Session Management in Mobile Data Communications

15 October 1999

Table of Contents

Preface

This Master Thesis work was done at Ericsson Radio Systems, Core Unit of Technology and Research(RCUR), in Stockholm, Sweden as the final part of my studies to get my Master of Science degree from the Royal Institute of Technology (KTH). The Core Unit of Technology and Research is an applied research organization within Ericsson working with mobile networks and systems technology in order to generate new business areas.

I want to thank my Ericsson supervisor Yuri Ismailov for all the help he has provided me with, my advisor at KTH Viktoria Elek for all the comments to improve my work, and my examiner Gunnar Karlsson for his interest. Finally I want to show my appreciation to Tomas Larsson for all the help and the tips he gave me during the thesis, and to Ignacio Mas Ivars, my opponent, for his valuable comments during the opposition.

Abstract

The area of mobile communications has become very important in academic and corporate research over recent times. People are getting more and more used to move around with their mobile terminals and they want to be provided with access to network resources in any place at any given point in time, that is, they want ubiquitous access.

In the strive to provide ubiquitous access, a lot of problems in different areas of mobile communications, like data transmission, are coming up. The major problem in mobile data communications relates the change of IP addresses in mobile terminals when the mobile user roams around different IP networks. Due to the nature of IP and TCP, a modification of IP address while ongoing a data transfer, implies that all currently opened TCP connections are terminated forcing the retransmission of all the data that was previously transmitted through these TCP connections.

IP address changes are very likely to happen in mobile environments. Mobile users get access to the networking resources through different ISPs, receiving usually different IP addresses from each ISP or network they visit. At the same time, network access is usually provided by wireless links, which offer less bandwidth, higher latency and higher bit error rates(BER) than wired links. The bad quality of the wireless link causes sometimes long periods of disconnection, which can also induce IP address changes in the mobile terminals.

When access to network resources is provided through a fast fixed network, restarting the data transfer might not be an unbearable annoyance for users, since data transfers usually do not require a long time to be completed. The situation changes when the network access is provided through low speed links, like wireless links, in which completing certain data transfers requires a long time, and restarting the data transfer becomes unacceptable.

Mobile data communications is an area in which solutions are needed to cope with the problems introduced by the low speed, bad quality network access links and the frequent IP address changes of the mobile terminals. The session management service is developed within this framework.

1.Introduction

Over recent years mobile computing has become a very hot matter of research in both corporate and academic level. Computer communications are getting closer to the people, and that has stimulated developers to try to adapt computer communications to people's uses and requirements. **What people want is to get access to the information independently of location and time.** In the last few years, the number of subscribers to mobile communications services like telephony and basic data transmission services like the Short Message Service (SMS) has grown substantially, and vendors and developers foresee a larger growth in the near future.

At the same time data communications are playing a more important role in the telecommunications sector, counting for a great part of the revenues of many companies. This is the one of the reasons why developers and researchers, are trying to integrate data communications into mobile environments. The goal is to improve the quality of the currently offered services and broaden the range of services available to mobile users, offering Internet content and other value-added services like e-mail. The areas that are being more thoroughly studied are:

•Wireless networks. These networks are being continuously studied and developed to improve the access and performance they offer. Some examples of this are the new data transmission services based on packet switching like GPRS.

•User interfaces. More user-friendly terminals are being designed taking in consideration the constraints introduced by computing in mobile environments.

•Protocol development. Protocols are being studied to try to add features that introduce communication improvements in mobile environments. One example of this is WAP(Wireless Application Protocol)[17].

The work presented in this thesis introduces a totally new proposal to enhance data transmission in mobile communications. The thesis is structured in four major chapters.

1. Chapter 1 is an introduction to the mobile computing environment, and some of the problems and challenges found in mobile data communications.

2. Chapter 2 is the most important chapter and titles the thesis. The concept of session in computer communications is introduced, and the **session management service**(SeMS), which is the most important contribution of the thesis, is described as an element to provide reliable data transmission in mobile environments.

3.Chapter 3 introduces the idea of the TCP multiplexer architecture to improve performance of data communications in mobile environments and describes an implementation as part of the work done in this thesis.

4. Chapter 4 describes a proposal to introduce security within the SeMS service.

1.1. Problem Statement

As mentioned before, the interest in computer networking has increased significantly. We have an example of this just by seeing how the Internet has developed and become a familiar term to everyone. At the same time computer communications are adapting continuously to people's habits providing for the development of fields like mobile communications.

Focusing on mobile communications, we can see that the range of services available in this type of communication is very short, mainly basic telephony and data services. The target is to be able to grow in this field in order to integrate Internet services in mobile environments, as well as providing as continuous connectivity as possible to the mobile user. This development has encountered, however, a series of obstacles which need to be surpassed if the target is to be achieved. In this section, we are going to discuss some of these obstacles or problems that prevent us from reaching the goal.

1.1.1.Mobility Issues.

Mobility offers the possibility for mobile users or terminals to move between different IP networks using a wide range of network access technologies. The main problem in providing mobility is that IP was designed having fixed communications in mind[16]. The address allocation schemes and the IP routing model were designed to address the mobility issue.

The current IP routing model neither allows addresses to dynamically change the physical location in the network nor allows open connections to be kept open when changing IP addresses.

This problem is aggravated if we account for the new ways of mobility that have appeared with the advent of third generation wireless systems. It is expected that each mobile terminal will have the ability to connect to the Internet via different access networks, wide and local area networks[2]. Mobile terminals will have to be able to change the network interface and physical link “on the move”, depending on the available communication media at the current geographical position. In the case in which several access networks are available simultaneously at a geographical position, the mobile terminal will have to choose the best access network. The process of reconfiguration should be done transparently to the user, interrupting the data transfer as little as possible. A certain period of disconnection is, however, unavoidable because of the time it takes to physically switch to a different communication media. It is important to have in mind that:

Users do not want to lose connectivity when changing the network access and/ or IP address.

The current proposed solutions to provide mobility are based on the assumption that the mobile terminal keeps it’s IP address independently on the IP network it attaches to. In order to provide continuous data transfer when changing to a different IP network it is necessary to establish proper routes to the mobile terminal while keeping the existing TCP connections. Clearly, this can be done only on the level below TCP since IP has no idea about TCP connections.

The most popular solution is Mobile IP. In the context of wide area wireless network, this is a very important solution because as long as the terminal stays within the same ISP coverage area (geographically it can be country wide) and exploits the same type of network there is no need to re-establish TCP connections even when moving to another IP network. Unfortunately, this solution does not support the scenario when a mobile host is forced to change an IP address and all currently opened TCP connections have to be re-established. Data transferred by the time TCP connections are terminated will be lost, because both client and server applications are memoryless in this sense. This is not unusual because **Mobile IP** is not universally deployed and **cannot not cope with all the possible events that may arise in mobile environments**. Moreover, it includes special overhead in the communications due to the fact that it uses triangular routing to deliver information to the mobile user.

The autonomy and quality of service for the mobile users is therefore limited, because they are forced to retain a connection to a certain ISP, even if it gets a a better service from a differ-ent one, until the data transfer is completed. It is necessary to find ideas to provide better QoS and to increase the autonomy of computing. This can also have a positive impact in the fees mobile users have to pay to have access to the desired services since the competition between the ISPs would increase, improving the services they offer and lowering the prices they charge.

1.1.2.Reliability Issues.

In mobile communications network access is mainly provided using wireless networks. Wireless networks have special characteristics that differ from those found in wired networks. There is less available bandwidth, higher latency and higher bit error rates(BER)[1]. Burst errors, disconnections, and delays due to link layer corrections and retransmissions are also usual in wireless networks. Thus, **the quality of the links in wireless networks is usually poor**, at times causing long disconnection periods in the mobile terminals, which can lead to IP address changes and termination of the TCP connections between the communicating hosts. We have already seen that this affects negatively the communications of the mobile user.

1.1.3.Performance Issues.

The performance in mobile communications is not as good as it should be. One of the major reasons for this is that **TCP shows an undesirable reaction to some situations commonly found in mobile environments**[3,6,7]. TCP reacts properly in traditional networks made up of wired links with stationary hosts, since it was designed using assumptions about the characteristics of communications in wired networks.

The quality of wireless links is usually worse than the quality of wired links. This disparity has as a consequence that many of the assumptions in which TCP is based do not apply in wireless environments, yielding an inferior behaviour of TCP over these networks. TCP behaves improperly because it was designed to react to packet losses as due to congestion in the network. When congestion in the network occurs, TCP triggers slow start and congestion avoidance algorithms[11,12] that reduce the amount of packets injected in the network.

Wireless networks are error prone links with high BER, suffering from a significant amount of non-congestion packet losses due to handovers and bit errors. When non-congestion losses occur in wireless links, TCP incorrectly assumes that there is congestion in the communication path and applies congestion control actions that reduce the throughput of the link. This can lead to a low utilization of the link, degrading the communication and the performance of applications[7,10].

1.1.4. Other Issues.

Other common problems in mobile data communications are related to the low processing power of terminals, the reduced battery capacity, the small size of the terminals, the weight, the reduced display, etc. The main issues we are focusing on in this thesis are mobility, reliability, and performance.

1.2.Objective of the Thesis

The thesis introduces a new approach to cope with all the problems previously discussed. The work describes an integrated service for mobile data communications that provides **reliable, high performance, globally mobile, and secure access to Internet services for mobile terminals** with multiple wireless interfaces and/or multiple subscriptions to different Internet Service Providers (ISP). The **approach is based on the concept of session** and it is basically directed for mobile communications, although it can also be used in wired networks where the benefits are lower.

1.3.State of the Art

The concept of session[15,19] has been known for long, i.e. it is included as a layer in the OSI standard. Many protocols use the session concept to provide different services(SIP, WSP, SSL, etc.), although it seems that there is not any other work that uses the session concept to pro-vide services similar to the ones introduced in this thesis. Here, session management focuses mainly on resynchronizing data transfers when TCP connections abruptly terminate in the middle of the data transfer, to avoid restarting data operations from scratch.

2. Session Management

This is the main chapter of the thesis. **Session management** is introduced and described as **a solution to cope with TCP connections being dropped and then re-established**. It uses information from the dropped TCP connections to support recovery in incomplete data transfers. It is, therefore, a solution **above** the TCP layer. It is based on the concept of session. The idea is to identify each individual TCP connection by a session identifier and save state about the different TCP connections to allow communications recovery and resumption if it is required. In this chapter we describe the architecture needed to support this service, how applications are affected, the underlying protocol, and how the session handover support is implemented.

The concept of session is well known and described in the OSI model[19]. It was not widely used because it was not absolutely necessary. Not long ago, most people got access to the Internet through wired networks. The nature of wired networks and the information available through the Internet made session management unnecessary. Most of these wired networks were LANs, and the size and characteristics of objects available to be fetched was not as big and varied as it is today. When a connection was interrupted in the middle of a data transfer, it was not too distressing to restart the data transfer from the beginning.

Over recent years, the Internet has grown consistently to become the world's largest public data network. The development of the Internet has been possible thanks to network access methods which have brought the Internet closer to the people. Most of the new users get access to the Internet through dial-up connections to ISPs, and lately, proposals have been raised to provide access to the Internet through wireless networks to enhance mobile computing. The development of the Internet has also been accompanied by the development of the services and information it offers, making retrievable files larger and more complex.

This development has made many early assumptions about data transfers no longer valid. Solutions are needed to keep up with the development of the Internet and to improve the services provided when accessing it. In the scope of evolution of the Internet mobility is also becoming interesting. The idea is to provide Internet services to users on the move.

Mobility introduces new problems which have to be studied for the sake of the data communications. The most important of these problems is related to the change of IP address in the mobile terminals when moving between different IP networks[1,4,5].

A change of IP address of the mobile terminal affects on-going TCP connections. TCP connections are defined by two tuples: the IP address and the port number used by both peers in the communication. When any of these values changes, the currently opened TCP connections suddenly terminate, and other TCP connections are established leading to the retransmission of data successfully received by a client at the moment the IP address changed, taking in consideration that TCP is memoryless in terms of data transfers.

Session management tries to cope with these situations. It tries to provide global mobility for mobile terminals independently of the available network type and ISPs used during the communication. In the following section, the concept of session as it appears in the OSI standard is shortly explained to allow the reader to get more acquainted with it.

2.1. The Session Layer

The concept of the session layer was introduced and standardized in the Open Systems Interconnection (OSI) protocol stack, see Figure 1. The session layer lies on top of the transport protocol and was included to solve some of the deficiencies present at the transport layer.

