Concurrent-Multipath-Communication SCTP A Novel Method For Multi-Path Data Transmission

Abbas Malekpour[1], Hamideh Jabalameli[2], Djamshid Tavangarian[1]
[1] Department of Computer Science
University of Rostock, Germany
{abbas.malekpour,djamshid.tavangarian}@uni-rostock.de

[2] Department of Computer Science
University of Esfahan, PNU, Iran
hamideh.jabalameli@gmail.com

Abstract— **Multi-homing at the transport layer is a mechanism that ties one transport layer association to several network interfaces at each communication host, which enabling the two end-nodes to exchange data over several network paths. Stream control transmission protocol (SCTP) supports multi-homing feature. We suggest Concurrent Multipath Communication-SCTP (CMC-SCTP) which uses multi-homing as a base for simultaneous data transmission over multiple paths to increase end-to-end path throughput and network path failover. In this paper we identify the negative side effects of re-sequencing due to the use of CMC that must be managed in order to get the full performance of CMC's parallel data transmission. To overcome this weakness we proposed new chunks and algorithms to eliminate these side effects. Sender maps a general transmission sequence number for each data packet to a dedicated sequence number for each path to controls the transmission status independently for each path. A control packet called status-chunk informs the receiver about the path number of each transmitted data chunk and the status of transmitted data on each path. An OPNET simulation evaluates the performance of our CMC-SCTP protocol extension.**

Keywords-component; SCTP; Multi-homing; Load-Sharing; QoS; multipath communication; congestion control; transport layer

I. INTRODUCTION

New devices on the market now often can connect over more than one network interfaces (both wired and wireless). Multi-homed hosts (i.e. hosts that can be accessed under several IP addresses) along with networked technology and network devices are available and are getting more and more economical.

In 1974 that Transport Control Protocol (TCP) was introduced, computer equipment and specially network devices were too expensive, and thus providing multi-home network node was beyond the research perspective [1]. Multi-homing at the network transport layer is a mechanism that ties one transport layer association to several network interfaces at each communication host, which enabling the two end-nodes to exchange data over several network paths (a.k.a. *Load-sharing*). Therefore, supporting multi-homing for having concurrent multipath communication became an attractive part of network researches and technology finances.

Several researchers have tried to apply simultaneously all available communication paths in order to transmit the user data on all available paths [8][9][11]. They provide solutions for the problems of utilizing multiple paths. As long as the *load-sharing* operations are provided in the upper network layer (i.e. Application Layer), they would be more flexible in term of user application connectivity and developing environment, but they have to pass the cross layer connectivity and related overheads to provide the necessary paths and interfaces information for the upper layer applications. In the opposite direction, as long as the *load-sharing* mechanism is provided in the lower layers, its implementation will be more complicated but normally would be work faster. Transport layer is the lowest network layer which provides an end-to-end connection. Although transport layer multi-homing is an old idea for multipath *load-sharing* for having better flexibility at the time of network failure to control multi path communication issues, the general transport layer protocols TCP and User Datagram Protocol (UDP), cannot support multi-homing. Both UDP and TCP can use one network address in both end-to-end connected nodes [2] [3]. Two recent IETF transport layer protocols, the Stream Control Transmission Protocol (SCTP) [4], and the Datagram Congestion Control Protocol (DCCP) essentially support multi-homing at the transport layer[5].

Using SCTP, we investigate possible design for Concurrent Multi-path Communication (CMC-SCTP). Each interface of a multihomed node may have different round-trip-time. Sending packets on several paths will cause un-order arrival of data on the receiver side. Consequently, the receiver has to inform its correspondent node the un-arrived packets (i.e. packet on the fly). This force the receiver to send the unnecessary packet lost message and consequently triggering the fast retransmission data on the sender side. The CMC-SCTP aims to address the related problems. The needed algorithms and a new *status chuck* are introduced. Theses algorithms controls the unwanted traffic caused by unnecessary fast retransmission. The suggested *status chuck* is used to inform the receiver about the status of transmitted data on each path.

