Int. J. Communications, Network and System Sciences, **2010, 3, 585-592**
doi:10.4236/ijcns.2010.37078 Published Online July 2010 (http://www.SciRP.org/journal/ijcns/).

Analysing TCP for Bursty Traffic

Israfil Biswas, Arjuna Sathiaseelan, Raffaello Secchi, Gorry Fairhurst
Electronics Research Group (*ERG*), *University of Aberdeen*, *AB*24 3*UE*, *King's College*, *Aberdeen*, *UK*
E-mail: {*israfil*, *arjuna*, *raffaello*, *gorry*}*@erg.abdn.ac.uk*
Received April 8, 2010; *revised May* 15, 2010; *accepted June* 19, 2010

Abstract

The Transmission Control Protocol (TCP) has been designed to support interactive and bulk applications, with performance tuned to support bulk applications that desire to continuously send data. In contrast, this paper analyses TCP performance for a class of applications that do not wish to send continuous data, but instead generate bursts of data separated by application-limited periods in which little or no data is sent. In this context, the paper evaluates an experimental method, Congestion Window Validation (CWV), proposed to mitigate the network impact of bursty TCP applications. Simulation results show that TCP-CWV exhibits a conservative behaviour during application-limited periods. The results also show that TCP-CWV is able to use the available capacity after an idle period over a shared path and that this can have benefit, especially over long delay paths, when compared to slow-start restart specified by standard TCP. The paper recommends the development of CWV-like algorithms to improve the performance for bursty applications while also providing an incentive for application designers to use congestion control.

Keywords: Congestion Window Validation, Slow Start, TCP, Congestion Control

1. Introduction

TCP [1] provides an Internet transport protocol that has been designed to support a range of applications. A TCP sender encapsulates data to from TCP segments, which are sent as packets over the Internet. TCP also incorporates congestion control to limit the impact of each flow on other flows that share the network.

The standardized TCP congestion control [2] techniques maintain a record of the currently available capacity along a network path in a variable, known as Congestion Window (cwnd). Senders increase the number of TCP segments in flight each time positive feedback is received indicating that the current rate is not contributing to congestion, and reduce cwnd when it is perceived that the network may be congested as indicated by loss or congestion marking. This regulates the number of packets an application can inject into the network (*i.e.*, the transmission rate). In this method, received ACKs may be thought of as clocking-out new data [1] based on the concrete evidence that recent path capacity was available.

Recent years have seen a change in way many applications use TCP. There has been a significant growth in applications that are "bursty", that is applications that alternate periods of data transmission at a rate with limited by the application with periods where there is no, or little data to be sent. We call a class of applications that send at a rate lower than the actual available rate or at a rate controlled by the application, "application-limited" [3]. Applications that result in such traffic include online games, video conferencing, etc. This behaviour can also result when already deployed applications, such as when the hyper-text transfer protocol (HTTP) [4] is used with persistent connections. Accompanying the growth of bursty application there has been increased deployment of residential Internet [5].

VoIP and video streaming have become popular real-time applications. However, conventional perception is that TCP may be inappropriate for such applications because of congestion controlled reliable delivery may lead to excessive end-to-end delays. More than 50% of the commercial streaming traffic is carried over TCP [6]. As wide-deployment of Network Address Translators, NATs and firewalls often prevent popular media applications over UDP traffic, bursty applications such as Skype [7] and Windows Media Services [6] use TCP. Researchers [7,8] have evaluated the feasibility of constant bit rate (CBR) over TCP and motivated us to evaluate bursty applications performance using CBR traffic over TCP.

Any application-limited TCP flow sends fewer packet probes along the path than allowed by the cwnd. In this case the TCP sender cannot validate that the current value of the cwnd is appropriate by the reception of

ACKs. Therefore, standard TCP reduces the cwnd to the Restart Window (RW) when the TCP sender leaves an idle period, resetting the window to min (IW,cwnd) [9]. This results in poor performance for bursty applications. It also could result in under-utilised capacity for several round trip times (RTTs).

Standard TCP also unnecessarily increases the cwnd during an application-limited period, extending this beyond the size confirmed by reception of ACKs.

To support the congestion control [2] for bursty flows over networks with variable characteristics, the traditional congestion control methods need to be revisited. This includes selection of an appropriate TCP-friendly transmission rate [10] inter-flow and intra-flow fairness, multimedia congestion control. We suggest TCP should allow a flow to return, after an idle period, to a recent previously permitted transmission rate, providing there is no indication that the capacity has changed.

Congestion Window Validation (CWV) [3] is an experimental modification to the TCP congestion control that was proposed to partially solve the problem of an inappropriate cwnd value. CWV modifies TCP congestion control to affect behaviour in two circumstances: when a connection needs to resume transmission after an idle period, and when the flow sending rate is limited by the rate that the application generates data (*i.e.*, application-limited). In both cases, the current value of the cwnd cannot be validated by reception of positive feedback at the sender, since the number of packet probes along the transmission path is lower than the congestion window itself. In other words, the reception of an ACK does not provide evidence that the network path is able to sustain the transmission rate recorded in the cwnd.

