Mitigating Bufferbloat with Receiver-based TCP Flow Control Mechanism in Cellular Networks

Xiaolan Liu, Fengyuan Ren, Ran Shu, Tong Zhang, Tao Dai
Tsinghua National Laboratory for Information Science and Technology, Beijing, China
Department of Computer Science and Technology, Tsinghua University, Beijing, China
{xl-liu12, shur11, zhang-t14, dait14}@mails.tsinghua.edu.cn, renfy@tsinghua.edu.cn

***Abstract*—Bufferbloat is an intractable phenomenon in both the Internet and cellular networks, which may cause excessively long delays without contribution to the improvement of throughput at the same time. The problem matters more for the latter, since large buffer is inevitable due to the demand of link layer retransmission mechanism as well as mitigating traffic burst. We unveil the root cause of bufferbloat to be the mismatching between the adjustment of sending windows and the dynamic variation of available bandwidth at the existence of large buffer. Being the bottleneck link and last hop of a connection in cellular networks, wireless link connects the mobile terminal directly. Besides, the mobile terminal can get all of the information of channel such as channel states and signal strength. So it is more feasible to resolve the bufferbloat at receiver side with the help of channel information, nevertheless, which is ignored in previous studies. So in this article we propose a receiver-based flow control strategy named as ABRWDA that retrieves available bandwidth at receiver side directly and use it to dynamically calculate the receiver window (*rwnd*) to mitigate the bufferbloat in cellular networks. We test our approach with the NS-2 simulation, and results indicate that ABRWDA achieves 0.1× and 0.4× shorter queues, 0.5× and 0.8× lower latency, while still maintaining the same high throughput as that of Newreno and a previous solution DRWA, respectively.**

***Index Terms*—Bufferbloat; TCP; Cellular networks.**

I. Introduction

The use of cellular networks is increasingly popular due to the ubiquitous deployment of mobile communication infrastructure, the quick updating of mobile terminals and mobile applications, as well as the development of mobile communication technologies. In Sandvine globe Internet phenomena report, the real-time entertainment, web browsing, and social networking make a 71.55% composition of average peak period traffic of the global mobile access [1], and the mean monthly usage of mobile access increases 16.6% in one year, while that of fixed access is only 13.4% [1], [2]. The explosive growth of mobile data traffic poses severe pressure on cellular network providers to provide better services, in which there still exists many performance problems. Bufferbloat [3] is a most typical one.

Bufferbloat is the phenomenon that packets are blocked in buffers when the sending window increases persistently at the existence of large buffers, which causes large delays and little improvement in throughput. The large buffer in cellular networks is the engineering choice for smoothing traffic burstiness and adapting to channel variability. Because the packet dropping is totally concealed by link layer retransmission mechanisms, TCP tends to fill up the buffer at congested links no matter how much the buffer size is. The heavily packed buffer contributes to excessive traffic delays and thus lose the ability to perform their intended function of absorbing traffic burst [4]. In cellular networks, the wireless links usually lie between base stations and mobile terminals. Because of the lower bandwidth of wireless links than that of wired ones, packets will accumulate in the buffer of base stations on the last hop when they are transferred from server to terminal. So the bufferbloat problem in cellular networks occurs in the downlink. In a prior study [5], up to several seconds of round trip delays caused by large buffers can be observed.

There are several proposals of mitigating bufferbloat in the Internet [4], [6]. Nonetheless, these schemes either need modifications to the base stations or require the sender-side to distinguish mobile access from fixed access for enhancing TCP, either of which can cause tremendous deployment costs. By contrast, a receiver-based solution is a more light-weight choice for the cellular case, since the mobile terminals, as receiver-side, are really alterable to users. The previous solution DRWA [5] introduces a delay-based window adjustment mechanism upon receiver-side. However, interfered by the receiver-side estimation error of RTTs and congestion windows, DRWA performs even worse than delay-based TCP versions like Vegas.

In this paper, we present an approach named as ABRWDA (Available Bandwidth based Receiver Window Dynamic Adjustment) to dynamically mitigate bufferbloat in cellular networks. Cellular networks have a special characteristic that the bottleneck link is always the last hop for downlink. On the other hand, the bandwidth of wireless channel has a certain relationship with the Signal to Interference plus Noise Ratio (SINR) and channel states. And UE can get all these information because of the direct connection with it. The key innovation of ABRWDA is to adjust sending window dynamically to adapt to the time-varying bottleneck link capacity in cellular networks. ABRWDA retrieves the bandwidth at receiver side directly by tracking the SINR. Meanwhile, RTT is also estimated at the receiver-side. By multiply bandwidth and RTT, the appropriate receiving window is obtained. To control sending window at receiver-side, ABRWDA modifies the TCP flow control mechanism with dynamically adjusted *rwnd* calculated by previous method. We insist that our solution is

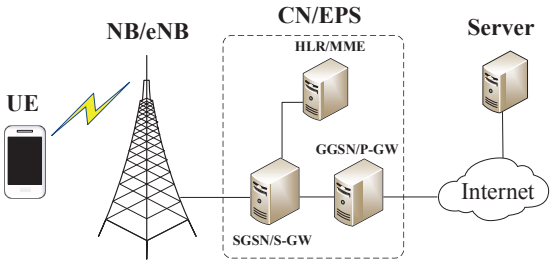


Fig. 1. The architecture of cellular networks.

easy to deploy because it can be functioned as either a kernel module or a netfilter, while requires no modifications to the network or application software and no support from ISPs.

