We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



12

On the Use of SCTP in Wireless Networks

Maria-Dolores Cano
Department of Information Technologies and Communications
Technical University of Cartagena
Spain

1. Introduction

Communications networks, particularly Internet, allow starting new businesses, to improve the current ones, and to offer an easiest access to new markets. Nowadays, Internet connects millions of terminals in the world, and it is a goal that this connection could be done with anyone, at any moment, and anywhere. In order to achieve this target, new lax and varied access requirements are needed. It is expected that a user would be able to access network services in a transparent way disregarding the location. The user terminal could seamlessly use the best available access technology (e.g., WLAN (Wireless Local Area Networks), LTE (Long Term Evolution), or PLC (Power Line Communications)), and service provisioning should agree with the user contract. This convergence of communications networks is giving rise to new challenges. The Internet Protocol (IP) has been selected to provide the necessary interconnection among all wireless and wired existing technologies. However, the use of IP does not solve all drawbacks. Multimedia applications show that current transport protocols like TCP (Transmission Control Protocol) or UDP (User Datagram Protocol) are not good enough to meet the new quality requirements.

To face these new challenges, the IETF (Internet Engineering Task Force) defined a new transport protocol called Stream Control Transmission Protocol (SCTP) (Stewart, 2007), whose main features are multihoming and multistreaming. Multistreaming allows transmission of several data streams within the same communication, splitting the application data into multiple streams that have the property of independently sequenced delivery, so that message losses in any one stream will only initially affect delivery within that stream, and not delivery in other streams. On the other hand, multihoming allows binding one transport layer's association to multiple addresses at each end of the SCTP association. The binding allows a sender to transmit data packets to a multihomed receiver through one of those different destination addresses. Therefore, SCTP is not only intended for signaling, but it can be used for any data application transport. The first studies about the performance of SCTP showed promising results. For instance, in (Kamal *et al.*, 2005), authors evaluate the benefits of using SCTP instead of TCP as the underlying transport protocol for a MPI (Message Passing Interface) middleware. Darche *et al.* (2006) presented a network architecture to enhance the cooperation of mobile and broadcast networks using SCTP as the transport layer protocol. In (Shaojian *et al.*, 2005), authors study the suitability of SCTP for satellite networks. Kim *et al.* investigate in (Kim *et al.*, 2006) the applicability of SCTP in MANET (Mobile Ad hoc NETworks). In (Kozlovszky *et al.*, 2006), authors carry out

performance measurements with TCP and SCTP as protocols to be used in distributed cluster environments. Finally, in (Natarajan *et al.*, 2006) authors propose the use of SCTP for HTTP-based applications, showing the benefits with real web servers compatible with SCTP. All these works showed the notable performance of SCTP as a multipurpose transport layer protocol.

This chapter reviews the specific use of SCTP in wireless networks and illustrates how to implement a multipurpose SCTP client/server application, compatible with IPv6, from a practical point of view. We describe how to enable multistreaming and multihoming capabilities. Through experimental tests in wired and wireless networks, we measure the SCTP performance regarding multistreaming and multihoming operation, compare it with the TCP protocol, and discuss its advantages and drawbacks. Therefore, the main contribution of this chapter is to present a survey in the work carried so far to turn the SCTP into a feasible transport-protocol option for wireless networks and to show the practical aspects of the design of a SCTP's open source client/server application, including some basic, but explanatory, experimental results in a single server – single client scenario. This work reveals that SCTP may be a competitive transport protocol for multimedia applications.

The rest of the chapter is organized as follows. Section 2 reviews the SCTP characteristics and its applicability in wireless networks. Section 3 explains how to make a SCTP client/server application. Experimental results are shown and discussed in Section 4. The chapter ends with conclusions in Section 5.

2. Related work

The SCTP features are described in this section. In addition, a survey about the applicability of SCTP in wireless environments has been also included. Among the advantages of using SCTP in wireless networks, mobility and multimedia transmission are highlighted, reviewing the most relevant works in these two areas. Other improvements like security or the introduction of redundancy for data delivery are also mentioned.

2.1 Stream control transmission protocol

SCTP is a message oriented transport protocol. Like TCP, SCTP provides a reliable transport service ensuring that data arrives in sequence and without errors. Like TCP, SCTP is a session-oriented mechanism, meaning that a relationship is created between the endpoints of a SCTP association prior to data being transmitted, and this relationship is maintained until all data transmission has been successfully completed. However, SCTP includes some new features (see Table 1) that evidence the advantages of using it in applications needing transport with additional performance and reliability.

Multihoming. A SCTP endpoint has the ability to work with more than one IP address, thus a session can remain active even in the presence of network failures. One of the main advantages is that in a conventional single-homed session, the failure of a local area network access can isolate the end system, but with multi-homing, redundant local area networks can be used to reinforce the local access. Multi-homing is not used for redundancy, as indicated in (Stewart, 2007). A pair of IP addresses <source, destination> is defined as the primary path, being used for data transmission. The other combinations of source and destination addresses will be considered as alternative paths, and will be employed in case of a primary path failure, which is detected by using the heartbeat mechanism (monitoring

function). The IP addresses of the SCTP association could be exchanged even if the association is already in use, i.e., it is possible to include new IP addresses during the communication (Stewart *et al.*, 2007). This feature is known as Dynamic Address Reconfiguration or Mobile SCTP.

Characteristics	TCP	UDP	SCTP
Unicast	Yes	Yes	Yes
Byte oriented	Yes	No	No
Message oriented	No	No	Yes
Reliable transport service	Yes	No	Yes
Multi-homing	No	No	Yes
Multi-stream	No	No	Yes
Cookie mechanisms	No	No	Yes
Rate adaptive	Yes	No	Yes
Heartbeat mechanism	No	No	Yes

Table 1. TCP, UDP, and SCTP comparison

Heartbeat Mechanism. A SCTP source should check if it is possible to reach the remote endpoint. This is done by means of the heartbeat mechanism. Alternative paths are monitored with heartbeat messages. Heartbeat messages are small messages with no user-data periodically sent to the destination addresses, and immediately acknowledged by the destination. The sender of a heartbeat message should increment a respective error counter of the destination address each time a heartbeat is sent to that address and not acknowledged within the corresponding time interval (RTO, Retransmission TimeOut). If this counter reaches a maximum value, the endpoint should mark this address as inactive. On the contrary, upon the receipt of a heartbeat acknowledgement, the sender of the heartbeat should clear the error counter of the destination address to which the heartbeat was sent, and mark the destination address as active.

Multistreaming. This feature allows splitting the application data into multiple streams that have the property of independent sequenced delivery, so that message losses in any one stream will only initially affect delivery within that stream, and not delivery in other streams. This is achieved by making independent data transmission and data delivery. SCTP uses a Transmission Sequence Number (TSN) for data transmission and detection of message losses, and also a Stream ID/Stream Sequence Number pair, which is used to determine the sequence of delivery of received data. Therefore in reception, the end point can continue to deliver messages to the unaffected streams while buffering messages in the affected stream until retransmission occurs.

Initiation. SCTP initiation procedure requires four messages. A cookie mechanism was incorporated to avoid Denial of Service (DoS) attacks. A SCTP client sends an init message to the SCTP server. The server replies with an init ack message that includes a cookie (a TCB (Transmission Control Block), a validity period, and a signature for authentication). Since the init ack is addressed to the source IP address of the init message, an attacker cannot get the cookie. A valid SCTP client would get the cookie, and send it back in a cookie echo message to the server. When this packet is received, the server starts giving resources to the client. The procedure finishes with a cookie ack message.

Data Exchange. Data exchange in SCTP is very similar to the TCP SACK procedure (Stewart, 2007). SCTP uses the same congestion and stream control algorithms as TCP.

Shutdown. SCTP shutdown procedure uses three messages: shutdown, shutdown ack, and shutdown complete. Each endpoint has an ack of the data packets received by the remote endpoint before closing the connection. SCTP does not support a half open connection, but it is assumed that if the shutdown initiates, then both endpoints will stop transmitting data.

2.2 SCTP in wireless networks

Seamless mobility is one of the challenges in wireless networks. With the proliferation of new types of wireless access technologies (e.g., WiFi, WiMAX, 3G, vehicular networks, etc.), a user, through his/her mobile device, should be able to change his/her location maintaining the Quality of Service (QoS) performance disregarding the roaming, either horizontal (under the same technology) or vertical (crossing different technologies). SCTP is a competitive solution for mobility due to its multihoming capability. Multimedia transmission is another challenge in wireless networks due to the higher likelihood of packet losses (error-prone channels). In this case, SCTP multistreaming improves the data rate throughput since streams are independently delivered; hence, the multimedia application is less sensitive to packet losses. Finally, some new modifications to SCTP have been presented in the related literature to increase its performance, e.g., allowing redundancy in multihomed devices. This section reviews the most relevant works in these areas.

