Efficiency of a PID-based Congestion Control for High-speed IP-networks

Nikolai Mareev[1], Dmitry Kachan[1], Kirill Karpov[1], Dmytro Syzov[1],
Eduard Siemens[1] and Yurii Babich[2]
[1]Department of Electrical, Mechanical and Industrial Engineering,
Anhalt University of Applied Sciences, Bernburger Str. 55, 06366 Köthen, Germany,
[2]Department of Telecommunication Networks,
Odessa National O.S. Popov Academy of Telecommunications, Odessa, Ukraine
nikolai.mareev@student.hs-anhalt.de, {dmitry.kachan, kirill.karpov, dmytro.syzov, eduard.siemens}@hs-anhalt.de,
y.babich@onat.edu.ua

Keywords: Congestion Control, Transport Protocol, Ip Networks, Pid Control, Rmdt, High-Speed Networks

Abstract: The current situation in IP networks shows the need for new congestion control algorithms that can be flexible, scalable, and capable of avoiding additional queue delays caused by loading the bottleneck buffers. Most common internet flows use loss-based congestion controls, which can achieve high bottleneck bandwidth utilization and fair resource sharing but cause overload bottleneck buffers. In this paper we present an investigation of the performance of a PID-based congestion control solution for high-speed IP networks. It uses measurements of a round trip time and receiver delivery rate to reach and keep maximum available bottleneck performance and constant node buffer load caused by bottleneck queue on some low level. This algorithm can be effective in high-speed IP networks and delay-sensitive applications. It is designed to be flexible and scalable for different connectivity cases. This algorithm then is investigated on the example of RMDT.

1 INTRODUCTION

Network congestion occurs, when a receiving node is receiving more data than it can handle or forward to an output interface. It leads to a significant performance degradation: additional delays and massive packet losses. Congestion control algorithms are aimed to solve such problems. This is an automatic control of a sender's parameters, which describe performance of data send process, adaptability for different connection cases and the ability to share link resources fairly with other connections.

The aim of this paper is to present an investigation on PID-based congestion control in terms of CloudBDT and BitBooster projects. These projects use the Reliable Multi-Destination Transport protocol RMDT [1][2], where the results of the present work may be used.

The idea behind the usage of PID (Proportional – Integral – Derivative) control in congestion control algorithm lies in the fact that this type of control can be very flexible, scalable and adaptive. It can be easily extended by additional modules like auto tune loop or artificial neural network.

The main challenges for modern congestion control are: high bottleneck bandwidth utilization, low bottleneck queue delays, automatic scalability to different channel conditions (different bandwidth and different delays) [3], adaptation for sudden changes over connection like rerouting, applicability in wireless networks and resource sharing.

The rest of this paper is organized as follows: In section 2 a short overview of modern congestion control solutions and their main disadvantages is presented. Section 3 describes main states of PID-based congestion control solution and its principles. The experimental setup is presented in Section 4. Test results and evaluations are given in Section 5. Section

6 includes conclusions based on the evaluation results, further work and describes benefits of such solutions.

2 RELATED WORK

In [4] various approaches to TCP host-to-host congestion control algorithms and its evolution due to modern network sharing issues have been reviewed.

Loss based congestion control algorithms (Reno, Cubic) interprets packet losses as an indicator of a congestion. TCP Cubic [5] (which is the default congestion control in Linux kernels 2.6.19 and above.) can be very effective - with high bottleneck capacity utilization and fair resource sharing. However, they cause significant bottleneck queue delays and performance degradation in cases with tiny bottleneck queue buffers. Moreover, packet losses can be caused not only by a congestion in a network, but also by a link itself as well (e.g. wireless connections).