During the time, paths quality may be changed especially for wireless networks. Therefore, a dynamic mechanism to monitor the communication paths in order to measure their

quality parameters must be considered. The path selection method should choose the best possible paths. Inserting any un-qualified (e.g. big round-trip-time) path will only enlarge the end-to-end communication latency time. In addition, we design a two-layer path selection method for two-side path selection, which specifies the source and destination addresses to send the data chunks on the proper paths. We study CMC-SCTP at transport layer in a broad scheme. We think our mechanisms and related algorithms will introduce a base for multi-home-support in different transport protocols. This paper introduces and enhances transport layer mechanisms to use CMC-SCTP for the multi-home hosts to improve application throughput.

This paper is organized as follows: Section II describes in detail CMC-SCTP design. Section III specifies the receiver side control algorithms. Section IV provides a performance study for CMC-SCTP. Finally, Section V concludes the paper.

II. SCTP BASICS AND RELATED WORKS

SCTP was introduced to be an all-purpose transport protocol for message-oriented communications. SCTP is a novel transport layer network protocol, which supports multi-homing and is the lowest and nearest layer to the network layer (i.e. IP) which has the most accurate information about internet addresses and path connections. SCTP is a Connection-Orientated Protocol and has in common many characteristics with TCP. Two communication end-points must establish a "connection", which is known as *Association* in SCTP. It has been formed using a 4-way handshake, which protects the communication against the Denial-of-Service (DoS) attacks [6]. SCTP puts messages and control information into different chunks (data chunks and control chunks). A message can be fragmented over a number of data chunks, which bundled into SCTP packets. Between all-available paths, one will be chose as a primary path (the fastest one). Standard SCTP uses primary path for data transmission. Other paths will be used either when primary path fails or retransmissions lose data packets. Chunks can be a control or a data chunk. For the purposes of reliability and congestion control, each data chunk in an association is assigned a unique transmission sequence number (TSN) for the entire DATA stream (used in fragmentation for reassembly). A selective ACK chunk called SACK is used in SCTP. Receiver uses SACK chunks to inform the last received TSN and the group of missed data chuck as gaps [4]. SCTP's congestion control algorithms like TCP are based on RFC2581. Like TCP, SCTP uses three congestion control variables: receiver's advertised window (rwnd), sender's congestion window (cwnd), and sender's slow start threshold (ssthresh). According to [7], SCTP congestion control engages with the following problems; 1) Well-organized recovery after a packet loss is only possible if SCTP generates the optional Gap Ack Blocks in SACK chunks. 2) The SCTP fast retransmit method is open to being mistakenly triggered multiple times leading to under-utilization of the network during recovery and duplicate retransmissions of the lost packet. 3) The SCTP fast retransmit procedure must wait for half a window of data to be acknowledged before retransmitting a lost packet yielding a slow response to packet loss. These problems will be increased when all paths are aggregated and used simultaneously. The retransmission performance of the standard SCTP was studied in [11]. The authors explained retransmission through another path is not always beneficial regarding throughput of the standard SCTP.

The authors of [8] introduced Concurrent Multi-path Transfer mechanism based on SCTP for simultaneous transmission and tried to solve the problems related to congestion control by introducing three algorithms: (1) avoiding unnecessary fast retransmissions; (2) delaying selective acknowledgements appropriately; (3) allowing faster updates of the congestion window. These algorithms could not properly control the both-side path selection and unnecessary SACK transmissions at the receiver side. Another method is a bandwidth aggregation technique that is called LS-SCTP, introduces a new data chunk which carries a path identifier (PID) and path sequence number (PSN) for all transmitted data chunks on the different paths [9]. Based on our knowledge the previous solutions have no control on the both-side multi-path transmission control. Because, those solutions do not provide the necessary sender path information to the receiver side. The authors did not clearly address the control of unwanted retransmission packet due to different path delay. Moreover, their method raises some new security issues relate to data transmission at the time of path changeover.

III. CONCURRENT-MULTIPATH-COMMUNICATION SCTP (CMC-SCTP)

The SCTP association operations divided in two parts, controlling the connection paths (Control-Path) and controlling the communicated data over these paths (Data Path). Figure 1. shows a general view of the standard SCTP and its extended modules architecture. The Light gray boxes are the global SCTP modules and the standard SCTP association modules which are adopted for multipath communication. We added some new modules to the standard SCTP association modules to provide our new protocol CMC-SCTP. Our new extended modules are shown with gray color in the blow figure.

