A Randomized Online Algorithm for Bandwidth Utilization

Sanjeev Arora *
Princeton University

Bo Brinkman†
Princeton University

Abstract

Protocols for data transmission over a TCP-like computer network should not only lead to efficient network utilization but also be fair to different users. Current networks accomplish these goals by some form of end-to-end congestion control. However, existing protocols assume somewhat altruistic behavior from hosts, and Karp et al. [7] have initiated a study of whether or not the host's optimum strategy is altruistic. We carry this exploration further by developing an efficient randomized algorithm for bandwidth utilization in their model. The competitive ratio of this algorithm is optimal up to a constant factor. Karp et al. had earlier studied the deterministic case and left open the randomized case.

What may be of some interest is that our algorithm is essentially the classical MIMD (multiplicative increase, multiplicative decrease) strategy, which is very aggressive and non-altruistic.

1 Introduction

In TCP networks, such as the Internet, there is no central authority which allots bandwidth to hosts. Instead, individual hosts are responsible for setting their sending rate appropriately. Each host would like to send data as fast as possible, but if they all do this, router queues throughout the network overflow and start dropping packets, which dramatically reduces network utilization. Such a *congestion collapse* was a frequent occurrence on the Internet in the 1980s. Thus there is an inherent tension between the interests of the hosts and those of the network designer.

In the 1980s, network designers introduced end-to-end congestion control [6]. In these schemes an individual host notices that its packets are being dropped, and adjusts its sending rate down in the hope of preventing future congestion. Specifically, in current implementations it halves its rate (this is termed *multiplicative decrease*). If, on the other hand, the host's packets are getting through, it increases its sending rate by 1 (*additive increase*) because it speculates that the network might be under-utilized. This protocol, called *additive-increase-multiplicative-decrease* (AIMD), has proved successful in preventing congestion

*Supported by a David and Lucile Packard Foundation Fellowship. email: `arora@cs.princeton.edu`

†Supported by a Frances Upton Fellowship. email: `brinkman@cs.princeton.edu`

problems on the Internet. A string of papers starting with Chiu and Jain [3] (see also the recent [8]) provide a heuristic argument that AIMD quickly stabilizes send rates to levels corresponding to a fair and efficient sharing of network resources. Other strategies such as MIMD (multiplicative-increase-multiplicative-decrease, whereby the hosts raise their transmission rates by a multiplicative factor so long as they do not experience dropped packets) lead to oscillatory behavior and inefficient network use.

Note that the AIMD protocol was developed by the designers to promote a social good, namely, efficient and fair use of the network. However, participation in TCP congestion control is voluntary, and opting out involves trivial modifications to the host's TCP/IP code. With the increased economic importance of the Internet, and the increasing variety of network applications, such opting out has become more frequent. Many programmers choose to use UDP, which does not have congestion control, for applications where sudden drops in sending rate might be unacceptable. Bansal and Balakrishnan [1] have developed binomial congestion control methods which are compatible with AIMD, but better suited to streaming media.

Thus there is a need to study congestion control from the hosts' perspectives. Understanding their optimum (possibly selfish) ways of responding to their network environment may provide insight into how to design networks where the hosts do not have an incentive to follow strategies that conflict with the common good.

Karp et al. [7] initiated such a study by proposing a simple model in which an adversary is used to model the unpredictable nature of bandwidth availability on the network. This allows them to abstract away the specific nature of the congestion control used in the network, and concentrate on exploring the sender's strategy for making best use of the available network resources.

In this model, the goal is to understand the problem of regulating the rate of a unicast flow from host A to host B. An adversary causes fluctuations in the available bandwidth at each step, but these fluctuations have to satisfy certain "continuity assumptions," which ensure that the available bandwidth cannot change too sharply in a single step. Host A does not have direct knowledge of the bandwidth available to it, but it does have indirect knowledge of this bandwidth—an upper bound or a lower bound—since it knows how many bytes it tried to send in the previous step and whether or not they got through (that is, were acknowledged by host B).

