International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 04 Issue: 03 | Mar -2017 www.irjet.net p-ISSN: 2395-0072

# CONCURRENT MULTI - PATH REAL TIME COMMUNICATION CONTROL PROTOCOL (CMPRTCP)

**[*1] Mr. Martin Paul Rufus Kumar K.,[*2] Mr. N.S. Rajanadan.,**

*[*1] M.Phil Research Scholar, PG & Research Department of Computer Science & Information Technology KMG College, Gudiyattam, Tamil Nadu, India*

*[*2]. Assistant Professor, PG & Research Department of Computer Science & Information Technology KMG College, Gudiyattam, Vellore, Tamil Nadu, India*

---

**Abstract** - ***In this thesis, a new transport protocol, the Concurrent Multi-Path Concurrent Transmission Control Protocol (CmpSTCP) is proposed. The proposed protocol has been designed to handle real-time streams (video and audio) over IP-networks. One of the key strengths of this protocol lies in its ability to intelligently exploit the availability of multiple paths between multi-homed hosts for Synchronized transmission of unicast Synchronized streams. This work describes the architecture and operation of CmpSTCP in detail. In addition, the limitations of currently used transport protocols in handling real-time streams are also discussed. These limitations of other protocols have played a vital role in the design process of the proposed protocol. Experiments to evaluate the performance of CmpSTCP against other protocols and the results obtained therein are also documented in this work. Results show that CmpSTCP is a best effort protocol that tries to maximize the amount of data that is successfully delivered to the destination in a timely manner under varying drop and delay conditions of the network.***

***Key Words:*** *synchronized Transmission Control, MAC, Transport Protocol*

## I. INTRODUCTION

Recent advances in digital networking technology coupled with the rapid increase in consumption of digital content over the intra / internet have placed a greater emphasis on bandwidth aggregation, network load balancing (NLB) and reliable communication. The challenges to be tackled only get bigger when considering real-time video/audio streams owing to the time-sensitive nature of real-time data. As stated in, application level end-to-end delays exceeding 250 ms affect data delivery of real-time streams leading to unintelligible real-time interaction from an end-user's perspective. Also, real-time data transfers cannot be compared with voluminous file data transfers because the idea is not to utilize the highest available network bandwidth for fast transmission but rather transmit data at the rate at which it is dispatched by the real-time source while ensuring minimal jitter.

Currently, real-time applications utilize the user datagram protocol (UDP) at the transport layer for their transmissions. UDP is connection-less and does not re-transmit packets, making it a lightweight protocol. Also, UDP does not take care of re-ordering packets arriving out of order at the destination. Applications using UDP have no knowledge of network status and hence may under-utilize available bandwidth or worsen the congestion in the network. The standard Transmission Control Protocol (TCP) on the other hand is connection-oriented, takes care of retransmission and does re-ordering of packets arriving out of order at the destination. Although TCP does have some knowledge of the network congestion status; TCP's retransmission to ensure that each and every packet does reach the destination is an expensive (time consuming) process for real-time streams. Retransmitted packets over networks with reasonable delays have little value at the receiving end in real-time applications such as Voice over IP (VoIP) and Digital Video over IP (DVIP) because of their late arrival. Also, in the process of retransmission of a set of data packets, newer data being dispatched from the source gets held up until the retransmission is complete [3]. Thus a cycle of constantly increasing delay in data delivery sets in during the length of the transmission which is unacceptable for real-time streams.

Another limitation of both TCP and UDP is their inability to probe for and utilize multiple paths if available between hosts equipped with multiple network interfaces (multi-homed hosts). TCP / UDP can bind to only one IP-endpoint at either end. Applications can however split data across multiple connections to enable multi-homed streaming. It is shown in that multi-homed streaming can improve quality of reception (Q) by 30% or more. While some studies on non real-time traffic have pro-posed multipath data transfer solutions at application layer and network layer, it has been clearly shown in that it is the transport layer that is best equipped with end-to-end

information and hence most suitable for positioning the multipath data transport capability.

## II. RELATED WORK

The Multipath State Aware Concurrent Multipath Transfer- Redundant Transmission (MSACMT-RT) algorithm observes the path status and assigns the path priorities before transmission. The observation is to identify a path that is expected to face failure (Weak Path). In order to support this weak path previous findings proposed a suitable path. Due to the dynamic nature of the internet traffic flow, the characteristic of the path also varies dynamically. Therefore, it is injustice to maintain the initially identified weak path as weak throughout the period of transmission. Thus in order to fulfil the dynamic nature of the internet path characteristics, the MSACMT-RT algorithm is reviewed periodically and the path priority is reassigned before re-scheduling the CMT. The MSACMT-RT reviewing considers the loss probability and additional unnecessary overhead.

