Stream Control Transmission Protocol as a Transport for SIP: a case study

Giuseppe De Marco[†], Maurizio Longo[‡], Dario De Vito
Dipartimento di Ingegneria dell'Informazione e Ingegneria Elettrica, University of Salerno
Fisciano,84084, Italy
[†‡] {gdemarco,longo}@unisa.it
and

Salvatore Loreto
CoRiTeL
Fisciano 84084, Italy
loreto@coritel.it

ABSTRACT

The dominant signalling protocol both in future wireless and wired networks will be the Session Initiation Protocol (SIP), as pointed out in the 3G IP-based mobile networks specifications, entailing a fully Internet integrated network. The use of SIP in the IP Multimedia Subsytem (IMS) of Release 5 involves the development of servers capable to handle a large number of call requests. The signaling traffic associated to such requests could explode, if an intelligent congestion control were not introduced. Stream Control Transmission Protocol (SCTP) was born to support transport of SS7 signaling messages. However, many of the SCTP features are also useful for transport of SIP messages, as: congestion control mechanism, good separation among independent messages, multihoming. Indeed, adoption of SCTP as transport of SIP signaling might prove useful in some situations where usual transport protocols, like TCP and UDP, suffer performance degradation.

In this paper, we analyse the general framework wherein SIP operates and we discuss the benefits of using SCTP as a transport for SIP, toward fair sharing of network resources. This study is carried on in the context of the implementation of an high-performance SIP Proxy Server. We also present some preliminar results of an implementation of SIP over SCTP/UDP in a real LAN environment.

Keywords: SCTP, SIP, 3GPP, transport protocols, network emulation.

1 INTRODUCTION

SIP is a modular design: it is independent from the lower-layer transport protocol used, and it can run over either reliable or unreliable message or stream transport. This does not simply mean that a SIP implementation operates in the same way whatever the transport protocol, rather it adapts itself, in particular its interface between the lower layers and the type of transport protocol, according to the chosen transport layers. The SIP protocol delivers messages reliably. This reliability can be obtained through a connected transport protocol. Originally SIP was implemented with TCP in mind, also for the affinity between SIP and HTTP (HTTP, in fact, only works on TCP). In low traffic conditions UDP becomes more advantageous than TCP and therefore has become more widely used in the commercial and open source implementation of the SIP User Agent (UA) and the Stateless Server. The design of a new transport protocol such as SCTP within the IETF, intended to transport telephony signaling and generally to transport message-based protocols, has involved the evaluation of SCTP also for SIP signaling messages. This evaluation arose because of the risk of congestion in the points of the network handling high number of signalling messages (more than 10000 per hour), e.g. among Proxy servers in the 3GPP architecture. The risk of congestion has been addressed in [3] from a signalling point of view, e.g. the Proxies acquire knowledge about a condition of "congestion-safety" through the analysis of special fields inserted in the Session Description Protocol (SDP) header; accordingly Proxies may accept or refuse the request of the sessions. As in [3] we agree that the risk of congestion depends on the transport protocol used between the EndPoints, being either Proxy server or UserAgent (UA). In this article we analyze the behaviour of SIP over the different transport protocols, focusing our attention on the case of Proxy-Proxy communication, Proxy being either a stateful or stateless server. We show that SCTP can guarantee the condition of "congestion-safety between Proxies.

This paper is structured as follows: Section 2 gives an overview of the SCTP protocol. Section 3 gives a quick evaluation of SIP over different transport protocol for UA-Proxy communication. In Section 4, we describe the scenario Proxy-Proxy and we analyse some solutions of SIP transaction mapping into SCTP streams to enhance performance. Section 8 describes some results obtained through our SIP module implementation over a Linux machine, with its kernel modified in order to support the SCTP protocol, as described in Section

