The Impact of Transport Problems on Large Scale Web Services

Ethan Katz-Bassett
University of Southern California
NSF/FCC QoE Workshop White Paper

Introduction

"You're a networking guy. Go figure out why `phone-model-X` *has poor TCP performance on* `cell-provider-Y`*."* And so started my first day working at Google. I was given some TCP packet captures from a single Google server and asked to figure out–and hopefully fix–performanced. There were a few problems with this:

- EVERY flow looked like it had MANY problems with it. How could I tell the difference between issues that were weird compared to my basic textbook knowledge of TCP and things that were weird even compared to how TCP looks at the scale of Google? And how could I do this quickly, given that I had no frame of reference for what the traces should look like or what different patterns might signify?
- Suppose I could figure out what was going on in a single flow. How could I be sure that the issues I was looking at was prevalent in traces for `phone-model-x` on `cell-provider-Y` across Google?
- Suppose I was able to explain what was going on, AND the issue was widespread across `phone-model-x` on `cell-provider-Y`. With this isolated TCP-level, server-side trace, how could I be sure that the issues I saw even impacted the clients on the other end of the connection? Were they masked by mechanisms above the TCP layer or dwarfed by other issues that played a larger role in determining user experience?
- Suppose I was able to explain what was going on, AND the issue was widespread across `phone-model-x` on `cell-provider-Y`, AND it was impacting user experience. How was I ever going to be able to improve user experience? I had never even looked at kernel code before, and TCP performance has been an area of serious effort by many experts for years. What was the chance I could think of an improvement that they had missed?
- Suppose I was able to explain what was going on, AND the issue was widespread across users of `phone-model-x` on `cell-provider-Y`, AND it was impacting user experience, AND I was able to come up with a fix to improve performance. That seems like a lot of work! How would I know that that had been the right problem to focus on? What if there were other problems we didn't even know about that were causing even more problems?

These questions came to define one of the strands of my research: how can we identify and fix the transport problems that are having the biggest impact on Quality of Experience worldwide?

Prior Work: Reducing Web Latency: The Virtue of Gentle Aggression

The need for faster loss recovery. To demonstrate a common cause of high delays, we collected packet captures from multiple Google frontend servers yielding a dataset of billions of TCP flows. As seen in Figure 1, web transfers with loss are five times slower on average. It depicts the TCP latency in the traces (the time between the first byte the server sent to its receipt of the last ACK), separating the transfers that experienced loss from those that did not experience loss. While loss-free transfers take little more than the ideal duration, their lossy counterparts take five times longer. The slow loss recovery is a result of the delay of timeout-based loss recovery, in which the sender assumes that a packet is lost if it does not receive an ACK within a certain time frame. In our study, 77% of the losses were recovered through this recovery type rather than faster mechanisms such as fast retransmit.

Improve latency through redundancy. We designed three TCP enhancements that add different amounts of redundancy (and therefore different amounts of overhead) to transfers to avoid timeouts and improve latency. A key challenge is the embedding of these packets into the existing TCP stream while maintaining compatibility with middleboxes and without worsening network congestion. Please see our paper for details on the techniques [2]. We tested

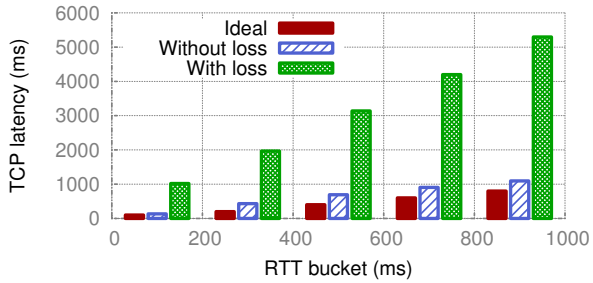

Figure 1: Mean TCP latency to transfer an HTTP response from Web server to a client. Measurements are bucketed by packet round-trip time between the frontend and the client.

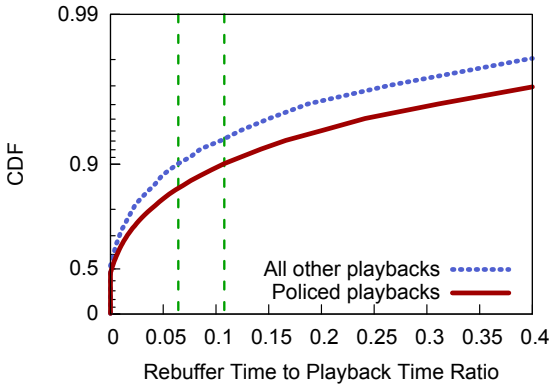

Figure 2: Rebuffering ratios for a sample of $\sim$ 1 million unpoliced and policed playbacks respectively. Each of the playbacks had at least one of its chunks delivered at a goodput rate between 250 and 350 kbps. The vertical dashed lines mark 90^{th} percentile values for each curve.

the techniques on real client traffic through a deployment in Google's production network. Combined, they improved the average web transfer latency by 23%, and in the 99th percentile we observed gains of 47%. One of the techniques is now default on in Linux.

Takeaway. Once we knew to focus on slow timeout-based loss recovery, it was possible to modify TCP for large gains. However, identifying the problem to address had taken a long time, and it had required the efforts of a number of TCP experts. We next asked: Can we encode some of their expert knowledge in heuristics, then apply those heuristics at Internet scale, in order to remove the human-in-the-loop bottleneck in identifying which problems are important?