This test finds the appropriate period when the MSACMT-RT algorithm should be reviewed. The experiment is carried out for various file sizes, and the resulting MSACMT-RT review period is ideal when reviewed after every ten successful transmissions. Since Internet setups have dynamic path characteristics, the SCTP's MSACMT-RT policy is investigated in challenging scenarios of equal and unequal number of interfaces with failure and non-failure conditions in each scenario. As the network load is uncertain, various robustness tests are performed on systems, with symmetric and asymmetric interfaces, to ensure and to exploit the multihoming benefits of SCTP. Extensive simulation studies have been performed using the University of Delaware's ns-2 SCTP/CMT module (ns-2 V2.29, 2005) (Caro and Iyengar 2006)..

| | Message Oriented | Multi-streaming | Bundling |
|---|---|---|---|
| UDP | Yes | | |
| TCP | | | |
| SCTP | Yes | Yes | Yes |
| SCTP-PR | Yes | Yes | Yes |
| SCTP (CMT) | Yes | Yes | Yes |
| cmpTCP | Yes | Yes | Yes |
| SCTP-PR (CMT) | Yes | Yes | Yes |

**Table 2.1: Protocol Feature Comparison Chart**

### 2.1 TCP

Transmission Control Protocol is the core protocol used on the internet for reliable transmission of data. TCP is categorized as a single-path, loss aware, reliable and fully ordered delivery transport (refer Table 2.1). TCP is stream oriented in nature which means that data is received by applications at the destination as a continuous stream of bytes without any demarcation at message boundaries. The strict ordering and reliability of TCP makes it extremely useful for lossless transfer of data from source to destination. However, there are problems when using TCP for real-time data (refer section 2.4).

### 2.2 Multi-path Transport

Protocols that belong to this family, bind to multiple IP endpoints at both the source and the destination ends (refer Fig. 2.1) at the beginning of a user session. Data transmission is along one or more paths connecting the multiple IP endpoints for the session. Some of these protocols also support addition of new IP endpoints as well as removal of IP endpoints when a session is in progress. SCTP, SCTP-PR, SCTP (CDT), SCTP-PR (CDT) and cmpTCP belong to this family of protocols.

### 2.3 Multi-path Non-concurrent Transport

These are protocols that establish multiple paths between source and destination but utilize only one path for transmission while reserving the rest for fail-over.

### 2.4 SCTP

The Stream Control Transmission Protocol (SCTP) was the first protocol of its kind that enabled multi-homed hosts to communicate via multiple paths. SCTP provides features such as sequenced delivery of user messages within multiple streams, optional bundling of multiple user messages into a single SCTP packet and network-level fault tolerance through supporting of multi-homing at either or both ends of an SCTP association.

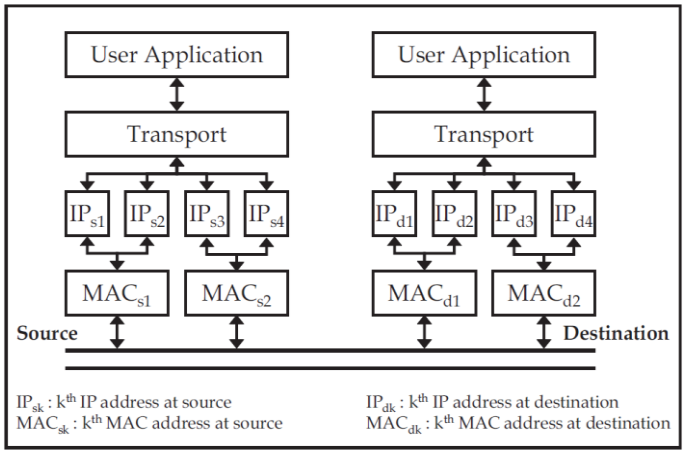


**Fig 2.1 Block Diagram of Multi-Path Transport provides**

On a multi-homed host, SCTP has the capability to tie down multiple IP addresses of the host to a common SCTP endpoint that can be used for data transmission or reception. This means that when an SCTP association is established between two multi-homed hosts, all potential network paths for data transfer are scouted for and kept track of by the protocol. SCTP's ability to scout for and establish multiple paths for communication has in fact made it the backbone for concurrent data transport protocols

## III. PREVIOUS IMPLEMENTATIONS

SmpRTCP establishes a multi-homed connection between the source and destination hosts in the same manner as cmpTCP (very similar to the mechanism in SCTP). The process of connection establishment is described in section 3.1.2. The entire design of smpRTCP beyond the connection establishment is based upon the simplistic goal that the sender must make a best effort to ensure that every packet / data chunk reaches the destination with no retransmission. For this purpose, the transport protocol at the sender must be equipped with

- A congestion window manager that continually tracks the network congestion status of the multiple paths that have been setup for concurrent data transport.
- A real-time scheduler that schedules packets over the multiple paths based on the inputs from the congestion window manager.

