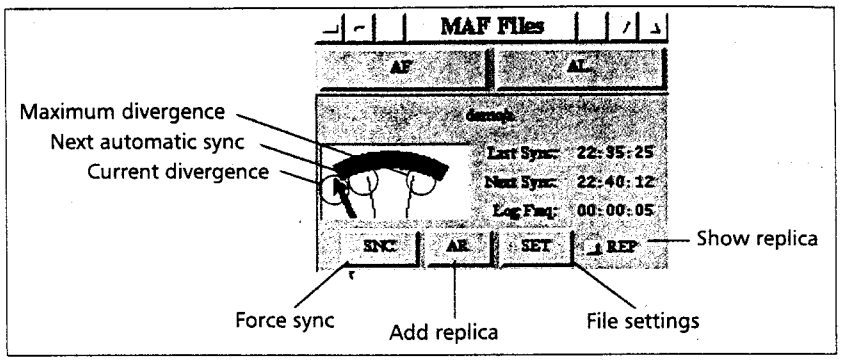
## Communications Scheduling

Networking applications make inadequate provision for the user to monitor and control communications cost. This is problematic when the high cost of mobile data

networks is considered. The "Scheduling" service provided as part of the Mobile Application Framework allows users to control communications cost by defining a monetary communications budget. User-initiated communications are monitored and the available budget adjusted accordingly. Maintaining the budget is achieved in conjunction with the File Logger, which can support disconnected operation for variable periods of time. This allows the scheduler to arrange reconnections freely and delay the transmission of data to:

- Save on overhead involved in establishing the data connection itself
- Take advantage of off-peak times when connection time is cheaper
- Apply compression more efficiently to batches of data

The effectiveness of our communications scheduler depends on the accuracy with which we can predict communications cost and the communications requirement.




■ **Figure 10.** *The file divergence meter presented by the File Logger.*

*Scheduling Strategy* – In the context of networking, "scheduling" focuses on scheduling packet transmissions to support different applications across homogeneous network environments through either "scheduling techniques," which determine the order in which packets are flushed from the buffer onto the network, or "dropping techniques," which determine which data packets from the buffer are dropped in favor of "higher-priority" data traffic; for a summary see [25, 26]. Such techniques are of limited value for our application.

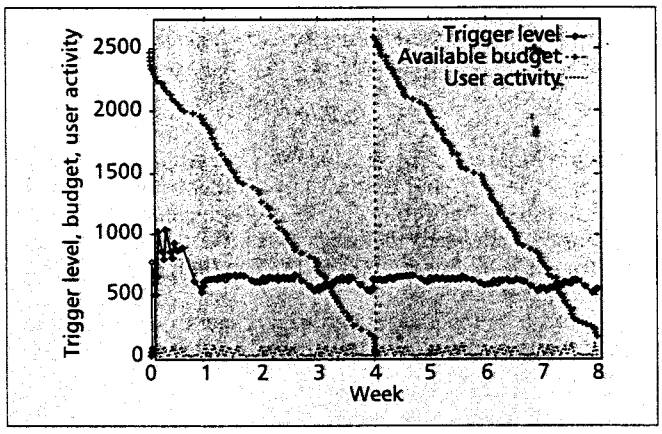
In simple terms, the scheduling algorithm determines when data is to be transmitted, given the predicted communications cost, the predicted communications requirement, and the available budget.

Our scheduler operates based on "Constant Maximum Backlog Scheduling," which reconnects as soon as the amount of logged data (i.e., the amount of data awaiting transmission) reaches a trigger-level $l$ which is determined by the cost of the network and the available budget. Thus, the user experiences at most $l$ outstanding data. Assuming that the predicted communications requirement at time $t$ is $D(t)$, the cost of communicating one unit of data is $C(t)$, the cost $c$ of $n$ reconnection during the budgeting period is

$$c = \sum_{m=1}^{n} C(D^{-1}(m \times l)) \times l,$$

where $D^{-1}(t)$ is the inverse of $D(t)$ and $n$ can easily be determined from the length of the budgeting period, the expected total communications requirement, and $l$.

The above formula is used to determine a trigger level $l$ for the available budget by iteratively evaluating $c$ for different values of $l$. Further rules are added to augment the behavior of the system. In particular, the user is asked to specify a maximum age for any logged data. This ensures that even small amounts of data are eventually propagated.