CWV also modifies the TCP congestion control procedure by updating cwnd during application-limited to match the application rate. It saves the latest cwnd for use when the flow resumes after an idle or application-limited period.

Many operating systems adopt a conservative approach where TCP resumes transmission in the slow-start phase from a RW of only one packet immediately following an idle period [11]. This indicates that implementers may prefer the safe approach of RFC 5681 which obsoletes RFC 2581, rather than the more recent CWV proposed in RFC 2861. The remainder of this paper explores the limited success of CWV. The next section contains analysis on CWV. Section III analyses the performance of CWV using simulation. Section IV discusses CWV issues. Finally, the last section provides a conclusion.

2. Congestion Window Validation

To understand the response of standard TCP after an idle period, this section describes three application behaviours (this forms the basis of the tests performed in [11] and [12]).

We refer to the case where an application stops sending for a period less than the TCP Retransmission Time Out (RTO) as a "short idle" period [3]. In this case, standard TCP allows a sender to resume transmission at a rate constrained by the cwnd (*i.e.*, at the same rate of increase of sequence number with time). Therefore, TCP can potentially send a cwnd-size line-rate burst into the network after such an idle period. The hypothesis here is that the previously determined cwnd is still valid when the application resumes transmission.

An application that is idle for a period greater than the RTO using standard TCP must restart with slow-start [2]. This resets the cwnd to no more than the Restart Window (RW) and results in exponential growth of the sequence number with time up to the stored ssthresh value. Hence, TCP restarts the ACK clock as at the beginning of a transfer.

An application that stops sending for a period greater several RTOs should not make assumptions about the previous congestion state of the path that it was using, nor that it is necessarily using the same path. Hence, standard TCP recommends a sender that is idle over several RTOs should continue from the RW by also resetting the cwnd.

In [3], a technique was proposed to provide a conservative estimate of cwnd. If the TCP connection has been inactive (*i.e.*, no packets in flight) for a period larger than one retransmission timeout (RTO), cwnd is reduced by a half as many times as the number of RTTs the TCP sender had been idle. This is equivalent to exponentially decaying cwnd during the idle period. Since TCP halves cwnd each time a negative feedback is received (that is, at most once per RTT), CWV provides a safe value for cwnd, which is then expected to be validated by the reception of ACKs during the first round of transmission following the idle period.

TCP also resets the slow-start threshold (ssthresh) to max (ssthresh, 3 × cwnd/4), which keeps previous information on the available path capacity. Since TCP resumes with a cwnd larger than the restart window (RW), the TCP sender can quickly recover its previous transmission rate. CWV also suggests that an application that stops sending for a period greater several RTOs should make no assumptions about the previous congestion state of the path it is using. Hence, a sender using TCP-CWV will exhibit a performance resembling standard TCP.

When the TCP sender detects that the cwnd has not been fully used for a period larger than an RTO (e.g. observing that each packet transmission does not use a full cwnd with an empty transmission buffer for more than RTO seconds), cwnd is reduced to (cwnd + w_used)/2. Here, w_used is an estimate of the portion of cwnd that was effectively used by the application. Hence, this mechanism avoids growth of cwnd to an arbitrary larger

value than the window-size actually used (because of slow-start or congestion-avoidance cwnd increase) and allows cwnd to be validated by reception of ACKs by the sender.

Figure 1 shows the dynamics of cwnd for standard TCP and TCP-CWV. In this case, we consider an application that generates data at an application-defined rate, interspersed by idle times when the application is inactive. The figure shows that at point 'a' both standard TCP and TCP-CWV start using slow start. Point 'b' denotes the point where the maximum rate permitted by cwnd is less than the application rate. Standard TCP continues to grow the cwnd, while CWV does not increase cend above that corresponding to the used rate. At point 'd', there is an idle period greater than a RTO, CWV reduces cwnd by a half, but standard TCP resumes from the RW (point 'c').

The discussion so far has focused on stable paths, and it may be argued that path conditions often remain relatively stable, at least for periods of several minutes [13]. However there are also cases where the Internet path characteristics can change (e.g., routing topology changes [14], mobility changes [15], intermittent wireless links) or a change in traffic scenarios could invalidate the congestion window. This may mean that a previously safe rate may become unsuitable, if too a long a time has passed since the network the path was last used. This may also be a cause of an inappropriate cwnd.