Similarly, the receiver must be equipped with the ability to aid the sender by informing it of

- Packets that are arriving late on particular paths
- Packets that have not shown up at all within a reasonable time limit.

This is of course in addition to the normal multi-path acknowledgements with gap reports (refer cmpTCP).

Medications for full memory protection. Rather than using contexts as the task mechanism, kernel-level threads are used. When the system is initialized, n such threads are created, all with a priority lower than that of the backup threads. In this scheme, the backup threads play a more important role than before. The backup threads share access to scheduling data structures. They select tasks to run on each core, and cause them to do so by forcing them to migrate to the appropriate core and setting their priority to be higher than that of the backup threads. Task completion is carried out when a task sets its priority back to the lower setting, returning control to the backup thread on that core. Timer-driven signals for releasing tasks are set up in a way that causes them to be delivered to the task running on the relevant core. When a task receives such a signal, it returns control to the backup thread on that core by lowering its priority.

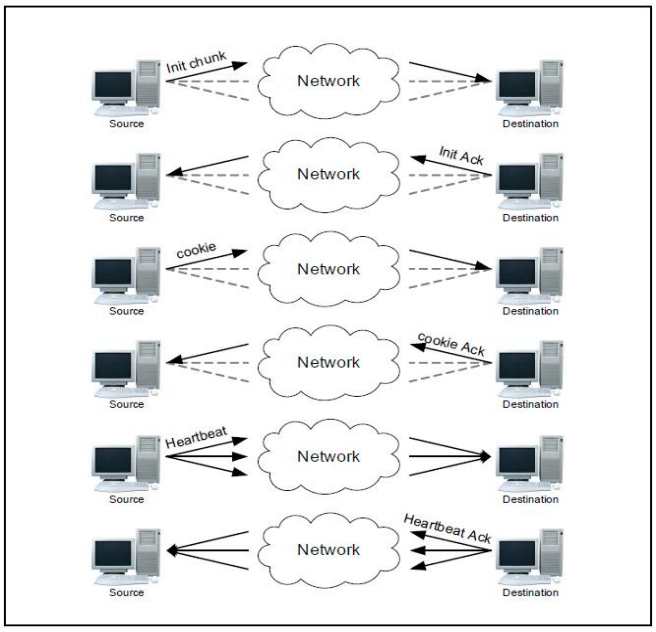


**Fig 3.1 Connection Establishment Process**

The connection establishment procedure is a 6-way handshake as shown in Fig. 3.1. The process can be carried out on top of IPv4 or IPv6 layers. Firstly, the sender sends a packet with a connection initialization data chunk (INIT chunk) which primarily contains information about the sender's multiple IP addresses. The receiver responds back with a connection initialization acknowledgement chunk (INIT ACK chunk) which contains the multiple IP addresses that the receiver is ready to accept data on along with a state cookie and a message authentication code (MAC).

### 3.1 Sender Design

Depicts the overall architecture of smpRTCP at the sender end. In addition to the two core modules (the congestion window manager and the real-time scheduler, the other modules that perform the relevant supporting roles are the Stream engine, the SACK processing module and the Packet dispatcher. The upper layer is allowed to send multiple parallel streams of real-time data for transmission through the established connection. In order to accommodate and manage the flow of the various data streams that need to be transported, there exists the stream engine which acts as a stream multiplexing, message fragmenting and time-stamping unit, creating data chunks out of the messages from the upper layer. The scheduler picks up these chunks queuing up in the transmission queue and chooses a path for dispatching them. The choice of path is based upon a heuristic that combines the following four factors (i) size of the congestion window of the path, (ii) outstanding bytes in flight (bytes that are awaiting acknowledgement) on the path, (iii) number of chunks that are appar¬ently missing (dropped / delayed) on the path, (iv) Round Trip Time (RTT) of the path. The window manager tracks the above factors to aid the scheduler.

