Scientific Journal Impact Factor (SJIF): 1.711

e-ISSN: 2349-9745
p-ISSN: 2393-8161

International Journal of Modern Trends in Engineering and Research
www.ijmter.com

Congestion Control for Streaming Data Broadcasting over Internet

Ashish Parejiya[1], Dr. Vinod Desai[2]
[1]*Ph.D Research Scholar-CSS, Mewar University, Rajasthan, India,*
[2] *Computer Science Department, Govt. Science College, Navsari, India*

Abstract- In the last few years, video streaming facilities over TCP or UDP, such as YouTube, Face-time, Daily-motion, Mobile video calling have become more and more popular. The important challenge in streaming broadcasting over the Internet is to spread the uppermost potential quality, observe to the broadcasting play out time limitation, and efficiently and equally share the offered bandwidth with TCP or UDP, and additional traffic types. This work familiarizes the Streaming Media Data Congestion Control protocol (SMDCC), a new adaptive broadcasting streaming congestion management protocol in which the connection's data packets transmission frequency is adjusted allowing to the dynamic bandwidth share of connection using SMDCC, the bandwidth share of a connection is projected using algorithms similar to those introduced in TCP Westwood. SMDCC avoids the Slow Jump phase in TCP. As a result, SMDCC does not show the pronounced rate alternations distinguishing of modern TCP, so providing congestion control that is more appropriate for streaming broadcasting applications. Besides, SMDCC is fair, sharing the bandwidth equitably among a set of SMDCC connections. Main benefit is robustness when packet harms are due to indiscriminate errors, which is typical of wireless links and is becoming an increasing concern due to the emergence of wireless Internet access. In the presence of indiscriminate errors, SMDCC is also approachable to TCP Tahoe and Reno (TTR). We provide simulation results using the ns3 simulator for our protocol running together with TCP Tahoe and Reno.

Keywords- Streaming Broadcasting application, Congestion, Streaming Media Congestion Control protocol (SMDCC), TCP Tahoe and Reno (TTR), Quality of Service, End-to-end protocols,

I. INTRODUCTION

TCP has effectively maintained data applications through end-to-end consistent in-order packet communication. With increase in link bandwidth above the past period many interactive program broadcasting applications that stream video and audio over the Internet have emerged. The popularity of such type of applications has produced multimedia data to be progressively present in the Internet. Protocols need to be advanced that for equally share network bandwidth between multimedia and other connection types such as TCP, UDP, etc.

Multimedia is regularly not transferred using the uncontaminated TCP standard because the services providing by TCP are more helpful to applications requiring dependable data transfer, such as the HTTP, E-mail, and FTP. The latter applications place greater priority on reliable delivery, rather than the data transfer time. In contrast, the video streaming application has severer transfer potential requirements. The key specific characteristic of such applications is that audio and video data needs to be nonstop played out at the client machine. This connections a play out time with each transmitted data packet. If the packet arrives late, it is unusable to the application, and might as well have been missing. A streaming data transfer rate that does not fluctuate radically is therefore more suited for multimedia communications.

SMDCC approximate the bottleneck bandwidth segment of a connection at the client machine (receiver machine) like Smart Phone, laptop, tablet, etc. by algorithms similar to those announced in TCP Westwood [4]. The receiver machine does not send response acknowledgement to the broadcaster for every one received stream data packets. Instead, negative responses acknowledge (NRACK) is returned to the broadcaster when the client perceives a data packet loss. This NRACK serves two resolutions. First, it is a request for retransmission of the data lost packet. Depending on whether it can be delivered on time the broadcaster may or may not retransmit the packet. Secondly, in SMDCC, a NRACK brings to the broadcaster the current Bandwidth Share Estimate (BSE). Thus a NRACK message conveys a measure of congestion, and the broadcaster adjusts its sending rate accordingly.

At always time being0, the sender simulators TCP's congestion avoidance stage by increasing its data sending rate by one packet per round trip time (RTT) until a NRACK message is received, upon which the sending rate is set to the BSE. After this modification in the sending rate, the server resumes a linear sending rate increase of one packet per RTT.

