University of Helsinki
Department of Computer Science
Series of Publications C, No. C-1996-1

Data Channel Service for Wireless Telephone Links

Jani Kiiskinen, Markku Kojo, Mika Liljeberg, and Kimmo Raatikainen

Helsinki, January 1996

Report C-1996-1

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

Data Channel Service for Wireless Telephone Links

Jani Kiiskinen, Markku Kojo, Mika Liljeberg, and Kimmo Raatikainen

Department of Computer Science, University of Helsinki
Report C-1996-1
January 1996
17 pages

Abstract. The characteristics of wireless telephone links are quite different from those of wireline links. In particular, standard TCP/IP performs poorly in an environment where these links are combined. We have designed and implemented a communication architecture for the efficient operation over an internetwork consisting of fixed networks and wireless telephone networks. In our architecture the Mobile-Connection Host is the bridge between the wired and wireless networks. It allows us to transparently replace TCP/IP over the wireless medium with a special transport service called the Mowgli Data Channel Service (MDCS). The MDCS is designed to take into account long latencies, narrow bandwidth, highly variable transmission delays, and sudden disconnections. As a result, the MDCS significantly improves the fault tolerance and performance of communication over wireless telephone links. In this paper we describe the MDCS and its implementation as the Mowgli Data Channel Protocol (MDCP). We also give performance results and compare the performance of our approach to that of standard TCP/IP.

Key Words: mobility, wireless connections, cellular telephone

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

Contents

1. Introduction

By utilizing the data services of a cellular telephone system, for example the digital GSM system [Rah93], a nomadic user can establish access to a fixed computer network. 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 different from those of wireline links. They have narrow bandwidth, long latencies, and highly variable transmission delays. Furthermore, sudden disconnections are not uncommon. These characteristics create problems for many existing applications designed to operate with fast and reliable connections. The problems with TCP/IP-based communication over wireless networks have been discussed recently in [AKL+94] and [CI94].

The non-transparent mode of the GSM Data Service provides a reliable data link with a nominal bandwidth of 9600 bps; in the ETSI specification the bit error rate is required to be less than 10^{-8} [ETS90]. The high reliability is obtained by using the *Radio Link Protocol* (RLP) [ETS92] but the price is highly variable transmission delays. Our performance measurements reported in [AKL+94] indicate that in favorable conditions the round-trip time is about one second and the maximum sustainable transfer rate for user level data is about 900 bytes per second (7200 bps). In unfavorable conditions we have observed round-trip times longer than 10 seconds.

In [KRA94] we introduced the Mowgli[1] communication architecture designed to alleviate the problems due to cellular telephone networks. Further enhancements to the basic architecture are described in [KALR95]. The key idea in the Mowgli communication architecture is to replace the client-server paradigm with a client-mediator-server paradigm and the peer-to-peer paradigm with a peer-to-peer-thru-mediator paradigm.

The mediator splits the end-to-end connection into two separate parts: one over the fixed network, the other over the wireless link. This enables each part of the connection to be treated in a way that is well suited to the underlying communication media. Similar approaches have successfully been used to improve the performance of TCP-based communication in networks where the last link to a mobile workstation is implemented over wireless LAN (for example [BB95, BSAK95, YB94]). These approaches focus on improving operation at the transport layer. However, our approach is more general since it takes into account also the layers above the transport layer. This is important with slow links since even greater improvements can be achieved by optimizing the operation at the application level. Our mediator supports intelligent agents which are an evolving approach to many problems in computing (see e.g. [ACM94]). Furthermore, in order to enable efficient data transmission over the wireless telephone link we have taken into account the characteristics of this low-performing link with a special care.

The *Mowgli Data Channel Service* (MDCS) is an essential part of the Mowgli communication architecture. It is a special transport service that transparently replaces standard

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

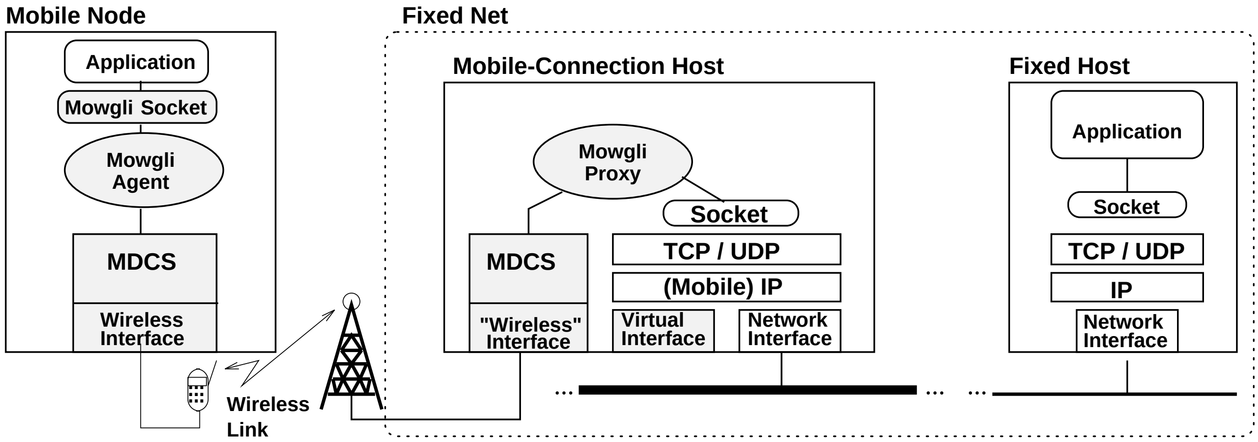

Figure 1: The Mowgli Communication Architecture for Connecting Mobile Nodes to Fixed Networks

