RESEARCH ARTICLE OPEN ACCESS

Multi-Streaming Behavior in Protocol Independent Transport API

Sangram Keshari Nayak [1], Sarojananda Mishra [2]
[1], [2] Department of Computer Science and Engineering, Indira Gandhi Institute of Technology, Sarang - India

ABSTRACT
Rapid growth of Internet leads to development of many innovative applications. A lot of research has been carried out in introducing new concepts in the existing transport protocol. One of the innovative transport layer service is SCTP multi-streaming. But there is a chance of high risk of failure of using specific protocol in the application development and OS development scenario. Some researchers[1] have initiated to introduce the implementation of transport layer protocol as services in a protocol independent way. They proposed common API that only offers the services as requirement. In this paper first we discussed the advantages and disadvantages of TCP, SCTP and existing solutions proposed by[1]. Second we demonstrated their approach and identified the benefits for SCTP multi-streaming. In the current development scenario BSD sockets API provides system calls that are not tied to any specific protocol. Hence we demonstrated the abstractions provided by socket API that can be treated as services. We have identified various network characteristics and protocol parameters that have been implemented in a protocol independent way.

Keywords: SCTP, TCP, UDP, UMTS, RTT, PI_API, DCCP, UDP-Lite.

I. INTRODUCTION

Currently a lot of applications have been developed in the current networking scenario. Lots of research has been carried out to develop network protocols to meet the application needs. Major protocols used in this regard are TCP[2] and UDP[3] which are standardized by Internet Engineering Task Force (IETF). Continuous work and research have been carried out over time New features are also added to the existing protocols to satisfy the requirements. But little importance is given for deployment of new protocols in the application development, operating system (OS) and in middle-boxes.

Application developers are the first group that tries to get best performance out of the program with minimal programming effort. They never try a chance to deploy new protocol until unless the work is seen as good chance of success. There might be some risk associated with it having the chance of failure. Hence application developers might not take interest towards deployment of new protocol. A few would be interested to invest extra effort with the good probability of using new thing with an intention of getting some benefit out of it.

Second group is operating system (OS) developers those try to minimize the risk and deliver best performance under some known risk constraints. This hinders them to use new protocols although having some beneficial features.

Third group is middle-box designers those try to focus on security and eliminate vulnerability. As a policy to block unwanted and unnecessary applications that might be having potential security risk. This prohibits deployment of new protocols and another reason is also the intention to deliver good performance to the Internet service provides.

Stream Control Transmission Protocol (SCTP)[5] is a unicast general purpose reliable connection oriented transport layer protocol, which is standardised by IETF, which provides ordered delivery of data.

Although TCP and UDP are the most widely used protocol, neither of them provide network fault tolerance capabilities. TCP faces the problem of head of line blocking. Due to independent messaging and order preservation property, it encounters head of line blocking. This may lead to control timers to expire and set up failures.

TCP also lacks path level redundancy supports. Since development of SCTP was motivated in finding a better transport mechanism for telephony signalling, it was evolved to more general use to satisfy need of applications which require a message oriented protocol. This is a requirement for using SCTP protocol, which have TCP like mechanism and additional features, that not present in TCP or UDP. SCTP provides sequencing, flow control, reliability and full duplex data transfer like TCP. However SCTP[11] provides some enhanced set of capabilities that are not available in TCP, which makes applications more susceptible to loss. Like UDP, SCTP supports framing of data and data transport is message oriented. SCTP is session oriented and communicates by establishing the connection between two endpoints, called an association. SCTP association can represent multiple IP addresses and ports at two endpoints, where as a TCP connection is bound to one IP address. SCTP supports two types of sockets[7][10]. These are one-to-one style and one-to-many style sockets. Since design objective of SCTP being to adapt TCP applications with little effort, here one-to-one style socket provides the function. Similar to UDP one-to-many style a single socket can communicate with multiple SCTP associations. SCTP also supports multiple logical streams within one association. Each stream is independent and provides sequential message delivery. So there might

occur a loss of one stream and does not affect other streams. It is useful in a way that overcomes the limitations of TCP and burrows beneficial features of UDP. Unlike other transport protocols, it offers advantages like multi-homing and multi-streaming capabilities.

The next section describes some unique features of SCTP like multi-homing and multi-streaming[8] .

II. SCTP FEATURES

It has some unique features like multi-homing and multi-streaming. The intention of multi-streaming is to decrease the impact of head-of-line blocking. A stream in SCTP is unidirectional and multiple streams can constitute one association.

