Summary Review Documentation for

“Tackling Bufferbloat in 3G/4G Networks”

Authors: H. Jiang, Y. Wang, K. Lee, I. Rhee

Reviewer #1

Summary: This paper starts off with a measurement based study of the bufferbloat phenomenon in cellular networks, and show that bufferbloat affects the RTT and throughput of TCP connections significantly across many different US cellular networks and some networks in Korea. They also measure TCP connection performance on several smartphones and find that Android phones exhibit a strange phenomenon of flat pipes since they by default set the TCP rmem parameter to some static value whereas the BDP of these networks is larger. So while TCP stacks in other phones keep probing for available bandwidth, Android phones cap it off at some value. They solve the problem by proposing a scheme called DRWA (dynamic receive window adjustment) where essentially the rmem parameter is dynamically adjusted to some factor λ of RTT_{min} (propagation delay). So, that way queue size never grows beyond this factor keeping latencies in check. They demonstrate it works well in practice using a real Android implementation.

Strengths:

- Good measurement study to illustrate the problem
- Nice cute idea to solve the problem.
- Very well written paper with nice solid evaluation and experimentation.

Weaknesses:

- DRWA idea itself is quite incremental given prior work.
- This paper only solves the receiver side TCP problems, but what about the sender side?

Comments to authors: I like this paper since it is a nice overall story of identifying a problem through extensive measurements, suggesting a simple working solution, and finally evaluating the efficacy of the solution through a nice prototype implementation in real networks.

The main downside of this paper is that the ideas are relatively straightforward and are quite incremental to prior work. The solution can be termed a cute ‘hack’. But the hack only works in the receive path, but not on the transmit path. One could argue mobile phones are download intensive, so they may be solving a more important problem, but it would be important to also consider the transmit direction as well.

Despite their argument that it is different from DRS, the algorithm is remarkably similar to DRS, and involves are a very small

change. In that sense, the algorithm is very incremental. I understand DRS has been proposed for a different reason, but the algorithm is too similar to DRS that I am not able to appreciate the solution’s novelty as much as I want to.

I am a little worried about the robustness of the hack. Specifically, 13(a) indicates quite a bit of oscillations in the receive window size. I am worried that this may affect the TCP throughput under some conditions. Unfortunately, for the same scenario, they don’t compare TCP throughput with and without their hack. They only measure the receive window size and the RTT values.

I am also a little worried that their work mainly in networks that have very high propagation delay such as almost 400-700ms compared to static values. While I can understand such delays when connecting to foreign servers, most connections tend to be local without such large propagation delays.

Also, as the authors point out, delay based solutions such as TCP Vegas can easily work well without facing this bufferbloat issue. The main argument is that this is a sender-based solution which require changing servers. I think given bufferbloat problem is not just common among cellular networks but also cable modems and such, delay-based congestion control should be used at servers more often. Any ways, that is orthogonal to this paper.

Reviewer #2

Summary: This paper discusses the impact of “bufferbloat” (the use of large buffers) in cellular networks. The paper reports on detailed end-to-end performance measurements (RTT, throughput, and object download times) across multiple cellular network providers and different wireless technologies. They demonstrate the existence and impact of bufferbloat, and discuss the performance implications of different TCP variations and implementations. They finally propose a new TCP variant - Dynamic Receive Window Adjustment (DRWA) - designed to enhance performance in such networks through changes only on the receiver side (UE). They demonstrate that DRWA promises significant improvements both for RTT and throughput under various conditions.

Strengths: The paper was well written and interesting to read. The paper presented a reasonably thorough experimental investigation of the performance of TCP variants over different production cellular networks. They demonstrated the impact of buffer-bloat in 3G and LTE networks on RTT, throughput and download times. They also went on to propose an apparently very effective TCP variant that could be implemented with code modifications on the receive (handset) only.

I enjoyed reading the paper and thought that it was very interesting work.

Weaknesses: The authors focus on fairly specific scenarios and do not pull in as much of the "bigger picture", which would benefit the reader in terms of placing the importance of these TCP limitations in their context. For example, by (primarily) neglecting the state transitions and by focusing on downloads of large files from very remote servers, they are likely neglecting the vast majority of traffic. How much do the limitations revealed in this paper and the proposed solution really impact mobile users?

Comments to authors: Do the authors have information regarding how significant the TCP performance issues are for realistic cellular traffic? The authors have not demonstrated how often these problems happen in practice and if they happen with adequate severity that people will care.

The paper mostly ignores the impact of RRC state transitions and its (very significant) impacts on performance. More discussion on this would help put the discussions here in better perspective. The state transitions can significantly dominate the performance of cellular applications.