As will be exposed in the remainder this paper work content, each SMDCC sender gets an accurate estimation of the connection's fair-minded share of the bottleneck bandwidth, and effectually fine-tunes the data sending rate to the changing estimate. This enables the receiver to continuously play out the delivered media content, although at fluctuating quality, even further down highly dynamic network conditions. SMDCC is also a fair protocol in that it shares the bottleneck bandwidth equally between a set of SMDCC connection networks. As a final point of view, SMDCC performs well in random loss in background environment, since the bandwidth approximate is strong to sporadic data packet loss, unlike TCP Reno-like adaptive schemes, which tend to overdramatize to random errors with substantial data throughput degradation. Furthermore, SMDCC is TCP friendly when packet losses are due to random errors.

The remainder of this research paper work as follows: Part 2 summaries related work. Part 3 presents SMDCC, the streaming protocol proposed in this research paper. This is followed by an assessment of SMDCC bandwidth estimation accuracy, throughput performance, objectivity, and its responsiveness to TCP, all in Part 4. Part 5 concludes by summarizing outcomes and discussing existing and thinks about future research work.

II. CORRELATED WORK

A latest research has target on designing TCP responsive protocols that meet the demands of multimedia streaming applications for video streaming facilities over TCP or UDP, such as YouTube, Face-time, Daily-motion, Mobile video calling have become more and more popular. In many cases these efforts have produced multimedia streaming protocols that behave very close related to TCP.

Time-lined TCP (TL-TCP) [11] is one such protocol that follows exactingly to the TCP congestion mechanism for streaming data application of broadcaster, while allowing data to be linked with target deadline or receiver end. Once the deadline of a unit of data has distributed the rest of the data linked with that

Broadcast a Streaming Data Congestion Control using Bandwidth Estimation

Targeted or receiver machine is dropped a stream data. Although this protocol rejects some of the timing duration issues introduced by TCP, it still stands by to the dependability and in-order distribution constraints of TCP, which hinder the transfer of streaming data.

SCTP [7] get used to the basic mechanisms of TCP congestion controlling to multimedia stream of traffic necessities. SCTP allows the disabling of in-order packet transfer and reliable

packet distribution, in the times of network congestion. However, SCTP still employs the slow-start mechanism of TCP, foremost to detrimental sending rate fluctuations in streaming data for broadcasting.

RAP [13] also simulators the additive increase multiplicative decrease congestion system of TCP. While in congestion escaping point, the sending or broadcasting rate is increased by one per round-trip-time. Upon detection of congestion, the broadcasting rate is multiplicatively decreased. RAP filters out the throughput difference level, caused by the multiplicatively sending rate reductions, by depend on receiver machine buffering strategies.

Feaster et al. [3] cover the RAP protocol by using non-AIMD algorithms end to end receiver buffer management to further decrease sending rate fluctuations. Their SQRT algorithm in specific progresses throughput differences by decreasing the sending rate by sort (window size), as opposite to AIMD algorithms where the decline in broadcast transmission rate is related to the window size.

TCP Friendly Rate Control (TFRC) [6] adjusts the server's broadcasting data rate upon bottleneck according to a TCP throughput calculation. This equation formula contains packet loss rate (due to congestion) and RTT, which are observed and maintained at the receiver machine. TFRC however cannot control random losses in the sense that such losses will cause unnecessary loss of quantity stream data.

One key design difference between SMDCC and the protocols mentioned overhead is that SMDCC does not require the receiver data packet acknowledgement of every data packet sent by sender of broadcaster. This is an important saving since multimedia data packets are typically small in size, ~200 bytes. Requiring a 40byte acknowledgment for each data packet automatically adds a 20% overhead to the protocol of SMDCC.

A new issue has been spoken by Tang in the Rate Control Scheme (RCS) and proposed by him. [13] Is robustness of the rate adaptation algorithm to high link loss rates on wireless channels with high Bandwidth X Delay products. Such cases correspond to satellite links as well as wireless access links to remote Internet servers. This scenario is likely to attract considerable interest as wireless Internet access to multimedia services is gaining popularity. The RCS scheme is a clever mechanism based on dummy packets to distinguish between congestion loss and random loss. The proper operation of RCS, however, requires the implementation of a priority level "below TCP best effort" in the router. Such support is not provided by most Internet routers. SMDCC on the other hand provides robustness to random errors by virtue of the intrinsic insensitivity of the Bandwidth Share Estimate to isolated losses. No extra network level support is required.

III. STREAMING MEDIA DATA CONGESTION CONTROL PROTOCOL (SMDCC)

In this paper we are propose the SMDCC protocol, 1) is adept to adept the sending stream data rate according to the linked estimated bandwidth on sharing base, 2) is fait to existing connection, and 3) does not suffer from pronounced data sending rate fluctuations data send typical of most window protocols.