Another important solution is a delay-based congestion control, like TCP Vegas described in [6] and its future improvements. It is a proactive algorithm that uses bottleneck queue delay and packet losses as congestion indicator. Such strategy allows to predict a congestion before losses occur, caused by bottleneck buffer overload happen and also to keep queue delays on the levels, lower than the loss-based algorithms. Anyway, those algorithms have no aim to keep bottleneck buffer load at a low level, they keep it at some constant level. The most significant disadvantage of such algorithms is an unfair network resource sharing – especially with loss-based congestion control algorithms [7]. In addition, use of packet losses as a secondary congestion indicator can lead to the same problem of non-congestion caused losses as with pure loss-based algorithms.

BBR [8] algorithm (Bottleneck Bandwidth and Round-trip propagation time) is a new solution in congestion control. It uses round trip time and bottleneck bandwidth probing cycle to keep bottleneck queue load on a low level along with queueing delays and tries to reach effective bottleneck capacity utilization. Such technique under some conditions leads to a higher performance in comparison to loss-based or “delay-loss-based” algorithms. However, the probing cycle leads to data rate decrease and in some cases to unfair resource sharing [9].

3 PID-BASED CONGESTION CONTROL

A PID controller is a widely used control loop feedback mechanism, it continuously calculates an error value as a difference between a desired level of a controlled value (Setpoint, SP) and a measured process value (PV). It applies a correction based on proportional, integral and derivative terms. In case of this congestion control solution, the process value is round trip time. The correction can be done by changing the send data rate.

The first state of an algorithm is a “*Gain*” state (see figure 1), used to quickly estimate bottleneck bandwidth (BBW). PID congestion control requires presets of main parameters such as round trip time SP and factors for send data rate correction. To estimate that the algorithm has a second state named “*Manage*” state. The third, “*Control*” state is a PID-controller itself. Figure 1 illustrates main states of PID-based congestion control.

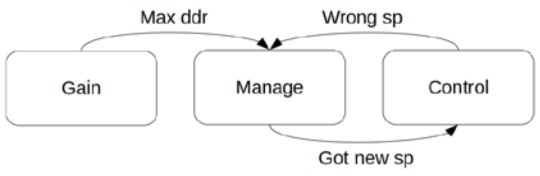

Figure 1: Main states of a PID-based congestion control.

3.1 Gain

Algorithm enters the *Gain* state at the very beginning of the transmission. It tracks the delivered data rate (DDR) and rapidly increases the send data rate. When last three reports show that there is no significant growth of delivered data rate, then congestion control switches to *Manage* state. *Gain* state enables the algorithm to quickly reach a bottleneck bandwidth limit and to make delay measurements for future processing.

3.2 Manage

Manage state tries to get the minimal round trip time (RTT) of a transmission by omitting the bottleneck queue buffer. It is achieved by decreasing data rate by half for 50 ms. It allows to set an acceptable round trip time setpoint (SP). For the current implementation, the acceptable setpoint is:

$$SP = max\ (minRTT + \alpha;\ 1.25\ minRTT), \qquad (1)$$

where α is the minimal growth of SP.

Minimal level value of SP (10 ms in this work) is caused by some instability of RTT measurements in the current solution. Otherwise, it can be even lower.

This state is also useful for resetting a setpoint if sudden rerouting is detected (RTT significantly dropped/raised). The third role of *Manage* state is fairndwidth sharing. For PID based congestion control, fair share is possible if chosen SPs of both links are almost equal.

3.3 Control

In the *Control* state, a modified PID digital controller tracks the delivery data rate and round trip time and tries to keep RTT near a setpoint by changing send data rate. If SP is not reachable for the last 10 packets or DDR has suddenly dropped down (more than 20% DDR drop), algorithm goes to *Manage* state to estimate a new RTT setpoint.

4 EXPERIMENTAL SETUP

Figure 2 shows testbed network topology. All tests have been performed in 40 GE Laboratory of Future Internet Lab Anhalt [10] (FILA).