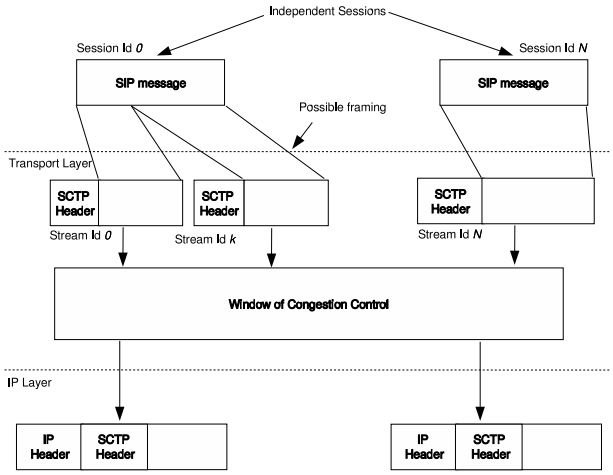

Figure 1: Description of a possible mapping of different SIP sessions into different streams. In SCTP streams are grouped and sent into a single association. By this way one realizes congestion control and suitable separation among independent messages.

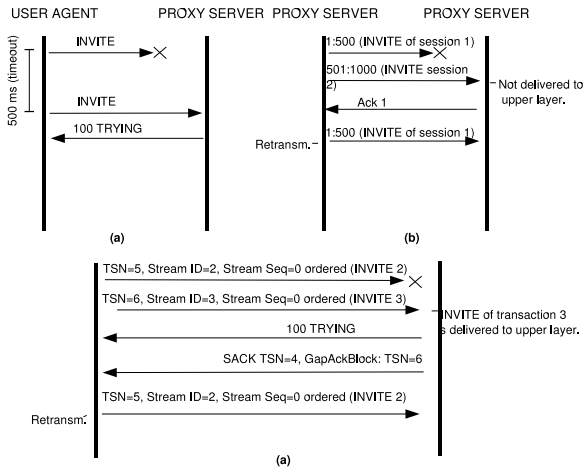

Figure 2: Example of signaling flow messages for UA-Proxy with UDP (a), Proxy–Proxy with TCP (b) and with mapping of transactions into SCTP Streams (c).

7. We suppose that the reader is familiar with the fundamental concepts of computer networks and with the basic features of the SIP protocol.

2 SCTP OVERVIEW

SCTP is the new transport protocol for signaling messages on IP networks; the specification core [8] is at the Proposed Standard level in the IP standardization process . Like TCP, SCTP offers an end-to-end, connection-oriented, reliable delivery transport service for applications communicating over an IP network. It inherits many of the functions developed for TCP, including powerful congestion control and packet loss recovery functionalities. The most significant differences regard around the support of multistreaming, partial ordering and multihoming.

When two EndPoints communicate through SCTP, an "association" is set between them, to guarantee the connection-oriented nature of the protocol. SCTP comes out with a concept of "stream", as depicted in Figure 1. A stream can be thougth of a sublayer between the transport layer and the upper layer. SCTP provides sequenced delivery within streams (partial ordering), by assuming a delivery mechanism similar to TCP in every stream. Nevertheless, an SCTP association may include several streams that are independent of each other (multistreaming). Thus, SCTP can give an inter-stream sequenceless delivery by which the throughput of transport can be improved.

Multihoming enables an SCTP host to establish an association with another SCTP host over multiple interfaces identified by separate IP addresses. The EndPoint can choose an optimal or suitable path towards multi-homed destination. This feature adds to SCTP network level a certain degree fault tolerance capability: When one of the paths fails, SCTP can still choose another path to replace the previous one.

3 USER AGENT – PROXY

This scenario usually involves small bursts of signalling information between UA and Proxy. A non-connected protocol such as UDP is to be preferred over TCP since it avoids useless delays in connection establishment, due to the aggressiveness of retransmission management entrusted to the SIP protocol (RetrasmissionTimeOut for SIP is 500ms, RTO for TCP is 1500ms). However, if one compares UDP versus SCTP, the performance gain is less significant. In fact, the SCTP association establishment is faster than the TCP connection (SCTP Initiation Procedure relies on a 4-way handshake where DATA can be included into the 3^{rd} and the 4^{th} message of the sequence; TCP can start exchanging messages only after completion of a 3-way handshake). Moreover, the initial RTO of SCTP can be set up according to user needs. Therefore in this scenario it is not clear in advance which of the two, between UDP and SCTP, should be preferred. UDP is certainly faster, and the fact that it doesn't implement any congestion control mechanism is not harmful for a 3GPP network where the path between UA and Proxy is a dedicated link established during the Packet Data Protocol context activation [8]. However, SCTP has some useful features with regard to the transport of Multimedia Messages and, generally, of Instant Messaging [9], not considered here.