There is no congestion window in SMDCC, although SMDCC adjusts its sending rate so as to simulator TCP's congestion escaping phase with its linear bandwidth analytical,

i.e. one extra packet is sent via broadcaster application on per round trip time so long as no congestion is identified. When congestion is encountered in SMDCC, the sending data rate is going down to the current Bandwidth Share Estimate (BSE), and linear probing is continuous. Never is the sending data rate dropped to 1 packet per round trip time following a timeout, as in the TCP protocol, unless the existing BSE dictates this summaries value. In other words, the "slow-start" phase with exponential progress as in TCP is not present in SMDCC.

3.1 SMDCC Broadcaster or sender Procedure

In this protocol, after getting acknowledgement connection setup using a three-way-handshake between SMDCC broadcaster server and receiver machine is complete, the broadcaster starts to send stream media content. In this execution process, we assume the network can handle the lowest quality data sending rate, and use that rate as the primary sending rate. The preliminary round trip time approximation is gathered from the handshaking connection process in between broadcaster and receiver machine. After each sequential RTT, the broadcaster increases the speed of one by one data packet sending rate. This has the consequence of mimicking TCP congestion escaping phase. The broadcaster settles in its RTT approximation each and every round trip time, by requesting a responds acknowledgement for the first data packet in every new RTT window.

At whatever time the broadcaster receives a NRACK, it arrange differently the sending rate to the BSE contained in the NRACK message (the BSE calculation is discussed below), and resumes linear inquisitive. The broadcaster can control if there is enough time to retransmit the lost packet, based on information it gets from the receivers machine regarding the receiver's current buffer and current play out time.

3.2 SMDCC Client or Receiver Machine Procedure

As we know for all data packet received, the receiver machine may recalculate the BSE. Upon getting a data packet that the broadcaster requests to be acknowledged, the receiver sends an ACK for that received data packets. As said above, this message interchange is used to set the RTT at the broadcaster side. Once a lost or dropped packet is identified, a NRACK message is sent enclosing the current bandwidth approximation. As said above, it is responsibility of broadcaster to control if the packet should be resending.

3.3 Bandwidth Calculation in SMDCC

Just as in TCP Westwood, the Bandwidth Sharing Approximation, i.e. the approximation of the data rate to be used by the broadcaster in order to share the bottleneck bandwidth fairly, is resolute by exponentially be an average of data rate samples. Though, where TCP Westwood measures ACK comings at the broadcaster, SMDCC uses the inter-arrival time between two subsequent packets at the receiver machine (Client) to calculate a data rate sample on Streaming Media Data Congestion Control using Bandwidth Approximation.

The onward path of the linked connection. The benefit of the SMDCC approach is that it measures the data rate on the forward path of the linked connection, clarifying out the effects of congestion on the same reserve path.

Fundamentally, the receiver side calculates the connection bandwidth share using the Receiver Based Packet Pair (RBPP) method described in [12]. RBPP requires the use of two consecutively sent packets to determine a bandwidth share sample. The adherence of SMCC to the RBPP sequence number requirement is fundamental for not overestimating bandwidth share and thus enhancing fairness and friendliness of SMCC.

A second another restraint is to make sure that 'time compression' has not happened on the successive packets. Here we try to define time compression as stirring when the difference between arrival times of the packets is less than the difference between sending times of the packets [12]. If the packets are time compressed, they are not used in the BSE calculation method as the resulting approximate would be too destructive. The intent is to estimate bandwidth only up to the server's known as broadcaster instantaneous sending data packet rate, even if more bandwidth is available.

If two successively received packets have passed the two tests above, they are used to calculate a bandwidth sample as follows:

$$\text{Bandwidth} = x_2 / (a_2 - a_1) \quad (1)$$

Where x2 is the size of the second packet, a1 and a2 are the arrival times of the first and second packet individually. This bandwidth taster is then used just as the data packet rate estimation in TCP Westwood; it is plugged into the exponential strainer to produce the current BSE.

IV. EXPERIMENTAL OUTCOMES

We had executed several simulation experiments to assess the performance of SMDCC with respect to relevant output, equality, and TCP-friendliness. Unless otherwise stated, all simulations test result use the topology shown in Figure 1, thru all linked connections going through the same bottleneck link using RED line management, and broadcasting data in the same track of destination of receiver machine. Simulations were run for 200sec. To filter out transitory system behavior, all simulations were run for 100 seconds earlier measurements were taken. Table 1 shows the constant configuration parameters.