As you can see in the Figure 1. , the standard SCTP Control-Path has two base modules, SCTP controller and Path-Management. We extend the standard SCTP Path-Management module with two new modules; Multipath Status Controller and Multipath Rout Management to provide the path management functionality for multipath communication scenarios separately.

In the Data-Path section of SCTP association in Figure 1. , we extend the flow control and congestion control of standard SCTP to support multipath flow/congestion control. Since we aimed to transmit data over the all available communication paths, we also add a Traffic-Partitioning-Module to provide the load-sharing over the several paths. Traffic-Partitioning-Module uses the previously standard SCTP bundle/unbundle module plus our enhanced Splitting Module to support the needed tasks for partitioning the traffics over the several available active paths.

CMC-SCTP is a method for multi-path data transmission that uses the available paths for a synchronous transmission of data chunks. In order to inform the correspondent receiver node

about the status of data on each communication path we introduced a new chunk, called *status-chunk* (see section A).

CMC-SCTP maintains separately *Congestion Control* and *Flow Control* on each path to keep the both sides fair *load-sharing* integration.

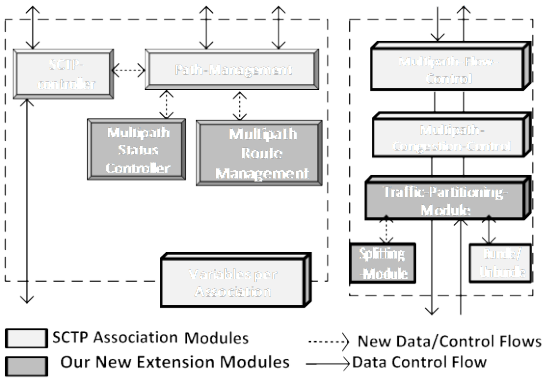

Figure 1. Standard SCTP and its Extended Modules Architecture

All data chunks in the standard SCTP are sent only on the primary path, but CMC-SCTP is able to communicate through all active interfaces between the two end-nodes in parallel. Communicating over multi-paths normally engages with unnecessary fast retransmission by the sender, and sending unnecessary selective acknowledgement (SACKS) by the receiver. This happens because different paths may have different flight time delays (i.e. RTT= 2*flight time) and cause a different order of received packets other than what they have been sent by the sender. A user messages may be divided into several data chunks and transmitted through several SCTP packets over the all available communication paths. Before any group of data chunks are sent on any path in a specified period of time, the status information of data chunks (i.e. Path Identifier, TSN, and the related flags) will be sent through our new chunk called *Status-Chunk* over the primary path. This status information helps the receiver to distinguish the range of transmitted data on each specific path in order to prevent the incorrect Gap report in the selective acknowledgement chunk. Sender sends the *status-chunk* on the primary path before any transmitted data chunk, to announce the path number before the arrival of data chunk in the receiver side. Primary path is supposed to be the best and fastest path (see Figure 2.). In case the primary path's RTT value is not too different to the other sender paths RTT, or several *status chuck* losses, the *status-chunk* of the transmitted data chunks can be transferred on the other communicated path.

Receiver establishes a dynamics *status table* to store the sender status information. The receiver inserts a new row in its *status table* for each received *status-chunk* on the primary path (see Table 1). The *status-chunk* informs the receiver about the specific path on which the data chunks (TSNs) was transmitted. The receiver uses this stored information to control and prevent sending unnecessary incorrect Gap reports toward the sender. In other words, receiver node with the help of *status table* does not report a TSN as a Gap when it is on the fly and has not been lost on the transmitted path. A timer will set for each transferred data, and then data chunks whose time enumerator reach the specified timeout, would be marked as a Gap and will be informed to be retransmitted.

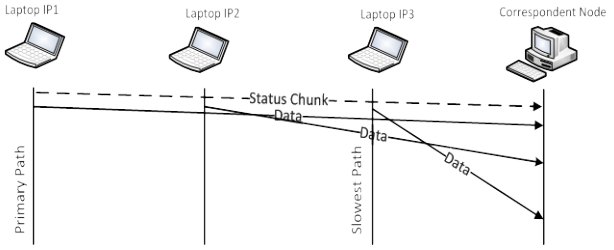

Figure 2. CMC-SCTP Message Diagram for Data and Status Chunk

Like *Sync-attack* in traditional TCP protocol, a possible malicious attacker may disuse the receiver's *status table* as a Denial-of-Service attack against any victim receiver node anywhere on the network. There are some solutions to prevent this kind of attacks; one may use these algorithms for the communication parties, which know each other due to a previous reason. The other solution may build based on a credit base algorithm, which starts to use CMC-SCTP algorithm after some data exchange. In fact, the receiver starts to make the *status table* if the sender acquired enough credit by the amount of data exchange in each association. Status will be kept only when at least one data chunk is exchanged. C. Vogt in [10] addresses the same related problem annihilation for Mobile IP6 protocol.

