International Journal of Future Generation Communication and Networking
Vol. 9, No. 7 (2016), pp. 33-42
http://dx.doi.org/10.14257/ijfgcn.2016.9.7.04

A Drop Aware TCP Rate Control Algorithm for Heterogenic Internet Access

Bin Zeng, Lu Yao

Center of Information Management, Naval University of Engineering, Wuhan, China
zbtrueice@163.com

Abstract

In this paper, an end-to-end TCP throughput control mechanism called DARCA has been presented. DARCA controls TCP throughput through introducing extra packet drop rate or round trip time. The extra packet drop rate and round trip time are calculated from Throughput Control Mode. It provides single point control of end-to-end TCP throughput even if some of the bottlenecks are remote. DARCA identifies remotely bottlenecked TCP connections and provide polices of excess bandwidth redistribution. Further, per class control mechanism is used to reduce the overhead of rate enforcement. Simulation results show that DARCA gives good policy-based throughput control performance for both homogeneous and heterogeneous TCP connections.

Keywords*: TCP Rate Control, Network bottleneck, TCP-Friendly, Quality of Service*

1. Introduction

TCP (Transmission Control Protocol) is by far the most widely used transport layer protocol. It provides a connection-oriented reliable service to many Internet applications. The widespread diffusion of TCP is a result of the capability of TCP to dynamically adapt to heterogeneous underlying networks with different capacities and propagation delays.

However, the TCP dynamic behavior results in some drawbacks such as unfairness between TCP connections themselves [1-2]. Thus, for performance and fairness goals, some throughput control mechanisms are needed.

A large number of solutions have been proposed to improve TCP performance or fairness. Some solutions suggest modification or replacement of transport layer protocol at end hosts, such as different TCP versions [3-4] or the design of "TCP-Friendly" protocols [5-6]. Another part of solutions suggest enhancing the functionality of network elements, for example, the implementation of different scheduling mechanisms such as round-robin scheduling, priority queuing or active queue management such as RED [7]. At last, some solutions control TCP's transmission rate through modification of fields in TCP header and transmission rate of ack packets [8-9].

There are three distinct motives for this work. One motivation of this paper is that the throughput of a TCP connection is resulted from dynamic conditions along the end-to-end connection path which most of current one-hop control mechanisms such as scheduling cannot precisely control.

The second motivation is that though most of TCP rate control is good enough, it's not easy to be implemented as well. First, to modify TCP header filed means we should extract and recognize TCP header at network level which suffers from packet fragmentation and recalculation of TCP header checksum and worst of all, the encryption of IP packets for security purpose such as IPSec in VPN. Moreover, the effect of ack rate control is restricted by "ack compression" [10].

The third motivation is that some fundamental aspects of the behavior of TCP still need investigation. The bandwidth of the leased line is limited. However, the properties of TCP connections sharing this leased line are quite different. For example, on-line trading connections are much more important than web browsing, but they compete for the limited bandwidth of leased line. On the other hand, two TCP connections from different sources may have equal importance. Thus, a rate control mechanism is needed to manage the bandwidth allocation to assure the corporate operation.

2. Algorithm Overview

For controlling outgoing data traffic, previous works show that through scheduling, one can more easily control the bandwidth used by individual TCP connection on outgoing link. However, for controlling incoming data traffic, the fact is that Internet is administrated by different parties. Therefore, traffic aggregation happens at different places. When traffic aggregates in ISP's network, it's impossible for an enterprise to execute its traffic management policy in ISP's network. The management of incoming data traffic faces the following problems.

Generally speaking, a TCP connection's end-to-end connection path contains more than one links with different available bandwidth. The link with least available bandwidth is the bottleneck link [11, 12]. If the bottleneck link is not the link we want to control, it is called remotely bottlenecked. The maximum possible transmission rate of the TCP connection is bounded by the remote bottleneck. As shown in Fig. 1, the end-to-end connections path from TCP sender1 to the receiver contains three links: $L_{1,}$ $L_{3,}$ and L_4; the path from TCP sender2 to the receiver contains $L_{2,}$ $L_{3,}$ and L_4. Assume all links have the same capacity *C* and down end point of $L_{4,}$ *z* is the point which we can apply the control.