TCP/IP protocols over the wireless link. The MDCS is designed to cope with the specific features of the cellular telephone links. The MDCS provides multiplexing and priority based scheduling of data channels over the wireless link, which is vital for the efficient use of the limited bandwidth. In this paper we show how the MDCS, implemented on top of the Linux operating system, improves the fault-tolerance and performance of communication over wireless cellular telephone links.

The rest of paper is organized as follows. In Section 2 we outline the essential features of the Mowgli communication architecture. In Section 3 we describe the Mowgli Data Channel Services. In Section 4 we present the essential features of the Mowgli Data Channel Protocol. We also discuss the specific problems of controlling the wireless data link. The results of performance measurements are reported in Section 5. In Section 6 we conclude with some remarks and discuss future research.

2. Overview of the Mowgli Communication Architecture

In the Mowgli communication architecture a mobile node is connected to the fixed network through a wireless telephone link. The *Mobile-Connection Host* (MCH) is a host in the fixed net that provides the mobile node with the connection point to the wireline Internet (see Figure 1). A proxy on the MCH in cooperation with an agent on the mobile node acts as a mediator for all data delivered between an application on the mobile node and its peer.

The MCH dynamically allocates a virtual network interface for each mobile node connected to it. The MCH assigns the IP address of the mobile node to this interface. Thus, the MCH is indistinguishable from the mobile node when communicating on its behalf with other hosts. If the mobile node connects to an MCH not in its home network, the Mobile IP routing [Per95] is exploited. In our architecture, there are two primary ways to implement

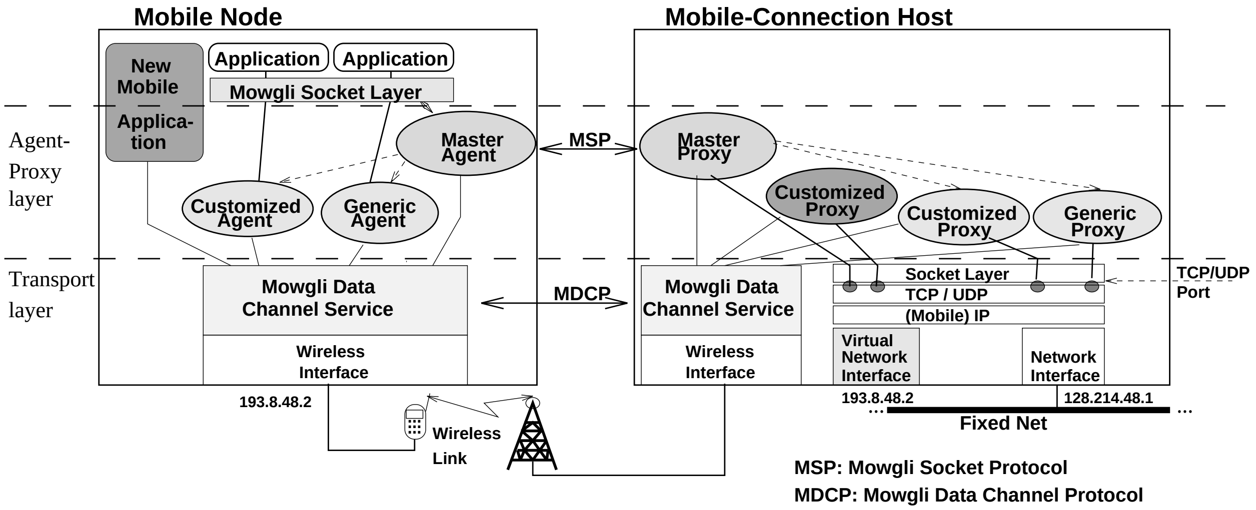

Figure 2: Software components in the Mowgli Sockets implementation

the Mobile IP routing. The MCH can either include the complete functionality of the Foreign Agent or only incorporate those functions needed to communicate with a separate Foreign Agent in the local network.

Existing applications on the mobile node can access the TCP/IP communication services through an application programming interface called the Mowgli socket interface. The basic communication services offered through Mowgli sockets include the socket operations in the traditional BSD socket interface [LMKQ89]. Thus, old applications can be executed without modifications.

The basic software components used in our Unix[2] (Linux) implementation are shown in Figure 2. The implementation of the architecture is divided into three logical layers: the Mowgli socket layer, the Agent-Proxy layer, and the Transport layer. In order to support old applications, the socket layer in the Linux kernel has been modified to notify the *master agent* whenever an application on the mobile node invokes a socket operation. The master agent then sends a message to the *master proxy* which invokes the corresponding operation on behalf of the application. The *Mowgli Socket Protocol* (MSP) defines the necessary messages.