Thus the problem of deciding the send rate is reduced to one of *bandwidth utilization*. Moreover, the general setup is one familiar from online algorithms: How can one prevail against an adversary who, though arbitrarily powerful, decides its strategy independent of the algorithm's actions? Karp et al. designed several online algorithms for variations on their model obtained by varying the continuity assumption. In several situations, they were able to design optimal online algorithms. However, they failed to do so in some cases, notably when the algorithm is allowed to be randomized and the bandwidths at successive steps are only required to be within some multiplicative factor μ of the bandwidth at the previous step. They conjectured that a randomized algorithm would be much better than their optimal deterministic algorithm.

In this paper we show that Karp et al.'s conjecture was correct by showing a randomized online algorithm that has a better performance ratio ($\Theta(\log \mu)$ instead of μ) than the deterministic algorithm. Furthermore, we show that this algorithm is optimal up to constant factors. Thus we provide a new problem in online algorithms in which randomized algorithms are provably better than deterministic algorithms. (This has happened before, most notably in paging [4].)

Another reason for interest in our work is that our algorithm is a randomized version of MIMD, thus suggesting that at least in the Karp et al. model, the optimum strategy for the hosts may conflict with the public good.

Of course, we stress—as Karp et al. did also—that this model is simplistic in two major ways since in real life fluctuations in available bandwidth are not (a) adversarial or (b) independent of the actions of host A (indeed, if host A sends too much data, it *affects* the amount of congestion in the network).

Nevertheless, we hope that our result provides some food for thought and discuss this aspect some more at the end of the paper.

1.1 Problem Statement

We use the adversarial model of Karp et al. from [7]:

- Time is divided into discrete steps (the duration of each step could be some multiple of the packet round-trip time).

- At each time step i, the amount of available bandwidth, $b_i \geq 1$, is chosen by an adversary. We assume that the adversary is oblivious, which is equivalent to saying that the algorithm's actions do not affect the amount of available bandwidth. This may be a reasonable assumption if the number of hosts using each link is large.

- At each time step, the algorithm picks x_i, its estimate of b_i, and sends x_i bytes of data. When the host sends too much data during a time step, all of that data is dropped by the network.[1] The actual bandwidth for a step, a_i, is therefore
$$a_i = \begin{cases} x_i & \text{if } x_i \leq b_i \\ 0 & \text{otherwise} \end{cases}$$
This corresponds to Karp et al.'s severe cost function.

- The total throughput achieved by the algorithm is $act = \sum a_i$ (where the sum is over all steps required to transmit the flow), and the optimal offline throughput is $opt = \sum b_i$. We seek to minimize the competitive ratio, $\frac{opt}{act}$. As is usual in online algorithms, we consider the asymptotic value of this ratio, after an arbitrarily large number of steps.

[1] While this may seem like a drastic assumption, note that it actually corresponds to withholding information from the algorithm. We could assume that if $b_i < x_i$ then *some* packets get through, say in some manner proportional to the difference between b_i and x_i. Thus the algorithm could learn a little about b_i. However, such changes to the model would still not change the basic result that MIMD is the optimal algorithm.

Karp et al. observe that an unrestricted adversary has too much power: it can easily force the competitive ratio to be arbitrarily large. They propose restrictions on the adversary, which correspond to placing some continuity conditions on the bandwidth available in successive steps. We present their suggested restrictions, though we will focus on the first two.

Definition *(Fixed range adversary)* A fixed range adversary is constrained to always pick b_i in some fixed range $[c, d]$. This is a natural idea, because there will be some upper bound on the available bandwidth due to hardware limitations. We also consider there to be an implied lower-bound of 1 for the sending rate, because the flow never runs out of data to send.

Definition *(μ-multiplicative adversary)* At each time step i, the adversary must pick $b_i \in [\frac{b_{i-1}}{\mu}, \mu b_{i-1}]$, satisfying $b_i \geq 1$. The amount of available bandwidth at a step depends on the available amount in the previous step. Available bandwidth might increase or decrease by a multiplicative factor (perhaps a new host is trying to use the same route, or a host which was previously sharing the route quit), but cannot change arbitrarily.

Definition *($\{\alpha, \beta\}$-additive adversary)* At each time step i, the adversary must pick $b_i \in [b_{i-1} - \alpha, b_{i-1} + \alpha]$, satisfying $b_i \geq \beta$. The bandwidth increases or decreases by an additive factor, but never goes below some lower bound β (usually equal to 1).

2 Results

Our main results concern randomized algorithms in the μ-multiplicative model. For deterministic algorithms, Karp et al. showed that there is an algorithm that achieves a competitive ratio μ and that no algorithm can do any better.