The objective of the transport layer is to provide a user-oriented connection service. It takes the responsibility for creating and maintaining a logical connection between end points. The function of the session layer is simply to improve the services offered by the transport layer.

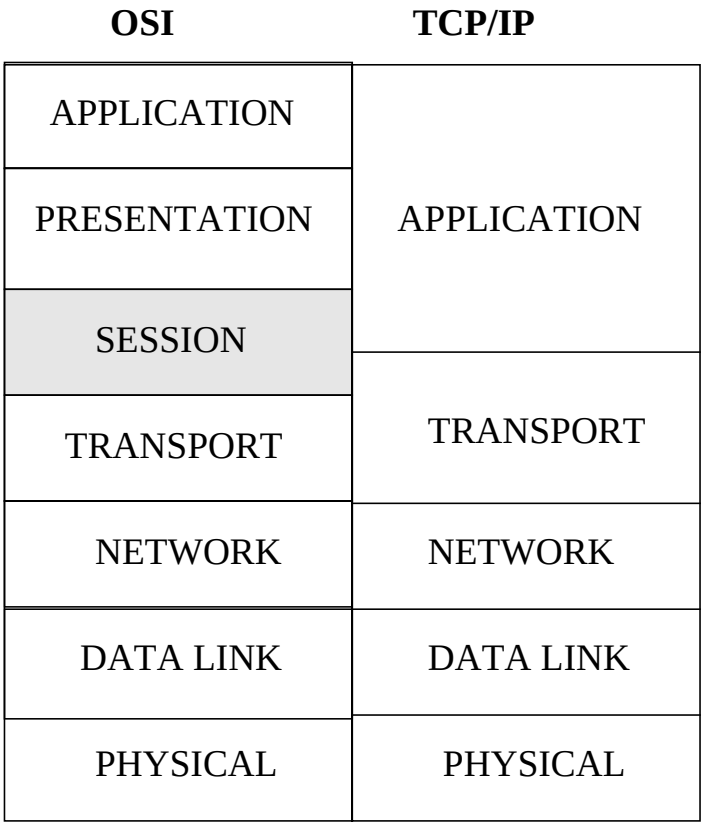


Figure 1. *Mapping between OSI and TCP/IP*.

2.1.1. Session Layer Services.

The value added features to the connection service, introduced by the session layer in the model proposed in the OSI standard are:

- Session establishment.
- Session maintenance.
- Dialogue management.
- Session termination.
- Session recovery.

A full description of these states can be found in many books about this model[19]. In the thesis, we are going to discuss just two of them: dialogue management and session recovery that will help us to get a deeper understanding of the session concept.

2.1.2. Dialogue Management.

When two users establish a connection, their respective session entities will create a session, identified by a session ID. The session will be mapped onto a transport connection, and the parameters of the session will be negotiated. The communication between the user and the session protocol entity generates units of data called records, which are then encapsulated into a Session Protocol Data Unit (SPDU). One session entity establishes a dialogue with the other session entity exchanging the SPDUs corresponding to the records received from the session user. The dialogue management between the two session entities can occur in several ways:

- Two-way simultaneous.
- One-way mode.
- Two-way alternate.

The most complex is the two-way alternate, in which both session entities have to establish a dialogue to determine when their respective turns start and end. The sequence of records sent during one user's turn is called session interaction unit. On the last record of its turn, the user would include an end-of-interaction unit mark. This is in fact a token that is passed to the other user. A user is prevented from sending data unless it is his turn, although the user can send special data called interrupt data to demand, for example, the turn. The other two ways of interaction are simple to understand and we are not going to deepen into them.

2.1.3. Session Recovery.

The most interesting feature provided by the session layer, in the scope of this thesis, is the session recovery service. The approach suggested in this thesis to provide session recovery in mobile data communications is related to the idea of session recovery in the OSI standard.

Before introducing our approach, let's go shortly over the session recovery in the OSI model. In the OSI model, logical units of work or activities are divided into dialogue units which comprise the so called session recovery units. An end user specifies the point at which a recovery unit ends, and each recovery unit is assigned a sequence number.

The dialogue units are delimited by major synchronization points[19], and can be further divided into smaller units by minor synchronization points. There are two different kinds of synchronization points, see Figure 2.

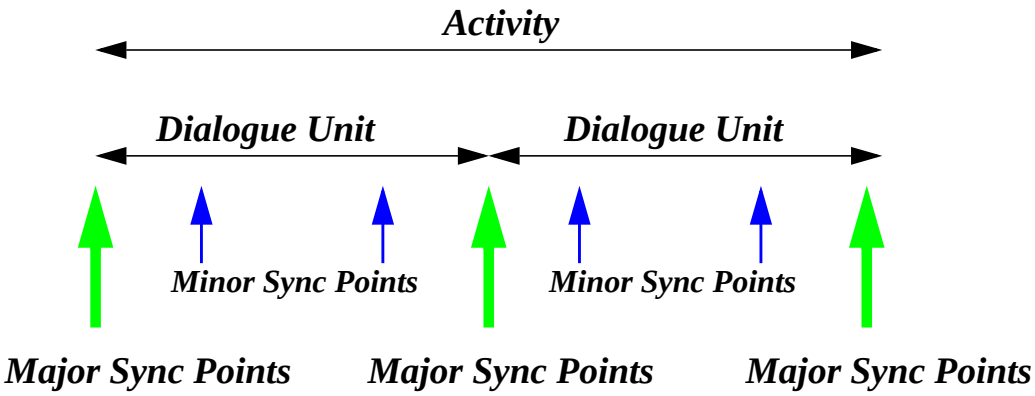


Figure 2. *Types of synchronization points.*

1.Major synchronization points. These points are numbered sequentially creating the intervals that delimit the dialogue units. They have to be acknowledged in order. A major synchronization point cannot be acknowledged if the previous one has not already been acknowledged. These dialogue units are the key element in providing session recovery.

2.Minor synchronization points. These points are used to structure the dialogue units and need not be confirmed.

To recover lost data, the user issues a recover command using the session recovery unit number to identify the point to which the session has to be backed up. Once the session has been backed up to that point, some form of recovery mechanism has to be attempted. It may be up to levels higher than the session layer to take care of the retransmission, or it may be a matter of the session entity to worry about this.

In the OSI standard, it is not the responsibility of the session layer to save any data that has already been transmitted, it only marks data streams with synchronization points to be able to go back to the point of resynchronization. It is the responsibility of the application in the sender to save and retransmit the data from the point of resynchronization.

If the task relies on the session entities, they must maintain a copy of each record exchanged with the session user. To avoid unbounded storage problems, the user must release periodically the storage buffer so that previously serviced records can be discarded.

The session service as it appears in the OSI standard, although it is helpful to cope with temporary disconnections, leads to consumption of additional resources, due to protocol overhead, and does not address the mobility issue. Besides, the use of this session protocol requires that the communicating hosts have the session service implemented, and this is not the current case in Internet applications. After this introduction to the concept of session management, we will start with the description of the work done during this project.

2.2. SeMS Architecture

Session management service (SeMS) is **a new solution to enhance the service provided to users in data transfers.** It introduces an architecture and a protocol that allow the synchronization of data transfers. It is intended for mobile data communications since this is the computing environment in which the benefits of the service are higher, but it can also be used in wired networks.

SeMS is proposed to avoid retransmitting successfully received data in mobile environments. In mobile environments TCP connections often terminate due to the change of IP address in the terminals. It introduces reliability in the connection and a great flexibility in choosing the access network, because it abstracts from it.

The solution is based on the session concept introduced in the OSI model. **In SeMS, each individual TCP connection from each application is assigned a session ID** so that it can be classified and identified. At the same time, **state for each session is saved**, so that it can be recovered in case something happens during the data transfer using the session ID.

The architecture to support the session management service is middleware based. Middleware based solutions are not new, they have been used in many proposals to enhance mobile communications[4,6,8,10]. The idea is to interleave intermediate agents at the interface between the wired and wireless parts of the communications path. The main goal is to speed up web browsing when using low capacity wireless links in Internet access.

In this project, we suggest a middleware based approach to cope with the lack of suitability of TCP/IP to mobile environments. In the suggested solution, the intermediate agents are the session managers. The session managers are agents or proxies that act as separate modules and are able to communicate via the TCP protocol to an application and to another session manager.

Our approach splits the end-to-end TCP connection in three parts, see Figure 3:

1.Between the client application and the client side session manager (CS SMA).

2.Between the CS SMA and the server side session manager (SS SMA).

3.Between the SS SMA and the server application.

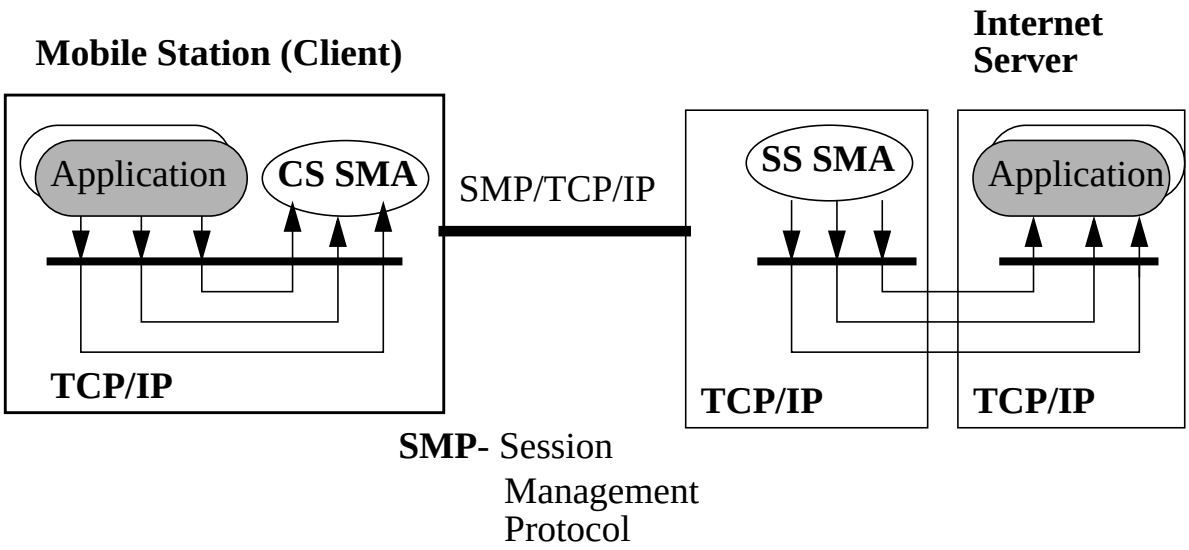


Figure 3. *SeMS architecture.*

The CS SMA and the SS SMA take advantage of the TCP multiplexing which will be thoroughly described in the next chapter. It is an existing architecture[21] that fits in session management because it provides two values that can be used as session ID for the different TCP connections. TCP multiplexing and session management are different things and can be implemented one without the other. But they are also compatible and the TCP multiplexing architecture can be easily integrated in session management.

TCP multiplexing improves performance in mobile communications. It uses a single persistent TCP connection between the CS SMA and the SS SMA to transmit data. Hence, TCP multiplexing supports persistent connections. Persistent connections improve the performance of browsing applications because they use the same TCP connection for the transfer of multiple requests and reception of multiple objects. That way, the overhead introduced by the TCP connection handshake and the slow start algorithm for each new TCP connection is avoided. It could be considered as an improvement to HTTP 1.1.[20]. But TCP multiplexing does not only solve problems due to multiple TCP connections in heterogeneous environments, it also contributes to improve the behaviour of TCP in mobile environments[7,22].

In TCP multiplexing, the TCP connections from the client applications are first classified by session IDs, multiplexed at the CS SMA, transmitted through the single persistent TCP connection between the SMAs, and demultiplexed at the SS SMA. Multiplexing of web server connections is carried out the other way around. Multiplexing and demultiplexing at both session managers require means to identify data corresponding to the different connections. The session IDs assigned at the beginning to each TCP connection are used for this.

The integration of TCP multiplexing into session management generates the session management service(SeMS). SeMS speeds up the communication when using wireless links for any application that supports the session service. At the same time, since SeMS uses TCP/IP, any improvement introduced at the TCP layer or layers below, which does not result in breaking end-to-end TCP connections, is supported by the SeMS architecture. **The main component of the SeMS is the session management protocol (SMP)**. The SMP is implemented at the application layer. It includes the SMP protocol data units (PDU) which are used by the session managers to interact. The session is uniquely bound to the mobile terminal, regardless of how many applications and TCP connections are needed for the user activity.