For each stream socket a *generic* agent and proxy is created once a TCP connection with a remote peer is established. The agent-proxy pair then takes care of delivering the data over the wireless link. If desired, the generic agent and proxy can be configured to be replaced by *customized* versions specially designed for the application protocol in use. Normally the master agent and proxy take care of delivering UDP datagrams. Customized agents and proxies can also be created for datagram sockets. The agents and proxies use the transport services provided by the *Mowgli Data Channel Service* (MDCS).

[2]Unix is a registered trademark in the U.S. and other countries licensed exclusively through X/Open Company, Ltd.

Customized agents and proxies are useful for adding application specific functionality. For example, our agent-proxy pair for optimizing World-Wide Web browsing has given promising results (see [LAK+95]). Similarly, new mobile node applications can be implemented in accordance with the client-mediator-server paradigm: the traditional client application is divided into a cooperating mobile node client and a special (customized) proxy on the MCH. In this approach the proxy takes care of all communication with the server, thus significantly reducing the traffic over the low-performing wireless link. On the mobile node the functionality of the customized agent is integrated into the mobile application software (see Figure 2).

3. Mowgli Data Channel Service

In the Mowgli communication architecture, the Mowgli Data Channel Service is used over the wireless network instead of the standard TCP/IP protocol stack. The main design objectives of the MDCS are to improve the fault-tolerance and performance of data communication over wireless cellular telephone links.

3.1 MDCS Overview

The Mowgli Data Channel Service provides a flexible set of communication services to MDCS-clients: agents and proxies as well as future mobile node applications designed for the wireless environment. As a basic service, the MDCS offers bidirectional data channels for data transfer over the wireless link. Data channels can be opened as *stream channels* providing TCP-like functionality or as *message channels* if UDP-like datagram service is desired. However, the message channels differ from UDP in that the delivery of the messages is reliable between the mobile node and the MCH.

The MDCS supports up to 256 simultaneous data channels. Each channel has independent flow control. Data channels are multiplexed according to channel priorities. Priority-based scheduling allows the MDCS to provide reasonable response times to interactive applications even if there are bandwidth-intensive background transfers, for example file transfers, using up the link capacity. Conventional TCP/IP over PPP [Sim94] or SLIP [Rom88] performs very poorly in this respect.

Another convenient feature of the MDCS is improved fault-tolerance. Particularly, the MDCS is designed so that it recovers efficiently from unexpected temporary disconnections which are quite common in cellular environments. The MDCS maintains the state information of the ongoing transmission over each channel so that an interrupted transmission can later be resumed. Each data channel has its own attributes that control the behavior of the channel.

The programming interface offered to MDCS-clients is very similar to the traditional BSD socket interface. Thus, implementing communication over the wireless link in agents and proxies as well as in new mobile node applications can be based on concepts that are familiar from traditional networking software.

3.2 Data Channel Attributes

Each data channel has a number of attributes that can be used to fine-tune the channel behavior to meet the needs of an MDCS-client. The MDCS also provides attributes for automatically controlling the wireless link. The most important attributes are those controlling the exception handling, priority, and establishment of the wireless link on-demand.

The attribute values are initialized when a data channel is opened. However, a specific operation for changing these attribute values is offered to the MDCS-clients. Below we briefly describe the most important attributes.

Priority. Each data channel has a priority which can be assigned separately for each direction. The priorities are used in multiplexing the data channels.

Link Connection Permission. This channel specific attribute is used to inform the MDCS that the wireless link may be established whenever a particular channel wants to transfer data. The value *auto* instructs the MDCS to automatically establish link connection when the channel is opened. If the value is *ask* the MDCS will prompt the end-user to decide whether or not the link connection should be established. If the value is *use* the data channel can only be opened if the link connection is already established.

Link Reconnection Permission. This attribute specifies whether the channel has the right to re-establish the link connection after an unexpected disconnection. Possible values are *auto*, *when-data*, and *none*. If the value is *auto* for any open channel, the MDCS immediately tries to re-establish the link after a disconnection. If the value is *when-data*, the link connection is automatically re-established when there is outgoing data in the channel. If the value is *none*, this channel will not cause the MDCS to re-establish the link connection.

State Expiration Timeout. This attribute specifies how long the MDCS should retain the state information of the data channel after a disconnection. If the timeout expires, the data channel will be closed and the state information and all the buffered data will be discarded.

Idletime. This attribute determines how long the data channel has to remain idle before the MDCS may disconnect the wireless link. This does not close the data channel; the wireless link can be re-established when there is further data to be transferred.

In addition to the attributes described above, there are attributes for restricting the amount of transmitted data and the duration of the connection. Further on, there is a selection of read-only attributes containing usage statistics and state information.

3.3 Mowgli Data Channel Service API

The programming interface to the data channels includes the same operations as the BSD socket interface. Hence, MDCS-clients can be programmed to open and close stream channel connections as well as to receive data from and send data to a channel in a similar fashion as clients and servers using Internet domain sockets. In addition, an MDCS-client can also set and get the attributes of channels and get usage statistics.

