Comparative Analysis of TCP over Multi Streaming Protocol (SCTP) Based on Data Centers

Nandgaonkar Vikas N. Patil Chandrashekhar G. Patil Sonali C.

***Abstract:*-In this paper we are presenting the literature analysis over the Stream Control Transmission Protocol (SCTP) as well as performance analysis against TCP. We also identify overheads associated with FTP, attributed to separate TCP connections for data and control, non-persistence of the data connections, and the sequential nature of command exchanges. The SCTP protocol becomes the replacement option against the use of TCP protocol due to kind of reliable services offered by it. In addition to this, the mechanisms of reducing head of line blocking in hardware assist in order calculate the CRC32 and hence this makes this protocol superior communication protocol. During the literature we studied that SCTP protocol is having many strong features. Apart from these advantages of SCTP, still there are some limitations like TCP. Most of the work of SCTP is related to WAN. In this paper our analysis over SCTP is done with a perspective of data center.**

***Keywords:*-TCP, SCTP, FTP, Data Center , Multistreaming**

I. Introduction

The past decade has witnessed an exponential growth of traffic in the Internet, with a proportionate increase in Hyper Text Transfer Protocol (HTTP) [BFF96] and decline in File Transfer Protocol (FTP) [PR85], both in terms of use and the amount of traffic. The decline in FTP traffic is chiefly attributed to the inflexible nature of its interface and inefficiency in its end-to-end delay performance. Many network researches on data center related issues continue to focus on the front end. We are interested on the fabric needs of inside the data center. Data centers are main part of high performance computing and e-business. It offers web-based services and increased complexity or size of the data to be manipulated; performance and availability requirements of the data to be manipulated, these requirements of data centers continue to grow. Although TCP/IP & Ethernet are well entrenched in data centers, Here stack does not carry data traffic. Ethernet will continue to remain the technology of choice for the mass market because of a variety of reasons including entrenchment, incompatibility of IBA with Ethernet right down at the connector level, familiarity, commodity nature, etc. IP is expected to scale well into the future, there are questions about TCP for supporting data center. In network a high data rate 10 Gb/sec, protocol offload providing low overhead/latency and maturing of IP based storage such as iSCSI, TCP/IP/Ethernet is likely to emerge as the unified fabric carrying all types' traffic, possibly on the same "pipe". Yet, although IP is expected to scale well into the future, there are legitimate questions about TCP for supporting data center applications demanding very low latencies, high data rates and high availability/robustness. Although TCP's weaknesses are well known, the force of legacy makes substantial changes to it almost impossible. SCTP (control transmission protocol control transmission protocol) is a connection oriented transport protocol designed for running over existing IP/Ethernet infrastructure. This protocol shared some features with TCP particularly the flow and congestion control; it is designed to be a lot more robust, flexible and extensible than TCP. SCTP was originally intended as the transport of choice for inter-working of SS7 and VoIP networks and was therefore designed with a view to support SS7 layers (e.g., MTP2 and MTP3) which have a number of unique requirements. Many of these features are useful in the data center environment which makes SCTP a better candidate for data center transport. The lack of SCTP makes it easier to change the protocol or its implementation in order to fine tune it in data centers. It provides a preliminary look at SCTP from the data center along with the impact of some optimizations that we studied. Further in this paper, in section 2, gives brief about TCP, Section 3 overview of SCTP is provided as well as discussed the major differences between WAN and data centers from this protocol point of view. In next section 3, presents performance results of SCTP and TCP open source implementations in terms of their efficiency and presented reasons of poor performance of SCTP. Section 4: presenting the literature study over many optimizations results those are already made by various researchers. Finally, section: 5 concludes the paper.