All *status-chunks* and possible retransmitted data chunks should be exchanged on the primary path to minimize the status update delay as well as *head-of-line-blocking* problems [13]. If sender did not receive any acknowledgement for any *status-chunk* after a specified time (e.g. 200ms according to the standard SCTP), it will retransmitted again on the primary path. The maximum delay time for any chunk acknowledgement, may be configured by the CMC-SCTP implementer. The maximum delay may be lower than 500 ms but must not be increased above 500 ms (according to the standard SCTP).

The standard SCTP does not isolate the flow and *Congestion Control* mechanisms, as it was designed to deal with a single path and both mechanisms use the same transmission sequence number. CMC-SCTP *Flow Control* is on each path basis to ensure fair integration with other traffic on the receiver side paths, and SCTP *Congestion Control* performs on each path separately, in order to ensure fair integration with other traffic on sender side paths. The Round Trip Time (RTT), congestion window size (*cwnd*), slow-start threshold (*ssthresh*), retransmission time out (RTO), etc. is applied as *Congestion Control* variables for each path separately. All the sender interfaces provide an independent congestion window size, which is equivalent to the combined of the *cwnd* from the all paths included in the association which are applied for CMC-SCTP. In addition, The *HEARTBEAT* chunks allow CMC-SCTP to monitor the reach-ability of idle destination addresses periodically. If the sender does not receive a *HEARTBEAT*-Ack chunk on any path, then it determines that the destination address is not available. In addition, CMC-SCTP monitors the status of the available paths

and as their status changes, i.e. new paths become active or existing paths break, it updates the active paths list (*Dynamic Address Reconfiguration*). When path failure increases, an association throughput will eliminate it from the active paths list. If the average loss rate and delay increase on any path to a given threshold, it will be inactivated. When the inactive path is recovered, it will be inserted to the active paths list and the sender starts to use it again in CMC-SCTP communication mode.

A. CMC-SCTP Status Chunk

In a multipath communication mode each path may has different transmission delay time, a receiver cannot detect whether the un-received data are missed or they are still on the fly on the other paths. Sender sends CMC-SCTP *status-chunk* to prevent the insertion of unnecessary Gap block in the receiver's selective acknowledgements (SACK) messages. The format of CMC-status-chunk is shown in Figure 3. . *Status-chunk* caries all the TSN ranges which are sent over one or many packets via the all communication paths in a specific time period. There are five new parameters in our status chunk. The latest Cumulative TSN Ack that the sender received through the received SACK is inserted within *Cumulative TSN* parameter in the related status chunk. *TSN Start (TS)* Block is containing the start offset of the TSNs range, which was sent on a given path in a specific period in any transmission. *TSN End (TE)* Block also is containing the end offset of the TSNs range, which was sent on a given path in a specific period in any transmission. In fact, Cumulative TSN plus each of these values will generate the start and end TSN value of this transferred block in this path (i.e. TS-TE range). In addition, CMC-SCTP continues to internally use *Status-chunk Sequence Number (SCSN)* parameter to assure sequenced delivery of the status chunks within an association. In order to inform the transmission path used for each packet transmission, the end point use *Path Identifier (PI)* parameter which is contains the transmission path identifier.

