University of Helsinki
Department of Computer Science
Series of Publications C, No. C-1995-15

Enhanced Communication Services for Mobile TCP/IP Networking

Markku Kojo, Timo Alanko, Mika Liljeberg, and Kimmo Raatikainen

Helsinki, April 1995

Report C-1995-15

University of Helsinki
Department of Computer Science
P. O. Box 26 (Teollisuuskatu 23)
FIN-00014 University of Helsinki, FINLAND

Enhanced Communication Services for Mobile TCP/IP Networking

Markku Kojo, Timo Alanko, Mika Liljeberg, and Kimmo Raatikainen

Department of Computer Science, University of Helsinki
Report C-1995-15
April 1995
22 pages

Abstract. Modern digital cellular telephones and portable personal computers have created a new, practical, and challenging platform for distributed information processing. We have designed and implemented a communication architecture for the efficient integration of wired and wireless networks. The architecture is based on the principle of indirect interaction. The mediating interceptor, the Mobile-Connection Host, is the bridge between the wired and wireless networks. The architecture significantly improves the end-to-end performance of TCP/IP-based communication between mobile nodes and fixed hosts. The introduction of application-specific mediators in the mobile node and in the Mobile-Connection Host provides means for enhanced application-level functionality, better user-level control of the wireless link, and increased performance.

Key Words: mobility, wireless connections, cellular telephone, mobile workstations, distributed applications

CR Classification: C.2.1, C.2.4, D.4.4

Contents

1. Introduction

Today, cellular telephones — such as GSM [Rah93] — provide access from portable personal computers to fixed networks. This makes it possible for nomadic users to access their organizations' information processing services from almost anywhere. Another important trend in information processing is the expansion of the user base. The majority of users will be professionals in fields other than computing. Therefore, it is extremely important that applications behave consistently in the mobile and the fixed network environments.

The advantage of using an existing cellular network like the GSM is that it takes care of terminal mobility: location updates, handovers, etc. However, the characteristics of cellular telephone links are quite different from those of wireline links. Cellular links have a low and variable throughput. They are also prone to sudden disconnections. Therefore, wireless links create problems for many existing applications, which are designed for fast and reliable connections. For example, problems in TCP/IP-based communication have been discussed recently in [AKL+94] and [CI94]. The characteristics of cellular telephone links also create needs for new kinds of functionality. For example, the user may want to have explicit control over the wireless link. Alternatively the user may want the wireless mobility to be as transparent as possible, unexpected disconnections included.

In [KRA94] we introduced the Mowgli[1] communication architecture, which is designed to solve the problems inherent in cellular telephone links and to take advantage of the special features of cellular telephone networks. The main design goals in this architecture are the following:

1) to allow the TCP/IP-based communication architecture to remain unmodified at existing hosts in the fixed network so that neither the existing network applications nor the protocol software on the fixed hosts need to be modified,
2) to provide an application programming interface which allows the development of new mobile (client) applications that are able to cooperate with existing services on fixed hosts, and
3) to obtain efficient use of the limited bandwidth and to improve fault tolerance and performance in spite of the intermittent connection.

In the Mowgli communication architecture the key idea is to replace the client-server paradigm with the client-mediator-server paradigm and to replace the peer-to-peer paradigm

[1] acronym of project name *Mobile Office Workstations using GSM Links*

with the peer-to-peer-thru-mediator paradigm. The mediator splits the end-to-end communication path into two parts: one over the fixed network, the other over the wireless medium. Similar approaches have been used successfully to improve the performance of TCP-based communication; for example [ABSK95, BB95, YB94]. However, our approach is more general and involves also higher levels of communication. It is important to take into account also the layers above the transport layer since most of the improvements can be achieved at the application level. Our mediator is an example of an intelligent agent which is an evolving approach to many problems in computing (see e.g. [ACM94]). In the area of mobile computing, exploiting such agents have also been proposed by others [ZD95, Wat95].

In the Mowgli communication architecture there are two kinds of mediators: agents in the mobile node and proxies in a fixed node on the border between the wireline and wireless networks. The essential features in our approach can be summarized as follows. Firstly, agents and proxies can be used to implement both generic services and enhanced functionality based on the application semantics. Secondly, operations can be executed in three different modes: 1) *strongly connected* (a LAN connection or a fast and reliable PSTN connection), 2) *weakly connected* (a slow and vulnerable cellular telephone connection), and 3) *disconnected* (the mobile node works temporarily as a stand-alone system). Thirdly, users can explicitely control how these three modes affect the execution of operations. The user can also request the operations to be executed according to the prevailing mode (with the right to change the mode).

The rest of the paper is organized as follows. In Section 2 we give an outline of user expectations in a wireless environment. In Section 3 we outline the Mowgli communication architecture. A more detailed description of communication services provided in our architecture can be found in Sections 4, 5, and 6.

2. Communication Services in a Wireless Environment

Usability of cellular telephone links — or of any wireless wide-area links — in existing and future applications requires communication services different from those provided in fixed networks. The Mowgli communication architecture and its services have their origins in the typical sandwich structure: between user needs and the constraints posed by available technology. The user needs and technology constraints leave a gap that must be filled by the communication services.

2.1 Pragmatic Perspective to Communication Services

As noted in the Introduction, the problems related to the use of wireless links have been discussed recently in several articles. In a mobile system based on cellular telephones — GSM in our case — we consider the most serious problems to be 1) the possibility of unexpected disconnections, 2) the modest performance (round-trip times around 1 second and transfer rates of less than 900 bytes/sec [AKL+94]), and 3) the inadequacy of the end-to-end flow control in the current TCP protocol. Although future systems, such as UMTS [Com95], will provide broadband capabilities (2 Mbits/sec), the performance of mobile telephone links will remain modest when compared to ATM and to future wireline links.

