Multipath TCP
Internet Draft
Intended status: Standard Track
Expires: July 2016

Mingwei Xu
Tsinghua University
Yu Cao
NDSC, China
Enhuan Dong
Tsinghua University
January 4, 2016

Delay-based Congestion Control for MPTCP
draft-xu-mptcp-congestion-control-03.txt

Abstract

This document describes the mechanism of wVegas (weighted Vegas), which is a delay-based congestion control for MPTCP. The current congestion control algorithm of MPTCP, LIA, achieves only course-grained load balancing, since it is based on packet loss event. On the contrary, wVegas adopts packet queuing delay as congestion signals, thus achieving fine-grained load balancing. Compared with loss-based algorithms, wVegas is more sensitive to the changes of network congestion and thus achieves more timely traffic shifting and quicker convergence. WVegas has been implemented in the Linux Kernel and is part of the UCLouvain's MPTCP implementation now.

This Internet-Draft will expire on July 4, 2016.

Copyright Notice

Table of Contents

1. Introduction 3
1.1. Requirements Language..... 4
1.2. Terminology 4
2. Weighted Vegas Algorithm..... 6
3. Practical considerations..... 8
4. Discussion 9
5. Security Considerations..... 10
6. IANA Considerations 10
7. References 10
7.1. Normative References..... 10
7.2. Informative References..... 10

1. Introduction

Performing congestion control independently on each path cannot guarantee the fairness for multipath transportation. So the major goal of multipath congestion control is to couple all the subflows belonging to a flow in order to achieve both fairness and efficiency. The current MPTCP adopts a congestion control algorithm called Linked Increases algorithm [RFC6356]. It provides the ability to shift traffic from more congested paths to less ones. However, it achieves only course-grained load balancing, since it uses the packet losses as the signal of network congestion and will shift traffic only after the loss event. Other alternative congestion control algorithms, such as OLIA [OLIA] or Balia [BALIA], have the same way to judge the congestion. These proposals lack the finer-grained information related to the degree of congestion.

In this draft, we introduce weighted Vegas (wVegas), which is a delay-based multipath congestion control algorithm. Comparing to LIA, OLIA and Balia wVegas adopts packet queuing delay as congestion signals, which is more sensitive to the changes of network congestion, thus achieving fine-grained load balancing. [WVEGAS]

WVegas is developed using the network utility maximization model [ADMTQM]. By solving the maximization problem, we get a general framework for designing an algorithm of multipath congestion control, which constitutes the Congestion Equality Principle and an approximate iterative algorithm [WVEGAS]. The Congestion Equality Principle illustrates that a fair and efficient traffic shifting implies every flow strives to equalize the extent of congestion that it perceives on all its available paths. And the approximate iterative algorithm makes the solution practical in real networks. The wVegas is precisely derived from the framework.

As the name shows, wVegas is originated from TCP-vegas [VEGAS] that measures packet queuing delay to estimate the extent of network congestion. TCP-vegas has two configurable parameters alpha and beta, for adjusting the congestion window during the congestion avoidance phase. Since the two parameters are commonly very close to each other, wVegas uses only one parameter for brevity. The design philosophy of wVegas is depicted as follows. First, WVegas performs in the same way as TCP-vegas on each path. Second, each flow has a fixed sum of parameters alpha, in spite of the number of subflows the flow has. Third, wVegas adaptively adjusts the parameter alpha resulting in the change of the transmission rate of the related subflows so as to equalize the extent of congestion on the path.

WVegas has been implemented in the Linux Kernel and is part of UCLouvain's MPTCP implementation now [MPLKI]. In [VEGAS], we study the traffic shifting ability of wVegas, reveal that it can guarantee the intra-protocol fairness and show the domino effect which is an indication of good performance.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.2. Terminology

Regular TCP: The standard version of TCP, which is currently restricted to a single path per connection.

Multipath TCP (MPTCP): A modified version of the regular TCP that provides the ability to simultaneously use multiple paths between peers.

LIA: The Linked-Increases Algorithm of MPTCP. [RFC6356]

OLIA: The Opportunistic Linked-Increases Algorithm for MPTCP. [OLIA]

Balia: Balanced Linked Adaptation Congestion Control Algorithm for MPTCP [BALIA]

WVegas: Weighted Vegas, which is a delay-based multipath congestion control algorithm for MPTCP. [WVEGAS]

Subflow (path): A single path TCP connection always belonging to a MPTCP connection.

