Improving TCP Performance and Reducing Self-Induced Congestion with Receive Window Modulation

Francesco Ciaccia*†, Oriol Arcas-Abella†, Diego Montero†‡, Ivan Romero†
Rodolfo Milito†, René Serral-Gracià*, Mario Nemirovsky†
*Universitat Politècnica de Catalunya (UPC), Barcelona, Spain {fciaccia, rserral}@ac.upc.edu
‡Universidad de Cuenca, Cuenca, Ecuador {diego.monterob}@ucuenca.edu.ec
†Clevernet Inc., San Francisco, USA {fciaccia, dmontero, iromero, oriol, rmilito, mario}@clevernet.io

***Abstract*—We present a control module for software edge routers called Receive Window Modulation - RWM. Its main objective is to mitigate what we define as *self-induced congestion*: the result of traffic emission patterns at the source that cause buffering and packet losses in any of the intermediate routers along the path between the connection's endpoints. The controller modifies the receiver's TCP advertised window to match the computed bandwidth-delay product, based on the connection round-trip time estimation and the bandwidth locally available at the edge router. The implemented controller does not need any endpoint modification, allowing it to be deployed in corporate edge routers, increasing visibility and control capabilities. This scheme, when used in real-world experiments with loss-based congestion control algorithms such as CUBIC, is shown to optimize access link utilization and per-connection goodput, and to reduce latency variability and packet losses.**

***Index Terms*—Edge router, TCP flow control, self-induced congestion, rate limiting.**

I. Introduction

TCP is the protocol of choice for many of the distributed application being developed. It provides guarantees in terms of data integrity and delivery; it controls and modulates the sending rate according to estimated network conditions by means of two mechanisms: i) congestion control and ii) flow control. The former is a sender mechanism aimed at adjusting the sending rate, according to congestion events in the network as estimated by the congestion control algorithm. The latter provides the receiver with a mechanism to signal to the sender the amount of data it can receive. In fact, TCP flow control relies on explicitly signaling the available receive window in the protocol header. It was designed considering slow receivers, which were not able to process all the received data because of constrained computational resources. However, TCP flow control is rarely involved in a normal TCP connection in modern Internet era, as processing power has increased dramatically since the protocol definition, preventing the receiver to become the bottleneck. While network infrastructure has evolved, TCP design choices in congestion control can still represent a limitation in network resources exploitation.

Many congestion control proposals, e.g. CUBIC [1], act too aggressively, contributing to what we define as *self-induced congestion*: intermediate routers start dropping packets causing consistent throughput reduction, especially in the presence of loss-based congestion control algorithms. Buffering increases, reducing the responsiveness of latency-sensitive concurrent flows, such as those of interactive applications.

In order to address these problems, we present Receive Window Modulation—RWM, a control module for edge routers:

- RWM mitigates self-induced congestion and improves end-to-end TCP performance, latency and fairness. Average application goodput is improved up to 70% and latency is reduced by a 2.5x factor in some scenarios.
- RWM adjusts the advertised receive window of packets traversing the edge router. This is intended to provide an upper bound to the sender's congestion window growth; the window, which is an upper limit to the connection's bandwidth-delay product (BDP), is based on the router's locally available bandwidth, the estimated connection RTT, and the policies for each class of traffic.
- This mechanism does not require any modification of the TCP stack, as it is transparent to the end-points, nor connections are terminated by a proxy.
- RWM preserves the characteristics of the sender congestion control deployed in the end-point and does not interfere nor substitute Active Queue Management strategies deployed in the router.

The rest of the paper is structured as follows: Section II describes the related work, especially previous proposals for edge router architectures and control strategies based on explicit TCP packet modification in intermediate nodes. Section III describes the controller architecture's building blocks. Section IV presents an experimental evaluation and analyzes the results of RWM in a real Internet environment. Section V discusses some improvement opportunities for the schema proposed. Section VI summarizes the key contributions and discusses future work.

II. Related Work

In [2] an architecture for bandwidth fair sharing is proposed; it focuses on home gateways and a credit-based resource allocation system. The solution, although, requires some level of interaction with the ISP core network to negotiate the amount of credits the gateway can spend during a congestion period. They also envision a control mechanism based on TCP advertised window modification, but it is not based on BDP estimation. They adapt the TCP receive window by proxying the connection and controlling the receive window at the socket level in the gateway. Instead, RWM proposal for TCP window modification is based on in-flight packet modification.