4 PROXY – PROXY

In the 3GPP architecture, a large amount of signaling traffic is typically generated among Proxies and especially among core network Proxies. In fact, each Proxy handles a large number of sessions and forwards signaling traffic to other Proxies, e.g. the signaling traffic between P-CSCF and S-CSCF or between two S-CSCF of a 3GPP IP Multimedia Subsystem [4][5]. In this case we argue that UDP is definitely the worst solution among the three, while, between SCTP and TCP, the

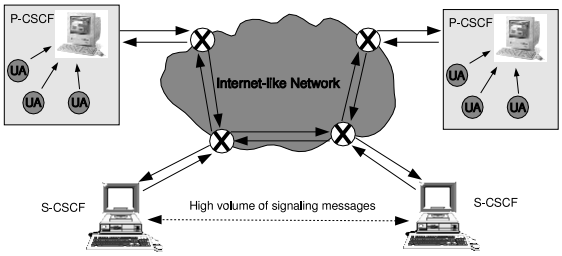

Figure 3: Simplified 3GPP architecture

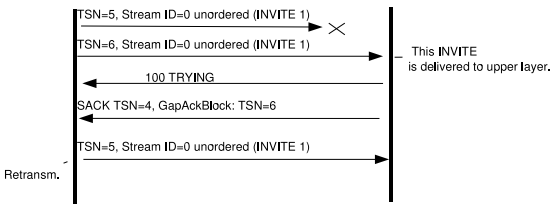

Figure 4: Example of mapping transactions into SCTP Stream 0 with flag "unordered" set

former is preferable. The advantages of SCTP over the other two protocols are outlined in the next subparagraphs.

SIP over UDP

SIP over UDP can suffer a degradation known as "retransmission storm". We explain this phenomenon by means of a typical situation arising in a 3GPP network. The analysis is based on euristics, omitting mathematical details. In Figure 3, an IP network where two SIP Proxies serve a large amount of UA terminals is depicted. One could imagine these Proxies as serving a specific geographical area. The transport protocol between the Proxies is assumed to be UDP. If traffic is generated by a moderate number of sender-receivers sessions, no significant risk of congestion arises. On the contrary, as the number of sessions increases, the SIP retransmission mechanism in conjunction with UDP is at risk of causing a flood of retransmissions. In fact, if we assume that a fraction P_0 of the total signaling traffic λ is backlogged due to the SIP timer expiration, at the next retransmission time the situation makes worse, as the traffic is now $\lambda(1 + P_0)$. The signaling traffic increases even more if a request has to be fragmented (according to some probabilistic model) into several packets. As in draft [3], we agree that "*It is not unreasonable to assume that there may be tens of thousands of UAs on each side of the network*".

5 SIP OVER TCP

A reliable transport layer as TCP undertakes the task of delivering the message to the next hop, so that retransmissions are not needed at an application level. However, TCP incurs the Head of the Line (HOL) blocking problem, as shown in Figure 2-b. HOL blocking occurs when the signaling associated with multiple sessions is sent over a single TCP connection between two servers. By using TCP, the loss of one message stops the immediate delivery to the SIP layer of further messages arriving at the Proxy. Therefore all other messages in the same flow, even if they belong to unrelated sessions, are affected by the loss of a message in a single session.

6 SIP OVER SCTP

From the previous analyses, it is clear that SCTP can overcome the problems of congestion and HOL. Now we analyze procedures for transport of SIP messages over SCTP. Some of these procedures have already been described in [4]. There are three possible ways to use SCTP with SIP: 1) mapping of SIP sessions into Streams; 2) mapping of SIP transactions into Streams; 3) using Stream 0 and the unordered flag.