When an MDCS-client creates a new data channel socket, it specifies the Mowgli Data Channel Service as the communications domain within which communication will take place. A stream channel is opened when the MDCS-client invokes the `connect` operation destined to a port which is listened by its peer on the other side of the wireless link. Before calling the `connect` operation, the MDCS-client should specify the opening method and the initial attributes for the new channel by using the `setsockattrs` operation. If this is not done, the default values of attributes are used. The opening method is either optimistic or pessimistic. In the former case, the `connect` operation does not wait for an acknowledgement from the other end as it does in the latter case.

The normal `close` and `shutdown` operations are provided to close a socket. The MDCS-client can also reset either or both directions of the data channel by invoking the `reset` operation. The `reset` operation will discard any buffered data immediately, instead of sending it over the link.

The operations for sending and receiving data have the same functionality as the corresponding operations in the BSD socket interface. However, there are differences in the exception handling. If the wireless link is down when the MDCS-client writes the data, the link reconnection permission is first checked. If the data channel has not been given the permission to re-establish the link, the data will be buffered. The MDCS-client can request to be notified of changes in link state either with a signal or a message.

In many situations it is necessary to modify the attribute values of an active channel. For example, the agent may want to increase the priority of the data channel because the channel contains urgent data to the proxy. The MDCS-client can modify attribute values by invoking the `setsockopt` operation or by invoking the `setsockattrs` which enables setting of several attributes in one operation.

Because of the unpredictable conditions in the wireless environment, the end-user will sometimes need direct control over the wireless link connection. For example, the user may want to stop all data transmission over the wireless link if the field strength is too low. Therefore, the MDCS provides wireless link services including operations for opening and closing the link connection.

In many situations the MDCS-clients can make use of various statistics of the wireless link. The client can invoke the `get_link_statistics` operation which returns the requested information. The statistics maintained by the MDCS include the state of the link and the observed throughput in both directions.

4. Mowgli Data Channel Protocol

The communication over the wireless medium is defined in the *Mowgli Data Channel Protocol* (MDCP). The main design objectives in the MDCP protocol were high performance, ability to recover from unexpected disconnections and tolerance for long and highly variable delays.

High performance has been achieved by minimizing the amount of protocol overhead, specifically protocol headers and round-trips over the wireless link. In addition, the MDCP protocol always tries to transmit data at the best possible speed the wireless link is capable of and uses acknowledgements sparingly. These solutions are different from standard TCP/IP, which has fairly large protocol headers, is vulnerable to high variance in round-trip times. In addition, TCP/IP employs the *slow-start* congestion control algorithm [Jac88] at the start of each new connection as well as after each time the retransmission timer expires. While being proper behavior within a fixed network, the use of slow-start results in poor performance in the wireless environment [CI94].

4.1 MDCP Overview

The MDCP is a light-weight packet-based protocol. There are two kinds of packets in the MDCP protocol: control packets and data packets. Control packets convey channel creation and deletion requests, as well as resynchronization, scheduling, and flow-control information. Data packets convey data transmitted on individual channels.

The protocol headers in data packets fit in 1–3 bytes. Control packets, being much less common than data packets, occupy 2–16 bytes, 4 bytes on the average. The high optimization level of the MDCP protocol is reflected in the large variety of packet types. Most of the packet types correspond to some common situations where not all of the usual header information needs to be transmitted. For instance, the data channel identifier is not needed when a packet is addressed to the same channel as the previous one. Furthermore, an optimized packet format is defined for small packets carrying less than 16 bytes of data. We decided not to implement generic data compression in the MDCP protocol. One reason is that data compression is currently being standardized for the GSM Data Service. On the other hand, application specific data compression is usually much more effective and can easily be done by customized agents and proxies.

The MDCP protocol can operate in two different modes. In the default mode, the MDCP relies on lower protocol layers and on the underlying hardware to provide an error-free data link. For instance, the Radio Link Protocol used in the GSM provides a strong error correction mechanism. Therefore, acknowledgements are only needed for flow control and during the recovery phase after an unexpected disconnection. In the error-monitoring mode, the lower layer is not assumed to be completely error-free, although errors are assumed to be rare. In this case, the protocol overhead is slightly higher because an additional checksum is needed for error detection. Recovery from an error is handled in the same manner as recovery from wireless link disconnections. The rationale for the error-monitoring mode is that while, for instance, the RLP protocol provides relatively reliable transmission over the GSM network, occasional bit errors can occur also in other components between the mobile node and the MCH.

4.2 Channel Scheduling Policy

The MDCS schedules data channels based on channel priorities. The current implementation supports several priority classes. The highest priority class is reserved for transmitting control packets. Data channels in different priority classes are scheduled pre-emptively. Whenever several channels want to transmit packets simultaneously, the channel having the highest priority class is always selected. This means that urgent data always gets link capacity in a timely manner. Data channels within the same priority class are scheduled using the round-robin algorithm. This provides a degree of fairness for channels within the same priority class.