Other works have been proposed in the past that take advantage of the TCP flow control mechanism to optimize the connection behavior. Explicit Window Adaptation (EWA) [3] also modifies the advertised window of TCP packets as a means of congestion control in intermediate nodes. The goal is to reduce buffer bloat and self-induced congestion due to TCP's window probing. Their testing environment is based on TCP/IP connections over ATM virtual networks. The value of the advertised window is a function of the available buffer space in the ATM router, and the behavior of EWA is compared against a typical Random Early Detection (RED) buffer management mechanism. RWM extends this study in several ways. Window adaptation is a powerful mechanism as demonstrated in EWA with ATM networks, but this technology has been largely superseded by IP-only networks. In this paper, the effects of window adaptation are tested in a more up-to-date environment where RWM is agnostic to the type of network segments the flow will traverse. RWM does not need to be deployed at the bottleneck as its objective is to maximize end-to-end behavior of the TCP flows according to the local resources available to the edge router. The feedback function does not need to be exclusively coupled to the buffer space though, as it can be dynamically derived from the current locally available bandwidth and some user defined traffic policies, computing a BDP value for each connection. In addition, the benefits of window adaptation in this work are not only focused on goodput and buffer utilization, but also a study on the impact on latency is included. In [4] a similar flow control based mechanism is proposed but the scheme is developed with satellite networks in mind and its deployment model foresees the controller to be right before the satellite link bottleneck.

Recently, Google has proposed a new hybrid scheme for congestion control called BBR [5]: its objective is to characterize the current BDP of the connection. It accomplishes so estimating the minimum RTT and the end-to-end available bandwidth, by periodically inducing queueing and then draining the buffer. In [6] and [7] it has been demonstrated that BBR is not fair towards loss-based congestion control flows and its performance can fluctuate in presence of small buffers on the path. Our solution, as of today, does not estimate the available bandwidth between the connection endpoints; the receive window is set accordingly to the edge router access link locally available capacity and the measured connection RTT, providing an upper boundary to the sender congestion control. It is not a substitute of the congestion control algorithm deployed in the sender: the response of the sender algorithm to transient network conditions is preserved. Given this, if our system is coupled to senders that use loss-based congestion controls like CUBIC, the fairness of such an algorithm towards other flows is retained. Finally, in contrast to BBR, this system is targeted to routers and does not require modifying the end-points.

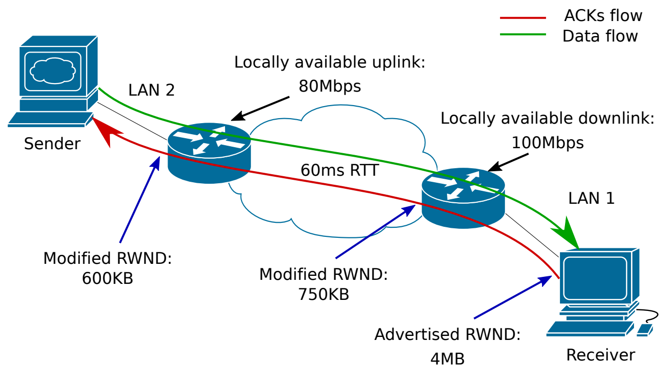


Fig. 1: Receive Window Modulation deploy schema. The window advertised by the receiver is modified according to the router's locally available bandwidth and the estimated RTT for the connection.

III. Architecture

The main idea behind RWM is to enforce an upper bound to the sending rate of the sender, exploiting TCP flow control. This is achieved by modifying the receiver's advertised window of in-flight acknowledgement packets. This type of control tries to mitigate some limitations of loss-driven congestion control algorithms, which are still extensively used nowadays. Tail drop in intermediate routers can be reduced and bufferbloat avoided if the sending rate in the end-point is adjusted to the path's nominal BDP. The edge deployment vantage point gives the router visibility on all the flows being generated in the Internet access link, which allows a fair share of the edge router resources in terms of bandwidth: the router can throttle flows in case they exceed a specific policy or the local resources of the router without the need to drop packets.

RWM computes the BDP value for each flow based on:

- RTT measurement during connection establishment.
- The router access link available capacity, where capacity can refer to the router network card nominal line rate or a user provided parameter indicating the bandwidth throttling enforced by the ISP on the access link.