2.2.1 Mobility and handovers in wireless networks

Several works in the related literature had demonstrated the advantages of using SCTP to improve both vertical or horizontal handovers and signaling in wireless networks. Authors in (Afif *et al.*, 2006a) proposed to include a new type of chunk in SCTP able to send QoS transmission parameters over the radio interface from an EGPRS mobile to the SCTP peer. By doing so, SCTP could adapt the transmission rate depending on the radio transmission conditions (e.g., LLC error rate, RLC/MAC block error rate, etc.). The reason to incorporate this new chunk, as stated by the authors, can be explained as follows. Even though SCTP is able to change the IP addresses in use, data packets are sent to old IP address before the alternative addresses become the primary ones. Therefore, there are packet losses during the exchange process. The simulation study in an EGPRS network with handovers between cells showed that the achieved throughput is higher with this modification than with the standard SCTP implementation because fewer packets are lost during handovers. From a similar perspective, same authors verified in (Afif *et al.*, 2006b) that their modification is also useful for handovers between EGPRS and Wireless Local Area Networks (WLAN).

Honda *et al.* proposed a new handover mechanism based on SCTP and a new data retransmission feature for smooth handover. In their work, authors state that the exchange of addresses in SCTP, assuming the new addresses to use are unknown at the beginning of the SCTP association (i.e., using Dynamic Address Reconfiguration), suffers a high delay mainly due to the multiple RTO expirations required to identify the failure. To overcome this situation, authors propose to include two algorithms called FastAssociation Reconfiguration and Fast Transmission Recovery. The former minimizes the RTO needed to substitute the addresses in use, whereas the latter allows sending data just after the establishment of the new addresses. Observe that in the standard, it was necessary to wait an RTO after a new path is configured to send data. The evaluation, carried out in an experimental network with WLAN links, showed that the handover latency was notably reduced using the authors' approach.

Focusing on vertical handover between WLAN and cellular networks, particularly UMTS (Universal Mobile Telecommunication System), authors in (Ma *et al.*, 2007) proposed a very interesting error recovery scheme called Sending-buffer Multicast-Aided Retransmission with Fast Retransmission that increases the throughput achieved during the SCTP connection in the presence of forced vertical handovers from WLAN to UMTS. A forced vertical handover occurs when the mobile node leaves the WLAN coverage due to the loss of signal and switches to the cell network. The advantages of using SCTP for vertical handovers were clearly identified in (Ma *et al.*, 2004): higher throughput, shorter delay, a simpler network architecture, and ease to adapt network congestion and flow control parameters to the new network; but a scenario with forced handovers involves important packet losses. Ma, Yu & Leung (2007) categorized these packet losses as dropping consecutive packets because of the loss of signal (WLAN) and random packet losses over the cellular link. To deal with these different types of errors, the authors propose to use two solutions. First, packet losses due to the loss of signal enable the Sending-buffer Multicast-Aided Retransmission algorithm, which multicast all buffered data on both the primary and the alternate address (observe that in a standard implementation SCTP only retransmits data to the alternate address if the error was due to a time out). The same applies to new data that needs to be sent. Second, packet losses likely due to random packet losses over the link (detected by the reception of duplicated acknowledgments) activate the Fast Retransmission algorithm, which force the retransmission to be done to the same destination IP address. With these two algorithms, long waiting delays are avoided, thus increasing the achieved throughput. Working on the same heterogeneous scenario with WLAN and UMTS networks, Shieh *et al.* (2008) detected that SCTP significantly decreases the congestion window when new primary addresses are used in the SCTP association (i.e., during a handover). Therefore, they proposed to assign an adequate initial congestion window according to the bandwidth available in the new path, so the association can skip the slow-start phase and enter the congestion avoidance phase directly. Packet-pair bandwidth proving is used to estimate the available bandwidth in the new path. Authors demonstrated the feasibility and goodness of their proposal through simulation. From an experimental point of view, authors in (Bokor *et al.*, 2009) designed and implemented a real native IPv6 UMTS-WLAN testbed to evaluate the effect of SCTP parameter configuration in terms of handover effectiveness, link changeover characteristics, throughput, and transmission delay. Among the most important parameters that have an effect on handover are: *RTO.Min*, *RTO.Max*, *Path.Max.Retransmission*, and *HB.Interval*. Authors verified that with the standard parameters, the handover delay would rise exponentially due to RTO redoubling, but using a more appropriate setting the handover delay rises linearly when the RTO is incremented. They also recommended keeping the *HB.Interval* (the time that elapses between consecutive heartbeat monitoring messages) as low as possible. Finally, they found that the SCTP performance in terms of delay, jitter, and throughput was better in UMTS than in WLAN.

From another perspective, authors in (Lee *et al.*, 2009) studied a mobile web agent framework based on SCTP. Typical web agents use TCP as transport protocol. However, mobile web agents using TCP present the following drawbacks: performance degradation, head-of-line (HOL) blocking, and unsupported mobility (as identified by IEEE Std 802.11-1997 and IEEE 802.16e-2005). By transmitting each object in a separate stream, SCTP solves the HOL problem. Mobility is achieved by the SCTP multihoming capability. To improve the performance, authors assumed that mean response time between HTTP requests and

replies is the most important performance parameter in a web environment. Therefore, they proposed to use SCTP to decrease the response time compared to the classical TCP implementation of web agents. Authors described the complete architecture for the mobile SCTP web agent framework. By simulation, they found that the mean response time decreased notably (around 30%) by using SCTP. The mean packet loss was also smaller with SCTP, and the faster the moving speed the better the SCTP performance in terms of packet loss compared to TCP.

Regarding the option of introducing crosslayer techniques to combine the SCTP features with information available at lower levels, the IEEE introduced the IEEE 802.21-2008 Media Independent Handover (MIH) as a way to provide link layer intelligence and other related network information to upper layers. MIH does not carry out the network handover, but it provides information to allow handover within a wide range of networks (e.g., WiFi, WiMAX, 3G, etc.). In (Fallon *et al.*, 2009) authors proposed to separate path performance evaluation (i.e., how SCTP detects that a path is no longer available) from path switching (i.e., update the new addresses of the primary path in the SCTP association). Whereas the first task will be done with MIH, SCTP will only be in charge of the second task (path performance is disabled in SCTP). By simulation, authors demonstrated that the combination of SCTP and MIH reacts to sudden performance degradation resulting from obscured line of sight in a heterogeneous scenario with WiMAX and HSDPA technologies. Indeed, the throughput of the SCTP connection improved notably (from 5% to 45%) compared to the standard SCTP implementations.

Network Mobility (NEMO), commonly used in military or vehicular applications, has been also studied from a SCTP perspective. In host mobility, a network in which terminals change their location, mobility is managed through the mobile node itself. In a mobile network, mobility is managed by a central node (e.g., a bus providing a WLAN service that moves around a city, hence changing the access point from which obtains Internet access). Leu & Ko (2008) proposed a method that combines SIP and SCTP with the aim of minimizing delay and packet losses during the handovers of a mobile network. With the authors' proposal, packet losses decreased significantly. Similarly, Huang & Lin (2010) presented a method to improve the bandwidth use and the achieved throughput in vehicular networks by using SCTP. Their approach is explained as follows. In a Vehicle to Infrastructure network (V2I), moving vehicular nodes communicate with Road Side Units (RSU) deployed in a specific area. RSU are connected to the wired infrastructure, e.g., providing Internet access to mobile vehicular nodes. Usually, several RSU share the same gateway to access the infrastructure. Therefore, authors proposed to use this gateway as a SCTP-packet monitoring station, buffering all SCTP packets containing data chunks. In the event of a packet loss, the gateway (not the destination node, which is assumed to be in the wired part of the network) will be in charge of retransmitting lost packets in the wireless link. With this scheme, the wired part of the communication is used more efficiently because no retransmissions are sent (unless the packet loss occurs in the wired part of the network). Moreover, since the destination node is not informed about packet losses in the wireless part of the network, its congestion window does not decrease as much, keeping a higher throughout rate in average. The performance of this proposal was done through simulation. Authors verified that the achieved throughput, the transmission time, and the congestion window behaved better with their approach than with the standard SCTP implementation.