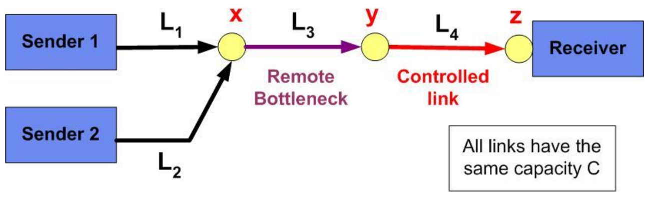

Figure 1. Remotely Bottlenecked TCP Connections

Because traffics from sender1 and sender2 already congest at L_3, scheduling mechanisms at ***z*** can not control the transmission rate effectively. That is, the downstream network element can not control the behavior of upstream network elements. In other words, implementation of scheduling mechanisms at downstream is not sufficient when the congestion occurs somewhere upstream. In Fig. 1, the enterprise can not control the speed it would receive the data form Content server A, B, and C according to its policy.

Consider *N* TCP connections, denoted as TCP_i flowing through the control point *P*. Each TCP connection has its bottleneck link which may or may not be the controlled link of interest in this paper. A flag $IsRB_i$ (i.e. Is Remotely Bottlenecked) is used to indicate whether the bottleneck link of TCP_i is the controlled link or not. Let the bottleneck bandwidth of TCP_i denoted as BBW_i. BBW_i is also the arrival rate of TCP_i at control point *P*. In this network model, we assume that if TCP_i is remotely bottlenecked, its BBW_i is bounded by some upstream network elements.

In the model, each TCP connection has a target bandwidth allocation that we would like to enforce, denoted as $TBW_{i,}$ and let the throughput RBW_i denotes the realized throughput resulted from the execution of the TCP throughput control policy. Notice that the target bandwidth is an administrative decision and the realized bandwidth is the result of the interaction between TCP and the TCP rate control mechanism. Let $p_{orig,i}$ be the original packet drop rate and $RTT_{orig,i}$ be the original round trip time (i.e. before the execution of the TCP throughput policy). The proposed TCP rate control mechanism would introduce RTT_{con} to the end-to-end

Copyright ⓒ 2016 SERSC

round trip time and reduce the TCP throughput. we propose a mechanism to control TCP throughput by introduce extra packet drop rate p_{con} (stands for controlled packet drop rate) because TCP's transmission rate is inverse proportional to square root of end-to-end packet drop rate.

We proposed a Drop-Based TCP Rate Control Scheme (DARCA) for TCP throughput management. DARCA is an end-to-end scheme which controls the average throughput of TCP connections. DARCA doesn't require the modification or enhancement of routers and end systems along the connection path. DARCA bases on the idea of using packet drops to regulate the behavior of TCP connections. To reduce the rate of a TCP connection, DARCA introduces extra packet drop. If the TCP sender detects a packet drop, it will decrease its sending rate accordingly. Through carefully setting of extra packet drop rate, the TCP throughput can be controlled with fine granularity.

As in Fig. 2, DARCA monitors bidirectional traffic (DATA and ACK) for each connection. A meter is used to measure the RTT and packet drop of a connection. A Flow Record is used to record *TBW*, *RBW* for each TCP connection. *Throughput* Control Model is used to calculate the control parameters p_{con} and RTT_{con}.

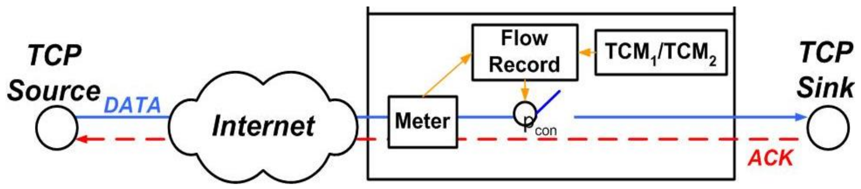

Figure 2. Abstract DARCA Architecture

In Fig. 3, the state transition of the DARCA is shown. DARCA operates in one of the three states: *Initial*, *Learning*, and *Stable*. A complete *Control Cycle* is composed of three phases, *Initial Phase*, *Learning Phase*, and *Stable Phase*. Except the *Initial Phase,* there may have one or more rounds in the *Learning Phase* and *Stable Phase*.

There are two reasons for DARCA to operate round by round instead of executing rate enforcement policy continuously. First, round-based execution of rate enforcement reduces the overhead of continuous network monitoring. Second, TCP can not increase or decrease its transmission rate to the target rate immediately at the execution of Rate Enforcement [13]. The duration between the execution of Rate Enforcement and the time that TCP's throughput has been raised or reduced to its target rate is the *Convergence period*. Moreover, for monitoring the TCP throughput, there's also a lower bound of the *Measurement Period,* which represents the frequency of throughput measurement. Therefore, there is also a lower bound of the interval of executing rate enforcement and the *Round* is adopted to fit this lower bound.