To explore this, we examine a highly dynamic network scenario where, not only significant variations in traffic rate, but also changes in the transmission path (due for instance to terminal mobility) modifying capacity or delay characteristics of a path may happen. The problem is that any path change experienced while the sender was idle could result in a significant increase of drop rate. It is therefore important to assess whether it is reasonable to allow an application to send faster after idle. Hence,

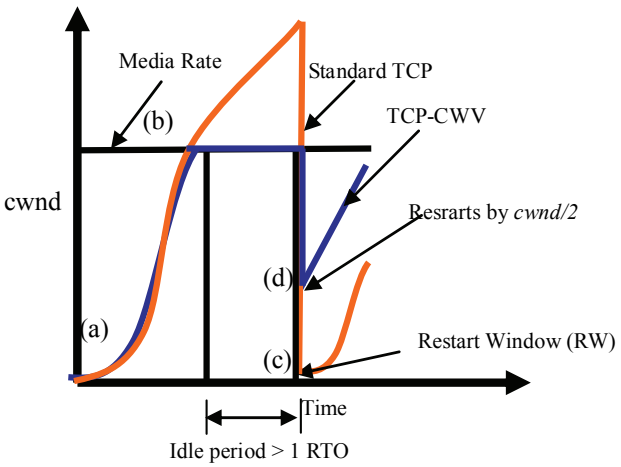

Figure 1. Illustration of cwnd dynamics for standard TCP and TCP-CWV.

the assessment could contribute to limited transient congestion in times of change, in return for improving application responsiveness. Next section analyses the impact of these methods that send faster after idle period in transient conditions using TCP-CWV.

The section explores this hypothesis for a set of simultaneous bursty flows that share a single bottleneck path. We analyse performance in a severely congested scenario for both idle period and application-limited period case.

3. Simulation Analysis of CWV

This section compares the performance of TCP-CWV and standard TCP following a significant change of the path characteristics.

The section considers two cases for analysis: 1) several idle TCP connections restart simultaneously (idle-period case), 2) several application-limited TCP flows simultaneously subjected to a sudden variation of application sending rate (application-limited case).

3.1. Idle Period Case

We analyse the performance of CWV after an idle period in two cases:

1) when the bottleneck is shared between equal numbers of standard TCP and CWV flows (heterogeneous-flow scenario). Hence, heterogeneous flows are mixed flows that are competing with flows using one alternate algorithm at a time.

2) when CWV or standard TCP only flows are present (homogeneous-flow scenario). Hence, all flows use one congestion control algorithm. Here a path is occupied by one kind or same type of flows and not sharing with other types of flow.

All simulations used a large advertised receiver window (1.5 MB) so that the TCP sender could send constrained only by the cwnd or CWV. Hence, flows at steady-state are interrupted and restarted after a short period of time. The interrupt and restart time of each flow was chosen from a uniform random distribution. Varying the duration of the idle period allowed investigation of CWV behaviour compared to standard TCP. **Table 1** summarises the simulation parameters.

Figure 2 illustrates the simulation topology where multiple TCP flows [S_1, S_2 …S_n, S_{n+1}] contribute traffic at the node n_0 and destinations are [R_1, R_2 …R_n, R_{n+1}]. These flows used a rate appropriate to medium quality video [16] over IP with an encoding rate of 512 kbps (packet size of 1500 bytes). To reach the destination via node n_3, one path is n_4-n_5 (capacity of 100 Mbps) and the alternate path is n_1-n_2.

Assuming a scenario where there is a path break on the path n_4-n_5 and n_0 chooses n_1-n_2 with a currently

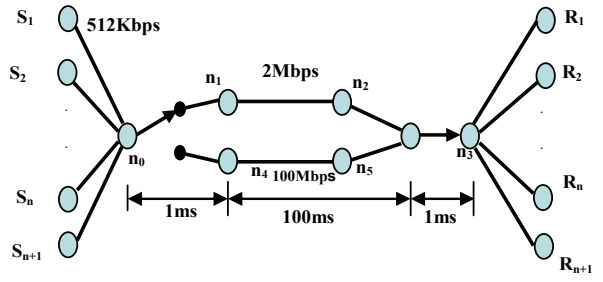

Figure 2. Single bottleneck topology.

Table 1. Configuration parameters for idle period simulations.

Bottleneck Link	
Bandwidth (Mb/s)	100
One-way Link Delay (ms)	100
Router Buffer Size	BDP
Access Links	
Bandwidth (Mb/s)	100
One-way Link Delay (ms)	1
TCP Configurations	
Maximum Segment Size (B)	1460
Maximum Advertised Window Size (kB)	1500
Minimum Retransmission Timeout (sec)	1
Simulation duration (sec)	40
CBR traffic Parameters	
Size (bytes)	1460
Rate (Kbps)	512
Idle period	
Duration of periods (sec)	0.5 to 5
Changed Bottleneck Bandwidth (Mb/s)	2
Flow/Drop monitor duration (RTT)	5

available capacity of 2 Mb/s. We assume both paths have the same path delay of 100 ms.