2.2.2 Multimedia transmission over wireless networks

The use of multimedia services and applications over wireless links is another important research area. Authors in (Wang *et al.*, 2003) presented one of the first works evaluating the performance of Partial Reliability SCTP (PR SCTP), a modification of SCTP that provides unreliable transmission service to part of the data to be sent, as the transport protocol for video (MPEG-4) transmission in a wireless local area network. Results showed an improvement in the video quality comparing PR SCTP with UDP. Another interesting works regarding MPEG-4 video transmission over wireless technologies are presented in (Nosheen *et al.*, 2007) and (Chughtai *et al.*, 2009). In the first work, authors compared SCTP with UDP and DCCP (Datagram Congestion Control Protocol) (Kohler *et al.*, 2006). By simulation, they found that the throughput achieved by UDP could be more than 20% smaller than the throughput achieved by SCTP or DCCP in a wireless environment. However, the delay was higher in SCTP due to the congestion control mechanism. In the presence of background traffic, the results also showed that SCTP and DCCP outperformed UDP. As an extension to this work, Chughtai *et al.* (2009) carried out a similar study to compare the QoS performance of SCTP, UDP, and SCTP transmitting video in a WiMAX network. The simulation scenarios included downloading or uploading MPEG-4 video traffic using a different number of subscribers, different packet sizes, and a variable video rate. Results showed that delay and jitter were lower with SCTP than with UDP or DCCP. In terms of throughput, DCCP performed slightly better than SCTP, and both exceeded UDP performance.

Wang *et al.* (2008) also studied video delivery over wireless networks using SCTP. They focused on the multistreaming feature of SCTP, and how to use it to optimize video quality. Previous works from the literature such as (Balk *et al.*, 2002) showed the benefits of using multistreaming for MPEG-4 video transmission in wired network by applying a differential treatment among streams in a SCTP association. Differing from previous works, Wang *et al.* (2008) proposed MPEG-4 transmission with optimized partial reliability among streams in a heterogeneous scenario with error-prone 802.11 wireless channels. Their proposal was based on retransmitting packets belonging to stream of I-frames until packets are eventually received, while no retransmissions are attempted for packets in stream of B- and P- frames. In terms of retransmission overhead delay, simulation results showed that adjusting SCTP fast retransmit threshold can reduce the retransmission overhead delay, hence increasing the I-frame data rate, and the video quality. Furthering the results obtained in this work, the same authors introduced in (Wang *et al.*, 2009) an extension to the SCTP protocol. The goal was to improve the transmission of delay sensitive multimedia data by including a selective retransmission of lost packets depending on whether the lost packets would still arrive before the schedule time. Assuming that there is clock synchronization between the SCTP associated peers, authors included a new field to the SCTP header with the time a packet is sent, so that the endpoint after reception can estimate the one-way delay. This value is sent to the sender from the receiver in the acknowledgement packet. Then, in the receiver side, the time of each frame of MPEG-4 to be played out is calculated, so if the frame is not received before this schedule time will be considered as non-useful and its retransmission will not be necessary. By simulation, authors achieved interesting results, confirming the improvement in the MPEG-4 video transmission performance.

Voice over IP (VoIP) is another important application that is gaining momentum. Chang *et al.* (2009) presented a middleware to transfer the session initiation protocol (SIP) signaling and real-time transmission protocol (RTP) messages from using UDP or TCP to SCTP.

Switching from UDP or TCP to SCTP (with Dynamic Address Reconfiguration) provides a seamless way for the user to roam maintaining the QoS level of the VoIP call. Authors analyzed their proposal in a real testbed. Nevertheless, results showed that although mobility was achieved, the delay was higher with their proposal.
Live TV broadcasting over wireless technologies could also benefit from the use of SCTP. Liu *et al.* (2010) introduced a method to provide an economic way of live news broadcasting by using SCTP. Satellite News Gathering (SNG) vehicles, which usually use satellite links for transmission, are an expensive service for TV companies, mainly due to the required equipment. In this case, the current deployment of WiMAX networks is a feasible alternative to satellite communication, but the bandwidth offered by WiMAX is not enough to provide a live TV service with QoS demands. Therefore, the authors proposed to take advantage of all available wireless networks, not only WiMAX but also HSDPA or WiFi, thus increasing the available bandwidth. A SCTP multi-link connection with both multihoming and multi-streaming was a key point for this implementation. SCTP Concurrent Multipath Transfer, which will be explained in next section, is also needed. With an experimental testbed, authors demonstrated the feasibility of their proposal, not only achieving a cost-effective system to provide live TV broadcasting but also increasing the coverage of previous SNG systems.

2.2.3 Other SCTP improvements

Concurrent Multipath Transfer (CMT) consists of simultaneously sending data over all available paths, hence, increasing the bandwidth of the SCTP association (Iyengar *et al.*, 2006). In environments where the paths of the SCTP association exhibit very different network conditions (e.g., round trip times or bandwidth), packet reordering is required in the receiver side, and this might cause retransmission, lowering the connection rate. To avoid this situation, authors in (Perotto *et al.*, 2007) compared the performance of two SCTP modifications: Sender-Based Packet Pair SCTP (SBPP-SCTP) and Westwood SCTP (W-SCTP). The former uses the sender-based packet pair technique, mentioned in the previous section, to estimate the bottleneck bandwidth of each path. The latter uses the same algorithm as in TCP Westwood (Mascolo *et al.*, 2004) for the bandwidth estimation. Both aim at minimizing packet reordering. In presence of intermittent interfering cross-traffic, authors showed that W-SCTP achieves a higher throughput than SBPP-SCTP. Aydin & Shen (2009) studied the performance of CMT SCTP over 802.11 static multihop wireless networks. They compared CMT SCTP with three different techniques: i) standard SCTP using just one path (the best one in terms of bandwidth) to send data, ii) standard SCTP using just one path (the worst one in terms of bandwidth) to send data, and iii) standard SCTP using all available paths to send data (splitting the traffic into the different available paths of the SCTP association). Results showed that in a multihop wireless scenario the achieved throughput is higher with CMT SCTP than with any of the three alternatives used for comparison. Nevertheless, CMT SCTP still presents a drawback to be completely useful for wireless networks: the received buffer blocking problem. This problem was clearly stated in (Wang *et al.*, 2010): "In SCTP transmission, data streams between each other are logically independent, if receiver has received all data chunks of a certain stream. The data of this stream can be delivered to the application layer. But in traditional CMT, because data chunks of the same stream maybe transferred to different paths, the data chunks could not arrive at the receiver orderly and duly, so the receive buffer blocking problem happens. This problem can seriously influence network performance, especially in high error rate and

delay wireless network." Consequently, authors proposed a new modification of the SCTP called Wireless Concurrent Multipath Transfer SCTP (WCMT SCTP). With this modification, each SCTP path delivers packets belonging to the same stream (one or more than one). For instance, if there are three paths available and there are five streams, then the first path only transmits packets from the first stream, the second path only transmits packets from streams two and three, and the third path only transmits packets from streams four and five. Authors also added other changes to the standard CMT implementation: a per-path congestion control mechanism, a new congestion control mechanism and a new retransmission mechanism that takes into account the type of error. Results obtained by simulation showed that WCMT SCTP performs better than CMT SCTP in ad hoc networks. In a similar way, Yuan *et al.* (2010) improved the CMT SCTP mechanism by categorizing the streams depending on their specific QoS requirements, and grouping those streams with similar QoS needs in subflows that are sent through the more appropriate paths available in the SCTP association. Finally, the work done in (Xu *et al.*, 2011) showed how to optimize CMT SCTP for video and multimedia content distribution.

Another interesting works that improve the performance of SCTP in wireless environments from different perspectives are (Cui *et al.*, 2007; Cano *et al.*, 2008; Lee & Atiquzzaman, 2009; Cheng *et al.*, 2010; Funasaka *et al.*, 2010). Cui *et al.* (2007) proposed to use a hierarchical checksum method that improves the retransmission procedure, thus increasing the achieved throughput in links with high packet losses. Cano *et al.* (2008) investigated how to combine the use of IPSec (Internet Protocol Security) with SCTP to enhance the security of the wireless communication. The work done in (Lee & Atiquzzaman, 2009) presented an analytical model to estimate the delay of HTTP over SCTP in wireless scenarios. Last, Cheng *et al.* (2010) proposed to use two new methods for bandwidth estimation and per-stream-based error recovery.