The other side of the coin reflects the user needs. In a typical usage scenario, the nomadic user wants to read the new mail messages which have arrived in his/her mailbox in the home network. The first step is to get the headers of all unread mail messages. While reading the headers the user selects the messages to be read. The user may want to specify the order in which the messages are to be received: the important ones first, small before large, for example.

Suppose that a mail message relates to some interesting product announcements to be found on a WWW page. The user activates a WWW browser to fetch that page. Unfortunately the WWW page is not designed to be read over a slow link. It contains several inline images, some of them of considerable size. A mobility-aware browser may inform the user of the expected transfer times. The user can then tell the system to drop the large images and to fetch the small and medium size images in the background. The larger ones will be transferred only if explicitly requested by the user.

Another mail message informs the user about an important document available at an FTP server. Now the user activates the FTP client in order to download the document. Since the document is important, the user wants to give to the FTP transfer a higher priority than to the rest of the transfers, which are still flowing to the mobile node.

The user needs to answer to some of the mail messages. The user writes the reply messages and sends them. Because some reply messages are long — they contain attachments — and the link is temporarily weak, a queue of as yet unsent mail messages builds up. Later during the session the user notices that the power of the battery is rapidly decreasing. In order to be sure that at least the most urgent messages will be transferred, the user changes the transfer order of the remaining mail messages.

Yet another important aspect: in the cellular networks where the billing is time-based — as in the current GSM — the user does not necessarily want to keep the link open while reading the mail and preparing the answers. Anyway, keeping the link open consumes power. So the user may want to instruct the system to close down the telephone connection immediately when all incoming mail has arrived and to re-establish it when a sufficient amount of transferrable messages has accumulated.

To summarize the scenario: the user needs a fine-grained control over the transfer. However, since the majority of users will not be computing professionals, the control interface must be based on concepts, terms, and allegories that the user can easily understand.

2.2 Conceptual Model of Application

In the mobile node one or more applications can run in parallel. The applications we are interested in are those which involve communication with one or more peers in the fixed net. Such an application can be considered as a component in a multi-component application distributed over the network. The component in the mobile node can have the role of a client, server, or peer.

When we examine the communication, we can model the application used by the end-user as a sequence of operation invocations. What an "operation" really is depends on the application and on the used level of abstraction. In our context it is reasonable to take into account only the operations that can cause information transfer between the mobile node and hosts in the fixed network. In our model we also assume that an operation is something that an end-user understands as an operation: typically an action which the end-user invokes by pressing a button or by entering a command. Examples of operations include "retrieve the headers of new mail messages," "retrieve a file," and "deliver the pending mail messages."

From the communication control point of view, the information to be transferred is either structured or unstructured. The basic element of structured information is called a *Data Transfer Object* (DTO). An operation invokes the transfer of one or more DTOs, and the transmission of each can be controlled individually. Examples of DTOs are files, print jobs, and inline images of HTML pages, just to name a few.

There are many applications where the communication cannot be modelled as a sequence of discrete, individually controllable DTO transfers. In our model such communication, consisting of the exchange of unstructured information, is called a (user-level) *information flow*. Examples are the information transfer during a Telnet session and streams of multimedia data. The basic difference between a DTO and an information flow is that within an information flow there are no meaningful ways to provide transmission control over the individual pieces of information.

When the communication is based on the exchange of structured information, a DTO is the basic operational unit of transmission control. Each DTO has its own attributes which specify the details of the control. A DTO can be scheduled for sending. A DTO can be deleted. A DTO can be suspended to wait for more favorable conditions. In general, a DTO is something that the user considers as an independent unit of information, the transfer of which he/she may want to control.

The user can also treat the DTOs as groups: for example, the DTOs of one operation or of one application can be treated as a group. The concept of group is used to simplify the user control: the user can control the whole group of DTOs as a single operational unit.

In addition to information transfer, there are control operations that affect the wireless link and the functionality of the associated mediators (the agents and proxies). The major difference between the control operations and the transfer operations is that a control operation is instantaneous — it cannot be stopped and resumed like a transfer operation. When a control operation is not local, information transfer takes place, but this information is just an atomic "control signal".

2.3 User Control over Information Transfer

As the usage scenario in Section 2.1 depicts, the usability of a slow wireless link requires fine-grained control and a sophisticated control interface. An operation intended to affect the transfer of user information can be specified to involve either a single DTO, a group of DTOs (for example, all DTOs of an application), or an information flow. The control interface provides the means by which the user can specify which DTO, which group of DTOs, or which information flow is intended.

First we consider the transfer of DTOs. The transfer can be in one of the four states: 1) *deferred* (the transfer of DTO may not start), 2) *ready* (the transfer of DTO may start), 3) *in_transit* (the transfer of DTO is in progress), and 4) *suspended* (the transfer of DTO may not proceed). Each DTO has several attributes that are used to control the transfer. The main purpose of these attributes is to specify whether or not an immediate transfer is allowed and — when the transfer is allowed — to specify the level of its urgency. Below we discuss the most important attributes.

Conditions. Each DTO has a list of condition attributes which are the basis for the transfer state. Typical conditions include:

time — transfer may start after the given time of day,

link quality — transfer is allowed only if the transfer rate is expected to exceed the given threshold,

connection type — the value "strongly connected" means that the DTO may not be transferred when the mobile node is in a weakly connected mode,

wait — the user has explicitly forbidden the transfer until further notice.

The condition attributes are evaluated by the Mowgli communication system. Changes in their values immediately affect the concerned DTOs. For example, the effects of link disconnections and reconnections can be handled through this attribute system without any explicit intervention by the user.

The condition attributes determine the transfer state of the DTO. The DTO is in the *deferred* state when at least one condition attribute is "against" the transfer. The DTO is moved into the *ready* state when none of the condition attributes prevents the transfer. The user can change the state either by changing condition specifications or by using an explicit command.