The current scheduling scheme is quite functional and easily configured for different types of applications. In the case of old TCP/IP applications the external configuration of priorities is important since the applications cannot change the priorities. However, we recognize that static priorities do not work well if the traffic patterns change over time depending on user activities. Also, there is the danger of misconfiguration so that some applications starve for link capacity. We are currently working on a more sophisticated dynamic scheduling algorithm similar to the process scheduling algorithms in multitasking operating systems. The basic idea is that bursty interactive traffic will cause the channel priority to temporarily increase while constant bulk transfer will cause the priority to remain the same or even decrease.

4.3 Data Channel Flow Control

Data channel flow control in the MDCP is based on the sliding window mechanism. A "lazy-ack" policy is used for sending acknowledgements, which means that acknowledgements are not sent for every received packet. Instead, acknowledgements are delayed until several packets can be acknowledged with a single ack packet.

As in TCP, a credit based flow-control algorithm is used. Initially, a sending end is given as much credit as there is room in the remote end's receive window. Whenever data is sent along the data channel, the credit is decreased by the amount sent. When the data channel runs out of credit, its window closes and the sender stops sending data in that channel. The window is reopened when an acknowledgement is received from the remote end. The acknowledgement packet indicates how much more credit the data channel is given.

The MDCP protocol differs from TCP in the way the acknowledgement policy is implemented. The receiving end keeps track of how much unused credit the remote end has for a specific data channel. In addition, the receiving end keeps track of how much "new quota" the sending end has, i.e. how much more credit it can be given. Initially, the sending end has all the credit, but no new quota. When an MDCP data packet arrives, the receiving end decreases its view of the remote end's remaining credit. On the other hand, when a received packet is delivered to the layer above, the receiving end increases the new quota of the remote end. The receiving end always sends out a flow control acknowledgement when both of the following conditions become true: the remote end's credit is below a specified *minimum credit* threshold and the remote end has enough new quota, i.e. more than a specified *minimum new quota* threshold. The acknowledgement gives all the new quota over to the remote end.

The selection of the threshold values is crucial for the flow control to function smoothly. If the minimum credit threshold is too low, the sender runs the risk of running out of credit and stopping unnecessarily. On the other hand, if the threshold is too high acknowledgements will be sent too often, which wastes the wireless link capacity. The minimum new quota threshold ensures that acknowledgements are not sent every time a small amount of data is taken from a full receive window and delivered to the layer above.

4.4 Wireless Link Flow Control

The wireless link flow control is used to optimize the data transmission over the wireless link. The flow control is needed because quite a lot of data can be buffered along the data communication path between the mobile node and the MCH. The problem is rather obvious when sending data to the mobile node, because the communication link between the MCH and the mobile switching center is faster than the link between the switching center and the mobile node. Thus, if the MCH is sending data at full speed, the mobile node is unable to match the pace and data will begin accumulating in buffers. The effect is that the observed round-trip time between the mobile node and the MCH will increase rapidly to tens of seconds if the amount of buffering is not controlled in some way. Obviously unrestrained buffering would make impossible any attempt to multiplex and schedule data channels intelligently, let alone deliver control packets and data packets with high priority in a timely manner.

In order to solve the problem of excessive buffering, we are currently adding a separate flow control mechanism to the wireless link. The algorithm is a simplification of the credit based data channel flow control we described earlier. Since the objective is to limit the amount of buffering, we need to select as small a window size as possible without affecting throughput. An alternative is to integrate rate based flow control into the feedback mechanism with acknowledgements.

4.5 Recovery after an Unexpected Disconnection

An unexpected disconnection occurs when conditions on the radio link change so that the wireless connection can no longer be maintained. In the case of cellular telephones this means that the cellular telephone network decides to hang up the telephone call. In many cases the poor conditions are temporary and the wireless link can be re-established in a short time;

for example, after driving through a tunnel. Sometimes, however, the connection cannot be immediately re-established due to insufficient coverage. The MDCS deals gracefully with both of these alternatives.

MDCS-clients can request the MDCS to give a notification with a signal or message when the wireless link is disconnected. This allows them to adapt to the changed connectivity, in many cases enabling the user to continue working in disconnected mode. However, old TCP/IP applications are completely isolated from the knowledge of unexpected disconnections. The MDCS may automatically try to re-establish the broken wireless connection. This activity is controlled by the data channel attribute Link Reconnection Permission (see section 3.2).

After a successful establishment of the wireless link connection, the MDCS enters a negotiation phase. During this phase, the user is authenticated (currently only normal Unix login authentication is supported) and a session identifier together with the other parameters for the session are negotiated (e.g. mobile node's IP-address). While negotiating the session identifier, the mobile node and the MCH detect whether they are in the process of establishing a completely new session or re-establishing an old one. If they are re-establishing an old session they will, after the negotiation, enter a recovery phase. The purpose of this phase is to enable all open channels to continue transmitting data from the point where they were when the wireless link got disconnected.

In the recovery phase each end sends one synchronization packet per data channel to the remote end. The synchronization packet is similar to an acknowledgement packet. When a synchronization packet is received all the packets acknowledged by the remote end are removed from the send buffer. Then both ends will resend all unacknowledged packets and begin normal operation. At this point MDCS-clients are informed that the wireless link has been re-established.