## IV. SYSTEM IMPLEMETNATION

The proposed solution aimed to strengthen the key the scheduler as described previously (section 3.1.3) has the responsibility to load balance across the multiple paths based on their availability. Algorithms 1 and 2 illustrated in this section are two scheduling algorithms that were developed and deployed to understand the importance of good scheduling. The basic smpRTCP protocol uses the first algorithm shown while the second algorithm is used in cm- pRTCPa (a variant of SmpRTCP). These algorithms execute as atomic operations. New events from layers in the protocol stack above or below; queue up until the algorithm completes a full pass across all paths. From the algorithms, it can be seen that during every round, when a burst of packets arrive from the upper layer, smpRTCP fills up each available path to its full capacity (capacity of each path is determined by the window manager - 3.1.3.2) before moving on to the next path in a round robin fashion. Every successive round takes over from the path that was previously used if it was not filled completely in the previous round; the next path by round robin otherwise.

Nonetheless, the implementation given leaves critical shared scheduling data structures vulnerable to corruption. If one of these data structures were to become corrupted, it could cause the failure of all tasks in the system. The only way around this problem is to prevent any task application code from being able to write to these shared scheduling data structures. (We assume that the virtual scheduler runtime library code is trusted not to cause corruption; ultimately, some code in the system must be trusted to update scheduling data structures.) Below, we outline how the implementation from Section 3 can be modified to achieve this property. (There are many specific implementation tradeoffs that may be worth investigating in future work. Here, our concern is simply to show that our virtual scheduler mechanism can be modified to enable memory protection.) smpRTCP a the variant of smpRTCP, is based on the idea that information about the missing packets can not only be used to control the amount of data being dispatched on each path but also direct the decision control of choosing a path.

**Algorithm1: smpRTCP Scheduler**

For all i such that i is a valid path number do

If obpa (i) < cwnd(i) then

Transmit on path i until obpa (i) = cwnd(i);

If more data is pending transmission in queue then

Choose next path i;

End

End

If no path was available for transmission and data is pending transmission

Then

Find path j that has the minimum ratio of obpa (j)

Cwnd (j) over all paths;

Transmit one MTU of data on path j;

End

**Algorithm 2: smpRTCPa Scheduler**

**Data: PathSet** ← set of valid path numbers sorted in ascending order of the number of packets missing on the respective paths

Let j be the first path from PathSet;

While data is pending transmission do

If (obpa(j) < cwnd(j)) or (missing(j) + sentAlready(j) < missing(j +1))

Then

transmit on path j;

End

Else

Choose path j + 1;

If j + 1 is not a valid path number then

Break out of the loop;

End
End

The variant thus operates by sorting the paths in the increasing order of the number of packets missing on the respective paths and then choosing paths in that order, dispatches packets as much as their respective congestion windows would permit. In addition, smpRTCPa may dispatch data exceeding the path capacity if it finds that loss of all of that data in excess of the capacity still does not make that path worse than the next best.

**Idle Path**

**Condition:** No transmission on path p for duration of 1x ATO

Cwnd (p) = integral multiple of MTU(p)

- To ensure that minimum data is sent over lossy paths when multiple paths are actually available for transmission, the congestion window is allowed to shrink to a minimum of a single MTU.
- To prevent a sudden burst loss from immediately sealing the window, the time interval between successive collapses of the congestion window is chosen to be a single RTT of the corresponding path (as opposed to a fast retransmit phase that locks the window).
- If an incoming acknowledgment packet indicates new data received at the destination, then the amount of bytes corresponding to the data packets acknowledged is used for appropriately incrementing the window sizes of the corresponding paths on which those data packets were dispatched. On the other hand, if the incoming acknowledgment indicates data being flushed to the upper layer at the destination (refer section LTSNF), all unacknowledged data prior to the data flush indicated by the incoming acknowledgment, are considered lost and window sizes of paths on which the unacknowledged data was originally dispatched are collapsed appropriately.

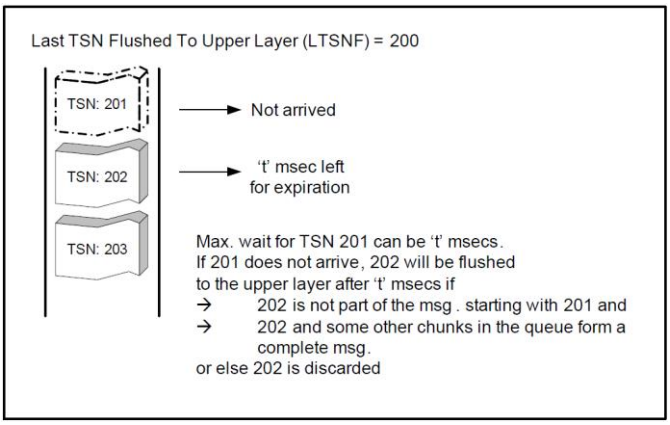


**Fig 4.1: Message Flushing Mechanism at Receiver**

When packets arrive at the destination over multiple network paths, a packet with a lower TSN may take longer over one network path while a packet with a higher TSN may have already reached the receiver buffer. The receiver cannot wait for the packet with the lower TSN indefinitely because the lifetime of the packet with the higher TSN in the buffer would run out. The maximum duration of waiting for a packet can only be as long as the time left for the first packet at the head of the receiver buffer to expire. After this duration, the first packet in the buffer is flushed to the upper layer and the sender is notified about the last TSN flushed to the upper layer via a selective acknowledgement (SACK) packet.