To evaluating the performances of ABRWDA, several simulations on NS2 platform is conducted. The experiment results prove that ABRWDA can reduce delay while maintain the appropriate throughput of TCP. Further experiment results show that the Average Flow Completion Time (AFCT) is reduced by 25% to 78% compared to TCP NewReno and at most 50% improvements compared to previous solutions. Results also reveal better network environment adaptability of ABRWDA.

The rest of the paper is organized as follows. We presents the background, related work and motivation in section II. Detailed algorithm design and analysis of ABRWDA are presented in section III. Section IV outlines the experimental configurations thoroughly, followed by the experimental results and performance comparison with that of other approaches in section V. We conclude our paper and discuss our future work in section VI.

II. Background, Related Work and Motivation

In this section, we describe the background, related work and motivation of this paper.

A. Background

1) Cellular Networks: The cellular network is a heterogeneous network that consists of wired and wireless parts, as illustrated in figure 1. The wired part has the characteristics such as traffic burst. The wireless part has several compositions such as node of base-station (NB) and user equipment (UE), and UE is the downlink terminal of a connection.

The link bandwidth of wireless part is variable due to the signal variation of wireless channels. The link between base-station (BS) and UE is always the last-hop of a downlink wireless link, and it also is the bottleneck of the connection [5]. Cellular networks also use TCP as it's transport layer protocol. However, due to the time-varying nature of the wireless channels, TCP does not work well enough in wireless networks. So in many cellular networks, a large buffer is deployed to absorb the burst traffic and achieve the retransmission mechanism of the link layer [5], [7].

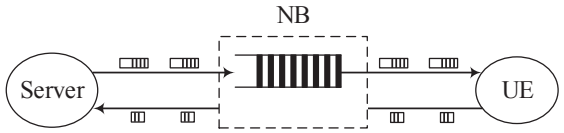


Fig. 2. Illustration of bufferbloat problem in cellular networks.

2) Bufferbloat: Bufferbloat [3]–[5], [8] is an intractable phenomenon in both the Internet and cellular networks, which may cause excessively long delays without contribution to the improvement of throughput at the same time. The problem matters more for the latter, since large buffer is the inevitable choice due to the demand of link layer retransmission mechanism as well as the need of mitigating traffic burst.

In a bufferbloated circumstance we consider in this paper in figure 2, the source will increase it's sending window persistently because the packets dropping is concealed by the large buffer. So the RTT latency increases sharply when packets are blocked in large buffer.

3) TCP Flow Control Mechanism: The TCP receive window is originally designed to prevent a fast sender from overwhelming a slow receiver with limited buffer space, used in TCP flow control mechanism [9]. This mechanism governs the size of a sending window together with the congestion control mechanism. It reflects the available buffer size on the receiver side so that the sender will not send more packets than the receiver can accommodate.

4) Available Bandwidth: The available bandwidth of a connection is the difference of the link capacity and the data rate of cross traffic. End-to-end available bandwidth estimation is important for a range of applications such as network management, flow control and rate adaptation in real-time multi-media streaming [10]. It depends on that of the bottleneck link. In a practical cellular network communication, the wireless link forms the last hop from the base station to the mobile clients, which is usually the bottleneck of the routes in terms of congestion [5]. Therefore, the available bandwidth of the wireless link is usually that of the whole connection.

B. Related Works

1) Bufferbloat: The issue about bufferbloat was first exposed by Dave P Reed in [11]. He found large RTTs along the routes but without packet loss. Kathleen Nichols and Van Jacobson proposed a modern AQM, CoDel [4], which aims to solve the bufferbloat by gauging the packet-sojourn time through the queue with the help of the timestamp. Rong Pan et al. issued a latency-based design for controlling bufferbloat in the Internet in [6], which could effectively cap the average queueing latency with a reference value.

All these works are about tackling bufferbloat in the Internet, and bufferbloat also exists in cellular networks. DRWA, a dynamic receive window adjustment approach, was proposed in [5] to tackle the bufferbloat by estimating the *cwnd* with the received data at receiver side. In order to mitigate the ACK

delays by eliminating TCP ACK clocking, a new TCP variant TCP-RRE was presented in [12]. It used TCP timestamp to estimate the receive rate at receiver side and determines the sending rate accordingly, which can keep the occupancy of the downlink buffer low. In [13] Yung-Chih Chen et al. researched the bufferbloat's influence of on multi-path TCP (MPTCP) with WiFi and cellular networks. They show that MPTCP might suffer from bufferbloat when there is another long-lived WiFi flow and the severer the bufferbloat is, the more harm MPTCP's performance will sustain.

2) Receiver-side Flow Control in Cellular Networks: TCP uses the effective sliding window mechanism to adjust the sending rate at the sender side. The sending window size is the minimum of the congestion window *cwnd* and the advertised receiver window *rwnd*. But TCP works poor because of the time-varying wireless channels in cellular networks. RCP [14] is a TCP clone that performs all important tasks including congestion control and reliability mechanism at receiver side. In [10] the TCP sending rate is varied adapting to the dynamic adjustment of advertised receive window. Yin Xu et al. use a receiver-side flow control to regulate the 3G/HSPA uplink buffer in [15].

