Homework 2 - Congestion Control

Cenk Baykal (baykal@mit.edu)
Wilko Schwarting (wilkos@mit.edu)
Alex Wallar (wallar@mit.edu)

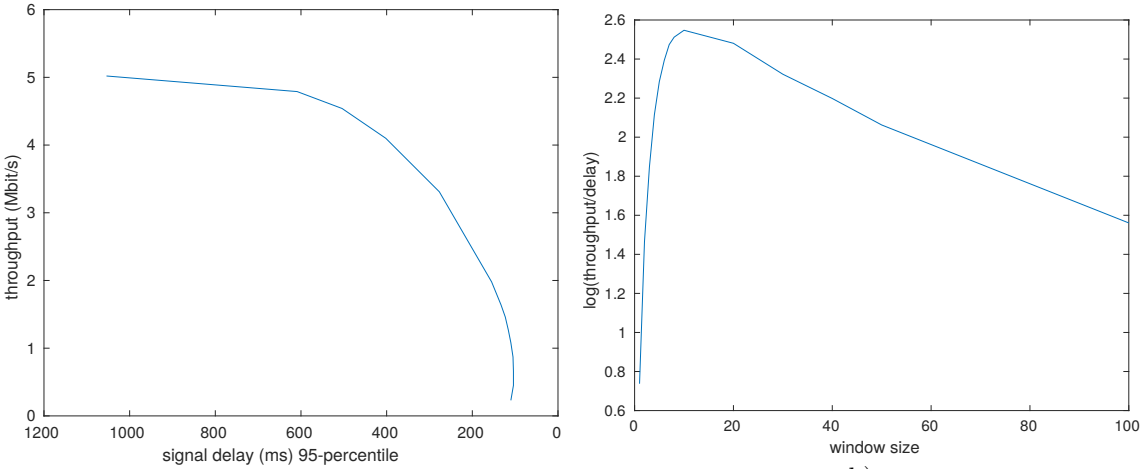
October 27, 2016

1 Fixed Window Size Scheme

We implemented a naive approach to congestion control that is based on a fixed window size, W. We measured the throughput and the 95-percentile delay for various values of congestion window size $W \in [1, 100]$. Fig. 1 depicts plots of throughput vs. 95-percentile signal delay and the score of each corresponding window size W computed as a function of the throughput and 95-percentile delay, i.e.,

$$score(W) = \log\left(\frac{T(W)}{D(W)}\right)$$

where $T(W)$ and $D(W)$ denote the throughput and 95-percentile delay under congestion window of size W respectively.



a) b)

Figure 1: a) Plot of throughput vs. 95-percentile signal delay obtained by experimenting with varying values of fixed congestion window size. The trade-off between high throughput and low delay can be seen by the Pareto frontier-like curve above. b) The logarithm of the throughput to 95-percentile delay ratio. The congestion window size that attains the maximum score was found to be $W^* = 10$.

Fig. 1 depicts the trade-off between the two competing objectives and resembles a Pareto frontier. We can see from the figure that higher throughput generally comes at the cost of higher induced signal delay and vice-versa. The plot of the score shows that values of congestion window size in the range $[10, 20]$ generally perform the best in the emulated TCP network. In particular, we found that the best single window size that maximizes the overall score is $W^* = 10$ packets.

Due to the highly dynamic and erratic nature of bandwidth on a cellular network, the measurements obtained with the same window size can vary significantly over multiple runs. For instance, if a large amount of bandwidth is available in one run, then we would expect a large window size to be more appropriate, whereas a smaller window size may be more suited towards instances with

lower amounts of available bandwidth. The experiments are fairly repeatable with deviations of $\pm 2ms$ in delay and $\pm 0.05 MBit/s$ in throughput.

2 Additive Increase/Multiplicative Decrease (AIMD)

We implemented a simple Additive Increase/Multiplicative Decrease (AIMD) scheme, similar to TCP's feedback control algorithm for congestion control, and evaluated the performance of AIMD on an emulated cellular network. The AIMD scheme generates the congestion window size as a function of discrete time steps $t \in \mathbb{N}_+$, $W(t)$ and is characterized by two constants $\alpha \in \mathbb{N}_+ \, \beta \in (0, 1)$ that denote the additive increase and multiplicative decrease respectively, i.e.,

$$W(t+1) = \begin{cases} \beta W(t) & \text{if packet loss occurs} \\ W(t) + \alpha & \text{otherwise,} \end{cases} \tag{1}$$

where $W(0) = 1$ initially.