Theorem 1 *No randomized algorithm can achieve competitive ratio better than* $\ln(\mu) + 1$ *in the μ-multiplicative model.*

The next theorem gives an upper bound that matches the lower bound up to a constant factor.

Theorem 2 *There is a randomized algorithm for bandwidth allocation in the μ-multiplicative model which achieves a competitive ratio of* $\frac{8}{3}\lg(\mu) + 16$.

The proof of Theorem 1 is a simple corollary of the following theorem of Karp et al.:

Theorem 3 (Karp et al.) *No randomized algorithm can achieve competitive ratio better than* $\ln(\frac{d}{c}) + 1$ *in the fixed range model with range* $[c, d]$.

For completeness, we restate the proof of Theorem 3 from [7].

Proof *(Karp et al.)* Apply Yao's minimax technique [12]. Instead of trying to prove a lower-bound for randomized algorithms against an adversary, we can consider deterministic algorithms against a specific randomized adversary.

In this case, Karp et al. have the adversary pick $b_i = y$ with the probability density function $g(y) = \frac{c}{y^2}$ for $y < d$. This assigns a total probability $\int_c^d \frac{c}{y^2} dy = 1 - \frac{c}{d}$ to the interval $[c, d)$. The remaining $\frac{c}{d}$ probability is assigned to d. If the algorithm picks $x_i = x$, then the expected amount transmitted is $x \int_x^d g(y)dy + x\frac{c}{d} = c$. The expected optimal bandwidth is $\int_c^d g(y)ydy + d\frac{c}{d} = c(\ln(\frac{d}{c}) + 1)$. Hence the competitive ratio is $\ln(\frac{d}{c}) + 1$.

Since the adversary in the μ-multiplicative model could choose to always keep the b_i in $[1, \mu]$, Theorem 1 follows directly.

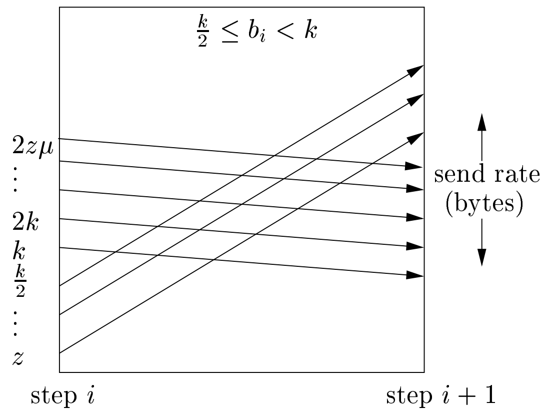
2.1 Algorithm

We now present our algorithm. For simplicity of presentation, we give a version that achieves competitive ratio $4\lg(\mu) + 12$. We assume that μ is a power of two, and that b_1 (the bandwidth available during the first step) is in the range $[1, 2\mu)$. Note that these conditions are also for simplicity, and neither are necessary. We indicate later how to remove both assumptions.

Algorithm Let x_i denote the amount of data which the algorithm chooses to send at step i, where $i \geq 1$. The algorithm picks x_1 uniformly at random from the powers of 2 between 1 and 2μ. At step $i + 1$, it picks x_{i+1} as follows: If at step i, the transmission of x_i bytes failed to complete, let $x_{i+1} = \frac{x_i}{2}$. Otherwise let $x_{i+1} = 2\mu x_i$.

We claim that this algorithm achieves competitive ratio at most $4\lg(\mu) + 12$.

Proof Note that our algorithm uses randomness only in the first "setup" step. In essence we are choosing one deterministic algorithm at random from $\lg(2\mu) + 1$ possible choices. In order to analyze the performance of our randomized algorithm we will consider these deterministic algorithms as an ensemble running in parallel. We will show that the ensemble manages to send at least $opt/4$ bytes (recall that $opt = \sum_i b_i$ is the throughput of the optimum algorithm that has complete information), whence it follows that picking one of the $\lg(\mu) + 2$ algorithms at random gives an expected throughput at least $opt/4(\lg(\mu) + 2)$.