C. Motivation

From the above subsection we know many works have been performed in cellular networks to tackle bufferbloat. Seminal as they are, all these approaches do not uncover the root cause of bufferbloat, which is the mismatching between the adjustment of sending windows and the dynamic variation of available bandwidth at the existence of large buffer. In this case, the key to solving bufferbloat is to make the sending rate adapted to the time-varying available bandwidth of the connection.

In cellular networks the wireless link is always the bottleneck link of the network, so it's available bandwidth is usually that of the connection. The UE acts as receiver for data sent from the server in the wired data network, which consist of the downlink of the connection. Being adjacent to the wireless link, which is the last-hop of the connection, UE obviously can obtain first-hand information of the wireless link [14], and it has all the information necessary to determine the rate at which packets should be sent by the server [16]. These information (such as MCS and SINR) has congruent relationship for certain network type, as stated in [17], [18]. Accordingly, UE can get the variable bandwidth value with the variation of wireless channel states.

In a word, we can retrieve the wireless link bandwidth at UE directly with the help of the channel information, estimate RTT with the same method in [5], calculate *rwnd* with the former two and adjust the sending rate with calculated *rwnd* finally under TCP flow control mechanism.

III. Algorithm Design and Analysis

In this section, we present algorithm design and analysis of ABRWDA.

Algorithm 1 ABRWDA

Initialization:

$wind\_ \Leftarrow CWND\_INIT$;
$last\_wind\_ \Leftarrow CWND\_INIT$;
$data\_cumd\_ \Leftarrow 0.0$;
$rtt\_min\_ \Leftarrow a\ large\ enough\ value$;

if $data\_cumd\_ == 0.0$ **then**
  $start\_time\_ \leftarrow now$;
end if
$data\_cumd\_ += PacketSize$;
if $data\_cumd\_ > last\_wind\_$ **then**
  $rtt\_est\_ \leftarrow now - start\_time\_$;
  if $rtt\_min\_ > rtt\_est\_$ **then**
    $rtt\_min\_ \leftarrow rtt\_est$;
  end if
  $last\_wind\_ \leftarrow wind\_$;
  $Dbw\_ \leftarrow (1-\alpha) * Dbw\_ + \alpha * bandwidth\_$;
  $wind\_ \leftarrow \lambda * Dbw\_ * rtt\_min\_$;
  $rwnd\_ \leftarrow wind\_ > last\_wind\_?wind\_ : last\_wind\_$;
  $data\_cumd\_ = 0.0$;
else
  $rwnd\_ \leftarrow wind\_$;
end if

A. Retrieving the Available Bandwidth

The UE can obtain the overall information about the physical channels, and the transport layer communication protocols will then get these information directly from UE.

In the Internet there may be several flows at the same time. So the link bandwidth is shared by these flows. But research result in [19] shows that in LTE there is only one TCP flow actively downloading data in 72.1% of the time, and this percentage might be even larger for smart phone users because a fraction of users that using LTE data cards on their laptops may be included in their data set, which may have high TCP flow concurrency. That is to say, in most cases there is only one TCP flow in a short time scale in cellular networks. So the available bandwidth of a connection is exactly that of the wireless link. Thus we think that the link bandwidth is used by one flow in cellular networks in one time. We get the link bandwidth as the method displayed in the above subsection. Due to its dynamic nature, we use a filter with a moving average coefficient α to smooth its fluctuation.

B. RTT Estimation

We get RTT estimation by averaging the RTT samples obtained from the timestamp within the last RTT when timestamp option is on. Otherwise, we estimate the RTT value with the time interval between a first acknowledged byte and a byte receipted whose sequence number is at least one window size forward, the same method in [20].

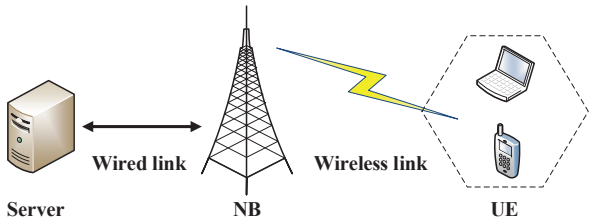


Fig. 3. Network topology in simulation.

TABLE I
SIMULATION PARAMETERS

Parameter	Value
Packet size	1500B
Queue management	Drop Tail
Buffer size in NB	500 packets
Bandwidth between sender and NB	1000 Mbps
Bandwidth between NB and UE: downlink	(3.1∼9.3)Mbps
Bandwidth between NB and UE: uplink	(1.8∼5.4)Mbps
Link delay between server and NB	50ms
Link delay between NB and UE	25ms
CWND_INIT	10 packets

C. Rwnd Calculation

We calculate *rwnd* with the retrieved available bandwidth and estimated RTT and adjust the sending window with TCP flow control mechanism. The Detailed realization of window acknowledgement mechanism needs a RTT period. The adjustment effect to sending window of *rwnd* will be seen at least after a RTT. But in a RTT, the bandwidth may be varied from a small value to a big one, which can causes the tendency to zero buffer and throughput decreasing. Thereafter, we set a scaling factor λ to handle this case.