2.2.1. SeMS Services.

The SMP protocol is the base of the SeMS service and offers the following services:

- Establishment and Maintenance of a session between SMAs.

- Session Termination. We can further classify termination:

  1. *Authorized Termination*, inducted by the client.

  2. *Unauthorized Termination*, can be inducted by the server or can happen due to the link interruption. Any unauthorized termination requires saving the current session by both client and server side SMA.

If the TCP connection between the two session managers terminates abruptly, the CS SMA would catch a notification coming from the interface saying that the connection went down. On the other hand, the SS SMA would not detect by itself that the connection went down, but the TCP connection keep alive timer would expire, so it would notice that the connection between the two session managers went down.

•Session Recovery. The resumption of an interrupted data transfer, starting from some synchronization point agreed by both CS and SS SMAs.

•Session Handover Support. The current SS SMA, based on information provided by the CS SMA, should be able to retrieve state from the SS SMA that was serving the client before it changed SS SMAs.

•Secure Session Transfer. The transfer of state between SS SMAs when session interruption occurs due to the fact that the mobile user changes serving SMA, is secure.

•Access Provider Independence. The use of SMA is not limited to a current service provider due to the fact that the session management service (SeMS) can be located at any place in the Internet. For example a mobile terminal is not forced to use an SMA at the currently serving ISP. On the other hand the ISPs have the chance of not using these service.

•Data Conversion. The session management agents may perform any type of multimedia data conversion/compression for any application requiring that kind of services.

By providing these features, the users of the mobile terminals have the chance of getting a reliable data transfer due to the session recovery and resumption, a better performance due to the implementation of the TCP multiplexer in the session management service, and globally mobile services thanks to the session handover support.

2.3. SeMS Aware Applications

Current applications are not designed to support the SeMS. There are several approaches to make applications SeMs aware. Since SeMS is a service that lies on top of the TCP layer, an application configurable through a proxy is the most basic solution to make applications SeMS aware. SeMS can only be used with protocols that convey information about the final server. Supporting proxy configuration allow us to run the application through the CS SMA, a proxy itself, being this the basic requirement to support SeMS.

There are applications that support proxy configuration. Web browsers are good examples of applications which do not need adjustments to be SeMS aware, because they have the ability to communicate via TCP proxies and because the application protocol, in this case HTTP, conveys information about the original server.

Other applications can also be adapted to support SeMS. We are going to describe three possible approaches to integrate SeMS in the applications.

2.3.1. Application Redesign .

This approach suggests that applications should be redesigned to support SeMS. Although this seems complex and burdensome, it is not so. It is enough with designing applications so that they are capable of running through a proxy and use protocols that convey information about the server.

2.3.2. Implementing the Session Socket Layer.

A second approach is to introduce a session socket layer for an application. The Application Interface (API) for the session socket layer could match exactly the ordinary socket API (might be platform dependent). Socket calls can be extended with some prefix, for example **sems_connect, sems_create, sems_listen, sems_open,** etc. Basically the task of the session socket layer is to translate sems_ calls to ordinary socket calls and to support needed functionality for the SeMS service. A layered structure of this approach is shown in Figure 4.

Actually, the session socket layer needs to understand all types of operation applicable to the active sockets. This include IOCTL calls to the opened sockets which control the socket interface during run time. IOCTL calls are calls to the operating system to control the I/O operations. IOCTL calls do not affect session semantics,

The disadvantage of this approach is that it requires a change in the standard socket calls in the application source code to the **sems_** socket calls. After the application would have to be recompiled and rebuilt. It seems more complex that the application redesign suggested in the previous approach.

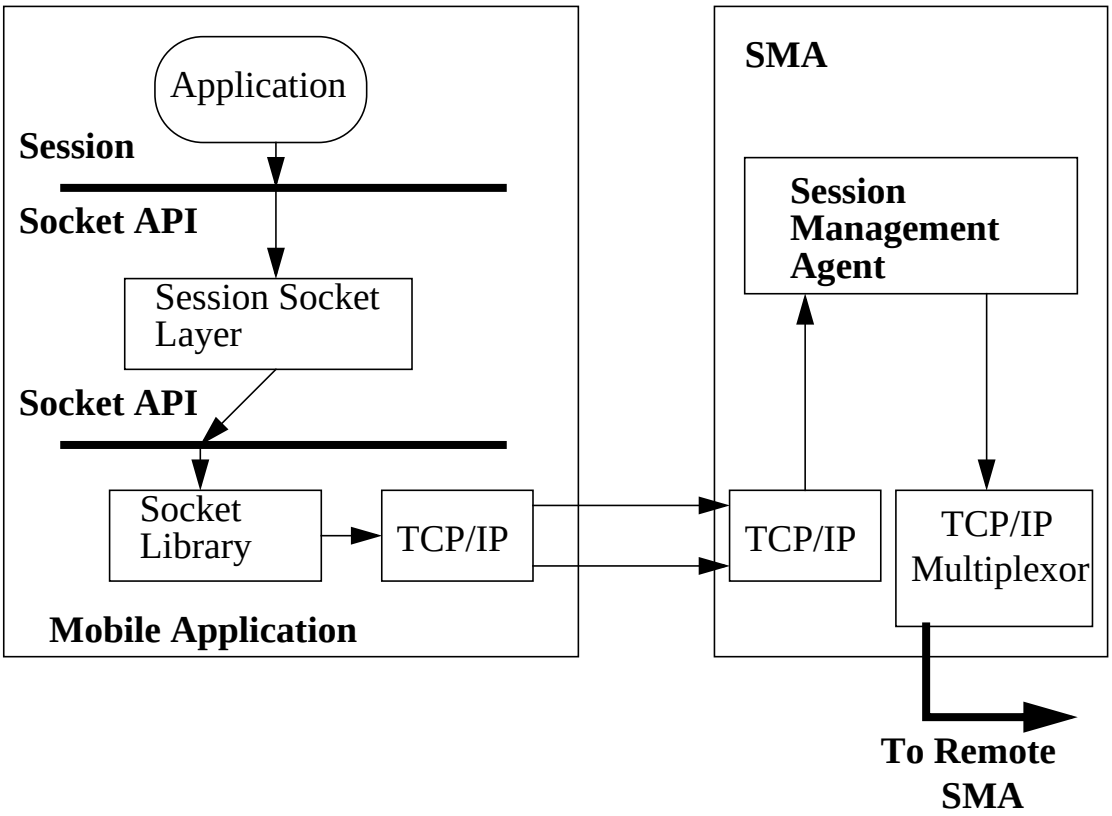


Figure 4. *Implementation of the session socket layer.*

2.3.3. Interception of Socket Calls.

The third approach is based on intercepting and redirecting packets, both unchanged and modified, to the local CS SMA using a socket call interceptor. This can be done either transparently or on a selective basis depending on application.

This feature should be controlled in some way in order to be disabled/enabled for the particular session. The advantage of this approach is that applications would not have to be changed at all in order to use the session management service.

The basic idea of the socket call interception is shown in Figure 5. It is supposed that the socket call interceptor analyses all packets sent to the original server and is able, if needed, to redirect them to the local SMA.

Incoming packets from local CS SMA will be correctly treated by TCP/IP stack and passed through the call interceptor or filter for the modification. If no modification is required the packets will be transparently passed on to the application. This approach seems to be workable although is quite unusual.

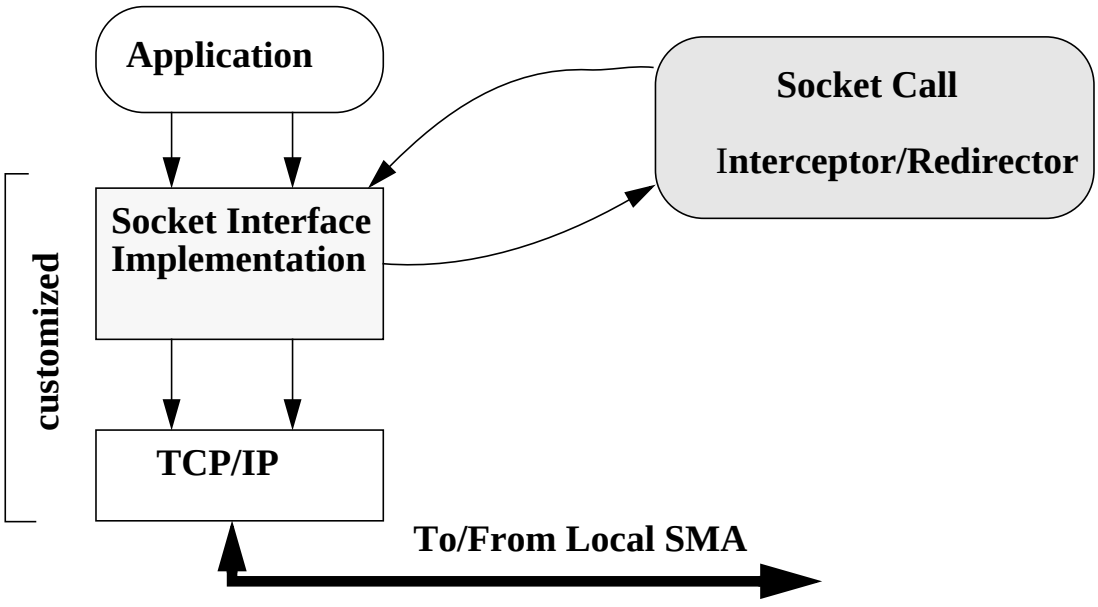


Figure 5. *Socket call interception and redirection.*

2.4. Session Management Protocol

The session management protocol (SMP) is the core element in the SeMS architecture. It is connection oriented and stateful, In Figures 6 and 7 we show the state machines for each of the session managers. SMP has the following states:

•ESTABLISH. This state conforms the initiation of the session between the CS and SS SMAs. The SS SMA enters this state when a new session is going to be established or when some old session is going to be resumed by the client. During this state the data transfer between client and server is forbidden.

•IN_PROGRESS. This state implies an open session where client and server can freely exchange data.

•SAVING. This state is entered whenever the CS SMA or the SS SMA finds out that the TCP connection between them has been terminated abruptly, or that data transmission is not feasible because of some other reason.
•RESUME. This state is entered by the CS SMA when it discovers that the object requested by the client exists in the list of saved states. If the application has some session support implemented, it may send a session resume request to the CS SMA. The SS SMA enters this state when it receives a resume request from the CS SMA.

•CLOSE. This state is entered in different situations for the session manager on the client side or on the server side.

a) CS SMA
1.If the client closes the TCP connection.
2.If a close PDU is received from the SS SMA.
3.If a time out takes place for a session in the SAVING state.
4.If the CS SMA fails during the ESTABLISH state.

b) SS SMA

1.If a close PDU is received from the CS SMA.

2.If the server or the next SMA in the chain towards the original server closes the corresponding TCP connection.

3.If the session times out during the SAVING state.