A. General Properties

Like TCP, SCTP is connection oriented protocol meaning that an association must be established before data sending. SCTP is multi-homed[9]. This means an association can involve several IP addresses at each end, where as TCP an handle only one IP address at a time at each end. For this several paths can be made to transfer of packets. Benefits of using different alternate paths is that one path will be used for transfer and others responsible for re-transmission or path failures. Within the association, logical streams exist and stream is unidirectional, that means stream can specified in both directions. SCTP endpoint specifies number of streams it would receive. Thus endpoints can have different number of streams ready to receive on. So one SCTP association haves at least one stream in each direction.

SCTP[10] has congestion control functionality, which is similar to TCP. But this functionality is different from TCP.

III. TRANSPORT TUSSLE

Although SCTP have a number of advantages over TCP, there exists a tussle between three parties involved in the development process. One is the application developers, which have the goal of getting best performance from the application. Second group being the OS developers, those have the focus of minimizing risk of new technology. Third group are the developers of middle boxes e.g. firewalls. This group have the focus on the security issues and maintainability. Thus there is a tussle among these groups in adopting new protocols like SCTP. So there is need of a new Application Programming Interface (API), could help of easier adoption of new protocols. So these arguments have the conclusion that a new transport API is needed that will work in a protocol independent way, called as Protocol Independent Application Programming Interface (PI_API). Thus it is interesting to think in the light of using API in a protocol independent way.

IV. MOTIVATION

APIs of TCP, UDP, SCTP are quite complex and applications require name of the protocol. Assuming that these protocols get deployed as common transfer protocols, the application programmers face different choices from a set of protocols. SCTP faces partial reliability. Thus although DCCP[4] provides this, it has also different forms of congestion control. So there will be a choice of DCCP or SCTP for application programmers. DCCP has ACK congestion control but SCTP has partial reliability. Another option is UDP-Lite[6], which has better control over congestion control. The constraints behind these protocol are that these are not widely available in client server applications. Thus from these choice of protocols and features, the application programmers should decide one of these to be used.

DCCP does not have a standardised API yet. SCTP is the second alternative to be used, but this is also quite complex. So gradual development of simplified access to these protocols came under progress. The application programmers may have a choice of getting services as per choice and does not have a decision between TCP and SCTP.

API is a difficult and endless task as the design space is large and requires quite knowledge of understanding. So the possible solution is to use the abstractions of existing API. It is possible to use best required services from available services. This will simplify the API and make easier to be done. Out of possible choices of services, it is also possible to add or remove the functions. Removal of services may be done from simplicity point of view or not required. We make a summary of listing all services and prepare the services that will be required. So these functionality should be placed underneath API. In the next section we have a discussion of API available and also the considerable works in this related area.

V. RELATED WORKS

Socket API has been extended for SCTP[5]. XTI is another API provides was designed for ISO/OSI model. It provides abstractions towards use of transport protocol in an independent way. These APIs are quite complex and does not provide simplified abstractions to the users. Socket API is one of the good choice among the available APIs. Socket API truly supports currently available transport protocols. Socket API is the simplest API and easily configurable with little knowledge of programming.

These APIs discussed are rather complex and socket API is the good choice to start. So it is possible to simplify the abstractions provided by socket API to make our current requirement.

VI. DESIGN

Considerable work has been done by jorer[1]. As per their design specification protocol features or services provided by SCTP, DCCP, UDP-Lite are discussed. The capabilities of these protocols have been studied. SCTP features are compared with TCP and UDP also. For DCCP the features are given in RFC 4340[4] and UDP-Lite is given in RFC 3828[6]. So the approach starts with a list of

services that all the three protocols provide and some of the services are removed as not required. Some of the included and excluded services are discussed here.

A. Included Services

*1) **Connection Oriented:***

Since protocols have different connection establishment procedures, this feature is investigated and compared for all the three protocols.

*2) **Flow control:***

TCP and SCTP simply supports flow control.

*3) **Congestion control:***

All the protocols support congestion control mechanism by different procedures.

*4) **Application Protocol Data Unit (PDU) Handling:***

It is useful for transmission overhead. Higher latency enables this feature but SCTP implementation requires different procedures.

*5) **Error Detection:***

Protocols provide full or partial error detection feature. This is possible by different mechanisms. Error detection is done by checksum applied to entire packet or partial specifying range of checksums.

*6) **Reliability:***

TCP supports total reliability where as SCTP provides partial reliability mechanism.