Type	Reserved	Length
Status Chunk Sequence Number(SCSN)		
Cumulative TSN		
TSN Start (TS) Block #1	TSN End (TE) Block #1	Reserved
TSN Start (TS) Block #2	TSN End (TE) Block #2	Reserved
.....		
TSN Start (TS) Block #N	TSN End (TE) Block #N	Reserved
Path Identifier(PI)		
Optional/variable parameter		

Figure 3. CMC-SCTP Status Chunk

According to our traffic-partitioning module, a specific data range (i.e. TS-TE range) has to be sent on a specific path. This data may be formed as one or more data chunks and can be fed in one or more SCTP packets. Receiver replays an Acknowledgment in response to each *status-chunk*. The *status-chunk-Ack* is very simple and only contains a header chunk and the *status-chunk sequence number*, which copied from the received *status-chunk* to show the transmitted Ack belongs to which received *status-chunk*. All other settings will follow the standard SCTP stated in [4]. Since the *status-chunk* is only sent on the primary path, the Ack chunk must be replied on the primary path as well. CMC-SCTP acknowledges all the chunks on the same received path. Receiver must immediacy reply the related Ack of each *status-chunk*. In order to decrease the waiting time on the receiver side because of *status-chunk* loss, the sender after 2*RTT delay (i.e. on primary path), upon the receipt of any SACK on any path, retransmits the status chunk. This will be done for 4 consecutive retransmissions and then the primary path will be supposed to be inactive and will be removed and will be changed.

B. Traffic-Partitioning Module

Optimal traffic partitioning specifies the amount of data percentage which should be assigned and dispatched over each communication path. The following equation calculates the data Division Factor (DF_i) that should be sent on each path:

$$Path_i DivisionFactor = PathWeight_i \Big/ \sum_{i=1}^{n} PathWeight_i = DF_i \qquad \text{(B.1)}$$

Suppose the Message Length (ML) is the length of the given message which should be sent to the correspondent party, therefore:

$ML*DF_1$	$ML*DF_2$	…	$ML*DF_n$

Figure 4. Message Dividing parts

1, 2, .., n are adopted to the number of active paths. Summation of the all above pieces will produce the whole message, in fact:

$$(ML*DF_1) + (ML*DF_2) + \ldots (ML*DF_n) = ML \qquad \text{(B.2)}$$

There are different possible ways for delegating the user data on different paths. The following methods can be used for different scenarios or policies.

- *Splitting modul*e: receives each message from association buffer and split the messages in regards to the different paths' factors (e.g. RTT, bandwidth and etc.). *Splitting module* may unnecessarily fragment a message. If part of a message in any path is lost or delayed, the rest of the message will be faced *head-of-line-blocking* problem. Therefore, a proper policy should be adopted for retransmission of the lost data in this method.
- *Multi-Level Priority Queues (MLPQ)*: configures a *multiple priority-queues* based on different class of paths. Each class gets a priority based on the path quality factors. As long as the higher priority queue has space, data will be transmitted on this path and data will not deliver to the lower priorities class of path.
- *Round Robin:* this packet distribution method forms only one packet (i.e. which can have the maximum of the path MTU) for each network path in each round-period. If an interface sender-buffer is full, it will skip to the next interface. Of course, the *round robin circulation* will start with higher priority interface to the lower one. Moreover, the round-robin will be reconfigured each time a SACK message or control message is received, which may be an announcement of new sender buffer size.

IV. STATUS-CHUNK HANDLING

We use Path Sequence Number (PSN) to facilitate their operation instead of using TSN. Accordingly, although the receiver provides the Cumulative TSN Ack Point but it is not used directly by sender in CMC-SCTP anymore. In fact, the Cumulative PSN Ack Point is used indirectly for each path according to the received Cumulative TSN Ack Point (i.e. for each path and not for whole association). The association follows the standard rules stated in [4] for TSN, whereas each path uses the PSN concept likewise. The Cumulative PSN Ack Point represents the sender's view of the highest PSN that has been received so far from a path at the receiver. It is worth noting that since sender uses a standard data chunk therefore, it sends TSN on toward its correspondent but has a map of TSN and PSN for each path locally.

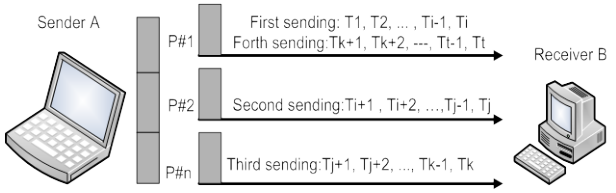

Figure 5. CMC-SCTP packet Load-sharing

Figure 5. displays an example to show our *status-chunk* behavior which is sent to the receiver on the primary path (P_p). In this example, four ranges of TSNs over the three paths have been sent. Table 1 shows our dynamic *status table* on the receiver, which is regulated based on Figure 5. . Each row will be added upon the arrival of each *status-chunk* on the receiver side. Table 1 assumes the related data chunks have not yet arrived to the receiver node. In case they arrive before their *status-chunk*, they will be buffered until the receipt of the related status chunk. This buffer will be kept only for a specified period of time (i.e. till the expected received time).