As expected, the AIMD congestion control policy performed very poorly when evaluated against the emulated Verizon LTE (4G) connection across varying values for α and β. The poor performance of AIMD stems from the fact that endpoints cannot rely on packet drops to learn of unacceptable congestion along a cellular network path [1]. In fact, in the evaluations that we conducted, we never saw an occurrence of a single packet drop! As a result, the AIMD feedback control was not reactive to the build-up of congestion and in effect, the queue was completely saturated with packets, leading to extremely large signal delays and low throughput overall.

Although the AIMD scheme performed poorly for virtually all values of α and β, we empirically found that the value of 1 for α worked the best. Which is intuitive, since the window-size will increase even quicker with high α and will not be reset regardless of α, since no packet drops occurred. Interestingly enough, the value for β did not influence our results at all since no packet drops occurred during our evaluations against the emulated cellular network.

3 Delay-triggered Scheme

In the previous section we observed that the occurrence of a packet drop is extremely rare and that packet loss is not a good indicator of congestion in a mobile network. Thus, in this section we implemented a delay-triggered scheme, where the congestion window rises or falls based on whether the Round-trip Time (RTT) is above a certain threshold θ, i.e.,

$$W(t+1) = \begin{cases} \beta W(t) & \text{if RTT} > \theta \\ W(t) + \alpha & \text{otherwise,} \end{cases} \tag{2}$$

where $W(0) = 1$, α and β represent the additive increase and multiplicative decrease constants as before, and the variable RTT denotes the Round-trip Time of the most recently acknowledged packet.

α	β	θ	Throughput (Mbits/s)	Signal Delay (ms)	$\log\left(\frac{\text{Throughput}}{\text{Delay}}\right)$
1	0.5	100	2.93	257	2.43
1	0.5	50	0.36	126	1.05
1	0.5	75	2.15	166	**2.56**
1	0.1	100	2.82	253	2.41
5	0.75	50	0.44	115	1.34
5	0.5	50	0.99	154	1.86

Table 1: The performance of our delay-triggered scheme with respect to throughput, 95-percentile signal delay, and score under varying parameters. The maximal score that was attained by varying the parameters is highlighted in bold.

We experimented with various values of α, β, and θ to determine the parameters that yield the best performance when evaluated against the emulated Verizon LTE (4G) connection. Table 1 depicts the performance of our delay-triggered scheme with respect to throughput, 95-percentile signal delay, and score for varying thresholds and parameters. We note from the table that a lower value for the RTT threshold θ generally results in higher throughput, but lower signal delay. The optimal parameters that maximize the score were found to be $\alpha^* = 1$, $\beta^* = 0.5$, and $\theta^* = 75$.

4 Our Approach (The Contest)

We implemented two congestion control schemes for solving the problem of optimal congestion control: (i) an algorithm inspired by Sprout and (ii) a theoretically-inspired algorithm that hinges on ideas from Multi-armed Bandit (MAB) literature.

4.1 Proportionally Driven Particle Filter with Resampling (Spoderman)

In this section we are describing a method that consistently outperforms Sprout by changing key aspects of the algorithm. We have implemented a scheme very similar to that of Sprout where we use a particle filter to estimate the rate in which the queue is being drained to the receiver. However, there are two main differences with the approaches. Firstly, Sprout does not resample the particle (rates) probability distributions and thus is prone to having one particle dominate the prediction. Secondly, Sprout uses a Wiener process to evolve the probability distribution without observation in order to produce a forecast of the rates in the near future. For our approach, we have added a resampling procedure that is triggered when a certain threshold for particle dominance is met. We also do not have a forecasting procedure as we have found that it does not produce a significant difference in the efficiency of the network.

4.1.1 Proportional Control

Instead of trying to evolve the probability distribution without observation, we determine a congestion window by estimating the amount of packets in the queue and subtracting this from the expected drain from the queue. We then use a proportionaly controller to drive the congestion window to the goal congestion window at each tick. This reduces the jitter of the congestion window and allows it to not "overreact" to changes in the queue drainage rate. We have found this approach to perform as well as Sprout on some testing scenarios.

4.1.2 Particle Filter with Resampling

Since our newly developed Multi-Armed Bandit Approach did not deliver the expected results and no improvement over the performance of Sprout we have developed two techniques to improve the performance of Sprout itself.

Sprout is estimating the probability distribution of the rates λ in a Particle Filter like structure. In this setting each particle is a bin of the 256 bins ranging from 0 to 1000 MTU-sized packets per second, (larger than the maximum ware we observed). Each particle has a weight w_i. All weights are initialized to equal values in the beginning.