*7) **Delivery type:***

Two types of delivery types are used in protocols. One is message based and another is stream based.

*8) **Multi-homing:***

Multi-homing is not supported by protocols other than SCTP.

*9) **Multi-streaming:***

Multi-streaming feature is only supported by SCTP.

B. Excluded services

*1) **Full duplex:***

All the transport protocols support this feature. Hence this is not a feature to be compared for the protocols.

*2) **ECN capable:***

It is the feature included in congestion control. So this is not further investigated.

*3) **Selective ACK:***

It is also included in congestion control. So this feature is not a comparable feature.

*4) **Path MTU Delivery(PMTUD):***

PMTU delivery can be used with any layer. Current Linux implementation does not allow a sender to send longer datagrams. So it can not be used as service among protocols.

*5) **Protection against SYN flood attack:***

Protection against this attack is an important feature. So all the protocols used this feature and can not be denied as a required service. So this feature is not a distinguishable feature for transport user.

*6) **Allows half closed connection:***

This is an important core level protocol feature. So it is not taken taken into consideration.

*7) **Reachability check:***

This is required for multi-homed protocols to check the endpoint reachability. Since this is included in multi-homing, this is not taken a distinguishable feature.

*8) **Time wait state:***

This is a protocol internal mechanism. So this is not taken as a service.

The services described above is compared and required to have a choice of services rather than choosing a protocol. So by handling protocol from application, performance can be gradually be imported by merely changing the transport system underneath the API or OS.

VII. IMPLEMENTATION

We have chosen the socket API for our implementation and we have taken the adaptation of native functions related to the SCTP socket. Here we need to incorporate multi-streaming feature.

A. Socket adaptation

Our choice of socket is based on connection oriented socket API [12]. The user is required to pass the service as parameter in stead of six parameters in the native socket function. The creation of a socket looks like int socket(int domain, int service). All the service characteristics are retrieved through getsocket(). Moreover the setsocket() function is used to set the parameters of configurable features. Multi-streaming feature is incorporated by function that is identical to API identical to current transport APIs.

B. Socket options

All the service characteristics are revised and set by functions setsockopt() and getcockopt(). The setsocket() function sets the configurable parameters. All the socket options are used by the protocol independent socket API are set with level argument PI_API. So for protocol independent a service is renamed with prefix as PI.

All the service characteristics are retrieved through getsocket() function. Moreover setsocket() function contains configurable features. Although the options affect the behavior of the API, all the options of

socket level (SOL_SOCKET) are still accessible. Here the API is simplified and it offers two different send/receive mechanisms in stead of four different methods in the standardised socket API[12]. The send/receive functions are shown in figure – 1.

```
sendmsg(int socket, const struct msghdr *message, int flags);
recvmsg(int socket, struct msghdr *message, int flags);
send(int socket, const void *buffer, size_t length, int flags);
recv(int socket, void *buffer, size_t length, int flags);
```

Fig. 1 Example of an unacceptable low-resolution image

They are identical to current versions of standardized socket API.

It is to mention that the socket() call has been altered to meet the PI_API requirements. All other functions remain same as the single UNIX System Specifications[13]. Previously authors introduced the prefix pi for all the functions to distinguish PI_API calls and normal socket calls.

C. Implementation scenario

The application scenario is set up by two hosts. The proposed application is compared with existing socket API and the application is based on client server model. The client produces four parallel streams with original TCP connection. The TCP protocol is partially replaced by one multi-streaming association of SCTP. For performing tests to compare TCP and SCTP , we used two hosts that run Ubuntu[15] version 14.04 as different nodes.

1) TCP to SCTP Protocol Translation Software:

The need of protocol translation software is needed to enable TCP applications to support SCTP. One of the tool is a available in Linux is *withsctp*[16]. It is a user space terminal application that works under Linux and included in *lksctp-tools* package. These tools are already included in our kernel version 3.4.10 and available on both server and client nodes.

2) Network emulation:

In our experiment in addition to server and client computer, another third computer is regarded as intermediate node and acts as Ubuntu router. This computer uses *netem* as network emulator, which is an integral part of Linux operating system. Network emulator is used to simulate properties of network. Emulator software is applied to the intermediate node to generate constant path delay. The reason for emulation is that practical measurements would be meaningless without applying delay in network paths. The emulation of higher path delay require router queue adaption, otherwise packets will be lost and show a different network behaviour.