Library	Description
netinet/sctp.h	It contains definitions for SCTP primitives and data structures.
netdb.h	It contains definitions for network database operation, e.g. translation.
sys/socket.h	It defines macros for the Internet Protocol family such as the datagram socket or the byte-stream socket among others.
netinet/in.h	It contains definitions of different types for the Internet Protocol family, e.g. sockaddr_in to store the socket parameters (IP address, etc.).
arpa/inet.h	To manage numeric IP addresses, making available some of the types defined in netinet/in.h

Table 2. Description of the libraries related to SCTP network communication

3. Implementation

For the sake of simplicity, we implement three SCTP client/server applications. The first one is called single SCTP, the second one is called multistream SCTP, and the last one is called multihomed SCTP. Single SCTP is very similar to TCP, since it will be able to transmit just

one data stream between source and destination endpoints. Multistream SCTP includes multistreaming, and finally, multihomed SCTP incorporates multihoming. The three implementations are written in C code. We use the libraries provided by the Berkeley Socket Application Programming Interface, which are briefly described in Table 2. Next sections detail the practical SCTP implementation issues.

```
1  int main(int argc, char *argv[])
2  {
3   int sockfd;
4   struct hostent *host;
//Structures to manage IP address
5   struct sockaddr_in remote_addr;   //IPv4
6   host = gethostbyname(argv[1]);
7   ra_family = host->h_addrtype;  //AF_INET
//IPv4 socket
8   sockfd = socket( ra_family, SOCK_STREAM, IPPROTO_SCTP);
9  if(sockfd == -1)
10    {perror("Socket:");exit(1);}
//Set server IP address
11  remote_addr.sin_family=AF_INET;
12  remote_addr.sin_port=htons(REM_PORT);
13  remote_addr.sin_addr=*((struct in_addr *)host->h_addr);
14  bzero(&(remote_addr.sin_zero),8);
//Connect to server
15  if(connect(sockfd,(struct sockaddr*)&remote_addr,sizeof(struct sockaddr))==-1)
16    {perror("connect:"); exit(1);}
//Omitting lines of code to receive a file
//Close socket
17  close(sockfd);
18  return 0;
19 }
```

Fig. 1. Extract of the original SCTP client code in a single file transmission

3.1 Single SCTP

SCTP server and SCTP client structures are very similar to those used in TCP. Fig. 1 shows how to implement a SCTP client. The only difference with TCP is in the *socket()* function, where the protocol type field should be IPPROTO_SCTP instead of the common parameter 0 used for TCP or UDP transport protocols (see code line 8 in Fig. 1). The rest of the implementation is done as in TCP; i.e., once the socket is created, the server IP address is set (see code lines 11-14 in Fig. 1), and the client connects to the server (see code line 15 in Fig. 1). Observe that we use the server IP address as an argument in the command line (see code line 6 in Fig. 1). If we want to use IPv6 instead of IPv4, some simple changes included in Table 3 are needed. First, it is necessary an appropriate structure to store an IPv6 address. Second, the *gethostbyname()* function, needs to know that the IP address is an IPv6 one, and the same applies to all lines of code where we use the IP address.

In Fig. 2, we define how to implement a SCTP server. In this case, to execute the server, no parameters are needed in the command line. We define a constant called MYPORT to include the port number associated to the server IP address (see code line 12 in Fig. 2). The server IP address is automatically set to any local IP address available (see code line 13 in Fig. 2). Then, we follow the usual sequence to set up the server. First, we create the socket with the *socket()* function. As indicated before, the socket protocol is set to IPPROTO_SCTP (see code line 15 in Fig. 2). Then, we set the socket parameters with the *bind()* function (see code line 18 in Fig. 2). Afterwards, we execute *listen()* so that the server can receive a specific number of client requests (see code line 20 in Fig. 2). The *accept()* function makes the server to wait for client requests (see code line 25 in Fig. 2). Finally, if a client request is received, the client is served by a child process due to the *fork()* function (see code line 27 in Fig. 2). To make it compatible with IPv6, lines indicated in Table 4 should be replaced.

```
1  int main(int argc, char *argv[])
2  {
3   int sockfd, newfd;
4   socklen_t sin_size;
5   struct sockaddr_in local;
6   struct sockaddr_in remota;
7   struct hostent *host;
8   sa_family_t la_family;
9   la_family = host->h_addrtype;
10  host = gethostbyname(argv[1]);
11  local.sin_family = AF_INET;
12  local.sin_port = htons(MY_PORT);
13  local.sin_addr.s_addr = htonl(INADDR_ANY);//Any local IP address
14  bzero(&(local.sin_zero),8);
15  sockfd = socket( la_family, SOCK_STREAM, IPPROTO_SCTP);
16  if(sockfd == -1)
17    {perror("Socket:"); exit(1);}
18  if((bind(sockfd, (struct sockaddr*)&local, sizeof(struct sockaddr)))==-1)
19    {perror("bind");exit(1);}
20  if(listen(sockfd,5) == -1)
21    {perror("listen");exit(1);}
22  for(;;)
23  {
24    sin_size=sizeof(struct sockaddr_in);
25    if((newfd = accept(sockfd, (struct sockaddr*)&local,&sin_size)) == -1)
26      {perror("accept");exit(1);}
27    if (!fork()
//Omitting lines of code to send a file
28  while(waitpid(-1,NULL,WNOHANG)>0);}
```

Fig. 2. Extract of the original SCTP server code in a single file transmission

Line# in Fig. 1	New code for IPv6

5	struct sockaddr_in6 remote_addr6; //IPv6
	struct in6_addr ipv6; //To store IPv6 address

6	host = gethostbyname2(argv[1], AF_INET6); //get IP address
	sockfd = socket(ra_family, SOCK_STREAM, IPPROTO_SCTP);
	if(sockfd == -1) {perror("Socket:");exit(1);}

8 to 14	remote_addr6.sin6_family = AF_INET6;
	remote_addr6.sin6_flowinfo = 0;
	remote_addr6.sin6_port = htons(REM_PORT);
	inet_pton(AF_INET6, argv[2], ipv6.s6_addr);
	remote_addr6.sin6_addr = ipv6;

15	if(connect(sockfd,(struct sockaddr*)&remote_addr6,sizeof(struct sockaddr))==-1)

Table 3. How to make the SCTP client implementation compatible with IPv6. Lines indicated in the first column should be replaced with lines shown in the second column

Line# in Fig. 1	New code for IPv6
5-6	struct sockaddr_in6 local6; struct sockaddr_in6 remota6;
10	host = gethostbyname2(argv[1], AF_INET6);

11 to 14	local6.sin6_family = AF_INET6;
	local6.sin6_flowinfo = 0;
	local6.sin6_port = htons(MY_PORT);
	local6.sin6_addr = in6addr_any;

18	if((bind(sockfd, (struct sockaddr*)&local6, sizeof(struct sockaddr))) == -1)
24-25	sin_size=sizeof(struct sockaddr_in6); if((newfd = accept(sockfd, (struct sockaddr*)&local6,&sin_size)) == -1)

Table 4. How to make the SCTP server implementation compatible with IPv6

3.2 Multistream SCTP

A SCTP client/server application with multistreaming allows sending/receiving multiple streams simultaneously. For instance, these different streams could belong to different files, so it would be possible transferring several files with the same SCTP association. Thus, the client only uses one request to the SCTP server. Nowadays, file downloading (music, games, software, etc.) is one of the most important services driving the usage of Internet. With the multistreaming SCTP feature, a unique association between SCTP client and SCTP server may accept many multimedia file transmissions, resulting in bandwidth saving as it will be shown in Section 4. The less traffic in the network, the more efficient the use.