1. Propagation: Each particle is perturbed by Brownian motion with $\sigma^2 * \tau$. To make sure that all bins continue to exist, the weight of the perturbed particle is distributed among the new particle location. This way we can maintain a fixed discrete set of particles.

2. Measurement update: Based on the observed k number of bytes during the tick, each particles weights w_i are updated based on the poisson pdf

$$w_i = w_i \frac{(x\tau)^k}{k!} \exp(-x\tau) \tag{3}$$

3. Normalization: After normalization

$$w_i = \frac{w_i}{\sum_j w_j} \tag{4}$$

we will end up with a posterior distribution, where each weight describes the likelihood of the particle, i.e. the rate bin, being true.

4. Resampling: Due to frequent non changing measurement updates, which occurred quite often during our test runs the posterior can become degenerate. This means that some of the particles weights are nearly zero. Even if their likelihood becomes high in future measurement updates, they will not be able to recover. The observed behavior is a very slow recovery from low throughput occurrences, e.g. when the mobile client is changing connection between cellular towers. The recursive estimation literature has developed a large variety to counteract this sub-optimal behavior. A good estimate for the number of effective particles

$$N_{\text{eff}} = \frac{1}{\sum_i w_i^2} \tag{5}$$

, i.e. particles which are degenerate and not close to zero. We resample in the case that

$$N_{\text{eff}} < 30, \tag{6}$$

where the threshold was estimated empirically. While other resampling technqiues exist we weill rely on uniform resampling.

$$w_i = w_i + \frac{\eta}{N} \tag{7}$$

Where η is proportional to the amount of weight that is added to all particles to reactivate degenerate particles. In a next step we re-normalize the weights again and continue.

Adding the resampling step in our framework has shown significant improvement in increasing the throughput after the network performance decreased dramatically. In general, the improvement made it possible to recover and adapt in a much faster rate to changes in network performance.

More specifically, we were able to outperform Sprout, as our implementation of the congestion control scheme achieved a score of 38.35, a throughput of 3.95 Mbps, and 95th percentile signal delay of 103 ms.

4.2 Multi-Armed Bandit Approach (MAB)

4.2.1 Introduction and Related Work

In this subsection, we outline an approach that is based on recasting the problem of congestion control in terms of a well-known problem in stochastic optimization. The problem of congestion control is challenging predominantly because the available bandwidth is time-varying and unknown in advance. Thus, generating an optimal sequence of congestion window sizes at each discrete time step requires two competing considerations: (i) setting the congestion window to be small enough so that the available bandwidth can be inferred without inducing high signal delays and (ii) large enough so that high utilization is achieved with respect to the available (time-varying) bandwidth.

These two objectives can recast in terms of simultaneous *exploration* of the current and future capacity by varying the congestion window size and *exploitation* of the bandwidth estimate to ensure high utilization. This dilemma is commonly known as the *exploration-exploitation trade-off* and has been rigorously studied in Multi-Armed Bandit (MAB) literature [2–4]. The MAB problem is a notoriously difficult problem that has been around since 1950s. As such, the MAB literature is rich with algorithms that have been rigorously analyzed and shown to exhibit favorable theoretical properties in both stochastic and adversarial settings. Examples include the Upper Confidence Bound (UCB) and Exponentially-weighs algorithm for exploration and exploitation (Exp3) [3, 4].

4.2.2 The Multi-Armed Bandit Problem

The canonical stochastic MAB setting concerns a sequential optimization problem where a gambler seeks to maximize the obtained reward by playing one of the $n \in \mathbb{N}_+$ slot machines at each discrete time step at a casino. When the gambler plays an arbitrary machine $i \in [n]$ at time-step $t \in \mathbb{N}_+$ a stochastic reward $R_i(t)$ with mean λ_k is generated (the action of playing machine i is also referred to as pulling *arm* i); however, the statistics $\lambda_1, \ldots, \lambda_n$ are not available a priori. Thus, the gambler must balance exploration and exploitation by experimenting with playing different machines (exploration) and playing the best machine thus far (exploitation). Given a time horizon $T \in \mathbb{N}_+$, the MAB problem is to generate an optimal sequence of slot indices $\pi^* = (I_1^*, \ldots, I_T^*)$ so that the expected reward is maximized, i.e.,