1. **Mapping of whole SIP sessions into streams**. This procedure is possible only if the involved Proxies are Call-Stateful Proxies. (A Proxy is said "call-stateful" if it retains the call-state for a dialog, from the initiating INVITE to the terminating BYE request). The advantages are the same as described in the following point, referring to the whole session and not to a single transaction. This method presents two disadvantages. First, since SIP Proxies in the net are more often transaction-stateful than call-stateful, this mapping can be used only unfrequently. Second, and more important, is a problem related to the length of the Stream Identifier field (16 bits) in the DATA chunk header. The SCTP Stream ID zero must be used to send every request and every response generated by a SIP entity, when it chooses not to use SCTP stream ID lightweight transaction identifiers. The highest stream ID $2^{16} - 1$ must not be used to send SIP traffic, because a CANCEL request must be sent over a Stream ID equal to the one on which the request to be cancelled was sent plus one. ("*Since Stream zero and Stream* 2^{16-1} *cannot be used as transaction identifiers, there are* $2^{15} - 1 = 32767$ *available Stream IDs*", [4]). This constraint on the number of SIP sessions that can be managed simultaneously might turn into a critical limitation for the supported signaling traffic, more if one considers that SIP session have a typical duration of over one minute.

2. **Mapping of SIP transactions into streams**. This method gives the chance to use a simple computational technique to identify the membership of a message to a transaction. In SIP the transaction identifier consists of the header fields: To, From, Call-ID, Cseq and topmost Via. The advantages of this mapping method is that the transport can deliver the proper message to the proper transaction state machine without parsing To, From, etc. The Stream-Id provides a transaction identifier that improves the performance of the receiving SIP server. In fact when the Server sends the first message or when it starts (depending on how one wants to configure its network), it activates an association with the other server. At the start of the association it sets the stream number to be used (typically, the maximum allowed value,

namely 32767). Nevertheless, mapping of SIP transactions into streams turns out to be a valid solution only for INVITE transactions because it includes more messages than other transactions (for instance, provisional responses and ACK messages). In fact, for transactions based on a few messages, the load of the Proxy for the creation of mapping (processing time and resources in general) might easily surpass the benefits. This issue is out of the scope of this paper and we will face it in further studies. An other issued is that method requires that one Stream ID value must be reserved to each transaction. However, this should not be an important issue, due to the moderate number of transaction per session.In summary, the mapping of transactions into streams helps in the design of the internal architecture of a SIP server, but otherwise is not beneficial to the network operations.

3. **Sending every SIP request and response via the SCTP Stream ID zero with the"unordered" flag set**. The unordered SCTP service delivers messages to the receiver as soon as they are received from the network, regardless of the order they are produced by the sender. Since a SIP client does not send a new request until the previous transaction has finished, ordinarily SIP requests arrive ordered. In only two exceptional situations, SIP clients send overlapping requests: an INVITE followed by a CANCEL and an INVITE followed by a BYE. In these cases, before sending a CANCEL (or BYE) for an INVITE, one must wait for the 100Trying for the INVITE. Thus one ensures that the CANCEL (or BYE) arrives always after the INVITE. About response messages, a provisional response arriving out of order, i.e. after the arrival of definitive response, is simply ignored at SIP level (see Figure 4).

In comparing the three methods methods above, one notes that the last one is the highest and is the only one can be used for SCTP transport to a Stateless Proxy. Whereas the other two methods requires Statefull Server at both ends. With Statefull Proxy, one could ordinarily use the first method, while adopting the second method in the special case of INVITE transactions. We note that all three methods are unaffected by HOL blocking problem as we now outline. In Figure 2-c, we assume that two ordered Streams (Stream Id=2, Stream Id=3) are used according to method one. It should be clear that a loss in one Stream does not introduce any delay on the other Stream. Similar arguments hold for other two methods. Summing up, sending every SIP message (either requests or responses) over stream zero with the unordered flag set is generally the best option, under criteria of performance and simplicity. Moreover, SIP [1] now provides a transaction identifier in the branch parameter of the Via entries. This identifier may replace at the application level the SCTP stream ID in demultiplexing of the SIP transactions[15]. This might affect the performances with respect of processing time, although we have not performed any further analysis in this respect.