Convergence Period and Measurement Period are estimated as following:

$$\textit{Convergence Peirod} = \max(\sqrt{3/2p_{tot,i}}RTT_{tot,i}),\ \mathrm{i}=1\sim \mathrm{N} \tag{1}$$

$$\textit{Measurement Period} = \max(\sqrt{2/3p_{tot,i}}RTT_{tot,i}),\ \mathrm{i}=1\sim \mathrm{N} \tag{2}$$

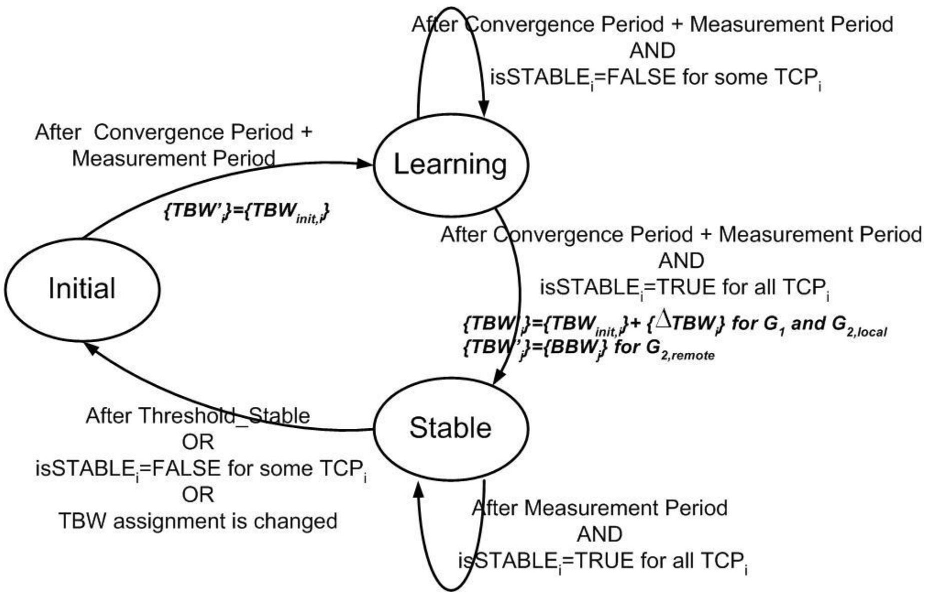
After Convergence Period + Measurement Period
AND
isSTABLE_i=FALSE for some TCP_i
After Convergence Period + Measurement Period
Learning
{TBW'_i}={TBW_init,i}
After Convergence Period + Measurement Period
AND
isSTABLE_i=TRUE for all TCP_i
Initial
{TBW'_i}={TBW_init,i}+ {ΔTBW_i} for G_1 and G_2,local
{TBW'_i}={BBW_i} for G_2,remote
After Threshold_Stable
OR
isSTABLE_i=FALSE for some TCP_i
OR
TBW assignment is changed
Stable
After Measurement Period
AND
isSTABLE_i=TRUE for all TCP_i

Figure 3. State Transition of DARCA

An example of state transition of DARCA is shown in Fig. 4.