The *suspended* state means that the transfer has been started but has later been temporarily stopped, either because of a user intervention or because of an internal change in the conditions. An example of such changes is deterioration of the link quality. The change back to the *in_transit* state from the *suspended* state can result from a user intervention or from an internal change in the conditions.

Transfer priority. The attribute specifies the order in which the DTOs in the *ready* state are scheduled for transfer. Furthermore, when the DTO is in the *in_transit* state this attribute determines the DTO's share of the capacity of the underlying wireless link. Priorities are very useful when we want to guarantee reasonable response times to DTOs of an interactive application even if there are DTOs of bandwidth-intensive background applications (e.g., file transfers) using up the link capacity.

Action in the case of an implied state change. A control operation — concerning the wireless link, a DTO, an attribute, etc. — can even indirectly lead to a change in the transfer state of a DTO. The attribute specifies the action to be taken when an implied state change occurs. For example, when the user requests for a disconnection while the transfer is in progress — i.e., the DTO is in the *in_transit* state — the possible alternatives are 1) to postpone the disconnection until the transfer has been completed, 2) to change the transfer state from

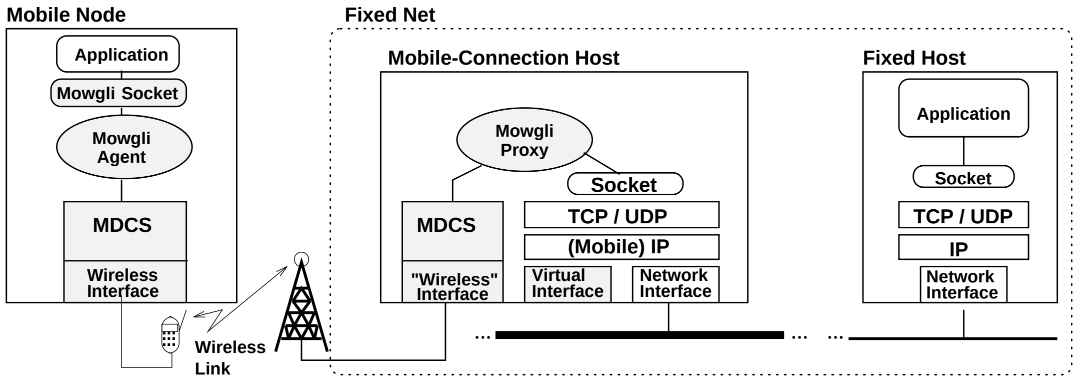

Figure 1: An architectural model of communication for mobile nodes

in_transit to *suspended*, 3) to terminate the transfer or 4) to request for further guidelines from the user.

Action in the case of an unexpected state change. The attribute specifies the action to be taken when the state change has not been caused by a user action. An unexpected disconnection of the link is a typical example. The possible alternatives differ from those mentioned above in that a postponement of the disconnection is not possible. Instead, an automatic recovery can be attempted.

An information flow has a similar set of attributes. In the *in_transit* state the data within the flow is transferred immediately. In the *suspended* state the outgoing data can be directed to a local buffer, and from there it can be retrieved when the flow is again in the *in_transit* state. The transfer priority of a flow is applied to those units of information which the system chooses to use in the transmission. Thus, it affects the flow's share of the link capacity.

3. Architectural Overview

The Mowgli communication architecture is based on the idea of indirect interaction [BBIM93, KRA94]. The mobile node is connected to a *Mobile-Connection Host* (MCH) through a wireless link; see Figure 1. The Mobile-Connection Host is a fixed-net host which provides the mobile node a connection to the wireline Internet. A proxy on the MCH — together with a cooperating agent on the mobile node — acts as a mediator for all the data delivered between an application on the mobile node and its counterpart somewhere in the fixed net.

This approach allows us to implement two separate, but flexibly cooperating data communication subsystems: one wireline oriented, the other wireless oriented. Existing TCP/IP

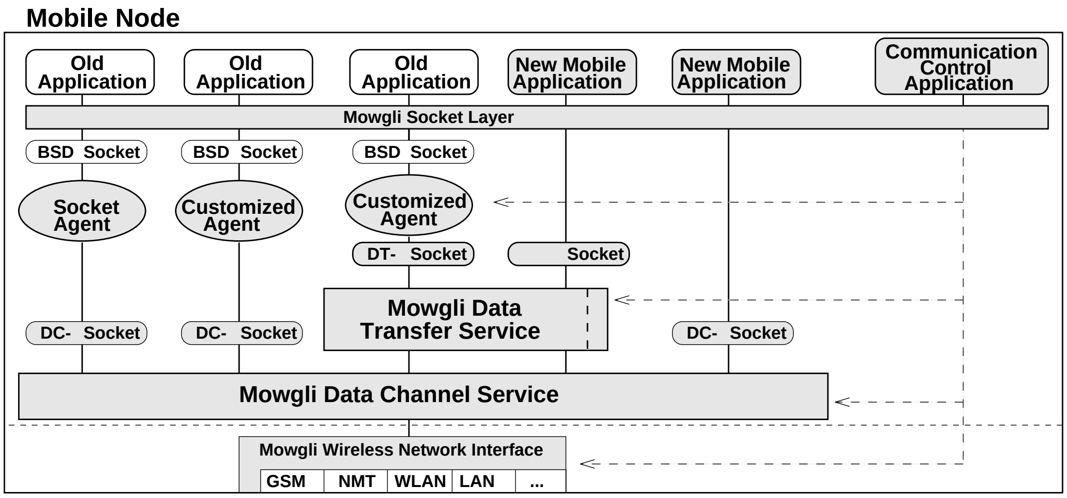

Figure 2: Layering of services offered through the Mowgli Socket Interface

protocols are used over the fixed net between the MCH and the fixed hosts. This implies that the TCP/IP protocol software in the fixed net remains unmodified, but special purpose protocols can be used to transmit data over the wireless medium.