While TCP flows are idle, the common path changes from 100 Mb/s to 2 Mb/s. The rate of 2 Mb/s was chosen because if each TCP connection carries a 512 Kb/s constant bit-rate flow. Increasing the number of connections to 32 allows evaluation of the TCP performance under high congestion. When TCP is used for bursty applications, it is expected to match the application transmission rate or media rate. We measure the 'Average received rate', this is the arrival rate at the receiver over a short period of time. The performance is measured over 5 RTT following an idle period (as an indication of the goodness of restart response).

The less conservative approach of CWV after an idle period can result in more packet losses compared to standard TCP.

The packet drop rate was evaluated at the bottleneck as an indication of protocol aggressiveness. This allowed investigation of the trade-off between user-perceived performance, reflected by TCP received packet rate, and network performance (*i.e.*, the loss rate). This drop-rate is only relevant in the homogeneous-flows scenario, where the cause of buffer overflows can be attributed to the investigated protocol.

3.1.1. Heterogeneous-Flows Scenario

When the idle period is less than one RTO (about 0.5 sec in the simulations), cwnd remained unchanged using both TCP-CWV and standard TCP. These simulations confirm that the two protocols achieve the same performance. However, when the idle period is larger than one RTO, TCP-CWV achieves better performance in terms of packet arrival rate at the receiver.

Figure 3 shows performance for an idle period of 1.5 sec with the heterogeneous-flows. In this case, standard TCP resumes from the RW (one packet in this experiment) and performs slow-start, whereas CWV restarts from a level significantly larger than RW. Finally, if the idle period is larger than several RTOs, the simulations show that TCP-CWV achieves the same performance as standard TCP and that CWV reduces the cwnd to the RW as in standard TCP.

3.1.2. Homogeneous-Flows Scenario

When TCP flows compete with flows of the same type (*i.e.*, using the same congestion control algorithm), similar behaviour to the heterogeneous-flow case is observed. That is, when the idle period is smaller than an RTO or sufficiently large (e.g., 5.0 sec), standard TCP and TCP-CWV achieves the same performance in received rate and drop rate. Whereas for an idle period of few RTOs (e.g., 1.5 sec) TCP-CWV outperforms standard TCP in terms of achieved bit rate (**Figure 4**). However, this also produces higher congestion at the bottleneck as illustrated by the drop rate graph (**Figure 5**).

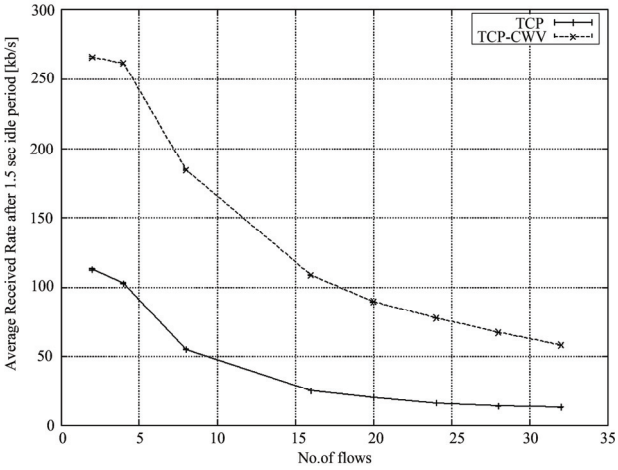

Figure 3. Average received rate vs. number of flows for the heterogeneous flows after 1.5 sec idle period.

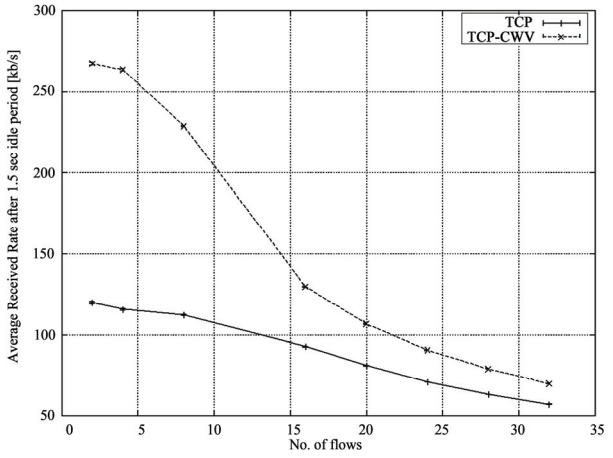

Figure 4. Average received rate vs. number of flows after 1.5 sec idle period for homogeneous-flows.