Figure 1 shows a typical functional schema of a RWM deployment. Both edge routers implement RWM: the controller can be deployed close to both the sender and/or the receiver. In the scenario presented in Figure 1 the connection is already established and both edge routers were able to estimate the connection RTT based on the three-way handshake. Each of them transparently modify the receive window advertised by the receiver in the acknowledgement packets, in order to match the locally available bandwidth. The sender will then conform its sending rate to the enforced receive window in case its congestion window exceeds it.

The control logic is triggered on a per-event basis: when a new incoming flow is registered, it is associated a specific traffic profile so that RWM can recompute the optimal value of the receive window for all active flows; a data plane component will then enforce it on the target flow. Particular attention is put in extracting the Window Scaling factor negotiated by both endpoints on connection establishment. Bandwidth allocation is done on a per-class basis, where classes can be defined by the user based on the application port. An upper limit is assigned for the bandwidth of each class. The user is free to allocate bandwidth to the classes within those limits.

Being *C* the capacity of the edge router link as previously defined; *P* the allocation policy defined for a specific traffic class *j*, expressed as percentage of the access link capacity; *i* the i-th flow for the class *j*; RWM recomputes the receive window *RWND* to be assigned to each flow *i* so that it always respects the following relation:

$$\sum_j (\sum_i \frac{RWND_i}{RTT_i}) * P_j = C \tag{1}$$

Equation 1 guarantees that the link capacity is distributed equally between flows of the same class and that each class is assigned the user defined share of network resources. The operation of RWM's logic implementing equation 1 is described in Algorithm 1. Given the link capacity and the traffic class policies, RWM computes the amount of link share to provide to each class. Then it iterates over all classes and their active flows, and computes the BDP associated to each of them starting from an estimation of the round trip time. It finally computes the receive window to enforce on the acknowledging flow based on the window scaling factor it registered at connection establishment. If the window advertised by the receiver is smaller than the one computed, the adjustment is not applied (see section V-A).

Algorithm 1 RWM operation

```
procedure NEW FLOW REGISTERED
    $C \leftarrow access\_link\_capacity$
    for $c$ in traffic_classes do
        $P_c \leftarrow class\_policy$
        $C_c \leftarrow C \cdot P_c$
        for $f$ in flow_table_c do
            if $WSCALE_f$ not recorded then
                skip to next flow
            end if
            $RTT_f \leftarrow estimated\_flow\_rtt$
            $BDP_f \leftarrow (C_c \cdot RTT_f)/len(flow\_table_c)$
            $RWND_{BDP_f} \leftarrow BDP_f/2^{WSCALE_f}$
            $RWND_f \leftarrow min(RWND_f, RWND_{BDP_f})$
        end for
    end for
end procedure
```

A. Use Cases

RWM's primary use case targets an enterprise edge gateway or router for WAN links. Our proposed architecture allows for dynamic adaptation of resource utilization, and combined with policy definition and traffic classification, is a powerful tool to be used in corporate networks. RWM is particularly appropriate to improve traffic control at ingress/egress WAN access links, especially when they represent the bottleneck—e.g. in site to site corporate communication. As previously exposed, RWM specifically addresses cases of self-induced congestion; while certain robustness in a real scenario has been proven through experiments (further detailed in section IV), its behavior has not been studied in networks where massive amount of cross traffic is the cause of intermediate congested nodes (see section V-C).