For each mobile node that communicates through the MCH, the MCH dynamically allocates a virtual network interface. The MCH assigns the IP address of the mobile node to the virtual interface. Thus, the MCH is able to play the role of the mobile node when communicating on its behalf with the fixed hosts.

The Mowgli communication architecture has been designed to be integrated with the developing Mobile IP standard [Per95]. To accomplish this, the MCH takes care of registering with the Home Agent on behalf of the mobile node. The MCH can implement either the functionality of the Foreign Agent or the functions needed to interact with a local Foreign Agent, thus acting as the mobile node. Once the mobile node has been connected to the MCH, the cellular telephone network hides all further mobility of the mobile node. Therefore, handoffs during a single wireless connection do not require any support in the Mowgli software. Furthermore, the IP mobility support is not required at all if the mobile node always connects to an MCH in its home network. However, in many circumstances it is more economical and practical for the nomadic user to connect to the nearest available MCH.

An application programming interface (API) called the Mowgli Socket Interface is offered to the applications on the mobile node. The Mowgli Socket Interface provides access to communication services defined on several levels of abstraction. Figure 2 depicts the conceptual organization of the communication system on the mobile node. The basic communication services include the socket operations available in the traditional BSD socket interface [LMKQ89]. Hence, existing applications that use TCP or UDP sockets can be executed on the mobile node without modification. In order to achieve this, a generic socket agent is used as the mediator.

The Mowgli sockets also provide enhanced features not available in conventional TCP/IP sockets. The enhanced services are provided on three different levels: the agent level, the data transfer level, and the data channel level. The Mowgli Socket Interface defines a number of new socket attributes and operations. The new socket attributes can, for instance, be used to assign a priority for the data passing through the socket, as well as to control the failure semantics of the socket in case of unexpected disconnections of the wireless link. Furthermore, the special attributes can be used to assign a customized agent — and a customized proxy on the MCH — to a BSD socket. In this way, we can introduce higher-level functionality to the sockets: the agent and the proxy can intercept application-level messages and optimize the traffic over the wireless link.

Most of the new socket attributes can be configured statically. External configuration of the attributes is necessary in order to assign appropriate values to the attributes for existing applications without modifying the source code of the application. However, a direct and dynamic way of control is needed for the applications that are aware of mobility.

A set of control functions is introduced in the Mowgli Socket Interface to allow application programmers to write applications that directly control communication. The functions include explicit operations for setting up and closing down the dial-up connection as well as operations for monitoring and controlling link level behavior.

The Mowgli Socket Interface provides communication operations for new mobile (client) applications aware of the impacts of mobility and wirelessness. When new applications are implemented for mobile users, the functionality of the customized agent can be integrated into the client software. Such mobile clients can cooperate directly with the customized proxy on the MCH. This approach amounts to splitting the traditional client program into two parts — one on each side of the wireless link. The new socket operations are specifically designed to be used for communication over the wireless (telephone) link with an application-specific proxy. The same operations are available when implementing communication between agents and proxies. These operations offer access to the communication services available in the Mowgli Data Transfer Service and in the Mowgli Data Channel Service; see Figure 2.

The *Mowgli Data Transfer Service* (MDTS) offers facilities for the efficient transfer of Data Transfer Objects (DTOs) over the wireless link. The MDTS provides an API for invoking the transfer of a DTO, for managing transfer queues, and for changing attributes of specified DTOs. These operations are provided through a Data Transfer socket (DT-socket). Services provided by the MDTS are described in Section 6.

The *Mowgli Data Channel Service* (MDCS) takes care of transmitting data over the wireless link between the mobile node and the MCH. All communication at higher levels of abstraction depends on the data transport services offered by MDCS. The communication is based on the concept of priorized data channels. The programming interface to the MDCS is provided in the form of a low-level Data Channel socket (DC-socket). A more detailed description of the MDCS is given in Section 5.

An application program called *Communication Control Application* provides a user interface to control all communication. This program exploits the control functions in the Mowgli Socket Interface. The control interface also enables the reconfiguration of some of the socket attributes as well as the on-line modification of certain attributes.

To give the reader some insight into the benefits of using customized agents and proxies with existing applications, we examine two concrete examples: electronic mail and WWW browsing.

Since a cellular telephone link can be expensive to use, sending e-mail is a good candidate for deferred delivery of DTOs. Thus, a disconnected mode of operation is the norm rather than the exception. It does not make sense to establish an expensive wireless link each time the user sends a mail message. As an example of deferred delivery of DTOs, we have implemented a special mailer agent and proxy that can defer sending mail messages according to several user-configurable parameters. For example, the transfer can be triggered at a certain time or when a sufficient number of messages has accumulated.

The HyperText Transfer Protocol (HTTP) used by the World-Wide Web is grossly inefficient over links with long round-trip times. A new TCP connection is established for each HTTP request, and since several HTTP requests are normally required to fetch a single hypertext document, the associated overhead quickly amounts to an excessive delay.

In [LAK+95] we describe how a special Mowgli HTTP Agent and Proxy can solve this problem. Our experiments have shown that such an approach reduces the user-level response time by about 25–50 % in WWW browsing. The basic idea is that the agent and proxy communicate with each other using an optimized version of the HTTP protocol. Only a single request is needed to fetch a hypertext document. The agent and the proxy employ low-overhead data channels for delivery of the HTTP requests and responses over the wireless

