International Journal of Future Generation Communication and Networking
Vol.7, No.1 (2014), pp.97-104
http://dx.doi.org/10.14257/ijfgcn.2014.7.1.10

Delay-based Congestion Control for Multipath TCP

Wenlan Guo[1], Jin Huang[1] and Yun Zhang[2]

[1]*School of Computer Science and Technology,Harbin University of Science and Technology, Harbin, China*
[2]*Department of Electrical and Computer Engineering, University of Texas at Austin*
guowenlan@sohu.com, huangjin0706@126.com, zhangyun@utexas.edu

Abstract

A single flow stream could be divided into several sub-flows to be sent across multiple paths. This method obtains obvious advantages against traditional TCP as it maintains higher reliability and makes bettaer use of the network resources. Congestion control algorithm is an essential part to be considered. This paper first reviews the existing multipath TCP congestion control algorithms and then analyzes and formulates the goals and problems need to be achieved and solved. A delay-based congestion control algorithm named Weighted Vegas (wVegas) is provided. Finally, two possible modifications are demonstrated including adjusting the congestion control window according to how far the path is from congestion and redefining the behavior when loss occurs.

Keywords: *Congestion control; Multipath TCP; Weighted Vegas*

1. Introduction

There are three goals during the design of multipath TCP congestion control algorithm. The first one is the efficiency which means it must ensure that he throughput of all the sub-flows of one multipath connection should have better all at least the same performance as a single TCP flow on the best path. The second goal is known as fairness which means that all the sub-flows on one link should not take more capacity than a single TCP flow would get. The last requirement is to utilize each path dependent on the congestion on the path. In another word, it should be able to balance the traffic based on the indication of congestion, could shift traffic from more congested path to less congested path.

A number of multipath TCP algorithms have already been proposed. According to the method to split the flow over multiple paths, they could be divided into uncoupled, coupled and semi-coupled congestion control algorithm. The uncoupled algorithms apply congestion control on each path independently. This kind of algorithms is difficult to ensure the fairness, like Ptcp [1], CMT-SCTP [2], and M/TCP [3]. mTCP [4] also performs uncoupled congestion control on each path. In an attempt to detect shared congestion at bottlenecks it computes the correlation between fast retransmit intervals on different subflows. It is not clear how robust this detector is.Some uncoupled methods like EWTCP [5] applies weight on each sub-flow could ensure fairness, but it could hardly make full use of network resource. The fully coupled algorithms will couple the congestion control on each involved path. However it leads to the result that the traffic would totally shift to the less congested flow which would sometimes causes link to shut down. The robust algorithm always applies semi-coupled method in which

ISSN: 2233-7857 IJFGCN

the traffic will prefer the less congested paths while keep a reasonable traffic on the congested ones, such as MPTCP [6, 7].

Most of current multipath TCP congestion control algorithms use the loss of packets as the indication of the congestion. However it provides congestion detection, but not the avoidance. Network resources have been wasted if the congestion control starts to shift the traffic when the loss occurs, as losing packet indicates the path has reached congested level. One more problem is the buffer overflow loss in the loss-based congestion control as the sending window would increase infinitely if no loss happens. Therefore, another approach is proposed which uses queuing delay to indicate extension of the congestion. It could balance the traffic as the path approaches the congested situation which could avoid loss and congestion to some degree. The buffer overflow loss could also be avoided [8]. The following sections proposed a new delay-based congestion control algorithm for multipath TCP.

2. Congestion Control Goal Formulation

Assuming the network holds a set of L of physical links with the finite capacity $c_1 (1\ belongsL\ )$. A path $r(belongs\ \ to\ \ R)$ holds a set of link L_r. a_{1r} could be used to presents the relationship between the paths and the links. If l belongs to the set of links belongs to path L_r, then the $a_{1,r} = 1$, otherwise it stays zero. Each flow s belongs to a set of flow S would hold a set of path R_s. $b_{s,r}$ could be used to presents the relationship between the paths and the flows. If path r is one of the transmission tube of flow $s(r\ belongs\ \ to\ R_s)$, then the $b_{s,r} = 1$, otherwise it would stay zero. $x_{s,r}$presents the transmission rate of flow s on the path r. and $Y_s = \sum_{reR} x_{s,r}$ denotes the total transmission rate. Then assuming an increasing, strictly concave function $U(y_s)$ named as utility function. This function denotes how much benefit the user could be obtained, if the flow s has a total transmission rate y_s. The congestion control goal of the multipath TCP is to maximizing the utility function within the finite link capacity. It could be formulated as below.