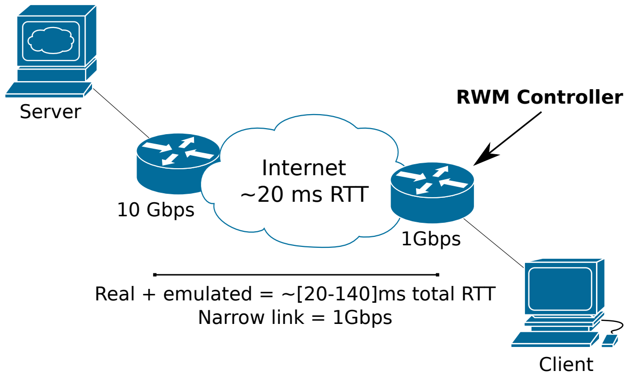


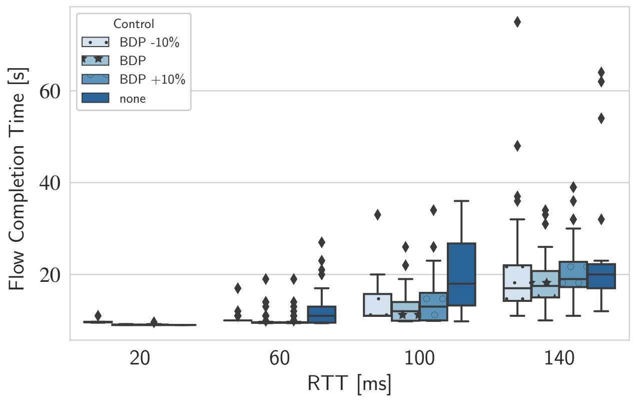
Fig. 2: Network topology and configuration for the experiments (RWM only in the client-side router).

IV. Experiments

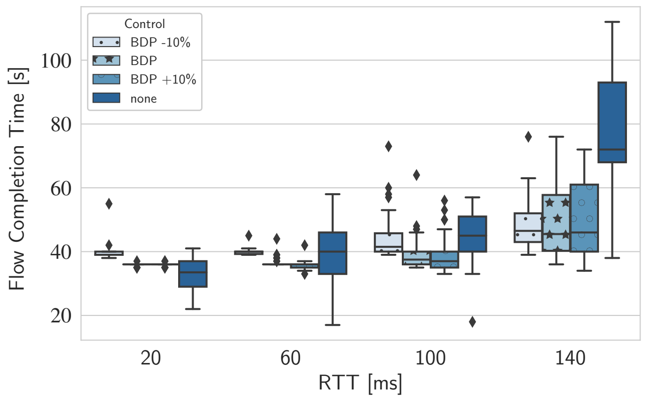
A series of experiments were conducted to validate the proposed architecture. The experimental evaluation addresses a scenario where all the traffic flows handled by the router are bulk data transfers belonging to a single traffic class. The objective of the experiments is to validate the benefits obtained when activating the controller in different network scenarios in terms of Flow Completion Time (FCT), goodput, total latency and fairness. In our testbed we emulate different network conditions to generate a range of BDP scenarios. In this evaluation we test the controller between two known locations and deploy one single point of control in the edge router close to the receiver; this edge router has the smallest access link between the two locations. We do not have control over the intermediate hops along the Internet path connecting the two locations. As such the BDP computed in the edge router is only dependent on the local bottleneck and does not take into account possible variability in the inner section of the network. To compensate, we define a $\pm\delta$ around the locally computed BDP value. We've set the value of δ to 10%, in order to study the response of the control when under and over estimating the network conditions.

A. Testbed

The testbed general configuration is shown in Figure 2. The testing environment includes two pairs of nodes in different



(a)



(b)

Fig. 3: Flow completion time for one (3a) and then four (3b) downloads starting simultaneously with or without the controller for different values of RTT.

geographic locations. Each pair is composed of a node that acts as a server or client and another one that acts as router/gateway for the paired instance. The nodes involved are all virtual machines running Debian Linux Stretch. One site has 1Gbps WAN access link, while the other has a 10Gbps one. The typical round-trip time over the Internet between the two sites is 20ms±0.2ms. In the 1Gbps link site is located the client; it connects to a web server connected through the 10 Gbps link. The virtual machines with the access link of 10Gbps are EC2 Amazon instances. Both instances are m5.xlarge type which are guaranteed to run on Intel Xeon Platinum 8000 series processors, have 4 virtual CPUs assigned and 16GB of RAM. The other two VMs with the 1Gbps access link run in our lab environment; each VM has 4 virtual CPUs and 4GB of RAM assigned. They are hosted on a server blade with 96GB of RAM, running an Intel Xeon CPU E5-2620 v4 with 8 physical cores with hyperthreading.

The only parameters modified at the endpoints are the maximum receive and send buffer size of TCP, which have been tuned to 60MB. This allows the endpoints to reach full theoretical link utilization in all BDP scenarios and avoids the endpoint receive window to become the bottleneck. The congestion control algorithm at the sender is CUBIC (Linux's default). On the router close to the client we use the Linux Traffic Control module called NetEm [8], to change the environment network conditions. We add delay to the link, while keeping the fixed capacity of 1Gbps. Latency introduced has a variability of ±3ms; this variability induces some packet reordering as consecutive packets scheduling can be delayed or anticipated to emulate transient network variability. We do not enforce any artificial loss through NetEm, relying on the intrinsic variability of the Internet between the two sites.