The CMC-SCTP receiver uses the algorithm to detect whether the delayed packets are on the fly via the other path or they have been lost. On a single path, the transmission and receiving sequence on both sides are usually the same. Of course except for any changes on the intermediate routing table in a short period or packet loss). This rationale is used to form our *unnecessary retransmission control* algorithm to eliminate the incorrect Gap in a SACK chunk. The first part of this algorithm updates the *status table* upon the receipt of any packet (see Figure 6.).

The unnecessary retransmission control algorithm uses two variables to check a data chunks reception on a specific path. They are one-bit flags and are called Internal Flag (IF) and General Flag (GF). The "IF" flag will be set over the related given *status table* row after the receipt of any data chunks between TS and TE. The "GF" flag will set on the related given *status table* row after the receipt of all data chunks between TS and TE. Figure 6. has two steps: the first step checks whether a CMC-status-chunk is received before or after its related data chunk. The second step checks whether a part or whole of the TSNs are received in TS-TE range.

Table 1: Status table

TS	TE	PI	IF	GF
T1	Ti	1	0	0
Ti+1	Tj	2	0	0
Tj+1	Tk	3	0	0
Tk+1	Tt	1	0	0

Figure 7. shows the second part of the *unnecessary retransmission control* algorithm which prevents the incorrect Gap report toward the sender. Based on *Congestion Control*, if a TSN arrives to the receiver after an un-arrived TSN on the same path, it indicates this un-arrived TSN is lost and can be marked for its retransmission. Therefore, for a range of TSNs (i.e. TS-TE) on a specific path, the same concept can be applied to report any un-arrived TSN with a bigger number after any received TSN as a Gap in its SACK chunk. Figure 7. checks whether any TSN number after any un-arrived TSN is received or not. This is done for each specific path separately. In the case of receiving any TSN after an un-arrived TSN on the same path, it will be reported as a Gap in SACK chunk.

1. For any receipt *status-chunk* **do**
i. If (TSNs between TS and TE have been arrived) **then**
/ *status-chunk* received after all of its related data chunks **/**
Begin
Insert TS, TE, PI, IF=1, GF=1 to the status table;
Go to Delete algorithm;
End
ii. Else
If (TSNs between TS and TE have not been arrived) **then**
/ *status-chunk* received after some of its related data chunks **/**
Insert TS, TE, PI, IF=1, GF=0 to the status table;
iii. Else
If (TSNs between TS and TE have not been arrived) **then**
Insert TS, TE, PI, IF=0, GF=0 to the status table;
2. For any receipt data chunk check status table **do**
/checking whether the part or whole of the data chunks receipt in TS-TE range received or not**/**
i. If (TSNs between TS and TE have not been arrived) **then**
Begin
Set IF=1, GF=1 for related row;
Go to Delete algorithm;
End
ii. Else
If (TSN between TS and TE have not been arrived) **then**

Figure 6. Unnecessary retransmission control Algorithm Part#1

If Tg is assumed to be a Gap TSN, Figure 7. step b checks if at least one TSN has arrived after Tg in the finding row. Then Tg is determined to be a loss chunk. In Figure 7. step c recaps the next transmission on the same path related to the same path identifier. If the aforesaid row is found, and IF or GF flag has been had set to one, it shows that the receiver has received a data chunk after the Tg in the next transmission on the same path and shows this Tg has been lost. Otherwise, the receiver waits for arrival of these un-arrived TSNs. Since the *status table* will be grown up during the time, an algorithm removes the unused-rows from the status table. In two cases, the receiver does not need any more to keep the aforesaid row

