R02 Network Architectures Essay 2

Alex Chau (ac2822), Churchill College

1 Introduction

In this essay I will be covering two papers; PCC Vivace – Online-Learning Congestion Control and DOTE. Both of these papers show how ML techniques can be applied to networks, the former being an application of a learning framework to congestion control. The latter applying ML to enable robust traffic engineering. Whilst both techniques overall aim to maintain network performance and prevent overload, congestion control is reactive, whereas traffic engineering is proactive. A naive approach would be to treat ML/AI as a magic-wand to networking, in reality the applicability of these techniques is highly dependent on the network constraints and how best you can tailor the ML technique to solve the problem.

2 PCC Vivace

PCC Vivace is a novel rate-control protocol which realizes the PCC framework from an online-learning perspective. The original realization of PCC, known as "PCC Allegro" takes a black box network-modeling approach to the problem. Allegro divides time into consecutive Monitor Intervals (MIs), where each interval is used to test the performance implications of sending at a specific rate. These metrics are collected and fed into a utility function, outputting a numerical value. It then adapts the sending rate in the direction that is empirically associated with higher utility. The paper's experiments indicate that this technique fails to achieve optimal low latency, has an overly long convergence time and are highly aggressive towards TCP. Some of the reasons for the far-from optimal performance is Allegro's simple utility function (essentially plotting rate against utility) and lack of latency-awareness. One of the reasons for the simple utility function is that adding more parameters such as latency makes reasoning about tradeoffs in utility all the more difficult, and adding more parameters to the utility function further complicates this issue. Allegro also has a naive aproach to rate control, as it uses a fixed step size which is bound to be too much or too little to converge or react appropriately to maximum utility. These choices prevent Allegro from attaining good performance under rapidly changing network conditions and fails to alleivate bufferbloat. Whilst Allegro produces a convergence rate/stability tradeoff that is better than TCP, it leaves much to be desired and causes high packet loss upon

convergence.

PCC Vivace addresses the limitations with Allegro by using ideas from online convex optimisation, replacing the utility function framework with an online learning theory-informed framework for utility derivation, incorporating latency into the utility function. Latency-awareness is incorporated into the utility function via an RTT gradient, where this value represents the change in latency experienced within the MI. This decision guarantees multiple competing Vivace senders converge to a near-optimal, unique, and fair stable rate configuration. Vivace then employs a gradient-ascent based no-regret online optimisation to adjust the sending rates. Experimentally, Vivace significantly improved on the state of the art (at the time of 2018) achieving fast convergence, better performance, greater reactivity and being more TCP-friendly. Additionally Vivace is easily deployed, requiring sender-only changes.

The approach to Vivace is well-intentioned, you can imagine that with the interest around AI there would be attempts to overhaul the entire PCC framework with a deep neural net to leverage their inherent inference/predictive abilities. At the timescale of MIs, you cannot afford the heavy inference provided by DNNs and state changes too quickly for offline training to be useful. Instead, the authors subtly lean into machine learning techniques rather than just flatly applying predictive modeling via a neural network as would be done with a ”hammer” approach. By appropriately borrowing ideas from convex optimisation and online-learning, the authors are able to provide (mathematically provable) performance guarantees about their utility function. On the flip-side, lets say you were to implement a DNN for congestion control (assuming it can be done fast enough). Whilst you can account for other metrics other than throughput, latency and loss over an MI, there are no performance guarantees about the outputs of the DNN. The solution would largely be a black-box, a DNN could probably discover some useful parameter configurations which trade off metrics even more optimally than Vivace does over MIs but would require substantial compute, which isn't feasible for typical end devices like mobile devices. Additionally how would a DNN react to network conditions that differ substantially from what they're trained on? Vivace can loosely be described as an ML/AI approach, framing the PCC problem from an online-learning persepctive and using a subset of optimisation techniques to overcome the identified issues seen in Allegro.

3 DOTE

4 Comparisons

Between Vivace and DOTE there's this trade-off between time-scale and complexity.

5 Further work

In 2020, the same research group followed up with PCC Proteus, applying the PCC Vivace framework to "Scavenger" traffic. In this paper they show that the same Vivace Utility function can be generalized to adopt priority??????