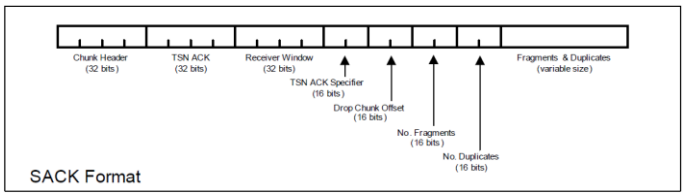


**Fig : 4.2 Format of SACK generated at the receiver**

The TSN ACK field can hold one of two values; either the cumulative TSN acknowledgment (CTSNA) or the last TSN flushed to upper layer (LTSNF). A CTSNA indicates that all packets up to and including the TSN in the TSN ACK field have been received at the receiver. An LTSNF indicates that the TSN in the TSN ACK field is the last TSN that has been flushed to the upper layer. To distinguish the CTSNA from the LTSNF, one bit of the TSN ACK Specifier field is used. If the receiver drops a late arriving packet, the TSN of the dropped packet is put into the Drop Chunk Offset field as an offset from the CTSNA / LTSNF.

.

**LTSNF Cross-Over Check**

Condition: → (Current LTSNF = Current CTSNA and Current CTSNA = Previous CTSNA) or (Current LTSNF > Current CTSNA)

Action:

Dispatch LTSNF in SACK

Make Previous CTSNA = LTSNF

If Condition is not satisfied, dispatch CTSNA in SACK

**Fig 4.3: CTSNA / LTSNF Crossover Algorithm**

### 4.3 Video Encoding and Real-time Payload Generation

All experiments presented here involved transmission of a 10.2 MB pre-encoded H.264 stream of the foreman video clip (2098 frames; YUV 4:2:0) at CIF resolution - 352x288 at 25 fps with a GOP structure (12,3) and average I, P and B-frame sizes of 16.56 KB, 6.1 KB and 3.14 KB respectively. The H.264 encoding was done such that every encoded frame would be sliced and packed into real-time payload (RTP) packets, the size of each packet being close to 1200 bytes (less than a single MTU of 1500 bytes). The distribution of frame sizes is shown in Fig. 4.2. Fig. 4.3 shows the frame sizes of the clip, sorted in descending order of frame size.

### 4.4 Round Trip Delay and Bandwidth Constraints

**Round Trip Delay**

Round Trip Delay for a path also referred to as RTT (Round Trip Time) is the total amount of time that it takes for a packet to travel from the source to the destination along that path and for the acknowledgement from the destination to return back to the source. Typical RTT between hosts within the United States range from less than 10 ms to as large as 100 ms. In all experimental scenarios described, unless otherwise stated, the round

trip delay has been set to 40 ms (20 ms in either direction).

The idea behind this experiment was to study the performance of smpRTCP over networks that exhibit a drop rate differential across multiple paths that are available for data transmission. While setting the packet drop rate on path I at 1%, the drop rate on path II was varied from 1% to 19%. Fig. 4.4 contrasts the effective loss rate (percentage of packets lost) between smpRTCP, smpRTCPa and an Application level UDP streamer.

## V EVALUATION RESULT:

Clear that as the drop rate differential across the paths increase, smpRTCP and its variant perform increasingly better than the others. When the drop rate on path II is set to 19%, it can be seen that smpRTCP shows a 60% improvement while its variant smpRTCPa, an even higher 80% improvement over UDP. The improvement can be attributed to the fact that smpRTCP uses the packet drop detection mechanism to control the amount of data flowing through each path. smpRTCPa takes it one step further and makes a best effort to choose a path with minimum number of missing packets during every round of transmission. The rapidly shrinking congestion window of the bad path with increasing drop rate differential is clearly seen in Fig. 4.5 and 4.6 for smpRTCP and smpRTCPa respectively