Table 1 : Simulation Properties

Delay of all side links	3ms
Side links capacity	5Mbps
Bottleneck buffer	10*RTT*bottleneck B/W
FTP/SMDCC data packets	200 bytes
FTP/SMDCC ack packet size	40 bytes
Min. Threshold	0.05*Bottleneck buffer
Max. Threshold	0.5*Bottleneck buffer
weight	0.002

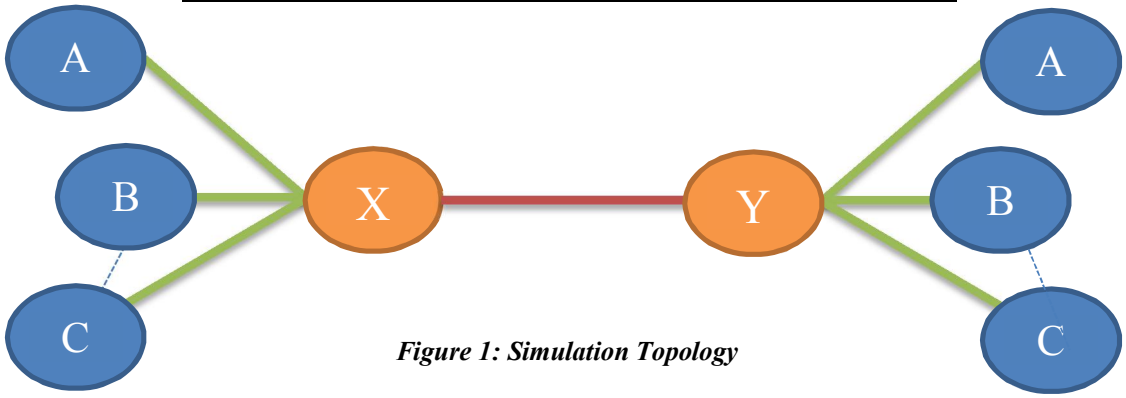

Figure 1: Simulation Topology

4.1 Precision of Bandwidth Estimation Result

At experiment stage First of all, we need to confirm the accuracy of SMDCC's bandwidth approximation result. This is completed by changing the link data broadcasting volume during the lifetime of a SMDCC connection. We modify the link capacity by introducing CBR transportation load at different rates over the bottleneck link. Figure 2 shows how SMDCC become accustomed to changing bottleneck bandwidth with no competing SMDCC or TCP connections. The figure shows both SMDCC's BSE value, and its output data result as a function of time. The figure shows that SMDCC calculation method the BSE precisely reach to within 10% of the available bottleneck bandwidth.