The data channel resynchronization mechanism is also employed when the MDCS has been configured to operate in error-monitoring mode. If a corrupted packet is detected, the both ends will first need to find a safe point in the data stream for continuing. This is done by exchanging special link synchronization packets. When both ends have found the safe point they enter the data channel recovery phase and resynchronize each data channel.

5. Performance Results

The performance of the MDCP protocol was tested in four sets of experiments totaling in 50 hours of "air time". Our experiments indicate that in a wireless environment the performance of our approach is superior to regular TCP/IP on top of PPP with header compression [Jac90] enabled.

Table 1: Hardware Used in the Measurements

Mobile Node Mobile-Connection Host LAN Fixed Host WAN Fixed Host	DECpc 325 SL, 386SX 25 MHz, 8 Mbytes of main memory AMBRA Sprinta 486, 486SX 25 MHz, 8 Mbytes of main memory PMC, Pentium 90 MHz, 16 Mbytes of main memory SPARCstation 2, Sun 4/75 40 MHz, 32 Mbytes of main memory
GSM Data Adapter	Nokia DTP-2 Cellular Data Card, (PCMCIA Type II Version 2)
MCH Modem	Nokia ECM Fast SW (DS61660), V.32bis V.42bis
GSM Phone	Nokia 2110

DECpc is a trademark of Digital Equipment Corporation. AMBRA Sprinta 486 is a trademark of Individual Computer Products International. Nokia is a registered trademark of Nokia Corporation. Pentium is a registered trademark of Intel Corporation. SPARCstation is a registered trademark of SPARC International, Inc., licensed exclusively to Sun Microsystems, Inc.

5.1 Test Arrangements

The data processing configuration in our experiments was similar to traditional client-server environments. The mobile node was a "notebook PC" running the Linux operating system. The Mobile-Connection Host (MCH) was a "desktop PC", running Linux, located in the local-area network at University of Helsinki, Finland. In the experiments we used two different fixed hosts. The first one (LAN fixed host) was also a Linux workstation, located in the same LAN with the MCH. The second one (WAN fixed host) was a Unix-workstation, running SunOS 4.1.2, located at University of Waterloo, Toronto, Canada. A detailed list of hardware used in the tests is given in Table 1.

The mobile node was connected to the MCH through GSM non-transparent Data Service[3] (bearer service with the RLP protocol) and PSTN with modem. The connection between Finland and Canada was over the Internet. The experiments were conducted in a normal office environment with good conditions on the GSM radio link and a lightly loaded local-area network. By executing the tests at different times of day we tried to reduce the effect of Internet traffic in the WAN tests.

In the experiments we used the `ttcp` measurement tool. The receiver was modified to log a timestamp for every completed `read` operation so that we could monitor the progress of transfer.

In the first set of experiments we measured throughput between the mobile node and fixed host. Both transfer directions were examined by transferring 100 kbytes in each experiment. In addition, both locations of the fixed host were used: LAN and WAN. Each experiment was repeated 20 times.

In the second set of experiments we measured throughput of three simultaneous transfers in one direction. Over each connection 100 kbytes was transferred. Again both directions and locations were examined. Each experiment was repeated 20 times.

[3]In Finland GSM Data Service is commercially available and provide by Telecom Finland.

Table 2: Summary of Transfer Times (sec) for Files of 100 kbytes

Single transfer (20 repetitions)

Net-work	Pro-tocol	From fixed host to mobile node					From mobile node to fixed host				
		Mean	Min	Med	Max	IQR	Mean	Min	Med	Max	IQR
LAN	MDCP	108.8	107.8	108.5	110.2	1.6	108.4	108.3	108.4	108.6	0.1
	TCP	139.2	116.6	143.1	168.2	36.3	126.3	114.4	114.7	160.5	23.4
WAN	MDCP	109.2	107.5	108.5	116.9	1.1	109.2	107.1	108.5	115.7	0.5
	TCP	171.9	117.6	154.6	252.7	89.2	231.2	121.4	152.6	497.1	197.4

Three parallel transfers (20 repetitions)

Net-work	Pro-tocol	From fixed host to mobile node					From mobile node to fixed host				
		Mean	Min	Med	Max	IQR	Mean	Min	Med	Max	IQR
LAN	MDCP	326.4	320.8	326.2	330.4	2.4	331.5	315.8	324.2	494.3	2.4
	TCP	513.8	413.3	495.4	591.4	87.3	344.8	299.7	325.9	490.9	75.7
WAN	MDCP	328.5	254.6	325.6	385.2	6.4	339.7	307.4	324.1	584.8	33.3
	TCP	362.0	251.0	364.1	480.0	45.8	388.9	253.0	358.9	759.9	94.4

In the third set of experiments we examined the situation in which we had six simultaneous transfers: three in each direction. These preliminary experiments were only carried out with the LAN fixed host.

In the fourth set of experiments we examined how on-going bulk data transfer affects bursty interactive exchange of data. In these experiments we used `ftp` to transfer a large file from a fixed host to the mobile node while the mobile node was simultaneously sending one byte messages to a TCP-echo server in the same fixed host as the ftp-sender. We measured the round-trip times of 40 echo messages.The time gap between the reply and the next request was 15 seconds. These experiments were only conducted with the LAN fixed host. In the MDCP experiments we configured a higher priority channel to be used for the echo messages.