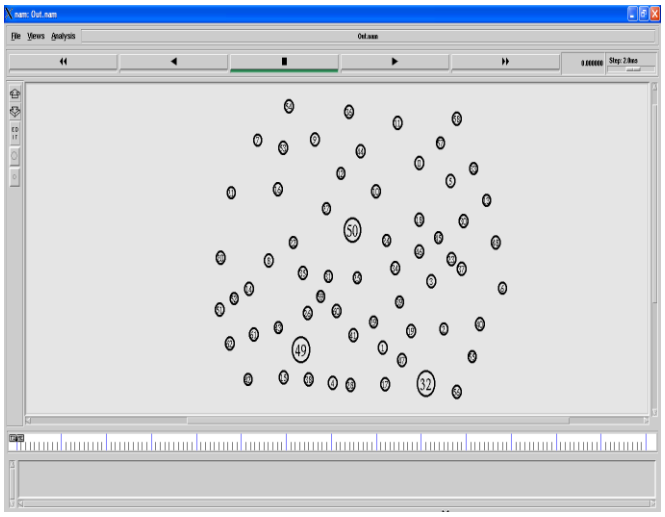

**Fig 5.1: Node Creation in NAM Editor**

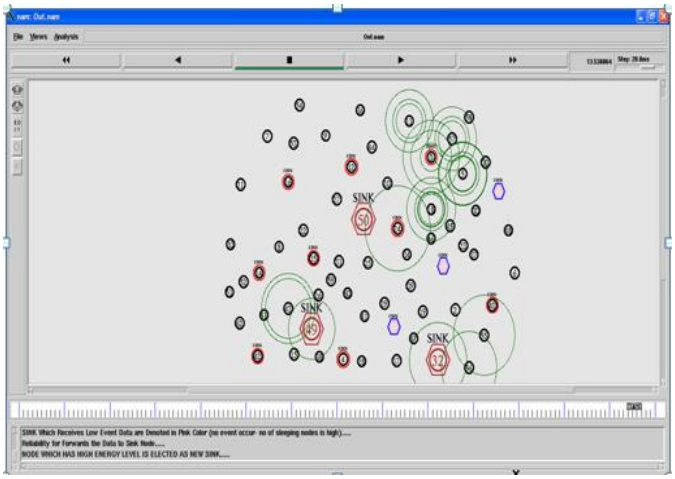


**Fig 5.2: Nodes Send Sensed Data to Respective Sink**

### 5.1 Packet Delivery Ratio

Many protocols in wireless sensor networks use packet delivery ratio (PDR) as a metric to select the best route, transmission rate or power. PDR is normally estimated either by counting the number of received hello/data messages in a small period of time, i.e., less than 1 second, or by taking the history of PDR into account. The first method is accurate but requires many packets to be sent, which costs too much energy. The second one is energy efficient, but fails to achieve good accuracy. A Sensor Network consists of multiple detection stations called sensor nodes, each of which is small, lightweight and portable. Every sensor node is equipped with a transducer, microcomputer, transceiver and power source.

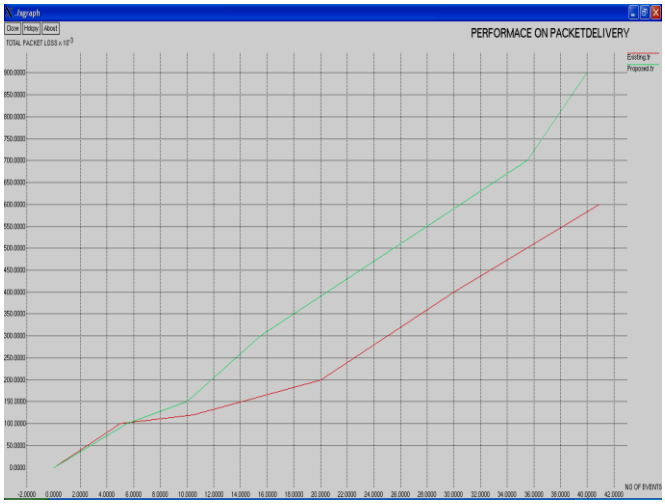


**Fig 5.3: Packet Delivery Ratio**

### 5.2 Packet Drop

A Wireless Sensor Network (WSN) is a collection of nodes organized into a cooperative network. Each node consists of processing capability which acts as transceiver. Packet dropping is a compromised node which drops all or some of the packets that is supposed to forward. Packet modification is a compromised node which modifies all or some of the packets that is supposed to forward. Packet dropping and modification are common attacks that can be launched by an adversary to disrupt communication in Wireless Sensor Network.