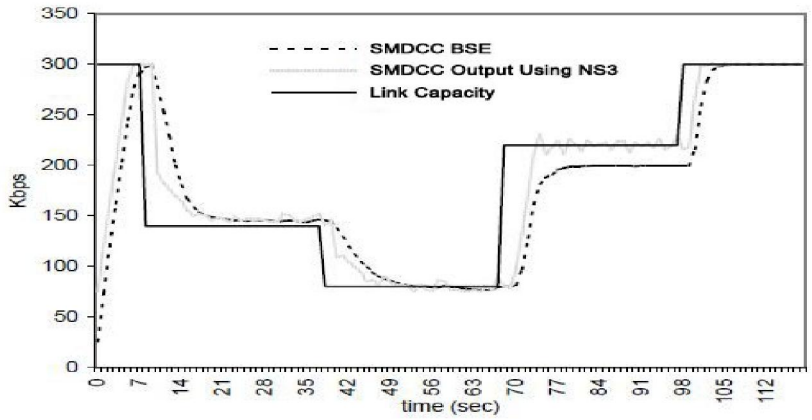

Figure 2 Adaptation of BSE to changing bottleneck bandwidth

4.2 Output Performance

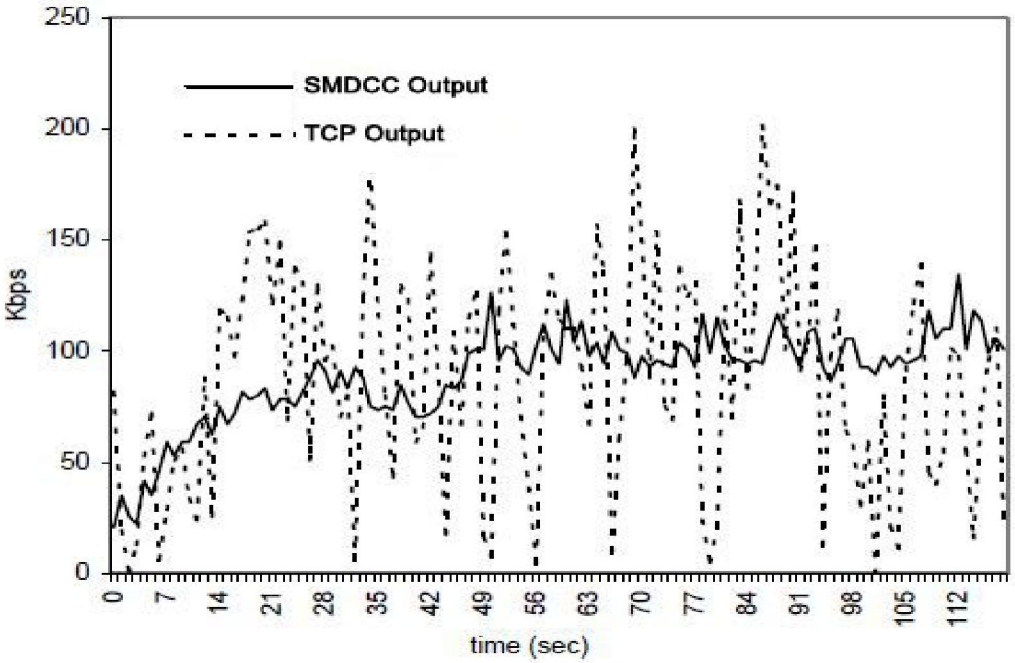

Figure 3 Output curvatures of 1 SMDCC and 1 FTP/TCP New Reno connections

Figure 3 plots the output curve of one SMDCC and one TCP New Reno connection, in 1-second intermissions for a 120 second simulation. A total of 5 TCP and 5 SMDCC connections were actively work, but only one output curve of each type of connection is shown if Figure 3. The bottleneck volume size was 1Mbps, with a 10msec delay accordingly the fair share per connection is 100Kbps. Figure 3 shows that while still getting its fair share value of 100Kbps, SMDCC's output curvature is much smoother than TCP New Reno's oscillatory data sending rate. This is because SMDCC does not reduce its data broadcasting rate by one half or to a minimum of one packet per round trip time as TCP does. Rather, SMDCC adjusts its data broadcasting rate to the BSE, and never performs an exponential rate increase. As a result, SMDCC takes longer to reach the fair share value size, but it does not suffer from marked sending rate fluctuations, and accordingly make suitable for streaming media data such as audio and video.

TCP was formerly aimed to work in a totally wired environment, where a packet loss meant that a buffer overflow, when happened somewhere in the network, and for that reason, served as a sign of congestion. However, in the case of loss links of connectivity, packets may not only be lost due to congestion, but also due to random errors. TCP Reno incorrectly interprets this event as congestion, and unnecessarily reduces its window size with consequent reduction in output. This prohibits TCP from fully exploiting the error-prone link.

SMDCC is more forceful to random loss because of the relative inconsiderateness of bandwidth estimation to irregular errors. In fact as per result, the loss of an isolated packet has only marginal impact on the bandwidth estimation. No time out and slow start are suffered in SMDCC.

To clarify the strength of SMDCC in the wireless domain, Figure 4 presents results from a simulation result with one SMDCC and one TCP New Reno connection running over a 0.3Mbps blockage link with 35ms delay, and an error rate varying from 1 to 20%. We see that SMDCC is competent to maintain a large percentage of the 150Kbps fair share value self-regulating of the error rate, while TCP output continues to reduce as the error rate rises.