$$\max \sum_{ses} U(y_s)\ \ y_s = \sum_{reR_s} x_{s,r} \sum_{r \in R} a_{1,r} x_r < c_1 \tag{1}$$

Our objective is to find the optimal solution for each $x_{s,r}$. The Lagrangian function could be used to find the optimal transmission rate.

$$L(x,\varphi) = \max \sum_{ses} U(y_s) + \sum_{l \in L} \varphi_1 (C_1 - \sum_{r \in R} a_{1,r} x_r) \tag{2}$$

φ_1 could be explained as the price for the transmission on the link l. The physical meaning of the Lagrangian function in multipath TCP congestion control could be explained as that $\sum_{l \in L} \varphi_1 (c_1 - \sum_{r \in R} a_{1,r} x_r)$ presents the total price for the utilization of the left bandwidth of the links and the $\max \sum_{s \in S} U(y_s)$ presents the maximum benefit to transmit all the flows which is in another word that indicates the minimum price to transmit at rate y_s. So the

target is to obtain the minimum value of the Lagrangian function. After some transformation of the Lagrangian function, the following function could be obtained.

$$L(x, \varphi) = \sum_{ses} \max U_s(\sum_{r \in R_s} x_{s,r}) - \sum_{r \in R} q_r x_{s,r} + \varphi_c \quad (3)$$

$$q_r = \sum_{i \in R} a_{l,r} \varphi_1 \quad (4)$$

q_r could be seen as the accumulated link price of the path. And then the Lagrangian function could be divided into two parts.

$$G_s(\varphi) = \max U_s(\sum_{r \in R_s} x_{s,r}) - \sum_{s \in R_s} q_r x_{s,r} \quad (5)$$

$$D(\varphi) = \sum_{s \in S} G_s(\varphi) + \varphi_c \quad (6)$$

From the above equations, the optimal congestion control goal could be divided into two goals. A master goal to minimize the $D(\varphi)$ over the whole network and several sub-goals for each flow in the network. The physical meaning of $G_s(\varphi)$ could be explained as the difference between the expected cost using a specific transmission rate and the sum of actual cost on each path. They are independent local optimization goals only depending on the local path price q_r. Therefore the each path using the path price and the optimal minimum cost assigned by the network to find the current optimal transmission rate and then board cast this rate to the whole network. The network would find the new optimal price for each link and the network then assign new optimal cost for each flow according to the new path price and the transmission rate. Executing the above procedure itinerantly. The optimal situation for both transmission rate and path price could arrive. This itineration of congestion control could be formulated as below.

$$\frac{dG_s(\varphi)}{dx_{s,r}} = U_S'(\sum_{r \in R_s} x_{s,r}) - q_r \quad (7)$$

$$x_{s,r}(n+1) = x_{s,r}(n) + \theta(U_s'(\sum_{r \in R_s} x_{s,r}) - q_r), r \in R_s \quad (8)$$

$$\varphi_1(n+1) = \varphi_1(n) - \gamma(c_1 - \sum_{r, l \notin L_r} a_{l,r} x_{s,r}) \quad (9)$$

For the first equation, $U_s'(\sum_{r \in R_s} x_{s,r})$ stands for the expected price when transmit using a single path at a specific transmission rate and stands for the actual price. The difference between them is used as a factor to determine how much improvement we need to made on the transmission rate. Another factor determines the step (how fast to approach the optimal value). After obtaining the new optimal transmission rate, the link price is modified according to the left link capacity. Another factor determines how fast the link price changes. Therefore, if the utility function, the expression of the path

price and the step factors could be determined, the congestion control algorithm could be designed.