We make a few simple observations. First, at each time step the $\lg(\mu) + 2$ algorithms always pick a *consecutive* sequence of $\lg(\mu) + 2$ powers of two. That each algorithm picks x_i to be a power of two is obvious by induction because μ is a power of two. The fact that the powers of two are consecutive is also an easy induction. If b_i were less than x_i for all the algorithms, then all the algorithms cut their send rate in half, and the x_i remain consecutive. If b_i were more than all the x_i, the x_i would similarly increase in lock-step. What if b_i fell between two consecutive x_i? Let us say that the algorithms are currently covering the powers of two between z and $2z\mu$. The algorithm that picked z will succeed, and will pick $2z\mu$ in the next step. The algorithm that picked $2z\mu$ will fail and then pick $z\mu$ during the next step. Hence the x_i remain consecutive (refer to the figure).

Now we can consider how much data the ensemble actually transmits. We say the ensemble is *successful at step i* if b_i falls within the range of values picked during that step. We will say that the ensemble *achieves e_i bytes at step i* if the largest number of bytes sent by any of the algorithms was e_i. Thus the ensemble's throughput is at least the sum of the number of bytes it achieves during successful steps. Of course, we have to relate this throughput to *opt*.

Let us say the ensemble is successful at step i. Since it achieved e_i, and b_i must fall between two algorithms which picked consecutive powers of two, $b_i < 2e_i$.

At the next step, if the bandwidth increases or stays constant the ensemble is guaranteed to be successful again. An algorithm in the ensemble achieved e_i at step i, and at the next step this algorithm will pick $2\mu e_i$. Since $b_i < 2e_i$, and the continuity assumption requires $b_{i+1} \leq \mu b_i$, we conclude that $b_{i+1} < 2\mu e_i$. Furthermore, there is some algorithm in the ensemble which picked $2e_i$ in step i, and failed, so it will pick e_i in the next step. Thus if $b_{i+1} \geq b_i$ the bandwidth lies in the range covered by the ensemble. In particular, this proves that the amount of available bandwidth can never be above the range covered by the ensemble.

For this reason we only need to worry about the case when $b_{i+1} < b_i$. Here the bandwidth may decrease very fast and the ensemble may not be successful at the next step. Let us say that the ensemble achieved e_i bytes at step i as above. Now if the ensemble is not successful at step $i+1$, we know that there could not have been more than e_i bandwidth available, because we have an algorithm that picked e_i. At step $i+2$, all the algorithms divide their sending rates by 2, which means that if the ensemble is again not successful, the missed bandwidth cannot be more than $\frac{e_i}{2}$. By simple induction we see that the amount of available bandwidth must be decreasing geometrically, and the total amount of missed bandwidth until the next successful step must be less than $e_i + e_i/2 + e_i/4 + \cdots = 2e_i$.

Thus the optimum algorithm could have achieved at most $4e_i$ in all these steps (up to $2e_i$ at step i because it is not constrained to pick only powers of 2, and $2e_i$ during the steps where the ensemble was unsuccessful) whereas the ensemble achieved e_i. Summing over all time steps we conclude that the ensemble manages to transmit at least *opt*/4 bytes in total.

This finishes the proof.

Removing the assumption that μ is a power of 2 is very simple: Round μ to the next larger power of 2. This can result in $\lg(\mu)+3$ algorithms, instead of $\lg(\mu)+2$, which gives the claimed result. Removing the need to start between 1 and 2μ is also easy. At the start the algorithm can simply increase its send rate by a factor of (say) μ^2 until it is successful. This will take some constant number of steps, and since we are concerned with steady-state performance, this one-time cost is unimportant.

2.2 Improved Constants

In the above algorithm we divide by 2 when we decrease, and assume that μ is a power of 2. One can change the algorithm to divide by c, multiply by $c\mu$, and pick a random power of c between 1 and $c\mu$. Tuning the constant c gives the final result, $\frac{8}{3}\lg(\mu)+16$ when $c=4$.

The reader may wonder why we increase by a multiple of μ whenever we are successful. The key insight is that when the algorithm over-estimates the amount of available bandwidth, and thus fails to send anything during a step, we can always charge this unused bandwidth to a successful step. Underestimating, however, causes the algorithm to fall farther and farther behind the available bandwidth, and this is where the adversary can gain the most against the algorithm. Since the continuity condition restricts the bandwidth from increasing by more than a factor μ, multiplying by some multiple of μ is sufficiently aggressive.