From the SCTP client point of view, the multistream operation has to be enabled by setting some particular properties. The sequence is as follows. First, we create all data structures. Second, we create the SCTP socket as explained in the previous section (see code line 10 in Fig. 3). Then, the maximum number of ingoing and outgoing streams should be indicated. Accordingly, *setsockopt()* is used to set the number of flows or streams in the client/server

```
1   int main()
2   {
3   int connSock, in, ret;
4   struct sockaddr_in servaddr;
5   struct sctp_status status;
6   struct sctp_sndrcvinfo sndrcvinfo;
7   struct sctp_event_subscribe events;
8   struct sctp_initmsg initmsg;
9   int numElem=0, firstTime=1;
10  connSock = socket( AF_INET, SOCK_STREAM, IPPROTO_SCTP ); //IPv4
11  memset( &initmsg, 0, sizeof(initmsg) );
12  initmsg.sinit_num_ostreams = 30; //max streams
13  initmsg.sinit_max_instreams = 30; //max streams
14  initmsg.sinit_max_attempts = 5; //max attempts
15  ret = setsockopt( connSock, IPPROTO_SCTP, SCTP_INITMSG,&initmsg,
    sizeof(initmsg) );
16  bzero( (void *)&servaddr, sizeof(servaddr) ); //server to connect to
17  servaddr.sin_family = AF_INET;
18  servaddr.sin_port = htons(MY_PORT_NUM);
19  servaddr.sin_addr.s_addr = inet_addr("192.168.1.10" );
20  ret = connect( connSock, (struct sockaddr *)&servaddr, sizeof(servaddr) ); //connect
    to the server
21  memset( (void *)&events, 0, sizeof(events) );
22  events.sctp_data_io_event = 1;
23  ret = setsockopt( connSock, SOL_SCTP, SCTP_EVENTS,(const void *)&events,
    sizeof(events) );
24  //File transfer
25  //Loop to receive the different streams
26  in = sctp_recvmsg( connSock, (void *)buffer, sizeof(buffer),(struct sockaddr *)NULL,
    0, &sndrcvinfo, &flags );
27  if(in==0)
28     break;
29  //Store each stream in its corresponding file
30  if (sndrcvinfo.sinfo_stream == STREAM1)
31    { if(firstTime)
32        {fp=fopen("reciboweb.txt","wb"); firstTime=0;}
33    numElem=fwrite(buffer, 1, 1024, fp);
34    if(num_elementos<1024) {fclose(fp); break;} }
35  else if (sndrcvinfo.sinfo_stream == STREAM2) 42
36  {//Save this stream in its corresponding file, lines 31-41}
37  else if ...//Save each stream in its corresponding place
38  //End loop to receive different streams
39  fclose(fp);
40  close(connSock);
41  return 0; }
```

Fig. 3. Extract of the original SCTP client code in a multistream transmission

SCTP association. Both client and server agree on this parameter (see code lines 12-15 in Fig. 3). Afterwards, both the server IP address and the port number to connect to are indicated (see code lines 16-19 in Fig. 3). Next, the client connects to the server (see code line 20 in Fig. 3). Finally, we enable data delivery with the function *setsockopt()* (see code lines 21-23 in Fig. 3). By doing so, the client is able to use the primitive *sctp_recvmsg()* for data delivery. At this point, the client is ready to receive data in multiple streams within the same SCTP association.

On the other hand, the SCTP server multistream implementation also needs some variations compared to the SCTP server single implementation. First, we declare data structures. After that, we create the SCTP socket as explained in the previous section (see code line 8 in Fig. 4). Then, the server IP address is automatically set to any local IP address available, the port is assigned, and the *bind()* function is called (see code line 9-13 in Fig. 4). The maximum number of ingoing and outgoing streams is specified now (see code lines 15-18 in Fig. 4). Observe that it is the same value used previously for the client implementation. Next, the server remains listening for client requests (see code line 19 in Fig. 4). If there is a client request, then the server accepts the connection, and it starts sending the corresponding files.

Once a client is connected to the server, the information sent from the server to the client should be identified, so that the client knows what file (stream) the data belong to. Whereas the source (the server in this case) is in charge of assigning an identifier to each stream, which is done with the *sctp_sendmsg()*function and a stream number (see line 28 and 35 in Fig. 4), each stream is identified using the *sndrcvinfo.sinfo_stream* field (see line 30 and 35 in Fig. 3) in the receiving side (the client in this case).

3.3 Multihomed SCTP

In this section, we describe the additional code necessary to facilitate the SCTP multihomed feature. After calling the *bind()* function and before the SCTP association is established, any additional address should be enabled. Otherwise, multihoming cannot be used unless Dynamic Address Reconfiguration is set. Enabling addresses is done with the *sctp_bindx()* function. *sctp_bindx()* links any IP address (IPv4 or IPv6) to the SCTP association. It can be also used to delete an IP address from an association. Table 5 shows the new lines of code.

New code for multihoming

hst_adicional = gethostbyname(argv[3]);//get additional address/es
sctp_bindx(sockfd, (struct sockaddr*)ip4, 1, SCTP_BINDX_ADD_ADDR);

Table 5. How to make the SCTP client/server implementation with multihoming

4. Experimental results

Three different scenarios are evaluated to compare the performance of SCTP vs. TCP. In the first scenario, our SCTP application transfers a single text file, a single mp3 file, or a single mpeg file from server to client. We called it the single operation. In the second scenario, our SCTP application transmits different types of files simultaneously from server to client. We called it the multistream operation. The former is like a normal TCP transfer file operation. The latter could emulate a web loading, where usually different types of multimedia files are involved. In the third scenario, we test the multihoming feature in what we called the multihomed operation. Next we describe the experimental topology, and discuss the experimental results.

```
1   int main()
2   {
3   int listenSock, connSock, ret, msglen;
4   struct sockaddr_in servaddr;
5   struct sctp_initmsg initmsg;
6   FILE *fp;
7   int num_bytes=0;
8   listenSock = socket( AF_INET, SOCK_STREAM, IPPROTO_SCTP );
9   bzero( (void *)&servaddr, sizeof(servaddr) );
10  servaddr.sin_family = AF_INET;
11  servaddr.sin_addr.s_addr = htonl( INADDR_ANY );
12  servaddr.sin_port = htons(MY_PORT_NUM);
13  ret = bind( listenSock, (struct sockaddr *)&servaddr, sizeof(servaddr) );
14  memset( &initmsg, 0, sizeof(initmsg) );
15  initmsg.sinit_num_ostreams = 30;
16  initmsg.sinit_max_instreams = 30;
17  initmsg.sinit_max_attempts = 5;
18  ret = setsockopt( listenSock, IPPROTO_SCTP, SCTP_INITMSG, &initmsg,
    sizeof(initmsg) );
19  listen( listenSock, 5 );
20  int i=0;
21  while( 1 )
22  {
23    connSock = accept( listenSock, (struct sockaddr *)NULL, (int *)NULL );
24    fp = fopen("textoweb.txt","rb");
25    do
26    {
27      num_bytes=fread( (void *)buffer, 1,1024, fp);
28      ret = sctp_sendmsg( connSock, (void *)buffer, (size_t)strlen(buffer),NULL, 0, 0, 0,
      STREAM1, 0, 0 );
29    }while(!feof(fp));
30    fclose(fp);
31    fp = fopen("vaquero.jpg","rb");
32    do
33    {
34      num_bytes=fread( (void *)buffer, 1,1024, fp);
35      ret = sctp_sendmsg( connSock, (void *)buffer, (size_t)strlen(buffer),NULL, 0, 0, 0,
      STREAM2, 0, 0 );
36    }while(!feof(fp));
37    fclose(fp);
38    ...//Send each file with its corresponding stream identifier
39  }
40  return 0; }
```

Fig. 4. Extract of the original SCTP server code in a multistream transmission

4.1 Experimental scenario

The experimental topology is illustrated in Fig. 5. We measure both the time required to initialize the TCP or SCTP socket(s), and the time that it takes to transfer the file(s) with TCP or SCTP. Tests are carried out with two laptops in a 10 Mbps wired Ethernet local area network. Both laptops also have wireless cards to verify the multihoming feature. During the tests, there was no other traffic in the network, but the one from these experiments. Likewise, the only application running on the laptops is our TCP or SCTP application.

In the single operation tests, we transmit a 1 MB file from the server to the client through the wired local area network, and repeat the experiment for a 3MB file, and a 50MB file. Each transmission is repeated 100 times. In the multistream operation tests, the client should load a multimedia web page from the server. Therefore, the client should download a variety of multimedia files. Since we have not implemented a web server compatible with SCTP, we carry out experiments assuming that the client downloads two or four multimedia files of different sizes. Both tests (downloading two or four multimedia files) are performed 100 times. Experimental results have a confidence interval of 95% that has been calculated with a normal distribution function using 100 samples.

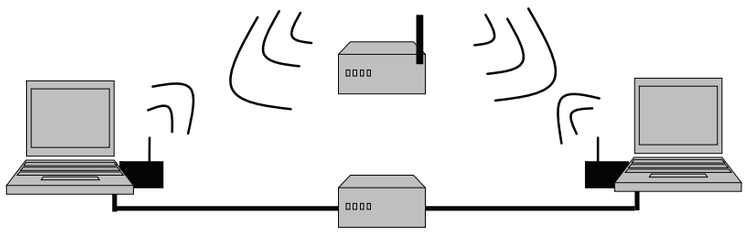


Fig. 5. Experimental topology. Laptops have Intel Centrino platforms, Intel Pentium M 740/1.73 GHz processors, and 1GB RAM. Operating system is Linux (SuSe 10.0)

4.2 Results

Results from the single operation tests show that TCP is slightly faster than SCTP in a single file transmission. Table 6 includes the average transmission time for single-file transmissions with TCP and SCTP and the corresponding confidence intervals. For instance, we observe that the transmission of a 3 MB file with SCTP lasts 2.73 seconds compared to the 2.6 seconds of TCP. SCTP is slower than TCP for two reasons. Firstly because its socket initiation time is 1ms larger (it uses four packets, adding the effect of the cookie mechanism). Secondly, the monitoring of the path that the SCTP carries out periodically (heartbeat mechanism) also introduces some overhead. As a result, the SCTP transmission lasts approximately 3% more than the TCP one.