3. Weighted Vegas Congestion Control Algorithm

Given that the BASE_RTT presents the minimum RTT has been measured so far which means the RTT with no queuing delay. RTT presents the average RTT during the previous round. The CWND presents the size of the congestion control window. The DIFF presents for how many packages are backlogged in the queue to say the length of the queue. The DIFF is calculated as below after the end of each round.

$$\begin{aligned} DIFF &= \left(\frac{CWND}{BASE\_RTT} - \frac{CWND}{RTT}\right) BASE\_RTT \\ &= CWND\,(RTT - BASE\_RTT) \Big/ RTT \end{aligned} \tag{10}$$

The DIFF could be defined as $a_{s,r}$ presents the backlogged package, the price of the path expresses as $q_r = RTT - BASE\_RTT$ and the transmission rate is expressed as $\frac{CWND}{RTT}$. The above equation could be written as below:

$$x_{s,r} = \frac{a_{s,r}}{q_r},\ r \in R_s \tag{11}$$

When the network reaches the optimal situation, the price of each path approaches the equal value, say q_s. And the total rate of a flow could be expressed as $y_s = \frac{a_s}{q_s}$ where the a_s represents the total backlogged packages over all path for a flow. And in the optimal situation, the expected price $U_s'(\sum_{r \in R_s} x_{s,r}) = q_s = \frac{a_s}{q_s}$ (proved in [9]) and the utility function could be obtained as $U(y_s) = a_s \log y_s$. Then assign the increasing step factor $\theta = \frac{x_{s,r}(n)}{q_s}$. Put the expression defined into the transmission rate equation in the previous section, the optimal transmission rate for one itineration could be obtained as below.

$$x_{s,r}(n+1) = \frac{x_{s,r}(n)}{y_s}\frac{a_s}{q_r} = \frac{k_{s,r}(n)a_s}{q_r} \tag{12}$$

$k_{s,r}(n) = \frac{x_{s,r}(n)}{y_s}$ is expressed as the weight of flow s on the path r. From the above equation, it could also be obtained that the new expected backlogged packages of flow s on path r are $a_{s,r}(n+1) = k_{s,r}(n)a_s$. Therefore the expected backlogged packages are treated as the Weighted Vegas in this algorithm.

The basic algorithm could be shown as below. The system would measure the minimum round time BASE_RTT of each path when there is no queuing delay. During each itineration, the average round time RTT is measured and used to calculate the backlogged package in the queue. Compare the actual backlogged packages and the expected backlogged packages obtained from the path weight. If there are more actual backlogged packages, it indicates the path is more congested than expected and reduce the sending window size. Otherwise it indicates the path is less congested than expected and the congestion control window should be expanded. After obtaining the new window size (transmission rate), recalculate the weight and expected backlogged packages and repeat the same itineration until the actual number of backlogged packages reach the expected one.

The total backlogged packages of each flow is never influenced by the number of sub-flow it divided, it is a fixed value assigned by the network when the flow arrives. The expected number of packages backlogged on each path is determined by how much weight the path occupies corresponding to the flow which is decided by the ratio of the rate the flow assigned to a specific path. If one path is occupied by a number of flows, the number of packages backlogged in the queue buffer will decide how much bandwidth the sub-flow would take on a specific path. Each path contain several sub-flows will hold a congestion window, the increase of the congestion window (increased transmission rate would be assigned to each sub-flows according to their backlogged package ratio. And one more thing, the increase transmission rate on one path would cause the increase of the weight the path occupies and backlogged packages on this path. The weight of other paths and the backlogged packages would drop which would also cause a dropping transmission rate (transmission rate shifting). This ensures the load balancing. Both the fairness and efficiency could be ensured from above analysis using the Weighted Vegas.

4. Implementation of Weighted Vegas

The paper gives a specific implementation of the Weighted Vegas Congestion Control Algorithm. In this implementation, it gives a variable array **equilibrium_rates[r]** to store the transmission rate of path r to calculate the weighted of the sub-flow, a **queue_delay[r]** to store the minimum queue delay of path r and **alpha[r]** to store the expected backlogged package of path r. The pseudo code and the analysis of the specific algorithm are shown as below.