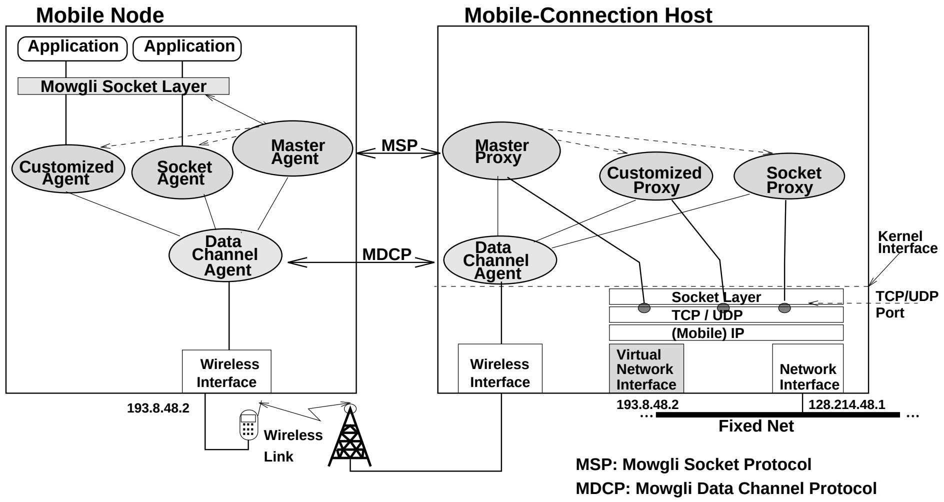

Figure 3: Software components in the Mowgli Sockets implementation

medium. In addition to reducing the protocol overhead, the agent and proxy can compress the data transferred over the wireless link in various ways, depending on the presentation format of the transferred document object. For example, the proxy can reduce the resolution and number of colors of large images before sending them to the agent.

4. Communication Support for Existing Applications

In this Section we describe in more detail how existing applications can take advantage of the communications services of the Mowgli communication architecture. We first describe the implementation of the basic communication service available at the application program interface. We then describe some of the special attributes available in the Mowgli Socket Interface and how these attributes can be configured for existing applications.

4.1 Implementation of the Basic Communications Services

The basic services include conventional TCP/IP communication through sockets. The Mowgli sockets provide functionality that is downwards compatible with the BSD sockets. The conventional socket types, such as stream and datagram sockets, are supported. To be more specific, the Mowgli sockets provide the same functionality as corresponding sockets provide in the Internet domain (PF_INET). Support for network database operations, such as retrieving the IP address for a given host's domain name, is also provided. The software components that are needed to implement the functionality of the traditional UDP and TCP sockets on the Linux operating system are shown in Figure 3.

When an application invokes a socket operation, the Mowgli socket layer notifies the *master agent* of the mobile node. The master agent forwards the socket operation to the *master proxy* on the MCH, which then performs the socket operation on behalf of the mobile-node application. The delivery of the information about the socket operations between the master agent and the master proxy is controlled by the Mowgli Socket Protocol (MSP).

For each TCP connection a new socket agent on the mobile node and a new socket proxy on the MCH are created. The agent and proxy cooperate to deliver the data that flows within the stream connection. In their simplest form, the socket agent and proxy act as relays for the intervening traffic. In addition, they take care of retaining out-of-band data "markers" within the data flow. The agents and proxies communicate using the data channels provided by the Mowgli Data Channel Service. It should be noted that when the TCP connection has been established, the master agent and master proxy are no longer involved in the delivery of normal data. Instead, the socket agent and proxy take care of all transmission activity.

The master agent and master proxy take care of delivering UDP datagrams. Separate agents and proxies are usually not created. The exception to this rule is when an application is configured to make use of customized agents and proxies to transfer UDP datagrams.

In most cases the master agent informs the master proxy immediately when the application requests a socket operation. However, in some cases the master proxy does not need to be informed immediately. For example, a request to create a new TCP socket and bind it to a local address will not be delivered to the proxy until the application calls for a *connect* or *listen* operation. In this way, we can avoid unnecessary round trips over the wireless link.

One novel feature in the implementation of Mowgli sockets is related to the closing of sockets. When the application on the mobile node closes its socket, a control message with high priority is sent to the MCH to reset the data channel. In this way the data flow over the slow link to the mobile node is terminated as soon as possible. This can be valuable on many occasions when sockets are closed unexpectedly. For example, when the user interrupts an incoming FTP file transfer, there may be several data segments in transit towards the mobile node and the wireless link may be blocked for several seconds.

We have a prototype implementation of basic communications services running on a Linux platform [AKKP95]. Almost all of the software is implemented as user-level processes, including all the agents and proxies. Only about 500 lines of C were inserted into the Linux kernel of the mobile node so that the existing socket layer can interact with the master agent. By modifying the kernel we ensure that the kernel socket call interface remains unmodified and that no modifications are needed in existing applications. Since we have not yet integrated the Mobile IP routing in our prototype, all the functions needed on the MCH have been implemented as user-level software, except the virtual network interface for which we used the existing extension of the loopback network driver for Linux. Incorporating the Mobile IP support into the MCH software will result in some kernel-level modifications but also enables a more convenient solution.

4.2 Special Attributes in Mowgli Sockets

The BSD socket service available in the Mowgli Socket Interface is augmented with enhanced functionality by a set of special attributes. These attributes are implemented as new socket options. The new socket attributes are used to control both communication through the socket and the establishment of lower-level connection. Below we describe some of the features affected by the new socket attributes.

Socket priority. Each stream socket has a priority which determines its share of the capacity of the wireless link. A datagram socket can also have a priority. If a datagram socket has not been assigned a priority, each UDP datagram originating from or destined to that socket has a priority based on the source or destination of the datagram.

Wireless connection establishment. When there is data to be delivered, the wireless connection can be automatically established by the Mowgli system without explicit user intervention. The alternative is that the user is prompted for permission to establish a connection.

Mediator selection. A customized agent and/or proxy can be assigned to deliver the data sent from or destined to the socket. The customized agent and proxy can filter data or exploit application protocol specific compression methods. A customized proxy can also perform enhanced operations on behalf of the application on the mobile node; this permits a reduction — sometimes even a substantial one — of the traffic over the wireless link.