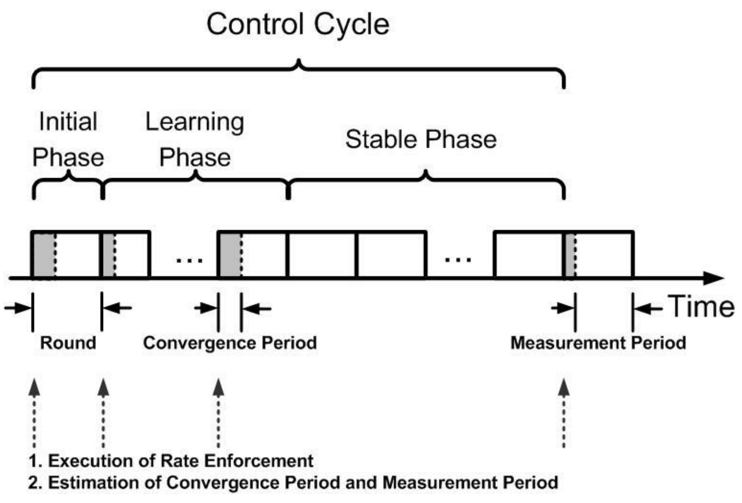

Figure 4. An Example of State Transition

In the *Initial Phase*, the system specifies bandwidth based on administrative policy. Corresponding control parameters are calculated and the rate enforcement is executed. Here, "administrative policy" means that the target bandwidths are specified without advance understanding of the underlying TCP connections' capabilities. It takes sometime for all TCP connections affected to converge. Here each TCP's *TBW* is set to $TBW_{,i}$, and the flag $isSTABLE_i$ is initialized to FALSE. *isSTABLE* is used to record whether the realized throughput of a TCP connection close to its target rate, *TBW*. The measurement of RTT_{orig} and p_{orig} and the calculation of p_{con} are executed. DARCA also schedules a *RBW* measurement task after the estimated *Convergence Perio*. After the measurement of realized bandwidth, DARCA checks each flow's RBW and update the *isSTABLE* flag. Then, DARCA moves to the *Learning State*.

In the *Learning Phase*, a TCP connection's target bandwidth is set to the bottleneck bandwidth for those connections that are remotely bottlenecked. The excess bandwidth is redistributed among connections that have target bandwidth less than the bottleneck bandwidth. At the beginning of a round in *Learning Phase*, the rate enforcement is also executed and the realized throughputs are measured after the system converged. When the realized throughputs of TCP connections meet stable criteria, DARCA moves to the *Stable Phase*. Note that, the system may go through one or more rounds to become stable. Similarly, the system may stay in the *Stable Phase* for more than one round.

For TCP connections whose *TBW* is less than its *BBW* (G_1) or whose *TBW* is greater than its *BBW* while it is local bottlenecked ($G_{2,local}$), their *RBW* should be eventually reduce or raise up to its T*BW*.

However, for TCP connections whose TBW is greater than its BBW and is also remote bottlenecked ($G_{2,remote}$), its transmission rate would never greater than its bottlenecked bandwidth *BBW*. The difference between its *TBW* and *BBW* is the excess bandwidth allocation. The major tasks of DARCA in *Learning State* are to learn the remotely bottlenecked TCP connections and revise the target bandwidth allocations.

At the beginning of each *Learning Round*, for each TCP_i which is inferred to be remotely bottlenecked, DARCA updates its TBW_i to TBW_i*0.8. Here, 0.8 is a bandwidth updating factor which is used to smoothly stepwise reduce the target bandwidth of TCP connections. After reducing target bandwidth for remotely bottlenecked TCP connections, DARCA redistributes the excess bandwidth to other TCP connections..

Rate Enforcement would be executed for TCP_i whose target bandwidth has been revised. DARCA schedules a *RBW* measurement task after estimated *Convergence Period*. After *RBWs* are measured, DARCA checks if the measured *RBWs* meet the stable criteria which decides whether DARCA would move to *Stable State* or stay in *Learning State*. The stable criteria here is that “All *isSTABLE* flags have been set to TRUE”.

In the *Stable Phase*, DARCA continues to measure the realized throughputs. When the realized throughputs violate the stable criteria, DARCA moves to *Initial State* and restart a *Control Cycle*.

3. Class Control

In this section, rate control algorithm on per class control base is described. The objective of per class control is to reduce he complexity and overhead of DARCA while maintains a stable aggregate throughput of a TCP class when TCP connections join or leave. Here, we define a TCP class to be a set of TCP connections with similar properties.

We know the overhead of DARCA mainly results from three tasks:

- Measurement of network parameters.
- Calculation of $p_{con.}$
- Arithmetic operations of drop counters upon packet arrivals.

Intuitively, in order to reduce the overhead, we should reduce the number of times DARCA performs these three tasks. The idea is to group TCP connections according to their properties because TCP connections with similar network properties would also behave similarly under the same condition. Thus, the network measurement of one of the TCP connections can be shared by other TCP connections of the same TCP class. Moreover, TCP connections of the same TCP class also share the control parameters. In our work, we use RTT as the grouping criterion.

To support per class control, DARCA should track the number of connections within each TCP class. If the target bandwidth of one TCP class is *TBW* and there are *fn* connections in this class, then the calculation of p_{con} should base on the target bandwidth *TBW / fn*. Hence, each TCP connection’s transmission rate would be controlled at *TBW / fn* and the aggregate throughput of the TCP class would be *TBW*.

Here, we assume RTTs of TCP connections in a TCP Class are identical. However, if we want to group TCP connections with a range of RTT, we have to calculate a representative RTT for each TCP Class. There are two options to reach this RTT:

- Simply choose the mean of the RTT range as the representative RTT.The representative RTT is the middle of the RTT range. This method is easy and DARCA does not measure the RTT of each TCP connection.
- Calculate the representative RTT from individual per flow RTT.