in the status table: 1. All data chunks in a TS-TE range are arrived (IF and GF have been set to one); 2. The loss chunks have been identified and reported as Gap blocks (Tg's TSN).

```
For any un-arrived TSN do
      i.    If  Tg's timer expired then {Tg is un-arrived TSN}
Let Tg is a loss chunk;
      ii.   Else
    Begin
      a.    For all table rows do
 If (TS<=Tg<=TE) then
Begin
            PathNumberTemp = PI;                    /**Tg's      path
identifier**/
MayNotEnd=TE;
Break ;                          /** go out of For a.**/
      End
  b. If(TSN between Tg+1 and TE have been arrived) then
/** at least one acked TSN of Tg+1 to TE was sent to the sender. **/
          Begin
              Let Tgis a loss chunk;
           Go to Delete Algorithm;
           End
c. Else
For this row to the last row  do
If [(PI==PathNumberTemp) and (GF==1or IF==1) and (TS>MayNotEnd)]
then
          /** at least one TSN arrived in the next transmission on path
          PI to the receiver and acked to the sender. **/
  Begin
                Let Tg is a loss chunk;
                Go to the Delete Algorithm;
                Break;
```

Figure 7. Unnecessary retransmission control Algorithm Part#2

V. PERFORMANCE STUDY

Our CMC-SCTP protocol extension has been simulated in the network simulator [12] and probed with different network configurations. The purpose of the extensive simulations is to inspect the transmission performance of the proposed path selections methods for CMC-SCTP with various network interfaces. We also compare our approach with the standard SCTP and LS-SCTP [9].

In our simulation, we created a network topology with two communication hosts where one side has 3 interfaces and the other side 2 interfaces. We supposed all intermediate link paths have the same path MTU; also, they tested with same packet losses to investigate the impact of different path quality and performance.

In our simulation, we evaluated the strength of the CMC-SCTP protocol in dynamic conditions. We assumed that we have 6 possible paths (i.e. 2*3 or *m***n*). Each time a maximum of 3 can be activated simultaneously (see below figure). The paths have different RTT between 40 and 160 ms. The sender paths have 1, 3 and 6 Mbps bandwidth while the receiver paths have 4 and 6 Mbps available bandwidth. In our performance study, we used the association throughput as a performance metrics, defined as an amount of data per-second which is delivered to the receiver's application-layer.

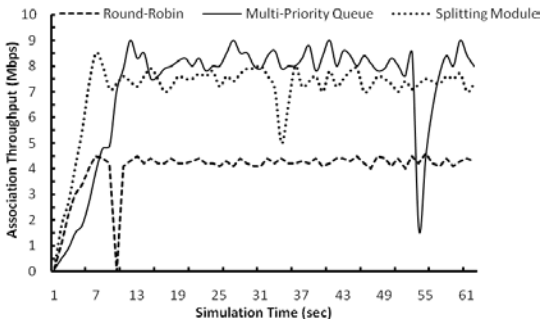

Figure 8. Big block messages transmission throughput

Figure 8. shows different path selection throughput during a big block of data messages transmission such as video files. The x-axis in Figure 8. represents the time, while the y-axis represents the effective association throughput. As can be seen from Figure 8. , MLPQ path selection has the best average throughput, but the increase reaches its maximum throughput a little slower than other methods.

The splitting module reaches its maximum throughput faster but in overall is slower than MLPQ. As we expected, the impact of slow line will degrade its whole association throughput in splitting and round-robin methods. We reconfigure the simulation with the same configuration, but with small blocks of application messages fed to the paths (see Figure 9.). All methods have a lower throughput compared to the last measurement with big blocks of data. In this simulation, all diagram results are unsteady, especially for the splitting module which has more oscillations. As we saw in both scenarios, the MLPQ had better overall throughput.

The different throughput of Multi-Level-Priority-Queue is compared with LS-SCTP and standard SCTP in Figure 10. . LS-SCTP has a gain better performance than standard SCTP, and relatively its behavior looks smoother, while the MLPQ method has far better performance, thanks to the use of path selection by considering the both sides' communication parties.