Exception handling. On both sides of the wireless link each socket has an associated buffer and associated state information to be used in recovery after unexpected link disconnections. The state information contains a timer which can be set to limit the keep time of the information after a disconnection. An automatic reconnection can be requested with the reconnect attribute.

4.3 External Configuration of Mowgli Socket Attributes

Existing applications can not call *setsockopt* socket operation in order to set values for the new attributes. Thus, the ability to configure the attributes is a necessary prerequisite to the ability to control communication in an existing application. A configuration table contains the default values of the attributes to be automatically assigned to a socket when it becomes operational. The configuration is based on local and remote ports (and IP addresses) which are associated to the socket when a stream connection is created (TCP sockets) or when a user datagram is sent or received (UDP sockets).

Separate configurations can be defined for active and passive sockets. A stream socket is defined to be active if it is used to initiate a new stream connection, and passive if it is used to listen to incoming connection requests. A datagram socket is defined to be active if it is used by a client to send request messages and receive associated reply messages, and passive if it is used by a server program to receive service requests and to send the replies. Typically, passive sockets are used in servers, and active sockets are used in clients.

For an active stream socket, attributes are usually configured by using the remote port to which a TCP connection is to be established. Similarly, for an active datagram socket the attributes to be used are usually associated to the remote port to which the request message is sent. Attributes of a passive socket are usually configured by using the local port the socket listens to for incoming TCP connection requests, or the local port from which the socket receives UDP datagrams.

5. Mowgli Data Channel Service

The Mowgli Data Channel Service (MDCS) provides data channels for transmitting data over the wireless link between the mobile node and the MCH. Two types of data channels are provided: stream channels and message channels. A stream channel offers a reliable octet stream between the end-points of the channel; it is used, for example, for TCP sockets. A message channel offers a reliable and sequenced delivery of messages; it is used, for example, by the master agent and master proxy to deliver UDP datagrams. Both channel types offer bidirectional data flow. Flow control is provided independently for each channel.

A data channel has number of attributes that are used in controlling the behavior of the channel. Values for these attributes are assigned when the channel is opened, but the values can be changed later at any time through the management functions in the MDCS programming interface. For example, each data channel has a priority. The priority is assigned separately for both directions. The priority of a TCP socket is assigned to the corresponding stream channel, and the priority of a DTO can be assigned to the channel over which it is delivered.

Other attributes are mostly used in controlling the on demand establishment of wireless connection and the recovery actions when the wireless link breaks down. A data channel is able to retain the state information of the ongoing transmission so that in the case of an unexpected disconnection the interrupted transmission can later be continued. Maintenance of the state information is separately controlled for each channel by attributes that specify for example how long the state is to be retained. However, a data channel does not survive the crash of the MCH or the mobile node. To obtain more fault-tolerant service the Mowgli Data Transfer Service should be employed.

Channels can be organized into groups, and these can have their own attributes, for example a group priority. The group priority defines the channel group's share of the available bandwidth. Within a channel group the data flows of different channels are scheduled independently of channels outside the group. Thus, the priority of a channel within a group can be changed without interfering with the data flows in channels outside the group.

The cellular telephone network hides the normal mobility of a workstation. However, after a link-level disconnection the user may wish to reconnect the workstation to some other MCH. The baseline MDCS can easily be augmented so that when an "MCH-handoff" takes place, the new MCH can act as a relay between the previous MCH and the mobile node. This extension will be implemented later.

The Mowgli Data Channel Service is implemented as a pair of *Data Channel Agents* (DCA); see Figure 3. These agents multiplex several data channels over the wireless point-to-point link. They also take care of priority-based scheduling of data channels. The communication between the DCAs is defined in the Mowgli Data Channel Protocol (MDCP). The main design goals in the MDCP protocol include the minimization of the protocol overhead and the round-trips over the wireless link.

Opening a new data channel is intended to be a light-weight operation. The open operation takes as its parameters the attributes for the data channel. Typically only 5 bytes of information are transmitted over the wireless link. Opening a data channel is based on an optimistic approach: the DCA opening the channel informs the other end of the existence of the new channel but does not wait for an acknowledgment. The channel is immediately ready for use. Because the round-trip time over a cellular telephone link can be fairly long, the optimistic approach is appropriate. Furthermore, when the lower-layer connection already exists, it is very unlikely that the open operation will fail. However, by setting a flag for open operation the DCA can be explicitly requested to wait for acknowledgment before completing the open operation.

The overhead of the protocol headers and of the control packets in the MDCP protocol has been kept very low. A typical packet only contains 1–3 bytes of header information. The typical header only includes packet type, length, and channel number. The channel number can be left out of consecutive packets for the same channel. Even the packet length information can be reduced to one bit when maximum length packets are delivered. Thus, in a typical situation where a single application is actively transmitting or receiving data, only one byte of header information per MDCP packet is needed.

The MDCP protocol can operate in two different modes. In the default mode, the MDCP layer relies on the lower layer and the underlying hardware to provide an error-free data link. For instance, the Radio Link Protocol [ETS92] used in the GSM provides a strong error correction mechanism. Thus, acknowledgments are only used for flow control and for the synchronization of the data channels after an unexpected break-down of the wireless link. In the other mode, the lower layer is assumed to be more unreliable. This results in an increased amount of protocol information.

6. Mowgli Data Transfer Service

The Mowgli Data Transfer Service (MDTS) offers efficient delivery of DTOs over the wireless link. The management of the DTOs in different states (*deferred*, *ready*, *in_transit*, or *suspended*) is the responsibility of the MDTS. The MDTS supports spooling and queueing of DTOs for transfer in a prioritized order. Transfers can also be deferred until specified conditions are satisfied. These features allow the MDTS to efficiently schedule transfers with varying urgency requirements.