Assume there are *fn* connections in a TCP class. Each connection has it own RTT, R_i and hence the throughput of each connection is proportional to $1/R_i$. Let the representative RTT denoted as R_{class}, The target throughput of each TCP connection is proportional to $1/R_{class}$. To minimize the difference between the aggregate target bandwidth and the realized bandwidth, we have the following equation:

$$\sum_{i=1}^{fn}\left(\frac{1}{R_{class}}-\frac{1}{R_i}\right)=0 \tag{3}$$

which yields

$$\frac{fn}{R_{class}}-\sum_{i=1}^{fn}\frac{1}{R_i}=0 \tag{4}$$

This gives

$$\frac{fn}{R_{class}}=\sum_{i=1}^{fn}\frac{1}{R_i}$$

$$R_{class}=\frac{fn}{\sum_{i=1}^{fn}\frac{1}{R_i}} \tag{5}$$

We can then calculate the representative RTT by Eq. (5). This method would result in better throughput control performance in the cost of per flow RTT measurement.

4. Performance Evaluation

In the experiments, a controlled link *L* with capacity *C*=10Mbps shared by *n* traffic sources is simulated. The TCP rate control point is located at the downstream router denoted as *ER*. The upstream router with respect to the control point is denoted as *AR*. Traffic source $source_i$ connects to *AR* by a link with capacity BBW_i. The round trip time between $source_i$ and *ER* is denoted as $RTT_{orig,i}$. The queue management of the controlled link is drop-tail [14, 15]. *TCP Reno* is used. Here, TCP connections are assumed to be greedy FTP connections. They always have data to send. The maximum transmission unit (*MTU)* of TCP is 512 or 1460 bytes; the receiver advertised window is 65536 bytes which is the maximum TCP receiver's buffer size; and the slow start threshold is initialized to 65536 bytes.

4.1. Remotely Bottlenecked Connection

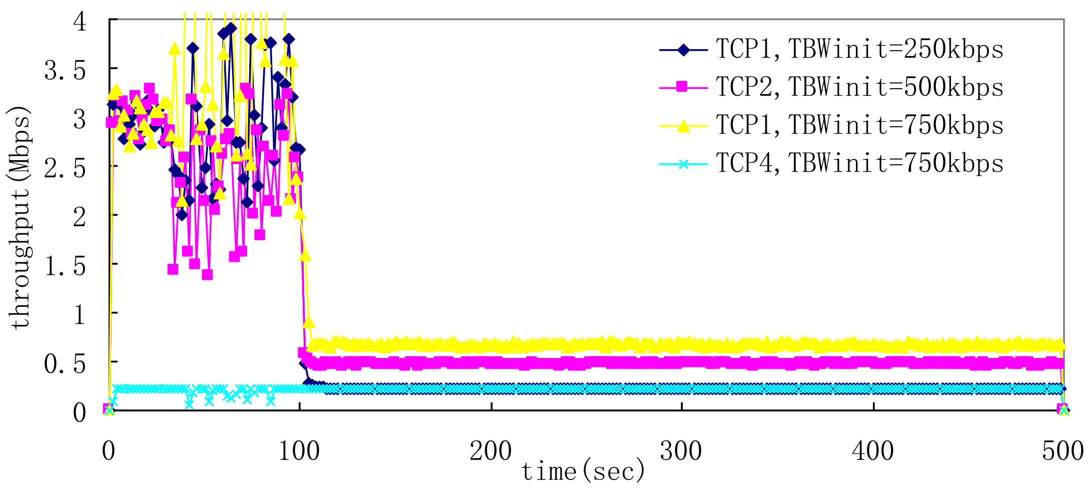

Figure 5. Excess Bandwidth Redistribution over the Time

In this experiment, we study TCP connections that are remotely bottlenecked. For these connections, DARCA updates their target bandwidths to the bottleneck bandwidth and redistributes excess bandwidth to other connections needed. There are four TCP connections. Among them, TCP_4 is remotely bottlenecked with the bottleneck bandwidth 250kbps; the other three connections are locally bottlenecked. The target bandwidths of the four connections are 250kbps, 500kbps, 750 kbps and 750kbps, respectively.

Fig.5 shows throughput dynamics of the four connections before and after applying target rate control. DARCA is activated at 100 sec. It moves to the *Learning State* at 104 sec. During the *Learning Phase* (105 sec ~ 112 sec) TBW_4 is revised and reduced to 257kbps. At 112 sec, DARCA enters the *Stable State*.