5.2 Measured Results

The results of performance measurements for experiments 1 and 2 are summarized in Table 2. The table contains the means of transfer times together with the minima, medians, and maxima. We also give the interquartile range (IQR), which is the difference between the 75th and 25th percentile.

The mean is the most often reported index in performance studies. However, the mean is, particularly in the WAN tests, quite sensitive to exceptionally large observations. Therefore, we prefer to draw the conclusions from the medians. Usually the standard deviation

Table 3: Summary of Interactive Response Times (sec): 40 repetitions

Protocol	Mean	Min	Med	Max	IQR
MDCP	2.57	2.18	2.24	3.67	0.8
TCP	7.88	3.64	8.12	9.12	0.5

is reported as the index of variation. However, it is very sensitive to exceptionally large observations. Therefore, we use the IQR as the index of variation.

The Table 2 clearly shows that our approach with MDCP is superior to TCP. Typically, the transfer time with MDCP is 10–50 % shorter than with TCP. In particular, MDCP is able to provide stable transfer times as the IQRs indicate. The same can be seen in Figure 3, in which we present typical results from experiments 1–3. The figure shows how the transfers progress in different cases.

There are several reasons leading to the differences. 1) The MDCP protocol reaches the full bandwidth much faster than TCP. This happens because TCP uses the slow-start congestion control mechanism at the beginning of each new connection instead of immediately transmitting at full speed like MDCP does. 2) TCP does a number of unnecessary retransmissions before the retransmission timers adjust to the long latency of the GSM link. TCP also enters the slow-start phase every now and then during the transfer. 3) Splitting the end-to-end connection into two parts in our approach reduces unnecessary retransmissions and variation in round-trip delays. 4) The header compression in MDCP is more efficient than the Van Jacobson method for TCP/IP with PPP.

In all the parallel transfers we can observe that MDCP remains very stable since MDCP provides equal share of the wireless link for each parallel transfer. In TCP experiments the allocation of link capacity is more random-like. Especially, in the case of parallel transfers in both directions we can see how the TCP transfers interfere with each other leading into very poor performance (See Figure 3, cases 3A and 3B).

In the case of a single interactive connection both TCP and MDCP perform reasonably well; the response time is roughly one second. In the fourth set of experiments we studied how the response time changes when an intensive bulk data transfer is simultaneously performed with an interactive connection. The results are summarized in Table 3. We observed that with the MDCP protocol the interactive response times are stable: 2–3 seconds. In contrast, with TCP the response times are fluctuating between four and eight seconds; typically 7–8 seconds. This clearly shows the benefits of scheduling data channels in a prioritized manner. The fluctuations in the TCP case are caused by TCP retransmission timers triggering an occasional retransmission followed by TCP slow-start. During this phase only few TCP segments, if any, are transmitted over the bulk data transfer connection. In this way the interactive channel gets more link capacity for a short period.