7 IMPLEMENTATION OF SIP OVER SCTP

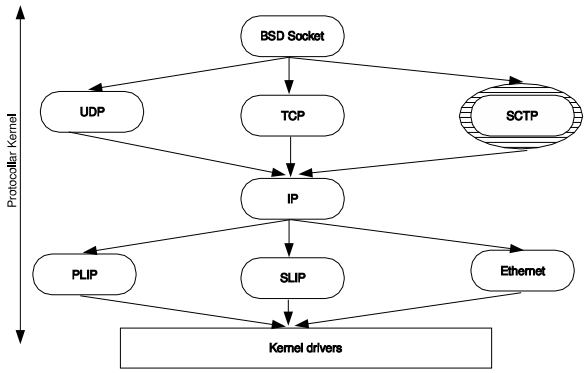

Figure 5: Structure of API layers in Linux Kernel

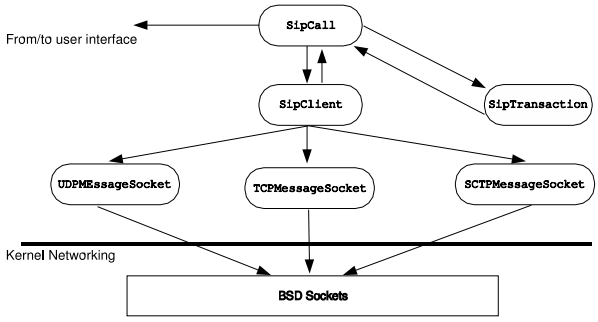

Figure 6: Macroclasses implemented in the SIP module

From previous consideration, SCTP should be preferred as the transport protocol for SIP in a Proxy-Proxy, although some increase of processing delay might be incurred due to involvement of end hosts at the application level in the SCTP operation. To investigate further on such trade-off, we have implemented a SIP module in C++ that can exploit SCTP in addition to UDP and TCP. As to the implementation of SCTP, we have followed the *lksctp* project [6], in which some of the authors of the present paper are participating as developers. This project is aimed at implementing SCTP inside the Linux kernel. Although *lksctp* project is not complete as yet, it is, to the best knowledge of the authors, the most effective current implementation of the SCTP, at least for the Linux platform in the open source world. In Figure 5 we show the networking structure of the Linux kernel as modified according to *lksctp*. The *lksctp* project implements API's that match those normally used for the IP network (BSD-Berkeley socket model), which general structure is depicted in Figure 5. This choice should render of SCTP easier for existing applications, as substantial changes are not required. The SCTP version of these API's is defined in the Internet Draft "Sockets API Extensions for SCTP" [11]. It defines two interfaces: UDP-Style Interface and TCP-Style Interface. Currently, *lksctp* only implements the UDP-Style API, as it appears more effective in supporting the innovations of SCTP. The UDP-style API's are indeed similar to those defined for the UDP protocol. The SCTP stack with UDP-style interface handles automatically both outbound association setups and shutdowns. A typical server in this model uses the following socket calls in sequence, in order

to prepare an EndPoint for servicing requests:

- `socket()`, `bind()`, `listen()`
- `recvmsg()`, `sendmsg()`
- `close()`

.The important difference between UDP and UDP-style SCTP API's are:

- **Multihoming**. An SCTP EndPoint must be associated with multiple addresses. To this aim, a new call has been introduced: `sctp_bindx()`.
- **Ancillary data**. Two new ancillary structures have been introduced ad hoc for SCTP to setting the number of streams at the beginning of the association, and to distributing the data by the chunks over the streams made available. It is also possible to set a default stream so that ancillary data may be omitted.
- **Option**. This function leads to set or to get the value of some parameters of SCTP. For example, it can modify the value of the initial RTO (`SCTP_RTOINFO` option), or it can set primary address (`SCTP_SET_PRIMARY_ADDR` option).

We have tested, according to procedures described in draft SIP-bis-09 [1] or adaptation thereof, the SIP behaviou when running over: 1) UDP; 2) TCP. In analogue fashion, to test SIP on SCTP, we have chosen to send and receive every request or response using Stream-ID zero with the"unordered" flag set. In further development of the module, implementationof Mapping of SIP transactions into streams is also be planned. The structure of macroclasses that manage the transmission and the reception of SIP messages is shown in Figure 6 . The large part of the SIP state machine is implemented in SipCall and SipTransaction. SipClient gets and sends messages from and to one and the same socket layer using functions implemented in `SCTPMessageSocket`, `TCPMessageSocket` and `UDPMessageSocket`. These three classes invoke system calls to the Linux kernel (BSD interface) that cause the actual transmission and reception of messages with the chosen transport protocol. The implemented module supports a "non standard feature", specific for our tests: a request is sent with the transport protocol chosen by the user, while a back answer is sent with the same protocol as the request which it refers to (indicated in the topmost Via header field of the request message).