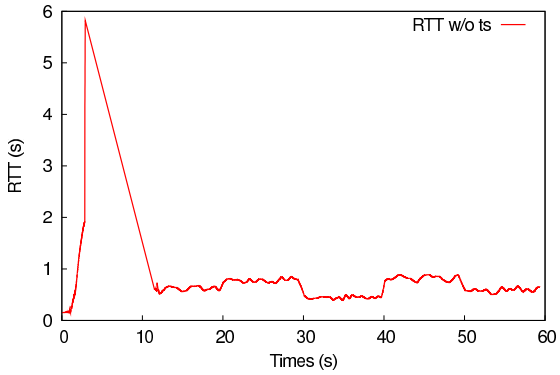
Because of the dynamic variation of wireless link bandwidth, the size of current *rwnd* window may be smaller than that of the previous one when there is a sudden increase in the wireless link bandwidth, which could cause the abnormal phenomenon that the estimated RTT value is smaller than the theoretical minimum (propagation delay only). So in our design, we select the bigger one between the current window and the previous one as new *rwnd*.

The pseudocode of ABRWDA is presented in algorithm 1.

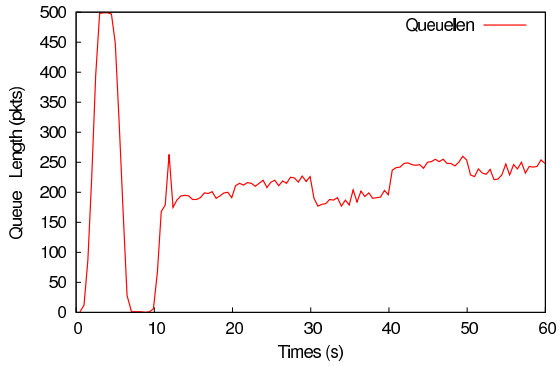
IV. Simulation Configurations

In light of the limitations stated in aforementioned literatures, we propose a method named ABRWDA to mitigate the bufferbloat in the cellular networks. We perform several simulation experiments to test the effectiveness of our approach. The simulations are performed using NS-2 version 2.30, which is running on the Ubuntu 10.04 LTS with 2.6.32-21-generic Linux kernel. And the simulation topology is shown in figure 3.

There are three kinds of nodes in the simulations, denoting the remote server, the base-station (NB) and the UE, respectively. The remote server lies in the wired network. The link bandwidth between the server and NB is 1Gbps, and



(a) RTT



(b) Queue Length

Fig. 4. The RTT and queue length of NewReno in a bufferbloated circumstance.

the propagation delay is 50ms. In a real experiment in [5], they observed more than 800KB (almost 560 packets with 1500B packet size) packets in flight for a certain carrier. The minimum BDP is 58KB for this network state. So in our simulation we set the buffer size of NB to be 500 packets, far greater than the minimum BDP. The queue management mechanism is drop-tail. The UE can be mobile phone or laptop.

In NS-2, any two nodes are connected by an *OTcl* object [21]. The link bandwidth and propagation delay are stored in two member variables of *OTcl* class *DelayLink* respectively. In our simulation, we obtain the available bandwidth in file *tcp_sink.cc* using a customized static member variable. It inherits the member function *bandwidth()* of its parent class, which returns the link bandwidth value bound with it given in *Tcl* file. We determine the configurations of the wireless link according to the UMTS specifications, and the model presented in [22] is also referred to. The bandwidth is set to be the theoretical value of the CDMA2000. All the parameters are listed in table I.

In real network circumstances, wireless physical link rate is dynamically variable due to the variation of signal strength and other radio environment [23]. In our simulation, the raw TCP version is NewReno. We simulate the dynamism of the wireless link bandwidth with random variables in *Tcl* file. We use two random variables to reappear the dynamical variation

of the wireless link bandwidth. The first random variable simulates different experiment circumstance with good or weak signal condition. And the second one simulates the dynamic variation of wireless link bandwidth. Our simulation results in figure 4 agree with that in [5], which are measured in real networks. So configurations in our simulation can depict the practical networks conditions perfectly.

V. Experimental Results and Performance Evaluation

In this section, we analyze the performance of ABRWDA and make a contrast with that of DRWA and some other TCP versions. There have been many works about TCP performance in wired networks, but what it is like in bufferbloated circumstance is still undiscovered. In this section we present the performance evaluation of ABRWDA, DRWA and Vegas in a bufferbloated cellular network.

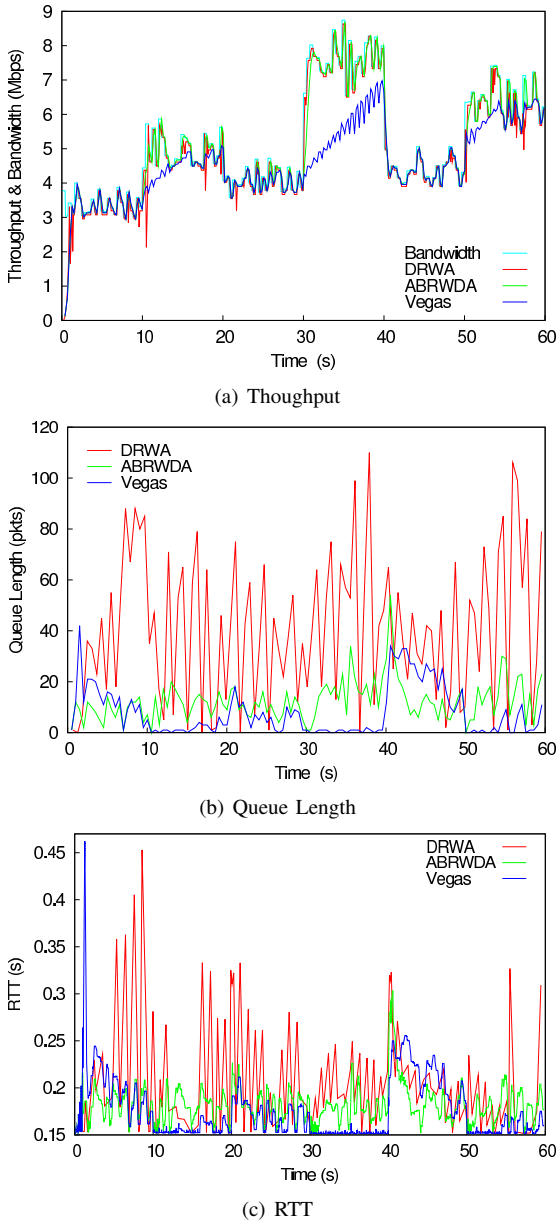
A. TCP Performance Analysis Under Bufferbloated Circumstance