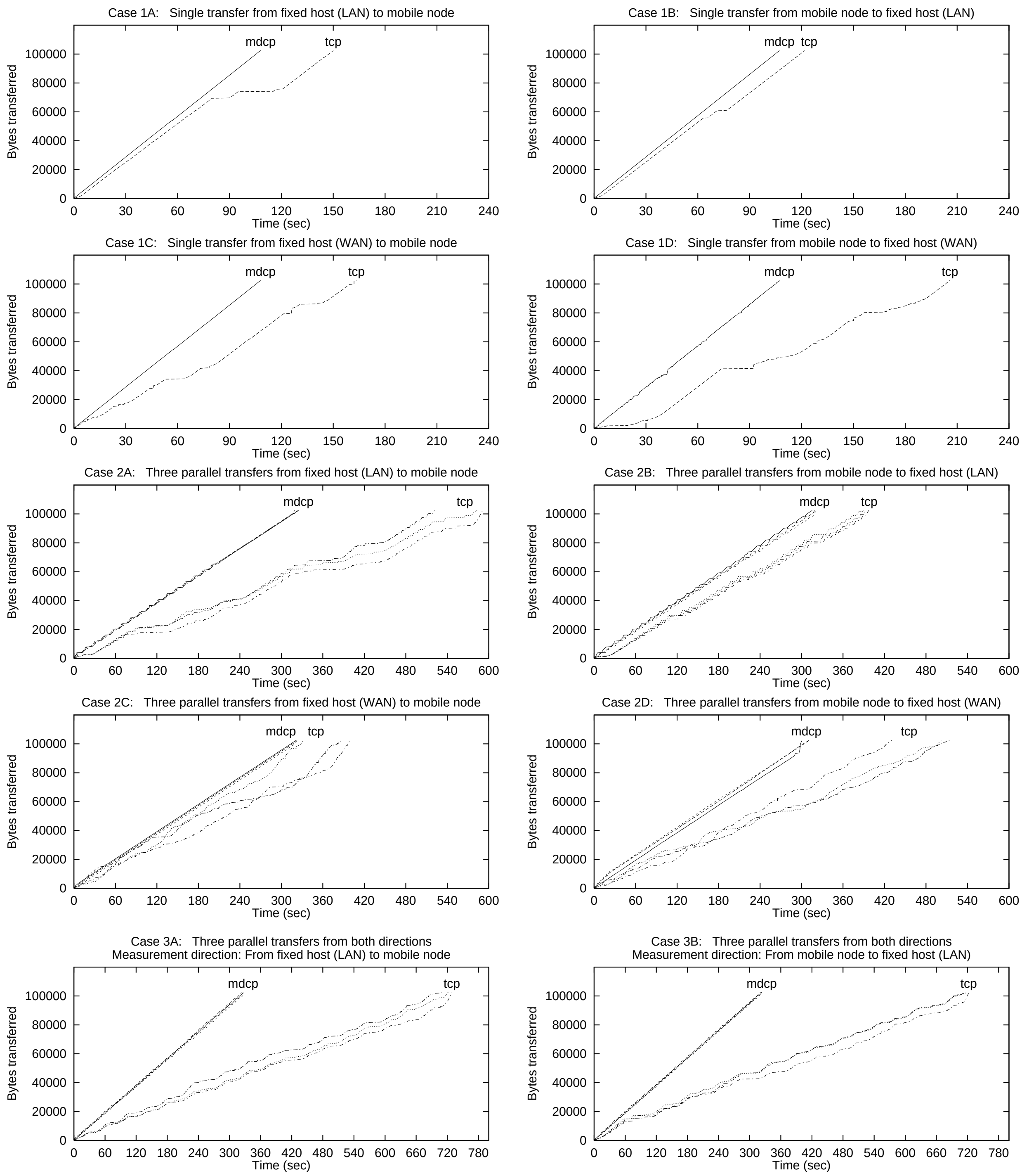

Figure 3: Examples of Typical Transfer Times for Files of 100 kbytes

Protocols: MDCP (max packet size 232) and TCP/IP with PPP (MSS 232)

6. Conclusions and Future Work

In the Mowgli-project we have developed a communication architecture that builds a bridge between the worlds of wireless and wireline communication. The new service interface called the *Mowgli Data Channel Service* (MDCS) and the *Mowgli Data Channel Protocol* (MDCP) have significantly improved the fault tolerance and performance of communication. The basic functionality was relatively straightforward to implement in the Unix-environment. In addition to the Linux platform, we have a prototype implementation of the mobile node software running on Microsoft Windows[4] 3.1 platform.

The experiments conducted indicate that even in favorable conditions (i.e. no exceptional transmission delays and unexpected disconnections) the Mowgli architecture significantly improves TCP/IP based communication over cellular telephone networks. It should be noted that in poor conditions the MDCS is even more clearly superior, since it can automatically recover from disconnections of the wireless link. When TCP/IP is used, the wireless link must usually be manually re-established and any interrupted user activities must be restarted from the very beginning.

In the near future we are going to measure the performance of the MDCP protocol more extensively in a laboratory environment where we can simulate different radio conditions over the wireless link. Meanwhile, we are working on finding good default values of various protocol parameters and channel attributes. We are also refining the algorithms for priority based scheduling as well as error recovery and flow control over the wireless link.

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 Mobile Phones, Nokia Telecommunications, Telecom Finland, and the Finnish Ministry of Education.

The authors are thankful to the rest of the Mowgli team, particularly to Timo Alanko, Petteri Kaskenpalo, and Heimo Laamanen, for the fruitful comments and discussions during the preparation of this paper. The authors also want to thank the Mowet94 student group from the fall 1994 Software Engineering Course, who implemented the initial baseline version of the Mowgli Data Channel Service as well as some other essential components.

References

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

[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.

[4]Windows is a trademark of Microsoft Corp.

[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.

[BSAK95] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz. Improving TCP/IP Performance over Wireless Networks. In *Proc. of the First ACM International Conf. on Mobile Computing and Networking (MobiCom '95)*, Berkeley, California, USA, November 1995.

[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.

[ETS90] ETSI. Quality of Service. GSM specification 02.08, version 3.0.0, ETSI/TC GSM, March 1990.

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

[Jac88] V. Jacobson. Congestion Avoidance and Control. In *Proc. ACM SIGCOMM '88 Symposium on Communications Architectures and Protocols*, pages 314–329, Stanford, Calif., August 1988.

[Jac90] V. Jacobson. Compressing TCP/IP Headers for Low-Speed Serial Links. Request for Comments 1144, Network Information Center, February 1990.

[KALR95] M. Kojo, T. Alanko, M. Liljeberg, and K. Raatikainen. Enhanced Communication Services for Mobile TCP/IP Networking. Report C-1995-15, Univ. of Helsinki, Dept. of Computer Science, Helsinki, Finland, April 1995.

[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. Revised version to appear in Mobile Computing, pp. 253–270, Kluwer, 1996.

[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)*, pages 132–139, 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 (work in progress), IETF, December 1995.

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

[Rom88] J. Romkey. A Nonstandard for Transmission of IP Datagrams over Serial Lines: SLIP. Request for Comments 1055, Network Information Center, June 1988.

[Sim94] W. Simpson. The Point-to-Point Protocol (PPP). Request for Comments 1661, Network Information Center, July 1994.

[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.