The core element here is the WAN emulator Netropy 40G [11] that can be used to create an emulation of WAN links up to 40 Gbps throughput and up to 10^6 ms delay. Sender and receiver both run in Ubuntu 16.04 (kernel: GNU/Linux 4.13.0-17-generic x86_64) and are equipped with Intel(R) Xeon(R) CPU E5-2643 v4 3.40GHz, 64GB of RAM and 40000baseSR4/Full supported link modes on Emulex Corporation OneConnect NIC.

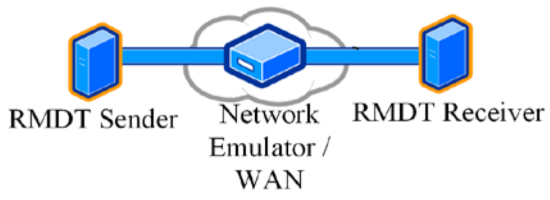

Figure 2: Testbed network topology.

5 EXPERIMENTAL RESULTS

Tests have been performed in next scenarios: single flow test with BBW 10 Gbps and {10, 50, 100, 150, 200}ms RTT; resource sharing test with two flows, bottleneck bandwidth 5 Gbps and 50 ms RTT. Queue management is set to drop-tail in all test. Figures from 3 to 7 demonstrate experimental results of congestion control: high bottleneck bandwidth utilization and queue load level control in high speed IP network with different base RTT (one way delay is one half of RTT). The fair resource sharing of proposed congestion control method is shown in Figure 8. All statistics are collected by WAN Emulator. It can collect only per-second mean statistics, which leads to unclear view of bottleneck buffer load level in the different states on some plots.

Results of first experiment with 10Gbps bottleneck bandwidth and 10 ms RTT are presented in figure 3. Such bottleneck buffer load level deviations (highlighted zones) are caused by the current RTT measurement solution in RMDT ver. 0.97 alpha, however this issue has no significant effect on bottleneck bandwidth utilization or performance of the control itself. 10 Gbps data rate was achieved with 6 Mbytes bottleneck buffer load in the *Control* state.

Second experiment with 10 Gbps bottleneck bandwidth and 50 ms RTT is shown in figure 4. The highlighted zone shows here the bottleneck buffer load during the *Gain* state (rapid growth of send data rate and bottleneck bandwidth estimation). It allows to make measurements of maxRTT and congestion reporting. 10 Gbps data rate was achieved with 13 Mbytes bottleneck buffer load in the *Control* state.

As shown in figure 5, in terms of next experiment with 10 Gbps bottleneck bandwidth and 100 ms RTT.

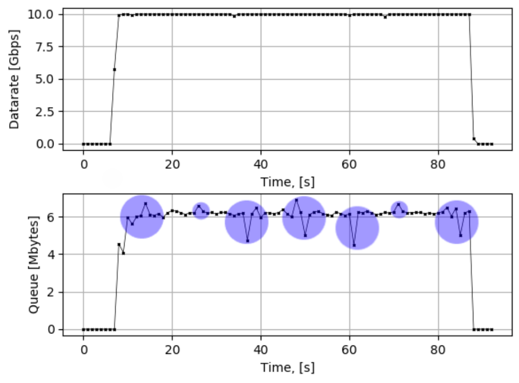

Figure 3: Test 1: BBW 10 Gbps, base RTT 10 ms, single flow.

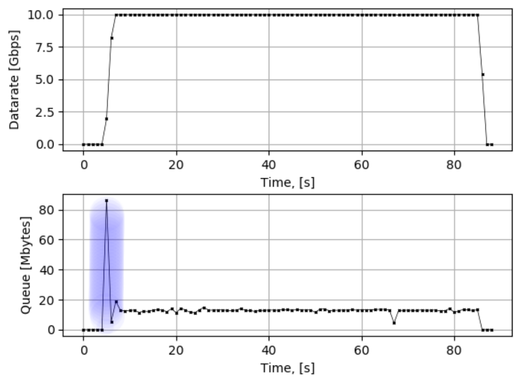

Figure 4: Test 2: BBW 10 Gbps, base RTT 50 ms, single flow.