II. Brief about TCP

Here continuity "patching" of TCP over last two decades have made TCP flow of congestion control in a hostile WAN environment. Before the discussion over SCTP, first we are discussing about TCP. TCP lacks in a number of areas including lack of integrity and robustness, poor support for quality of service, etc. Since TCP was never designed for use within a data center, some of its weaknesses become particularly acute in such environments. First we see brief overview of TCP. A TCP connection provides a reliable, ordered byte stream abstraction between the two transport end-points. These uses a greedy scheme to increase flow of bytes at a rate commensurate with the round-trip times (RTTs) until it cannot do. The basic flow of congestion control algorithm used by TCP is the window based AIMD (additive increase and multiplicative decrease) where the window is increased slowly as the network is probed for more bandwidth but cut down rapidly in the face of congestion.

III. TCP Versus SCTP

Both LTE and LTE-Advanced use either TCP or SCTP, to send messages between two peers. Since both protocols are connection-oriented, the connection between two peers has to be established before sending any command. The SCTP is

a transport protocol identified, which works at an equivalent level in the stack as TCP and UDP.
Compared to TCP and UDP, the SCTP is superior in functionality and more robust against the failures in the network connections. The purpose in employing SCTP is to provide an efficient and reliable signaling bearer. To achieve this, the SCTP provides appropriate congestion control techniques with fast retransmission in the case of packet loss and enhanced reliability. Furthermore, it provides extra security against blind attacks and enhances security feature, when connected on top of UMTS and other 3G systems with different operators.
When the functionalities of TCP and SCTP are compared, it is evident that the later provides two key features, multi-streaming and multi-homing, which lacks in TCP. In the SCTP domain, a stream is a unidirectional sequence of user packets to be distributed to upper layers. Consequently, bidirectional communication between two entities includes at least a pair of streams, one in each direction. The multi-streaming is the feature, from which the name of STCP is actually derived. It permits setting up several independent streams between two peers. In such a case, when a transmission error happens on one stream, it does not affect the transmission on the other streams.

Table 1 . Comparison of TCP/UDP/SCTP

Feature	TCP	UDP	SCTP
Connection oriented	Y	N	Y
Reliable transport	Y	N	Y
Preserve message Boundary	N	Y	Y
In-order delivery	Y	N	Y
Un-order deliver	N	Y	Y
Data checksum	Y(16bit)	Y(16bit)	Y (32bit)
Flow & Congestion Control	Y	N	Y
Multiple streams within a session	N	N	Y
Multi-homing Support	N	N	Y

In contrast, TCP only provides one stream for a given connection between IP peers, which may cause additional data transmission delay when packets dropped. When a transmission loss happens on a TCP connection, the packet delivery is suspended until the missing parts are restored. SCTP provides new services and features for IP communication. However, the TCP provides reliable communication service and the UDP provides unreliable service but neither TCP nor UDP can handle multi-homing or have the ability to send information to an alternate address, if the primary becomes unreachable. Many of the features found in TCP and UDP can be found in SCTP. Comparative results between SCTP, TCP and UDP are provided in Table 1.