1) TCP performance of ABRWDA in a bufferbloated circumstance: The buffer size is excessively bigger than BDP in a bufferbloated network circumstance. The sender will persistently send packets when there is no packet dropping. So the queue length in buffer will grow up quickly, as shown with red line in below picture of figure 4. The queue size reaches to the maximum at 3.51 seconds, and it begins to decrease after a stable state because of the packet dropping. It decreases to the zero at 8.92 seconds. Though after a new window evolvement, buffer size still keeps in a great value in the simulation.

The big buffer size can cause extra long delay. The red line in upper picture of figure 4 stands for the RTT in a bufferbloated circumstance, estimated with the approach in the algorithm 1. We can see that RTT in real networks with NewReno is greater than that with ABRWDA, represented with green line in bottom picture of figure 5. The great spike of RTT is caused by the absence of packets which should be received normally in standard TCP. The zero buffer size harms the throughput seriously just as the full buffer does to RTT. The throughput collapses steeply when buffer size decreases to zero. We don't present throughput for NewReno due to the space limitation.

The essence of ABRWDA is the adaption of the sending rate to the bandwidth variation of wireless links at the existence of large buffer. In figure 5, the green line stands for the performance of ABRWDA. The pink line in the upper picture represents the variation of bandwidth. We can see that ABRWDA can adapt well to the variation of bandwidth dynamically.

The appropriate queue length ensures the perfect throughput as well as low delays. The green line in the middle picture presents the measured queue length of ABRWDA in a 60 seconds simulation. We can see that the maximum queue length does not exceed 63 packets, being bigger than BDP calculated with minimum bottleneck bandwidth (about 40 packets size). The RTT measurements are represented in the bottom picture of figure 5. We can find that it's variation is in a opposite tendency to that of bandwidth, and the RTT values are steadily varied in the range of 0.154 (the delay without the queuing delay) to 0.33 second, which is a great improvement contrasted to the standard TCP without ABRWDA, shown in the upper picture of figure 4.



(a) Thoughput

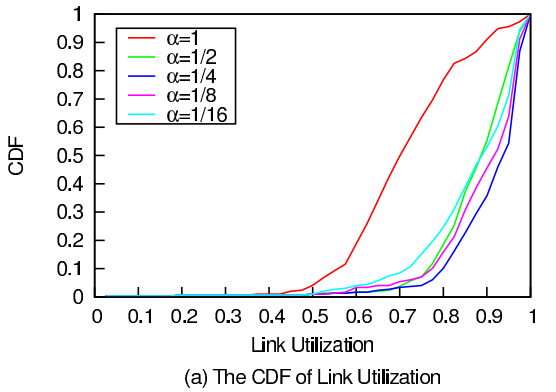
(b) Queue Length

(c) RTT

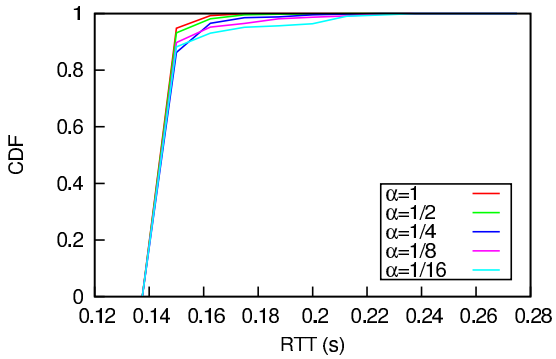
Fig. 5. Throughput, queue length and RTT performance of ABRWDA, DRWA and Vegas in a 60 seconds simulation.

From the figure 5 we can see that ABRWDA keeps an appropriate queue length that can maintain the high throughput and low delay at the same time.

2) TCP Performance of DRWA in a Bufferbloated Circumstance: The performances of DRWA are presented with red line in figure 5. DRWA estimates the *cwnd* with received