10 Gbps data rate was achieved with 30 Mbytes bottleneck buffer load in the *Control* state.

Figure 6 demonstrate single flow experiment with 10 Gbps bottleneck bandwidth and 150 ms RTT. IP in high speed networks with large RTT. However, various RTT/ BBW cases require

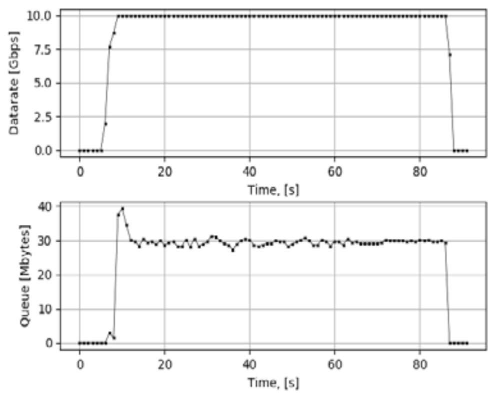

Figure 5: Test 3: BBW 10 Gbps, base RTT 100 ms, single flow.

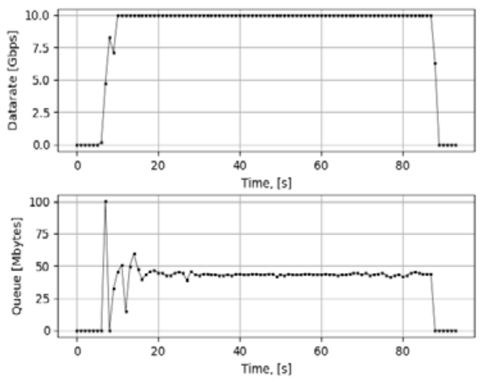

Figure 6: Test 4: BBW 10 Gbps, base RTT 150 ms, single flow.

different presets done by *Manage* state for PID-controller. Higher RTT require more bottleneck buffer memory in a relatively current realization of an algorithm (figures 3-7). 10 Gbps data rate was achieved with 43 Mbytes bottleneck buffer load in the *Control* state.

Last single flow experiment with 10 Gbps bottleneck bandwidth and 200 ms RTT is presented in figure 7. Highlighted zone shows the transition from *Manage* state (drop data rate and omitting bottleneck buffer) to *Control* state (growth of buffer load to SP). 10 Gbps data rate was achieved with 59 Mbytes bottleneck buffer load in the *Control* state.

Figure 8 shows fair network resource sharing test by two PID-based congestion control flows, with

lution has following features: it is scalable, keeps bottleneck buffer load on some low level and achieves high throughput with minimal losses. Fair resource sharing is achieved by dynamically

network is a system with high transport delays, however PID-based congestion control acts quite effective even

5 Gbps bottleneck bandwidth and 50 ms RTT. Highlighted zone shows overload of bottleneck buffers in the *Gain* state of flow 2. After *Gain* state cross-interference, both flows come to such SP‘s, that result in low bottleneck buffer load. 2.53 Gbps rate was achieved in sharing by flow 1 and 2.46 Gbps by flow 2 (5 Gbps in total). Bottleneck buffer load before interaction is near 14 Mbytes by only flow 1 and 17 Mbytes in sharing by two flows.

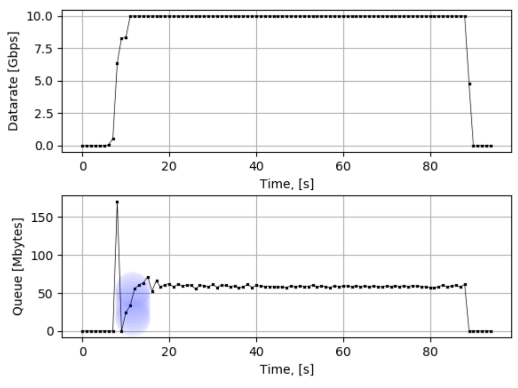

Figure 7: Test 5: BBW 10 Gbps, base RTT 200 ms, single flow.