4.2. Events Handle

Consider two TCP connections. TCP_1 is locally bottlenecked and TCP_2 is remotely bottlenecked with bottleneck bandwidth 250kbps. *TBWs* are 250kbps and 500kbps, respectively. First, DARCA is activated at 1 sec. In Fig. 6, we can see that TCP connections operate at the target bandwidths under the proportional policy. (i.e the excess bandwidth 250kbps from TCP_2 goes to TCP_1). Then a new TCP connection TCP_3 with *TBW* 250kbps joins at 100 sec. The excess bandwidth is now redistributed to TCP_1 and TCP_3 fairly because TBW_1 equals to TBW_3. At 200 sec, TCP_1 leaves. The excess bandwidth originally received by TCP_1 is now redistributed to TCP_3.

Consider two TCP connections. TCP_1 is locally bottlenecked and TCP_2 is remotely bottlenecked with bottleneck bandwidth 250kbps. *TBWs* are 250kbps and 500kbps, respectively. DARCA is activated at 1 sec. The TCP connections operate at the target bandwidths under the proportional policy.

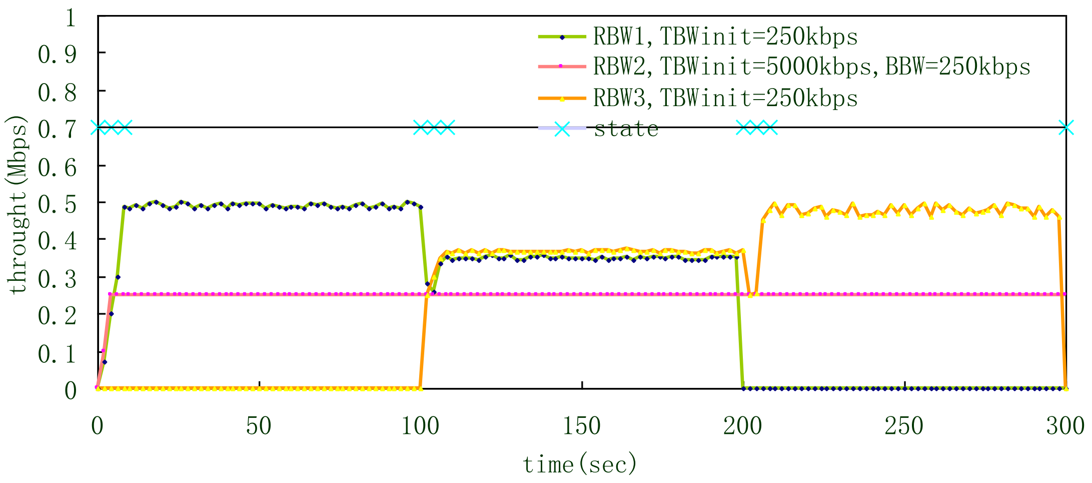

Figure 6. TCP$_3$ joins at 100 sec and TCP$_1$ leaves at 200 sec

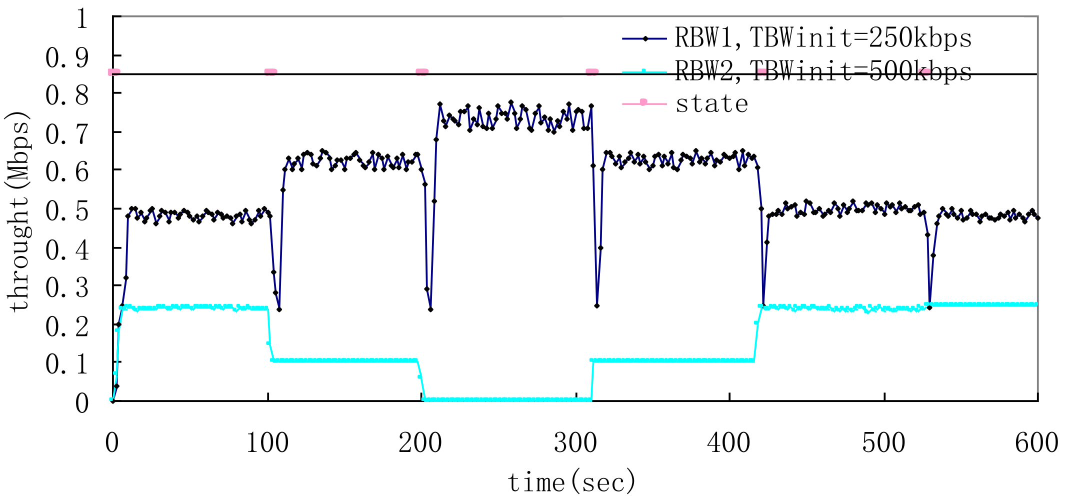

Figure 7. DARCA Adapts the Network State

There are four events in Fig.7:

- The bottleneck bandwidth of TCP$_2$ decreased to 100kbps at 100 sec
- TCP2 stopped sending data at 200 sec
- TCP2 started sending data at 300 sec
- The bottleneck bandwidth of TCP2 increased to 250kbps at 400 sec

At 100 sec, event (a) occurs and the additional excess bandwidth 150kbps (250-100) is redistributed to *TCP$_1$*, and the updated *TBW*s are 650kbps and 100kbps. We see that the realized bandwidth of *TCP$_1$* is now close to 650kbps. At 200 sec, event (b) occurs, and the target bandwidth of stopped *TCP$_2$* is reduced to zero. Again, the additional excess bandwidth 100kbps is redistributed to *TCP$_1$*. The updated TBWs are 750kbps and 0kbps. *TCP$_1$*'s realized bandwidth further increases to 750kbps.

At 300 sec, event (c) occurs. However, this event cannot be detected by DARCA in *Stable Phase*, and the throughput of *TCP$_2$* is still restricted to zero. In this experiment, the length of *Stable Phase* is restrict to near 100 sec. Therefore, DARCA restarts a *Control Cycle* at 310 sec and *TCP$_2$* can now transmit data at bottleneck bandwidth 100kbps. At 400 sec, event (d) occurs. This event cannot be detected, either. While DARCA restarts a *Control Cycle* at 415 sec, the increased bottleneck bandwidth can be captured and TCP2's transmission rate could increase to 250kbps.

Note that at 100 sec, 200 sec, 310 sec, and 415 sec, the throughput of *TCP*$_1$ drops to 250kbps drastically. This is because every time DARCA restarts a *Control Cycle*, the target bandwidth of *TCP*$_1$ would be reset to 250kbps.

4.3. Per Class Control

Consider there are three TCP classes in the system. There are three TCP connections in each class and *TCP Class*$_1$ is remotely bottlenecked at 1.5Mbps. *TBW*$_1$, *TBW*$_2$, and *TBW*$_3$ are 3Mbps, 2Mbps and 1Mbps.

Fig.8 shows the dynamics of aggregate throughputs of TCP classes are well controlled at target bandwidth and the excess bandwidth is proportionally distributed.

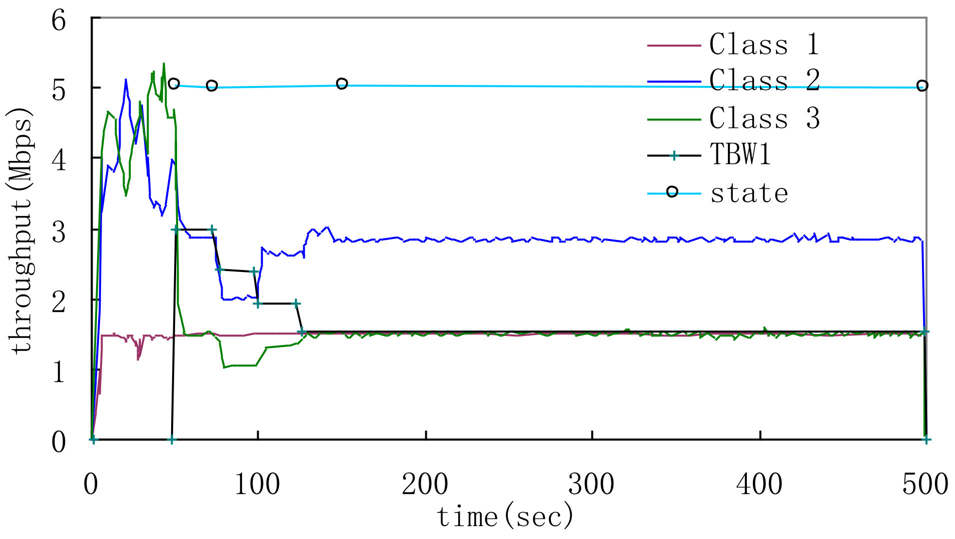

Figure 8. Excess Bandwidth Redistribution of per Class Control over the Time (Proportional)

In this section, we have shown that DARCA with per class control still performs well, including the efficiency of rate control and the distribution of excess bandwidth. However, per class control also increases the oscillation of per flow throughput. There is a tradeoff between the overhead reduced by per class control and the increased oscillation of throughput.