8 RESULTS

The module outlined above was used to emulate the behavior of a generic SIP Proxy, by implementing an INVITE generator, which sends messages at an user selected rate. A simple picture of the configuration used for the tests is shown in Figure 7. A and B EndPoints are Linux o.s. machines, with Kernel version 2.4.17 modified to support the *lksctp* project [7]. The traffic generation software runs on C-EndPoint, aiming to congest the network.

First, a test has been performed to measure the time interval between the sending of an INVITE message and the arrival on

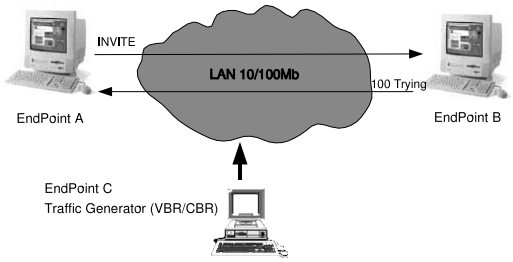

Figure 7: Test bed used to test the SIP module

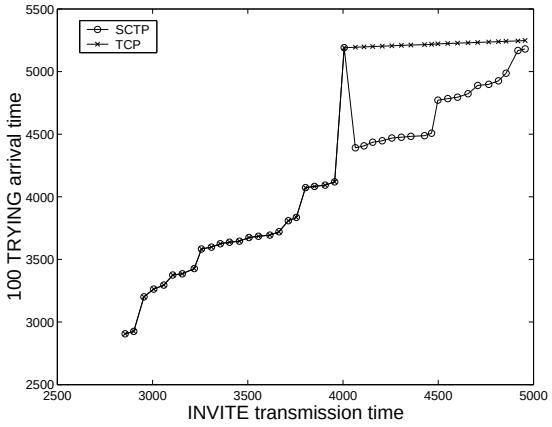

Figure 8: Comparison of SIP on both SCTP and TCP when a packet loss occurs. SCTP delivers consecutive 100 Trying even if the 5191 Trying packet has been lost.

the same computer, at the SIP layer, of the related provisional response 100Trying. The plot in Figure 8 shows that the SCTP use avoids the HOL problem: The loss of a 100Trying does not affect the ready delivery to the SIP application of the next 100Trying responses that arrive in the meantime. On the contrary, an hypothetical TCP loss in the same point causes the buffering at transport layer of messages arriving before reception of the retransmission of the lost message. Second, a test has been performed in order to verify the behavior of SIP over UDP under heavy load of the network. The plot in Figure 8 shows that, in congested situation (as in the middle zone of the plot), many useless INVITE retransmissions arise, which only increase the delays, thus creating a vicious cycle in the network. Note that in our simulation. the traffic generated by EndPoint A and B is lower than the entire local traffic, which is mainly generated by node C, so that the boomerang effect was only observed in the middle zone of plot. In a real network situation wherein the path linking the two proxies is traversed by very heavy signaling traffic, the boomerang effect of SIP over UDP retransmissions is expected to come out more dramatically. The fraction of INVITE messages incurred in retransmition with respect to the total is reported in the following table. One can see that the higher the call generation rate, the higher the re-transmission percentage. This is mainly due to the lack of congestion and flow control.

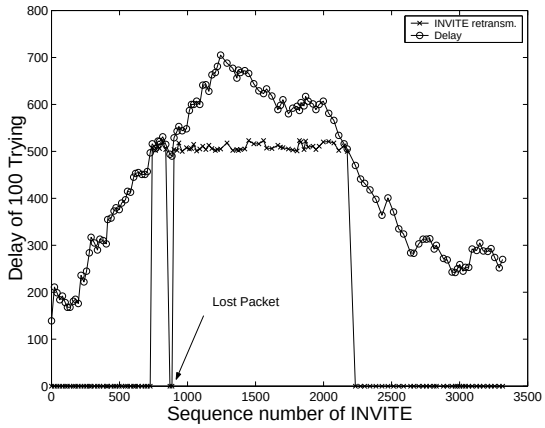

Figure 9: SIP over UDP: The 100 Trying arrives just after SIP timer expiration, causing "aggressive" retransmissions.