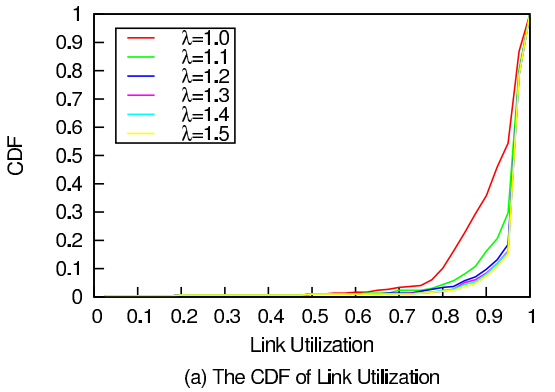


(a) The CDF of Link Utilization

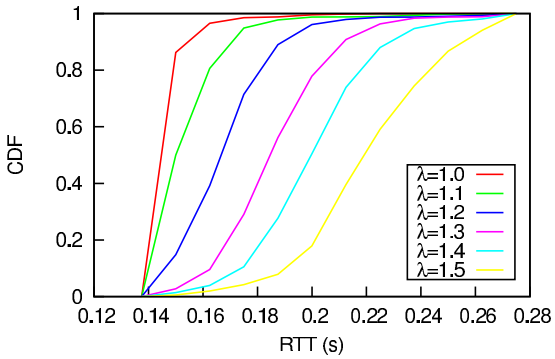


(b) The CDF of RTT

Fig. 6. The CDF of link utilization and RTT with different α values (λ=1).



(a) The CDF of Link Utilization



(b) The CDF of RTT

Fig. 7. The CDF of link utilization and RTT with different λ values (α=1/4).

data at receiver side, and restrains the sending rate with *rwnd* calculated with the estimated *cwnd* and RTT.

Since the estimation of *rwnd* is only based on the transport layer information, DRWA has no knowledge of the real link available bandwidth. So the sending rate can't adapt to the variation of link bandwidth, which can explain the several collapses and the dramatic fluctuation in queue length, as shown in middle picture. And RTT plotted in bottom picture fluctuates accordingly.

From figure 5 we can find that ABRWDA's adaptability to the variation of network states is superior to that of DRWA. ABRWDA can perfectly adapt to the variation of network states, keep queue length within an appropriate range and reduce the RTT naturally.

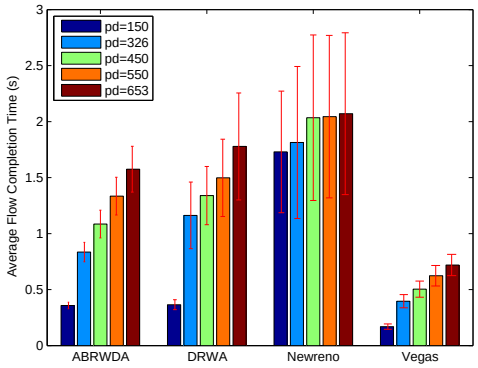
3) TCP Performance of Vegas in a Bufferbloated Circumstance: Vegas is a TCP variant that confines all the changes at the sending side [24]. It adjusts the *cwnd* with the difference of expected sending rate and actual sending one. When there is congestion in network, the difference will increase, then Vegas decreases the sending rate to avoid congestion. Thus, the congestion avoiding is achieved in return for the decrease of throughput. We can see in the upper picture of figure 5 that Vegas's adaptability to the variation of bandwidth is severely poor. The throughput collapses severely in several period of times, and does not vary with the variation of bandwidth, which can been seen at 10.001, 18.109, 40.001 seconds respectively. The average queue length is only 6 packets , with the maximum of 84 packets size at the beginning, as shown in the middle picture of figure 5. In 40.76% of simulation time, the queue length is zero, under which the RTT is relatively small. The average value is 0.166 second, except for a 0.449 second of maximum at the very beginning.

B. The Effect on TCP Performance Caused by Parameters' Selection

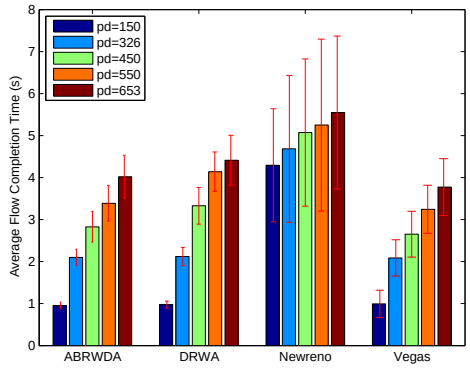
The essence of ABRWDA is the adaption of sending rate to the variation of wireless link state at the existence of large buffer. In the upper picture of figure 5, the pink line indicates the variation of throughput well keeps up with the variation of bandwidth. Several little jitters triggered by bandwidth variation can be found, but they dissipate very soon.

There are two parameters to be configured in algorithm 1: one is α, which is the coefficient of the moving average filter, the other is λ, the scaling factor to adapt to the bandwidth variation. We seek for their optimal values in 2 steps: first, we set λ as 1,and test optimal α with different values; second, we test the optimal λ value with the optimal α value found in the former step.

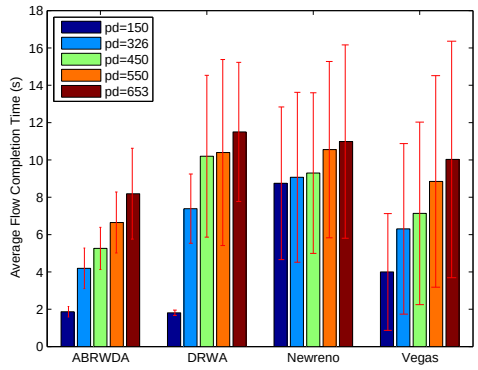
In algorithm 1, we dispose the variation of bandwidth with a smooth average filter. We test the appropriate α in experiments and calculate both the CDF of link utilization and RTT under different α values, shown in figure 6. We can see that compared with other α values, the link utilization is significantly lower when α is 1. All the α values less than 1 produce similar utilizations, and the value of 1/4 produces the best utilization.