IV. MULTI STREAMING SCTP

SCTP adopts congestion/flow control scheme of TCP except for some minor differences [3]. This makes SCTP not only "TCP friendly", but mostly indistinguishable from TCP in its congestion behavior. The negative side to this is QoS related issues and complexity as discussed later. However, SCTP does provide the following enhancements over TCP. We point these out primarily for their usefulness in a data center. 1. Multi-streaming: An SCTP association (or loosely speaking a connection) can have multiple "streams", each of which defines a logical channel, somewhat like the virtual-lane in the IBA context [5]. The flow and congestion control are still on a per association basis, however. Streams can be exploited, for example, to accord higher priority to IPC control messages over IPC data messages and for other purposes [7]. However, inter-stream priority is currently not a standard feature. 2. Flexible ordering: Each SCTP stream can be designated for an in-order or immediate delivery to the upper layer. Unordered delivery reduces latency, which is often more important than strict ordering for transactional applications. 3. Multi-homing: An SCTP association can specify multiple "endpoints" on each end of the connection, which increases connection level fault tolerance (depending on available path diversity). This is an essential feature to achieve five 9's availability that data centers desire [4]. 4. Protection against denial of service: SCTP connection setup involves 4 messages (unlike 3 for TCP) and avoids depositing any state at the "called" endpoint until it has ensured that the other end is genuinely interested in setting up an association. This makes SCTP less prone to DoS attacks. 5. Robust association establishment: An SCTP association establishes a verification tag which must be supplied for all subsequent data transfer. This feature, coupled with 32-bit CRC and heartbeat mechanism makes SCTP more robust. This is crucial within the data center at high data rates.
The key to SCTP's extensibility is the "chunk" feature. Each SCTP operation (data send, heartbeat send, connection init,) is sent as a "chunk" with its own header to identify such things as type, size, and other parameters. A SCTP packet can optionally "bundle" as many chunks as will fit in the specified MTU size. Chunks are never split between successive SCTP packets. Chunks are picked up for transmission in the order they are posted to the queue except that control chunks always get priority over data chunks. SCTP does not provide any ordering or reliable transmission of control chunks (but does so for data chunks). New chunk types can be introduced to provide a new capability, which makes SCTP quite extensible.

A) SCTP Environments Analysis

Data centers have a number of requirements that are quite different from those for general WAN. In addition, data centers require much higher levels of availability, robustness, flexible ordering, efficient multiparty communication, etc. which are crucial but beyond the scope of this paper. In a WAN environment, the primary concerns for a reliable connection protocol are (a) each flow should adapt automatically to the environment and provide the highest possible throughput under packet loss, and (b) be fair to other competing flows. Although these goals are still important in data centers, there are other, often more important goals, that the protocol must satisfy. Data centers are usually organized in tiers with client traffic originating/terminating at the front end server. The interior of the data centers portray multiple connected clusters, which is the main focus here. The key characteristics of these communications (compared with WAN) include: (a) much higher data rates, (b) much smaller and less variable round-trip times (RTTs), (c) higher installed capacity and hence less chances of severe congestion, (d) low to very low end to end latency requirements, and (e) unique quality of service (QoS) needs. These requirements have several consequences. First and foremost, a low protocol processing overhead is far more important than improvements in achievable throughput under packet losses. Second, achieving low communication latency is more important than using the available BW most effectively. This results in very different tradeoffs and protocol architecture than in a WAN. A look at the existing data center protocols such as IBA or Myrinet would confirm these observations. For example, a crucial performance metric for data center transport is number of CPU cycles per transfer (or CPU utilization for a given throughput). It is interesting to note in this regard that most WAN focused papers do not even bother to report CPU utilization. The data center characteristics also imply other differences. For example, the aforementioned fairness property is less important than the ability to provide different applications bandwidths in proportion of their needs, negotiated SLAs, or other criteria determined by the administrator. Also, the data center environment demands a much higher level of availability, diagonosability and robustness. The robustness requirements generally increase with the speeds involved. For example, the 16-bit CRC used by TCP is finadequate at multi-gigabit rates. Protocol implementations have traditionally relied on multiple memory-to-memory (M2M) copies as a means of convenient interfacing of disparate software layers. For example, in the traditional socket based communications, socket buffers are maintained by the OS separate from user buffers. This requires not only a M2M copy but also a context switch both of which are expensive. In particular, for large data transfers (e.g., in case of iSCSI transferring 8KB or large data chunks), M2M copies may result in substantial cost in terms of CPU cycles, processor bus BW, memory controller BW and, of course, the latency. Ideally, one would like to implement 0-copy sends and receives and standard mechanisms are becoming available for the purpose. In particular, RDMA (remote DMA) is gaining wide acceptance as an efficient 0-copy transfer protocol [8,11]. However, an effective implementation of RDMA becomes very difficult on top of a byte stream abstraction. In particular, implementing RDMA on top of TCP requires a shim layer called MPA (Marker PDU Alignment) which is a high data touch layer that could be problematic at high data rates (due to memory BW, caching and access latency issues). A message oriented protocol such as SCTP is by comparison can interface with RDMA much more easily. There are other differences as well between WAN and data center environments. We shall address them in the context of optimizing SCTP for data centers.