5. Conclusion

Compared with TCP Rate Control, DARCA doesn't manipulate the content (TCP header) of the packets, thus it is applicable even if the packets are authenticated or encrypted in the emergent network service, VPN. Moreover, DARCA could control any other "TCP-friendly" transport layer protocols which uses packet drop rate to estimate the available bandwidth of the network.

In our work, we focus on the scenario that the control point is the input port of the edge router which is at the downstream end of the connections path. To place the control point at any position along the connection path, there are still two future works. First, the measurement of end-to-end network parameters should include the measurement from the (source-to-control point) part and (control point-to-sink) part of the connection. Second, Eq. (2) which formulates the end-to-end packet drop rate,

p_{tot} by p_{con} and p_{orig} should also be re-conducted which would changes how we reach the control parameters, p_{con}.

References

[1] M. Mizutani, Y. Miyoshi, K. Tsukamoto, M. Tsuru and Y. Oie, "Network-supported TCP rate control for high-speed power line communications environments", Simulation Modelling Practice and Theory, vol. 19, no. 1, **(2011)**, pp. 69–83.

[2] D. Wischik, C. Raiciu, A. Greenhalgh, "Design, Implementation and Evaluation of Congestion Control for Multipath TCP", 8th USENIX Symposium on Networked Systems Design and Implementation, Boston, USA, **(2011)**.

[3] Y. N. Xie, G. L. Sun, J. Su, "Improvement and Research on Congestion Control Algorithm of TCP Vegas", Journal of Harbin University of Science and Technology, vol. 16, no. 3, **(2011)**, pp. 26-30.

[4] G. Zodi, A. Lusilao and J. N. Kakandel, "Performance Evaluation of TCP-Friendly Rate Control Enhanced with Adaptive Filters", International Journal of Advancements in Computing Technology, vol. 3, no. 1, **(2011)**, pp. 132-145.

[5] S. Jung, J. Lee and G. Lee, "Novel Fastest Retransmission and Rate Control Schemes for Improving TCP Performance in Wireless Ad Hoc Networks", Wireless personal communications, vol. 75, no. 1, **(2014)**, pp. 557-567

[6] K. R. Chowdhury, M. Di Felice and I. F. Akyildiz, "TCP CRAHN: A transport control protocol for cognitive radio ad hoc networks", IEEE Transactions on Mobile Computing, vol. 12, no. 4, **(2013)**, pp. 790-803

[7] K. Winstein and H. Balakrishnan, "Tcp ex machina: Computer-generated congestion control", ACM SIGCOMM Computer Communication Review, vol. 43, no. 4, **(2013)**, pp. 123-134

[8] C. Liu, I. Bouazizi and M. Gabbouj, "Rate adaptation for adaptive HTTP streaming", Proceedings of the second annual ACM conference on Multimedia systems, New York, NY, USA, **(2011)**, pp. 169-174.

[9] S. Soundararajan and R. S. Bhuvaneswaran, "Multipath load balancing & rate based congestion control for mobile ad hoc networks (MANET)", 2012 Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP), Bangkok, **(2012)**, pp. 30-35.

[10] H. Yuan, M. K. Du and C. B. Li, "Improve TFRC Mechanism for Streaming Media in Wireless Network", Journal of Harbin University of Science and Technology, vol. 16, no. 2, **(2011)**, pp. 40-43.

[11] V. Singh, J. Ott and I. D. Curcio, "Rate-control for conversational video communication in heterogeneous networks", 2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), San Francisco, CA, USA, **(2012)**, pp. 1-7.

[12] A. Sathiaseelan and G. Fairhurst, "TCP-Friendly Rate Control (TFRC) for bursty media flows", Computer Communications, vol. 34, no. 15, **(2011)**, pp. 1836-1847

[13] S. Akhshabi, A. C. Begen and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP", Proceedings of the second annual ACM conference on Multimedia systems, New York, USA, **(2011)**, pp. 157-168.

[14] K. D. Huang, K. R. Duffy and D. Malone, "H-RCA: 802.11 collision-aware rate control", IEEE Transactions on Networking, vol. 21, no. 4, **(2013)**, pp. 1021-1034.

[15] J. Lee, H. W. Lee and Y. Yi, "Improving TCP performance over optimal CSMA in wireless multi-hop networks", IEEE Communications Letters, vol. 16, no. 9, **(2012)**, pp. 1388-1391.

Copyright ⓒ 2016 SERSC