B. Evaluation

The experiments consist of HTTP GET transfers performed through the client with the `wget` tool. The server provides 1GB files through a `nginx` web server. The client acts as receiver and the server as sender. We combine the following parameters during testing:

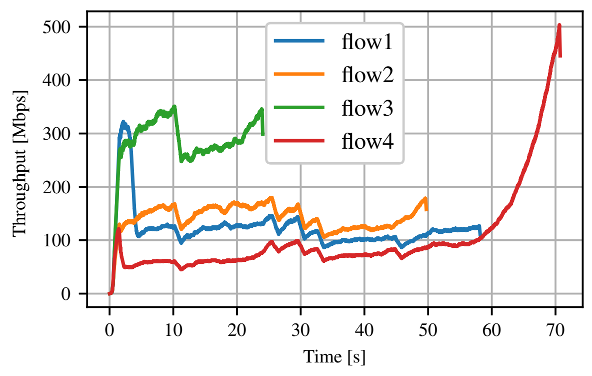
- BDP of the path - varied by means of controlling the latency in the client router. Total RTT varies between 20ms and 140ms.
- Number of concurrent transfers - one and four.
- Controller BDP set: i) exact BDP, ii) exact BDP-10%, iii) exact BDP $+10\%$, and iv) No Controller.

Each combination of parameters is tested 50 times for a total of over 3000 tests. We study the FCT of the downloads. We also recollect statistics from the endpoints by means of an `eBPF` [9] based analysis tool. In particular, in the endpoints we measure:

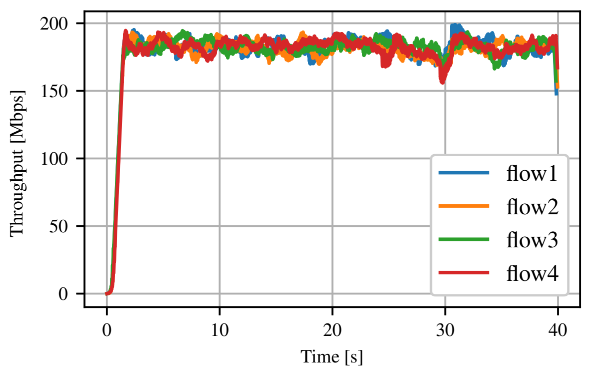
- The throughput of each download.
- The evolution of the RTT during the data transfer.
- The evolution of the receive window advertised by the receiver and the actual receive window seen by the sender (that could have been modified by our controller).
- The evolution of the congestion window of the sender.

We show the results of FCT using categorical boxplots (see Figure 3). Each category represents all the tests performed for a specific value of total RTT for a certain number of concurrent transfers. In each category four boxes are shown: three represent the results when enabling the controller and providing it different values for the BDP estimation as previously described; the fourth box represents the results without the controller.

1) Flow Completion Time: Figure 3a depicts the series of tests with one single transfer. We can observe that the range for the non-controlled scenario has considerable variability. The controller improves the behavior of the flows by keeping a more consistent rate during the whole transfer. While improvement is clear for the 60ms and 100ms scenarios, we can see that for 140ms the margin is reduced. Results suggest that, when increasing the RTT, and thus the BDP, the amount of in-flight data injected by the sender is enough to fill one or multiple queues along the path. This holds true even when controlling the receive window to match the nominal BDP. As a matter of fact we start measuring tail drop in our own router's buffer.



(a)



(b)

Fig. 4: Throughput of four concurrent transfers. In 4a a legacy TCP CUBIC scenario. In 4b the same experimental environment is being controlled by RWM.