2.3 Additive Adversary

While it is not a focus of this paper, we should also note that the same algorithm achieves similar performance in the additive model (and this performance matches known lower bounds). For an adversary that remains in the range $[\beta, \beta+\alpha]$, application of the Karp et al. fixed range lower-bound gives a $1+\ln(\frac{\alpha+\beta}{\beta})$ lower-bound for randomized algorithms against the $\{\alpha,\beta\}$-additive adversary. In order to apply our algorithm, we may just set $\mu = \frac{\alpha+\beta}{\beta}$. This guarantees that $\forall x \geq \beta$, $x + \alpha \leq \mu x$. This immediately results in an algorithm with competitive ratio $\frac{8}{3}\lg(\frac{\alpha+\beta}{\beta}) + 16$.

3 Conclusions

We already mentioned that the Karp et al. model uses a simplified view of reality, and hence any results proved in this model do not necessarily provide any guidance for network designers. In fact, Karp et al. suggest that AIMD ought to be the optimum protocol for hosts once the model has been refined to correctly reflect real-life constraints.

Nevertheless, the fact that our algorithm is a multiplicative-increase-multiplicative-decrease (MIMD) algorithm should be interesting. MIMD algorithms are efficient at rapidly finding the amount of available bandwidth, but it is believed that they lead to instability as well as unfair sharing of resources; see Chiu and Jain [3].

On the other hand, it has been observed for some time ([5],[10],[11]) that if most hosts follow AIMD ("are altruistic") then aggressive individual hosts are able to subvert AIMD congestion control in TCP and receive an unfair share of the bandwidth. Our result (though the model is simple) suggests that this may not be a fault of TCP, but a simple fact of life: In an uncertain environment, aggressiveness is key to finding available bandwidth.

Several future projects suggest themselves. One might be to modify the Karp et al. adversarial model to better reflect real network behavior. Swings in available bandwidth can be large but modeling them as adversarial may be overkill. We suspect that in most plausible models, greed will prove better than altruism.

This leads to another (long standing) open problem: How can one change current network protocols to discourage cheating in TCP-like networks? It may well be that routers have to enforce some kind of fairness. Recent papers ([5],[11]) suggest some ways to do so, but are not as yet backed by any solid theoretical results.

Acknowledgments

We thank Larry Peterson for many helpful discussions, and Kevin Lai and Frank Kelly for pointing us to some previous work on congestion control. We would also like to thank Ding Liu for noticing the application of our algorithm to the $\{\alpha, \beta\}$-additive case.

References

[1] D. Bansal and H. Balakrishnan. Binomial congestion control algorithms. In *Proceedings of INFOCOM*, 2001. Extended version available as MIT-LCS-TR-806 from http://nms.lcs.mit.edu/papers/cm-binomial.html.

[2] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. Williamson. Adversarial queueing theory. In *ACM Symposium on Theory of Computing*, pages 376–385, 1996.

[3] D. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Journal of Computer Networks and ISDN Systems*, 17:1–14, 1989.

[4] A. Fiat, R. Karp, M. Luby, L. McGeoch, D. Sleator, and N. Young. Competitive paging algorithms. *Journal of Algorithms*, 12(4):685–699, 1991.

[5] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, 1999.

[6] V. Jacobson. Congestion avoidance and control. In *Proceedings of ACM SIGCOMM*, pages 314–329, 1988.

[7] R. M. Karp, E. Koutsoupias, C. H. Papadimitriou, and S. Shenker. Algorithmic problems in congestion control. In *41st IEEE Symposium on Foundations of Computer Science*, pages 66–74, 2000.

[8] F. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: shadow prices. *Journal of the Operational Research Society*, 49:237–252, 1998.

[9] L. Peterson and B. Davie. *Computer Networks: A Systems Approach*, pages 446–515. Morgan Kaufmann Publishers, 2000.

[10] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson. TCP congestion control with a misbehaving receiver. *Computer Communication Review*, 29(5), 1999.

[11] S. Shenker. Making greed work in networks: A game-theoretic analysis of switch service disciplines. *IEEETNWKG: IEEE/ACM Transactions on Networking*, 3, 1995.

[12] A.C.C. Yao. Probabilistic computations: Towards a unified measure of complexity. In *18th Symposium on Foundations of Computer Science*, pages 222–227, 1977.