Regarding the multistream operation, the first clear conclusion is that TCP requires more IP packets to proceed with these transmissions. A TCP connection requires three packets for negotiation and four packets for shutdown. Therefore, the more files to transmit with TCP the more packets, because it is necessary to establish a different connection to download each file (each stream) with TCP. Likewise, a SCTP association needs four packets for negotiation and three for shutdown, however, SCTP will only require an association for downloading multiple files. Fig. 6 shows the overhead amount produced with SCTP and TCP, where the x axis represents the number of files to be transmitted and the y axis the number of bytes used. We represent in this figure the number of bytes used in TCP for initiation and shutdown, as well as the number of bytes consumed by SCTP in initiation, shutdown, and heartbeat packets. For the heartbeat mechanism, we consider sending the

heartbeat signal every 100ms, 250ms, 500ms, and 1 s. Observe that the time interval for sending the heartbeat is an adjustable parameter. Clearly, the more frequent the heartbeat the more bandwidth consumed. Assuming the minimum possible packet sizes for TCP and SCTP, and taking into account the SCTP heartbeat mechanism, the overhead introduced by TCP would be smaller than the one introduced by SCTP only if the heartbeat is very aggressive. Otherwise, the fact of establishing one TCP connection for each file transmission produces higher bandwidth consumption.

	1 MB file		3 MB file		50 MB file	
	TCP	**SCTP**	**TCP**	**SCTP**	**TCP**	**SCTP**
Average transmission time (s)	1.06	1.09	2.60	2.73	47.11	48.52
Confidence interval	0.24	0.29	0.32	0.20	3.78	2.26

Table 6. SCTP vs. TCP average transmission times in *single* operation tests

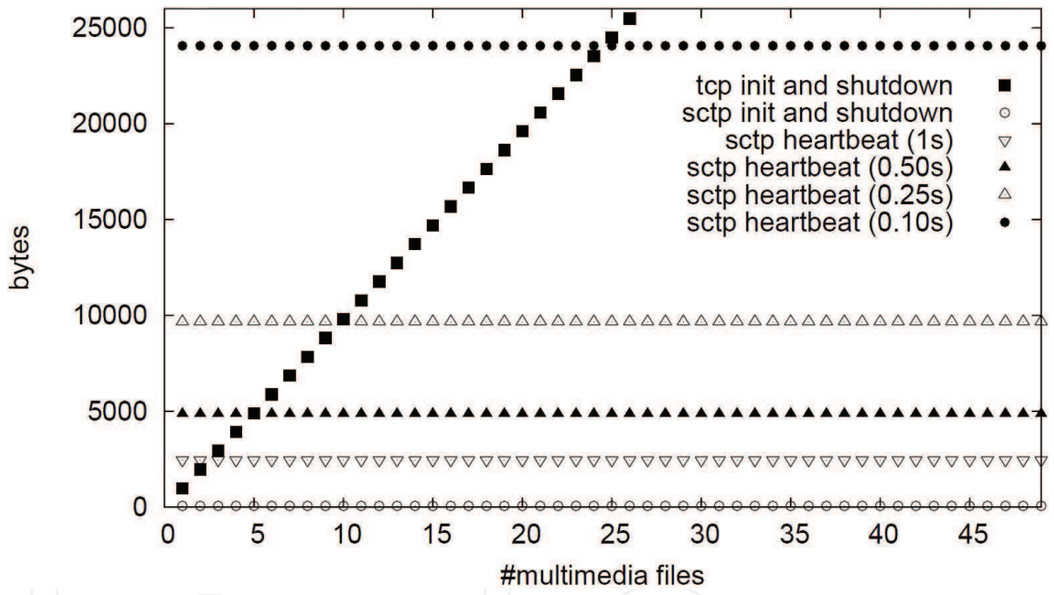


Fig. 6. Overhead introduced by TCP and SCTP. For TCP, the packet size is 20 bytes (we assume no data is sent with the first ACK packet). For SCTP, we take the following minimum packet sizes as indicated in (Stewart, 2007): INIT 20 bytes, INIT ACK 20 bytes, COOKIE ECHO 8 bytes, COOKIE ACK 4 bytes, HEARTBEAT REQUEST 4 bytes, HEARTBEAT ACK 4 bytes

On the other hand, socket initiation is still faster in TCP. However, since more sockets need to be used in TCP, the total initiation time difference between TCP and STCP is shorter and shorter as the number of files to be transmitted increases. Fig. 7 and Fig. 8 represent the duration of initiating sockets in TCP versus initiating sockets in SCTP. Indeed, when two multimedia files are transmitted (Fig. 7), the average time dedicated to sockets initiation in TCP is 1.69 ms, while the average time is 1.73 ms for SCTP. However, if we send four multimedia files, the average time increases to 3.4 ms average in TCP whereas approximately the same value remains in SCTP (Fig. 8). Thus, when four files are transmitted, SCTP total initiation time is half of the TCP total initiation time. Consequently, results show that not only the SCTP multiple file transmission is faster than the TCP one,

but it consumes less bandwidth. Table 7 includes the average times for a multiple-file transmission and the corresponding confidence intervals.