Problem 1 (Canonical MAB Problem)**.**

$$\pi^* = \underset{\pi=(I_1,\ldots,I_T)}{\operatorname{argmax}} \sum_{t=1}^{T} \mathbb{E}[R_{I_t}(t)] = \underset{\pi=(I_1,\ldots,I_T)}{\operatorname{argmax}} \sum_{t=1}^{T} \lambda_{I_t}.$$

As we note more rigorously in the following subsection (Sec. 4.2.3), the problem of choosing an optimal arm at each time step can be seen as optimizing the congestion window at each time step. Moreover, the expected reward associated with each arm can be seen as the expected performance of a particular choice of congestion window size. Unfortunately, however, the problem of optimal congestion control differs from the traditional MAB formulation in that the stochastic rewards in the context of congestion control are time-varying and are a function of the available bandwidth at each time step. Hence, we extend a novel MAB algorithm that is able to handle MAB settings in which the rewards' statistics evolve over time, which is analogous to the time-varying nature of bandwidth.

In [5], the authors consider a non-stationary MAB formulation where the variation of the rewards are bounded by a variation budget V_T. The authors propose an algorithm for generating policies that achieve a regret of order $(nV_T)^{1/3}T^{2/3}$, which is long-run average optimal if the total variation of the arm V_T is sub-linear with respect to the time horizon T [5]. In the context of congestion control, the variation budget V_T is analogous can be seen as analogous to the total variation of the available capacity within a given time frame T and is more rigorously defined in Eq. 8.

4.2.3 Recasting Optimal Congestion Control Problem to MAB

Our mapping between the optimal congestion control problem and MAB are as follows:

- Arms $\equiv$ Congestion Window Sizes: The discrete optimization parameter to be optimized at each decision step is the size of the congestion window W. Since congestion window sizes can be quite large, we discretize the space of possible congestion windows by partitioning the interval $[0, W_{\max}]$ into $\lceil \frac{W_{\max}}{\delta_W} \rceil$ intervals of size δ_W, where $W_{\max}$ is a user-specified upper bound on the window size. Effectively, this partitioning scheme enables computational tractability by reducing the number of optimization parameters, i.e., arms, to consider, as each congestion window interval is treated as a separate "arm." For example, $W \in [0, \delta_W)$ maps to arm 0, $W \in [\delta_W, 2\delta_W)$ maps to arm 1, and so on. To map an arm index a to a congestion window size, we use discrete uniform sampling: $W \sim \text{Uniform}\left(\delta_W a, \delta_W (a+1)\right)$.
- Rewards $R_t^{\text{W}} \equiv$ rate of acknowledgments: there are two competing objectives (high throughput and low 95-percentile signal delay) in the problem of congestion control, but we can map the reward associated with a congestion window of W into a single scalar value by considering the *rate of acknowledgments received* under window size of W. Although this definition of reward is not precisely the overarching score function, it acts as an objective function that incentivizes high throughput and low signal delays.
- Arm statistics $\lambda_1(t), \ldots, \lambda_n(t) \equiv$ *expected rate of acknowledgments* ***given the available bandwidth*** *at time step* t: the problem of optimal congestion control differs from the canonical MAB formulation in that the stochastic "rewards" in the context of congestion control are time-varying, i.e., **the expected reward for each congestion window size is a function of the available bandwidth at time step** t.

4.2.4 Problem Definition

We formally define the total variation of expected reward associated with each arm as a function of the available bandwidth

$$\forall i \in [n] \quad \sup_{P \in \mathcal{P}} \sum_{j=1}^{n_p - 1} \left| \lambda_i(p_{j+1}) - \lambda_i(p_j) \right| \leq V_T, \tag{8}$$

where $\mathcal{P} = \left\{ P = \{p_1, \ldots, p_{n_P}\} \mid P \text{ is a partition of } [0, T] \right\}$. For our empirical results, we empirically measured the total variation of the available bandwidth using the available sample mobile network traces and used a value of $V_T = 100$.

The objective of maximizing expected reward after T time steps can also be thought of as minimizing *expected regret*, i.e., the differential between a sub-optimal policy π and the best policy in hindsight denoted by $R(\pi, T)$:

$$\mathbb{E}[R(\pi, T)] = \sum_{t=1}^{T} \lambda^*(t) - \sum_{t=1}^{T} \mathbb{E}[\lambda_{I_t}(t)] \tag{9}$$

where $\lambda^*(t)$ is the expected reward of the best slot machine at the time slot t, i.e., $\lambda^*(t) = \max_{i \in [n]} \lambda_i(t)$.