3) Iperf as performance measurement tool:

Iperf[14] is a terminal tool available in Linux as network performance measurement tool. Initial performance measurement was done by using *iperf*. Although *iperf* is used as a measurement tool for TCP and UDP, SCTP support is carried out by using *iperf* in combination of *withsctp* software. The working of withsctp is to exchange the TCP packet with SCTP packet.

Experiments were done using latest stable version of protocol stack implementation. The details of hardware and software uses is listed in table-I and table II.

iperf[14] with current timestamps of its machine and placed in socket buffer as message queue. Then messages are transmitted over the network to the server application. User input data for each send() socket system call by using the server code.

VIII. PERFORMANCE AND DISCUSSION

In this section we discuss the comparative measurements for TCP and SCTP with same work

TABLE I
HARDWARE CONFIGURATION

Switch	Netgear Prosafe 5 port fast Ethernet switch
Network interface A	Qualcomm Atheros Lite-On Communications Inc Device
Network interface B	Qualcomm Atheros Lite-On Communications Inc Device
Ethernet controller	Realtek Semiconductor Co., Ltd. RTL8111/8168/8411 PCI Express Gigabit Ethernet Controller

TABLE II
SOFTWARE SPECIFICATION

Operating system	Ubuntu 14.04
Kernel version	3.4.10
Network emulator	Netem kernel component(already enabled)
LKSCTP versions	Lksctp-tools 1.0.9
Network measurement tool	Iperf [14] 2.0.5

flow for both protocols. Experiments were carried out with standard test case. The network test bed for practical measurements are dependent on prevailing network condition and appropriate protocol adaptation is applied for good performance. Table-III summarizes necessary network emulation parameters. Higher bandwidth may be provided, but 100Mbps is emulated using this test as per a practical standard value.

TABLE III
EMULATION PARAMETERS

Network details	Specific values
IP version	IPV4
Network bandwidth	100Mbit
Path delay	100ms
Additional delay	10ms
Next Random value	25%
Delay link	35ms

Figure-2 shows bandwidth utilization for four parallel TCP streams and one SCTP association having four streams. It shows bandwidth values for different Round Trip Time(RTT). In this test scenario both TCP and SCTP performs quite similarly. Although variation occurs in bandwidth utilization, TCP has an average bandwidth utilization to 4.16 Mbits/s and corresponding value of SCTP multi-stream as 4.18 Mbits/s. Although variation occurs for bandwidth utilization, it is comparable with TCP. So we can accept that our PI_API program code works well.

Comparison result in figure-3 shows that throughput values with corresponding round trip

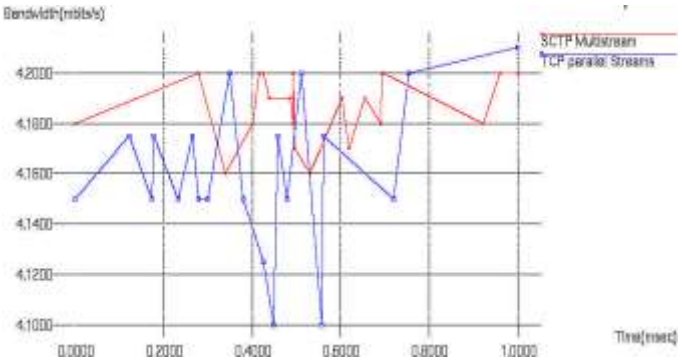

Fig. 2 Bandwidth utilization of SCTP vs TCP

time under 100ms bandwidth. Summary data obtained is given in table-IV. Value of RTT is

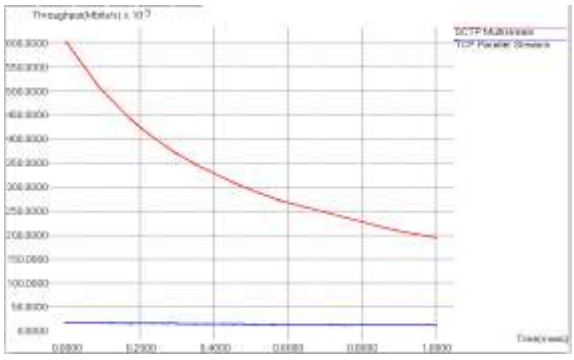

Fig. 3 Throughput vs RTT

obtained by taking random element depending 25% of the last one. The result obtained shows throughput decreases considerably for our PI_API multi-streaming as compared to four TCP parallel streams. Although behaviour of SCTP multi-streaming is quite unnatural, the decrease in value is due to larger transfer time. TCP parallel stream behaviour is slightly decreasing with transfer time. It is therefore considerable gain in performance of SCTP multi-streaming over TCP parallel stream, which is clear from the graphs.