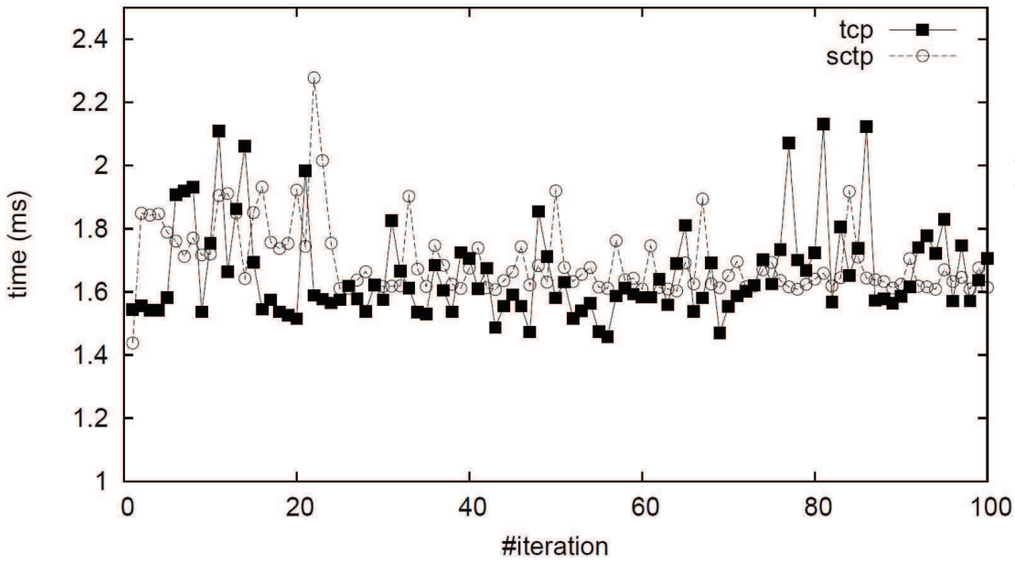


Fig. 7. Socket initiation time in TCP and SCTP in two-file downloading

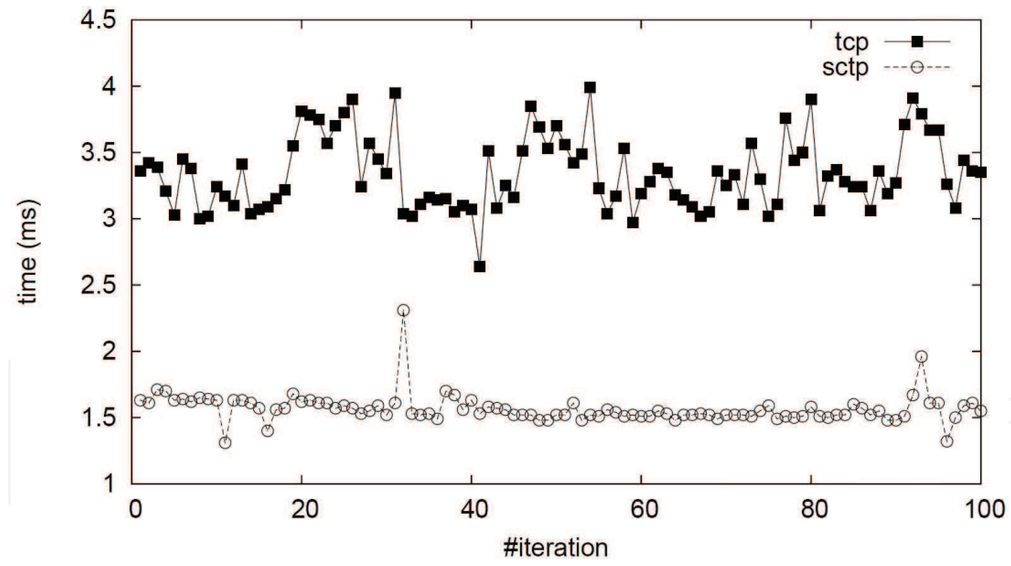


Fig. 8. Socket initiation time in TCP and SCTP in four-file downloading

	2 multimedia files		4 multimedia files	
	TCP	**SCTP**	**TCP**	**SCTP**
Average Transmission time (s)	6.20	3.10	18.02	6.90
Confidence intervals	1.55	0.86	1.61	1.07

Table 7. SCTP vs. TCP average transmission times in *multistream* operation tests

Finally, we test the multihoming SCTP feature in the topology shown in Fig. 5, where two PCs are connected to each other through two interfaces (one is wired, the other is wireless). We use the multistreaming SCTP client and server implementations shown in Fig. 3 and Fig. 4 respectively, including the new lines shown in Table 5. At first, client and server are using the wired network (primary IP addresses). Then, one of the wired network interface card is disabled. Experimental results show that in less than 1 second SCTP reacts in the presence of the network failure by replacing primary IP addresses with the alternative one (wireless one) to continue with the transmission. The time to change the IP addresses in use includes the ARP resolution, which is almost negligible in this scenario. Table 8 shows the exchange of IP addresses in use.

No.	Time	Source	Destination	Protocol	Info
55559	185.79019	192.168.1.10	192.168.1.11	SCTP	DATA
55560	185.79022	192.168.1.11	192.168.1.10	SCTP	SACK
55561	185.79108	192.168.1.10	192.168.1.11	SCTP	DATA
55562	185.79215	192.168.1.10	192.168.1.11	SCTP	DATA
55563	185.79218	192.168.1.11	192.168.1.10	SCTP	SACK
55564	185.79304	192.168.1.10	192.168.1.11	SCTP	DATA
55565	185.99060	192.168.1.11	192.168.1.10	SCTP	SACK
55570	186.79958	linuxpedro.local		ARP	who has 192.168.2.33? Tell 192.168.2.34
55571	186.79959	192.168.2.33		ARP	192.168.2.33 is at 00:80:5a:32:cb:c0
55572	186.80009	linuxpedro.local		ARP	who has 192.168.2.33? Tell 192.168.2.34
55573	186.80009	192.168.2.33		ARP	192.168.2.33 is at 00:80:5a:32:cb:c0
55574	186.81128	192.168.2.34	192.168.2.33	SCTP	DATA
55575	186.81132	192.168.2.33	192.168.2.34	SCTP	SACK
55576	186.83170	192.168.2.34	192.168.2.33	SCTP	DATA

- Frame 55565 (64 bytes on wire, 64 bytes captured)
- Linux cooked capture
- Internet Protocol, Src: 192.168.1.11 (192.168.1.11), Dst: 192.168.1.10 (192.168.1.10)
- Stream Control Transmission Protocol, Src Port: 5200 (5200), Dst Port: 20000 (20000)
  - Source port: 5200
  - Destination port: 20000
  - Verification tag: 0x 52c9c5b0
  - Checksum: 0xe5b04b29 [correct CRC32C]
  - SACK chunk (Cumulative TSN: 261016901, a_rwnd: 112640, gaps:0, TSNs: 0)

Table 8. Extract of the traffic captured with Wireshark (Wireshark, 2011). The first 6 SCTP packets use the primary IP addresses. After the network failure (packet# 55565), alternative addresses are used

5. Conclusion

In this work, we have presented a survey with the most relevant works on the applicability of SCTP in wireless networks. We have categorized the benefits of SCTP for wireless technologies in the following categories: mobility and handovers, multimedia transmission, and other improvements related to multiple path transmission or security. We have also shown the practical aspects of the design of a SCTP client/server application. In our example, the SCTP application is used to download files from a server. We have described the basics of how to enable multihoming and multistreaming capabilities in SCTP. We have observed that it is quite easy to adapt current applications to the SCTP protocol. When comparing to TCP, the advantages of SCTP are numerous (e.g., faster average transmission times and resources saving), above all in applications that require the transmission of multiple files. Moreover, multihoming allows increasing reliability, a key additional requirement in multimedia applications over wireless networks.

6. Acknowledgment

This research has been supported by the MICINN/FEDER project grant TEC2010-21405-C02-02/TCM (CALM).

7. References

Afif, M., Martins, P., Tabbane, S., & Godlewski, P. (2006a). Radio aware SCTP extension for handover data in EGPRS. *Proceedings 17th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications PIMRC'06*, pp. 1-5.

Afif, M., Martins, P., Tabbane, S., & Godlewski, P. (2006b). SCTP Extension for EGPRS/WLAN Handover Data. *Proceedings 31st IEEE Conference on Local Computer Networks*, pp. 746-750.

Aydin, I., & Shen, C.-C. (2009). Performance Evaluation of Concurrent Multipath Transfer Using SCTP Multihoming in Multihop Wireless Networks. *Proceedings 8th IEEE International Symposium on Network Computing and Applications*, pp. 234-241.

Balk, A., Sigler, M., Gerla, M., & Sandidi, M. Y. (2002). Investigation of MPEG-4 video streaming over SCTP. *Proceedings 6th World Multiconference on Systemics. Cybernetics. and Informatics SCI'02*, pp. 1-4.

Begg, C. L., Pawlikowski, K., Sirisena, H., & De Silva, P. (2007). Suitability of SCTP for High Quality Video Streaming over CDMA2000. Proceedings Australasian Telecommunication Networks and Applications Conference, pp. 496-502.

Bokor, L., Huszák, A., & Jeney, G. (2009). On SCTP Multihoming Performance in Native IPv6 UMTS–WLAN Environments. *Proceedings 5th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities and Workshops TridentCom'09*, pp. 1-10.

Cano, M.-D., Romero, J.A., & Cerdan, F. (2008). Experimental Tests on SCTP over IPSec. *Proceedings IFIP International Conference on Network and Parallel Computing NPC'08*, pp. 96-102.

Chang, L.-H., Huang, P.-H., Chu, H.-C., & Tsai, H.-H. (2009). Mobility Management of VoIP services using SCTP Handoff Mechanism. *Proceedings Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*, pp. 330-335.

Cheng, R.-S., D.-J., Chao, H.-C., & Chen, W.-E. (2010). An Adaptive Bandwidth Estimation Mechanism for SCTP over Wireless Networks. *Proceedings 5th International Conference on Future Information Technology*, pp. 1-5.

Chughtai, H. M. O., Malik, S. A., & Yousaf, M. (2009) Performance Evaluation of Transport Layer Protocols for Video Traffic over WiMax. *Proceedings IEEE 13th International Multioptic Conference*, pp. 1-6.

Cui, X., Cui, L., & Koh, S. J. (2007). A Hierarchical Checksum Scheme for SCTP over Wireless Networks with Worse Channel Condition. *Proceedings International Conference on Wireless Communications, Networking and Mobile Computing WiCom'07*, pp.1845-1848.

Darche, D., Kopp, R., Mazieres, B., Lepage, F., & Gnaedinger, E. (2006). Using SCTP to improve performances of hybrid broadcast/telecommunication network system. *Proceedings of IEEE Consumer Communications & Networking Conference*, Vol. I, pp. 371-375.

Fallon, E., Murphy, L., & Murphy, J. (2009). Optimizing Metropolitan Area Wireless Path Selection Using Media Independent Handover. *Proceedings Second International Workshop on Cross Layer Design IWCLD'09*, pp. 1-5.