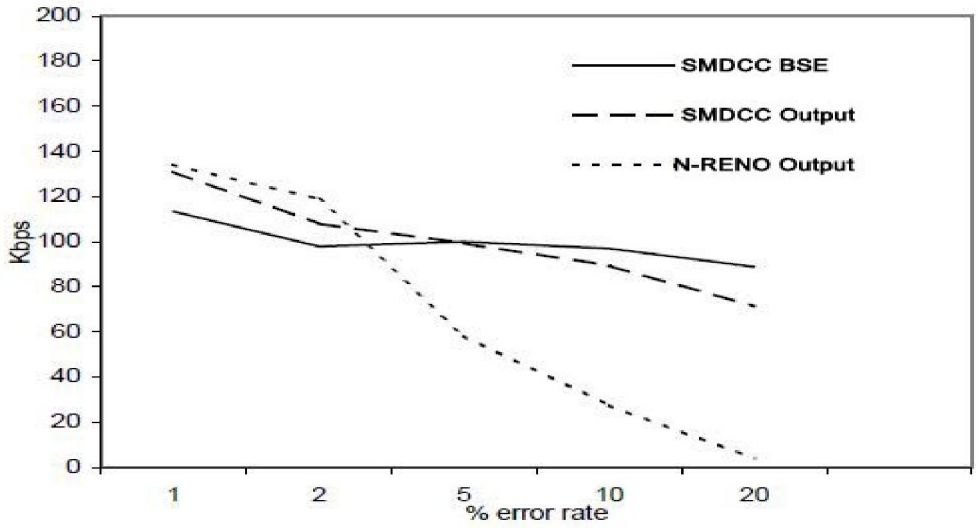

Figure 4 One SMDCC versus One TCP New Reno connection running over a 300Kbps bottleneck lossy link with a 35ms delay

4.3 SMDCC Objectivity

The objectivity of a protocol is a check and measurement of how many number of connections successively linked in the same protocol share the bottleneck bandwidth. The objective of SMDCC is calculated below by simultaneously running 10 SMDCC connections. The time is an average of the out and BSE of each connection is shown in Figure 5. The results show that the connections very little in their output and BSE averages. We also entered different value and make experiments changing the number of connections from 2 to 50. The bottleneck volume size is scaled to maintain a fair share value of 100Kbps, while the delay was held continuous at 10ms. All runs showed good equality output behavior: Jain's Fairness Index was higher than .99 for all runs of instruction.

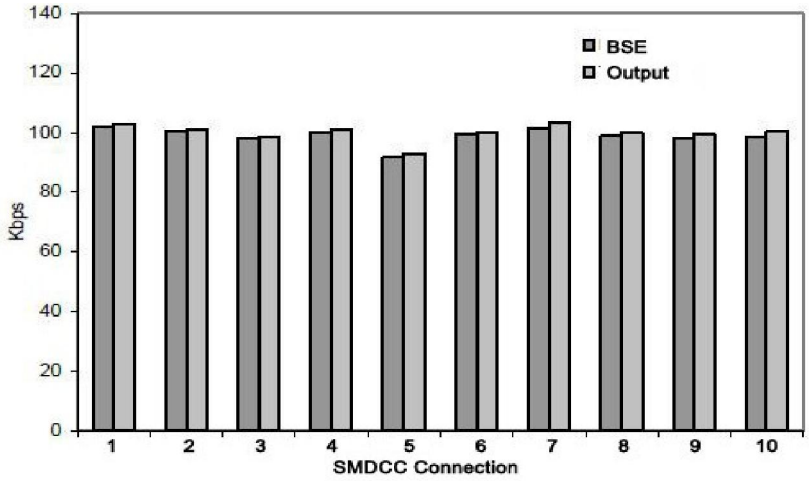

Figure 5 Time being an average of BSE and Output on 10 SMDCC connections

4.4 TCP Friendliness

We also tried and tested approachability for a different selection of network conditions, together with number of flows, bottleneck connection link delay, bottleneck bandwidth, and random irregular errors. Varying the number of flows did not show major impact on responsiveness. As a result we will limit our discussion in this section to how bottleneck delay, bandwidth and random errors impact approachability. All simulations in this section were performed using a balanced set of 10 SMDCC and 10 TCP New Reno connections.

We use the Efficiency Interchange Graph introduced in [4] to better visualize the productivity tradeoff behavior when introducing SMDCC connections to share a common bottleneck link with TCP New Reno connections. The following two tests produce a single point on the chart:

A. A simulation with 20 TCP New Reno flows is run for standardization connection, to establish a reference value. The result output of each flow, and the total link consumption, are measured.

B. Another simulation is then run with half of the TCP flows replaced with SMDCC flows. The new result outputs and utilization are measured.