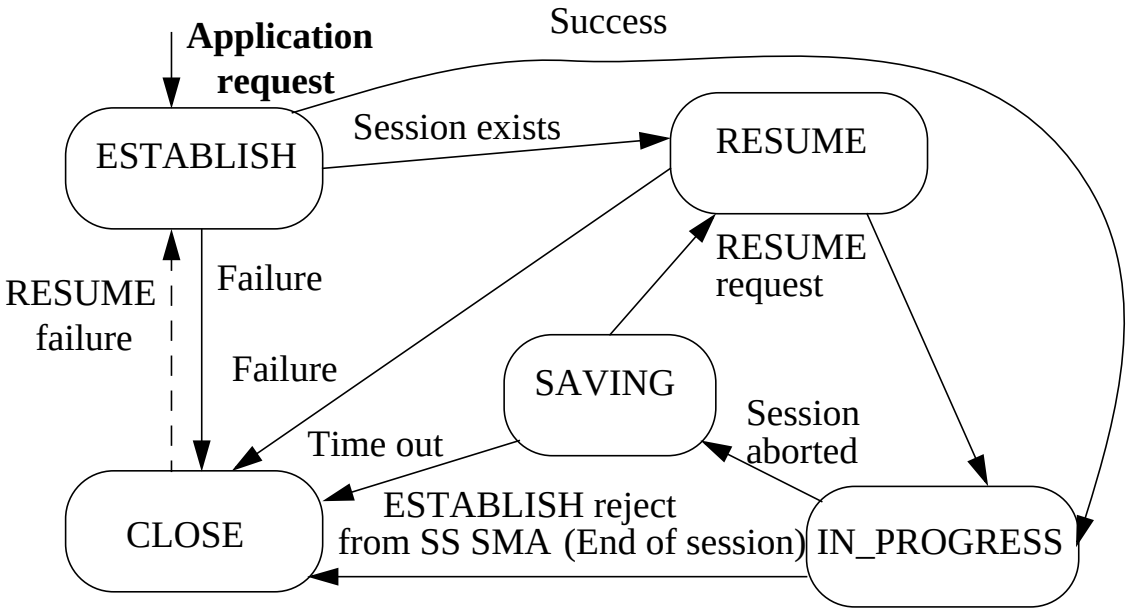


Figure 6. *CS SMA state machine.*

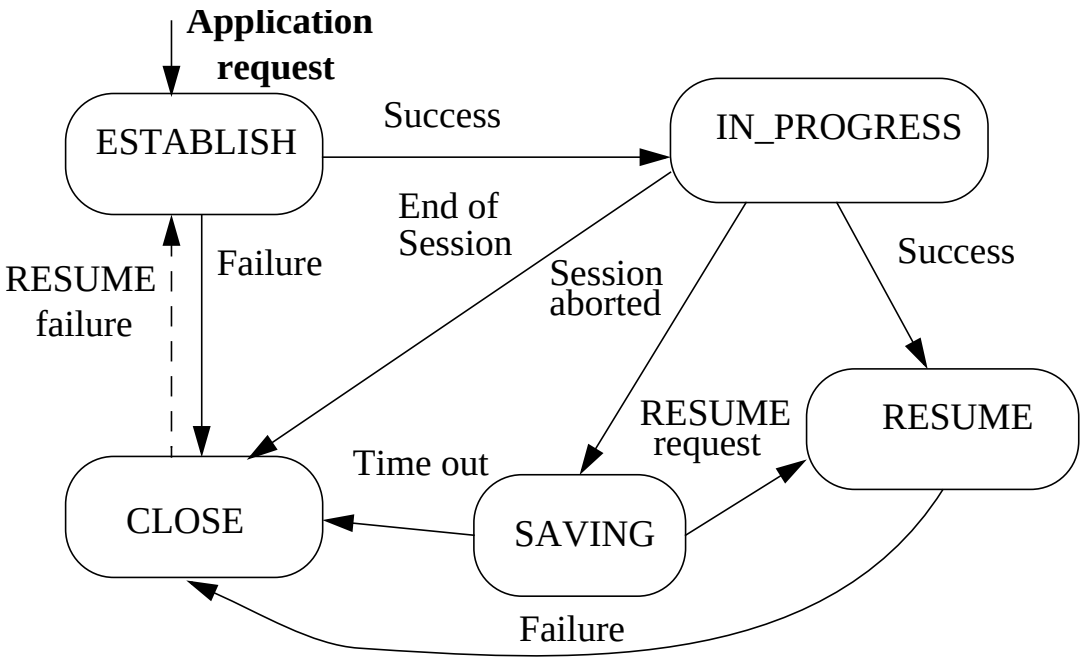


Figure 7. *SS SMA state machine.*

2.4.1. Session Establishment.

2.4.1.1. CS SMA . The CS SMA enters the ESTABLISH state when a new TCP connection is requested by the client's application. In this state, the CS SMA adds the new session to the list of active sessions maintained and waits for data coming from the application.

The client's application has to send protocol specific information of the session being established to the CS SMA. In case the CS SMA does not receive this information, it considers the session a failure and passes on to the CLOSE state without having to notify SS SMA in anyway. If the protocol specific information is received, the CS SMA looks through the list of saved states to see if the data requested is in the list. If the data is in the list, the CSMA moves to the RESUME session state. If the requested data is not found in the list of saved states, the CS SMA using the protocol specific information fills in all the required fields in the data structures, moves to the IN_PROGRESS state and sends the request unchanged to the SS SMA.

The CS SMA does not have to wait for any positive acknowledgment from the SS SMA to go to the ESTABLISH state. In case the establishment is refused by the SS SMA, it sends a negative acknowledgment to the CS SMA making it move to the CLOSE state and gracefully closes the corresponding TCP connection to the client. The shutdown of the connection can be led by a message to the user.

2.4.1.2. SS SMA . When the SS SMA receives a request from the CS SMA, it allocates all the necessary data structures and fills in all the information in the different fields of these data structures. Immediately afterwards, it tries to establish a TCP connection to the original server requested by the client. If it succeeds in establishing the connection, the request is passed on to this server and the SS SMA enters the IN_PROGRESS state. Otherwise, it sends a negative ESTABLISH acknowledgment to the CS SMA.

2.4.2. In Progress State.

2.4.2.1. CS SMA . While the CS SMA stays in this state, data flows are allowed between the client and the original server through both SMAs as long as neither CS SMA nor SS SMA close the corresponding TCP connection, or the link between the SMAs becomes unavailable.

If the client terminates a session, the CS SMA moves to the CLOSE state. In case the session is interrupted accidentally, the CS SMA moves to the SAVING state. Discovery of the availability or not of the link between the SMAs is not discussed here. How the CS SMA handles the TCP connections to the client when entering this state is discussed in a later section.

2.4.2.2. SS SMA . While the SS SMA stays in this state, data flows are allowed between the client and the original server through both SMAs as long as neither CS SMA nor SS SMA close the corresponding TCP connection, or the link between the SMAs becomes unavailable.

If the client or the server terminates the session, the SS SMA moves to the CLOSE state. If the session is interrupted accidentally, the SS SMA moves to the SAVING state.

2.4.3. Saving Session State.

2.4.3.1. CS SMA . Being in the SAVING state, the CS SMA is not able to figure out the reasons of the link failure. There are many situations that can lead to different periods of disconnection:

- Changing the network adapter and connecting to the network again can be done in relatively in short time periods and local TCP connections can be kept as long the client application can tolerate the delay and does not time out itself.

- Batteries in the mobile terminal have to be changed. In this case the mobile terminal has to be turned off, which implies that all TCP connections between the client and the CS SMA will be dropped.

- Wireless network reachability. The disconnection time in this case is totally unpredictable and depends on many factors like the geographical zone where the user is located.

Situations like the previously described show that it is important to implement a time-out mechanism while entering this state and, even a direct interaction with the user so that he can decide how long he wishes to wait to re-establish the communication. In this line, a good compromise solution would be to use a time out value directly defined by the user.

If the timer expires, the user receives a notification saying that all the TCP connections will be closed. This does not imply that all the data received so far will be lost, because all the data received before the TCP connection termination is saved in memory, in an external storage device like a disk, or a session file. If the application wants to resume the data transfer, the saved data is delivered immediately from the storage device to the CS SMA and a resume request is sent to the SS SMA to complete the transfer.

After a close notification, the user decides whether he wants to close the local TCP connections or not. If the user wishes to continue with them, the time out timer will be restarted and the user will get a notification message again. If during this procedure, the application times out, the CS SMA moves to the CLOSE state. It is responsibility of the CS SMA to check the time-outs of the saved sessions at start up time. If there are sessions that have expired, the CS SMA should request permission to remove them. The CS SMA moves to the RESUME state in two different conditions:

1.When the CS SMA finds out that the communications link to the SS SMA has become available after the interruption.

2.When it receives a protocol specific message while in the ESTABLISH state and finds a saved session for the requested object.

2.4.3.2. SS SMA . The SS SMA presents a different behaviour as compared to that shown by the CS SMA. After entering the SAVING state, the SS SMA keeps up the communication with the original server for the requests still not served, because the original server is not aware of any link failure between the CS SMA and the SS SMA and does not have any idea about the SeMS service in the communication, so it believes that is communicating directly with the client.

This is the main reason why the communication between the original server and the SS SMA must be kept up while the SS SMA has requests to be served, despite the fact that the SS SMA can not pass the data on to the CS SMA. The data received by the SS SMA while in the SAVING state is saved using data caching techniques like the ones used by regular proxies in the Internet.

2.4.4. Close Session State.

2.4.4.1. CS SMA . The CLOSE state is entered when the clients application closes the corresponding TCP connection, by receiving a CLOSE message from the SS SMA, by receiving a RESUME reject message from the SS SMA, or after a time-out of the session while in the SAVING state. In this last situation, the CS SMA has to close, if it is necessary, the TCP connections and save the transmitted data if required.

2.4.4.2. SS SMA . The CLOSE state is entered when the server closes the corresponding TCP connection, by receiving a CLOSE message from the CS SMA, or after a time-out of the session while in the SAVING state. In this last situation, the SS SMA has to close, if it is necessary, the TCP connections and save the transmitted data if required.

2.4.5. Resume Session State.

2.4.5.1. CS SMA . The CS SMA enters this state when it discovers that the requested session is among the sessions in the SAVING state. if this is the case, the CS SMA send a RESUME request to the SS SMA and if the SS SMA finds the correspondent session among its sessions in the SAVING state, it begins the data transfer without contacting the original server. There are no explicit positive acknowledgments for the RESUME request.

If the RESUME request cannot be completed, the SS SMA sends a RESUME reject to the CS SMA. Upon receiving the RESUME reject message, the CS SMA moves to the CLOSE state. However, this does not mean that the requested session by the client will be rejected. Internally, the CS SMA moves to the ESTABLISH state in order to establish a new session to the SS SMA.

2.4.5.2. SS SMA . The SS SMA enters the RESUME state when it receives a RESUME request from the CS SMA. If the requested session is found in the list of sessions in the saved state, the SS SMA transfers the data beginning by the packet number specified by the RESUME request from the CS SMA. if not, RESUME reject is sent to the CS SMA, and the CS SMA proceeds to establish a new session.

2.5. SeMS Location and Handover

Handover or handoff as it is called in the US is defined as the period and actions required for a mobile to move from one network to another[6,8,9]. In our case, depending on how we place the SS SMAs, handovers occur or not. Here we are going to see two different cases, one in which handover is not needed, and another one in which it is needed. In this last case, we are going to see how handovers are supported in the SeMS.

2.5.1. One SS SMA per group of ISPs.

In this scenario, there is one SS SMA placed at a particular network in the Internet taking care of the server part of the SeMS as we can see in Figure 8. Regardless of the ISP used by the mobile user, if the user wants to make use of the SeMS services, all the data has to be routed via that SS SMA. If the user decides to switch to another ISP, the traffic will yet have to be routed via the same SS SMA and no handover scheme is needed.

The advantage of this approach is that no handovers are required, even in the case in which the user decides to switch to a different ISP. The disadvantages are the following:

- Scalability problems. Due to the fact that one SS SMA has to serve all the clients willing to use the SeMS service.

- Triangular routing. All the packets have to be routed via the network where the SS SMA is located.

Many ridiculous situations may result from this. For example, let's suppose we are sitting in an office in Stockholm and we want to fetch a web page in our department. If the SS SMA is located in the US, all the data up and down the communications path would have to go via the US. This is not reasonable and would add extra latency to the communication.