We examined transfer time for uniform loss rates of 0, 0.01, 0.03, 0.06, 0.1. Table - V, VI, VII, VII, IX summarizes different transfer time obtained for both TCP streams and SCTP multi-stream for transferring 1MB, 5MB, 10MB, 25MB files for each simulated loss rates.

TABLE IV
MAXIMUM AND MINIMUM THHROUGHPUT

Maximum throughput for TCP	0.017
Minimum throughput for TCP	0.011
Minimum throughput for SCTP	0.603
Minimum throughput for SCTP	0.194

TABLE V
FILE TRANSFER, 1MBITS/S /35MS DELAY LINK WITHOUT LOSS

File size	Transfer time (Sec)	
	TCP	SCTP
1MB	2	1.7
5MB	9.4	9.9
10MB	18.5	20.8
25MB	47.3	50.3

TABLE VI
FILE TRANSFER, 1MBITS/S /35MS DELAY LINK WITH 1% LOSS

File size	Transfer time (Sec)	
	TCP	SCTP
1MB	2	1.7
5MB	9.3	9.9
10MB	19.4	19.8
25MB	45.3	23.9

TABLE VII
FILE TRANSFER, 1MBITS/S /35MS DELAY LINK WITH 3% LOSS

File size	Transfer time (Sec)

	TCP	SCTP
1MB	1.9	1.7
5MB	9.4	9.9
10MB	18.4	20.6
25MB	47.5	50

TABLE VIII
FILE TRANSFER: 1MBITS/S /35MS DELAY LINK WITH 6% LOSS

File size	Transfer time (Sec)	
	TCP	SCTP
1MB	1.9	1.7
5MB	9.3	10
10MB	19.4	19.8
25MB	46.5	50

TABLE IX
FILE TRANSFER: 1MBITS/S /35MS DELAY LINK WITH 10% LOSS

File size	Transfer time (Sec)	
	TCP	SCTP
1MB	1.9	1.7
5MB	9.3	9.9
10MB	19.4	19.7
25MB	45.4	49.9

Corresponding graphs obtained for file transfer and shown in figure-4, 5, 6, 7, 8. The graphs yield major observations about file transfer time over a TCP connection verses SCTP association. First in situations without any network loss is applied and consequent file transfer time obtained for various loss rates. Although some unexpected behaviour of SCTP observed in figure-5, graphs in figure 4, 6, 7 and figure-8 show similar behaviour in transfer time. This is apparent to accept that our PI_API for multi-streaming using socket that works in a similar fashion with TCP.