INVITE rate	Fraction of retransmitted INVITEs
20 ms	4%
18 ms	8%
13 ms	13%
10 ms	38%
8 ms	66%

REFERENCES

[1] J. Rosemberg, H. Schulzrine *et al.*: **SIP: Session Initiation Protocol**, IETF draft, [Online]Available: http://www.ietf.org/internet-drafts/draft-ietf-sip-rfc2543bis-09.ps, February 27, 2002

[2] 3GPP TSG SSA, **IP Multimedia Subsystem (IMS) – Stage 2 (Release 5)**, TS 23.228.v5.6.0, 2002-09, [Online]Available:http://www.3gpp.org

[3] D. Wills, B. Campbell, **SIP Extension to Assure Congestion Safety**, IETF draft, [Online]Available: http://www.softarmor.com/wgdb/docs/draft-willis-sip-congestsafe-00.txt, June 20, 2002

[4] 3GPP TSG CN, **Signaling Flows for the IP Multimedia Call Control Based on SIP and SDP**– Stage 3 (Release 5), TS 24.228 vXXXX, [Online]Available:/http://www.3gpp.org

[5] 3GPP TSG CN, **IP Multimedia Call Control Protocol Based on SIP and SDP**– Stage 3 (Release 5), TS 24.229 vXXXX, [Online]Available:http://www.3gpp.org

[6] G. Camarillo, H. Schulzrinne, R. Kantola, **Evaluation of Transport Trotocols for the Session Initiation Protocol**,Network, IEEE , Vol. 17 , Issue: 5 , Sept.-Oct. 2003, pages:40 - 46

[7] G. Camarillo, H. Schulzrinne, R. Kantola, **Signaling Transport Protocol**, [Online]Available: http://www.cs.columbia.edu/ library/TR-repository/reports/reports-2002/cucs-002-02.pdf, February 12, 2002

[8] R. Stewart, Q. Xie, *et al*, **Stream Control Transmission Protocol**, IETF RFC2960, [Online]Available:http://www.ieft.org/rfc/rfc2960.txt

[9] La Monte H.P. Yarroll, K. Knutson, **Linux Kernel SCTP: The Third Transport**, [Online]Available: http://old.lwn.net/2001/features/OLS/pdf/pdf/sctp.pdf

[10] **lksctp**, [Online]Available:sourceforge.net/projects/lksctp

[11] 3GPP TS 23.060 **General Packet Radio Services (GPRS); Service description; Stage 2**, [Online]Available:http://www.3gpp.org

[12] J. Rosemberg, C. Huitema, R. Osborne, A. Houri,**A Proposal for IM Transport**, [Online]Available: http://www.softarmor.com/wgdb/docs/draft-rosenberg-simple-im-transport-00.txt, November 14, 2001

[13] M. Handley, V. Jacobson, C. Perkins, **SDP: Session Description Protocol**, IETF RFC2327, [Online]Availbale: http://www.ietf.org/rfc/rfc2327.txt, April, 1998

[14] R. Stewart, Q. Xie *et al*,**Socket API Extension for SCTP**, IETF draft, [Online]Available: http://www.ietf.org/internet-drafts/draft-ietf-tsvwg-sctpsocket-08.txt,April 1, 2004

[15] J. Rosemberg, H. Shulzrinne, G. Camarillo,**The Stream Control Transmission Protocol as a Transport for the Session Initiation Protocol**, IETF draft, [Online]Available: http://www.softarmor.com/wgdb/docs/draft-ietf-sip-sctp-00.txt,August 14, 2001

[16] A.P. Markopoulou, F.A. Tobagi, M.J. Karam, **Assessment of VoIP Quality over Internet Backbones**,Proceeding of IEEE INFOCOMM 2002, Vol. 1, pages: 150-159, 2002