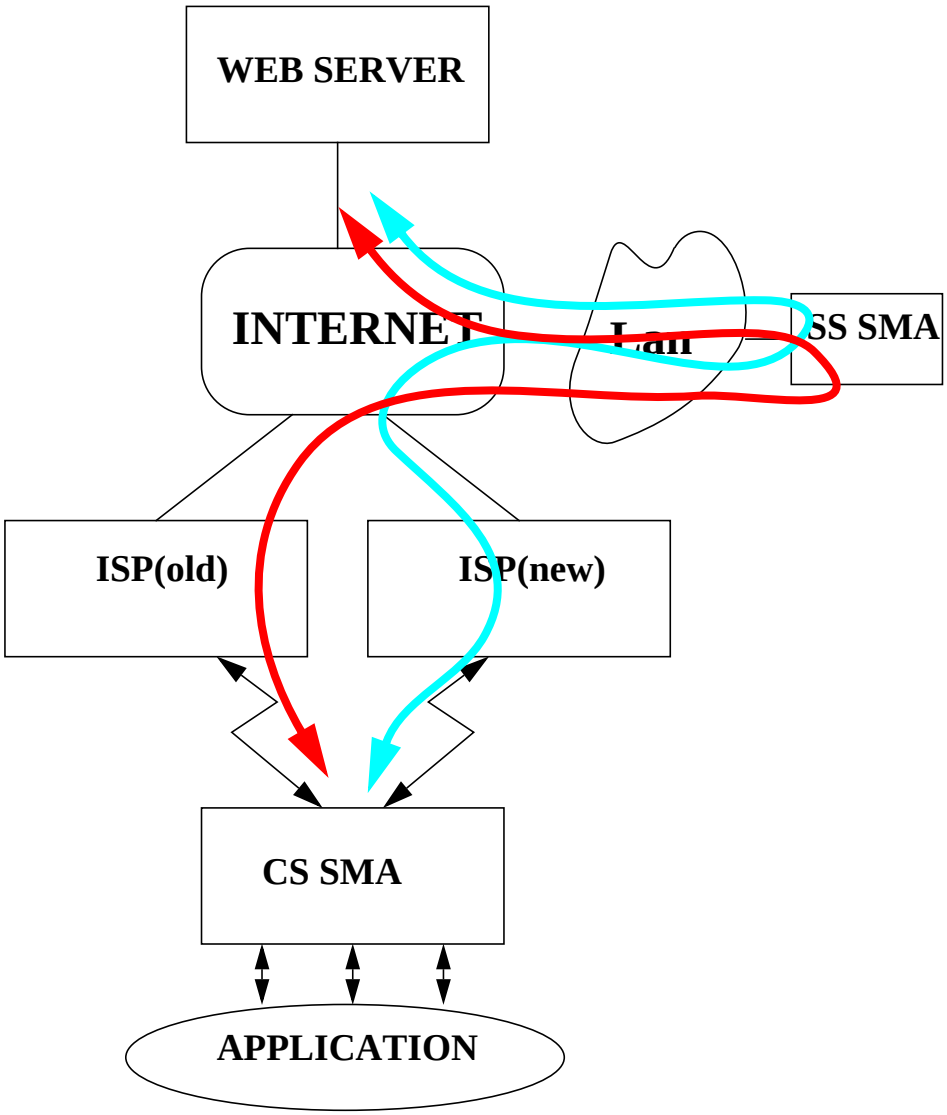


Figure 8. *Scenario with one SS SMA pergroup of ISPs.*

2.5.2. One SS SMA per ISP.

There is another approach in terms of the location of the SS SMA, see Figure 9. In this case, a SS SMA is placed in each ISP, and handovers are necessary when changing ISPs because saved data have to be transferred from the old SS SMA to the new one. This is the major disadvantage in this case. The advantage is that we avoid the triangular routing of packets of in the previous approach, except when transferring data from the old to the new SS SMA.

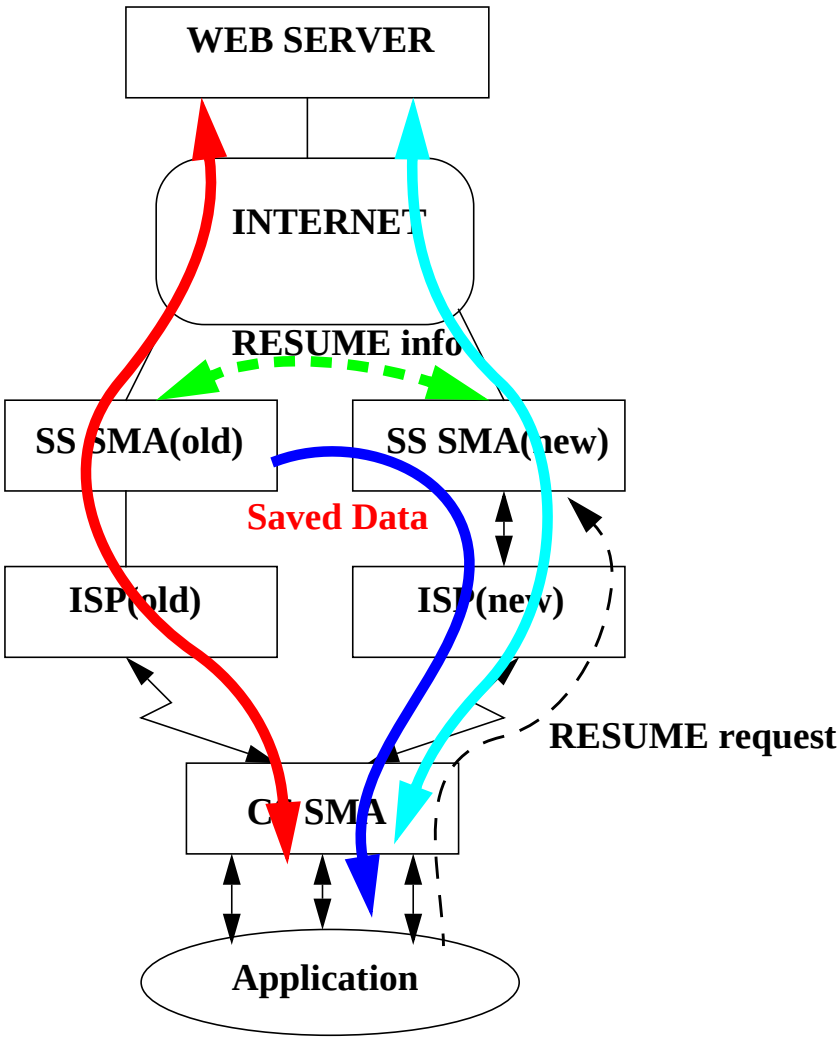


Figure 9. *Scenario with one SS SMA per ISP.*

During the data transfer, the user of the application may consider that there is an ISP that offers a service better than the one provided by the current ISP. In that case, with SeMS being supported in every ISP by a SS SMA, the user connects to the new ISP who will process the requests from the client's applications from that moment on. In the way the SMP is designed, the old SS SMA continues to receive data from the server corresponding to all the TCP requests from the client received before switching to the new ISP.

This information is catalogued with the saving tag and stored for a certain time in some storage place using current caching techniques of proxies in the Internet. A time out can be implemented to check for how long the information should be saved. Different criteria could be used to decide the time out, e.g. the type of service the user subscribes to.

This saved data is a part of the data being transferred and it has to be passed up to the client's application. In order to do so, the new SS SMA contacts the old SS SMA and they establish a dialogue to transfer the saved data from the old SS SMA to the new SS SMA. Finally, the new SS SMA takes care of relaying the information on to the client application. To launch this process the client has to issue a special resume request to the new SS SMA.

The RESUME request includes the following information:

- The IP address of the old SS SMA, so that the new SS SMA can contact the old one.

- Old session ID. The old client number and the TCP connection number.

- New session ID. The new client number and TCP connection number.

In the dialogue between the SS SMAs, the new SS SMA tells the old one that the client wants to resume communications for a certain TCP connection, marked with a certain old session ID in the old SS SMA. The old SS SMA transfers the data corresponding to that TCP connection down to the new SS SMA.

Whenever the new SS SMA receives this data, it maps the session in the old SS SMA to the session in the new SS SMA for that same TCP connection, using the old and new session IDs. Finally the new SS SMA passes the data on to the CS SMA, who takes care of delivering it to the application.

Some of this information like the TCP connection number is redundant since it is assigned by the CS SMA and does not change when changing the network access. In order to increase the sense of mobility when roaming around, this handover should be done as fast as possible[6,9]. Nevertheless, there is a problem in this approach, there is triangular routing of part of the transferred data. This is, however, a common problem in the majority of the handover schemes and it is difficult to avoid.

3. TCP Multiplexing

It is an existing idea that fits specially well in SeMS because it returns a pair of values that can be easily used to provide session identification and classification(session ID in SeMS). It also introduces performance improvements that can be very valuable in mobile environments.

All the steps forward in order to improve TCP performance on low speed wireless links, can be shortened taking in consideration that many applications are not well suited to work over them[1,5,7]. Parallel research has to be conducted in TCP and applications performance over wireless links to get to solutions that offer a better service to users.

The idea of TCP multiplexing is enclosed in this framework, and was raised to try to overcome, to a certain degree, the lack of understanding of applications about the underlying transport protocol TCP in wireless links.

The results show that there are in fact gains of performance in the service when using TCP multiplexing of between 20 to 50% in comparison to the normal TCP implementation[22]. Performance gains are not the only positive aspects of using this architecture. TCP multiplexing allows the mobile terminal to assume ordinary TCP/IP protocol stack, making most of the mobility and security solutions implemented in TCP/IP suitable to be used also in this architecture.

3.1. Performance characteristics over low-speed links

Before getting into the architecture of TCP multiplexing, we are going to study the impact of simultaneous TCP connections, to and from the same host for data transfers in communication paths that have different capacities and that include a low speed link.

In low speed links, the distribution at the outgoing buffer of data packets belonging to multiple connections is bursty. This causes the RTTs for some data packets to vary in a large way[7,22], generating TCP time-outs in the sender which will lead to retransmissions. This is a burden on links offering high latency and high BER, which require an usage of links as optimal as possible.

3.1.1. TCP Behaviour in Bulk Data Transfers.

In bulk data transfers over wireless links, TCP shows an undesirable behaviour at the start-up time of the data transfer[7,9]. This is due to the so called initial retransmission syndrome (IRS). The observations consist of a data transfer of 256 kilobytes using FTP via a GSM connection to the Internet. The MSS is 1500 bytes and the initial round trip time (IRTT) is set to 200 milliseconds.

TCP has an undesirable behaviour at start-up due to the initial retransmission time-out (IRTO) value of some servers used during the experiments. The small IRTO leads to multiple retransmissions of packets during the time required for the acknowledgments to return. If the number of retransmissions is greater than three and all the packets and corresponding acknowledgments reach their destination, then the sender receives multiple acknowledgments with the same sequence number and assumes that all succeeding packets are lost. This causes even more retransmissions and further repeated acknowledgments.

When RTO < RTT, packets are unnecessarily retransmitted (because RTO will expire before an ACK for the packet can arrive) and the connection enters slow start and congestion avoidance. Generally, the algorithms for computing RTO avoid this problem by adding a positive term to the estimated RTT. However, when we have a new connection, it must use some estimate for RTO, and if it picks a value less than RTT, the above discussed problems will come up. Furthermore, when the initial RTO < RTT, it can take a long time or the TCP to correct the problem by adapting the RTT estimate, because the use of Karn’s algorithm (RFC 1122,(4.2.3.1)) will discard many of the candidate RTT measurements made after the first time-out, since they will be measurements of retransmitted segments. In Figure 10, we can see the average number of retransmissions per packet when sending 50 packets.

As we can see in this figure, when the IRTO is very small, about 200 milliseconds, the first forty packets are retransmitted, in some cases reaching up to 5 to 7 retransmissions. In similar observations, it was also found that when the IRTO has a value of between 1 to 3 seconds, more in the range of the RFC 1122 recommendation, there were only retransmissions for the first two or three packets. This is because in this case, the TCP timers adjust much faster to the transmission through the slow link.

In any case, a simple increase in the IRTO is not the solution because it may lead to bad consequences in high speed networks. This is a matter of study and several options to improve this situation could be studied, like for example a previous negotiation of this parameter when establishing the TCP connection.

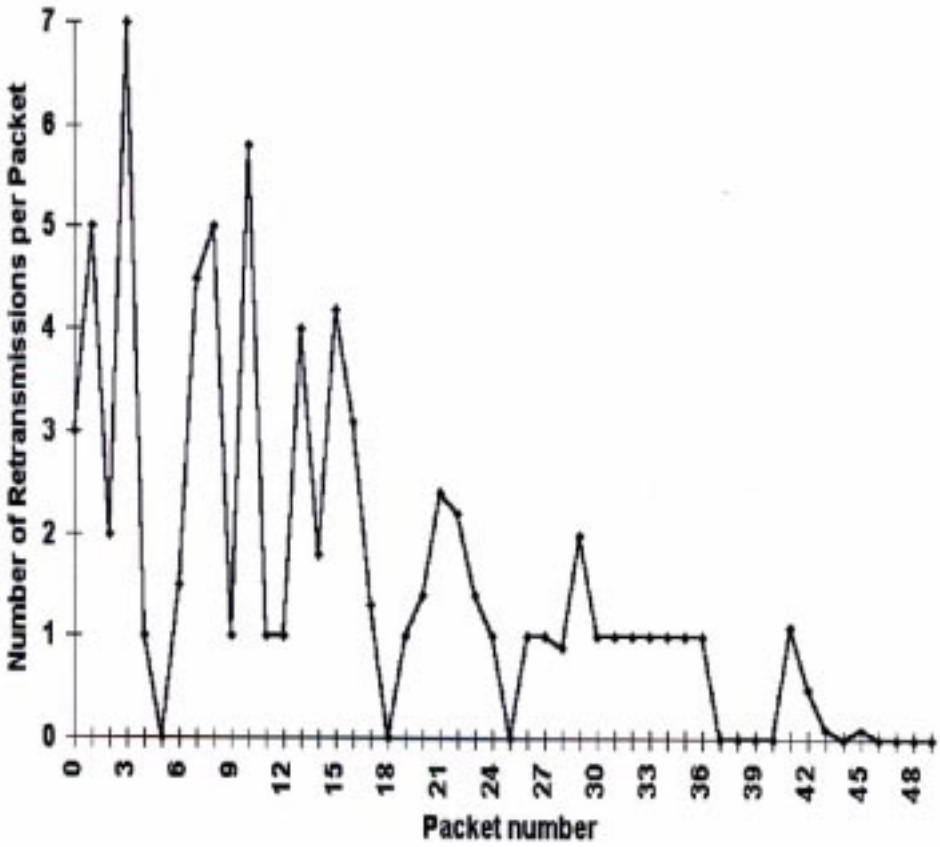


Figure 10.