V. TCP VS. SCTP PERFORMANCE ANALYSIS

A major stumbling block in making performance comparison between TCP and SCTP is the vast difference in the maturity level of the two protocols. SCTP being relatively new, good open-source implementations simply do not exist. Two native-mode, non-proprietary implementations that we examined are (a) LK-SCTP (http://lksctp.sourceforge.net/): An open-source version that runs under Linux 2.6 Kernel, and (b) KAME (http://www.sctp.org): a free-BSD implementation developed by Cisco. We chose the first one for the experimentation because of difficulties in running the latter, lack of tools (e.g., VTuneOprofile, Emon, SAR) for free BSD, and more familiarity with Linux. LK-SCTP was installed on two 2.8 GHz Prestonia Pentium IV machines (HT disabled) -- one used as a server and the other as a client --with 512 KB second level cache (no level 3 cache) – each running R.H 9.0 with 2.6 Kernel. Each machine had one or more Intel GBGb NICs. One machine was used as a server and the other as a client. Many of the tests involved unidirectional data transfer (a bit like TTCP) using a tweaked version of the iPerf tool that comes with LK-SCTP distribution. iPerf sends back to back messages of a given size. iPerf doesn't have multi-streaming capability. Multi-streaming tests were done using a small traffic generator that we cobbled up. Before making a comparison between TCP and SCTP, it is important ensure that they are configured identically. One major issue is that of HW offloading. Most current NICs provide the capability of TCP checksum offload and Transport Segmentation Offload. (TSO). None of these features are available for SCTP. In particular, SCTP uses CRC-32 (as opposed CRC-16). We found that checksum calculation is very CPU intensive. In terms of CPU cycles, CRC-32 increases the protocol processing cost by 24% on the send side and a whopping 42% on the receive side! Quite clearly, high speed operation of SCTP demands CRC32 offload. Therefore, we simply removed the CRC code from SCTP implementation, which is almost equivalent to doing it in special purpose HW. TSO for SCTP would have to be lot more complex than that for TCP and will clearly require new hardware. Therefore, we disabled TSO for TCP, so that both protocols would do segmentation in SW. However, one discrepancy remains. The message based nature of SCTP requires some additional work (e.g., message boundary recognition on both ends) which TCP does not need to do. Also, the byte stream view makes it much easier for TCP to coalesce all bytes together.

A) Base Performance Comparisons

Table 2 shows the comparison between TCP and SCTP for a single connection running over the GBGb NIC and pushing 8 KB packets as fast as possible under zero packet drops. (SCTP was configured with only one stream in this case.) The receive window size was set to 64 KB and is more than adequate considering the small RTT (about 56 us) for the setup. The reported performance includes the following key parameters: (a) Average CPU cycles per instruction (CPI), (b) Path-length or number of instructions per transfer (PL), (c) No of cache misses per instruction in the highest level cache (MPI), and (d) CPU utilization. Not surprisingly, SCTP can achieve approximately the same throughput as TCP. SCTP send, however, @@@@@@ is 2.1X processing intensive than TCP send in terms of CPU utilization. The CPI, PL and MPI numbers shed further light on the nature of this inefficiency. SCTP is actually executing 3.7X as many instructions as TCP; however, these instructions are, on the average, simpler and have a much better caching behavior so that the overall CPI is only 60%. This is a tell-tale sign of a lot of data manipulation. In fact, much of the additional SCTP path length derives from inefficient implementation of data chunking, chunk bundling, maintaining several linked data structures, SACK processing, etc. On the receive end, STCP is somewhat more efficient (1.7X of TCP). This is because SCTP receive requires significantly less work beyond the basic TCP. The 8 KB data transfer case is presented here to illustrate performance for applications such as iSCSI where the data transfer sizes are fairly large and operations such as memory to memory copy substantially impact the performance. It is also instructive to consider performance v/s small transfer sizes (e.g., 64 bytes). In this case, packet processing overwhelms the CPU for both protocols (as expected); therefore, the key measure of efficiency is the throughput rather than the CPU utilization.