First hop buffering: The authors demonstrate that the buffering is in the "cellular network" as compared with the Internet by showing that the delay is up to the first hop IP. From the paper - "The results show that the queue is mainly in the first IP hop which definitely contains the cellular link." The authors seem to be implying that the buffering is likely over the wireless link between the user device and the cell tower (or on the cell tower/NodeB/eNodeB). However, the "cellular network" that carries traffic to the first hop IP is actually a large number of network elements - in a 3G network this would include the wireless portion, the NodeB, RNC, SGSN, GGSN, and the potentially tens of network elements that provide the connectivity between these core mobility network nodes - often including the same IP network that carries Internet traffic. Thus, proving that the delay is up to the first hop IP is not a solid proof that it is in the wireless portion of the network - if this is what the authors are implying. The authors should thus be careful about how they word this and how strongly they draw inferences.

"However, in cellular networks, a base station typically has separate buffer space for each user". Can the authors please provide a reference for this? Buffering in NodeBs/eNodeBs is complicated, with multiple different buffers. And at least one eNodeB vendor document states that buffering is performed in a common pool without fixed per UE partitions.

The authors should also complete a proof read of the paper - there are a number of typos etc.

Reviewer #3

Summary: The paper conducts measurements of the 3G/4G networks to study the extent of bufferbloat in these networks. The paper then proposes using dynamic receive window adjustment to reduce latency and improve throughput in the case of over-buffered cellular networks. The paper performs extensive measurements of cellular networks, examines the existing TCP stacks in smartphones, and proposes modifications to the TCP stack on these phones to reduce bufferbloat problems. The paper uncovers some interesting artefacts of TCP stacks in smartphones, such as a "flat TCP", which keeps the congestion window constant for the duration of the connection.

Strengths: Issues of bufferbloat in various parts of the network are receiving much attention, and this is the first extensive study of its kind of over-buffering in cellular networks. The paper also uncovers some interesting artefacts of the TCP stacks of existing smartphones. This paper is a very good (and citable) study on the current state of TCP behavior of smartphones in 3G and 4G networks.

Weaknesses: The findings in the paper are fairly expected (and mundane), and the proposed solutions to bufferbloat (which the paper spends perhaps almost half of its size on) are not particularly interesting, and not thoroughly compared to other solutions. For example, a well-known solution to bufferbloat is to perform traffic shaping at a rate that is slightly less than the rate of the uplink. The paper does not compare its approach to any of the many simpler possible solutions to the bufferbloat problem.

Comments to authors: Overall, I enjoyed reading this paper, and I think it could make for a good paper at the conference. Many readers will likely have experienced bufferbloat problems in cellular networks, so the problem that this paper is studying is certainly of interest and relevance. Some of the behaviors of the TCP stacks on the smartphones, such as the observed "flat TCP" behavior, were interesting and surprising.

That said, I did not find the dynamic receive window adjustment algorithm to add much to my understanding of bufferbloat, and it is not clear that this approach is the only (or best) way to mitigate bufferbloat effects.. This section of the paper actually weakens the paper because the solution is relatively straightforward, and yet it is not compared to existing (and even simpler) solutions to the bufferbloat problem, such as performing traffic shaping at the client (see, for example, the SIGCOMM 2011 paper from Sundaresan et al., which proposes using Wondershaper on the client or first hop to mitigate the effects of bufferbloat).

Although it is clear that DRWA works well, the increment over DRS is small, and the proposal is not compared to DRS or any of the other possible solutions to the problem. The contribution of this part of the work is thus less (and less rigorous) than it could be.

Reviewer #4

Summary: This paper measures the effects of "bufferbloat" cellular networks using 4 major US providers and one carrier in Korea. The paper shows that buffer-bloat is indeed severe in all carriers and for all 3G/4G technologies tested, that it affects user experience by increasing RTT and that high-speed tcp variants compound the problem.

The paper proposes a very simple algorithm to dynamically adapt the receive window to tackle RTT inflation, implements it and tests it in practice. The results are quite promising, showing much better delays and even higher throughput compared to the current tactic using a small, static, max receive window.

Strengths: I like your paper - it is very clear, well-written and thorough.

Buffer-bloat and the associated RTT inflation is an important problem to address, although it is probably still not critical today because users rarely do downloads & interactive traffic at the same time. However, this could easily become an issue in the near future, as the authors suggest.

It is about time someone took the time and solved this problem for cellular networks - it is much easier to solve in this case than in general.

The reason this simple approach works is because, as the authors say, cellular base-stations keep per user buffers; as long as this is true, and the bottleneck is the cellular network (again, this is the case today) this simple solution seems to get the job done well. And it might just be deployed: it is much better than the ad-hoc small max rwnd of today.

Weaknesses: I would like to see what happens if the bottleneck is not the cellular link - how does DRWA work when fighting with regular TCP? Although this is not the case today, it is good to understand proposed solutions that could affect millions of devices! You need to vary buffer size at the bottleneck (i assume DRWA will work just fine with buffers $\geq 2 \times \text{BDP}$, and start losing out afterwards.