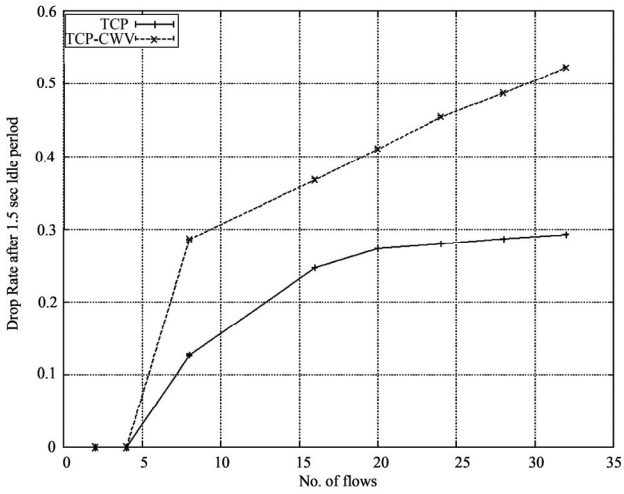

Figure 5. Drop Rate after 1.5 sec idle period for homogeneous-flows.

In conclusion, simulations show CWV restarts quickly and hence higher received rate compare to standard TCP. CWV shows best performance over a less congested path by dropping fewer packets. The heterogeneous case shows a higher received rate after an idle period CWV utilises more bottleneck capacity than standard TCP. A quick restart helps CWV to be fair to itself, and also shows fairness to standard TCP by reducing the rate similar to standard TCP rate in the longer period.

3.2. Impact over Long Network Path

We also considered the performance of standard TCP and TCP-CWV over a Long Fat Network (LFN) [17], specifically one with a long network path. Hence, in this experiment, the one-way delay is increased to 300 ms (*i.e.*, more than 600 ms RTT). CWV has an advantage of quick restart after an idle period. **Figure 6** shows the average received rate and **Figure 7** shows the loss rate for the

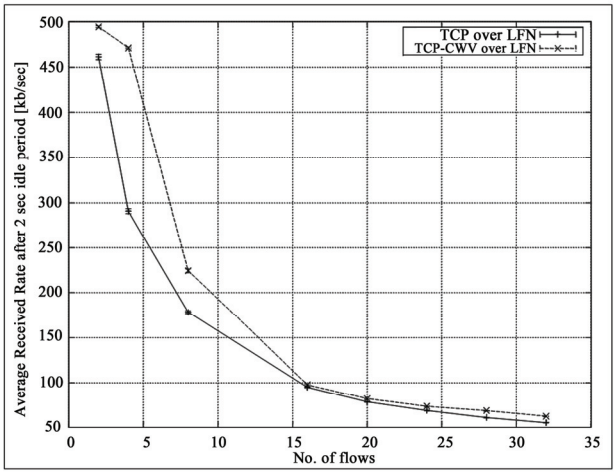

Figure 6. Average received rate vs. number of flows after a 2 sec idle period with homogeneous-flows over a long network path.

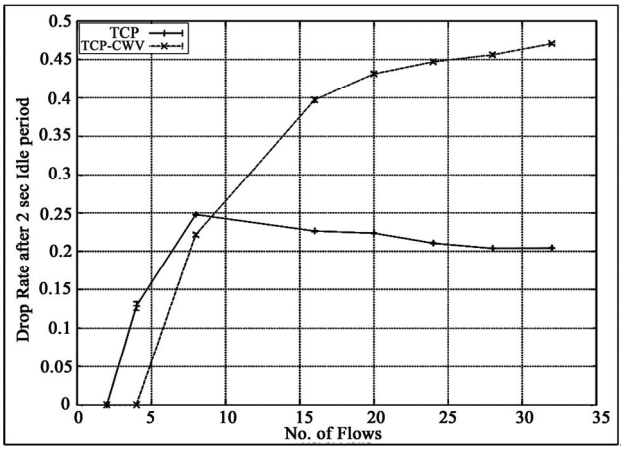

Figure 7. Drop Rate 2 sec idle period in homogeneous-flows over a long network path.

scenario with homogeneous-flows.

This shows that TCP-CWV is able to achieve a higher received rate at moderate congestion without severe router buffer overflow. The graphs also show that TCP-CWV allows a sender to maintain a cwnd sufficiently large to utilise the capacity after an idle period.

3.3. Delay Jitter

Figure 8 shows the one-way delay measured as the application-generated packet time at the sender and reception time at the receiver socket buffer by the application against the packet sequence number. The queuing delay induced by TCP-CWV is substantially decreased with respect to standard TCP for a restart time after an idle period longer than one RTO. This benefit can be observed providing that cwnd is not reduced to RW, which