TABLE 2: 8 KB transfers, 1 CPU, 1 connection

Case	Total CPI	Path Len.	CPU Utility	Tput (mb/s)
TCP Send (w/o TSO, w/o Chksum)	93	16607	41.6	929
SCTP Send (w/o TSO, w/o Chksum)	2.94	60706	89.0	916
TCP Receive (w/o TSO, w/o Chksum)	3.89	20590	40.3	917
SCTP Receive (w/o TSO & Chksum)	3.92	35920	69.4	904

Again, TCP was found more efficient than SCTP, however the differences are very much dependent on receive window size and data coalescing as discussed below.

Since TCP is a byte-stream oriented protocol, it can accumulate one MTU worth of data before making a driver call to prepare the IP datagram and send. This is, in fact, the default TCP behavior. However, if the data is not arriving from the application layer as a continuous stream, this would introduce delays that may be undesirable. TCP provides a NO-DELAY option for this (turned off by default). SCTP, on the other hand, is message oriented and provides chunk bundling as the primary scheme for stuffing up an MTU. SCTP also provides a NO-DELAY option which is turned on by default. That is, by default, whenever the window allows a MTU to be sent, SCTP will build a packet from the available application messages instead of waiting for more to arrive.

B) Multi-streaming Performance

One of the justifications for providing multi-streaming in SCTP has been the lightweight nature of streams as compared to associations. Indeed, some of the crucial transport functionality in SCTP (e.g., flow and congestion control) is common to all streams and thus more easily implemented than if it were stream specific. Consequently, one would expect better multi-streaming performance than multi-association performance for the same CPU utilization.

Table 3: 64 B transfers, 1 CPU, 1 connection

Case	Tput (64 KB)	Tput (128KB)
TCP Send (w/o TSO, w/o Chksum)	72	134
SCTP Send (w/o TSO, w/o chksum)	66	100
TCP Receive (w/o TSO,w/o Chksum)	76	276
SCTP Receive (w/o TSO, w/o Chksum)	74	223

Since the streams of a single association cannot be split over multiple NICs, we considered the scenario of a single NIC with one connection (or association). However, to avoid the single NIC becoming the bottleneck, we changed the transfer size from the usual 8 KB down to 1.28 KB. Note that no segmentation will take place with this transfer size. We also used a DP (dual processor) configuration here in order to ensure that the CPU does not become a bottleneck. Table 3 shows the results. Again, for the single stream case, although both SCTP and TCP are able to achieve approximately the same throughput, SCTP is even less efficient than for the single connection case. This indicates some deficiencies in TCB structure and handling for SCTP which we confirmed in our experiments as well. Now, with SCTP alone, contrary to expectations, the overall throughput with 2 streams over one association is about 28% less than that for 2 associations. However, the CPU utilization for the 2 stream case is also about 28% lower than for the 2 association case. These observations are approximately true for both sender and receiver. So, in effect, the streams are about the same weight as associations; furthermore, they are also unable to drive the CPU to 100% utilization. This smacks of a locking/ synchronization issue. On closer examination, it was found that the streams come up short both in terms of implementation and in terms of protocol specification. The implementation problem is that the sendmsg() implementation of LKSCTP locks the socket at the beginning of the function & unlocks it when the message is delivered to the IP-Layer. The resulting lock contention limits stream throughput severely. This problem can be alleviated by a change in the TCB structure along with finer granularity locking. A more serious issue is on the receive end – since the stream id is not known until the arriving SCTP has been processed and the chunks removed, there is little scope for simultaneous processing of both streams. This is a key shortcoming of the stream feature and can only be fixed by encoding stream information in the common header, so that threads can start working on their target stream as early as possible.