(a) The AFCT of a 1 packet-size flow



(b) The AFCT of a 30 packets-size flow



(c) The AFCT of a 200 packets-size flow

Fig. 8. The comparison of average flow completion time (AFCT) of ABRWDA, DRWA, NewReno and Vegas under different network conditions.

In figure 6,though we can see that there is little difference in RTT CDF in the cases of different α values, we can still find that the smaller α is, the less RTT varies. Combining the CDF of link utilization with that of RTT, we choose the 1/4 as the optimal value of α.

We test different values of λ with the optimal α value selected in former experiment, the results are shown in figure 7. In the upper picture, we can see that the link utilizations

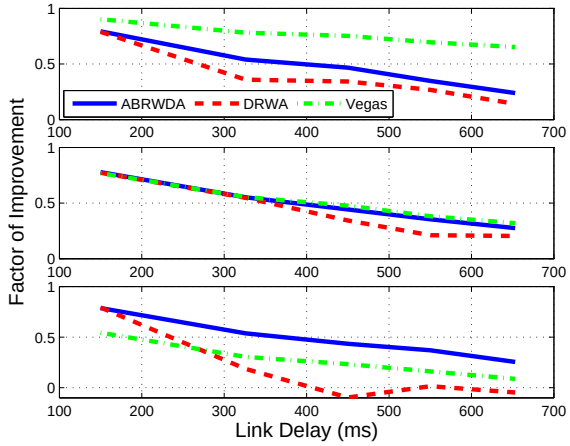


Fig. 9. The performance improvement about AFCT of ABRWDA, DRWA and Vegas with different flow size and network circumstance (Top: 1 packet-size; Middle: 30 packets-size; Bottom: 200 packets-size).

are all close to 1 with different λ values, except for the case when λ is 1.0. Furthermore, as is shown in the bottom picture of figure 7, the bigger λ is, the greater RTT varies. Taking into account both the link utilization and the RTT variation, we set the optimal value of λ as 1.2.

C. The Improvement in User Experiences

Nowadays, mobile applications (such as web browsing, file downloading and online game playing etc.) have flooded into people's lives. When it comes to network service, different applications vary greatly in data volumes. A typical mobile scenario is simultaneous large file downloading and web browsing. In this case, short flows and long flows will coexist on the same network path. In a bufferbloated circumstance, the packets of the long flow will occupy most of the buffer space, causing great queueing delay to short ones. The consequence of the bufferbloat problem could be the slow response to a delay-sensitive application such as simple web page request, which seriously impacts the user experience. So the completion time of short flows is a vital merit to evaluate the user experience in this situation. In this experiment, we compare the averaged short flow completion times (AFCT) among ABRWDA, DRWA, NewReno and Vegas. We start 100 flows in each test, one is long, the others are short. The long flow starts at the beginning of the test, and the short ones start at different time in turn. We emphasize that none of the short flows overlap with each other on the time dimension to avoid the inter-influence among different short flows. The outcome is shown in figure 8 and figure 9.

In this experiment we set three kinds of flow size (1, 30 and 200 packets) to represent clicking the mouse, small web pages and large web components separately. For each flow size, we change the distance between the UE and the server by setting different propagation delays, which are selected according to the settings in [5]. The values in figure 8 are the AFCTs of short flows in each scenario. Comparing the three

subgraph, we can see that as a delay-based protocol acting on the source, Vegas achieves the shortest AFCT when the flow size is 1 packet. However, when as the flow size becomes larger, the advantage of Vegas becomes less obvious, which can be seen from the second and third subgraph of figure 8. In addition, the low delay of Vegas always come with the cost of throughput loss. DRWA and ABRWDA are both receiver-based algorithms, and they both perform well with the flow size being 1 or 30. Nevertheless, when the flow size is 200 packets, DRWA loses effect, and in some cases its performance is even poorer than NewReno. In the four algorithm, only ABRWDA keeps good performance with three flow sizes. What's more, AFCT's growth of ABRWDA is very steady in each network circumstance. Compared with NewReno, The performance improvement factor of AFCT for ABRWDA, DRWA and Vegas are shown in figure 9. We can find that the performance of ABRWDA is very steady, however, neither of Vegas and DRWA can maintain the performance as the flow size increases.

VI. Conclusion

Cellular networks deploy large buffers to absorb traffic bursts and to achieve the link layer retransmission mechanism for better performance. As a result of the existence of link layer reliable retransmissions and large buffers, the packet dropping, which is referred to as the signal of the congestion by TCP, is totally concealed. Without the packet-loss-driven decrease in the sending window, excessively long delays are caused, and then deteriorate the TCP performance. This phenomenon is named as bufferbloat. In this paper, we propose ABRWDA, a receiver-based solution to mitigate bufferbloat in cellular networks. ABRWDA obtains the wireless link bandwidth by means of the relationship between bandwidth and SINR, then uses it and the estimated RTT to calculate the receiving window which will be used to adjust the sending window. We implement ABRWDA with NS-2, and compare its performance with the current receiver-based solution DRWA and the delay-based TCP version Vegas. The experiment results manifest that ABRWDA can achieve considerable performance benefits compared with other approaches.

Acknowledgment

The authors gratefully acknowledge the anonymous reviewers for their constructive comments. This work is supported in part by National Basic Research Program of China (973 Program) under Grant No. 2012CB315803 and National Natural Science Foundation of China (NSFC) under Grant No. 61225011.