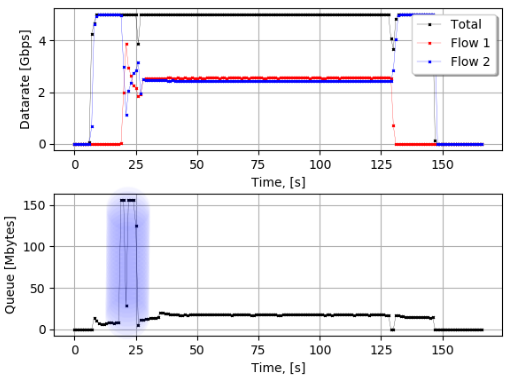

Figure 8: Test 6: BBW 5 Gbps, base RTT 50 ms, two flows in one link.

6 CONCLUSION & FURTHER WORK

This article presents the results of the investigation of a PID-based congestion control solution. Basic tests with UDP-based transport protocol show that proposed so

changing the RTT setpoint within the Manage state of the proposed algorithm. For long fat networks a PID-based congestion control is also can be used, but it requires additional RTT setpoint fitness algorithm, for keeping bottleneck buffers on lower level. It is under active development; next steps include additional auto tune loop for more precise scalability to any bandwidth and any delays; more intelligent setpoint management; fairness with TCP congestion control algorithms; boost performance in wireless networks.

ACKNOWLEDGMENT

This work has been funded by Volkswagen Foundation for trilateral partnership between scholars and scientists from Ukraine, Russia and Germany within the project CloudBDT: Algorithms and Methods for Big Data Transport in Cloud Environments.

REFERENCES

[1] E. Siemens, D. Syzov and D. Kachan, “High-speed UDP Data Transmission with Multithreading and Automatic Resource Allocation” Proc. of: the 4th International Conference on Applied Innovations in IT, (ICAIIT 2016), pp. 51-56, Koethen, 2016

[2] S. Maksymov, D. Kachan and E. Siemens, “Connection Establishment Algorithm for Multi-destination Protocol” Proc. of: the 4th International Conference on Applied Innovations in IT, (ICAIIT 2014), pp. 57-60, Koethen, 2016

[3] Eduard Siemens, Ralf Einhorn, Andreas Aust, Lars Fuerst, "Multi-Gigabit Challenges: Similarities between Scientific Environments and Media Production" Proc. of: The IASTED International Conference on Automation, Control, and Information Technology (ACIT-ICT 2010), Novosibirsk, Russia.

[4] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, “Host-to-Host Congestion Control for TCP,” Communications Surveys Tutorials, IEEE, vol. 12, no. 3, pp. 304–342, Third 2010.

[5] I. Rhee and L. Xu, “CUBIC: A New TCP-Friendly High-Speed TCP Variant,” Proc. of PFLDnet 2005, February 2005, Lyon, France.

[6] L. S. Brakmo, S. W. O’Malley, and L. L. Peterson, “TCP Vegas: New Techniques for Congestion Detection and Avoidance,” in SIGCOMM ’94. Techniques for Congestion Detection and Avoidance,” in SIGCOMM ’94.

[7] M. Hock, R. Bless, and M. Zitterbart, “Toward Coexistence of Different Congestion Control Mechanisms,” in 2016 IEEE 41st Conference on Local Computer Networks, November 2016, pp. 567–570.

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

[9] M. Hock, R. Bless and M. Zitterbart, “Experimental Evaluation of BBR Congestion Control,” in 2017 IEEE 25th International Conference on Network Protocols (ICNP), Oct 2017.

[10] FILA. Future Internet Lab Anhalt [Online]. Available from: https://fila-lab.de. 2017.11.12

[11] Apposite. Apposite Technologies: Linktropy and Netropy Comparison. [Online]. Available from: http://www.apposite-tech.com/products/index.html. 2017.11.12