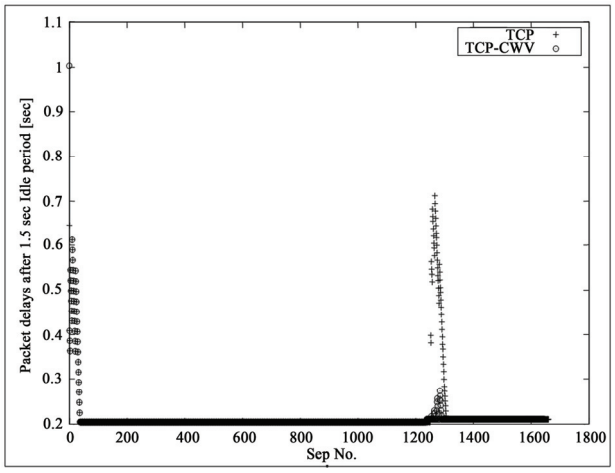

Figure 8. Average one-way delay over 25 standard TCP and TCP-CWV connections in the homogeneous-flows scenario (1.5 sec idle period and 200 ms propagation delay).

is for idle periods of up to several RTOs. This result illustrates that the TCP-CWV flows suffer high delay only at the beginning of a connection because of the three-way handshake and the initial slow-start phase.

The maximum delay variation is strongly dependent on the duration of the idle period. In this simulation, the best performance improvements are observed between five to twenty-two RTTs of idle period duration. For less than five or shorter RTT, CWV performs the same as standard TCP (*i.e.*, there is no reduction of cwnd) and after a longer RTT (e.g., more than 20 RTTs) CWV decays the cwnd to a value almost equal to that of the initial window of standard TCP.

3.4. Application-Limited Period

The increase of cwnd allowed by standard TCP during an application-limited period can be inappropriate (*i.e.*, allowing the cwnd to grow unnecessarily while transmitting at a rate lower than the available capacity). If there is a change in application transmission rate, this behaviour could lead to many packet drops. This is because the value of cwnd no longer reflects the current path capacity. CWV was proposed to enhance congestion control by continuously updating cwnd during application-limited periods to avoid unnecessary increase during application-limited periods.

To create an application-limited scenario, we arrange for the sending rate of a CBR application to be lowered to a steady-state from 512 kb/s to 12 kb/s (for instance, when a high bit rate media flow switches to a low bit rate media flow). After an interval of several RTOs, the application rate is restored to its original rate.

At the same time, the shared link capacity is changed from 100 Mb/s to 2 Mb/s, as in the previous set of simulations. The average rate of packet arrival at the receiver was measured (over a several RTOs starting from the capacity discontinuity) as a measure of the response time, and the packet drop rate at the bottleneck as a measure of capacity sharing of the congestion control protocol. **Table 2** shows the configuration parameters and we used the same topology of **Figure 2**.

3.4.1. Heterogeneous-Flow Scenario

Results for an application-limited scenario show that TCP-CWV is conservative and cannot respond quickly to a change in the application rate, whereas standard TCP allows TCP to immediately send additional packets (see **Figure 9**). Hence, application-limited period responses are opposite to the idle period responses for standard TCP and TCP-CWV.

Table 2. Configuration parameters for simulations with an application-limited period.

Bottleneck Link	
Bandwidth (Mb/s)	100
One-way Link Delay (ms)	100
Router Buffer Size	BDP
Access Links	
Bandwidth (Mb/s)	100
One-way Link Delay (ms)	1
TCP Configurations	
Maximum Segment Size (B)	1460
Maximum Advertised Window Size (kB)	1500
Minimum Retransmission Timeout (sec)	1
Simulation duration (sec)	40
CBR traffic Parameters	
Size (bytes)	1460
Rate (Kbps)	512
Change of Rate-application-limited period (Kbps)	12
Application-limited period	
Duration of periods (sec)	0.5 to 5
Changed Bottleneck Bandwidth (Mb/s)	2
Flow/Drop monitor duration (RTT)	10

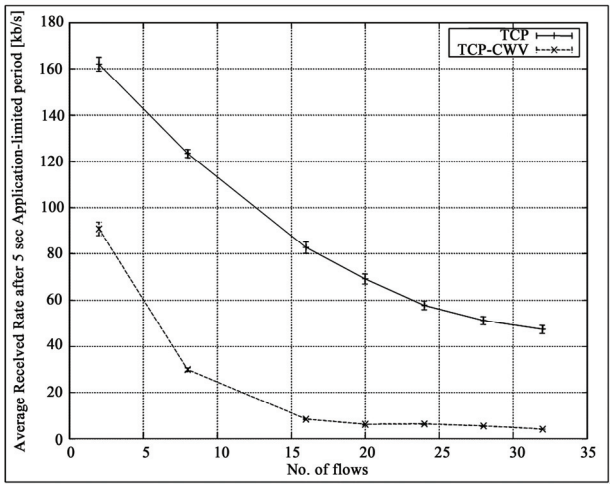

Figure 9. Average packet arrival rate at the receiver for standard TCP and TCP-CWV connections in the heterogeneous-flows scenario after 5 sec application-limited period and 100 ms delay path.

3.4.2. Homogeneous-Flow Scenario

Figure 10 confirms that in the case of homogeneous-flows standard TCP provides higher received rate than TCP CWV.