References

[1] "GLOBAL INTERNET PHENOMENA REPORT 1h 2014." https://www.sandvine.com/downloads/general/global-internet-phenomena/2014/1h-2014-global-internet-phenomena-report.pdf, 2014. [Online; accessed 19-July-2014].

[2] "GLOBAL INTERNET PHENOMENA REPORT 1h 2013." https://www.sandvine.com/downloads/general/global-internet-phenomena/2013/sandvine-global-internet-phenomena-report-1h-2013.pdf, 2014. [Online; accessed 19-July-2014].

[3] J. Gettys and K. Nichols, "Bufferbloat: Dark Buffers in the Internet," *Queue*, vol. 9, no. 11, p. 40, 2011.

[4] K. Nichols and V. Jacobson, "Controlling Queue Delay," *Communications of the ACM*, vol. 55, no. 7, pp. 42–50, 2012.

[5] H. Jiang, Y. Wang, K. Lee, and I. Rhee, "Tackling Bufferbloat in 3G/4G Networks," in *Proceedings of the 2012 ACM conference on Internet measurement conference*, pp. 329–342, ACM, 2012.

[6] R. Pan, P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg, "PIE: A Lightweight Control Scheme to Address the Bufferbloat Problem," in *High Performance Switching and Routing (HPSR), 2013 IEEE 14th International Conference on*, pp. 148–155, IEEE, 2013.

[7] X. Liu, A. Sridharan, S. Machiraju, M. Seshadri, and H. Zang, "Experiences in a 3G Network: Interplay Between the Wireless Channel and Applications," in *Proceedings of the 14th ACM international conference on Mobile computing and networking*, pp. 211–222, ACM, 2008.

[8] H. Jiang, Z. Liu, Y. Wang, K. Lee, and I. Rhee, "Understanding Bufferbloat in Cellular Networks," in *Proceedings of the 2012 ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design*, pp. 1–6, ACM, 2012.

[9] V. Paxson, M. Allman, and W. Stevens, "TCP Congestion Control," *RFC2581, April*, 1999.

[10] M. Li, Y.-L. Wu, and C.-R. Chang, "Available Bandwidth Estimation for the Network Paths with Multiple Tight Links and Bursty Traffic," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 353–367, 2013.

[11] D. Reed, "What's Wrong with This Picture." http://mailman.postel.org/pipermail/end2end-interest/2009-September/007742.html, 2014. [Online; accessed 09-July-2014].

[12] W. K. Leong, Y. Xu, B. Leong, and Z. Wang, "Mitigating Egregious ACK Delays in Cellular Data Networks by Eliminating TCP ACK Clocking," in *ICNP*, pp. 1–10, 2013.

[13] Y.-C. Chen and D. Towsley, "On Bufferbloat and Delay Analysis of Multipath TCP in Wireless Networks," in *Networking Conference, 2014 IFIP*, pp. 1–9, IEEE, 2014.

[14] H.-Y. Hsieh, K.-H. Kim, Y. Zhu, and R. Sivakumar, "A Receiver-Centric Transport Protocol for Mobile Hosts with Heterogeneous Wireless Interfaces," in *Proceedings of the 9th annual international conference on Mobile computing and networking*, pp. 1–15, ACM, 2003.

[15] Y. Xu, W. K. Leong, B. Leong, and A. Razeen, "Dynamic Regulation of Mobile 3G/HSPA Uplink Buffer with Receiver-Side Flow Control," in *Network Protocols (ICNP), 2012 20th IEEE International Conference on*, pp. 1–10, IEEE, 2012.

[16] N. T. Spring, M. Chesire, M. Berryman, V. Sahasranaman, T. Anderson, and B. Bershad, "Receiver Based Management of Low Bandwidth Access Links," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, pp. 245–254, IEEE, 2000.

[17] H. Hu, H. Yanikomeroglu, D. D. Falconer, and S. Periyalwar, "Range Extension without Capacity Penalty in Cellular Networks with Digital Fixed Relays." http://www.sce.carleton.ca/faculty/yanikomeroglu/Pub/gc04_hh.pdf, 2004. [Online; accessed 15-July-2014].

[18] "Average Cell Throughput Calculations for LTE." http://www.raymaps.com/index.php/average-cell-throughput-calculations-for-lte/, 2011. [Online; accessed 15-July-2014].

[19] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck, "An In-Depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance," in *ACM SIGCOMM Computer Communication Review*, vol. 43, pp. 363–374, ACM, 2013.

[20] W.-c. Feng, M. Fisk, M. Gardner, and E. Weigle, "Dynamic Right-Sizing: An Automated, Lightweight, and Scalable Technique for Enhancing Grid Performance," in *Protocols for High Speed Networks*, pp. 69–83, Springer, 2002.

[21] T. Issariyakul and E. Hossain, *Introduction to Network Simulator NS2*. Springer Science & Business Media, 2011.

[22] A. Gurtov and S. Floyd, "Modeling Wireless Links for Transport Protocols," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 85–96, 2004.

[23] F. Ren and C. Lin, "Modeling and Improving TCP Performance over Cellular Link with Variable Bandwidth," *Mobile Computing, IEEE Transactions on*, vol. 10, no. 8, pp. 1057–1070, 2011.

[24] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," *Selected Areas in Communications, IEEE Journal on*, vol. 13, no. 8, pp. 1465–1480, 1995.