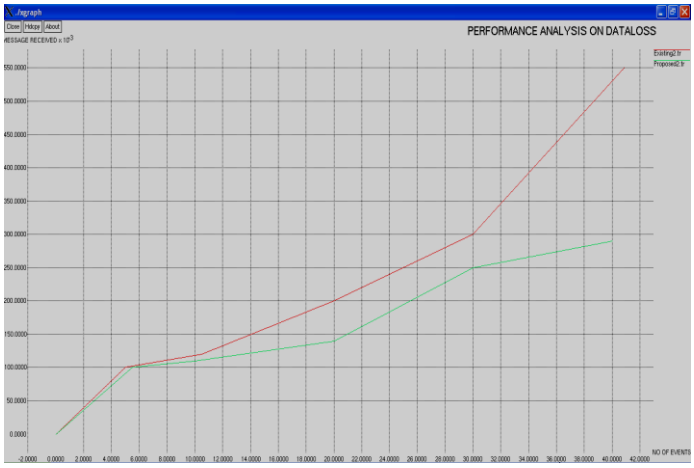


**Fig 5.4 : Packet Drop**

### 5.3 Throughput

Most of studies only consider that wireless sensor networks are equipped with only Omni-directional antennas, which can cause high collisions. It is shown that the per node throughput in such networks is decreased with the increased number of nodes. Thus, the transmission with multiple short - range hops is preferred to reduce the interference. However, other studies show that the transmission delay increases with the increased number of hops. Found that using directional antennas not only can increase the throughput capacity but also can decrease the delay by reducing the number of hops.

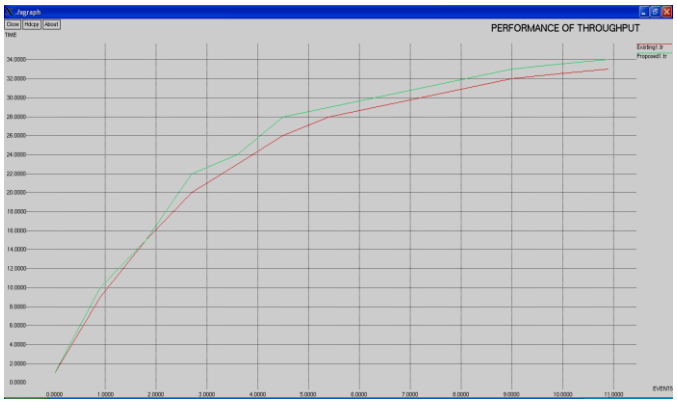


**Fig 5.5 : Throughput**

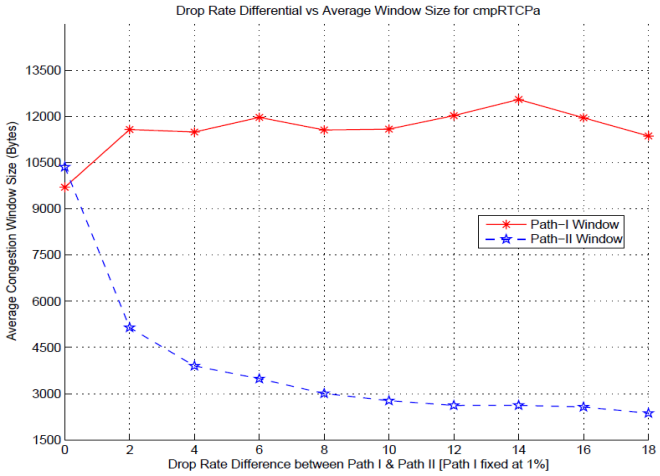


**Fig 5.6: Congestion Window Plot for smpRTCPa**

The transport layer fragments the messages from the upper layer into chunks that are no more than the size of an MTU, loss of a fragment would render the rest of the fragments of a message useless to the upper layer at the receiving end. This raises a question about the plot in Fig. 5.6. Is it possible to infer the loss rates in terms of bytes? It is for this reason that the RTP packet sizes were restricted to less than a single MTU during their generation. This prevents the problem of fragmentation and also helps achieve an almost perfect correlation between the percentage of bytes lost and percentage of packets lost as shown in Fig. 5.7 (correlation coefficient = 0.998). Hence, the effective loss rates shown in Fig. 5.6 are the same irrespective of the metric (lost bytes or lost packets).