Prior Work: The Case Against Policing Speed Limits on the Information Superhighway

Expert analysis at scale. With collaborators at Google, we built a pipeline that captures packet flows for a small fraction of YouTube streams worldwide and uses heuristics developed by TCP experts to identify problems.

Detecting traffic policing. One of the heuristics we encoded identifies flows that experience policing. A traffic policer enforces a preconfigured rate per flow by dropping any packets that exceed the rate, whereas a shaper buffers the excess. ISPs have resorted to policing and similar techniques to manage the heavy volumes of data flowing into their network from video and other popular services. In our dataset of over 270 billion packets served to YouTube clients in over 28,400 ASes, between 2.8% and 6.6% of the traffic is policed, depending on region. Policing has a measurable impact on the users quality of experience. Figure 2 shows the distribution of the rebuffer time to playback time ratio for playbacks with at least one of their video segments delivered at a goodput rate between 250 and 350 kbps. This rate is the minimum required to play videos of the lowest rendering quality, and therefore leaves little opportunity to bridge delayed transmissions by consuming already buffered data. About half of the playbacks do not see any rebuffer events. For playbacks with rebuffering, policed playbacks perform up to 40% worse than the unpoliced ones. For example, in the 90^{th} percentile, policed flows experience a rebuffering ratio of 10.8% vs. 6.4% when not policed. Prior work found that a 1% increase in the rebuffering ratio can reduce user engagement by 3 minutes [1].

Future Work: Structuring Packet Captures With Neural Networks

The two studies above were both preceded by a manual analysis of packet traces conjecturing about potential causes for the observed performance degradation, or anecdotal evidence by other parties providing possible explanations. This human intervention enabled our investigations and design of solutions to improve performance. But what can we do about all the other situations where we observe high latency yet do not have any clue about the root causes? Were there more important problems that we should have looked at instead of policing that had a larger impact on QoE? Was our TCP experts' time best spent tuning heuristics to recognize policing? For example, there are many traces wth slow transfers yet low loss rates, indicating that something besides packet loss has to be responsible for transfer times being inflated. We do not have the means to manually look at all these flows for a first assessment.

This motivates our future work: Our goal is to find common features in the vast amount of measurements available to us that can provide hints towards potential root causes for bad QoE, pointing us to potential issues that warrant further investigation. Since we do not know the root causes yet, we want to identify large groups of flows with similar features, assuming that similar problems lead to similar effects. However, these effects are not necessarily reflected in a single metric like packet loss. For example, while traffic policers drop packets, loss rates vary widely depending on the interaction between TCP and the policer. More generally, we want to learn more about the *hidden structure* of the data which ultimately lets us identify the causes for bad performance.

We are exploring using neural networks to find structure hidden in our data sets to derive new knowledge. In our provisional work, we are exploring extensions to our prior work above.

Sequence loss prediction. Can we predict with high probability if a particular packet is lost or not, given data about preceding packets? Answering this question is similar in flavor to previous studies that tried to predict the next word in a text sequence [3]. Accurate predictions of the likelihood of upcoming loss can inform the sender to transmit data less aggressively or employ redundancy measures like the ones discussed in the prior work section.

Anomaly detection. Then we will use supervised learning to detect anomalies, like Bufferbloat or traffic policing, training using our heuristics to label each data entry. A network that finds more (or less) anomalies in the test dataset provides indicators about possible limitations of the heuristic used in the first place, as when the network learns a different way to classify inputs.

Ultimately, we plan to use QoE information to label video playbacks (e.g., does the rebuffering ratio of a playback indicate a good or bad experience?), then use these labels to train a network to recognize good versus bad playbacks from TCP-level features. We want to find different clusters of packet traces that exhibit similar characteristics (i.e., similar activation patterns in the network) leading to poor experiences, without knowing ahead of time what the defining features for each cluster are. The hope is that the clusters will pull out problems that would be hard for a human to discover, such as anomalies that are sparse within any given trace but significant when considered across geography and time. By grouping together traces with similar underlying structure, we can give to a human expert relevant traces to help understand the problems most impacting QoE, as a starting point towards fixes.

Relationship to the Broader QoE Research Agenda

Our dataset uses a different vantage point than much of the academic work: it is a global view from the server side, rather than the more limited scope of client side views from, for example, Sam Knows or BISmark. Tackling QoE issues from both sides will likely result in a better solution than either side in isolation. Our focus on understanding the effect of transport-layer issues on QoE will become more fruitful as we incorporate others' research on the metrics that most impact QoE [1].

References

[1] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. A. Joseph, A. Ganjam, J. Zhan, and H. Zhang. Understanding the Impact of Video Quality on User Engagement. In *Proc. of SIGCOMM*, 2011.

[2] T. Flach, N. Dukkipati, A. Terzis, B. Raghavan, N. Cardwell, Y. Cheng, A. Jain, S. Hao, E. Katz-Bassett, and R. Govindan. Reducing Web Latency: The Virtue of Gentle Aggression. In *Proc. of SIGCOMM*, 2013.

[3] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to Sequence Learning with Neural Networks. In *Proc. of NIPS*, 2014.