Let tR1 be the average output of the TCP New Reno flows in the first testing simulation, and U1 be the total link used in experiment operations. In the same way, let tR2 be the average output of the TCP New Reno flows in the second simulation, and U2 be the total link consumption by all connections that is SMDCC as well as TCP connections. We then define:

$$\text{Efficiency Improvement EI} = U2/U1 \quad (2)$$

$$\text{Friendliness FL} = (tR2/tR1) \quad (3)$$

For each network situation of interest, EI and F are determined and an (FL, EI) point is placed on the Efficiency / Friendliness Tradeoff graph. We expect that, in most cases, an increase in Efficiency Improvement (EI) is compensated for by a decrease in Friendliness (F).

We had executed a set of experiments, varying the bottleneck capacity from 5 to 20Mbps. The bottleneck delay was also varied between 10, 50, 100 and 150ms. We also make known to random data packet dropping of media data over the bottleneck link to simulate random loss connected with data transfer over a wireless channel.

Figure 6 shows the Efficiency / Friendliness adjustment behavior when SMDCC connections share a bottleneck link with TCP New Reno connections. Each curve in Figure 6 relates to fixed bottleneck bandwidth and error rate. Furthermore, the points on each curve represent the impact of successively changing the traffic jam delay. For expediency, the arrows mark the direction of increasing bottleneck delay.

We see that for all three 0% error rate cases, as the delay increases, SMDCC decreases TCP's output without any resulting improvement in efficiency. The reason for this is that in these situations there is not much room for enhancement in efficiency, since TCP by itself can accomplish a high operation of the bottleneck link. There is a friendliness problem in the current version of SMDCC, signified by the flat portions of the 0% error curves. This is one area of further required enhancement, and can be addressed by applying adaptive share estimation methods, as in [4, 5], as opposed to the current packet pair model. The packet pair model and the related available bandwidth approximation have been shown to be too aggressive at times. Results presented in [4, 5] have shown major developments in friendliness using the rate estimation model.

Figure 6 also shows that SMDCC is approachable to TCP New Reno in the irregular error case (wireless network situation).

As we said in Section 4.2, when a data packet loss is due to irregular unexpected error, TCP unnecessarily decreases its window size, with resulting reduction in output. This forbids TCP from fully exploiting the capacity of an error-prone link. This incompetence is more noticeable as the bandwidth X delay product increases.

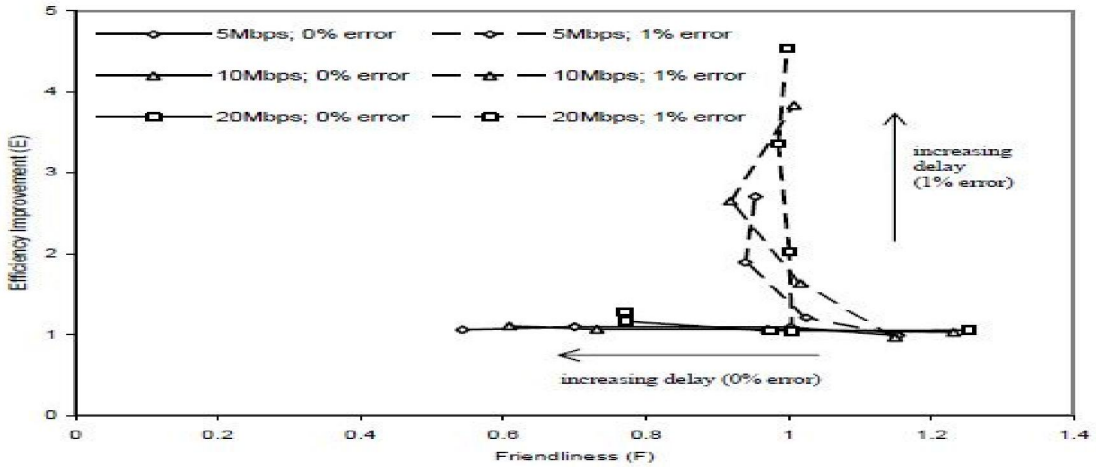

Figure 6: Efficiency / Friendliness tradeoff graph

In SMDCC, the loss of an isolated data packet has only marginal impact on the bandwidth approximation, so it achieves a higher operation of an error-prone link. When SMDCC connections are presented, they readily use the residual bandwidth that TCP is incapable of utilizing. This is apparent in Figure 6 where the 1% error rate curves show that SMDCC remains friendly to TCP irrespective of bottleneck volume capacity and delay. Also, the gain in total consumption resulting from the introduction of SMDCC increases as the bandwidth X delay product increases.