Comments to authors:

- please explain lambda a bit earlier (or just say it is ≥ 1) so that it makes it easier to understand your algorithm without reaching the end of its description.
- I would be inclined to choose lambda = 2 instead of 3 - it gives much lower RTTs at a modest decrease in throughput. Also, it stops the sender from keeping more than BDP data in the buffer, which would fight just fine with regular TCPs at a bottleneck with BDP-sized buffers.
- the discussion of why it is difficult to reduce buffer size in cellular networks is a bit confused - surely the main reason is the lack of knowledge of end-to-end path properties, such as RTT? Can't these be estimated also?

Reviewer #5

Summary: This paper is experimentally quantifying the impact of bufferbloat on TCP in 3G/4G networks. The authors have conducted a large number of experiments in 4 cellular networks in the USA and 1 in Korea and assessed the extent of queue buildup and its impact on TCP performance. They have interestingly revealed the approach that Google has taken to limit the effect of bufferbloat and discussed the problems that may arise with the static setting of the TCP receiver window. They propose a simple fix that could allow cellular devices to make the best use of the cellular pipe.

Strengths:

- interesting and extensive set of measurements quantifying the extent of bufferbloat
- discussion of the solutions implemented by smartphones to deal with bufferbloat
- solution on dynamic adjustment of TCP receiver window and evaluation

Weaknesses: The evaluation of DRWA is rather limited since it is not contrasted to Vegas neither DRS itself

Comments to authors: I really liked this work. The authors start from a solid measurement framework to study the extent of bufferbloat in today's networks and its impact on TCP performance. Each experiment is carefully selected to make the right point and lead to the subsequent part of the paper. While the original measurements are taken on laptops the authors proceed to test actual smartphone TCP implementations. This is where the authors reveal Google's interesting approach to deal with bufferbloat. They use that observation and propose a very reasonable solution - to dynamically adjust the TCP receiver window. They manage to patch Android phones using 100 lines of code and demonstrate performance improvements for bufferbloat short BDP paths, as well as long BDP paths.

While the solution is simple I think its power is its simplicity and the fact that with 100 lines of code one can get the presented performance improvements.

Having said that the evaluation in the current manuscript could certainly be improved. You discuss the potential for Vegas to provide a solution to the problem. I would like to see a comparison of DRWA and Vegas. Moreover, given that DRWA is based on DRS, I think such a comparison should also be provided. Then you can better discuss the elements of DRWA that work/do not work more effectively. A final experiment with some latency sensitive application, like Skype, would also be interesting. Could you see a perceivable difference?

You pick a lambda of 3 based on Figure 14. I would argue that a value of 2 gives you the majority of the throughput improvement without greatly inflating your RTT.

Lastly, the paper could greatly improve in terms of its clarity. You tend to provide the results of the experiments without describing the experiment itself - for instance for Fig.9 you do not describe the setup, nor how many times the experiment was run. This is essential for a measurement paper and most of your results are not accompanied by a proper experiment description (the only time where I saw this was below Fig. 18).

Some more detailed comments:

- Figure 3 what happens to the red line after 3000?
- you should have a table with the BDP of the different network experiments since you do discuss quite a lot the relationship between the congestion window and BDP (section 2.1)
- Fig.8: It is curious to see that iPhone and MacBook perform similarly, but laptop with Windows 7 and Windows Phone are different. Why? Do they use a different TCP stack?

Response from the Authors

We thank the anonymous reviewers for their insightful comments. We have modified the paper in the following major aspects to address these comments:

1. We added Table 1 in the appendix to summarize the setup of each experiment so that the readers can look it up easily. We also strengthened Section 2.1 and Figure 2 to offer more details of our measurement setup.
2. We carried out additional experiments in Section 7.1 to compare DRWA with alternative solutions like delay-based congestion control (Figure 19) and traffic shaping at the sender

(Figure 20). We also refined our arguments against other solutions that require infrastructure change and hence cannot be compared experimentally.

3. We added throughput performance to Figure 12 to address the concern of Reviewer A. We also added the performance of DRWA in non-bufferbloat networks such as WiFi in Figure 13. Both verified the robustness of our proposal.
4. We clarified in Section 6.1 that a high propagation delay is common in cellular networks due to the fact that all cellular data traffic must detour through a few IP gateways. Hence, the throughput improvement brought by DRWA is observable even if the client and server are close to each other geographically.
5. We replotted Figure 14 so that the performance of different types of networks are shown in different scales. The origi-

nal figure had everything in the same scale. Since LTE has a much higher throughput than EVDO and HSPA+, it obscures the throughput differences in those networks under different λ settings. Hence, a number of reviewers preferred $\lambda = 2$ to $\lambda = 3$. In the current figure, it is clear that $\lambda = 2$ may suffer from throughput loss in certain scenarios and hence we picked $\lambda = 3$.

6. As suggested by the reviewers, We toned down a few statements in the paper to make them more meticulous. Particularly, instead of trying to argue that DRWA is a systematic and complete solution to bufferbloat, we emphasize that it is a simple and easy-to-deploy solution that does the job.
7. We also fixed all other minor issues (typos etc.) in the paper.