VI. Conclusion and Future Work

So overall in this research paper, we have presented the literature study over SCTP over data center and WAN with respect to TCP. During the analysis, we presented many differences among data center and WAN environments as well as exposed some unsolved issues associated with SCTP. The protocol and implementation are main issues during these studies. From the implementation point of view, it becomes important to minimize memory to memory copies, simplify chunking data structures, reduce SACK overhead, forego chunk bundling for larger messages, simplify implement finer grain TCB locking mechanisms, and TCB structure. Whereas from the side of protocol, the major findings include re-architecting of streaming feature to maximize parallelism and to provide a simple embedded ACK procedure with SACK made optional. For the future study, we will focus on improving the performance of SCTP for multi streaming communication applications as compared to TCP. In addition to this, there are a few other issues already addressed by others, e.g., need for stream priorities [3] and simultaneous transmission across multi-homed interfaces [4] that will be very used within a data center.

References

[1] R. Seggelmann, M. Tuexen, and E. P. Rathgeb, “Stream Scheduling Considerations for SCTP,” in Proceedings of the 18th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Sep. 2010.
[2] R. Stewart, K. Poon, M. Tuexen, V. Yasevich, and P. Lei, “Sockets API Extensions for Stream Control Transmission Protocol (SCTP),” Internet Draft draft-ietftsvwg- sctpsocket-24, Oct. 2010, work in progress.
[3] R. Stewart, “Stream Control Transmission Protocol,” IETF RFC 4960, Sep. 2007.
[4] R. Stewart, I. Arias-Rodriguez, K. Poon, A. Caro, and M. Tuexen, “Stream Control Transmission Protocol (SCTP) Specification Errata and Issues,” IETF RFC 4460, Apr. 2006.
[5] R. Stewart, “Stream Control Transmission Protocol (SCTP) Remote Direct Memory Access (RDMA) Direct Data Placement (DDP) Adaptation”, Sept 2004.
[6] G. Regnier, S. Makineni, et. al., “TCP onloading for data center servers”, Special issue of IEEE Computer on Internet data centers, Nov 2004.
[7] R. Stewart and Q. Xie, Stream Control Transmission Protocol (SCTP): A Reference Guide. Addison Wesley, New York, 2001.
[8] P.T. Conrad and G.J. Heinz, “SCTP in battlefield networks,” MILCOM 2001, Washington, DC, pp. 289–295, Oct 2001.
[9] B. Sikdar, S. Kalyanaraman, K.D. Vastola, “Analytic models and comparative study of latency and steady state throughput of TCP Tahoe, Reno and SACK”, Proc. Of Globecom 2001.
[10] E.A. Heredia, C. Teng, and M. Ozkan, “Using multiresolution and multistreaming for faster access in image database broadcast,” 1998 International Conference on Image Processing, Chicago, IL, pp. 784–788, Oct 4-7, 1998.
[11] R. Simon, T. Znati, and R. Sclabassi, “XCAP: a multistream routing algorithm for multimedia traffic,” Third IEEE International Conference on Multimedia Computing and Systems, Hiroshima, Japan, pp. 104–107, June 17-23 1996.

Author’s Profile

Prof. Nandgaonkar V. N. received his B.E. and M.E. Degree in Computer Engineering from Amravati University and Pune University, India respectively. He is Head of Department , Computer Engineering in NMIET, University of Pune, India, He is also a life member of ISTE. His research interest includes study and analysis of different communication protocols and try to speed up the network

Prof. Patil C. G. working as Asst. Prof. at Siddhant C.O.E., Pune, University of Pune, India. He is also looking after the Academic Research Department to encourage the faculties and students for research work.

Prof. Patil S. C. received her B.E. in Computer Engineering from BATU. She is working as Asst. Prof. at Siddhant C.O.E., University of Pune, India.