In the Initialization phase:

- **total_alpha <- fixed number----** The network would assign a fixed number of backlogged packages to the coming flow and save it in the total_alpha.
- **Equilibrium_rates[r] <- 0, queue_delay[r] <- 0, alpha[r] <-random number**
- **Base_RTT[r]----**Measure the minimum RTT of path with no queuing delay

During the congestion avoidance phase after each round of path r:

- **RTT <- Sampled_rtts[r]/Sampled_num[r]----** Calculate the average RTT of the previous round
- **Diff <- cwnd[r] (RTT-Base_RTT[r]) / RTT----**Calculate the actual number of backlogged packages of path r. It is the queuing delay indicates extension of congestion
- **If Diff>=alpha[r] then**
- **Equilibrium_rates[r]<-cwnd[r]/RTT**
- **Adjust weight[r]<-Equilibrium_rates[r]/total_rate**
- **alpha[r]<-weight[r] multiply total_alpha**
- **alpha[r]<-max (alpha[r], 2) end if**

This if block is used for updating the transmission rate, weight and expected backlogged packages of path r. The transmission rate and the weight is only updated when the number of actual backlogged packages is larger than the number of expected backlogged packages which means the path is more congested than expected. Because when the actual backlogged packages are not as much as expected backlogged packages, the path could tolerate more capacity. The weight of path would also increase and the expected backlogged packages would be more than previous expected. However the previous expected number of backlogged packages has not been even exceeded, so it holds no need to update the backlogged packages in the queue until the previous one is exceeded. This will definitely save the network resource and ensure a faster convergence. The last action in if block ensures that even if the path occupies no weight, it is still not abandoned and assigned a reasonable load.

- **If Diff<alpha[r] then cwnd[r] <- cwnd[r]+1**
- **If Diff>alpha[r] then cwnd[r] <- cwnd[r]-1**

----Update the sending window size according to the relationship between expected backlogged packages and actual backlogged packages.

- **q<-RTT - Base_RTT**
- **if queue_delay[r]=0 or queue_delay[r]>q then queue_delay[r] <- q**
- **if q>2 multiply queue_delay[r]**
- **backoff_factor <- 0.5 multiply Base_RTT[r]/RTT**
- **queue_delay[r]<-0**

----The RTT is obtained from the average RTT value of the previous round. Each RTT value is obtained when each package in a round is sent. If the RTT value covers a large range, say that the difference between the largest RTT value and smallest RTT value is large, the measured result would be inaccurate which would definitely induce wrong action on the congestion control value. Therefore, the minimum queuing delay so far would be recorded. When the new measured queuing delay is several times larger than the smallest one, the congestion control would reduce according to a adaptive factor which would reduce the weight of the path and drain off the backlogged packages. After draining off the backlogged package, it is possible for the path to measure more accurate RTT.

- **cwnd[r] <- max (cwnd[r], 2)**---Ensure that no path is abandoned totally

If package loss occurs on path r:

- **equilibrium_rates[r] <- 0, queue_delay[r] <- 0**----When loss occurs, the path is

abandoned. All the traffic is shifted to other paths. The weights of other paths remain the same.

5. Modification and Problem

One problem lies on the accuracy of the RTT measurement. When the arriving rate is much higher than the transmission rate on a path. It could hardly be able to obtain the accurate RTT. Assuming that a new arrival flow assign the backlogged packets on a path which could be seen as an arrival rate R_c. The left bandwidth of the path is A. The queue will build up at a speed $R_c - A$. The RTT is measured during each packet is sent, assuming the current transmission rate is R and packet length is L, the RTT measuring period is $\frac{L}{R}$. It must be ensured that the increase of the queuing delay detected before the overflow of the buffer to

give the corresponding congestion control. Therefore, the boundary $\frac{L}{R} << \frac{B}{R_C - A}$ [10] holds for the delay-based congestion control, or the measure RTT would be incorrect which lead to wrong action. Sometimes will even cause loss due to overflow. The arriving rate of the path can never be too large which is a limitation to the Weighted Vegas algorithm.