4.5 Evaluation of SMDCC to other Streaming Protocols

As mentioned in Section 2, the common experiment for any media data streaming protocol is to fine-tune its sending rate to a correct value once congestion occurs. Most protocols so far proposed adhere to the multiplicative reduction paradigm of TCP. Others, such as TFRC attempt to calculate

the output that TCP would achieve on the same connection, and use that calculation to set the data sending rate. The downside of these methods is that in striving to achieve the same output as TCP with worse sending rate vacillations, such protocols can inherit the fundamental limits of TCP. [13] Shows that the RAP protocol using fine-grain rate revision can achieve the same output as TCP regardless of the bottleneck delay. Similarly, TFRC [6] is based on an essential control equation that make sure that the protocol uses no more bandwidth, in steady-state, than a TCP conforming protocol running similar conditions. From figures 5 and 6, we see that in the random and unexpected error cases, reaching the same output as TCP. leaves the bottleneck link severely underutilized. SMDCC allows the media data streaming application to access this unexploited bandwidth in these cases, without hindering prevailing TCP connections. As wireless Internet access becomes progressively standard and technology increases the capacity of links, this issue will become all the time more important. Naturally, in the try to improve utilization, SMDCC at times also grabs bandwidth from TCP and vice-versa. We think to address these issues in depth in future work.

V. CONCLUSION

In this paper, we had tried to present on Streaming Media Data Congestion Control (SMCC), a protocol based on bandwidth approximation concepts, which provides flat sending rate changes and good utilization when available network bandwidth fluctuates. SMDCC is fair with respect to other SMDCC flows. It is practically friendly to TCP, particularly with lossy connection links. SMDCC mimics the linear probing for additional bandwidth in TCP congestion escaping phase. Upon finding of a lost data packets, which the receiver explicitly NRACKs along with the existing BSE, the broadcaster adjusts its data broadcasting rate to the present Bandwidth Share Estimation. SMDCC impersonators the Congestion Prevention phase of TCP congestion control, but never the Slow Start phase.

SMDCC dictates the broadcasting rate of a streaming media data application, but not the quality of the stream. The quality reworking is a separate issue. It not only depends upon the existing broadcasting rate, but also upon the receiver's or client machines buffer, and the user preferences and profile as well.

The resulting performance shows that SMDCC output is quite flat and smooth, as equaled to the unstable behavior of predictable TCP. This is a required property for media data streaming applications. Toughness to link errors and random loss was also revealed via simulations.

Areas of additional research include: friendliness of bandwidth approximation method particularly when congestion, not irregular error, is the predominant basis of packet losses; and the extension to multicast broadcasting applications (both single and multi-source). This protocol will also be tried on interactive media data, while the tight time delay limitations will introduce more severe buffer space requirements.

REFERENCES

1. Roberto Canonico, Carmen Guerrero, and Andreas Mauthe. Content distribution infrastructures for community networks. Computer Networks, 53(4):431–433, 2009.
2. D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker. Dynamic Behavior of Slowly-Responsive Congestion Control Algorithms. In Proceedings of Sigcomm 2012.
3. N. Feamster, D. Bansal, and H. Balakrishnan. On The Interactions Between Layered Quality Adaptation and Congestion Control for Streaming Video. 11th International Packet Video Workshop, Kyongju, Korea. May 2009.
4. M. Gerla, M. Sanadidi, K. Ng, M. Valla, R.Wang. Efficiency/Friendliness Tradeoff in TCP Westwood. ISCC 2012.

5. M. Handley, J. Padhya, S. Floyd, and J. Widmer. TCP Friendly Rate Control (TFRC): Protocol Specification. Internet Engineering Task Force, July 2010.
6. Parth H. Pathak and Rudra Dutta. A survey of network design problems and joint design approaches in wireless mesh networks. IEEE mmunications Surveys and Tutorials, 13(3):396–428, 2011.
7. Thomas Plagemann, Roberto Canonico, Jordi Domingo-pascual, Carmen Guerrero, and Andreas Mauthe. Chapter 15 Infrastructures for Community Networks.
8. V. Paxon. Measurements and Analysis of End-to-End Internet Dynamics. Ph.D. thesis, University of California, Berkeley, April 1977.
9. R. Rejaie, M. Handley, D. Estrin. RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet. University of Southern California, Information Sciences Institute. July 1998.