3.1.2. TCP Behaviour for Multiple Concurrent TCP Connections.

Now we have an scenario in which there are multiple concurrent TCP connections, to and from the same host, through a heterogeneous communications path including a low speed link. In this case, TCP shows an undesirable behaviour even though the IRTO was inside the range recommended by RFC 1122, that is 3 seconds. The experiments were done using a GSM connection. The application used was a Netscape browser (version 4.04), and it used four simultaneous TCP connections.

There is an improper behaviour of the system because of the long packet delays caused by burstiness at the outgoing buffer to the link. See Figure 10. The interleaving of the packets from the different simultaneous TCP connections at the outgoing buffer to the low speed link causes long RTT variations. Another factor that introduces RTT variation are retransmissions at the radio link layer. For example, the Radio Link Protocol (RLP) in GSM provides high reliability through forward error correction along with retransmission of corrupted segments over the air interface.

This kind of retransmission at the link level is a possible source of bad behaviour due to the fact that it can cause time-outs that lead to TCP retransmissions. However, on low speed links TCP and RLP have time-outs in different time scales. RLP does not retransmit the whole TCP/IP segment, but frames of 240 bytes. Sending these frames over low speed links does not take a long time. The retransmission time-out (RTO) of TCP depends on the RTT which grows proportionally with the congestion window size. Since RLP provides a reliable delivery of data reducing packet losses. the RTT variations introduced by RLP retransmissions are not significant TCP’s calculation of RTO.

Hence, retransmissions at the link layer have a lower impact on TCP performance than the RTT variations caused by the interleaving of packets at the outgoing buffer to the slow link. At slow start, starting with a relatively small IRTT, long packet delays and high RTT variations causes performance degradation due to TCP retransmissions. As the RTT increases, the TCP parameters adapt better to the available bandwidth and throughput converges to a steady state.

3.2. TCP Multiplexer Design

The implementation of the TCP Multiplexer was done by implementing a client side proxy (CSP) running on a mobile computer and a server side proxy (SSP) running at the other side of the low speed link. While the CSP is recommended to run in the mobile device there are no special constraints about where in the wired part of the communications path the SSP has to be located.

In order to be able to support the TCP multiplexer, it is necessary that the application is capable of supporting proxy communication, e.g. a Netscape browser. When this is the case, the application is configured to communicate with the CSP using the loopback interface of the mobile device (127.0.0.1).

The process starts with the CSP trying to establish a connection to the SSP, see Figure 11. Once the connection is established, it is persistent and only disappears if the TCP connection between the two proxies terminates. The user of the mobile terminal has total control of this connection and can terminate and re-establish it at will, using for example a graphical user interface.

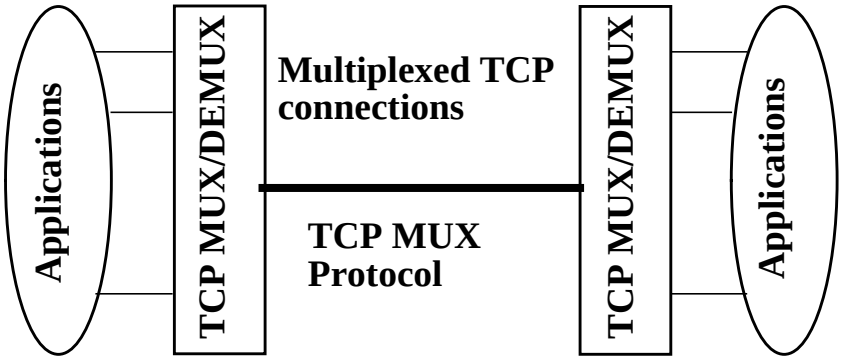


Figure 11. *TCP multiplexer architecture.*

The SSP is capable of supporting an arbitrary number of clients simultaneously. Hence, it is necessary to identify each of the clients with a certain number so that the SSP knows to whom the data received from the server belongs. This client number identifies the single TCP(STCP) connection to the CSP, see Figure 11.

The STCP connection is used by the mobile client to perform all subsequent communications through the low speed link. The client number is assigned by the SSP during the initial handshake.

3.2.1. Handshake sequence.

1. The CSP tries to establish a connection to the SSP by sending the message "Hello Proxy".

2. The SSP decides whether to accepts the connection or not.

    •If the SSP accepts the connection, it assigns a client number to the connection. This number is the first unused number in the set 1..MAXINT. Number 0 is set for a client of special use. MAXINT sets the limit in terms of the number of connections that the SSP can support simultaneously.

    The SSP maintains the client number by introducing the connection in a list of active connections and allocates the structures necessary for the communication with the client. Finally, the SSP sends a message back to the CSP saying "Proxy OK" along with the client number associated with the connection. The CSP is not allowed to change the number during the STCP connection's lifetime. The only entity capable of doing it is the SSP and only when the STCP terminates.

    In this case, the SSP will tear down the connection that it holds and deallocate all the corresponding resources. If the CSP requests a new connection, it will follow the handshake procedure from the very beginning, getting another client number for its new connection to the SSP.

    •If the SSP rejects the connection it simply sends a message back to the CSP saying "Proxy No".

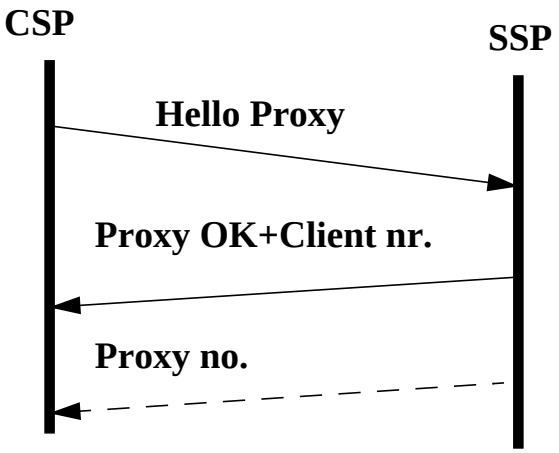


Figure 12. *Handshake sequence.*

3.2.2. TCP MUX architecture.

Once the single TCP connection is established the CSP is ready to receive requests for service from the mobile terminal applications. The multiplexing protocol is used to map uniquely the TCP connections, both at the CSP and the SSP.

When the application establishes a TCP connection with the CSP, the CSP allocates a TCP connection number, again between 1 and MAXINT, to identify each particular TCP connection. In this case, it is the CSP that assigns and maintains the connection numbers for the application. Each session is identified by the client number and the TCP connection number.

Once the TCP connection with the application has been classified by the pair {client number, TCP connection number}, the CSP fills in the header, see Figure 13, appends the data received in that connection to the header and sends it to the SSP. The data along with the header comprise the protocol data unit (PDU) of the multiplexer protocol.

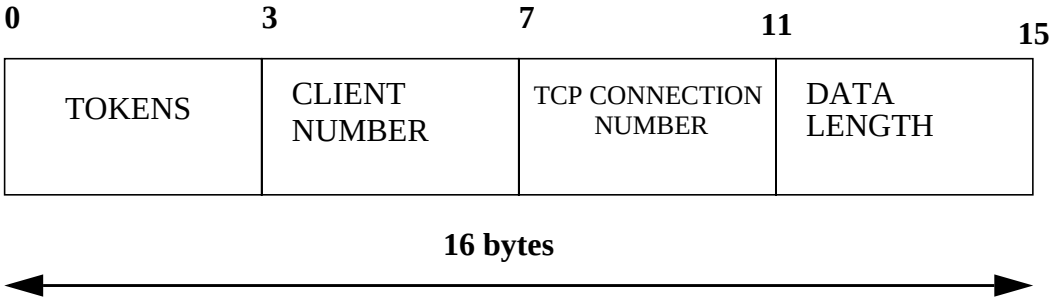


Figure 13. *TCP mux header.*

When the SSP receives the PDU, it first reads the client number to identify to whom the connection belongs, then it reads the TCP number. This number along with the client number allows the SSP to map the different connections it establishes with the server. This makes it easy to conclude that, for a certain client, the SSP allocates a number of connections to the server that is equal to the number of connections the application establishes with the CSP.

When the SSP receives data from the server, it first identifies the connection through which the data was received and then finds out the client to whom it corresponds. It creates the PDU by filling in the header and by appending the data received from the server. Finally, it relays the PDU on to the CSP using the STCP connection identified by the client number.

The procedure is very similar when the CSP receives information through the STCP connection to the SSP. It first reads the TCP connection number from the header of the PDU and then checks the amount of data corresponding to that particular data unit. Then, the process accesses the data structure where the information about the different TCP connections with the application is held. This allows the process to identify the socket to which that TCP connection is associated, and to send the information through that socket.

3.2.3. TCP MUX implementation.

The implementation was done in a regular Hewlett-Packard Vectra PC with 166 MHz using Red Hat Linux 5.2. It comprises two major processes, one representing the CSP and the other one representing the SSP. A multi-threaded approach was adopted. Different Posix threads take care of the different simultaneous functions supported by each one of the agents, see Figure14.

The threads interact among themselves. They communicate using the information stored in the data structures. The data structures constitute shared memory since they are common to all threads. Thus, mechanisms of synchronization have to be used when accessing these data structures. In this case, a semaphore is used. Without the synchronization undesirable results come up, yielding in a useless multiplexer.

To make a good use of the CPU resources. polling is avoided. This is achieved by using condition signals that put a process to sleep until a certain condition happens, therefore avoiding the unnecessarily bad use of available computer resources. The sleeping process wakes up only when it is signalled that the condition has occurred. In Figure 14, we show a simple scheme of how the program works. These are the basic functionalities of each thread in the program:

A->Thread that listens to connections from the clients.

B->Thread that listens to connections from the applications.

C->Thread that takes care of a certain TCP connection (application side) at the CS SMA.

D->Thread that sends the PDUs through the STCP.

E->Thread that reads the PDUs from the STCP.

F->Thread that takes care of a certain TCP connection (server side) at the SS SMA.

? -> The shared data structures.