Problem 2 (Congestion Control Problem)**.** *Compute the optimal policy that minimizes expected regret, i.e.,*

$$\pi^* = \operatorname*{argmin}_{\pi} \mathbb{E}[R(\pi, T)]. \tag{10}$$

4.2.5 Exp3CC: MAB-based Congestion Control Algorithm

Algorithm 1: Exp3CC: The **Exp3** algorithm for **Congestion Control**

Input: Time horizon T, variation budget V_T, an upper bound on the congestion window $W_{\max}$, and the congestion window discretization constant δ_W.

// Compute the number of "arms" by discretizing the congestion window interval
$n \leftarrow \lceil \frac{W_{\max}}{\delta_W} \rceil$;
// Compute the length of each epoch
$\tau \leftarrow (n \log n)^{1/3} (T/V_T)^{2/3}$;
// Compute the exploration and exploitation constant γ
$\gamma \leftarrow \min\left(1, \sqrt{\frac{n \log n}{(e-1)\tau}}\right)$;
while $t_{current} \leq T$ **do**
  // Reset the weights for each arm to remain adaptive to changes in bandwidth
  $w_i \leftarrow 1 \quad \forall i \in [n]$;
  // Determine the end point of the current epoch
  $T_{\text{end}} \leftarrow t_{\text{current}} + \tau$;
  while $t_{current} \leq T_{end}$ **do**
    // Set congestion window probabilistically based on weights
    $p_i \leftarrow (1-\gamma)\frac{w_i}{\sum_{j=1}^{n} w_j} + \frac{\gamma}{n} \quad \forall i \in [n]$;
    Randomly sample an arm $\text{arm}_{\text{current}}$ based on the distribution defined by $p_1, \ldots, p_n$;
    $W \sim \text{Uniform}\left(\delta_W a_{\text{current}}, \delta_W (a\text{current} + 1)\right)$;
    Set the congestion window to W and send all packets W;
    $t_{\text{sent}} \leftarrow t_{\text{current}}$;
    if *W acknowledgments are all received* **then**
      // Compute the reward associated with the congestion window W
      $R \leftarrow \frac{W}{t_{\text{current}} - t_{\text{sent}}}$;
      // Update the weight of the corresponding arm
      $w_{a_{\text{current}}} \leftarrow w_{a_{\text{current}}} \exp\left(\frac{\gamma R}{n p_{a_{\text{current}}}}\right)$;

4.2.6 Results and Discussion

Despite the favorable theoretical properties of our MAB-based algorithm, our method did not perform well when evaluated against the emulated Verizon LTE (4G) mobile network. In particular, our algorithm achieved a score of ≈ 20 and its performance was bottlenecked by high 95-percentile signal delays.

Upon further inspection of our algorithm and its performance against emulated data, we compiled a list of possible reasons that could explain the poor performance of a MAB-based approach (including our method) for the problem of optimal congestion control.

1. Exceedingly high temporal variations: the available bandwidth was subject to very high temporal variation and was exceedingly dynamic.

2. Lack of prediction step/reactiveness: since our algorithm did not explicitly attempt to perform prediction as, the signal delays during times of exceedingly low bandwidth were unexpected and heavily penalized.

4.3 Conclusion

In this section we summarized the two approaches that we developed and implemented for purposes of optimal congestion control in mobile networks. Our predictive approach, named Spoderman, was inspired by the Sprout algorithm [1] and performed very well when evaluated against the emulated Verizon LTE (4G) mobile network–it even outperformed Sprout! Our second approach hinged on recasting the optimization problem into the MAB framework and building upon existing algorithms popular in Theoretical Computer Science literature. We modified a recent algorithm for generating long-run average optimal policies in dynamic settings where the rewards are subject to temporal variations [5]. However, despite the favorable theoretical properties of the algorithm that we designed (Alg. 1), our method performed poorly when we evaluated it against the simulated mobile network.

References

[1] K. Winstein, A. Sivaraman, and H. Balakrishnan, "Stochastic forecasts achieve high throughput and low delay over cellular networks," in *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pp. 459–471, 2013.

[2] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *arXiv preprint arXiv:1204.5721*, 2012.

[3] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[4] P. Auer and R. Ortner, "Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem," *Periodica Mathematica Hungarica*, vol. 61, no. 1-2, pp. 55–65, 2010.

[5] O. Besbes, Y. Gur, and A. Zeevi, "Non-stationary stochastic optimization," *Operations Research*, vol. 63, no. 5, pp. 1227–1244, 2015.