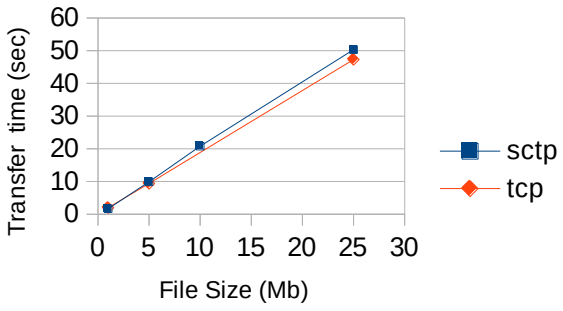

Fig. 4 Transfer Time vs. File si ze over 1 Mbps/35ms and loss rate 0%

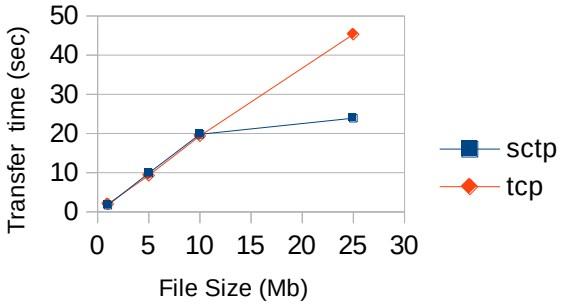

Fig. 5 Transfer Time vs. File size over 1 Mbps/35ms and loss rate 1%

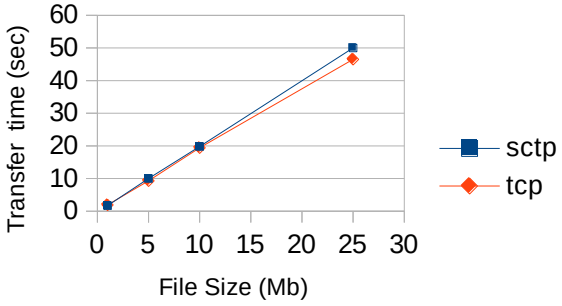

Fig. 6 Transfer Time vs. File size over 1 Mbps/35ms and loss rate 3%

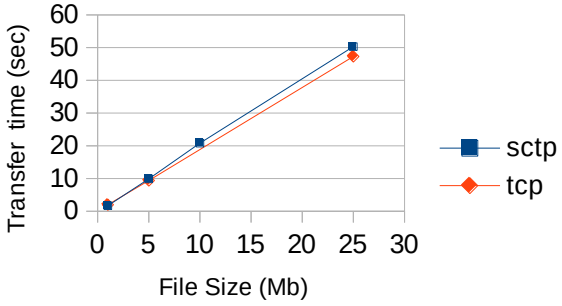

Fig. 7 Transfer Time vs. File size over 1 Mbps/35ms and loss rate 6%

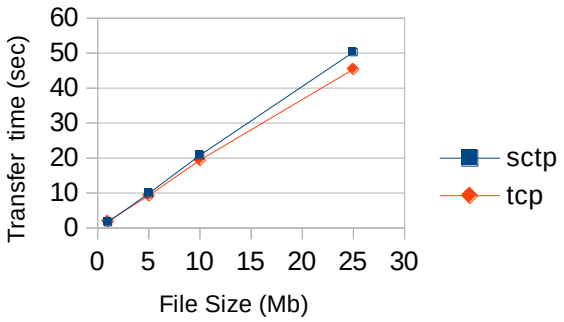

Fig. 8 Transfer Time vs. File size over 1 Mbps/35ms and loss rate 10%

IX. CONCLUSION

We feel major observations from our experiments that protocol translation is fairly feasible and works effectively in common network applications and provides performance that is equivalent what is possible using TCP.

REFERENCES

[1] S. Jorer, "A Protocol-Independent Internet Transport API", 2010. http://home.ifi.uio.no/michawe/teaching/dipls/stefan_joerer.pdf

[2] J. Postel. Transmission Control Protocol. RFC 793 (Standard), September 1981. Updated by RFCs 1122, 3168.

[3] J. Postel. User Datagram Protocol. RFC 768 (Standard), August 1980.

[4] E. Kohler, M. Handley, and S. Floyd. Datagram Congestion Control Protocol (DCCP). RFC 4340 (Proposed Standard), March 2006

[5] R. Stewart, K. Poon, M. Tuexen, V. Yasevich, and P. Lei. Sockets API Extensions for Stream Control Transmission Protocol (SCTP).

[6] A. Larzon, M. Degermark, S. Pink, LE. Jonsson, and G. Fairhurst, "The Lightweight User Datagram Protocol (UDP-Lite)," RFC 3828(Proposed Standard), Jul. 2004.

[7] Internet-Draft(work in progress),July 2010. http://tools.ietf.org/html/draft-ietf-tsvwg-sctpsocket-23.

[8] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, M. Kalla I. Rytina, L. Zhang, and V. Paxson. The stream control transmission protocol (SCTP). Available from http://www.ietf.org/rfc/rfc2960.txt, October 2000.

[9] P. Conrad, G. Heinz, A. Caro, P. Amer, and J. Fiore. SCTP in battleled networks. Proceedings MILCOM 2001, Washington, DC , October 2001.

[10] R. Stewart, K. Poon, M. Tuexen, V. Yasevich, and P. Lei, "Sockets API Extensions for Stream Control Transmission Protocol (SCTP)," Internet draft draft-ietf-tsvwg-sctpsocket-25 (work in progress), January 2011.

[11] R. Stewart and P. Amer, "Why is SCTP needed given TCP and UDP are widely available?" March 2010. [Online]. Available: http://www.isoc.org/briefings/017/

[12] "Networking Services, Issue 5 " February 2010, http://www.opengroup.org/bookstore/ catalog/c523.htm.

[13] Portable Operating System Interface (POSIX) Base Specifications, Issue 7. The Open Group. Technical Standard, December 2008. http://www.opengroup.org/bookstore/catalog/c082.htm .

[14] IPERF. URL http://iperf.sourceforge.net/

[15] Ubuntu Linux. URL http://www.ubuntu.com

[16] The Linux Kernel Stream Control Transmission Protocol Project - LKSCTP. URL http://lksctp.sourceforge.net