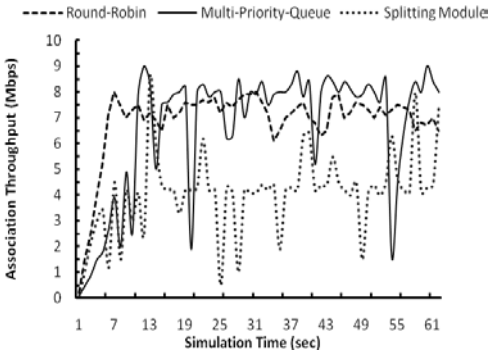

Figure 9. Small block messages transmission throughput

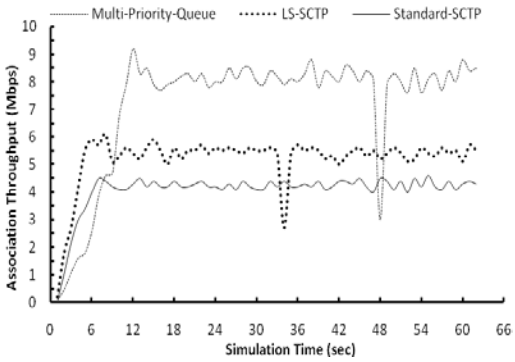

Figure 10. Different Methods transmission throughput

VI. CONCLUSION

SCTP support multi-homing. Multi-home nodes are the nodes which can be reached under several IP addresses. With the increase of multi-homed devices, we can simultaneously connected to different networks through different interfaces. In this paper, we suggest an extension for SCTP which is called CMC-SCTP. It can use all the available paths for simultaneous transmission between the sender and receiver. The standard SCTP is designed for single path communication and not suitable for simultaneous multi-path communication. Then we extend the SCTP to support concurrent multipath communication. The CMC-SCTP proposed modifying the sender to split the data on the available paths based on an estimate of the line weight. Furthermore we create the CMC-*status-chunk* for informing the receiver from transmitted TSNs rang and related path. The CMC-SCTP performance study shows that it benefits the use of simultaneous data transmission on all available paths. CMC-SCTP not only increased the communication path throughput but also controls incorrect gap report in the SACK chunks. We believe our proposal also benefits the mobile nodes on the overlapped wireless networks. A mobile node can be multi-homed when detects a new subnet and communicate still on its previous subnet.

REFERENCES

[1] Vinton G. Cerf and Robert E. Kahn, “A Protocol for Packet Network Intercommunication”, 1974 IEEE. Reprinted, with permission, from IEEE Trans on Comms, Vol Com-22, No 5 May 1974

[2] RFC 793, “Transmission Control Protocol”, Defense Advanced Research Projects Agency, Information Processing Techniques Office, September 1981.

[3] J. Postel, “User Datagram Protocol”, RFC 768, 28 August 1980

[4] Arias-Rodriguez, R. Stewart, A. Caro, K. Poon, and M. Tuexen. “Stream Control Transmission Protocol (SCTP) Specification Errata and Issues”. RFC 4460, April 2006.

[5] M. Handley, E. Kohler, and S. Floyd, “Datagram Congestion Control Protocol (DCCP)”, RFC 4340 March 2006.

[6] Zhongying Bai, Yuan Bai, “4-Way Handshake Solutions to Avoid Denial of Service Attack in Ultra Wideband Networks.”, Issue Date: 21-22 Nov. 2009, ISBN: 978-0-7695-3859-4

[7] M. Allman,V. Paxson, E. Blanton, “TCP Congestion Control” RFC 5681, IETF, September 2009.

[8] Janardhan R. Iyengar, Keyur C. Shah, Paul D. Amer. “Concurrent Multipath Transfer Using SCTP Multi-homing.” SPECTS 2004, July 2004.

[9] Ahmed Abd El Al*, Tarek Saadawi, Myung Lee, “a bandwidth aggregation technique for stream control transmission protocol.” Computer Communications, VOL 27, NO. 10, pp. 1012-1024, 2004.

[10] C. Vogt, “Credit-Based Authorization for Concurrent IP-Address Tests”, 2005

[11] A. Caro, R. Stewart and P. D. Amer, “Transport Layer Multi-homing for Fault Tolerance in FCS Networks”, Collaborative Technology Allicances (CTA), 2003.

[12] OPNET Technologies, Inc. A Leading provider of solutions for managing networks and applications, http://www.opnet.com/

[13] M. Scharf, S. Kiesel, “Head-of-line Blocking in TCP and SCTP: Analysis and Measurements.”, April 2007

[14] R. Stewart, M. Tuexen, G. Camarillo, Security Attacks Found Against the Stream Control Transmission Protocol (SCTP) and Current Countermeasures. RFC 5062 (Informational), September 2007