Mux GSM -
No Mux GSM --
Mux 33.6 kbps modem -
No Mux 33.6 kbps modem --

Figure 15. *Mux vs No Mux*

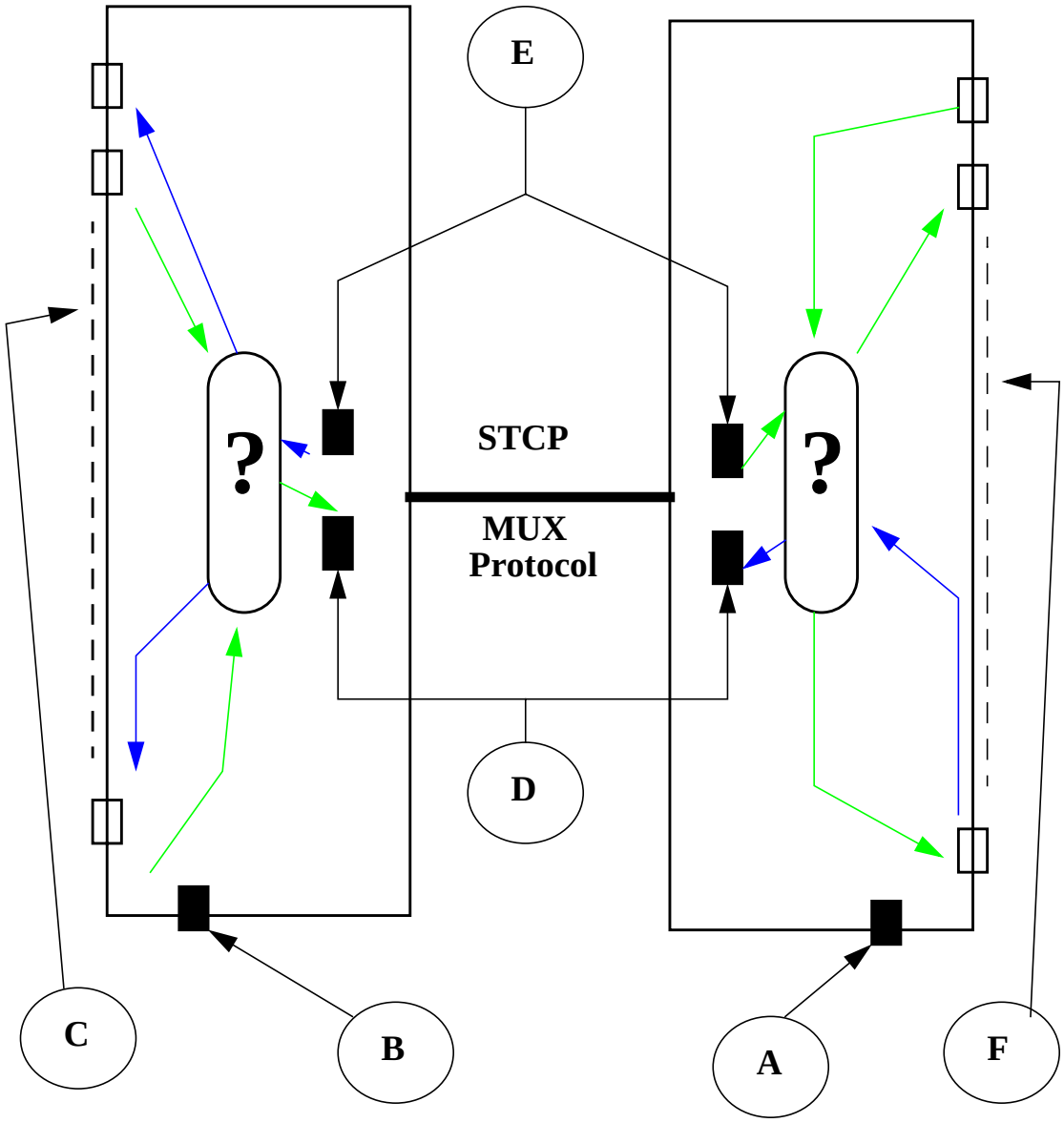


Figure 14.

The only threads that are running all the time without dying, are the ones that take care of reading and writing from/to the STCP connection, and the ones that are listening for connections from the application and the CSPs.

The other threads, the ones that take care of the different TCP connections both in the SSP and the CSP are alive only while there is something to do for the TCP connection they represent, otherwise they die. There one exception, and that is the case in which the application decides to reuse a connection like many browsers already do. In this case, the CSP does not let the thread corresponding to that connection die, the corresponding threads in the SSP follow the normal behaviour.

3.3. Improvements Introduced by TCP Multiplexing

The introduction of the TCP multiplexing brings along performance improvements for low speed links:

•The use of TCP multiplexing eliminates the 3 way handshake over the low speed link that is normally performed whenever a new TCP connection is created.

•The single TCP connection between the CSP and the SSP is persistent during the whole session. It stays up as long as the link between the CSP and the SSP does not go down. Thus, slow start is avoided for every new TCP connection.

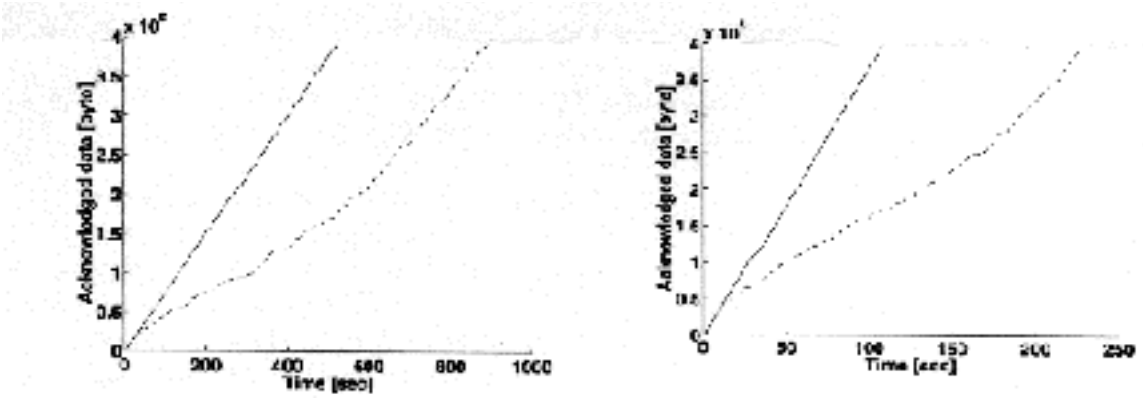
•When there are several TCP connections transmitting data simultaneously in one direction, the queuing times in the outgoing buffer to the slow link are dependent on the interleaving of segments from the different connections. Thus, the interleaving of packets affects the calculations of the RTT and consequently the RTO.

The TCP multiplexer architecture causes a reduction in the interleaving of packets from the different TCP connections and therefore, there is a reduction in the variation of the calculated RTT and RTO. In a previous section we said that high variations in the calculated RTT led to retransmissions due to time-outs. While using the TCP multiplexer it appeared to be free of retransmissions of this kind.

•By working with the TCP/IP protocol stack we avoid problems of compatibility. Many solutions propose split-up connections with special stacks taking care of the wireless part of the link. This is fine when the mobile devices communicate always through an intermediate agent like a base station in wireless links. But what happens with devices that can get access through both dial up and wired connections. A simple example is a laptop. Current laptop usage patterns conclude that the devices use dial up access when out of the office, but wired LANs when inside the office, solutions compatible with TCP/IP are more suited for this environment. Since the multiplexer is based on TCP/IP, any improvement to this protocol stack will also improve the multiplexer.

•The solution offered by the multiplexer offers a lot of flexibility in placing the proxies compared to other solutions. For example, the solutions that split the connection in two, one for the wired part and one for the wireless part of the link, are forced to locate their proxies at the interface with the wireless link.

•In Figure 15, we show two examples of how the throughput in low speed links can be increased by using the TCP Multiplexing architecture. We show acknowledged data at the mobile client when downloading four objects of 100 kilobytes using GSM respectively 33.6 kilobit/s modem[22].



4. Data Transfer Security

It is well known that the TCP/IP suite does not provide any security. Many transactions are done via insecure networks, and the information exchanged can be eavesdropped, altered, and modified. Solutions are needed if the Internet wants to be used for something more than exchanging trivial and low significant information. Solutions in security will reduce the obstacles for a faster, better development and expansion of the Internet. So far, the task of providing a secure method to communicate over the Internet has been somewhat elusive.

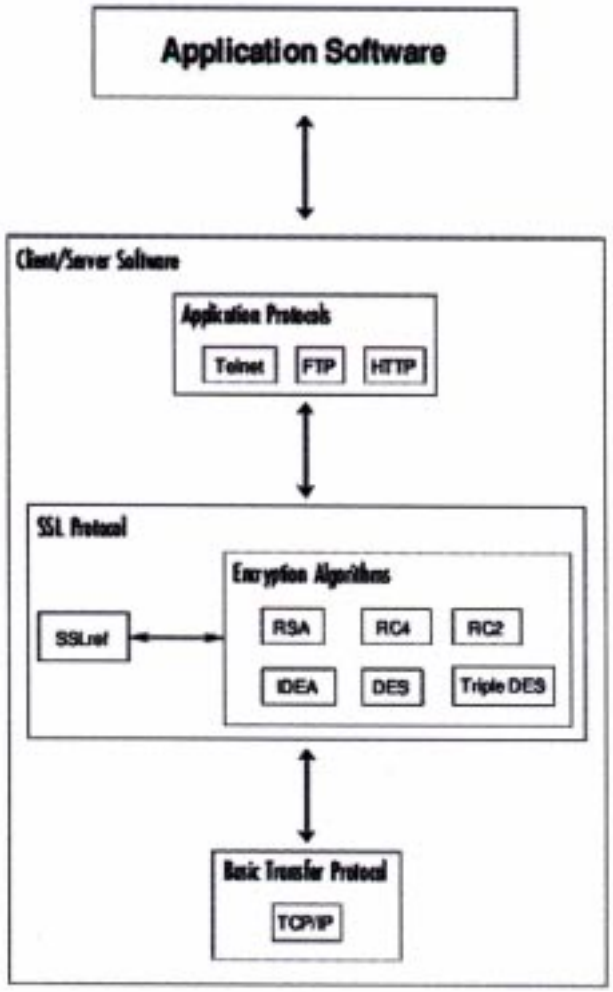


Figure 16. *The SSL Protocol*

There is not a perfect standard but several choices have been gaining popularity lately, like the Secure Socket Layer (SSL)[23,24]. SSL, see Figure 16, is a protocol introduced by the Netscape corporation and it is already supported by major browsers and web servers. It lies on top of any transport protocol and it is not TCP/IP dependent. It can run application protocols such as HTTP, TELNET, and FTP.

SSL provides authentication, confidentiality, and integrity. The implementation of the SSL can be easily done within SeMS because both SSL and SeMS use the notion of a session. A session in SSL is defined as a set of parameters and data compression algorithms. As a session identifier, the SSL uses an arbitrary byte sequence which does not exceed 32 bytes and that is assigned by the server.

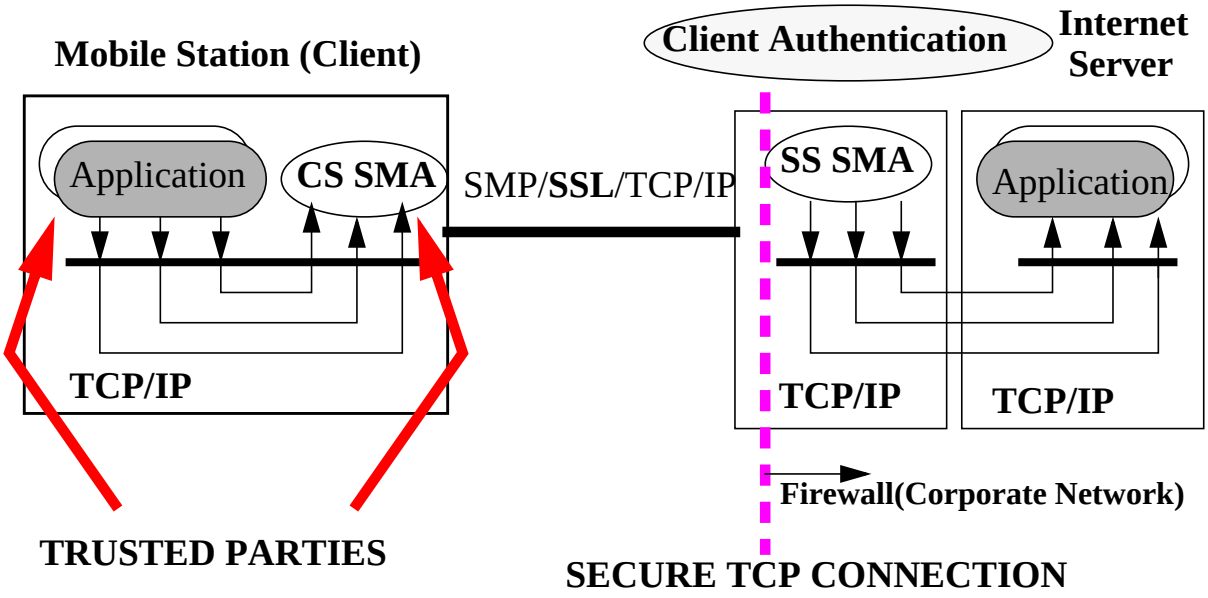


Figure 17. *Security in session management.*

We have already seen that in the SMA the session identifier is also assigned by the server side SMA during the session establishment phase. Using SSL with the TCP connection between the SMAs allows to extend the SSL session to the notion of session in the SMA, that is, the idea is more or less to map an SSL session with a communications session in SeMS. There are no restrictions on creating multiple SSL sessions into a single SMA session.