■ **Figure 11.** *Trigger level l, available budget, and user activity during the first two months.*

*Predicting User Activity and Network Cost* – For the above computation we have assumed that the cost of the network and the user activity can be predicted. Our system bases these predictions on past observations. The accuracy of these predictions depends on the regularity with which the mobile is used.

The observations are stored as running averages for each hour in a week in 7 x 24 tables. The user activity measurements are based on the size of the logs recorded by the File Logging service. Network availability is measured similarly in a separate table for each network, where each observation yields 1 to indicate network availability and 0 otherwise.

Given an order function $o(nw)$ which determines the priority of each network $nw$ against all other networks such that

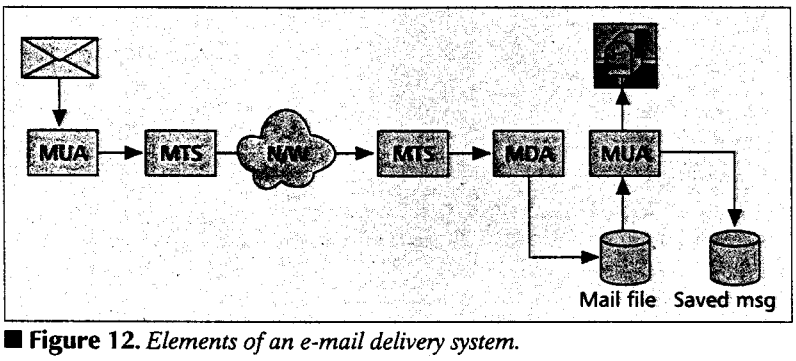
**if** $nw_1$ is preferred over $nw_2$ **then** $o(nw_1) > o(nw_2)$,

and the probability that the network $nw$ is available at some time $t$ in the future is recorded in table $np_{nw}(t)$, we can deduce the probability $p_{nw}(t)$ that we will be using $nw$ at time $t$:

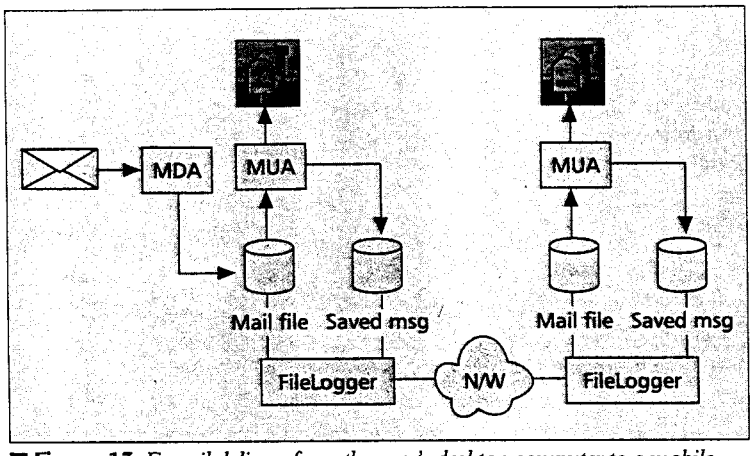
$$p_{nw}(t) = np_{nw}(t) \times \prod_{\forall y: o(y) > o(nw)} (1 - np_y(t))$$

or, in other words, the probability that the network is available at that time multiplied by the probability that all other preferred networks are not available.

We can now easily compute the expected network parameters by evaluating their expected value.




■ **Figure 12.** *Elements of an e-mail delivery system.*




■ **Figure 13.** *E-mail delivery from the user's desktop computer to a mobile notebook with the facilities provided by the Mobile Application Framework.*

*A Sample Budget* – Figure 11 shows user activity, the identified trigger level $l$, and the available budget during the first two months of a budgeting run. The graph clearly shows how the algorithm adapts to the usage pattern and identifies a constant trigger level around 600 bytes. The budget is met at the end of each month.

The jitters in the budget and trigger curves during the first week are due to the algorithm adjusting to usage and network availability patterns.

# An MAF Application: E-mail

Electronic mail is not only the oldest, but also one of the most popular and widespread applications within the Internet. In this section, we demonstrate how we can use the facilities within the Mobile Application Framework to mobilize e-mail.

As illustrated in Fig. 12, e-mail systems consist of three elements. "Mail user agents" (MUAs) are the actual e-mail readers, such as "xmh" or "elm." They display e-mail and provide tools for composing new messages and replies. If a message is ready for sending, it is passed to the MTA ("mail transfer agent"), which is responsible for the actual delivery of the message across the network to the target destination. At the target host the "Mail Delivery Agent" (MDA) delivers it to the user's e-mail file, from which it is displayed by the MUA. The MUA then provides facilities to archive the received file in alternative mail files.

## Motivation

Modern mail transfer agents now allow e-mail messages to be batched prior to transmission to reduce communications cost when connected across slow networks; some allow the user to indicate priority for each outgoing mail message, which in turn determines the maximum length of time the message may remain on the local queue.

However, these systems are frequently tied into particular MUAs to provide adequate control over the added features (e.g., Lotus cc:mail). As noted previously, specialized mobile applications are often not favored by users who would prefer a uniform application interface across all the platforms that they use, mobile or stationary.

## Implementation

The Mobile Application Framework can provide similar functions in conjunction with conventional e-mail systems. We now detail both receiving e-mail and mail delivery within a sample MAF architecture.

We assume that the user has a conventional e-mail account on a stationary computer (e.g., in the office), and that the mobile computer is used to access the same e-mail account. Thus, the mobile computer acts as an "outpost" for the stationary account.

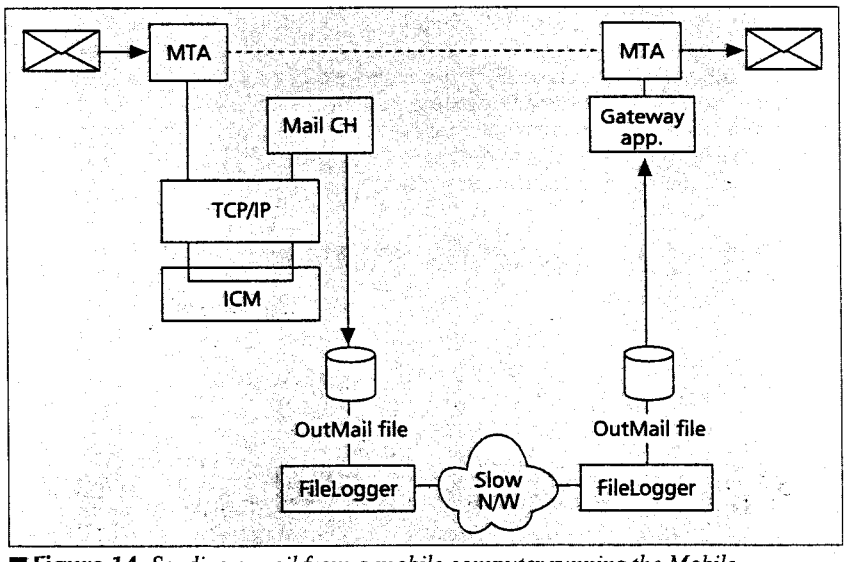
*Receiving Mail* – Forwarding e-mail from the desktop machine to the mobile is implemented easily. Figure 13 illustrates the principal architecture: to the left of the slow network is the user's desktop e-mail system. E-mail delivered to the MDA is appended to the e-mail file; the MUA then indicates that a new message has arrived.

On the right side is the user's mobile computer, with an identical e-mail system. As indicated in Fig. 13, e-mail forwarding between the desktop computer and the notebook is accomplished by use of the File Logger by instating a replica-original relationship between the e-mail files (inbox file and archives) on the mobile computer and on the stationary computer. By reading a new message it is deleted from the mail file and possibly moved to one of the mail archive files. The File Logger makes sure such activity is automatically duplicated on both hosts.

Through the File Logger the user has access to the built-in functions for communications budgeting and scheduling.

*Sending Mail* – Sending e-mail involves all three facilities provided by the Mobile Application Framework. We recall that, to send an e-mail, the message is passed by an MUA to the local MTA, which forwards the message across the network toward the mail's destination address through other MTAs.

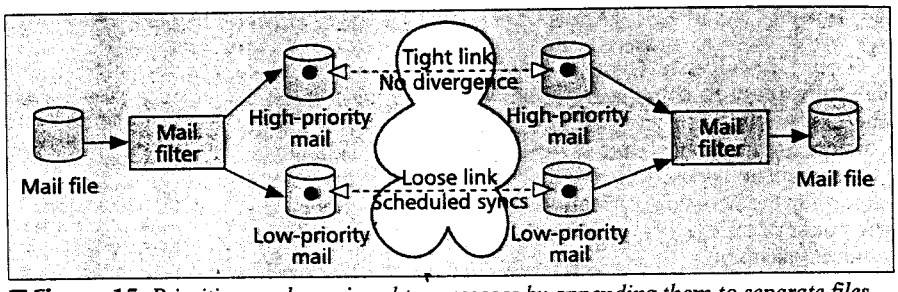
Figure 14 illustrates the sending of e-mail from a computer running the MAF. Here, to the left of the slow network is the mobile computer from which an e-mail is being sent. The MTA is an unmodified one which attempts to pass the message on to its counterpart on the next hop toward the destination of the message. By use of the call redirection system, the communication



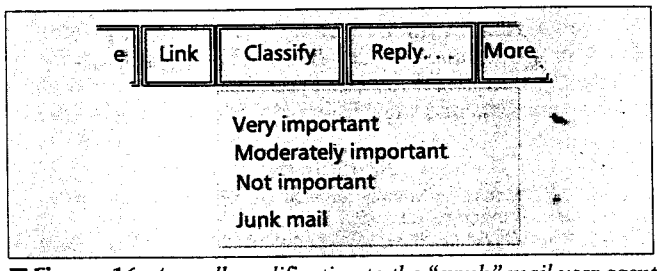

■ **Figure 14.** *Sending e-mail from a mobile computer running the Mobile Application Framework.*

call from the local MTA is intercepted and redirected to a mail call handler on the local machine, which in the following conversation mimics the responses from the remote MTA. Instead of passing the mail across the network, however, the call handler writes it into a locally held buffer file for outgoing mail. This file is replicated onto a stationary gateway system connected to a wired network (e.g., the user's desktop machine) using the File Logging facility. On the stationary system, a small gateway application passes appends to the replicated buffer file on to the local MTA, which continues mail delivery.

Again, the use of the File Logging facility and the related Scheduling allows the user to control mail delivery by the central budget defined for the Scheduler.




■ **Figure 15.** *Priorities can be assigned to messages by appending them to separate files and setting the replication parameters accordingly.*




■ **Figure 16.** *A small modification to the "exmh" mail user agent allows the user to classify outgoing mail messages through a small pulldown menu.*

## System Extensions

In our setup, budgeting facilities provided by the Scheduler allow the user to monetarily control how quickly e-mail is delivered and received. However, all mail messages are handled indiscriminately; the architecture does not yet allow the user to distinguish between high-priority messages, which ought to be delivered immediately, and lower-priority mail, which can await the next scheduled reconnection.

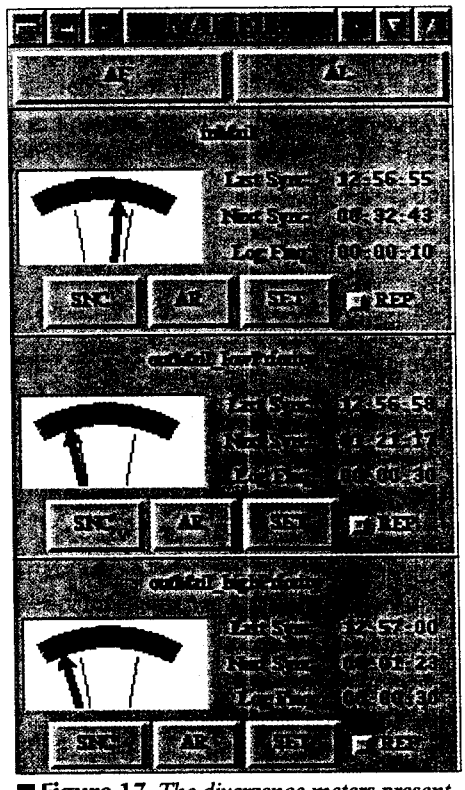
This problem can be alleviated easily. All outgoing and incoming mail is forwarded between the mobile and stationary host through files, which are handed by the File Logging system. The File Logger allows the user to define maximum divergence values; by installing separate files for high- and low-priority mail and setting the maximum divergence values accordingly (i.e., very small or zero for high-priority mail, and large for low-priority mail), high- and low-priority messages can be handed differently by appending them to separate files (Fig. 15).

Assigning priorities can be tricky; common e-mail filters usually rely on simple keyword matching. More elaborate "learning" e-mail filters (such as [27]) are still in the experimental stage. The model mail filter employed within the MAF relies on the user to indicate priority by including a line of the form `priority level: [1..9]` as part of the mail envelope. We have also made a small modification to "exmh" [28] to allow this to be done through a convenient pulldown menu (Fig. 16).

## User Interface

The e-mail call handler itself does not provide a user interface, but relies on the File Logger to provide user feedback.

Figure 17 shows the divergence meters for the mail files used in our sample implementation (one file for incoming e-mail, one for high-priority outgoing e-mail, and one for low-priority outgoing e-mail). Note that the "Next Sync:" times indicated by the File Logging subsystem reflect the user settings for acceptable divergence for the various files.



■ **Figure 17.** *The divergence meters presented by the File Logging subsystem provide feedback regarding the state of the mail buffer files for incoming e-mail, and high- and low-priority outgoing e-mail.*

# Conclusions

Wireless networks are a logical and necessary extension to wired data networks. Unfortunately, their cost, speed, and transmission characteristics are less favorable than those found in wired links, and existing networking applications are often inefficient.

In this article we have presented a system of background services that help to augment existing communications systems to make efficient and controllable use of mobile data communication networks by:

- Optimizing outgoing communication calls and providing means for transparently replacing one communications protocol with another
- Supporting disconnected operation on personal data files through a File Logging system which logs changes to data files, and reconciles these changes into other replicas held at other locations
- Allowing the user to specify a monetary communications budget, and by scheduling reconnection events such that minimum divergence is experienced for the given budget

We have also shown how the services of the Mobile Application Framework play together to mobilize e-mail, one of the most popular applications within the Internet.

Our solution has the advantage that no modifications are required to the software configuration of the e-mail system on the stationary or mobile machine. The Mobile Application Framework enriches the e-mail system with similar functionality to that provided by dedicated mobile e-mail clients while retaining the conventional user interface. The system

can fully adapt to changes in the networking environment to the extent that, if connected to a cost-free and fast LAN connection, it behaves exactly like a conventional system.

The article has used the e-mail system as an example application for the Mobile Application Framework. It is hoped that the reader can easily translate the techniques provided to other applications.

## Acknowledgments

The authors gratefully thank the members of the Rainbow Group at the University of Cambridge Computer Laboratory and the members of Keith Bryant-Jeffries' Mobile Communications Group at IBM Hursley for their helpful comments and suggestions.