One possible modification could be made on the how to change the congestion control window according to the queuing delay. In the algorithm discussed in the previous session, it calculate the actual backlogged packages and compare it with the expected ones to determine whether increase or decrease the sending window by 1. However, the queuing delay could do more than just judging whether the path is congested or not, it could also tell how far it is from the situation of congestion (the expected backlogged package in the previous algorithm). When it is still far from the congestion, the congestion window could be expanded greatly during each round. As it approaches the congestion situation, the increase of the sending window should be made smaller and smaller. It holds the same behavior when the path has already been congested and reduces the sending window size accordingly. This modified point could obvious accelerate the process of converging. To realize this idea, several thresholds could be made on the difference between the expected backlogged packages and the actual backlogged packages and assign reasonable sending window changing sizes to each interval.

Another modification could be made on the behavior when loss occurs. As we know, congestion of the path is not the only reason for the loss of package. The error occurs in the received package will also lead to a missing message. In the algorithm discussed above, the path is abandoned directly after the loss package occurs. This would be too drastic and will waste the bandwidth if the loss is not due to congestion of the network. To obtain a better performance, it should determine the behavior according to how far the path is from the congestion when loss package occurs. When the difference between expected backlogged packages and actual backlogged packages (the indication of congestion) is very large, the loss is probably caused by the receiving error or other reasons. The sending window could be just reduced by an adaptive factor, to say $\mu \times (alpha[r] - Diff) / alpha[r]$ (μ *is the reduce fact*). When it indicates that it has already exceeded the expected balance value (already been congested), the network could assign a reasonable small sending window size. The reason for the small sending window is that if the path is fully shut down, there would be never any traffic on the path and it would be never used even when the path is not congested anymore. It would waste the resource of the network.

6. Conclusion

Based on the Lagrangian optimization of the congestion control goal, we proposed a delay based iterative congestion control algorithm. A specific implementation of this algorithm is also discussed. After implementing proposed algorithm in MATLAB and comparing with other methods like MPTCP, this Weighted Vegas algorithm obviously ensures the fairness and efficiency requirement and could also better balance a dynamic traffic. The limitation on the input packets rate and two possible modifications to improve the performance are also discussed.

References

[1] H. Y. Hsieh and R. Sivakumar, "A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts", Proceedings of ACM MobiCom '02, New York, NY, USA, **(2002)**, pp. 83-94.
[2] J. R. Iyengar, P. D. Amer and R. Stewart, "Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths", IEEE/ACM Trans. Netw., vol. 14, no. 5, **(2006)**, pp. 951-964.
[3] K. Rojviboonchai and H. Aida, "An evaluation of multi-path transmission control protocol (M/TCP) with robust acknowledgement schemes", IEICE Trans. Communications, **(2004)**.
[4] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson and R. Wang, "A transport layer approach for improving end-to-end performance and robustness using redundant paths", Proceedings of USENIX '04, **(2004)**.
[5] M. Honda, Y. Nishida, L. Eggert, P. Sarolahti and H. Tokuda, "Multipath Congestion Control for Shared Bottleneck", Proc. PFLDNeT workshop, **(2009)** May.
[6] D. Wischik, C. Raiciu, A. Greenhalgh and M. Handley, "Implementation and Evaluation of Congestion Control for Multipath TCP", Proc. USENIX NSDI, **(2011)**.
[7] A. Ford, C. Raiciu and M. Handley, "TCP extensions for multipath operation with multiple addresses", Internet-draft, IETF, **(2011)**.
[8] S. Liu, T. Basar and R. Srikant, "TCP-Illinois: A Loss and Delay Based Congestion Control Algorithm for High-Speed Network", Performance Evaluation, vol. 65, **(2008)**, pp.417-440.
[9] Y. Cao, M. Xu and X. Fu, "Delay-based Congestion Control for Multipath TCP", ICNP, **(2012)**, pp. 1-10.
[10] R. S. Prasad, M. Jain and C. Dovrolis, "On the Effectiveness of Delay-based Congestion Avoidance", Proc. PLFDNet, **(2004)**, pp. 3-4.