6.1 General Operation

The typical life cycle of a DTO is rather straightforward. The contents of a DTO is initially produced by some application, for example a mail user interface. When the application has completed its part, for example a mail message is ready to be sent, the agent intercepts the product and creates the corresponding DTO. The condition attributes are evaluated and the DTO is given to the *Data Transfer Manager* (DTM). Depending on the condition attributes, the DTO is now either in the *ready* state or in the *deferred* state. The values of the attributes can originate from the user or application profile, from the group to which the DTO belongs, or they can depend on the system state. Moreover, the user can specify them explicitly. If the application has been designed for use in a mobile environment, then the DTO can be created directly by the application; the application can also determine — at the operation invocation — values of attributes.

The DTM provides high-level scheduling for DTOs in the *ready* state. It selects a DTO, allocates a data channel, and initiates the transfer of the DTO to the receiving DTM over the wireless link. The MDCS is employed to establish the lower-level data link connection when needed.

State changes between the *ready* and *deferred* states are trivial. They are controlled by the system state and by the user. As a matter of fact, this aspect of functionality in the MDTS has similarities with the concept of guarded statements [And91]: a change in a condition may cause or prevent the transfer operation of a certain DTO.

After the DTM has started the transfer of a DTO, the DTO is in the *in_transit* state. This means that both the mobile node and the MCH know about the transfer, and both of them know the local state of the ongoing transfer. If the transfer is interrupted, the local state information is preserved. How long the information will be preserved depends on a DTO-specific timer attribute — or on explicit user interventions. The user can also cause the interruption of a transfer in an indirect way. For example, if the user changes the DTO's "connection type" attribute from "weakly connected" to "strongly connected," the transfer of the DTO will be interrupted, and the DTO will stay in the *suspended* state until the mobile node is attached to a fast data communication network.

The DTOs which are in the *ready* state are scheduled for transfer in order of priority. If an DTO of higher priority enters the *ready* state while a lower priority DTO is being transferred, the transfer of the new DTO starts immediately on a new MDCS channel. When several DTOs are simultaneously in the *in_transit* state, the MDCS takes care of the allocation of bandwidth for the corresponding channels.

The information flow is controlled roughly in the same way. However, the only possible states are *in_transit* and *suspended*. A flow becomes active as soon as the application creates it. From the application's point of view, an output flow remains active even in the *suspended* state, the main difference being that now the DTM redirects the flow into a local intermediate storage. The local state information of a suspended flow is preserved both in the mobile node and in the MCH. However, this default behavior can be changed with help of the attribute system.

6.2 Attributes of Data Transfer Objects

In Section 2 we introduced the most interesting condition attributes. Here we introduce some other attributes which are more technical and are related to the functionality of the MDTS.

Reliability. This attribute specifies how the system should treat the DTO in the case of an abnormal termination of the transmission. If the delivery is specified to be "reliable," the state of the transmission is retained on both nodes; the DTO remains in the *suspended* state. When the transfer is resumed, the transmission continues from the point of interruption. If the delivery is not specified to be "reliable," the transfer state can be discarded. The reliable transfer can be requested not only to deal with unexpected disconnections, but also to recover from a mobile node crash.

Persistence. A DTO that requires reliable delivery can be declared persistent. This means that if the data link connection is lost, the MDCS will actively try to re-establish it for the given period of time.

Operation. This attribute contains information about DTO-specific operations to be invoked in the sending node before the transmission starts and about the corresponding operations at the receiving end. For example, the user can specify that the DTO is to be compressed before the transmission and uncompressed at the other end. The user can specify an agent that is to be invoked to transfer the DTO. As soon as the DTO is scheduled for transfer, the DTM allocates a channel, starts the specified agent, and gives the DTO to it. The agent then proceeds with the transfer using the services of the MDCS. This powerful feature can be used to invoke operations for DTOs which need a more complex treatment.

Notification. The user can ask for a notification of the completion status of the transfer. The notification is first destined to the communication control application which, in turn, notifies the end user in a prespecified manner: for example by showing a pop-up window.

6.3 Queue Management

The DTOs which have not yet been sent can be in different states and can belong to different groups. Technically, the Data Transfer Manager manages these DTOs as transfer queues where the DTOs wait to be processed. To an end user these as yet unsent DTOs are visualized as queues. Each queue corresponds to the DTOs of one group in a certain state; for example, ready mail messages and deferred mail messages.

The DTM provides operations for adding DTOs to a queue, deleting DTOs from a queue, reordering the DTOs in a queue and moving DTOs from one queue to another. In general, the operations can affect either a single DTO or a group of DTOs. For instance, the user can put all the DTOs of one group into the *deferred* state using a control operation with the group as the denoted operand. In a similar fashion, the user can change the priority of all DTOs in a group. If an operation is applied to an active transfer that is a DTO in the *in_transit* state, the transfer may be aborted, or suspended and later resumed. The outcome depends on the attributes of the DTO; most often it depends on the user's choice.

7. Summary

We have presented a communication architecture which builds a bridge between the worlds of wireless and wireline communication. Our architecture retains the existing TCP/IP-based communication architecture unmodified at hosts in the fixed network. The existing network applications on fixed hosts and mobile nodes need not to be modified.

We have paid particular attention to the higher level functionality required to adjust network applications to the new challenges of mobile environments. Our architecture reduces the traffic over the wireless link and improves the quality of service at the application level, especially fault tolerance and performance. The bottom line in our architecture is to split communication paths with end-to-end control into two parts using a store-and-forward interceptor.

Our architecture provides an application programming interface that is downwards compatible with the traditional BSD socket interface. We have specified a number of services to support the integration of existing network applications into the mobile environment. The services are designed for implementing a higher level of functionality that allows the applications to cooperate efficiently with existing services on the fixed network. The higher level of functionality takes the form of an agent-proxy pair that intercepts the intervening traffic over the wireless link. In this way we can significantly improve the performance from the user point of view. For instance, our experience has shown that applying this approach to WWW browsing can halve the response times seen by the user. In addition, we provide support for the development of new mobile applications based on the new client-mediator-server and peer-to-peer-thru-mediator paradigms. When new mobile applications that cooperate with an application specific proxy in the MCH are implemented, the new operations introduced in the Mowgli Socket Interface are useful and valuable.