Honda, M., Sakakibara, H., Nishida, Y., & Tokuda, H. (2007). SmSCTP: A Fast Transport Layer Handover Method Using Single Wireless Interface. *Proceedings 12th IEEE International Symposium on Computers and Communications ISCC'07*, pp.319-324.

Huang, C.-M., & Lin, M.-S. (2010). RG-SCTP: Using the Relay Gateway Approach for Applying SCTP in Vehicular Networks. *Proceedings IEEE International Symposium on Computers and Communications ISCC'10*, pp. 139-144.

IEEE Std 802.11-1997. (1997). IEEE 802.11 wireless LAN medium access control (MAC) and physical layer (PHY) specifications.

IEEE Std. 802.16e-2005. (2005). IEEE Standard for Local and metropolitan area networks. Part 16: Air interface for fixed broadband wireless access systems.

IEEE Std. 802.21-2008. (2008). IEEE Standard for Local and metropolitan area networks. Part 21: Media Independent Handover Services.

Iyengar, J. R., Amer, P., & Stewart, R. (2006). Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths. *IEEE/ACM Transactions on Networking*, Vol. 14, No. 5, pp. 951–964.

Kamal, H., Penoff, B., & Wagner, A. (2005). SCTP versus TCP for MPI. *Proceedings of ACM/IEEE SuperComputing Conference SC'05*, pp. 30-44.

Kim, D., Song, J., Kim, J., Yoo, H., Park, J., & Cano, J.C. (2006). The Applicability of SCTP to Mobile Ad Hoc Networks. *Proceedings International Conference on Advanced Communication Technology ICACT'06*, Vol. 3, pp. 1979-1984.

Kohler, E., Handley, M., & Floyd, S. (2006). Datagram Congestion Control Protocol. RFC 4340.

Kozlovszky, M., Berceli, T., & Kutor, L. (2006). Analysis of SCTP and TCP based communications in high speed clusters. *Nuclear Instruments and Methods in Physics Research Section A*, Vol. 559, Issue 1, pp.85-896.

Lee, Y.-J. & Atiquzzaman, M. (2009). Mean Waiting Delay for Web Object Transfer in Wireless SCTP Environment. *Proceedings IEEE International Conference on Communications ICC'09*, pp. 1-5.

Lee, Y.-J., Lee, D.-W., & Atiquzzaman, M. (2009). Novel web agent framework to support seamless mobility for data networks. *IET Communications Journal*, Vol. 3, No. 12, pp. 1861–1869.

Leu, F.-Y. & Ko, Z.-J. (2008). A Novel Network Mobility Scheme Using SIP and SCTP for Multimedia Applications. *Proceedings of International Conference on Multimedia and Ubiquitous Engineering*, pp. 564-569.

Liu, H.-S., Hsieh, C.-C., Chen, H.-C., Hsieh, C.-H., Liao, W., Chu, P.-C., & Wang, C.-H. (2010). Exploiting Multi-link SCTP for Live TV Broadcasting Service. *Proceedings IEEE 71st Vehicular Technology Conference*, pp. 1-6.

Ma, L., Yu, F., Leung, V. C. M., & Randhawa, T. (2004). A new method to support UMTS/WLAN vertical handover using SCTP. IEEE Wireless Communications, Vol. 11, No. 4, pp. 44-51.

Ma, L., Yu, F. R., & Leung, V. V. M. (2007). Performance Improvements of Mobile SCTP in Integrated Heterogeneous Wireless Networks. *IEEE Transactions on Wireless Communications*, Vol. 6, No. 10, pp. 3567-3577.

Mascolo, S., Grieco, L. A., Ferorelli, R., Camarda, P., & Piscitelli, G. (2004). Performance evaluation of Westwood+ TCP congestion control. *Performance Evaluation*, Vol. 4, No. 55, pp. 93–111.

Natarajan, P., Iyengar, J. R., Amer, P. D., & Stewart, R. (2006). SCTP: An innovative transport layer protocol for the web. *Proceedings 15th International World Wide Web Conference, WWW'06*, pp. 615-624.

Nosheen, S., Malik, S. A., Zikria, Y. B., & Afzal, M., K. (2007). Performance Evaluation of DCCP and SCTP for MPEG4 Video over Wireless Networks. *Proceedings IEEE 11th International Multitopic Conference*, pp. 1-6.

Perotto, F., Casetti, C., & Galante, G. (2007). SCTP-based Transport Protocols for Concurrent Multipath Transfer. *Proceedings IEEE Wireless Communications and Networking Conference WCNC'07*, pp. 2969-2974.

Shaojian, F., Atiquzzaman, M., & Ivancic, W. (2005). Evaluation of SCTP for space networks. *IEEE Wireless Communications*, Vol. 12, No. 5, pp. 54-62.

Shieh, C.-S., Lin, I-C., & Lai, W. K. (2008). Improvement of SCTP Performance in Vertical Handover. *Proceedings of Eighth International Conference on Intelligent Systems Design and Applications*, pp. 494-498.

Stewart, R. (2007). Stream Control Transmission Protocol. RFC 4960.

Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., & Kozuka, M. (2007). Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration. RFC 5061.

Wang, B., Feng, W., Zhang, S.-D., & Zhang, H.-K. (2010). Concurrent multipath transfer protocol used in ad hoc networks. *IET Communications Journal*, Vol. 4, No. 7, pp. 884–893.

Wang, H., Jin, Y., Wang, W., Ma, J., & Zhang, D. (2003). The performance comparison of PRSCTP, TCP and UDP for MPEG-4 multimedia traffic in mobile network. *Proceedings International Conference on Communication Technology (ICCT),* Vol. 1, pp. 403-406.

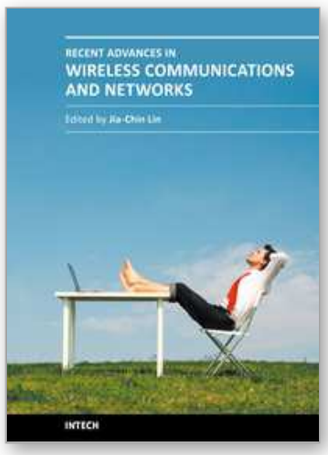
Wang, L., Kawanishi, K., & Onozato, Y. (2008). MPEG-4 Optimal Transmission over SCTP Multi-streaming in 802.11 Wireless Access. *Proceedings 7th Asian-Pacific Symposium on Information and Telecommunication Technologies*, pp. 172-177.

Wang, L., Kawanishi, K., & Onozato, Y. (2009).Achieving Robust Fairness of SCTP Extension for MPEG-4 Streaming. *Proceedings 20th Personal, Indoor and Mobile Radio Communications Symposium PIMRC '09*, pp. 2970-2974.

Wireshark. <http://www.wireshark.org>. Last visited March 20th, 2011.

Xu, C., Fallon, E., Qiao, Y., Zhong, L., & Muntean, G.-M. (2011). Performance Evaluation of Multimedia Content Distribution Over Multi-Homed Wireless Networks. *IEEE Transactions on Broadcasting*, Vol. PP (99), pp. 1-12.

Yuan, Y., Zhang, Z., Li, J., Shi, J., Zhou, J., Fang, G., & Dutkiewicz, E. (2010). Extension of SCTP for Concurrent Multi-Path Transfer with Parallel Subflows. *Proceedings IEEE Wireless Communications and Networking Conference WCNC'10*, pp. 1-6.



Recent Advances in Wireless Communications and Networks
Edited by Prof. Jia-Chin Lin

ISBN 978-953-307-274-6
Hard cover, 454 pages
Publisher InTech
Published online 23, August, 2011
Published in print edition August, 2011

This book focuses on the current hottest issues from the lowest layers to the upper layers of wireless communication networks and provides â€œreal-timeâ€ research progress on these issues. The authors have made every effort to systematically organize the information on these topics to make it easily accessible to readers of any level. This book also maintains the balance between current research results and their theoretical support. In this book, a variety of novel techniques in wireless communications and networks are investigated. The authors attempt to present these topics in detail. Insightful and reader-friendly descriptions are presented to nourish readers of any level, from practicing and knowledgeable communication engineers to beginning or professional researchers. All interested readers can easily find noteworthy materials in much greater detail than in previous publications and in the references cited in these chapters.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Maria-Dolores Cano (2011). On the Use of SCTP in Wireless Networks, Recent Advances in Wireless Communications and Networks, Prof. Jia-Chin Lin (Ed.), ISBN: 978-953-307-274-6, InTech, Available from: http://www.intechopen.com/books/recent-advances-in-wireless-communications-and-networks/on-the-use-of-sctp-in-wireless-networks

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2011 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen