Overview of the Mowgli Project

Dr. Timo Alanko, Markku Kojo,
Prof. Kimmo Raatikainen

Department of Computer Science, University of Helsinki

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

E-mail: {alanko, kojo, kraatika}@cs.Helsinki.FI

Abstract

Modern cellular telephone systems extend the usability of portable personal computers enormously. A nomadic user can be given ubiquitous access to remote information stores and computing services. However, the behavior of wireless links creates severe inconveniences within the traditional data communication paradigm. In this paper we give an overview of the problems related to wireless mobility. We also present a new software architecture for mastering the problems and discuss a new paradigm for designing mobile distributed applications. The key idea in the architecture is to place a mediator, a distributed intelligent agent, between the mobile node and the wireline network.

1 Introduction

Developments in mobile communication technology have radically improved the information processing capabilities of man. Usage of modern cellular telephone systems is expanding dramatically; in Finland the penetration has exceeded 40%, and other industrialized countries are following at a rapid pace. At the same time most of the urban areas tend to be covered by mobile telephone networks. Hence, even nomadic users, not

working in their offices, can now have ubiquitous access to various information processing services, such as retrieval of reports, access to e–mail, data base queries, WWW browsing. In addition, the current technology of personal computers has achieved a level where nomadic users more or less expect to get, even out of office, a similar quality of service as when working with their desktops. In practice, however, there are several reasons why these expectations cannot always be fulfilled.

The wireless world has behavioral characteristics for which the wireline world—neither the communication infrastructure, nor the applications—is prepared. The first aspect is the modest performance of the cellular telephone system: the throughput is low and the latency is high. The transfer of a 3 megabyte picture over a 9.6 kilobits/s link takes more than one hour—possibly a surprise to a customer used to retrieve imposing WWW pages in some few seconds.

A second source of problems is the instability of the wireless link. The user may move out of the coverage area, or the radio conditions may deteriorate. Transmission errors are detected by the wireless link protocol, and, to some extent, the protocol can recover from them. However, there is a cost involved: an increased delay. This delay may create problems in the fixed network. The protocols interpret increased delays as a symptom of a congestion, and trying to relieve the problem they decrease the transfer rate. This recovery action does not cure the illness, and it decreases the performance.

Changing conditions can even lead to a disconnection of the link. In fixed networks a broken link typically means an irrecoverable failure for the end user. In a wireless environment the problem may disappear within minutes or even seconds. However, current systems expect the user to repeat the whole dialup–login procedure, and the applications typically have lost the information that belonged to the interrupted session. In some field experiments up to 40% of the sessions were broken unexpectedly, and the user had to reestablish the session.

Protocols tuned for the traditional, well–known wireline world are sometimes grossly inadequate and sometimes even counterproductive when cellular telephone networks become involved. This is true for all protocol layers: link control, control of network usage, and control of application functionality.

The wireless telephone link is the bottleneck of the data communi-

cation path. Thus, tuning of the protocols that control the traffic over it is of vital interest, and this tuning should focus on new areas: local efficiency, adaptability to changing conditions, and ability to recover from certain types of failures.

We have developed the Mowgli architecture to alleviate these problems [5]. The key idea is to place a mediator, a distributed intelligent agent, between the mobile node and the wireline network. The mediator controls the underlying wireless link according to rules specified by the end user. It can adapt its behavior to different conditions of the wireless link and to different wishes of the end user. In a disconnected state both of its parts can act rather independently in the role of the "non–accessible" partner of distributed computation. It can also be used as a platform for user–accessible control tools. In essence, there is a shift of paradigm: the traditional "client–server" paradigm has been replaced with a new "client–mediator–server" paradigm.

Similar kinds of mediator–based approaches have been introduced for wireless LAN's. In these approaches the problems are treated only at the transport layer. However, when slow wireless links are involved, all protocol layers need to be addressed. In the Mowgli approach, we cover all the layers up to the application layer and the user interface.

A prototype implementation of the Mowgli architecture exists in an environment consisting of Linux and Windows platforms and the GSM network.

2 TCP and Wireless Links

2.1 Problems in TCP/IP Over Wireless Links

Under favourable radio conditions a wireless telephone link behaves almost like a normal PSTN link. Therefore, it can be expected that existing Internet applications still work when the conditions are good. However, differences between wired and wireless links cause complications. The common data communication protocols, like the TCP/IP Internet protocols, do not expect problems like those occurring in a cellular telephone network.

The TCP/IP protocols are designed for wired networks, and recent research shows performance problems due to inappropriate operation of

current TCP/IP when wireless links are involved [1, 2, 6]. The experimental results indicate that even minor disturbances in connections may lead to suboptimal performance—sometimes to delays which irritate users—and even to failures at the application level.

To improve the user–visible performance several enhancements are needed:

- a link–level protocol tuned for a slow wireless link,
- facilities for parallel data transfer operations some of which are to be executed in the background, and
- an elaborate multiplexing system giving higher priorities for the more urgent tasks (especially for interactive tasks with a waiting user).

The transmission control mechanisms of TCP are based on one basic performance metric: the round–trip time. An abruptly increased round–trip time may cause the TCP retransmission timer to go off, which in turn is interpreted as a sign of insufficient capacity somewhere in the network; the recovery mechanism is to decrease the sending rate. For wireless connections, like the GSM non–transparent data service, highly variable transmission delays are typical, but they have totally different origins: link-level retransmissions generated by bursts of errors on the radio link. Hence, the reaction of TCP is incorrect. What is needed is a separation of control: the wireless side and the wired side should be controlled independently of each other.

The wireless link is vulnerable. The mobile workstation may move through an uncovered area, or the radio conditions may temporarily deteriorate. In both cases the link becomes inaccessible for some period of time. Neither the TCP/IP protocols nor applications are prepared for this; in the wired network a broken link is considered to be a fatal failure, and typically all active operations are terminated. A mediator can obviously be used to mask intermittent breaks.

Present cellular telephone systems are circuit switched, and charging is based on the connection time. Thus, the customary habit of working with the link open over the whole session is not reasonable. Clearly, a more elaborate control of the wireless link is needed.

2.2 TCP and GSM Data

Figure 1 shows how the commercial non–transparent GSM data service can be used to connect a mobile workstation to the fixed Internet using standard Internet protocols.

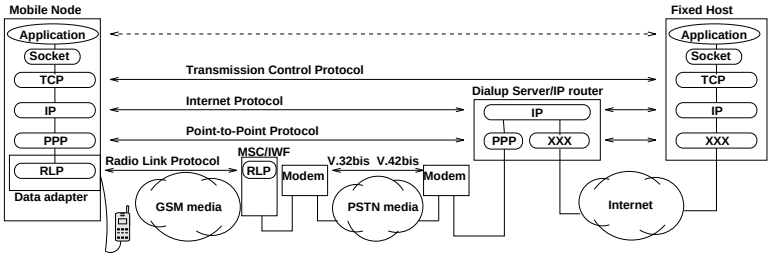

Figure 1. TCP/IP and PPP over GSM data service

The Point-to-Point protocol (PPP), or alternatively the Serial Line Internet Protocol (SLIP), is used as the framing protocol between the mobile workstation and a dial–up server within the fixed network. The dial–up server functions as a router between the point-to-point link to the mobile workstation and rest of the Internet.

IP datagrams are encapsulated within PPP (or SLIP) frames so that TCP/IP protocols can be used in an end-to-end manner between the mobile workstation and any other Internet host. This is a very typical arrangement, that has worked well for connecting modem users to the Internet over PSTN. However, the uncertainty of the wireless medium brings problems not found in the PSTN environment.

Many of the performance problems are related to the slow–start congestion control algorithm [3] used by TCP. The slow–start algorithm is employed on two occasions: at the start of each new TCP connection and whenever the network is congested. The slow–start algorithm is applied at the start of each TCP connection just for the case that the network might be congested at that particular moment. The theory is that this inefficiency at the beginning of the connection is negligible because it only occurs at the start of the connection. This is true for connections that

transfer a significant amount of data. However, in very short–lived TCP connections, such as those created by WWW type activity, the inefficiency is significant. In the GSM data environment with its high latency, the slow–start together with the three–way handshake employed at the start of each TCP connection creates a delay visible to the end–user.

The TCP protocol has been tuned for the relatively reliable fixed networks, in which a packet loss due to transmission errors is rare. Therefore, the protocol assumes that each packet loss is a sign of network congestion and the slow start is triggered whenever a packet is lost. Unfortunately, in a wireless environment packet losses due to transmission errors are quite common. A TCP connection, spanning both wireless and fixed networks, will have poor performance because TCP misinterprets every lost packet as network congestion and slows down the rate of transmission. The result is that even though new data could immediately be delivered over the wireless link, the sending TCP layer, instead of transmitting packets, waits for a non–existent congestion to disappear.

In the non–transparent GSM data service the Radio Link Protocol (RLP) is used to provide reliable transfer over the wireless link. In order to correct transmission errors the RLP protocol retransmits corrupted frames over the wireless link. The result is that transmission errors corrected by RLP will be perceived as transmission delays. Usually the delays are long enough for TCP to interpret them as lost packets, since acknowledgements do not arrive in time. Again, TCP assumes that there are congestion problems in the network and slows down the rate of transmission.

Delays due to RLP retransmissions also cause other performance problems. A notable one is that, in addition to triggering congestion control, such a delay may also cause the sending TCP layer to unnecessarily retransmit packets. Naturally, the retransmissions will also go over the wireless link consuming bandwidth that could be better used for other traffic. Further retransmissions are caused by the fact that the components in the GSM data service can buffer tens of kilobytes of data along the communication path, particularly in modems and in the dial–up server that functions as a router between the fixed and wireless networks. The resulting queuing effect increases the round–trip time causing further retransmissions followed by slow starts. Unnecessary retransmission accumulation is a problem that occurs more probably with TCP connections where a significant amount of data is transferred, e.g. in file transfers.

Problems due to delayed acknowledgements have also been reported in an earlier study [8]. The phenomenon was called ack–compression.

A particularly difficult case of the unnecessary retransmission syndrome manifests when the TCP send window is fully open (transmission has gone well for a while), most of the segments in the send window have been buffered along the communication path, and a long delay on the wireless link causes all those segments to time out. According to the slow–start algorithm the congestion window is set to one segment and the first timed–out segment is retransmitted. Soon, the sending TCP receives an acknowledgement for the first segment—however, not in response to the retransmission but to the original transmission. Since the sending TCP is receiving acknowledgements, it assumes that everything is all right, increases the congestion window and retransmits some more segments in accordance with the slow–start algorithm. Thus, retransmitted TCP segments keep piling into the buffers along the communication path. The syndrome may repeat for awhile but eventually it dies out: at each repetition the congestion window must increase at a slower rate.

It should be noted, that TCP implementations differ in the way they handle retransmissions. Some of them adhere to the TCP specification more strictly than others. In addition, the specification is not entirely clear in how to deal with multiple packets "timeouting" simultaneously. In TCP the retransmission time–out value (RTO) is constantly adjusted according to measured round–trip time estimates [3]. In our experience, the algorithm does not adapt well to the highly variable delays of a wireless environment, causing either numerous unnecessary retransmissions or too slow retransmissions when packets are actually being lost. In addition, some TCP implementations, which are optimized for fixed networks, incorrectly use a quite short initial retransmission time–out at the start of a TCP connection.

The low initial value of RTO causes unnecessary retransmissions before the algorithm adjusts to the high–latency link. This is not a serious problem in long–lived TCP connections, in which a lot of data is transferred, but in short–lived TCP connections a significant fraction of the transmitted data segments can in fact be retransmissions. Part of the problem is that adjusting the RTO is slow, because the algorithm attempts to smooth variations in measured round–trips. The main reason is that round–trip times can not be reliably measured from retransmitted

segments [4]. Thus, the unnecessary retransmissions in the beginning of the connection slow down the RTO adjustment causing more retransmissions.

A further issue with multiple TCP connections over a low–bandwidth link is that they can interfere with each other. Even a single bulk transfer over the wireless link causes the round–trip times observed by other connections to reach several (5–10) seconds [6].

Again the reason is that data is buffered along the data communication path below the TCP layer. Especially new connections have difficulties: the retransmission timer takes a very long time to adjust upwards to the relatively high observed round–trip time. The too short initial RTO causes numerous unnecessary retransmissions, and each time the sender applies the slow–start algorithm, making the adjustment very slow. In our experience, it may be almost impossible to start an interactive connection when there are ongoing bulk–data transfers.

A small additional problem with TCP/IP performance over a low–bandwidth link is that the combined IP and TCP headers are quite verbose (typically 40 bytes). While Van Jacobson's header compression [3] is typically used with PPP framing, the overhead still shaves off a few percent of the available bandwidth.

To summarize, there are a number of subtle performance problems in using the TCP protocol over the non–transparent GSM data service. Because of the relatively high latency of the wireless link, short–lived TCP connections are slowed down by the initial three–way handshake, TCP slow–start, and the unnecessary retransmissions caused by a too tight initial retransmission time–out. Long–lived TCP connections, on the other hand, suffer from an excessive queuing effect on the wireless path, coupled with subtle interactions between congestion control and the round–trip time estimation algorithm used to calculate the retransmission time–out. Furthermore, parallel TCP connections competing for the same constrained link interfere with each other in a way that causes additional retransmissions.

3 Mowgli Communication Architecture

3.1 Architectural Overview

In the Mowgli architecture a mobile node is connected to the fixed network through a wireless telephone link. The key idea in the architecture is to separate the control of the behaviorally different wireline and wireless worlds. The central role in this separation is given to the node on the border of these two worlds, which provides the mobile node with a connection point to the wireline Internet; the node is called the mobile–connection host (MCH). This node also offers a platform for intelligent agents, called proxies. The proxies represent mobile–node applications in the wireline network, and they are capable of operating autonomously, even when the mobile node is not reachable.

The mediator approach allows us to replace the regular TCP/IP protocols of the Internet with a specialized set of communication services for the slow wireless link. These services are provided at three different layers: at the agent–proxy layer, at the data transfer layer, and at the data transport layer.

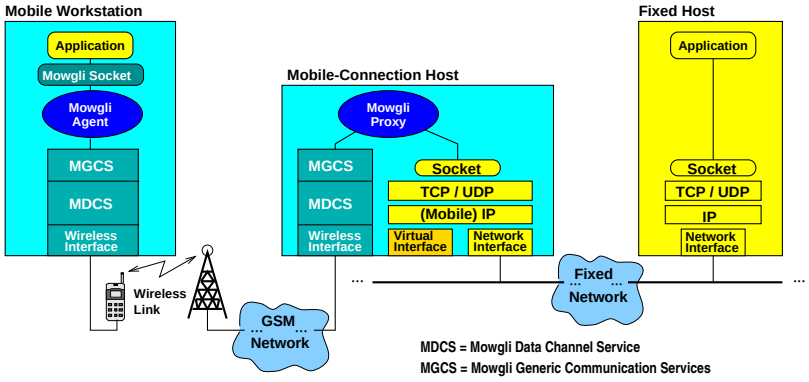

Figure 2. Mowgli architecture

Figure 2 depicts the logical organization of the service architecture, provided on three different levels: on the agent–proxy level, on the generic

communication level, and on the data transport level. The basic communication services, which are available through an application programming interface called the Mowgli socket interface, include the socket operations in the traditional Berkeley socket interface. Because the Mowgli sockets have the same functionality as the traditional TCP/IP sockets, existing applications can be executed on the mobile node without modifications. This approach amounts to splitting the traditional client program into two parts—one on each side of the wireless link.

3.2 Agent and Proxy

On the agent level the communication services are implemented as agent–proxy teams. In the Mowgli system we have two basic kinds of teams: generic agent–proxies and application–specific agent–proxies.

Generic agent and proxy take care of delivering data over the wireless link; for each TCP socket a pair is generated. A generic pair does not perform any specific optimizations; agent and proxy simply relay data over the wireless link using the Mowgli data transport level services.

Application–specific agent and proxy are tailored for a specific application protocol; when controlling data transfer they can adapt their behavior depending on the characteristics of the application, the needs of the user, and on the quality of service available in the wireless environment. The Mowgli WWW software is a good example of an agent–proxy team taking advantage of application semantics [7].

In Mowgli, the communication services available in the generic communication service level are specifically designed to be used in communication over a wireless (telephone) link, and between an application–specific agent and the corresponding proxy.

3.3 Mowgli Data Channel Service

At the data transport layer, the TCP/IP core protocols are replaced with the Mowgli Data Channel Service (MDCS) [6], which is designed to cope with the special characteristics of slow wireless links. All communication above the data transport layer uses the transport services of the MDCS.

The communication in the MDCS is based on data channels. Each data channel has a priority and a set of additional attributes for controlling

the behavior of the channel in case of exceptional events. Multiplexing of data channels is based on a priority scheme. This technique can be used to significantly improve the user–visible performance of data transfers. For example, when interactive communication is directed over a high–priority channel, the response times are in practice independent of the existence of any low–priority background data transfers. In addition, the user may start several simultaneous file transfers, which can be executed either serially or in parallel.

Another important feature of the MDCS is improved fault–tolerance. Particularly, the MDCS is designed to make a recovery from unexpected temporary disconnections of the wireless link efficient. In order to accomplish this, MDCS maintains the state information of the ongoing transmissions; after an interruption the transmissions can be resumed in one round–trip time.

In the design of the Mowgli Data Channel Protocol (MDCP) we have solutions that are different from those of the standard TCP/IP. As was discussed in Section 2, the TCP/IP protocols do not suit well to the characteristics of the wireless links: TCP employs sophisticated techniques not suitable in a wireless environment, the headers are long, and the usage of link capacity is wasteful.

Another source of problems was buffering within the GSM: the data communication path between the mobile node and the MCH typically involves several components with a significant amount of buffer space. If the amount of buffering is not controlled, the observed round–trip time between the mobile node and the MCH may rapidly increase to several seconds. Obviously, unrestrained buffering also makes smooth multiplexing of data channels more difficult; it even hampers delivering control packets and high–priority data packets in a timely manner. Hence, the MDCP protocol employs link–level flow control to limit the amount of buffering along the path between the mobile node and the MCH.

The high performance of the MDCP protocol has been achieved by minimizing the amount of protocol overhead, specifically protocol headers and superfluous round–trips over the wireless link. The protocol information in data packets occupies only 1–3 bytes plus an optional checksum field. In addition, MDCP always tries to transmit data at the best possible speed the wireless link is capable of: it does not employ slow–start, and a three–way handshake during connection establishment is optional.

It also sends acknowledgements only sparingly. Thus, MDCP reaches the full bandwidth much faster than TCP, and it is capable of retaining the transfer rate rather stable even when temporary delays occur on the wireless link [6].

3.4 Mowgli Data Transfer Service

The Mowgli Data Transfer Service (MDTS) is one of the services offered at the generic communication service level. This service is able to take over the responsibility of transferring structured user data over the wireless link. The basic element of information for transfer operations is an Information eXchange Unit (IXU). In general, an IXU is something that the user considers as an independent unit of information, the transfer of which he or she may want to control. Examples of IXU's include mail messages, files, print jobs, and WWW pages or inline images of WWW pages.

Each IXU has a set of attributes, which are used in controlling the transfer. The MDTS provides operations for creating IXU's to be sent, for receiving IXU's, for managing transfer queues, and for changing attributes of specified IXU's. Due to the attribute system, the MDTS is able to operate rather independently of various kinds of background transfers: it can make decisions about invoking transfers when conditions are favorable, about postponing transfers when conditions deteriorate, about trying to recover from failures, and about canceling operations.

The application–specific agents and proxies are capable of extracting appropriate data units, creating IXU's, and sending them, as well as receiving IXU's and translating them back to application protocol data.

4 An Example: WWW Agent and Proxy

The Mowgli WWW software [7] is an excellent example of customized agent–proxy team that is optimized for a specific application. Mowgli WWW design is based on the following key principles:

- Allow use of popular WWW browser and server software unmodified.

- Minimize round trips and the volume of data transmitted over the slow link.
- Discourage burstiness on all levels, since bursty data traffic translates to low utilization of the link.
- Allow operation without network connectivity.
- Enable asynchronous background fetching of documents.
- Provide the user with fine–grained control over how the wireless (or slow) link should be used.

The Mowgli WWW Agent and Proxy cooperate in order to fetch hypermedia documents from WWW servers to the mobile node. They communicate with each other using the optimized Mowgli HTTP (MHTTP) protocol, which minimizes the data traffic over the wireless link.

The Mowgli WWW Agent and Proxy establish a binding when the user starts a browsing session. The binding is maintained until the user declares that the browsing session is over. This means that the Mowgli WWW Proxy retains awareness of the user even over periods when the mobile workstation is disconnected from the network. In this way the proxy can perform operations in the fixed network on behalf of the mobile node when the node is momentarily unreachable.

The MHTTP is highly optimized; for example, it exploits binary encoding of HTTP headers to reduce protocol overhead. In addition, it employs the MDCP protocol for transport; thus, it is not exposed to delays due to the three–way handshake and slow start, which reduce the transfer speed of the normal HTTP on top of regular TCP. The true power of the approach, however, lies in the way the agent and proxy exploit available knowledge about the behavior of the application. They make extensive use of predictive upload of documents and document objects by prefetching document objects embedded in WWW documents. When the proxy receives a request for a WWW document from the agent, it fetches the document from a WWW server and forwards the document to the mobile node. While doing this the proxy looks at the document and starts prefetching the embedded objects from the WWW server and forwarding them to the agent with the assumption that the WWW browser will request them soon in any case. Figure 3 outlines how a Web page with two

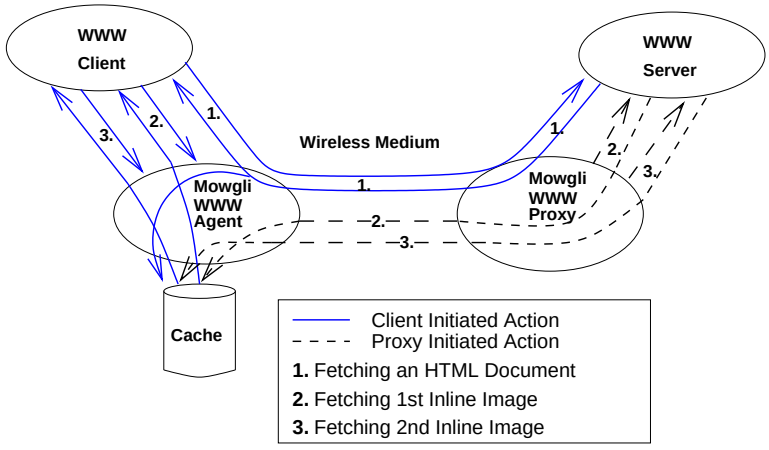

Figure 3. Reducing round–trip delays in Web browsing

inline images is fetched. This technique decreases the number of round–trips and makes the data traffic less bursty, which, in turn, reduces the response times observed by the end–user. Our performance measurements indicate that total transfer time in fetching typical WWW documents is reduced significantly [7].

5 Discussion

The conceptual design of the Mowgli system has its origins in the different natures of the wireless and wireline worlds, in some specific features of the wireless link, and in the problems of a nomadic user. The main idea is to develop new functionality using the concept of a mediator that consists of an agent–proxy team and of associated infrastructure.

This approach leads to an essentially improved performance when compared to the regular TCP/IP–based solution [6, 7]:

- The overall performance was improved, and the number of very long delays was decreased dramatically; for example, in bulk data transfer

benchmarks the upper quartile of the transfer time was decreased by 50%.

- Response times for interactive queries could be kept at a low and predictable level, independent of the background traffic; in the benchmarks the response times remained at the level of 2–3 seconds instead of about 8 seconds with regular TCP/IP.

The idea of splitting the end-to-end control turned out to be fruitful. Firstly, it helps to utilize the capacity of the bottleneck device, the wireless link. The proxy on the MCH acts as a high–level buffer balancing variability in arrival rates from the wireless side, which are due to distortions on the radio link, and from the wireline side, which are due to congestion in the Internet. Secondly, the splitting prevents irregular behavior on either side from harming the control of the other side: wireless delays do not cause unnecessary end-to-end retransmissions, nor does wireline congestion affect the operation over the wireless link. Of more importance for the end users are the qualitative improvements: occasional breaks in the radio link have no visible effects at the user interface level, automatic control of the telephone connection is a blessing for unaccustomed nomadic users, and the possibility to transparent background operations relieves the nervous end user from a lot of idle waiting.

Today the research in mobility is primarily concentrated on the problems of physical mobility: The system perceives how the nodes move. In the Mowgli environment the cellular telephone system hides certain issues in the terminal mobility like paging, hand–overs, and location updates. Therefore, we are able to focus on the next set of problems: when to connect, how to control disconnected agents, and—in general—how to improve the dependability of applications comprising wireless data communications.

Acknowledgements

This work was partially supported by Nokia Mobile Phones, Nokia Telecommunications and Sonera (prev. Telecom Finland).

References

[1] *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), pages 24–44, October 1994.

[2] *R. Caceres and L. Iftode.* Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments. *IEEE Journal on Selected Areas of Communication*, 13(5), pages 314–329, June 1995.

[3] *V. Jacobson.* Congestion Avoidance and Control. *Proceedings of the ACM SIGCOMM '88 Symposium on Communications Architectures and Protocols*, pages 314–329, August 1988.

[4] *P. Karn and C. Partridge.* Improving Round–Trip Time Estimates in Reliable Transport Protocols. *ACM Transactions on Computer Systems*, 6(4), pages 364–373, November 1991.

[5] *M. Kojo, K. Raatikainen, and T. Alanko.* Connecting Mobile Workstations to the Internet over a Digital Cellular Telephone Network. In *Mobile Computing* (ed. T. Imielinski and H.F. Korth), pages 253–270. Kluwer Academic Publishers, 1996.

[6] *M. Kojo, K. Raatikainen, M. Liljeberg, J. Kiiskinen, and T. Alanko.* An Efficient Transport Service for Slow Wireless Telephone Links. *IEEE Journal on Selected Areas of Communication; Special Issue on Networking and Performance Issues of Personal Mobile Communications*, 15(7), pages 1337–1348, September 1997.

[7] *M. Liljeberg, H. Helin, M. Kojo, and K. Raatikainen.* Mowgli WWW Software: Improved Usability of WWW in Mobile WAN Environments. *Proc. IEEE Global Internet 1996 Conference*, pages 33–37, London, UK, November 1996.

[8] *L. Zhang, S. Shenker, and D.D. Clark.* Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two–Way Traffic. *Proceedings of the ACM SIGCOMM '91 Symposium on Communications Architectures and Protocols*, pages 133–147, September 1991.