The basic functionality of the architecture was relatively straightforward to implement in a Unix (Linux) environment. We are currently working on the implementation of enhanced communication services and on the further development of the communication control interface. In addition, we have also implemented the mobile node software for a Microsoft Windows (3.11) platform. The Windows version has been implemented as a dynamic-link library (DLL).

Future telecommunication systems promise remarkable improvements in performance and quality. For example, in the Universal Mobile Telecommunication System transmission speeds of 2 Mbits/sec are expected. However, even if these improvements alleviate the most unpleasant shortcomings of mobile computing, they do not solve the fundamental problems. The heterogeneity of the network performance will not disappear. In fact, the difference in performance between the wireless and wireline worlds will probably grow in the future. The wireline technology has essentially wider prospects. It will be a long time before the new fast wireless networks cover the scarcely populated areas. By the time they do, their capacity in the densely populated areas may be severely overstrained. Nor will the new technology be able to remove the possibility of unexpected disconnections. We expect that the use of mediators will remain a relevant approach in the integration of future wireline and wireless networks.

Acknowledgements

This work was carried out as a part of the Mowgli research project supervised by Professor Martti Tienari and funded by Digital Equipment Corporation, Nokia Telecommunications, Nokia Mobile Phones, Telecom Finland, and the Finnish Ministry of Education.

The authors express their thanks to the rest of the Mowgli team, particularly to Jani Kiiskinen and Heimo Laamanen, for the fruitful comments and discussions during the preparation of this paper.

References

[ABSK95] E. Amir, H. Balakrishnan, S. Seshan, and R. Katz. Efficient TCP over Networks with Wireless Links. In *Proc. of the Fifth Workshop on Hot Topics in Operating Systems (HotOS-V)*, Rosario Resort, Orcas Island, Washington, USA, May 1995.

[ACM94] Commun. ACM. Special issue on intelligent agents. Vol. 37, 7, July 1994.

[AKKP95] A. Albayrak, V. Karhu, J. Kiiskinen, and M. Perttilä. Implementation of Mowgli Communication Services for a Mobile Linux Workstation. SE Project Documentation, Univ. of Helsinki, Dept. of Computer Science, Helsinki, Finland, February 1995. In Finnish.

[AKL+94] T. Alanko, M. Kojo, H. Laamanen, M. Liljeberg, M. Moilanen, and K. Raatikainen. Measured Performance of Data Transmission over Cellular Telephone Networks. *Computer Communications Review*, 24(5):24–44, October 1994.

[And91] G. R. Andrews. *Concurrent Programming: Principles and Practice.* The Benjamin/ Cummings Publishing Company, Inc., first edition, 1991.

[BB95] A. Bakre and B.R. Badrinath. I-TCP: Indirect TCP for Mobile Hosts. In *Proc. IEEE 15th International Conference on Distributed Computer Systems*, Vancouver, Canada, May 1995.

[BBIM93] B. R. Badrinath, A. Bakre, T. Imielinski, and R. Marantz. Handling Mobile Clients: A Case for Indirect Interaction. In *Proc. of the Fourth Workshop on Workstastion Operating Systems (WWOS-IV).* IEEE, October 1993.

[CI94] R. Cáceres and L. Iftode. The Effects of Mobility on Reliable Transport Protocols. In *Proc. IEEE 14th International Conference on Distributed Computer Systems*, pages 12–20, Poznan, Poland, June 1994. IEEE, IEEE Computer Society Press.

[Com95] IEEE Personal Communications. Special Issue on the European Path Towards UMTS. Vol. 2, 1, February 1995.

[ETS92] ETSI. Radio link protocol for data and telematic services on the ms-bss interface. GSM specification 04.22, ETSI/TC GSM, 1992.

[KRA94] M. Kojo, K. Raatikainen, and T. Alanko. Connecting Mobile Workstations to the Internet over a Digital Cellular Telephone Network. Report C-1994-39, Univ. of Helsinki, Dept. of Computer Science, Helsinki, Finland, September 1994. Published also in Proc. Workshop on Mobile and Wireless Information Systems (MOBIDATA), November 1994.

[LAK+95] M. Liljeberg, T. Alanko, M. Kojo, H. Laamanen, and K. Raatikainen. Optimizing World-Wide Web for Weakly-Connected Mobile Workstations: An Indirect Approach. In *Proc. 2nd International Workshop on Services in Distributed and Networked Environments (SDNE)*, Whistler, Canada, June 1995.

[LMKQ89] S. J. Leffler, M. K. McKusick, M. J. Karels, and J. S. Quarterman. *The Design and Implementation of the 4.3BSD UNIX Operating System*. Addison Wesley, first edition, 1989.

[Per95] C. Perkins (ed.). IP Mobility Support. Internet Draft, IETF, July 1995.

[Rah93] M. Rahnema. Overview of the GSM System and Protocol Architecture. *IEEE Communication Magazine*, 31(4):92–100, April 1993.

[Wat95] T. Watson. Effective Wireless Communication Through Application Partitioning. In *Proc. Fifth Workshop on Hot topics in Operating Systems (HotOS-V)*, Rosario Resort, Orcas Island, WA, May 1995. IEEE.

[YB94] R. Yavatkar and N. Bhagawat. Improving End-to-End Performance of TCP over Mobile Internetworks. In *Proc. IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, California, December 1994.

[ZD95] B. Zenel and D. Duchamp. Intelligent Communication Filtering for Limited Bandwidth Environments. In *Proc. Fifth Workshop on Hot topics in Operating Systems (HotOS-V)*, Rosario Resort, Orcas Island, WA, May 1995. IEEE.