Expected sending rate: The best rate that a single path flow can get under the current congestion window condition.

Actual sending rate: The actual rate that a single path flow can get under the current congestion window condition.

Diff_r: For subflow r, the difference between Expected sending rate and Actual sending rate.

Cwnd_r: The congestion window on a subflow r (maintained in packets).

Ssthresh_r: The slow start threshold of a subflow r.

Rtt_r: The average RTT in the last round on a subflow r.

Rate_r: The rate of subflow r, which is used to estimate the equilibrium rate of subflow r.

Base_rtt_r: The RTT of a subflow r when the sub-connection is not congested.

Alpha_r: TCP-vegas tries to keep the extra bytes between two configurable parameters in a single path flow. For subflow r, we use only one parameter, alpha, for brevity.

Total_alpha: The total bytes backlogged in the network for all subflows belonging to one MPTCP flow.

Weight_r: The rate of subflow r divided by the total rate among all subflows belonging to one MPTCP flow.

Gamma: When the Actual sending rate falls below the Expected sending rate by the gamma threshold, TCP-vegas changes from slow start to congestion avoidance phase.

Queue_delay_r: For subflow r, queue_delay_r records the minimal queuing delay measured after the last backoff.

2. Weighted Vegas Algorithm

In this section, we introduce wVegas. WVegas is a delay-based congestion control algorithm and also uses the unmodified TCP behavior in the case of loss event. WVegas is an alternative for LIA, the current congestion control of MPTCP.

The algorithm mainly applies to the congestion avoidance phase and, in slow start phase, it also add a chance to enter the congestion avoidance phase earlier, which is similar with the implementation of the TCP-vegas [VEGAS]. The decrease of the congestion avoidance phase, the fast retransmit and fast recovery algorithms are the same as TCP [RFC5681].

The following operations of wVegas must be performed on the end of each transmission round.

For a subflow r, the difference between the Expected sending rate and Actual sending rate is calculated as follows:

```
diff_r = (cwnd_r / base_rtt_r - cwnd_r / rtt_r) * base_rtt_r
```

If the subflow is in the slow start phase and the diff_r is larger than gamma, it must enter the congestion avoidance phase. This operation is inherited from TCP-vegas [VEGAS] and can be achieved by setting the ssthresh_r to (cwnd_r - 1). On the other hand, if the diff_r is no more than gamma in slow start phase, wVegas must act the same way as TCP [RFC5681].

In the congestion avoidance phase, if the diff_r is no less than alpha_r, the rate_r must be updated. The weight_r and the alpha_r must be tweaked before the window adjustment and the improvement of the base_rtt_r accuracy. The rate_r , the weight_r and the alpha_r must be calculated as follows:

```
rate_r = cwnd_r / rtt_r                    (1)

weight_r = rate_r / SUM(rate_r)            (2)

alpha_r = weight_r * total_alpha
```

SUM(rate_r) is the total rate of all subflows belonging to one MPTCP flow. After the tweak of the alpha_r, if the tweak is needed, the cwnd_r must be adjusted as follows:

- If diff_r is larger than alpha_r, then

cwnd_r = cwnd_r - 1

- If diff_r is less than alpha_r, then

cwnd_r = cwnd_r + 1

The last task wVegas has to do is to try to drain link queues. This operation is to improve the accuracy of base_rtt_r. And the specific method is described in [DRLK]. The idea is to make congestion window back off once detecting the queuing delay is larger than some threshold, so that the bottleneck link can drain off the backlogged packets. And thus all the flows involved have a chance to obtain the more accurate propagation delay.

First, wVegas has to calculate the current queuing delay as follows:

queue_delay_r = rtt_r - base_rtt_r

If the current queuing delay is less than the saved queue_delay_r, the queue_delay_r must be replaced by the current one. And if the current queuing delay is two times larger than queue_delay_r, the following operations must be performed:

cwnd_r = cwnd_r * 0.5 * base_rtt_r / rtt (3)

3. Practical considerations

In practice, it is difficult to get the RTT of a subflow r when the sub-connection is not congested. WVegas should set the base_rtt_r to the minimum of all the measured round trip times. The total_alpha should be implemented as a configurable parameter.