On the other hand, we can see from Figure 3b that when adding multiple traffic sources the aggressiveness of the congestion control algorithm at the sender is enough to incur in a consistent performance penalty even at lower latencies. At 140ms of RTT is possible to observe that the 50th percentile for the FCT of the uncontrolled scenario is almost 1.5 times higher than the controlled scenario. The difference between the different levels of control applied is marginal with a clear trend: an underestimation of the BDP of 10% brings less variability but average higher values for FCT, while the best results are obtained when controlling at the ideal BDP point; overestimation shows lower 25th percentile values for higher RTT values, but brings more variability to the overall statistic. In this representation each data transfer counts as independent event. We computed the distance between the 50th percentiles of the different categories: RWM improvement in term of FCT in the one transfer scenario is up to 46%, while in the 4 transfers scenario the improvement goes up to 70%. Another relevant observation can be done by looking at the 140ms category in the scenario of Figure 3b: the 75th percentile of any of the controlled tests is better than the 25th percentile of the non-controlled case.

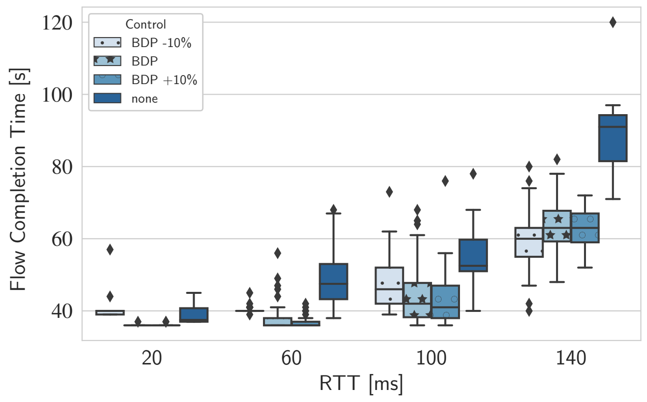


Fig. 5: Maximum flow completion time of each set of four concurrent downloads with or without the controller, for different values of RTT.

2) Fairness: In figure 4 we compare a non controlled experiment using CUBIC in the endpoints, with the same scenario deploying RWM in the receiver's edge router. Four downloads start simultaneously; total RTT is of 60ms. Figure 4a shows the throughput of the four connections having a very varied behavior with inconsistent performance. One of the flows is greedier and finishes faster, penalizing the congestion windows of the other three flows. The final FCT for the set is around 70 seconds. In Figure 4b the throughput for the test performed using the controller is shown. The throughput is consistent along the whole duration of the data transfer and the congestion windows always works above the level of the receive windows being enforced. The four flows reach fair sharing of the link capacity and all finish at the same time, taking around 40 seconds.

In Figure 5 we repropose the results seen in Figure 3b but this time considering only the FCT of the slowest of the group of four simultaneously started transfers. Figure 5 confirms the trend seen in the previous section but also provides stronger validation for the fairness properties of RWM.

3) Latency: Table I summarizes the statistics for the RTT of a subset of combinations of the tests performed. In case of lower latencies (20ms case) is possible to observe how the average standing queue induced by a loss-based congestion control is within the order of magnitude of the latency itself. The controller avoids buffering in any of the intermediate nodes. Mean value corresponds to the nominal RTT, and statistical variation is negligible. In all the cases shown, the variability of the non-controlled scenarios exhibits a higher standard deviation.

control	flows	std	mean	25th	50th	75th	max
none	1	**12.8**	**53.2**	42.8	56.0	64.4	71.2
none	4	**15.9**	**52.8**	41.9	58.5	65.1	71.0
RWM	1	**0.2**	**21.0**	21.0	21.0	21.0	21.3
RWM	4	**0.2**	**20.4**	20.3	20.4	20.6	25.5

TABLE I: Quartiles and standard deviation for the 20 ms RTT for some combinations of number of concurrent flows. For simplicity, statistics for the scenarios controlled with BDP$\pm\delta$ are not shown.

V. Discussion and Future Work

A. RWM Compliance with TCP

TCP is a transport-level protocol, where flow control and congestion control policies are applied by the endpoints. On-route packet processing and modification is not required by design. As a matter of fact, TCP has a checksum mechanism to detect errors that can originate in the network. RWM guarantees TCP data integrity by recomputing the TCP checksum of each modified packet. At the same time the flow control semantic is kept intact: if the original advertised receive window is lower than the value the controller wants to enforce, no changes are applied to the packet. This guarantees that the receiving endpoint can still apply flow control in case of need (e.g if it is not able to process the amount of data received).