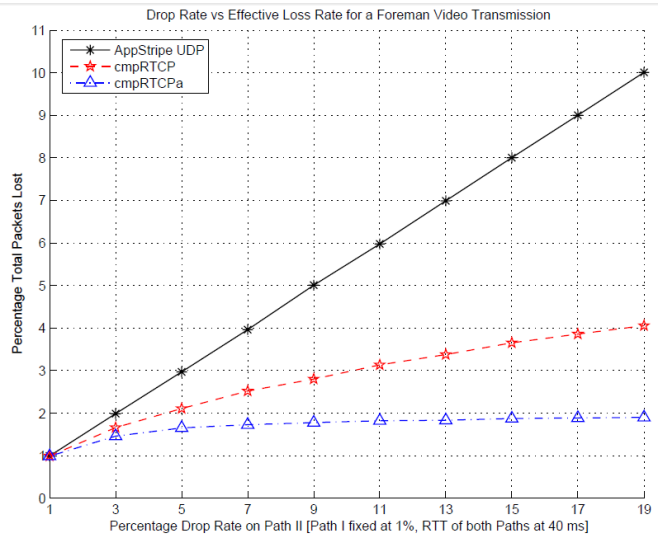


**Fig 5.7: Comparison of Effective Loss Rates for Drop Rate Imbalance**

The idea behind this experiment was to study the performance of smpRTCP over networks that exhibit a drop rate differential across multiple paths that are available for data transmission. While setting the packet

drop rate on path I at 1%, the drop rate on path II was varied from 1% to 19%. Fig. 5.7 contrasts the effective loss rate (percentage of packets lost) between smpRTCP, smpRTCP and an Application level UDP streamer.

## CONCLUSION

The primary purpose of this work was to come up with a robust transport protocol for transmission of real-time streams between multi-homed hosts. In this paper, the advantage of having a protocol that makes use of a TCP like congestion controller work in conjunction with a packet scheduler to achieve substantial gains has been clearly highlighted. Studies and experiments have shown that this protocol is indeed capable of performing very well when streaming real-time video over IP-networks with fixed as well as varying drop rate and delay characteristics. Some of the important tasks ahead include performing an exhaustive study of the protocols performance under dynamic conditions of varying bandwidth, extending the protocol to support retransmission of select packets under application request, replacing the AIMD congestion controller with a more sophisticated bandwidth manager and developing an analytical model for the protocol.

Although we evaluated Round-Robin and Weighted Round-Robin scheduling mechanisms, there are several other mechanisms being developed by many researchers. They need to be implemented and evaluated on the real-time test bed setup. There are various other parameters like delay, jitter etc. which can also be measured and evaluated in these scenarios. In this work, 4G USB modems of the same type at the source and a single-homed destination were used for the implementation. However, investigations can be made making use of different wireless networks and multi-homed destination in real-time scenarios which might bring more challenges in configuring.

## REFERENCES:

1 A. E. Al, T. Saadawi, and M. Lee. LS-SCTP: A bandwidth aggregation technique for stream control transmission protocol. Computer Communications, 27(10), 2004.

2 E. Al, T. Saadawi, and M. Lee. A Transport Layer Load Sharing Mechanism for Mobile Wireless Hosts. In IEEE PerCom 2004 , 2004.

3 Argyriou and V. Madisetti. Bandwidth aggregation with SCTP. In IEEE Globecom, San Fransisco, CA, Dec 2003.

4 S. Bangolae, A. P. Jayasumana, and V. Chandrasekar. TCP-friendly congestion control mechanism for a UDP-based high-speed radar application and character-ization of its fariness. In ICCS, Singapore, Nov 2002.

5 J. But, U. Keller, and G. Armitage. Passive TCP stream estimation of RTT and jitter parameters. In LCN '05: Proceedings of the The IEEE Conference on Local Computer Networks 30th Anniversary, pages 433-441, Washington, DC, USA, 2005. IEEE Computer Society.

6 M. Carson and D. Santay. NIST Net: a Linux-based network emulation tool. ACM SIGCOMM Computer Communication Review, 33(3):111-126, 2003.

7 Casetti and M. Meo. Westwood SCTP: Load balancing over multipaths using bandwidth-aware source scheduling. In IEEE Vehicular Technology Conference, 2004.

8 J. Day and H. Zimmermann. The osi reference model. Proceedings of the IEEE, 71(12):1334-1340, Dec. 1983.

9 M. Fiore and C. Casetti. An adaptive transport protocol for balanced multihom¬ing of real-time traffic. In Globecom, 2005.

10 Habib and J. Chuang. Multi-homing media streaming. In IPCCC, 2005.

11 S. Institute. Transmission Control Protocol (TCP). RFC 793, September 1981.

12 Iyengar, P. Amer, and R. Stewart. Concurrent multipath transfer using trans-port layer multihoming: Performance under varying band-width proportions. In MILCOM, Monterey. CA, Oct 2004.