## References

[1] S. G. Hild, "Managing Mobile Connections," Ph.D. dissertation, Comp. Lab., Univ. Cambridge, U.K., 1997.
[2] AT&T, *WaveLAN ISA Card-User's Guide*, doc. no. 008-0127332, rev. A, 1996.
[3] J. Ioannidis, D. Duchamp, and G. Maguire, "IP-Based Protocols for Mobile Internetworking," *ACM Comp. Commun. Rev.*, vol. 21, no. 4, Sept. 1991.
[4] C. Perkins, "IP Mobility Support," RFC 2002, Network WG, Oct. 1996.
[5] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," Tech. rep. DCS-TR-314, Dept. Comp. Sci., Rutgers Univ., 1994.
[6] A. Bakre and B. R. Badrinath, "Handoff and System Support for Indirect TCP/IP," *Proc. 2nd USENIX Symp. Mobile and Location-Independent Comp.*, Apr. 1995.
[7] A. Bakre and B. R. Badrinath, "Implementation and Performance Evaluation of Indirect TCP," *IEEE Trans. Comp.*, vol. 46, no. 3, Mar. 1997.
[8] J. Svarke, T. Reich, and B. Anderson, "Semi-Connected TCP/IP in a Mobile Computing Environment," *Proc. IMC '96*, Feb. 1996.
[9] H. Balakrishnan, S. Seshan, and R. Katz, "Improving TCP/IP Performance over Wireless Networks," *Proc. Mobicom*, 1995.
[10] S. L. Pope, "Application Support for Mobile Computing," Ph.D. dissertation, Comp. Lab., Univ. Cambridge, U.K., 1996.
[11] J. E. White, "Telescript Technology: Scenes from the Electronic Marketplace," General Magic, Inc., 2465 Latham Street, Mountain View, CA 94040, 1994; General Magic White Paper.
[12] J. Postel, "Simple Mail Transfer Protocol," RFC821, Network WG, Aug. 1982.
[13] M. Butler *et al.*, "Post Office Protocol: Version 2," RFC937, Network WG, Feb. 1985.
[14] M. Crispin, "Internet Message Access Protocol – Version 4," RFC1730, Network WG, Dec. 1994.
[15] P. Peinl and C. Kröll, "The Design of a Platform for Mobile Data Communication," *Proc. IEEE Global Data Networking Conf.*, 1993.
[16] V. N. Padmanabhan and J. C. Mogul, "Using Predictive Prefetching to Improve World Wide Web Latency," *ACM SIGCOMM Comp. Commun. Rev.*, July 1996.
[17] SUN Microsystems, Inc., "NFS: Network File System Protocol Specification," RFC 1094, Network Info. Ctr., SRI Int'l., Menlo Park, CA, 1989.
[18] M. Burrows, "Efficient Data Sharing," Ph.D. dissertation, Comp. Lab., Univ. Cambridge, U.K., 1988.
[19] M. L. Kazar, "Synchronization and Caching Issues in the Andrew File System," *Proc. Winter USENIX Conf.*, Feb. 1988.
[20] M. J. Carey and M. Livny, "Conflict Detection Tradeoffs for Replicated Data," *ACM Trans. Database Sys.*, vol. 16, no. 4, Dec. 1991.
[21] C. D. Tait, "A File System for Mobile Computing," Ph.D. dissertation, Columbia Univ., 1993.
[22] J. J. Kistler and M. Satyanarayanan, "Disconnected Operation in the Coda File System," *ACM Trans. Comp. Sys.*, vol. 10, no. 1, Feb. 1992.
[23] Microsoft Corp., "Microsoft Windows NT Workstation," document no. 94945-0996.
[24] Lotus Dev. Corp., "LOTUS cc:MAIL Technical Overview," rel. 6 & 7, 1996.
[25] J. M. Peha and F. A. Tobagi, "Evaluating Scheduling Algorithms for Traffic with Heterogeneous Performance Objectives," *Proc. IEEE GLOBECOM*, Dec. 1990.
[26] J. M. Peha, "Heterogeneous-Criteria Scheduling: Minimizing Weighted Number of Tardy Jobs and Weighted Completion Time," *Comps. and Ops. Res.*, vol. 22, no. 10, Dec. 1995.
[27] T. R. Payne and P. Edwards, "Interface Agents that Learn: An Investigation of Learning Issues in a Mail Agent Interface," *Applied AI*, vol. 11, no. 1, Jan. 1997.
[28] J. Peek, *MH & xmh: Email for Users & Programmers*, O'Reilly & Associates, 1991, Chs. 20–22.

## Biographies

STEFAN G. HILD [M] (Stefan.Hild@cl.cam.ac.uk) received a B.Sc. in computer science in 1991 from King's College, University of London. He is currently completing a Ph.D. in mobile data communications at the University of Cambridge Computer Laboratory, and is affiliated with IBM UK Laboratories, Hursley.

PETER ROBINSON (Peter.Robinson@cl.cam.ac.uk) is a lecturer in the Computer Laboratory at the University of Cambridge in England, where he is part of the Rainbow Group working on computer graphics and interaction. He is also a Fellow and Director of Studies in Computer Science at Gonville and Caius College. His research interests are human-computer interaction, where he works on the use of video and paper as part of the user interface, and electronic CAD for self-timed circuits. He studied mathematics at the University of Cambridge and continued with a Ph.D. in computer science. He is a member of the BCS and a chartered engineer.