Both SMP and SSL are connection oriented protocols, so it is possible to combine the SSL's connection state with the SMA's connection establishment to form the common secure connection establishment state.

SSL has the ability to be tunnelled via proxies. This provides for secure data transfer because the SMA can act as an intermediate agent for the client[25]. It can also provide for secure transfer of data in handovers in which different SMAs exchange data.

The TCP connections between the client's application and the CS SMA are secure since both entities are trusted parties running on the same terminal or mobile station. The only thing we need to introduce security in the data transfer using SeMS is to introduce security in the single TCP connection between the two session managers, see Figure 17. It is in the STCP where the SSL session and the SeMS session are mapped. This is a very flexible approach and by providing security in the STCP connection we also provide security for all the TCP connections from the client's applications.

As we can see in Figure 17, this approach is suitable to access servers on corporate networks. The only thing we would need it is provide an authentication mechanism in the SS SMA. The data transfer would be secure up to the SS SMA, then using the authentication mechanism, the client is identified. If access to the corporate network is granted, the TCP connections between the SS SMA and the server inside the corporate network could go through the firewall and the data transfer would be secure.

4.1. Secure Socket Layer

4.1.1. Protocol Overview.

SSL is a security protocol developed by Netscape that provides communications privacy over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or forgery of messages. As we can see in Figure 18, the protocol comprises two basic layers:

1.SSL Handshake Protocol It allows client and server to authenticate each other and to negotiate an encryption algorithm with cryptographic keys before the application protocol transmits or receives its first byte of data.

2.SSL Record Protocol It lies on top of a reliable transport protocol like TCP and is used to encapsulate higher layer protocols like the SSL Handshake Protocol.

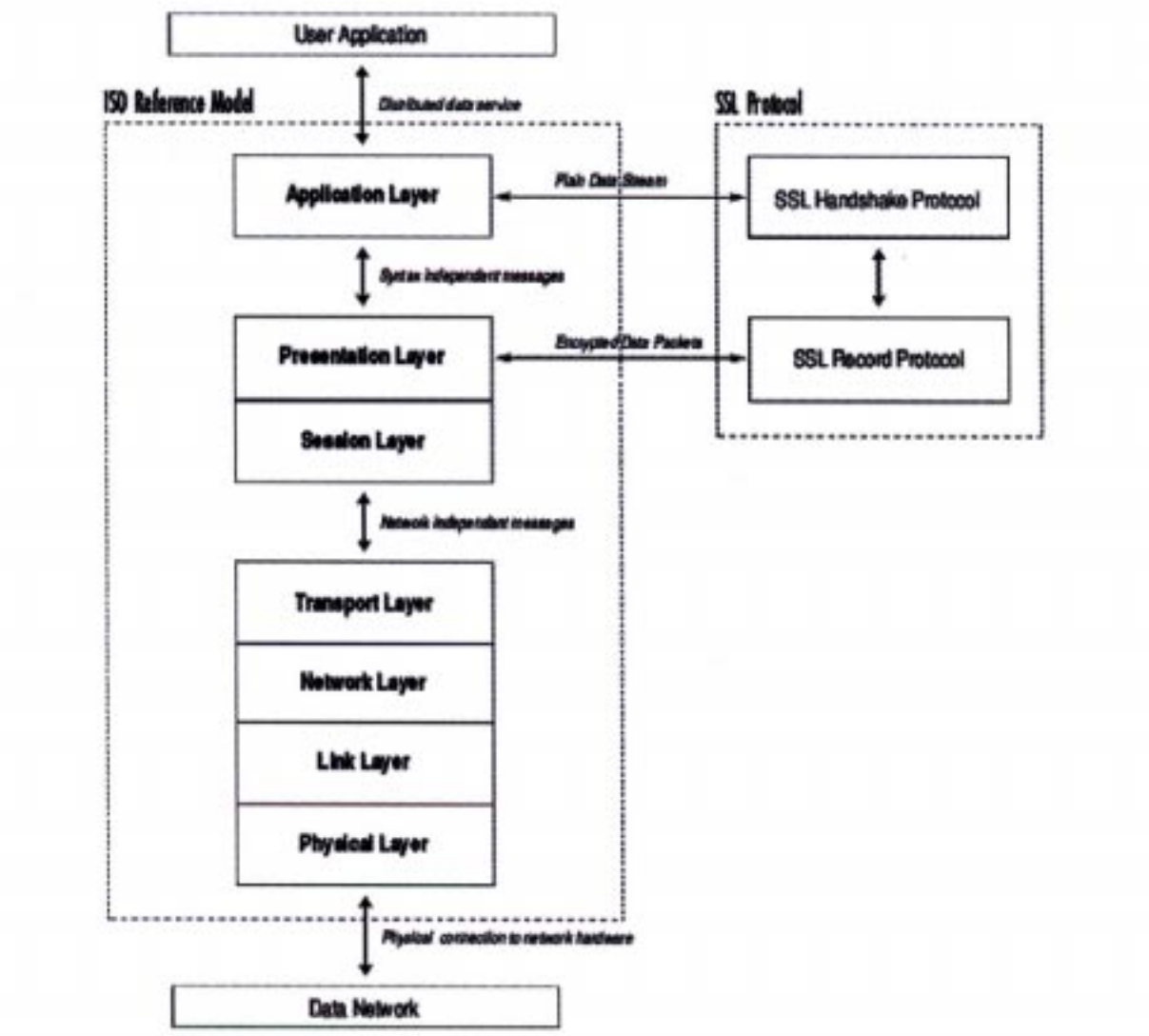


Figure 18. *The SSL Protocol*

The **basic properties** of SSL:

1.It provides private connection by the use of symmetric cryptography.

2.It provides peer authentication by the use of asymmetric cryptography or public key encryption.

3.It provides a reliable connection by the use of a message authentication code.

The **goals** that the protocol should achieve in order of priority are:

1..Cryptographic Security To provide a secure connection between 2 parties. Figure 18. The SSL Protocol.

2.Compatibility Independent programmers should be able to develop applications using SSL that successfully exchange cryptographic parameters without knowledge of each other's code.

3.Extensibility To provide a framework into which public key and bulk encryption methods can be incorporated as necessary.

4.Relative efficiency To reduce the intensity of CPU usage when doing cryptographic operations a session caching scheme.

SSL is a layered protocol that takes messages to be transmitted, fragments the data into manageable blocks, compresses it(if desired), applies a message authentication code and encrypts it. Finally, it transmits the results. A single SSL session can include multiple secure connections, and in addition, parties may have multiple simultaneous sessions.

5. Conclusions

Session management is a good and flexible approach to support and enhance data communications in mobile environments. It is a totally new approach compared to other approaches trying to provide a similar functionality, like Mobile IP. It solves many of the problems present in mobility by providing reliability, high performance, global mobility, and security for Internet access in mobile environments. It provides all of this by introducing few changes to existing architectures and applications.

SeMS is flexible because it abstracts away from address allocation schemes, network access types, and number of subscriptions to ISPs of the mobile user. It is compatible with the TCP/IP protocol stack, supporting most of the improvements introduced in this stack for end to end TCP connections. It is also compatible with other middleware solutions whose objective is to improve the performance of mobile data communications.

6. References

[1] P. Johansson, “Wireless IP Networks:Performance and Properties,” Ericsson internal document, 19 Mars 1998.

[2] UMTS Forum, “The Future Mobile Market: Global Trends and Developments with a Focus on Western Europe”. <http://www.umts-forum.org/reports.html>

[3] R. Caceres, L. Iftode, “The Effects of Mobility on Reliable Transport Protocols,” Proc. IEEE 14th International Conference on Distributed Computer Systems, Poznan, Poland, June 1994.

[4] E. Brewer, R. Katz, E. Amir, H. Balakrishnarr, Y. Chawathe, A. Fox, S. Gribble, T. Hodes, G. Nguyen, V. Pedmanabhan, M. Stemm, S. Seshan, and T. Henderson. University of California at Berkeley, “A Network Architecture for Heterogeneous Mobile Computing,” Proceedings Spring COMPCON Conference 1996.

[5] G.Montenegro, S.Dawkins, “Wireless Networking for the MNCRS,” Internet draft <draft-montenegro-mncrs-00. txt>, August 7, 1998.

[6] H. Balakrishnan, S. Seshan, and R. Katz, “Improving Reliable Transport and Handoff Per-formance in Cellular Wireless Networks,” ACM Wireless Networks, December 1995.

[7] Y. Ismailov, “Observations on TCP Behaviour in a Low Capacity Wide Area Wireless Environment,” Ericsson Report 97:203, 1997.

[8] M.Stemm and R. Katz, “Vertical Handoffs in Wireless Overlay Networks,” ACM Mobile Nertworking(MONET), Special Issue on Mobile Networking on the Internet, Summer 1998.

[9] R. Caceres and V. Pedmanabhan, “Fast and Scalable Handoffs for Wireless Internetworks,” Proc. ACM Mobicom96, Rye, NY, November 1996.

[10] E. Amir, H. Balakrishnan, and R. Katz, “Efficient TCP over Networks With Wireless Links,” Proc. Fifth IEEE Workshop of Hot Topics in Operating Systems, May 1995.

[11] M.Allman, V.Paxon, W. R. Stevens, “TCP Congestion Control,” Request for Comments 2581, April 1999.

[12] Van Jacobson, “Congestion Avoidance and Control,” Proceedings of ACM SIGCOMM, August 1988.

[13] X. Zhao, C. Castelluccia, and M. Baker, “Flexible Network Support for Mobility,” Proceedings of Mobicom’98, Dallas, Texas, October 1998.

[14] C. Perkins, “IP Mobility Support,” Request for Comments 2002, October 1996.

[15] A. Lo, V. Chandrasekaran, K. Seah, and C. Soh, “A Session Management Protocol for Mobile Computing,” Proceedings of IEEE Global Communications Conference, 8-12 Nov 1998, Sydney, Australia, pp 2592-2598.

[16] W. R. Stevens, “TCP/IP Illustrated Volume 1,” Addison-Wesley 1998.

[17] WAP Forum: “The Specifications of WAP version 1.0,” 14.3.1999 <http://www.wapforum.org/>

[18] W.R. Stevens, “Unix Network Programming,” Addison-Wesley 1998.

[19] W. Stallings. “Data and Computer Communications,” 3rd edition. Maxwell MacMillan International Editions, 1991.Pp 618-652.

[20] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, “Hypertext Transfer Protocol-- HTTP/1.1.”, IETF Request for Comments 2068.

[21] W. Stevens, “TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms”, RFC 2001, January 1997.

[22] T. Larsson, Y. Ismailov and A. Nilsson, “Performance Characteristics of Multiplexed TCP Connections,” February 1999.

[23] A. Freier, P. Karlton and P. Kocher, “The SSL Version 3.0.,” Internet draft <draft-freier-ssl-version3-02.txt>, Netscape Communications, 15.3.1998, <http://home.netscape.com/eng/ssl3/draft302.txt>

[24] Luotonen, A., "Tunneling TCP based protocols through Web proxy servers", Internet-Draft (Work In Progress) <draft-luotonen-web-proxy-tunneling-01>, August 1998.

[25] A. Luotonen, “Tunneling SSL Through a WWW Proxy,” Internet Draft. <http://search.netscape.com/newsret/std/tunneling_ssl.html>

7. Abbreviations

API	Application Programming Interface.
BER	Bit Error Rate.
CSP	Client Side Proxy.
CS SMA	Client Side Session Manager.
FTP	File Transfer Protocol.
GPRS	General Packet Radio Service.
GSM	Global System for Mobile Communications.
HTTP	Hypertext Transfer Protocol.
IP	Internet Protocol.
IRS	Initial Retransmission Syndrome.
IRTO	Initial Retransmission Time-out.
IRTT	Initial Round Trip Time.
ISP	Internet Service Provider.
LAN	Local Area Network.
MAC	Message Authentication Code.
MSS	Maximum Segment Size.
OSI	Open Systems Interconnection.
RFC	Request for Comments.
RLP	Radio Link Protocol.
SeMS	Session Management Service.
SIP	Session Initiation Protocol.
SMP	Session Management Protocol.
SPDU	Session Protocol Data Unit.
SSL	Secure Socket Layer.
SSP	Server Side Proxy.
SS SMA	Server Side Session Manager.
STCP	Single TCP Connection.
TCP	Transport Control Protocol.
WAN	Wide Area Network.
WAP	Wireless Application Protocol.
WSP	Wireless Session Protocol.