B. Path Symmetry

The controller, in order to be able to estimate the connection RTT, to retrieve the window scaling factor and finally to apply the computed receiver window to all packets of the flow, needs to have visibility on packets from both directions of a TCP flow. This is usually the case in edge routers acting as gateways in corporate LANs, which is the typical use case envisioned for RWM as stated in section III-A. On the other hand this limits the applicability of the scheme as routers deeper in the Internet core could not have access to both directions of the flow due to path heterogeneity.

C. Available Bandwidth Estimation

Currently RWM bases the BDP computation exclusively on its locally available bandwidth and the RTT measured for the connection. In the testing environment presented in section IV the edge router manages the smallest Internet access link between the sender and the receiver. While this schema was shown to be effective in this scenario, there is no guarantee that it would be as effective in presence of consistent cross traffic deep down in the network. In such a case the congestion control of the sender will be triggered by packet losses happening in the network core, not just due to self-induced congestion. Effectiveness of RWM in these scenarios could be improved by developing a BDP estimator that takes into account transient network conditions by employing available bandwidth estimation techniques, predicting variations in network conditions so to control the flow with a more conservative window value before loss events could take place. Such an improvement for RWM is currently under development.

D. Distributed deployment of points of control

We have developed RWM for the following scenario: traffic across two remote sites traversing the Internet, with RWM deployed in just one of the two sites edge router. This solution does not require the controller deployed in more than one node. In a future study we will study the convergence properties of multiple sites topologies with an RWM deployed at the edge router of each site.

VI. Conclusions

In this paper we presented a novel controller for an edge router that improves end-to-end TCP connections' behavior by throttling the flows modifying the TCP receive window advertised: we call it Receive Window Modulation—RWM. RWM computes the window value to enforce by estimating the BDP of the connection based on its locally available bandwidth and the end-to-end RTT. RWM has been proven to enhance TCP goodput, while reducing dramatically the buffering caused in intermediate nodes by TCP loss-based congestion control mechanisms. As a consequence, bufferbloat is contained and TCP connections behavior in terms of latency improves. The experimental evaluation focuses on a scenario where the traffic is mostly composed of bulk data transfers. In this case RWM shows an improvement for average application goodput of up to 70%, while avoiding buffering in the intermediate nodes and consistently reducing latency in respect to legacy CUBIC TCP connections. The best results have been obtained in presence of multiple concurrent flows where the RWM schema is able to provide high level of fairness when sharing link resources. Future work includes the development of an available bandwidth detection mechanism to make RWM more robust to cross traffic. Furthermore we envision a study to optimize resource sharing between different classes of traffic.

Acknowledgment

This work was supported by the grant 2015DI023 in the framework of the Industrial PhD grants of AGAUR and Generalitat de Catalunya. The authors would like to thank Ignasi Errando for his valuable contribution to the definition of the problem addressed in this work.

References

[1] S. Ha, I. Rhee, and L. Xu, "CUBIC: A New TCP-friendly High-speed TCP Variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008.

[2] F. M. F. Wong, C. Joe-Wong, S. Ha, Z. Liu, and M. Chiang, "Improving user QoE for residential broadband: Adaptive traffic management at the network edge," in *Quality of Service (IWQoS), 2015 IEEE 23rd International Symposium on*. IEEE, 2015, pp. 105–114.

[3] L. Kalampoukas, A. Varma, and K. Ramakrishnan, "Explicit Window Adaptation: a Method to Enhance TCP Performance," vol. 1, pp. 242–251, 1998.

[4] T. Taleb, N. Kato, and Y. Nemoto, "An explicit and fair window adjustment method to enhance TCP efficiency and fairness over multihops satellite networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 2, pp. 371–387, 2004.

[5] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-Based Congestion Control," *Queue*, vol. 14, no. 5, pp. 50:20–50:53, Oct. 2016.

[6] M. Hock, R. Bless, and M. Zitterbart, "Experimental evaluation of BBR congestion control," in *2017 IEEE 25th International Conference on Network Protocols (ICNP)*. IEEE, 2017, pp. 1–10.

[7] P. Farrow, "Performance analysis of heterogeneous TCP congestion control environments," in *Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN), 2017 International Conference on*. IEEE, 2017, pp. 1–6.

[8] S. Hemminger *et al.*, "Network emulation with NetEm," in *Linux conf au*, 2005, pp. 18–23.

[9] IO Visor Project. BCC: BPF Compiler Collection. [Online]. Available: https://github.com/iovisor/bcc