Equation (1) and (2) imply that the rate and the weight are floating point values. However, in many kernels, floating point operations are disabled. There is an easy way to approximate the above calculations using integer arithmetic.

Let rate_scale be an integer. When computing the rate, use cwnd_r * rate_scale instead of cwnd_r and the related operations can be done in integer arithmetic. The rate_r should be calculated as follows:

```
rate_r = cwnd_r * rate_scale / rtt_r
```

The rate_scale implies the precision we need for computing the rate. With this change, the rate can be stored as an integer variable. Besides, when we need to calculate the sum rate of all subflows belonging to one flow, the operations can be done in integer arithmetic.

When updating the weight_r, we also need a weight_scale to avoid floating point operations. So the weight_r should be computed as follows:

```
weight_r = rate_r * weight_scale / SUM(rate_r)
```

The weight_scale supplies a similar function with rate_scale. With the weight_scale, alpha_r can be much more accurate. But it also need scale down as follows:

```
alpha_r = weight_r * total_alpha / weight_scale
```

For the sake of brevity, we combine the rate_scale and weight_scale to one scale parameter. We name it wvegas_scale. It would be better to set the wvegas_scale as a power of two, which allows faster shift operations rather than multiplication and division.

In equation (3), the multiplication by 0.5 can be implemented by shift operation.

4. Discussion

Congestion Equality Principle shows that a fair and efficient traffic shifting implies that every flow strives to equalize the extent of congestion that it perceives on all its available paths. This principle has been proved in [WVEGAS]. By instantiating the approximate iterative algorithm, weighted Vegas (wVegas), a delay-based algorithm for multipath congestion control, was developed, which uses packet queuing delay as congestion signals, thus achieving fine-grained load balancing. Our simulations show that, compared with loss-based algorithms, wVegas is more sensitive to the changes of network congestion and achieves more timely traffic shifting and quicker convergence. Additionally, as it occupies fewer link buffers, wVegas rarely causes packet losses and shows better intra-protocol fairness.

5. Security Considerations

Security considerations discussed in [RFC6181] and [RFC6356] are to be taken into account.

6. IANA Considerations

This document has no actions for IANA.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.

7.2. Informative References

[DRLK] D. Leith, R. Shorten, G. McCullagh, L. Dunn, and F. Baker, "Making Available Base-RTT for Use in Congestion Control Applications", IEEE Communications Letters, vol. 12, no. 6, pp. 429-431, Jun. 2008.

[RFC6356] Raiciu, C., Handley, M., and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols", RFC 6356, October 2011.

[OLIA] R. Khalili, N. Gast, M. Popovic, J.-Y. Le Boudec, "Opportunistic Linked-Increases Congestion Control Algorithm for MPTCP", draft-khalili-mptcp-congestion-control-05.

[BALIA] A. Walid, Q. Peng, J. Hwang, S. Low, "Balanced Linked Adaptation Congestion Control Algorithm for MPTCP", draft-walid-mptcp-congestion-control-03.

[MPLKI] UCL, Louvain-la-Neuve, Belgium, "MultiPath TCP-Linux kernel implementation," 2013 [Online]. Available: http://mptcp.info.ucl.ac.be/.

[WVEGAS] Y. Cao, M. Xu, X. Fu, "Delay-based congestion control for multipath TCP", ICNP 2012.

[ADMTQM] S. H. Low, "A Duality Model of TCP and Queue Management Algorithms", IEEE/ACM Transactions on Networking, 11(4):525-536, Aug. 2003.

[VEGAS] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: new techniques for congestion detection and avoidance, " in Proc. of ACM SIGCOMM, 1994, pp. 24-35.

[RFC6181] Bagnulo, M., "Threat Analysis for TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6181, March 2011.

Authors' Addresses

Mingwei Xu
Tsinghua University
Department of Computer Science, Tsinghua University
Beijing 100084
P.R.China

Phone: +86-10-6278-5822
EMail: xmw@cernet.edu.cn

Yu Cao
NDSC
National Digital Switching System Engineering and Technological Research Center
Zhengzhou 450002
P.R.China

Email: caoyu08@tsinghua.org.cn

Enhuan Dong
Tsinghua University
Department of Computer Science, Tsinghua University
Beijing 100084
P.R.China

Email: deh13@mails.tsinghua.edu.cn