On the other hand, the inefficient cwnd growth offered by standard TCP results in a much larger packet drop rate (**Figure 11** highlights) with respect to CWV. Changing the size of the interval the application rate remains at low rate, we observed that the longer the interval, the lower the correlation between the cwnd and the bandwidth-delay product.

Finally, CWV has no benefit to the application because it offers a lower received rate compared to standard TCP. CWV also is less desirable for the shared bottleneck compared to current TCP. However, it is observed that the CWV application-limited period approach is safe for these transient events.

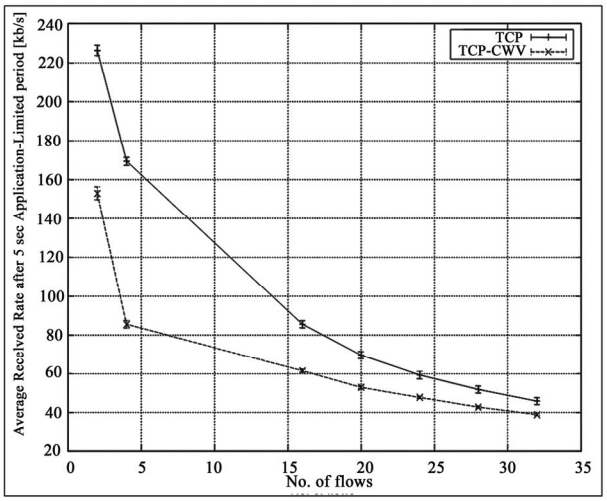

Figure 10. Average arrival rate at the receiver after 5 sec idle period in homogeneous-flows scenario for CWV and standard TCP connections.

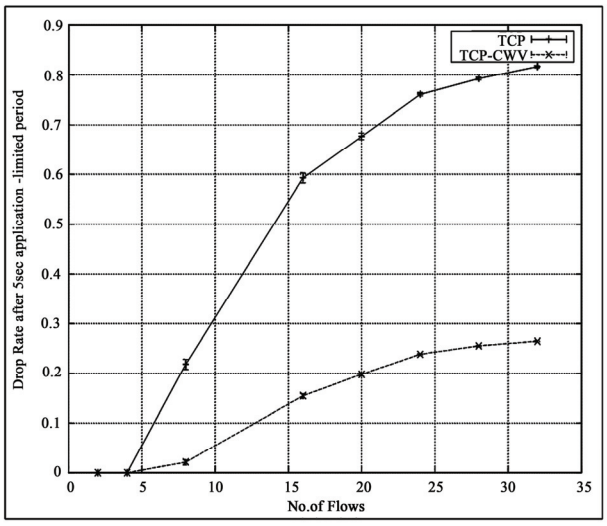

Figure 11. Drop rate after 5 sec application-limited period in homogeneous-flows scenario for CWV and standard TCP connections.

4. Discussion

Standard TCP takes a conservative approach during an idle period that lasts more than one RTO. It reduces the cwnd to the RW and then slow-starting back to the application rate. This approach is beneficial to the network, but does not benefit the application (as shown by the reduced average received rates for the case of Heterogeneous, Homogeneous flows and the packet drop rates in **Figures 3-5** respectively).

TCP-CWV mitigates the poor network performance of standard TCP by reducing the cwnd by one half for every RTO period that the application is idle. This benefits the application allowing an application to send packets much faster after a restart from an idle period. However, CWV impacts the network more if there is a transient event that changes the network path characteristics while the application was idle (shown by the increased average receive rates and packet drop rates in **Figures 4** & **5**).

During an application-limited period of more than one RTO, standard TCP behaves more aggressively compared to TCP-CWV. Standard TCP benefits an application-limited application by the faster sending, whereas TCP-CWV behaves more conservatively. The conservative approach of TCP-CWV ensures that in transient conditions, the impact caused by TCP-CWV flows restarting from a application-limited period is less compared to standard TCP (as shown by the lower average received rates and packet drop rates in **Figures 10** & **11**). **Table 3** shows the comparison as response after an idle or application-limited period.

Table 3. Comparison of response after idle or application-limited period: Standard TCP and TCP-CWV.

Approaches	Period	Standard TCP [RFC 5681]	TCP-CWV [RFC 2861]
1. Fairness to Application	Idle period	Conservative probing but losses application fairness.	Aggressive probing. Fair to the application.
	Appliction-limited period	Unnecessarily increases the cwnd, good for the application	Decay cwnd to utilise the capacity. Conservative approach, not fair to the application
2. Fairness to Path	Idle period	Safe probing. Good for the Internet path	Probing is not safe So, no benefit for the path.
	Appliction-limited period	Higher drops in transient events, bad for the path	Less drops to transient events So fair to the path

Our analysis of TCP-CWV poses a question: What is best for application designers that develop bursty applications? TCP-CWV would benefit an application if it exhibits regular idleness. However TCP-CWV would be of benefit only if the idle period was several RTOs. Applications exhibiting very large idle periods (tens of seconds) would experience no benefit from using TCP-CWV, since the behaviour would be the same as for standard TCP. Although TCP-CWV benefits the network in an application-limited scenario, the conservative approach of TCP-CWV does not provide an incentive to application to use this.

5. Conclusions

The current TCP specification defines a conservative slow start algorithm that can penalise an application which restarts from an idle period, making it undesirable for interactive bursty applications. TCP-CWV suggests a remedy to this problem, allowing a faster restart after an application was idle. This is seen to be beneficial to the application, and suggests the need for appropriate methods can be found, that balance the threat of network collapse against application performance. TCP-CWV exhibits a much more aggressive faster restart behavior after idle, however when an application is limited by the application rate, TCP-CWV has a much more conservative approach. Standard TCP has a more aggressive approach for application-limited flows. This non uniform approach of TCP-CWV has been a deterrent for it being deployed widely in the Internet, with TCP-CWV only deployed in the Linux OS (enabled by default).

Our future work will propose a new method(s) applicable to both idle and application-limited periods. We hope our work would lead to standards paving a way for application designers. The availability of methods that effectively support burst applications will provide an incentive for application designers to change to use a standard method to share the network resources in a more efficient and friendly manner.

6. References

[1] J. Postel, "Transmission Control Protocol," STD 7, RFC 793, September 1981.

[2] V. Jacobson, "Congestion Avoidance and Control," *ACM SIGCOMM Computer Communication Review*, Vol. 25, No. 1, 1995, pp. 157-187.

[3] M. Handley, J. Padhye and S. Floyd, "TCP Congestion Window Validation," RFC 2861, June 2000.

[4] H. F. Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H. Lie and C. Lilley, "Network Performance Effects of HTTP/1.1, CSS1, and PNG," *Proceedings of the ACM SIGCOMM'*97 *Conference on Applications*, *Technologies*, *Architectures*, *and Protocols for Computer Communication*, Cannes, France, September 1997, pp. 155-166.

[5] J. Heidemann, K. Obraczka and J. Touch, "Modeling the Performance of HTTP over Several Transport Protocols," *IEEE*/*ACM Transactions on Networking*, Vol. 5, No. 5, 1997, pp. 616-630.

[6] L. Guo, E. Tan, S. Chen, Z. Xiao, O. Spatscheck and X. Zhang, "Delving into Internet Streaming Media Delivery: A Quality and Resource Utilization Perspective," *Proceedings of the* 6*th ACM SIGCOMM Conference on Internet Measurement*, Rio de Janeriro, Brazil, 2006, pp. 217-230.

[7] B. Eli, B. S. Abdul, R. Dan and S. Henning, "The Delay-Friendliness of TCP," *Proceedings of the* 2008 *ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, Annapolis, MD, USA, 2008, pp. 49-60.

[8] S. Baset, E. Brosh, V. Misra, D. Rubenstein and H. Schulzrinne, "Understanding the Behavior of TCP for Real-Time CBR Workloads," *Proceedings of the* 2006 *ACM CoNEXT Conference*, Lisboa, Portugal, 2006.

[9] M. Allman, V. Paxson and E. Blanton, "TCP Congestion Control," RFC 5681, September 2009.

[10] M. Handley, S. Floyd, J. Padhye and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," RFC 3448, January 2003.

[11] Md. I. Biswas and G. Fairhurst, "A Practical Evaluation of Congestion Window Validation Behaviour," 9*th Annual Postgraduate Symposium in the Convergence of Telecommunications*, *Networking and Broadcasting PGNet*, Liverpool, UK, 2008.

[12] Md. I. Biswas and G. Fairhurst, "An Investigation of TCP Congestion Window Validation over Satellite Paths," 4*th Advanced Satellite Mobile Systems Conference*, Bologna, Italy, 2008, pp. 37-42.

[13] H. Balakrishnan, S. Seshan, M. Stemm and R. Katz, "Analysing Stability in Wide-Area Network Performance," *ACM Sigmetrics Performance Evaluation Review*, Vol. 25, No. 1, 1997, pp. 2-12.

[14] J. Ni, H. Xie, S. Tatikonda and Y. R. Yang, "Network Routing Topology Inference from End-to-End Measurements," Technical Report, Yale University, 2007.

[15] A. C. Snoeren and H. Balakrishnan, "An End-to-End Approach to Host Mobility," 6*th ACM*/*IEEE International Conference on Mobile Computing and Networking*, Boston, Massachusetts, August 2000, pp. 155-166.

[16] S. A. Baset and H. Schulzrinne, "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol," *IEEE INFOCOM*, Barcelona, Spain, 2006, pp. 1-12.

[17] V. Jacobson and R. Braden, "TCP Extensions for Long-Delay Paths," RFC 1072, October 1988.

